A Secure Integrated Cloud Storage Framework Using Lorenz Chaotic Key Generation, AES-256-GCM, and PBFT-Based Blockchain Verification

Walde Rajesh Baliram[1*], Bashir Alam[2], Mohammad Najmud Doja[3]
Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India[1, 2]
Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India (Retired)[3]

***Abstract*—Cloud storage security remains a major challenge owing to threats related to confidentiality, integrity, and unauthorized access. In this study, a novel secure cloud storage system is proposed by integrating Lorenz 3D chaotic key generation, AES-256-GCM authenticated encryption, and blockchain verification based on PBFT. High-entropy keys were generated using the Lorenz chaotic system and then passed through SHA-256 to create secure 256-bit AES keys. The data is encrypted using the AES-256-GCM algorithm and stored in the cloud, and integrity metadata is stored in a blockchain that offers tamper detection, auditability, and Byzantine fault tolerance. The experimental results demonstrate strong cryptographic performance with near-ideal entropy (7.99), a high avalanche effect (50.05%), successful NIST randomness validation, and low blockchain overhead. The results of the security analysis prove resistance against brute force, statistical, differential, replay, known plaintext, chosen ciphertext, and Byzantine attacks. The proposed framework offers a secure, efficient, and reliable multilayered security solution for cloud storage systems.**

***Keywords*—Cloud security; AES-256-GCM; Lorenz chaotic system; blockchain; PBFT; data integrity; secure cloud storage; cryptography**

I. Introduction

Cloud computing has revolutionized the modern era of data management in terms of scalability, flexibility, and cost efficiency [1]. Cloud computing infrastructure is the foundation of the new economy, where companies rely on it to store sensitive data, such as financial records, healthcare information, and proprietary research data [2]. But when it comes to outsourcing data to external cloud service providers, there are significant security concerns, such as unauthorized access, data breaches, exposure of sensitive data, and privacy invasion [3].

The Advanced Encryption Standard (AES) is another symmetric encryption algorithm frequently employed to reduce the potential threat to confidentiality, as it offers high security assurance and a low computational load [4]. AES-256, especially, offers resistance to known cryptanalytic attacks, as well as brute force attacks under normal security conditions [5]. However, for centralized cloud storage systems, data encryption does not provide any data integrity checking or insider interference protection [6]. In addition, traditional cloud architecture has drawbacks, such as a lack of transparency, insufficient decentralization, and single points of failure, impacting trust in cloud-based systems [7].

The generation of keys is important in cryptography and is a critical component of security. The strongest encryption is not strong enough if the keys are predictable or not sufficiently random. In chaos-based cryptography, the high-entropy keys are generated using nonlinear dynamic systems that exhibit extreme sensitivity to initial conditions and have strong diffusion properties [8]. Chaos-based key generation with traditional encryption schemes has been proven to be a more random and efficient method [9]. The Lorenz 3D chaotic system is a complex nonlinear system that is highly sensitive to initial conditions and exhibits randomness, making it suitable for cryptographic applications. This makes it significantly more secure than low-dimensional chaotic systems and highly compatible with the latest encryption algorithms, including AES-256.

It is viewed as a decentralized solution to overcome trust and integrity problems in distributed systems using blockchain technology. The mechanisms of consensus, cryptographic hash linkage, and permanent ledger systems that blockchain offers can be used to store and record transactions in a verifiable manner without the possibility of being tampered with [10].

Recent studies have discussed the use of blockchain and cloud computing for access control auditing, logging security, and data integrity [11]. To resolve the problems of scalability in blockchain, researchers have suggested hybrid architectures, integration of cloud storage, and validation layers based on blockchain. In these systems, encrypted documents are stored in the cloud, and cryptographic hash values and transaction metadata are recorded on-chain.

Although cloud security has advanced, some areas of research could still be pursued. Existing AES-based cloud services offer confidentiality, but not decentralized integrity verification. Cloud-based frameworks using blockchain tend to lack cryptographic randomness enhancement, but are nonetheless tamper-proof. Cloud frameworks based on blockchain are not only tamper-proof, but also have no improvement in this regard with respect to cryptographic randomness. Chaos-based encryption methods provide more unpredictability but are not generally used in combination with distributed integrity validation mechanisms. Many existing systems lack formalized threat modelling or comparative security-performance evaluation. Limited research has been

*Corresponding author

conducted on incorporating authenticated encryption, chaotic key generation, and blockchain with PBFT within a single architecture.

The following contributions were made to overcome these limitations:

- Design and development of a secure cloud storage system with multi-level security, application of chaos-based cryptography, and verification by blockchain.
- Design of a Lorenz 3D chaotic key generation mechanism strengthened by the SHA-256 hash function.
- Incorporation of AES-256-GCM authenticated encryption with confidentiality and integrity.
- Implementation of a dual-hash integrity verification system for ciphertext and plaintext checks.
- Adoption of the PBFT consensus mechanism for Byzantine fault tolerance in blockchains.
- Complete experimental evaluation, including NIST randomness tests, entropy, avalanche effect, throughput, and latency measurements.
- Comparative analysis of existing AES-based and Chaos-based cloud security models.

The novelty of this work is the unified integration of the following four concepts uniformly: chaos theory-based key generation, authenticated encryption, dual hash integrity checking, and PBFT-based blockchain consensus to ensure secure cloud storage.

The structure of the rest of this study is as follows: A literature review of cloud security, chaos-based cryptography, and blockchain-based integrity verification is provided in Section II. In Section III, the proposed architecture for secure cloud storage is introduced. The methodology and algorithms are described in Section IV. Details of the experimental environment and setup are given in Section V. The experimental results and performance analyses are presented in Section VI. The threat model is introduced in Section VII, and a detailed security analysis is presented in Section VIII. A formal security analysis of the proposed framework is presented in Section IX. The comparison of the proposed approach with existing studies is made in Section X. The results and implications are discussed in Section XI. Finally, the conclusions and future research directions are presented in Section XII and Section XIII, respectively.

II. Literature Review

Data security remains a significant concern after cloud computing has transformed computing and storage services in recent years. Despite the use of encryption to reduce the risks of confidentiality, preliminary studies have pointed out that there are still some shortcomings, such as multi-tenancy, data exposure, and even user control over external resources [3], [12], [13]. Insider threats, improper access control, and post-storage tampering cannot be mitigated through encryption [14].

Symmetric encryption, particularly AES, is a fundamental aspect of secure cloud storage; however, its shortcomings in key generation and integrity checks have led to interest in other options. Because of its efficiency and robust cryptographic guarantees, AES is still the most popular symmetric encryption algorithm in cloud security architectures [15].

AES-256 is designed to be secure with respect to known cryptanalytic attacks and brute force attacks known to the designer, based on the known security assumptions. The recent research has also applied AES in the storage and transmission of information in cloud storage systems to protect the confidentiality of information [16]. AES is confidential but not necessarily decentralized, ensuring integrity verification and protection against unauthorized modifications in cloud storage systems.

Most existing schemes are built on traditional key generation schemes, which are susceptible to being compromised by insufficient entropy sources. The chaos-based cryptography utilizes a nonlinear dynamical system to improve the randomness and unpredictability of the key generation process [17]. The chaotic maps, such as logistic maps, are very sensitive to initial conditions and have positive Lyapunov exponents, matching the cryptographic randomness conditions [8], [18]. Research has shown that keys generated by the chaos technique are robustly diffused and confused, making them suitable for use to supplement the symmetric encryption technique [19]. Recently, studies have been carried out on using chaos-based methods to enhance unpredictability and protection against brute force attacks on cryptographic systems [20]. However, few studies have considered both chaos-based key generation and integrity models for blockchain-based cloud systems.

Blockchain technology has been called a new technology with the potential to be used to protect the integrity of data and distributed trust. The novel idea of decentralized consensus and hash-linked blocks to prevent tampering was introduced by Nakamoto [10]. Additional studies have led to the development of blockchain models and consensus algorithms for securing decentralized applications.

Alghuried and Wenhua provided a comprehensive overview of blockchain security concerns and privacy implications [21], [22]. The immutability of blockchain, its consensus-driven validation, and the decentralized system structure make it appropriate for cloud system integrity (Zheng et al., 2017). Even with these enhancements, several blockchain-based systems add performance overheads from the consensus protocol or complicated cryptographic primitives.

Some studies have focused on how to use the blockchain and cloud storage to enhance auditability and tamper resistance [16]. To improve scalability, blockchain-based cloud systems store large-scale, encrypted data off-chain and maintain blockchain metadata on-chain within the blockchain [11], [23]. However, blockchain integration with cloud services still faces many challenges related to performance overhead, transaction latency, and security concerns [24]. This has made lightweight blockchain architectures and hybrid storage solutions interesting potential solutions. However, large amount of data cannot be directly stored on the blockchain network due to

scalability and delay latency issues, and storage issues [25]. For this reason, more scalable solutions have been developed, which are hybrid solutions storing encrypted data off-chain with integrity metadata on-chain [26].

The application of chaos theory, authenticated encryption, blockchain technology, and distributed consensus has been investigated in recent studies to enhance cloud security. Rahman et al. proposed a chaos-enhanced AES key generation mechanism, which showed enhanced key randomness and unpredictability in an IoT environment [9]. Alatawi proposed a novel approach to decentralized authentication and authorization using smart contracts that run on a blockchain in cloud security [27]. In addition, Yao et al. and Liu et al. proposed optimal PBFT consensus mechanisms to enhance the throughput, latency, and Byzantine fault tolerance in the consortium blockchain system [28], [29]. Although these advancements reflect a considerable leap forward in secure cloud storage architectures, most current solutions focus on encryption, access control, or blockchain verification individually.

The proposed framework is distinct because of the combination of Lorenz 3D chaotic key generation and AES-256-GCM authenticated encryption, PBFT-based blockchain verification, and dual-hash integrity validation in a single cloud security framework.

III. Proposed System Architecture

The proposed system architecture is divided into six interoperable layers, as shown in Fig. 1. The proposed architecture is a multi-layered security architecture that combines symmetric encryption (AES-256-GCM), Lorenz 3D chaotic key generation, cloud storage, blockchain-based integrity verification using PBFT consensus, and a smart contract. These layers serve different functions to ensure the confidentiality, integrity, authentication, and traceability of the data. This multilayered design ensures a clear separation of concerns and separates cryptographic functions from blockchain consensus and the application logic. This modularity allows systems to be more resilient in the event of a compromise in one layer and to maintain a secure flow of data within and between the components. The layers are given below.

A. Application Layer

The Application Layer is the main layer where users interact with the proposed secure cloud storage system. It handles user authentication, access control, upload/download of files from/to the blockchain, and blockchain authorization. When a file is uploaded, the layer generates a Lorenz 3D chaotic key and applies AES-256-GCM authenticated encryption to protect the confidentiality and integrity of the data. The ciphertext is encrypted and stored in the cloud, with file hashes and transaction metadata stored on the blockchain, which is maintained on the PBFT protocol for integrity checks and auditing purposes. During the retrieval phase, AES-GCM authentication tag verification and blockchain validation are performed prior to decryption, which is robust enough to thwart attempts at unauthorized access, insider attacks, replay attacks, and data tampering.

