MESSAGE ENCRYPTION DECRYPTION USING ONE TIME PAD

Submitted in partial fulfillment of the requirements of the degree of

Bachelor of Engineering

by

Name	Roll No
Agarwal Gunjan Hemant	312002
Ishan Ahmed	312018
Tamim Ahmad	312055
Tarmale Sudarshan Sanjay	312056

Project Guide:

Prof. Ahlam Ansari
&
Prof. Asadullah Shaikh

(Computer Engineering)

M.H. Saboo Siddik College of Engineering University of Mumbai
2022-23

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING
8, Saboo Siddik Road, Byculla, Mumbai - 400 008.

This is to certify that,

Name	Roll No
Agarwal Gunjan Hemant	312002
Ishan Ahmed	312018
Tamim Ahmad	312055
Tarmale Sudarshan Sanjay	312056

Of Third Year (T.E. Semester VI) degree course in Computer Engineering, have completed the specified project report on,

MESSAGE ENCRYPTION DECRYPTION USING ONE TIME PAD

As partial fulfillment of the project work in a satisfactory manner as per the rules of the curriculum laid by the University of Mumbai, during the Academic Year Jan 2023 - May 2023.

Internal Guide

Project Report Approval for T. E.

This project report entitled **“Message Encryption Decryption using One Time Pad”** by **Agarwal Gunjan Hemant, Ishan Ahmed, Tamim Ahmad and Tarmale Sudarshan Sanjay** is approved for the degree of Computer Engineering.

EXAMINERS

1. ______________________________

2. ______________________________

SUPERVISORS

1. ______________________________

2. ______________________________

Date:

Place: Mumbai

Acknowledgment

We would like to express our gratitude and appreciation to our parents for motivating and encouraging us throughout the career.

We wish to express our sincere thanks to our Director Dr. Mohiuddin Ahmed and our Principal Dr. Ganesh Kame, M.H. Saboo Siddik College of Engineering for providing us all the facilities, support, and wonderful environment to meet our project requirements.

We would also take the opportunity to express our humble gratitude to our Head of the Department of Computer Engineering Dr. Zainab Pirani for supporting us in all aspects and for encouraging us with her valuable suggestions to make our project successful.

We are highly thankful to our internal project guide Prof. Ahlam Ansari and Prof. Asadullah Shaikh whose valuable guidance helped us understand the project better, her constant guidance and willingness to share her vast knowledge made us understand this project and its manifestations in great depth and helped us to complete the project successfully.

We would also like to acknowledge with much appreciation the role of the Computer Department staffs, especially the Laboratory staff, who permitted us to use the labs when needed and the necessary material to complete the project.

We would like to express our gratitude and appreciate the guidance given by other supervisors and project guides, their comments and tips helped us in improving our presentation skills.

Although there may be many who remain unacknowledged in this humble note of appreciation, there are none who remain unappreciated.

Table of Content

Abstract

The implementation of a message encryption and decryption system utilising the One Time Pad approach, a very safe cryptographic algorithm, is the main goal of this small project. The encryption technique obscures the original message by creating a random key that is the same length as it, and the decryption procedure entails utilising that same key to restore the message. Our solution performs the encryption and decryption procedures using the ASCII values of the characters in the message and key.

The user can input a message and key of the same length to encrypt and decrypt messages, respectively. Our implementation offers a simple and effective way to encrypt and decrypt messages using One Time Pad, which relies on the randomness of the key and the key's one-time use to guarantee security. This mini project provides an excellent starting point for anyone interested in learning One Time Pad encryption and decryption, while also providing a foundation for more advanced cryptographic techniques.

1. Introduction

In order to protect sensitive information from unauthorised access, message encryption is a crucial approach. The One-Time Pad (OTP) encryption algorithm, a cryptographic method that ensures full secrecy, will be used in this project.

