南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

现代密码学实验报告

实验二　分组密码算法 DES

于文明

年级：2020 级

专业：信息安全

指导教师：古力

2022 年 11 月 26 日

摘要

关键字：Block cipher DES Avalanche effect test

目录

一、 实验内容

(一) 实验步骤 [1]

1. 对课本中 DES 算法进行深入分析，对初始置换、E 扩展置换，S 盒代换、轮函数、密钥生成等环节要有清晰的了解，并考虑其每一个环节的实现过程。

2. DES 实现程序的总体设计：在第一步的基础上，对整个 DES 加密函数的实现进行总体设计，考虑数据的存储格式，参数的传递格式，程序实现的总体层次等，画出程序实现的流程图。

3. 在总体设计完成后，开始具体的编码，在编码过程中，注意要尽量使用高效的编码方式。

4. 利用 3 中实现的程序，对 DES 的密文进行雪崩效应检验。即固定密钥，仅改变明文中的一位，统计密文改变的位数；固定明文，仅改变密钥中的一位，统计密文改变的位数。

(二) 实验报告要求

(1) 分别实现 DES 的加密和解密，提交程序代码和执行结果。

(2) 在检验雪崩效应中，要求至少改变明文和密文中各八位，给出统计结果并计算出平均值。

二、 程序流程图

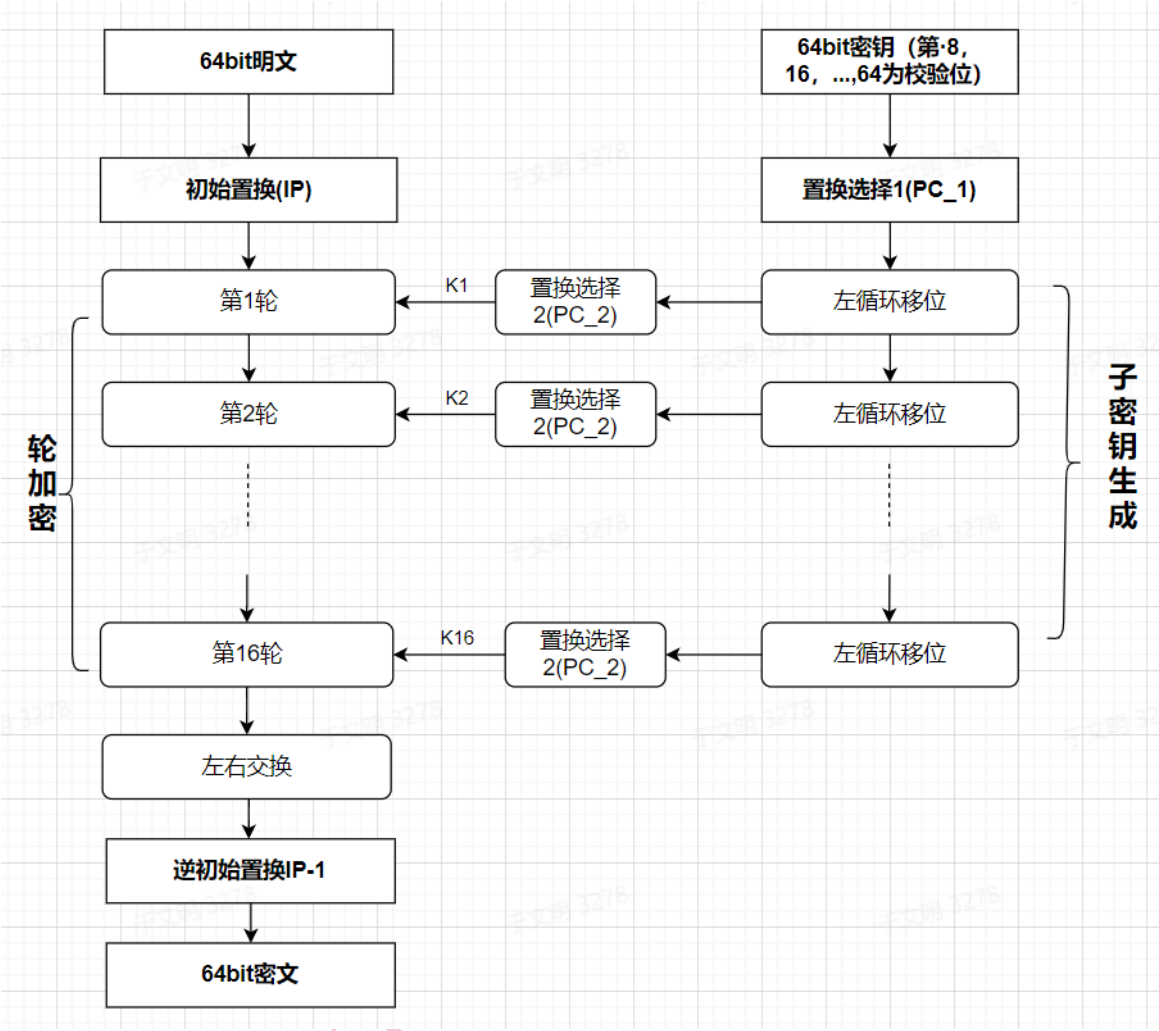

图 1: DES 加密算法框图

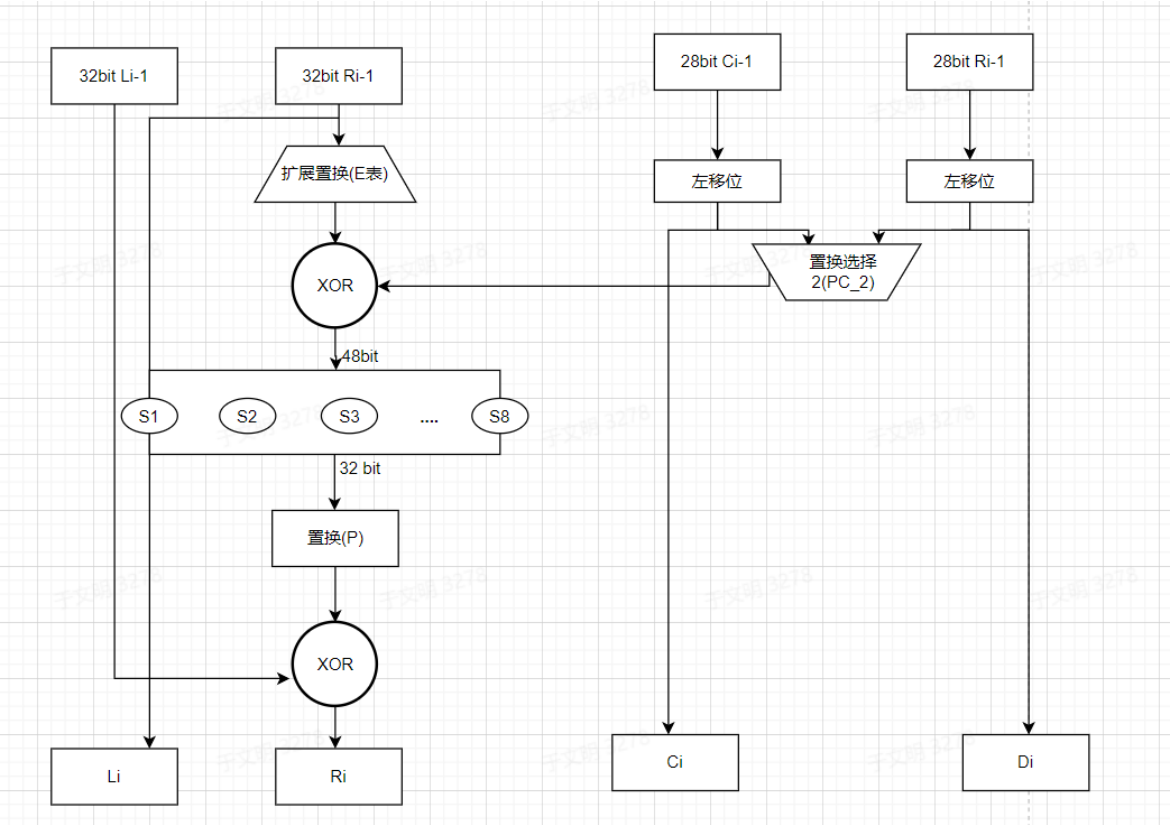

图 2: DES 加密算法的轮结构

DES 的解密和加密使用同一算法，但是子密钥的顺序相反。

三、 核心代码

(一) 数据编码

题目的测试数据定义了一个结构体 des_test_case 如下:

des_test_case 结构体

```
static const struct des\_test\_case {
    int num, mode; // mode 1 = encrypt
    unsigned char key[8], txt[8], out[8];
}
```

其中密钥，明文，密文都是以 16 进制 char 数组的形式给出,DES 算法需要的数据都是二进制数据，所以通过 STL 标准库的 bitset 来存储二进制数据