B. Key Generation Layer

The Key Generation Layer is developed to produce a safe and high-entropy key of 256 bits from the Lorenz 3D chaotic system to a cryptographic hashing function called SHA-256. This blended approach yields the unpredictability and cryptographic strength of the generated keys. A highly sensitive, nonlinear sequence is generated by the Lorenz 3D chaotic system and quantized to be fed into SHA-256 to generate a secure, uniformly distributed AES-256 key for encrypting the vector.

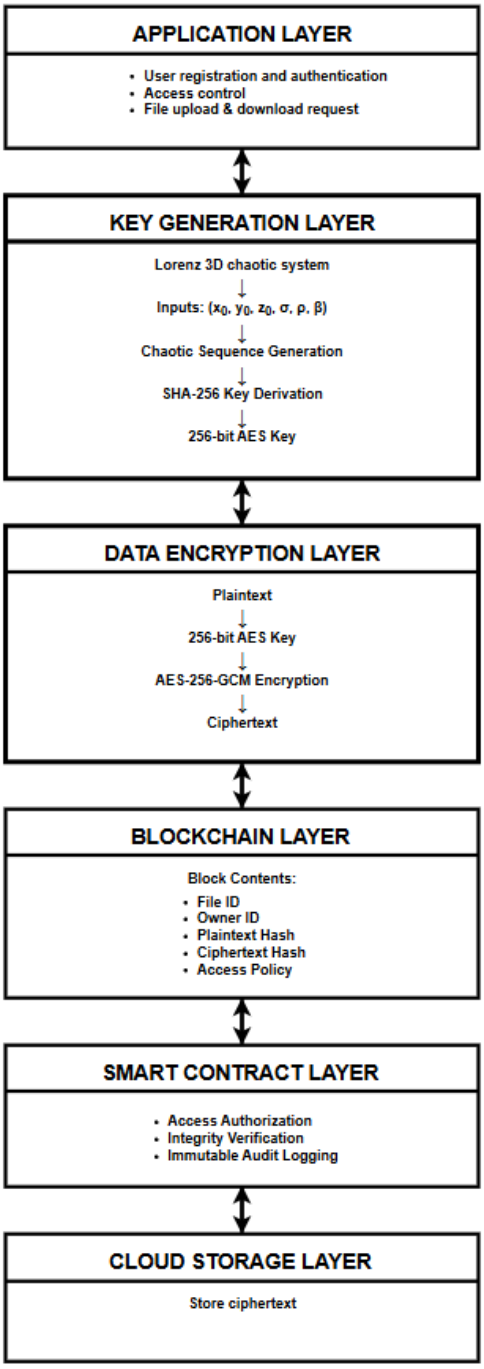

Fig. 1. System architecture diagram

C. Encryption Layer

The main security layer in the proposed framework that offers data confidentiality, integrity, and cryptographic randomness is the Encryption Layer. The chaotic Lorenz system is employed to produce a 256-bit chaotic key that is highly random, and then this 256-bit chaotic key is utilized for authenticated encryption with the AES-256-GCM. AES-GCM provides a ciphertext with a high degree of randomness and resistance to differential and statistical cryptanalysis. AES-GCM was created by AES to prevent modification or replay attacks on the ciphertext and ensure that the data remains intact. The integrity metadata and hash value of the encrypted ciphertext are recorded on the PBFT-based blockchain ledger for security checks and auditability, and the ciphertext is uploaded to the cloud server.

D. Cloud Storage Layer

Only encrypted data files are stored in the Cloud Storage Layer; no plaintext data is stored on the cloud servers.

E. Blockchain with PBFT Consensus Layer

Blockchain with the PBFT Consensus Layer provides auditability, integrity verification, and tamper-proof logging. The system does not store the entire file on-chain; instead, it stores the following:

- SHA-256 hash of the encrypted file
- SHA-256 hash of plaintext file
- Timestamp
- Previous block hash
- Block index

The blockchain layer offers security characteristics, such as immutability, tamper detection, audit trails, and decentralized verification. If any change is made to the stored file, the recalculated hash will not match the blockchain record, which will be the first sign of manipulation.

F. Smart Contract Layer

The Smart Contract Layer specifies and implements access control policies. Smart contracts, which are built on blockchains, offer complete control over access to cloud data. Downloading the resource is only possible after a successful on-chain authorization of a download request to prevent unauthorized access, even by privileged insiders of the cloud. Blockchains' smart contracts ensure data integrity, enable decentralized access control, store immutable hash, and deliver transparent auditability. Both serve crucial functions in enhancing the privacy and security of cloud storage.

IV. METHODOLOGY

A. System Workflow

The workflow of the proposed secure cloud storage system is defined by the entire cycle of data, from uploading to access and verification (Fig. 2). This system consists of client-side encryption, decentralized access control, blockchain-based integrity verification, and cloud storage to ensure confidentiality, integrity, and accountability.

Step 1: User Registration. Users and owners of the data are registered into the system with unique identifiers. Blockchain credentials are issued to each user to safely interact with smart contracts.

Step 2: AES key Generation. The AES key is generated based on a highly sensitive and nonlinear sequence generated by the Lorenz 3D chaotic system. Then, the SHA-256 hashing algorithm was used to create a 256-bit encryption key.

Step 3: File Encryption. The data owner selects a file to be stored in the cloud. The data is encrypted on the client using AES-256-GCM encryption, which ensures confidential data storage.

Step 4: Secure File Upload. The encrypted file is then uploaded to the cloud storage. The access permissions of the files are defined by the data owner before they are uploaded.

Step 5: Metadata Storage in Blockchain. A cryptographic hash of the plaintext file and a cryptographic hash of the encrypted file are generated using a hashing algorithm called SHA-256 to help verify the integrity of the file. The file hash and file ownership information were recorded on the blockchain. The access policy and file identifier, along with a hash, are stored and verified on the blockchain via a smart contract.

Step 6: Access Request by User. Once the user requests access to a stored file, it is sent to the blockchain layer. The smart contract then receives the access control policy and proceeds to determine whether the requesting user has access to that file.

Step 7: Decentralized Access Control Decision. Access is granted or denied based on smart contract verification. The information requested, and the decision made is logged on to the blockchain as an audit trail which is auditable and non-repudiable.

Step 8: Secure File Download. Once authenticated, encrypted files are downloaded from the cloud storage and sent to the authorized end user. The cloud service provider does not know the content of the files and does not control access to the files.

Step 9: File Tamper Detection. It computes the SHA-256 hash of the downloaded encrypted file prior to decryption. This hash is compared with the original hash that was added to the blockchain. If any discrepancy was found, it meant that the data or the process were corrupted, and the process was killed.

Step 10: File Decryption. If tamper detection test is passed successfully, the encrypted file is decrypted at the client side using the AES key to reconstruct the plain text file.

Step 11: File Integrity Verification. SHA-256 computes the hash of the decrypted file in the system. This hash is compared with the hash of the original plaintext file stored in the blockchain. If there is any discrepancy, it means that there is some corruption or tampering with the data, and the process is killed.

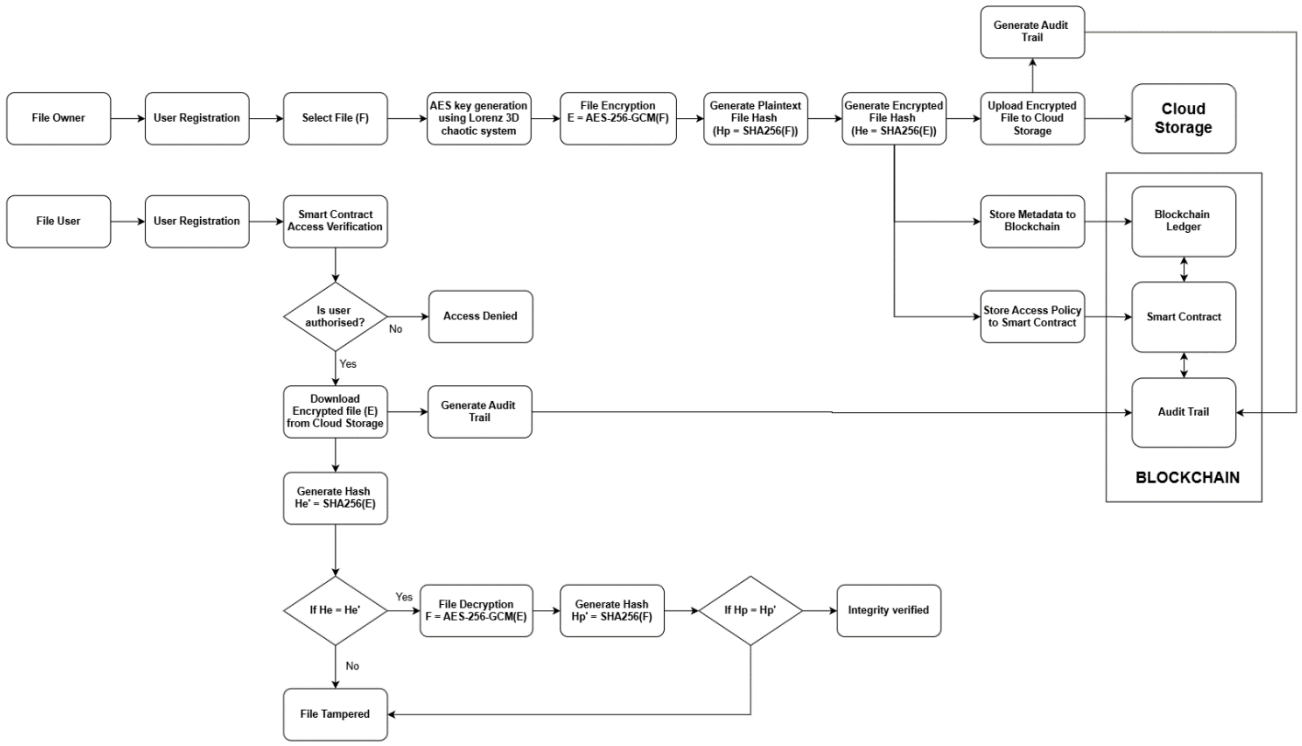

Fig. 2. System workflow diagram

B. Algorithms

To implement the proposed secure cloud storage framework, five algorithms are used to deliver confidentiality, integrity, authentication, and fault tolerance. Algorithm 1 produces a cryptographic key with high entropy by applying the Lorenz 3D chaotic system. Algorithm 2 is used to perform AES-256-GCM authenticated encryption, which provides data confidentiality and integrity. Algorithm 3 securely uploads encrypted files onto cloud storage and records integrity metadata on the blockchain. Algorithm 4 uses the Practical Byzantine Fault Tolerance (PBFT) consensus protocol for blockchain network validation and committing metadata transactions. Lastly, Algorithm 5 retrieves the encrypted files from the cloud storage and checks for integrity on both the encrypted and decrypted files based on blockchain-stored metadata. These algorithms create a robust, transparent, and unalterable cloud storage system. For clarity and ease of understanding, the notations and abbreviations used in the proposed algorithms, along with their descriptions, are provided in Table IX at the end of the study.