The plaintext is encrypted by the OTP algorithm using a random key that is only used once. It offers absolute secrecy, meaning that even with infinite computing capacity, an adversary who intercepts the encrypted message will not be able to deduce any useful information about the plaintext. The randomness and single use of the key are essential components of OTP encryption security.

In this project, we'll put a Python programme in place that enables users to input plaintext messages and produce randomly generated keys. The message will then be encrypted by the programme using the key and the OTP algorithm. By using the same key that was used for encryption, the user can also decrypt the ciphertext. We will construct the OTP algorithm and create random keys using Python's built-in random package. A straightforward and user-friendly interface will be provided by the programme so that users may enter plaintext messages, encrypt them with OTP, and decrypt them with the same key. The OTP encryption technique will be practically implemented as part of this project, enabling the safe transmission of important messages while preventing unauthorised access.

The early 1900s saw the invention of the traditional cryptographic method known as one-time pad encryption. Despite being more than a century old, it is still a strong encryption algorithm that, when used properly, offers perfect secrecy. Many secure communication systems, including those used for military and diplomatic communication, still employ the OTP algorithm.

The use of OTP encryption and decryption in this project will serve as an example of how crucial it is to secure sensitive data using robust encryption methods. In today's linked world, maintaining the confidentiality of sensitive information is essential, and encryption is an essential instrument to do it. Users can take precautions to guarantee their data is private and secure by knowing the fundamentals of encryption and adopting a strong encryption method like OTP. The initiative will assist in educating users on the fundamentals of secure communication and the significance of encryption in safeguarding private data.

.

2. Implementation

Implementing Message Encryption Decryption using One Time Pad involves several key steps, including:

Planning:

- Plan to implement a message encryption and decryption system using the One Time Pad technique.
- Define the input/output requirements and overall functionality of the system.
- Choose Python as the programming language and tools needed for implementation, such as a text editor or IDE.
- Plan the encryption and decryption algorithms using One Time Pad, utilizing Python libraries such as the random module for generating a random key and the ord() and chr() functions to convert characters to their ASCII values.
- Decide on the testing procedures to ensure the system's accuracy and security.

Language used:

- Use Python as the programming language for implementation of the encryption and decryption algorithms using One Time Pad.
- Utilize built-in Python libraries such as the random module for generating a random key and the ord() and chr() functions to convert characters to their ASCII values.
- Use Python's text encoding/decoding capabilities to ensure compatibility with different character sets.

Testing:

- Develop a testing plan to ensure the system's accuracy and security.
- Test the system using different input message and key combinations.
- Verify the encrypted messages can be decrypted correctly using the same key.
- Test the system's security by attempting to decrypt the message without the correct key.
- Utilize Python's built-in unit testing framework or other third-party testing libraries to automate testing and ensure consistent results.
- Analyze the results and make any necessary adjustments to the implementation to improve the system's accuracy and security.

Pseudo Code:

1. Import the random module.

2. Define a function named generate_key that takes a message string as input.

3. Inside the generate_key function, generate a random key of the same length as the message by iterating over the message and using the chr and randint functions from the random module.

4. Return the generated key.

5. Define a function named encrypt that takes a message string and a key string as input.

6. Inside the encrypt function, XOR each character in the message with the corresponding character in the key by iterating over the message and using the chr and ord functions.

7. Return the encrypted message.

8. Define a function named decrypt that takes an encrypted message string and a key string as input.

9. Inside the decrypt function, XOR each character in the encrypted message with the corresponding character in the key by iterating over the encrypted message and using the chr and ord functions.

