*Journal of Computer Science*

Research Article

# Design and Evaluation of a Blockchain-Integrated Secure Cloud Storage System

**Walde Rajesh Baliram, Bashir Alam and M. N. Doja**

*Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India*

*Article history*
Received: 08-02-2026
Revised: 07-06-2026
Accepted: 08-06-2026

**Corresponding Author:**
Walde Rajesh Baliram
Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India
Email: rajesh.walde@gmail.com

**Abstract:** Cloud storage is a critical part of modern computer systems that has a centralized structure, which opens sensitive data to security risks, including unauthorized access, insider threats, and data tampering. In order to mitigate these issues, this paper proposed a novel cloud storage system with built-in blockchain, AES-256 encryption, access control using smart contracts, and a dual-hash integrity check system. The encrypted data files are then outsourced to the cloud, and cryptographic hash, ownership, and access control policies are stored on a blockchain ledger. As AES encryption uses randomized initialization vectors, encryption of the same file with subsequent encryption yields a different ciphertext. To prevent falsely identifying integrity failures, the proposed system calculates and stores the hash of the plaintext file on the blockchain prior to the encryption of the file, and ensures that the integrity can be reliably and consistently checked. The system is entirely executed in Python as a hash-linked blockchain with a JSON-based persistence and is tested by the experiments using Cloud Sim-style metrics. It analyses performance metrics such as latency, throughput, AES overhead, and blockchain overhead in three scenarios: cloud-only, cloud with AES encryption, and cloud with AES encryption and blockchain. The experimental result proves that the proposed system substantially improved data confidentiality, integrity, access control, and auditability at the expense of moderate and tolerable performance overhead.

**Keywords:** Cloud Storage, Data Security, Blockchain, Encryption, AES-256, Smart Contract

## Introduction

Cloud computing is currently a fundamental technology in existing information systems and is capable of delivering on-demand and scalable storage as well as computing services at minimal cost of infrastructure (Lui et al., 2011). Cloud storage is being used by enterprises and individual users as an option to store a considerable amount of data due to its flexibility, availability, and accessibility (Yang et al., 2020). Nevertheless, the fact that cloud storage is centralized poses severe risks to data confidentiality, integrity, and access control (Tabrizchi and Rafsanjani, 2020). As the information is stored and operated by third-party cloud systems, clients have to trust those systems implicitly, and cloud environments are prone to insider attacks, unauthorized access, and data manipulations (El Kafhali et al., 2022).

To secure confidential data stored on the cloud, encryption techniques are normally used. AES is the best-known symmetric-key algorithm due to its usability and the high level of security. It is believed that the AES-256 encryption standard is very tough towards brute-force attacks, and it is widely employed to secure data during transit and rest (NIST, 2023). Even though encryption is an effective way of guaranteeing the confidentiality of data, it does not necessarily provide data integrity verification mechanisms, transparent access control mechanisms, or reliable audit trails. As a result, even encrypted data on centralized cloud solutions can be modified or removed without the data owner knowing about it.

The blockchain technology has become one of the potential solutions to resolve trust and integrity concerns in distributed systems (Yaga et al., 2018). Invented by Nakamoto as the core technology behind Bitcoin, blockchain allows creating a decentralized, immutable registry with the use of cryptographic hashing and distributed consensus (Nakamoto, 2008). When data is entered into the blockchain, it is computationally infeasible to modify without being detected. These characteristics precondition the fact that blockchain is

especially applicable in the case of the implementation of applications with integrity verification, non-repudiation, and open auditing (Zou et al., 2021). Blockchain may serve to store metadata in cloud storage, including file hashes, file ownership, file access policies, and file access logs, which decreases the dependency on a centralized trusted entity (Sharma et al., 2020).

A number of current research studies from academics and industry have explored blockchain-based cloud storage for decentralization, immutability, and traceability (Huang and Yi, 2024). To offer fine-grained access control and data privacy, some solutions combine blockchain with attribute-based encryption (Zhang et al., 2020), searchable encryption (Li et al., 2019; Zhang et al., 2018), or decentralized storage platforms (Sharma et al., 2021). In spite of the fact that these solutions have good security guarantees, they are usually characterized by high computational complexities, high latency, and low scalability. Besides, most of the current literature does not provide an in-depth performance assessment at realistic cloud operating conditions, thus it is not easy to determine their applicability.

It is based on these issues that this paper suggests a novel, lightweight, and practical blockchain-based secure cloud storage model, which incorporates AES-256 encryption with blockchain-based access control and data integrity verification. In the proposed system, the data is encrypted locally and uploaded to a cloud in such a way that it is not revealed to the storage provider in plaintext form. Blockchain directly handles only metadata (including cryptographic hash and access control policies) to lower storage expenses without impacting the privacy of metadata-based access control (metadata is not stored). This system is developed in Python and measured with Cloud Sim-style metrics that allow control of the experiment and obtain reproducible results in the performance indicators, including latency, throughput, AES overhead, and blockchain overhead. The obtained results of the experiment have shown that the suggested strategy provides a significant improvement in data security and tamper resistance, along with introducing an acceptable overhead in performance, which is acceptable in applications of secure cloud storage.

## *Literature Review*

The need to secure cloud storage has been an engaging field of study owing to the increase in the use of third parties to provide their cloud services and the amount of sensitive information that is being stored outside the premises. The initial studies concentrated on cryptographic practices to ensure that the data privacy in clouds is secured (Rehman et al., 2021; Sun, 2020; Yang et al., 2020). Conventional methods used asymmetric and symmetric encryption techniques to protect data at rest and in transit (Agarwal et al., 2020; Mareddy, 2026). One of them is the Advanced Encryption Standard (AES), which has been popular with its efficiency, speed, and high-security assurance (Wang et al., 2025). Nevertheless, these encryption-based solutions are mostly based on an assumed trusted cloud provider and lack proper coverage of integrity checks, insider threats, and transparent access auditing.

To address the shortcomings of centralized models of trust, scientists have tried to apply blockchain technology to promote cloud security (Han et al., 2022; Khanna et al., 2022; Li et al., 2020, 2021). Decentralization of blockchain and irreversibility allow safe storage of metadata, logs, and data operation verification is completely transparent (Darwish et al., 2020; Sharma et al., 2020). In Bitcoin, it was Nakamoto who proposed blockchain as the fundamental technology that depicted the probability of having an append-only distributed ledger that was not controlled by a central authority (Nakamoto, 2008). The different works proposed blockchain-oriented systems in which the cryptographic hash of the files stored in the cloud is stored on the blockchain to ensure data integrity and non-repudiation (Alharby, 2024; Ganesan and Arulkumaran, 2018).

One of the more general literatures is the combination of blockchain and more advanced encryption algorithms, such as the Attribute-Based Encryption (ABE), to provide fine-grained access control to the cloud storage system (Musa et al., 2025; Punia et al., 2024; Yan et al., 2023). These schemes give the ability to define access rights on the basis of attributes of users as opposed to identities, which gives flexibility and high security assurance. Nevertheless, ABE-based systems typically have high computational cost, complicated key management, and scalability issues, which do not make them appropriate for practice in large-scale cloud deployment.

The alternative area of study is the combination of blockchain and decentralized data storage platforms such as the Inter-Planetary File System (IPFS). In these methods, access control and metadata are managed with the help of blockchain, and the actual data is stored in IPFS in a peer-to-peer manner (Hasan et al., 2019; Sultana et al., 2023). Even though such systems are not dependent on centralized cloud providers, they create unpredictable latencies, availability issues, and complexity in systems. Besides, decentralized storage systems cannot be easily tested with standard cloud simulation frameworks, restricting the ability to replicate and compare the performance outcomes.

The recent survey articles highlight the necessity of lightweight and practical blockchain-cloud integration models that may balance the increase in security with acceptable performance overhead. Latency is one of the factors that can influence the performance of an application greatly, and it is a vital business aspect that may influence user experience and customer satisfaction

directly (Alex, 2025; Patil, 2025). The point of these surveys is that most of the current solutions are not practically feasible because they are either too slow (latency) or too large (storage overhead), or they have not been experimentally validated. Specifically, one may not pay much attention to such performance metrics, such as latency, throughput, and blockchain overhead storage, which are relevant to real cloud applications.

