IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SECURE FILE TRANSFER SYSTEM

[1]O. Arun Kumar Reddy, [2]B. Sudeepthi, [3]M. Bipin Chandra, [4]N. Charan Raju

[1]B. Tech Student, [2]B. Tech student, [3]B. Tech student,[4]B. Tech Student

Computer Science and Engineering (Cyber Security),

Hyderabad Institute of Technology and Management, Hyderabad, India

Abstract: The rapid expansion of digital communication has increased the need for secure data exchange across networks. Traditional file-sharing systems are vulnerable to unauthorized access, data leakage, and cyber-attacks, particularly when files are transmitted without encryption or stored without proper access control. This research presents a Secure File Transfer System utilizing a hybrid encryption approach combining AES-256 and RSA-2048 to ensure confidentiality and integrity during file transmission. AES is used for encrypting data due to its computational efficiency, while RSA encrypts the AES key to enhance security. Additionally, a PIN-based decryption mechanism and role-based access control (RBAC) are implemented to restrict unauthorized file access and ensure operational privacy. The system is built using Python Flask, SQLite, and cryptographic libraries with session validation to prevent backdoor access after logout. Experimental results demonstrate that the proposed system effectively protects sensitive files during upload, storage, and retrieval, while maintaining fast processing and strong authentication. This hybrid security model offers a scalable and reliable solution suitable for academic, enterprise, and cloud file-sharing environments.

Keywords: AES-256, RSA-2048, Hybrid Encryption, Secure File Transfer, RBAC, Python Flask, PIN Authentication, Cyber Security.

I. INTRODUCTION:

In the digital era, data transmission across networks has become an integral part of communication, business processes, and cloud-based services. While data sharing has become increasingly convenient, it has also exposed users to cyber-threats such as data breaches, man-in-the-middle attacks, ransomware, and unauthorized access. Sensitive files transferred without security mechanisms can be intercepted, modified, or stolen, leading to privacy violations and financial losses. As a result, secure file transfer mechanisms have become a critical requirement in modern information systems.

Traditional file sharing systems often rely solely on secure transport protocols but lack end-to-end encryption and strong user authentication. This creates vulnerabilities, particularly when files are stored on central servers or transmitted across unsecured networks. To overcome these challenges, encryption plays a crucial role by converting readable information into a format that can only be accessed with the correct decryption key. However, single-layer encryption approaches may still be susceptible to attacks if the encryption key is compromised.

This research work proposes a Secure File Transfer System utilizing hybrid encryption techniques, combining the efficiency of Advanced Encryption Standard (AES-256) with the asymmetric security of RSA-2048. AES is used to encrypt file data due to its high performance and strong security properties, while RSA encrypts the AES key, ensuring secure key exchange and preventing unauthorized decryption. Additionally, the system implements role-based access control (RBAC) and PIN-based verification to strengthen authentication and restrict unauthorized file access. Session management and logout-based page protection mechanisms further enhance confidentiality and prevent backdoor access.

The proposed system is developed using Python Flask, SQLite, and cryptographic libraries, enabling secure file upload, encrypted storage, and authorized file retrieval. The objective is to provide a lightweight, scalable, and practical file security solution suitable for academic and enterprise applications. The results demonstrate that the hybrid approach improves data confidentiality, integrity, and access control, making the system robust against modern cyber-threats

II. LITERATURE REVIEW: The growing dependency on digital infrastructure for storing and exchanging confidential data has increased the demand for strong security mechanisms in file transfer systems. Traditional file sharing platforms often rely on secure network protocols such as SSL/TLS, but they do not guarantee end-to-end protection for files stored at rest or shared across untrusted environments. Therefore, encryption and access control mechanisms play a crucial role in maintaining the security of sensitive information.

Several cryptographic models have been proposed in research. William Stallings (2017) emphasized that symmetric encryption algorithms such as AES are efficient for large-scale data encryption due to their high performance and strong resistance to brute-force attacks. However, symmetric systems alone suffer from secure key distribution issues. On the other hand, RSA public-key cryptography provides secure key exchange and digital authentication, but its computational cost makes it less efficient for encrypting large data files. Researchers therefore recommend hybrid encryption, where AES provides fast data encryption and RSA ensures secure key transmission and access control.