10. Return the decrypted message.

11. Define a function named i that prints a menu of options for the user to choose from.

12. Print a logo for the program.

13. Get a message from the user.

14. Set a variable x to 0.

15. Start an infinite loop.

16. If x is not equal to 1, generate a key for the message using the generate_key function.

17. Encrypt the message using the encrypt function and the key.

18. Decrypt the encrypted message using the decrypt function and the key.

19. Print the menu of options using the i function.

20. Get a choice from the user.

21. If the choice is 1, get a new message from the user and set x to 0.

22. If the choice is 2, print a logo for encryption, print the key and encrypted message, and set x to 1.

23. If the choice is 3, print a logo for decryption, print the key and decrypted message, and set x to 0.

24. If the choice is 4, print "Exiting........" and break out of the infinite loop.

25. End the program.

CODE:

```
import random

def generate_key(message):
    # Generate a random key of the same length as the message
    key = ''
    for i in range(len(message)):
        key += chr(random.randint(33, 126))
    return key

def encrypt(message, key):
    # XOR each character in the message with the corresponding character in the key
    encrypted_message = ''

    for i in range(len(message)):
        encrypted_message += chr(ord(message[i]) ^ ord(key[i]))

    return encrypted_message

def decrypt(encrypted_message, key):
    # XOR each character in the encrypted message with the corresponding character in
the key
    decrypted_message = ''
    for i in range(len(encrypted_message)):
        decrypted_message += chr(ord(encrypted_message[i]) ^ ord(key[i]))
    return decrypted_message

def i():
    print('''\n-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit'''
          )

logo = """
  ___   ____     ___      ______  ____  ___ ___    ___      ____   ____  ___
 /   \ |    \   /  _]    |      ||    ||   |   |  /  _]    |    \ /    ||   \
|     ||  _  | /  [_     |      | |  | | _   _ | /  [_     |  o  )  o  ||    \
|  O  ||  |  ||    _]    |_|  |_| |  | |  \_/  ||    _]    |   _/|     ||  D  |
|     ||  |  ||   [_       |  |   |  | |   |   ||   [_     |  |  |  _  ||     |
|     ||  |  ||     |      |  |   |  | |   |   ||     |    |  |  |  |  ||     |
 \___/ |__|__||_____|      |__|  |____||___|___||_____|    |__|  |__|__||_____|

"""
print(logo)
```

