

密码学综合设计实验

实验 2：DES 加密解密算法实现

学号：031803108

姓名：苏煜程

2019 年 9 月 29 日

一、 实验要求

1. 实现 Feistel 密码结构 (64bit 分组长度)

- a) 输入 64bit 明文分组, 轮数 Round, 轮函数 F, 子密钥数组 K
- b) 输出 64bit 密文分组
- c) 提示: 如果是 C 语言实现的话, 轮函数用函数指针

2. DES 算法实现

- d) 初始置换实现
- e) 子密钥生成实现
- f) DES 轮函数实现
- g) 逆初始置换实现
- h) 加密分组实现
- i) 解密分组实现

3. 附加内容:

- j) 自定义轮函数, 实现一个基于 Feistel 结构的加密解密自定义算法。
- k) 比如自定义轮函数: $F(W, K) = (1 \ll W) + K$, 即 W 先循环左移 1 位再与 K 异或。
- l) 用电码本模式对文件进行加密和解密。

二、 实验原理

1. 所需参数

key: 8 个字节共 64 位的工作密钥

data: 8 个字节共 64 位的需要被加密或被解密的数据

mode: DES 工作方式, 加密或者解密

2. 初始置换

DES 算法使用 64 位的密钥 key 将 64 位的明文输入块变为 64 位的密文输出块, 并把输出块分为 L0、R0 两部分, 每部分均为 32 位。初始置换规则如下:

```
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
```

3. 轮结构

将 64 比特的轮输入分为 32 比特的左、右两半，分别记为 L 和 R。和 Feistel 网络一样，每轮变换可由以下公式表示：

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

K_i 是向第 N 层输入的 48 位的密钥，F 是以 R_{i-1} 和 K_i 为变量的输出 32 位的函数。

4. 函数 F

A. 扩展置换 E

通过扩展置换 E，数据的右半部分 R 从 32 位扩展到 48 位。
扩展置换 E 规则如下：

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,
28,29,30,31,32,01

B. S-盒代替

R 扩展置换之后与子密钥 K_i 异或以后的结果作为输入块进行 S 盒代替运算，功能是把 48 比特数据变为 32 比特。然后再通过一个 S 盒，产生 32 比特的输出。

代替运算由 8 个不同的代替盒 (S 盒) 完成。每个 S 盒有 6 位输入，4 位输出。

C. P-盒置换

S-盒代替运算，每一盒得到 4 位，8 盒共得到 32 位输出。这 32 位输出作为 P 盒置换的输入块。

P 盒定义如下：

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

5. 子密钥的产生

DES 算法由 64 位密钥产生 16 轮的 48 位子密钥。在每一轮的迭代过程中，使用不同的子密钥。

A. 置换选择 1

将 64 位密钥通过缩小选择置换表（PC-1）的变换变成 56 位，然后将置换后的 56 位密钥分为 C_0, D_0 两半。PC-1 的定义如下：

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,
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

B. 置换选择 2

在第 i 轮分别对 C_{i-1} 和 D_{i-1} 进行左循环移位，循环左移每轮移动的位数如下：

每轮移动的位数表

轮	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

移位后的结果作为下一轮求子密钥的输入，同时也作为置换选择 2 的输入。通过置换选择 2 产生的 48 比特的 K_i ，即为本轮的子密钥，作为函数 F 的输入。其中置换选择 2 的定义如下：

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, 49, 39, 56, 34, 53,
46, 42, 50, 36, 29, 32

6. 解密

和 Feistel 密码一样，DES 的解密和加密使用同一算法，但子密钥使用的顺序相反。

软件系统设计

实现Feistel密码结构（64bit分组长度）

单独通过一个Feistel函数抽象了Feistel结构

```
def Feistel(ClearTxt,Key,Model):
    #Step 1 is CreateKey
    keylist = createkey(Key)
    print()
    if Model == 1:
        keylist = keylist
    if Model == 2:
        keylist = keylist[::-1] #INVERSEKEY
    #Step 2 is ClearTxt
    text = DES_ECB(ClearTxt,keylist)
    return text
```

通过不同的加密轮函数和密钥生成结构就可以组合成不同的Feistel密码，通过不同的Key生成函数可以生成不同的子密钥数组

DES算法实现

初始置换实现

Java实现：

```
public int[] IP(int[] BinaryClearTxt){
    int[] IPTABLE = {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};
    if(BinaryClearTxt.length != 64){
        System.out.println("Your key Text lenth is Error!");
    }
    else {
        for (int i = 0;i < IPTABLE.length;i++){
            ret[i] = BinaryClearTxt[IPTABLE[i]-1];
        }
        System.out.print("Source ClearText : ");
        for(int i = 0;i<BinaryClearTxt.length;i++){
            System.out.print(BinaryClearTxt[i]);
        }
        return ret;
    }
}
```