同时定义以下辅助函数来将 16 进制 char 类型数据转换为二进制数据，以及其逆过程

辅助函数

```
bitset<64> strTobitset(char s[8])
{
        bitset<64> binset;
        for (int i = 0; i < 8; i++)
        {
```

```
                for (int j = 0; j < 8; j++)
                {
                        binset[i * 8 + j] = (s[i] >> (7-j)) & 1;
                }
        }
        return binset;
}
char* bitsetTostr(bitset<64> binset) {
        char s[8];
        bitset<8> tmp;
        for (int i = 0; i < 8; i++)
        {
                for (int j = 0; j < 8; j++)
                {
                        tmp[j] = binset[i * 8 + j];
                }
                s[i] = static_cast<unsigned char>(tmp.to_ulong());
        }
        return s;
}
```

(二) DES 类框架

1. DES 类

DES 算法通过定义 class 实现，其中类成员主要是一些置换矩阵和轮函数的子密钥，以及 S 盒矩阵, 此外，将 DES 加密算法的操作封装为类成员函数具体实现如下:

辅助函数

```
class DES {
        bitset<48> subKeys[16];//轮函数的子密钥
        const int ip[64];
        const int PC_1[56];
        const int PC_2[56];
        const int E[48];
        const int P[32];
        const int S_BOX[8][4][16];
        const int ip_1[64];
        bitset<64> IP(const bitset<64> plain);
    bitset<64> IP_1(const bitset<64>& S);
    bitset<32> F(const bitset<32>& r, const bitset<48>& subkey);//轮函数
        void gernerateSubkey(const bitset<64>& key);//生成子密钥

        //DES();
        bitset<64> encrypt(const bitset<64>& plain, const bitset<64>& key);
        bitset<64> decrypt(const bitset<64>& plain, const bitset<64>& key);
};
```

2. 初始置换 IP 和逆初始置换

初始置换的作用是把输入的 64 位数据块的排列顺序打乱，每位数据按照下面的置换规则重新排列，即将第 58 位换到第一位，第 50 位换打第 2 位，…，依次类推。置换后的 64 位输出分为 L0 、R0（左、右）两部分，每部分分别为 32 位。

辅助函数

```
bitset<64> IP(const bitset<64> plain)
{
        bitset<64> IPS;
        for (int i = 0; i < 64; i++)
        {
                IPS[i] = plain[ip[i]-1];
        }
        return IPS;
}
bitset<64> IP_1(const bitset<64>& S)
{
        bitset<64> cipher;
        //构造ip逆置换
        for (int i = 0; i < 64; i++)
        {
                cipher[i] = S[ip_1[i] - 1];
        }
        return cipher;
}
```

3. 子密钥生成模块

具体子密钥的生成如流程图所示。输入的初始密钥值为 64 位，但 DES 算法规定，其中第 8、16、…、64 位为奇偶校验位，不参予 DES 的运算。所以，实际可用位数只有 56 位，经过缩小选择位表 1（表 1 － 2）即密钥置换 PC-1 的变换后，初始密钥的位数由 64 位变成了 56 位，将其平分位两部分 C0，D0。然后分别进行第一次循环左移，得到 C1 和 D1，将 C1（28 位）、D1（28 位）合并后得到 56 位的输出结果，再经过压缩置换 PC-2(表 1 － 3)，从而得到了密钥 K1（48 位）。依次类推，便可得到 K2、…、K16。需要注意的是，16 次循环左移对应的左移位数要依据表 1 － 1 的规则进行，实现如下:

子密钥生成模块

```
void DES::gernerateSubkey(const bitset<64>& key) {
        bitset<56> cur;
        bitset<28> left;
        bitset<28> right;

        //pc_1置换
        for (int i = 0; i < 56; i++)
        {
                cur[i] = key[PC_1[i]-1];
                //cout << cur[i];
```

```
        }
        //cout << endl;
        for (int i = 0; i < 28; i++)
        {
                left[i] = cur[i];
        }
        for (int i = 0; i < 28; i++)
                right[i] = cur[i+28];

        //循环左移位+压缩置换
        int shiftBits[] = { 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };
        for (int round = 0; round < 16; round++)
        {
                leftShift(left, shiftBits[round]);
                leftShift(right, shiftBits[round]);
                //cout << "left" << round << ':';
                /*for (int j = 0; j < 28; j++)
                        cout << left[j];
                cout << endl;
                cout << "right" << round << ':';
                for (int j = 0; j < 28; j++)
                        cout << right[j];*/
                //合并
                for (int i = 0; i < 56; i++)
                {
                        if (i < 28)
                                cur[i] = left[i];
                        else
                                cur[i] = right[i-28];
                }

                //压缩置换
                //cout << "第i轮的密钥为:";
                for (int j = 0; j < 48; j++)
                {
                        subKeys[round][j] = cur[PC_2[j]-1];
                        //cout<< subKeys[round][j];
                }
                //cout << endl;
        }
}
```

