National College of Ireland

Blockchain-Based Framework for Secure Cloud Storage with Data Integrity Verification

MSc Research Project
MSc Cloud Computing

Harshit Bhalla
Student ID: x23208813

School of Computing
National College of Ireland

Supervisor: Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Harshit Bhalla
Student ID:	x23208813
Programme:	MSc Cloud Computing
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Ahmed Makki
Submission Due Date:	12/12/2024
Project Title:	Blockchain-Based Framework for Secure Cloud Storage with Data Integrity Verification
Word Count:	6851
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Blockchain-Based Framework for Secure Cloud Storage with Data Integrity Verification

Harshit Bhalla
x23208813

Abstract

Today, Data is considered as an invaluable asset. Cloud computing have drastically transformed management of data by providing cost efficient and scalable data storage solutions, however their centralised architecture poses considerable challenges such as unauthorised access and data breaches and attacks to data integrity, hence it is essential to protect sensitive data. These challenges are addressed by combining decentralised technology blockchain and cryptographic techniques to develop a secure platform for storing sensitive data to cloud. The proposed system employs Advanced Encryption Standard (AES-256) to encrypt user data ensuring confidentiality on cloud storage and to preserve integrity of data,achieving a 100% detection accuracy in integrity verification tests. Merkle tree-based approach is implemented, allowing users to verify authenticity of the data and detect tempering of data. The system also supports periodic integrity verification feature that allows user to enable automated integrity check for their data. This approach offers real time alerts in the event of file integrity violation. The platform is designed as a web application, providing users with interface to securely store, manage and verify integrity of data over cloud. The result of study shows that blockchain technology can enhance the security and ensure integrity of the data over cloud storage system.

1 Introduction

In this era of information, data is considered as one of the most valuable assets and as the dependency on data is increasing, so do the challenges related to ensuring its security and integrity. Cyber-attacks and unauthorised access to sensitive data poses significant risks that can potentially weaken the trust users have in the systems managing their data. Amongst the principle of data security CIA (Confidentiality, Integrity, and Availability) data integrity is considered as critical concern as it's violation can damage trust and decision-making processes.

1.1 Research Motivation:

Traditional cloud computing services have brought a paradigm shift in the way we manage data, it is considerably more secure, reliable and efficient. Whether for a small-scale business or large corporations, there is a significant shift to cloud storage solutions for addressing the growing demands driven by lower cost, auto scaling and high availability. However, the centralized architecture of traditional cloud storage service introduces

several challenges.Data stored over a centralised servers is susceptible to unauthorized access and data breaches (Barona and Anita, 2017). A Research on cloud computing vulnerably shows data breaches in centralized cloud environment not only threaten the data confidentiality but can also cause violate of integrity resulting in user's data manipulation and corrupted data.(Feng and Li, 2022),Such data breach can violate users trust over data, additionally user faces lack of transparency regarding how their data is stored and managed, compelling them to solely rely on service provider's integrity. One of the primary concerns of traditional cloud storage system is security of sensitive data. Valuable and sensitive data are often stored by organizations on cloud servers that is essential for their daily operations, making it potential target for cyber attacks. Research shows that data breaches and unauthorized access to cloud data continue to be significant issues in cloud storage systems, many a times resulting in financial losses, reputation damage, and reduces trust towards service providers (Sharma et al., 2019).

Neto et al. (2020) examines the Capital One data breach, where personal data of around 106 million customers was compromised. Attackers exploited vulnerabilities in the AWS cloud infrastructure, enabling a Server Side Request Forgery (SSRF) attack. Hence there is a need of a system that provides method to ensure integrity of user uploaded data while ensuring confidentiality of data.

1.2 Project Specification

The web application enables users to securely store data over third party cloud storage service, ensuring that the stored data is only accessible to actual uploader. Before accessing the application, user must authenticate themselves using MetaMask wallet. By implementing blockchain technology, application applies an extra layer of security by storing critical metadata including file hashes and Merkle Roots on the ethereum blockchain to prevent tampering of data. Simultaneously, it splits the data into chunks and encrypts each chunk using AES (Advanced Encryption Standard) encryption technique and stores encrypted chunks over cloud storage like Amazon web services Simple Storage Service (S3) , at any time user can perform integrity check to verify their data is not tempered. Additionally, periodic automatic integrity are performed using smart contract and Lambda function, user is informed immediately if any discrepancy in data is detected.

1.3 Research questions:

- How effectively the integration of blockchain and encryption and Merkle Tree structure identify data integrity violations in cloud storage ?
- What are the performance metrics for encryption and decryption using Cryptography technique (AES-256) when used on fragmented file stored in the cloud?
- How does periodical verification of integrity improve the identification of tampered data in cloud storage environment?
- What are the trade-offs in performance when using Ethereum smart contract for metadata storage and data integrity verification regarding cost and scalability?

1.4 Research objectives:

- To create a secure framework for cloud data storage by integrating blockchain technology along with cryptographic methods such as AES-256 encryption and Secure Hash Algorithm 256 (SHA-256) hashing.
- To maintain data integrity using Merkle Trees and the immutable storage of Merkle Roots over a blockchain network for ensuring immutable data storage.
- To address data privacy issues of cloud storage solutions by implementing encryption and offering decentralized authentication methods.
- To provide integrity verification methods with both manual and automatic periodic integrity testing with a real-time notification system in case of data tampering.

1.5 Research Contribution:

This Research aims to enhance the data security and ensure data integrity in centralised cloud storage solutions

- **Integrity Verification:** The study focuses on ensuring the data integrity of cloud storage data by leveraging blockchain technology to immutably store file metadata over Ethereum smart contract. Merkle roots is calculated using cryptographic hashing of data fragments, gives an efficient method to validate large dataset. Any tempering to the data can be detected by recalculating the Merkle root, ensuring that data remains authentic and unaltered.
- **Automated Integrity tests:** The system provides automated periodic integrity test by using smart contracts and AWS Lambda functions. This ensures continuous integrity test are performed on cloud stored data and if any inconsistencies or tempering or data is detected, users receive real-time notification via registered Email with the help of AWS Simple notification service (SNS).
- **Enhanced Security:** By utilising AES-256 (Advanced Encryption Standard) encryption along with data fragmentation, the system protects the data from any unauthorised access, ensuring confidentiality of data stored in cloud storage server while providing efficient way to verify integrity of data.

2 Related Work

2.1 Ensuring Data Integrity in Cloud Computing

To address the data integrity threats in cloud computing settings (Sharma et al., 2019) proposes a blockchain based solution that uses decentralised and immutable property of blockchain technology. Their solution focus on employing a combination of blockchain and Merkle tree to verify integrity of data while ensuring transparency and authentication. Smart contract within blockchain ecosystem is used for verification process. The proposed system makes use of P2P (point to point) storage model that is distributed amongst multiple nodes to mitigate the risk associated with a centralised architecture storage like single point of failure. This immutable ledger eliminated the risk of tampering, providing

a reliable source for storing Merkle root of data. This process makes sure that only verified user can interact with the system and data remains intact. While the system proposes an innovative approach to verify integrity, the author mentioned some limitation including scalability issue in due to blockchain's high computational power requirements also the system mainly focus on integrity this study entends this research to ensure integrity along with condifentiality make a more secure hybrid model for integrity verification over cloud envionment.

