ISSN : 2350-0743

International Journal of Recent Advances in Multidisciplinary Research

Vol. 12, Issue 04, pp.11095-11100, April, 2025

RESEARCH ARTICLE

SECURE DEDUPLICATION OF TEXTUAL DATA IN CLOUD ENVIRONMENTS

Priyanka, S., Anirudh Koushik, B.A., Dr. Nagaraja, G.S.

Department of Computer Science and Engineering, RV College of Engineering, Bangalore, India

ARTICLE INFO

Article History

Received 30th January, 2025
Received in revised form
17th February, 2025
Accepted 26th March, 2025
Published online 30th April, 2025

Keywords:

Secure Deduplication, Cloud Storage, Textual Data, Client-Side Deduplication, Encryption, AES, Triple DES, Data Security, Data Integrity, Redundancy Removal.

****Corresponding author: Priyanka***

ABSTRACT

The rapid expansion of textual data in cloud environments presents significant challenges for managing large-scale storage systems. Data deduplication has become a valuable strategy for reducing storage requirements, but it also introduces security concerns. This paper presents a deduplication method specifically designed for textual data, utilizing a client-side approach to achieve high compression rates while ensuring data security. Our method is designed to resist side-channel attacks, enhancing overall security. Through experimental evaluations, we demonstrate how this approach significantly reduces storage demands while maintaining data confidentiality. This efficiency can lead to substantial cost savings and improved performance in large-scale data management systems.

Citation: Priyanka, S., Anirudh Koushik, B.A., Dr. Nagaraja, G.S. 2025. *"*Secure Deduplication of Textual Data in Cloud Environments.*", International Journal of Recent Advances in Multidisciplinary Research,* 12, (04), 11095-11100.

INTRODUCTION

Data deduplication is a powerful technique that optimizes storage by eliminating duplicate files or data blocks, ensuring that only unique data is stored. Instead of saving multiple copies of identical files, a single instance is kept, and references are created for duplicate occurrences. This method is particularly beneficial in cloud storage environments, where vast amounts of data are continuously uploaded and managed. In backup applications, deduplication can reduce storage requirements by up to 90–95%, while in standard file systems, reductions of up to 68% are achievable. Deduplication techniques vary based on granularity and implementation. At the granularity level, the three primary methods include file-level, fixed-size block, and variable-sized block deduplication. File-level deduplication removes exact duplicate files, while fixed-size block deduplication divides files into uniform blocks and eliminates redundant ones. Variable-sized block deduplication, on the other hand, adapts chunk sizes dynamically to identify duplicates, but it requires additional metadata management and may lead to hash collisions. Among these, block-level deduplication is generally more effective, as it can detect duplicate data even if it appears in different files or locations within a system. Deduplication can also be categorized by its execution method: server-side or client-side. Server-side deduplication occurs on the cloud server, reducing the need for users to process data locally. However, this approach may not fully address communication overhead. Client-side deduplication, in contrast, takes place on the user's device before data is uploaded to the cloud. By detecting and removing duplicates before transmission, this approach minimizes bandwidth usage. However, it also introduces security risks, such as side-channel attacks and potential data leaks. Traditional deduplication (Classic Deduplication, CD) primarily focuses on identifying and removing duplicate files, which can be inefficient when files contain similar but non-identical content. To address this limitation, Generalized Deduplication (GD) has emerged as an improved approach. GD extends traditional deduplication by recognizing and eliminating similar, rather than identical, data chunks. This reduces storage requirements more effectively, eliminates unnecessary redundancy, and enhances data management efficiency.

CONTRIBUTION

This paper proposes an advanced deduplication method tailored specifically for textual data, integrating existing frameworks to enhance both storage efficiency and security. Our approach introduces multiple key contributions to address the challenges associated with secure deduplication in cloud environments. Firstly, Client-Side Deduplication is implemented to minimize redundant data transfers and optimize bandwidth utilization. By performing deduplication before data is uploaded to the cloud, this method significantly reduces communication overhead, mitigates potential data leakage risks, and enhances data processing speed. Unlike server-side deduplication, our approach ensures that sensitive information remains encrypted before transmission, eliminating the need for cloud servers to process raw data. Secondly, Data Security remains a fundamental aspect of our framework.