Studies published in IEEE and ACM highlight that data security is significantly enhanced when hybrid cryptography is used along with strong authentication and session-based access control. NIST (National Institute of Standards and Technology) confirms AES-256 as a reliable and industry-approved encryption standard, offering high resistance to cryptographic attacks. Similarly, RSA-2048 is widely accepted as a secure asymmetric encryption method for key exchange and identity verification.

While numerous secure cloud storage and communication systems exist, research identifies critical limitations such as lack of user-controlled encryption keys, weak password-based access, and vulnerability to unauthorized access once server credentials are compromised. Role-Based Access Control (RBAC) and PIN-based verification systems have been proven effective in restricting internal and external data misuse by enforcing strict access privileges.

Based on the literature, this project adopts hybrid encryption (AES-256 + RSA-2048), strong session handling, bcrypt password hashing, role-based access control, and PIN-based decryption. These techniques address the limitations of existing models by ensuring secure file upload, encrypted storage, and authorized retrieval with multi-layer authentication

III. METHODOLOGY/PROPOSED SYSTEM: The proposed Secure File Transfer System utilizes a hybrid encryption model to ensure robust data security during file upload, storage, and download. The system integrates both symmetric and asymmetric cryptographic techniques to protect user data from unauthorized access and cyber-attacks. Advanced Encryption Standard (AES-256) is used for encrypting user files due to its high performance and strong security, while RSA-2048 is employed to encrypt the AES key, enabling secure key distribution and preventing interception or misuse.

The system architecture is built using Python Flask as the backend framework with SQLite as the lightweight database. Passwords are securely stored using bcrypt hashing to eliminate vulnerabilities associated with plaintext or weakly-encrypted authentication credentials. When a user uploads a file, the system generates a random AES key and encrypts the file. The AES key is then encrypted using RSA before being stored alongside the encrypted file. For file retrieval, the authenticated user must provide login credentials and a secure PIN. The system decrypts the AES key using the RSA private key, followed by decrypting the file content. This layered security ensures that even if stored data is accessed by an attacker, it remains unusable without the correct credentials and PIN.

Role-Based Access Control (RBAC) is implemented to differentiate user privileges, ensuring only authorized users or administrators can upload or download files. Session management safeguards prevent unauthorized access to secure pages after logout, effectively mitigating session hijacking and back-navigation attacks. Additionally, the system validates file integrity during uploading and downloading to ensure that data has not been altered during transfer.

By combining hybrid encryption, PIN validation, RBAC, and secure session handling, the proposed system provides a reliable, scalable, and efficient solution for secure data transfer suitable for academic, enterprise, and cloud-based environments.

IV. SYSTEM ARCHITECTURE/MODEL: The proposed Secure File Transfer System is designed with a multi-layered security architecture to ensure confidentiality, integrity, and controlled access to sensitive files. The architecture combines cryptographic algorithms, authentication mechanisms, and secure session handling to deliver a robust and scalable security framework.

