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

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

**Date: 26/08/2024**

**Assignment 5**

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

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

**Ans:**The Data Encryption Standard (DES) is a symmetric-key algorithm for the encryption of digital data. Although DES is now considered insecure for many applications due to its small key size, it is still an important algorithm for understanding the basics of cryptography.

**DES (Data Encryption Standard)**

1. **Key Size**: DES uses a 56-bit key for encryption. While the original key is 64 bits, 8 bits are used for parity (error checking), making the effective key size 56 bits.
2. **Plaintext Input Size**: DES operates on 64-bit blocks of plaintext, meaning it encrypts 64 bits of data at a time.
3. **Number of Rounds**: DES performs 16 rounds of encryption. Each round includes a unique transformation of the data using the key and various mathematical operations.
4. **Key Steps**:
   * **Initial Permutation**: The plaintext block undergoes an initial permutation (rearrangement).
   * **Rounds**: For each round, a different 48-bit subkey is generated from the main 56-bit key and applied to the data using substitution (S-boxes) and permutation.
   * **Final Permutation**: After all 16 rounds, a final permutation (reverse of the initial one) is applied to produce the ciphertext.

**Note**: DES is considered insecure by modern standards because its 56-bit key size is too small, making it vulnerable to brute-force attacks.

**Practical Application of DES Algorithm**

To apply the DES algorithm in a practical application, we can use the **pycryptodome** library in Python, which provides an implementation of DES. Below is an example that demonstrates how to use DES to encrypt and decrypt a message.

**Python Code:**

from Crypto.Cipher import DES

from Crypto.Util.Padding import pad, unpad

from Crypto.Random import get\_random\_bytes

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

    """

    Encrypt the plain text using DES algorithm.

    Parameters:

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

    key (bytes): The encryption key (must be 8 bytes long).

    Returns:

    bytes: The encrypted cipher text.

    """

    print(f"\n[Encryption Process]")

    print(f"Plain Text: {plain\_text}")

    # Initialize the DES cipher in ECB mode

    cipher = DES.new(key, DES.MODE\_ECB)

    print(f"DES Cipher initialized with ECB mode and key: {key.hex()}")

    # Padding the plain text to match DES block size (8 bytes)

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

    print(f"Padded Plain Text (in hexadecimal): {padded\_text.hex()}")

    # Encrypting the padded text

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

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

    return encrypted\_text

def des\_decrypt(cipher\_text, key):

    """

    Decrypt the cipher text using DES algorithm.

    Parameters:

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

    key (bytes): The decryption key (must be 8 bytes long).

    Returns:

    str: The decrypted plain text.

    """

    print(f"\n[Decryption Process]")

    print(f"Cipher Text to Decrypt (in hexadecimal): {cipher\_text.hex()}")

    # Initialize the DES cipher in ECB mode for decryption

    cipher = DES.new(key, DES.MODE\_ECB)

    print(f"DES Cipher initialized with ECB mode and key: {key.hex()}")

    # Decrypt the ciphertext

    decrypted\_padded\_text = cipher.decrypt(cipher\_text)

    print(f"Decrypted Padded Text (in hexadecimal): {decrypted\_padded\_text.hex()}")

    # Unpad the decrypted text

    decrypted\_text = unpad(decrypted\_padded\_text, DES.block\_size)

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

    return decrypted\_text.decode()

def main():

    """

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

    """

    # Generate a random 8-byte key for DES

    key = get\_random\_bytes(8)

    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: ")

            encrypted\_text = des\_encrypt(plain\_text, key)

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

        elif choice == '2':

            # Decrypt a message

            encrypted\_text\_hex = input("\nEnter the Encrypted Text in hexadecimal to decrypt: ")

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

            # Decrypt the ciphertext

            try:

                decrypted\_text = des\_decrypt(encrypted\_text, key)

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

            except ValueError as e:

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

        elif choice == '3':

            print("\nExiting the program.")

