Assignment 1: Exploring Cybersecurity Tools for AES Encryption and Decryption

Chaitanya Bharathi Institute of Technology
Department of Information Technology
Class: V Semester IT1
Name: D Chandana
Roll No: 160123737012

Abstract

This assignment explores the implementation of **AES encryption and decryption** for secure text and file handling. Using the **Python Cryptography library** in Google Colab, text strings and files are encrypted with AES-256 in CBC mode. The encrypted outputs are then validated using **CyberChef**, a web-based cryptographic tool, to ensure correctness and interoperability. The project demonstrates practical methods for maintaining data confidentiality and provides hands-on experience in applying cryptographic techniques for secure data handling in cybersecurity contexts.

Aim

To implement AES-based encryption and decryption for text and files in Python and validate the outputs using CyberChef to demonstrate secure data handling.

Problem Statement

The primary challenge addressed in this assignment is ensuring the **confidentiality of sensitive information** during storage and transmission. This involves encrypting both text and files using AES and verifying that the encrypted data can be accurately decrypted in another environment. The problem demonstrates interoperability, correctness, and reliability of cryptographic solutions in practical cybersecurity scenarios.

Tools Used

1. **Python Cryptography Library** – Used in Google Colab to implement AES encryption and decryption for text and files.

2. **CyberChef** – Web-based cryptography tool used to validate encrypted data and ensure correct decryption using the same key and IV.

Description of CyberChef

CyberChef is a free, open-source web application developed by GCHQ. It provides a wide range of tools for **data analysis, cryptography, encoding, and decoding**. CyberChef allows users to construct “recipes” to apply multiple operations on input data, such as encrypting, decrypting, encoding, or hashing. Its visual interface simplifies the process of verifying cryptographic outputs, making it a valuable tool for learning and validating encryption techniques like AES.

Approach

1. Implement **AES encryption in Python** using Google Colab for both text and files.

2. Print the **Key** and **IV** in hex format for easy copy-paste into CyberChef.

3. Encrypt text and file inputs in Colab.

4. Validate results in CyberChef by performing decryption with the same key and IV.

5. Compare decrypted outputs with original inputs to ensure correctness.

Code Implementation

Text Encryption and Decryption in Colab

```
# AES Text Encryption/Decryption in Google Colab



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

from cryptography.hazmat.primitives import padding

from base64 import b64encode, b64decode



# Fixed AES Key and IV for reproducibility

aes_key = bytes.fromhex("6c318977531e763533d3ca6eb242d412")

iv = bytes.fromhex("89ea40e81d6b0cb2f4c6801d8b7cfa15")



print("AES Key (hex, copy to CyberChef):", aes_key.hex())

print("IV (hex, copy to CyberChef):", iv.hex())



# User input for encryption

plaintext = input("Enter a message to encrypt: ").encode()



# Apply PKCS7 padding

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

padded_data = padder.update(plaintext) + padder.finalize()



# AES-CBC Encryption
```

```
cipher = Cipher(algorithms.AES(aes_key), modes.CBC(iv))

encryptor = cipher.encryptor()

ciphertext = encryptor.update(padded_data) + encryptor.finalize()


# Convert to Hex and Base64 for display

ciphertext_hex = ciphertext.hex()

ciphertext_b64 = b64encode(ciphertext).decode()


print("\nEncrypted message (Hex):", ciphertext_hex)

print("Encrypted message (Base64):", ciphertext_b64)


# AES-CBC Decryption

decryptor = cipher.decryptor()

decrypted_padded = decryptor.update(ciphertext) + decryptor.finalize()


# Remove padding

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

decrypted = unpadder.update(decrypted_padded) + unpadder.finalize()


print("\nDecrypted message:", decrypted.decode())
```

File Encryption and Decryption in Colab

```
# AES File Encryption/Decryption in Google Colab

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

from cryptography.hazmat.primitives import padding

from google.colab import files



# Fixed AES Key and IV

aes_key = bytes.fromhex("6c318977531e763533d3ca6eb242d412")

iv = bytes.fromhex("89ea40e81d6b0cb2f4c6801d8b7cfa15")



print("AES Key (hex, copy to CyberChef):", aes_key.hex())

print("IV (hex, copy to CyberChef):", iv.hex())



# Upload file

uploaded = files.upload()

for filename in uploaded.keys():

    with open(filename, "rb") as f:

        plaintext = f.read()



    # Padding

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

    padded_data = padder.update(plaintext) + padder.finalize()
```

```python
# Encrypt

cipher = Cipher(algorithms.AES(aes_key), modes.CBC(iv))

encryptor = cipher.encryptor()

ciphertext = encryptor.update(padded_data) + encryptor.finalize()


# Save ciphertext as hex file (for CyberChef)

hex_ciphertext = ciphertext.hex()

encrypted_filename = "encrypted_" + filename + ".hex"

with open(encrypted_filename, "w") as f:

    f.write(hex_ciphertext)

files.download(encrypted_filename)

print("Encrypted file (HEX) ready for download:", encrypted_filename)


# Decrypt back (Python verification)

decryptor = cipher.decryptor()

decrypted_padded = decryptor.update(ciphertext) + decryptor.finalize()

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

decrypted = unpadder.update(decrypted_padded) + unpadder.finalize()


decrypted_filename = "decrypted_" + filename

with open(decrypted_filename, "wb") as f:

    f.write(decrypted)

files.download(decrypted_filename)
```

```
    print("Decrypted file ready for download:", decrypted_filename)
```

Demonstration in CyberChef

1. **Text Encryption & Decryption**: Paste the Hex-encoded ciphertext in CyberChef. Use AES Decrypt with the provided Key and IV. Output matches original input.

2. **File Encryption & Decryption**: Upload the Hex file in CyberChef, set AES Decrypt mode with the same Key and IV. Decrypted output matches original file content.

CyberChef provides a **visual, web-based interface** that allows users to apply cryptographic operations, encode/decode data, and validate encrypted outputs easily. It is highly useful for verifying programmatic encryption results.

Please refer to the link https://github.com/Chandana909/CS_CyberChef-_tool_exploration for th code files and implementation videos.

Conclusion

The project successfully demonstrates AES-based encryption and decryption for both text and files. Google Colab was used for implementation and automation, while CyberChef verified correctness across platforms. The solution highlights the reliability and interoperability of AES as a cryptographic tool, emphasizing its practical utility in cybersecurity applications.