(Liu et al., 2023) introduces a data integrity auditing system that makes use of Quad Merkle Tree (QMT) structure. It extents the concept of Merkle Tree by dividing the data in to four branches per node. System is interconnected with blockchain technology for immutable storage of QMT root. This scheme addresses the issues seen in traditional cloud storage by offering decentralized and temper resistant way for data verification, with use of QMT, the system offers improved data integrity checks but use of QMT has high effect on computational cost specially in case of large datasets.

(Goswami et al., 2024) Suggests an integrity verification scheme specially build for cloud storage IOT data. It makes use of Merkle Hash Trees (MHT) and techniques like Proofs of Retrievability (PoR) and Provable Data Possession (PDP), the system mainly focus on ensuring data integrity and enable efficient public auditing. Furthermore, the capabilities of blockchain have been examined for ensuring immutability and transparency of data. The study expands upon these developments by using blockchain authentication, AES-256 encryption, and Merkle Tree hash verification, providing scalable approach for data integrity verification in cloud storage settings.

(Sari and Sipos, 2019) introduced a blockchain based system for handling Secure File Sharing over decentralised cloud storage that uses Interplanetary File System (IPFS) over Etherum network The paper suggests a dynamic technique to ensure user file integrity by using a group consensus mechanism and secure data over P2P(Peer to Peer) network, offers decentralisation access control and authentication to address drawbacks of centralized cloud storage.

2.2 Blockchain and Cryptography for Secure Cloud Storage

In order to address the security challenges of cloud data storage, (Idrus et al., 2023) present a blockchain based framework that aims to improve the privacy and ownership of data stored in cloud storage. The proposed framework tackles with vulnerabilities seen in conventional cloud storage solutions like centralised control and vulnerability to cyber-attacks. By using blockchain immutable ledger, the authors imply cryptographic methods, AES (Advanced Encryption Standard), for data encryption and SHA 256 (Secure Hash Algorithm 256) for performing hashing. To ensure confidentiality of data, data is transformed into encrypted blocks and stored permanently on blockchain, with each block connected to its successor by utilising cryptographic hashes and hence ensuring transparency and tamperproof technique to securely store data. A significant aspect of this approach is implementation of Proof of Authority (PoA) consensus mechanism.In order to validate a block, confirmation from all the validators is required, in comparison with other consensus mechanism like Proof of work (PoW), PoA provides faster block confirmation and offers higher throughput but sacrifices on decentralization.

(Shakor et al., 2024) suggests a solution for improving cloud data security by integrating dynamic AES encryption and blockchain based key management. The system modifies encryption key in real time based on the sensitivity of data and the policies defined by the user. Blockchain technology is used to securely store immutable encryption keys, blockchain smart contracts are used to manage encryption keys. The proposed system handles risk of unauthorized access and maintains data confidentiality, performance is measured in terms of encryption and decryption speed and systems capability to secure data against any attack or unauthorized access.

2.3 Authentication With Blockchain

(Jha et al., 2023) proposed a system called Certifier Dapp that uses decentralized approach to managing certificates with the use of blockchain. The system makes use of MetaMask wallet for providing secure authentication system where user can connect with their meta mask wallet and application can retrieves the wallet address to identify the user. Smart contract ensures issuing new certificates to registered users and verify the authenticity of existing certificates. Each transaction is stored over smart contract ensuring high transparency of transactions and security.

(Almadani and Hussain, 2023) introduces a secure blockchain based wallet system with MFA (multi factor authentication), the system uses advanced authentication settings, including passwords,TOTP(time based one time passwords), and facial recognition. This paper mainly focus on how secure authentication can help in reducing the risk of phishing attacks and unauthorised access, the system is implemented using Ethereum blockchain and python. Author conducted bunch of experiments to validate its security against attack simulations, this system demonstrates a highly secure authentication mechanism.

(Castro-Medina et al., 2019) reviews different ways of performing data fragmentation and replication of data in a cloud environment, highlights their role in improving the overall performance and security, paper discus on various techniques implemented over the past eight years giving insights on their use case and suitability.

2.4 Research Gap Analysis

The existing work on secure cloud storage mainly focus on either confidentiality or integrity, (Sharma et al., 2019) research mainly focus on integrity protection, (Idrus et al., 2023) and Shakor et al. (2024) focus on ensuring confidentiality neglecting solution that focus on both, although both AES encryption and SHA-256 based hashing are commonly used but their interaction with blockchain for periodically verifying integrity and automated of verification is not explored widely, limited attention is given to challenges like real time breach notification and hybrid solutions that verifies integrity while ensuring data remains confidential over cloud environment even in situations like data breach, furthermore existing solutions like Quad Merkle Tree (Liu et al., 2023) and Proofs of Retrievability Goswami et al. (2024) are processor-intensive, making them less efficient for large-scale cloud applications.

This research aims to address these gaps by creating a hybrid solution that allows a organisation or a user to verify integrity while ensuring confidentiality of data by using AES-256 encryption and blockchain storage. This research addresses the existing limitations literature aims to develop a secure, scalable and efficient system to upload data on

cloud storage while ensuring both confidentiality and integrity of data.

Reference	Focus	Findings
(Sharma et al., 2019)	Blockchain-based cloud integrity	Uses merkle tree approach to ensure data integrity, daily Focused on P2P.
(Liu et al., 2023)	Data integrity (QMT)	Suggest QMT approach to improve integrity of data, shows High computational costs for large datasets.
(Goswami et al., 2024)	IoT data integrity	Integrated AES-256 encryption and Merkle Tree for data security and integrity.
(Sari and Sipos, 2019)	Decentralized file sharing	Offer group consensus and secure file storage in a P2P network, Addressed centralization drawbacks of traditional cloud storage.
(Idrus et al., 2023)	Offers blockchain based cloud security	Combined AES encryption and SHA hashing with blockchain for secure data storage.
(Shakor et al., 2024)	Dynamic AES encryption	Real-time dynamic AES encryption modifies keys based on sensitivity and policies.
(Jha et al., 2023)	Decentralized certification	Smart contracts for transparent certificate issuance; MetaMask integration for secure user authentication
(Almadani and Hussain, 2023)	Blockchain wallet security	Offers integrated ways of encryption (TOTP and password authentication)
(Castro-Medina et al., 2019)	Data fragmentation/replication	Highlights the role of fragmentation and replication in improving performance

Table 1: References in the Order Discussed

3 Methodology

The proposed system is designed to implement a web application that ensure file integrity while securing the data stored on cloud storage solution by leveraging blockchain technology and cryptographic techniques such as encryption. The main goal is to provide a tamper proof and efficient method by which users can verify integrity of their data while ensuring security. By combining immutable blockchain storage, cryptographic techniques and cloud storage the research implementation aims to overcome the drawbacks of centralized storage systems like data breaches and lack of transparency inefficient data integrity verification methods.

3.1 User Interface and Web Application

The web application is built on Angular frontend framework, it allows user to register and login securely with blockchain based wallet integration where user authentication is done through smart contracts deployed over Ethereum (ETH) blockchain. Once user is authenticated they can upload a file, the application processes the file locally, where it performs data fragmentation. The divided segments are then encrypted, these encrypted chunks are then uploaded to cloud storage and hash for each file chunks is calculated using and these hashes are then combined to form Merkle tree which represents integrity of the entire file. This root is further immutably stored over smart contract along with file metadata. This web application ensures both confidentiality of the file content and allows user to test integrity of the file. The application allows both manual an automated testing of file integrity, where user have the option to enable automated verification of integrity. Furthermore, system allows to enable periodic hash verification where file is periodically tested for verifying integrity.