Compared with the previous work, the proposed system has a pragmatic design that employs AES-256 encryption in conjunction with access control and integrity verification on blockchain and traditional cloud storage of encrypted data. The system reduces the storage overhead of the blockchain and computation cost by placing only cryptographic hashes, access policies, and audit logs on the blockchain. The off-chain storage was advantageous to blockchain as it was scalable, storage speed was validated, and efficient (Gai et al., 2020). Moreover, the fact that CloudSim is used allows for conducting systematic and repeatable performance testing in a variety of scenarios. However, unlike numerous other existing studies, this one contains the experiment of the attack simulation to empirically prove the improvement of the security and tamper resistance.

Even though there has been massive progress in ensuring the security of cloud storage by the integration of encryption and blockchain, there are still several significant gaps in the research. Firstly, most of the currently available cloud security systems use encryption methods only, which guarantee confidentiality but do not provide transparent integrity checks, decentralized access control, and immutable audit. This drawback exposes systems to insider attacks and unnoticed data interference.

Second, many of the blockchain-based cloud storage systems combine intricate cryptographic algorithms like attribute-based encryption or decentralized storage systems like IPFS. Although these methods provide enhanced security, they tend to create a high level of computational overhead, sophisticated key management, erratic latency, and poor scalability. Besides, they do not have lightweight designs that limit their use in the real-world enterprise cloud.

Third, a majority of the available works concentrate on the conceptual architectures or security analysis and offer few experimental validations. Standard performance measurement based on standard measures like latency, throughput, blockchain overhead, and storage overhead is often lacking. Due to this, the trade-offs in the performance and feasibility of the blockchain-enabled solutions of cloud security have not been sufficiently investigated.

Lastly, attack simulation and empirical security validation are not typically done in previous works. Numerous studies assume the hypothetical security assurances but do not explain how efficiently suggested systems can overcome actual attack situations, including unauthorized access, tampering with data, or replay attacks.

This study addresses these shortcomings by developing a novel, lightweight, and user-friendly unified framework that integrates AES-256, dual-hash integrity verification (plaintext and ciphertext), and blockchain-based auditability for secure cloud storage. The main goal of the study is to design and deploy an efficient, secure, and feasible cloud storage system that surpasses the failure of the traditional models of centralized cloud security requirements. The objectives of the proposed work are:

- Integration of AES-256 encryption, cloud storage, and blockchain
- Tamper-proof audit and verification model based on blockchain
- Dual-hash (plaintext and ciphertext) integrity check system
- Performance analysis through CloudSim-style metrics
- Mathematical attack modeling and experimental attack simulation
- Security-performance trade-off evaluation

The above objectives were met in the proposed system by bridging the differences between the theoretical framework of blockchain-based cloud security and actual and performance-oriented cloud storage solutions.

## Methods

### *Architectural Design*

The architecture design is comprised of the three interoperable layers, that is, the cloud storage layer, the blockchain ledger, and the smart-contract layer, as shown in Fig. 1.

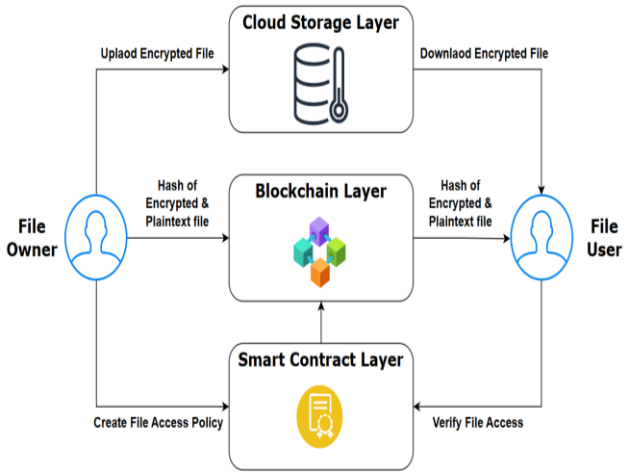


**Fig. 1:** Proposed Architecture Diagram

The lowest tier is the cloud storage layer, which archives the user data in encrypted format through the assistance of the Advanced Encryption Standard (AES-256). AES offers data confidentiality both at rest and in transit. Encryption is of the form:

$$C = E_K(F)$$

Where:

$F$ is a plaintext file
$K$ is a symmetric encryption key
$C$ denotes the encrypted text that is stored on the cloud

The second layer is the blockchain layer, which writes metadata of all files stored, including the hash digest, identifier of the uploader, date of creation, and cloud object reference. The links between any block and the one before it is cryptographically expressed, by recursive computation on the hash function SHA-256, as:

$$H_{\text{block}} = SHA256(H_{\text{prev}} \;||\; Tx_{\text{data}})$$

Where:

$H_{prev}$ stands for the previous block hash
$Tx_{data}$ represents data content within a transaction
$H_{block}$ stands for the current block hash

It is impossible to modify the component of the chain of hashes with this forwarding connection. Any unauthenticated modification renders the full nullity of the entire chain of hash, and thus, the detection of the manipulation is immediate and without centralized authority. Trustless access control, integrity, and auditability are presented in the blockchain layer.

The third layer is the smart-contract layer, which specifies and implements access control policies. The use of smart contracts based on blockchain provides strict control over access to cloud data. Only successful on-chain authorization of a download request results in downloading the resource, thus eliminating unauthorized access even by privileged insiders of the cloud. The smart contracts of blockchains implement decentralized access control, ensure data integrity, and offer an immutable store of hash, transparency, and auditability. Both are instrumental in enhancing privacy and security within cloud storage systems.

## *Algorithm*

**Algorithm 1: Secure File Upload.**

**Purpose:** To upload a file to the cloud storage and ensure data confidentiality, integrity, and access control utilizing AES-256 encryption and blockchain smart contracts.

**Input:**

- Plain file $F$
- The symmetric encryption key $K$ of AES-256.
- Data Owner Identifier *OwnerID*
- File Identifier *FileID*
- Access policy $P$

**Output:**

- Encrypted file $E_K(F)$ stored on cloud storage.
- Dual hash metadata recorded on Blockchain.

**Steps:**

1. **File Selection:** The owner of the data picks a file $F$ to be stored safely in the cloud.
2. **Generation of plaintext Hash:** To allow an integrity check in the future, a hash $H_P$ of the plaintext file is calculated using SHA256.
$$H_P = SHA256(F)$$
3. **Key Generation:** The data owner generates an AES encryption key $K$ of 256 bits.
4. **Client-Side Encryption:** The chosen file $F$ is encrypted on the client-side with the aid of the AES-256 algorithm in key $K$, resulting in the encrypted file $E_K(F)$.
5. **Generation of encrypted file Hash:** To allow file tampering check in the future, a hash $H_E$ of the encrypted file is calculated using SHA256.
$$H_E = SHA256(E_K(F))$$
6. **Cloud Upload:** This is the encrypted file $E_K(F)$ uploaded to the cloud storage service.
7. **Creation of a blockchain Transaction:** This involves the creation of a blockchain transaction consisting of,
   - File identifier *FileID*
   - Owner identifier *OwnerID*
   - Value of plaintext file hash $H_P$.
   - Value of encrypted file hash $H_E$.
   - The policy of file access (list of authorized users).
8. **Smart Contract Execution:** It confirms the transaction and stores permanently the metadata, the hash, and the access policy on the blockchain ledger.
9. **Upload Confirmation:** Once the encrypted file is stored in the cloud and the metadata is stored in the blockchain, the system sends an upload confirmation to the data owner.

End of Algorithm 1

**Algorithms 2: Blockchain-based access control.**

**Purpose:** To implement secure, decentralized, and tamper-resistant access control to cloud-stored encrypted files with the help of blockchain smart contracts.

**Input:**

- User Identifier *UserID*
- File Identifier *FileID*
- Access request type (read /download)
- Smart contract and blockchain ledger.

**Output:**

- Grant/Deny access decision
- Immutable access log stored in the blockchain.

**Steps:**

1. **Access Request Initiation:** An authentic or an unauthenticated user sends a request to gain access to a file stored in the clouds using the *FileID.*
2. **Smart Contract Invocation:** To the deployed smart contract that controls access to files, the access request is sent to the blockchain network.

