INTERNET OF THINGS & CRYPTOGRAPHY

ACTIVITY – 1

AES Encryption

Using Python libraries

Submitted by:

Gunav . S

3rd Yr.MSc.DFIS

24MSRDF041

Submitted to :

MRS. PRIYANKSHA DAS

Assistant Professor

Department of Forensic Science

JAIN (DEEMED-TO-BE University)

School of Sciences

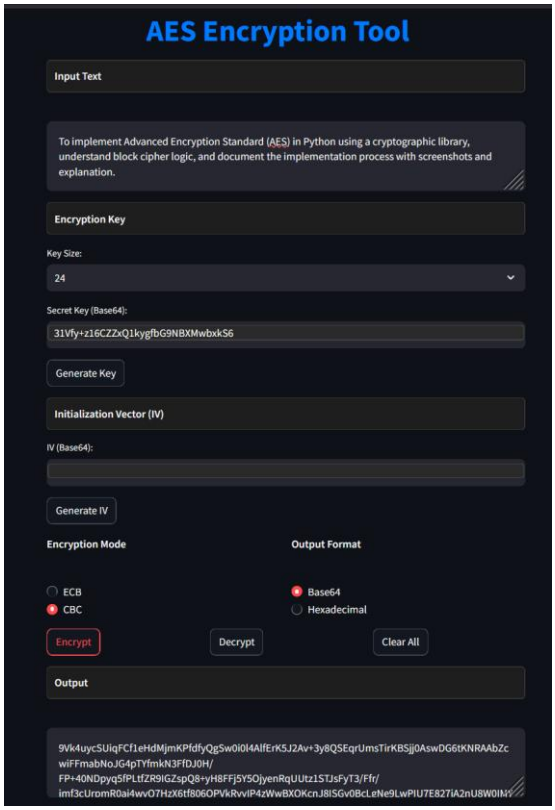
1. Code Overview with Comments and Screenshots

The following is a detailed explanation of the aes_tool.py script, including its structure, functionality, and usage. The code implements an interactive AES encryption/decryption tool using Streamlit for web-based UI.

Code (with inline comments)

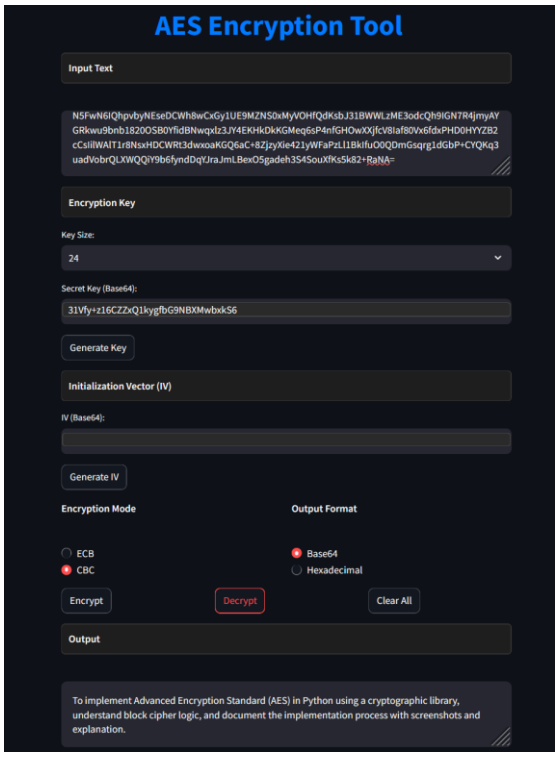
```
import streamlit as st

import os

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from cryptography.hazmat.primitives import padding

import base64

import binascii

# --- AES Functions ---

def pad_data(data):

"""Pad data to be multiple of 16 bytes using PKCS7."""

padder = padding.PKCS7(128).padder()

padded_data = padder.update(data)

padded_data += padder.finalize()

return padded_data

def unpad_data(padded_data):

"""Remove PKCS7 padding."""

unpadder = padding.PKCS7(128).unpadder()

data = unpadder.update(padded_data)

data += unpadder.finalize()

return data

def encrypt_aes(plaintext, key, mode, iv=None):

"""

Encrypt plaintext using AES in specified mode.

Returns: encrypted data (Base64 or Hex)

"""

if mode == 'ECB':

cipher = Cipher(algorithms.AES(key), modes.ECB())

else:  # CBC
```

