File Encryption/Decryption Tool using Python and Cryptography

Title:

File Encryption/Decryption Tool using Python and the Cryptography Library.

Summary:

This project involves the development of a simple file encryption and decryption tool using Python and the Cryptography library. It enables secure communication by converting plain text files into encrypted files and then back into readable text using a secret key.

Objective:

To create a Python-based encryption/decryption system that can:

- Securely encrypt text files
- Decrypt them back using the same secret key
- Help understand practical cryptographic applications using the Fernet module of the cryptography library

Technologies Used:

- Programming Language: Python

- Library: `cryptography` (Fernet)
- Editor: Notepad
- Terminal: Command Prompt (CMD)

Why I Chose This Project:

I chose this project to explore encryption and decryption techniques, which are fundamental to securing sensitive information. As cybersecurity is a field, I'm passionate about building this tool allows me to gain practical experience with the **cryptography** library in Python and understand real-world applications of encryption. This project also aligns with my goal of enhancing my skills in data security and cryptographic principles.

What is Encryption/Decryption?

Encryption converts readable data into unreadable ciphertext using a key. **Decryption** is the reverse process, which restores the ciphertext back to its original readable form.

What is Cryptography?

Cryptography is a technique used to protect information by transforming it into a secure format. Python's `cryptography` library offers high-level encryption capabilities.

Why Fernet?

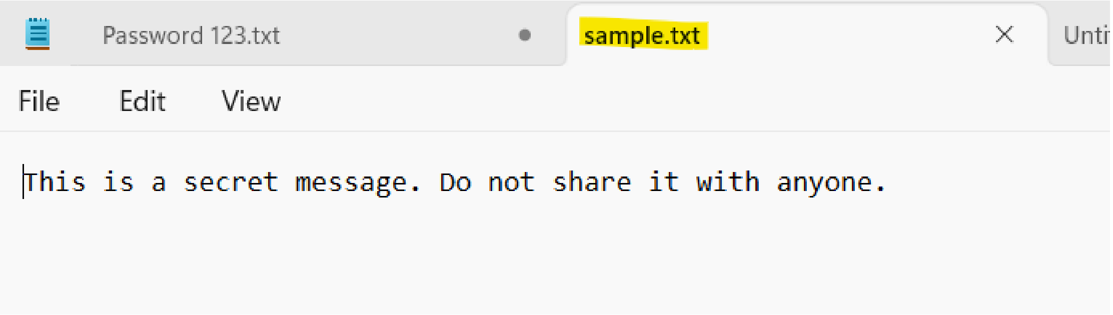
Fernet is a symmetric encryption tool provided by the `cryptography` library. It:

- Uses AES for secure encryption
- Ensures data integrity and confidentiality
- Is easy to implement

Project Workflow

Step 1: Create a Normal Text File

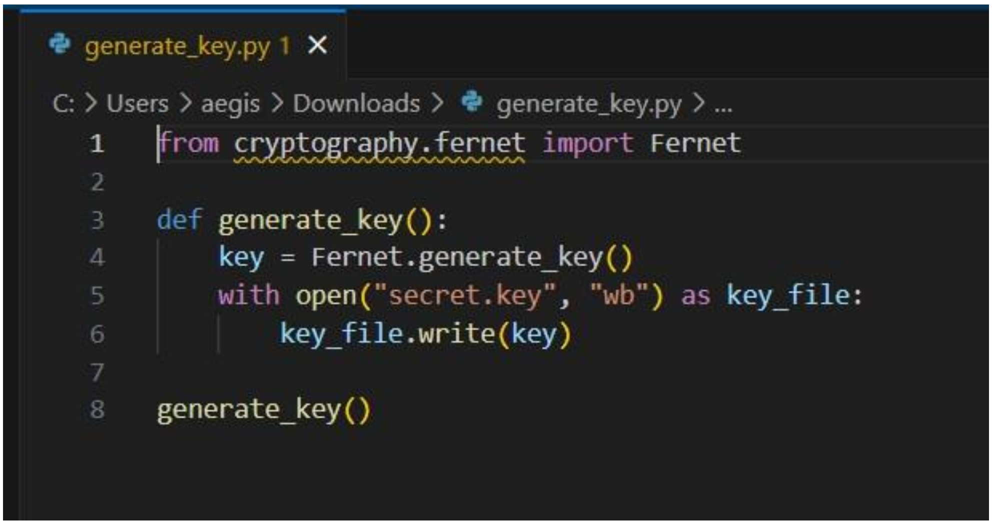
- Create a file named `sample.txt`
- Add some secret message