3. **Policy Retrieval:** The smart contract gets the blockchain ledger of the access control policy of *FileID*. The policy includes:
   - File owner identifier
   - List of authorized users
   - Allowed operations (read/download)
4. **User Authentication:** The smart contract inspects the identity of the requesting user, *UserID*, with blockchain credentials (e.g., registered address or digital identity).
5. **Authorization Check:** Use of a smart contract ensures that the *UserID* meets the access policy assigned to *the FileID*.
6. **Access Decision:**
   - The smart contract can provide access based on permission to the user; in case he/she is authorized.
   - In case of unauthentication of the user, the smart contract rejects the request.
7. **Access Logging:** An access attempt is logged on the blockchain as an unalterable log record with the following contents:
   - *UserID*
   - *FileID*
   - Timestamp
   - Access decision (Grant/Deny)
8. **Decision Notification:** The access decision is sent back to the requesting user and the cloud storage service, which permits or denies the file download as required.

End of Algorithm 2

**Algorithm 3: File Integrity Checking.**

**Purpose:** To ensure the integrity of cloud-encrypted files and check whether any unauthorized modification or tampering of the files occurred based on the blockchain-stored cryptographic hash.

**Input:**
- Downloaded encrypted file, $E_K(F)$.
- File Identifier *FileID*
- Blockchain registry with stored hash values $H_P$ and $H_E$.

**Output:**
- Status of integrity check (Valid or Tampered)
- Authority to carry out decryption or abort operation.

**Steps:**
1) **File Download:** After the access is approved by the means of access control based on blockchain, the encrypted file $E_K(F)$ is downloaded to the cloud storage.
2) **Local Hash Computation:** The client calculates a hash $H_E'$ of the downloaded encrypted file using SHA256.
$$H_E' = SHA256(E_K(F))$$
3) **Blockchain Hash Retrieval of Encrypted File:** The system recovers the original hash $H_E$ of *FileID* of the blockchain ledger through the smart contract.
4) **Tamper Detection:**
   - If $H_E' = H_E$, then the file is assumed to be untampered.
   - If $H_E' \neq H_E$, then the file has been reported to be tampered with or corrupted.
5) **Tamper Handling:** When a mismatch occurs, the system ends the decryption process and sends the user and system administrator a tampering alert.
6) **Decryption Authorization:** Once tamper detection is passed, the encrypted file is then decrypted with the help of the AES-256 key *K* in order to recover the original plaintext file $(D_K(E_K(F)))$.
7) **Computation of decrypted file Hash:** The client calculates a hash (*Hp'*) of the decrypted file using SHA256.
$$H_D' = SHA256((D_K(E_K(F)))$$
8) **Blockchain Hash Retrieval of Plaintext File:** The system recovers the original hash $H_P$ of *FileID* of the blockchain ledger through the smart contract.
9) **Integrity Verification:**
   - If $H_D' = H_P$, then the integrity of the file has been established.
   - If $H_D's \neq H_P$, then the file has been reported to be tampered with or corrupted.
10) **Integrity Verification Logging:** The verification result (Valid or Tampered), timestamp, and file identifier are optionally stored on the blockchain to audit and perform forensic analysis.

End of Algorithm 3

**Algorithm 4: Smart Contract-Based File Access Policy Enforcement**

**Purpose:** The smart contract checks whether a user is authorized to access a file.

**Input:**
- File_ID – Identifier of the requested file
- User_ID – Identifier of the requesting user
- Action – Requested operation (Upload / Download / Access)

**Output:**
- Access_Granted or Access_Denied

**Steps:**
1) **Receive Access Request:** The user sends a file access request to the blockchain network containing File_ID, User_ID, and the requested Action.
2) **Retrieve File Metadata:** The smart contract requests the blockchain ledger to derive metadata of the File ID, which includes:
   - File owner ID
   - Authorized user list (access policy)
   - Stored cryptographic hashes
   - File transaction history.
3) **Validate File Existence:** If the file metadata does not exist on the blockchain:
   - Reject the request.
   - Return Access Denied.
4) **Verify User Authorization**
   - If the User_ID matches the owner ID, grant access immediately.
   - Otherwise, check whether User_ID exists in the authorized user list stored in the blockchain.
5) **Enforce Access Policy:**
   - If User_ID is included in the access policy: Grant access to the requested operation.
   - Otherwise: Deny access and terminate the request.
6) **Record Access Event:** Log the access decision (granted or denied) as a new blockchain transaction to ensure transparency and auditability.
7) **Return Decision:** Return the final access decision to the client:

- Access Granted
- Access Denied

End of Algorithm 4

## *System Workflow*

The proposed blockchain-based secure cloud storage system has a workflow that defines the entire cycle of data, i.e., from uploading to access and verification (Fig. 2). This system combines client-side encryption, decentralized access control, blockchain-based integrity verification, and cloud storage in order to permit confidentiality, integrity, and accountability.

Step 1: User Registration and Initialization: The users and the owner of the data are registered in the system with unique identifiers. Each user is provided with blockchain credentials in order to interact with smart contracts safely. Before files are uploaded, their access permissions are specified by the owner of the data.

Step 2: Secure File Upload: The data owner chooses a file to be stored in the cloud. To keep it confidential, the data is encrypted with AES-256 encryption on the client side. To facilitate integrity verification, a cryptographic hash of an encrypted file is produced using SHA-256. The computation of the file hash and the ownership information are stored in the blockchain using a smart contract, and the uploaded file is encrypted and transferred to the cloud.

Step 3: Policy Storage Blockchain: The blockchain ledger stores and verifies the access policy, file identifier, and hash through the smart contract. This information is made permanent and publicly available to be checked in the future in terms of access control and integrity.

Step 4: Access Request by User: When a user is requested to access a stored file, the request is forwarded to the blockchain layer. The smart contract will then get the corresponding access control policy and proceed to check if the requesting user is entitled to access the file or not.

Step 5: Decentralized Access Control Decision: Depending on the smart contract verification, access is granted or denied. The requested information and the decision taken on the access are logged on the blockchain as an audit log that cannot be changed and allows accountability and non-repudiation.

