Problem Statement:

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

Project by:-

Deep Laljibhai Ribadia

MGM College of Engineering and Technology.

Unique Idea (Solution)

FileCryptor:-

The logic behind this file encryption/decryption application involves several key steps and principles aimed at ensuring data security and usability. Here's a breakdown of the core ideas and logic used in the application:

1. User Interaction through GUI:

- **The application uses tkinter to create a graphical user interface (GUI) that allows users to interact with the application easily.**
- **Users can select files, enter passwords, and initiate encryption or decryption through buttons and text fields.**

2. Password Management:

- **The application collects a password from the user and ensures it is no longer than 20 characters. The password can include any character.**
- **This password is used to generate an encryption key using a key derivation function (PBKDF2) to enhance security.**

3. Key Derivation:

- **PBKDF2 (Password-Based Key Derivation Function 2):**
  - **The password provided by the user is combined with a randomly generated salt.**
  - **PBKDF2 applies SHA-256 hashing with the salt for 100,000 iterations to derive a cryptographic key.**
  - **This process makes it computationally expensive to guess the password using brute force or dictionary attacks.**

4. Encryption Process:

- **AES Encryption:**
  - **The derived key is used in the AES (Advanced Encryption Standard) algorithm with CBC (Cipher Block Chaining) mode.**
  - **A random initialization vector (IV) is generated to ensure that the same plaintext encrypts to different ciphertext each time.**
  - **Padding:**
    - **The plaintext file data is padded using PKCS7 padding to ensure it fits into the AES block size.**
  - **File Structure:**
    - **The salt, IV, and original file extension are prepended to the encrypted data.**
    - **This additional metadata allows the decryption process to reconstruct the original file correctly.**

5. Decryption Process:

- **Reading Encrypted File:**
  - **The application reads the salt, IV, original file extension, and encrypted data from the encrypted file.**
- **Key Derivation:**
  - **The same PBKDF2 process is applied using the stored salt and user-provided password to derive the decryption key.**
- **AES Decryption:**
  - **The AES algorithm in CBC mode is used with the derived key and IV to decrypt the data.**
  - **Unpadding:**
    - **The decrypted data is unpadded using PKCS7 to restore the original plaintext.**
- **Restoring Original File:**
  - **The decrypted data is saved to a new file with the original file extension.**

6. File Management:

- **After a file is successfully encrypted or decrypted, the original file is deleted to ensure only the processed (encrypted or decrypted) file remains.**
- **This step helps maintain data security by removing any leftover plaintext files.**

7. Error Handling:

- **The application includes extensive error handling to manage common issues such as:**
  - **Missing file selection.**
  - **Incorrect password entry.**
  - **Invalid decryption keys or signatures, indicating possible tampering.**
- **Users are informed of errors through message boxes and console output.**

Features Offered

GUI Features:

1. **User-Friendly Interface:**
   - **Intuitive layout with clearly labeled buttons and text fields.**
   - **Uses tkinter for creating a desktop application with visual components.**
2. **File Selection:**
   - **Allows users to select the file to be encrypted or decrypted through a file dialog.**
3. **Password Input:**
   - **Secure password entry with masked input (asterisks).**
   - **Password length is limited to 20 characters, but allows any character type.**
4. **Console Output:**
   - **Displays messages and updates to the user about the current status and results of operations.**
   - **Provides feedback on selected files, successful operations, and errors.**

Functional Features:

