**Final Year B.Tech. (CSE) – VII [ 2024-25]**

**6CS451: Cryptography and Network Security Lab (C&NS Lab)**

**Date: 26/08/2024**

**Assignment 6**

**PRN:** 21510017  **Name:** Onkar Anand Yemul

1. **Apply AES algorithm for practical applications**

Ans:

The Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm that is both fast and secure. It is the standard encryption algorithm used by governments, financial institutions, and many other organizations. Unlike DES, which is now considered insecure, AES is robust and provides a high level of security.

**AES (Advanced Encryption Standard)**

1. **Key Size**: AES supports three key sizes: 128 bits, 192 bits, and 256 bits, depending on the security level required.
2. **Plaintext Input Size**: AES operates on 128-bit blocks of plaintext, meaning it encrypts 128 bits of data at a time.
3. **Number of Rounds**:
   * **AES-128**: 10 rounds.
   * **AES-192**: 12 rounds.
   * **AES-256**: 14 rounds. Each round has a unique transformation, with more rounds for longer keys.
4. **Key Steps**:
   * **Initial Round (AddRoundKey)**: XOR the plaintext block with the initial round key.
   * **Rounds (Repeated Steps)**:
     + **SubBytes**: Each byte is substituted with another based on a substitution table (S-box).
     + **ShiftRows**: Rows of the data block are shifted to the left by varying offsets.
     + **MixColumns**: Columns are transformed for diffusion (spreads out the plaintext bits).
     + **AddRoundKey**: XOR with a round-specific key derived from the main key.
   * **Final Round**: The last round omits the **MixColumns** step and applies only **SubBytes**, **ShiftRows**, and **AddRoundKey** to complete encryption.

**Note**: AES is considered very secure for modern encryption needs due to its large key sizes and rigorous transformations across multiple rounds.

The output size is **128 bits** (16 bytes), regardless of the key size (128, 192, or 256 bits). Each 128-bit block of plaintext produces a 128-bit block of ciphertext.

**Practical Application of AES Algorithm**

We can use the **pycryptodome** library in Python to implement AES encryption and decryption. The AES algorithm can work with key sizes of 128, 192, or 256 bits, and it operates on 128-bit blocks. In this example, we'll use AES with a 256-bit key in Cipher Block Chaining (CBC) mode.

**Python Code:**

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

from Crypto.Random import get\_random\_bytes

def aes\_encrypt(plain\_text, key):

    """

    Encrypt the plain text using AES algorithm.

    Parameters:

    plain\_text (str): The text to be encrypted.

    key (bytes): The encryption key (must be 16, 24, or 32 bytes long).

    Returns:

    bytes: The initialization vector (IV) and the encrypted cipher text.

    """

    cipher = AES.new(key, AES.MODE\_CBC)

    iv = cipher.iv  # Initialization vector

    padded\_text = pad(plain\_text.encode(), AES.block\_size)

    encrypted\_text = cipher.encrypt(padded\_text)

    return iv, encrypted\_text

def aes\_decrypt(iv, cipher\_text, key):

    """

    Decrypt the cipher text using AES algorithm.

    Parameters:

    iv (bytes): The initialization vector used during encryption.

    cipher\_text (bytes): The encrypted text to be decrypted.

    key (bytes): The decryption key (must be 16, 24, or 32 bytes long).

    Returns:

    str: The decrypted plain text.

    """

    cipher = AES.new(key, AES.MODE\_CBC, iv)

    decrypted\_text = unpad(cipher.decrypt(cipher\_text), AES.block\_size)

    return decrypted\_text.decode()

def main():

    """

    The main function to run the AES encryption and decryption program.

    """

    key = get\_random\_bytes(32)  # Generate a random 32-byte key for AES (256-bit)

    print(f"\nGenerated Key (in hexadecimal): {key.hex()}")

    while True:

        print("\nMenu:")

        print("1. Encrypt Text")

        print("2. Decrypt Text")

        print("3. Quit")

        choice = input("\nEnter your choice: ")

        if choice == '1':

            # Encrypt a message

            plain\_text = input("\nEnter the plain text to encrypt: ")

            # Encrypt the plaintext

            iv, encrypted\_text = aes\_encrypt(plain\_text, key)

            print(f"\n[Encryption Process]")

            print(f"Initialization Vector (IV) (in hexadecimal): {iv.hex()}")

            print(f"Padded Plain Text (in hexadecimal): {pad(plain\_text.encode(), AES.block\_size).hex()}")

            print(f"Encrypted Text (in hexadecimal): {encrypted\_text.hex()}")

        elif choice == '2':

            # Decrypt a message

            iv\_hex = input("\nEnter the Initialization Vector (IV) in hexadecimal: ")

            encrypted\_text\_hex = input("Enter the Encrypted Text in hexadecimal: ")

            # Convert hex inputs to bytes

            iv = bytes.fromhex(iv\_hex)

            encrypted\_text = bytes.fromhex(encrypted\_text\_hex)

            # Decrypt the ciphertext

            try:

                decrypted\_text = aes\_decrypt(iv, encrypted\_text, key)

                print(f"\n[Decryption Process]")

                print(f"Decrypted Text: {decrypted\_text}")

            except ValueError as e:

                print(f"\nError during decryption: {e}")

        elif choice == '3':

            print("\nExiting the program.")