Algorithm 1: Key Generation Using Lorenz 3D Chaotic System

The purpose of this algorithm is to generate an AES key using the Lorenz 3D chaotic system [30].

Input:

x0, y0, z0: Initial conditions

σ, ρ, β: Lorenz control parameters

h: Step size

N: Number of iterations

T: Number of transient samples to discard

L: Required key length (256 bits)

Output:

K: 256-bit chaotic key

Begin

1. x ← x0
2. y ← y0
3. z ← z0
4. S ← EmptySequence
5. for i ← 1 to N do
6. x ← x + h × σ × (y - x)
7. y ← y + h × (x × (ρ - z) - y)
8. z ← z + h × (x × y - β × z)
9. Append(S, x)
10. end for
11. S ← RemoveFirstTElements(S, T)
12. B ← EmptyBitstream
13. for each value s in S do
14. x_norm ← |s| mod 1
15. if x_norm > 0.5 then
16. bit ← 1
17. else
18. bit ← 0
19. end if

```
20.     Append(B, bit)
21.     if Length(B) ≥ L then
22.         break
23.     end if
24. end for
25. B ← FirstLBits(B)
26. K ← ConvertBitsToBytes(B)
27. return K
End
```

Algorithm 2: AES-256-GCM Authenticated Encryption

The purpose of this algorithm is to securely encrypt user data using the AES-256-GCM authenticated encryption scheme specified in the Advanced Encryption Standard (AES) [4] and Galois/Counter Mode (GCM) [31].

```
Input:
  P: Plaintext divided into 128-bit blocks {P1, P2, ..., Pm}
  K_AES: 256-bit AES encryption key
  IV: 96-bit initialization vector or nonce
  AAD: Associated authenticated data
Output:
  C: Ciphertext
  T: 128-bit authentication tag
Begin
1. RoundKeys ← KeyExpansion(K_AES)
2. J0 ← IV || 0^31 || 1
3. C ← EmptyCiphertext
4. for i ← 1 to m do
5.     Ji ← inc32(Ji-1)
6.     Si ← AES_Encrypt(K_AES, Ji)
7.     Ci ← Pi ⊕ Si
8.     Append(C, Ci)
9. end for
10. H ← AES_Encrypt(K_AES, 0^128)
11. Y ← AAD || C || len(AAD) || len(C)
12. Divide Y into 128-bit blocks {Y1,Y2,…, Yn}
13. X0 ← 0^128
14. for j ← 1 to n do
15.     Xj ← (Xj-1 ⊕ Yj) · H over GF(2^128)
16. end for
17. G ← Xn
18. T ← AES_Encrypt(K_AES, J0) ⊕ G
19. return (C, T)
End
```

Algorithm 3: Secure File Upload

The purpose of this algorithm is to encrypt user data safely with the help of a chaos-based AES-256 key, save the ciphertext in the cloud, and store metadata about integrity in the blockchain for tamper-proof verification.

```
Input:
  F: Plaintext file
  UserCred: User authentication credentials
  x0, y0, z0: Initial conditions of Lorenz
3D chaotic system
  AAD: Associated authenticated data
Output:
  UploadStatus
Begin
1. if Authenticate(UserCred) = FALSE then
2.     return "Access Denied"
3. end if
4. K_chaos ← Lorenz3D_KeyGeneration(x0,y0,z0)
5. K_AES ← SHA256(K_chaos)
6. IV ← CSPRNG_Generate_Unique_Nonce(96 bits)
7. EncParams ← Encrypt({x0,y0,z0}, MasterKey)
8. StoreKeyMetadata(FileID, EncParams)
9. (C,T) ← AES256_GCM_Encrypt(F,K_AES,IV, AAD)
10. H_plain ← SHA256(F)
11. H_cipher ← SHA256(C)
12. StoreCloud(C, T, IV)
13. Metadata ← {
      FileID,
      H_plain,
      H_cipher,
      Timestamp
    }
14. Tx ← CreateBlockchainTransaction(Metadata)
15. ExecuteSmartContract(Tx)
16. UploadStatus ← StoreBlockchainRecord(Tx)
17. return UploadStatus
End
```

1) Secure chaotic parameter management: The initial conditions (x_0, y_0, z_0) of the Lorenz system are considered as confidential cryptographic parameters. These parameters are not stored in plain text in the cloud storage layer or blockchain ledger to prevent disclosure. Rather, they are encrypted with a master key specific to the user and stored in a secure key-management module. An encrypted parameter record is linked

with the appropriate file identifier during file upload. If the user is authenticated successfully during the retrieval of the file, the encrypted parameters are recovered and decrypted securely in local mode, which helps to regenerate the chaotic key sequence and derive the AES-256 encryption key. As a result, the confidentiality of the chaotic parameters is maintained, and the key is regenerated reliably by authorized users.

2) Nonce generation and management: A unique 96-bit nonce (IV) is generated for each file upload using a cryptographically secure random number generator (CSPRNG) to prevent nonce reuse in AES-256-GCM. This unique nonce is tied only to the encryption session in which it is used, and is not repeated when using the same AES key. The nonce is saved along with the ciphertext and authentication tag in the cloud storage and recalled at the decryption phase. The probability of nonce collision is negligible due to the following reasons: new AES key is generated each time the uploaded file is uploaded, and fresh nonce is used every time the data is encrypted. With this nonce management strategy, AES-GCM assures confidentiality and authentication while mitigating on nonce reuse.

Algorithm 4: PBFT-Based Blockchain Metadata Verification

The purpose of this algorithm is to validate and commit blockchain transactions with file integrity metadata using Practical Byzantine Fault Tolerance. The algorithm verifies that only valid ciphertext hashes, plaintext hashes, timestamps, and audit records are accepted after the validators in the consortium reach an agreement.

Input:
  Tx: File metadata transaction
  H_cipher: Ciphertext hash
  H_plain: Plaintext hash
  T: Timestamp
  FileID: File identifier
  V: Validator node set {v1, v2, ..., vn}
Output:
  B_i: Committed blockchain block
  ConsensusStatus: Accepted or Rejected
Begin
1. Tx ← {
    FileID,
    H_cipher,
    H_plain,
    T,
    UserID
  }
2. SignTransaction(Tx)
3. PrimaryNode ← SelectPrimary(V)
4. if VerifyTransaction(Tx) = FALSE then
5. return Rejected
6. end if
7. B_i ← Hash(B_(i-1) || Tx || T)
8. Broadcast(PRE_PREPARE, v, s, Digest(Tx))
9. for each validator v_i ∈ V do
10. if ValidatePrimaryNode() = TRUE
    and VerifyDigest(Tx) = TRUE
    and VerifySequenceNumber(s) = TRUE
    and VrifyPreviousBlockHash() = TRUE
    and VerifyTransaction(Tx) = TRUE then
11. Broadcast(PREPARE,v,s,Digest(Tx),NodeID)
12. end if
13. end for
14. if PrepareCount ≥ 2f then
15. Broadcast(COMMIT,v,s,Digest(Tx),NodeID)
16. end if
17. if CommitCount ≥ 2f + 1 then
18. Blockchain ← Blockchain ∪ B_i
19. ConsensusStatus ← Accepted
20. else
21. ConsensusStatus ← Rejected
22. RaiseAlert("Consensus Failure")
23. end if
24. return (B_i, ConsensusStatus)
End

Algorithm 5: File Retrieval and Verification

The purpose of this algorithm is to extract encrypted data from cloud storage and check the integrity of these data before and after decryption using the blockchain metadata.

Input:
  FileID: Unique file identifier
  UserCred: User authentication credentials
  K_AES: AES-256 decryption key
  AAD: Associated authenticated data
Output:
  F: Plaintext file
  RetrievalStatus: Verified or Rejected
Begin
1. if Authenticate(UserCred) = FALSE then

```
2.      return "Access Denied"
3. end if
4. EncParams ← RetrieveKeyMetadata(FileID)
5. (x0,y0,z0) ← Decrypt(EncParams, MasterKey)
6. K_chaos ← Lorenz3D_KeyGeneration(x0,y0,z0)
7. K_AES ← SHA256(K_chaos)
8. (C, T, IV) ← RetrieveCloud(FileID)
9. H_plain_ch,H_cipher_ch) ← QueryBlockchain(FileID)
10. H_cipher_local ← SHA256(C)
11. if H_cipher_local ≠ H_cipher_ch then
12.    RaiseAlert("Ciphertext Tampering Detected")
13.    RetrievalStatus ← Rejected
14.   Abort
15. end if
16. F ← AES256_GCM_Decrypt(C,K_AES,IV,AAD,T)
17. if DecryptionStatus = AuthenticationFailed then
18.     RaiseAlert("AES-GCM Authentication Failed")
19.     RetrievalStatus ← Rejected
20.     Abort
21. end if
22. H_plain_local ← SHA256(F)
23. if H_plain_local ≠ H_plain_chain then
24.    RaiseAlert("Plaintext Integrity Violation")
25.    RetrievalStatus ← Rejected
26.     Abort
27. end if
28. RetrievalStatus ← Verified
29. return (F, RetrievalStatus)
End
```

C. Secure Chaotic Parameter Management

The initial conditions of the Lorenz system (x_0, y_0, z_0) are treated as confidential cryptographic parameters. To prevent disclosure, these parameters are not stored in plaintext in the cloud storage layer or blockchain ledger. Instead, they are encrypted using a user-specific master key and stored within a secure key-management module. During file upload, the encrypted parameter record is associated with the corresponding file identifier. Upon successful user authentication during file retrieval, the encrypted parameters are securely recovered and decrypted locally to regenerate the chaotic key sequence and to derive the AES encryption key. Consequently, the confidentiality of the chaotic parameters is preserved while ensuring reliable key regeneration for the authorized users.

V. Experimental Setup

The following configuration was used to build a prototype system of the hybrid cloud storage solution and evaluate it in a sound-testing environment.

A. Software Environment

The software stack was selected for its cryptographic efficiency and ability to handle complex chaotic-map computations with high precision.

- Programming Language: Python 3.x was used because of its flexibility and extensive support for numerical and cryptographic libraries.
- Cryptographic Library: PyCryptodome was employed specifically for its AES-256-GCM implementation, which provides data confidentiality and authentication features.
- Hash Function: SHA-256 (via the hashlib library) was used to generate tamper-proof file digests and to distil the Lorenz chaotic state variables into fixed-length 256-bit keys.
- Blockchain Model: To enable secure logging of file metadata and verification hashes, a custom implementation of a blockchain using Python was developed for prototyping purposes. This implementation aims to demonstrate the PBFT consensus mechanism for verifying integrity in a controlled environment.
- Blockchain Consensus Method: Practical Byzantine Fault Tolerance was added to ensure that transactions are final and that the ledger is consistent across all the distributed nodes.
- Operating System: Windows 11.
- Integrated Development Environment: PyCharm 2025.2.1.1.

