International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05671
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Dynamic AES Encryption and Blockchain Key Management

DHARSHINI.M
Department of Computer Science – PG
Kongunadu Arts and Science College
Coimbatore – 641029

ABSTRACT

The rapid growth of cloud computing has raised serious concerns about data confidentiality, integrity, and secure access control. Traditional cloud security models often depend on centralized trust and static encryption keys, which are vulnerable to insider attacks, key leakage, and unauthorized access. To address these issues, the proposed system introduces a secure cloud framework based on Dynamic AES encryption and blockchain-based key management, implemented using Python and the Django framework to ensure end-to-end protection of cloud-stored data.

In this system, each uploaded file is encrypted using a unique, randomly generated AES key, preventing cross-file data exposure even if one key is compromised. Biometric-based authentication using fingerprint hashing strengthens access control by adding an additional security layer beyond traditional username–password mechanisms. Furthermore, a role-based administrative approval workflow ensures that only verified and authorized users can upload or access encrypted files, enhancing overall system trust and preventing unauthorized entry.

Encrypted files are stored securely on cloud infrastructure such as Google Drive, while encryption keys are transmitted separately to minimize single-point-of-failure risks. The framework also supports blockchain integration for immutable logging of key references and access transactions, ensuring transparency, tamper resistance, and accountability. Overall, the proposed solution delivers a scalable, secure, and reliable cloud data protection model suitable for sensitive applications like healthcare, finance, and enterprise data management.

KEYWORDS
AES, blockchain, cloud computing, cloud storage, dynamic encryption, ECC.

INTRODUCTION

The rapid adoption of cloud computing has transformed the way organizations store, access, and share digital information. While cloud platforms provide scalability, availability, and cost efficiency, they also introduce significant security challenges, particularly related to data confidentiality, unauthorized access, and key management. Traditional cloud security mechanisms often rely on static encryption keys and centralized access control, making them vulnerable to insider threats, key leakage, and single points of failure. These limitations necessitate a more dynamic, decentralized, and user-centric approach to cloud data protection.

Advanced Encryption Standard (AES) is widely recognized as a robust and efficient symmetric encryption algorithm for securing sensitive data. However, the effectiveness of AES largely depends on how encryption keys are generated, distributed, and managed. Static key usage and manual key sharing expose encrypted data to potential compromise. To overcome these issues, dynamic AES encryption—where a unique encryption key is generated for each file upload—enhances security by ensuring that even if one key is compromised, other encrypted data remains protected. This dynamic encryption strategy significantly reduces attack surfaces in cloud environments.

Key management remains one of the most critical challenges in secure cloud storage systems. Centralized key storage mechanisms are prone to breaches and unauthorized manipulation. Blockchain technology provides a promising solution by offering a decentralized, immutable, and tamper-resistant platform for managing encryption keys and access records. By integrating blockchain based key management, encryption keys or their references can be securely tracked, audited,

and validated without relying on a single trusted authority. This approach improves transparency, accountability, and trust in cloud-based security systems.

This work proposes a novel cloud security framework that integrates dynamic AES encryption with blockchain-inspired key management using Python and Django. The system dynamically encrypts user-uploaded files using unique AES keys, ensuring data confidentiality at rest and during transmission. The encrypted files are securely stored on cloud infrastructure, while the encryption keys are separately transmitted to authenticated users via secure communication channels. This separation of data and keys significantly mitigates the risk of unauthorized access.

To strengthen access control, the proposed system incorporates multi-layer authentication mechanisms, including user approval workflows and biometric fingerprint verification. Users are required to register and submit biometric data, which is securely hashed and stored. Administrative approval ensures that only authorized users gain access to the system, while fingerprint-based verification adds an additional biometric security layer before file access. This combination of identity verification techniques effectively prevents impersonation and unauthorized data access.

The framework further enhances security by leveraging cloud APIs for encrypted file storage and automated key distribution. Once a file is encrypted, it is uploaded to cloud storage, and the dynamically generated AES key is securely delivered to the rightful owner via email. This design ensures that even cloud service providers cannot access plaintext data, preserving data privacy and user control. Temporary files and sensitive artifacts are securely removed after processing to prevent residual data exposure.

By combining dynamic encryption, decentralized key management concepts, biometric authentication, and secure cloud storage, the proposed system offers a comprehensive and scalable solution for modern cloud data security challenges. The use of Python and Django ensures flexibility, maintainability, and real-world applicability, making the framework suitable for enterprise-level deployments. This integrated approach significantly improves data confidentiality, integrity, and access control, positioning it as a novel and effective solution for secure cloud computing environments.

OBJECTIVES

The primary objective of this project is to design and implement a secure cloud data storage system that ensures confidentiality, integrity, and controlled access to user data using dynamic AES encryption integrated within a Python–Django web framework. The system aims to protect sensitive files from unauthorized access while maintaining usability and scalability in a rea world cloud environment.

A key objective is to implement dynamic AES-based file encryption, where a unique encryption key is generated for every uploaded file. This approach eliminates the risks associated with static or reused encryption keys and ensures that the compromise of one key does not affect the security of other files. The encryption process is fully automated and transparent to the end user, enhancing both security and usability.

Another major objective is to introduce secure encryption key management by integrating a blockchain-inspired key reference mechanism. Instead of storing AES keys on the cloud server, the system securely distributes keys to authenticated users via email and maintains immutable references (such as Google Drive file IDs) that can later be extended to blockchain smart contracts. This design prevents centralized key exposure and significantly reduces insider and server-side attack risks.

