**PROTECTING USER PASSWORD KEYS AT REST**

protecting user password keys at rest is paramount to ensuring the security and privacy of personal and sensitive information. "At rest" refers to data that is stored on a device or in a database, as opposed to data that is actively being transferred across networks.

The need for robust protection mechanisms for user password keys is driven by several key factors:

1. **Data Breaches:** With the increasing frequency and sophistication of cyber-attacks, data breaches have become a significant threat. If user password keys are not properly protected, they can be easily exploited by malicious actors, leading to unauthorized access to accounts and services.
2. **Regulatory Compliance:** Various regulations and standards, such as the General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS), mandate stringent measures for protecting sensitive data, including password keys. Non-compliance can result in severe penalties and reputational damage.
3. **User Trust:** Users expect their personal information to be handled with the highest level of security. Failure to protect user password keys can erode trust, leading to a loss of customers and a negative impact on the organization's brand.

To mitigate these risks, several best practices and technologies are employed to protect user password keys at rest:

1. **Encryption:** Encrypting password keys using strong encryption algorithms ensures that even if the data is accessed by unauthorized individuals, it remains unreadable without the appropriate decryption key.
2. **Hashing:** Password hashing involves converting the original password into a fixed-length string of characters, which is stored instead of the actual password. Hash functions are designed to be one-way, making it extremely difficult to reverse-engineer the original password from the hash.
3. **Salting:** Adding a unique random value, known as a salt, to each password before hashing ensures that even identical passwords result in different hashes. This prevents attackers from using precomputed tables (rainbow tables) to crack passwords.
4. **Key Management:** Proper management of encryption keys is crucial. This includes using secure key storage solutions, regularly rotating keys, and implementing access controls to restrict who can access the keys.
5. **Access Controls and Monitoring:** Implementing strict access controls ensures that only authorized personnel have access to password keys. Additionally, continuous monitoring and logging of access to these keys can help detect and respond to any suspicious activity.

In conclusion, protecting user password keys at rest is an essential aspect of securing sensitive data and maintaining user trust. By employing a combination of encryption, hashing, salting, key management, and access controls, organizations can significantly reduce the risk of data breaches and comply with regulatory requirements.

**FEATURES**

**1.AES Encryption:** Utilizes AES (Advanced Encryption Standard) with a 256-bit key size for robust data encryption, ensuring confidentiality of files and directories.

**2.Passphrase-based Encryption:** Allows users to securely encrypt files and directories using a passphrase, ensuring only authorized users can decrypt them.

**3.File and Directory Handling:**

* Supports encryption and decryption of both individual files and entire directories.
* Preserves directory structure during encryption and restores it during decryption, maintaining organizational integrity.

**4.Key Management:**

* Generates random file encryption keys for each encrypted file, enhancing security.
* Derives a secure key from the user-provided passphrase and a randomly generated salt using PBKDF2 for encryption key encryption.

