Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 6, December 2026, pp. 5569~5583
ISSN: 2302-9285, DOI: 10.11591/eei.v15i6.11877

Advanced encryption standard with layered patching mechanism for secure cloud data storage

Harish Kumar Narayanaswamy[1], Deepak Gopal[2], Annapoorna Bangalore Ramesh[3], Sarala Duggonahalli Veeresh[4]
[1]Department of Computer Science and Engineering, BMS Institute of Technology and Management, Bengaluru, India
[2]Department of Information Science and Engineering, Global Academy of Technology, Bengaluru, India
[3]Department of Computer Science and Engineering, Dayananda Sagar College of Engineering, Bengaluru, India
[4]Department of Computer Science and Engineering, B.M.S. College of Engineering, Basavanagudi, Bengaluru, India

Article Info

Article history:

Received Nov 17, 2025
Revised Sep 19, 2026
Accepted Oct 4, 2026

Keywords:

Advanced encryption standard
Attribute-based encryption
Cloud computing
Cloud service provider
Layered patching

ABSTRACT

Cloud computing (CC) offers significant flexibility and cost efficiency for data owners (DOs) by enabling the outsourcing of intricate data management tasks from local sites to public cloud. However, encrypted-data updates introduce computational latency when key management and ciphertext modification require repeated processing, limiting the efficiency of secure cloud storage. This research proposes a novel advanced encryption standard with layered patching (AES-LP) to enhance cloud data security while minimizing computational overhead. The framework employs a multi-layered patching (LP) strategy that applies direct updates to ciphertext through delta vectors, thereby enhancing scalability and preserving confidentiality. AES-LP integrates a patch authentication layer using a hash-based message authentication code (HMAC) and a patch-aware decryption layer, which enables localized changes for effective security management. The system model incorporates attribute-based encryption (ABE) as an access-control mechanism within a hierarchical cloud architecture consisting of DOs, data users (DUs), attribute authorities (AAs), and the cloud service provider (CSP). The results illustrate that the proposed AES-LP achieved an encryption time of 3.50 s and memory utilization of 360 KB for a 100 KB data block, which performs well than the evaluated Rivest-Shamir-Adleman (RSA), elliptic curve cryptography (ECC), and AES-ECC-MHT-based approaches under the harmonized experimental conditions.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Harish Kumar Narayanaswamy
Department of Computer Science and Engineering, BMS Institute of Technology and Management
Yelahanka, Bengaluru-560119, India
Email: harishkumar@bmsit.in

1. INTRODUCTION

Cloud computing (CC) security refers to the processes, technologies, and policies designed to protect data, applications, and resources within cloud-based infrastructures [1]. A cloud service provider (CSP) delivers these services to end users. CC offers scalable storage capabilities, enabling organizations in domains such as healthcare, finance, manufacturing, retail, education, and government to manage and store sensitive data in cloud environments [2]. To reduce operational costs, CSPs provide services at different levels, including presentation, network, and data layers [3]. Cloud infrastructures support effective management through virtualization and storage technologies, which contribute to next-generation data centers [4]. Owing to its ability to provide resources cost-efficient, on-demand scalability, and unlimited storage, CC has rapidly

expanded within the IT industry. Consequently, it has become a significant innovation, attracting the interest of technologists worldwide [5], [6]. This cloud-based platform integrates various layers, from physical hardware and network connectivity to software and application interfaces, into a unified operational model [7], [8]. With the growing number of end-user devices, volume of data transferred to cloud has increased rapidly, leading to challenges such as network congestion and delays in CC environments [9]. The computing industry has developed strategies to mitigate these challenges. One of the primary approaches to improve cloud security is the application of cryptographic technologies, which are key elements of cloud security [10]. Virtual machines enable the dynamic allocation and management of virtualized resources, thereby contributing to a secure and distributed computing infrastructure [11]. Hybrid cryptographic methods that combine various encryption techniques further improve security. Elliptic curve cryptography (ECC) is a form of public-key cryptography based on elliptic curve, which generates fast, efficient, and compact encryption keys [12], [13]. In addition to cryptographic algorithms, the strength and length of cryptographic keys play a significant role in ensuring robust security [14]. A compromised key directly affects the security and privacy of cloud data, reduces user trust, and can lead to significant organizational consequences [15].

However, many systems fall short due to insufficient emphasis on security metrics and the inconsistent implementation of protective measures. Furthermore, traditional blockchain frameworks suffer from limitations such as high energy consumption and poor scalability, which affect their integration into modern cloud environments [16]–[18]. Sundar *et al.* [19] proposed a quantum cryptography-based cloud security model (QC-CSM) for secure cloud data storage and access. QC-CSM employed a quantum key distribution protocol (QKDP) to share secret keys among parties. Attribute-based encryption (ABE) was applied to assure the security of shared data in cloud. Data access was achieved through keys generated via secure quantum channels. Furthermore, ABE was applied for key generation, which allowed secure data sharing and decryption at the client end. Kousalya and Baik [20] developed an improved Rivest-Shamir-Adleman (RSA)-based role-based access control (RBAC) model with extensible access control markup language (XACML) for data encryption. This provided information storage using cryptographic attributes within a basic admission-management mechanism. To secure sensitive data, encryption combined a conventional homogeneous process with unbalanced data distribution. This hybrid technique allowed users to retrieve data securely. Shrivastava *et al.* [21] developed a hybrid lightweight elliptic ElGamal-based encryption (HLEEE) scheme for cloud security. Key generation was accomplished using ECC, and key selection was optimized with flamingo search optimization (FSO). Public and private keys were determined using FSO, while data encryption was carried out with the ElGamal scheme. An optimal hash was generated using SHA-256 in blockchain, and hash value, encrypted data, and timestamp were stored in each block to enhance security. Mohammed *et al.* [22] presented a cloud-based data security system (C-DSS) using a five-tiered trust model for cloud data sharing. The system selected the required trust levels and applied cyber threat intelligence (CTI) sanitization strategies. Depending on the organization's confidence level, this approach was executed by a CSP or an end device. C-DSS prevented unauthorized access, disclosure, and modification caused by breaches, exfiltration, and fraudulent activities.

Gangadharaiah and Shrinivasacharya [23] suggested a hybrid scheme combining advanced encryption standard (AES), ECC, and a Merkle hash tree (MHT) for secure and efficient cloud data. Hybrid AES-ECC was used for key generation and encryption, producing both private and public keys. Encrypted user data were divided into blocks, and encrypted blocks was ensured using SHA-256 to generate hash values. The CSP created an MHT with these hash values to verify and restore user data integrity. Tang *et al.* [24] introduced a secure and lightweight ciphertext-policy ABE (CP-ABE) scheme for access control and key management. Private cloud server managed keys and enforced access control, while the public cloud server ensured secure data storage. CP-ABE provided mutual zero-interaction verification among private and public cloud servers, thereby using ElGamal encryption to ensure data unforgeability. Security assessment ensured that CP-ABE provided both data security and integrity. From this analysis, existing cloud data security approaches face several limitations. Quantum-based models suffer from limited infrastructure availability. RSA-based models lack scalability due to higher encryption time with large data block size. Hybrid methods such as ECC and ElGamal introduce greater computational overhead. Trust-based systems lack encryption standards, exposing data to potential threats, while MHT-based schemes require higher memory and processing time as the data volume increases. This research was motivated by the need to address the limitations of governance models and strengthen security in cloud data access. It leads to improve data protection by integrating innovative technologies into a unified, efficient, and scalable security framework. However, traditional cryptographic requires full re-encryption for minor data updates which enhances the computational overhead and latency. This limitation is significant in dynamic cloud environments where encrypted data suffers frequent modifications which resultant in poor scalability and inefficient performance for real-time cloud data storage. Existing methods lacks authenticated patch mechanisms and patch-aware decryption which are required for efficient and secure modification of encrypted cloud data. The AES-based model enables secure ciphertext

patching with authentication and patch-aware decryption for dynamic cloud data updates. The advanced encryption standard with layered patching (AES-LP) is specifically relevant to dynamic cloud environment where encrypted data go through the frequent updates. For example, the health records require secure modification of patient data, multimedia editing platform update encrypted images and IoT data streams generate continuous encrypted updates. The localized ciphertext patching in AES-LP provides effective and secure handling of real-time modifications. The proposed AES-LP provides scalable and efficient for secure cloud data storage, specifically in dynamic environments where encrypted data requires frequent updates with reduced computational overhead. The proposed model enables controlled data access by attribute-based authorization while secure encryption, patching, and decryption are handled by AES-LP. The major contributions of this research are as follows:

