# Secure File Storage System with AES-256 Encryption

## Introduction

In today's digital landscape, securing sensitive data is more crucial than ever. This project, **Secure File Storage System with AES-256 Encryption**, is designed to provide robust local file encryption and decryption. The system is capable of handling multiple files, detecting tampering, and logging all activities. It aims to enhance personal or organizational data security through a user-friendly interface and solid cryptographic practices.

## Abstract

The Secure File Storage System provides a local encryption and decryption mechanism using **Advanced Encryption Standard (AES)** with **256-bit key strength**. Files are encrypted with unique, obfuscated filenames, and associated metadata is logged for integrity verification. The system includes both a **command-line interface (CLI)** and a **PyQt5-based GUI** with support for drag-and-drop, progress feedback, and secure logging. Additionally, users can export metadata or decrypted file paths, making the system suitable for both technical and non-technical users.

## Tools Used

- **Python 3.11**
- **PyQt5** – GUI development
- **cryptography** – AES encryption using Fernet
- **hashlib & base64** – Hashing and encoding
- **argparse** – CLI interface
- **FPDF** – For generating PDF reports
- **JSON & CSV** – Metadata storage and export
- **OS & UUID** – File operations and obfuscation

## Steps Involved in Building the Project

1. *Project Planning*
   Defined clear objectives like secure AES encryption, tamper detection, and a friendly GUI.
2. *Key Derivation*
   Used PBKDF2 with SHA-256 to derive strong keys from user passwords and salt.

3. *File Encryption*
   Implemented AES-256 encryption using Fernet. Each file is uniquely salted and securely stored.
4. *Metadata Logging*
   Logged filename, timestamp, and SHA-256 hashes in a secure metadata.json file.
5. *GUI Development*
   Built a PyQt5 GUI allowing drag-and-drop file selection, password prompts, and progress display.
6. *CLI Interface*
   Added argparse-powered command-line control with batch encryption, --force overwrite, and output folder options.
7. *Testing & Validation*
   Wrote unit tests for encryption, decryption, hash verification, and error handling using pytest.
8. *Additional Features*
   Included CSV export for metadata, file-based logging (secure_vault.log), overwrite prevention, and visual feedback.

---

## Conclusion

The Secure File Storage System is a practical tool that balances strong encryption with user accessibility. By using AES-256 encryption and SHA-256 integrity checks, it ensures both **confidentiality and integrity** of stored data. The modular structure supports both GUI and CLI users and is ready for enhancements like cloud sync or multi-factor authentication. Overall, this project highlights how secure, local file storage can be achieved effectively on personal or enterprise systems.