B. Hardware Configuration

The following hardware was used to obtain consistent performance in the execution of the chaotic integration and encryption algorithms:

- Processor: Intel(R) Core i7 processor
- RAM: 16 GB
- Storage: SSD-based storage to reduce the latency of I/O operations for large datasets.

C. PBFT Implementation

The proposed architecture is based on a custom consortium blockchain implementation with Python for experimentation to demonstrate integrity verification using PBFT in a secure cloud storage architecture. Implementation attempts to analyze the feasibility and performance influence of integrating the PBFT consensus into AES-256-GCM encryption and Lorenz chaotic key generation.

The experiment was conducted in a controlled local distributed environment with one client node, one primary (leader) validator node, and three replica validator nodes, totaling four PBFT validator nodes. The PBFT protocol is executed in the usual way: first, a pre-prepare phase, followed by a prepare phase, and finally a commit phase, to ensure that blockchain transactions that include ciphertext hashes, integrity metadata, timestamps, and audit records are validated. A transaction is committed if it is approved by $(2f + 1)$ validator nodes, with at most ($f \leq \frac{n-1}{3}$) Byzantine nodes.

This consortium blockchain design assumes that the validator nodes are maintained by various participating organizations or entities, including cloud administrators, organizational clients, and audit authorities, rather than a single centralized cloud provider. This distributed trust model can achieve Byzantine fault tolerance and prevent malicious insiders from unilaterally modifying the integrity of the metadata.

The locally distributed implementation was intentionally designed as a lightweight prototype to test the latency of the blockchain, the validation of transactions, the feasibility of the consensus, and the performance of the integration with the proposed encryption framework. Therefore, the reported blockchain latency values are experimental and proof-of-concept results, conducted on a small scale in a controlled test environment, and not in a large-scale deployment with a cloud-based environment.

Currently, the proposed framework is not designed to ensure the large-scale scalability of the proposed PBFT protocol. In the cloud-scale scenario, the number of validator nodes could increase the network overhead of the practical communication complexity of PBFT as ($O(n^2)$). However, for a typical consortium-based enterprise cloud environment, the number of trusted validator nodes is relatively small, and in practical circumstances, integrity verification and auditability applications are suitable for the PBFT.

D. Test Dataset

The framework was tested using a multi-format dataset to assess its performance at different levels of data redundancy and correlation. Experiments were carried out with file sizes ranging from 10 to 50 MB, such as:

- Standard text files
- High-correlation image datasets
- Unstructured binary files
- Complex PDF documents

All tests were conducted under controlled laboratory conditions to obtain consistent results. The main points of the experimental design were reproducibility and complete security validation. Each experiment was performed several times, and the average of the performance metrics was calculated to obtain the final performance metrics to maintain the statistical validity of the experiments and reduce the influence of random errors or outlier values.

VI. Experimental Results

A. Performance Evaluation of Metrics

The metrics used to evaluate the framework in this section are encryption, decryption, upload, download, blockchain consensus latency, and overall throughput.

1) Measurement methodology: The experimental tests were conducted on a locally distributed setup with file sizes ranging from 10 to 50 MB, and without using external networks to avoid the influence of network variations. All time-related measurements were reported in seconds (s).

a) Latency Calculation (T_{total}): The total processing time for a single file transaction is defined as the sum of its independent computational stages:

$$T_{total} = T_{encryption} + T_{upload} + T_{blockchain}$$

where,

- $T_{encryption}$: is the duration required for the Lorenz 3D chaotic key generation, SHA-256 distillation, and the AES-256-GCM encryption process.
- T_{upload}: is the time taken to transfer the 128-bit blocks of ciphertext to the hybrid cloud storage module.
- $T_{blockchain}$: is the time required for the PBFT consensus protocol to validate the metadata and commit the SHA-256 integrity hashes to the distributed ledger.

b) Throughput Calculation ($Throughput$): Throughput is defined as the number of file transactions completed per unit of time and is calculated as:

$$Throughput = \frac{Number\ of\ Files}{Total\ Time}$$

where, $N = 1$ representing a single file of a specific size processed from initialization to blockchain commitment.

2) Performance analysis: A detailed performance analysis of the proposed system was performed and executed for various file sizes ranging from 10 to 50MB. The experimental results for the different performance metrics are summarized in Table I and presented graphically in Fig. 3.

TABLE I. Performance Evaluation of Metrics

Metric	10MB	20MB	30MB	40MB	50MB
Chaotic Latency (Sec)	0.07	0.08	0.07	0.08	0.07
Encryption Latency (Sec)	0.05	0.08	0.13	0.18	0.23
Upload Latency (Sec)	0.08	0.12	0.18	0.22	0.27
Blockchain Latency (Sec)	0.01	0.01	0.01	0.01	0.01
Total Latency (Sec)	0.20	0.28	0.39	0.49	0.58
Throughput (Files/Sec)	4.95	3.59	2.59	2.05	1.74

As shown in Table I and Fig. 3, the proposed framework can maintain its performance when the file size increases from 10 to 50 MB. The total latency increases gradually from 0.20 sec to 0.58 sec due to higher encryption and upload times for larger files. By contrast, chaotic key generation latency is almost stable (0.07- 0.08 sec), and the blockchain latency is constant (0.01 sec), showing low overhead of the PBFT-based

verification process. As the file size increased, the throughput decreased from 4.95 files/s to 1.74 files/s, as anticipated, because larger files require more processing and transmission time.

Fig. 3. Performance evaluation of metrics chart.

The performance of the proposed system is compared with that of existing methods in terms of latency, throughput, and security overhead. In order to make a fair comparison, maximum file size used in the performance evaluation was 50MB. The comparative results are presented in Table II and Fig. 4.

TABLE II. PERFORMANCE METRICS AND CUMULATIVE SECURITY OVERHEADS.

Metric	Cloud system	Cloud + AES system	Cloud + AES + Chaotic system	Proposed system
Latency (Sec)	0.27	0.50	0.57	0.58
Throughput (Files/Sec)	3.73	2.01	1.76	1.74
AES Overhead (%)	0	85.73	85.73	85.73
Chaotic Overhead (%)	0	0	14.29	14.29
Blockchain Overhead (%)	0	0	0	1.20

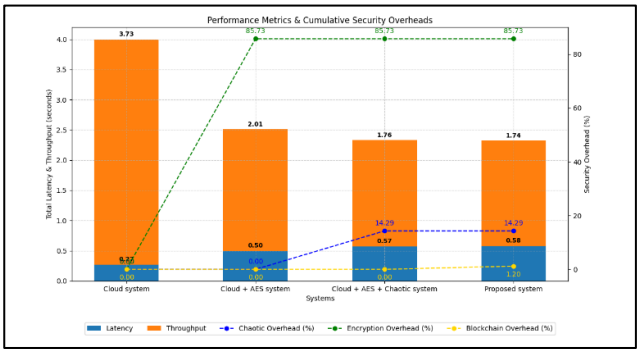

Fig. 4. Performance metrics and cumulative security overheads.

As shown in Table II and Fig. 4, the proposed system causes only a minor latency increase from 0.57 sec to 0.58 sec when compared with the Cloud + AES + Chaotic system. The slight increase is due to the extra verification layer added to the blockchain, which is based on the PBFT protocol. There is also a slight drop in throughput from 1.76 files/s to 1.74 files/s, demonstrating that blockchain integration has a negligible effect on the processing efficiency. Although the proposed system introduces AES, chaotic key generation, and blockchain verification overheads, the proposed blockchain overhead is only 1.20%, which shows that the proposed system provides increased integrity, auditability, and Byzantine fault tolerance at a small performance cost.

B. Chaotic Key Randomness Analysis

The Lorenz 3D chaotic key sequences were evaluated using several randomness evaluation measures, including Shannon entropy, bit balance analysis, chi-square testing, autocorrelation analysis, and key sensitivity analysis, as listed in Table III.

TABLE III. STATISTICAL ANALYSIS OF GENERATED CHAOTIC KEY SEQUENCE.

Category	Metric / Test	Result	Interpretation
Entropy Analysis	Shannon Entropy	7.99	Near-ideal randomness close to the theoretical maximum for Shannon entropy H=8
Bit Balance Analysis	Number of 0s	128242	Nearly balanced binary distribution
	Number of 1s	127758	Indicates low statistical bias
Chi-Square Analysis	Chi-Square Value	229.50	Near-uniform byte distribution with minimal deviation
Autocorrelation Analysis	Lag 0	1.00000	Perfect self-correlation at zero lag
	Lag 1	0.00245	Near-zero correlation
	Lag 2	0.00101	Near-zero correlation
	Lag 3	-0.00212	Near-zero correlation
	Lag 4	0.00128	Near-zero correlation
	Lag 5	0.00181	Near-zero correlation
	Lag 6	-0.00148	Near-zero correlation
	Lag 7	-0.00133	Near-zero correlation
	Lag 8	-0.00227	Near-zero correlation
	Lag 9	-0.00165	Near-zero correlation
Sensitivity Analysis	Key Sensitivity	0.51562	Strong chaotic sensitivity and diffusion behavior
NIST SP 800-22 Tests	Frequency Test	0.33877 (PASS)	Balanced distribution of bits
	Block Frequency Test	0.80179 (PASS)	Uniformity across bit blocks
	Runs Test	0.21520 (PASS)	Absence of abnormal repetitive patterns
	Longest Run Test	0.75065 (PASS)	Random distribution of continuous bit runs
	Approximate Entropy Test	0.52712 (PASS)	High sequence complexity and unpredictability

The results shown in Table III demonstrate that the Lorenz 3D chaotic key sequence exhibits good randomness and cryptographic features. The entropy value of 7.99 is close to the theoretical maximum of 8, and the nearly balanced distribution of 0s and 1s and the low chi-square value indicate that there is no statistical bias. In addition, the near-zero autocorrelation values and high key sensitivity indicate high

unpredictability and dependence on the initial conditions. The NIST SP 800-22 tests were passed successfully, thereby demonstrating that the keys produced were statistically random and appropriate for secure cryptographic use.

C. Ciphertext Statistical Analysis

A ciphertext statistical analysis is conducted to assess the effectiveness of the proposed encryption framework in creating statistically random encrypted data. To analyze the encryption diffusion and statistical security, the ciphertexts were tested by measuring the entropy, chi-square, autocorrelation, and avalanche effects (Table IV).

