Secure File Storage System with AES

🔒 Introduction

In the modern digital world, securing sensitive files is a critical requirement. This project focuses on building a **local file encryption and decryption system using AES-256**, ensuring data confidentiality and integrity. The system allows users to securely store files, verify file integrity, and detect tampering. A **Flask-based GUI** was implemented to provide a user-friendly interface for encryption, decryption, and metadata management.

🔼 Abstract

The Secure File Storage System with AES is designed to protect sensitive files from unauthorized access. By leveraging **AES-256 encryption** with password-based key derivation, the system encrypts files, stores metadata securely, and allows decryption only with the correct password. The project incorporates **hash-based integrity verification** to detect file tampering. A web-based GUI enhances usability by providing intuitive workflows for uploading, encrypting, decrypting, and downloading files. This project demonstrates practical skills in **cryptography, secure file handling, and web application development**.

퉩 Tools Used

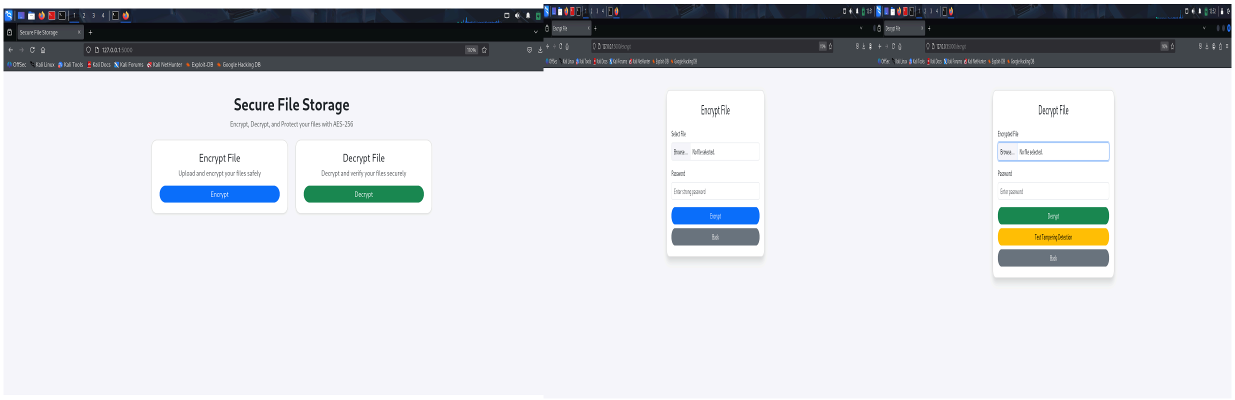
- **Python 3.13** – Programming language for core logic
- **cryptography library** – AES encryption and hashing
- **Flask** – Web framework for GUI
- **Bootstrap 5 / TailwindCSS** – Frontend styling for responsive design
- **Virtual Environment (venv)** – Isolated Python environment
- **Windows & Kali Linux** – Development and testing platforms

🛠️ Steps Involved in Building the Project

1. **Environment Setup**
   - Installed Python, pip, and virtual environment.
   - Created a virtual environment and installed required packages: cryptography and Flask.
2. **Core Encryption & Decryption Logic**
   - Developed secure_store.py to handle AES-256 encryption and decryption.
   - Implemented **password-based key derivation (PBKDF2-HMAC-SHA256)**.
   - Added **hash verification** for integrity checks to detect tampering.

3. **Metadata Management**
   - Stored file metadata including filename, timestamp, and SHA256 hash securely.
   - Verified metadata upon decryption to ensure file integrity.
4. **Testing & Tamper Detection**
   - Created sample files and tested encryption/decryption workflows.
   - Simulated tampering by modifying encrypted files and verifying hash detection.
5. **GUI Development with Flask**
   - Built a **Flask web app** with routes for encryption, decryption, tamper testing, and file download.
   - Integrated **Bootstrap 5** for a modern, responsive UI.
   - Displayed file metadata, hash verification results, and warnings for tampered files.

📸 Screenshot



Secure File Storage

Encrypt, Decrypt, and Protect your files with AES-256

Encrypt File

Upload and encrypt your files safely

Decrypt File

Decrypt and verify your files securely

Encrypt File

Select File: Browse... No file selected.

Password: Enter strong password

Decrypt File

Encrypted File: Browse... No file selected.

Password: Enter password

🔔 Conclusion

The Secure File Storage System with AES successfully demonstrates a **complete end-to-end secure file storage workflow**. It allows users to safely encrypt, store, and decrypt files while ensuring integrity through hash verification. The Flask-based GUI makes the tool accessible and user-friendly. This project strengthened practical skills in **cryptography, secure coding, and web development**, providing a foundation for building more advanced cybersecurity tools in the future.