A white paper with blue text

Description automatically generated

A close up of a document

Description automatically generated

A white paper with black text

Description automatically generated

Contents

[Abstract: 2](#_Toc142610694)

[Objectives: 2](#_Toc142610695)

[Step-by-step explanation of how the Secure Messaging Application works: 7](#_Toc142610696)

[secure\_messenger.py: 8](#_Toc142610697)

[key\_generator.py: 8](#_Toc142610698)

[chat\_room.py: 9](#_Toc142610699)

[cui.py: 10](#_Toc142610700)

[Gui.py: 12](#_Toc142610701)

[Main.py: 14](#_Toc142610702)

[Testing: 15](#_Toc142610703)

[CLI 15](#_Toc142610704)

[GUI 16](#_Toc142610708)

[Conclusion: 18](#_Toc142610716)

# Table of Figures:

[Figure 1 7](#_Toc142610799)

[Figure 2 8](#_Toc142610800)

[Figure 3 9](#_Toc142610801)

[Figure 4 10](#_Toc142610802)

[Figure 5 10](#_Toc142610803)

[Figure 6 12](#_Toc142610804)

[Figure 7 12](#_Toc142610805)

[Figure 8 13](#_Toc142610806)

[Figure 9 14](#_Toc142610807)

[Figure 10 15](#_Toc142610808)

[Figure 11 15](#_Toc142610809)

[Figure 12 16](#_Toc142610810)

[Figure 13 16](#_Toc142610811)

[Figure 14 16](#_Toc142610812)

[Figure 15 16](#_Toc142610813)

[Figure 16 17](#_Toc142610814)

[Figure 17 17](#_Toc142610815)

[Figure 18 17](#_Toc142610816)

[Figure 19 17](#_Toc142610817)

# Step-by-step explanation of how the Secure Messaging Application works:

1. **When the application starts (`main.py`):**
   1. sers are presented with a menu to choose between the Command-line Interface (CUI) or the Graphical User Interface (GUI) for interaction.
2. **If the user chooses the Command-line Interface (CUI):**
   1. The `run\_cui()` function from `cui.py` is called.
   2. An encryption key is provided to the `ChatRoom` object to handle message encryption and decryption.
   3. The CUI displays a menu with options to register a user, send a message, retrieve messages, or exit the application.
3. **Registering a user (CUI):**
   1. The user is prompted to enter a username they want to register.
   2. The `ChatRoom` object checks if the username is available and adds the user to the list of registered users if it's unique.
4. **Sending a message (CUI):**
   1. The user is prompted to enter the sender, receiver, and message they want to send.
   2. The `ChatRoom` object verifies if both the sender and receiver exist as registered users.
   3. The message is encrypted using the `SecureMessenger` class and stored in the receiver's message list.
5. **Retrieving messages (CUI):**
   1. The user is prompted to enter their user ID to retrieve messages.
   2. The `ChatRoom` object checks if the user ID exists and retrieves the encrypted messages from the user's message list.
   3. The messages are decrypted using the `SecureMessenger` class and displayed on the terminal.
6. **If the user chooses the Graphical User Interface (GUI):**
   1. An instance of the `ChatGUI` class from `gui.py` is created.
   2. The GUI window is set up with labels, entry fields, and buttons for user interaction.
7. **Sending a message (GUI):**
   1. Users can enter the sender, receiver, and message in the GUI entry fields.
   2. Upon clicking the "Send" button, the `send\_message()` method of the `ChatGUI` class is called.
   3. The `ChatRoom` object handles the message encryption and storage, and a message box displays the status of the sent message.
8. **Registering a user (GUI):**
   1. Users can enter a username in the GUI entry field.
   2. Upon clicking the "Register User" button, the `register\_user()` method of the `ChatGUI` class is called.
   3. The `ChatRoom` object handles the user registration, and a message box displays the status of the registration.
9. **Retrieving messages (GUI):**
   1. Users can enter their user ID in the GUI entry field.
   2. Upon clicking the "View Messages" button, the `view\_messages()` method of the `ChatGUI` class is called.
   3. The `ChatRoom` object handles message retrieval, decryption, and display, and a message box shows the decrypted messages.
10. **Exiting the application (CUI and GUI):**
    1. Users can choose to exit the application from either the CUI or the GUI interfaces.
    2. In the CUI, choosing "4" exits the application, and in the GUI, clicking the close button on the window does the same.

# secure\_messenger.py:

**A screen shot of a computer program

Description automatically generated**

Figure 1

**Explanation:**

* This module defines the SecureMessenger class, responsible for message encryption and decryption.
* It imports Fernet from the cryptography.fernet library, which is used for encryption and decryption.
* SecureMessenger class:
  + The \_\_init\_\_ method initializes the SecureMessenger object with an encryption key provided as an argument. It stores the key and creates a Fernet cipher suite using that key for later encryption and decryption operations.
  + encrypt\_message method takes a plaintext message as input, converts it to bytes, encrypts it using the Fernet cipher suite, and returns the encrypted message.
  + decrypt\_message method takes an encrypted message (in bytes) as input, decrypts it using the Fernet cipher suite, and returns the decrypted plaintext message as a string.

# key\_generator.py:

**A screen shot of a computer code

Description automatically generated**

Figure 2

**Explanation:**

* This module defines the generate\_key function, which generates an encryption key and saves it to a file.
* It imports Fernet from the cryptography.fernet library to create a key.
* generate\_key function:
  + The function takes a file path as an argument to specify where to save the generated key.
  + It generates a random 32-byte (256-bit) encryption key using Fernet.generate\_key().
  + The generated key is then written to the specified file in binary mode ('wb').