- AES-LP is proposed in this research which introduced a delta-based ciphertext patching mechanism that enables localized encrypted updates without full re-encryption thereby reducing computational overhead, encryption time, and latency in dynamic cloud environments.
- Hash-based message authentication code (HMAC) based patch authentication layer is integrated to each delta update with version and sequence metadata thereby ensuring integrity, preventing replay, reordering attacks, and enabling secure ciphertext modification.
- The framework integrates patch-wise decryption with ABE based access control, where ABE manages user authorization while AES-LP performs efficient ciphertext-level updates.

The remaining of this paper is arranged as follows: section 2 describes the proposed method in detail including system model and algorithms. Section 3 presents results analysis along with comparative analysis and discussion, and section 4 concludes this research with future work.

2. PROPOSED METHOD

The proposed secure cloud data storage framework based on a cryptographic scheme includes four primary entities: attribute authorities (AAs), data owners (DOs), data users (DUs), and a CSP. AAs consist of central and distributed authorities that act as trusted servers. They are responsible for system initialization, generating public parameters, issuing private keys based on user attributes, and managing key updates and revocations. DOs encrypt sensitive files according to access control policies and upload them securely to the cloud. DUs retrieve and decrypt files if their attributes match the access policy. Their decryption keys are updated through AAs. The CSP provides storage and file delivery services but cannot access plaintext data, thereby ensuring confidentiality.

ABE is applied to the distribution of the symmetric AES key to the authorized recipients and not to the actual encryption of the data. When a valid user retrieves the AES key, all file changes, patching, and checking processes are completed only with the help of symmetric cryptography. That is, ABE has no impact on the performance of encryption, patching, or the decryption of the normal operations but affects the access control layer. That is why the performance evaluation will consider the time and latency of AES-LP execution and the use of memory: ABE will play its role only in user onboarding, revocation, or refresh of key events. This division is in accordance with the usual hybrid encryption custom in safe storing of the clouds. In the model, therefore, ABE is to provide policy-based user authorization and not to enhance efficient computation. The experiments thus only portray the performance-critical symmetric cryptographic operations.

The CSP is considered to semi-trusted entity which stores and transfers encrypted data but is incapable of altering the encrypted data or producing valid patches. Patch verification is done by the DO using the authentication information that is attached to each patch. Patches are only accepted that are of the appropriate file version and patch sequence number. This eliminates confusion as to the responsibility of validation. The DO encrypts data, generates delta patches and uploads ciphertext to the cloud. The DU retrieves encrypted data and applies validated patches during decryption, AAs manage attribute-based key generation, distribution, and revocation for authorized users. The CSP gathers ciphertext and patch updates but does not access plaintext to ensure confidentiality of cloud data. ABE is utilized for secure distribution of symmetric AES key and execution of access control policies. The AES-LP perform encryption, delta-based patching, authentication, and patch-aware decryption operations individually. Therefore, ABE handles user authentication while AES-LP focuses on effective ciphertext-level updates and secure data modification. The interactions among these entities are illustrated in Figure 1.

- System parameter distribution and user private key generation

In the initialization phase, the central authority (CA) and AAs collaboratively executes cryptographic algorithms to generate public system parameters. These parameters define encryption, decryption, and key management processes. Once generated, the parameters are distributed to DOs and the CSP, ensuring that all entities operate securely within the framework. AAs generate private decryption keys for DUs based on their

attribute sets. These keys are securely delivered to users, enabling access to encrypted files if their attributes meet the data-access policy. This ensures fine-grained access control, allowing only authorized users to decrypt data.

– User revocation and key update broadcast

To manage dynamic access, the system maintains a revocation list that tracks users who lose authorization. When a user is revoked, their identity is added to the list, preventing future access even if they retain an old private key. AAs generate and broadcast key updates across the cloud environment. These updates allow only non-revoked users to renew their decryption abilities, thereby maintaining secure access to the encrypted data.

– File encryption and upload and decryption key generation

DOs encrypt files using public system parameters and define access policies based on user attributes. The encrypted files are then uploaded to the CSP for storage. Since encryption is performed before uploading, the CSP cannot access the data. Authorized DUs combine their attribute-based private keys with the most recent key update to generate a temporary decryption key. This key remains valid only for a limited period and is regenerated during each update cycle, ensuring time-bound access control.

– File decryption and attack prevention

Using the temporary decryption key, DUs download and decrypt encrypted files from the CSP. Decryption succeeds only if the user's attributes satisfies DO-defined access policy. This ensured unauthorized users are denied access. The system incorporates mechanisms to defend against security threats in cloud environments. It mitigates attempts by malicious users to combine attribute sets for unauthorized access. To minimize key exposure risks, the system utilizes frequent key updates and randomized cryptographic parameters, thereby enhancing the robustness of key management.

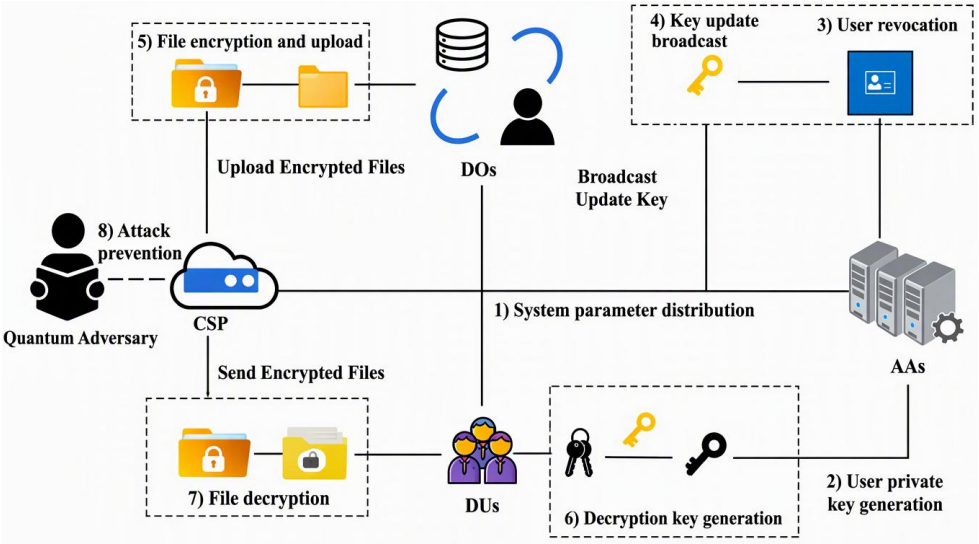


Figure 1. System model for cloud data security

2.1. Encryption standard with layered patching

AES is selected because it is a fast and secure symmetric encryption method which reduces computational overhead at repeated data updates. In the proposed approach, the DO computes delta vectors locally through recognizing the modified portions and produce difference among actual and updated data. The delta vector is transmitted as an authenticated ciphertext-update representation together with the required version and sequence metadata; the CSP stores the update without possessing the decryption key. The novelty of AES-LP does not arise from modifying the AES cipher, introducing a new AES operating mode, or claiming XOR-based delta computation as a new cryptographic primitive. Instead, the contribution lies in the system-level integration of three mechanisms for dynamic encrypted cloud storage: i) localized delta-based ciphertext updating, ii) HMAC-authenticated patch metadata containing file-version and sequence information, and iii) patch-aware selective decryption combined with ABE-based access control. The version and sequence information binds each patch to the expected ciphertext state and provides protection against replay, reordering, and rollback-related inconsistencies. Thus, AES-LP distinguishes itself from conventional hybrid encryption frameworks by focusing specifically on authenticated, version-sensitive ciphertext updates rather than only on initial encryption, key exchange, or integrity verification.