We employ a hybrid encryption mechanism that safeguards data confidentiality throughout the deduplication process. This is particularly beneficial for resource-constrained environments such as IoT systems, mobile devices, and edge computing applications, where security and efficiency must be balanced without compromising performance. Our method is also designed to withstand potential side-channel attacks, offering an additional layer of resilience against unauthorized access.

Lastly, Load Balancing and Scalability are key considerations in our design. To optimize system performance, deduplication tasks from multiple clients can be offloaded to a dedicated deduplication server. This distributed processing approach helps balance the computational load, alleviating strain on both client devices and cloud infrastructure while maintaining an efficient storage management system. By reducing redundancy and improving storage allocation, this approach contributes to overall cost reduction in cloud-based storage services.

The remainder of this paper is organized as follows: Section II presents the Proposed Design and underlying system models that support our deduplication framework. Section III elaborates on the Methodology and Implementation, detailing the encryption and deduplication mechanisms. Section IV discusses the Performance Metrics, evaluating the impact of our approach on storage efficiency and security. Section V assesses Security Considerations, addressing potential vulnerabilities and mitigation strategies. Section VI explores techniques for Preventing Information Loss during deduplication. Section VII provides Experimental Results and Evaluation, demonstrating the effectiveness of our approach in real-world scenarios. Section VIII reviews related work, comparing our framework to existing deduplication strategies. Finally, Section IX presents our Conclusion and Future Directions, summarizing our contributions and outlining potential areas for further research.

PROPOSED DESIGN: The design of the Secure Text framework ensures that encrypted textual data is deduplicated efficiently without exposing sensitive information. The workflow begins with the initial step of data splitting, where input files are divided into smaller blocks. These blocks undergo encryption using both AES and Triple DES, providing a dual-layer security mechanism. A Cyclic Redundancy Check (CRC) is then computed to maintain data integrity and detect any unauthorized modifications. Before storing the encrypted data in the cloud, the system checks for duplicates by comparing encrypted blocks. If an identical encrypted block already exists, duplication is prevented, thereby optimizing storage space. In cases where no duplicate is detected, the unique encrypted block is securely stored in the cloud. The sequence of these operations ensures minimal redundancy, optimized storage utilization, and secure data handling.

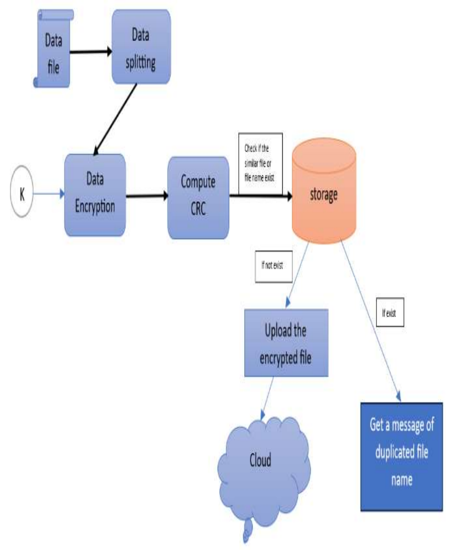

Data file
Data splitting
K
Data Encryption
Compute CRC
Check if the similar file or file name exist
storage
If not exist
If exist
Upload the encrypted file
Cloud
Get a message of duplicated file name

