



**JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE AND
TECHNOLOGY**

SCHOOL OF INFORMATICS AND INNOVATIVE SYSTEMS

DEPARTMENT OF COMPUTER SCIENCE SECURITY AND FORENSICS

**TITLE: A SECURE ERASURE CODE-BASED CLOUD STORAGE WITH
DATA ALGORITHM MATCHING**

CONCEPT PAPER BY:

**OMULO SAMUEL OKOTH – I132/1857 / 2020
KINOTI JOSEPH KIRERA – I132 /0281 / 2020**

**THIS CONCEPT PAPER IS PRESENTED IN PARTIAL FULFILMENT
FOR THE AWARD OF BACHELORS OF COMPUTER SECURITY AND
FORENSIC AND AS REQUEST FOR APPROVAL TO PROCEED WITH THE
FINAL YEAR PROJECT PROPOSAL AND DOCUMENTATION.**

DECLARATION

We hereby declare that this project work entitled “**A SECURE ERASURE CODE-BASED CLOUD STORAGE WITH DATA ALGORITHM MATCHING**” submitted towards partial fulfillment for the award of bachelors of computer security and Forensics is our original work and it has not been submitted for the basis for award of any diploma, degree or exam body to the best of my knowledge.

Signature.....

Date.....

Omulo Samuel Okoth

I132/1857/2020

Signature.....

Date.....

Kinoti Joseph Kirera

I132/0281/2020

UNIVERSITY SUPERVISORS APPROVAL

This project has been submitted with our approval as university supervisors.

Signature.....

Date.....

Name.....

COPYRIGHT

All rights are reserved. No part of this thesis or information herein may be reproduced, stored in a retrieval system or transmitted in any form or by any means of electronic, mechanical, photocopying, recoding or otherwise, without the prior written permission of the author or Jaramogi Oginga Odinga University of Science and Technology.

DEDICATION

This project is dedicated to the almighty God who has given us life, good health and strength to peruse this course. We would also like to dedicate this work to our families and friends for having shown us love, patience and courage towards the completion of this project.

ACKNOWLEDGEMENTS

This endeavor has taken a lot of time and work on our part. The completion of this project, however, would not have been feasible without the help and advice of a large number of people. We would like to express our heartfelt gratitude to each one of them. We owe Prof. Antony Rodriguez a huge debt of gratitude for his assistance and oversight. We would like to express our gratitude to him for giving with the information and resources we needed for this endeavor. We would want to thank our parents and friends for their unwavering support and encouragement, which has greatly aided us in finishing this project.

Our gratitude and appreciation also extend to our classmates who assisted in the development of the project. Thank you to everyone who has volunteered his or her time and talents to assist us.

ABSTRACT

Cloud computing is a delivery model for computing resources in which various servers, applications, data and other resources are integrated and provided as a service over the internet. The cloud storage system consists of a collection of key servers and storage servers. As it is evident, storing data in third party cloud systems causes serious concern on data confidentiality, so a user divides the data into blocks, encrypts the data and stores them in various designated storage servers. The storage server encodes the data using erasure code symbol. During data sharing process, a re-encryption key to the storage server is integrated. The storage server then re-encrypts the original code symbol into a re-encrypted code symbol. The re-encryption scheme is integrated with a secure decentralized erasure code so that a secure distributed system is designed. At key distribution, the proxy re-encryption scheme assists in encoding operations over encrypted data as well as forwarding operations. The key server retrieves re-encrypted code symbols and performs partial decryption so that the receiver combines the blocks to retrieve the data. By fusing the efficiency of data, matching with the resilience of secure erasure coding improves content retrieval in cloud-based contexts, optimizing storage use. Integration of an advanced Data matching algorithm which plays an important role in secure erasure code-based cloud storage, allowing for quick comparison of uploaded and downloaded data for integrity verification. Scalability and security concerns are handled through the use of machine learning and cryptography approaches. As cloud storage advances, data matching algorithms will improve security and efficiency, assuring dependable data management on the cloud.

Keywords: Decentralized erasure code, proxy re-encryption, secure storage system, data matching algorithm.

TABLE OF CONTENTS

DECLARATION	2
DEDICATION	4
ACKNOWLEDGEMENTS	5
ABSTRACT	6
TABLE OF CONTENTS	7
LIST TABLES	10
CHAPTER 1: INTRODUCTION	13
1.2 Background information	14
1.3 Executive summary	15
1.4 Problem Statement:	15
1.5 Objectives	16
1.5.1 General Objectives	16
1.5.2 Specific objectives	16
1.6 Project significance	16
1.6.1 Data Security and Integrity	16
1.6.2 Efficiency in storage Utilization	16
1.6.3 Cost Saving:	16
1.6.4 Fault Tolerance:	16
1.6.5 Efficient Data Management and Verification	16
1.6.6 Disaster Recovery:	16
1.6.7 Privacy and Confidentiality	17
1.6.8 Innovation in Cloud Services	17
1.7 Project Scope	17
1.7.1 Enhanced Information Security	17
1.7.2 Multi-Layered Access Control	17
1.7.3 Data Integrity Verification	17
1.7.4 Redundancy and Fault Tolerance	17
1.7.5 Standard Security audits	17
1.7.6 Disaster Recovery Planning	18
1.7.7 Scalability and Performance	18
1.8 Advantages over the existing systems	18
1.9 Assumptions	18
CHAPTER II	19
2.0 LITERATURE REVIEW	19

2.1	Distributed Storage Systems.....	19
2.2	Proxy Re-Encryption Scheme	19
2.3	Decentralized Erasure Code	19
2.4	Data Matching Algorithm.....	20
2.5	A Straightforward Solution	20
2.6	Existing system.....	20
2.7	Disadvantages of existing system.....	21
2.8	Proposed system	21
2.8.1	Advantages.....	21
CHAPTER III		23
3.0	METHODOLOGY	23
3.1.1	Stages of Simulation and Modeling Methodology	23
3.1.2	Advantages of Simulation and Modeling Methodology	23
3.1.3	Disadvantages of Simulation and Modeling Methodology	23
CHAPTER IV		24
4.0	SYSTEM REQUIRMENT ANALYSIS AND DESIGN	24
4.1.1	Software Requirements Specification.....	24
4.1.2	Function Requirements	24
4.1.3	Usability.....	24
4.1.4	Reliability.....	24
4.1.5	Implementation	24
4.1.6	Hardware requirements:.....	25
4.1.7	Software requirements:	25
4.1.8	Non-functional requirements	25
4.2	Logical design.....	25
4.2.1	Symbols of Flow Chart Diagram	25
4.2.2	Combined Secure Erasure code-based and Data Matching Algorithm	26
4.2.3	Secure erasure code Reed-Solomon encoding algorithm	26
4.2.4	Data matching algorithm flow chart	27
4.2.5	data integrity check algorithm	27
CHAPTER V		28
5.0	SYSTEM IMPLEMENTATION AND TESTING	28
5.1.1	Cloud Deployment	28
5.1.2	5.30 Extraction of Features and Data Capture	28
5.1.3	3. The Algorithm for Data Matching	28

5.1.4	Testing and Validation	28
5.1.5	Performance Evaluation.....	28
5.2	Physical design.....	29
CHAPTER VI.....		33
6.0	TESTING	33
6.1.1	System integration and testing	33
6.1.2	Unit testing.....	33
6.1.3	Functionality testing.....	33
6.1.4	Test for Navigation	33
6.2.0	Test cases	34
6.2.1	Positive test cases.....	34
6.3.0	USER GUIDE.....	34
CHAPTER VII.....		36
7.0	FINDINGS AND CONCLUSION	36
7.1	FINDINGS.....	36
7.1.0	Data Matching Algorithm Efficiency	36
7.1.1	Secure Erasure Coding Effectiveness	36
7.2	CONCLUSION.....	36
APPENDIX I		37
8.0	PROJECT TIMELINE.....	37
APENDIX II		38
9.0	COST BENEFIT ANALYSIS	38
REFERENCES		39