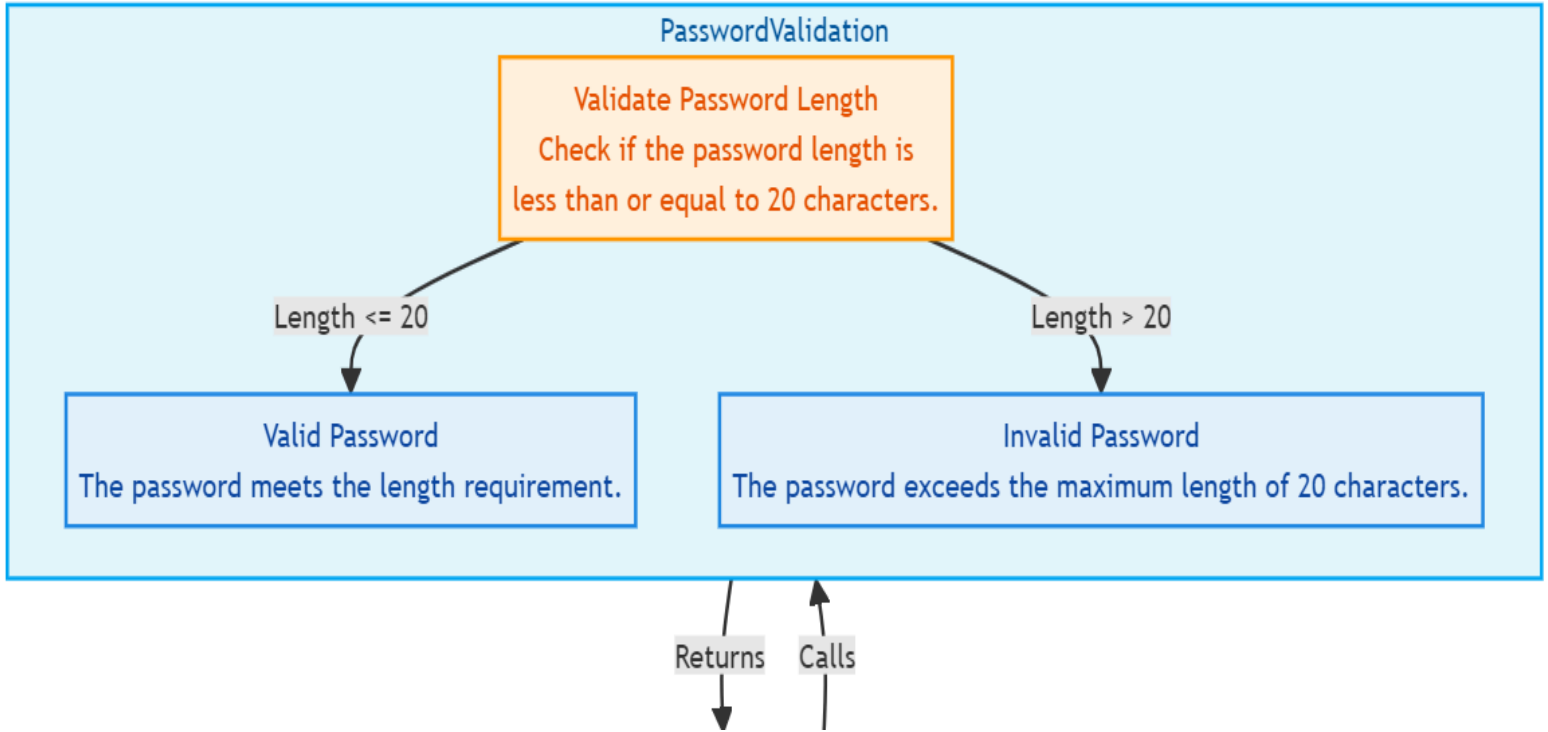
1. **Encryption:**
   - **Uses AES (Advanced Encryption Standard) for secure encryption.**
   - **Implements PKCS7 padding to handle data that is not a multiple of the block size.**
   - **Generates and uses a random salt and initialization vector (IV) for each encryption operation to enhance security.**
   - **Derives encryption keys from the user's password using PBKDF2 (Password-Based Key Derivation Function 2) with SHA-256.**
   - **Stores the salt, IV, and original file extension as part of the encrypted file.**

