program for Encrypt and decrypt in cipher block chaining mode using one of the following ciphers: affine modulo 256, Hill modulo 256, S-DES, DES. Test data for S-DES using a binary initialization vector of 1010 1010. A binary plaintext of 0000 0001 0010 0011 encrypted with a binary key of 01111 11101 should give a binary plaintext of 1111 0100 0000 1011. Decryption should work correspondingly.

# Simplified DES (S-DES) in CBC Mode

# Works with binary input and output

# No extra modules needed

# --- Helper functions ---

def permute(bits, order):

return ''.join(bits[i-1] for i in order)

def left\_shift(bits, n):

return bits[n:] + bits[:n]

def xor(a, b):

return ''.join('1' if x != y else '0' for x, y in zip(a, b))

# --- S-DES Key Generation ---

def generate\_keys(key10):

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

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

key = permute(key10, P10)

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

left, right = left\_shift(left, 1), left\_shift(right, 1)

K1 = permute(left + right, P8)

left, right = left\_shift(left, 2), left\_shift(right, 2)

K2 = permute(left + right, P8)

return K1, K2

# --- S-Boxes ---

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 sbox(bits, sbox):

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

col = int(bits[1]+bits[2], 2)

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

# --- S-DES Round Function ---

def fk(bits8, key):

IP8\_L, IP8\_R = bits8[:4], bits8[4:]

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

temp = permute(IP8\_R, EP)

temp = xor(temp, key)

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

left, right = sbox(left, S0), sbox(right, S1)

P4 = [2,4,3,1]

temp = permute(left + right, P4)

return xor(IP8\_L, temp) + IP8\_R

# --- S-DES Encrypt/Decrypt ---

def sdes\_encrypt(plaintext8, K1, K2):

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

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

bits = permute(plaintext8, IP)

bits = fk(bits, K1)

bits = bits[4:] + bits[:4] # swap halves

bits = fk(bits, K2)

return permute(bits, IP\_inv)

def sdes\_decrypt(ciphertext8, K1, K2):

# same as encrypt but reverse key order

return sdes\_encrypt(ciphertext8, K2, K1)

# --- CBC Mode Encryption/Decryption ---

def cbc\_encrypt(plaintext, key10, iv):

K1, K2 = generate\_keys(key10)

blocks = [plaintext[i:i+8] for i in range(0, len(plaintext), 8)]

prev = iv

ciphertext = ''

for block in blocks:

block = xor(block, prev)

enc = sdes\_encrypt(block, K1, K2)

ciphertext += enc

prev = enc

return ciphertext

def cbc\_decrypt(ciphertext, key10, iv):

K1, K2 = generate\_keys(key10)

blocks = [ciphertext[i:i+8] for i in range(0, len(ciphertext), 8)]

prev = iv

plaintext = ''

for block in blocks:

dec = sdes\_decrypt(block, K1, K2)

dec = xor(dec, prev)

plaintext += dec

prev = block

return plaintext

# --- Main Program ---

iv = '10101010'

key = '0111111101'

plaintext = '0000000100100011' # 16-bit plaintext

cipher = cbc\_encrypt(plaintext, key, iv)

decrypted = cbc\_decrypt(cipher, key, iv)

print("=== CBC Mode with S-DES ===")

print("IV: ", iv)

print("Key: ", key)

print("Plaintext: ", plaintext)

print("Ciphertext: ", cipher)

print("Decrypted: ", decrypted) 