Python实现：

```

def IP(ClearTxt):
    IPTABLE = [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]
    if len(ClearTxt) != 64:
        print("Your key Text lenth is Error!")
        assert len(ClearTxt) == 64 # if no 64bit error
    else:
        ret = ""
        for i in IPTABLE:
            ret = ret + ClearTxt[i - 1]
        print("Source ClearText : ",ClearTxt)
        print("IP Replace : ",ret)
        return ret

```

子密钥生成实现

Java实现

```

class Key{
    int[] Movetimes = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
    int[] PC_1_LTABLE = {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};
    int[] PC_1_RTABLE = {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};
    int[] PC_2_TABLE = {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, 49, 39, 56, 34, 53,
                        46, 42, 50, 36, 29, 32};
    public int[] Binarykey = new int[64];
    public int[][] Sonkey = new int[16][48];
    public int len = 0;
    public void set()//(test pass) set Binarykey
    {
        Scanner scan = new Scanner(System.in);
        int[] tmp = new int[64];
        System.out.println("Please input your Key : ");
        String tmpchar = scan.next();
        HextoBin hexto = new HextoBin();
        hexto.set(tmpchar);
        String str2 = hexto.out();
        for(int i = 0;i < 64;i++){
            Binarykey[i] = str2.charAt(i)-'0';
        }
    }
}

```

```

    }
    public int[] result(int[] txt,int count){
        int lenth = txt.length;
        int[] tmptxt = new int[lenth];
        if(lenth > 28){
            System.out.println("Your Result length is Error");
        }
        else{
            int tmpnum = 0;
            for(int i = count;i < lenth ;i++){
                tmptxt[tmpnum] = txt[i];
                tmpnum ++;
            }
            for(int i = 0;i < count;i++){
                tmptxt[tmpnum+i] = txt[i];
            }
        }
        return tmptxt;
    }
    public int[][] Main(){
        int lenth = Binarykey.length;
        if(lenth != 64){
            System.out.println("Your Key lenth is Error");
        }
        String BinarykeyST = "";
        for(int i =0;i < lenth;i++){
            BinarykeyST += Binarykey[i];
        }
        System.out.println("Your Binary Key is : "+BinarykeyST);
        int[] L0 = new int[28];
        int[] R0 = new int[28];
        for(int i = 0;i < PC_1_LTABLE.length;i++){
            L0[i] = Binarykey[PC_1_LTABLE[i] - 1];
        }
        for(int i = 0;i < PC_1_RTABLE.length;i++){
            R0[i] = Binarykey[PC_1_RTABLE[i] - 1];
        }
        for(int i = 0;i < 16;i++){
            L0 = result(L0,Movetimes[i]);
            R0 = result(R0,Movetimes[i]);
            int[] mergedKey = new int[56];
            for(int j = 0;j < 28;j++){
                mergedKey[j] = L0[j];
            }
            for(int j = 28;j < 56;j++){
                mergedKey[j] = R0[j-28];
            }
            int[] tempkey = new int[48];
            for(int j = 0;j < PC_2_TABLE.length;j++){
                tempkey[j] = mergedKey[PC_2_TABLE[j]-1];
            }
            for(int j = 0;j<48;j++){
                Sonkey[i][j] = tempkey[j];
            }
        }
        return Sonkey;
    }
}

```

Python实现

```
def createkey(key):
    Movetimes = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
    PC_1_LTABLE = [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]
    PC_1_RTABLE = [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]
    PC_2_TABLE = [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, 49, 39, 56, 34, 53,
                  46, 42, 50, 36, 29, 32]

    if len(key) != 64 :
        print("Your Key lenth is Error!")
        assert len(key) == 64#if no 64bit error
    else:
        L0 = ""
        R0 = ""
        for i in PC_1_LTABLE:
            L0 += key[i - 1]
        for i in PC_1_RTABLE:
            R0 += key[i - 1]
        assert len(L0) == 28 #if no 28bit error
        assert len(R0) == 28
        Sonkey = []
        for i in range(0, 16):
            print("Movetimes : ",i)
            L0 = result(L0, Movetimes[i])
            R0 = result(R0, Movetimes[i])
            print("L0 : ", L0)
            print("R0 : ", R0)
            mergedKey = L0 + R0
            tempkey = ""
            for j in PC_2_TABLE:
                tempkey += mergedKey[j - 1]
            assert len(tempkey) == 48
            print("Your NO.",i," Sonkey :",tempkey)
            Sonkey.append(tempkey)
        return Sonkey
```