The AES-LP utilizes AES in counter (CTR) mode with 128-bit key for symmetric encryption. The delta vectors are generated to represent localized data modifications and each patch is authenticated using HMAC. The ABE is utilized for access control and key distribution which ciphertext patching operations are handled by AES-LP. The patch metadata contains file identifier, version number, and patch sequence index to ensure correct ordering and integrity verification. The layered design provides an explicit balance between update efficiency and security assurance. Delta-based patching reduces unnecessary processing for localized modifications, whereas HMAC authentication and version-aware metadata introduce additional processing and storage requirements. These additional operations are intentional because a ciphertext update mechanism without integrity and state validation could allow unauthorized modification, replay, or reordering of patches.

a. Substitution (SubBytes)

SubBytes is a nonlinear transformation applied to each byte in 4×4 matrix using a substitution box (S-box). The AES S-box is based on multiplicative inverse and transformation in finite field ($GF2^8$), which introduces nonlinearity and withstands algebraic attacks. This process introduces confusion, a core cryptographic principle, by substituting the relationship between ciphertext and plaintext. Both encryption and decryption perform the same S-box.

b. ShiftRows

ShiftRows cyclically shifts row of the state matrix to left. The first row is unchanged, second row shifts one byte, third shifts two bytes, and the fourth shifts three bytes. This ensures inter-column diffusion, forcing each column to use the output of all bytes in subsequent rounds. Vertical symmetry is disrupted, which increases the avalanche effect. Decryption uses right circular shifts to reverse the operation.

c. MixColumns

MixColumns considers every column of state as a polynomial over ($GF2^8$) and multiplies it through a fixed matrix modulo $x^4 + 1$. This ensures that each byte in a column influences all other bytes in the same column. The operation provides intra-column diffusion, so a change in one plaintext byte influences four ciphertext bytes after a single round. Mathematically, this is equivalent to matrix multiplication in a finite field, which enhances resistance to differential cryptanalysis. In decryption, a different matrix is used to invert the MixColumns operation and restore the original state.

d. AddRoundKey

In AddRoundKey, state matrix is combined with a round-specific subkey derived from cipher key using bitwise XOR. This is the only AES step in which the key directly influences the plaintext. It introduces key dependency, ensuring that small modifications in a key result in better differences in the ciphertext. Although simple and efficient, this step is essential because it secures each round with a unique key. It is carried out at the start of AES and in every round except the final round.

e. Key expansion

Key expansion provides round keys from the initial cipher key using RotWord, SubWord, and XOR operations. It operates on 32-bit words and produces 44 words for AES-128, which supports 10 rounds plus the initial round key. Each round incorporates a key schedule constant (Rcon), which prevents symmetry and repetition in keys. The S-box provides nonlinearity in the SubWord transformation, while recursive XOR operations with preceding words ensure diversity. This process makes each round key unique and cryptographically resistant. In the extension of [25], a hyperchaotic variable S-box was integrated into the AES architecture to support block-wise image encryption.

This research introduces a new ciphertext-based update mechanism, termed layered patching (LP). While the variable S-box design improves both key diversity and data confusion during the initial encryption process, the modified AES still suffers from a drawback common to all block-cipher systems: any alteration in the plaintext requires re-encryption of at least the affected block, and in many cases, the entire image. This inefficiency is particularly significant in cloud-based data storage and multimedia editing, where encrypted data undergoes frequent and localized changes. To overcome this limitation, the proposed AES-LP, which incorporates an additional security layer-a patching mechanism that permits direct patching of ciphertext using delta vectors, without decryption and without access to the key. The mechanism operates through three-layered stages, which is discussed as follows. The patch through ciphertext delta in layer 1 of the LP directly updates encrypted data without decryption. Instead of re-encrypting the entire data block for minor changes, only the binary difference between the old and new plaintext is estimated and applied to the ciphertext, as shown in (1):

$$\Delta = P_{new} \oplus P_{old} \quad (1)$$

where Δ denotes the patch vector and P represents the plaintext. LP significantly reduces the computational complexity and latency, specifically when changes are recurrent but localized. For each update, the DOs compares the original plaintext representation with the modified representation at the byte level. The delta vector records the locations and values of the modifications between the two versions. For unchanged positions, the delta is zero; for modified positions, the delta represents the corresponding binary difference. The DOs

performs this comparison locally before transmitting the update, so plaintext information is not exposed to the CSP. The resulting patch is associated with the file identifier, target version, and patch sequence index.

Before application, the patch metadata and delta content are authenticated using HMAC-SHA-256. A patch is accepted only when its authentication tag is valid and its version and sequence information corresponds to the expected ciphertext state. This procedure provides deterministic patch identification and prevents an old or incorrectly ordered patch from being applied to an incompatible file version. The delta vector is determined as binary difference among original plaintext block and modified plaintext block. This vector represents localized changes and enables partial ciphertext updates without re-encrypting the entire data block. Moreover, it enhances security by removing the need to expose plaintext or regenerate keystreams during updates, thereby improving the scalability and security of cloud storage. The next layer of the LP is the patch authentication layer, which protects data against unauthorized ciphertext alternations. By integrating a HMAC into each patched block, the model ensures that every modification is traceable and authenticated. This validates the source and origin of the patch and facilitates the creation of immutable audit logs. To ensure authenticity and integrity, each delta patch is based on authentication tag generated using HMAC. The authentication fixes the patch with metadata including file identifier, version number, and patch sequence index. Every patched block is authenticated using HMAC, as shown in (2):

$$HMAC_i = HMAC_K\left(C_i' || timestamp_i || source_i\right) \quad (2)$$

The authentication tag encloses the file identifier, current version of the file, patch sequence number, and delta update itself. This metadata ensures that any unauthorized change, replay of the old patches, or patch reordering will be detected. Therefore, the authentication does not only ensure the integrity of the data, but also controls the order of patches as well as rollback attacks. The last layer is patch-aware decryption, in which updated ciphertext is decrypted without re-encrypting entire block. When a patch is applied to a ciphertext block through a delta update, the correct plaintext is recovered by XORing the patched ciphertext with the keystream. Since encryption applies only to data, the decryption process incorporates changes without requiring explicit knowledge of the patch. Patch-aware decryption creates the updated plaintext by applying the delta-modified ciphertext to same keystream produced during encryption which enables selective decryption of modified blocks. The process uses the same keystream generator and key, as shown in (3):

$$B_i' = AES_{K,S_i}^{-1}(C_i') = Keystream_i \oplus C_i' \quad (3)$$

Thus, the patch is reflected in the plaintext without requiring re-encryption of the entire data. The AES-LP lies in providing secure and authenticated ciphertext-level updates without providing plaintext encryption keys. Unlike conventional AES-based system which require re-encryption of whole blocks with minor data changes, the AES-LP utilizes delta-based ciphertext patching that reduces computational overhead. The integration of patch authentication layer using HMAC ensures that every patch is integrity-protected and controlled which prevents unauthorized modification and reordering of updates. Furthermore, the patch-aware decryption enables selective decryption of affected ciphertext blocks, enhancing scalability and ensuring that the model is efficient for dynamic cloud data where frequent updates occur. This enables localized updates to be applied precisely, preserving both data integrity and confidentiality. Only the affected blocks are accessed and decrypted, thereby reducing computational complexity, memory usage, and latency. Furthermore, patch-aware decryption improves the scalability of encrypted systems by differentiating the decryption process from the overall data size.