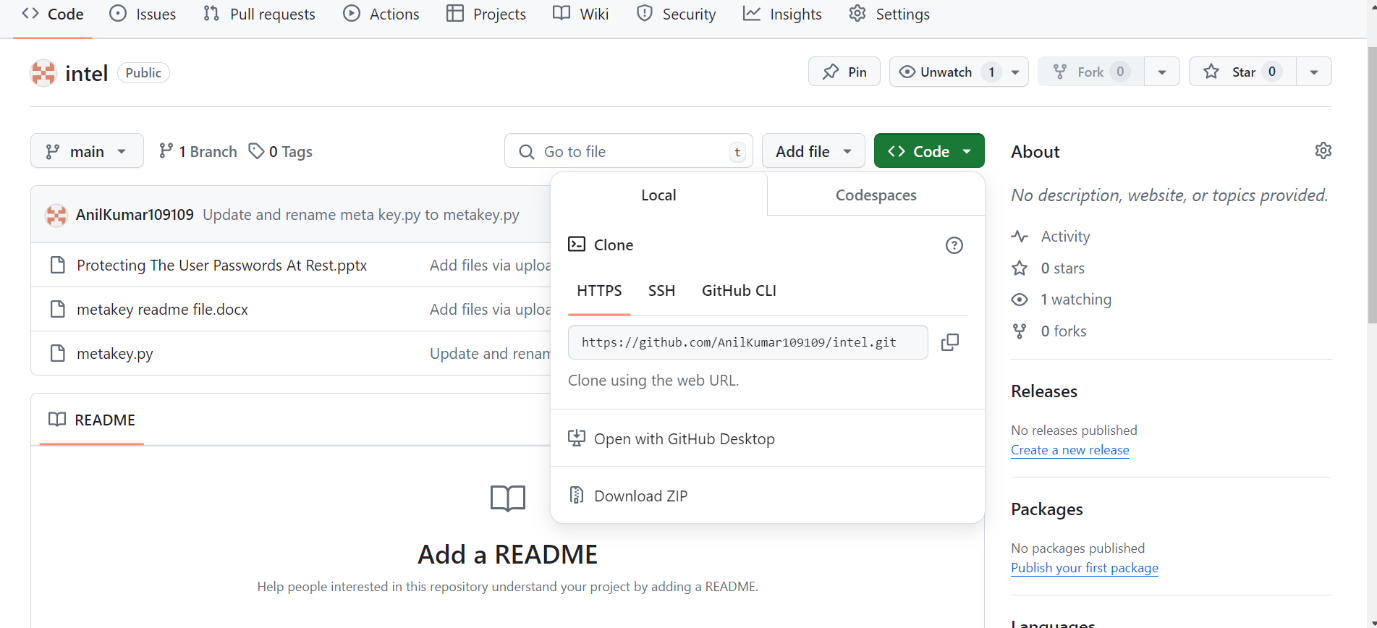
**5.Secure Storage of Encrypted Keys:**

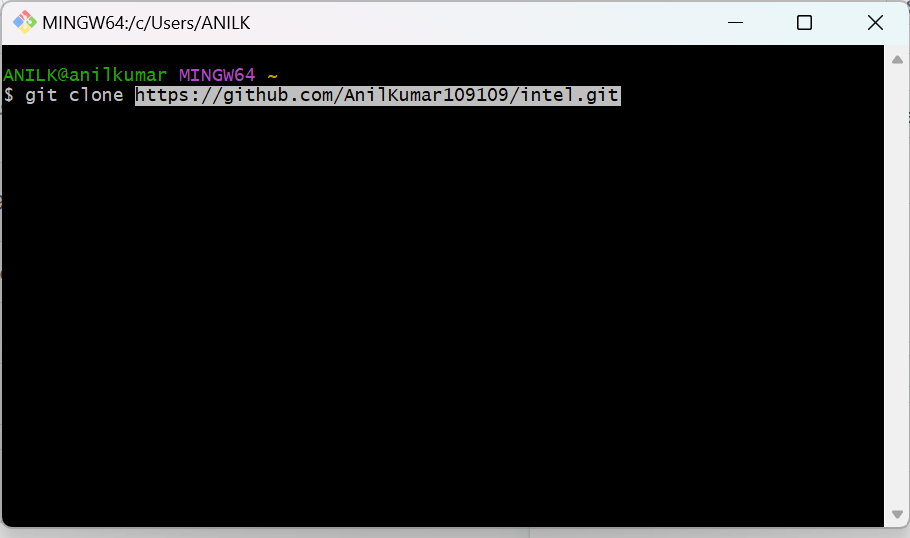
* Encrypts file encryption keys using AES-CFB mode with a key derived from the passphrase and stores them alongside the encrypted data.
* Ensures the integrity of encrypted file keys using HMAC-SHA256 for authentication.

**6.Graphical User Interface (GUI):**

* Provides a user-friendly GUI using tkinter, allowing easy file selection, passphrase input, and interaction with encryption and decryption functionalities.
* Supports browsing for files and folders, simplifying user interaction with the application.

**PROCESS TO INSTALL PROTECTING USER PASSORDS AT REST (ON DISK)ON WINDOWS USING GITBASH**

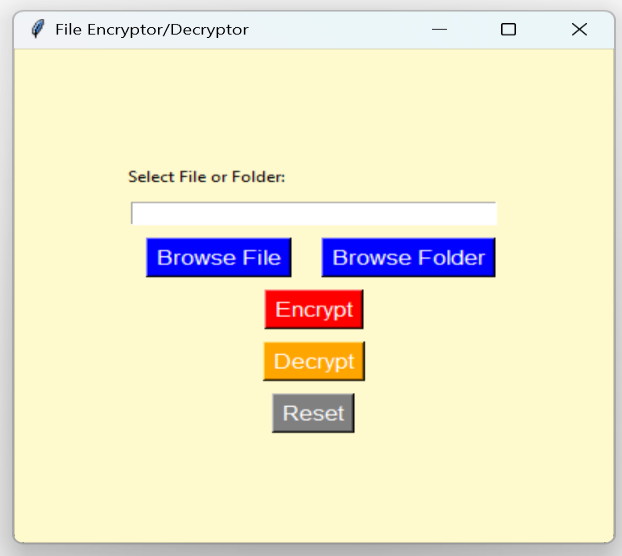
* Getting the link of the repository from github.
* 
* First we need to run the git clone of that repository



