CIS 547 – Network Security & Data Assurance

Project Title: Dynamic OTP-Based Secure Message Locker

Professor Ashok Patel

Group Number: 10

Team Members:

1. Prerak Panwar (02081587)

2. Khushi Hareshkumar Dubal (02134550)

3. Ravi Sai Alekhya (02101882)

Project Summary

1. Project Overview

In an era dominated by digital communication, the importance of secure messaging cannot be overstated. With organizations and individuals relying on internet-based platforms for transferring sensitive information, it is crucial to ensure confidentiality, integrity, and authenticity in communication. This project, Dynamic OTP-Based Secure Message Locker, addresses the growing need for secure messaging by combining AES (Advanced Encryption Standard) and TOTP (Time-based One-Time Password) authentication in an intuitive and robust application.

Key Concepts in Security:

- **AES Encryption:** This widely accepted standard provides unparalleled security for encrypting messages, making them indecipherable without the appropriate decryption key.
- **TOTP Authentication:** Adding a second layer of security, TOTP ensures that only users with a dynamically generated time-sensitive password can access encrypted messages. This OTP expires in a limited time window (e.g., 30 seconds), reducing vulnerabilities to interception.

In addition to encryption and authentication, the project incorporates rate limiting and lockout mechanisms to mitigate brute-force attacks, ensuring even further security robustness. For a seamless user experience, it includes a graphical user interface (GUI) developed using Tkinter.

2. Objectives

To achieve its goals, the project is designed with the following objectives in mind:

Security-Oriented Objectives:

- Message Confidentiality: Safeguard messages by using AES encryption with a Cipher Block Chaining (CBC) mode to ensure confidentiality and resistance to brute-force attacks.
- Dynamic Authentication: Use TOTP to authenticate users dynamically, ensuring that OTPs are unique for every session.
- Security Auditing: Maintain detailed logs of security-related events, such as failed OTP attempts, lockouts, and successful authentications.

Usability-Oriented Objectives:

- User-Friendly Design: Simplify the encryption/decryption process with an easy-to-use GUI.
- Resilience Against Attacks: Incorporate rate-limiting mechanisms to block excessive failed attempts, along with a temporary lockout feature.

3. System Design and Methodology

The system design reflects a combination of cryptographic principles, user-centric design, and secure coding practices. Below is a breakdown of the methodology:

1-QR Code Generation for TOTP:

- The pyotp library generates a unique Base32-encoded secret key, which is embedded into a QR code using the qrcode library.
- Users scan this QR code with the Google Authenticator app, establishing their unique TOTP setup.

2-Message Encryption & Decryption:

- Encryption employs AES in CBC mode, requiring a 256-bit key. This key is securely stored in an environment variable, ensuring it is not exposed in the codebase.
- Decryption uses the same key to transform ciphertext back into plaintext, provided the OTP entered by the user is valid.

3-Authentication with TOTP:

- The user enters the OTP displayed in Google Authenticator into the application. pyotp verifies the OTP for validity within the defined expiration window (e.g., 30 seconds).

4-Lockout and Rate Limiting:

- To prevent brute-force OTP guessing, the application tracks failed OTP attempts.
- After three invalid attempts, the user is locked out for 60 seconds, as seen in the screenshots. These events are logged for auditing purposes.

5-Logging:

- All key actions, such as OTP validation, encryption, decryption, and lockouts, are logged with timestamps and associated IP addresses.

4. Tools and Technologies Used

The project leverages the following tools and technologies to achieve its objectives:

- **Python:** Chosen for its versatility and extensive libraries for cryptography, GUI design, and QR code generation.
- **pyotp:** Simplifies TOTP generation and validation, ensuring compatibility with Google Authenticator.
- **cryptography:** Offers secure implementations of AES encryption, ensuring compliance with industry standards.
- **Tkinter:** Provides a user-friendly GUI framework for seamless user interactions.
- **qrcode:** Simplifies QR code generation for sharing TOTP secret keys securely.
- **ipinfo.io API:** Enhances logging capabilities by providing geographical and ISP information for logged IPs.

Why These Tools?

- Python is widely used for cryptography and GUI applications due to its simplicity and powerful libraries.
- Tkinter ensures the application is platform-independent, working seamlessly on Windows, Mac, and Linux systems.

5. Code Explanation

Each script in the project has a specific role:

encrypt_config.py:

Encrypts the TOTP secret stored in a JSON file. This script ensures that sensitive configuration files are stored securely using AES encryption.

decrypt_config.py:

Decrypts the encrypted TOTP secret using an AES key stored in an environment variable. This ensures the application retrieves sensitive information without compromising security.