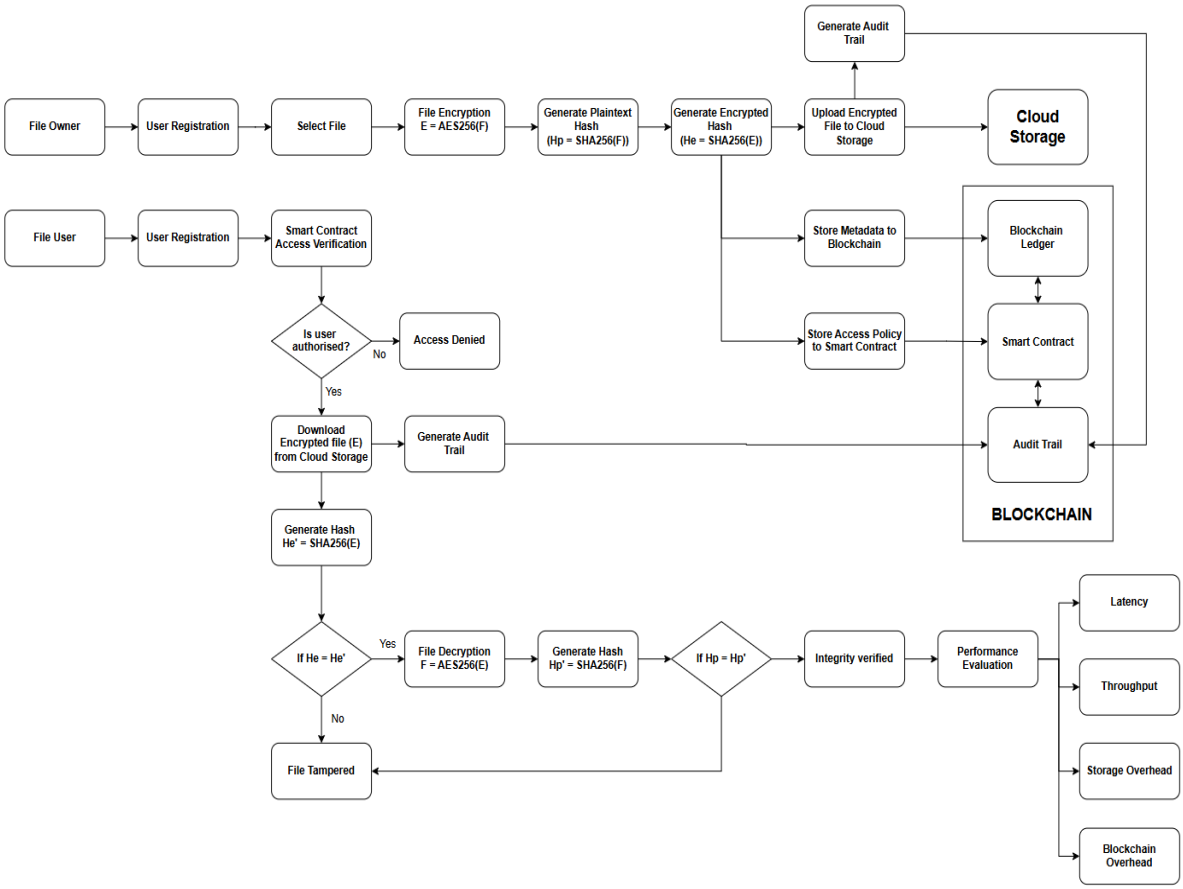


**Fig. 2:** System workflow

Step 6: Secure File Download: After authentication, the encrypted file is downloaded from the cloud store and transmitted to the end user who has been authorized. The cloud service provider is unaware of the content and access control of files.

Step 7: To Check File Tamper Detection: The system calculates the SHA-256 of the downloaded encrypted file before decryption. This hash is compared with the original one that was put in the blockchain. Any discrepancy depicts corruption or tampering of data, and the process is killed.

Step 8: File Decryption: In case of a successful tamper detection test, it decrypts the encrypted file at the client-side with the AES-256 key so as to reassemble the original plaintext file.

Step 9: Check the integrity of the file: SHA-256 computes the hash of the decrypted file in a system. This hash is compared with the hash of the original plaintext file that was put in the blockchain. Any discrepancy depicts corruption or tampering of data, and the process is killed.

Step 10: Monitoring and Performance Evaluation: During the working process, the performance indicators are recorded, which include upload latency, download latency, throughput (files per second), AES overhead, and blockchain overhead. Security events and access logs are kept for use in the assessment and analysis.

### *System Evaluation*

The system was tested in four dimensions of performance, namely, the upload latency, throughput, blockchain overhead, and storage overhead. All of these metrics are used to measure the computational efficiency, scalability, and cost of blockchain integration of the system.

### *Latency ($L_T$)*

The total delay caused by encryption ($T_{enc}$), cloud uploading ($T_{upload}$), and blockchain transaction confirmation ($T_{commit}$):

$$L_T = T_{\text{enc}} + T_{\text{upload}} + T_{\text{commit}}$$

### *Throughput (Tp)*

Throughput can be described as the number of files that can be processed successfully in one second:

$$T_p = \frac{N_{\text{files}}}{t_{\text{total}}}$$

### *AES Overhead ($O_e$)*

The increase in latency caused by adding AES encryption before uploading the file into the cloud storage (in percentage):

$$O_e = \frac{L_T^{ENC+UPLOAD} - L_T^{UPLOAD}}{L_T^{UPLOAD}} \times 100\%$$

### *Blockchain Overhead ($O_b$)*

The increase in latency caused by adding blockchain on top of AES encryption (in percentage):

$$O_b = \frac{L_T^{ENC+BC} - L_T^{ENC}}{L_T^{ENC}} \times 100\%$$

## *Experimental Evaluation*

### *Experimental Setup*

The proposed framework is implemented using a Python-based system to simulate the interaction between the encryption modules, blockchain network, and cloud storage system. Python was selected due to its extensive cryptographic libraries, flexible scripting capabilities, and ease of integrating distributed components. The experiments were conducted on a workstation equipped with the following configuration:

### *Hardware Configuration*

- Processor: Intel(R) Core(TM) i7
- RAM: 16 GB
- Storage: SSD-based storage

### *Software Environment*

- Programming Language: Python 3.x
- Cryptographic Library: PyCryptodome (AES-256 implementation)
- Hash Function: SHA-256 (hashlib library)
- Blockchain Framework: Custom Python-based blockchain implementation
- Operating System: Windows 11
- Integrated Development Environment (IDE): PyCharm 2025.2.1.1 (Community Edition)

All tests are performed in a controlled laboratory environment to maintain the condition of all the parameters equivalent. The aim of the experimental setup is to conduct performance evaluation and security analysis. The experiments were repeated 100 times with five different file sizes (10 MB, 20 MB, 30 MB, 40 MB, and 50 MB) in order to increase the consistency and validity of the findings. The average of the results taken for 100 trials is calculated to minimize the impact of random errors and outliers. The experimental plan has been prepared on the basis of the following three models.

### *Cloud-Only Models*

It uploads a plaintext file to the cloud storage.

### *Cloud With AES Models*

It encrypts the file with the AES256 algorithm and then uploads the encrypted file to the cloud storage.

### *Cloud With AES and Blockchain Model*

It encrypts a file with the AES256 algorithm and then uploads the encrypted file to the cloud storage. Additionally, it calculates a dual SHA256 hash of both the original and encrypted files and stores these hashes on the blockchain.

The encryption and upload capabilities of all the models are equal in order to make the comparison fair and to control it. The operations of blockchain are designed such that metadata is committed only, irrespective of the size of the file.

### *Performance Evaluation*

The metrics, such as latency, throughput, AES overhead, and blockchain overhead, defined in the methodology are compared for their result. To improve the consistency and validity of the findings, the experiments were repeated several times (100 times), and the results were averaged. Table. 1 provides the evaluation of all three models.

The findings shown in Table 1 indicate that the combination of AES encryption and blockchain mechanism enhances the security level with a slight performance overhead. The model shows an increase in latency and a decrease in throughput when compared to the Cloud Only model, but the overhead observed is within acceptable limits for secure cloud storage applications. With the proposed design demonstrating a relatively low blockchain overhead, especially for larger file sizes, it suggests that the architecture is scalable and could be implemented in practical applications that demand robust data protection and accountability measures.

### *Latency Analysis*

The cloud-only model has the lowest latency because it involves uploading and storing data only. The latency of the cloud + AES model is a little more triggered by the processing of AES256 encryption and the computation of the hash with SHA-256. Additional latency is introduced with the Cloud + AES + Blockchain model due to the formation of blocks, validation of the chain, recording access policy, verification of transactions, and immutable audit logs. In all three models, the latency increases with the increase in file sizes. Fig. 3 demonstrates the comparison of the latency of all three models.

**Table 1:** Performance Evaluation

| Metric | Cloud Only | | | | | Cloud + AES | | | | | Cloud + AES + Blockchain | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| File Size | 10 MB | 20 MB | 30 MB | 40 MB | 50 MB | 10 MB | 20 MB | 30 MB | 40 MB | 50 MB | 10 MB | 20 MB | 30 MB | 40 MB | 50 MB |
| Latency (sec) | 0.06 | 0.11 | 0.16 | 0.21 | 0.26 | 0.11 | 0.21 | 0.32 | 0.44 | 0.54 | 0.13 | 0.23 | 0.34 | 0.45 | 0.56 |
| Throughput (files/sec) | 15.74 | 9.20 | 6.23 | 4.76 | 3.90 | 8.70 | 4.68 | 3.10 | 2.28 | 1.85 | 7.83 | 4.39 | 2.97 | 2.21 | 1.80 |
| AES Overhead (%) | -- | -- | -- | -- | -- | 81 % | 97 % | 101 % | 109 % | 111 % | 81 % | 97 % | 101 % | 109 % | 111 % |
| Blockchain Overhead (%) | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | 11 % | 6% | 4% | 3% | 3% |