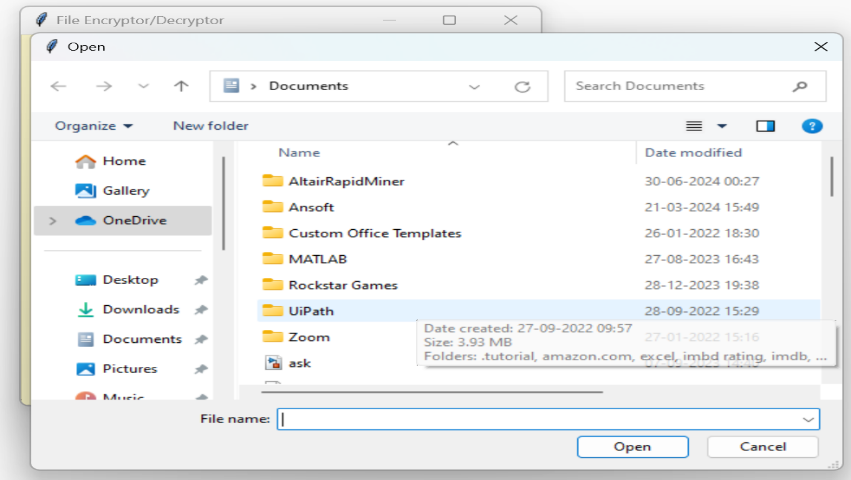
* Next we need to run ls to see what is that in directory
* Next we need to run cd intel in order to enter the directory.
* Next we need to run ls after getting into intel directory.
* Next we need to run python metakey.py to display the dialog box

FILE AND FOLDER ENCRYPTION AND DECRYPTION PROCESS:

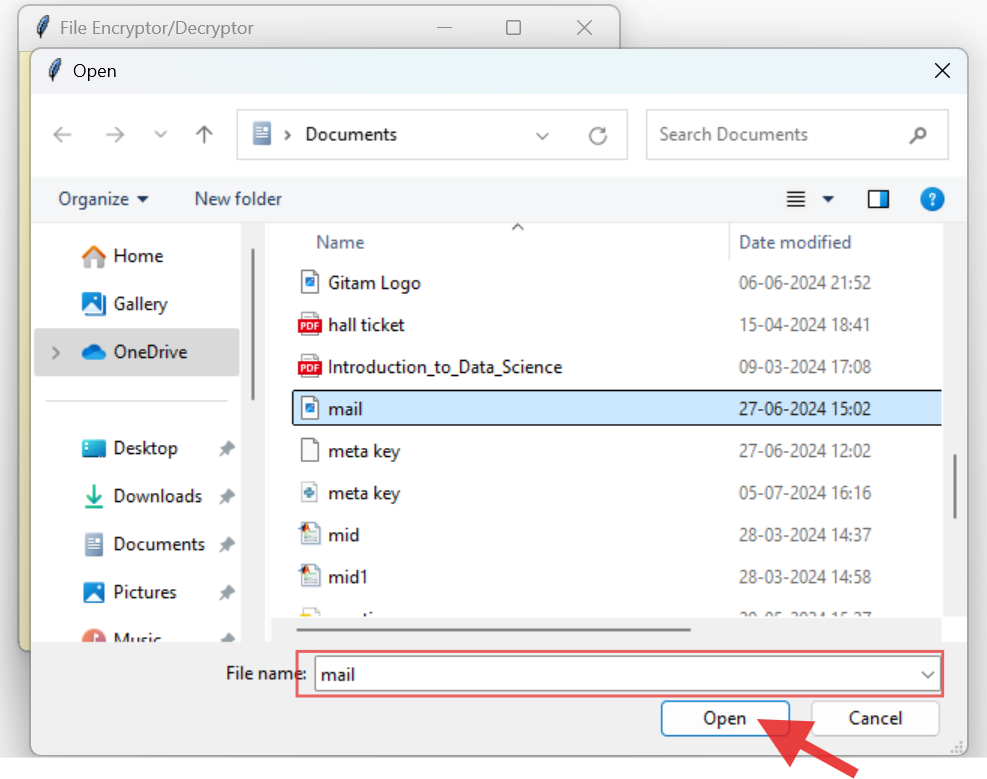
***FILE ENCRYPTION PROCESS***:

STEP: Upon Successful implementation of python program in the VSCode, the File Encryptor/ Decryptor dialog box will appear on the screen.

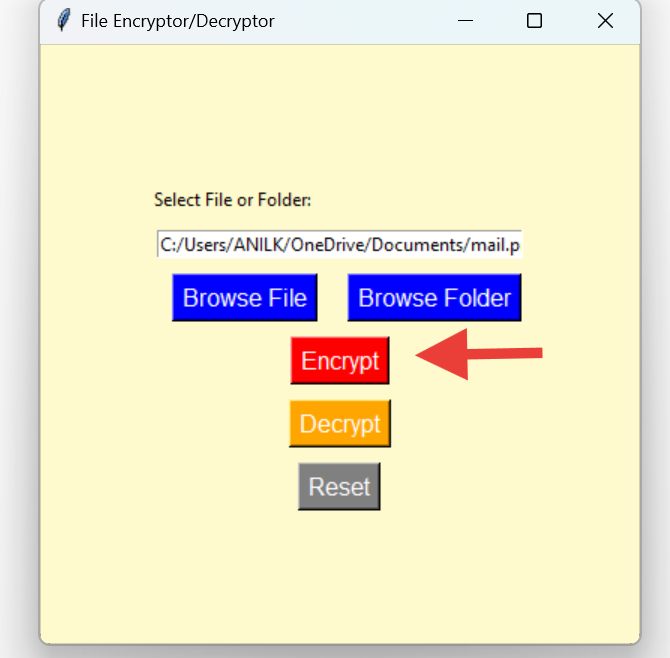
Step: click Browse File to browse the file that you need to encrypt.