The decryption process in AES-LP is designed to efficiently retrieve plaintext from either original or patched ciphertext blocks by using AES in CTR. In this mode, each ciphertext block is generated by XORing the plaintext with keystream derived from the encryption key and a unique counter. AES-LP utilizes this to support patch-aware decryption, where updates applied to ciphertext as delta vectors are reflected in the decrypted output. During decryption, the same keystream is regenerated using the identical key and counter, and XOR is performed with the patched ciphertext block. This allows AES-LP to retrieve the updated plaintext without requiring the original plaintext or re-encrypting the dataset. Therefore, only the modified blocks are decrypted, enhancing scalability, particularly in a system where encrypted data is frequently updated.

To prevent keystream reuse in AES-LP, is used on each version of the file based on the file identifier and version number. The counter is worked up on an encrypted block. Consequently, every pair is utilized only once and the reuse of the keystream across updates is avoided. The patches also contain a version id and a patch sequence index. In case an attacker tries to reuse or rearrange patches, the version and index values will not correspond to the desired state, and a patch will be rejected. This novelty of the work is not based on the modification of AES or the development of a new encryption mode. Rather, it is in the empowerment of secure, authenticated, and version-sensitive ciphertext patching without plaintext exposure. Though XOR-based delta

updates are theoretically known, these are insecure without patch ordering, version control, and integrity protection. The scheme being proposed has version numbers, patch sequence indices, and signed metadata binding to guarantee that each patch is only applied to the state corresponding ciphertext. Every update also carries a version identifier and an index number such that no patch is accepted as old or out of order. The authentication of each patch also encompasses the contextual information of the patch such that a change cannot be forged, reordered, removed, or replayed.

The patch authentication was implemented using HMAC with SHA-256 as the hash function. Each patch contains file identifier, version number, and sequence index which counter values were derived from file identifier and version number to avoid keystream reuse. The AES-LP utilizes AES in CTR mode to support independent block updates and effective keystream generation. The patch authentication is utilized using HMAC with SHA-256 as hash function which ensures secure, scalable, and lightweight ciphertext-level updates.

Pseudocode for the proposed AES-LP:

```
Input: Plaintext P, Secret Key K, Access Policy A, Attributes S, Delta Vector Δ
Output: Patched Ciphertext C', Decrypted Plaintext P'
1. System Initialization:
    Params ← Setup(K,A) for generating system parameters
    RoundKeys ← KeyExpansion(K) for AES key expansion
2. Encryption:
    For each block B_i in P:
        B_i ← SubBytes(B_i) for byte substitution
        B_i ← ShiftRows(B_i) for row shifting
        B_i ← MixColumns(B_i) for column mixing
        B_i ← AddRoundKey(Bi, RoundKeys) for add round key
   C ← {B_1, B_2, ..., B_n} for encrypted ciphertext blocks
3. Layered Patching (Layer 1):
    If update ΔP is detected:
        Compute patch vector using (1)
        Attach version ID and sequence number
        Apply delta patch
4. Patch Authentication (Layer 2):
    Generate HMAC using (2)
    Store patch metadata
5. Patch-Aware Decryption (Layer 3):
    Verify HMAC
    Verify version ID and sequence order
    If mismatch → Reject patch
    Else → Apply patch and decrypt using (3)
6. Access Control and Revocation:
    If S satisfies A:
        Grant decryption access with PrivateKey(S)
    Else:
        Deny access and Broadcast key update if revoked
Return C', P'
```

3. RESULTS AND DISCUSSION

The proposed AES-LP framework was implemented in Python 3.10 using PyCryptodome 3.20 for AES-CTR encryption and HMAC-SHA-256 computation. For each encrypted block, the 128-bit AES-CTR counter is constructed as FileID (32 bits), BlockIndex (64 bits), thereby providing a unique counter value for each patch under a given encryption key. The HMAC authentication key is derived separately from the AES encryption key using HKDF-SHA256, with the AES key as the input keying material (IKM). The RSA baseline employs RSA-1024 with OAEP padding using SHA-256, while the ECC baseline employs the NIST P-256 (secp256r1) curve. The post-quantum cryptography (PQC) baseline uses ML-KEM-512, standardized in FIPS 203 and historically referred to as CRYSTALS-Kyber-512. The same cryptographic library was used to implement the RSA and ECC baselines, while liboqs-python 0.10 was used for the PQC baseline. Across the evaluated block sizes, the measured patch size was approximately 5% of the corresponding data block, with observed values ranging from approximately 4% to 9%. This localized-update configuration reduces the amount of data requiring cryptographic processing compared with full-block re-encryption, thereby contributing to the lower computational overhead observed for AES-LP.

The data were uniformly distributed by synthetic binary data block randomly with size of 1 KB to 200 KB to simulate dynamic cloud storage updates. The use of synthetic binary data provides controlled and reproducible input sizes for measuring cryptographic processing time and memory consumption; however, it does not capture all characteristics of real-world cloud workloads. All baseline methods were implemented using identical data block sizes, hardware configuration, and evaluation parameters to ensure a consistent comparison.

The experiments are repeated over multiple runs and the average values with standard deviations (SDs) were estimated to ensure statistical reliability of the results. The proposed model is applicable to real-time data types such as text files, multimedia content, and system logs, since AES-LP operates at the ciphertext block level independent of data semantics. The synthetic data were utilized to ensure reproducibility and controlled evaluation of cryptographic performance. The AES-LP mitigates various attack scenarios such as replay attacks are prevented using version identifiers and patch sequence numbers embedded in authenticated metadata. The rollback attacks are avoided by validating patch before application. The selected ciphertext attacks are mitigated as delta patches are authenticated by HMAC and unauthorized modifications are rejected. The confidentiality is ensured by AES-128 in CTR model while integrity is maintained by HMAC to detect tampering.

ABE provides secure access control and key distribution thereby ensuring only authorized user attain decryption keys. The replay and reordering attacks are prevented by versioning and sequence-based metadata. The following performance metrics were evaluated: encryption time, decryption time, response time, key generation time, latency, and memory utilization. Data block size of 1 KB, 10 KB, 50 KB, 100 KB, and 200 KB were used for performance assessment. In this research, the block size refers to the total data size processed in each experiments and not the internal AES block sized used for encryption. AES operates on fixed 128-bit block size which is basic unit of encryption and decryption in all experiments. The larger data sizes are divided into multiple 128-bit blocks and sequentially processed in this research. Here, the encryption time estimates the duration required to encrypt data blocks; decryption time presents the time for plaintext recovery; response time denotes total processing delay; latency denotes to update propagation delay; key generation time presents cryptographic key computation time; and memory utilization estimates peak memory consumed during execution. All the experiments are conducted on standalone system simulating cloud storage operations such as encryption, updates and access control. AES-LP operates at algorithmic level and the performance is depended on computational efficiency rather than infrastructure thereby the results are generalized to cloud environments.