LIST TABLES

Table 1. positive test cases.....	34
Table 2. Negative Test Case	34
Table 3 Project Timeline.....	37
Table 4 Cost benefit analysis	38

LIST OF FIGURES

Figure 1: Symbols for the flow chart diagram	25
Figure 2 Flowchart diagram for secure erasure code and Data matching algorithm	26
Figure 3 Flow chart diagram for data matching algorithm	27
Figure 4. Erasure encoding	29
Figure 5. erasure decoding	30
Figure 6. Data preprocessing	30
Figure 7. Data matching cord.....	31
Figure 8. Data integrity check.....	31
Figure 9. Erased data	32
Figure 10. file upload.....	34
Figure 11. prompt on successful	35
Figure 12. File download	35
Figure 13 Data mismatch	35

List of acronyms and abbreviations

RAM- Random Access Memory

IDE- Integrated Development Environment

EM – Expectation-Maximization

NAS - Network-Attached Storage

NFS – Network File System

BoW - Bag-of-Words

TF-IDF - Term Frequency-Inverse Document Frequency

PCA - Principal Component Analysis.

OpenCV – open Computer Vision

MFCC - Mel-frequency cepstral coefficients

Definition of terms

Cloud - Virtual Storage System

Node(s) - A storage node is a node that provides remote data storage and distribution services

Secure erasure code - Process of dividing data into smaller datasets and distributing them across different storage nodes

Cypher text - An encrypted text.

Code word – A phrase or symbol that has a secret meaning.

CHAPTER 1: INTRODUCTION

Cloud computing is the use of computing resources that are delivered as a service over a network. For example, the email service is probably the most popular one (Aatish & Avinash, 2020). Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed. This paper focuses on designing a cloud storage system for robustness, confidentiality, and functionality. It also aims at creating redundancy through Load Balancing in the cloud storage. Lin & Tzeng (2012) emphasize that cloud storage system is considered as a large scale distributed storage system that consists of many independent storage servers and usually contains terabytes of data especially when data is compressed before storage.

Ensuring the resilience of data is a crucial necessity for storage systems (Liu et al., 2020). One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is usually huge because the message can be retrieved as long as one storage server survives (Nivedhaa & Justus, 2018). Another way is to encode a message of k symbols into a code word of n symbols by erasure coding. To store a message, each of its code word symbols is stored in a different storage server. A storage server failure corresponds to an erasure error of the code word symbol. As long as the number of failure servers is under the tolerance threshold of the erasure code, the message can be recovered from the code word symbols stored in the available storage servers by the decoding process.

Rashmi et al (2017) discusses that a decentralized erasure code is a code that independently computes each compressed code symbol for each message. A decentralized erasure code is suitable for use in a distributed storage system. After the compressed message symbols are sent to storage servers, each storage server independently computes a code word symbol for the received message symbols and stores it. This finishes the encoding and storing process.

It is well known that storing data in a third party's cloud system causes serious concern on data confidentiality. To provide strong confidentiality for messages in storage servers, a user encrypts messages by a cryptographic method before applying an erasure code method to encode and store messages (Vasanthi et al., 2021). When they want to use a message, they need to retrieve the code word symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. There are three problems in the above straightforward integration of encryption and encoding.

First, the user has to do most computation and the communication traffic between the user and storage servers which is high. Second, the user has to manage his cryptographic keys. If the user's device used to store the keys is lost or compromised, the security is broken.

We propose a data matching algorithm and integration into secure erasure code-based cloud storage systems is a significant advancement in ensuring data integrity and reliability. This algorithm serves as vital for effectively comparing uploaded and downloaded data, as they use approaches for unique feature extraction and comparison. These methods verify data authenticity by examining metadata, file attributes, and content fingerprints, highlighting matches and mismatches. This integration improves the system's ability to detect discrepancies, giving the consumer confidence in data integrity. In this regard, the introduction emphasizes the significance of including data matching algorithms to strengthen secure erasure code-based cloud storage systems.

The contributions are:

The data matching algorithm in cloud storage assumes the responsibilities of data integrity verification through comparison and checksum validations, the algorithm ensures that the stored data remains unaltered and free from corruption. The data matching algorithm's responsibilities extend beyond mere pattern recognition; which actively contributes to tenets of data management, optimizing resource utilization and safeguarding the integrity of information in cloud storage.

1.2 Background information

As more individuals and organizations are relocating their information to the cloud, the security and dependability of cloud storage arrangements have become progressively significant. Traditional cloud storage systems depend on replication for data redundancy, which can be exorbitant as far as storage space and is vulnerable to data breaches. Erasure coding is an effective option in contrast to replication, yet it presents security concerns and difficulties in cloud storage systems e.g. reconstruction attacks where an attacker might attempt to corrupt data while in the reconstruction process thus losing integrity. The problem lies in developing a secure erasure code-based cloud storage system that offers data matching algorithm while encapsulating data redundancy and data security. However, a solution lies in matching data, such as documents, music, and videos specific algorithms available, with their exact original copies as uploaded by the user and showing the results as a match or a mismatch in the case of data corruption and integrity violations. This approach offers significant benefits, including enhancements in data integrity through efficient comparison of the uploaded and downloaded data, ensuring authenticity and reliability, additionally the use of unique data attributes and comparison techniques ensures accurate detection of discrepancies therefore minimizing the risk of data corruption or loss, providing users with robust data management solutions.

1.3 Executive summary

Secure Erasure Code-Based Cloud Storage with Data Matching Algorithm In the evolving landscape of cloud storage, the fusion of secure erasure code-based methodologies with advanced data matching algorithms heralds a transformative approach to data integrity, efficiency, and security. This innovative integration combines the robustness of erasure coding techniques, ensuring fault tolerance and optimized storage utilization, with the precision and intelligence of data matching algorithms, refining the handling and retrieval of data within cloud environments. This comprehensive solution addresses fundamental challenges in contemporary cloud storage systems. The utilization of secure erasure code-based principles minimizes redundancy while maintaining data integrity. Simultaneously, the incorporation of sophisticated data matching algorithms empowers efficient detection and notification of a mismatch, enhancing storage optimization and minimizing resource waste. The synergy between Secure Erasure Code-Based Cloud Storage and Data Matching Algorithms not only fortifies data reliability but also significantly advances operational efficiency. This innovative framework not only ensures robust fault tolerance and data recovery but also optimizes resource utilization, making it a cornerstone for large-scale cloud storage systems and revolutionizing the way organizations manage, secure, and extract value from their data assets.

1.4 Problem Statement:

As more individuals and organizations are relocating their information to the cloud, the security and dependability of cloud storage arrangements have become progressively significant. Traditional cloud storage systems depend on replication for data redundancy, which can be exorbitant as far as storage space and is vulnerable to data breaches. Erasure coding is an effective option in contrast to replication, yet it presents security concerns and difficulties in cloud storage systems. The problem lies in developing a secure erasure code-based cloud storage system that offers data matching algorithm while encapsulating Integrity and data security. However, a solution lies in matching data, such as documents, music, and videos specific algorithms available, with their original user uploaded copies. By doing so, these data can be stored in the cloud on erasure and later integrity assessed on user request download. This approach offers significant benefits, including enhancements in data integrity through efficient comparison of the uploaded and downloaded data, ensuring authenticity and reliability, additionally the use of unique data attributes and comparison techniques ensures accurate detection of discrepancies therefore minimizing the risk of data corruption or loss, providing users with robust data management solutions.

