INTEL UNNATI INDUSTRIAL TRAINING

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

TEAM BIT MINERS

Nithya Santhoshini M

Surepally Vishnnu Vardhini

A project report on

Problem Statement -15

Protecting User Password Keys at Rest (on the Disk)

Problem Statement

Problem Statement - 15

TITLE: Protecting User Password Keys at Rest (on the Disk)

Category: System Software, Security

Scope: Developing an application for file encryption which is in turn protected by user pass phrase

Following are the high level features:

1. Encrypt [AES-256] a user chosen file or directory using a random key a.k.a File Encryption Key.
2. Store the random key in a file, which has to be protected via user pass phrase.
3. The user pass phrase as well as the random key cannot be stored in plain form in the text file.
4. If the user pass phrase authentication is successful retrieve i.e., decrypt the file using File Encryption Key.

Unique Idea Brief (Solution)

Two-Factor Authentication (TOTP): The application uses time-based OTPs for enhanced security during the password setup process, ensuring that only the rightful user can set or verify the password.

User Verification: Upon attempting to set up a password, users must enter the OTP received via email, adding an extra verification step that significantly reduces the risk of unauthorized access.

Enhanced Security: This application ensures that even if a user's password is compromised, an attacker cannot gain access without the OTP sent to the user's registered email.

Dynamic Codes: The OTPs are time-sensitive and change every 30 seconds, further enhancing security by ensuring that the codes cannot be reused.

Bulk File Processing: Allows encryption and decryption of multiple files at once. Increases usability and efficiency for users managing large volumes of data.

Features Offered

Secure File Encryption: Utilizes AES-256 encryption to securely encrypt individual files or entire directories.

Secure Password Setup: Users can set and confirm a password that is securely hashed and used for key derivation.

Automatic Key Management: The application generates a random File Encryption Key (FEK) for each encryption session and securely stores it encrypted with the user's password-derived key.

File Decryption: Users can decrypt previously encrypted files or directories, restoring original content securely.

Logging: Detailed logging of operations for tracking purposes, enhancing the application's transparency and debugging capabilities.

File Handling: Automatic removal of original files after encryption to ensure data security, while allowing recovery of original files through decryption.

Time-Based OTP Verification: Incorporates TOTP for two-factor authentication during the password setup process, enhancing security.

Key Derivation Function (KDF): Uses a KDF to derive encryption keys from the user’s passphrase, ensuring that keys are not stored in plaintext.

Batch File Processing: Allows users to encrypt or decrypt all files in a specified directory at once, improving workflow efficiency.

User-Friendly GUI: Provides an intuitive graphical interface built with Tkinter and Customtkinter, making it accessible for users to encrypt and decrypt files easily, along with password management.

Error Handling and Notifications: Offers detailed error messages and notifications for various scenarios (e.g., password mismatch, failed OTP verification, decryption errors), which helps users troubleshoot issues effectively aiding in user experience.

File Integrity: Ensures that files can only be decrypted with the correct passphrase, maintaining data integrity and confidentiality.

Process flow

1. Application Start

The application initializes and presents the user with a setup screen.

2. User Setup

User inputs their email address and password.

The application sends an OTP (One-Time Password) to the provided email for verification.

User receives the OTP and inputs it into the application.

The application verifies the OTP using the TOTP method.

3. Password Confirmation

If OTP verification is successful, the user confirms their password.

The application checks if the entered password matches the confirmation password.

4. Key Derivation

Upon successful password confirmation, the application derives a key using PBKDF2 (Password-Based Key Derivation Function) with the provided password. The derived key is used for encryption and decryption processes.

5. File/Directory Encryption

User selects a file or directory to encrypt.

The application reads the selected file's contents.

It encrypts the content using AES-256 encryption with a randomly generated File Encryption Key (FEK).

The FEK is securely stored (encrypted) using the derived key.

6. File/Directory Decryption

User selects an encrypted file or directory for decryption.

The application reads the encrypted content.

It retrieves the FEK using the stored information and decrypts the file using AES-256.