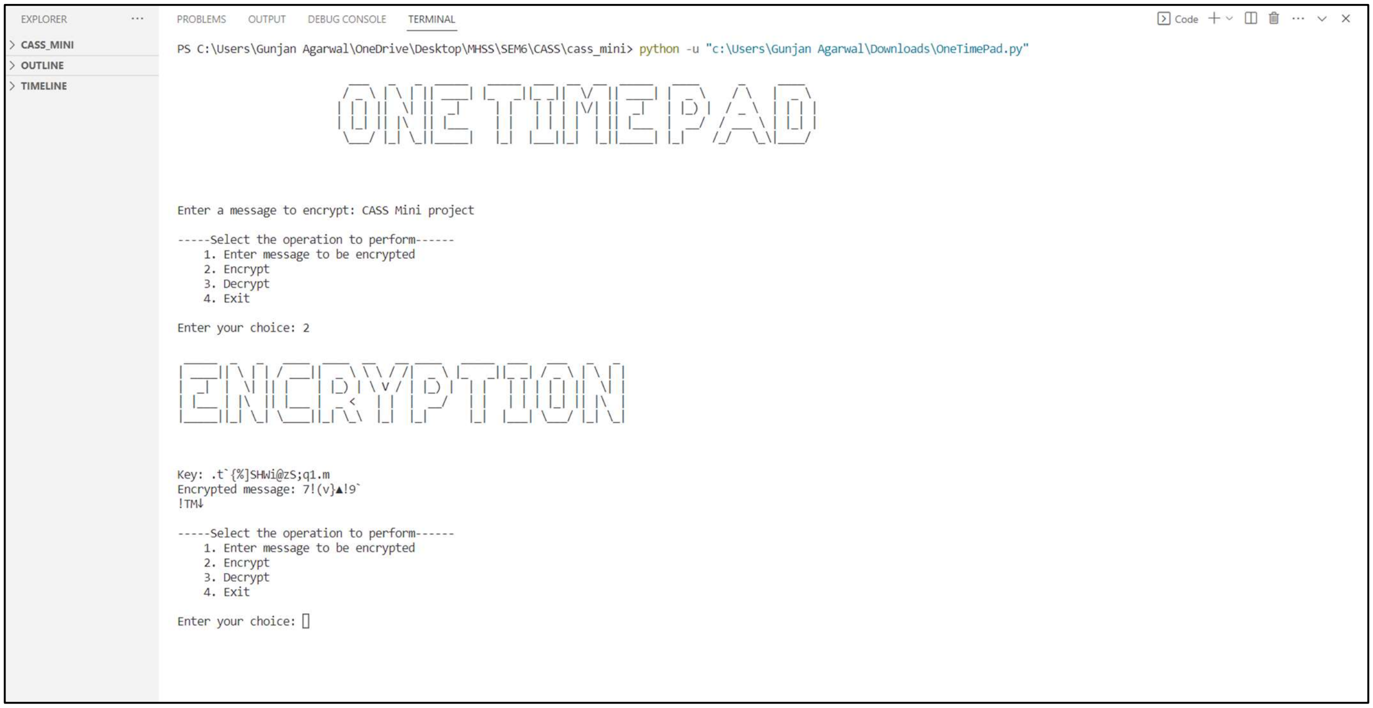
```
message = input("\nEnter a message to encrypt:")
x = 0
while True:
    if(x != 1):
        key = generate_key(message)

    encrypted_message = encrypt(message, key)
    decrypted_message = decrypt(encrypted_message, key)
    i()
    choice = int(input("\nEnter your choice: "))

    if choice == 1:
        message = input("Enter a message:")
        x = 0
    if(choice == 2):
        encrypt_logo = '''
 _____  _   _   ____  ____  __   __ ____   _____  ___   ___   _   _
| ____|| \ | | / ___||  _ \ \ \ / /|  _ \ |_   _||_ _| / _ \ | \ | |
|  _|  |  \| || |    | |_) | \ V / | |_) |  | |   | | | | | ||  \| |
| |___ | |\  || |___ |  _ <   | |  |  __/   | |   | | | |_| || |\  |
|_____||_| \_| \____||_| \_\  |_|  |_|      |_|  |___| \___/ |_| \_|

'''
        print(encrypt_logo)
        print("\nKey: " + key)
        # Print the encrypted message and key
        print("Encrypted message: " + encrypted_message)
        x = 1

    if(choice == 3):
        decrypt_logo = '''
 ____   _____   ____  ____  __   __ ____   _____  ___   ___   _   _
|  _ \ | ____| / ___||  _ \ \ \ / /|  _ \ |_   _||_ _| / _ \ | \ | |
| | | ||  _|  | |    | |_) | \ V / | |_) |  | |   | | | | | ||  \| |
| |_| || |___ | |___ |  _ <   | |  |  __/   | |   | | | |_| || |\  |
|____/ |_____| \____||_| \_\  |_|  |_|      |_|  |___| \___/ |_| \_|

        '''
        print(decrypt_logo)
        x = 0
        print("\nKey: " + key)
        # Print the decrypted message
        print("Decrypted message: " + decrypted_message)

    elif choice == 4:
        print("Exiting........")
        break
```

