Цель работы:

Изучить основные принципы работы алгоритмы DES.

Задачи работы:

1. Выполнение 1 цикла раундовой функции алгоритма DES вручную (т.е. выполнение всех функций, входящих в раундовую функцию DES, для фиксированного входного двоичного вектора с отображением промежуточных значений шифрования) или программная реализация 1 раунда (или полной системы) DES.
2. Анализ визуализации алгоритма DES и примитивных атак на шифр, используя Cryptool 2.

Ход работы:

Часть 1.

Демонстрация работы программной реализации полного процесса шифрования для 1 раунда DES с генерацией 64 битного блока, начальной перестановкой IP, выполнением 1 раунда функции расширения ключа.

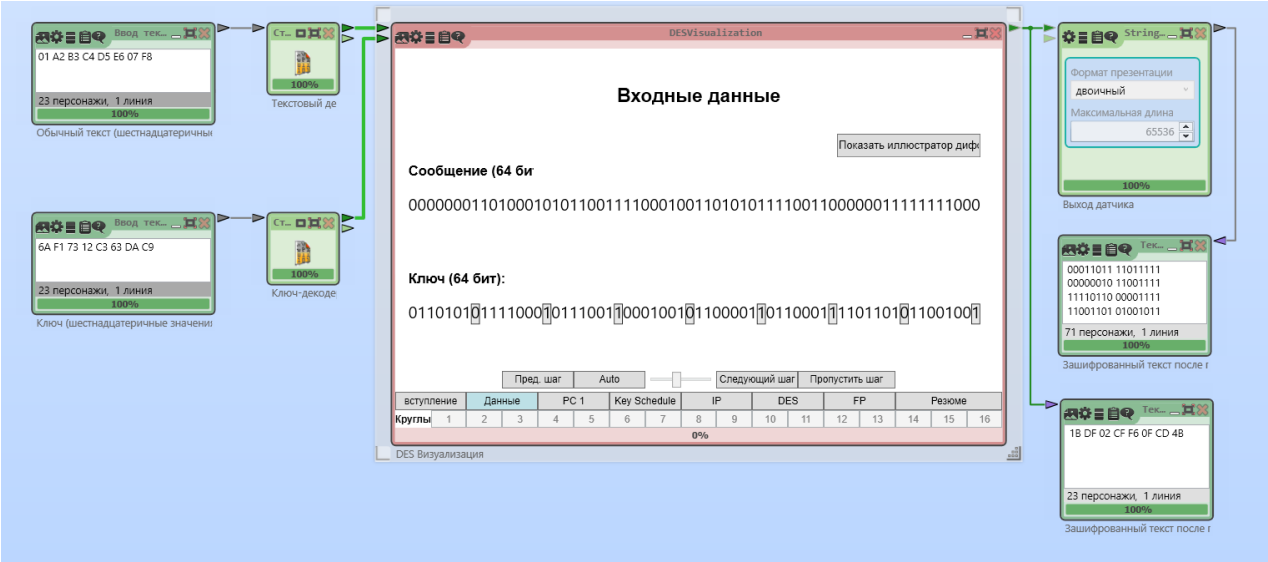
Алгоритм работы программы:

1. Генерирование 64-бит сообщения и 56-бит ключа.
2. Начальная перестановка IP.
3. Разделение сообщения на 2 части L0 и R0.
4. Расширение ключа до 64-бит путем добавления бит в полиции каратные 8.
5. Формирование блоков C0 и D0 из 64-бит ключа.
6. Циклический левый сдвиг блоков на 1 бит (для 1 раунда).
7. Формирование 48-бит ключа путем выборки из вектора C1D1.
8. Расширение “E” сообщения R0 до 48-бит сообщения E.
9. Сложение E с ключом по модулю 2.
10. Операция преобразования “S” при помощи S-box.
11. Перестановка P.
12. Получение сообщений L1 и R1 для 2 раунда DES.