DES轮函数实现

Java实现

```
class DES{
    private static String intToHex(int n) {
        StringBuffer s = new StringBuffer();
```



```

String a;
char []b =
{'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
while(n != 0){
    s = s.append(b[n%16]);
    n = n/16;
}
a = s.reverse().toString();
return a;
}

private static String hexStr = "0123456789ABCDEF";
private static String[] binaryArray =
{"0000","0001","0010","0011",
    "0100","0101","0110","0111",
    "1000","1001","1010","1011",
    "1100","1101","1110","1111"};
public static String bin2HexStr(byte[] bytes){
    String result = "";
    String hex = "";
    for(int i=0;i<bytes.length;i++){
        //字节高4位
        hex = String.valueOf(hexStr.charAt((bytes[i]&0xF0)>>4));
        //字节低4位
        hex += String.valueOf(hexStr.charAt(bytes[i]&0x0F));
        result +=hex; //+" "
    }
    return result;
}

String ClearTxt = "";
public int[] BinaryClearTxt = new int[64];
public int[][] Sonkey = new int[16][48];
int[] ret = new int[64];
public int[] IP(int[] BinaryClearTxt){
    int[] IPTABLE = {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};
    if(BinaryClearTxt.length != 64){
        System.out.println("Your key Text lenth is Error!");
    }
    else {
        for (int i = 0;i < IPTABLE.length;i++){
            ret[i] = BinaryClearTxt[IPTABLE[i]-1];
        }
        System.out.print("Source ClearText : ");
        for(int i = 0;i<BinaryClearTxt.length;i++){
            System.out.print(BinaryClearTxt[i]);
        }
    }
    return ret;
}

public String expend(String Rn){
    int[] ETABLE = {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 retRn = "";
for(int i =0;i<ETABLE.length;i++){
    retRn += Rn.charAt(ETABLE[i]-1);
}
return retRn;
}

public String S_sub(String S_Input){
    int[][] STABLE = {{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},
    {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},
    {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},
    {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},
    {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},
    {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},
    {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},
    {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}};

String retstr = "";
while (S_Input.length()<48){
    S_Input = "0" + S_Input;
}
int index = 0;
for(int i = 0;i < 8;i++){
    int[] slist = new int[64];
    for(int j = 0;j < 64;j++){
        slist[j] = STABLE[i][j];
    }
    int row = (S_Input.charAt(index)-'0')*2+(S_Input.charAt(index+5)-
'0');

```

```

        int col = (S_Input.charAt(index+1)-'0')*8 +
(S_Input.charAt(index+2)-'0')*4+(S_Input.charAt(index+3)-'0')*2+
(S_Input.charAt(index+4)-'0');

        String ret_single = Integer.toBinaryString(Slist[row*16+col]);
        while (ret_single.length() < 4){
            ret_single = "0" + ret_single;
        }
        retstr += ret_single;
        index += 6;
    }
    if (retstr.length() != 32){
        System.out.println("Your S_sub retstr lenth is erroe!\n");
    }
    return retstr;
}

public String[] P(String Ln,String S_sub_str,String Rn){
    int[] PTABLE = {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};
    String tmp = "";
    for(int i = 0;i < PTABLE.length;i++){
        tmp += S_sub_str.charAt(PTABLE[i]-1);
    }
    String Lnnew = "";
    for(int i = 0;i < tmp.length();i++){
        Lnnew += (tmp.charAt(i) - '0') ^ (Ln.charAt(i) - '0');
    }
    while (Lnnew.length()<32){
        Lnnew = "0" + Lnnew;
    }
    if (Lnnew.length() != 32){
        System.out.println("Your Lnnew lenth is error!\n");
    }
    Ln = Rn;
    Rn = Lnnew;
    String[] strstr = new String[2];
    strstr[0] = Ln;
    strstr[1] = Rn;
    return strstr;
}

public String IP_inverse(String L16,String R16){
    int[] IPINVERSETABLE = {40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15,
55, 23, 63, 31,
        38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29,
        36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27,
        34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25};
    String tmp = L16 + R16;
    String retstr = "";
    for(int i = 0;i < IPINVERSETABLE.length;i++){
        retstr += tmp.charAt(IPINVERSETABLE[i] - 1);
    }
    if (retstr.length() != 64){
        System.out.println("Your IP_inverse is error!\n");
    }
    return retstr;
}

```

```

}
public void Main(){
    Scanner scan = new Scanner(System.in);
    ClearTxt = scan.next();
    HextoBin hexto = new HextoBin();
    hexto.set(ClearTxt);
    String str2 = hexto.out();
    for(int i = 0;i < 64;i++){
        BinaryClearTxt[i] = str2.charAt(i)-'0';
    }
    BinaryClearTxt = IP(BinaryClearTxt);
    int[] tmpLn = new int[32];
    int[] tmpRn = new int[32];
    String Ln = "";
    String Rn = "";

    for(int i = 0;i <32;i++){
        Ln += BinaryClearTxt[i];
    }
    for(int i = 32;i <64;i++){
        Rn += BinaryClearTxt[i];
    }

    Key Key = new Key();
    Key.set();

    Sonkey = Key.Main();

    for(int i = 0;i <16;i++){
        int[] tmpkey = new int[48];
        for(int j = 0;j < 48;j++){
            tmpkey[j] = Sonkey[i][j];
        }
        while (Rn.length() < 32){
            Rn = "0" + Rn;
        }
        while (Ln.length() < 32){
            Ln = "0" + Ln;
        }
        String Rn_expand = expend(Rn);
        String S_input = "";
        String S_sub_str = "";
        for(int j =0;j<48;j++){
            S_input += ((Rn_expand.charAt(j) - '0') ^ (tmpkey[j]));
        }
        S_sub_str = S_sub(S_input);
        String[] StrStr = new String[2];
        StrStr = P(Ln,S_sub_str,Rn);
        Ln = StrStr[0];
        Rn = StrStr[1];
    }
    String tmpp = "";
    tmpp = Ln;
    Ln = Rn;
    Rn = tmpp;
    String re_text = IP_inverse(Ln,Rn);