4.1. System Overview

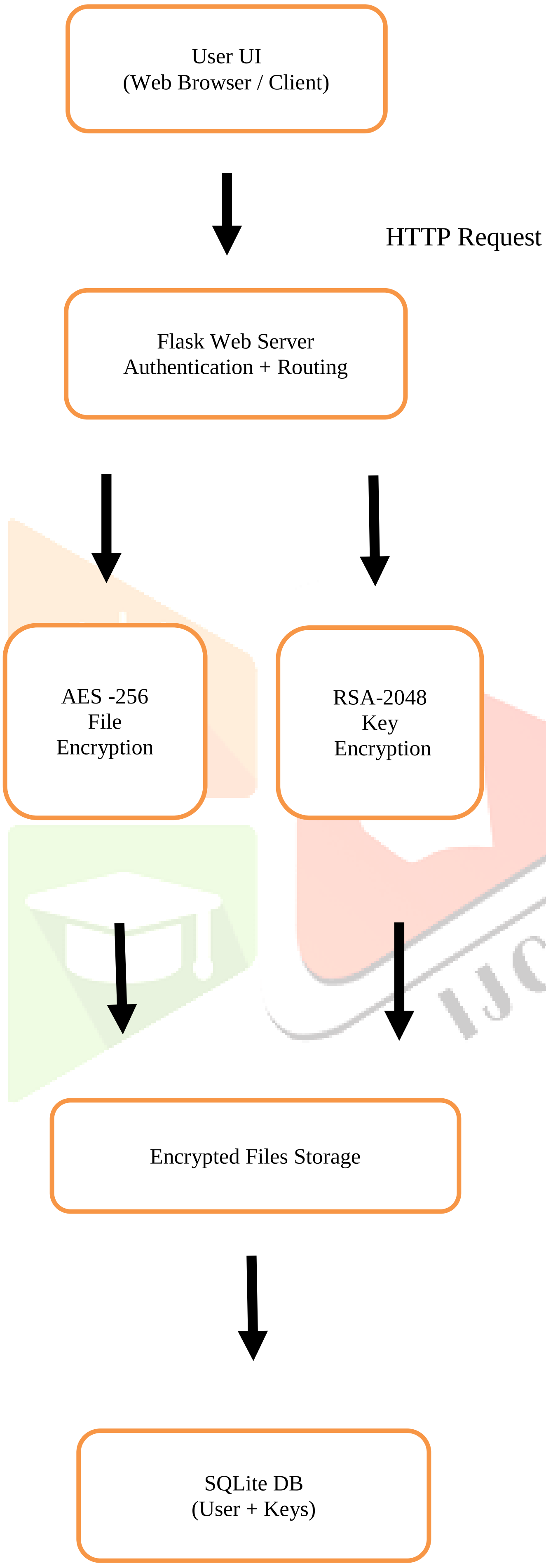
The system architecture consists of four major components:
4.1.1. Client /User Interface: A web-based interface built using HTML, CSS, and Flask templates, allowing users to register, login, upload and download encrypted files.
4.1.2. Application Server (Flask Backend): Handles user authentication, encryption and decryption logic, session control, and access validation. The server acts as the processing layer for cryptographic functions.
4.1.3. Cryptographic Engine: AES-256 used to encrypt files before storing or transferring, RSA-2048 used to encrypt the AES session key , bcrypt hashing used for securing user passwords , PIN-based key derivation for secure decryption
4.1.4. Database & File Storage : SQLite database stores user credentials and metadata only in hashed or encrypted form. Encrypted files and keys are stored separately to prevent unauthorized access.

