# Optimized S-DES Implementation with User Input Key

# Permutation function

def permute(bits, perm):

return [bits[i - 1] for i in perm]

# Circular left shift

def left\_shift(bits, shifts=1):

return bits[shifts:] + bits[:shifts]

# XOR operation

def xor(bits1, bits2):

return [b1 ^ b2 for b1, b2 in zip(bits1, bits2)]

# S-Box substitution

def sbox\_substitution(bits, sbox):

row, col = (bits[0] << 1) | bits[3], (bits[1] << 1) | bits[2]

return [int(b) for b in format(sbox[row][col], '02b')]

# Key scheduling

def generate\_keys(key):

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

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

key = permute(key, P10)

left, right = key[:5], key[5:]

keys = []

for s in [1, 2]:

left, right = left\_shift(left, s), left\_shift(right, s)

keys.append(permute(left + right, P8))

return keys

# Feistel function

def f\_function(bits, key):

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]]

xored = xor(permute(bits, EP), key)

return permute(sbox\_substitution(xored[:4], S0) + sbox\_substitution(xored[4:], S1), P4)

# S-DES Encryption/Decryption

def sdes(text, key, encrypt=True):

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

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

K1, K2 = generate\_keys(key)

keys = (K1, K2) if encrypt else (K2, K1)

bits = permute(text, IP)

left, right = bits[:4], bits[4:]

for k in keys:

left, right = right, xor(left, f\_function(right, k))

return permute(right + left, IP\_inv)

# Convert text to/from binary

def text\_to\_binary(text):

return [int(bit) for char in text for bit in format(ord(char), '08b')]

def binary\_to\_text(binary):

return ''.join(chr(int(''.join(map(str, binary[i:i+8])), 2)) for i in range(0, len(binary), 8))

# User input key

def get\_user\_key():

while True:

key\_input = input("Enter a 10-bit key (e.g., 1010101010): ")

if len(key\_input) == 10 and all(bit in '01' for bit in key\_input):

return [int(bit) for bit in key\_input]

print("Invalid key. Please enter exactly 10 binary digits.")

# Main execution

plain\_text = input("Enter the plaintext message: ")

plain\_binary = text\_to\_binary(plain\_text)

key = get\_user\_key()

# Encrypt plaintext

cipher\_binary = [bit for i in range(0, len(plain\_binary), 8) for bit in sdes(plain\_binary[i:i+8], key, encrypt=True)]

cipher\_text = binary\_to\_text(cipher\_binary)

print("Plain Text:", plain\_text)

print("Cipher Text:", cipher\_text)

# Decrypt ciphertext

re\_encoded\_binary = text\_to\_binary(cipher\_text)

decrypted\_binary = [bit for i in range(0, len(re\_encoded\_binary), 8) for bit in sdes(re\_encoded\_binary[i:i+8], key, encrypt=False)]

decrypted\_text = binary\_to\_text(decrypted\_binary)

print("Decrypted Text:", decrypted\_text)

Enter the plaintext message: helloworld

Enter a 10-bit key (e.g., 1010101010): 1110011100

Plain Text: helloworld

Cipher Text: ÇÇYtYhÇé

Decrypted Text: helloworld

PS C:\Users\shashank zende\OneDrive\Desktop\mini-project>