Step: Select which file you need to encrypt.

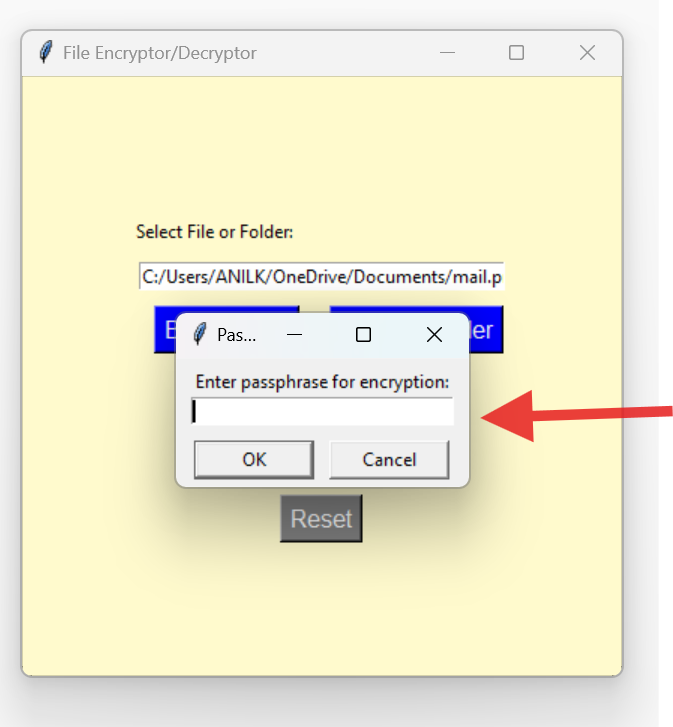
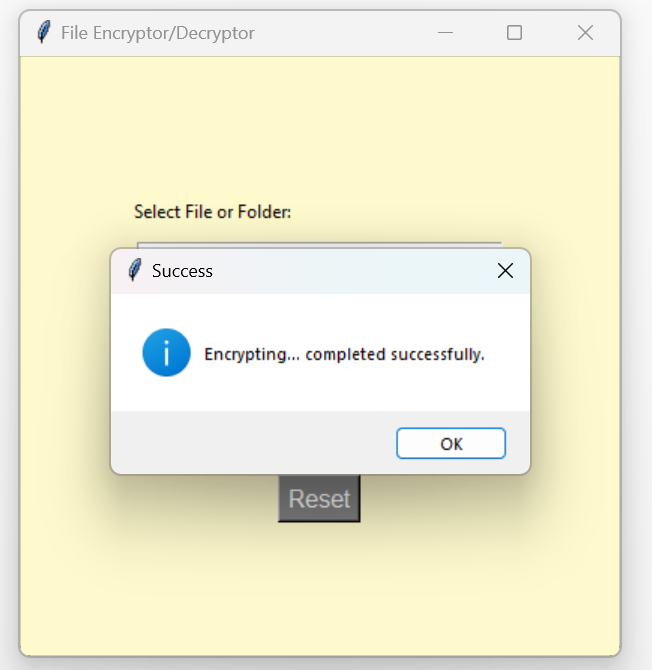
Step: Select which file you need to encrypt. click open.

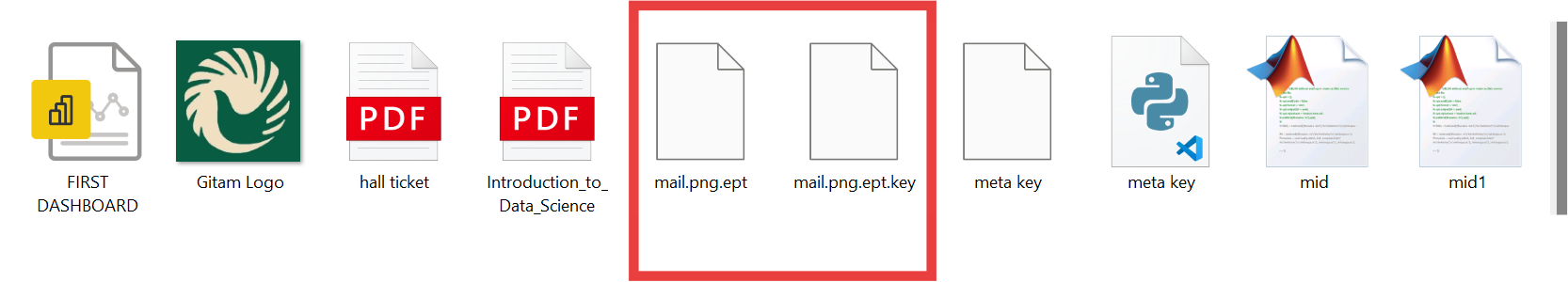


Step: click encrypt.