TABLE IV. STATISTICAL EVALUATION OF CIPHERTEXT

Category	Metric / Test	Result	Interpretation
Entropy Analysis	Shannon Entropy	7.99	Near-ideal ciphertext randomness close to the theoretical maximum H=8
Chi-Square Analysis	Chi-Square Value	295.75	Near-uniform ciphertext distribution with low statistical deviation
Autocorrelation Analysis	Lag 0	1.00000	Perfect self-correlation at zero lag
	Lag 1	0.00345	Near-zero correlation
	Lag 2	0.00104	Near-zero correlation
	Lag 3	0.00160	Near-zero correlation
	Lag 4	0.00188	Near-zero correlation
	Lag 5	0.00123	Near-zero correlation
	Lag 6	-0.00632	Near-zero correlation
	Lag 7	0.00194	Near-zero correlation
	Lag 8	-0.00840	Near-zero correlation
	Lag 9	-0.00400	Near-zero correlation
Avalanche Effect Analysis	1-bit Plaintext Change	50.05%	Strong diffusion and nonlinearity
	2-bit Plaintext Change	50.03%	Consistent ciphertext diffusion
	3-bit Plaintext Change	49.99%	Near-ideal avalanche behavior
	4-bit Plaintext Change	49.99%	Stable diffusion performance
Overall Ciphertext Behavior	Average Autocorrelation	Near Zero	Strong statistical independence between ciphertext bytes

The results presented in Table IV demonstrate that the proposed framework provides strong resistance to a wide range of security threats. The large key space (2^{256}), high entropy (7.99), and avalanche effect (50.05%) indicate robustness against brute-force, statistical, and differential attacks. In addition, AES-256-GCM authenticated encryption effectively protects against known-plaintext and chosen-ciphertext attacks, whereas dual-hash verification and blockchain immutability ensure reliable detection of data tampering and replay attacks.

D. Key Space Analysis and Histogram

The proposed scheme is based on the Lorenz 3D chaotic system, which has high sensitivity to generate chaotic sequences, and processes the chaotic sequence using the SHA-256 hash function to obtain a fixed-length AES-256-GCM key. The last encryption key is:

$$K = SHA256(f(x_0, y_0, z_0))$$

where, $f(x_0, y_0, z_0)$ is the output of the Lorenz chaotic system, and K is the key of AES (256 bits).

The chaotic system has a large parameter space and is very sensitive to the initial conditions; however, the key space of the final AES key specifies the space that can be used for cryptography.

Since SHA-256 outputs a 256-bit digest, the maximum effective key space is (2^{256}). This key space is too large and computationally impractical for exhaustive searches using existing classical computing methods. The key space histogram is shown in Fig. 5.

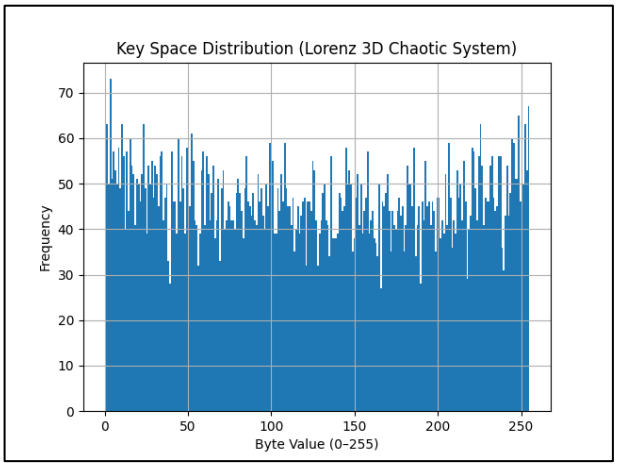

Fig. 5. Key space histogram

Fig. 5 illustrates that the proposed system achieves a key space of (2^{256}). The histogram exhibits a nearly uniform distribution of key space values evenly distributed throughout all 256 possible byte values, suggesting high randomness and no statistical bias in the key-space. This provides excellent resistance to both brute-force and statistical attacks.

E. Plaintext Histogram

The histogram of plaintext file is shown in Fig. 6.

In Fig. 6, the histogram shows a nonhomogeneous data distribution, has noticeable peaks, reflects inherent data patterns, and has a high degree of statistical redundancy. This irregularity makes the plaintext vulnerable to attacks, such as frequency analysis, statistical attacks, and pattern recognition.

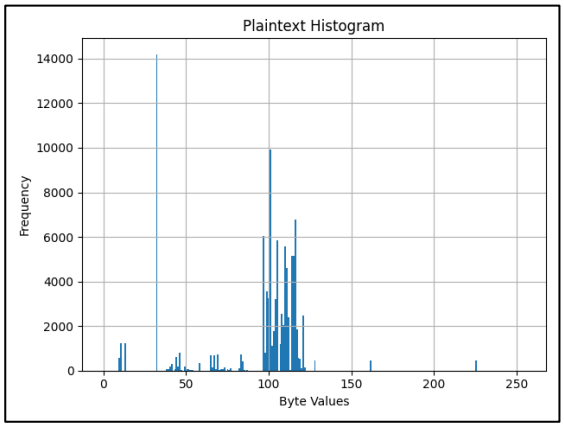

Fig. 6. Plaintext histogram

F. Ciphertext Histogram

The histogram of ciphertext generated by the proposed system is shown in Fig. 7.

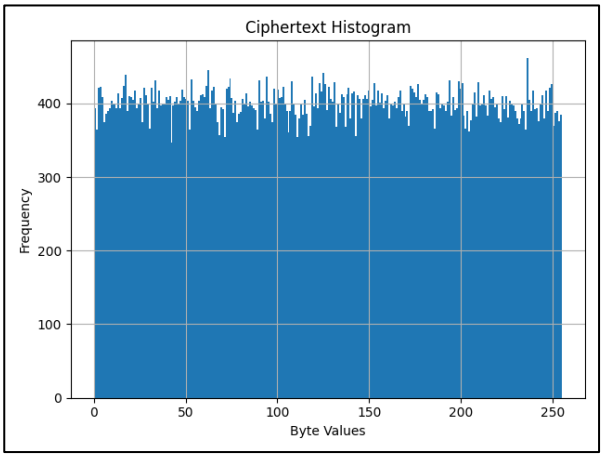

Fig. 7. Ciphertext histogram

The ciphertext histogram in Fig. 7 is almost unbiased, shows an absence of visible patterns, has a flattened frequency distribution curve, and features an equal distribution of all 256-byte counts. This indicates that diffusion, confusion, and statistical randomness were high. This chaotic key is used to increase the randomness of the AES cipher, thereby making the histogram more homogeneous. Thus, chaotic enhancement offers a considerable degree of uncertainty, randomness, and resistance to differential analyses.

G. Attacks and Resistance

Different multi-file real-attack simulations were performed, including unauthorized access, file tampering, blockchain tampering, brute-force, statistical, differential, known-plaintext, chosen ciphertext, replay, and Byzantine fault-tolerance attacks. The proposed framework exhibited excellent anti-attack capabilities against various attack types. Lorenz chaotic key generation, AES-256-GCM encryption and PBFT blockchain offer robustness against cryptographic, statistical, and system-level attacks. The security performance against various attacks is presented in Table V.

TABLE V. SECURITY PERFORMANCE AGAINST VARIOUS ATTACKS

Attack Type	Evaluation Metric	Result	Security Outcome
Unauthorized Access Attack	Access Control Success	100%	Access denied
Data Tampering Attack	Integrity Detection Rate	100%	Tampering detected
Blockchain Tampering	Block Validation Failure	100%	Invalid block rejected
Brute-Force Attack	Key Space	2^{256}	Computationally infeasible
Statistical Attack	Entropy	7.99 bits	Near-ideal randomness
Differential Attack	Avalanche Effect	50.05%	Strong diffusion
Known Plaintext Attack	Key Recovery Probability	Negligible	Resistant
Chosen Ciphertext Attack	Authentication Failure Detection Rate	100%	Attack Prevented
Replay Attack	Detection Rate	100%	Attack prevented
Man-in-the-Middle Attack	Hash Verification Success	100%	Attack detected
Byzantine Fault Tolerance	Maximum Faulty Nodes Tolerated	$f \leq \frac{n-1}{3}$	Consensus Maintained

Table V shows that the proposed framework offers robust resistance against several cryptographic and blockchain attacks. The user authentication and access control mechanisms prevent unauthorized access to the stored data. A large key space of (2^{256}), avalanche effect of 50.05% and entropy value of 7.99 assure strong resistance to brute force, statistical and differential attacks. The combination of Lorenz chaotic key generation and AES-256-GCM provides an additional layer of security against known plaintext and chosen ciphertext attacks. Blockchain Timestamp Validation, AES-GCM Authentication Tags, and Dual-Hash Integrity Verification help prevent replay attacks, data tampering, and ciphertext modification. Moreover, blockchain tampering is detected using immutable ledger records and hash verification mechanisms. Finally, PBFT ensures accurate network functioning even in situations where malicious validator nodes are present. Consensus is guaranteed as long as the number of faulty nodes does not exceed one-third of the total validators ($f \leq (n-1)/3$), demonstrating the robustness of the proposed framework against Byzantine failures. Overall, the results confirm that the proposed framework provides confidentiality, integrity, authenticity, and fault tolerance in cloud-storage environments.

VII. THREAT MODEL

The threat model defines the assumptions, attack surfaces, trust boundaries, and security goals of the proposed secure cloud-storage framework. The primary intent of this threat model is to evaluate how well the proposed framework can guarantee confidentiality, integrity, authenticity, auditability, and decentralized trust in realistic threat scenarios.

A. Adversary Assumptions

The adversary is assumed to have polynomial-time computational power and can attempt to attack various components of the system. An attacker can perform any of the following actions:

- Intercept communication channels
- Capture encrypted data packets
- Replay previously transmitted ciphertext
- Modify stored cloud data
- Attempt brute-force key recovery
- Perform statistical analysis on ciphertext
- Compromise a subset of blockchain validator nodes
- Attempt unauthorized access to cloud resources

This adversary is assumed to be capable of completely using the public communication network but lacking a legitimate cryptographic key or secret chaotic parameters.

B. Trust Assumptions

The following are the key assumptions of the proposed framework:

- AES-256-GCM remains computationally secure.
- SHA-256 is collision and pre-image-resistant.
- Initial chaotic parameters remain secret.
- Fewer than one-third of the PBFT validator nodes were malicious.
- It is assumed that the authentication mechanisms have been properly implemented.
- The random initialization vectors were generated securely.

These assumptions are based on widely used cryptographic and distributed-system security models.

C. Security Goals

The proposed framework is designed to achieve the following security goals (Table VI):

TABLE VI. SECURITY GOALS

Security Goal	Description
Confidentiality	Prevent unauthorized disclosure of plaintext
Integrity	Detect unauthorized modifications
Authentication	Verify legitimate users
Auditability	Maintain immutable access records
Availability	Ensure reliable data access
Non-Repudiation	Prevent denial of performed actions
Byzantine Fault Tolerance	Resist malicious validator behavior

The combination of cryptographic protection and decentralized verification in the framework provides multilayer security in a cloud storage environment.

VIII. SECURITY ANALYSIS

The security strength of the proposed framework is studied within the context of an extensive threat model of external attackers, insider attacks, and attempts to tamper with the data. The framework aims to maintain data confidentiality, integrity, authentication, and auditability through encryption using AES-256-GCM with Lorenz chaos and blockchain-assisted integrity verification (IV) techniques.