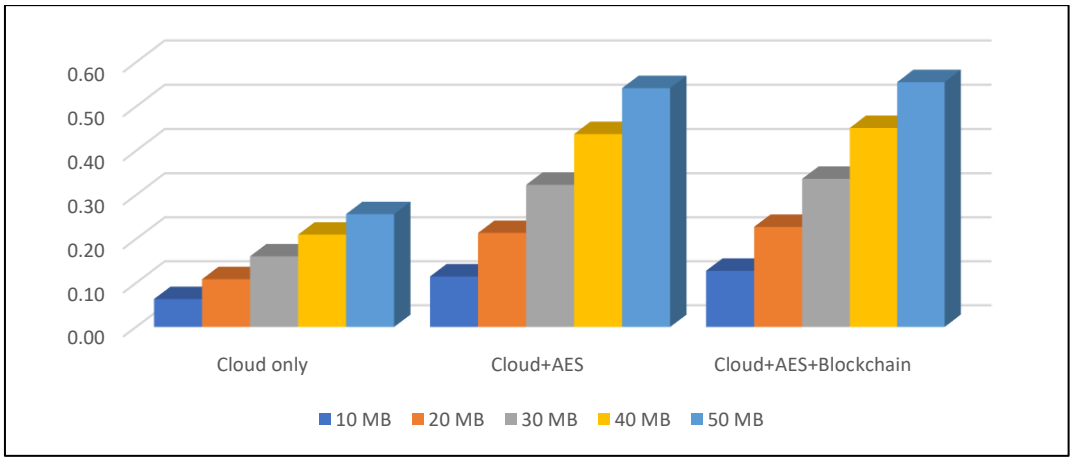


**Fig. 3:** Latency Comparison

Fig. 3 shows the clear effect of the security mechanisms on system latency. The Cloud Only model consistently has the lowest latency for all file sizes since there is no additional security processing. The use of AES-256 encryption adds latency because it requires more computing power to generate a hash and encrypt files before uploading. The Cloud + AES + Blockchain model has the maximum latency due to the operation of the blockchain, such as creating transactions, writing metadata, managing access policies, and verifying the blockchain. Despite this, the latency increment is relatively small and is directly proportional to the size of the file, demonstrating the acceptable performance of the proposed system, while it provides a significant improvement in confidentiality, integrity, access control, and auditability. The results showed that the minor latency impact on the system from the framework is outweighed by the security advantages that the blockchain system offers.

### *Throughput Analysis*

With the cloud-only model, there is very little processing prior to storage, and therefore, the system is able to get the best throughput. Through AES encryption and the generation of the SHA-256 hash, the throughput in the Cloud + AES model is reduced. The utilization of the blockchain operations in the Cloud + AES + Blockchain model decreases the throughput of the system. With the increase in file sizes, the throughput is decreased in all three models. Fig. 4 represents the comparison of the throughput of all three models.

As presented in Fig. 4, the throughput drops for all three models with an increase in file size due to extra processing required for large files. The Cloud Only model always provides the best throughput as it does not use cryptographic or blockchain processing when performing the storage operation. The Cloud + AES model has lower throughput because of the computational complexity of hash generation and encryption. The Cloud + AES + Blockchain model has the lowest throughput amongst the three models, due to the extra blockchain transactions that need to be performed, such as transaction validation and metadata recording. The throughput reduction, however, is moderate, suggesting that the proposed security enhancements do not significantly affect system efficiency.

### *AES and Blockchain Overhead Analysis*

The AES overhead in terms of latency is applicable to cloud + AES and Cloud + AES + Blockchain models, and the blockchain overhead in terms of latency is applicable to the Cloud + AES + Blockchain model only. Experimental evaluation shows that the AES overhead increases as file size increases, whereas the blockchain overhead decreases. Fig. 5 shows the comparison of the AES overhead and blockchain overhead for all three models with different file sizes.

Fig. 5 shows the system security-to-performance trade-off. The AES overhead increases with file size because larger files require more encryption and hashing operations, resulting in additional processing time. On the other hand, as the size of the file increases, the blockchain overhead decreases because the blockchain operations are not dependent on the file size and are mainly about the management of the metadata. The results show that the relative overhead incurred by this blockchain component is small and decreasing, and that the advantages offered in integrity verification, access control, and auditability are significant. As a result, the proposed architecture provides a desirable security-to-performance trade-off.

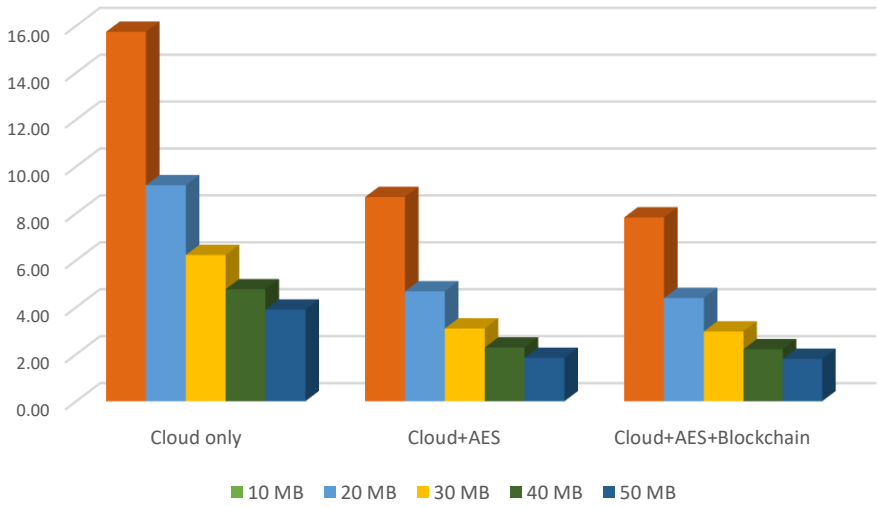


**Fig. 4:** Throughput Comparison

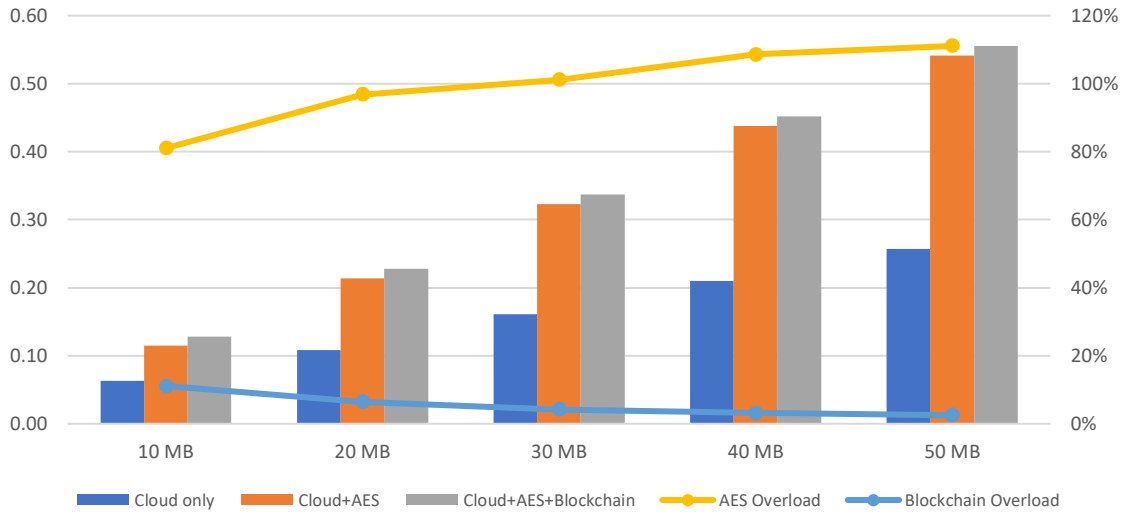


**Fig. 5:** Blockchain Overhead Comparison

### *Attack Scenarios and Protection*

We performed the following multi-file real attack simulation:

- Unauthorized access
- File tampering
- Blockchain tampering
- Brute force attack
- Reply attack

All the attacks are successful on the cloud-only model as the system is unable to prevent any of these attacks. Combining AES with cloud decreases the attack to half, whereas the integration of blockchain and AES with cloud prevents all the attacks successfully. Table 2 shows the comparison of these attacks performed on all three models.

