

22. Write a python 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.

```
# -----
```

```
# S-DES HELPER FUNCTIONS
```

```
# -----
```

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

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

```
P4 = [2, 4, 3, 1]
```

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

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

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

```
# 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]
```

```
]
```

Code:

```
def permute(bits, table):  
    return ''.join(bits[i-1] for i in table)  
  
def left_shift(bits, n):  
    return bits[n:] + bits[:n]  
  
def key_generation(key10):  
    k = permute(key10, P10)  
    L, R = k[:5], k[5:]  
    L1, R1 = left_shift(L, 1), left_shift(R, 1)  
    K1 = permute(L1 + R1, P8)  
    L2, R2 = left_shift(L1, 2), left_shift(R1, 2)  
    K2 = permute(L2 + R2, P8)  
    return K1, K2  
  
def f_function(bits, key):  
    L, R = bits[:4], bits[4:]  
    ER = permute(R, EP)  
    x = ''.join('1' if ER[i] != key[i] else '0' for i in range(8))  
    L0, R0 = x[:4], x[4:]  
    row, col = int(L0[0] + L0[3], 2), int(L0[1] + L0[2], 2)  
    s0 = format(S0[row][col], '02b')  
    row, col = int(R0[0] + R0[3], 2), int(R0[1] + R0[2], 2)  
    s1 = format(S1[row][col], '02b')  
    out = permute(s0 + s1, P4)  
    return ''.join('1' if L[i] != out[i] else '0' for i in range(4)) + R  
  
def sdes_encrypt(block8, K1, K2):  
    x = permute(block8, IP)  
    y = f_function(x, K1)
```

```

    y = y[4:] + y[:4]
    z = f_function(y, K2)
    return permute(z, IPinv)

def sdes_decrypt(block8, K1, K2):
    x = permute(block8, IP)
    y = f_function(x, K2)
    y = y[4:] + y[:4]
    z = f_function(y, K1)
    return permute(z, IPinv)

# -----

# CBC MODE

# -----

def xor_bits(a, b):
    return ''.join('1' if a[i] != b[i] else '0' for i in range(len(a)))

def CBC_encrypt(plaintext, K1, K2, IV):
    blocks = [plaintext[i:i+8] for i in range(0, len(plaintext), 8)]
    cipher = ""
    prev = IV
    for blk in blocks:
        xored = xor_bits(blk, prev)
        c = sdes_encrypt(xored, K1, K2)
        cipher += c
        prev = c
    return cipher

def CBC_decrypt(ciphertext, K1, K2, IV):
    blocks = [ciphertext[i:i+8] for i in range(0, len(ciphertext), 8)]
    plain = ""
    prev = IV

```

```

for blk in blocks:

    ptemp = sdes_decrypt(blk, K1, K2)

    p = xor_bits(ptemp, prev)

    plain += p

    prev = blk

return plain

# -----

# TEST CASE GIVEN IN QUESTION

# -----

key10    = "0111111101"

IV       = "10101010"

plaintext = "00000001" "00100011" # 2 blocks

expected  = "11110100" "00001011" # expected ciphertext

K1, K2 = key_generation(key10)

cipher = CBC_encrypt(plaintext, K1, K2, IV)

decrypt = CBC_decrypt(cipher, K1, K2, IV)

print("Generated Ciphertext :", cipher)

print("Expected Ciphertext :", expected)

print("Decrypted Plaintext :", decrypt)

```

```

IDLE Shell 3.14.0
File Edit Shell Debug Options Window Help
Python 3.14.0 (tags/v3.14.0:ebf955d, Oct 7 2025, 10:15:03) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> ===== RESTART: C:/Users/Maria/OneDrive/Documents/ex22.py =====
Generated Ciphertext : 1111010000001011
Expected Ciphertext : 1111010000001011
Decrypted Plaintext : 0000000100100011
>>>

```