**Document Title:  
Documentation for File Encryption and Decryption Program**

**Author:** Vanshuk  
**Date:** November 9, 2024

### **1. Introduction**

This document provides instructions and a detailed overview of a file encryption and decryption program created in Python. The program is designed to enhance file security through password-based encryption. It has been compiled into an executable (.exe) file for convenient use without needing Python installed. This documentation explains the program's structure, features, and instructions for usage.

### **2. Program Overview**

The encryption program allows users to protect files within a specified directory by encrypting them with a password. The same password is required for decrypting these files. This program leverages Python's cryptography library, ensuring strong encryption standards. Once packaged as an .exe file, the program can be executed directly, making it accessible to users without a Python setup.

### **3. Key Features**

* **Password-Based Security**: Users can provide a password to generate an encryption key, which is used to encrypt and decrypt files.
* **Retry Mechanism**: If the decryption password is incorrect, the program gives users three attempts to re-enter the password before it closes automatically.
* **Standalone Executable**: The Python script has been compiled into an .exe file, allowing it to run independently on Windows systems.

### **4. Code Explanation**

#### **a. Imports and Libraries**

The program uses the cryptography library in Python, specifically the Fernet encryption protocol, to ensure secure, symmetric encryption.

from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

from cryptography.hazmat.primitives import hashes

from cryptography.fernet import Fernet, InvalidToken

import os

import base64

#### **b. Key Generation from Password**

The function generate\_key\_from\_password takes the user-provided password and converts it into a cryptographic key using the PBKDF2 key derivation algorithm. This key enables consistent and secure encryption and decryption with Fernet.

def generate\_key\_from\_password(password):

salt = b'some\_fixed\_salt' # Use a unique salt per session in production

kdf = PBKDF2HMAC(

algorithm=hashes.SHA256(),

length=32,

salt=salt,

iterations=100000

)

key = base64.urlsafe\_b64encode(kdf.derive(password.encode()))

return key

#### **c. Encryption Function**

The encrypt\_files function goes through each file in the specified directory, encrypts it using the generated key, and replaces the original content with the encrypted data. After completion, a message informs the user that encryption was successful.

def encrypt\_files(directory, password):

key = generate\_key\_from\_password(password)

fernet = Fernet(key)

for filename in os.listdir(directory):

filepath = os.path.join(directory, filename)

if os.path.isfile(filepath):

with open(filepath, "rb") as file:

file\_data = file.read()

encrypted\_data = fernet.encrypt(file\_data)

with open(filepath, "wb") as file:

file.write(encrypted\_data)

print(f"{filename} has been encrypted.")

print("Encryption completed successfully. You may now close the program.")

#### **d. Decryption Function with Retry Attempts**

The decrypt\_files function attempts to decrypt each file in the specified directory using the provided password. If an incorrect password is entered, the program allows up to three retry attempts. After three incorrect attempts, the program closes.

def decrypt\_files(directory, password):

key = generate\_key\_from\_password(password)

fernet = Fernet(key)

for filename in os.listdir(directory):

filepath = os.path.join(directory, filename)

if os.path.isfile(filepath):

with open(filepath, "rb") as file:

encrypted\_data = file.read()

try:

decrypted\_data = fernet.decrypt(encrypted\_data)

with open(filepath, "wb") as file:

file.write(decrypted\_data)

print(f"{filename} has been decrypted.")

except InvalidToken:

return False # Incorrect password

return True # Decryption successful

### **5. Using the Program**

#### **a. Starting the Program**

To run the program, double-click the encryption\_script.exe file. This will open a command prompt window.

#### **b. Encryption Instructions**

1. When prompted, type encrypt and press Enter.
2. Enter the full path of the directory containing the files to encrypt.
3. Provide a password. This password will be required for decryption.
4. Upon successful encryption, the program displays: "Encryption completed successfully. You may now close the program."

#### **c. Decryption Instructions**

1. When prompted, type decrypt and press Enter.
2. Enter the directory path containing the encrypted files.
3. Provide the decryption password.
4. If the password is incorrect, you will be prompted to try again, with a maximum of 3 attempts. After 3 failed attempts, the program will display: "Too many unsuccessful attempts. Program will now close."

### **6. Security Recommendations**

* **Password Strength**: The effectiveness of file encryption depends on a strong password. It is recommended to use a unique, complex password.
* **Backup Consideration**: Always back up important files before encrypting, as a forgotten password or incorrect usage may result in data loss.

### **7. Conclusion**

This file encryption and decryption program is a practical tool for securing sensitive files. By using password-based encryption, it provides a straightforward approach to file protection. The retry mechanism in the decryption function ensures a user-friendly experience while maintaining security. Converting the Python script to an .exe file allows easy sharing and usage on Windows systems.

**End of Document**