SUB: Information Security

AY 2023-24 (Semester-V)

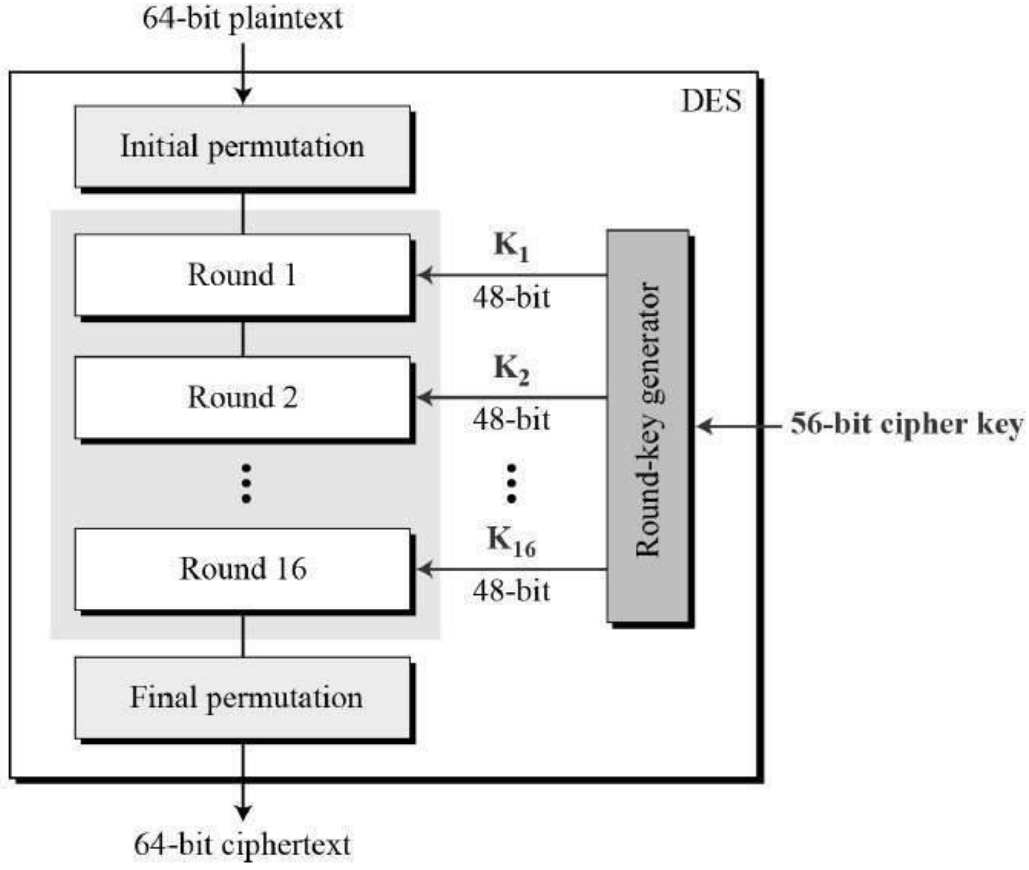
Experiment No: 6

Name: Bhuvi Ghosh
SAPID:60009210191

Aim: To implement Simplified DES.

Theory:

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −



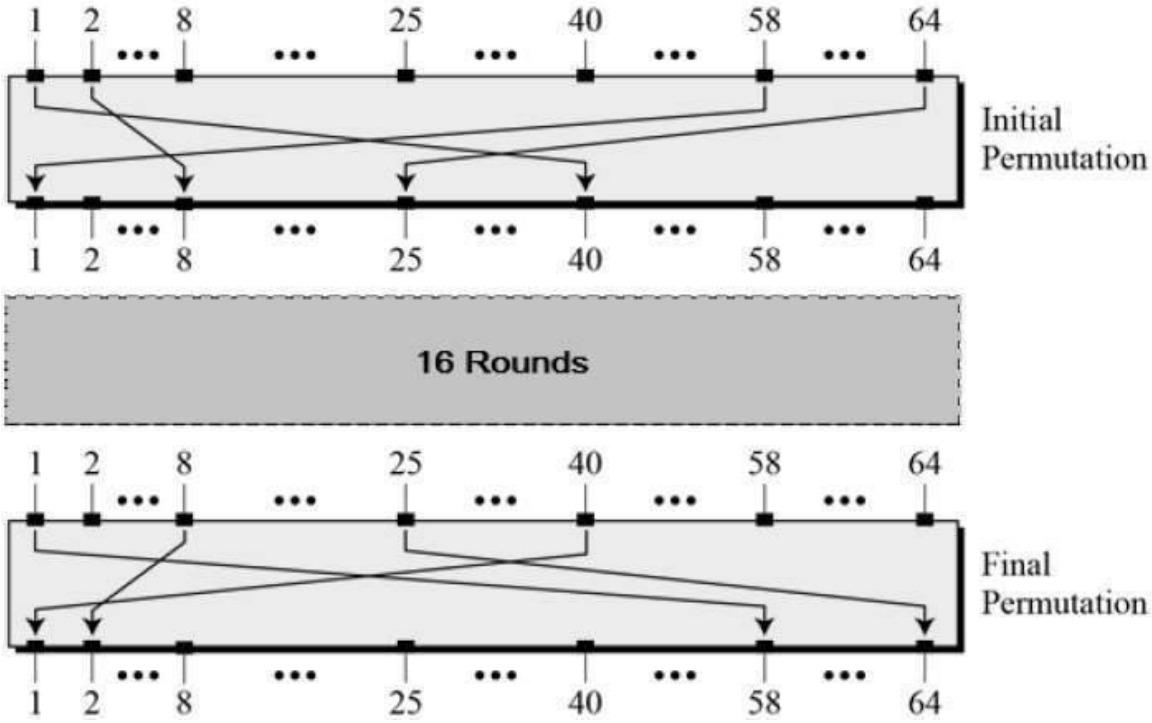
SUB: Information Security

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

Initial and Final Permutation

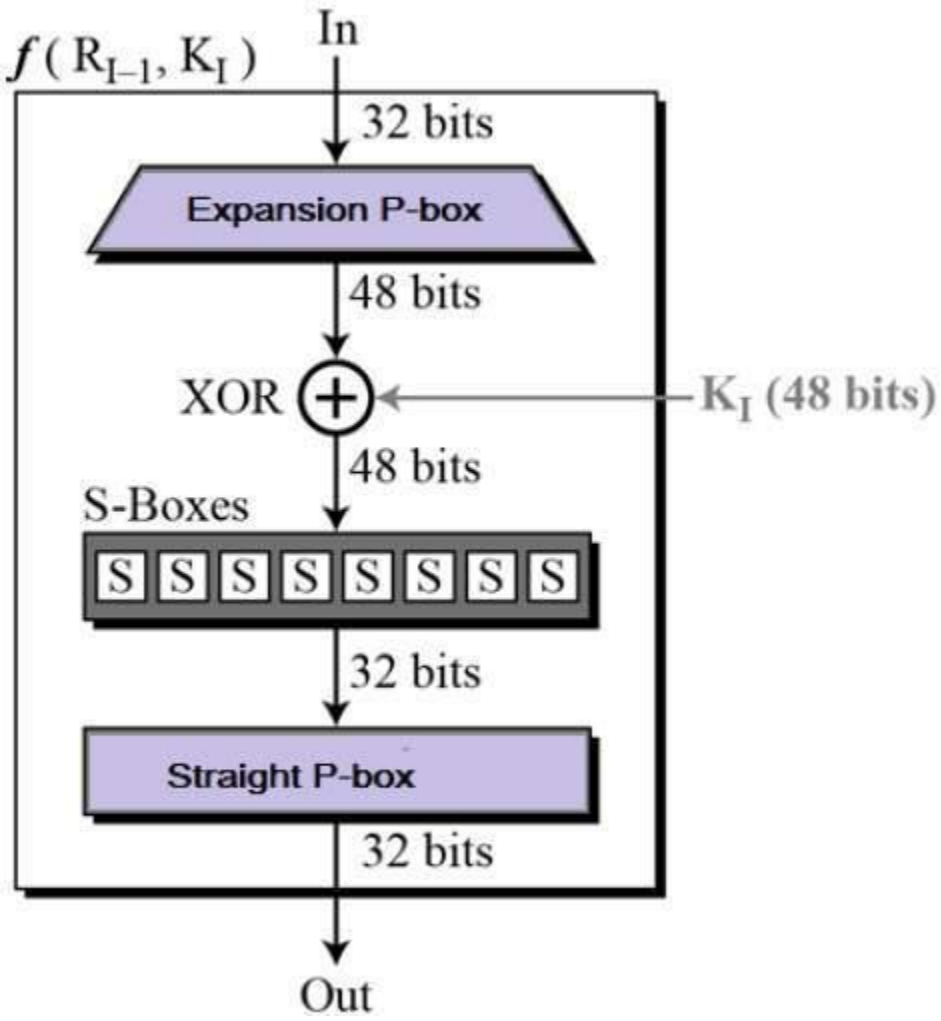
The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −