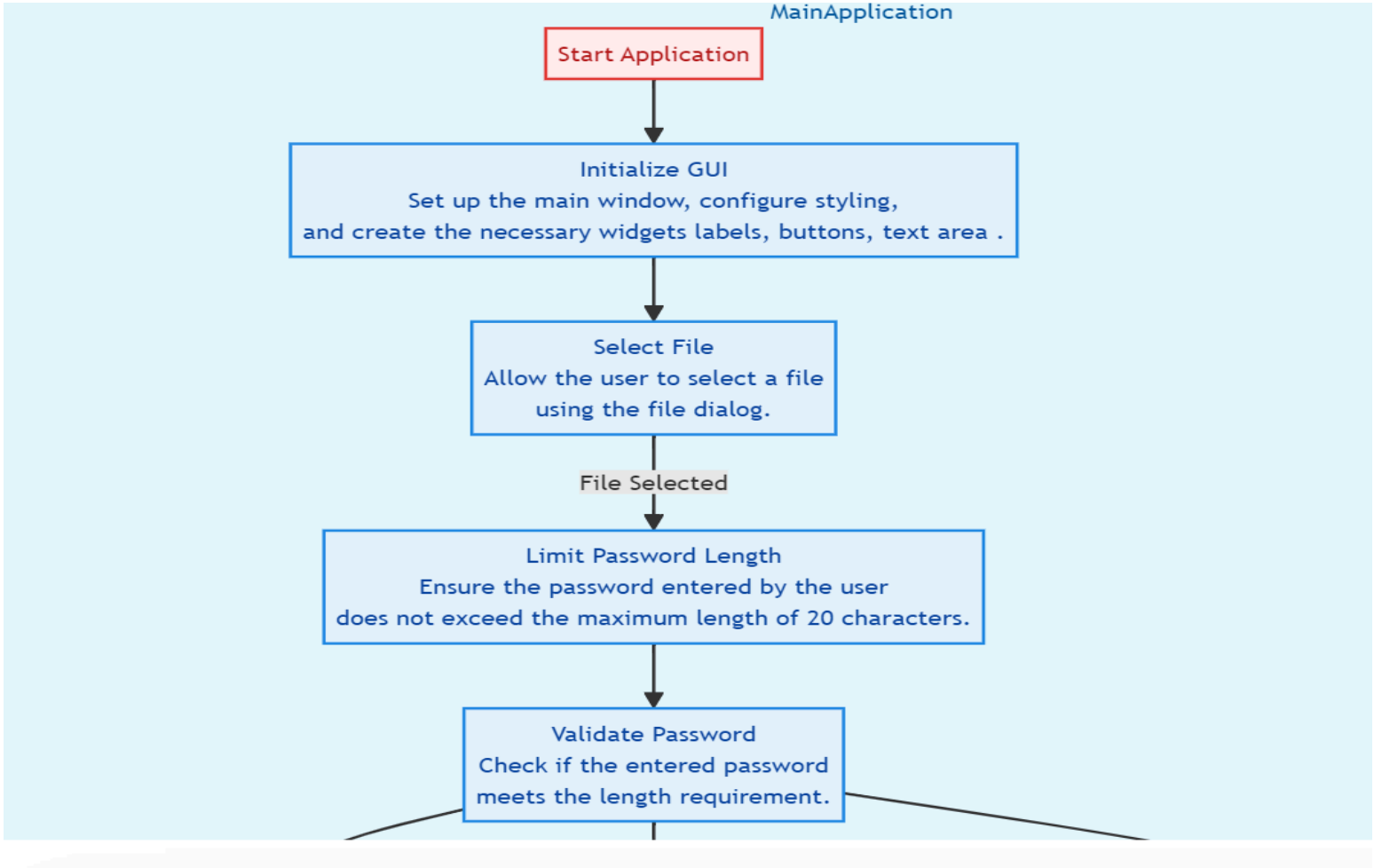
2. **Decryption:**
   - **Decrypts files that were encrypted by the application.**
   - **Uses the stored salt and IV to derive the correct decryption key from the user's password.**
   - **Verifies the integrity and correctness of the decrypted data.**
   - **Restores the original file extension to the decrypted file.**
3. **Error Handling:**
   - **Provides clear error messages for common issues, such as missing files, incorrect passwords, and potential file tampering.**
   - **Displays error messages in both the console output and as message boxes.**
4. **File Management:**
   - **Automatically deletes the original file after encryption or decryption to ensure only the processed file remains.**
   - **Handles file paths and extensions correctly to avoid overwriting important data.**