1.5 Objectives

1.5.1 General Objectives

- a. To implement Data matching Technique to ensure data integrity

1.5.2 Specific objectives

- a. To Study Different Types of Data Algorithm Matching.
- b. To Analyze and Optimize Data verification Process.
- c. To Implement Backup and Disaster Recovery.
- d. To Conduct Monitoring and Management of Data.

1.6 Project significance

Secure Erasure Code-based Cloud Storage with data matching algorithm signifies a forward-looking and robust approach to addressing the challenges associated with data storage in the cloud. It aligns with the growing demands for data security, integrity, efficiency, and scalability in the digital landscape and reveals its importance in the realm of cloud computing and data storage due to the below key factors:

1.6.1 Data Security and Integrity

Erasure coding integrated with Data matching algorithm enables detection of discrepancies, enhancing data integrity and enabling prompt action against unauthorized alterations, when implemented securely, enhances data protection by dispersing and encoding information across multiple locations. Which helps mitigate the risk of data loss, corruption, or unauthorized access.

1.6.2 Efficiency in storage Utilization

- Erasure coding provides a more efficient method of redundancy compared to traditional replication. This means that data can be stored with a lower storage overhead, optimizing storage resources while maintaining data integrity and availability.

1.6.3 Cost Saving:

The efficiency in storage utilization can lead to cost savings, as organizations can store the same amount of data with less storage infrastructure. This is particularly valuable as data volumes continue to grow, and storage costs become a significant factor for businesses.

1.6.4 Fault Tolerance:

Erasure coding enhances fault tolerance by allowing data recovery even if a portion of the storage infrastructure becomes unavailable or experiences failures. This is crucial for maintaining continuous access to data and preventing service disruptions.

1.6.5 Efficient Data Management and Verification

Data comparison automation processes streamlines verification, reducing human error and enhancing operational efficiency. Users gain from quicker decision-making and response time, especially in critical scenarios.

1.6.6 Disaster Recovery:

The distributed nature of erasure coding supports robust disaster recovery strategies. In the event of a data center failure or natural disaster, data can be reconstructed from the remaining fragments, ensuring minimal data loss and downtime.

1.6.7 Privacy and Confidentiality

Security is paramount in cloud storage, especially when dealing with sensitive or confidential information. The approach contributes to the privacy and confidentiality of stored data, protecting it from unauthorized access and cyber threats.

1.6.8 Innovation in Cloud Services

Secure erasure code-based storage represents an innovative approach to cloud services, pushing the boundaries of traditional storage methods. It reflects the industry's commitment to evolving technologies that offer improved security, efficiency, and reliability.

1.7 Project Scope

The project's goal is to integrate data matching algorithm into secure erasure code-based cloud storage system to improve data integrity, security, and efficiency. The scope includes investigating existing data matching algorithms, determining their viability for integration, and implementing the chosen method in the cloud storage. Testing and validation will be performed to confirm that the algorithm's functionality, performance, and security fulfill the project's requirements. The project scope encompasses detecting and addressing any issues or limitations that arise during implementation, with the ultimate goal of providing a strong and dependable cloud storage solution with integrated data matching capabilities.

1.7.1 Enhanced Information Security

Foster an eradication code-based distributed storage framework that integrates progressed encryption strategies to guarantee information security. Utilize end-to-end encryption to safeguard information at still and on transit. This ought to incorporate secure key management and encryption algorithms that are impervious to known attacks.

1.7.2 Multi-Layered Access Control

Execute robust access control mechanisms to forestall unauthorized access. This incorporates role-based access control, two-factor authentication, and audit trails. Characterize access define access policies and routinely monitor and update them to adjust to changing security threats.

1.7.3 Data Integrity Verification

Coordinate verification of data integrity mechanisms that detect and correct errors in the stored data. This should be possible through checksums, cryptographic hash capabilities, or other error-checking strategies to guarantee that information stays in one piece and unaltered.

1.7.4 Redundancy and Fault Tolerance

Execute erasure coding procedures to provide redundancy without the need for full replication. Disperse encoded data across multiple servers or data centers to guarantee fault tolerance. Use techniques, for example, Reed-Solomon codes to provide redundancy while minimizing storage overhead. Storing data in geographically distributed data centers enhance data accessibility. This guarantees data to remains available even in the case of regional outages, data center failures or catastrophic events.

1.7.5 Standard Security audits

Direct regular security audits and conduct pentesting to quickly recognize eminent vulnerabilities and address them immediately. Continuous monitoring and proactive threat detection can assist with keeping up with the security of the cloud storage system.

1.7.6 Disaster Recovery Planning

Foster a comprehensive disaster recovery plan for quick data restoration in the in case of a major outage, data corruption, or other catastrophic events. Consistently test and update this intend to guarantee its adequacy.

1.7.7 Scalability and Performance

Plan the system to be versatile, guaranteeing that it can deal with the developing data requests of clients and organizations without compromising performance or security.

1.8 Advantages over the existing systems

- a. Unlike storage other cloud technologies, data matching algorithms reduces Integrity violations by selectively identifying and compares unique instances of data.
- b. By intelligently recognizing similarities and Alerting Corruption and integrity violations across extensive datasets, this algorithm significantly minimizes the attack surface, optimization of storage resources without compromising data integrity.
- c. Erasure coding with Data matching algorithm in cloud storage systems enhance fault tolerance and data retrieval efficiency. In contrast to replication-based systems that rely on full data copies for recovery, these algorithms employ redundant information strategically generated through mathematical computations.
- d. This method ensures data recoverability even on the possibility of individual fragments or symbols are lost.
- e. It is cost effective and improves scalability, rendering them highly advantageous in large-scale cloud storage environments compared to traditional storage replication methods.

1.9 Assumptions

- a. It is assumed that the users of this system have the knowledge of uploading and downloading data into the cloud storage servers.
- b. It is assumed that the subject systems have secondary storages of 16GB minimum for optimal data disintegration during the secure erasure coding.

CHAPTER II

2.0 LITERATURE REVIEW

We briefly review distributed storage systems, proxy re-encryption schemes, decentralized erasure code and data matching algorithm.

2.1 Distributed Storage Systems

Xu et al (2020) clarifies that users can access store data via network connection provided by the storage devices over the network from the Network-Attached Storage (NAS) and the Network File System (NFS). Distributed architecture for storage systems offer good scalability, due to their ability to either enter or exit without the control of a central authority. To provide robustness against server failures, a simple method is to make copies of each data and store them in different servers. One way to reduce the expansion rate is to use erasure codes to encode data (Balaji et al., 2018). Data is encoded as a code word, which is a vector of symbols, and each storage server stores a code word symbol. A storage server failure is modeled as an erasure error of the stored code word symbol. To store a message of k blocks, each storage server linearly combines the blocks with randomly chosen coefficients and stores the code word symbol and coefficients. To retrieve the message, a user queries k storage servers for the stored code word symbols and coefficients and solves the linear system. The system has light data confidentiality because an attacker can compromise k storage servers to get the message.