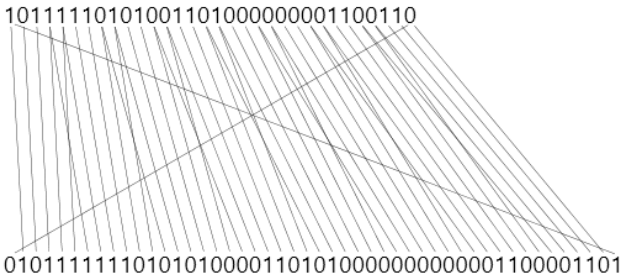
Демонстрация работы программного модуля:

```
Сообщение T (64 бит)     0000000110100010101100111100010011010101111001100000011111111000
Ключ k (56 бит)          01101011111000011100100010011100001011000111011011100100

Сообщение IP (64 бит)    10111000100101000111100001010101101111101010011010000000001100110
Сообщение L0 (32 бит)    10111000100101000111100001010101
Сообщение R0 (32 бит)    10111110101001101000000001100110

Ключ k0 (64 бит)         01101010111100000111001000010010110000100110001011011010110001000
Блок C0 (28 бит)         1101001011110111001001110100
Блок D0 (28 бит)         0111110100000000110000011110
Блок C1 (28 бит)         1010010111101110010011101001
Блок D1 (28 бит)         1111101000000001100000111100
Ключ k1 (48 бит)         101010111111100001000010011000101110100001000011

Расширение E (48 бит)    010111111101010100001101010000000000001100001101
XOR с ключом (48 бит)    ['111101', '000010', '110101', '001111', '001000', '101110', '101101', '001110']
Операция S-box (32 бит)  ['0110', '0001', '1110', '0011', '0111', '0011', '1010', '0001']
Перестановка P (32 бит)  10100000011001011110111110000101

Сообщение L1 (32 бит)    10111110101001101000000001100110
Сообщение R1 (32 бит)    00011000111100011001011111010000
```

Часть 2. Визуализация 1 раунда алгоритма DES.