Fig 1. Design of proposed model

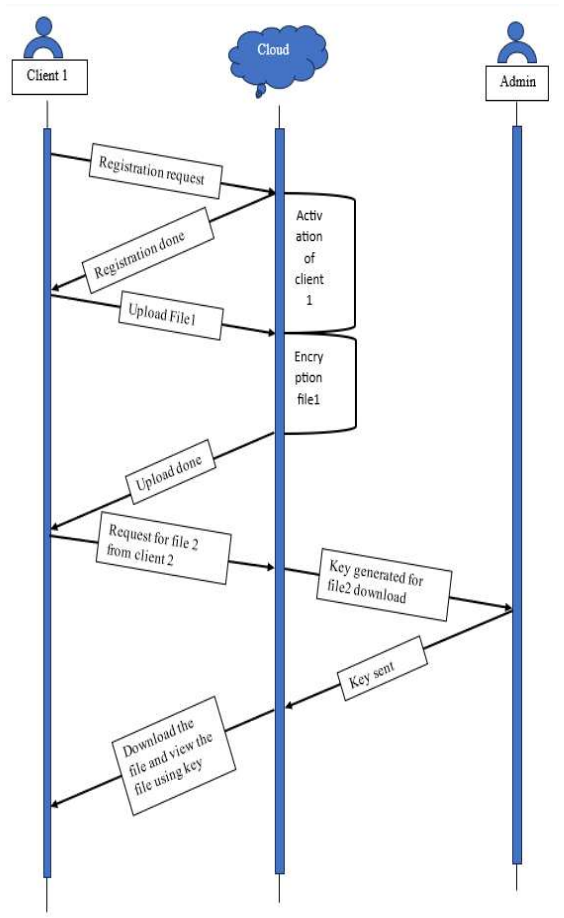


Fig 2. Sequence Diagram of proposed work

The sequence diagram illustrates a secure file-sharing process between clients and a cloud server, facilitated by an administrator. The process begins with Client 1 initiating a registration request to the cloud server. Upon receiving this request, the cloud server processes the registration and activates Client 1. Once activated, Client 1 uploads a file to the cloud, where it undergoes an encryption process before being stored. After a successful upload, a confirmation is sent back to Client 1. At a later stage, Client 2 submits a request to download a specific file stored on the cloud. The cloud server processes this request by generating a secure key necessary for file decryption and sending it to the administrator for validation. The administrator then forwards the key to Client 2. Upon receiving the key, Client 2 downloads the requested file from the cloud and decrypts it using the provided key, enabling them to access the file securely.

Proposed Approach: The methodology of the Secure Text framework consists of multiple phases, each ensuring secure and efficient data deduplication. In the User Registration and Authentication phase, users first provide credentials such as username, password, and email, which are securely stored in a database. The system employs secure authentication mechanisms to validate user identity before granting access. This step is crucial for ensuring that only authorized users can interact with the system. The next phase is File Upload and Encryption, where users upload textual data files to the system. Before storing the files in the cloud, they are encrypted using a hybrid approach combining AES and Triple DES algorithms. This encryption ensures that even if a file is duplicated, the stored data remains protected from unauthorized access. The encryption process follows the below steps:

Once encrypted, the Deduplication Mechanism is activated. The system extracts a unique hash from the encrypted file blocks and checks against the database to identify duplicates. If a duplicate exists, only a reference to the original encrypted block is maintained rather than storing the file again. The deduplication process is outlined below:

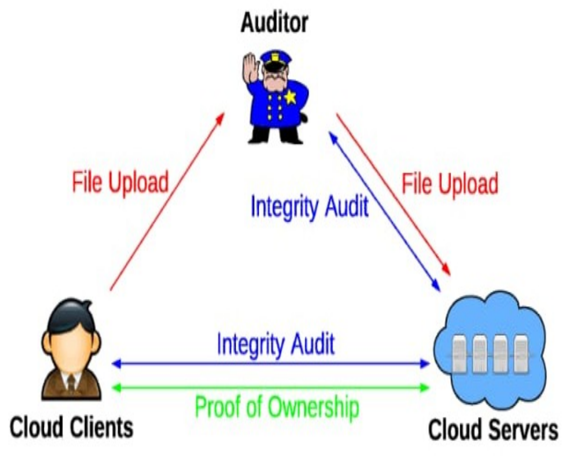


Fig 3. Proposed Model Architecture

Finally, the File Download and Decryption phase allows authorized users to retrieve stored data. When a user requests a file, the system decrypts the encrypted data by reversing the encryption process. This end-to-end encryption and deduplication process guarantees security and efficiency in cloud storage environments.

ALGORITHM DEVELOPED AND IMPLEMENTED: The Secure Text framework incorporates several key algorithms to achieve secure deduplication. The Database Connection Algorithm, implemented in the DBconnection class, establishes a connection to a MySQL database using JDBC, enabling efficient database operations. The FTP File Upload Algorithm, implemented in the Ftpcon class, facilitates secure file uploads to a remote FTP server using the Apache Commons Net library. This method ensures reliable file transfer and prevents data loss during transmission. Additionally, the Email Notification Algorithm, implemented in the MailSender class, allows the system to send automated notifications using the JavaMail API. This feature enhances user interaction by notifying users about key actions such as successful file uploads, retrievals, and authentication alerts. The Secure Text framework incorporates several key algorithms to achieve secure deduplication and encryption. The following sections outline these algorithms in detail.