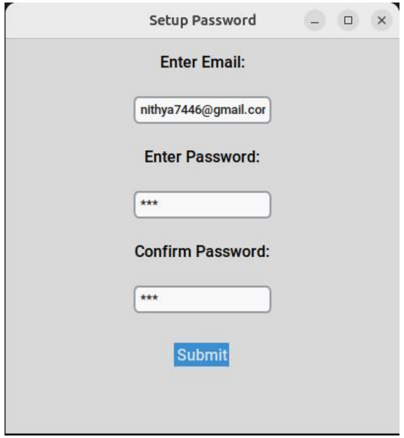
The original content is restored and saved.

7. Feedback to User

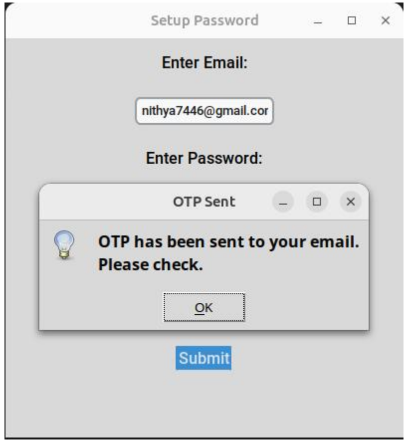
The application provides feedback to the user on the success or failure of the encryption/decryption operations through message boxes.

8. Exit Application

User can close the application once their operations are completed.

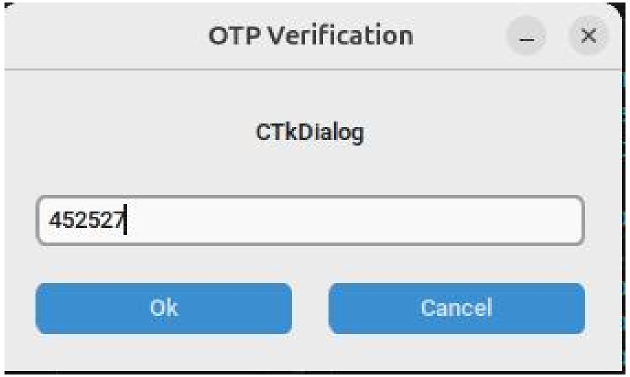


Password Setup

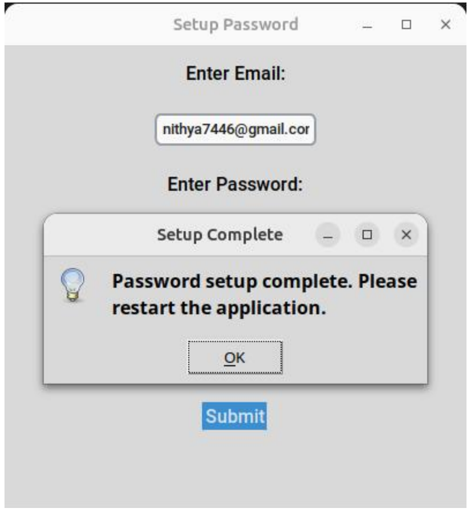


OTP has been sent to your email. Please check.

OTP Sent



OTP Verification


Setup Password

Enter Email:

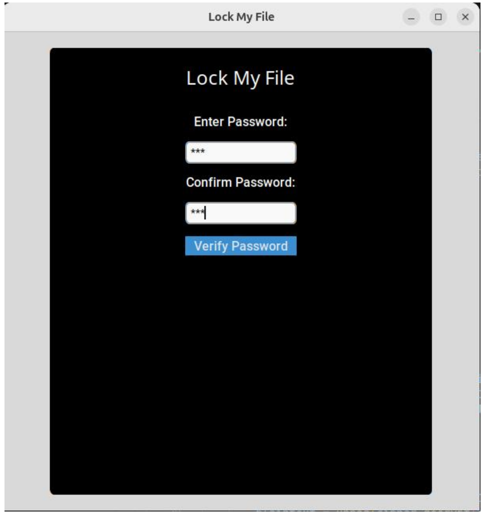
nithya7446@gmail.cor

Enter Password:

Setup Complete

Password setup complete. Please restart the application.