            break

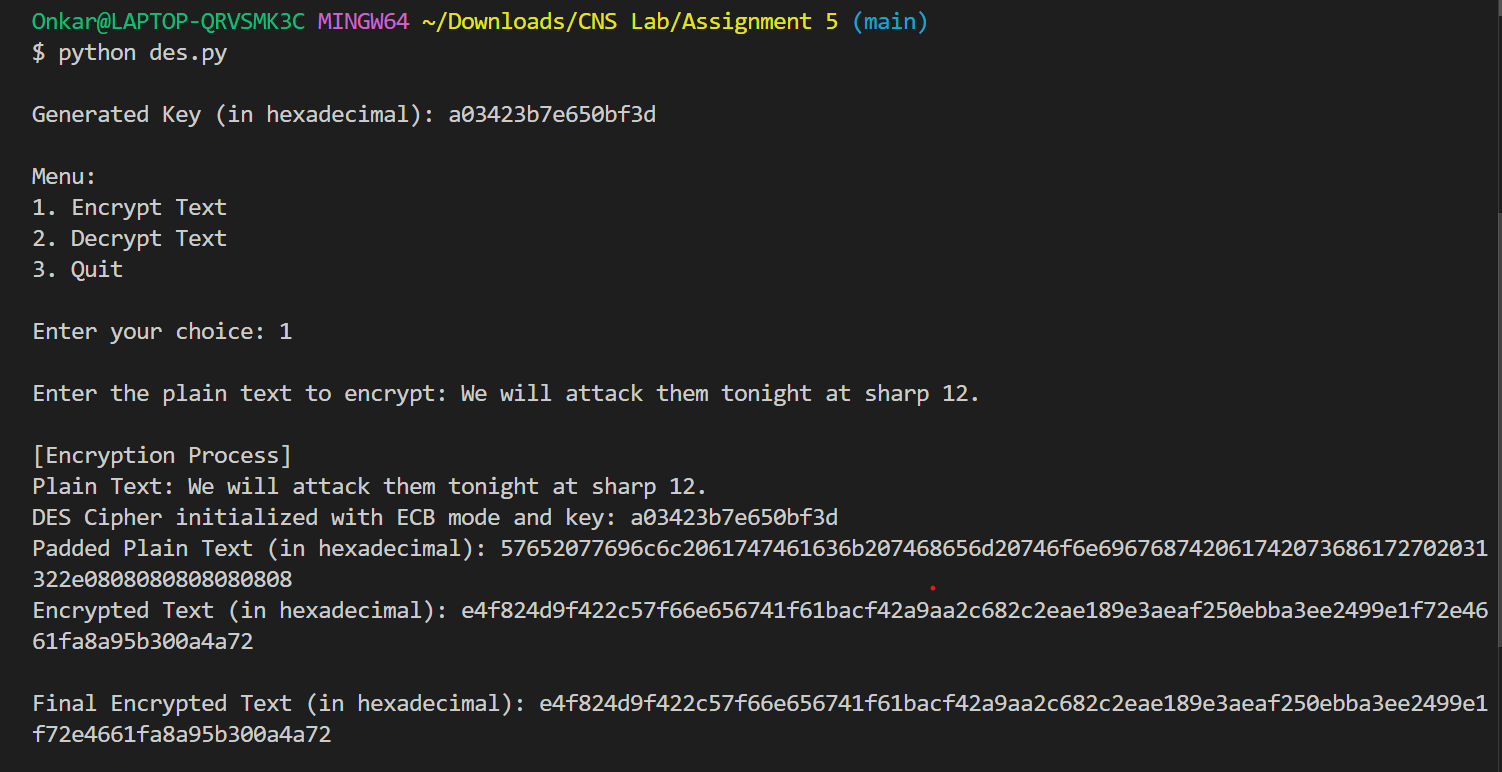
        else:

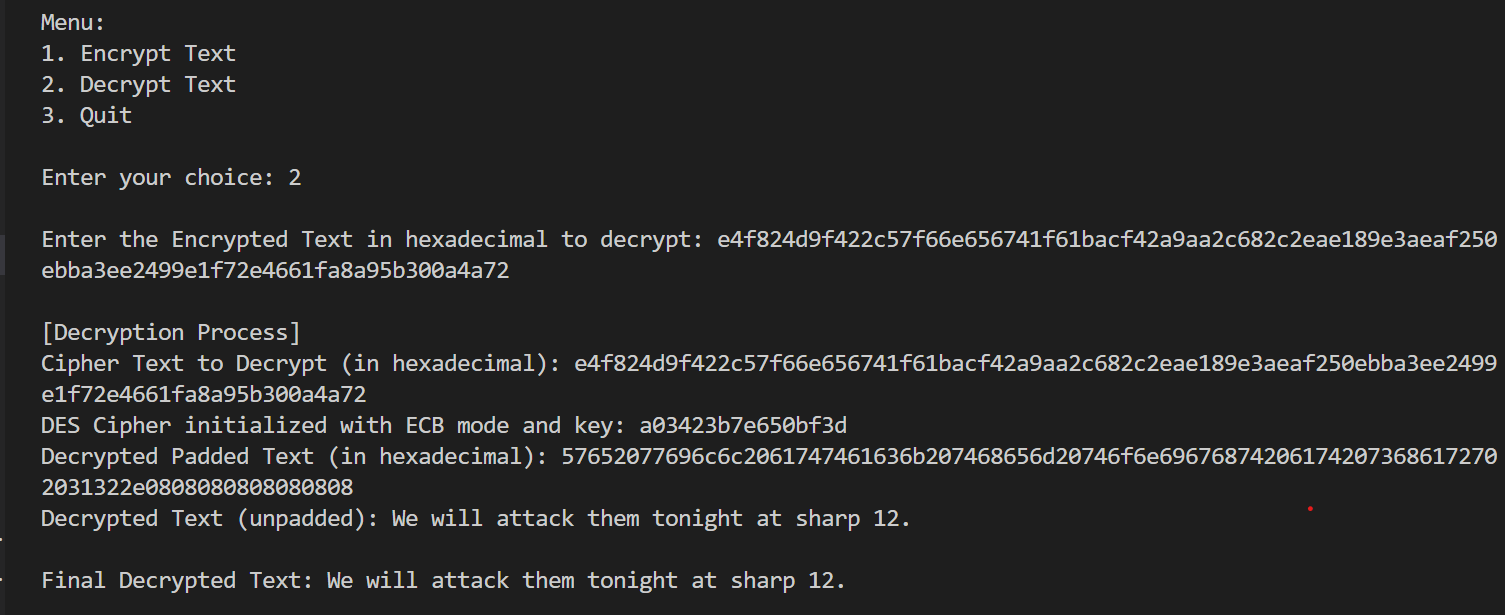
            print("\nInvalid choice. Please enter 1, 2, or 3.")

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

    main()

**Output:**

****

****

**Practical Applications:**

* **File Encryption:** DES can be used to encrypt sensitive files before storing them in insecure locations.
* **Secure Communication:** DES ensures that messages sent over a network are unreadable to unauthorized parties.
* **Password Storage:** Encrypting passwords before storing them in databases (though modern standards recommend stronger algorithms like AES).

While DES itself is outdated and not recommended for secure applications, understanding how it works is crucial for grasping more advanced encryption algorithms like AES.