Secure File Sharing System

Name: Olakunle Olasubomi Priscilla

Task 3: Secure File Sharing System

Program: Future Interns Cybersecurity Internship

Date: September 2025

Introduction

In this project, I developed a Secure File Sharing System that leverages Python Flask as the backend framework and AES encryption for data protection. The application enables users to upload files, which are automatically encrypted before storage, and download files, which are decrypted before being delivered. This simulates a practical solution for confidential data sharing in industries such as healthcare, law, and corporate organizations.

Tools and Technologies

- Python Flask – for backend web application development
- PyCryptodome (AES) – for file encryption and decryption
- HTML / CSS – for a simple and interactive user interface
- Virtual Environment (venv) – to isolate project dependencies
- Web Browser (localhost) – for accessing and testing the system

Implementation Process

1. Environment Setup

- Installed Flask and PyCryptodome libraries.
- Structured the project into relevant folders: templates/, uploads/.

- Generated a secure AES key and stored it as an environment variable for safe key management.

2. Application Development

- Implemented app.py with Flask routes for:
- Uploading files (with AES encryption)
- Displaying available files
- Downloading files (with AES decryption)
- Designed a simple user interface (index.html) inside the templates/ directory.

3. Running the Application

- Activated the virtual environment and launched the Flask development server.
- Accessed the system locally via: http://127.0.0.1:5000.

Security Features

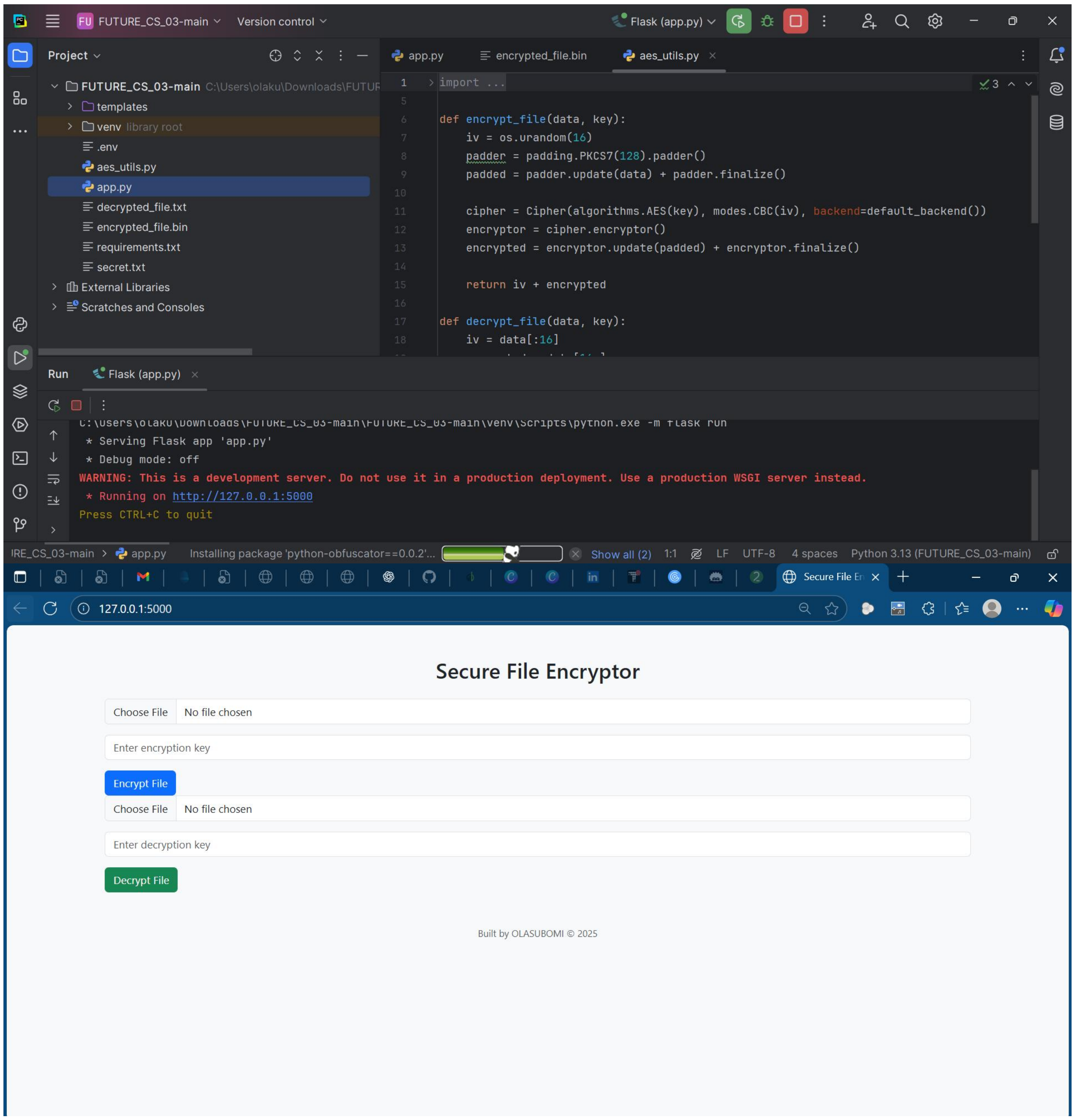
- AES Encryption (CBC Mode)
- Files are encrypted on upload using AES in CBC mode.
- CBC links each block of ciphertext to the previous one, ensuring stronger confidentiality.
- Requires padding for files not divisible by 16 bytes.
- Uses an Initialization Vector (IV) to prevent repeated patterns.