Password Setup Complete


Lock My File

Lock My File

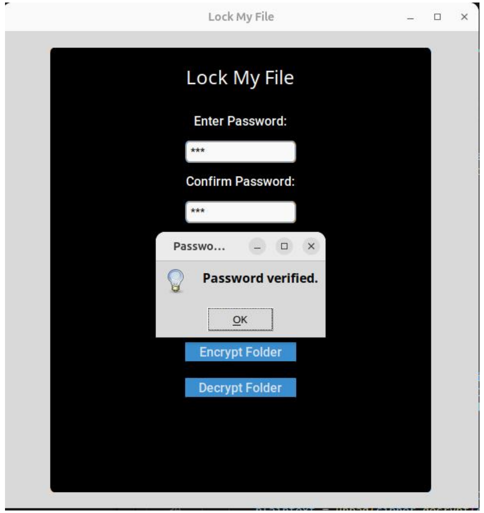
Enter Password:

***

Confirm Password:

***

Password Verification for Encryption or Decryption


Lock My File

Lock My File

Enter Password:

***

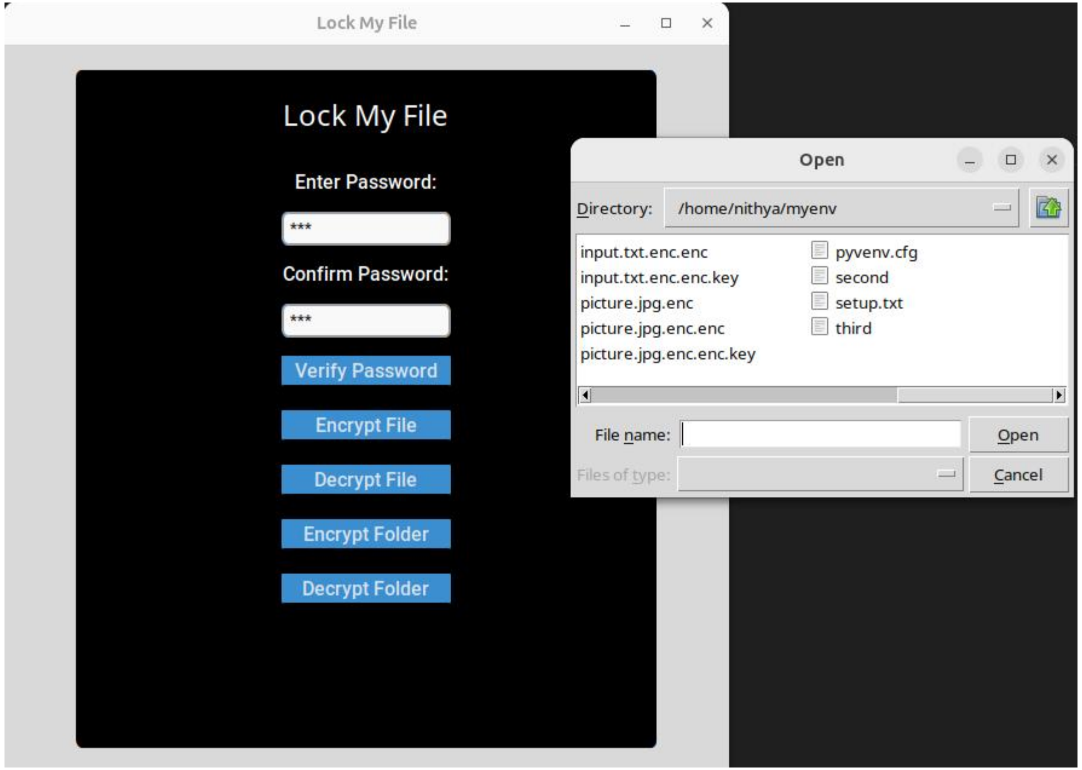
Confirm Password:

***

Passwo...