```

```

        int tmnum = 0;
        int tmindex = 0;
        int tmmindex = 3;
        String a = "";
        for(int i =0;i <64;i++){
            if(tmindex == 4){
                tmindex = 0;
                a += hexStr.charAt(tmnum);
                tmnum = 0;
                tmmindex = 3;
            }
            tmnum += Math.pow(2,tmmindex) * (re_text.charAt(i) - '0');
            tmmindex --;
            tmindex ++;
            if(i == 63){
                a += hexStr.charAt(tmnum);
            }
        }
        System.out.println("Your DES Result is : "+re_text);
        System.out.println("Your DES Result is : "+tmnum);
        System.out.println("Your DES Result is : "+a);
    }
}

```

Python实现

```

def expend(Rn):
    ETABLE = [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]

    retRn = ""
    for i in ETABLE:
        retRn += Rn[i - 1]
    assert len(retRn) == 48
    return retRn

def S_sub(S_Input):
    STABLE = [(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),
               (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),
               (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),
               (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),

```

```

        (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),
        (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),
        (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),
        (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)]

S_Input = bin(S_Input)[2:]
while len(S_Input) < 48:
    S_Input = "0" + S_Input
index = 0
retstr = ""
for Slist in STABLE:
    row = int(S_Input[index] + S_Input[index + 5], base=2)
    col = int(S_Input[index + 1:index + 5], base=2)
    ret_single = bin(Slist[row * 16 + col])[2:]
    while len(ret_single) < 4:
        ret_single = "0" + ret_single
    retstr += ret_single
    index += 6
assert len(retstr) == 32
return retstr
def P(Ln, S_sub_str, oldRn):
    PTABLE = [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]
    tmp = ""
    for i in PTABLE:
        tmp += S_sub_str[i - 1]
    LnNew = int(tmp, base=2) ^ int(Ln, base=2)
    LnNew = bin(LnNew)[2:]
    while len(LnNew) < 32:
        LnNew = "0" + LnNew
    assert len(LnNew) == 32
    (Ln, Rn) = (oldRn, LnNew)
    return (Ln, Rn)
def IP_inverse(L16, R16):
    IPINVERSETABLE = [40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63,
31,
                      38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21,
61, 29,
                      36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19,
59, 27,
                      34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17,
57, 25]
    tmp = L16 + R16
    retstr = ""
    for i in IPINVERSETABLE:
        retstr += tmp[i - 1]
    assert len(retstr) == 64

```

```

    return retstr
def DES_ECB(ClearTxt,Keylist):
    InitKeyCode = IP(ClearTxt)
    Ln = InitKeyCode[0:32]
    Rn = InitKeyCode[32:]
    for key in Keylist:
        while len(Rn) < 32:
            Rn = "0" + Rn
        while len(Ln) < 32:
            Ln = "0" + Ln
        print("Ln : ",Ln)
        print("Rn : ", Rn)
        Rn_expand = expend(Rn)
        print("Rn_expand : ",Rn_expand)
        S_Input = int(Rn_expand, base=2) ^ int(key, base=2)
        print("S_Input : ", S_Input)
        S_sub_str = S_sub(S_Input)
        print("S_sub_str : ", S_sub_str)
        (Ln, Rn) = P(Ln, S_sub_str, Rn)
    (Ln, Rn) = (Rn, Ln)
    re_text = IP_inverse(Ln, Rn)
    return re_text

```

加密分组实现

```

def Encryptmohex(tmplist):
    str_bin = ''.join(tmplist)
    groups = int(len(str_bin) / 64)
    # 生成加密分组
    M = np.zeros((groups, 64))
    index = -8
    for i in range(groups):
        index += 8
        strr = ""
        for j in range(8):
            strr += tmplist[index + j]
        for j in range(64):
            M[i][j] = int(strr[j])
    print("Your List Clear Text: ", tmplist)
    print("Your Binary Clear Text: ", str_bin)
    print("Your Hex Clear Text: ", hex(int(str_bin, base=2)).upper())
    print("Your Clear Text groups: ", groups)
    key_bin = inkey()
    # 打印加密群
    AllCiphertext = ""
    for i in range(groups):
        print("ClearText Group ", i, ":", end='')
        tmptext = ""
        for j in range(64):
            tmptext += str(int(M[i][j]))
        ciphertext = Feistel(tmptext, key_bin, 1)
        AllCiphertext += ciphertext
        print("Ciphertext: ", ciphertext)
    print("Your Binary Ciphertext: ", AllCiphertext)
    print("Your Hex Ciphertext: ", hex(int(AllCiphertext, base=2)).upper())