3. Technology Used

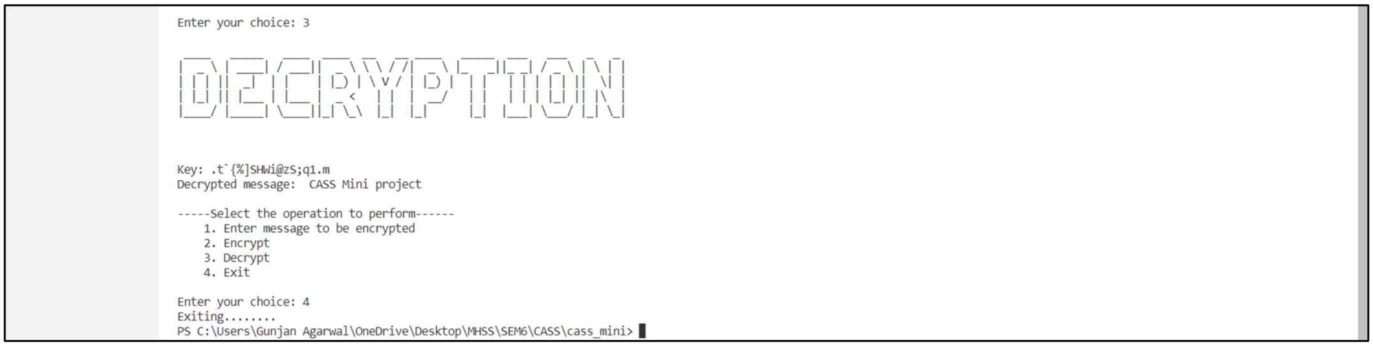
The following is the list of hardware and software requirements for the proposed system to be implemented.

- Hardware Requirements:
  - ❖ Any device with internet connectivity.
- Software Requirements:
  - ❖ PYTHON
  - ❖ VISUAL STUDIO CODE