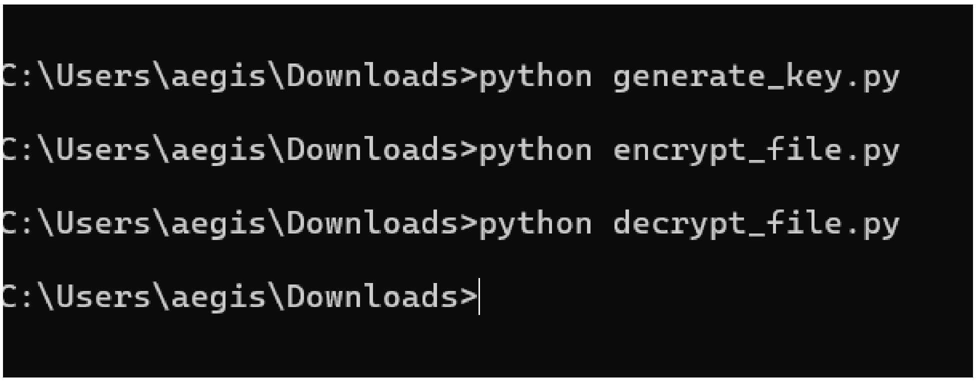
This is a secret message. Do not share it with anyone.

Step 2: Generate Secret Key

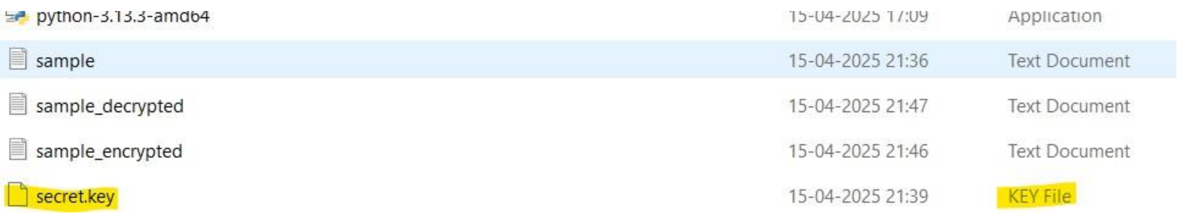
- Generate a secret key and save as `generate_key.py` and run in command prompt, then the secret key will be created as name



```python
from cryptography.fernet import Fernet

def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)

generate_key()
```

