Unit 8 Seminar

Title: Cryptography Programming Exercise

Read the Cryptography with Python blog at tutorialspoint.com (the link is in the reading list). Select one of the methods described/ examples given and create a Python program that can take a short piece of text and encrypt it.

Create a Python program in Codio (you can use the Jupyter Notebooks space provided in the Codio resources section) that can take a text file and output an encrypted version as a file in your folder on the Codio system. Demonstrate your program operation in this week’s seminar session.

Answer the following questions in your e-portfolio:

- Why did you select the algorithm you chose?
- Would it meet the GDPR regulations? Justify your answer.

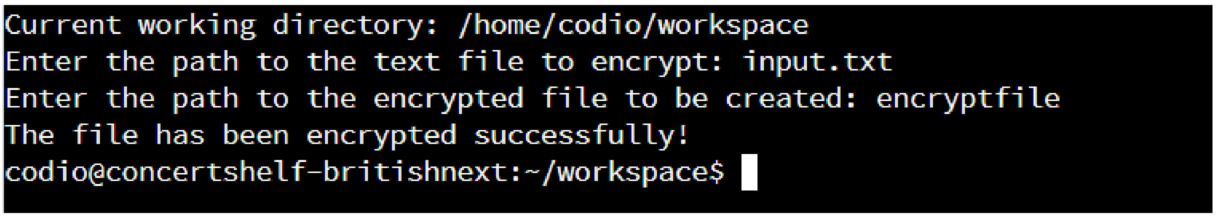
My Answers:

Cryptography is a technique that keeps messages private and secure by hiding them. It involves four key components - plain text (the original message), cipher text (the encrypted message), encryption (the process of converting plain text to cipher text), and decryption (the process of converting cipher text back to plain text). Hence, cryptography is a necessary process that ensures the privacy and security of messages (Tutorialspoint, N.D.).

Below is my code based on ideas of python.hotexamples.com. (N.D.):

```
"""Python program to encrypt a text file"""
import os
import cryptography.fernet as fernet


print("Current working directory:", os.getcwd())

def generate_key():
    """Generates a random encryption key."""
    return fernet.Fernet.generate_key()

def encrypt_file(input_file, output_file, key):
    """Encrypts a text file using Fernet."""
    # Read the contents of the input file into a byte object.
    with open(input_file, "rb") as f:
        input_data = f.read()

    # Initialize a Fernet object with the encryption key.
    f = fernet.Fernet(key)

    # Encrypt the byte object using the Fernet object.
    encrypted_data = f.encrypt(input_data)

    # Write the encrypted byte object to the output file.
    with open(output_file, "wb") as f:
        f.write(encrypted_data)

def main():
    """main"""
    # Generate a random encryption key.
    key = generate_key()

    # Get the paths to the input and output files.
    input_file_path = input("Enter the path to the text file to
encrypt: ")
    output_file_path = input("Enter the path to the encrypted
file to be created: ")

    # Encrypt the file.
    encrypt_file(input_file_path, output_file_path, key)

    # Print a success message.
    print("The file has been encrypted successfully!")



if __name__ == "__main__":
    main()
```

Output:



```
Current working directory: /home/codio/workspace
Enter the path to the text file to encrypt: input.txt
Enter the path to the encrypted file to be created: encryptfile
The file has been encrypted successfully!
codio@concertshelf-britishnext:~/workspace$
```

1. Why did I choose the Fernet algorithm?

I opted for the Fernet algorithm as it is a trusted symmetric encryption method offered by the cryptography library in Python. It operates on the Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode with PKCS7 padding, ensuring secure encryption and decryption of data using a shared secret key. Moreover, it provides data integrity and authentication using a message authentication code (MAC) based on HMAC. Fernet's added advantage of PKCS7 padding ensures secure handling of variable-length plaintexts. Its ease of use and robust security features have made it a popular choice in Python, with plenty of resources available to support implementation (Mate, 2023).

2. Would the Fernet algorithm meet the GDPR regulations?

Yes, the Fernet algorithm is fully compliant with GDPR. Intersoft Consulting (2019) states that the GDPR mandates suitable technical and organisational measures to safeguard personal data. Hence, the Fernet algorithm is a highly regarded and extensively utilised encryption.

Furthermore, Allen (2023) emphasises that data at rest carries less risk than data in transit. However, unencrypted data can still be accessed through vulnerabilities, insider threats, and phishing attacks. To ensure GDPR compliance and data protection, it is crucial to encrypt data at rest with securely managed keys. The Fernet algorithm is highly effective in encrypting data in both scenarios.

References:

Tutorialspoint (N.D.). *Cryptography with Python - Quick Guide - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_quick_guide.htm.

python.hotexamples.com. (N.D.). *Python Fernet.encrypt Examples, cryptography.fernet.Fernet.encrypt Python Examples - HotExamples*. [online] Available at: https://python.hotexamples.com/examples/cryptography.fernet/Fernet/encrypt/python-fernet-encrypt-method-examples.html.

Mate, E. (2023). *Text Encryption Methods for Securing Credentials and Secrets Sharing*. [online] Medium. Available at: https://medium.com/@ekantmate/text-encryption-methods-for-securing-credentials-and-secrets-sharing-f1550e4a1d03.

Intersoft Consulting (2019). *Encryption* | *General Data Protection Regulation (GDPR)*. [online] General Data Protection Regulation (GDPR). Available at: https://gdpr-info.eu/issues/encryption/.

Allen, C. (2023). *Encryption for GDPR Compliance*. [online] www.cryptomathic.com. Available at: https://www.cryptomathic.com/news-events/blog/encryption-for-gdpr-compliance.