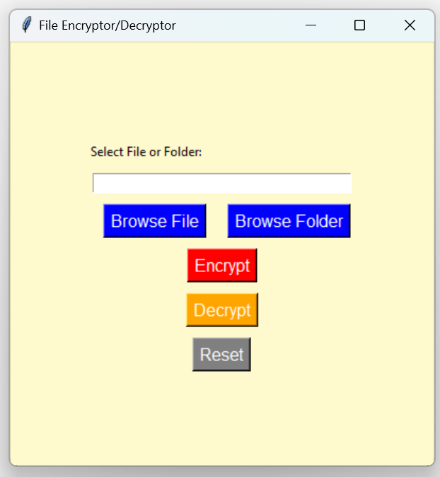


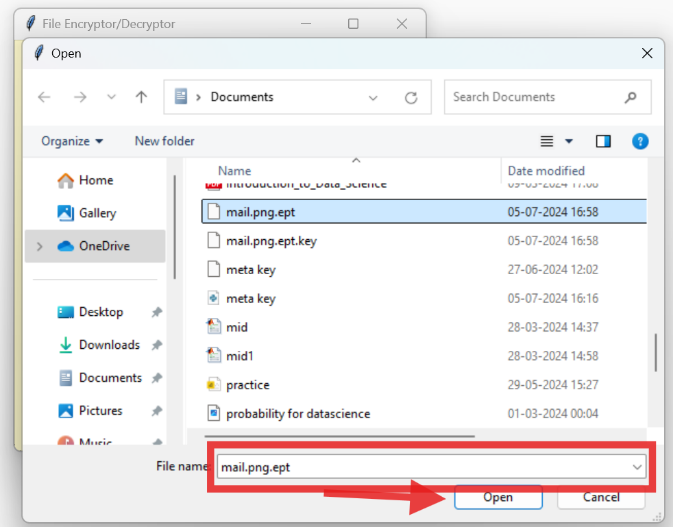
Step: Enter Passphrase for Encryption

step: After entering the passphrase, encryption is completed successfully.

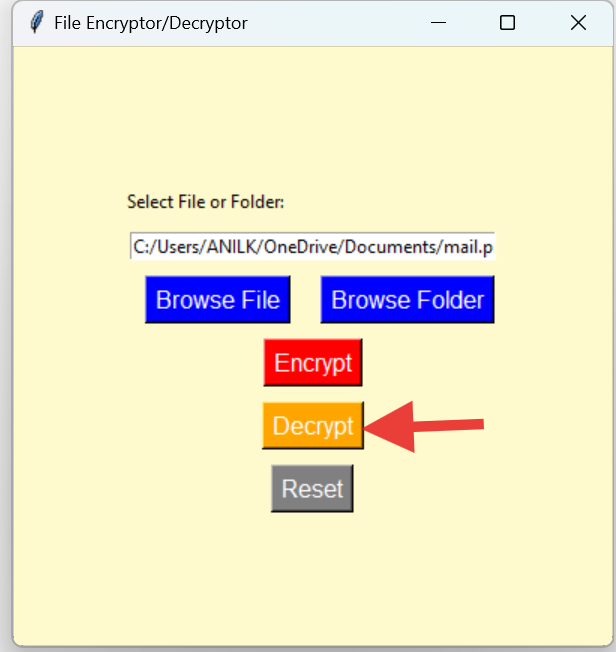
Step: After encryption, an encrypted file and key file are generated.

