In [62]:
```
#permutations for keys
p10_seq = (3, 5, 2, 7, 4, 10, 1, 9, 8, 6)
p8_seq = (6, 3, 7, 4, 8, 5, 10, 9)

#permutations for text
ip_seq = (2, 6, 3, 1, 4, 8, 5, 7)
inv_ip_seq = (4, 1, 3, 5, 7, 2, 8, 6)

#permutation to expand 4 bit to 8 bit
ep_seq = (4, 1, 2, 3, 2, 3, 4, 1)

#permutation for 4 bits
p4_seq = (2, 4, 3, 1)

#s boxes
s0_seq = [
            ["01", "00", "11", "10"],
            ["11", "10", "01", "00"],
            ["00", "10", "01", "11"],
            ["11", "01", "11", "10"]
        ]

s1_seq = [
            ["00", "01", "10", "11"],
            ["10", "00", "01", "11"],
            ["11", "00", "01", "00"],
            ["10", "01", "00", "11"]
        ]
```

In [63]:
```
def left_shift(s, bits):
    s = s[bits:] + s[:bits]

    return s
```

In [64]:
```
def permute_and_generate(ip, seq):
    s=""

    for val in seq:
        s += ip[val-1]

    return s
```

In [65]:
```
def generate_keys(key):

    p10 = permute_and_generate(key, p10_seq)

    p10_left = p10[0:5]
    p10_right = p10[5:10]

    ls1_left = left_shift(p10_left, 1)
    ls1_right = left_shift(p10_right, 1)

    k1 = permute_and_generate(ls1_left + ls1_right, p8_seq)

    print("k1 : ", k1)

    ls2_left = left_shift(ls1_left, 2)
    ls2_right = left_shift(ls1_right, 2)

    k2 = permute_and_generate(ls2_left + ls2_right, p8_seq)

    print("k2 : ", k2)

    return k1, k2
```

In [66]:
```
def find_xor(s1, s2):
    xor = ""

    for i in range(len(s1)):
        if s1[i] == s2[i]:
            xor += "0"
        else:
            xor += "1"

    return xor
```

In [67]:
```
def find_s0_s1(xor_half, lookup_table):
    r = (int(xor_half[0]) * 2) + int(xor_half[3])
    c = (int(xor_half[1]) * 2) + int(xor_half[2])

    return lookup_table[r][c]
```

In [68]:
```
def round_encrypt(ip, key):

    expanded_per = permute_and_generate(ip, ep_seq)
    expanded_per_xor = find_xor(expanded_per, key)

    left_ep = expanded_per_xor[:4]
    right_ep = expanded_per_xor[4:]

    s0 = find_s0_s1(left_ep, s0_seq)
    s1 = find_s0_s1(right_ep, s1_seq)

    p4 = permute_and_generate(s0 + s1, p4_seq)

    return p4
```

In [69]:
```
def encrypt_decrypt(ip, k1, k2):

    ip_per = permute_and_generate(ip, ip_seq)

    left_ip_per = ip_per[:4]
    right_ip_per = ip_per[4:]

    # round 1
    r1_output = round_encrypt(right_ip_per, k1)
```

```
    r1_output = find_xor(r1_output, left_ip_per)

    # round 2
    r2_output = round_encrypt(r1_output, k2)
    r2_output = find_xor(r2_output, right_ip_per)

    ip_inv = permute_and_generate(r2_output + r1_output, inv_ip_seq)

    return ip_inv
```

In [70]:
```
k1, k2 = generate_keys("1010000010")
```

```
k1 :  10100100
k2 :  01000011
```

In [71]:
```
plaintext = "01100011"

print("Plaintext : ", plaintext)

ciphertext = encrypt_decrypt(plaintext, k1, k2)

print("Ciphertext : ", ciphertext)

decryptedtext = encrypt_decrypt(ciphertext, k2, k1)

print("Decrypted text : ", decryptedtext)
```

```
Plaintext :  01100011
Ciphertext :  11101000
Decrypted text :  01100011
```

In [ ]:
```
```