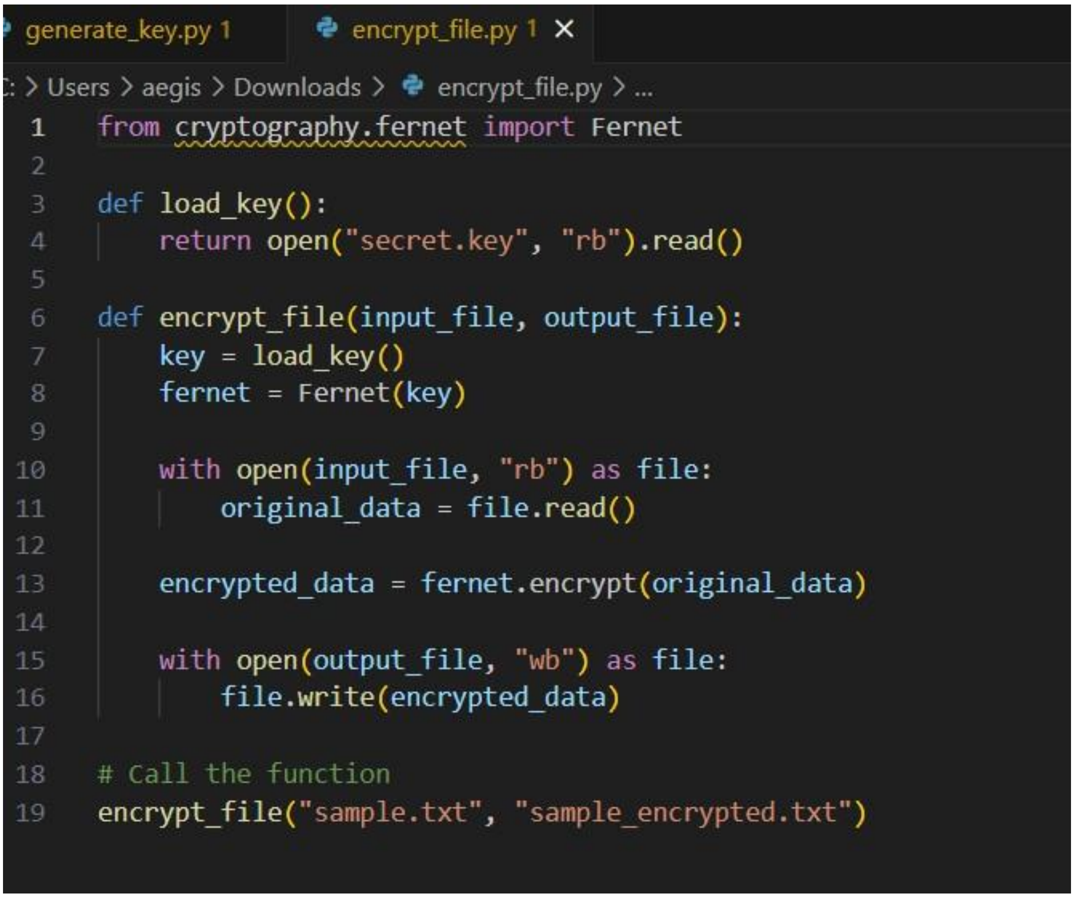


```
C:\Users\aegis\Downloads>python generate_key.py

C:\Users\aegis\Downloads>python encrypt_file.py

C:\Users\aegis\Downloads>python decrypt_file.py

C:\Users\aegis\Downloads>
```



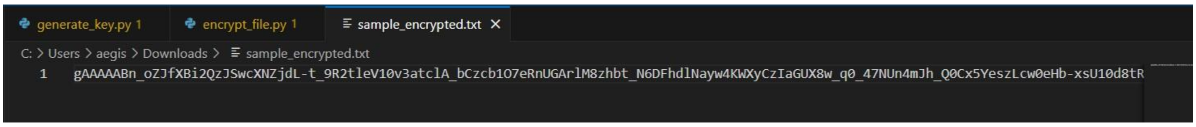
python-3.13.3-amd64	15-04-2025 17:09	Application
sample	15-04-2025 21:36	Text Document
sample_decrypted	15-04-2025 21:47	Text Document
sample_encrypted	15-04-2025 21:46	Text Document
secret.key	15-04-2025 21:39	KEY File