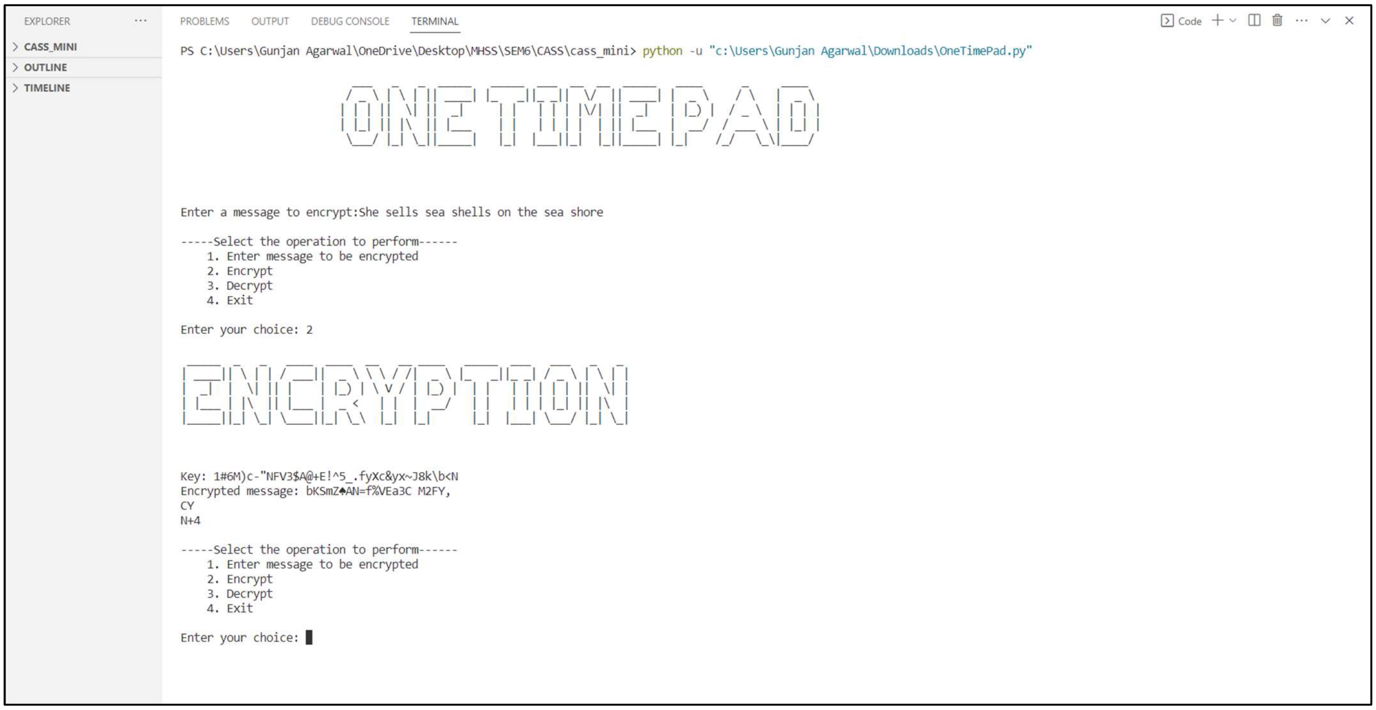
4. Outputs



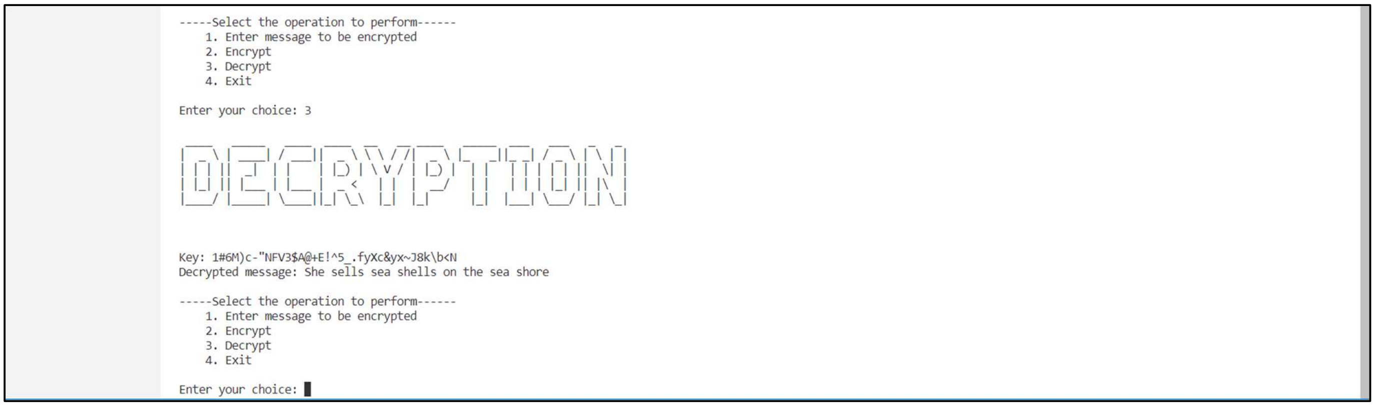
```
PS C:\Users\Gunjan Agarwal\OneDrive\Desktop\MHSS\SEM6\CASS\cass_mini> python -u "c:\Users\Gunjan Agarwal\Downloads\OneTimePad.py"

Enter a message to encrypt: CASS Mini project

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 2

Key: .t`{%]SHWi@zS;q1.m
Encrypted message: 7!(v}▲!9`
!TM↓

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 
```