- Decryption on Download
- Files are decrypted only at the point of download, ensuring end-to-end confidentiality.
- Secure Key Management
- AES key is securely generated and stored as an environment variable.
- Prevents key exposure by avoiding hardcoding in source code.

Security Overview

The system ensures confidentiality and integrity of shared files. Even if attackers gain access to stored data, it remains unreadable without the AES key.

This reflects best practices in secure file sharing, widely used in organizations handling sensitive information.



```python
import ...

def encrypt_file(data, key):
    iv = os.urandom(16)
    padder = padding.PKCS7(128).padder()
    padded = padder.update(data) + padder.finalize()

    cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    encrypted = encryptor.update(padded) + encryptor.finalize()

    return iv + encrypted

def decrypt_file(data, key):
    iv = data[:16]
```

```
C:\Users\olaku\Downloads\FUTURE_CS_03-main\FUTURE_CS_03-main\venv\Scripts\python.exe -m flask run
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Secure File Encryptor

Choose File No file chosen

Enter encryption key

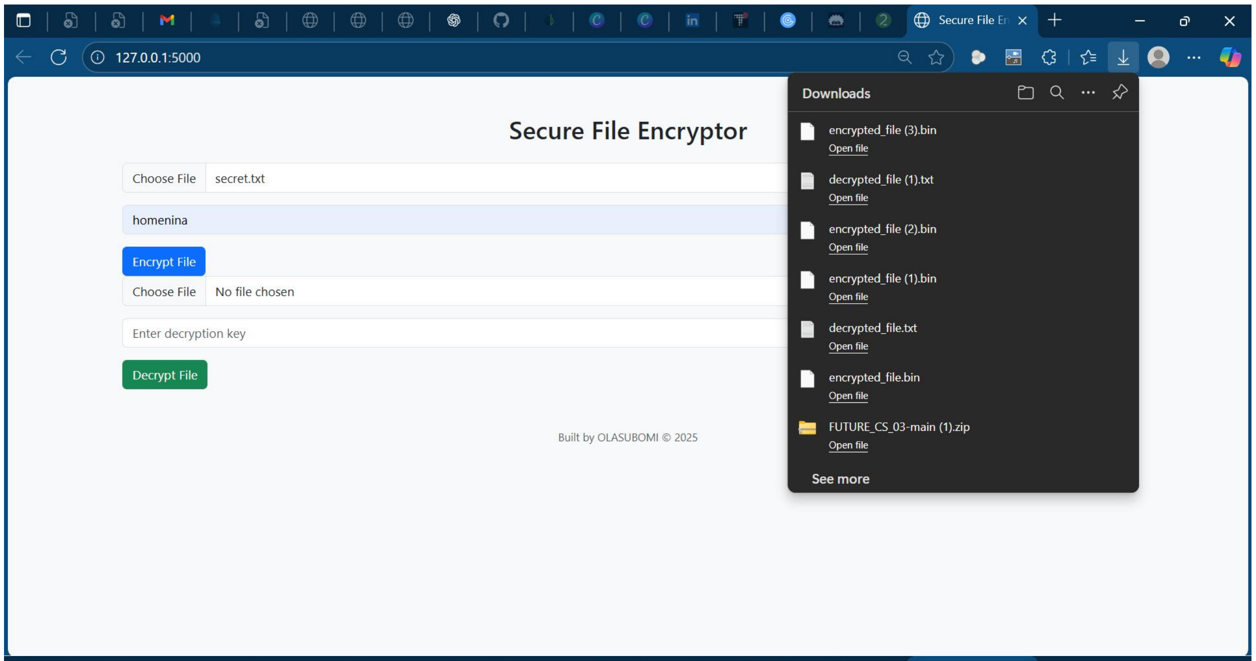
Encrypt File

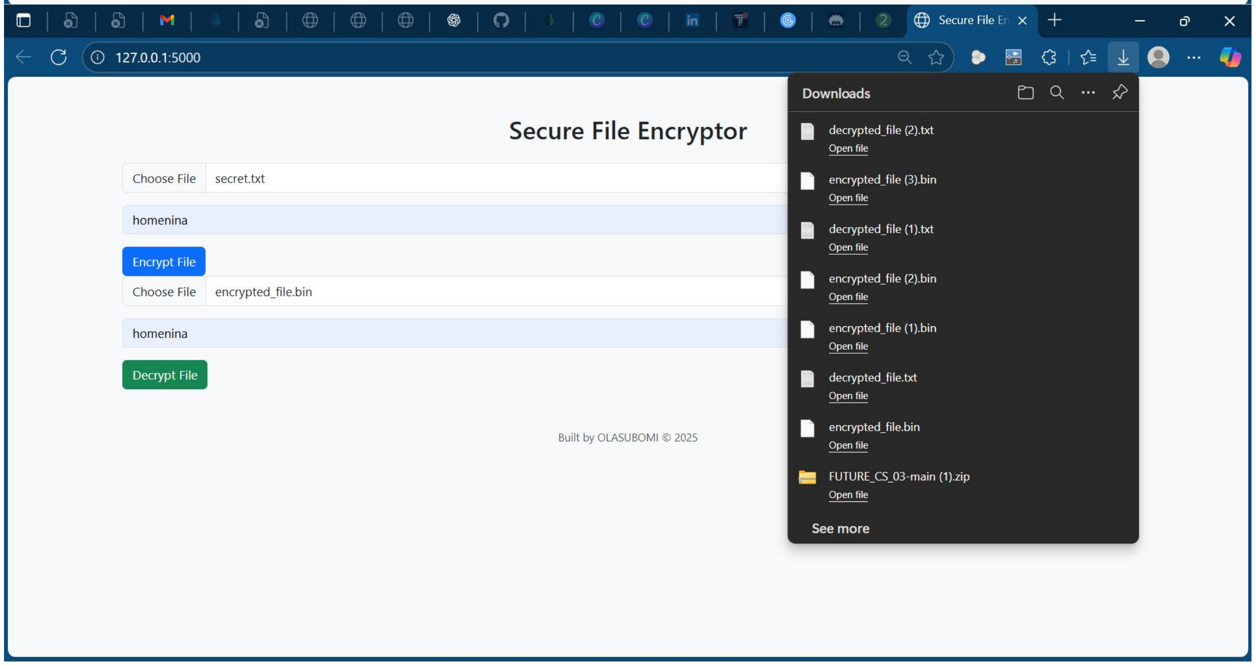
Choose File No file chosen

Enter decryption key

Decrypt File

Built by OLASUBOMI © 2025





Conclusion

This project provided hands-on experience in:

- Flask-based web application development
- Implementing AES encryption (CBC mode) for secure file handling
- Cryptographic key management
- Secure software design principles

It demonstrates how encryption ensures confidentiality and data protection, which are critical skills in cybersecurity and SOC operations.