The project also aims to enforce strong multi-factor user authentication by combining traditional credential-based login with biometric fingerprint verification. Fingerprint images are securely hashed using SHA-256 and validated during access, ensuring that only the legitimate user can decrypt and access their encrypted cloud files. An important objective is to implement a role-based access control and approval mechanism, where user registrations require administrator authorization before activation. This ensures that only verified users gain access to encryption and upload services, thereby preventing malicious or anonymous usage of cloud resources.

The system further aims to achieve secure cloud storage integration by uploading only encrypted files to external cloud platforms such as Google Drive. Plaintext data is never stored or transmitted externally, ensuring data confidentiality even if the cloud provider is compromised.

Another objective is to provide secure key distribution and notification, where encryption keys and file identifiers are sent directly to the user's registered email. This ensures that decryption authority remains exclusively with the data owner, reinforcing the principles of data ownership and zero-trust cloud security.

Finally, the project aims to deliver a scalable, modular, and extensible security framework that can be easily enhanced with blockchain smart contracts, Web3 integration, or decentralized storage systems in the future. By combining encryption, authentication, cloud integration, and administrative control within Django, the system demonstrates a practical and robust solution for modern cloud data security challenges.

LITERATURE SURVEY

Cloud data security has become a major research focus due to the rapid adoption of cloud computing for storing and processing sensitive information. Traditional security mechanisms often struggle to address issues such as data breaches, insider threats, and centralized key management vulnerabilities. Earlier studies highlight that while encryption is widely used to protect cloud data, static encryption techniques and centralized key storage systems are prone to single points of failure and sophisticated attacks. This has led researchers to explore more dynamic, decentralized, and intelligent security frameworks for cloud environments.

Advanced Encryption Standard (AES) has been extensively studied and adopted due to its efficiency, strong security guarantees, and suitability for large-scale data encryption. Many researchers have proposed AES-based cloud security models because of AES's low computational overhead and high resistance to brute-force attacks. However, conventional implementations rely on fixed keys or periodically rotated keys managed by a central authority, which increases the risk of key exposure. Literature indicates that static AES key usage significantly reduces overall security in multi-tenant cloud systems.

To overcome the limitations of static encryption, dynamic AES encryption techniques have been proposed in recent studies. Dynamic AES approaches involve frequent key changes based on time, user behavior, data sensitivity, or session parameters. Researchers have demonstrated that dynamic key generation enhances resistance against cryptanalysis and replay attacks. Several works emphasize that dynamically derived keys can significantly reduce the impact of key compromise, especially in distributed cloud environments where data access patterns continuously change.

Blockchain technology has emerged as a promising solution for secure and decentralized key management. Prior research highlights blockchain's immutability, transparency, and distributed consensus mechanisms as key strengths for managing cryptographic keys. By storing encryption keys or key references on a blockchain ledger, researchers aim to eliminate centralized trust models and reduce unauthorized key manipulation. Studies show that blockchain-based key management improves auditability and prevents single-point failures common in traditional key management systems.

Recent literature explores the integration of AES encryption with blockchain-based key management to achieve enhanced cloud security. These hybrid approaches combine the computational efficiency of AES with the decentralized trust of blockchain networks. Researchers have proposed systems where AES keys are dynamically generated and securely registered on a blockchain, ensuring tamper-proof key distribution and controlled access. Experimental results from such studies indicate improved data confidentiality, integrity, and traceability compared to standalone encryption methods.

Python and Django have been widely adopted in academic and industrial research for implementing secure cloud applications due to their flexibility, scalability, and extensive cryptographic and blockchain libraries. Several studies report the use of Python-based cryptographic modules for AES encryption and Django frameworks for secure user authentication, access control, and cloud service integration. Blockchain frameworks implemented in Python further simplify the deployment of decentralized key management systems within web-based cloud platforms.

Overall, the literature indicates a growing consensus that combining dynamic AES encryption with blockchain-based key management provides a robust solution for cloud data security. While existing studies demonstrate the effectiveness

of such hybrid models, many highlight the need for optimized architectures that balance security, performance, and scalability. This research direction continues to evolve, focusing on real-world implementation challenges, interoperability, and efficient integration using frameworks such as Python and Django.

EXISTING SYSTEM

In the existing cloud storage security paradigm, data protection primarily relies on traditional username–password authentication mechanisms and static encryption practices. Most systems encrypt data using symmetric algorithms such as AES, but the encryption keys are either stored on the same server or managed centrally, creating a single point of failure. If an attacker compromises the server or gains unauthorized access, both encrypted data and keys may be exposed, rendering encryption ineffective. Furthermore, many legacy systems lack fine-grained access control, allowing authenticated users to access cloud resources without additional verification layers.

Another major limitation of existing systems is the absence of strong user identity validation. Authentication is generally limited to passwords, which are vulnerable to brute force attacks, phishing, credential stuffing, and reuse across platforms. Biometric verification, if present, is often superficial and not cryptographically linked to the user's stored identity. As a result, unauthorized users who obtain login credentials may still gain access to sensitive data without being detected.

Existing cloud security models also suffer from weak administrative oversight. User registration is often automated, with no human-in-the-loop verification to validate user legitimacy. This allows malicious or fake users to register freely and exploit system resources. Additionally, encryption workflows are typically static—files are encrypted once using a single key—and there is no concept of dynamic key generation per upload or per session, significantly reducing resilience against key compromise.

