A Information Security Mini Project Report
on

"SECURE COMMUNICATION TOOL"

Submitted to the
Savitribai Phule Pune University

In partial fulfillment of

"Artificial Intelligence and Machine Learning"
By

Manasi Rathod [33559]

Under the guidance of

Subject In charge: Prof. Sapana Bhirud

Department Of Artificial Intelligence and Machine Learning
PES's Modern College of Engineering,
Shivaji Nagar, Pune-411005,
Maharashtra, India 2024-2025

CERTIFICATE

This is to certify that the project based learning-II report entitled **" Secure Communication Tool "** being submitted by **Manasi Rathod** is a record of bonafide work carried out by him/her under the supervision and guidance of **Prof. Sapana Bhirud Ma'am** in partial fulfillment of the requirement for **TE (**Artificial Intelligence and Machine Learning**) – 2019 course** of Savitribai Phule Pune University, Pune in the academic year 2024-2025

Date:07/10/24

Place: Pune

Prof.Mrs.SapanaBhirud
Guide of Information Security

Dr.Mrs.S.D.Deshpande
Head of the Department

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude towards all those who helped me in accomplishing this Information Security project work. I would like to thank P.E.S.'s Modern College of Engineering for giving me this opportunity to look in some concept from my curriculum subject. I would like to thank my guide **Prof. Sapana Bhirud** for her valuable comments and timely support. I would like to show my greatest appreciation to her.

I would also like to thank all other faculty members for supporting me directly or indirectly. The guidance and support received from all the Professors who contributed are vital for the success of the work. I am grateful for their constant support and help.

I would also like to thank my friends for their support and encouragement.

Manasi Rathod [33559]

Introduction

As communication moves increasingly into the digital realm, ensuring the privacy and security of transmitted information has become a top priority. Encryption is one of the key technologies used to protect sensitive data, converting readable messages into an unreadable format that can only be decrypted by someone who possesses the appropriate cryptographic key.

The **Secret Communication Tool** is designed to provide a basic but effective solution for encrypting and decrypting text messages. It uses symmetric encryption, meaning the same key is used for both encrypting and decrypting a message. The tool's main purpose is to provide a simple, user-friendly interface for users to protect their messages from being read by unauthorized individuals. This makes the application highly useful for learning and experimenting with the fundamentals of cryptography in a real-world setting.

This project employs **Fernet encryption**, a secure and widely trusted encryption method provided by the **cryptography** library in Python. The tool is implemented as a command-line application, making it accessible for anyone familiar with basic Python and programming concepts. Additionally, it can serve as the foundation for more advanced security projects involving encrypted communications.

Abstract

The **“ Secret Communication Tool ”** is a Python-based application developed to enable secure communication through encryption and decryption of messages. By leveraging the powerful cryptography library, specifically the Fernet module, the tool implements symmetric key encryption to ensure the confidentiality of messages.

The program allows users to generate cryptographic keys, encrypt plain text messages, and decrypt encrypted messages back to their original form. The simplicity of its design and implementation makes it suitable for small-scale secure communication purposes, serving as a practical introduction to cryptographic concepts.

This report provides a comprehensive overview of the tool's objectives, core functionality, and future potential improvements, illustrating its real-world application in safeguarding private communications.

Objective

The primary objective of the **" Secret Communication Tool "** is to provide users with a practical, easy-to-use application that allows them to:

- Encrypt plain text messages: Transform readable information into a secure, encrypted format that cannot be interpreted without the proper key.
- Decrypt encrypted messages: Convert an encrypted message back into its original readable form using the same cryptographic key.
- Securely manage cryptographic keys: Automatically generate and store cryptographic keys in a secure format for reuse in future sessions.
- Enhance understanding of cryptographic principles: Demonstrate the basic concepts of symmetric encryption, key management, and the cryptography process through an accessible, real-world example.

The tool is designed to be simple enough for educational purposes while remaining practical for use in small-scale personal communications. Through its functionality, it aims to highlight the importance of encryption in maintaining the privacy and integrity of sensitive data.

Basic Structure of Secure Communication

The **Secret Communication Tool** is a Python program that provides encryption and decryption of text messages using the cryptography library's Fernet symmetric encryption. The program is structured as follows:

1. **Key Management :**
   - The load_key() function checks if a key file (secret.key) exists. If so, it loads the key; if not, it generates a new one and saves it to the file. This key is crucial for both encryption and decryption.
2. **Encryption :**
   - The encrypt_message() method takes a plain text message, converts it to bytes, and encrypts it using the Fernet key, returning the encrypted message (ciphertext).
3. **Decryption :**
   - The decrypt_message() method takes an encrypted message, decrypts it using the key, and returns the original plain text.
4. **User Interface :**
   - The main() function provides a menu with options to encrypt or decrypt a message or exit the program. Based on the user's input, it performs the respective action and handles any errors during decryption.

In essence, the tool ensures secure communication by transforming readable text into unreadable encrypted data and vice versa, using the same key for both processes.