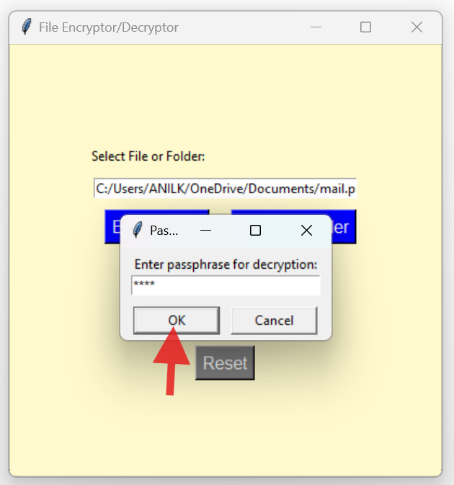
***FILE DECRYPTION PROCESS:***

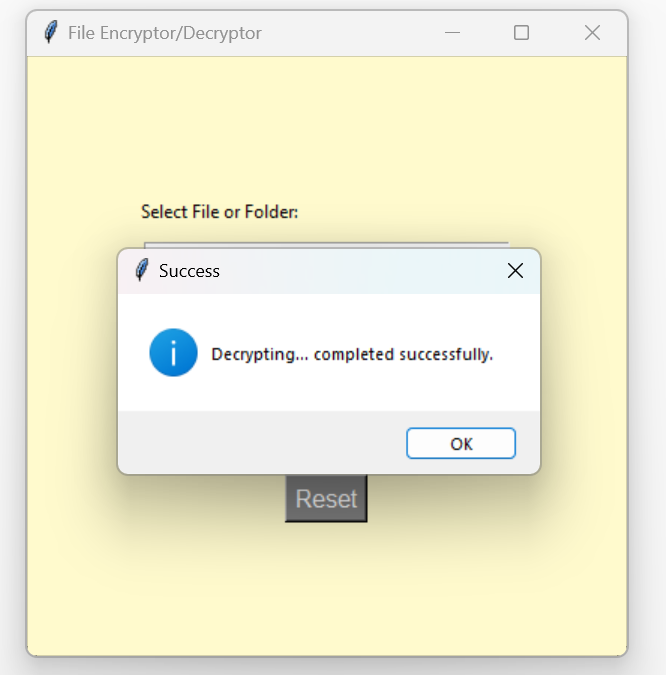
STEP: Click OK and then browse the file you need to decrypt.

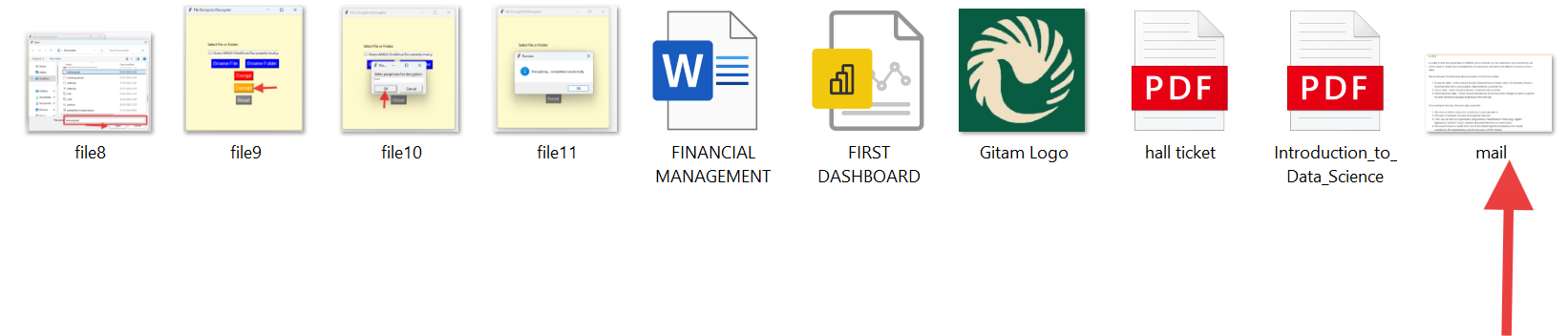
Step: After selecting the encrypted file, click open.

