Name: Tejas Redkar

Batch: C2; Roll no: 44

PRN: 1032210937

**ICS Lab Assignment 2**

**Lab A2**: Implement simple DES symmetric key algorithm using python or java or C++

**Objective of Lab**

1. To study understand and implement DES symmetric key algorithm

**Theory:**

**DES Algorithm:**

The Data Encryption Standard (DES) is a symmetric-key block cipher encryption algorithm that was widely used to secure data in the past. While DES is considered obsolete for modern cryptographic standards due to its short key length, understanding its operation provides insight into the fundamentals of block cipher design. Here's a detailed explanation of the DES algorithm:

1. Key Generation:

- DES starts with a 64-bit secret key, but only 56 bits are used for encryption. The other 8 bits are used for error checking and do not contribute to the key.

- The 56-bit key is divided into 16 subkeys, one for each of the 16 rounds. Each subkey is derived by shifting and permuting the original key bits.

2. Initial Permutation (IP):

- The 64-bit plaintext is permuted according to an initial permutation table (IP), which reorders the bits.

3. 16 Rounds of Feistel Network:

- DES uses a Feistel network structure, where the data is split into two 32-bit blocks (left and right) at the start of each round.

Each Round Consists of:

- Expansion (E-Box): The 32-bit right block is expanded to 48 bits by duplicating some of the bits.

- XOR with Round Key: The expanded right block is XORed with the current round's 48-bit subkey.

- Substitution (S-Boxes): The 48-bit result is divided into 8 6-bit blocks, each of which is substituted using predefined S-boxes. Each S-box produces a 4-bit output.

- Permutation (P-Box): The 32-bit output of the S-boxes is permuted according to a fixed permutation table (P-box).

- XOR with Left Block: The permuted 32-bit result is XORed with the left block from the previous round.

4. Final Permutation (FP):

- After 16 rounds, the left and right blocks are swapped, and the result is permuted according to a final permutation table (FP).

5. Output:

- The final 64-bit ciphertext is produced.

**Key points about DES:**

- DES uses 16 rounds of Feistel network, each with expansion, substitution (S-boxes), permutation, and XOR operations.

- It has a fixed block size of 64 bits and uses a 56-bit key, which is expanded into 16 round keys.

- DES is relatively slow compared to modern ciphers, and its 56-bit key length makes it vulnerable to brute-force attacks.

**Code (DES):**

P10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]

P8 = [6, 3, 7, 4, 8, 5, 10, 9]

KEY = [1, 0, 1, 0, 0, 0, 0, 0, 1, 0]

def key\_order(key, order):

    newOrder = []

    for o in order:

        newOrder.append(key[o - 1])

    return newOrder

def left\_shift(val, shift):

    return val[shift:] + val[:shift]

def split(arr):

    n = len(arr)

    return arr[: n // 2], arr[n // 2 :]

def key\_gen():

    p10 = key\_order(KEY, P10)

    l, r = split(p10)

    l\_ls1, r\_ls1 = left\_shift(l, 1), left\_shift(r, 1)

    k1 = key\_order(l\_ls1 + r\_ls1, P8)

    l\_ls2, r\_ls2 = left\_shift(l\_ls1, 2), left\_shift(r\_ls1, 2)

    k2 = key\_order(l\_ls2 + r\_ls2, P8)

    return k1, k2

IP = [2, 6, 3, 1, 4, 8, 5, 7]

IP\_inv = [4, 1, 3, 5, 7, 2, 8, 6]

EP = [4, 1, 2, 3, 2, 3, 4, 1]

P4 = [2, 4, 3, 1]

S0 = [[1, 0, 3, 2], [3, 2, 1, 0], [0, 2, 1, 3], [3, 1, 3, 2]]

S1 = [[0, 1, 2, 3], [2, 0, 1, 3], [3, 0, 1, 0], [2, 1, 0, 3]]

def xor(a, b):

    return [a[i] ^ b[i] for i in range(len(a))]

def F(text, k):

    l, r = split(text)

    r\_ep = key\_order(r, EP)

    xor\_op = xor(r\_ep, k)

    l\_xor, r\_xor = split(xor\_op)

    matrix\_op = matrix(S0, l\_xor) + matrix(S1, r\_xor)

    p4 = key\_order(matrix\_op, P4)

    xor\_op = xor(l, p4)

    return xor\_op, r

def matrix(m, text):

    row = int(str(text[0]) + str(text[3]), 2)

    col = int(str(text[1]) + str(text[2]), 2)

    b = bin(m[row][col]).replace("b", "")

    return list(map(int, [b[-2], b[-1]]))

def sdes():

    k1, k2 = key\_gen()

    ip = key\_order(plain\_text, IP)

    a, b = F(ip, k1)

    c, d = F(b + a, k2)

    ip\_inv = key\_order(c + d, IP\_inv)

    return ip\_inv

def getUserInput():

    userInput = input("Enter the plain text: ")

    if len(userInput) != 8:

        print("Invalid input")

        return getUserInput()

    return [int(x) for x in userInput]

plain\_text = getUserInput()

cipher\_text = sdes()

print(f"Cipher text: {cipher\_text}")

**Output Screen shots**:



**Conclusion**:

We successfully implemented simple DES symmetric key algorithm, in Python. This implementation provides a practical demonstration of how this algorithm works for encrypting messages.

# FAQs:

# 

# What is the concept of fiestel cipher.

# Draw and describe DES algorithm briefly

# List and state broad level operations used internally in DES algorithm.

# Compare various block ciphers such as DES, AES, Blowfish etc..

# What are the Block cipher design guidelines.