Database Connection Algorithm

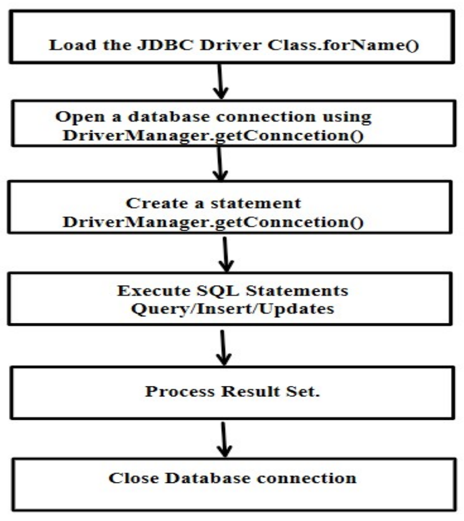

Load the JDBC Driver Class.forName()
Open a database connection using DriverManager.getConncetion()
Create a statement DriverManager.getConncetion()
Execute SQL Statements Query/Insert/Updates
Process Result Set.
Close Database connection

Fig 4. Algorithm: Database Connection

The Database Connection Algorithm, implemented in the DBconnection class, establishes a connection to a MySQL database using JDBC. This enables efficient interaction with the database for storing and retrieving metadata related to file deduplication and encryption. The algorithm is outlined below:

FTP File Upload Algorithm: The FTP File Upload Algorithm, implemented in the Ftpcon class, enables secure file uploads to a remote FTP server. It ensures reliable file transfer and prevents data loss. The steps are as follows:

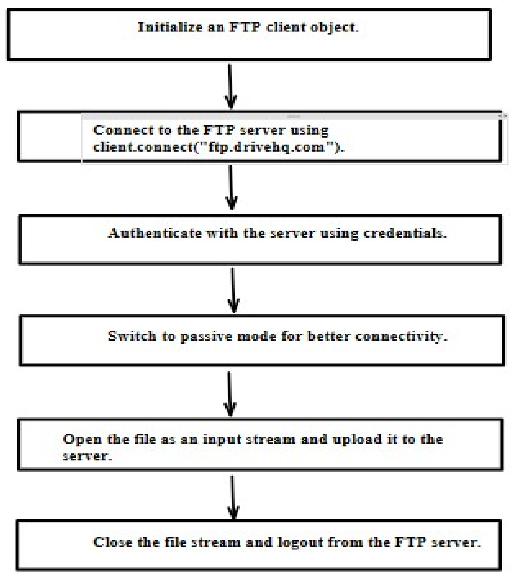
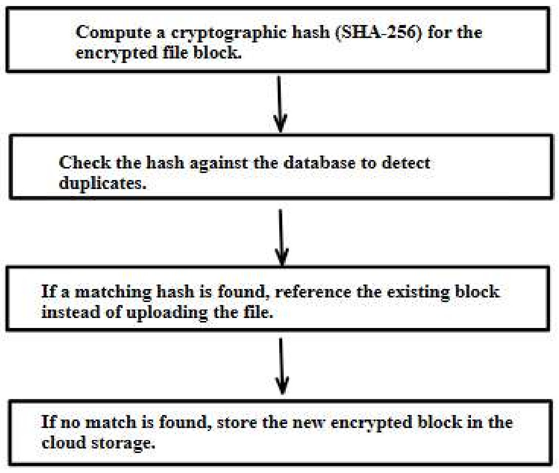

Initialize an FTP client object.
Connect to the FTP server using client.connect("ftp.drivehq.com").
Authenticate with the server using credentials.
Switch to passive mode for better connectivity.
Open the file as an input stream and upload it to the server.
Close the file stream and logout from the FTP server.

Fig 5. Algorithm: FTP File Upload

Secure Deduplication Mechanism: To reduce storage overhead, client-side deduplication is implemented, ensuring that identical encrypted data blocks are stored only once. The deduplication process works as follows:


Compute a cryptographic hash (SHA-256) for the encrypted file block.