A. Overview of the Security Objectives

The proposed framework uses the Lorenz 3D chaotic system to generate a chaotic key, AES-256-GCM symmetric encryption, and a PBFT-based blockchain system to achieve multi-dimensional security. It has the following major security objectives:

- The data were confidential, as ensured through AES-256-GCM encryption.
- Assign a random value to the 'key' to achieve key unpredictability (using Lorenz chaotic dynamics).
- Integrity & Immutability: Ensured with blockchain hashing.
- The authentication & consensus are enforced through PBFT.
- Attack Resistance: Against brute-force, known-plaintext, chosen-plaintext, replay, statistical, Byzantine, and side-channel attacks, unauthorized access, file tampering, and blockchain tampering.

Let the system be defined as:

$$C = E_K(P), \quad K = H(f(x_0, y_0, z_0))$$

where, (x_0, y_0, z_0) are Lorenz's initial conditions and $H(\cdot)$ is SHA-256.

B. Unauthorized Access Resistance

Unauthorized access resistance is provided by authenticated encryption, blockchain access verification, and secure key management. The key used to encrypt the data is the AES-256-GCM encryption key, which is dynamically generated from the Lorenz chaotic system and is not stored as plain text in the cloud environment.

Only authenticated and authorized users with valid cryptographic credentials can decrypt the stored ciphertext successfully. Moreover, verification on the blockchain does not allow any unauthorized changes to be made in the access records or metadata of transactions to enhance protection against unauthorized attempts to access the cloud storage.

C. Ciphertext and File Tampering Resistance

The framework uses a dual integrity verification mechanism: AES-256-GCM authentication tag validation and blockchain-based SHA-256 ciphertext hash validation. If any part of the ciphertext, nonce, authentication tag, or content of the stored file changes, the authentication fails, or the hash of the blockchain does not match. In addition, the ciphertext hashes stored in the blockchain were verified before decryption. Therefore, when unauthorized changes are made, they are immediately discovered and rejected. This mechanism strengthens the resistance against:

- Ciphertext tampering
- Malicious modification
- Unauthorized data manipulation
- File integrity attacks

D. Blockchain Tampering Resistance

The PBFT-based blockchain layer offers distributed integrity checks and BFT. The number of malicious nodes that can be accommodated by a consensus by PBFT is limited to $(f \leq \frac{n-1}{3})$ validator nodes.

Blockchain metadata, such as the ciphers and transactions, is immutable and cannot be altered without the agreement of the validators that are part of the consensus. Therefore, the integrity records and transaction histories would be impossible for the attackers to change without compromising most of the validator nodes.

E. Brute-Force Resistance

The proposed framework uses AES-256-GCM authenticated encryption with a 256-bit chaotic key generated by the Lorenz 3D chaotic system. The total key space is (2^{256}), which is computationally infeasible to exhaustively search using current classical computing capabilities.

Moreover, the Lorenz chaotic system is highly sensitive to the initial conditions, leading to greater unpredictability. The slightest change in the chaotic parameters produces entirely different encryption keys, which makes the system even harder to break using brute-force and key prediction.

F. Statistical Attack Resistance

The generated ciphertext exhibits the following properties based on statistical analysis:

- Near-ideal Shannon entropy (H=7.99)
- Balanced statistical distribution
- Strong avalanche characteristics

In addition, the autocorrelation values of the ciphertext are close to zero for all nonzero lags, which further proves, on a statistical level of significance, that the ciphertext bytes are statistically independent. These results showed resistance to:

- Frequency analysis
- Statistical cryptanalysis
- Differential attacks
- Pattern recognition attacks

G. Differential Attack Resistance

The strong diffusion and confusion properties of the proposed encryption framework provide resistance to differential attacks. Experimental avalanche effect analysis showed that small changes in the plaintext resulted in enormous changes in the ciphertext, and the average value of the avalanche effect was nearly 50%, which is the theoretical ideal.

The Lorenz chaotic key generation mechanism further enhances the nonlinear diffusion characteristics and the unpredictability. In this way, the attacker will not be able to establish meaningful relationships between the differences in plaintext and ciphertext, making the system more resistant to differential cryptanalysis.

H. Known-Plaintext Attack Resistance

The proposed framework can prevent known plaintext attacks using dynamic chaotic key generation and AES-256-GCM authentication encryption. In the event that an adversary acquires partial plaintext/ciphertext pairs, the encryption key cannot be reconstructed because the Lorenz chaotic parameters remain concealed, the SHA-256 key derivation is one-way, and the AES-256-GCM delivers computational confidentiality.

Additionally, high ciphertext randomness and near-zero autocorrelation make it difficult for an attacker to detect exploitable statistical relationships between the known plaintext and ciphertext data.

I. Chosen-Ciphertext Attack Resistance

The framework includes authenticated encryption using AES-256-GCM and integrity verification using blockchain, which resists adaptively chosen ciphertext attacks. AES-GCM validates the authentication tag before decrypting data.

If the ciphertext, nonce, or authentication tag is modified in any way, decryption fails: $Dec_K(C') = \perp$

Hence, any malicious manipulation of the ciphertext does not provide useful information regarding plaintext data.

Furthermore, the hash of the ciphertexts stored on the blockchain is checked before the block is decrypted, allowing for early detection of unauthorized changes to the ciphertexts. Together, these mechanisms enhance resistance to IND-CCA2 adversaries and ciphertext forgery attacks.

J. Replay Attack Resistance

Replay attack resistance is implemented using unique nonces for AES-GCM and transaction validation in the blockchain. Every encryption operation results in different nonce and transaction identifiers, so that the same ciphertext cannot be reused. Blockchain metadata validation ensures the authenticity and integrity of the transaction before it is decrypted during retrieval from the file. Duplicated or outdated transactions are rejected, thus avoiding a replay attack.

In addition, replay protection has been improved with unique AES-GCM nonces and blockchain timestamps. As each file is encrypted with a different nonce and with a unique blockchain transaction record, replayed ciphertexts or duplicated upload requests will be identified via metadata verification/timestamp validation.

K. Byzantine Fault Tolerance Resistance

To overcome malicious or compromised validators, the proposed consortium blockchain uses a PBFT consensus protocol. Using PBFT, the blockchain can continue to function properly, even if some of the validator nodes are malicious or fail unexpectedly. A transaction is committed only when at least $(2f + 1)$ validator nodes approve the block.

This means that Byzantine attackers have no ability to tamper with the blockchain integrity record, audit logs, or access verification metadata without compromising more than $(2/3)$ of the validator nodes.

This mechanism is strengthened by:

- Distributed trust
- Consensus reliability
- Blockchain integrity
- Resistance against insider consensus manipulation

L. Insider Attack Resistance

The proposed framework helps to reduce insider threats by implementing the Encrypted cloud storage, Blockchain-based access verification, Immutable audit logging, and Authenticated encryption.

Cloud administrators and malicious insiders cannot access plaintext data without the corresponding chaotic-encryption keys. Moreover, every file action is logged as an immutable transaction on the blockchain, allowing for open and transparent auditing and accountability of information. Blockchain integrity verification and AES-GCM authentication can detect any unauthorized attempts to modify the data.

M. Overall Security Discussion

The Lorenz 3D chaotic key generation, AES-256-GCM authenticated encryption, SHA-256 integrity verification, and PBFT blockchain consensus contribute to the security of the proposed framework. The experimental tests yielded the following results:

- Near-ideal entropy
- Low statistical correlation
- Strong avalanche behavior
- Successful NIST randomness validation was achieved.

The results indicate that the proposed framework is resistant to brute-force, statistical cryptanalysis, replay, ciphertext tampering, insider, and manipulation attacks on the blockchain.

N. Comparative Security Analysis

The data were encrypted using AES-256-GCM to ensure their confidentiality. The SHA-256 hashes of the encrypted and plaintext files are stored in the blockchain, guaranteeing data integrity. Blockchain immutability ensures that access policies and audit logs cannot be changed, which, in turn, ensures non-repudiation and accountability of the data. AES encryption protects data privacy and keeps those who are not authorized to access the plaintext data, such as cloud providers. Smart contracts are decentralized access controls, data integrity with irrevocable hashes, and auditability on blockchains. A comparative security analysis of the proposed and conventional systems is presented in Table VII.

TABLE VII. COMPARATIVE SECURITY ANALYSIS

Feature	AES System	AES + Chaotic System	Proposed System
Confidentiality	Yes	Yes	Yes
Integrity Verification	Limited	Limited	Yes
Blockchain Auditability	No	No	Yes
Chaotic Randomness	No	Yes	Yes
PBFT Consensus	No	No	Yes
Replay Protection	No	Limited	Yes
Byzantine Fault Tolerance	No	No	Yes
Dual Hash Verification	No	No	Yes
NIST Validation	No	Partial	Yes

The comparative results presented in Table VII demonstrate that the proposed framework provides a more comprehensive security solution than existing approaches. Unlike existing schemes, which are mostly concerned with chaotic encryption and key space enhancement, the proposed scheme further includes PBFT consensus, Byzantine fault tolerance, replay protection, blockchain-based auditability, and dual-hash integrity check. Although maintaining comparable entropy (approx. 7.99) and a large key space (2^{256}), the proposed system offers stronger resistance to data tampering, replay attacks, and blockchain-related threats. The findings indicate that the combination of Lorenz chaotic key generation, AES-256-GCM encryption, and blockchain verification using PBFT can significantly improve the security and trust of cloud storage systems.

IX. FORMAL SECURITY ANALYSIS

A. Security Assumptions

The proposed framework relies on the following assumptions:

- AES-256-GCM is IND-CCA2 secure when nonces are unique.
- SHA-256 is collision-resistant and preimage-resistant.
- The Lorenz chaotic parameters remain secret and unpredictable, ideally derived from high-entropy sources or secure key exchange protocols during system initialization to prevent their compromise.
- PBFT consensus is secure if fewer than one-third of validator nodes are Byzantine.
- Validator nodes are distributed across independent entities in a permissioned consortium blockchain.

B. IND-CCA2 Security of the Encryption Layer

Theorem 1: The encryption layer of the proposed framework achieves IND-CCA2 security under the assumption that AES-256-GCM is a secure authenticated encryption scheme and that nonces are never reused with the same key.

Game 0: Real IND-CCA2 Game

The challenger generates the encryption key: $K = H(f(x_0, y_0, z_0))$

where, $f(x_0, y_0, z_0)$ denotes the Lorenz chaotic key.

The adversary A is given access to: $Enc_K(\cdot)$ and $Dec_K(\cdot)$

The adversary submits two equal-length plaintexts: (P_0, P_1)

The challenger selects: $b \in \{0, 1\}$

and returns: $C^* = Enc_K(P_b, N, AAD)$

The adversary may continue querying the decryption oracle, except that it may not query the exact challenge ciphertext C^*. Finally, the adversary outputs b'.