4. f 函数

f 函数是多个置换函数和替代函数的组合函数，它将 32 位比特的输入变换为 32 位的输出，如图 1 — 2 所示。Ri 经过扩展运算 E 变换后扩展为 48 位的 E（Ri），与进行异或运算后输出的结果分成 8 组，每组 6 比特。每一组再经过一个 S 盒（共 8 个 S 盒）运算转换为 4 位，8 个 4 位合并为 32 位后再经过 P 变换输出为 32 位的。其中，扩展运算 E 与置换 P 主要作用是增加

算法的扩散效果。

f 函数

```
bitset<32> DES::F(const bitset<32>& r, const bitset<48>& subkey) {
        //扩展
        bitset<48> Exp;
        bitset<32> Fout;
        //cout << "Exp:";
        for (int i = 0; i < 48; i++)
        {
                Exp[i] = r[E[i] - 1];
                //cout << Exp[i];
        }
        //cout << endl;
        //与子密钥异或
        /*cout << "org Exp";
        for (int i = 0; i < 48; i++)
                cout << Exp[i];
        cout << endl;*/
        Exp = Exp ^ subkey;
        /*cout << "Exp";
        for (int i = 0; i < 48; i++)
                cout << Exp[i];
        cout << endl;*/
        //分组，经过S盒置换
        for (int i = 0; i < 8; i++)
        {
                int head = i*6;//每一组第一位的下标
                int row = Exp[head] * 2 + Exp[head + 5];
                int col = Exp[head+1] * 8 + Exp[head+2] * 4 + Exp[head+3] * 2
                    + Exp[head+4];
                //cout << "row" << row << "col " << col << endl;
                //置换
                for (int j = 0; j < 4; j++)
                {
                        Fout[i * 4 + j] = (S_BOX[i][row][col] >> (3-j)) & 1;
                }
        }
        /*cout << "S:";
        for (int i = 0; i < 32; i++)
                cout << Fout[i];
        cout << endl;*/
        //P置换
        bitset<32> tmp = Fout;
        //cout << "P:";
        for (int i = 0; i < 32; i++)
        {
                Fout[i] = tmp[P[i] - 1];
```

```
                //cout << Fout[i];
        }
        //cout << endl;
        return Fout;
}
```

5. 加密模块

具体加密过程首先是将输入的数据进行初始置换（IP），即将明文 M 中数据的排列顺序按一定的规则重新排列，生成新的数据序列，以打乱原来的次序。然后将变换后的数据平分成左右两部分，左边记为 L0，右边记为 R0，然后对 R0 实行在子密钥（由加密密钥产生）控制下的变换 f，结果记为 f（R0，K1），再与 L0 做逐位异或运算，其结果记为 R1，R0 则作为下一轮的 L1。如此循环 16 轮，最后得到 L16、R16，再对 L16、R16 实行逆初始置换 IP − 1，即可得到加密数据。

加密模块

```
bitset<64> DES:: encrypt(const bitset<64>& plain, const bitset<64>& key) {
        bitset<32> left;
        bitset<32> right;
        bitset<32> nextLeft;
        bitset<64> cur;

        cur = IP(plain);
        for (int i = 0; i < 32; i++)
        {
                left[i] = cur[i];
        }
        for (int i = 0; i < 32; i++)
        {
                right[i] = cur[i+32];
        }

        //16轮迭代
        gernerateSubkey(key);
        /*for (int i = 0; i < 16; i++)
        {
                cout << "第" << i << "轮的密钥为:" << endl;
                for (int j = 0; j < 48; j++)
                        cout << subKeys[i][j];
                cout << endl;
        }*/
        for (int i = 0; i < 16; i++)
        {
                nextLeft = right;
                right = left ^ F(right, subKeys[i]);//异或
                /*cout << "f:";
                for (int i = 0; i < 32; i++)
                        cout << right[i];
                cout << endl;*/
```

```
                left = nextLeft;
        }

        for (int i = 0; i < 32; i++)
                cur[i] = right[i];
        for (int i = 0; i < 32; i++)
                cur[i + 32] = left[i];
        return IP_1(cur);
}
```