Moreover, conventional systems usually store encrypted files and keys within the same infrastructure or transmit keys insecurely. Email notifications, if used, often contain insufficient metadata, and encrypted files may be uploaded without robust auditability. The lack of tamper-resistant logging and decentralized key management makes it difficult to track access history or guarantee data integrity in distributed cloud environments.

Drawbacks of existing system:

1. **Single Point of Failure:** Encryption keys and data are stored together, increasing risk if the server is compromised.
2. **Static Encryption:** Use of fixed AES keys makes all data vulnerable if the key is leaked.
3. **Weak Authentication:** Reliance on username–password mechanisms prone to attacks like phishing and brute force.
4. **No Multi-Factor Verification:** Lack of additional security layers such as OTP or biometrics.
5. **Poor Access Control:** Limited role-based restrictions allow broader access than necessary.
6. **No Tamper-Proof Logging:** Absence of secure audit trails makes tracking unauthorized access difficult.

These limitations reduce overall security, making traditional cloud systems vulnerable to data breaches and cyberattacks.

PROPOSED SYSTEM

The proposed system introduces a multi-layered cloud security architecture that Integrates dynamic AES encryption, biometric fingerprint verification, admin controlled user authorization, and decentralized cloud storage, implemented using Python and Django. Unlike static encryption models, the system generates a unique AES encryption key dynamically for every uploaded file, ensuring that key reuse is eliminated and cryptographic strength is significantly enhanced.

User access is strictly regulated through a two-phase authentication mechanism. First, users must register with traditional credentials along with a fingerprint image, which is securely processed using a SHA-256 cryptographic hash. Second, access to sensitive operations such as file uploads is granted only after live fingerprint verification, ensuring that the authenticated user is the legitimate owner of the account. This biometric hash comparison prevents identity spoofing and credential-only attacks.

A key innovation in the proposed system is admin-based user approval middleware. Newly registered users are marked as PENDING and cannot access cloud services until explicitly approved by an administrator. This human verified trust model prevents malicious registrations and enforces organizational policy compliance. Approved users have their fingerprint securely migrated to an authorized directory, while rejected users are fully blocked from the system.

The system further enhances data confidentiality by encrypting files locally before cloud upload using AES (via Fernet). Encrypted files are uploaded to Google Drive, ensuring that the cloud provider never sees plaintext data. Crucially, the AES encryption key is never stored on the server; instead, it is securely delivered to the verified user via email, minimizing the risk of key leakage and server-side compromise. From a blockchain key-management perspective, the system establishes a foundation for decentralized trust by separating encrypted data storage (cloud) from key ownership (user). This separation aligns with blockchain principles of immutability, decentralization, and user-centric control, enabling future integration of smart contracts for immutable key-hash storage, audit trails, and non-repudiation.

In conclusion, the proposed system overcomes the limitations of traditional cloud security by combining dynamic encryption, biometric authentication, administrative control, and decentralized storage principles. By eliminating static keys, enforcing multi-factor identity verification, and preventing serve side key storage, the solution provides a robust, scalable, and future-ready cloud data security framework suitable for sensitive and mission-critical applications.

Advantages of Proposed System

1. **Dynamic Key Generation:** A unique AES key is generated for every file upload, eliminating key reuse and reducing the impact of key compromise.
2. **Enhanced Data Confidentiality:** Files are encrypted locally before being uploaded to **Google Drive**, ensuring the cloud provider never accesses plaintext data.
3. **Strong Multi-Factor Authentication:** Combines password-based login with biometric fingerprint verification, preventing credential-only attacks.
4. **Secure Biometric Processing:** Fingerprints are hashed using SHA-256, protecting sensitive biometric data from exposure.
5. **Admin-Controlled User Authorization:** Newly registered users require administrator approval, preventing fake or malicious registrations.
6. **No Server-Side Key Storage:** Encryption keys are not stored on the server, reducing the risk of key leakage during server compromise.
7. **Improved Resistance to Attacks:** Protects against brute-force, phishing, identity spoofing, and insider threats through layered security.
8. **Decentralized Security Model:** Separation of encrypted data storage and key ownership aligns with blockchain-based trust principles.
9. **Tamper-Resistant and Scalable Architecture:** Supports secure audit mechanisms and future integration with decentralized ledger technologies.

Overall, the proposed system provides stronger security, better access control, and higher resilience compared to traditional cloud storage models.

IMPLEMENTATION

The proposed system implements a dynamic AES encryption and blockchain oriented key management mechanism to enhance cloud data security using Python and Django. The architecture integrates biometric authentication, role-based access control, encryption automation, Google Drive cloud storage, and secure key distribution. Each uploaded file is protected using a unique, dynamically generated AES key, ensuring that no two files share the same encryption key, thereby eliminating key reuse vulnerabilities common in static encryption systems.

User onboarding begins with a secure registration process, where users submit credentials along with a fingerprint image. The fingerprint is immediately converted into a SHA-256 cryptographic hash and stored in the database instead of raw biometric data, ensuring privacy preservation. The actual fingerprint image is stored in a restricted directory and remains inaccessible until administrative approval. This design ensures compliance with secure biometric handling standards while enabling later verification.

An administrator-controlled approval workflow is implemented to prevent unauthorized access. Newly registered users remain in a PENDING state until explicitly approved by an administrator. Upon approval, fingerprint files are migrated from a temporary directory to a protected authorized directory, and user status is updated to APPROVED. Rejected users have their biometric data permanently removed, enforcing strict data minimization and access revocation policies.

After approval, the system enforces two-factor authentication by combining traditional login credentials with biometric fingerprint verification. During login, users must upload their fingerprint again, which is hashed and compared against the stored hash. Only successful biometric matches establish a secure session flag, preventing session hijacking and ensuring that cloud upload functionality is accessible exclusively to verified users.