3.2 Data Encryption and Integrity Verification

To ensure confidentiality, system utilise cryptographic techniques during file processing, when a user uploads a file, it's divides it into smaller chunks by performing data fragmentation, and each data chunk is further encrypted using AES-256 (Advanced Encryption Standard) encryption. This ensures even if data is compromised its content remains hidden. AES-256 is a commonly used symmetric encryption method that uses a 256-bit key length. It is considered as highly secure and helps in protecting sensitive data (Akhil et al., 2017), because of its large key size it offers efficient encryption mechanism and offers key lengths varying from 128 to 256 bits, it supports a block-length of 128 bits and key size of 128 bits,192 bits and 256 bits. AES-256 provides a significant level of confidentiality and integrity. (Idrus et al., 2023) In application when user uploads a file, they are prompted to sign a cryptographic message using their blockchain wallet, further this signature is used to derive Encryption key using a Key Derivation Function (KDF) that combines the user signed message and wallet address. This approach avoids storage of sensitive data such as the salt or encryption key on the blockchain ensuring a secure encryption process. Then the derived key is used for AES-256 encryption of file chunks, making sure that only authenticated user with access to the wallet can decrypt the file.

Figure 1 illustrate the process of data fragmentation, encryption, storage and decryption. It shows how user data is fragmented into chunks, encrypted with AES-256, and stored securely over cloud, and for decryption process user must have same wallet signature to derive access key which they used for encrypting the file order to decrypt the file.

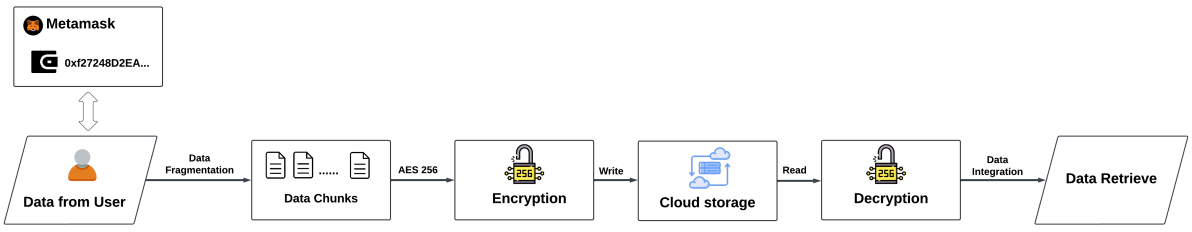

Figure 1: Cryptographic Process

Breaking down large files into smaller chunks allows the system to handle them in smaller parts, making processing and storage effective and feasible. (Liu et al., 2023) For Integrity verification each divided chunk is hashed using SHA-256 algorithm, SHA 256 (Secure Hash Algorithm 256) is a widely used cryptographic hash function designed for data security and authenticity of data. It provides high degree of collision resistance making sure that different inputs return unique hash values (Devi and Jayasri, 2023). The hash values generated for each chunk are then used to create a Merkle Tree. A Merkle Tree is a structure that is very commonly used in blockchain and cryptography applications to verify data integrity, each leaf node represents hash of a chunk and the node above it are generated by hashing the combination of its children node. This multi layered configuration produces a single hash node at the top of tree also called as root node, the Merkle tree computes Merkle root which acts as a single point integrity verifier, a point representing integrity of the entire file (Liu et al., 2021). Even if a slight change made to any of the chunk will result in new Merkle root, allowing system to detect any tempering of data, this calculated root is stored immutably on blockchain smart contract along with file meta data including file name, timestamp of uploaded file and the cloud stored chunk URLs.

3.3 Blockchain Integration

Integration of blockchain provides a decentralized framework for storing immutable metadata which is used for verifying file integrity, In this system Ethereum blockchain platform is used. ETH blockchain Smart contract are automatically executing contacts with its terms and conditions directly written in the code whenever these conditions are met. It executes the intended code, allowing secure management of agreements without anyone's involvement. Smart contract immutably stores the computed Merkle Root, and the URLs of encrypted file chunks stored in cloud storage. Once this data is recorded on smart contract it cannot be changed or deleted Any change to the original file stored over cloud services will result in a completely new merkle root when compared to the one stored over blockchain, allowing the system to detect any tampering of data.

This smart contract is written in Solidity, deployed over ETH Sapolia test network which automated key functionalities in the system, Smart contract is designed to offer the following functionalities in the system:

- **User Registration:** Initially when user visit the application for the first time, they are required to register in order to use the application, A user can register with their wallet by providing their name and Email address, once registered a user profile is created over smart contract, linked to their wallet address
- **Uploaded File metadata:** Whenever an authenticated user upload or modify a file, it's computed merkle root along with upload time stamp, file name, URLs of encrypted file chunks are stored over smart contract, in future if user want to update a file, smart contract ensures metadata is updated for previously uploaded files.
- **Integrity Verification:** The smart contract facilitates functionality to verify integrity by providing method to retrieve stored Merkle Root for comparison with a newly computed root. This ensures that any unauthorized changes made to the file or its chunks can be effectively detected.

- **Periodic Hash Verification:** The contract provides functionality to manage periodic hash verification preferences. A user can opt in and opt out if they no longer want their encrypted files to be monitored for integrity verification.

3.4 Cloud Storage and Chunk Management

For storing the Encrypted chunks of data over cloud, the system uses Amazon S3 service. I selected Amazon s3 because it's highly reliability and scalable and manages data in a manner that provides durability, high availability, minimal latency, and enhanced durability (M et al., 2022). File uploaded by users are divided into smaller chunks for computing Merkle root and allows parallel processing of data. Once each chunk is encrypted and uploaded to s3, S3 returns the uploaded chunks URLs, which are further stored over smart contract to facilitate integrity verification.

Whenever user wants to test integrity of their file or want to access the file, these encrypted chunks are retrieved from cloud and decrypted using the users unique key and are reassembled to create the original file uploaded by the user.

3.5 Periodic Integrity Verification:

Periodic integrity verification provides automated monitoring of file integrity this is implemented using blockchain and some cloud technologies, user gets an option to enable or disable periodic integrity checks on file. Their preferences are stored over smart contract, at regular intervals a Lambda function calculates recalculates their Merkle root using the SHA-256 hash algorithm on uploaded chunks and compare them with smart contract stored metadata, basically it performs integrity verification similar to the web application but it automates the process, enabling user to continuously assess their file for any potential damage or tampering without requiring manual intervention to the system.

4 Design Specification

The system Architecture of the suggested system will be discussed in this section, the proposed system is designed to manage secure file storage and retrieval while providing measures to verify data integrity by using a combination of decentralized authentication, data encryption and using blockchain smart contracts

4.1 Architectural Overview

Figure 2 illustrates the Architecture model of application, The Architecture is built over three main components: **Web Application**, **Blockchain (Smart Contract)**, and **Cloud Storage (AWS S3)**.