Table 1 specifies the configuration of the evaluated cryptographic methods under the harmonized experimental protocol. To ensure a fair and reproducible comparison, all cryptographic methods were evaluated using the same data block sizes, hardware environment, operating system, and performance-measurement procedure. The experiments considered data sizes of 1 KB, 10 KB, 50 KB, 100 KB, and 200 KB. The proposed AES-LP employed AES-128 in CTR mode with a 128-bit symmetric key, HMAC-SHA-256 for patch authentication, and delta-vector-based ciphertext patching with version and sequence metadata. The baseline methods, including AES, RSA, ECC, and PQC, were evaluated under the same computational environment without introducing the LP mechanism.

Table 1. Configuration of the evaluated cryptographic methods under the harmonized experimental protocol

Method	Cryptographic configuration	Input data size (KB)	Key/security configuration	Implementation/ processing	Hardware and software environment	Evaluation protocol
AES	Standard AES symmetric encryption	1, 10, 50, 100, 200	AES symmetric key; key size of AES-128 bits	Conventional AES encryption and decryption without layered ciphertext patching	Windows 10; Intel Core i5 processor; 8 GB RAM; Python	Same input sizes and hardware
RSA	RSA public-key encryption	1, 10, 50, 100, 200	RSA public/private key pair; key size of RSA-1024 bits	Conventional RSA-based encryption and decryption	Windows 10; Intel Core i5 processor; 8 GB RAM; Python	Same input sizes and hardware
ECC	ECC	1, 10, 50, 100, 200	ECC public/private key pair; key size of ECC-256 bits	Conventional ECC-based cryptographic processing	Windows 10; Intel Core i5 processor; 8 GB RAM; Python	Same input sizes and hardware
PQC	PQC	1, 10, 50, 100, 200	PQC key pair; key size of PQC-128 bits	Conventional PQC-based cryptographic processing	Windows 10; Intel Core i5 processor; 8 GB RAM; Python	Same input sizes and hardware
Proposed AES-LP	AES-128 in CTR mode+HMAC-SHA-256+delta-vector-based LP	1, 10, 50, 100, 200	128-bit AES symmetric key; HMAC-SHA-256 for patch authentication; file ID, version number, and sequence index for patch management	Localized ciphertext update using delta vectors, HMAC-authenticated patching, version/sequence validation, and patch-aware decryption	Windows 10; Intel Core i5 processor; 8 GB RAM; Python; NumPy and time libraries; no GPU or parallel optimization	Same input sizes and hardware

Figures 2 and 3 present the encryption and decryption times, respectively, for AES-LP compared with existing algorithms such as PQC, RSA, ECC, and AES. AES-LP achieved improved encryption times of 0.06 s, 1.25 s, 2.90 s, 3.50 s, and 5.65 s for data block size of 1 KB, 10 KB, 50 KB, 100 KB, and 200 KB, respectively. Corresponding decryption times were 0.04 s, 0.60 s, 2.75 s, 3.40 s, and 5.20 s. These improvements were achieved due to the LP mechanism, which avoids reprocessing entire ciphertext blocks when only partial data modifications occur. Furthermore, offloading decryption steps to the layered structure eliminates redundancy, making AES-LP more efficient for resource-constrained environments requiring rapid cryptographic operations without compromising security.

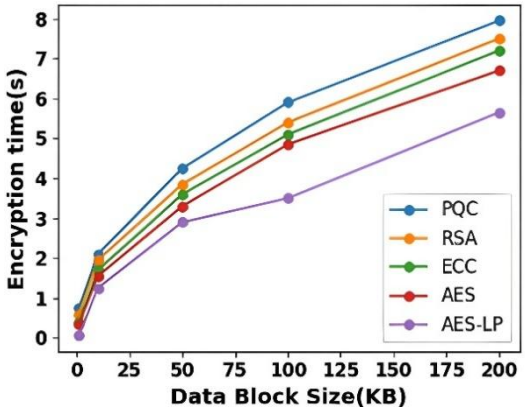


Figure 2. Performance of encryption time (s) with various data block size showing AES-LP reduces processing overhead for partial ciphertext updates

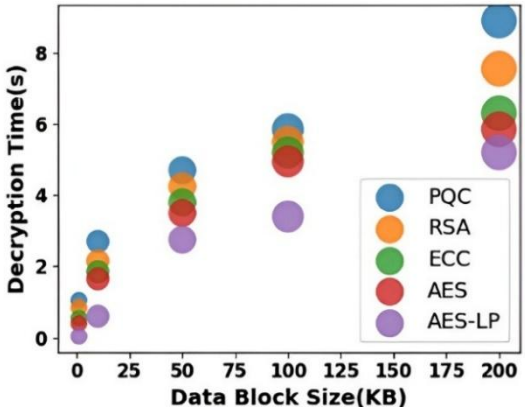


Figure 3. Performance of decryption time (s) with various data block size demonstrating AES-LP efficiently processes modified ciphertext blocks with lower latency

Figures 4 and 5 show the response time and key generation time, respectively. AES-LP achieved response times of 0.28 s, 1.40 s, 3.10 s, 4.60 s, and 6.40 s for the selected data block size, while key generation times were 0.22 s, 1.10 s, 2.60 s, 3.80 s, and 5.30 s. These improvements are attributed to the LP technique, which reduces both response and key generation times, thereby enhancing suitability for resource-constrained environments.

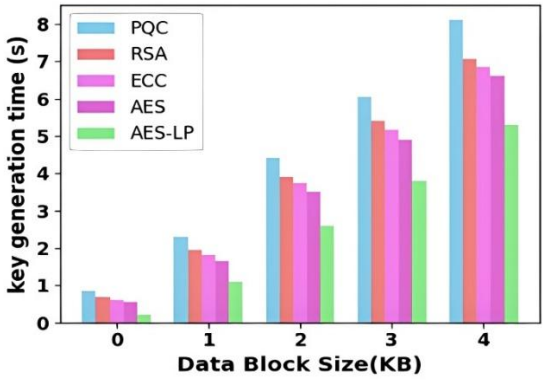


Figure 4. Performance of response time (s) with various data block size illustrating improved responsiveness of AES-LP due to layered ciphertext patching mechanism

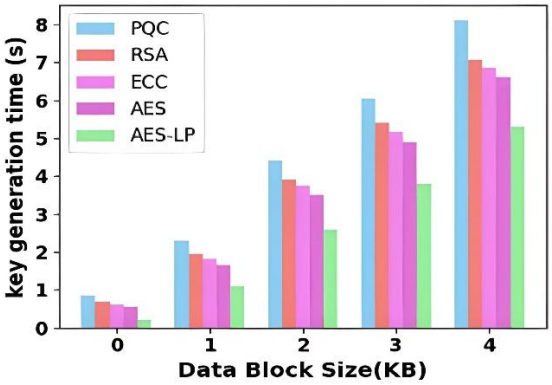


Figure 5. Performance of key generation time (s) with various data block size indicating AES-LP reduces computation through lightweight layered update strategy

Figures 6 and 7 show latency and memory utilization, respectively. AES-LP achieved latencies of 0.35 s, 1.50 s, 3.25 s, 4.80 s, and 6.45 s for data block size of 1 KB, 10 KB, 50 KB, 100 KB, and 200 KB. Corresponding memory utilization values were 30 KB, 100 KB, 240 KB, 360 KB, and 500 KB. These results

ensures that the layered structure reduces processing delays and avoids unnecessary reprocessing, thereby lowering latency. Additionally, the lightweight memory design and optimized patching, make AES-LP effective in resource-constrained environments.