Step 3.1: Encrypt the File



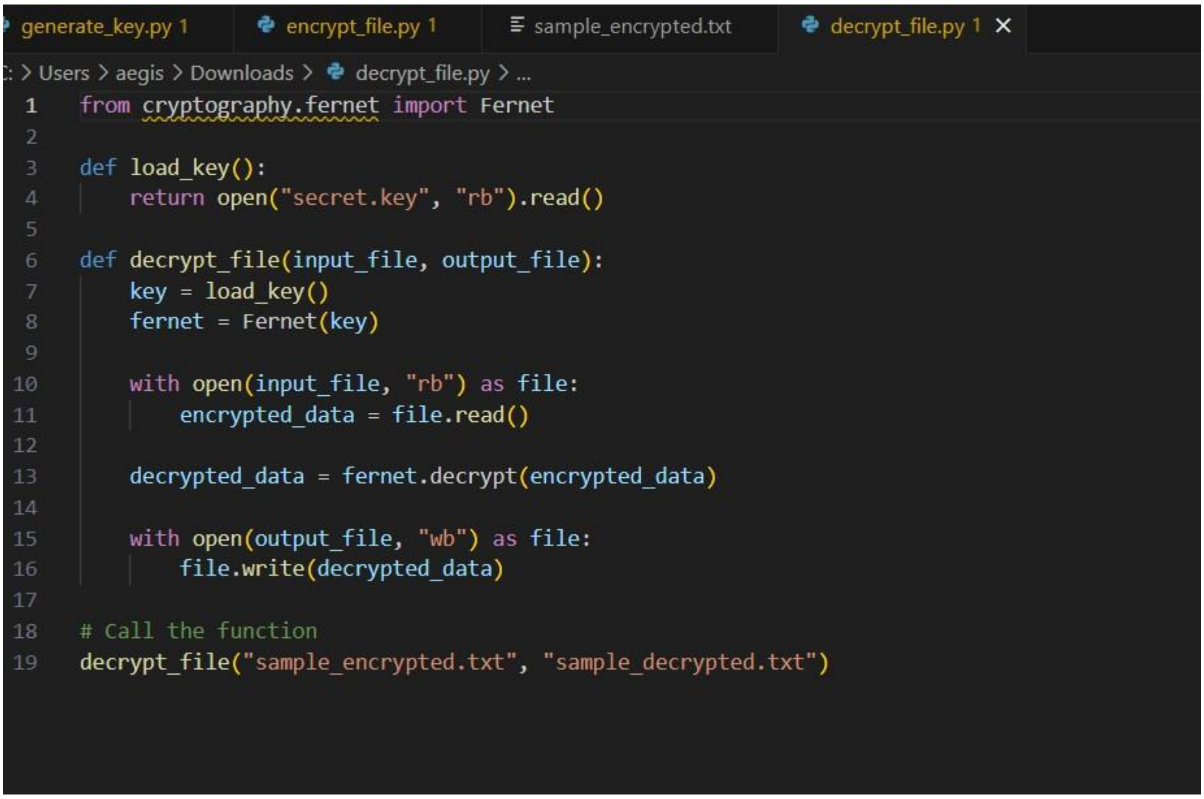
```python
from cryptography.fernet import Fernet

def load_key():
    return open("secret.key", "rb").read()

def encrypt_file(input_file, output_file):
    key = load_key()
    fernet = Fernet(key)

    with open(input_file, "rb") as file:
        original_data = file.read()

    encrypted_data = fernet.encrypt(original_data)

    with open(output_file, "wb") as file:
        file.write(encrypted_data)

# Call the function
encrypt_file("sample.txt", "sample_encrypted.txt")
```

After encrypting the new file created name is "sample_encrypted.txt"



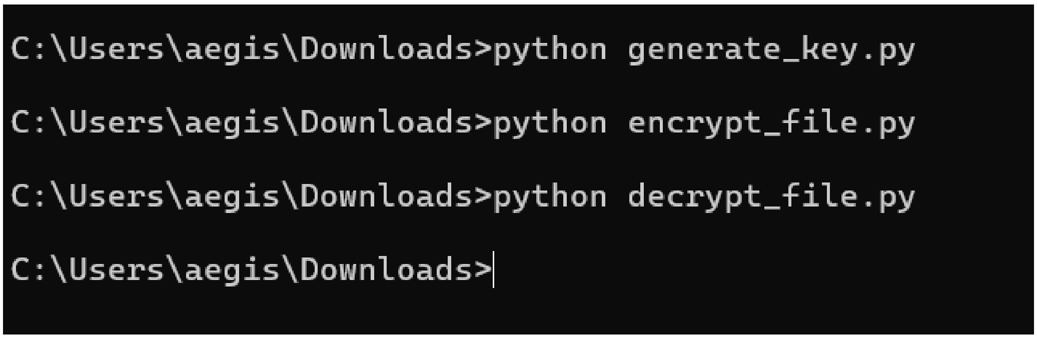
```
gAAAAABn_oZJfXBi2QzJSwcXNZjdL-t_9R2tleV10v3atclA_bCzcb1O7eRnUGArlM8zhbt_N6DFhdlNayw4KWXyCzIaGUX8w_q0_47NUn4mJh_Q0Cx5YeszLcw0eHb-xsU10d8tR
```