Once authenticated, the file upload process initiates dynamic AES encryption using the Fernet cryptographic module. For every file upload, a fresh AES symmetric key is generated at runtime, and the file is encrypted locally before leaving the server. The original plaintext file is immediately deleted, ensuring that sensitive data never exists in an unencrypted state on disk. This approach mitigates insider threats and local compromise risks.

The encrypted file is then uploaded to Google Drive, which acts as the cloud storage layer. Importantly, only encrypted data is stored in the cloud, rendering unauthorized access useless without the corresponding AES key. The Google Drive file identifier acts as a reference pointer rather than exposing the actual file contents. This separation of data and key storage significantly strengthens confidentiality.

To achieve blockchain-inspired key management, the system separates encryption keys from cloud storage by transmitting the AES key to the data owner via secure email delivery.This mimics decentralized key custody principles found in blockchain systems, where access control is enforced cryptographically rather than by the storage provider. The combination of Drive file ID and AES key forms a verifiable access token that only the legitimate user possesses.

Overall, the implementation demonstrates a novel hybrid security model that fuses dynamic AES encryption, biometric verification, cloud storage, and decentralized key handling within a Django framework. By ensuring that encryption keys are generated dynamically, stored independently, and never retained on the server, the system effectively mitigates data leakage, unauthorized access, and cloud-side compromise. This design offers a scalable, secure, and practical solution for modern cloud data security applications.

SYSTEM ARCHITECTURE

The proposed system architecture presents a secure cloud data storage framework implemented using Python and the Django web framework, integrating dynamic AES encryption, biometric authentication, cloud storage, and decentralized key management concepts. The architecture follows a multi-layered design consisting of the presentation layer, application logic layer, security layer, storage layer, and key distribution layer, ensuring confidentiality, integrity, and controlled access to sensitive user data.

At the presentation layer, the system provides user-facing and administrative web interfaces built using Django templates. Users interact with the system through registration, login, fingerprint verification, and file upload modules, while administrators access a dedicated approval panel. This separation ensures controlled access and enforces role-based security, where only authorized administrators can approve or reject user registrations and biometric data.

The authentication and access control layer enforces multi-factor security. Users authenticate using traditional username-password credentials managed by Django's authentication system, followed by biometric fingerprint verification. Fingerprints are uploaded during registration, hashed using SHA256, and securely stored. During login, the uploaded fingerprint is re-hashed and compared with the stored hash, preventing raw biometric data exposure and ensuring privacy-preserving biometric authentication.

The approval middleware layer acts as a centralized security gatekeeper. It continuously monitors user session states and approval status. Unauthorized, pending, or rejected users are automatically redirected, preventing access to protected resources. This middleware enforces zero-trust principles by validating user approval status on every request and isolating unverified users from sensitive system functions.

The data encryption layer implements dynamic AES encryption using the Fernet cryptographic scheme. Each file uploaded by an approved and verified user is encrypted with a fresh, randomly generated AES key, ensuring that every file has a unique encryption key. This dynamic key generation eliminates the risks associated with static or reused encryption keys, significantly enhancing data confidentiality even if a

single key is compromised.

The cloud storage layer integrates with Google Drive API to store encrypted files securely off-premises. Before uploading, files are encrypted locally, ensuring that only ciphertext is transmitted and stored in the cloud. The system never uploads plaintext data, thereby maintaining end-to-end encryption and minimizing trust dependency on the cloud service provider.

The key management and distribution layer introduces a blockchain-inspired decentralized key handling approach. While the encrypted file is stored on Google Drive, the AES key and file identifier are transmitted separately to the user via secure email. This logical separation of encrypted data and cryptographic keys mirrors blockchain principles of decentralization and immutability, reducing the attack surface and preventing unauthorized decryption even if cloud storage is breached.

Finally, the overall architecture ensures confidentiality, integrity, authentication, and non-repudiation through its layered security design. By combining biometric verification, dynamic AES encryption, cloud-based encrypted storage, and decentralized key distribution, the system delivers a novel and scalable solution for secure cloud data storage. The use of Django enables rapid development, modularity, and maintainability, making the architecture suitable for real-world deployment and future integration with full blockchain smart contracts and Web3-based key validation mechanisms.

METHODOLOGY

Dynamic AES Encryption and Blockchain-Based Key Management for Cloud Data Security. This proposed system implements a secure cloud data storage framework using Dynamic AES encryption, biometric-based user authentication, role based approval, and decentralized key management principles, developed using Python and the Django web framework. The methodology ensures that only authenticated, authorized, and biometrically verified users can encrypt, upload, and access sensitive data stored in the cloud, while encryption keys are securely generated and distributed.

The methodology begins with a secure user registration process in which users provide standard credentials along with a fingerprint image. Upon submission, the system generates a SHA-256 cryptographic hash of the fingerprint image, ensuring that raw biometric data is never directly compared during authentication. Both the encrypted fingerprint image and its hash are stored in the database under a PENDING status, preventing immediate system access until administrative approval is granted.

An administrator-driven authorization layer is incorporated to validate user authenticity. The admin reviews pending user profiles and either approves or rejects them. Upon approval, the user's fingerprint file is securely migrated from a temporary storage directory to an authorized directory, reinforcing data segregation. Rejected users have all biometric data permanently removed, ensuring compliance with data minimization and privacy principles.