Expansion



Побитовая операция XOR

K1: 101010111111100001000010011000101110100001000011

⊕

Exp: 010111111101010100001101010000000000001100001101

11110100001011010100111100100010111010110100111 0

S-коробки

111101 000010 110101 001111 001000 101110 101101 001110

S8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Входные данные: 001110

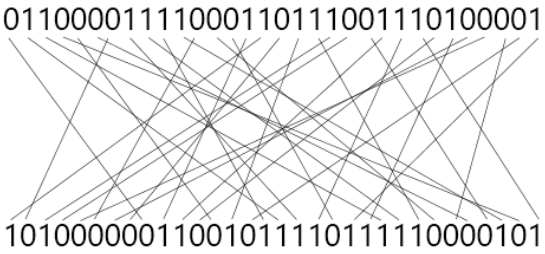
Ряд: 0____0 ≙ 0

Колонка: _0111_ ≙ 7

Вывод: 0001 ≙ 1

Вывод всех S-блоков: 01100001111000110111001110100001

Функция перестановки

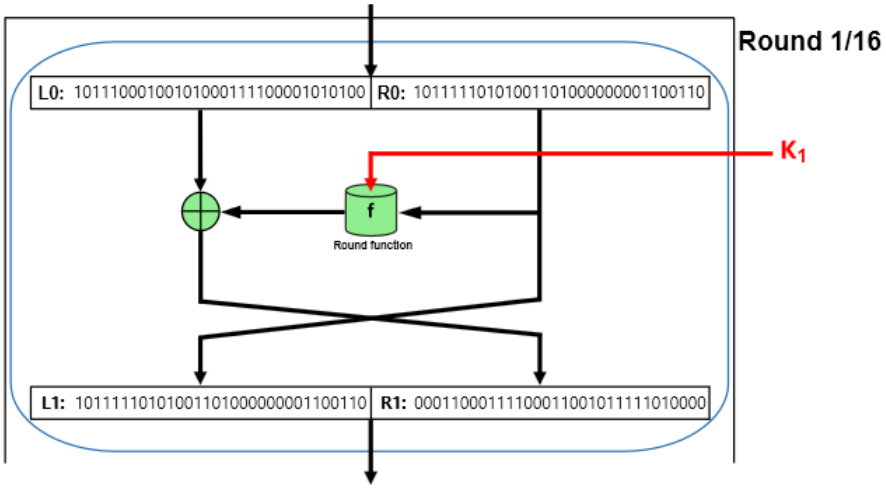


Bitwise XOR Operation

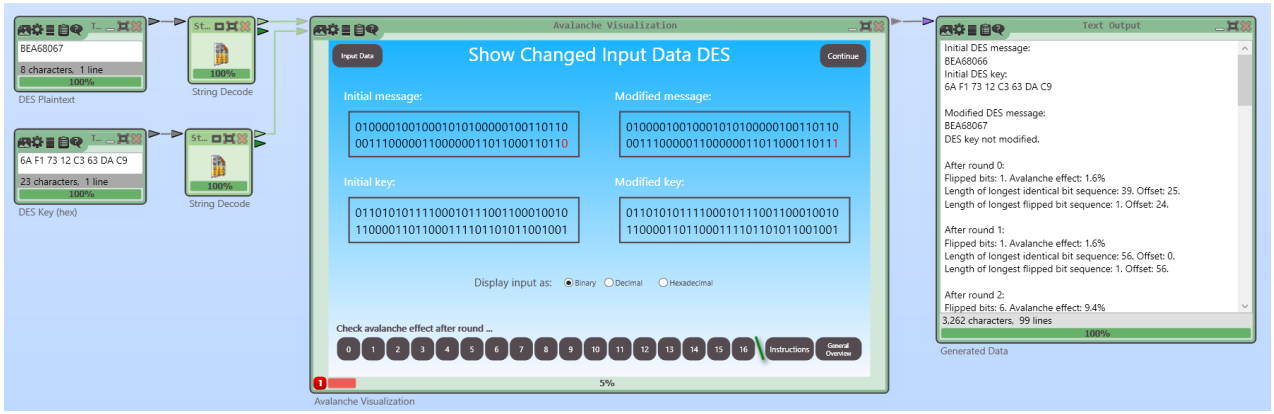
L0: 10111000100101000111100001010100

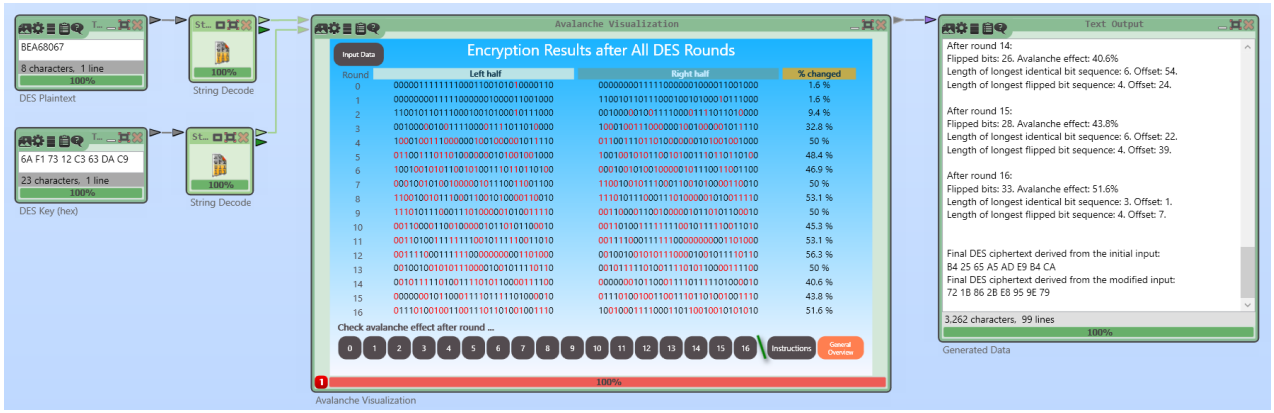
⊕

f(R0): 10100000011001011110111110000101

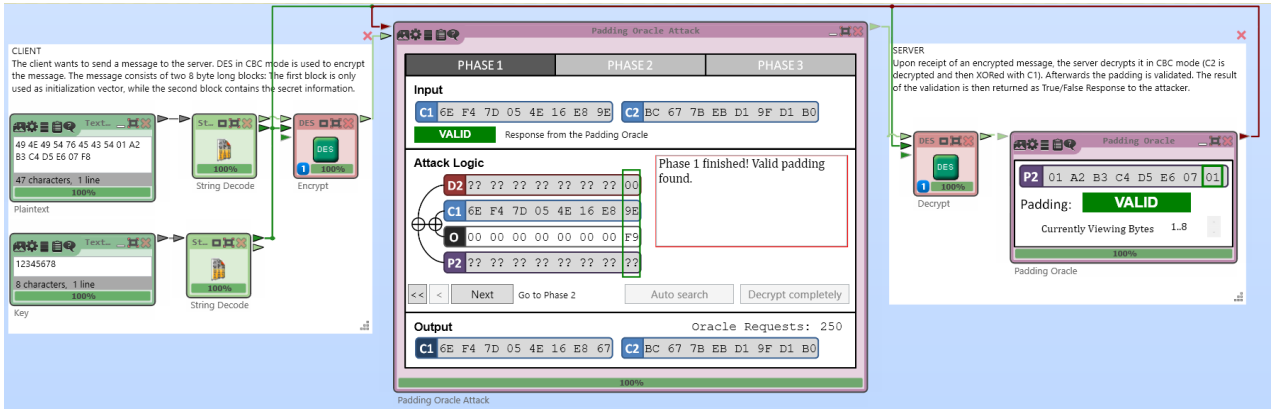


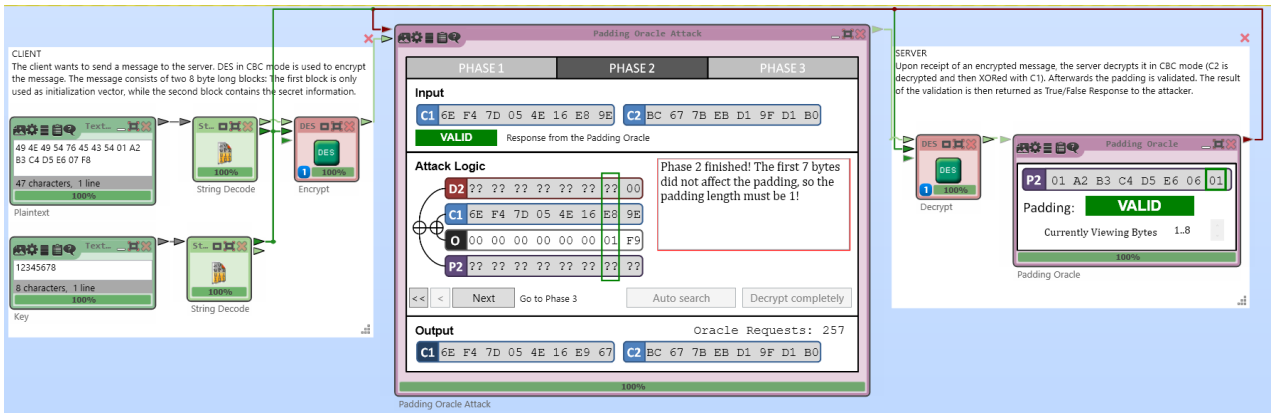
Часть 2. Визуализация «лавинного эффекта» DES.