- **Screenshot 1: Encryption Mode (CBC, Base64 Output)**



This shows the interface after entering text and clicking "Encrypt". The output is a long Base64-encoded string.

- **Screenshot 2: Decryption Mode**



After pasting the encrypted Base64 string and clicking "Decrypt", the original plaintext is recovered.

2. Explanation of Input/Output

Input Text	The plain text to be encrypted or decrypted. Supports any UTF-8 string.
Key Size	Selects AES key length: 128-bit (16 bytes), 192-bit (24 bytes), or 256-bit (32 bytes).
Secret Key (Base64)	A randomly generated key used for encryption/decryption. Stored in Base64 format. Must match between encryption and decryption.
IV (Base64)	Initialization vector (only used in CBC mode). Random 16-byte value prepended to ciphertext. Ensures uniqueness even with same key.
Mode	Choice between ECB (Electronic Codebook) and CBC (Cipher Block Chaining). CBC is more secure.
Output Format	Choose between Base64 (readable) or Hexadecimal (compact).
Output	Encrypted data (Base64/Hex) or decrypted plaintext.

3. Summary: What is AES and Where Is It Used?

What is AES?

AES (Advanced Encryption Standard) is a symmetric-key block cipher adopted by the U.S. government in 2001. It supports key sizes of 128, 192, and 256 bits and operates on 128-bit blocks.

It is considered highly secure and is widely used across industries due to its speed and resistance to known attacks.

How Does AES Work?

- **Symmetric**: Same key is used for encryption and decryption.
- **Block Cipher**: Processes data in fixed 128-bit blocks.
- **Rounds**: Uses multiple rounds (10, 12, or 14 depending on key size) involving substitution, permutation, mixing, and key addition.
- **Modes of Operation**: ECB, CBC, CFB, OFB, CTR — each defines how blocks are processed.

Where Is AES Used?

AES is foundational in modern cybersecurity:

Use Case	Description
Wi-Fi Security (WPA2/WPA3)	AES-CCMP protects wireless communications.
SSL/TLS	AES is used to encrypt data transmitted over HTTPS (e.g., banking, email).
Disk Encryption	Tools like BitLocker (Windows), FileVault (macOS), and LUKS use AES.
Cloud Storage	AWS S3, Google Drive, etc., use AES to protect stored files.
Secure Messaging Apps	Signal, WhatsApp use AES for end-to-end encryption.

4. Library Used: cryptography

Library Name: cryptography

- Official Python package for cryptographic primitives.
- Provides high-level APIs for encryption, hashing, signing, etc.
- Secure and actively maintained.

Installation Steps

```
pip install cryptography
```

Why This Library?

- **Secure**: Audited and follows best practices.
- **Easy to Use**: High-level abstractions like Cipher, Padding.
- **Cross-platform**: Works on Windows, macOS, Linux.

5. Conclusion

This Streamlit application demonstrates a practical implementation of AES encryption and decryption using the cryptography library. It allows users to:

- **Generate keys and IVs securely.**
- **Choose between ECB and CBC modes.**
- **Encode outputs in Base64 or Hex.**
- **Easily encrypt and decrypt text.**

While ECB is not recommended for real-world use, this tool helps visualize the differences between modes and reinforces understanding of block ciphers.

Final Notes

- Always use **CBC or better (GCM)** for production.
- Never reuse IVs in CBC mode.
- Store keys securely (e.g., environment variables, KMS).