2.2 Proxy Re-Encryption Scheme

In a proxy re-encryption scheme, a proxy server can convey a cipher text under a private key A to a new one under another public key B (Nivedhaa & Justus, 2018). The server does not know the plaintext during transformation. The data is first encrypted with a symmetric data encryption key and then stored in the cloud storage server. The cloud storage server uses a re-encryption algorithm to transfer the encrypted DEK into the format that can be decrypted by the recipient's private key. The recipient then can download the encrypted data from the cloud and use the DEK for decryption. A re-encryption key is generated from the data owner's private key and a recipient's public key. A data owner may share different files with different recipient groups. Therefore, a recipient cannot read data for a group it does not belong to. The cloud, on the other hand, acts as an intermediate proxy. It cannot read the data as it cannot get DEKs. Thus, the system has data confidentiality and supports the data forwarding function

2.3 Decentralized Erasure Code

Aatish & Avinash (2020) wrote that a decentralized erasure code refers to method of data encoding and storage that distribute redundancy and error correction information across multiple independent dispersed nodes without relying on a centralized. The generator matrix G constructed by an encoder is as follows: First, for each row, the encoder randomly marks an entry as 1 and repeats this process for an $\ln k$ times with replacement. Second, the encoder randomly sets a value from IF for each marked entry. This finishes the encoding process. A decoding is successful if and only if $k \times k$ submatrix formed by the k -chosen columns is invertible. Thus, the probability of a success decoding is equivalent to that of the chosen sub matrix being invertible.

The owner randomly selects v servers with replacement and sends a copy of M_i to each of them. Each server randomly selects a coefficient for each received cipher text and performs a linear combination of all received cipher texts (Lin & Tzeng, 2012). Those coefficients chosen by a server form a column of the matrix and the result of the linear combination is a code word

element. Each server can perform the computation independently. This makes the code decentralized (Vasanthi et al., 2021).

2.4 Data Matching Algorithm

Data matching algorithms are important tools in many real-world applications because they allow for reliable comparison and identification of comparable information across datasets. These algorithms are critical in a wide range of industries, including healthcare, banking, e-commerce, and data management. Probabilistic record linking approaches, such as Fellegi-Sunter and Expectation-Maximization (EM), are often employed in cases when datasets lack unique identifiers or contain errors (Sayers et al., 2015). String matching methods like as Levenshtein distance, Jaccard similarity, and cosine similarity perform well in text-based matching tasks, even in the presence of typographical errors. Indolia et al (2018) discussed that Machine learning techniques, such as supervised learning models (e.g., Support Vector Machines and Random Forests) and unsupervised learning algorithms (e.g., k-means clustering), use labeled or unlabeled data to classify or group related records. Deep learning techniques, particularly neural networks, have showed potential in harnessing Convolutional and recurrent architectures are used for accurate record linking. Feature extraction techniques such as Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Principal Component Analysis (PCA) are essential for translating raw data into comparable characteristics. These systems provide a variety of record matching solutions, with continuous research and development promising even greater accuracy and scalability (Gollapudi, 2019).

2.5 A Straightforward Solution

A straightforward solution to supporting the data forwarding function in a distributed storage system is as follows: when the owner A wants to forward a message to user B, he downloads the encrypted message and decrypts it by using his secret key. He then encrypts the message by using B's public key and uploads the new cipher text. When B wants to retrieve the forwarded message from A, he downloads the cipher text and decrypts it by his secret key. The whole data forwarding process needs three communication rounds for A's downloading and uploading and B's downloading. The communication cost is linear in the length of the forwarded message. The computation cost is the decryption and encryption of the owner A, and the decryption for user B. Proxy re-encryption schemes can significantly decrease communication and computation cost for the owner. In a proxy re-encryption scheme, the owner sends a re-encryption key to storage servers such that storage servers perform the re-encryption operation for him. Thus, the communication cost of the owner is independent of the length of forwarded message and the computation cost of re-encryption is taken care of by storage servers. Proxy re-encryption schemes significantly reduce the overhead of the data forwarding function in a secure storage system.

2.6 Existing system

Erasure code-based storage approaches to minimize redundancy while maintaining data integrity while consuming considerable space to ensure fault tolerance. Reed-Solomon method dramatically reduces the storage overhead compared to traditional replication. This optimized storage computation significantly reduces the overall storage size needed, but in return stores data from users as loads which brings forth a lump sum of data to the servers.

2.7 Disadvantages of existing system

In secure erasure code-based storage systems that lack integrated data matching techniques, the lack of real-time comparison between uploaded and downloaded feature extractions is a significant drawback. Without this capacity, the system cannot quickly detect differences between the original and returned data. As a result, potential data corruption or illegal modifications go unnoticed, compromising data integrity and security. Manual verification techniques are inefficient and error-prone, which causes delays in discovering and resolving issues. The system's inability to automate data matching reduces its response to security risks and compromises overall data integrity.

2.8 Proposed system

In the proposed system, we propose a data matching algorithm while integrating it with a decentralized erasure code such that a secure cloud storage system is designed. By using the data matching algorithm, we present a cloud storage system that is secure that enhances scalability, provides more space for storage while encapsulating secure data forwarding functionality in a decentralized structure. The distributed storage system is not only secure and but also enables the system to do a matching of the data that the user is backing up before encryption, safely retrieves the metadata and encrypts it and stores it. Our method fully integrates similarity metrics, pattern recognition techniques, data loading and preprocessing, encoding and decoding.

We propose a new data matching algorithm and integrates it with a secure decentralized code to form a secure cloud storage system. The algorithm holds critical responsibilities that contribute to the security, efficiency and overall effectiveness of the storage ecosystem. This algorithm plays a pivotal role in data deduplication, by detection and notification of a mismatch, enhancing storage optimization. By employing sophisticated matching techniques, the algorithm ensures that only the unique data instances are stored, optimizing storage space thus achieving scalability and mitigating the risks associated with data inconsistency. In realizing the responsibility, the system identifies integrity violations of data and fostering a streamlined and space efficient cloud environment. Through the comparisons and checksums validations, the algorithm checks that stored data has remains unaltered and free from corruption. The tight integration of similarity metrics, pattern recognition techniques, data processing, encrypting and encoding makes the storage system efficiently meet the requirements of data robustness. Regular validation of the integrity of stored data, the algorithm acts as a safeguard against potential errors, accidental corruption, or unauthorized modification, thereby upholding reliability. Accomplishing the integration with considerations of a cloud structure is challenging. Moreover, we consider the system in a more general setting than previous works. This setting contributes to the core tenets of managing data, efficient resource utilization and securing the integrity of information in cloud storage.

2.8.1 Advantages

The proposed data matching technique has various benefits for secure erasure code-based storage systems, including improved data integrity, security, and efficiency. First, by incorporating the technique within the storage system, real-time comparison of uploaded and downloaded feature extractions is enabled. This allows for the instant discovery of any inconsistencies between the original and recovered data, reducing the danger of undiscovered data corruption or unauthorized modifications. Furthermore, the automated nature of the data matching method streamlines the verification process, minimizing the need for manual checks, which are prone to errors and delays. This leads to faster issue detection and treatment, which improves overall system responsiveness.

Furthermore, the algorithm's capacity to automate data matching makes the system more resilient to security threats and breaches. The system enables proactive efforts to limit potential threats by quickly spotting discrepancies or suspicious patterns, so protecting data integrity and confidentiality. Furthermore, incorporating feature extraction techniques into the data matching process improves comparison accuracy and efficiency, resulting in exact detection of matches and mismatches between datasets. Overall, the proposed data matching algorithm transforms secure erasure code-based storage systems by providing unprecedented levels of data integrity, security, and operational efficiency.