```

解密分组实现

主函数同加密分组实现，只需要把密钥取反即可

```
keylist = keylist[::-1] #INVERSEKEY
```

附加内容

用电码本模式对文件进行加密和解密

```
def opentxt():
    f = open('/tmp/test.txt')
    str1 = ""
    while True:
        str1 = f.readline()
    return str1

def Encryptmostr(text):
    lenth = len(text)
    flag = 0
    offset = 0
    # 补位操作
    if (lenth % 8 == 0):
        flag = 0
    else:
        flag = 1
        offset = 8 - lenth % 8
    for i in range(offset):
        text += 'A'
    tmplist = encode(text)
    groups = int(len(tmplist) / 8)
    #对齐8位
    for i in range(len(tmplist)):
        if len(tmplist[i]) != 8:
            tmpp = ""
            for j in range(8 - len(tmplist[i])):
                tmpp += '0'
            tmplist[i] = tmpp + tmplist[i]
    str_bin = ''.join(tmplist)
    #生成加密分组
    M = np.zeros((groups, 64))
    index = -8
    for i in range(groups):
        index += 8
        strr = ""
        for j in range(8):
            for k in tmplist[index+j]:
                strr += k
        for j in range(64):
            M[i][j] = int(strr[j])
    print("Your List Clear Text: ", tmplist)
    print("Your Binary Clear Text: ", str_bin)
    print("Your Hex Clear Text: ", hex(int(str_bin, base=2)).upper())
    print("Your Clear Text offset: ", offset)
    print("Your Clear Text groups: ", groups)
    key_bin = inkey()
    #打印加密群
```



```

AllCiphertext = ""
for i in range(groups):
    print("ClearText Group ", i)
    tmpText = ""
    for j in range(64):
        tmpText += str(int(M[i][j]))
    ciphertext = Feistel(tmpText, key_bin, 1)
    AllCiphertext += ciphertext
    print("Ciphertext: ", ciphertext)
print("Your Binary Ciphertext: ", AllCiphertext)
print("Your Hex Ciphertext: ", hex(int(AllCiphertext, base=2)).upper())
def encode(s):#字符串转二进制
    tmp = []
    for c in s:
        tmp.append(bin(ord(c)).replace('0b', ''))
    str_bin = ' '.join(tmp)
    for i in range(len(tmp)):
        if len(tmp[i]) != 7:
            tmpp = ""
            for j in range(7-len(tmp[i])):
                tmpp += '0'
            tmp[i] = tmpp + tmp[i]
    return tmp
def ParityCheck(s):#偶校验
    num = 0
    for i in s:
        if i == '1':
            num = num + 1
    if num%2 == 0:
        s += '0'
    else:
        s += '1'
    return s

```

自定义轮函数

$F(W,K)=(1 \ll W)+K$, 即W先循环左移1位再与K异或

```

def DIYDES(ClearTxt,Key,Model):
    keylist = createkey(Key)
    InitKeyCode = IP(ClearTxt)
    for key in keylist:
        InitKeyCode = InitKeyCode << 1;
        InitKeyCode = InitKeyCode ^ key
    return InitKeyCode

```

重要的实现细节

- 随机密钥的生成

```

    if model == "y":
        tmplist = []
        for i in range(8):
            j = random.randint(1,3)
            if j == 1:
                tmplist.append(chr(random.randint(48, 57)))
            if j == 2:

```

```

        tmplist.append(chr(random.randint(65, 90)))
    if j == 3:
        tmplist.append(chr(random.randint(97, 122)))
    for i in tmplist:
        strr += i
    keylist = encode(strr)
    # 补充奇偶校验位
    for i in range(len(keylist)):
        keylist[i] = ParityCheck(keylist[i])
    key_bin = ''.join(keylist)
    print("Your Key: ", strr)
    print("Your Binary Key: ", key_bin)
    print("Your Hex Key: ", hex(int(key_bin, base=2)).upper())
    return key_bin

```

- 校验位的生成和补位操作

- ```

def ParityCheck(s):#偶校验
 num = 0
 for i in s:
 if i == '1':
 num = num +1
 if num%2 == 0:
 s += '0'
 else:
 s += '1'
 return s
补充奇偶校验位
for i in range(len(keylist)):
 keylist[i] = ParityCheck(keylist[i])
key_bin = ''.join(keylist)
print("Your Key: ", strr)
print("Your Binary Key: ", key_bin)
print("Your Hex Key: ", hex(int(key_bin, base=2)).upper())