1. **Web Application** Is built on Angular and provides User interface to interact, it allows the user to register, login, securely upload and mange user uploaded files, It offers the following functionalities:

    - **User Authentication:** The application allows the user to register and login themselves using their meta mask wallet where their wallet address act as a Unique identifier for authentication.

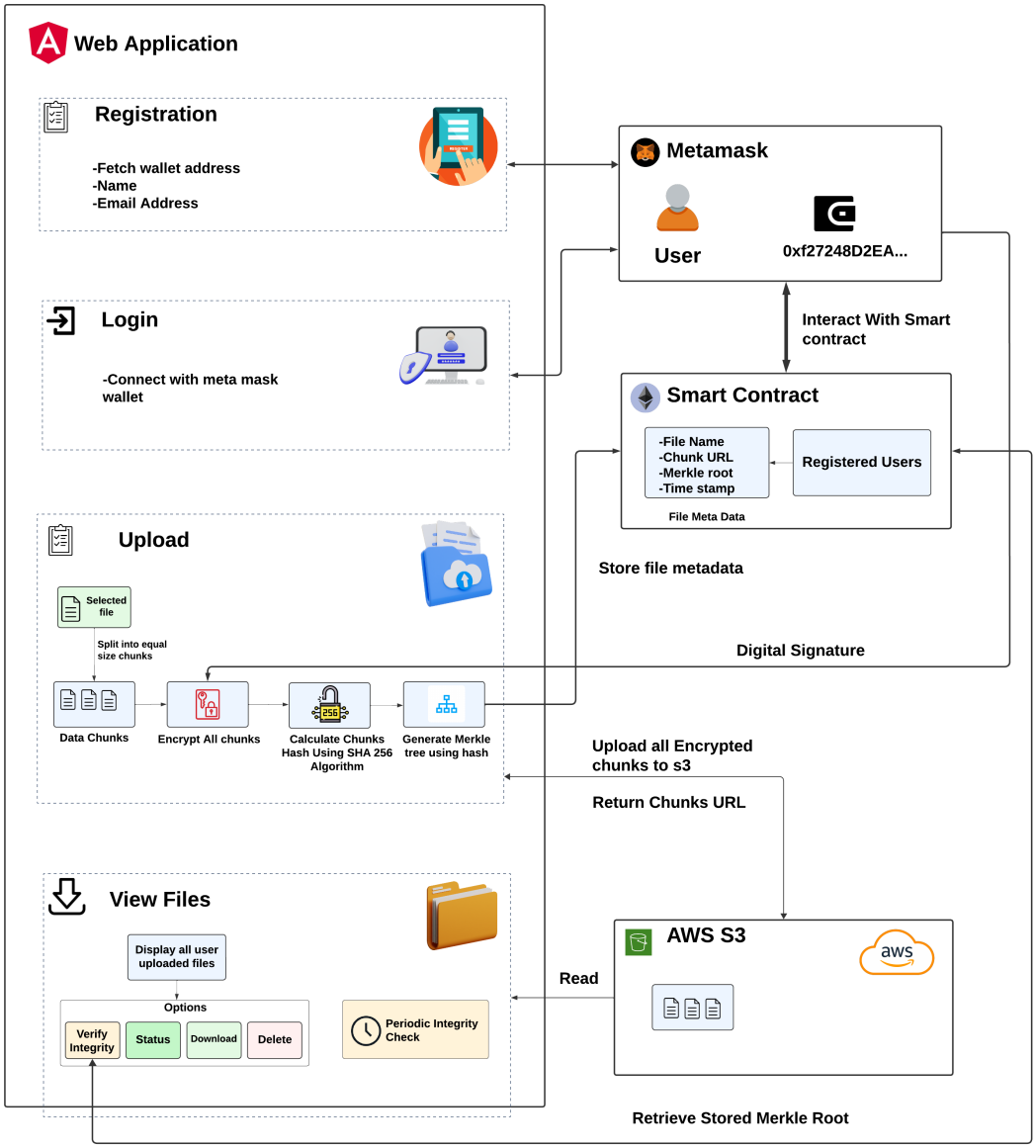

Figure 2: System Architecture

- **File Upload/Update:** Once user selects a file, the application divides it into equal-sized chunks, encrypt them using AES 256 encryption by deriving a key using KDF (key derivation function) and generates Merkle tree structure using chunk SHA -256 hashes, the encrypted chunks are then uploaded to AWS S3, and the metadata is stored over smart contract.
- **File Download:** User gets an option to download the uploaded file for which encrypted chunks are fetched from AWS S3 and the encryption key is regenerated using the same process by taking wallet signature ensuring that only the rightful owner of the file can access it.
- **Verify Integrity:** The application gives an option for all user uploaded files to test the integrity of file that recalculated Merkle root the cloud uploaded chunks hash and compared it with the stored Merkle root, if any tempering of data exists then these values can never be the same, also system allows user to enable periodic hash function which alerts the user about any potential

tempering of data using notification service.

2. **Blockchain (Smart Contract)** provides decentralized storage for maintaining immutable records, it provides a medium to storing and managing file metadata, by using blockchain technology system ensures immutable and temper proof storage, it provides the following functionalities:

   - **Secure authentication:** Stores user details like user name, email address and wallet address and provides functionality to check if a user is registered or not.
   - **Store File Metadata:** Stores the Metadata like the file name, Merkle root, cloud stored chunks URLs and file uploaded time stamp are stored.
   - **Integrity Verification:** Provides functionality to fetch the stored data on smart contract by providing metadata details of a file to the authenticated user.
   - **Periodic Integrity Verification Status:** Tracks whether a user has enabled periodic integrity checks to automate the testing process.

3. **Cloud Storage:** AWS S3 is used for storing the user uploaded file, It stored the file fragmented encrypted chunks, it offers two basic functionalities :

   - **File Chunk Storage:** Encrypted file chunks are stored securely over S3 bucket, with each chunk uniquely named and accessible through its URL.
   - **File Retrieval:** Provides encrypted file chunks to the Web Application whenever user request for file downloads or tries to perform integrity checks.

4.2 System Workflow :

The sequence diagram shown in Figure 2 explains the flow of interactions between the components for running the core functionalities of the system like user registration, authentication, file uploads, file download and file integrity verification.

4.3 Security and Design Justification

- **Encryption:** File chunks generated from user uploaded file are encrypted with AES 256(Advanced Encryption Standard) before being uploaded to public cloud. so, to ensure data remains confidential even if the file chunks are compromised in a data breach. By using a user-specific key for encryption the system ensures that only the authenticated user can access the files.
- **Data Integrity:** To ensure the integrity of uploaded files, application uses Merkle tree-based verification that uses blockchain immutability to ensure that any unauthorized modification or tampering with the file chunks is immediately detectable.
- **Decentralized Metadata Storage:** Storing file metadata including file details and Merkle root on the blockchain ensures data cannot be tempered with and provides immutable storage.
- **Wallet-Based Authentication:** Using blockchain wallets such as MetaMask for authentication make sure that only the authorised owner can access the application.