CHAPTER III

3.0 METHODOLOGY

3.1 SIMULATION AND MODELLING

In the context of this project, simulation and modeling methodologies play a crucial role in the design, evaluation, and optimization of the proposed solution. Below are the stages, advantages, and disadvantages of using simulation and modeling methodology for this project:

3.1.1 Stages of Simulation and Modeling Methodology

- a. Problem Definition
- b. Model Development
- c. Simulation Implementation
- d. Data Generation
- e. Experimentation and Analysis
- f. Optimization
- g. Validation

3.1.2 Advantages of Simulation and Modeling Methodology

- a. Cost-Effective
- b. Risk-Free Testing without risking data loss or security breaches.
- c. Iterative Development enabling quick iterations and refinement of the secure erasure code and data matching algorithm.
- d. Scenario Exploration by allowing for testing under various scenarios, including different data distributions, network conditions, and attack scenarios, to assess the robustness and security of the solution.

3.1.3 Disadvantages of Simulation and Modeling Methodology

- a. Assumption Limitations
- b. Model Validation Challenges.
- c. Complexity Management
- d. Potential Bias
- e. Limited Realism

CHAPTER IV

4.0 SYSTEM REQUIREMENT ANALYSIS AND DESIGN

4.1.1 Software Requirements Specification

A software requirements specification is developed as a consequence of analysis. Review is essential to ensure that the developer and customers have the same perception.

Software requirements specification (SRS) is the starting point of the software development activity. The Software Requirements Specification is produced at the culmination of the analysis task. The introduction of the software requirements specification states the goals and objectives of the software, describing it in the context of the computer based system. The software requirements specification includes an information description, functional description, behavioral description, validation criteria.

4.1.2 Function Requirements

These requirements will be categorized into software and hardware requirements and they will be responsible for the development of the system. All hands-on operations will directly depend on these requirements in order for them to function and conduct their functions effectively.

- a. Data Uploading
- b. Feature Extraction
- c. Erasure coding
- d. Receiving Data
- e. Data Matching
- f. Download Data

4.1.3 Usability

The system is designed with completely automated process hence there is no or less user intervention. The end user can easily navigate the entire system as it is developed in windows application. The application gives the status messages regularly based on the user actions performed. Thus the access to this system is very easy.

4.1.4 Reliability

The system is more reliable because of the qualities that are inherited Python language. This application does not depend on the external memory and hence its space complexity is very less.

4.1.5 Implementation

The system is implemented in windows environment using python language. The user interface is designed in such a way that it is very flexible and can be easily accessed by the end users.

4.1.6 Hardware requirements:

- Processor - intel
- Base - 64-bit system
- Speed - 2.6 Ghz
- RAM - 4 Gb (min)
- Hard Disk - standard
- Key Board - Standard Windows Keyboard

4.1.7 Software requirements:

- Operating System : Windows 10 professional
- Application Server : Flask Localhost Sever
- IDE : Visual studio, Pycharm
- Script language : Python

4.1.8 Non-functional requirements

- Accurate performance of the system efficiently and immediately.
- A user-friendly system with intuitive interfaces.
- The programming language should be user friendly

4.2 Logical design

4.2.1 Symbols of Flow Chart Diagram

Flow chart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

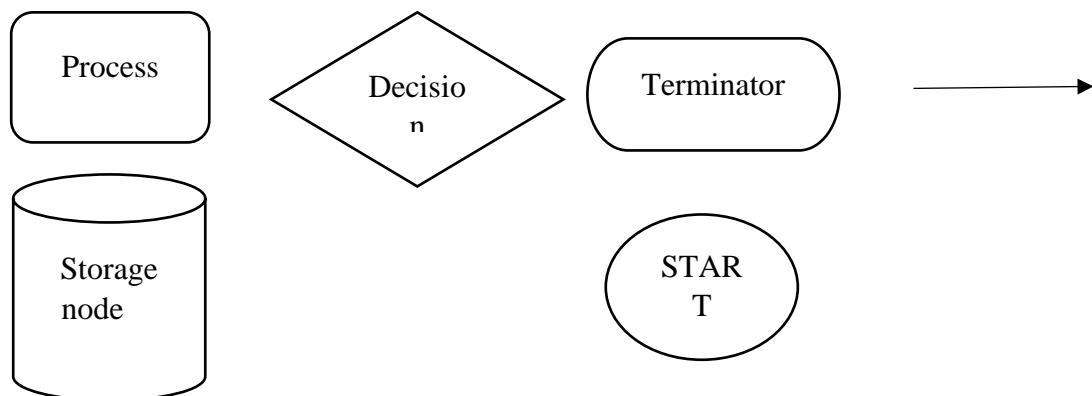


Figure 1: Symbols for the flow chart diagram

4.2.2 Combined Secure Erasure code-based and Data Matching Algorithm

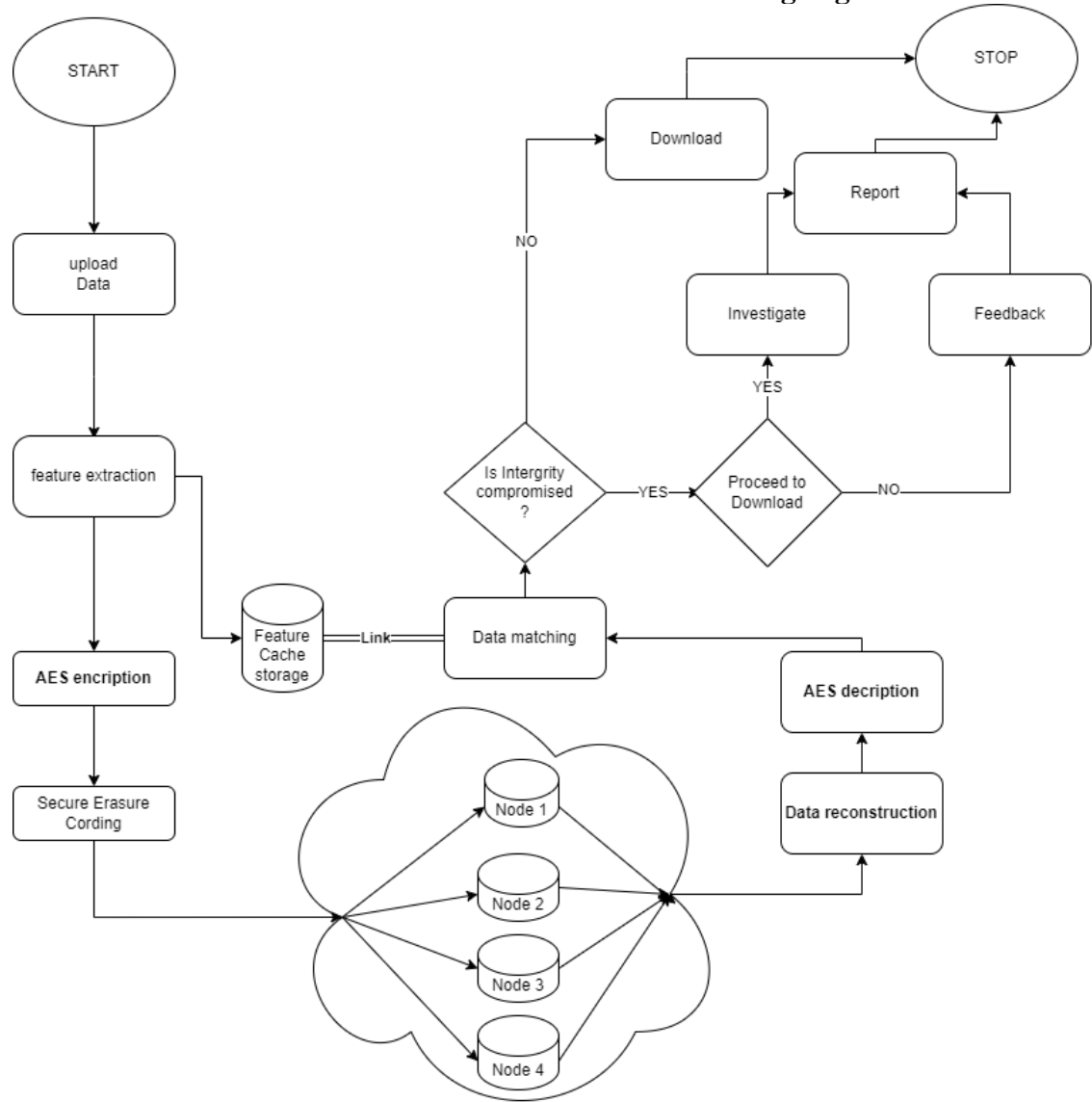


Figure 2 Flowchart diagram for secure erasure code and Data matching algorithm

4.2.3 Secure erasure code Reed-Solomon encoding algorithm

N: Total number of storage nodes.