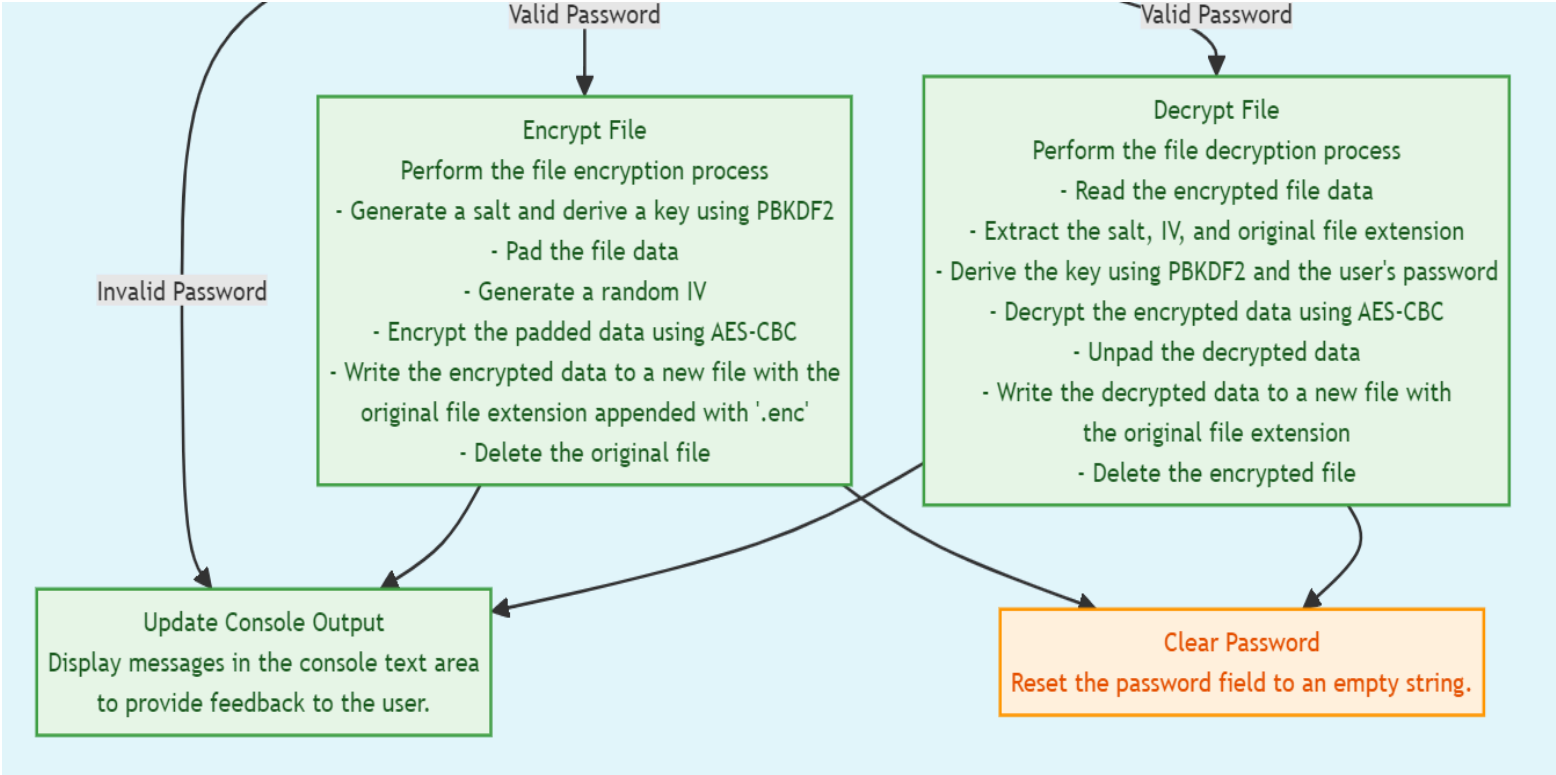
Security Features:

1. **Secure Password Management:**
   - **Uses PBKDF2 with a random salt to derive encryption keys from passwords, making it harder for attackers to use precomputed hashes (rainbow tables) to break passwords.**
   - **Limits password length to enhance usability while maintaining security.**
2. **Data Integrity:**
   - **Ensures that data is padded correctly before encryption and unpadded after decryption to maintain data integrity.**
   - **Uses AES in CBC mode to ensure secure encryption of data blocks.**
3. **Tamper Detection: Incorporates error handling to detect and report potential tampering with encrypted files, such as invalid signatures or incorrect decryption keys.**