The adversarial advantage is as follows:

$$Adv_A^{IND-CCA2} = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

Game 1: Replace Derived Key with Uniform Random Key

In this game, the derived key $K = H(f(x_0, y_0, z_0))$

is replaced with a uniformly random 256-bit key as follows:

$$K' \leftarrow \{0, 1\}^{256}$$

If the adversary distinguishes Game 0 from Game 1, then it can distinguish the derived key from a uniformly random AES key.

$$\text{Thus: } |\Pr[G_0] - \Pr[G_1]| \leq \epsilon_{KDF}$$

where, ϵ_{KDF} denotes the advantage of distinguishing the SHA-256-derived chaotic key from a uniformly random 256-bit key.

Game 2: AES-GCM IND-CCA2 Game

In Game 2, encryption and decryption are performed using AES-256-GCM with a uniformly random key K'.

AES-GCM provides authenticated encryption. Therefore, any modified ciphertext is rejected unless the adversary produces a valid authentication tag.

For a 128-bit authentication tag: $\Pr[Forge] \leq 2^{-128}$

Since AES-GCM is assumed IND-CCA2 secure under nonce uniqueness, the adversary cannot distinguish whether the challenge ciphertext encrypts P_0 or P_1 with non-negligible probability.

Therefore: $\Pr[G_2] = \frac{1}{2}$

and: $Adv_A^{IND-CCA2} \leq \epsilon_{KDF} + Adv_{AES-GCM}^{IND-CCA2}$

Since both terms are negligible: $Adv_A^{IND-CCA2} \approx 0$

Conclusion: The encryption layer is IND-CCA2 secure under the assumptions that AES-256-GCM is IND-CCA2 secure, nonces are unique, and the derived chaotic key is computationally indistinguishable from a random 256-bit key.

C. Blockchain Integrity Security

Theorem 2: The blockchain integrity layer detects ciphertext metadata tampering with an overwhelming probability, assuming that SHA-256 is a collision-resistant hash function.

The blockchain stores the ciphertext hash: $H_{\text{cipher}} = SHA256(C)$

During retrieval, the system computes: $H_{\text{local}} = SHA256(C')$

The file is accepted only if: $H_{\text{local}} = H_{\text{cipher}}$

If an adversary modifies the ciphertext: $C \rightarrow C'$

Then successful undetected tampering requires: $SHA256(C') = SHA256(C)$

where, $C' \neq C$

This implies a collision in SHA-256.

Therefore: $Pr[Undetected\ Tampering] \leq Adv_{SHA256}^{Collision}$

Since SHA-256 collision resistance is assumed secure: $Pr[Undetected\ Tampering] \approx 0$.

D. PBFT Consensus Security

Theorem 3: The PBFT-based blockchain maintains consensus safety if fewer than one-third of the validator nodes are Byzantine nodes.

For n validator nodes, PBFT tolerates: $f \leq \frac{n-1}{3}$ Byzantine nodes.

A block is committed only if at least: $2f + 1$ validators approve the transaction.

Therefore, an adversary can alter the blockchain metadata only if it controls more than one-third of the validator nodes or violates the PBFT consensus.

$$\text{Thus: } Pr[PBFT\ Failure] \leq Adv_{PBFT}$$

Under the stated validator trust model: $Adv_{PBFT} \approx 0$

E. Overall Security Bound

The overall framework security follows from the separate security properties:

$$Adv_{System} \leq Adv_{AES-GCM}^{IND-CCA2} + Adv_{SHA256}^{Collision} + Adv_{PBFT}$$

where,

- $Adv_{AES-GCM}^{IND-CCA2}$ represents the encryption-layer confidentiality advantage
- $Adv_{SHA256}^{Collision}$ represents the probability of undetected metadata tampering
- Adv_{PBFT} represents the probability of violating PBFT consensus assumptions

Unlike the previous version, blockchain integrity and PBFT consensus were not included as hybrid games in the IND-CCA2 proof. Instead, they are treated as separate system-level security properties.

Since each term is negligible under the stated assumptions: $Adv_{System} \approx 0$.

X. Comparative Study Table

Table VIII summarizes the main features of this study and some published studies and illustrates some important differences between them. Most current approaches are either encryption or blockchain-verification-driven, ignoring high-dimensional chaotic key generation and adequate formal security evaluation. The goal of the framework is to fill these key gaps by combining these elements and offering something that is more complete and provably secure than those proposed in the current literature.

The comparative results presented in Table VIII show that the proposed framework provides a more comprehensive and integrated security solution than existing approaches. The proposed framework integrates Lorenz 3D chaotic key generation, AES-256-GCM authenticated encryption, blockchain-based integrity verification, and PBFT consensus into a single architecture, whereas most previous studies focused on a single aspect of encryption, access control, or blockchain verification. This integration improves the privacy, integrity, audibility, and Byzantine fault tolerance of data while providing robust storage security in the cloud and efficient metadata verification. As a result, the proposed framework provides more security resilience and a comprehensive defense-in-depth protection for cloud storage systems.

TABLE VIII. Comparative Study Table

Features/ System	[11]	[32]	[33]	[34]	Proposed System
Architecture	Fully decentralized blockchain network	Blockchain + cloud server	Blockchain + cloud	Blockchain-enabled cloud architecture	Multi-layered architecture
Encryption Method	CP-ABE	SHA-256 + ECDSA	ABE	ChaCha20-Poly1305	AES-256-GCM
Key Generation Method	Bilinear mapping-based cryptography	--	ABE	ChaCha20-Poly1305	Lorenz chaotic 3D system
Blockchain Role	Access control + integrity	Transaction verification	Data verification	Access control + metadata	Decentralized integrity verification, immutable audit logging, and PBFT-based consensus for Byzantine fault tolerance.
Storage Model	Distributed cloud	Cloud server	Cloud + blockchain metadata	Cloud storage	Encrypted files are stored in cloud storage with integrity hashes and metadata maintained on a PBFT-based blockchain.
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.	Detected evaluated attack scenarios, demonstrating strong security resilience, reliable integrity verification, and effective Byzantine fault tolerance
Key Features	Authorization, decentralized storage	Secure medical record sharing	Secure data sharing and verification	Improved throughput and lower latency	Attack detection and prevention, AES-256-GCM authenticated encryption, 10D hyperchaotic key generation, blockchain-based integrity verification, and PBFT-based Byzantine fault tolerance.

XI. Discussion

The experimental findings show that the proposed secure cloud storage framework effectively combines chaos-based cryptography, authenticated encryption, and blockchain-based verification, thereby delivering a comprehensive security solution for cloud environments. The combination of Lorenz 3D chaotic key generation, AES-256-GCM encryption, and PBFT-based blockchain verification achieves multiple security objectives, including confidentiality, integrity, authentication, auditability, and fault-tolerance.

The randomness analysis of the generated chaotic keys validates the effectiveness of the Lorenz chaotic system in cryptographic key generation. The Shannon entropy value was 7.99, the bit distribution was balanced, the autocorrelation values were close to zero, and the NIST SP 800-22 test results were positive, which shows that the generated keys have good randomness properties and low statistical bias. These properties greatly improve the unpredictability and resistance to brute-force and statistical attacks.

The robustness of the encryption layer can also be validated using ciphertext analysis. The ciphertext had near-ideal entropy, low chi-square deviation, negligible auto-correlation, and a mean avalanche effect of approximately 50%, indicating good diffusion and confusion properties. The findings show that the proposed encryption scheme successfully hides the pattern of the plaintext and reduces information leakage, which enhances its differential and statistical cryptanalysis resistance.

The performance evaluation demonstrates that the latencies for encryption, upload, and total processing grow steadily as the file size increases, as expected for cryptographic operations on larger datasets. Although the integration of chaotic key generation and blockchain verification adds some computational overhead, the measured PBFT consensus latency remains very low (approximately 0.01 sec), suggesting that integrity verification can be performed efficiently without a significant impact on the overall system performance. As the file sizes increase, the throughput decreases; however, the framework remains efficient while offering significantly improved security promises.

The comparative analysis shows that the conventional cloud storage has a higher throughput and lower latency but lacks some advanced security capabilities, such as decentralized integrity verification, blockchain auditability, Byzantine fault tolerance, and chaos-enhanced cryptographic randomness. The proposed framework, on the other hand, has a

relatively small overhead and provides much greater protection. The blockchain layer provides immutability to audit records and tamper-proof features, whereas the PBFT consensus mechanism guarantees dependable operation even when dealing with malicious validator nodes.

The security evaluation and results under various attack scenarios demonstrate the effectiveness of the proposed design. The combination of the large key space of (2^{256}), good avalanche behavior, authenticated encryption, dual-hash verification and immutability of the blockchain offers significant protection against unauthorized access, brute force attacks, ciphertext manipulation, replay attacks, known-plaintext attacks, chosen-ciphertext attacks, and Byzantine failures. Based on these findings, a defense-in-depth security architecture is achieved in the proposed framework, which is applicable to sensitive cloud-based applications.

However, the present implementation was tested in a local consortium blockchain setup with a small number of validator nodes. Hence, the reported PBFT latency and throughput numbers might vary in geographically distributed cloud deployments, where network delays and communication overhead have a greater impact. Additionally, the communication complexity of the PBFT consensus is $(O(n^2))$, potentially impacting scalability as the number of validators increases.

Overall, the results demonstrate that the proposed framework effectively achieves a trade-off between security and performance through the use of chaos-based key generation, authenticated encryption, and blockchain verification of integrity. The architecture is well-suited for high-security needs and verifies the integrity of data, such as healthcare information systems, enterprise cloud storage platforms, financial services, and Internet of Things (IoT) applications.

A. Effect of Euler Integration on Chaotic Key Quality

The Lorenz system was numerically integrated using the Euler method; however, the statistical evaluation results showed that the chaotic sequences generated by this system were highly random throughout the experiments. The entropy value of 7.99 bits, successful NIST SP 800-22 test results, and low autocorrelation indicate that the generated key sequences were statistically uniform in the range of iterations tested and file sizes. Based on these results, it seems that the numerical errors caused by the Euler method did not significantly affect the cryptographic quality of the generated keys with the experimental settings. However, future studies could explore the use of higher-order integration methods, such as the fourth-order Runge–Kutta algorithm, to further enhance the long-term numerical accuracy and chaotic trajectory stability.

B. Limitations of the PBFT Performance Evaluation

The PBFT performance evaluation was performed on a four-node consortium blockchain prototype. Hence, the measured latency and throughput data must be considered as prototype-scale numbers under ideal experimental conditions. Although the results show the feasibility of integrating PBFT with the proposed cloud security framework, they do not completely reflect the communication overhead that can occur in a larger consortium blockchain application. As the number of validator nodes grows, the network traffic and consensus latency of PBFT are expected to grow as well, because its message complexity is $(O(n^2))$ of the number of nodes. Therefore, these reported performance metrics are not necessarily applicable to large-scale cloud environments. The framework will be tested for scalability in future work across geographically distributed nodes and larger networks of validators to validate it in realistic deployment scenarios.

