Problem Statement

- PS- 15 Protecting User Password Keys at rest (on the Disk)

Unique Idea Brief (Solution)

To develop an authorization application that encrypts files or directories using AES-256 and protects the encryption keys, we have followed these steps:

1. Encrypt a File or Directory with AES-256
2. Protect the File Encryption Key
3. Store Encrypted File Encryption Key
4. Authenticate User and Decrypt File

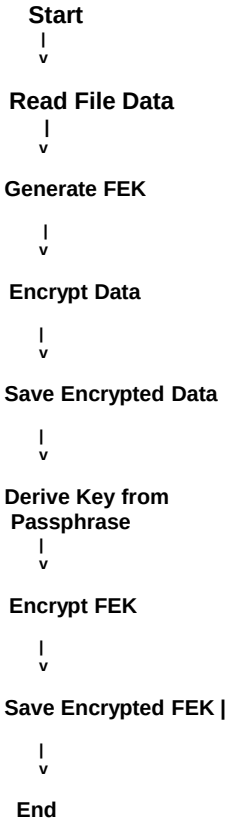
Features Offered

1. **Random Key Generation**:
   - Utilizes AES-GCM to generate a 256-bit random key for file encryption.
2. **File Encryption**:
   - Encrypts data using AES-GCM, ensuring both confidentiality and integrity.
   - Incorporates a secure nonce for each encryption to prevent data reuse attacks.
3. **File Decryption**:
   - Decrypts data encrypted with AES-GCM, retrieving the original file content.
4. **Key Derivation**:
   - Uses PBKDF2-HMAC with SHA-256 to derive a secure key from a user passphrase.
   - Employs a random salt to enhance security against rainbow table attacks.
5. **Key Protection**:
   - Encrypts the File Encryption Key (FEK) with a derived key from the user passphrase.
   - Saves the encrypted FEK and salt securely to a key file.
6. **Secure Storage**:
   - Separates encrypted file data and encryption keys into distinct files.
   - Ensures that the key used for file encryption is protected by a user-provided passphrase.
7. **Usability**:
   - Simple functions for protecting and decrypting files.
   - Example usage provided for easy integration and testing.

Process flow

- Generate a random File Encryption Key (FEK).
- Encrypt the file or directory using this FEK.
- Use a Key Derivation Function (KDF) to derive an encryption key from a user passphrase.
- Encrypt the FEK using the derived key.
- Save the encrypted FEK in a file.
- Prompt the user for the passphrase.
- Use the KDF to derive the key from the user passphrase.
- Decrypt the FEK using the derived key.
- Decrypt the file or directory using the decrypted FEK.

Architecture Diagram



Technologies Used

AES-GCM (Advanced Encryption Standard - Galois/Counter Mode):

- Symmetric encryption algorithm.
- Provides confidentiality and authenticity.
- Used for encrypting and decrypting data.

PBKDF2-HMAC (Password-Based Key Derivation Function 2 - HMAC):

- Key derivation function.
- Uses SHA-256 for hashing.
- Derives a secure key from a user passphrase.
- Adds computational cost to resist brute-force attacks.

SHA-256 (Secure Hash Algorithm 256-bit):

- Cryptographic hash function.
- Produces a 256-bit hash value.
- Used within PBKDF2 for secure key derivation.

OS Module:

- Generates cryptographically secure random numbers.
- Used for creating nonces and salts.

Serialization Module:

- Handles conversion of keys to and from various formats.
- Ensures secure storage and transmission of cryptographic keys.

Team members and contribution:

Siya Sharma: Topic Research, Code Writing, Code Testing, Documentation

Sanidhya Jehi: Topic Research, Code Testing, Documentation

Kushagra Sikka: Topic Research, Code Writing, Code Testing, Documentation

Conclusion

This project provides a robust solution for securely encrypting and decrypting files using AES-GCM. By leveraging PBKDF2-HMAC for key derivation and protecting the encryption keys with user passphrases, it ensures both data confidentiality and integrity, making it a reliable tool for secure data storage.