Step 3.2: Decrypt the File

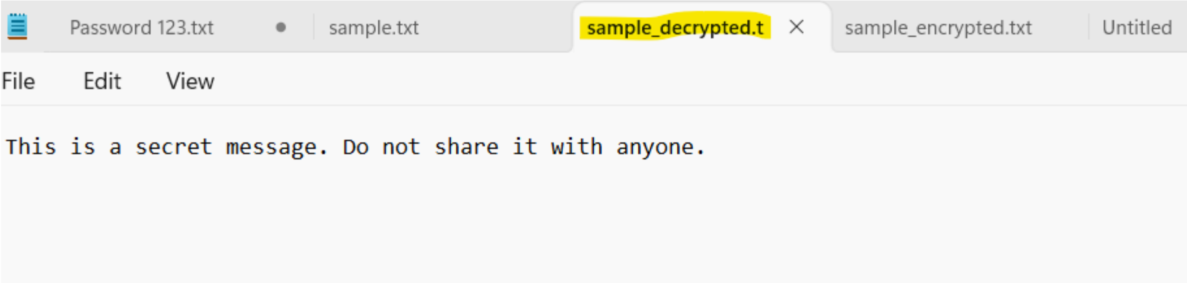


```python
from cryptography.fernet import Fernet

def load_key():
    return open("secret.key", "rb").read()

def decrypt_file(input_file, output_file):
    key = load_key()
    fernet = Fernet(key)

    with open(input_file, "rb") as file:
        encrypted_data = file.read()

    decrypted_data = fernet.decrypt(encrypted_data)

    with open(output_file, "wb") as file:
        file.write(decrypted_data)

# Call the function
decrypt_file("sample_encrypted.txt", "sample_decrypted.txt")
```

Run in command prompt



```
C:\Users\aegis\Downloads>python generate_key.py

C:\Users\aegis\Downloads>python encrypt_file.py

C:\Users\aegis\Downloads>python decrypt_file.py

C:\Users\aegis\Downloads>
```



```
This is a secret message. Do not share it with anyone.
```

File Flow Summary

- **sample.txt**: This is the original plain text file created by the user. It contains a readable message that needs to be protected.
- **secret.Key**: This file contains the secret key generated using the Fernet module. It is used for both encryption and decryption.
- **sample_encrypted.txt:** This is the encrypted version of the original file. The contents are unreadable and protected.
- **sample_decrypted.txt:** This is the decrypted version of the encrypted file. It restores the original readable content using the same secret key.

Conclusion:

The project successfully demonstrates how to implement encryption and decryption of text files using a secret key with Python. The Fernet module of the cryptography library makes it easy to secure data efficiently.

Learning Outcomes

- Technical Proficiency: I gained practical experience in various cybersecurity tools and techniques, including vulnerability scanning, penetration testing, and cryptographic analysis.
- Understanding Cybersecurity Lifecycle: I developed a comprehensive understanding of the steps involved in securing systems, from initial vulnerability assessment to implementing mitigations.
- Problem-Solving Skills: Tackling complex security challenges enhanced my analytical thinking and my ability to devise effective solutions under pressure.
- Professional Development: My experience improved my ability to work in a team, communicate technical information clearly, and manage time efficiently in a fast-paced environment.

Challenges and Solutions

- **Adapting to Complex Tools:** Initially, mastering advanced cybersecurity tools like Metasploit and Wireshark was challenging. I overcame this by dedicating time to study tutorials and practicing in a simulated environment, which significantly improved my proficiency.
- **Handling Advanced Security Scenarios:** The complexity of real-world security threats requires a deep understanding of underlying principles. I tackled this by engaging in continuous learning and seeking guidance from my coordinator and team members.

Conclusion

My internship at Hack Secure was an enriching experience that significantly expanded my knowledge and skills in cybersecurity. The practical exposure to real-world security challenges and the application of advanced tools have solidified my interest in pursuing a career in cybersecurity. This experience has been instrumental in preparing me for the complexities of the cybersecurity field.

Acknowledgment

I express my sincere gratitude to Hack Secure, especially my mentor, Mr. Aman Pandey, and assistant mentor Mr. Prabhat Raj, for their guidance and support throughout my internship. I also thank Amrita Vishwa Vidyapeetham for providing this internship opportunity, which has been crucial in my personal and professional development. This report encapsulates the essence of my internship experience, highlighting the integration of academic knowledge with practical skills in a professional setting. It reflects my journey of learning, growth, and development in the field of cybersecurity