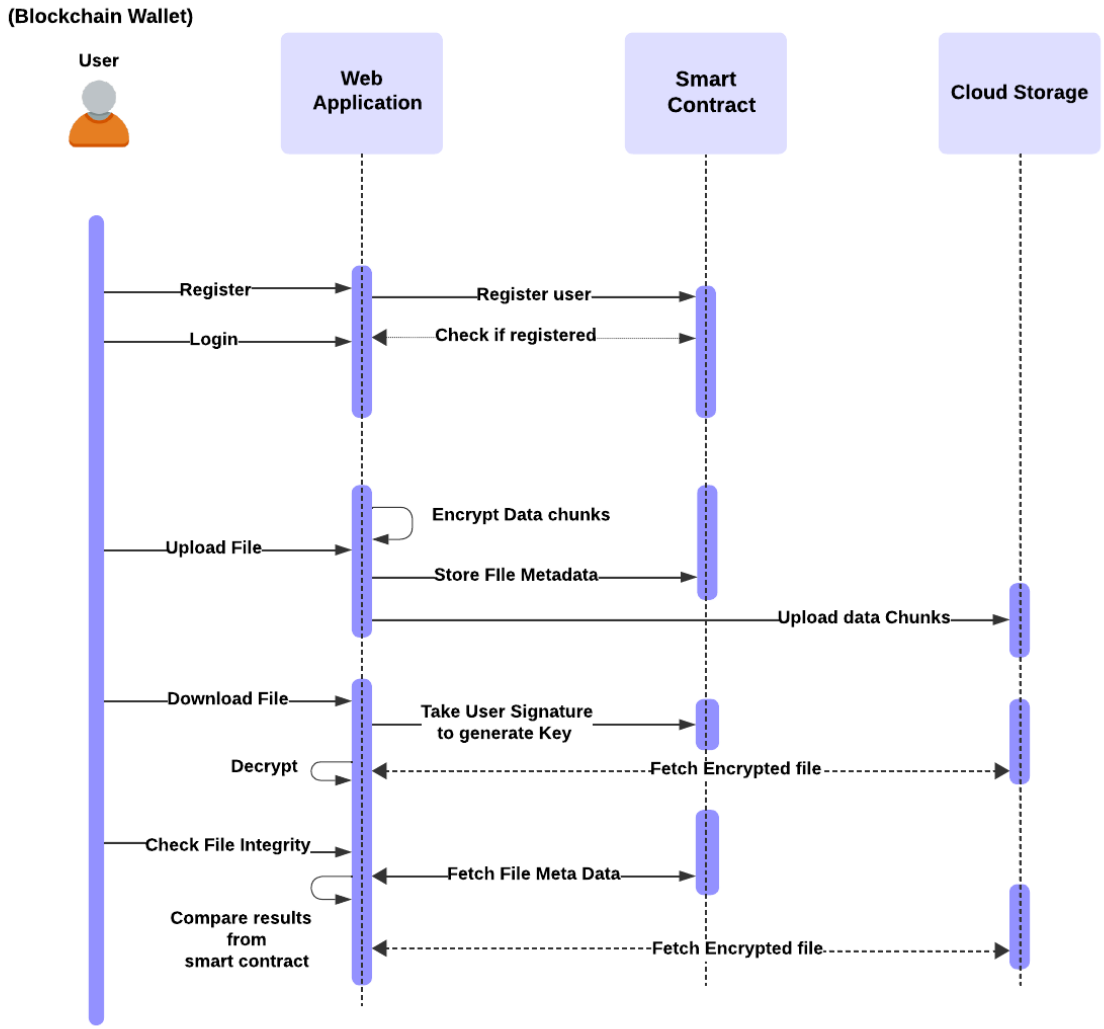

Figure 3: Application sequence diagram

5 Implementation

The system was developed using Typescript and the web application was built on Angular Framework, The webapp frontend user interface was built using HTML, SCSS and bootstrap. The implementation of the project includes development of a Dapp(Decentralized application) that integrates blockchain and cryptography to ensure secure data storage management over cloud and ensure integrity of data.

5.1 Creation of smart contract:

Smart contract is a core component of the application, implemented in solidity and deployed over ETH test network called Sapolia test network, Once the smart contract (blocksecure.sol) is deployed it will be able to execute its functionality without the need for any manual intervention, Remix IDE was used for creation and deployment of smart contract. The table 2 shows the transaction information of deployed contract.

The smart contract is responsible to provide immutable storage and some key func-

Field	Details
Transaction Hash	`0xdf37330c002691e63b0e679695b7f0064649d66523d08e92133e6b697481be4b`
Status	Success
Block Height	7122737
Contract Address	`0x19967531119842F341D2AE1D96208908EC914BDE`

Table 2: Smart Contract deployed on Sepolia Testnet

tionalities for application including user registration, storage of file meta data and data retrieval, some of it's key functionality are discussed in 3

Function Name	Description
`registerUser()`	Register the user by giving their name and email,Ensures each wallet address can only register once.
`isUserRegistered()`	Checks if a wallet address is registered. Facilitates authentication for Web App access.
`uploadFileMetadata()`	Uploads file metadata (file name, chunk URLs, Merkle Root).
`getFileMetadata()`	Fetches metadata of a file using the user's address and file index.
`updatePeriodicHash()`	Stores the status of periodic integrity verification(Enabled/Disabled).

Table 3: Functions defined in Smart Contract.

5.2 Data Transformation (Encryption and hashing of files):

For the calculation of Merke root representing the complete dataset, each file uploaded by the user is split into smaller chunks by performing data fragmentation after that data is encrypted using AES 256 in Cipher Block Chaining (CBC) mode and to carry out encryption, Node.js library CryptoJS is used. The process involved calculation of encryption key, key is generated from the user blockchain wallet signature using the PBKDF2 (Password-Based Key Derivation Function 2)and using this derived encryption key , IV (Initialization Vector) the AES algorithm encrypts each data chunk, for encryption process node library CryptoJS is used.

Post encryption each file is hashed using SHA-256 (Secure Hash Algorithm). Different input generates unique hash that is completely non-duplicate by any other input of data, Once all computations are complete user can view the calculated hash for each chunk along with the encrypted data and can also see the computed merkle root of chunk hashes as showing in Figure 4 and Figure 5.