K: Number of data blocks.

F: Maximum number of allowable node failure without data loss recovery.

GF(q): Galois field with q elements, where q is prime.

$f: D \rightarrow D = D$

$D = [d_1, d_2, \dots, d_n]$

$d'_i = f(d_i)$

Where;

(d_i') = secure erased data

$D' = [d_1', d_2', \dots, d_n']$

$\forall i \in [1, n], V(d_i') = 1$ (for each byte d_i' in D' , the verification function V returns 1)

4.2.4 Data matching algorithm flow chart

Secure erasure code and data matching algorithm takes a perfect path in ensuring the integrity of downloaded data. The data matching algorithm in this case helps in comparing data with its pre-scanned features in feature cache storage to ensure data integrity. System users have the privilege to make a download decision regardless of the integrity result Data matching typically involves comparison and identification of similarities between different data sets, often within semi-structured data stored as binary large objects. Process may vary depending on nature of the data.

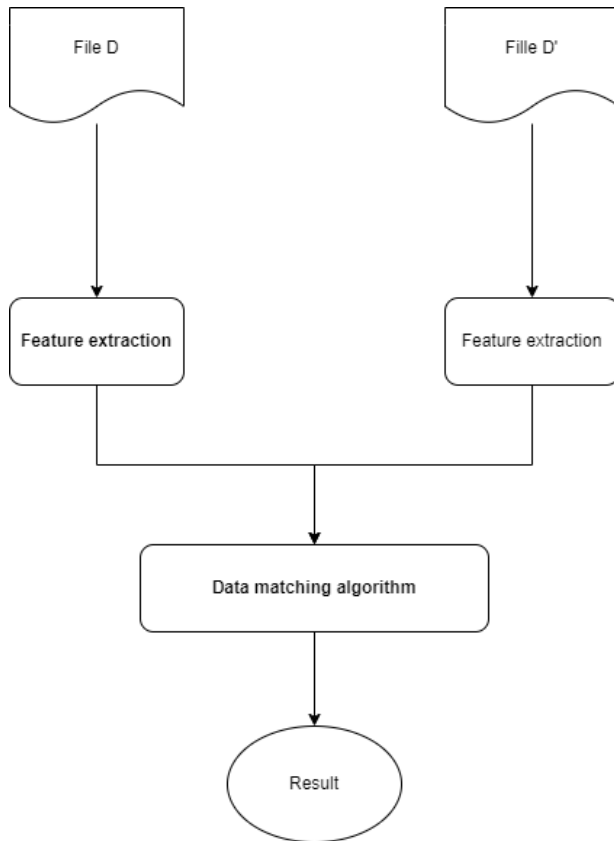


Figure 3 Flow chart diagram for data matching algorithm

4.2.5 data integrity check algorithm

Let $D = [d_1, d_2, \dots, d_n]$

$D' = [d_1', d_2', \dots, d_n']$

Compare each byte d_i' in D' with the corresponding byte d_i in D

If $D \neq D'$ for any i , flag an integrity violation

If $d \neq d_i'$ for any i , flag an integrity violation

CHAPTER V

5.0 SYSTEM IMPLEMENTATION AND TESTING

As we focus on the practical implementation of our secure erasure coding system in the "Implementation and Testing" phase of our project, with a special emphasis on the integration of a data matching algorithm. During this phase, we will deploy our system and thoroughly test its security and functioning. Our goal for the "Implementation and Testing" phase is to provide a reliable and safe cloud-based solution for sensitive data storage, together with a powerful data matching algorithm that guarantees data integrity and makes access and retrieval easy.

5.1.1 Cloud Deployment

To facilitate easy and scalable storage and retrieval of sensitive data, our data-matching algorithm-equipped secure erasure coding system is put onto cloud infrastructure. This deployment keeps the strongest security standards in place while guaranteeing user accessible from any location with an internet connection.

5.1.2 5.30 Extraction of Features and Data Capture

Upload Procedure: Our system recodes the attributes of the uploaded file when a user uploads data to the cloud. Time stamps, file sizes, metadata, and other pertinent information are a few examples of these qualities. Furthermore, the data is encoded using secure erasure coding techniques to guarantee data integrity and fault tolerance while being stored.

Download Process: In a similar vein, our system recodes the attributes of the files that users download from the cloud. In order to compare these features with the features of the original uploaded file, they are retrieved and preserved.

5.1.3 3. The Algorithm for Data Matching

Comparing Features a data matching algorithm is used to compare the features of the downloaded file with those of the original uploaded file. This formula examines a number of the files' properties, to verify if the files match.

- **Facilitating Match:** Encouraging a match between the downloaded and uploaded files is the main objective of the data matching algorithm. A match is created if, within a predetermined threshold, the characteristics of the downloaded file closely match those of the original uploaded file.

5.1.4 Testing and Validation

- **Functional Testing:** We carry out comprehensive testing to validate the proper functioning of the data matching algorithm and secure erasure coding system. This entails testing a range of scenarios, including file uploads, downloads, and matches, under varied workloads and conditions.

5.1.5 Performance Evaluation

- **Scalability:** We evaluate the scalability of our solutions by observing how well it performs under various user demands and datasets. In order to ascertain the system's ability to manage high data volumes and multiple user requests at once, stress testing is required

- **Efficiency:** We assess the computational resources, storage overhead, and reaction times of our data matching algorithm and secure erasure coding scheme. When necessary, optimization techniques are used to improve system performance.

5.1.6. Continuous Improvement:

We make iterative improvements to our implementation to address any issues found, improve system stability, and introduce new features or optimizations based on testing results and user input. This iterative process guarantees that our solution adapts to the changing requirements and difficulties of secure data storage and matching in the cloud environment.

5.2 Physical design

5.2.1 Erasure encoding

In the implementation phase of erasure coding, encoding involves dividing the data into blocks, performing matrix multiplication between the data blocks and a predefined parity matrix to generate redundant parity blocks, and then appending these parity blocks to the original data blocks to create encoded blocks ready for storage or transmission.

```
from hashlib import sha256
from random import randint
from sympy import Matrix

app = Flask(__name__)

class SecureErasureCode:
    def __init__(self, k, m):
        self.k = k # Number of data blocks
        self.m = m # Number of parity blocks
        self.n = k + m # Total number of blocks
        self.parity_matrix = self.generate_parity_matrix()

    def generate_parity_matrix(self):
        matrix = []
        for i in range(self.k):
            row = []
            for j in range(self.m):
                row.append(randint(0, 255))
            matrix.append(row)
        return Matrix(matrix)

    def encode(self, data):
        if len(data) % self.k != 0:
            raise ValueError("Data length must be divisible by k")
        encoded_blocks = []
        for i in range(0, len(data), self.k):
            block = data[i:i + self.k]
            block_matrix = Matrix([list(block)])
            parity_block = (block_matrix * self.parity_matrix).tolist()[0]
            encoded_parity_block = bytes((x % 256) for x in parity_block)
            encoded_blocks.append(block + encoded_parity_block)
        return encoded_blocks
```