Step: click Decrypt to decrypt the encrypted file

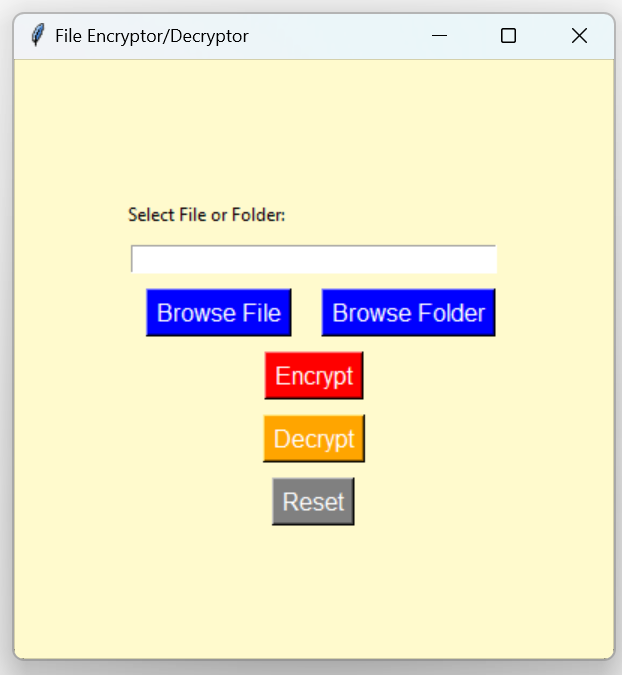


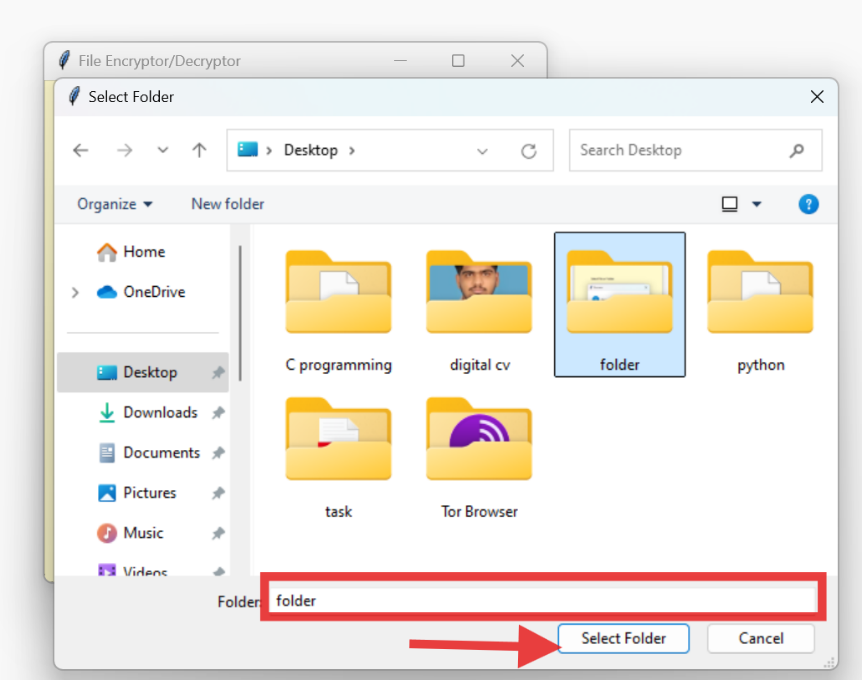
Step: After entering the passphrase for decryption, click ok.

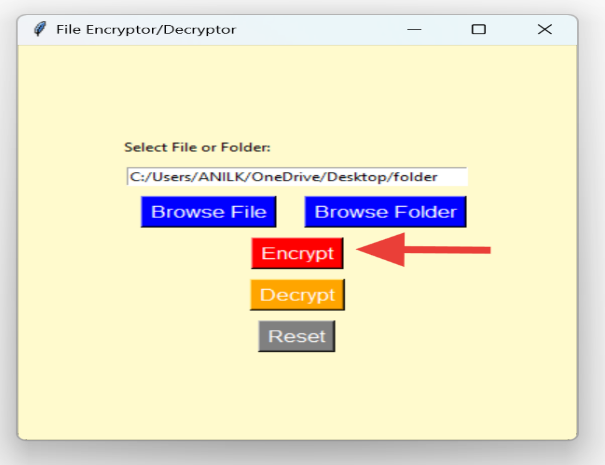
Step: After clicking Ok, the file was successfully decrypted.

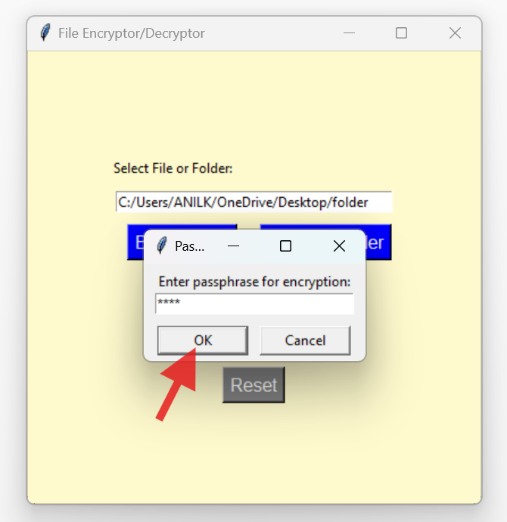


***FOLDER ENCRYPTION PROCESS:***

Step: click browse folder to encrypt the folder

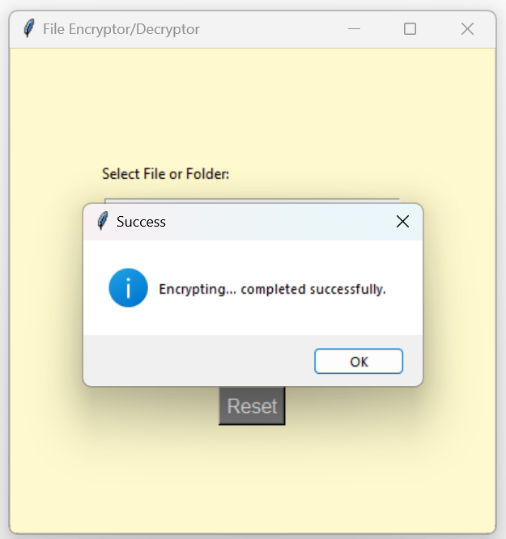
Step: Select the folder and click select folder

