Secure File Storage System with AES

Introduction

This project is a Secure File Storage System that encrypts and decrypts files locally using AES-256 encryption. It allows users to safely store sensitive data, prevent tampering, and retrieve files only with the correct password.

Abstract

The system was developed with Python, using the Cryptography library for AES, PyQt5 for GUI, and JSON for metadata storage. It provides both CLI and GUI interfaces, ensuring flexibility and strong data protection with integrity verification.

Tools Used

- Python 3.10+
- Cryptography library
- Argparse
- JSON
- PyQt5 (for GUI)

Steps Involved in Building the Project

Step 1: Environment Setup
Installed Python 3.10+, required libraries (cryptography, argparse, json, PyQt5), and created secure_file.py.

Step 2: AES-256 Key Derivation
Derived a 256-bit key from user password using PBKDF2-HMAC-SHA256 with salt and iterations.

Step 3: File Encryption
Implemented AES-GCM encryption. Files saved with .enc extension. Each encryption used a unique IV.

Step 4: File Decryption
Decrypted files with the correct password. Added error handling for wrong password attempts.

Step 5: Command-Line Interface (CLI)
Built CLI with argparse. Supported encrypt/decrypt commands with secure password input.

Step 6: Metadata Storage
Stored metadata in metadata.json including original filename, encrypted filename, timestamp, and SHA-256 hash.

Step 7: Integrity Verification
Verified file integrity by comparing stored SHA-256 hash with decrypted file hash.

Step 8: GUI Implementation (PyQt5)
Developed a GUI with buttons for Encrypt, Decrypt, and Metadata Viewer. Added popup alerts for errors.

Conclusion

The Secure File Storage System successfully combines AES-256 encryption, CLI, and GUI to provide a complete security solution. It ensures confidentiality, integrity, and usability, making it a practical application of cryptography for real-world data protection.