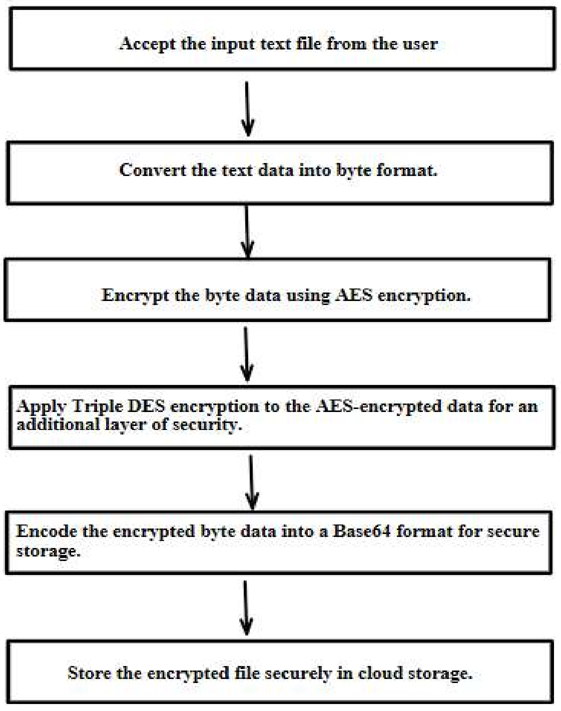
Check the hash against the database to detect duplicates.

If a matching hash is found, reference the existing block instead of uploading the file.

If no match is found, store the new encrypted block in the cloud storage.

Fig 6. Client-Side Deduplication

AES and Triple DES Encryption and Decryption Algorithm: The encryption mechanism enhances data confidentiality by employing AES and Triple DES encryption before file storage. The encryption steps are outlined below:


Accept the input text file from the user

Convert the text data into byte format.

Encrypt the byte data using AES encryption.

Apply Triple DES encryption to the AES-encrypted data for an additional layer of security.

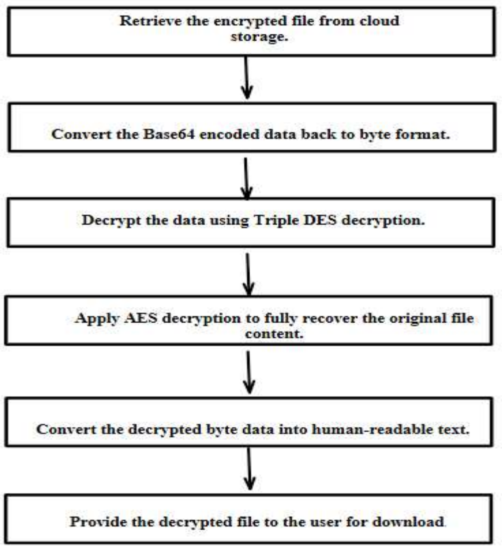
Encode the encrypted byte data into a Base64 format for secure storage.

Store the encrypted file securely in cloud storage.

Fig 7. AES and Triple DES Encryption

The decryption process retrieves and restores the original file when accessed by an authorized user. The decryption steps are as follows:

Email Notification Mechanism: To enhance user interaction, the MailSender class is implemented to notify users about important system events such as file uploads, authentication success, and deduplication reports. The notification mechanism follows these steps:


Retrieve the encrypted file from cloud storage.

Convert the Base64 encoded data back to byte format.

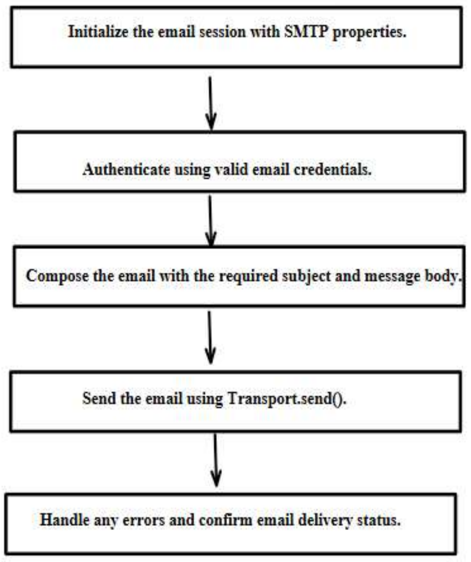
Decrypt the data using Triple DES decryption.

