Enterprise Secure File Storage System using AES-256-GCM Encryption

Internship Project Report

Prepared by: Shusanta Bargayary

Institute: Central Institute of Technology, Kokrajhar

Date of Submission: 27 October, 2025

Abstract

This project presents a Secure File Storage System built using Python, integrating AES-256-GCM encryption for protecting sensitive files. It ensures confidentiality, integrity, and authenticity of data by combining strong encryption, cryptographic hashing, and database logging. The system features a graphical user interface (GUI) to make encryption and decryption processes user-friendly while maintaining backend transparency through secure key management and automatic logging. It provides a practical demonstration of cybersecurity implementation in desktop environments.

Tools and Technologies Used

Programming Language	Python 3
GUI Framework	Tkinter
Database	SQLite3
Encryption Algorithm	AES-256-GCM (Cryptography Library)
Hashing Function	SHA-256
Threading	Python threading module

Steps Involved in Building the Project

1. Key Generation and Management: A 256-bit AES key is generated and securely stored in a file with restricted permissions.

2. GUI Development: Tkinter provides an intuitive interface to select, encrypt, decrypt, and view logs with real-time progress feedback.

3. Encryption Process: The selected file is encrypted in 64KB chunks using AES-256-GCM mode. A unique nonce ensures randomness, and an authentication tag is appended for integrity verification.

4. Decryption Process: During decryption, the system retrieves the nonce and tag, reconstructs the cipher, and verifies authenticity before restoring the file.

5. Hash Verification: The system generates SHA-256 hashes for both original and encrypted files to confirm integrity.

6. Database Logging: SQLite logs every encryption and decryption with filename, hashes, nonce, tag, and timestamp for accountability.

7. Testing and Validation: The system was tested on multiple file types to ensure efficiency, integrity, and backward database compatibility.

Conclusion

The Secure File Storage System effectively demonstrates modern encryption and data protection techniques using Python. By implementing AES-256-GCM encryption, SHA-256 hashing, and structured database logging, the system achieves a reliable balance between usability and security. It reinforces core cybersecurity principles such as confidentiality, integrity, and authentication. The application can be extended for enterprise use by integrating cloud storage and multi-user access control in future versions.