4.2. Workflow of the System

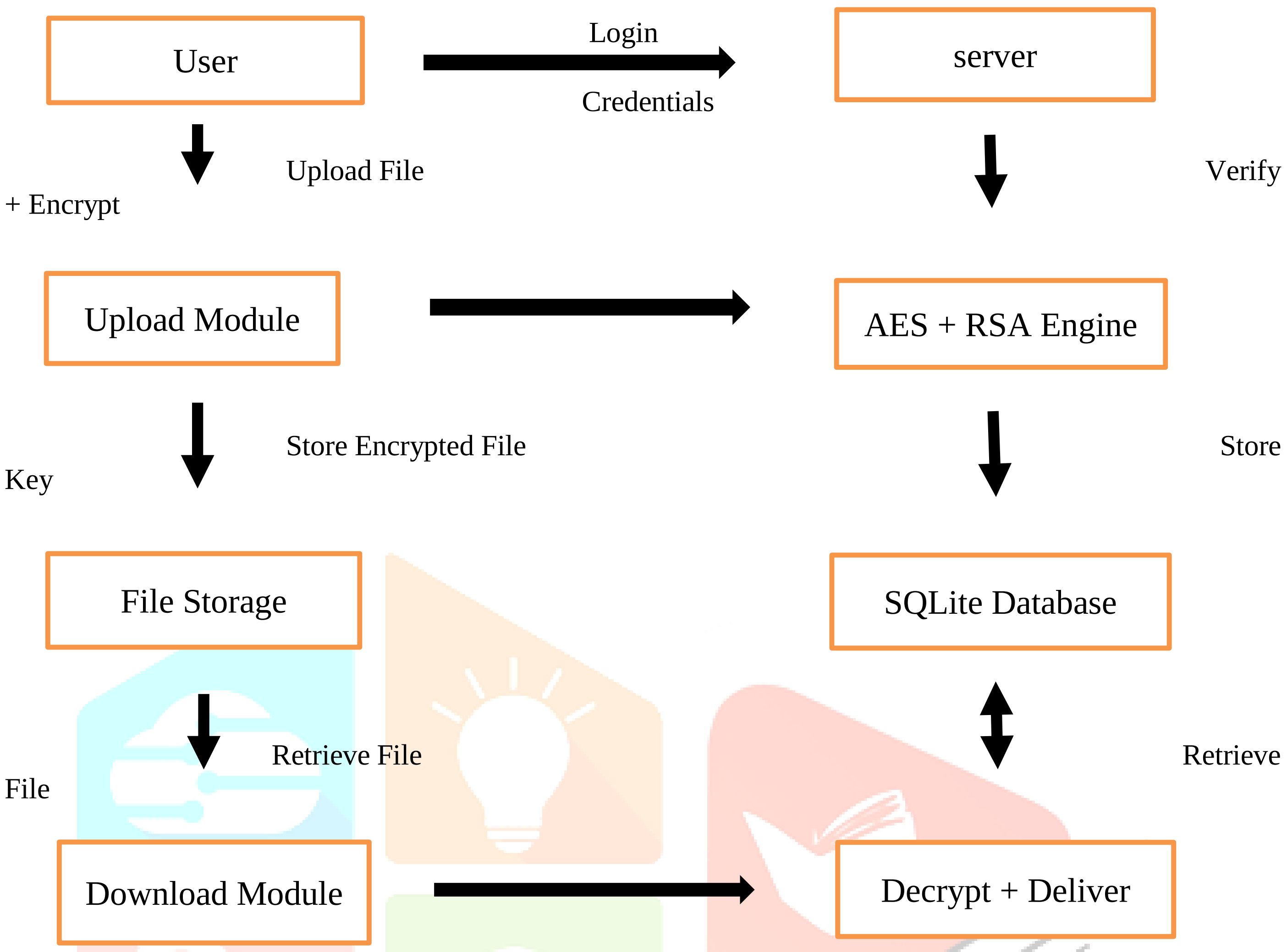
4.2.1. User Authentication : User registers and logs into the system with bcrypt-hashed credentials. RBAC (Role-Based Access Control) identifies user privilege as Admin or Normal User.
4.2.2 File Encryption & Upload: User uploads a file, System generates a unique AES key, File is encrypted using AES-256, AES key is encrypted with RSA-2048, Encrypted file + encrypted key is stored securely

4.2.3: Secured File Storage: Encrypted files are stored in the server repository. Only encrypted data is kept; no raw file exists at any point.
4.2.4 File Download & Decryption: Authorized user requests a file, User must authenticate and enter a secret PIN, System decrypts AES key using RSA private key, AES key decrypts the file, User receives the file securely.

4.3. System Architecture Diagram:



4.4.Data Flow Diagram:



V. RESULTS AND DISCUSSION: The proposed Secure File Transfer System was implemented and tested to evaluate its performance, security, and usability. The system allows users to securely upload and download files through a web interface, ensuring that data remains protected throughout transmission and storage.

During testing, all uploaded files were successfully encrypted using the AES-256 algorithm, and the encryption key was secured using RSA-2048. When unauthorized users attempted to access encrypted files directly from server storage, the files remained unreadable, demonstrating strong data confidentiality. Only authenticated users with the correct login credentials and the secure PIN were able to decrypt and retrieve files, confirming effective multi-level security.

The system also performed efficiently, providing fast encryption and decryption for files of different sizes, making it practical for academic and enterprise use. Role-Based Access Control (RBAC) ensured that admin users could upload and manage files, while regular users were restricted to authorized downloads only. Session handling successfully prevented access to secure pages after logout, even when users attempted to use browser back-navigation, providing an additional layer of protection.