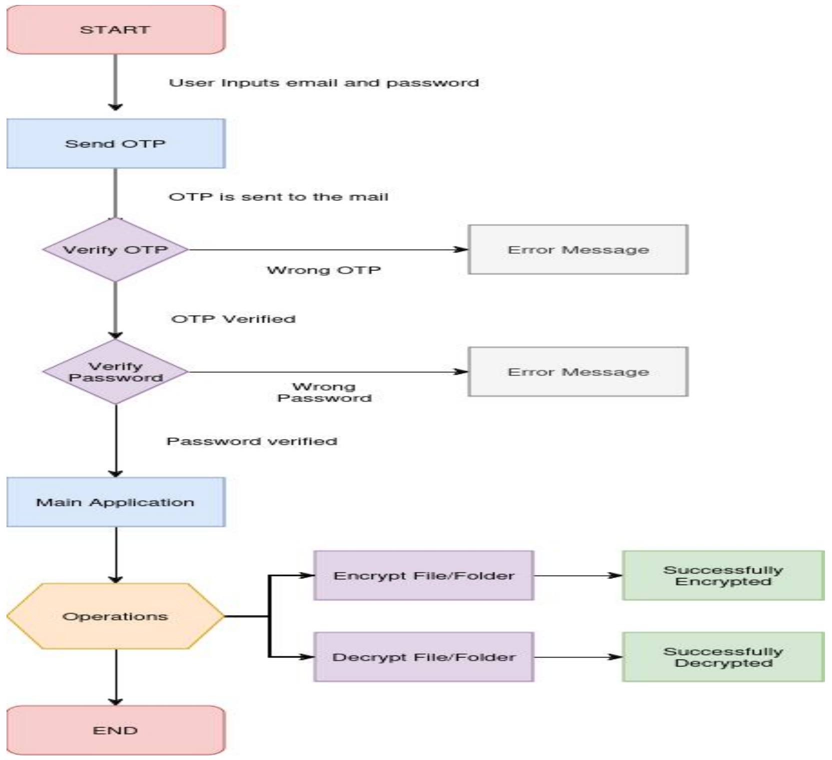
Password verified.

Password Verified



Choose File to Encrypt or Decrypt

Architecture Diagram



Technologies used

1. ***Python:*** The primary programming language used for development.

2. ***Tkinter:*** Python's standard GUI (Graphical User Interface) toolkit for creating desktop applications.

3. ***PyCryptodome:*** A comprehensive Python library for cryptography, providing implementations of various cryptographic algorithms including AES (Advanced Encryption Standard) with 256-bit keys, key derivation functions (KDF), and secure random number generation.

4. ***pyotp:*** A Python library for generating and verifying time-based one-time passwords (TOTP), used for implementing two-factor authentication (2FA).

5. ***smtplib:*** Python's built-in module for sending emails using the Simple Mail Transfer Protocol (SMTP). It's used here to send OTPs (One-Time Passwords) via email for user verification.

6. ***os:*** Python's built-in module for interacting with the operating system, used primarily for file and directory operations such as file reading, writing, and deletion.

7. ***EmailMessage***: For constructing email messages in a user-friendly format.

These libraries and modules collectively enable the functionality and security features of the file encryption application, ensuring robust encryption, user authentication, and a smooth user interface.

Team members and contribution:

Team Member 1: Surepally Vishnnu Vardhini

Role: Encryption and Decryption Development

Contributions: Developed the core encryption and decryption functionalities using AES-256. Implemented the file and directory encryption and decryption processes. Ensured secure key management through the use of a key derivation function (KDF). Integrated cryptographic functions into the overall application, ensuring data integrity and security.

Team Member 2: Nithya Santhoshini M

Role: GUI development and OTP Implementation

Contributions: Designed and developed the graphical user interface (GUI) using *Tkinter*, creating an intuitive and user-friendly experience. Implemented the setup process for user passwords, including password confirmation and storage. Integrated OTP (One-Time Password) functionality using *pyotp*, adding an additional layer of security through two-factor authentication. Developed the email functionality for sending OTPs using *smtplib*, ensuring secure user verification.

Conclusion

In conclusion, our project, focused on "Protecting User Password Keys at Rest," successfully addresses the critical need for secure file encryption. By implementing AES-256 encryption, we ensure robust protection of user data, while the integration of two-factor authentication enhances security by requiring OTP verification before any sensitive actions. This dual-layered approach not only safeguards the password keys but also ensures that users are actively involved in the security process, mitigating risks of unauthorized access.

Furthermore, the ability to encrypt and decrypt multiple files simultaneously streamlines the user experience, making the application both functional and efficient. Our solution showcases the importance of combining strong cryptographic practices with user-friendly interfaces, ultimately fostering greater trust and confidence in digital security measures. As we move forward, continued enhancements to the application's features and user interface will further solidify its position as a reliable tool for data protection in an increasingly digital world.