Secure File Sharing System Report – Task 3

Intern name : Mithileshwaran

Internship domain : Cyber Security

Task tittle : Secure File Sharing System

Date : August 2025

Tool used : Python, Flask, PyCryptodome

1. Objective :

The goal of this task was to build a **secure file sharing system** that supports:

- File upload by users
- AES encryption of uploaded files
- Secure download of decrypted files

This system simulates secure file transfer practices commonly used in cybersecurity and enterprise environments.

2. Tools and Technologies :

Tool/Library	**Purpose**
Python	Programming language
Flask	Web framework for building upload/download APIs
PyCryptodome	Encryption library for AES implementation
HTML (templates)	Frontend for file upload form
VS Code	Code editor
GitHub	Version control and code publishing

3. Encryption Method Used :

The system uses AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode.

• A random 128-bit AES key is generated using get_random_bytes(16).

• The uploaded file is padded and encrypted using PyCryptodome.

• An IV (initialization vector) is generated per file and prepended to the ciphertext.

• For decryption, the IV is extracted and used to decrypt using the same key.

4. Futures Implemented:

• User-friendly web interface for file upload and download.

• Automatic AES encryption of each uploaded file.

• Separate storage for original, encrypted, and decrypted files.

• Decryption and download feature for secure access to original content.

• Links to download both encrypted and decrypted versions of files.

• Organized folder structure for easy maintenance.

5. Folder Structure:

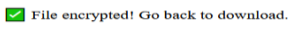
The project folder is organized as follows:

```
secure_file_share/
├── app.py
├── templates/
│   └── index.html
└── uploads/
    ├── original/
    ├── encrypted/
    └── decrypted/
```

6. Tested Performed:

1. Host is running

2. File Upload Page



3. Download link (encrypted & decrypted file)

The system was tested using multiple text files and other formats. Each file was successfully:

• Uploaded and saved in the original folder.

• Encrypted and saved in the encrypted folder.

• Decrypted and downloaded through the web interface.

The output matched the original input content, confirming encryption-decryption accuracy.

7)Security Considerations:

• AES encryption ensures strong protection for files at rest.

• Each file uses a random IV to prevent pattern attacks.

• Decryption happens only upon user request to avoid storing plaintext.

• Temporary decrypted files can be auto-deleted for better security (optional improvement).

• Key is generated per session, which is safe for demos; for production, proper key management is needed.