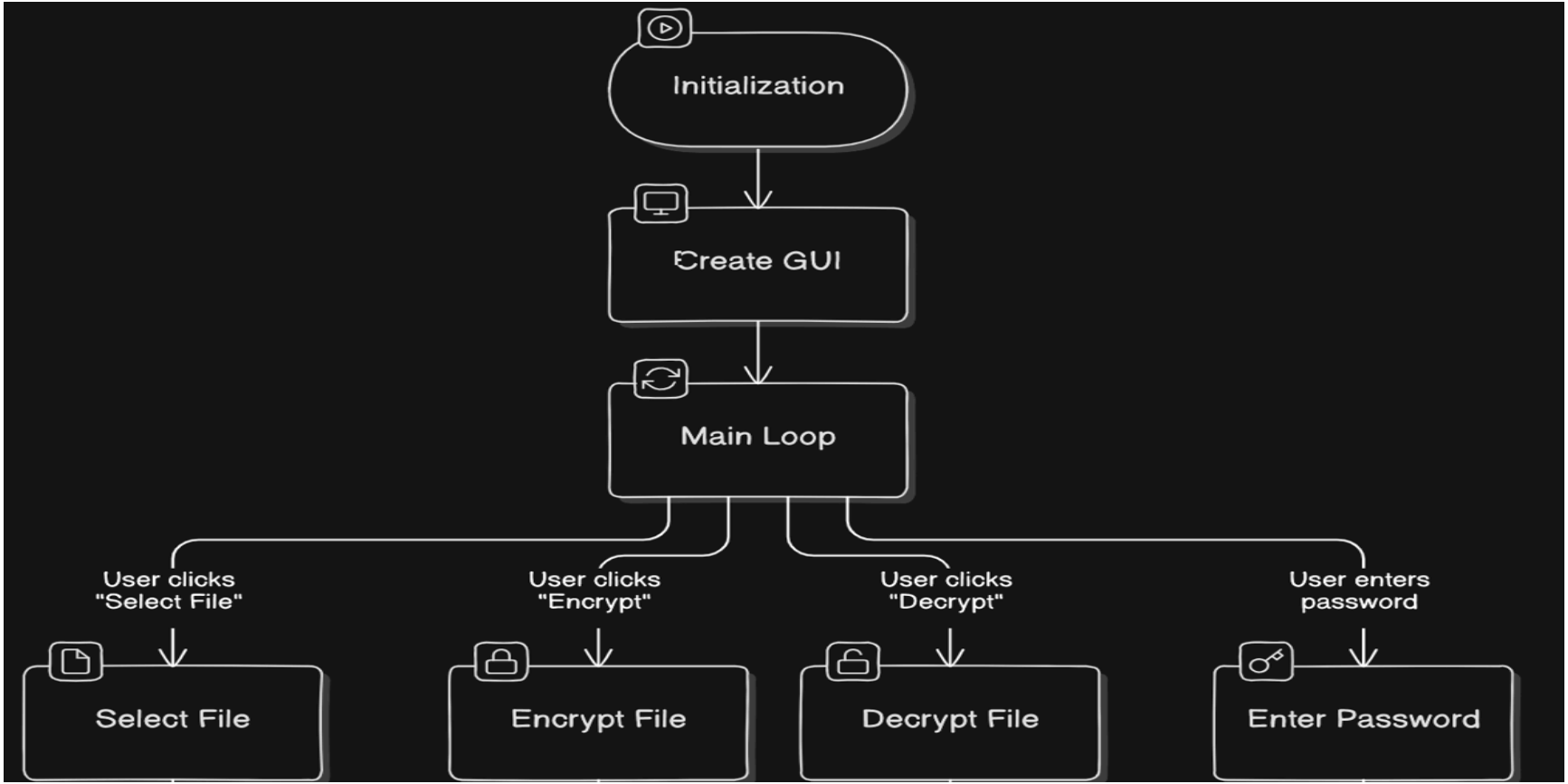
Process Flow

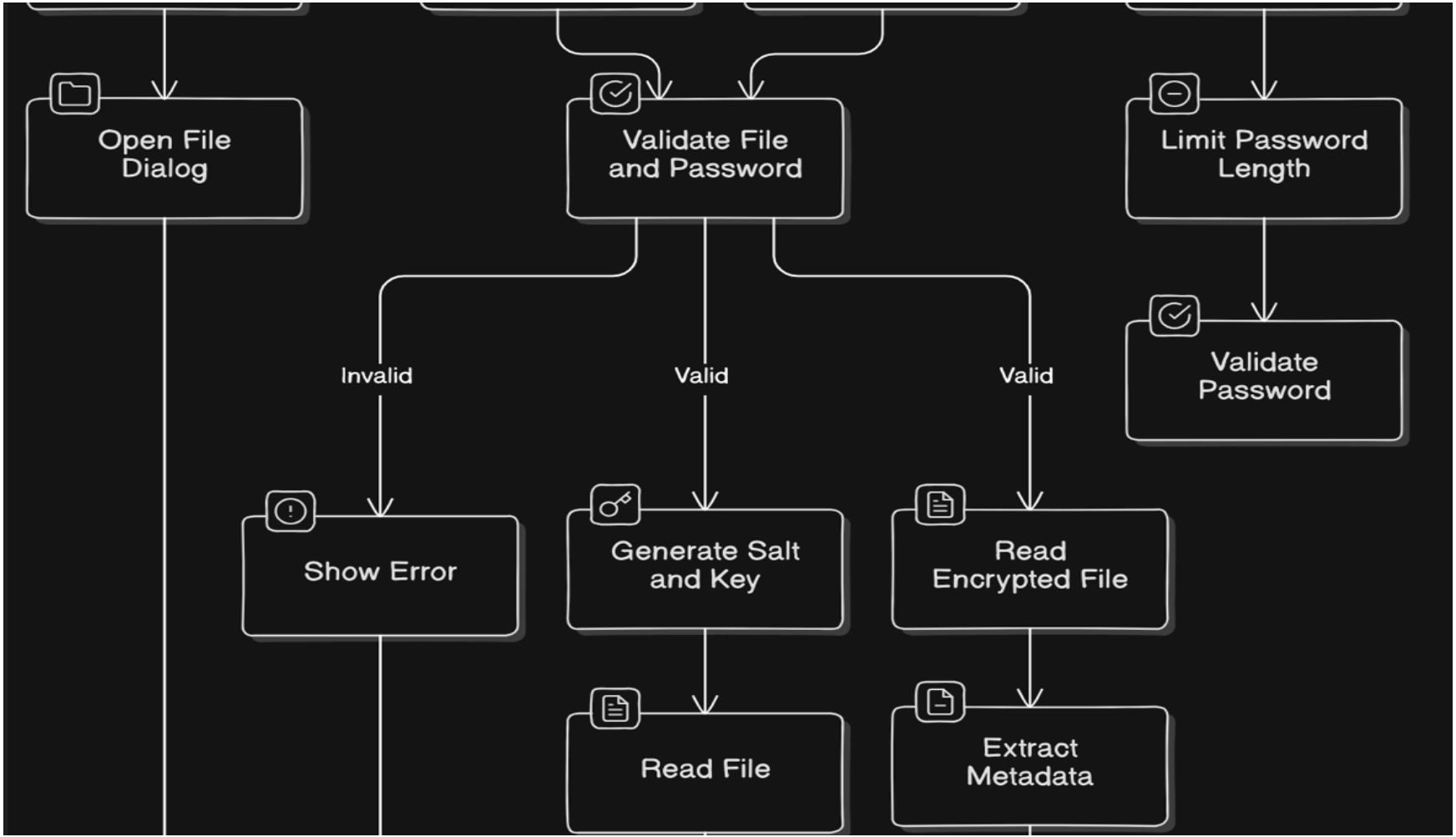


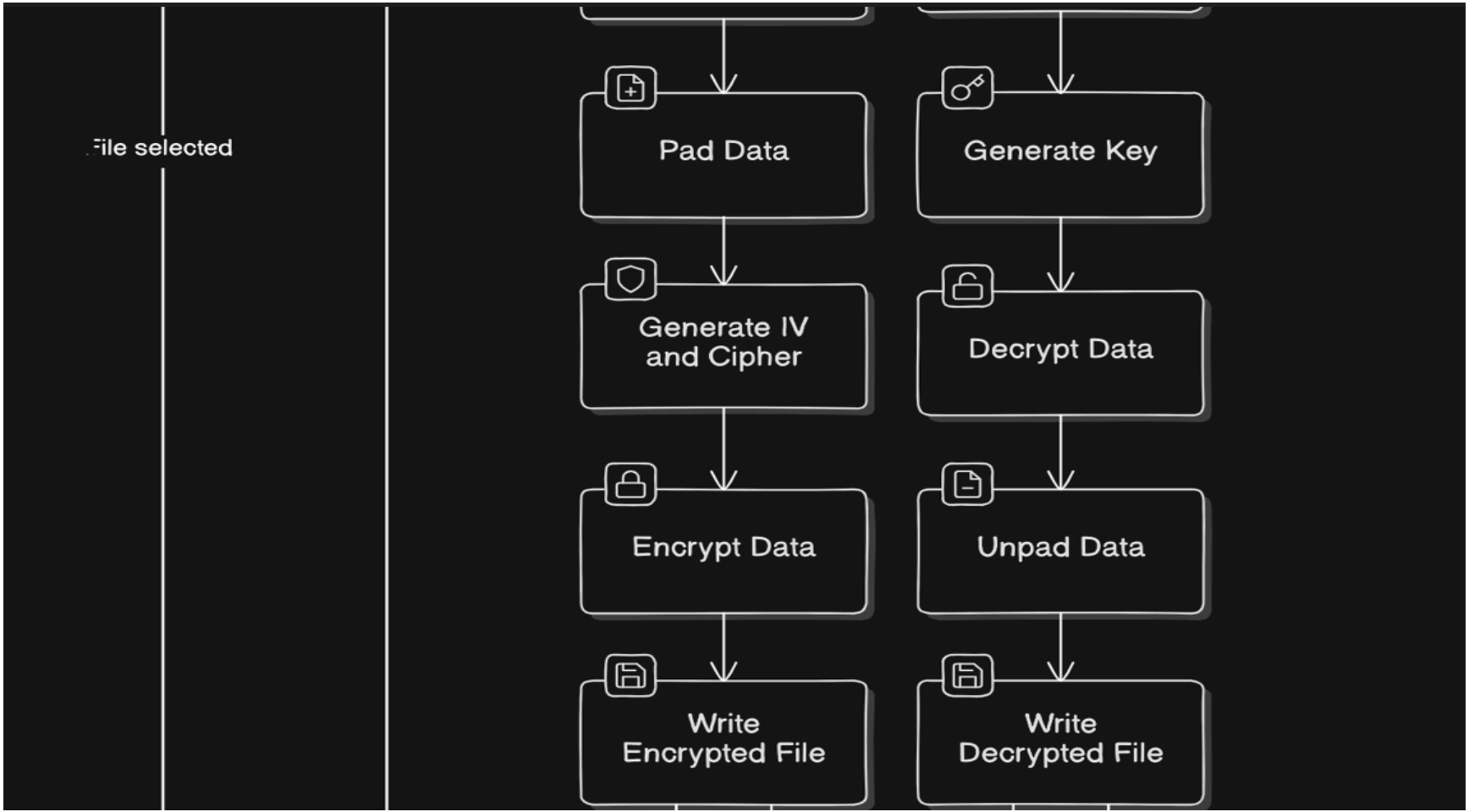

MainApplication
Start Application
Initialize GUI
Set up the main window, configure styling,
and create the necessary widgets labels, buttons, text area .
Select File
Allow the user to select a file
using the file dialog.
File Selected
Limit Password Length
Ensure the password entered by the user
does not exceed the maximum length of 20 characters.
Validate Password
Check if the entered password
meets the length requirement.