```

## 实现效果

### 实现文本加密

```

C:\Users\sysc\venv\Scripts\python.exe D:/build/F2U-15-404/密码学/DES/CUI_Me.py
Please choose Encrypt or Decrypt (1)Encrypt (2)Decrypt :2
Please choose your Model of Text (1)Hex (2)String :2
Please input your String : I Love you
Your List Clear Text: ['01001001', '0101100', '01011111', '0110110', '0100101', '0111001', '01101111', '0110101']
Your Binary Clear Text: 0100100101011000101110110110011001010110101010110110110101
Your Hex Clear Text: 0x09C67665796F75
Your Clear Text offset: 0
Your Clear Text groups: 1
Random Key (y/n): y
Your Key: xXU5750
Your Binary Key: 1111000010110001101010101100011000101010101010101010101010000
Your Hex Key: 0XF0B1AAC6A96A68
ClearText Group: 0
Movetimes: 0
L0: 010111101001101101100000
R0: 101010000001000011110000110
Your NO. 0 Sonkey: 011001001101101110001001011011010000100010
Movetimes: 1
L0: 10111101100111101101100000
R0: 01010000010000111100001101
Your NO. 1 Sonkey: 110110111000101101000000101001011011010001
Movetimes: 2
L0: 111110100111011100000010
R0: 0100000100001111000010101
Your NO. 2 Sonkey: 10001101100011011011101010110101010101010000
Movetimes: 3
L0: 11110110011101110000001011
R0: 0000010000111100001010100
Your NO. 3 Sonkey: 10101010011011101001100100100100100010000
Movetimes: 4
L0: 11011001110111000000101111
R0: 000110001111000011010100
Your NO. 4 Sonkey: 10101010011011101001100100100100100010000
Movetimes: 5
L0: 01100111101110000001011111
R0: 00100001111000010101010000

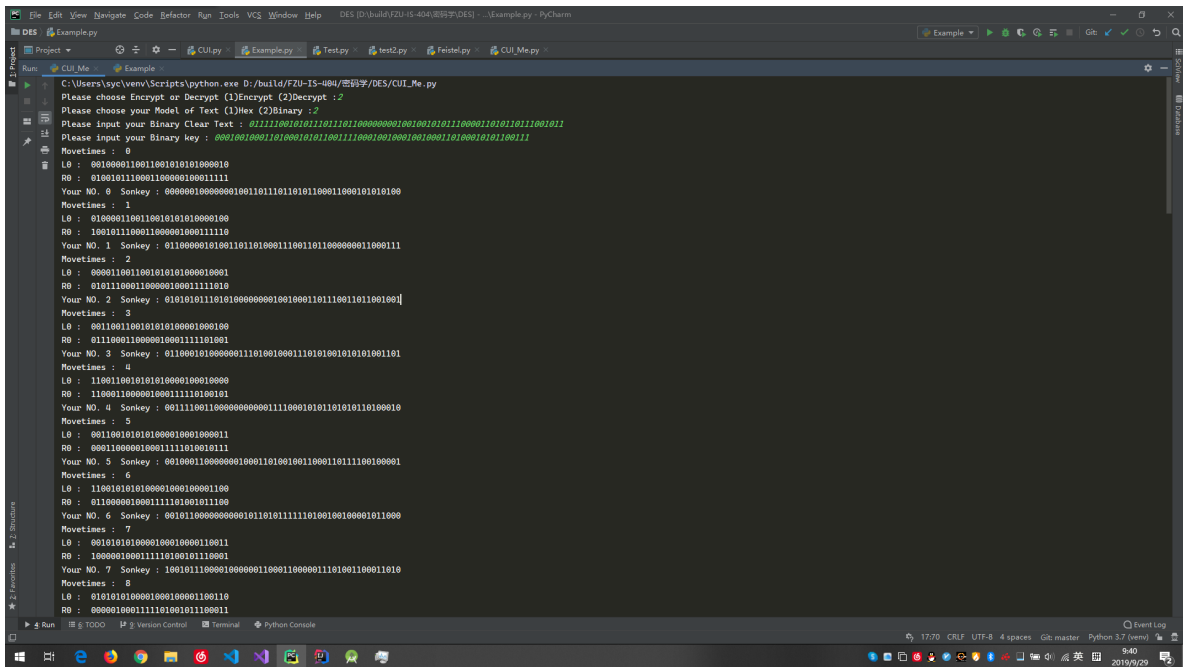
```

```
DES [D:\build\FZU-IS-404\密码学\DES] - CUI_Me.py - PyCharm
Project: CUI_Me.py
Run: CUI_Me.py
KeyList: keyList[1] #INVERSEKEY
Feistel() # Model == 2
Rn_expand: 0110111110000110101000000001110101100000001
S_input: 190267810072266
S_sub_str: 0010101100101010100000100110010
Ln: 110111001101000000101011000000
Rn: 1111111001111001001001001001
Rn_expand: 111111111010011111100001001001010101011
S_input: 53112904513117
S_sub_str: 1011100101100000010000001000001001
Ln: 1111111001111001001001001001001
Rn: 111100001100000001000110000110
Rn_expand: 011101000010110000000000101000111000000101
S_input: 1765132173439
S_sub_str: 110101100110000111111011000100
Ln: 111100001100000001000110000110
Rn: 100010110001010100100101010101010
Rn_expand: 0100010101111001010101010010010101010101
S_input: 20755911289732
S_sub_str: 01110011011001010110110110011000
Ln: 100010110001010100100101010101010
Rn: 00001111000000111011001000000
Rn_expand: 1001110001010100101101011111
S_input: 77002209295045
S_sub_str: 101000000011001110011001010101010
Ln: 100110001010100001010101010101010
Rn: 0010001010001110110110011110
Rn_expand: 0010000010101000001111101101110101111100
S_input: 77002209295045
S_sub_str: 101000000011001110011001010101010
Ln: 100110001010100001010101010101010
Rn: 0010001010001110110110011110
Rn_expand: 0010000010101000001111101101110101111100
S_input: 01110001011011001011101000100101111101010001000010111000
Ciphertext: 01110001011011001011101000100101111101010001000010111000
Your Binary Ciphertext: 01110001011011001011101000100101111101010001000010111000
Your Hex Ciphertext: 0X78802F465FDA2178
```

