Secure File Storage System with AES

# Introduction

In the digital era, sensitive information such as documents, financial data, academic records, and personal files must be securely stored to prevent unauthorized access. Traditional storage methods often leave files vulnerable to theft, tampering, or accidental leaks. Encryption provides a robust solution by transforming readable data into an unreadable format unless unlocked with the correct key.  
  
This project focuses on implementing a Secure File Storage System using the Advanced Encryption Standard (AES). By encrypting and decrypting files with password-derived keys, the system ensures that only authorized users can access protected information. Beyond serving as a storage utility, the project also educates students and professionals about modern cryptography’s importance in cybersecurity.

# Abstract

The Secure File Storage System is designed to encrypt and decrypt files using AES-256 in Galois/Counter Mode (AES-GCM), which guarantees both confidentiality and integrity. The system employs PBKDF2 with SHA-256 to convert a user-provided password into a strong, 256-bit encryption key. Randomly generated salts and nonces ensure uniqueness, making each encryption resistant to rainbow table or replay attacks.  
  
The project offers both a Command-Line Interface (CLI) and a Graphical User Interface (GUI) built with Tkinter. The CLI provides simplicity and flexibility for developers, while the GUI ensures user-friendliness by including file pickers, password inputs, and clear status messages. Together, these components make the system suitable for personal use, academic demonstrations, and small-scale professional environments.

# Tools Used

* Python: Core programming language for encryption logic and GUI development.
* Cryptography Library: Provides AES-GCM encryption, PBKDF2 key derivation, and secure random number generation.
* Tkinter: Used to create a minimal desktop-based GUI with file selection and password input.
* Argparse: For building CLI commands (encrypt, decrypt) with user-friendly options.
* OS & Sys Modules: Handle file operations, random salt/nonce generation, and system-level execution.

# Steps Involved in Building the Project

1. Key Derivation: Accept a user password and generate a random salt. Use PBKDF2-HMAC-SHA256 to derive a secure 256-bit key.
2. Encryption Process: Generate a random nonce. Encrypt the file with AES-GCM using the derived key. Save the result in the format: salt | nonce | ciphertext+tag.
3. Decryption Process: Read the salt and nonce from the encrypted file. Derive the same key using the provided password. Attempt decryption with AES-GCM, verifying authenticity and integrity. If successful, restore the original file.
4. CLI Interface: Build commands for encrypt and decrypt with argparse. Add error handling for missing files, mismatched passwords, or tampered files.
5. GUI Interface: Create a Tkinter-based frontend with: File picker, password entry (masked), Encrypt and Decrypt buttons, Status area, and optional progress bar.
6. Testing: Generate sample files of different sizes (text, PDFs, large binary files) and verify encryption/decryption functionality.

# Conclusion

This project demonstrates how encryption can be applied in a practical way to protect digital files. By combining AES-GCM encryption with password-based key derivation, it provides a secure mechanism for file storage that is both effective and user-friendly.  
  
The inclusion of a CLI makes the system flexible for developers and automation scripts, while the Tkinter GUI allows non-technical users to encrypt and decrypt files easily. Error handling for wrong passwords, corrupted files, or tampering ensures reliability.  
  
For learners, the project provides valuable hands-on experience with cryptography, key derivation, and GUI design. For professionals, it acts as a lightweight but effective tool for secure file handling. Future enhancements could include support for cloud integration, chunked streaming for very large files, and stronger key derivation using Argon2.