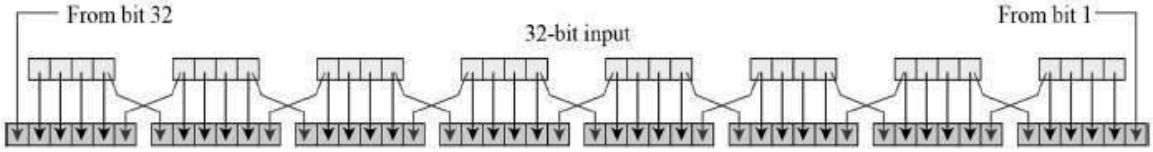
Round Function

The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

SUB: Information Security



- Expansion Permutation Box − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −

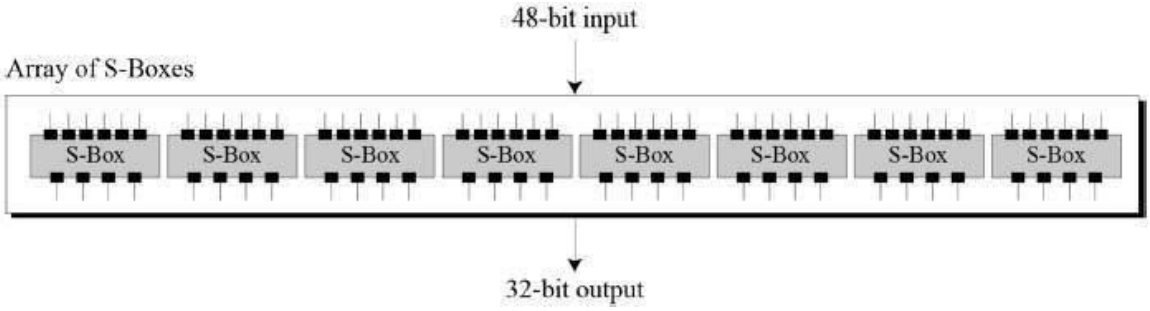


- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

SUB: Information Security

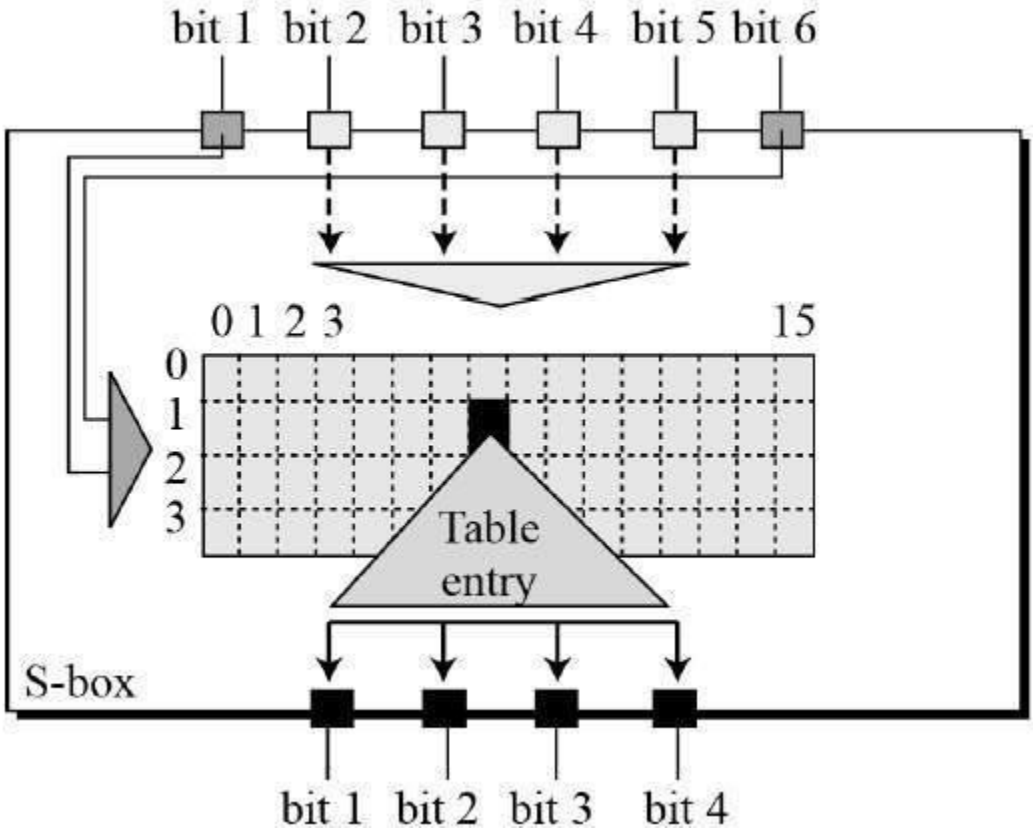