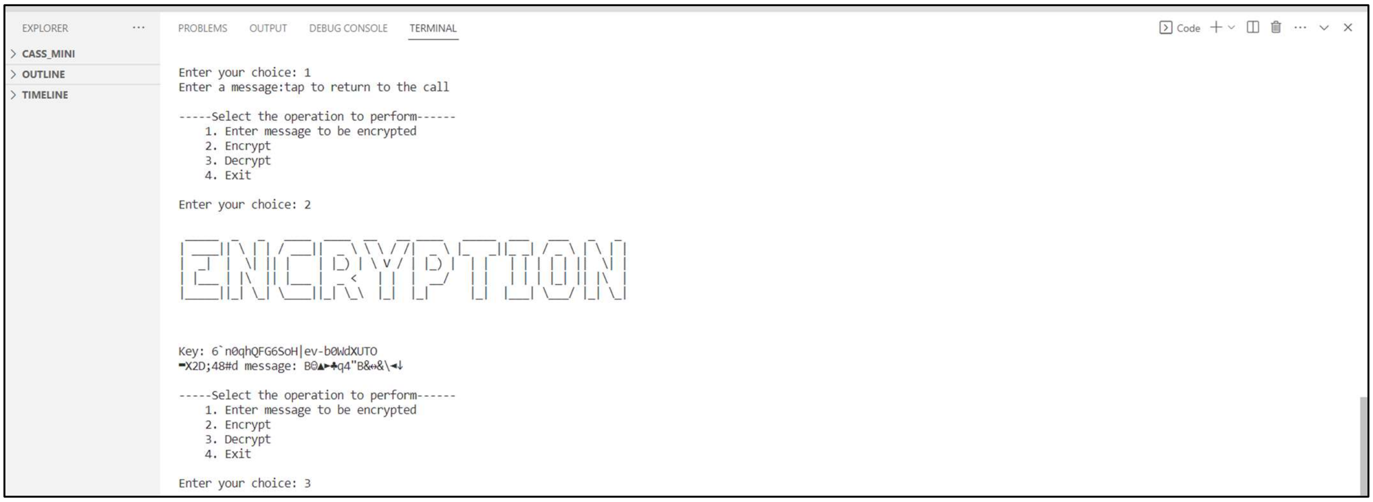
```
Enter your choice: 3

Key: .t`{%]SHWi@zS;q1.m
Decrypted message:  CASS Mini project

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 4
Exiting........
PS C:\Users\Gunjan Agarwal\OneDrive\Desktop\MHSS\SEM6\CASS\cass_mini> 
```