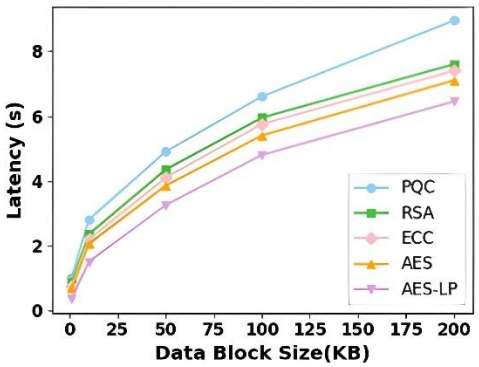


Figure 6. Performance of latency (s) with various data block size showing AES-LP minimizes delays for dynamic cloud updates

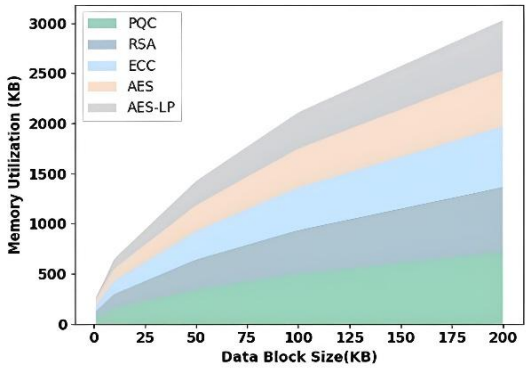


Figure 7. Performance of memory utilization (KB) with various data block size showing AES-LP reduces storage overhead using localized ciphertext patching approach

Table 2 presents the results of mean±SD attained from five independent experimental runs to estimate the consistency and stability of each model. The lower SD values denote that the performance of AES-LP remains stable across repeated executions. The AES-LP achieves lower values for encryption time, decryption time, response time, key generation time, latency, and memory utilization but smaller variations compared to baseline methods. This ensures that the proposed LP mechanism provides reliable performance under the same experimental condition.

Table 3 presents the performance stability analysis of proposed AES-LP for a 100 KB data block size. The mean and SD values are attained from multiple experimental runs (n=10) and 95% confidence interval (CI) is estimated to calculate result consistency. The CI indicates that AES-LP provides stable and reliable performance among repeated executions. The stability is achieved due to deterministic delta-based ciphertext patching and patch-aware decryption mechanisms which reduces variability in processing time and memory usage. The scalability results ensure that the proposed model maintains consistent efficiency under repeated runs thereby showing its suitability for practical cloud storage environments.

Table 2. Average performance with SD for data block size of 100 KB across different cryptographic methods

Metrics	AES	RSA	ECC	PQC	Proposed AES-LP
Encryption time (s)	4.10±0.08	8.45±0.15	5.20±0.10	6.80±0.12	3.50±0.05
Decryption time (s)	3.95±0.07	12.60±0.20	5.80±0.11	7.15±0.14	3.40±0.04
Response time (s)	5.20±0.10	15.30±0.25	7.40±0.15	9.90±0.18	4.60±0.06
Key generation time (s)	4.05±0.05	11.20±0.18	4.90±0.09	8.30±0.15	3.80±0.05
Latency (s)	5.40±0.12	16.80±0.30	7.90±0.16	10.5±0.20	4.80±0.07
Memory utilization (KB)	480±8	1240±20	610±12	920±15	360±3

Table 3. Performance stability analysis of proposed AES-LP framework for 100 KB data block size showing mean, SD, and 95% CI across repeated runs (n=10)

Metric	Mean	Std. dev	Correct 95% CI
Encryption time (s)	3.5	0.08	±0.05
Decryption time (s)	3.4	0.07	±0.04
Response time (s)	4.6	0.1	±0.06
Key generation time (s)	3.8	0.09	±0.06
Latency (s)	4.8	0.11	±0.07
Memory utilization (KB)	360	6	±3.72

3.1. Comparative analysis

A comparative analysis was performed between AES-LP and three existing approaches: QC-CSM [19], improved RSA with RBAC-XACML [20], and AES-ECC with MHT [23]. Table 4 shows the performance comparison for a data block size of 100 KB. Metrics include encryption time, decryption time, response time, key generation time, latency, and memory utilization.

Table 4. Comparison of proposed AES-LP and existing methods for the data block size of 100 KB

Metrics	QC-CSM [19]	Improved RSA and RBAC with XACML [20]	AES-ECC MHT [23]	Proposed AES-LP
Encryption time (s)	5.2	4.8	4.3	3.5
Decryption time (s)	5	4.5	4.1	3.4
Response time (s)	6.8	6.4	5.6	4.6
Key generation time (s)	6.1	5.7	5.1	3.8
Latency (s)	7.3	6.9	6.2	4.8
Memory utilization (KB)	520	460	400	360

AES-LP outperformed the existing metrics, achieving an encryption time of 3.50 s, decryption time of 3.40 s, response time of 4.60 s, key generation time of 3.80 s, latency of 4.80 s, and memory utilization of 360 KB. The existing models were re-simulated under the same conditions as the proposed method to ensure a fair evaluation. These results demonstrate the improved efficiency and lightweight of AES-LP for secure cloud data storage.

3.2. Discussion

AES-LP addresses operational inefficiencies and security gaps in traditional models to provide improved cloud data encryption. Unlike conventional block ciphers, which require complete re-encryption for even minor changes, the patching mechanism in AES-LP supports partial updates via delta vectors. This feature is especially valuable in dynamic cloud environments where small and frequent modifications occur. The integration of the HMAC strengthens security, thereby enabling verifiable data updates in the cloud without compromising integrity. The patch-aware decryption mechanism further reduces computational memory overhead by processing only updated data blocks during decryption. The adaptability AES-LP across various data block size highlights its effectiveness in large-scale encryption tasks. The layer-wise separation reduces latency without compromising encryption integrity. The observed performance improvement is primarily associated with the separation of full-data encryption from localized update processing. Conventional encryption approaches process the complete data representation when an update is applied, whereas AES-LP identifies the modified portion and represents the change using a delta vector. Consequently, unchanged ciphertext does not need to undergo the same processing during a localized update. The patch-aware decryption mechanism further limits processing to the affected ciphertext blocks, while HMAC authentication adds integrity verification to each patch. Therefore, the principal computational benefit originates from localized delta-based processing and selective decryption, whereas HMAC introduces a relatively small additional authentication operation. ABE contributes to access control and key distribution rather than to the measured symmetric encryption and patching workload. There is a security-efficiency trade-off associated with the layered design. The use of HMAC, version identifiers, and sequence indices introduces additional metadata and authentication processing and therefore creates some memory and computational overhead. However, this overhead is exchanged for stronger control over patch integrity, ordering, and replay-related attacks. The measured memory utilization of 360 KB for the 100 KB data block reflects this layered processing and associated metadata. Thus, the improvement should not be interpreted as eliminating all cryptographic overhead; rather, AES-LP shifts the computational cost from repeated full-data processing toward localized update processing with explicit authentication and version control.

The advantage of AES-LP becomes more relevant when modifications affect only a limited portion of a larger encrypted object. Hence, the principal use case for AES-LP is dynamic encrypted storage characterized by repeated partial modifications rather than complete file replacement. The role of ABE is separated from the performance-critical AES-LP operations. ABE is responsible for attribute-based authorization, key distribution, and revocation, whereas AES-LP performs symmetric encryption, ciphertext-level patching, HMAC-based patch authentication, and patch-aware decryption. Consequently, the encryption, decryption, response-time, latency, and memory measurements primarily represent the computational behavior of AES-LP rather than the overhead of ABE. This separation is intentional because the proposed framework uses hybrid cryptography in which public-key-based access control protects the symmetric key while symmetric cryptography handles the actual data processing. The performance improvement occurs as AES-LP processes only modified ciphertext compared to RSA, ECC, and

AES-ECC-MHT which require complex block processing. The similar efficiency enhancements for partial encryption updates are discussed in [23] without authenticated ciphertext patching. As the patch updates are applied at ciphertext block level and authentication independently, concurrent updates from multiple authorized users are processed sequentially without affecting data integrity thereby supporting scalable cloud environments with frequent updates. Security analysis shows that AES-LP attacks replay and rollback attacks through version identifiers and patch sequence numbers embedded within the HMAC-authenticated metadata. The patch authentication layer ensures integrity, while patch-aware decryption accepts validated updates thereby enhancing resistance against tampering attacks. These enhancement supports the objective of allowing lightweight cloud updates as delta-based patching avoids full re-encryption while patch-aware decryption process only modified blocks. The higher memory usage is due to the storage of delta patches, version identifiers and HMAC authentication tags. The additional memory overhead provides effective ciphertext-level updates and removes full re-encryption thereby resultant in enhanced overall model performance.