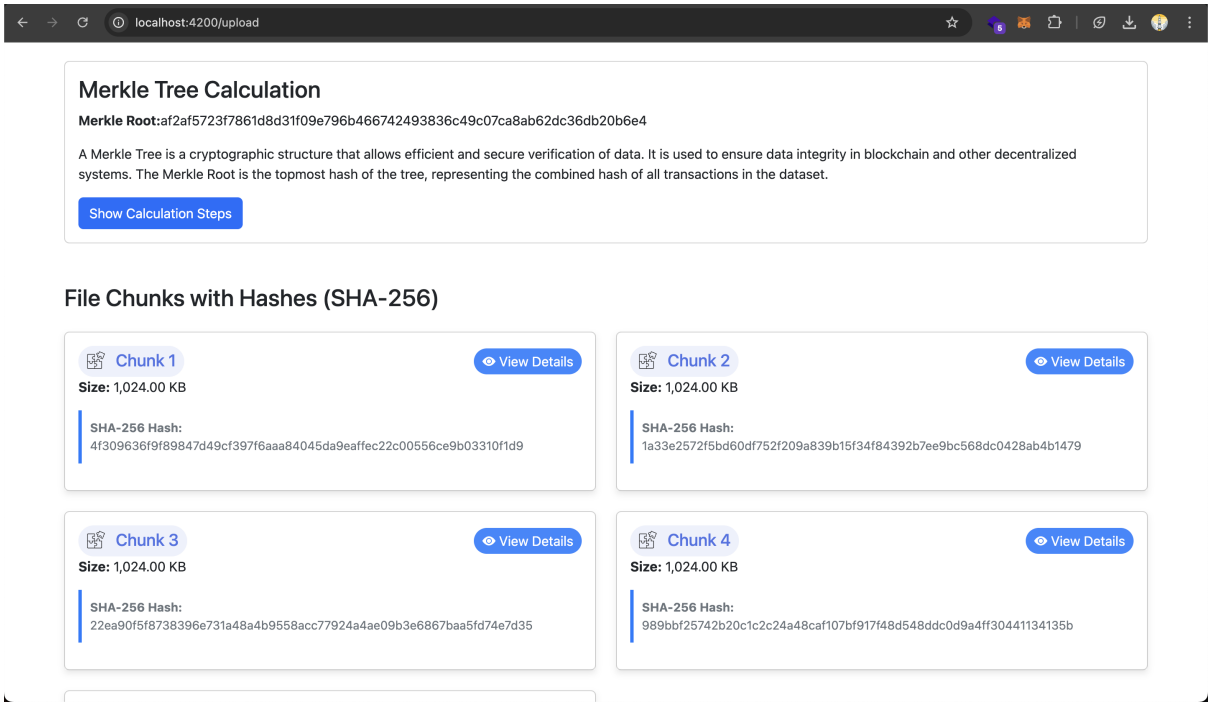

Figure 4: AWS S3 Bucket storing user uploaded file chunks

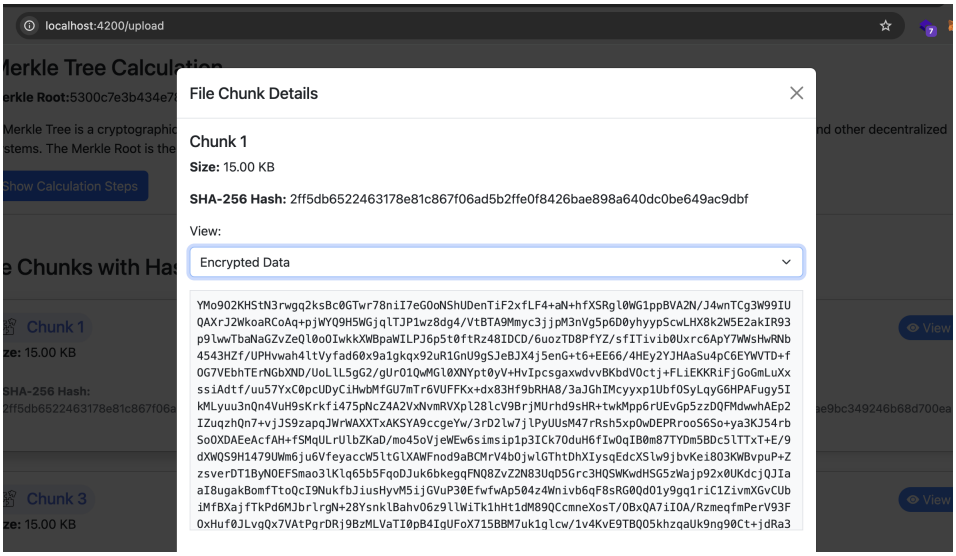

Figure 5: Page to view encrypted chunk data.

5.3 Integrating Amazon S3

Integration of AWS S3 with web application was carried out using AWS SDK. A bucket named 'blockstore-data' was created for storing encrypted file chunks, Figure 6. shows the S3 dashboard with uploaded content, once user uploads the file the generated encrypted chunks are stored to this bucket. Parameters like the region name (us-east-1) and bucket name were configured in the application in environment file and a IAM role was created to access the S3 service using access key ID and secret access key.

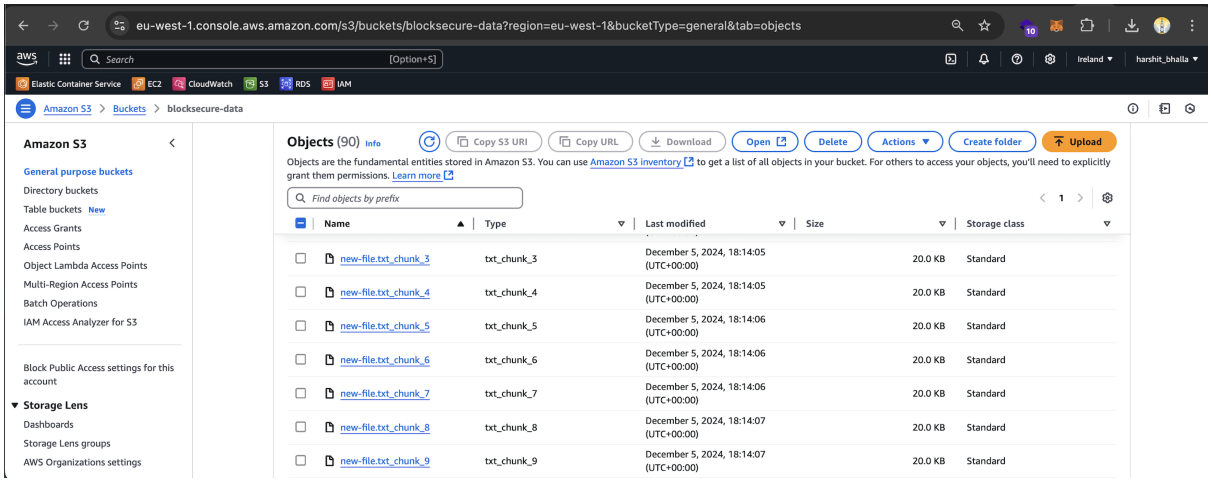

Figure 6: AWS S3 Bucket storing user uploaded file chunks

6 Evaluation

The Evaluation of the web application was conducted to measure its performance in terms of security and integrity of data. These performance analyses align with the research objectives and provides detail on how system solves the issues related to cloud storage security.

6.1 Experiment 1:

To maintain data integrity with the use of Merkle Trees and the immutable storage of Merkle Roots over a blockchain network for ensuring tamper-proof data storage.

The main objective of this research is to verify the integrity of user uploaded data under cloud environment, The web application solves this issue with the help of Merkle tree hash calculations, the evaluation is based on testing scenario when integrity of data is violated or uploaded data is tempered with, the system demonstrates this results by performing comparison between the newly calculated Merkle root and the one stored on blockchain.

The system was able to identify the violation of data successfully, To achieve this 10 diffrent files were uploaded from a user account to cloud (AWS S3) and 4 of them were manually altered to demonstrate a data breach or data tempering attempt by attacker.Results from web application are shown in Figure 7, the tempering of data were detected in real time for all altered files.

...

6.2 Experiment 2:

To address data privacy issues of cloud storage solutions by implementing and evaluating wallet based encryption and deception method.