Apply AES decryption to fully recover the original file content.

Convert the decrypted byte data into human-readable text.

Provide the decrypted file to the user for download

Fig 8. AES and Triple DES Decryption


Initialize the email session with SMTP properties.

Authenticate using valid email credentials.

Compose the email with the required subject and message body.

Send the email using Transport.send().

Handle any errors and confirm email delivery status.

Fig 9. Email Notification

These algorithms together ensure that the Secure Text framework achieves robust encryption, efficient deduplication, and seamless communication, providing a secure and optimized cloud storage solution.

RESULTS

User Registration and Login

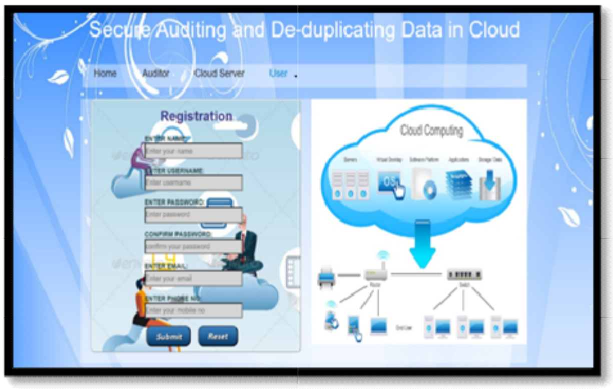


Fig 10. User Registration

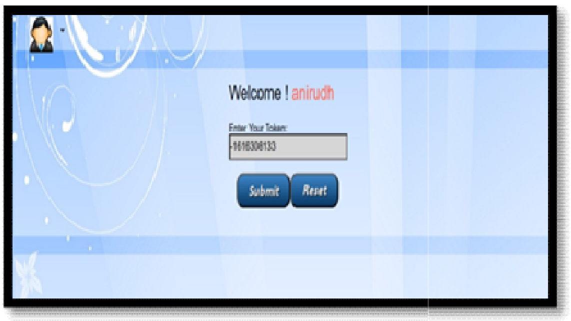
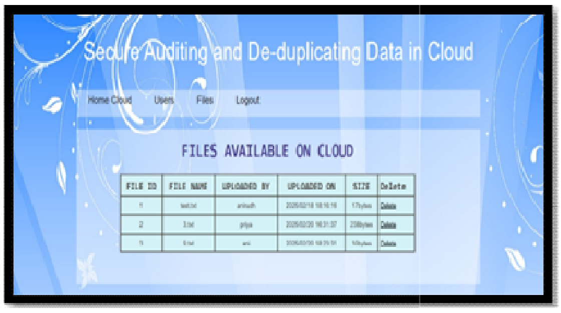


Fig 11. User Login

- **User Registration**: This step allows new users to create an account by providing their details (username, password, email, etc.). The information is securely stored in the database, ensuring that user credentials are protected. Testing will involve:
  - o Verifying that new users can successfully register.
  - o Checking that the system prevents duplicate usernames and validates input fields (e.g., email format).
  - o Ensuring that user data is correctly stored in the database.
- **User Login**: Existing users log into the system to access file management functionalities. This process involves:
  - o Validating user credentials against the stored data in the database.
  - o Testing for successful login and redirection to the appropriate user interface.
  - o Ensuring that incorrect credentials result in appropriate error messages without exposing sensitive information.

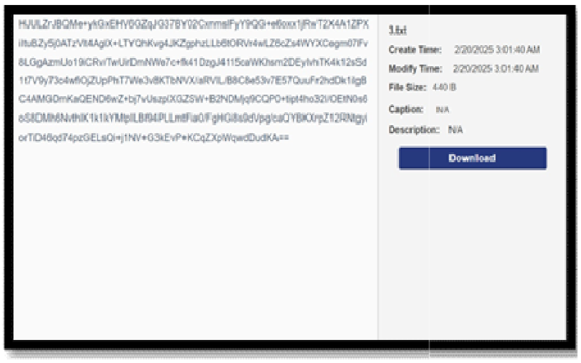
File Upload and Encryption



FILES AVAILABLE ON CLOUD

FILE ID	FILE NAME	UPLOADED BY	UPLOADED ON	SIZE	Delete
1	test.txt	anirudh	2025/02/18 18:16:16	17bytes	Delete
2	3.txt	priya	2025/02/20 16:31:37	238bytes	Delete
5	9.txt	ani	2025/02/20 18:19:59	50bytes	Delete