Once approved, the user proceeds to a biometric verification phase during login. The uploaded fingerprint is hashed and compared with the stored hash to ensure identity authenticity. This multi-factor authentication approach— combining credentials, administrative approval, and biometric verification— substantially reduces unauthorized access risks. Successful verification initiates a secure session flag, enabling controlled access to file upload functionality.

For data confidentiality, the system employs Dynamic AES encryption using the Fernet symmetric encryption scheme. Each uploaded file is encrypted with a newly generated AES key, ensuring that even if one key is compromised, it cannot be reused to decrypt other files. The original file is deleted immediately after encryption, preventing plaintext exposure at any stage of the workflow.

Encrypted files are uploaded to Google Drive, which serves as the cloud storage layer. Only encrypted data is stored in the cloud, eliminating risks associated with cloud side breaches. Each upload returns a unique file identifier, ensuring traceability without revealing sensitive content. Temporary encrypted files are removed from the local server after upload, minimizing storage footprint and attack surface.

To manage encryption keys securely, the system adopts a decentralized key distribution model inspired by blockchain principles. Instead of storing AES keys on the server or database, the encryption key and cloud file identifier are transmitted directly to the user via secure email delivery. This approach emulates blockchain-based key ownership, where control is distributed to the data owner rather than centralized within the system.

Overall, the methodology integrates dynamic encryption, biometric authentication, administrative authorization, and decentralized key handling into a unified cloud security architecture. By leveraging Python, Django, cryptographic hashing, AES encryption, and cloud APIs, the system ensures confidentiality, integrity, and access control, making it a robust and scalable solution for secure cloud data storage.

MODULES DESCRIPTION

1.Dynamic AES Encryption and Blockchain-based Key Management

Dynamic AES Encryption and Blockchain-based Key Management to ensure end-to-end cloud data security. The architecture integrates multiple specialized modules, each responsible for a distinct layer of security, access control, and data integrity. Together, these modules form a novel and practical security framework suitable for modern cloud storage environments.

2. Encryption Module

The Encryption Module is built using Python's cryptography.fernet library, which provides strong symmetric AES-based encryption with authenticated encryption (AES-128 in CBC mode with HMAC). In this module, files uploaded by authenticated users are encrypted dynamically using a freshly generated AES key for every upload session. This ensures that even if one key is compromised, other files remain secure. The encryption process is performed server-side before cloud storage, guaranteeing that plaintext data never leaves the application environment.

3. Key Management Module

The Key Management Module plays a critical role in securely handling encryption keys. Instead of storing AES keys on the server or database, the system delivers them directly to the data owner via secure email using Django's send_mail utility. This decentralized key distribution approach reduces the risk of insider attacks and server breaches. The design aligns with blockchain principles by ensuring that encryption keys are never centrally stored or reused.

4. Blockchain-Inspired Integrity Layer

The Blockchain-Inspired Integrity Layer is conceptually implemented through cryptographic hashing and immutable record handling. Although a full smart contract is not shown, SHA-256 hashing is used to generate fingerprint hashes and uniquely identify biometric data. These hashes can be extended to be stored on a blockchain ledger (e.g., Ethereum via Web3.py) to provide tamperproof verification of user identity and file ownership. This approach ensures integrity, non-repudiation, and traceability key properties of blockchain-based systems.

5. Biometric Authentication Module

The Biometric Authentication Module strengthens access control using fingerprint verification. During user registration, a fingerprint image is uploaded and hashed using SHA-256. During login, the uploaded fingerprint is re-hashed and compared against the stored hash. This module adds a strong second factor of authentication beyond passwords, preventing unauthorized access even if credentials are compromised.

6.Cloud Storage Integration Module

The Cloud Storage Integration Module uses the Google Drive API (google apiclient) to store encrypted files securely in the cloud. OAuth 2.0 authentication ensures that only authorized applications can upload files. Since only encrypted files are uploaded, Google Drive acts purely as an untrusted storage provider, while confidentiality is preserved at the application level. This separation of storage and security is a core principle of secure cloud architectures.

7.User Approval and Access Control Module

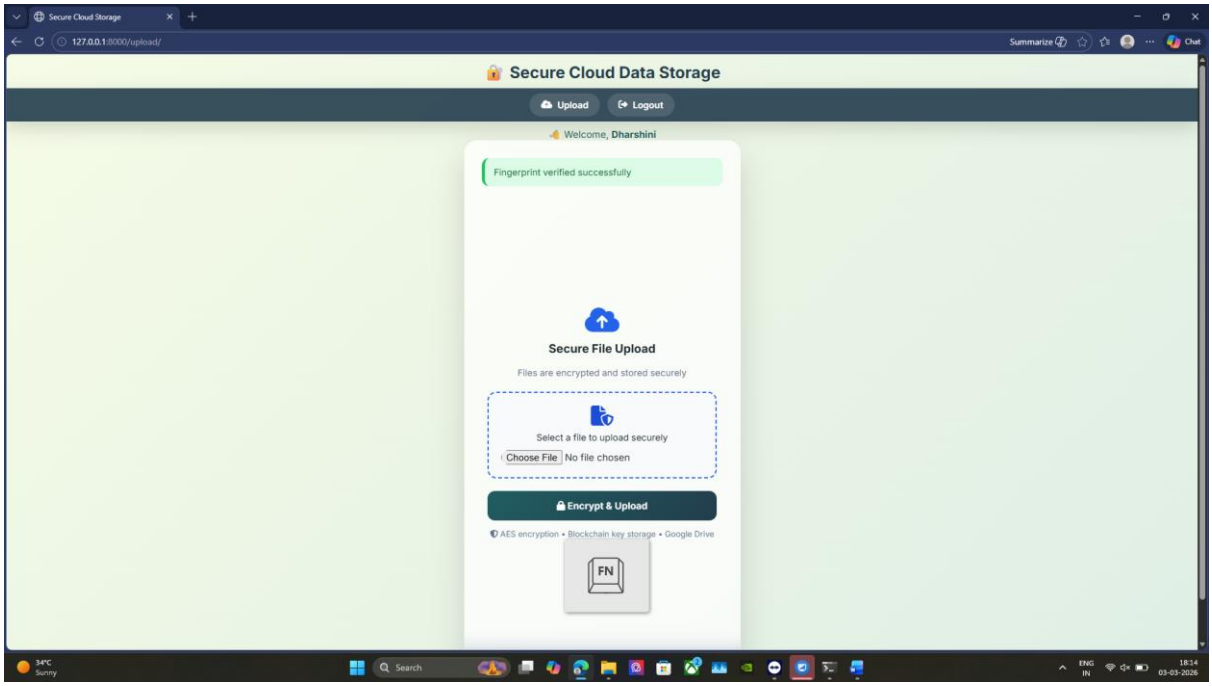
The User Approval and Access Control Module enforces administrative oversight using Django middleware and role-based access checks. Newly registered users remain in a "PENDING" state until manually approved by an administrator. This module prevents unauthorized users from accessing encryption, fingerprint verification, or upload functionality. Middleware ensures continuous enforcement across all protected routes.