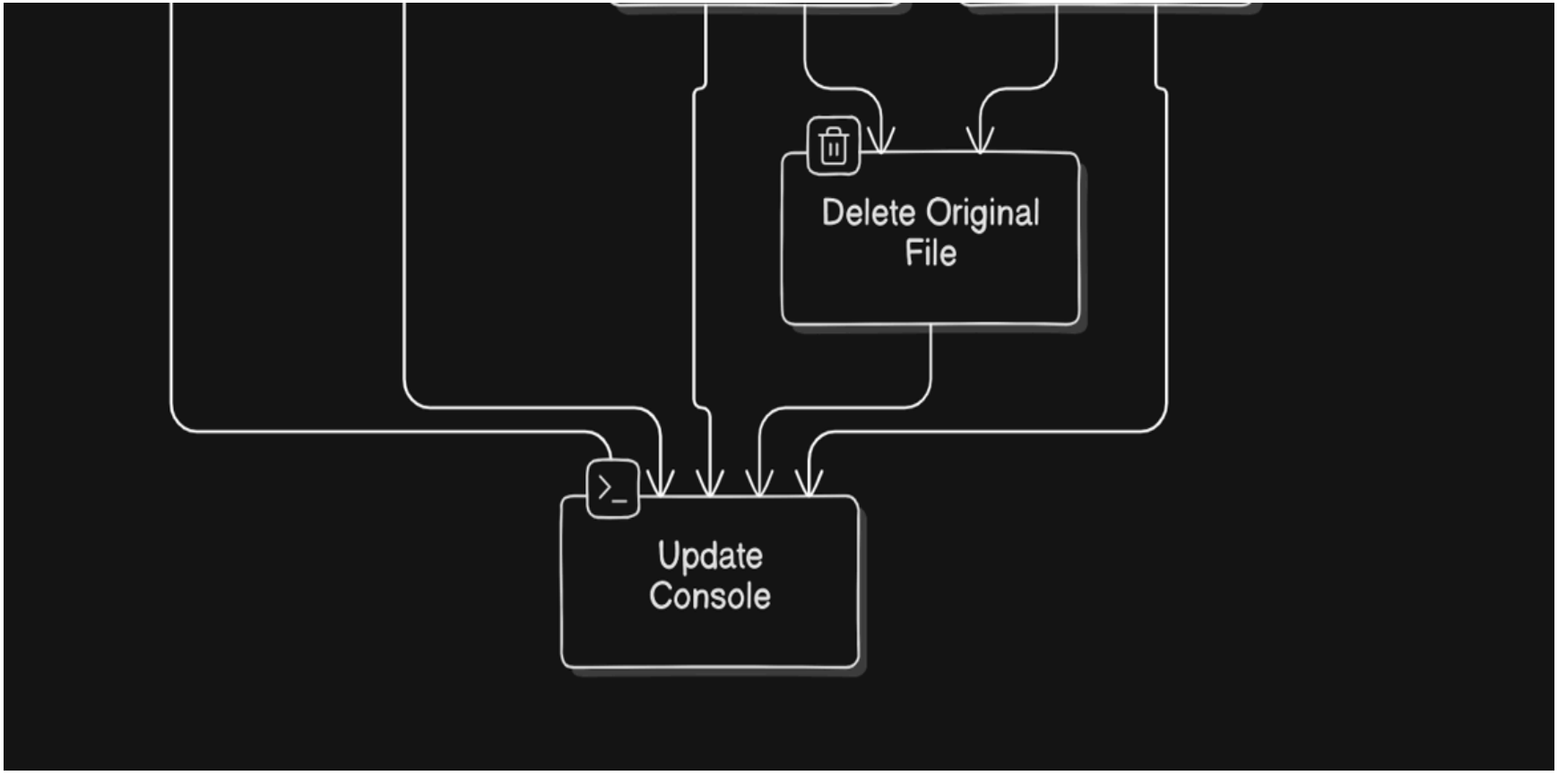

Valid Password
Valid Password
Encrypt File
Perform the file encryption process
- Generate a salt and derive a key using PBKDF2
- Pad the file data
- Generate a random IV
- Encrypt the padded data using AES-CBC
- Write the encrypted data to a new file with the original file extension appended with '.enc'
- Delete the original file
Decrypt File
Perform the file decryption process
- Read the encrypted file data
- Extract the salt, IV, and original file extension
- Derive the key using PBKDF2 and the user's password
- Decrypt the encrypted data using AES-CBC
- Unpad the decrypted data
- Write the decrypted data to a new file with the original file extension
- Delete the encrypted file
Invalid Password
Update Console Output
Display messages in the console text area to provide feedback to the user.
Clear Password
Reset the password field to an empty string.

Architecture Diagram




Open File Dialog
Validate File and Password
Limit Password Length
Validate Password
Invalid
Valid
Valid
Show Error
Generate Salt and Key
Read Encrypted File
Read File
Extract Metadata


File selected
Pad Data
Generate Key
Generate IV
and Cipher
Decrypt Data
Encrypt Data
Unpad Data
Write
Encrypted File
Write
Decrypted File


Delete Original File
Update Console

Technologies Used

Python Libraries:

1. **tkinter:**
   - **Purpose: Provides a GUI toolkit for creating the user interface.**
   - **Features: Used for creating the main application window, buttons, labels, and text fields.**

2. **filedialog and messagebox (from tkinter):**
   - **Purpose: Provides dialogs for file selection and message boxes for displaying alerts and errors.**
   - **Features: filedialog is used for selecting files, and messagebox is used for showing error and information messages.**

3. **cryptography.hazmat.primitives.ciphers:**
   - **Purpose: Provides cryptographic algorithms for encryption and decryption.**
   - **Features: Includes the Cipher, algorithms, and modes modules for implementing AES encryption in CBC mode.**

4. **cryptography.hazmat.backends:**