From an engineering perspective, AES-LP separates confidentiality, access control, and dynamic update management. AES-128-CTR provides efficient encryption, ABE enables attribute-based authorization and key distribution, while delta-based patching supports localized ciphertext updates. HMAC-SHA-256 authenticates each patch, and version and sequence information ensure state-aware update management. This separation reduces unnecessary processing of unchanged encrypted data during localized updates.

Although AES-LP is designed with lightweight and dynamic cloud-storage scenarios in mind, the present study does not constitute a deployment benchmark on a constrained or distributed cloud platform. The experiments were conducted using synthetic binary data on a standalone Windows 10 system with an Intel i5 processor and 8 GB RAM, without GPU or parallel optimization. Therefore, the reported reductions in processing time and memory utilization demonstrate algorithm-level efficiency under controlled conditions rather than guaranteed performance on specific cloud infrastructures. In practical deployment, factors such as network bandwidth, storage latency, virtualization overhead, concurrent users, cloud-provider architecture, and distributed synchronization may influence the observed performance. Moreover, actual cloud files may exhibit different redundancy, update locality, file-size distributions, access patterns, compression behavior, and concurrent-access characteristics. Consequently, the present results should be interpreted as controlled algorithm-level measurements rather than direct predictions of performance for specific production cloud workloads. Evaluation using real-world datasets and representative cloud traces is required to determine how these characteristics influence AES-LP under practical operating conditions. The study does not include a formal cryptanalytic security benchmark, penetration testing, distributed scalability experiment, or large-scale concurrent-user evaluation. These evaluations are important for establishing security and scalability under practical deployment conditions and are therefore considered directions for future work. A comprehensive scalability and resource-consumption assessment on distributed cloud and resource-constrained platforms is therefore identified as future work.

4. CONCLUSION

This research proposed a secure and scalable cloud data encryption method called AES-LP, designed to overcome the limitations of existing cryptographic models. Conventional encryption models are less effective in dynamic cloud environments because of its inefficiencies in handling partial updates, which reduces latency. Within the overall framework, ABE provides fine-grained attribute-based authorization and secure distribution of the symmetric AES key, while AES-LP performs the computationally intensive encryption, patching, authentication, and decryption operations. Therefore, the reported performance improvements are primarily attributable to the AES-LP ciphertext-processing mechanism, while ABE provides the access-control functionality required for controlled cloud data sharing. The proposed AES-LP framework combines AES-based encryption, delta-vector-driven localized ciphertext patching, HMAC-based authentication, and ABE-based access control for secure dynamic cloud data management. Delta vectors represent only the modified portions of data, reducing redundant processing and avoiding unnecessary re-encryption of unchanged ciphertext. This makes AES-LP suitable for lightweight cloud environments where computational and memory efficiency are important. Under the evaluated experimental conditions, AES-LP achieved an encryption time of 3.50 s, decryption time of 3.40 s, response time of 4.60 s, key generation time of 3.80 s, latency of 4.80 s, and memory utilization of 360 KB for a 100 KB data block.

Compared with the evaluated baseline approaches, including RSA, ECC, and AES-ECC-MHT, AES-LP demonstrated lower computational time and memory utilization under the harmonized experimental protocol. The improvement is primarily associated with localized ciphertext processing, which avoids unnecessary reprocessing when only a portion of the stored data is modified. The experimental evidence supports the computational efficiency of AES-LP under the controlled conditions evaluated in this study. The observed advantages are demonstrated using synthetic binary data, selected data block sizes, repeated

standalone executions, and a common hardware environment. Therefore, these results should not be interpreted as direct evidence of performance superiority in every cloud deployment scenario. In particular, distributed networking, concurrent users, cloud-provider infrastructure, and real-world workload characteristics were not evaluated. Future work will address these factors through distributed cloud experiments, real-world datasets, concurrent-update workloads, formal security analysis, and broader scalability testing.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Harish Kumar Narayanaswamy	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓
Deepak Gopal		✓	✓		✓	✓		✓	✓	✓	✓	✓		✓
Annapoorna Bangalore Ramesh	✓		✓	✓	✓		✓	✓		✓	✓		✓	✓
Sarala Duggonahalli Veeresh		✓		✓	✓		✓	✓	✓		✓	✓		✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] S. Zubair and H. M. A. Ahmed, "A hybrid algorithm-based optimization protocol to ensure data security in the cloud," *International Journal of Information Technology*, vol. 16, no. 5, pp. 3057–3064, Jun. 2024, doi: 10.1007/s41870-023-01546-7.
[2] I. Qiqieh, J. Alzubi, and O. Alzubi, "DNA cryptography based security framework for health-cloud data," *Computing*, vol. 107, no. 1, p. 35, Jan. 2025, doi: 10.1007/s00607-024-01393-9.
[3] S. Ahmad and S. Mehfuz, "Efficient time-oriented latency-based secure data encryption for cloud storage," *Cyber Security and Applications*, vol. 2, p. 100027, 2024, doi: 10.1016/j.csa.2023.100027.
[4] N. M. Reddy, G. Ramesh, S. B. Kasturi, D. Sharmila, G. Gopichand, and L. T. Robinson, "Secure data storage and retrieval system using hybridization of orthogonal knowledge swarm optimization and oblique cryptography algorithm in cloud," *Applied Nanoscience*, vol. 13, no. 3, pp. 2449–2461, Mar. 2023, doi: 10.1007/s13204-021-02174-y.
[5] A. Abdo, T. S. Karamany, and A. Yakoub, "A hybrid approach to secure and compress data streams in cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 3, p. 101999, Mar. 2024, doi: 10.1016/j.jksuci.2024.101999.
[6] M. Akbar, M. M. Waseem, S. H. Mehanoor, and P. Barmavatu, "Blockchain-based cyber-security trust model with multi-risk protection scheme for secure data transmission in cloud computing," *Cluster Computing*, vol. 27, no. 7, pp. 9091–9105, Oct. 2024, doi: 10.1007/s10586-024-04481-9.
[7] S. Guan, C. Zhang, Y. Wang, and W. Liu, "Hadoop-based secure storage solution for big data in cloud computing environment," *Digital Communications and Networks*, vol. 10, no. 1, pp. 227–236, Feb. 2024, doi: 10.1016/j.dcan.2023.01.014.
[8] A. S. Alqahtani, Y. Trabelsi, P. Ezhilarasi, R. Krishnamoorthy, S. Lakshmisridevi, and S. Shargunam, "Homomorphic encryption algorithm providing security and privacy for IoT with optical fiber communication," *Optical and Quantum Electronics*, vol. 56, no. 3, p. 487, Mar. 2024, doi: 10.1007/s11082-023-06098-5.
[9] R. Agrawal, S. Singhal, and A. Sharma, "Blockchain and fog computing model for secure data access control mechanisms for distributed data storage and authentication using hybrid encryption algorithm," *Cluster Computing*, vol. 27, no. 6, pp. 8015–8030, Sep. 2024, doi: 10.1007/s10586-024-04411-9.
[10] M. Suganya and T. Sasipraba, "Stochastic Gradient Descent long short-term memory based secure encryption algorithm for cloud