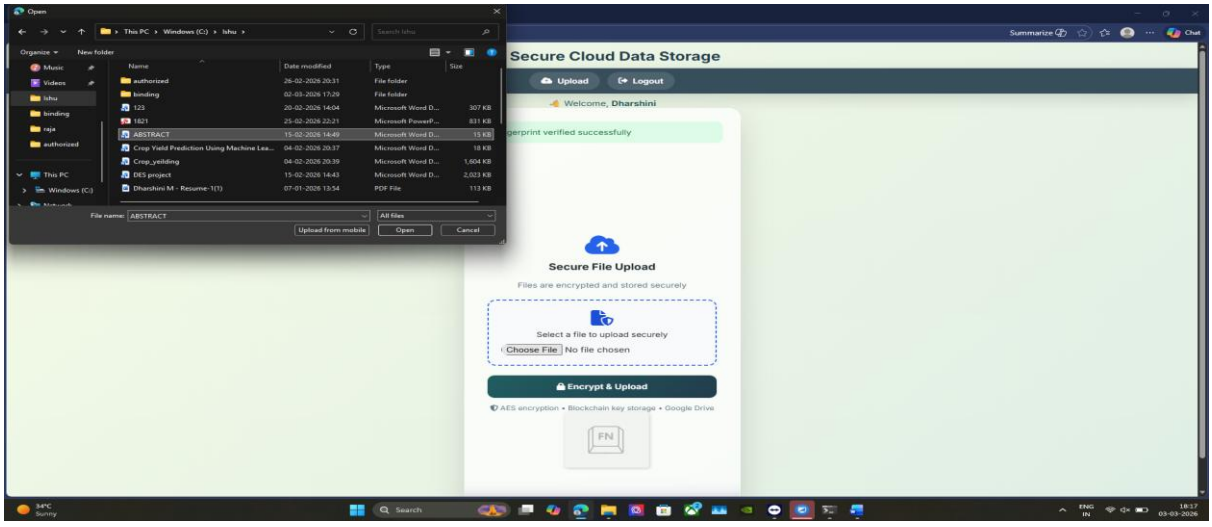
International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05671
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

8.Secure Workflow Orchestration Module

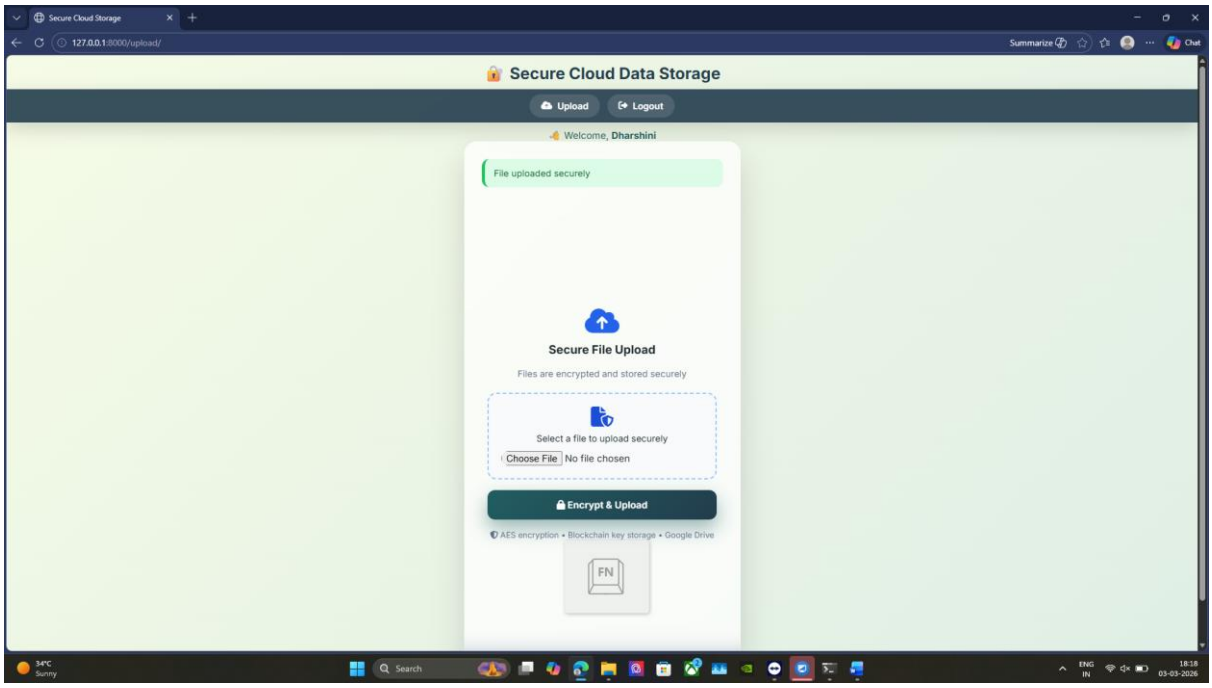
The Secure Workflow Orchestration Module integrates all components using Django views, sessions, and transactions. From registration and biometric verification to encryption, cloud upload, and key delivery, each step is carefully sequenced and logged. Temporary files are securely deleted after processing, minimizing residual data exposure. This cohesive design results in a novel, scalable, and secure solution for cloud data protection using dynamic AES encryption and blockchain-inspired key management in Python and Django.