Fig 12. List of File Uploaded



HUVLZrJBQMe+ykGxEHY0GZqJG37BY02CxmmaIFyYI9QG+el6oxx1jRwT2X4A1ZPX
iItuBZy5j0ATzVlt4AgIX+LTYQhKvg4JKZgphzLLb6tORVr4wLZ6cZs4WYXCegm07Fv
8LGgAzmUo19iCRviTwUirDmNWe7c+fk41DzgJ4115caWKhsm2DEyIvhTK4k12sSd
1t7V9y73c4wfiOjZUpPhT7We3v8KTbNVX/aRVIL/B8C8e53v7E57QuuFr2hdDk1lgB
C4AMGDmKaQEND6wZ+bj7vUszpIXGZSW+B2NDMjq9CQPO+tipt4ho32I/OEtN0s6
oS8DMh6NvthK1k1kYMtpILB/94PLLmtFia0/FgHG8s9dVpg/caQYBKXrpZ12RNtgyi
orTiD46qd74pzGiELsQi+j1NV+G3kEvP+KCqZXpWqwdDudKA==

3.txt
Create Time: 2/20/2025 3:01:40 AM
Modify Time: 2/20/2025 3:01:40 AM
File Size: 440 B
Caption: N/A
Description: N/A
Download

Fig 13. Encrypted File

- **File Upload**: Users provide files to the system for storage and management. This process includes:
- Testing the upload functionality with various file and sizes to ensure compatibility.
- Verifying that the system correctly handles file uploads, including error handling.
- Ensuring that uploaded files are encrypted using AES and Triple DES before being stored in the database.
- **Encryption Process**: After a file is uploaded, it undergoes encryption. Testing will involve:
- Verifying that the encrypted file can be decrypted accurately, restoring the original content without loss or corruption.
- Checking that unauthorized access to the encrypted files is prevented, ensuring data security.

File Download and Decryption

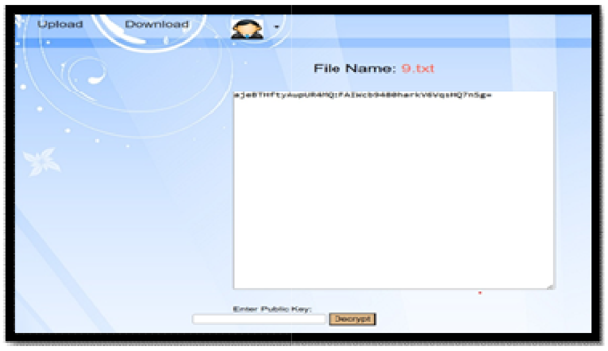


Fig 14. File Download of encrypted file

File Download: Users can download their uploaded files. This process includes:

- Testing the download functionality to ensure that users can retrieve their files without issues.
- Verifying that the downloaded files are decrypted correctly and match the original files.
- Ensuring that the system logs download activities for auditing purposes.

File Decryption: Users can Decrypt their or others uploaded files. This process includes:

- Requesting for the public and private key from the owner of the file
- Entering the key obtained from the owner both public and private key.
- The requested data can be decrypted using both the key and view the file content.

DISCUSSION

The experimental results demonstrate the effectiveness of the proposed deduplication method in reducing storage overhead while maintaining data confidentiality. By implementing client-side deduplication, the approach successfully minimizes communication overhead and prevents data leaks. The dual-layer encryption mechanism, using AES and Triple DES, enhances security and prevents side-channel attacks. Load balancing through a dedicated deduplication server significantly improves system efficiency and reduces strain on cloud servers. Performance evaluation shows a substantial reduction in storage space and faster data retrieval times compared to traditional deduplication techniques.

Furthermore, the system's ability to detect and eliminate duplicate textual data blocks contributes to cost savings and optimized cloud storage management.

