CADT IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល Institute of Digital Technology

Name: Yim Cheayong

Group: 02

IDTB100258

Individual's Project Report

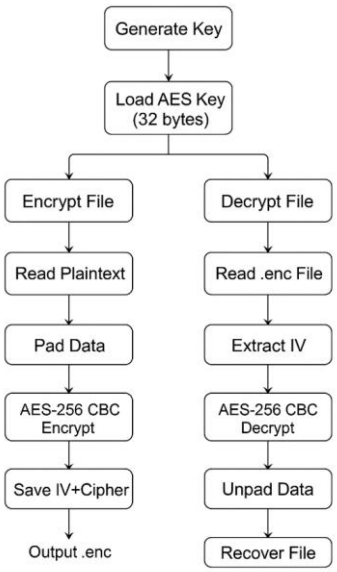
CrypLock: A Symmetric-Key File Protection Tool

I. Introduction

This project aims to create a simple and secure file-locking application called **CrypLock**, which allows users to encrypt and decrypt files using **AES-256**, one of the strongest symmetric encryption methods. The main goal is to protect sensitive data so that unauthorized people cannot access it. In today's digital environment, files can easily be copied, transferred, or leaked. Without proper encryption, private information becomes vulnerable. CrypLock solves this problem by providing a small, easy-to-use tool that locks files with a secret key. Only someone who has the correct key can decrypt and recover the original file. This project uses basic cryptographic concepts such as **symmetric key encryption**, **AES-256**, **CBC mode**, and **PKCS7 padding**. The system is designed to be simple enough for beginners, while still demonstrating real-world security practices.

II. System Design / Architecture

1) User generates a 256-bit AES key (stored as *CrypLock.key*).
2) User selects a file to encrypt.
3) System pads the file → encrypts using AES-256 CBC mode → saves as .enc
4) To decrypt, the user selects the encrypted file and loads the same key.
5) The system decrypts and removes padding to restore the original file.



III. Implementations

The program is written in Python and uses a few main components:

1. Libraries

- **os** – for generating random keys and handling files.
- **tkinter** – creates the simple GUI for buttons and file selection.
- **cryptography (AES)** – performs the actual encryption and decryption.
- **padding (PKCS7)** – makes sure the file fits the AES block size.

2. Key Functions

- **generate_key()** – creates a 256-bit AES key and saves it as *CrypLock.key.*
- **load_key()** – loads the saved key when encrypting or decrypting.
- **encrypt_file()** – reads a file, pads it, encrypts it with AES-256 CBC, and saves a .enc file.
- **decrypt_file()** – reads the .enc file, decrypts it, removes padding, and restores the original file.

3. GUI

A simple Tkinter window is used with three buttons:

- Generate Key
- Encrypt File
- Decrypt File

This makes the program easy to use for beginners with no command-line experience.

IV. User Guide

1) **Requirement**
   - Install Python3 and Cryptography Libraries in your computer.
2) **How to Run**
   - Save source code as ".py" and run it on terminal, PowerShell or VS Code.
3) **How to use CrypLock**
   - Generates A Key
   - Encrypt File
     - ✓ Choose a file that you want to protect.
     - ✓ CrypLock encrypts it and create a new encrypted file with ".enc".
   - Decrypt File
     - ✓ Select ".enc" file that you want to decrypt.
     - ✓ The program restores the original file automatically.

Note: The key (CrypLock.key) must be in the same folder as the program.

V. Conclusion And Future Works

CrypLock demonstrates how cryptography can be applied in simple tools to protect user data. The project shows understanding of AES-256, file handling, and GUI development. While basic, it successfully encrypts and decrypts files in a secure way.

In the future, the project can be improved by adding:

- Password-protected key generation
- RSA public-key encryption to protect the AES key
- A more modern GUI
- Drag-and-drop file support
- Multiple file encryption at once.

VI. References

https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/?utm_source=chatgpt.com

https://medium.com/%40dheeraj.mickey/how-to-encrypt-and-decrypt-files-in-python-using-aes-a-step-by-step-guide-d0eb6f525e4e

https://www.kiteworks.com/risk-compliance-glossary/aes-256-encryption/