# chat\_room.py:

A screen shot of a computer program

Description automatically generated

Figure 3

**Explanation:**

* This module defines the ChatRoom class, responsible for managing user registration, sending messages, and retrieving messages securely.
* ChatRoom class:
  + The \_\_init\_\_ method initializes the ChatRoom object with an encryption key provided as an argument. It creates a SecureMessenger object to handle encryption and decryption and initializes an empty dictionary self.users to store registered users and their messages.
  + register\_user method allows registering new users. It takes a username as input and adds the user to the self.users dictionary as a key with an empty list to store their messages.
  + send\_message method allows sending messages from one user to another. It takes the sender, receiver, and message as inputs, encrypts the message using the SecureMessenger, and stores it in the receiver's message list.
  + retrieve\_messages method retrieves messages for a specific user. It takes the user ID as input, checks if the user exists in the self.users dictionary, retrieves the messages, decrypts them using the SecureMessenger, and returns the decrypted messages as a list.

# cui.py:

**A screen shot of a computer program

Description automatically generated**

Figure 4

**A computer screen with text on it

Description automatically generated**

Figure 5

**Explanation:**

* This module provides the Command-line Interface (CUI) for user interaction with the application.
* It imports the ChatRoom class from chat\_room.py to handle user actions and message operations.
* run\_cui function:
  + It starts by setting up the encryption key (encryption\_key), which is hardcoded for demonstration purposes (but should be handled securely in a real application).
  + It creates a ChatRoom object (chat\_room) with the provided encryption key.
  + The function enters an infinite loop (while True) to continuously display a menu and take user input.
  + Depending on the user's choice, it calls the corresponding methods from the ChatRoom object and displays the results through the terminal.

# Gui.py:

**A screen shot of a computer program

Description automatically generated**

Figure 6

**A screen shot of a computer program

Description automatically generated**

Figure 7

**A computer screen shot of text

Description automatically generated**

Figure 8

**Explanation:**

* This module creates a Graphical User Interface (GUI) using the tkinter library for user interaction with the application.
* It imports tkinter modules, messagebox, and the ChatRoom class from chat\_room.py to handle user actions and message operations.
* ChatGUI class:
  + The constructor initializes the ChatGUI object with an encryption key and a ChatRoom object.
  + It sets up a window with labels, entry fields, and buttons for user input.
  + The send\_message, register\_user, and view\_messages methods handle their respective actions based on user input.
  + The run\_gui method starts the GUI main loop using self.window.mainloop().

# Main.py:

**A screen shot of a computer program

Description automatically generated**

Figure 9

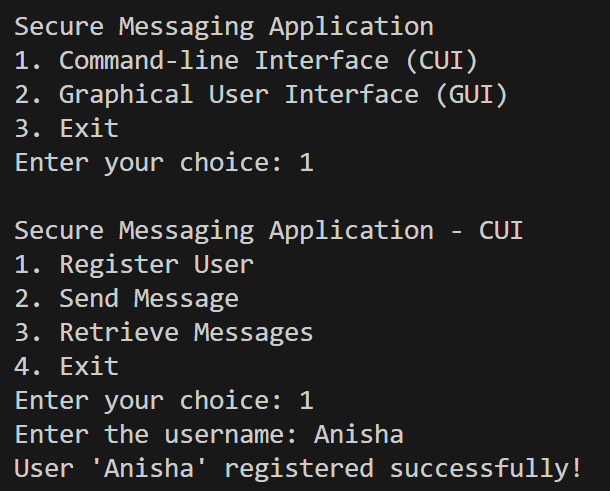
**Explanation:**

* This is the main script to run the Secure Messaging Application.
* It imports cui.py and gui.py modules to provide the user with a choice between CUI or GUI interfaces.
* main function:
  + It starts by presenting users with a menu to choose between CUI or GUI interfaces.
  + Depending on the user's choice, it calls the run\_cui() function from cui.py or creates a ChatGUI object and runs the GUI.

Overall, the Secure Messaging Application provides users with two ways to interact with the system: Command-line Interface (CUI) and Graphical User Interface (GUI). The application allows user registration, sending encrypted messages, and retrieving decrypted messages in a secure manner using encryption keys

Testing:

CLI



Figure

A screenshot of a computer screen

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

# GUI

A screen shot of a computer

Description automatically generated

Figure

A screenshot of a computer message

Description automatically generated

Figure

A close up of a word

Description automatically generated

Figure

A black and white text

Description automatically generated

Figure

A screenshot of a message

Description automatically generated

Figure

A screenshot of a computer screen

Description automatically generated

Figure

A screenshot of a message

Description automatically generated

Figure

Conclusion:  
  
In conclusion, the Secure Messaging Application provides a reliable and secure platform for users to communicate privately through encrypted messages. With its two interfaces, the Command-line Interface (CUI) and the Graphical User Interface (GUI), the application caters to diverse user preferences. By integrating cryptographic techniques from the cryptography.fernet library, the application ensures message encryption during transmission and decryption upon retrieval, guaranteeing the confidentiality of user communications.

The seamless collaboration of key components makes the application efficient. The secure\_messenger.py module, utilizing the SecureMessenger class, handles message encryption and decryption to safeguard exchanged information. On the other hand, the chat\_room.py module acts as the backbone, efficiently managing user registration, message storage, and retrieval while maintaining message integrity.

Moreover, the application's versatility lies in the cui.py module, providing a straightforward and text-based interface, and the gui.py module, offering a visually appealing and user-friendly graphical interface using tkinter. This adaptability enhances the application's accessibility for a wide range of users.

However, it is essential to recognize that this implementation serves educational purposes and may require additional considerations, such as user authentication, secure key management, and protection against potential security vulnerabilities, when used in real-world applications.