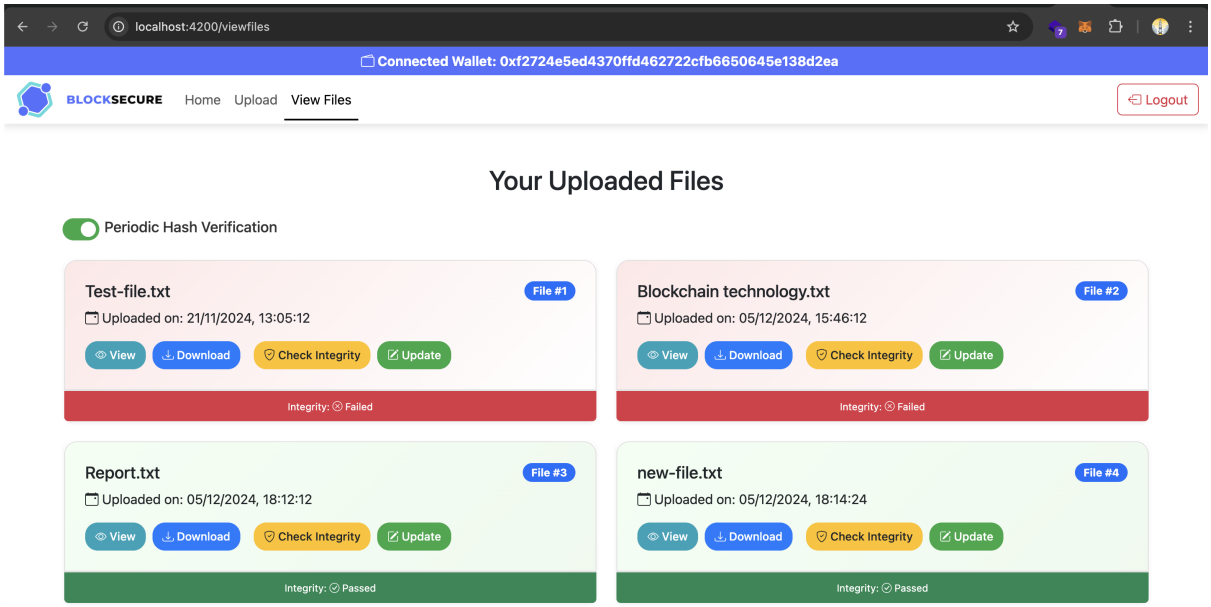

Figure 7: Page to view uploaded file integrity.

To Evaluate the encryption and decryption process for different size files, a test was performed to evaluate the effect of file size over encryption. Since we need to fragment the data to calculate SHA Merkle root hash, I setup the size of chunk to be 1MB. The table 4 shows the evaluation results and result are plotted in graph that demonstrates encryption and decryption times for different file sizes shown in Figure 8

The result from this experiment depicts a direct relation between the size file and both encryption and decryption times such as :

- As the size of file increases both encryption and decryption times increase linearly .
- encrypting a 1MB file took around 110ms, whereas a 100MB file requires over 11.4 seconds.
- Decryption times are slightly faster than encryption times across all file sizes, with average time of 114ms for encryption of and 101ms for decryption.
- System works uniformly even with 100mb file with around 100 chunks, this shows that system demonstrates scalability for handling larger files, making it suitable for real world application.

6.3 Experiment 3

To assess the automated periodic integrity verification system and measure its efficacy in detecting tampering of data stored over cloud (AWS S3).

File Size (MB)	Encryption Time (ms)	Decryption Time (ms)
1	110.34	106.11
5	775.97	656.77
10	1586.50	1367.87
50	5898.43	5345.73
100	11428.35	10017.34

Table 4: Encryption and Decryption Time (MS) / File Sizes(MB)

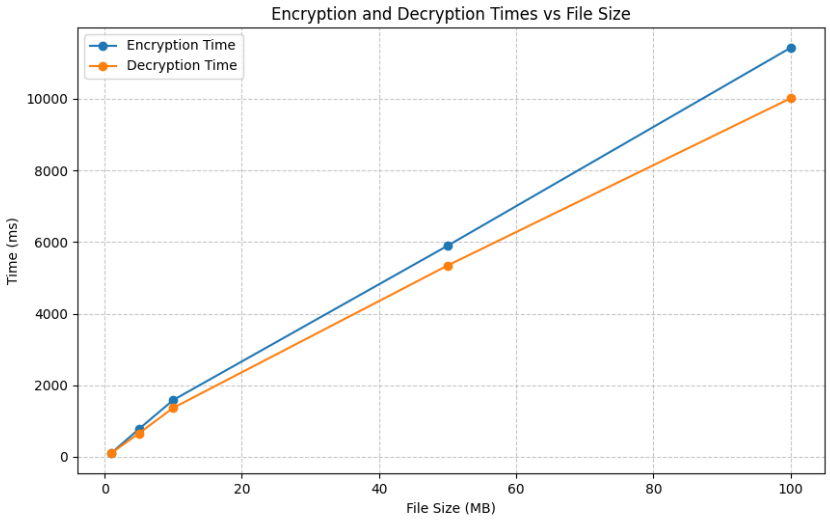

Figure 8: Encryption/Decryption Time for Different Size File.

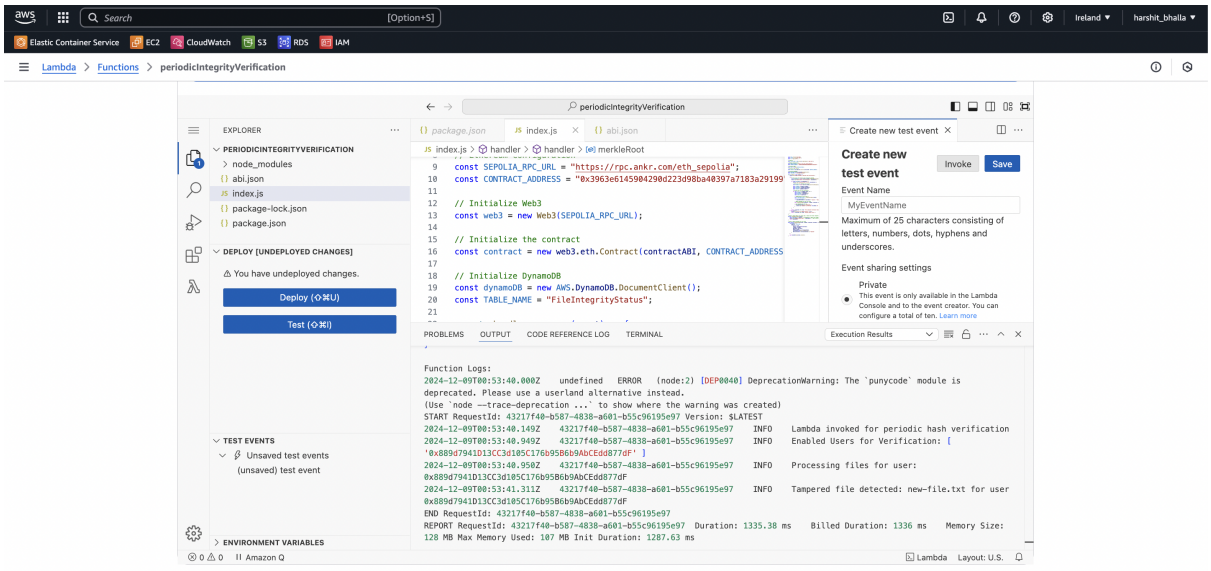

Figure 9: Periodic integrity Verification

To achieve this a files was uploaded from a user's account to cloud (AWS S3) was manually altered to demonstrate a data breach or data tempering attempt by attacker.To assess the system, A file tampering occurs and observe the system's response in notifying the user. Lambda function results are shown in Figure 9

6.4 Discussion

The findings of the research highlights through the experiments conducted that integrating blockchain technology, cryptographic techniques, and cloud storage has a potential to address the challenges of security, integrity of cloud storage. However, these experiments also shows the opportunities for improvement.