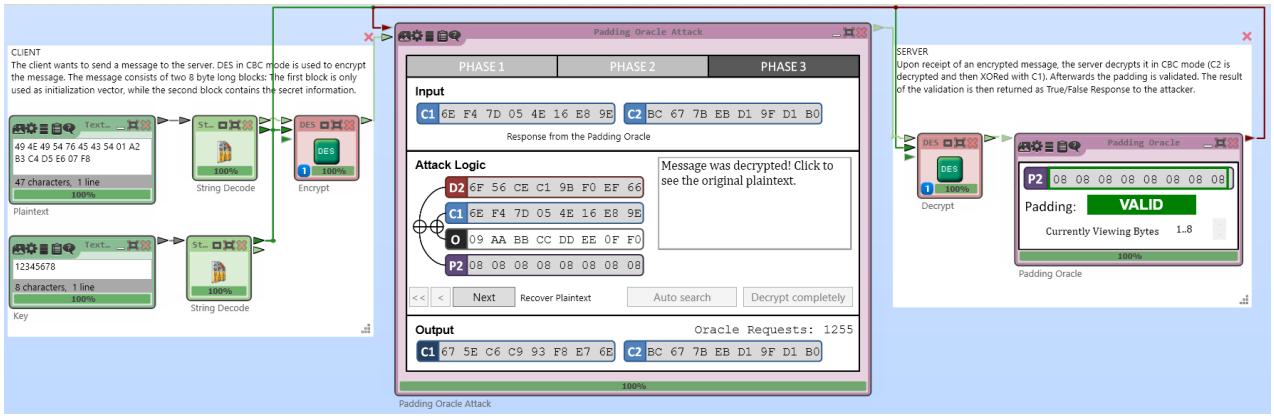


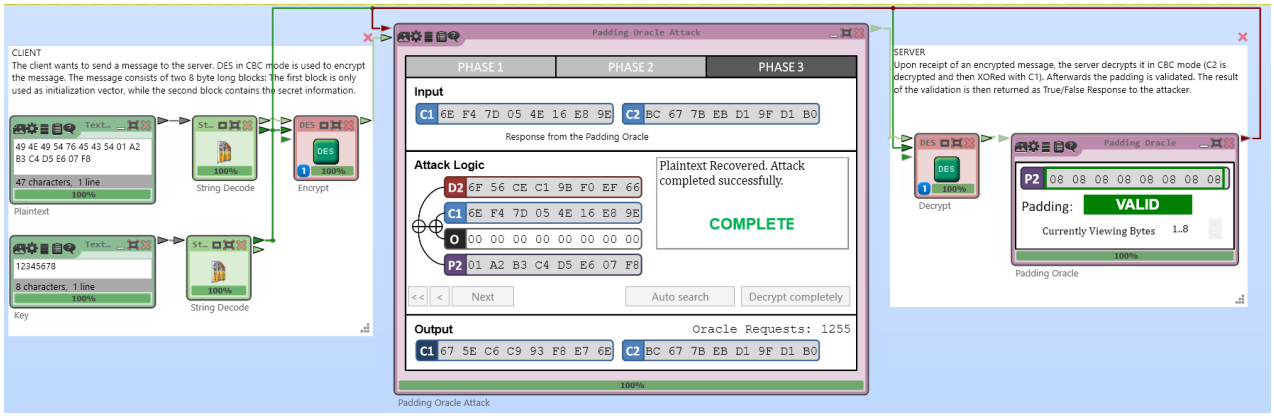


Часть 2. Проведение Padding Oracle атаки на DES.









Вывод:

В результате данной лабораторной работы были изучены основные принципы работы алгоритмы DES, программно реализован полный процесса шифрования для 1 раунда с генерацией 64 битного блока, начальной перестановкой IP, выполнением 1 раунда функции расширения ключа. Был выполнен анализ визуализации алгоритма DES, продемонстрирован лавинный эффект, проведена padding oracle атака на DES.