CONCLUSION

In this paper, we proposed a secure deduplication method for textual data in cloud environments. By leveraging client-side deduplication and hybrid encryption, our approach ensures data security and reduces communication overhead. The use of a dedicated deduplication server for load balancing further enhances system performance and efficiency. Experimental evaluations demonstrate significant storage space reduction, improved data retrieval speed, and robust protection against data leakage. This framework is particularly beneficial for resource-constrained environments like IoT systems and edge computing applications. Future work will focus on enhancing the deduplication mechanism to handle dynamic data updates and integrating blockchain technology for secure access control.

REFERENCES

Deshingkar, H. "Data Deduplication Using Python," in *Proc. 7th International Conference on Intelligent Computing and Communication (ICCubeA)*, 2023.

M. P. and S. Suganthidevi, "Comparative Study and Secure Data Deduplication Techniques for Cloud Computing Storage," in *Proc. 2024 IEEE International Conference on Cloud Computing Technology and Science*, 2024.

M. P. and S. Suganthidevi, "Comparative Study and Secure Data Deduplication Techniques for Cloud Computing Storage," 2024.

Ahmed S. *et al.*, "FASTEN: Towards a FAult-tolerant and STorage EfficieNt Cloud: Balancing Between Replication and Deduplication," *arXiv preprint arXiv:2312.08309*, 2023.

Sehat H. *et al.*, "Bonsai: A Generalized Look at Dual Deduplication," *arXiv preprint arXiv:2202.13925*, 2022.

Ma X. *et al.*, "A Secure and Efficient Data Deduplication Scheme with Dynamic Ownership Management in Cloud Computing," *arXiv preprint arXiv:2208.09030*, 2022.

Stanek J. and L. Kencl, "Enhanced Secure Thresholded Data Deduplication Scheme for Cloud Storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1010–1023, 2018.

Xia W. *et al.*, "A Comprehensive Study of the Past, Present, and Future of Data Deduplication," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 1–22, 2016.

Sehat H. *et al.*, "Yggdrasil: Privacy-Aware Dual Deduplication in Multi Client Settings," in *ICC 2021 - IEEE International Conference on Communications*, Montreal, QC, Canada, 2021, pp. 1–6.

Wang Z. *et al.*, "Lightweight Secure Deduplication Based on Data Popularity," *IEEE Systems Journal*, vol. 17, no. 1, pp. 1–10, 2023.

Zhao Y. and S. S. M. Chow, "Updatable Block-Level Message-Locked Encryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 1–14, 2021.

Prajapati P. and P. Shah, "A Review on Secure Data Deduplication: Cloud Storage Security Issue," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 1, pp. 1–10, 2022.

Sehat H. *et al.*, "Dual Deduplication: A New Approach to Data Deduplication in Cloud Storage," *arXiv preprint arXiv:2202.13925*, 2022.

Davea D. *et al.*, "Term Memory for Data Deduplication and Data Security," *Journal of Engineering Science and Technology*, vol. 19, no. 1, pp. 1–15, 2024.

Pahlevani P. *et al.*, "Deduplication of Textual Data by NLP Approaches," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Fall)*, London, UK, 2023, pp. 1–5.

Liu F. *et al.*, "Efficient Deduplication with Secure Data Ownership Management in Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 742–754, 2020.

Xu J. *et al.*, "Reducing Data Redundancy in Cloud Storage through Secure Deduplication," *IEEE Access*, vol. 9, pp. 14392–14404, 2021.

Li R. *et al.*, "A Blockchain-based Secure Deduplication Scheme for Cloud Storage," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

Wang C. *et al.*, "Secure Data Deduplication with Efficient Key Management in Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 2099–2111, 2019.

Xue K. *et al.*, "Secure and Efficient Deduplication for Encrypted Data in Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 768–779, 2020.

Jeong T. *et al.*, "Towards Secure Data Deduplication in Cloud Storage with Dynamic Proof of Ownership," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 13–25, 2021.

Kumar A. *et al.*, "A Hybrid Approach for Secure Data Deduplication in Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 45–57, 2021.

Chen Y. *et al.*, "Privacy-Preserving Data Deduplication with Dynamic Ownership Management," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 362–375, 2021.

Fan L. *et al.*, "Secure Multi-User Data Deduplication in Cloud Storage," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 517–531, 2020.

Zhang B. *et al.*, "Privacy-Preserving Deduplication with Public Auditing in Cloud Storage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2385–2398, 2021.

****