Figure 4. Erasure encoding

5.2.2 Erasure decoding

Decoding involves retrieving the original data from the encoded blocks. This process typically includes reconstructing missing or corrupted data blocks using the available redundant parity blocks and performing matrix operations, such as matrix inversion, to recover the original data. Once the missing or corrupted data blocks are reconstructed, the original data can be retrieved for further processing or usage.

```
def decode(self, encoded_blocks):
    if len(encoded_blocks[0]) % (self.k + self.m) != 0:
        raise ValueError("Encoded block size must be divisible by k+m")

    data_blocks = []
    for encoded_block in encoded_blocks:
        data_blocks.append(encoded_block[:self.k])

    data_matrix = Matrix([list(block) for block in data_blocks])

    # Pad the parity matrix to make it square
    if self.m < self.k:
        self.pad_parity_matrix()

    inverse_matrix = self.parity_matrix.inv_mod(256)
    decoded_blocks = []

    for encoded_block in encoded_blocks:
        encoded_block_data = encoded_block[:self.k]
        encoded_block_parity = encoded_block[self.k:]
        reconstructed_parity = (data_matrix * inverse_matrix).tolist()[0]
        reconstructed_block = bytes([a ^ b for a, b in zip(encoded_block_parity, reconstructed_parity)])
        if sha256(encoded_block_data).digest() == sha256(reconstructed_block).digest():
            decoded_blocks.append(reconstructed_block)
        else:
            print("Data integrity check failed. Block may have been tampered with.")
            return None
    return decoded_blocks
```

Figure 5. erasure decoding

5.2.3 Preprocessing data

The preprocessing phase in data matching involves standardizing the raw input data to ensure consistency across different types, such as images, audio, documents, or videos, before feature extraction and comparison. This step optimizes data for accurate matching and integrity assessment.

```
def load_and_preprocess_data(file_path, file_type):
    if file_type == 'image':
        return load_and_preprocess_image(file_path)
    elif file_type == 'audio':
        return load_and_preprocess_audio(file_path)
    elif file_type == 'document':
        return load_and_preprocess_document(file_path)
    elif file_type == 'video':
        return load_and_preprocess_video(file_path)
    else:
        raise ValueError("Invalid file type. Supported types are 'image', 'audio', 'document', 'video'.")
```

Figure 6. Data preprocessing

5.2.4 Data Matching

In simple terms, the data matching in the code involves comparing two files to see how similar they are. This comparison is done by first preparing the files for comparison (preprocessing) and then looking at certain characteristics of the files (features) to see how close they are using tools like OpenCV, TF-IDF and MFCC. Based on this comparison, the code determines the level of similarity between the files and checks if there are any signs of tampering or corruption.

```
def compare_features(features1, features2):  
    mse = np.mean((features1 - features2) ** 2)  
    return mse
```

Figure 7. Data matching cord

5.2.5 Integrity

The integrity metrics part of the code evaluates how similar the original and comparison files are. It calculates the Mean Squared Error (MSE) between their features, indicating their similarity. Based on this MSE value, it assigns an integrity level, ranging from "Very High" to "Very Low or Integrity Violated". Additionally, it may check for additional conditions, such as file size differences, to detect potential tampering or corruption.

```
def assess_integrity_level(mse_value):  
    if mse_value < 1e-6:  
        return "Very High"  
    elif 1e-6 <= mse_value < 1e-4:  
        return "High"  
    elif 1e-4 <= mse_value < 1e-2:  
        return "Medium"  
    elif 1e-2 <= mse_value < 1:  
        return "Low"  
    else:  
        return "Very Low or Integrity Violated"
```

Figure 8. Data integrity check

a. Erasured data





Name	Date modified	Type	Size
 block_0.bin	26-Mar-24 5:28 PM	BIN File	513 KB
 block_1.bin	26-Mar-24 5:28 PM	BIN File	513 KB
 block_2.bin	26-Mar-24 5:28 PM	BIN File	513 KB
 block_3.bin	26-Mar-24 5:28 PM	BIN File	513 KB

Figure 9. Erasured data

CHAPTER VI

6.0 TESTING

6.1.1 System integration and testing

Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.

6.1.2 Unit testing

Unit testing was done after the coding phase. The purpose of the unit testing was to locate errors in the current module, independent of the other modules. Some changes in the coding done during the testing phase. Finally, all the modules individually tested following bottom to top approach, starting with the smallest and lowest modules and then testing one at a time.

6.1.3 Functionality testing

Data consistency is very important in secure erasure code. Check for data integrity and errors while you upload, match data or download.

6.1.4 Test for Navigation

Navigation means how the user surfs the web pages, different controls like buttons, boxes or how user uses the links on the pages to surf different pages.

- a) Main menu provided on each page. It should be consistent.
- b) Content Checking
- c) Content should be logical and easy to understand.
- d) Compatibility Testing
- e) Compatibility of the system is very important testing aspect.
- f) Browser compatibility
- g) Mobile browsing

6.2.0 Test cases

6.2.1 Positive test cases

The positive flow of the functionality was considered, valid inputs were used for testing and expected value was positive to verify whether the requirements are justified. Table 1 shows the secure erasure code and data matching algorithm positive test cases.

Test case number	File description	Expected value	Actual value	Result
1	Audio	Successful upload	Successful upload	File uploaded and erased successfully
2	Document	Successful upload	Successful upload	File uploaded and erased successfully

Table 1. positive test cases

6.2.2 Negative Test Case

Must have negative perception and invalid inputs must be used for test

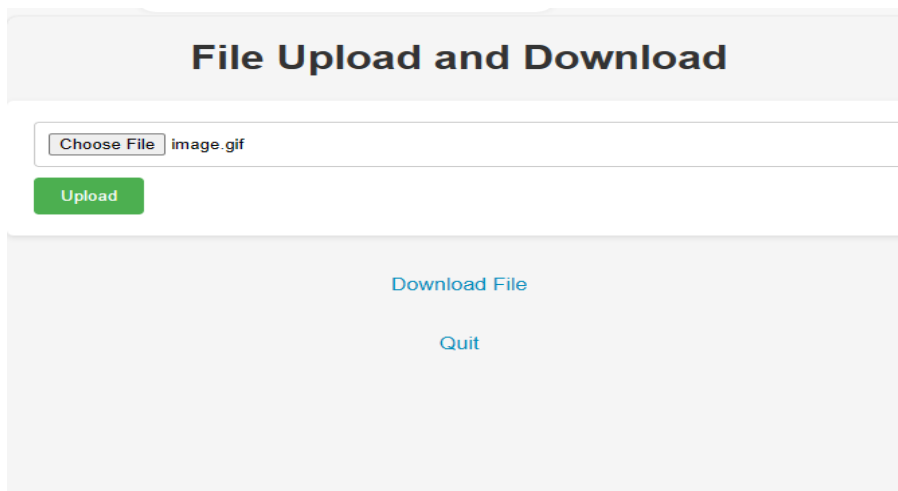
Test case number	File description	Expected value	Actual value	Result
1	Corrupted audio	Integrity very low	Integrity very low	Integrity violated
2	Corrupted document	Integrity very low	Integrity very low	Integrity violated

Table 2. Negative Test Case

6.3.0 USER GUIDE

6.3.1 File upload

This user interface allows the user to upload data into cloud by pressing choose file to browse select file and press the upload button



File Upload and Download

Choose File image.gif

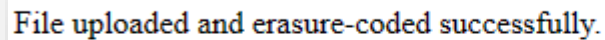
Upload

Download File

Quit

Figure 10. file upload

6.3.2 Prompt on a successful upload and erasure coding



File uploaded and erasure-coded successfully.