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- XOR (Whitener). − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- Substitution Boxes. − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- The S-box rule is illustrated below −

SUB: Information Security



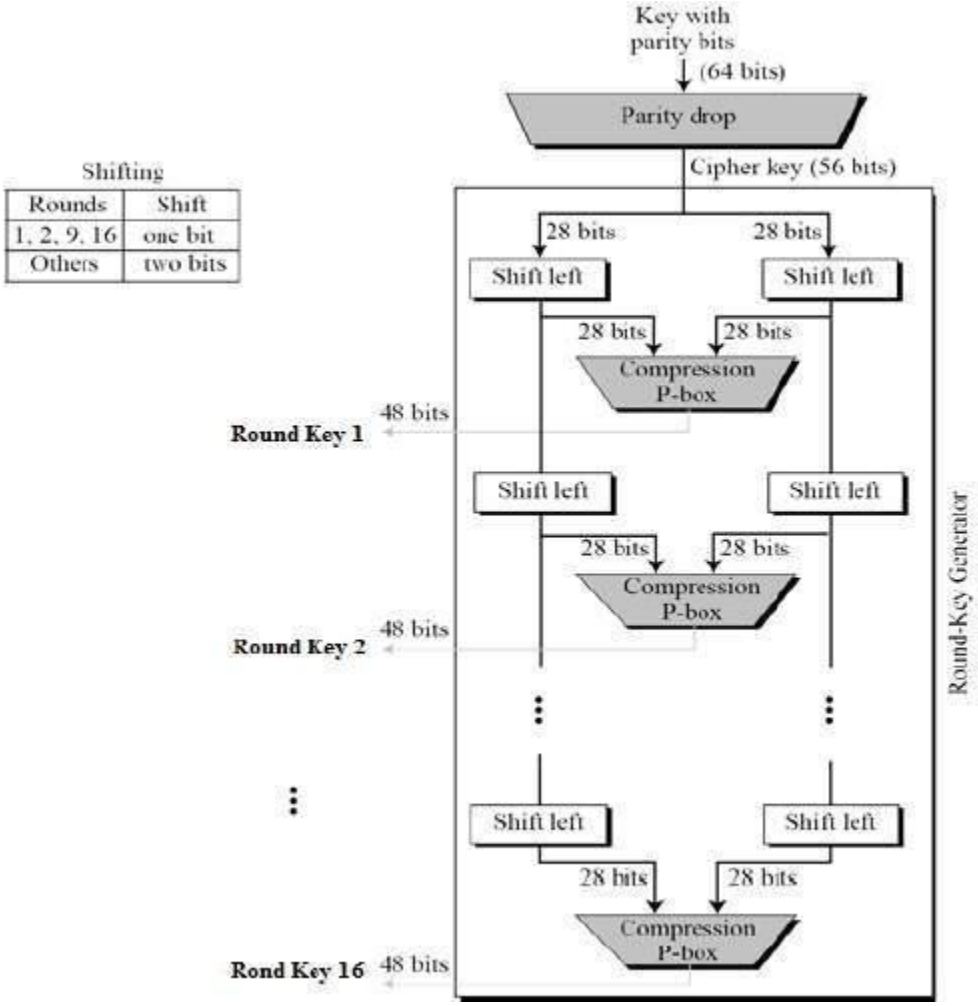
- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- Straight Permutation − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −

SUB: Information Security



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- Avalanche effect − A small change in plaintext results in the very great change in the ciphertext.
- Completeness − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

SUB: Information Security

```
def blocksof4(text):
    blocks = []
    while(text!=''):
        blocks.append(text[:4])
        text = text[4:]

    return blocks

def expansionPermutation(RPT, KEY):
    blocksRPT = blocksof4(RPT)
    n = len(blocksRPT)

    expandedRPT = []
    for inx, word in enumerate(blocksRPT):
        if inx==0:
            temp = blocksRPT[-1][-1] + word + blocksRPT[1][0]
        elif inx==(n-1):
            temp = blocksRPT[-2][-1] + word + blocksRPT[0][0]
        else:
            temp = blocksRPT[inx-1][-1] + word + blocksRPT[inx+1][0]

        expandedRPT.append(temp)

    expandedRPT_str = ''.join(expandedRPT)

    XORed = ''
    for pt, k in zip(expandedRPT_str, KEY):
        XORed += str(int(pt)^int(k))

    blocksof6 = []
    while XORed!='':
        blocksof6.append(XORed[:6])
        XORed = XORed[6:]
```

SUB: Information Security

```
    sbox = []
    for i in blocksof6:
        temp = i[1:5]
        sbox.append(temp)

    sbox_str = ''.join(sbox)

    return sbox_str

def encryption(PT, KEY1, KEY2):
    n = len(PT)
    n = n//2
    lpt = pt[:n]
    rpt = pt[n:]

    for i in range(2):
        print('\n\nLPT :', lpt)
        print('RPT :', rpt)
        if(i==0):
            sbox = expansionPermutation(rpt, KEY1)
        else:
            sbox = expansionPermutation(rpt, KEY2)

        xor = ''
        for l, k in zip(lpt, sbox):
            xor += str(int(l)^int(k))

        lpt = xor
        lpt, rpt = rpt, lpt


    return lpt + rpt

pt = '0100110010100001101101111110111000000110010111110100010001010010'
print(pt)
```

```
key1 = '00111111101011011100001111110001011011001111010 0'
key2 = '11000000010100100011110000001110100100110000101 1'
print(key1)
print(key2)

print("\nPLAIN TEXT:", pt)
ct = encryption(pt, key1, key2)
print("\n\nCIPHERED TEXT:", ct)
```

```
0100110010100001101101111110111000000110010111110100010001010010
0011111110101101110000111111000101101100111101 00
1100000001010010001111100000011101001001100001011

PLAIN TEXT: 0100110010100001101101111110111000000110010111110100010001010010


LPT : 01001100101000011011011111101110
RPT : 00000110010111111010001000101 0010


LPT : 00000110010111111010001000101 0010
RPT : 00110111010011110001100000100110


CIPHERED TEXT: 00110111010011110001100000100110101100110101111001001000000010001
```

Conclusion:DES for symmetric key cryptography has been successfully implemented in Python.