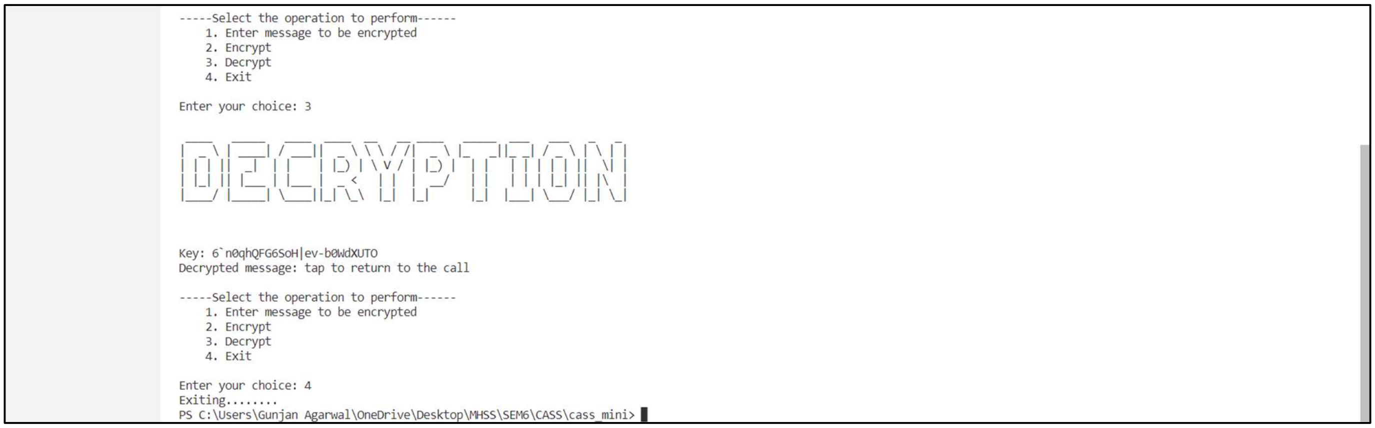
```
PS C:\Users\Gunjan Agarwal\OneDrive\Desktop\MHSS\SEM6\CASS\cass_mini> python -u "c:\Users\Gunjan Agarwal\Downloads\OneTimePad.py"

Enter a message to encrypt:She sells sea shells on the sea shore

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 2

Key: 1#6M)c-"NFV3$A@+E!^5_.fyXc&yx~J8k\b<N
Encrypted message: bKSmZ♠AN=f%VEa3C M2FY,
CY
N+4

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 
```



```
-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 3

Key: 1#6M)c-"NFV3$A@+E!^5_.fyXc&yx~J8k\b<N
Decrypted message: She sells sea shells on the sea shore

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 
```



```
Enter your choice: 1
Enter a message:tap to return to the call

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 2

 _____ _   _  ____ ______   ______ _____ ___ ___  _   _
| ____| \ | |/ ___|  _ \ \ / /  _ \_   _|_ _/ _ \| \ | |
|  _| |  \| | |   | |_) \ V /| |_) || |  | | | | |  \| |
| |___| |\  | |___|  _ < | | |  __/ | |  | | |_| | |\  |
|_____|_| \_|\____|_| \_\|_| |_|    |_| |___\___/|_| \_|



Key: 6`n0qhQFG6SoH|ev-b0WdXUTO
━X2D;48#d message: B⊖▲►♣q4"B&↔&\◄↓

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 3
```



```
-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 3

 ____  _____ ____ ______   ______ _____ ___ ___  _   _
|  _ \| ____/ ___|  _ \ \ / /  _ \_   _|_ _/ _ \| \ | |
| | | |  _|| |   | |_) \ V /| |_) || |  | | | | |  \| |
| |_| | |__| |___|  _ < | | |  __/ | |  | | |_| | |\  |
|____/|_____\____|_| \_\|_| |_|    |_| |___\___/|_| \_|



Key: 6`n0qhQFG6SoH|ev-b0WdXUTO
Decrypted message: tap to return to the call

-----Select the operation to perform------
    1. Enter message to be encrypted
    2. Encrypt
    3. Decrypt
    4. Exit

Enter your choice: 4
Exiting........
PS C:\Users\Gunjan Agarwal\OneDrive\Desktop\MHSS\SEM6\CASS\cass_mini> █
```

5. Conclusion

In this project, we used Python to create the One-Time Pad encryption scheme, which offers perfect secrecy and is a strong method to protect sensitive data. We gave users an easy-to-use interface that uses OTP to encrypt and decrypt messages, allowing them to safeguard their data from unauthorised access.

It is impossible to exaggerate the value of encryption, especially in the linked world of today where hacking and cyberattacks are a constant threat. OTP encryption ensures that the encryption key is random and only used once, which offers a practical method for safeguarding sensitive data. Without the key, it is practically impossible for an attacker to decipher the message. We have shown the effectiveness of encryption and underlined the value of utilising robust encryption methods to protect sensitive data by applying OTP encryption in this project. As a result, the project is a helpful source for individuals wanting to learn more about encryption and how crucial it is for protecting sensitive data.

The One-Time Pad encryption technique and its potential uses in protecting sensitive data are overall demonstrated in practise through this research. We have given consumers a practical and efficient method to safeguard their data from unauthorised access by implementing OTP encryption in Python.

REFERENCES

1. V. K. Bhargava and D. K. Gupta, "A Survey on Cryptography Techniques," 2015 International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2015, pp. 837-840, doi: 10.1109/INDIACom.2015.7100561.

2. C. E. Shannon, "Communication theory of secrecy systems," Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

3. K. D. Patel, "A Survey on Cryptography and Its Applications," 2015 International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, 2015, pp. 160-163, doi: 10.1109/CICT.2015.52.

4. M. T. Goodrich and R. Tamassia, "Introduction to Cryptography," in Algorithm Design and Applications, Hoboken, NJ: John Wiley & Sons, Inc., 2015, pp. 347-387, doi: 10.1002/9781118335916.ch13.