The results presented in Table 2 demonstrate the effectiveness of integrating AES encryption and blockchain technology for mitigating common cloud security threats. The Cloud Only model is susceptible to all of the attacks evaluated, as it does not provide any mechanisms for confidentiality protection, integrity verification, or decentralized access control. With the inclusion of AES encryption, unauthorized access and brute force attacks are prevented as file contents are not disclosed to unauthorized users. AES alone does not provide immutable integrity verification or version control, as a result of which data tampering and replay attacks are not prevented. By contrast, the Cloud + AES + Blockchain model is able to effectively deal with all evaluated attack scenarios. The blockchain ledger provides tamper-evident storage with immutable hash records, and smart contracts enforce the access control policies and keep transparent audit logs. These results prove that blockchain can be integrated into the cloud storage system, providing security beyond just confidentiality, and thus contributing to a complete and robust security framework in cloud storage.

### *Security Analysis*

### *Threat Model*

The proposed blockchain-based secure cloud storage system is designed with a realistic threat model, including external attackers and malicious insiders. The main goal of the adversary is to impact the confidentiality, integrity, availability, or access control of data stored in the cloud environment.

### *Adversarial Capabilities*

The adversary is assumed to have one or more of the following capabilities.

**Table 2:** Comparison of Attack Scenarios

| Attack Scenario | Cloud Only | Cloud + AES | Cloud + AES + Blockchain |
|---|---|---|---|
| Unauthorized access | Yes | No | No |
| Data tampering | Yes | Yes | No |
| Blockchain tampering | -- | -- | No |
| Brute Force Attack | Yes | No | No |
| Reply Attack | Yes | Yes | No |

- Unauthorized Cloud Access: An attacker can try to access stored files without proper permissions or cryptographic credentials
- Data Tampering: Attackers can replace, delete, or alter cloud-stored encrypted files to breach data integrity
- Replay Attacks: An attacker can overwrite the most recent version of a file with a previous valid version to mislead users or bypass updates
- Brute-Force Attacks: The attacker can try to obtain encryption keys by performing exhaustive key-search attacks on the AES-256 encryption mechanism
- Blockchain Manipulation Attempts: An attacker can try to change metadata stored in the blockchain, access policies, audit records, or integrity hashes on the blockchain ledger
- Malicious Insider Threats: Privileged cloud administrators and users can try to access, alter, or delete data stored in the cloud without authorization

### *Trust Assumptions*

The following trust assumptions are made in the proposed system:

- Trusted Client Environment: The data owner's local device is considered safe for key generation, encryption, and decryption procedures
- Cryptographic Security: AES-256 and SHA-256 are assumed to be computationally secure against attacks that are currently considered feasible
- Blockchain Integrity: The blockchain network is assumed to be correctly maintaining consensus, and the majority of participating nodes are honest
- Secure Smart Contract Execution: Smart contracts are assumed to be implemented correctly and without software vulnerabilities
- Untrusted Cloud Storage Provider: The cloud service provider is honest, but curious. It can store data reliably, but cannot be trusted with plaintext information, access control enforcement, or integrity verification

### *Trust Boundaries*

There are three major trust boundaries in the system:

1) Client Boundary: All encryption/decryption and key management take place at the client side and not in the trust domain of the cloud provider
2) Cloud Storage Boundary: Only encrypted data is stored in the cloud, and the cloud is not trusted to ensure integrity or access control
3) Blockchain Boundary: Integrity verification and authorization decisions are based on the blockchain, which stores file hashes, ownership information, access policies, and audit logs

Based on these conditions, the proposed architecture is designed to provide confidentiality via AES-256 encryption, integrity with a dual-hash system and blockchain immutability, access control via smart contracts, and auditability via tamper-resistant blockchain records.

### *Mathematical Attack Modeling*

To evaluate the robustness of the proposed blockchain-enabled secure cloud storage system, several potential attack scenarios are mathematically modeled. These include brute-force attacks, data tampering attacks, replay attacks, and unauthorized access attacks.

Let the secure cloud system be defined as:

$$S = \{U, F, C, B, K, H\}$$

Where:

$U$ = Set of users
$F$ = Set of files
$C$ = Cloud storage
$B$ = Blockchain ledger
$K$ = Encryption keys
$H$ = Cryptographic hash functions

### *Brute-Force Key Attack*

An attacker attempts to recover the AES encryption key through exhaustive search.

Let $K = 2^{256}$ represent the AES-256 key space.

The probability of successfully guessing the correct key is:

$$P_{success} = \frac{1}{2^{256}}$$

If the attacker performs $N$attempts:

$$P_{attack} = \frac{N}{2^{256}}$$

Since $2^{256} \approx 1.15 \times 10^{77}$, the attack probability is computationally infeasible.

This ensures strong confidentiality of stored cloud data.

### *Data Tampering Attack*

An adversary modifies the encrypted cloud file.

Let $E = AES(f, k)$ be the encrypted file stored in the cloud.

The blockchain stores the hash: $H_e = SHA256(E)$

If an attacker modifies the cloud file: $E' \neq E$

A new hash is computed: $H'_e = SHA256(E')$

Integrity condition: $H'_e = H_e$

If $H'_e \neq H_e$
then $Tampering = True$
Thus, blockchain hash verification detects 100% of unauthorized file modifications.

### *Replay Attack Model*

In a replay attack, an adversary replaces the latest encrypted file with an older version.
Let $E_i$ represent version $i$ of a file.
Blockchain stores: $H_{p_i}, H_{e_i}$
If attacker replaces the file: $E_i \rightarrow E_{i-1}$
Verification condition: $SHA256(E_{i-1}) \neq H_{e_i}$
Thus, $ReplyAttack = Detected$
This prevents old file injection attacks.

### *Unauthorized Access Attack*

An attacker attempts to access a file without authorization.
Access function:

$$A: U \times F \rightarrow \{0,1\}$$

Where:

$$A(u,f) = 1 \rightarrow Access\ Granted$$

$$A(u,f) = 0 \rightarrow Access\ Denied$$

Smart contract rule:

$$Access(u,f) = \begin{cases} 1 & u = Owner(f) \\ 1 & u \in Policy(f) \\ 0 & Otherwise \end{cases}$$

If attacker:

$$u_a \notin Policy(f)$$

Then:

$$Access(u_a, f) = 0$$

Thus, unauthorized access is automatically rejected by the blockchain smart contract.

### *Hash Collision Attack*

An attacker attempts to generate two files with the same SHA-256 hash.
Collision probability:

$$P_{collision} = \frac{1}{2^{256}}$$

Using the birthday paradox approximation:

$$P_{collision} \approx \frac{n^2}{2^{257}}$$

Where $n$ is the number of hash attempts.
Even with $n = 2^{64}$, the probability remains negligible.
Therefore: $CollisionAttack \approx 0$.

### *Attack Success Rate Model*

Let:

$$A_t = Total\ Attack\ Launched$$

$$A_s = Successful\ Attacks$$

Attack success rate:

$$ASR = \frac{A_s}{A_t}$$

In the proposed system: $ASR_{proposed} \approx 0$
while in traditional cloud storage: $ASR_{cloud} > 0$
This demonstrates the security improvement achieved through AES encryption and blockchain integration.

### *Final Security Model*

Overall system resistance to attacks is defined as:

$$Security = f(C_{conf}, I_{int}, A_{auth}, R_{audit})$$

Where:

$C_{conf}$ = Confidentiality (AES encryption)
$I_{int}$ = Integrity (SHA-256 + blockchain)
$A_{auth}$ = Access control (smart contracts)
$R_{audit}$ = Auditability (immutable blockchain logs)

The confidentiality of the data is ensured by the AES-256 encryption. AES encryption offers a level of security with respect to data confidentiality such that other parties, like the cloud providers, cannot access the plaintext data. The blockchain module keeps two cryptographic hash functions: plaintext hash used to verify deterministic integrity, and encryption file hash used to detect tampering on the cloud side, which removes the possibility of integrity failure due to randomized AES encryption. The immutability of the blockchain guarantees the impossibility of altering the access policies and audit logs, and provides good non-repudiation and accountability. Smart contracts that are applied to blockchain introduce a decentralized access control, the integrity of the data stored in the form of an immutable hash, and can be audited openly. Table 3 provides a comparison of security properties.