            break

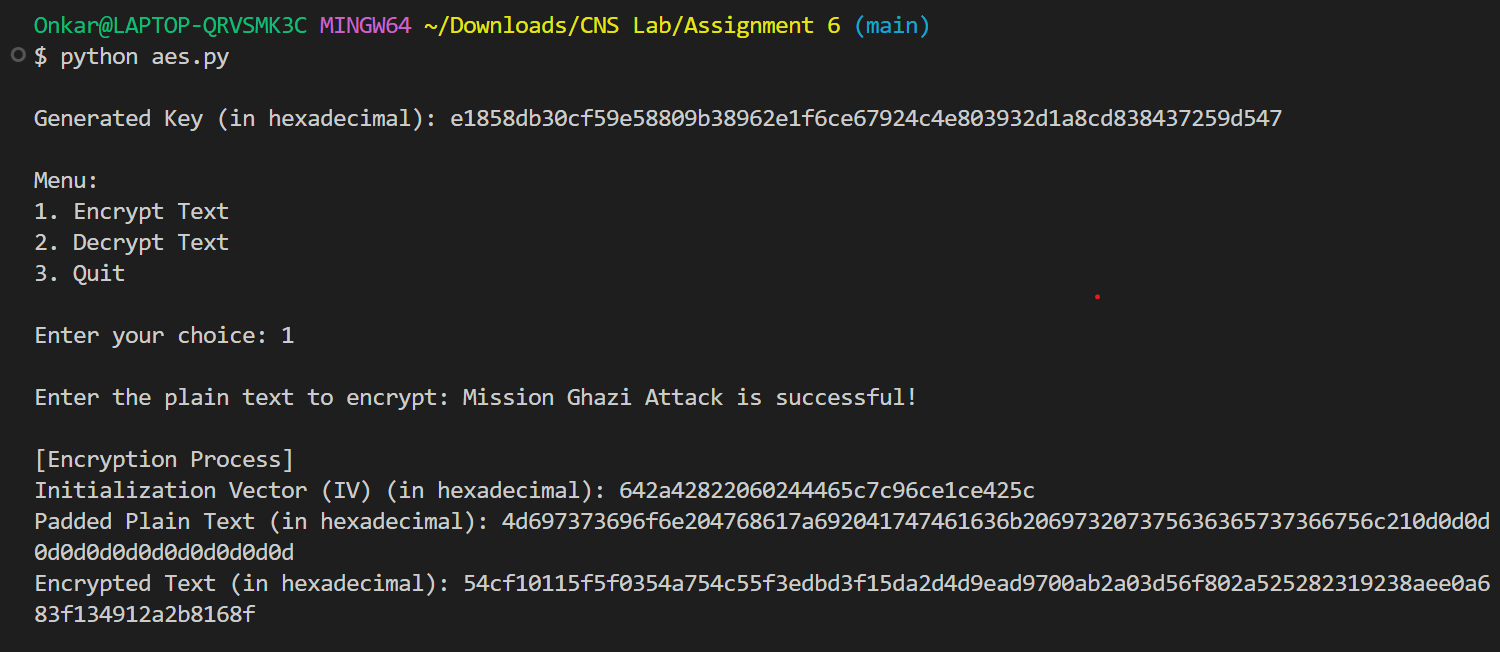
        else:

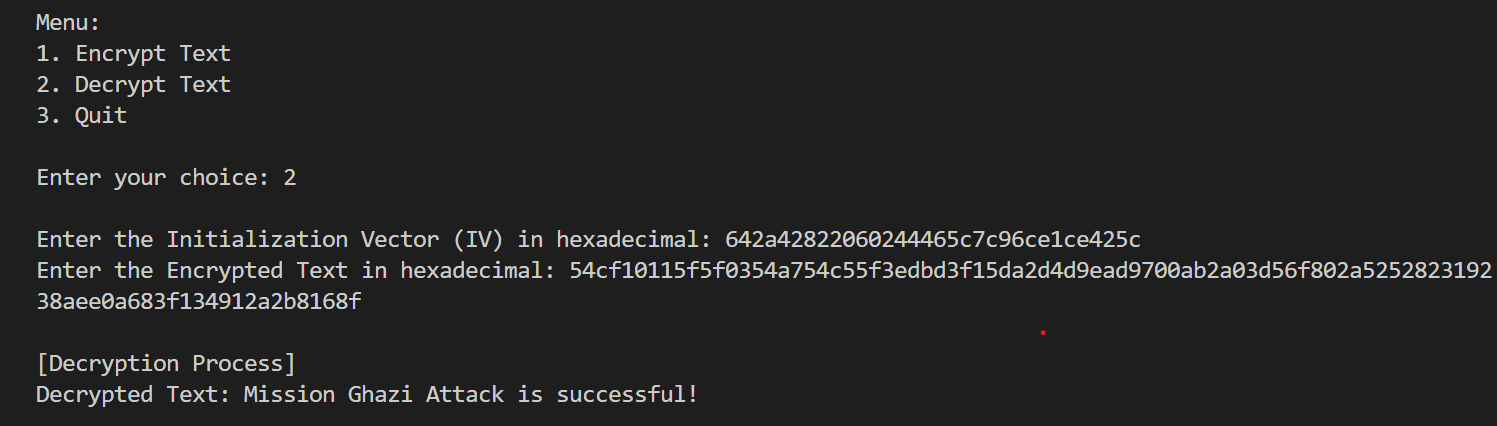
            print("\nInvalid choice. Please enter 1, 2, or 3.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Output:**

****



**Practical Applications of AES:**

* **File Encryption:** Encrypting sensitive files before storing them on disk.
* **Secure Communication:** Ensuring that data sent over the network remains confidential.
* **Data Protection in Applications:** Encrypting user data, such as passwords, to protect them from unauthorized access.

AES is widely adopted due to its strength and efficiency, and it remains the standard for securing digital data across various industries.