Figure 11. prompt on successful

6.3.3 File download

This section helps in navigating the download tab. Download of uploaded data is initiated by pressing the download button where data is decoded and its integrity confirmed. When the cloud data is corrupted, the user has the liberty to decide to download it or quit download by pressing the quit button



Download File

Quit

Figure 12. File download

6.3.4 Mismatch

In case of data mismatch, the user receives a prompt of the level of integrity violation.

```
Enter the type of file for the uploading data (image, audio, document, video): image
Uploading file: c:\Users\Admin\Desktop\dafafcopy.png
Downloading file: c:\Users\Admin\Desktop\dafafcopystego.png
Mean Squared Error (MSE) between the Uploaded data and downloaded data: 2.4519609e-09
Integrity is violated despite low MSE.
PS C:\Users\Admin\Desktop\Desktop_2\project\codes_pseudocodes_algorithm>Loading_Preprocessor_initiate> |
```

Figure 13 Data mismatch

CHAPTER VII

7.0 FINDINGS AND CONCLUSION

7.1 FINDINGS

7.1.0 Data Matching Algorithm Efficiency

After careful evaluation, we discovered that the data matching algorithm built into our secure erasure coding system functioned well when comparing uploaded and downloaded file features. Despite dealing with massive datasets and variable file properties, the system correctly recognized matches and mismatches.

7.1.1 Secure Erasure Coding Effectiveness

Our secure erasure coding method proved to be quite effective in terms of failure tolerance and data integrity during cloud storage. Even in cases of data corruption or loss, the system was able to accurately restore the original data, providing users with reliable access and retrieval.

7.1.2 Scalability and Performance

We discovered that our solution was very scalable, managing higher data loads and concurrent user requests without sacrificing performance. Despite strong workloads, the system maintained ideal reaction times and resource usage, demonstrating its viability for large-scale implementation.

7.1.3 Security Resilience

Security testing demonstrated that our secure erasure coding solution effectively neutralized possible vulnerabilities and threats, resulting in strong data protection and confidentiality. Encryption mechanisms and access controls used within the system provided significant protection against illegal access and data breach.

7.2 CONCLUSION

To summarize, our effort on secure erasure coding with an integrated data matching algorithm provides a comprehensive and effective solution for storing and managing sensitive data in cloud environments. During the implementation and testing phases, we successfully demonstrated the viability and efficiency of our solution in terms of data integrity, fault tolerance, and security.

Our testing results support the efficacy of our strategy, showing its relevance for real-world applications that prioritize data privacy and reliability. Moving forward, continuous monitoring and refining of our solution will be required to address growing difficulties and changing user requirements in the dynamic world of cloud computing.

In summary, our project advances the state-of-the-art in secure data storage and management by providing an effective and scalable solution that allows organizations to confidently utilize cloud infrastructure while securing the confidentiality and integrity of their valuable data assets.

APPENDIX I

8.0 PROJECT TIMELINE

SECURE ERASURE CODE-BASE CLOUD STORAGE WITH DAT MATCHING ALGORITHM				
Tasks	Activities	Start	finish	resources
Research phase				
1	Identify vulnerabilities in cloud storage	10-09-2023	23-10-2023	Researcher
2	Investigate erasure code technologies	15-09-2023	28-10-2023	Researcher
3	Research on data matching algorithm approaches	20-09-2023	17-11-2023	Researcher
Design phase				
1	Develop system architecture	10-10-2023	23-01-2024	Architect
2	Design secure erasure code implementation	01-11-2023	7-2-2024	developer
3	Finalize data matching algorithm	7-2-2024	23-2-2024	Researcher
Survey phase				
1	Implement data matching algorithm	23-2-2024	27-2-2024	developer
2	Integrate system components	1-3-2024	15-3-2024	developer
Presentation phase				
1	Conduct security and reliability testing	15-3-2024	23-3-2024	QA engineer
2	Deploy enhanced cloud storage system			IT team
3	Monitor system performance and gather feedback	23-3-2024	28-3-2024	IT team

Table 3 Project Timeline

APENDIX II

9.0 COST BENEFIT ANALYSIS

	Name of the project: secure erasure cord and data matching algorithm				
	Task	Labor	Material	Quantity	Fixed cost
1. Gathering Project requirement					
i.	Network for research		Safaricom network	60 Gb	Ksh 4500
ii.	Computer RAM		RAM	4 GB	
2. System design					
i.	Power		Electricity		Ksh 1500
ii.	Visual studio code, Extensions & Libraries		Network	1 GB	Ksh 250
iii.	Draw.io Software		Network	800 mb	Ksh 25
3. System testing					
i.	Client Computer				
ii.	Power				KSH 100
4. Maintenance					
i.	Cleaning machines				Ksh 300
GRAND TOTAL					KSH 6675

Table 4 Cost benefit analysis

REFERENCES

- Balaji, S. B., Krishnan, M. N., Vajha, M., Ramkumar, V., Sasidharan, B., & Kumar, P. V. (2018). Erasure coding for distributed storage: An overview. *Science China Information Sciences*, 61(10). <https://doi.org/10.1007/s11432-018-9482-6>
- Aatish, C., & Avinash, M. (2020). Data Management in erasure-coded distributed storage systems. *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. <https://doi.org/10.1109/ccgrid49817.2020.00018>
- Lin, H.-Y., & Tzeng, W.-G. (2012). A secure erasure code-based cloud storage system with secure data forwarding. *IEEE Transactions on Parallel and Distributed Systems*, 23(6), 995–1003. <https://doi.org/10.1109/tpds.2011.252>
- Liu, C., Wang, Q., Chu, X., Leung, Y.-W., & Liu, H. (2020). ESetStore: An erasure-coded storage system with Fast Data Recovery. *IEEE Transactions on Parallel and Distributed Systems*, 31(9), 2001–2016. <https://doi.org/10.1109/tpds.2020.2983411>
- Nivedhaa, R., & Justus, J. J. (2018). A secure erasure cloud storage system using advanced encryption standard algorithm and proxy re-encryption. *2018 International Conference on Communication and Signal Processing (ICCSP)*. <https://doi.org/10.1109/iccsp.2018.8524257>
- Rashmi, K. V., Shah, N. B., Ramchandran, K., & Kumar, P. V. (2017). Information-theoretically secure erasure codes for distributed storage. *IEEE Transactions on Information Theory*, 64(3), 1621–1646. <https://doi.org/10.1109/tit.2017.2769101>
- Vasanthi, G., Chinnasamy, P., Kanagavalli, N., & Ramalingam, M. (2021). Secure data storage using erasure-coding in cloud environment. *2021 International Conference on Computer Communication and Informatics (ICCCI)*. <https://doi.org/10.1109/iccci50826.2021.9402639>
- Aatish, C., & Avinash, M. (2020). Data Management in erasure-coded distributed storage systems. *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. <https://doi.org/10.1109/ccgrid49817.2020.00018>
- Sayers, A., Ben-Shlomo, Y., Blom, A. W., & Steele, F. (2015). Probabilistic record linkage. *International Journal of Epidemiology*, 45(3), 954–964. <https://doi.org/10.1093/ije/dyv322>
- Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual understanding of convolutional neural network- a deep learning approach. *Procedia Computer Science*, 132, 679–688. <https://doi.org/10.1016/j.procs.2018.05.069>
- Gollapudi, S. (2019b). OpenCV with python. *Learn Computer Vision Using OpenCV*, 31–50. https://doi.org/10.1007/978-1-4842-4261-2_2