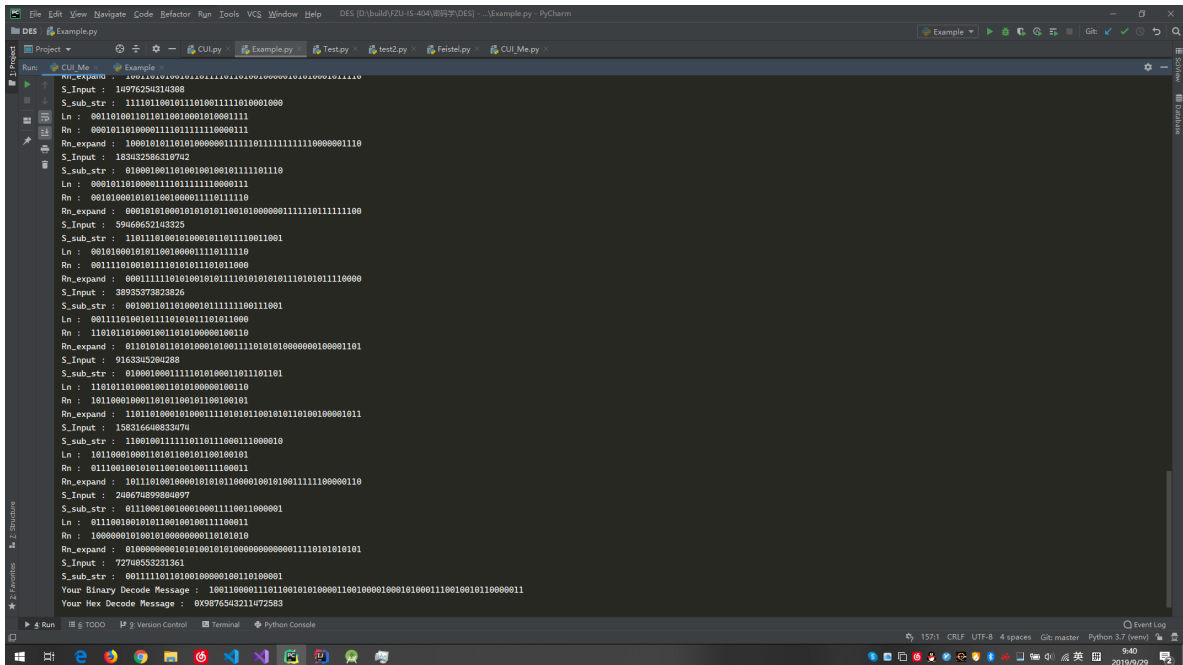
## 实现十六进制输入加密

```
DES [D:\build\FZU-IS-404\密码学\DES] - Example.py - PyCharm
Project: Example.py
Run: Example.py
Please choose Encrypt or Decrypt (1)Encrypt (2)Decrypt : 1
Please choose your Model of Text (1)Hex (2)String : 1
Please input your Hex (0x): 0X7CAEEC824AE1ADCB
Your List Clear Text: ['01111100', '10101110', '11101100', '00000010', '01001010', '11100001', '10101101', '11001011']
Your Binary Clear Text: 0111110010101101101000000010010010111000011010111001011
Your Hex Clear Text: 0X7CAEEC824AE1ADCB
Your Clear Text groups: 1
Random Key (y/n): y
Your Key: X0s8BuZ
Your Binary Key: 10110001011000001100111011000000110001110101100110011010100
Your Hex Key: 0XD160E76071E99904
ClearText Group 0: Movetimes: 0
L0: 11001010011110101111111011
R0: 010010010000100011000000010
Your NO. 0 Sonkey: 10111010011110001111110000100110000000100100
Movetimes: 1
L0: 1001010011110101111110111
R0: 100100100010001100000000100
Your NO. 1 Sonkey: 0111100111111110001100000000101010000000011
Movetimes: 2
L0: 0101001111010111111011101110
R0: 010010001000110000000010010
Your NO. 2 Sonkey: 01110000101010111111101000100010000000010000
Movetimes: 3
L0: 01001111010111111101110111001
R0: 0010000100011000000001001001
Your NO. 3 Sonkey: 1111010110011000110111010000100000000101001100
Movetimes: 4
L0: 00111101011111110111100101
R0: 1000010001100000000100100100
Your NO. 4 Sonkey: 1110011111010111010110000000001001010000010
Movetimes: 5
L0: 111110101111111101110010100
R0: 000100011000000001001001010
Your NO. 5 Sonkey: 10111101011011110101101010100000010000100101
Movetimes: 6
L0: 111010111111101110010100011
R0: 0100011000000001001001001000
Your NO. 6 Sonkey: 11111111000101100111011000010100000100011001000
```