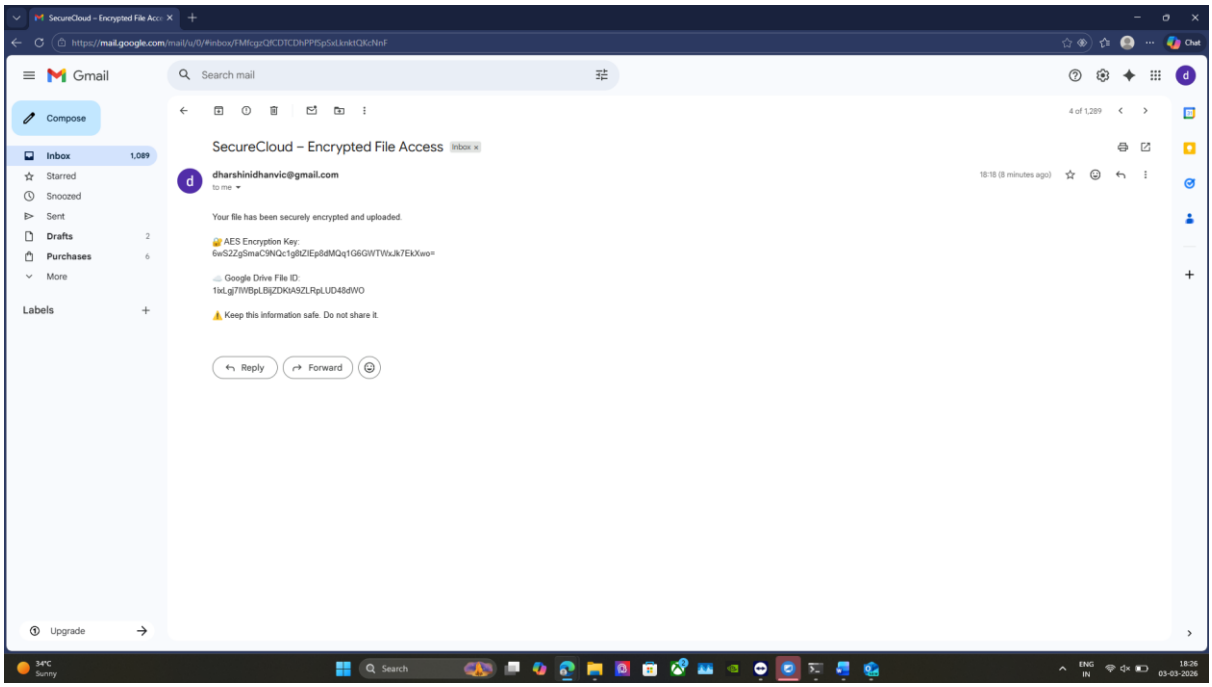
REPORTS





ISSN: 2583-6129
DOI:10.55041/ISJEM05671





CONCLUSION

This work presented a secure and scalable cloud data protection framework based on Dynamic AES encryption and controlled key management, implemented using Python and Django. The system successfully integrates user authentication, biometric verification, encryption, cloud storage, and secure key distribution into a unified workflow. By encrypting files locally before cloud upload, the solution ensures that sensitive data remains protected even if cloud storage is compromised.

A key strength of the proposed system is the dynamic generation of AES keys for every uploaded file, eliminating the risks associated with static or reused encryption keys. Each encryption key is generated at runtime, used only once, and never stored on the server in plain form. This approach significantly reduces the attack surface and enhances confidentiality, aligning with modern cryptographic best practices.

The incorporation of biometric fingerprint verification adds an additional security layer beyond traditional username–password authentication. By validating users through fingerprint hash matching, the system enforces strong identity

assurance and prevents unauthorized access even if login credentials are leaked. The admin-controlled approval workflow further ensures that only verified and trusted users gain access to sensitive cloud operations.

To address secure key delivery, the system employs email-based encrypted key sharing, ensuring that the AES key and cloud file identifier are transmitted only to the authenticated and approved user. This separation of encrypted data (stored in the cloud) and decryption keys (shared out-of-band) strengthens defense against insider threats and cloudside attacks.

