**Practical No : 08**

**Problem Statement:**

Write a program to implement S-DES cipher.

**Program:**

def P10(p): return ''.join(p[i-1] for i in [3, 5, 2, 7, 4, 10, 1, 9, 8, 6])

def P8(p): return ''.join(p[i-1] for i in [6, 3, 7, 4, 8, 5, 10, 9])

def P4(p): return ''.join(p[i-1] for i in [2, 4, 3, 1])

def LS1(p): return ''.join(p[i-1] for i in [2, 3, 4, 5, 1])

def LS2(p): return ''.join(p[i-1] for i in [3, 4, 5, 1, 2])

def IP(p): return ''.join(p[i-1] for i in [2, 6, 3, 1, 4, 8, 5, 7])

def IPinverse(p): return ''.join(p[i-1] for i in [4, 1, 3, 5, 7, 2, 8, 6])

def EP(p): return ''.join(p[i-1] for i in [4, 1, 2, 3, 2, 3, 4, 1])

def xor(s1, s2): return ''.join(str(int(a) ^ int (b)) for a, b in zip(s1, s2))

def sBox(p):

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 s(b, S):

return bin(S[int(b[0] + b[3], 2)][int(b[1] + b[2], 2)])[2:].zfill(2)

return s(p[:4], S0) + s(p[4:], S1)

def fk(p, k): return xor(p[:4], P4(sBox(xor(EP(p[4:]), k)))) + p[4:]

def keyGeneration(key):

key = P10(key)

k1, k2 = LS1(key[:5]), LS1(key[5:])

return P8(k1+k2), P8(LS2(k1) + LS2(k2))

def sDESEncryption(plaintext, key):

k1, k2 = keyGeneration(key)

print(f"K1: {k1}\nK2: {k2}")

ct = fk(IP(plaintext), k1)

return IPinverse(fk(ct[4:] + ct[:4], k2))

def sDESDecryption(ciphertext, key):

k1, k2 = keyGeneration(key)

pt = fk(IP(ciphertext),k2)

return IPinverse(fk(pt[4:] + pt[:4], k1))

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

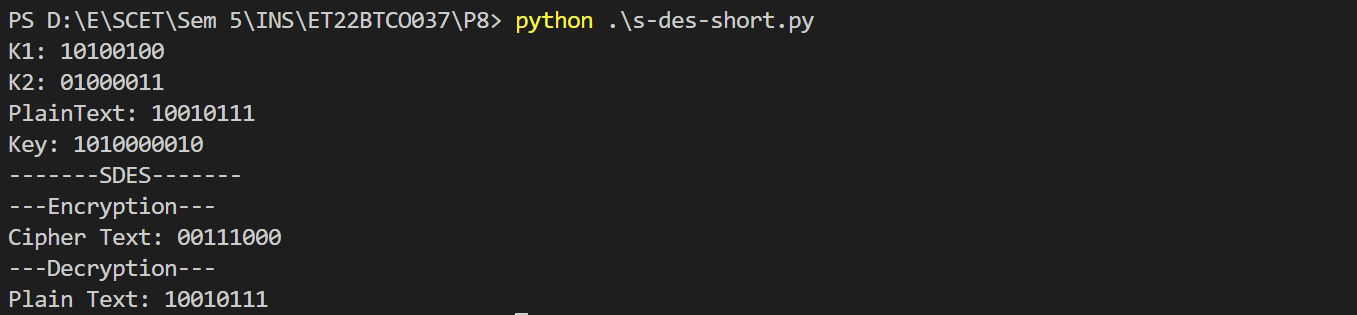
key, plaintext = "1010000010", "10010111"

ciphertext = sDESEncryption(plaintext, key)

print(f"PlainText: {plaintext}\nKey: {key}\n-------SDES-------\n---Encryption---")

print(f"Cipher Text: {ciphertext}\n---Decryption---\nPlain Text: {sDESDecryption(ciphertext, key)}")

**Output:**

****

**Conclusion:**

I implemented the Simplified Data Encryption Standard (S-DES) algorithm in Python. This algorithm enhances data security through a series of permutations and substitutions based on a 10-bit key, effectively encrypting and decrypting messages. However, its simplicity and short key length make it vulnerable to brute-force attacks. This emphasizes the importance of using more advanced encryption standards for robust security in modern applications.