**Table 3:** Comparison of Security Goals

| Security Goals | Cloud only | Cloud + AES | Cloud + AES + Blockchain |
|---|---|---|---|
| Confidentiality | No | Yes | Yes |
| Integrity | No | No | Yes |
| Access Control | No | No | Yes |
| Auditability | No | No | Yes |

Table 3 highlights the progressive enhancement of security properties achieved through the integration of AES encryption and blockchain technology. The Cloud Only model does not provide adequate confidentiality, integrity, access control, or auditability because data management relies entirely on a centralized cloud provider. The Cloud + AES model improves confidentiality by protecting data through strong encryption; however, it still lacks mechanisms for integrity verification, decentralized authorization, and transparent auditing. The proposed Cloud + AES + Blockchain model satisfies all four security objectives simultaneously. AES-256 ensures data confidentiality, while blockchain-based dual-hash verification guarantees data integrity. Smart contracts provide decentralized access control, and immutable blockchain records enable comprehensive auditability. The results demonstrate that the proposed architecture delivers a holistic security framework capable of addressing the major limitations of traditional cloud storage systems while maintaining practical deployment feasibility.

## Discussion

The outcome of the experiment shows that AES-256 encryption is associated with insignificant computational load and significant enhancement of data confidentiality in clouds. The results of measured encryption and decryption times show that the extra expense has not significantly increased the total file upload and download latency, which is why it is convenient to apply it in real-world scenarios. In addition to confidentiality, the inclusion of blockchain technology plays a major role in enhancing the security model because it provides decentralized integrity checks and policy enforcement. The system provides tamper-evident storage by placing the plaintext hash, ciphertext hash, and file access policy in the blockchain ledger. Any illegal change in the encrypted file stored in the cloud or the metadata causes a hash discrepancy that instantly exposes data integrity breaches. The dual hash scheme offers end-to-end assurance- from the original plaintext to the encrypted cloud storage.

Moreover, access control through blockchain increases transparency and accountability. In contrast to the early days of access management systems that were centralized, blockchain associations implement access policies that cannot be modified without agreement. This eliminates insider manipulation, unauthorized privilege escalation, and post hoc log manipulation. The security incidents are also advantageous to the forensic capability in case of a security incident, since the transactions are also highly auditable. Nonetheless, these security improvements have moderate performance overhead. This extra block creation, computation of hash values, and verification of the chain also mean a slight increase in the overall processing latency. There is also an overhead of storage because of the maintenance of blockchain metadata. However, this overhead can be analyzed experimentally to be within the acceptable bounds of safe cloud applications, especially those that have sensitive or controlled information. The sacrifice between better integrity assurance and a low performance increment is reasonable in an environment where both data security and traceability are important.

The proposed architecture is expected to scale well in multi-user environments, as AES-256 encryption and decryption are done at the client level, bypassing the centralized complexity of processing. Additionally, the blockchain carries only lightweight metadata, like file hashes and access policies, not the real contents of the files, reducing storage and transaction overhead. While concurrent access requests can affect the processing time for transactions in the blockchain, the minimal usage of blockchain operations for metadata management helps to ensure an acceptable performance. Hence, the proposed system is expected to be able to support multiple concurrent users while maintaining confidentiality, integrity, access control, and auditability. Future research will explore the system's behavior in large-scale concurrent workloads and in actual deployments of blockchain systems.

In general, the integrated AES-Blockchain system has an optimal security-performance trade-off. AES guarantees high and effective confidentiality of the data, whereas blockchain guarantees policy enforcement, tamper resistance, and decentralized integrity verification. The findings affirm that the suggested architecture is not only secure but also practically implementable to the current cloud storage systems that need high-assurance levels.

### *Comparative Analysis*

Unlike the current published literature, as presented in Table 4, the proposed system is the first to integrate lightweight AES encryption, dual-hash integrity verification, smart-contract-based access control, extensive performance analysis, and real attack testing, to create a more realistic, reproducible, and experimentally validated secure cloud storage system.

**Table 4:** Comparative Analysis of Existing System vs Proposed System

| Features/System | (Sharma et al., 2021) | (Amanat et al., 2022) | (Zhang et al., 2023) | (Golla Bala and Gnanavel, 2025) | Proposed System |
|---|---|---|---|---|---|
| Architecture | Fully decentralized blockchain network | Blockchain + cloud server | Blockchain + cloud | Blockchain-enabled cloud architecture | Hybrid layered architecture |
| Encryption Method | CP-ABE | SHA-256 + ECDSA | ABE | ChaCha20-Poly1305 | AES-256 |
| Blockchain Role | Access control + integrity | Transaction verification | Data verification | Access control + metadata | Hash storage + smart contract access control |
| Storage Model | Distributed cloud | Cloud server | Cloud + blockchain metadata | Cloud storage | Encrypted files in the cloud |
| Performance Observation | Fully decentralized architecture increases processing overhead | Cryptographic verification increases security overhead | Blockchain verification improves integrity but adds latency | It surpasses AES-GCM and RSA in various performance metrics. | Efficient AES encryption + lightweight blockchain metadata |
| Key Features | Authorization, decentralized storage | Secure medical record sharing | Secure data sharing and verification | Improved throughput and lower latency | Dual-hash integrity + blockchain audit |

## Conclusion

This study presented a blockchain-enabled secure cloud storage system that integrates AES-256 encryption with smart-contract-based access control and dual-hash integrity verification. By encrypting data at the client side and storing only cryptographic hashes and access policies on the blockchain, the proposed system effectively addresses critical cloud security challenges such as unauthorized access, data tampering, and lack of auditability. Unlike existing blockchain–cloud solutions that rely on computationally expensive cryptographic primitives or external decentralized storage platforms, the proposed approach adopts a lightweight and practical design based on symmetric encryption and hash-linked blockchain records. In the proposed system, a dual-hash mechanism has been used to provide a strong integrity verification. The plaintext file is hashed before encryption, and the result is placed in the blockchain to facilitate a reliable integrity check. Moreover, a hash of the encrypted file is saved so as to detect any tampering in the cloud storage. This design not only prevents false integrity failures due to randomized AES encryption but also improves tamper detection. All-inclusive performance evaluation using CloudSim-style metrics validates that the proposed system achieves a favorable balance between security and efficiency. Once the two aspects are compared to each other through a comprehensive performance assessment with the use of CloudSim-style metrics, it has been proven that the proposed system has both a decent security and efficiency balance. Although blockchain integration introduces moderate latency overhead, the measured overhead remains acceptable when compared to the significant gains in confidentiality, integrity, access control, and accountability. Furthermore, real attack simulation experiments confirm the system's robustness against common cloud security threats, with a substantial reduction in attack success rates. Overall, the proposed blockchain-enabled secure cloud storage system demonstrates that strong data confidentiality, integrity, accountability, and access control can be achieved with manageable performance costs.

### *Future Scope*

Future work will be focused on scalability optimization, fine-grained attribute-based access control, and deployment on real-world blockchain platforms to further enhance system applicability.

## Acknowledgment

The authors would like to show their deepest gratitude to the faculty and mentors who assisted them with their precious guidance and technical knowledge during the period of this research work. We also recognize the resources and assistance that the institution gave us, which helped to complete this work successfully. The development and maintenance of cryptographic and blockchain frameworks, which were the basis of the implementation, are credited to the open-source community. Lastly, we like the positive feedback that our colleagues provided in order to make this research better and more understandable.

## Funding Information

No financial support has been received from any organization to carry out this research work.

## Author's Contributions

All the authors have contributed equally to the

research work, along with the preparation of the manuscript.

## Ethics

The article is original and unpublished. The corresponding author ascertains that the manuscript has been read and approved by all the other authors and that there are no ethical concerns involved.