data storage and retrieval in cloud computing environment," *Journal of Cloud Computing*, vol. 12, no. 1, p. 74, May 2023, doi: 10.1186/s13677-023-00442-6.
[11] T. Kamble, M. Ghuge, R. Jain, and V. S. Bodade, "Secure data transmission in cloud computing using a cyber-security trust model with multi-risk protection scheme in smart IOT application," *Cluster Computing*, vol. 28, no. 2, p. 79, Apr. 2025, doi: 10.1007/s10586-024-04847-z.
[12] S. Vinothkumar and J. Amutharaj, "A hybrid public cryptography based group key generation for sensitive attribute protection in medical health care systems," *International Journal of Information Technology*, vol. 17, no. 1, pp. 645–654, Jan. 2025, doi: 10.1007/s41870-024-02306-x.
[13] Q. Liu, F. Zhou, and H. Chen, "Secure medical data on cloud storage via DNA homomorphic encryption technique," *Physical Communication*, vol. 64, p. 102295, Jun. 2024, doi: 10.1016/j.phycom.2024.102295.
[14] M. S. Jawed and M. Sajid, "CryptoHHO: a bio-inspired cryptosystem for data security in Fog–Cloud architecture," *The Journal of Supercomputing*, vol. 80, no. 11, pp. 15834–15867, Jul. 2024, doi: 10.1007/s11227-024-06055-3.
[15] A. Gupta, S. Namasudra, and P. Kumar, "A secure VM live migration technique in a cloud computing environment using blowfish and blockchain technology," *The Journal of Supercomputing*, vol. 80, no. 19, pp. 27370–27393, Dec. 2024, doi: 10.1007/s11227-024-06461-7.
[16] S. C. Krishnan, "AI-HybridChain: Picturized authentication and DRL based access control method with secure two fold revocation for ensuring cloud computing security," *Future Generation Computer Systems*, vol. 160, pp. 389–405, Nov. 2024, doi: 10.1016/j.future.2024.04.054.
[17] C. Tamizshelvan and V. Vijayalakshmi, "Cloud data access governance and data security using distributed infrastructure with hybrid machine learning architectures," *Wireless Networks*, vol. 30, no. 4, pp. 2099–2114, May 2024, doi: 10.1007/s11276-024-03658-9.
[18] Z. Zhang, N. Zhou, B. Sun, S. Banerjee, and J. Mou, "Multimedia healthcare cloud personal archives security system based on compressed sensing and multi-image encryption," *Journal of the Franklin Institute*, vol. 361, no. 8, p. 106844, May 2024, doi: 10.1016/j.jfranklin.2024.106844.
[19] K. Sundar, S. Sasikumar, C. Jayakumar, D. Nagarajan, and S. Karthick, "Quantum cryptography based cloud security model (QC-CSM) for ensuring cloud data security in storage and accessing," *Multimedia Tools and Applications*, vol. 82, no. 27, pp. 42817–42832, Nov. 2023, doi: 10.1007/s11042-023-15463-1.
[20] A. Kousalya and N. Baik, "Enhance cloud security and effectiveness using improved RSA-based RBAC with XACML technique," *International Journal of Intelligent Networks*, vol. 4, pp. 62–67, 2023, doi: 10.1016/j.ijin.2023.03.003.
[21] P. Shrivastava, B. Alam, and M. Alam, "A hybrid lightweight blockchain based encryption scheme for security enhancement in cloud computing," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 2683–2702, Jan. 2024, doi: 10.1007/s11042-023-17040-y.
[22] S. Mohammed, S. Nanthini, N. B. Krishna, I. V. Srinivas, M. Rajagopal, and M. A. Kumar, "A new lightweight data security system for data security in the cloud computing," *Measurement: Sensors*, vol. 29, p. 100856, Oct. 2023, doi: 10.1016/j.measen.2023.100856.
[23] S. Gangadharaiah and P. Shrinivasacharya, "Secure and efficient public auditing system of user data using hybrid AES-ECC crypto system with Merkle hash tree in blockchain," *Multimedia Tools and Applications*, vol. 83, no. 29, pp. 72301–72320, Feb. 2024, doi: 10.1007/s11042-024-18363-0.
[24] X. Tang, C. Guo, K.-K. R. Choo, X. Jiang, and Y. Liu, "A secure and lightweight cloud data deduplication scheme with efficient access control and key management," *Computer Communications*, vol. 222, pp. 209–219, Jun. 2024, doi: 10.1016/j.comcom.2024.05.003.
[25] A. H. Brahim, A. A. Pacha, and N. H. Said, "An image encryption scheme based on a modified AES algorithm by using a variable S-box," *Journal of Optics*, vol. 53, no. 2, pp. 1170–1185, Apr. 2024, doi: 10.1007/s12596-023-01232-8.

BIOGRAPHIES OF AUTHORS



Harish Kumar Narayanaswamy is currently serving as an associate professor in Department of Computer Science and Engineering at BMS Institute of Technology and Management, Bengaluru. He earned his Ph.D. in Computer Science and Engineering from Visvesvaraya Technological University (VTU), Belagavi. He received his master of Technology (M.Tech.) degree in the field of Computer Network Engineering from VTU, Belagavi in 2011 and bachelor of engineering (B.E.) degree in the field of Information Science and Engineering from VTU, Belagavi, in 2009. He has contributed to the academic community by publishing many international journals/conferences. His area of interests includes network security, IoT, artificial intelligence, and machine learning. He can be contacted at email: harishkumar@bmsit.in.



Deepak Gopal is a professor in the Department of Information Science and Engineering at Global Academy of Technology. He holds a Ph.D. in Computer Science and Engineering and has made significant contributions to academia through extensive research and innovation. He has authored numerous research papers in reputed journals and conferences and has also filed multiple patents in the field of computer science and engineering. His area of interests includes network security, cloud security, artificial intelligence, machine learning, and cybersecurity. He is actively involved in mentoring students and guiding research projects. With a passion for advancing technology and education, he continues to engage in cutting-edge research and academic development. He is committed to fostering innovation and preparing the next generation of engineers and researchers. He can be contacted at email: deepak.g@gat.ac.in.



Annapoorna Bangalore Ramesh received B.E. degree in Computer Science and Engineering and M.Tech. degree in computer network engineering, where she graduated as a topper of the university. She is a doctorate holder from Visvesvaraya Technological University in the field of computer science and engineering where her research focused on automation in agricultural field. She received a best Ph.D. thesis award from IIIT, Allahabad. She has about 11 years of teaching experience in core subjects of CSE. To add to her credit, she holds an Indian patent published towards her contribution in the field of agricultural sector. She has published 12 journals and 4 conference proceedings in various peer journals and conferences, respectively and also, she has contributed as a reviewer for various conferences. She is a lifetime member of Indian Society of Technical Education. She has been a recipient of a funded student project from Karnataka state government. Her research interests include AI, ML, data analytics, data science, and networking. She can be contacted at email: annapoorna-cs@dayanandasagar.edu.



Sarala Duggonahalli Veeresh is currently serving as an assistant professor in Department of Computer Science and Engineering at BMS College of Engineering, Bengaluru. She earned her Ph.D. in Computer Science and Engineering from Visvesvaraya Technological University (VTU), Belagavi. She received her master of Technology (M.Tech.) degree in the field of Computer Science and Engineering from VTU, Belagavi, in 2012 and bachelor of engineering (B.E.) degree in the field of Computer Science and Engineering from VTU, Belagavi, in 2010. She has contributed to the academic community by publishing many international journals/conferences. Her area of interests includes computer vision, machine learning, and deep learning. She can be contacted at email: saraladv.cse@bmsce.ac.in.