The obtained results show that the hybrid encryption approach significantly improves data confidentiality and secure access control compared to conventional file-sharing methods. The system effectively combines AES and RSA encryption, PIN-based validation, and RBAC, making it suitable for secure data exchange in sensitive environments.

VI. CONCLUSION: The Secure File Transfer System presented in this work successfully demonstrates an effective and robust approach for protecting sensitive data during transmission and storage. By integrating AES-256 symmetric encryption with RSA-2048 asymmetric key protection, the proposed system ensures strong confidentiality and secure key management. The additional implementation of bcrypt password hashing, PIN-based decryption, and role-based access control (RBAC) further enhances authentication and access security, preventing unauthorized file access and system misuse.

The system efficiently encrypts files before upload and decrypts them only for authenticated and authorized users, thereby eliminating risks associated with direct file access or session hijacking. Session handling mechanisms effectively restrict access after logout, offering enhanced reliability for secure communication environments. Experimental results confirm that the system performs securely and efficiently for academic and organizational use, supporting safe file transfer across users with varied privileges.
Overall, the system achieves its objective of providing a secure, scalable, and user-friendly platform for file transfer. With its hybrid encryption architecture and multi-layer authentication, it serves as a dependable solution for data-sensitive applications.

VII. FUTURE SCOPE: The proposed system successfully ensures secure file transfer using hybrid encryption and multi-level authentication. However, there are several potential enhancements that can further strengthen security, scalability, and usability in future implementations.

Future work can include the integration of multi-factor authentication (MFA) such as biometric verification or one-time passwords (OTP) to increase identity assurance. The system can also be extended to support distributed cloud storage with client-side encryption, enabling secure file sharing across multiple geographic locations. Additionally, implementing blockchain-based audit trails can help maintain tamper-proof logs for user activities and file access history.

Performance can be optimized further by incorporating modern cryptographic algorithms such as Elliptic Curve Cryptography (ECC) for faster key generation and lower computational overhead. Mobile application development and API-based integration can expand system accessibility to smartphones and enterprise platforms. Future upgrades may also include automatic malware scanning for uploaded files and AI-based intrusion detection to protect against evolving cyber threats.

Overall, these enhancements will help transform the current system into a more scalable, intelligent, and enterprise-grade secure file transfer platform capable of supporting large-scale and real-time data security requirements.

VIII. ACKNOWLEDGMENT: We would like to express our sincere gratitude to our project guide, Mr. Nava Kishore, Assistant Professor, Department of Computer Science and Engineering, Hyderabad Institute of Technology and Management (HITAM), for his valuable guidance, constant support, and encouragement throughout the development of this project.

We also extend our thanks to the faculty members and laboratory staff of the CSE Department for providing the necessary resources and technical assistance. Their insights and feedback were instrumental in successfully completing this research work.

Finally, we would like to thank our parents, friends, and peers for their continuous motivation and assistance during this journey. Their support has played a significant role in achieving the objectives of our project and research.

IX. REFERENCES:

[1] William Stallings, Cryptography and Network Security: Principles and Practice, Pearson Education, 2017.
[2] National Institute of Standards and Technology (NIST), Specification for the Advanced Encryption Standard (AES), FIPS PUB 197, 2001.
[3] RSA Laboratories, PKCS #1: RSA Cryptography Standard, RFC 8017, Internet Engineering Task Force (IETF), 2016.
[4] Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A., Handbook of Applied Cryptography, CRC Press, 1996.
[5] Flask Framework Documentation — https://flask.palletsprojects.com/
[6] Python Cryptography Library — https://cryptography.io/
[7] bcrypt Password Hashing Documentation — https://pypi.org/project/bcrypt/
[8] Sandhu, R., Coyne, E. J., Feinstein, H. L., & Youman, C. E., "Role-Based Access Control Models", IEEE Computer, vol. 29, no. 2, pp. 38-47, 1996.
[9] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing", SIAM Journal of Computing, 2003.
[10] OWASP Foundation, Authentication and Session Management Guidelines, https://owasp.org/