## References

Agarwal, V., Kaushal, A. K., & Chouhan, L. (2020). A Survey on Cloud Computing Security Issues and Cryptographic Techniques. *Social Networking and Computational Intelligence*, *100*, 119–134. https://doi.org/10.1007/978-981-15-2071-6_10

Alex, A. (2025). Network Latency in Cloud Computing Data Centers: Challenges and Innovations. *European Journal of Computer Science and Information Technology*, *13*(12), 106–115. https://doi.org/10.37745/ejcsit.2013/vol13n12106115

Alharby, M. (2024). Preserving Data Secrecy and Integrity for Cloud Storage Using Smart Contracts and Cryptographic Primitives. *Computers, Materials & Continua*, *79*(2), 2449–2463. https://doi.org/10.32604/cmc.2024.050425

Amanat, A., Rizwan, M., Maple, C., Zikria, Y. B., Almadhor, A. S., & Kim, S. W. (2022). Blockchain and cloud computing-based secure electronic healthcare records storage and sharing. *Frontiers in Public Health*, *10*, 938707. https://doi.org/10.3389/fpubh.2022.938707

Darwish, M. A., Yafi, E., Al Ghamdi, M. A., & Almasri, A. (2020). Decentralizing Privacy Implementation at Cloud Storage Using Blockchain-Based Hybrid Algorithm. *Arabian Journal for Science and Engineering*, *45*(4), 3369–3378. https://doi.org/10.1007/s13369-020-04394-w

El Kafhali, S., El Mir, I., & Hanini, M. (2022). Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing. *Archives of Computational Methods in Engineering*, *29*(1), 223–246. https://doi.org/10.1007/s11831-021-09573-y

Gai, K., Guo, J., Zhu, L., & Yu, S. (2020). Blockchain Meets Cloud Computing: A Survey. *IEEE Communications Surveys & Tutorials*, *22*(3), 2009–2030. https://doi.org/10.1109/comst.2020.2989392

Ganesan, S., & Arulkumaran, G. (2018). Blockchain-Enabled IoT-Cloud Storage Security: A Merkle Hash Tree and Cryptographic Hashing Approach "Blockchain-Enabled IoT-Cloud Storage Security: A Merkle Hash Tree and Cryptographic Hashing Approach'''. *Journal of Science and Technology*, *3*(01), 2456–5660.

Golla Bala, R., & Gnanavel, S. (2025). BC2P-1305: an enhanced data security in cloud computing network using blockchain based ChaCha20-Poly1305 cryptography. *Frontiers in Blockchain*, *8*, 1636056. https://doi.org/10.3389/fbloc.2025.1636056

Han, H., Fei, S., Yan, Z., & Zhou, X. (2022). A survey on blockchain-based integrity auditing for cloud data. *Digital Communications and Networks*, *8*(5), 591–603. https://doi.org/10.1016/j.dcan.2022.04.036

Hasan, S. S., Sultan, N. H., & Barbhuiya, F. A. (2019). Cloud Data Provenance using IPFS and Blockchain Technology. *Proceedings of the Seventh International Workshop on Security in Cloud Computing*, 46–53. https://doi.org/10.1145/3327962.3331457

Huang, J., & Yi, J. (2024). The key security management scheme of cloud storage based on blockchain and digital twins. *Journal of Cloud Computing*, *13*(1), 15. https://doi.org/10.1186/s13677-023-00587-4

Khanna, A., Sah, A., Bolshev, V., Burgio, A., Panchenko, V., & Jasiński, M. (2022). Blockchain–Cloud Integration: A Survey. *Sensors*, *22*(14), 5238. https://doi.org/10.3390/s22145238

Li, H., Tian, H., Zhang, F., & He, J. (2019). Blockchain-based searchable symmetric encryption scheme. *Computers & Electrical Engineering*, *73*, 3245. https://doi.org/10.1016/j.compeleceng.2018.10.015

Li, J., Wu, J., Jiang, G., & Srikanthan, T. (2020). Blockchain-based public auditing for big data in cloud storage. *Information Processing & Management*, *57*(6), 102382. https://doi.org/10.1016/j.ipm.2020.102382

Li, W., Wu, J., Cao, J., Chen, N., Zhang, Q., & Buyya, R. (2021). Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions. *Journal of Cloud Computing*, *10*(1), 35. https://doi.org/10.1186/s13677-021-00247-5

Lui, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). *NIST cloud computing referencearchitecture*. https://doi.org/10.6028/nist.sp.500-292

Mareddy, L. (2026). Security and Performance Assessment of Encryption Techniques for Cloud Platforms. *International Journal of Communication Networks and Information Security (IJCNIS)*, *2026*(1), 68–80.

Musa, N. S., Murugan, T., N., N. A., & Mirza, N. M. (2025). Research study on securing the cloud: utilizing conventional and blockchain-based access control mechanisms. *Journal of Cloud Computing*, *14*(1), 88. https://doi.org/10.1186/s13677-025-00803-3

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *Published Independently (Hosted at Bitcoin.Org)*, *27*, 1502.

NIST. (2023). *Advanced Encryption Standard (AES)*. https://doi.org/10.6028/NIST.FIPS.197

Patil, P. (2025). Optimizing low latency public cloud systems: Strategies for network, compute and storage efficiency. *World Journal of Advanced Research and Reviews*, *26*(1), 4003–4021. https://doi.org/10.30574/wjarr.2025.26.1.1538

Punia, A., Gulia, P., Gill, N. S., Ibeke, E., Iwendi, C., & Shukla, P. K. (2024). A systematic review on blockchain-based access control systems in cloud environment. *Journal of Cloud Computing*, *13*(1), 146. https://doi.org/10.1186/s13677-024-00697-7

Rehman, S., Talat Bajwa, N., Shah, M. A., Aseeri, A. O., & Anjum, A. (2021). Hybrid AES-ECC Model for the Security of Data over Cloud Storage. *Electronics*, *10*(21), 2673. https://doi.org/10.3390/electronics10212673

Sharma, P., Jindal, R., & Borah, M. D. (2020). Blockchain Technology for Cloud Storage. *ACM Computing Surveys*, *53*(4), 1–32. https://doi.org/10.1145/3403954

Sharma, P., Jindal, R., & Borah, M. D. (2021). Blockchain-based decentralized architecture for cloud storage system. *Journal of Information Security and Applications*, *62*, 102970. https://doi.org/10.1016/j.jisa.2021.102970

Sultana, S. A., Rupa, C., Malleswari, R. P., & Gadekallu, T. R. (2023). IPFS-Blockchain Smart Contracts Based Conceptual Framework to Reduce Certificate Frauds in the Academic Field. *Information*, *14*(8), 446. https://doi.org/10.3390/info14080446

Sun, P. (2020). Security and privacy protection in cloud computing: Discussions and challenges. *Journal of Network and Computer Applications*, *160*, 102642. https://doi.org/10.1016/j.jnca.2020.102642

Tabrizchi, H., & Rafsanjani, M. K. (2020). A survey on security challenges in cloud computing: issues, threats, and solutions. *The Journal of Supercomputing*, *76*(12), 9493–9532. https://doi.org/10.1007/s11227-020-03213-1

Wang, W., Ma, J., Zhao, B., Han, P., & Zeng, P. (2025). Information security transmission method of data exchange platform based on lightweight AES encryption algorithm. *Egyptian Informatics Journal*, *32*, 100831. https://doi.org/10.1016/j.eij.2025.100831

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). *Blockchain technology overview*. https://doi.org/10.6028/nist.ir.8202

Yan, L., Ge, L., Wang, Z., Zhang, G., Xu, J., & Hu, Z. (2023). Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. *Journal of Cloud Computing*, *12*(1), 61. https://doi.org/10.1186/s13677-023-00444-4

Yang, P., Xiong, N., & Ren, J. (2020). Data Security and Privacy Protection for Cloud Storage: A Survey. *IEEE Access*, *8*, 131723–131740. https://doi.org/10.1109/access.2020.3009876

Zhang, R., Xue, R., & Liu, L. (2018). Searchable Encryption for Healthcare Clouds: A Survey. *IEEE Transactions on Services Computing*, *11*(6), 978–996. https://doi.org/10.1109/tsc.2017.2762296

Zhang, T., Wang, C., & Chandrasena, M. I. U. (2023). Blockchain-assisted data sharing supports deduplication for cloud storage. *Connection Science*, *35*(1), 2174081. https://doi.org/10.1080/09540091.2023.2174081

Zhang, Y., Deng, R. H., Xu, S., Sun, J., Li, Q., & Zheng, D. (2020). Attribute-based Encryption for Cloud Computing Access Control: A Survey. *ACM Computing Surveys*, *53*(4), 1–41. https://doi.org/10.1145/3398036

Zou, J., He, D., Zeadally, S., Kumar, N., Wang, H., & Choo, K. R. (2021). Integrated Blockchain and Cloud Computing Systems: A Systematic Survey, Solutions, and Challenges. *ACM Computing Surveys*, *54*(8), 1–36. https://doi.org/10.1145/3456628