One of the findings based on Merkle Tree based Integrity Verification was that the system was able to detect the data tampering. The tampering is detected by comparing the recalculated Merkle root with the previous stored root on the blockchain. This guarantees the Merkle tree structures integrity consistent with findings in related work, such as (Liu et al., 2023). Immutability is a strength of blockchain that ensures instant detection of tampered data.

The second experiment that was conducted on wallet based encryption and decryption performance. The application's use of AES-256 encryption aligns with prior studies emphasizing its security, such as (Idrus et al., 2023). However, the encryption and decryption times increases with increase in file size. Decryption time is slightly faster that shows efficient key management. However, the encryption process has computational overhead that could be a bottleneck for data operations.

Another experiment on periodic integrity verification automated tampering detection and sent notifications for the same. The real time alerts enhances the user experience and brings in the trust. The use of AWS Lambda functions gives the efficiency in cost execution. However, having periodic triggers instead of continuous monitoring can result in delays.

There are few limitations observed throughout the experiments that are as follows:

- As Ethereum's transaction cost is high, it poses a challenge to keep the scalable blockchain. This limitation impacts the feasibility of scaling the system for high-frequency metadata updates, as highlighted in related studies, such as (Rouhani and Deters, 2018).
- The dependency on centralized cloud services (AWS S3) for storage partially contradicts the decentralized ethos of the system, might create single point of failure.
- Performance overheads as discussed above in experiments too, large files creates high-loads on data fragmentation, encryption and Merkle tree. These overheads could become significant in high load environments.

The improvements that can make application better could be exploring alternative blockchains to reduce costs and increasing the throughput. One of the other improvement can be real time integrity checks that is based on implementing integrity verification using services like AWS EventBridge for immediate responses. Incorporating decentralized storage solution could further decrease the dependency on centralized cloud providers. Bettering the user experience by providing more intuitive interfaces and improve engagement for periodic verification and displaying integrity logs could be another improvement.

7 Conclusion and Future Work

Nowadays, there are lot of data breaches, leading a threat to security and integrity of a critical information. This left me wondering about how can blockchain technology can ensure data encryption, security and integrity of data for cloud storage systems. Driven by this concern, my primary goal of the research became to address the security and integrity challenges of centralized cloud storage through blockchains technology. There are techniques such as data fragmentation, data encryption, automated integrity that could address the challenges.

The step by step method of research included was firstly creating a secured framework for cloud data storage by integrating blockchain technology with AES-256 encryption and SHA-256 hashing. Second step was about the integrity where the Merkle Tree concept was used. Next step was about addressing the data privacy. Data privacy was handled by authentication methods. There was an implementation of detection of any tampering with real-time notifications.

The research question successfully achieved the objectives and key findings were:

- The enhanced security due to implementation of AES-256 encryption ans SHA-256 hashing.
- To ensure the integrity of uploaded files, application uses Merkle tree-based verification that uses blockchain immutability to ensure that any unauthorized modification or tampering with the file chunks is immediately detectable.
- The use of blockchain technology leads to more transparency. A tamper proof mechanism was created by storing merkle roots in Ethereal blockchain for verification of data stored in S3.
- The integration of smart contracts and AWS Lambda helped in testing the file integrity periodically.

While the framework shows strong potential for improving the data security and integrity but there are few limitations such as high transaction costs and scalability issues of Ethereum and some performance issues data fragmentation and encryption for large datasets. There are future works that can be included as follows:

- Find an alternative blockchain that can lower the transactional costs and higher throughput.
- Developing a more efficient data fragmentation and encryption for large datasets

References

Akhil, K. M., Kumar, M. P. and Pushpa, B. R. (2017). Enhanced cloud data security using aes algorithm, *2017 International Conference on Intelligent Computing and Control (I2C2)*, pp. 1–5.

Almadani, M. S. and Hussain, F. K. (2023). Implementing a secure blockchain-based wallet system with multi-factor authentication, *2023 IEEE International Conference on e-Business Engineering (ICEBE)*, pp. 23–30.

Barona, R. and Anita, E. A. M. (2017). A survey on data breach challenges in cloud computing security: Issues and threats, *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1–8.

Castro-Medina, F., Rodríguez-Mazahua, L., Abud-Figueroa, M. A., Romero-Torres, C., Reyes-Hernández, L. A. and Alor-Hernández, G. (2019). Application of data fragmentation and replication methods in the cloud: a review, *2019 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pp. 47–54.

Devi, S. S. and Jayasri, K. (2023). Esha-256: An enhanced secure cryptographic hash algorithm for information security, *2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS)*, pp. 952–958.

Feng, Y. and Li, M. (2022). Data integrity and security in cloud storage: Challenges and emerging solutions, *IEEE Transactions on Cloud Computing* **10**(3): 809–822.

Goswami, P., Faujdar, N., Debnath, S. et al. (2024). Investigation on storage level data integrity strategies in cloud computing: classification, security obstructions, challenges and vulnerability, *Journal of Cloud Computing* **13**(45).
URL: *https://doi.org/10.1186/s13677-024-00605-z*

Idrus, M. A. Z. B., Rahman, F. D. A., Khalifa, O. O. and Yusoff, N. M. (2023). Blockchain-based security for cloud data storage, *2023 IEEE 9th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pp. 73–77.

Jha, S., Modak, A., Pise, R. and Patil, S. (2023). Certifier dapp - decentralized and secured certification system using blockchain, *2023 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, pp. 1–6.

Liu, H., Luo, X., Liu, H. and Xia, X. (2021). Merkle tree: A fundamental component of blockchains, *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pp. 556–561.

Liu, Z., Ren, L., Feng, Y., Wang, S. and Wei, J. (2023). Data integrity audit scheme based on quad merkle tree and blockchain, *IEEE Access* **11**: 59263–59273.

M, P. H., Shankaraiah, S. and R, S. (2022). Patient health information framework using aws s3 service, *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, pp. 1–5.

Neto, N. N., Madnick, S., de Paula, A. M. G. and Borges, N. M. (2020). A case study of the capital one data breach, *Working Paper CISL# 2020-07*, Cybersecurity Interdisciplinary Systems Laboratory (CISL), Massachusetts Institute of Technology.
URL: *https://web.mit.edu/smadnick/www/wp/2020-07.pdf*

Rouhani, S. and Deters, R. (2018). Performance analysis of ethereum transactions in private blockchain, *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74.

Sari, L. and Sipos, M. (2019). Filetribe: Blockchain-based secure file sharing on ipfs, *European Wireless 2019; 25th European Wireless Conference*, pp. 1–6.

Shakor, M. Y., Khaleel, M. I., Safran, M., Alfarhood, S. and Zhu, M. (2024). Dynamic aes encryption and blockchain key management: A novel solution for cloud data security, *IEEE Access* **12**: 26334–26343.

Sharma, P., Jindal, R. and Borah, M. D. (2019). Blockchain-based integrity protection system for cloud storage, *2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, pp. 1–5.