Generate_QR.py:

Generates a QR code that encodes the TOTP provisioning URI. This URI contains the secret key, account name, and issuer details. Users scan the QR code with Google Authenticator to initialize their TOTP configuration.

main.py:

Acts as the main interface for users. It includes:

- OTP Verification: Validates OTPs entered by the user.

- Encryption/Decryption: Processes user messages based on authentication results.
- Logging: Tracks successful/failed attempts and logs IP addresses.
- Lockout: Implements rate-limiting and lockout mechanisms for security.

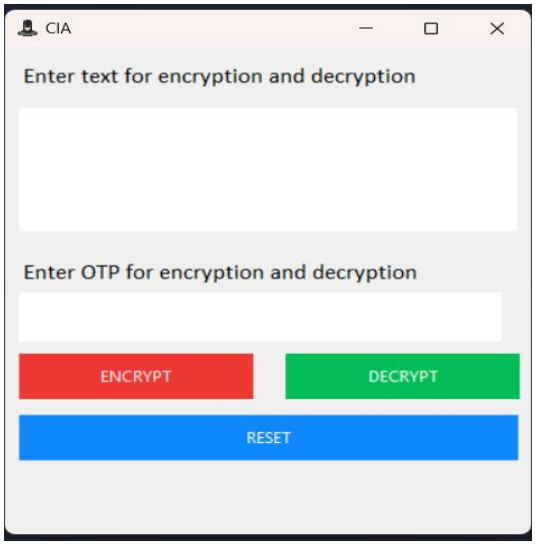
ipfinder.py:

Fetches details of IP addresses logged during user interactions, such as location and ISP information, for further analysis.

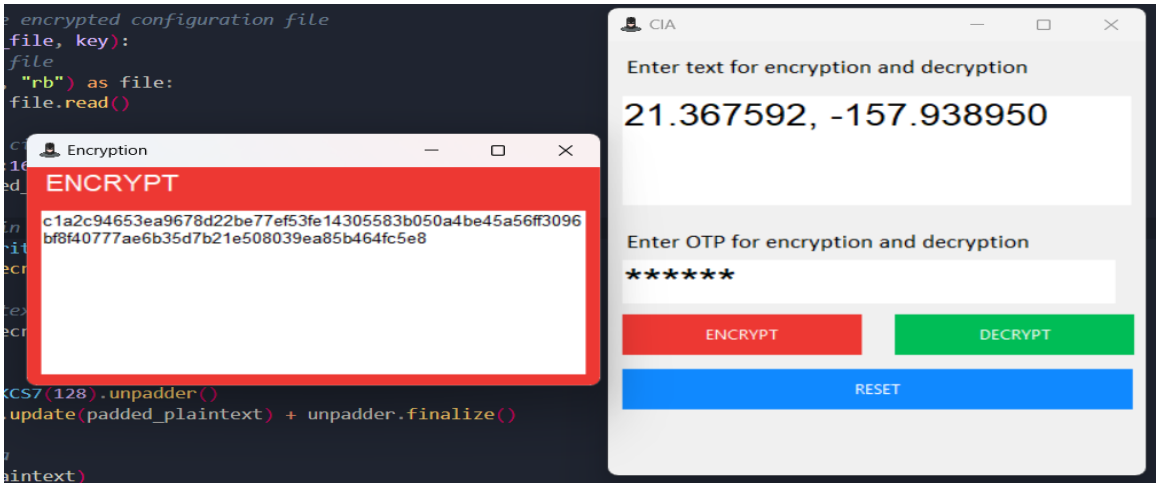
6. Screenshots and Results

Screenshot Descriptions:

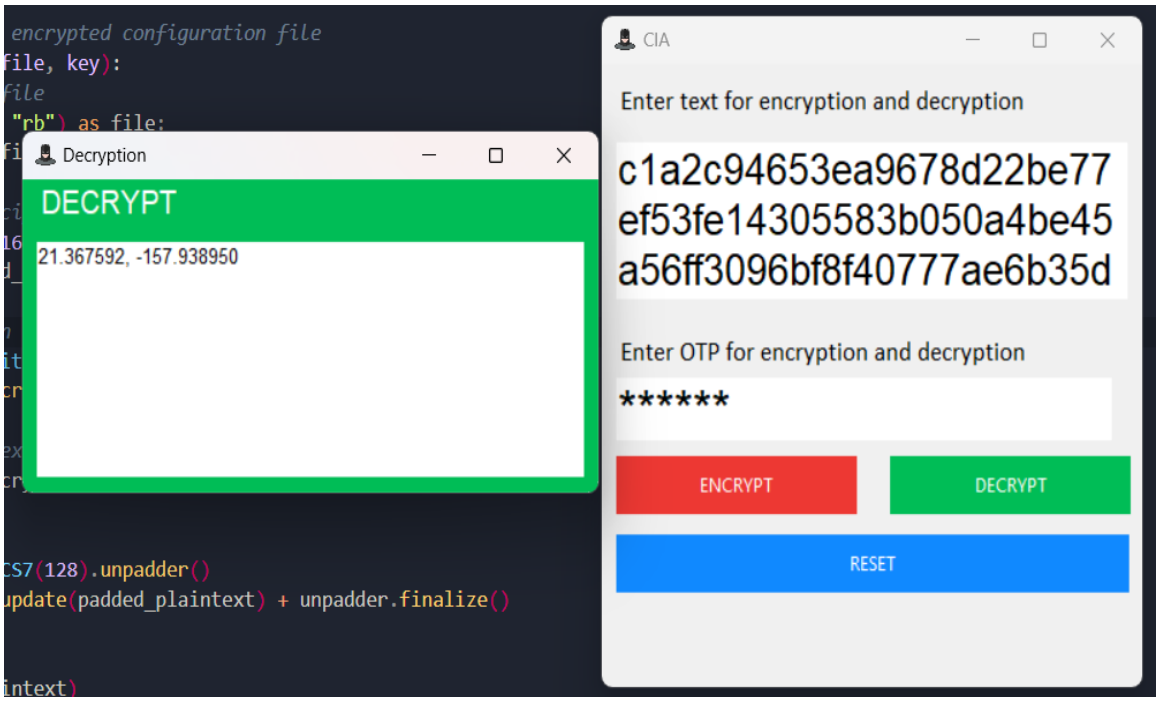
Main GUI: Users can enter text for encryption/decryption and OTP for validation.



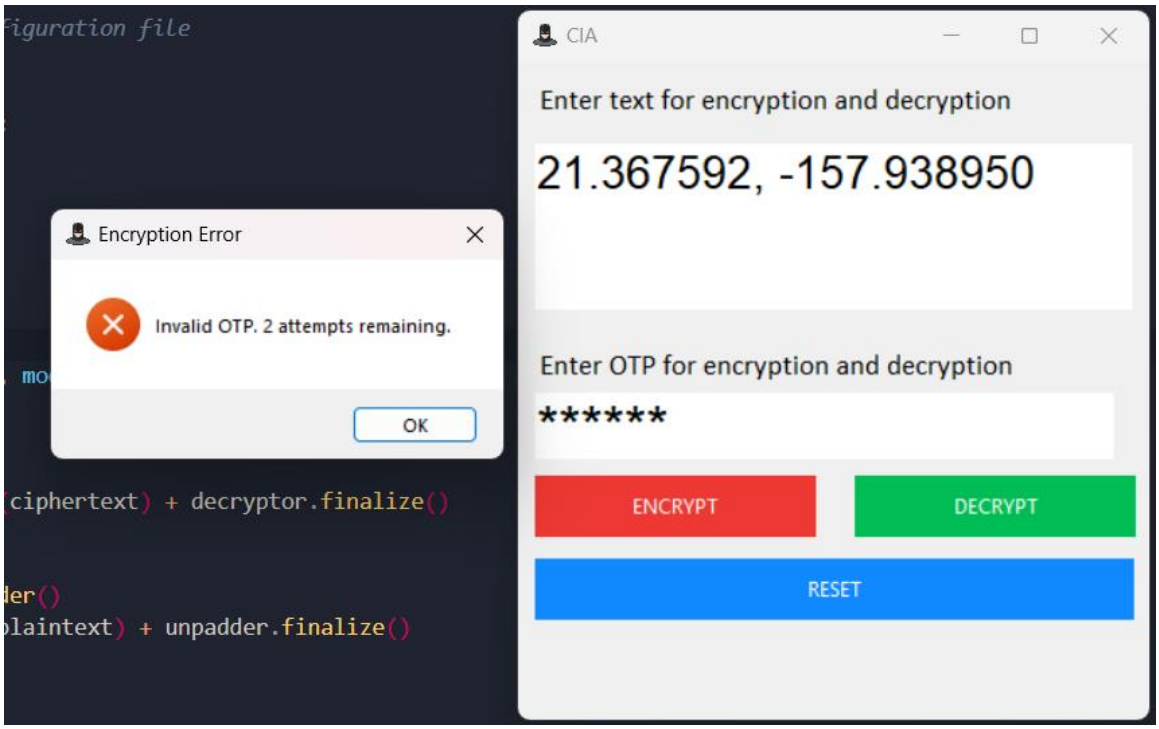
Encryption Success: Displays encrypted text after validating the OTP.