Приложение 1. Листинг программного кода.

```
import random

# msg_T = '0000000110100010101100111100010011010101111001100000011111111000'
# key56 = '01101011111000011100100010011100001011000111011011100100'
msg_T = key56 = ''
for i in range(64):
        msg_T += str(round(random.random()))
for i in range(56):
        key56 += str(round(random.random()))
print('Сообщение T (64 бит)    ', msg_T)
print('Ключ k (56 бит)                              ', key56, '\n')

# Начальная перестановка
msg_IP = ''
msg_IP_list =
(58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,62,54,46,38,30,22,14,6,64,56,48,40,32,24,16,8,57,49,41,33,
25,17,9,1,59,51,43,35,27,19,11,3,61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7)
for i in range(len(msg_IP_list)):
        msg_IP += msg_T[msg_IP_list[i]-1]
print('Сообщение IP (64 бит)   ', msg_IP)

# Разделение на L0 и R0
L_list = []
R_list = []
L_list.append(msg_IP[:32])
R_list.append(msg_IP[32:])
print('Сообщение L0 (32 бит)   ', L_list[0])
print('Сообщение R0 (32 бит)   ', R_list[0], '\n')

# Расширение ключа
key64 = ''
for i in range(0, len(key56), 7):
        for n in range(7):
                key64 += key56[i+n]
        key64 += '0'
print('Ключ k0 (64 бит)                ', key64)

# Формирование блоков C0 и D0
C_list = (57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36)
D_list = (63,55,47,39,31,23,15,7,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,28,20,12,4)
msg_C = ''
msg_D = ''
for i in range(len(C_list)):
        msg_C += key64[C_list[i]-1]
        msg_D += key64[D_list[i]-1]
print('Блок C0 (28 бит)                ', msg_C)
print('Блок D0 (28 бит)                ', msg_D)

# Циклический сдвиг
msg_C += msg_C[0]
msg_C = msg_C[1:]
msg_D += msg_D[0]
msg_D = msg_D[1:]
msg_CD = msg_C + msg_D
print('Блок C1 (28 бит)                ', msg_C)
print('Блок D1 (28 бит)                ', msg_D)

# Получение ключа 48 бит
key = ''
key_list =
(14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51,45,33,48,44,4
9,39,56,34,53,46,42,50,36,29,32)
for i in range(len(key_list)):
        key += msg_CD[key_list[i]-1]
print('Ключ k1 (48 бит)                ', key, '\n')

# Расширение E
msg_E = ''
expansion_E_list =
(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)
for i in range(len(expansion_E_list)):
        msg_E += R_list[0][expansion_E_list[i]-1]
print('Расширение E (48 бит)   ', msg_E)

# Сложение с ключом
msg_B = []
msg_temp = ''
for i in range(len(msg_E)):
        msg_temp += str((int(msg_E[i])+int(key[i]))%2)
        if (i+1)%6 == 0:
                msg_B.append(msg_temp)
                msg_temp = ''
print('XOR с ключом (48 бит)   ', msg_B)

# Преобразование (Операция подстановки S-box)
s1 = ((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))
s2 = ((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))
s3 = ((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))
s4 = ((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))
s5 = ((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))
s6 = ((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))
s7 = ((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))
s8 = ((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))
a = b = 0
msg_Bs = []
s_list = (s1, s2, s3, s4, s5, s6, s7, s8)
for i in range(len(msg_B)):
            a = int(str(msg_B[i][0])+str(msg_B[i][5]), 2)
            b = int(str(msg_B[i][1])+str(msg_B[i][2])+str(msg_B[i][3])+str(msg_B[i][4]), 2)
            msg_Bs.append(format(s_list[i][a][b], '04b'))
print('Операция S-box (32 бит) ', msg_Bs)
msg_temp = msg_Bs
msg_Bs = ''
for elem in msg_temp:
            msg_Bs += elem

# Перестановка P
msg_P = ''
permutation_P_list =
(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)
for i in range(len(permutation_P_list)):
            msg_P += msg_Bs[permutation_P_list[i]-1]
print('Перестановка P (32 бит) ', msg_P, '\n')

# Получение L1 и R1
L_list.append(R_list[0])
msg_temp = ''
for i in range(len(R_list[0])):
            msg_temp += str((int(L_list[0][i])+int(msg_P[i]))%2)
R_list.append(msg_temp)
print('Сообщение L1 (32 бит)   ', L_list[1])
print('Сообщение R1 (32 бит)   ', R_list[1])
```