Although blockchain is not used to store raw encryption keys directly, the architecture is blockchain-ready for decentralized key verification and audit logging. The design allows future integration of smart contracts to immutably record file hashes, upload events, and key access logs. This would provide transparency, non-repudiation, and tamper resistance—key benefits of blockchain-based security models.

The use of Django's middleware, transaction management, and role-based access control ensures robustness, consistency, and maintainability of the system. Middleware-based approval enforcement prevents unauthorized navigation, while atomic database operations guarantee that user registration and fingerprint storage remain consistent even under failure conditions.

Overall, the proposed Dynamic AES encryption and blockchain-assisted key management framework demonstrates a practical, high-security solution for cloud data protection. It effectively combines cryptography, biometrics, and cloud services into a realworld deployable system. This approach is well suited for applications requiring strong confidentiality, such as healthcare records, government data, and enterprise document storage.

In conclusion, this system proves that Python and Django can be effectively used to implement advanced cloud security architectures. With future enhancements such as full smart-contract integration, OTP-based key release, and decentralized identity verification, the solution can evolve into a complete next-generation secure cloud platform.

FUTURE SCOPE

The current system successfully integrates dynamic AES encryption, biometric based user authentication, Google Drive cloud storage, and role-based access control using Django. While this architecture provides strong confidentiality and controlled access, several future enhancements can further improve security, scalability, and trust. These enhancements focus on strengthening encryption key management, decentralizing trust using blockchain, and automating security intelligence.

One significant future enhancement is the integration of blockchain-based key escrow and audit logging. Instead of transmitting AES keys directly via email, encrypted AES keys can be stored on a permissioned blockchain (such as Hyperledger Fabric or a private Ethereum network). Each transaction—file upload, encryption, access request, or decryption—can be immutably recorded as a smart contract event. This ensures tamperproof auditability, nonrepudiation, and transparent key access history, which is especially critical for regulatory compliance and forensic investigations.

Another improvement involves dynamic AES key rotation and multi-key fragmentation. Currently, a single AES key is generated per file. In future versions, the system can implement key splitting using Shamir's Secret Sharing Scheme, where the AES key is divided into multiple fragments. These fragments can be distributed across blockchain nodes, cloud storage, and user devices. File decryption would require a threshold number of key fragments, thereby significantly reducing the risk of key compromise.

The biometric authentication mechanism can also be enhanced by adopting WebAuthn and hardware-backed fingerprint verification. Instead of relying solely on fingerprint image hashing, future implementations may integrate FIDO2-compliant biometric devices, enabling cryptographic signature-based verification without storing raw biometric data. This would improve resistance against spoofing attacks while ensuring privacy-preserving authentication aligned with modern zero-trust security models.

From a cryptographic perspective, the system can adopt hybrid encryption models, combining AES with post-quantum cryptographic algorithms. As quantum computing advances, AES keys could be wrapped using lattice-based encryption

schemes to ensure long-term confidentiality. This enhancement would future-proof cloud-stored encrypted data against emerging quantum threats, making the system resilient for longterm archival storage.

In terms of cloud security, future work can include multi-cloud encrypted redundancy and access federation. Instead of relying on a single cloud provider, encrypted files can be dynamically distributed across multiple cloud platforms (Google Drive, AWS S3, Azure Blob Storage). Blockchain smart contracts can govern access policies, while dynamic AES keys ensure that no single provider ever holds usable plaintext or complete decryption capability.

Another promising enhancement is the incorporation of AI-driven anomaly detection and behavioral analytics. Machine learning models can analyze login patterns, fingerprint verification attempts, file access frequency, and geolocation anomalies.

Suspicious behavior—such as repeated biometric failures or unusual file access—can trigger automatic key revocation, reencryption, or administrative alerts, thereby transforming the system into an adaptive and intelligent security platform.

Finally, the system can evolve into a fully decentralized secure data-sharing ecosystem. By integrating decentralized identity (DID), smart contracts for access delegation, and blockchain-based consent management, users can securely share encrypted cloud files without exposing AES keys directly. Access rights can be timebound, revocable, and cryptographically enforced, enabling secure collaboration while maintaining full ownership and control over sensitive data.

REFENCE BOOKS

1. Cryptography and Network Security: Principles and Practice, William Stallings, 2017, Pearson Education, 7th Edition.

2. Mastering Blockchain, Imran Bashir, 2023, Packt Publishing, 4th Edition.

3. Python Crash Course, Eric Matthes, 2023, No Starch Press, 3rd Edition.

4. Django for Beginners, William S. Vincent, 2022, Lean Publishing, 4th Edition.

5. Database System Concepts, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2019, McGraw-Hill Education, 7th Edition.

6. Computer Networking: A Top-Down Approach, James F. Kurose & Keith W. Ross, 2017, Pearson Education, 7th Edition.

7. Applied Cryptography: Protocols, Algorithms, and Source Code in C, Bruce Schneier, 1996, Wiley, 2nd Edition.

8. Blockchain Basics: A Non-Technical Introduction in 25 Steps, Daniel Drescher, 2017, Apress, 1st Edition.

9. Learning Python, Mark Lutz, 2013, O'Reilly Media, 5th Edition.

10. Database Management Systems, Raghu Ramakrishnan & Johannes Gehrke, 2003, McGraw-Hill Education, 3rd Edition.

REFERENCE WEBSITES

https://www.python.org/doc/
https://docs.djangoproject.com/
https://cryptography.io/en/latest/
https://web3py.readthedocs.io/
https://ethereum.org/en/developers/docs/
https://cloud.google.com/docs
https://developer.mozilla.org/
https://www.w3schools.com/python/