SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Pragati Shankar	USN: 1SI19CS090	Batch No: B2	Date: 06-12-2022

Evaluation:

Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	H K Vedamurthy	
2.	Gururaj S P	

Question No: 6

i) Given 64-bit output of (i-1)th round of DES, 48-bit ith round key K_i and E table, find the 48-bit input for S-box.

ii) Given 48-bit input to S-box and permutation table P, find the 32-bit output R_i of ith round of DES algorithm.

i) **Algorithm:** Follow the flow-chart and tables given below.

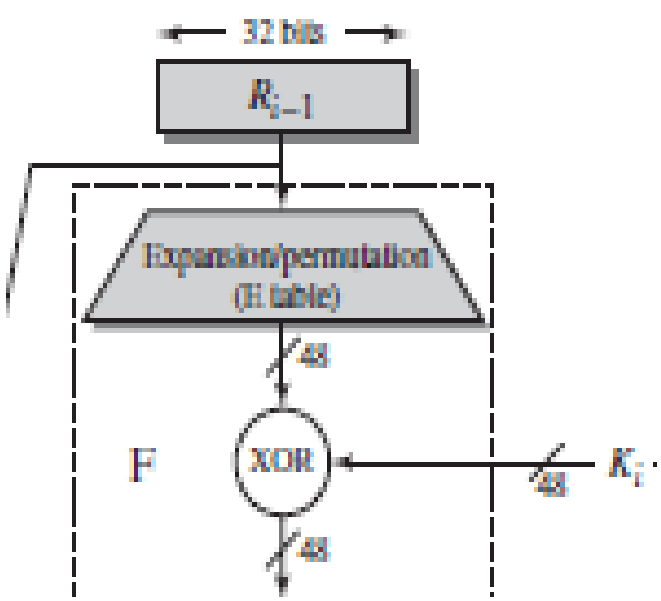

Figure: Generation of 48-bit input for S-box.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

Table: Expansion Permutation

ii) **Algorithm:** The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

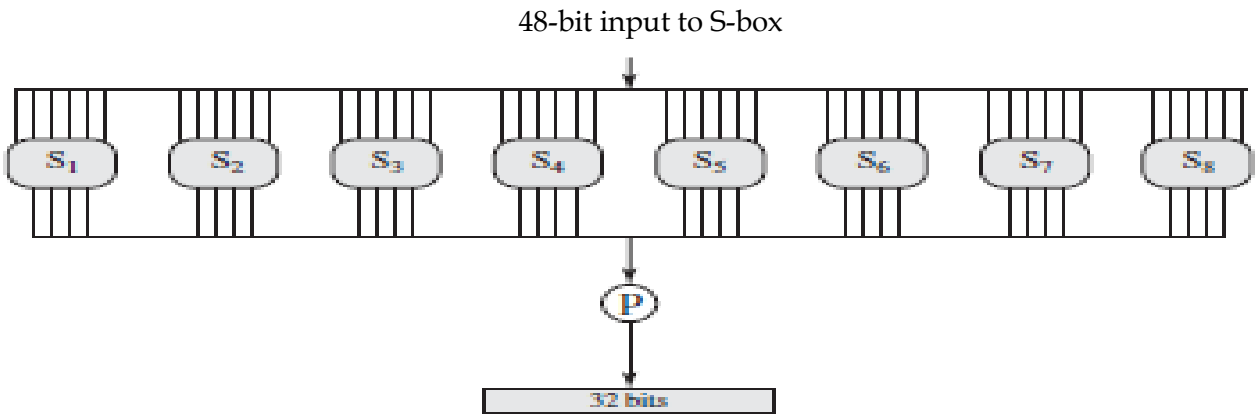

Figure: The 32-bit output R_i of ith round, given 48-bit input

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Table: Permutation Function (P)

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

CODE:

```
#include <bits/stdc++.h>

using namespace std;

int expPermute[] = {
                        32, 1 , 2 , 3 , 4 , 5 ,
                        4 , 5 , 6 , 7 , 8 , 9 ,
                        8 , 9 , 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, 30, 31, 32, 1 };

string expansionPermute(string input)
{

        string res = "";
        for(int i=0; i<48; i++)
        {
                res += input[expPermute[i]-1];
        }
        return res;
}
string XOR(string input1, string input2)
{
        string res = "";
        for(int i=0; i<input1.length(); i++)
        {
                res += (input1[i] == input2[i]) ? "0" : "1";
        }
        return res;
}
int main()
{
        int i; // round i
        unsigned long long hexInput;
        string Ki; // ith round key
        ifstream fin;
        cout << "\nEnter Round number (i) : ";
        cin >> i;
        cout << "Enter 64-bit (i-1)th round output in hex: " ;
        cin >> hex >> hexInput;
        string input = bitset<64>(hexInput).to_string();
        fin.open("keygen.txt");
        for(int j=1; j<=i; j++)
        {
                fin >> Ki;
```

```
        }
        if(Ki.length() == 0)
        {
                cout << "\nkeygen.txt not found !!! \n" << endl;
                exit(1);
        }
        cout << "\n64-bit Binary Input = " << input << endl ;
        cout << "key for ith round (Ki) = " << Ki << endl ;
        string Ri_1 = input.substr(32,32); // 32 bit Right half of input R[i-1]
        cout << "\nRight half of 64-bit input, Ri_1 = " << Ri_1 << endl;
        string R48 = expansionPermute(Ri_1);
        cout << "Ri_1 after expansion permutation = " << R48 << endl;
        string sBoxInput = XOR(R48, Ki);
        cout << "\nInput to s-box : " << sBoxInput << endl << endl;
}
```

Output Screenshots:

keygen.txt
~/1SI19CS090

```
1 1100101101100010101011001100011010000001101000 11
2 0011010110100100101011110010010111100001100110 10
3 1111001100001100100000100110010100010100010000 11
4 0111100010101010101101001100111010000000011011 10
5 1001010010110100000111100000010011011111110011 00
6 0110011000000110011101100001100010010100111100 01
7 1110111011011000001001001100101111001100001000 01
8 1000101010100011011110100000101001101111000110 00
9 1110001111000010110101111001010000011100001100 11
10 0011110111010011100000101000111100001010011101 00
11 0011001000010001111110110001000111101011110100 00
12 1011110101000000010101010011000110000100000101 01
13 0000011101001011100111001100101100100100100001 10
14 0001111000110001101101010010110001100011100011 01
15 1001111100001100011010010011001001010000110001 11
16 0100100110101001100110110110101000101001010101 10
```

Plain Text Tab Width: 8 Ln 14, Col 49 INS

user@linux-OptiPlex-5090: ~/1SI19CS090

```
user@linux-OptiPlex-5090:~/1SI19CS090$ gedit des2.cpp
user@linux-OptiPlex-5090:~/1SI19CS090$ g++ des2.cpp
user@linux-OptiPlex-5090:~/1SI19CS090$ ./a.out

Enter Round number (i) : 1
Enter 64-bit (i-1)th round output in hex: cc00ccfff0aaf0aa

64-bit Binary Input = 1100110000000000110011001111111111110000101010101111000010101010
key for ith round (Ki) = 1100101101100010101011001100011010000001101000 11

Right half of 64-bit input, Ri_1 = 11110000101010101111000010101010
Ri_1 after expansion permutation = 011110100001010101010101011110100001010101010101

Input to s-box : 101100010111011111111001101111001001010011110110

user@linux-OptiPlex-5090:~/1SI19CS090$
```