   - **Purpose: Provides backend interfaces for cryptographic primitives.**
   - **Features: Used to specify the backend for cryptographic operations (default_backend).**

5. **cryptography.hazmat.primitives.padding:**
   - **Purpose: Provides padding schemes for cryptographic data.**
   - **Features: Used to pad plaintext data to the correct block size for AES encryption.**

6. **cryptography.hazmat.primitives.hashes:**
   - **Purpose: Provides hashing algorithms.**
   - **Features: Used in key derivation (PBKDF2) to ensure secure keys.**

7. **cryptography.hazmat.primitives.kdf.pbkdf2:**
   - **Purpose: Provides the PBKDF2 key derivation function.**
   - **Features: Used to derive encryption keys from passwords with a salt.**

8. **cryptography.exceptions:**
   - **Purpose: Provides exception classes for cryptographic errors.**
   - **Features: Handles specific exceptions like InvalidKey and InvalidSignature.**

9. os:

- **Purpose: Provides miscellaneous operating system interfaces.**
- **Features: Used for file operations like reading, writing, and deleting files, and generating random data (salts and IVs).**

10. re:

- **Purpose: Provides regular expression operations.**
- **Features: Although imported, it is not used in the current code.**

11. struct:

- **Purpose: Provides functions to convert between Python values and C structs.**
- **Features: Used to pack and unpack binary data (like the length of the original file extension).**

Conclusion

It is a robust Python application for file encryption and decryption using the AES algorithm. It leverages the tkinter library to create a user-friendly graphical interface, allowing users to select files, enter passwords, and initiate encryption or decryption processes.

Overall, the File Cryptor application effectively combines a user-friendly interface with strong cryptographic techniques to provide a secure file encryption and decryption solution. It demonstrates good practices in cryptography and user interaction, making it a valuable tool for users seeking to protect their sensitive data.