Encryption & Decryption CODE

```python
import os
from cryptography.fernet import Fernet

class SecretCommunicationTool:
    def __init__(self, key_file='secret.key'):
        self.key_file = key_file
        self.key = self.load_key()
        self.fernet = Fernet(self.key)

    def load_key(self):
        if os.path.exists(self.key_file):
            with open(self.key_file, 'rb') as file:
                return file.read()
        else:
            key = Fernet.generate_key()
            with open(self.key_file, 'wb') as file:
                file.write(key)
            return key

    def encrypt_message(self, message):
        return self.fernet.encrypt(message.encode())

    def decrypt_message(self, encrypted_message):
        return self.fernet.decrypt(encrypted_message).decode()
```

```python
def main():
    tool = SecretCommunicationTool()

    while True:
        print("\nSecret Communication Tool")
        print("1. Encrypt Message")
        print("2. Decrypt Message")
        print("3. Exit")

        choice = input("Choose an option: ")

        if choice == '1':
            message = input("Enter the message to encrypt: ")
            encrypted = tool.encrypt_message(message)
            print(f"Encrypted Message: {encrypted.decode()}")
        elif choice == '2':
            encrypted_message = input("Enter the message to decrypt: ")
            try:
                decrypted = tool.decrypt_message(encrypted_message.encode())
                print(f"Decrypted Message: {decrypted}")
            except Exception as e:
                print("Error decrypting message:", str(e))
        elif choice == '3':
            break
        else:
            print("Invalid choice.")

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

HTML CODE

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Secret Communication Tool</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        textarea { width: 100%; height: 100px; }
        button { margin-top: 10px; }
    </style>
</head>
<body>
    <h1>Secret Communication Tool</h1>

    <h2>Encrypt Message</h2>
    <textarea id="plainText" placeholder="Enter your message here..."></textarea>
    <button onclick="encryptMessage()">Encrypt</button>
    <h3>Encrypted Message:</h3>
    <textarea id="encryptedText" readonly></textarea>

    <h2>Decrypt Message</h2>
    <textarea id="cipherText" placeholder="Enter encrypted message here..."></textarea>
    <button onclick="decryptMessage()">Decrypt</button>
    <h3>Decrypted Message:</h3>
    <textarea id="decryptedText" readonly></textarea>
```

```html
    <script>
        let key;

        async function generateKey() {
            key = await window.crypto.subtle.generateKey(
                {
                    name: "AES-GCM",
                    length: 256
                },
                true,
                ["encrypt", "decrypt"]
            );
        }

        async function encryptMessage() {
            const plaintext = document.getElementById("plainText").value;
            const encoder = new TextEncoder();
            const iv = window.crypto.getRandomValues(new Uint8Array(12)); // Initialization vector
            const encrypted = await window.crypto.subtle.encrypt(
                {
                    name: "AES-GCM",
                    iv: iv
                },
                key,
                encoder.encode(plaintext)
            );
```

```
        const encryptedBuffer = new Uint8Array(encrypted);
        const combinedBuffer = new Uint8Array(iv.length + encryptedBuffer.length);
        combinedBuffer.set(iv);
        combinedBuffer.set(encryptedBuffer, iv.length);
        document.getElementById("encryptedText").value = btoa(String.fromCharCode(...combinedBuffer));
    }

    async function decryptMessage() {
        const encryptedData = atob(
        const encryptedArray = new Uint8Array([...encryptedData].map(char => char.charCodeAt(0)));
        const iv = encryptedArray.slice(0, 12); // Extract IV
        const ciphertext = encryptedArray.slice(12); // Extract the actual ciphertext

        const decrypted = await window.crypto.subtle.decrypt(
            {
                name: "AES-GCM",
                iv: iv
            },
            key,
            ciphertext
        );
```

```
var Uint8Array: Uint8ArrayConstructor
new (elements: Iterable<number>) => Uint8Array (+4 overloads)
```

```
        const decoder = new TextDecoder();
        document.getElementById("decryptedText").value = decoder.decode(decrypted
    }

    // Generate the encryption key when the page loads
    window.onload = generateKey;
</script>
</body>
</html>
```

Output

Encypting Mechanism :

Secret Communication Tool

Encrypt Message

```
Hello Everyone
Myself MAnasi Rathod
```

Encrypt

Encrypted Message:

```
i43EfMDHohPD5ZlrmG5l61DLz+X9Ho+362negbr4SewriYwZB6KUiST+eRWlpi/A2DgWgdX0j0y94dFf9Y6KWw==
```

Decrypting Mechanism :

Decrypt Message

```
i43EfMDHohPD5ZlrmG5l61DLz+X9Ho+362negbr4SewriYwZB6KUiST+eRWlpi/A2DgWgdX0j0y94dFf9Y6KWw==
```

Decrypt

Decrypted Message:

```
Hello Everyone
Myself MAnasi Rathod
```

Conclusion

The **Secret Communication Tool** is a basic yet effective implementation of cryptography using Python. It demonstrates key cryptographic concepts such as encryption, decryption, and secure key management in a manner that is both practical and easy to understand. The tool highlights the importance of secure communication and showcases how encryption can be used to protect sensitive information.

The project achieves its primary objectives of providing an intuitive platform for secure communication and promoting a better understanding of cryptography. While simple, the tool serves as a solid foundation for more complex encryption systems and can be enhanced with additional features in the future. Possible improvements include implementing password protection for keys, adding message integrity verification, and developing a graphical user interface (GUI).

In summary, the **Secret Communication Tool** fulfills its intended purpose as a basic, educational, and functional encryption tool, offering a hands-on approach to understanding the critical role of encryption in modern digital communication .