## 实现二进制输入解密



```
C:\Users\sys\venv\Scripts\python.exe D:/build/FZU-IS-404/密码学/DES/CUI_Me.py
Please choose Encrypt or Decrypt (1)Encrypt (2)Decrypt : 2
Please choose your Model of Text (1)Hex (2)Binary : 2
Please input your Binary Clear Text : 011111001010111011000000010010010101100001101010110010011
Please input your Binary key : 0001001000101000101011001110001001000100100010101100111
Movetimes : 0
L0 : 0010000110011001010101000010
R0 : 0100101110001100000100011111
Your NO. 0 Sonkey : 00000100000001001101101011000110001000101010100
Movetimes : 1
L0 : 0100001100110010101010000100
R0 : 100101100011000001000011110
Your NO. 1 Sonkey : 01100001010010110100011001101100000011000111
Movetimes : 2
L0 : 00001100110010101000010001
R0 : 010111000110000010001111010
Your NO. 2 Sonkey : 0101011101010000000100100011011001101100100
Movetimes : 3
L0 : 00110011001010100001000100
R0 : 011100011000001000111101001
Your NO. 3 Sonkey : 0110001010000001110100100011101001010100101001101
Movetimes : 4
L0 : 110011001010100001000100010000
R0 : 110001100000100011110100101
Your NO. 4 Sonkey : 0011110011000000000001110001010110101010100010
Movetimes : 5
L0 : 001100101010000100010001000011
R0 : 000110000010001111010010111
Your NO. 5 Sonkey : 0010001100000001000110100100110111100100001
Movetimes : 6
L0 : 110010101010000100010001100
R0 : 01100000100011110100101100
Your NO. 6 Sonkey : 001011000000000010101011111010010010000101000
Movetimes : 7
L0 : 00101010000100010000110011
R0 : 1000010001111010010110001
Your NO. 7 Sonkey : 100101100010000001100011000001110100110011010
Movetimes : 8
L0 : 0101010000100010000110010
R0 : 00000100011110100101100011
DES [D:/build/FZU-IS-404/密码学/DES] ...Example.py - PyCharm
```



```
DES [D:/build/FZU-IS-404/密码学/DES] ...Example.py - PyCharm
S_Input : 109762504310308
S_sub_str : 11110100101101001111010001000
Ln : 00110100110110110010001010001111
Rn : 00010101000001110111110000111
Rn_expand : 10001010101010000001111101111111110000001110
S_Input : 18303206310702
S_sub_str : 01000100101001001001011101110
Ln : 0001010100001110111110000111
Rn : 0010000101010010000111011110
Rn_expand : 0001010001010101010100000011111011111100
S_Input : 5900060210325
S_sub_str : 110110100101000101001110011001
Ln : 001000010101001000111011110
Rn : 001110100101110101011001000
Rn_expand : 00011111010100101011101010101110000
S_Input : 3893573823826
S_sub_str : 00100110101000101111110011001
Ln : 0011101001011101010110101000
Rn : 11010101000100110101000010010
Rn_expand : 0110101010100001000111101010100000010000101
S_Input : 9163305204288
S_sub_str : 010001000111101010001011010101
Ln : 110101101000100101010000010010
Rn : 101000100001010110010100100101
Rn_expand : 110101000101000111010101001010100100001011
S_Input : 158316008033074
S_sub_str : 1100100111110110111000011000010
Ln : 10110001000101010010100100101
Rn : 0111001001010100100111100011
Rn_expand : 1011010100000101010100001001001001111100000110
S_Input : 200070090000097
S_sub_str : 01110001001000100001110011000001
Ln : 01110001001010100100011100011
Rn : 100000010100101000000010101010
Rn_expand : 01000000001010101010000000000001110101010101
S_Input : 7270053231361
S_sub_str : 001111010100100000100110100001
Your Binary Decode Message : 10011000011101001010000110010000110010001110010010110000011
Your Hex Decode Message : 0X9876543211072583
```

## 总结

### 系统亮点

- 实现了完成的Feistel密码结构的分层
- 实现了Key轮加密密钥的类抽象
- 实现了随机密钥的生成方法
- 提供了包括二进制、十六进制、字符串、文本文件在内的多种输入方式
- 提供了自定义轮函数的接口