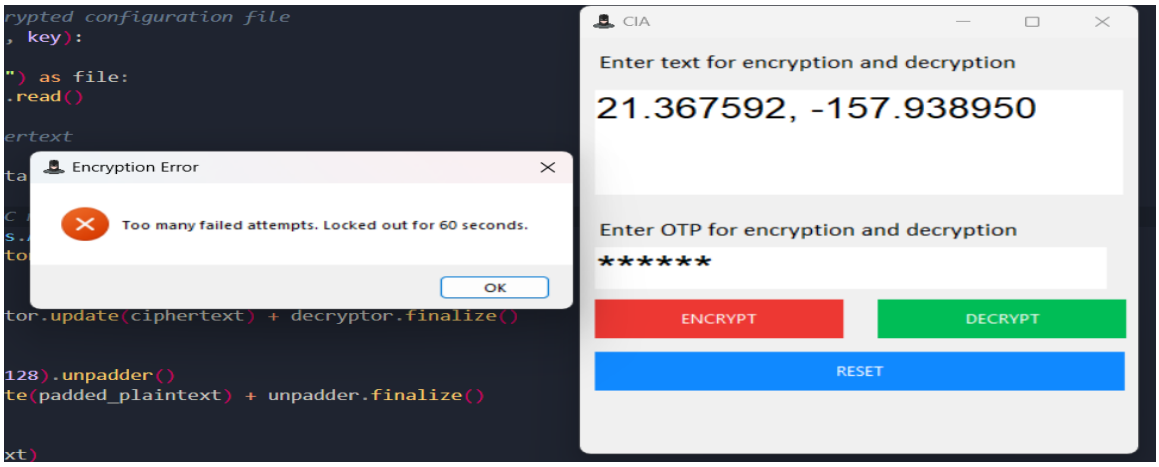
Decryption Success: Shows the original message after successful decryption.



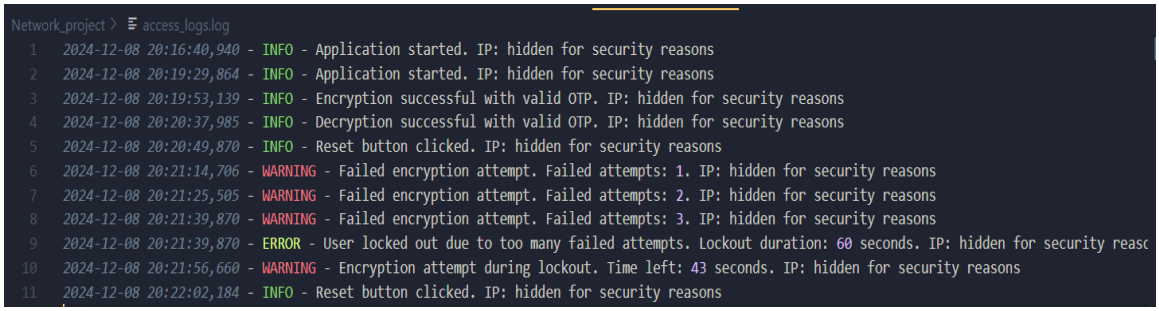
Failed OTP Attempts: Users receive warnings for invalid OTPs, with a countdown of remaining attempts.



Lockout Mechanism: After three failed OTP attempts, the user is locked out temporarily.



Logging: Logs are maintained with timestamps, actions, and IP addresses for monitoring.



8. Applications

This application has a wide range of potential applications, including:

- Business Communication: Providing a secure platform for transmitting sensitive business information.
- Education: Serving as a learning tool for students and developers interested in encryption and authentication techniques.
- Legal and Financial Sectors: Protecting sensitive client data and documents in legal and financial domains.

9. References and Citations

The project was developed using the following resources:

1. Python documentation: https://www.python.org/doc/
2. pyotp library: https://pyauth.github.io/pyotp/
3. cryptography library: https://cryptography.io/en/latest/
4. qrcode library: https://github.com/lincolnloop/python-qrcode
5. IP information API: https://ipinfo.io/