XII. Conclusion

This study proposed a secure integrated cloud storage system that integrates the Lorenz 3D chaotic key generation method, AES-256-GCM authenticated encryption, and blockchain verification technology based on PBFT. The proposed system enhances data confidentiality, integrity, auditability, and Byzantine fault tolerance by storing encrypted files in the cloud and integrity metadata on the blockchain. The experimental outcomes demonstrated excellent cryptographic performance, such as a nearly perfect entropy of 7.99, an avalanche effect of 50.05%, low autocorrelation, and successful randomness validation. The framework also proved to be resistant to brute-force, statistical, differential, replay, tampering, known-plaintext, chosen-ciphertext, and Byzantine attacks. Although the evaluation was performed in a local prototype setting, the outcome of the experiment validates the proposed framework of providing a secure and efficient multilayer solution for cloud storage applications.

XIII. Future Work

This work can be extended in the future by the addition of post-quantum cryptographic techniques and deployed in a real-world, large-scale distributed cloud system with enhancements for performance and scalability.

AI Usage Disclaimer

The authors acknowledge the use of artificial intelligence (AI)-assisted tools for language refinement, grammar correction, formatting assistance, and improving the clarity of technical explanations during manuscript preparation. AI tools were not used to generate research results, experimental data, scientific conclusions, or core technical contributions. All system designs, implementations, analyses, and validations were independently conducted and verified by the authors.

TABLE IX. List of Notations and Abbreviations

Symbol / Notation	Description
P	Plaintext file
C	Ciphertext
K_{AES}	256-bit AES encryption key
IV	Initialization Vector (Nonce)
AAD	Associated Authenticated Data
T	Authentication tag generated by GCM
H_{plain}	SHA-256 hash of plaintext
H_{cipher}	SHA-256 hash of ciphertext
J_0	Initial counter block in GCM

J_i	Counter block for encryption round i
S_i	AES-generated keystream block
P_i	Plaintext block i
C_i	Ciphertext block i
H	GHASH hash subkey
G	GHASH output value
x, y, z	Lorenz chaotic state variables
x_0, y_0, z_0	Initial conditions of Lorenz system
σ	Lorenz system parameter (Prandtl number)
ρ	Lorenz system parameter (Rayleigh number)
β	Lorenz system parameter
h	Numerical integration step size
N	Number of chaotic iterations
B	Generated chaotic bitstream
K	Generated chaotic secret key
SHA-256	Secure Hash Algorithm (256-bit)
AES-256-GCM	Advanced Encryption Standard with Galois/Counter Mode
PBFT	Practical Byzantine Fault Tolerance
Tx	Blockchain transaction
FileID	Unique file identifier
Timestamp	File upload time recorded on blockchain
Smart Contract	Blockchain-based access control logic
Validator Node	Blockchain participant responsible for consensus
PRE-PREPARE	First PBFT consensus phase
PREPARE	Second PBFT consensus phase
COMMIT	Final PBFT consensus phase

REFERENCES

[1] S. S. Gill and R. Buyya, "A taxonomy and future directions for sustainable cloud computing: 360 degree view," ACM Comput. Surv., vol. 51, no. 5, 2018, doi: 10.1145/3241038.

[2] R. Buyya et al., "A manifesto for future generation cloud computing: Research directions for the next decade," ACM Comput. Surv., vol. 51, no. 5, Sep. 2019, doi: 10.1145/3241737.

[3] P. Yang, N. Xiong, and J. Ren, "Data Security and Privacy Protection for Cloud Storage: A Survey," IEEE Access, vol. 8, pp. 131723–131740, 2020, doi: 10.1109/ACCESS.2020.3009876.

[4] NIST, "Advanced Encryption Standard (AES)," May 2023. doi: 10.6028/NIST.FIPS.197-upd1.

[5] D. P. Leech, S. Ferris, and J. T. Scott, "The economic impacts of the advanced encryption standard, 1996?2017," Gaithersburg, MD, Aug. 2018. doi: 10.6028/NIST.GCR.18-017.

[6] S. Rehman, N. Talat Bajwa, M. A. Shah, A. O. Aseeri, and A. Anjum, "Hybrid aes-ecc model for the security of data over cloud storage," Electronics (Switzerland), vol. 10, no. 21, Nov. 2021, doi: 10.3390/electronics10212673.

[7] N. S. Musa, T. Murugan, N. A. N, and N. M. Mirza, "Research study on securing the cloud: utilizing conventional and blockchain-based access control mechanisms," Dec. 01, 2025, Springer Science and Business Media Deutschland GmbH. doi: 10.1186/s13677-025-00803-3.

[8] W. Lu, C. Jin, J. Wang, X. Liu, J. Liu, and Z. Zhai, "A novel image encryption scheme using 3D chaotic maps with Josephus permutation and dynamic diffusion," Journal of King Saud University - Computer and Information Sciences, vol. 37, no. 8, Oct. 2025, doi: 10.1007/s44443-025-00284-z.

[9] Z. Rahman, X. Yi, M. Billah, M. Sumi, and A. Anwar, "Enhancing AES Using Chaos and Logistic Map-Based Key Generation Technique for Securing IoT-Based Smart Home," Electronics (Switzerland), vol. 11, no. 7, Apr. 2022, doi: 10.3390/electronics11071083.

[10] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008, doi: 10.1007/s10838-008-9062-0.

[11] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain-based decentralized architecture for cloud storage system," Journal of Information Security and Applications, vol. 62, Nov. 2021, doi: 10.1016/j.jisa.2021.102970.

[12] P. J. Sun, "Security and privacy protection in cloud computing: Discussions and challenges," Journal of Network and Computer Applications, vol. 160, no. August 2019, p. 102642, 2020, doi: 10.1016/j.jnca.2020.102642.

[13] M. N. Birje, P. S. Challagidad, R. H. Goudar, and M. T. Tapale, "Cloud computing review: concepts, technology, challenges and security," 2017.

[14] M. B. Qureshi et al., "Encryption Techniques for Smart Systems Data Security Offloaded to the Cloud," Symmetry (Basel)., vol. 14, no. 4, Apr. 2022, doi: 10.3390/sym14040695.

[15] R. Ganesh, B. U. I. Khan, A. R. Khan, and A. Bin Kamsin, "A panoramic survey of the advanced encryption standard: from architecture to security analysis, key management, real-world applications, and post-quantum challenges," Int. J. Inf. Secur., vol. 24, no. 5, Oct. 2025, doi: 10.1007/s10207-025-01116-x.

[16] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain Technology for Cloud Storage: A Systematic Literature Review," ACM Comput. Surv., vol. 53, no. 4, 2020, doi: 10.1145/3403954.

[17] C.-H. Lin, G.-H. Hu, C.-Y. Chan, J.-J. Yan, and J. Chaos-Based, "Chaos-Based Synchronized Dynamic Keys and Their Application to Image Encryption with an Improved AES Algorithm," 2021, doi: 10.3390/app.

[18] B. Zhang and L. Liu, "Chaos-Based Image Encryption: Review, Application, and Challenges," Jun. 01, 2023, MDPI. doi: 10.3390/math11112585.

[19] M. A. Khan, "New Image Encryption Scheme Using Chaotic Maps," in ACM International Conference Proceeding Series, Association for Computing Machinery, Aug. 2019. doi: 10.1145/3387168.3389111.

[20] A. Arab, M. J. Rostami, and B. Ghavami, "An image encryption method based on chaos system and AES algorithm," Journal of Supercomputing, vol. 75, no. 10, pp. 6663–6682, Oct. 2019, doi: 10.1007/s11227-019-02878-7.

[21] A. Alghuried, M. Alkinoon, M. Mohaisen, A. Wang, C. Zou, and D. Mohaisen, "Blockchain Security and Privacy: Threats, Challenges, Applications, and Tools," Distributed Ledger Technologies, vol. 5, no. 1, Dec. 2025, doi: 10.1145/3716323.

[22] Z. Wenhua, F. Qamar, T. A. N. Abdali, R. Hassan, S. T. A. Jafri, and Q. N. Nguyen, "Blockchain Technology: Security Issues, Healthcare Applications, Challenges and Future Trends," Feb. 01, 2023, MDPI. doi: 10.3390/electronics12030546.

[23] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A Survey on the Security of Blockchain Systems," 2020.

[24] J. Singh et al., "A Systematic Review of Blockchain, AI, and Cloud Integration for Secure Digital Ecosystems," Dec. 01, 2025, Springer Science and Business Media B.V. doi: 10.1007/s44227-025-00072-1.

[25] C. Yu, N. Mei, C. Du, and H. Luo, "Blockchain Data Scalability and Retrieval Scheme Based on On-Chain Storage Medium for Internet of Things Data," Electronics (Switzerland), vol. 12, no. 6, Mar. 2023, doi: 10.3390/electronics12061454.

[26] Y. Luo, M. Liu, J. Wang, C. Yuan, and T. Liu, "When Secure Data Sharing Meets Blockchain: Overview, Challenges and Future Prospects," in ACM International Conference Proceeding Series, Association for Computing Machinery, Mar. 2022, pp. 1–8. doi: 10.1145/3532640.3532641.

[27] M. N. Alatawi, "Blockchain-Driven Smart Contracts for Advanced Authorization and Authentication in Cloud Security," Electronics (Switzerland), vol. 14, no. 15, Aug. 2025, doi: 10.3390/electronics14153104.

[28] Z. Yao, Y. Fang, H. Pan, X. Wang, and X. Si, "A secure and highly efficient blockchain PBFT consensus algorithm for microgrid power trading," Sci. Rep., vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-58505-w.

[29] J. Liu, X. Deng, W. Li, and K. Li, "CG-PBFT: an efficient PBFT algorithm based on credit grouping," Journal of Cloud Computing, vol. 13, no. 1, Dec. 2024, doi: 10.1186/s13677-024-00643-7.

[30] F. Nujumudeen et al., "A hybrid chaos-based cryptographic framework for lightweight IoT security: enhancing efficiency and security in low-power devices," Peer. Peer. Netw. Appl., vol. 19, no. 2, p. 53, Feb. 2026, doi: 10.1007/s12083-025-02188-1.

[31] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-38D, Nov. 2007, doi: 10.6028/NIST.SP.800-38d.

[32] A. Amanat, M. Rizwan, C. Maple, Y. Bin Zikria, A. S. Almadhor, and S. W. Kim, "Blockchain and cloud computing-based secure electronic healthcare records storage and sharing," Front. Public Health, 2022, [Online]. Available: https://www.frontiersin.org/

[33] T. Zhang, C. Wang, and M. I. U. Chandrasena, "Blockchain-assisted data sharing supports deduplication for cloud storage," Conn. Sci., vol. 35, no. 1, 2023, doi: 10.1080/09540091.2023.2174081.

[34] R. Golla Bala and S. Gnanavel, "BC2P-1305: an enhanced data security in cloud computing network using blockchain based ChaCha20-Poly1305 cryptography," Frontiers in Blockchain, vol. 8, 2025, doi: 10.3389/fbloc.2025.1636056.