Step: Click encrypt to encrypt the folder

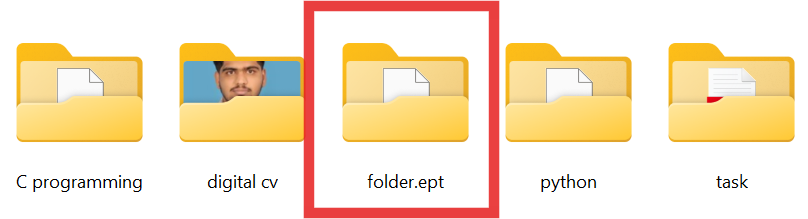
Step: After entering the passphrase for encryption click OK.

Step: After clicking "OK," the folder was successfully

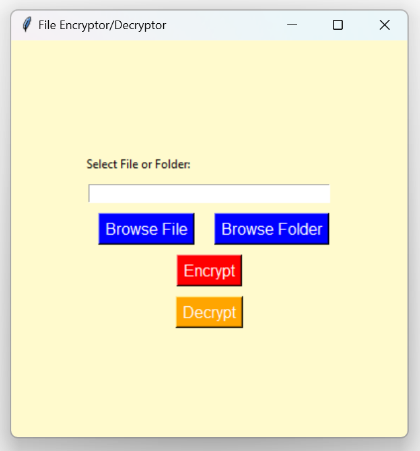
encrypted.

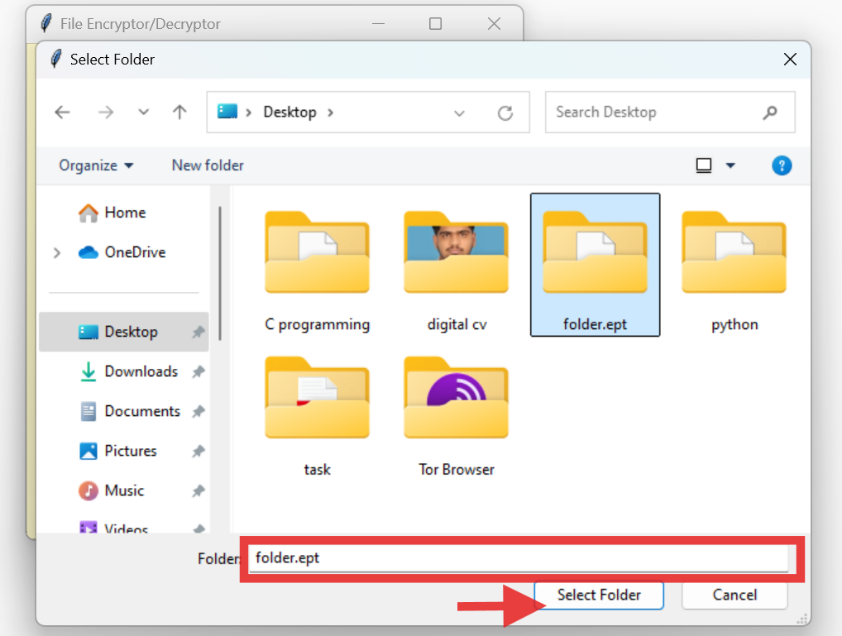


Step: encrypted folder “folder.ept” was generated.

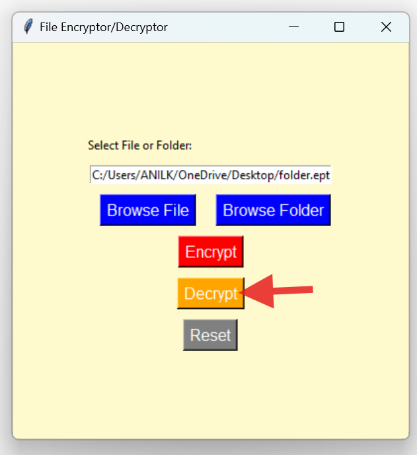
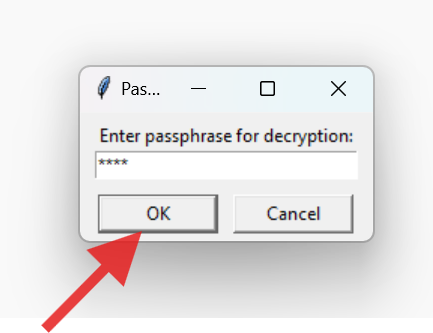


***FOLDER DECRYPTION PROCESS:***

Step: Click Browse Folder to decrypt the folder.

Step: After selecting encrypted folder, click select folder.

Step: Click Decrypt to decrypt the folder.

Step: After entering the passphrase for decryption click OK

Step: The folder was successfully decrypted.