6. 解密模块

解密过程与加密类似，不同之处仅在于子密钥的使用顺序正好相反。

解密模块

```
bitset<64> DES::decrypt(const bitset<64>& plain, const bitset<64>& key) {

        bitset<32> left; // 记录上半部分
        bitset<32> right; // 记录下半部分
        bitset<32> nextLeft; // 作为16轮迭代的中间临时变量
        bitset<64> cur; // 记录每一步置换的结果

        // 第一步: IP初始置换
        cur = IP(plain);

        // 获取L和R
        for (int i = 0; i < 32; i++)
                left[i] = cur[i];
        for (int i = 0; i < 32; i++)
                right[i] = cur[i+32];

        // 第二步: 16轮迭代T
        gernerateSubkey(key);//生成子密钥
        for (int i = 0; i < 16; i++) {
                nextLeft = right;
                right = left ^ F(right, subKeys[15 - i]); // 子密钥调度顺序与
                    加密时相反
                left = nextLeft;
        }

        // 第三步: 交换置换
        for (int i = 0; i < 32; i++)
                cur[i] = right[i];
        for (int i = 0; i < 32; i++)
                cur[i + 32] = left[i];

        // 第四步: IP_1逆置换
        return IP_1(cur);
```

```
}
```

7. main 函数

main 函数

```
int main() {

        for (int i = 0; i < 20; i++)
        {
                DES des;
                bitset<64> txt = strTobitset((char*)cases[i].txt);
                bitset<64> key=strTobitset((char *)cases[i].key);
                char* out;
                bitset<64> cipher;
                if (cases[i].mode)
                {
                        cipher=des.encrypt(txt, key);
                        //输出16进制加密结果
                        cout <<"第"<<i<<"组加密结果为:";
                        bitsetTohex(cipher);
                        cout << endl;
                }
                else
                {
                        cipher= des.decrypt(txt, key);
                        cout << "第" << i << "组解密结果为:";
                        bitsetTohex(cipher);
                        cout << endl;
                }
                bool flag = 1;
                bitset<64> bout = strTobitset((char*)cases[i].out);
                for (int i = 0; i < 64; i++)
                {
                        if (cipher[i] != bout[i])
                        {
                                flag = 0;
                                break;
                        }
                }
                if (flag)
                        cout << i << ":true" << endl;
                else
                        cout << i << ":false"<<endl;
        }
        //avalancheTest(cases[0], 0);

        //printf("%x", &s);
```

```
43          //初始置换
44
45          system("pause");
46          return 0;
47  }
```

四、 执行结果

```
第0组加密结果为:0x82 0xDC 0xBA 0xFB 0xDE 0xAB 0x66 0x2
0:  true
第1组加密结果为:0x80 0x0 0x0 0x0 0x0 0x0 0x0 0x0
1:  true
第2组加密结果为:0x40 0x0 0x0 0x0 0x0 0x0 0x0 0x0
2:  true
第3组加密结果为:0x20 0x0 0x0 0x0 0x0 0x0 0x0 0x0
3:  true
第4组加密结果为:0x10 0x0 0x0 0x0 0x0 0x0 0x0 0x0
4:  true
第5组加密结果为:0x8 0x0 0x0 0x0 0x0 0x0 0x0 0x0
5:  true
第6组加密结果为:0x4 0x0 0x0 0x0 0x0 0x0 0x0 0x0
6:  true
第7组加密结果为:0x2 0x0 0x0 0x0 0x0 0x0 0x0 0x0
7:  true
第8组加密结果为:0x1 0x0 0x0 0x0 0x0 0x0 0x0 0x0
8:  true
第9组加密结果为:0x0 0x80 0x0 0x0 0x0 0x0 0x0 0x0
9:  true
```

图 3: 加密结果

```
第10组解密结果为:0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
10:  true
第11组解密结果为:0x95 0xF8 0xA5 0xE5 0xDD 0x31 0xD9 0x0
11:  true
第12组解密结果为:0xDD 0x7F 0x12 0x1C 0xA5 0x1 0x56 0x19
12:  true
第13组解密结果为:0x2E 0x86 0x53 0x10 0x4F 0x38 0x34 0xEA
13:  true
第14组解密结果为:0x4B 0xD3 0x88 0xFF 0x6C 0xD8 0x1D 0x4F
14:  true
第15组解密结果为:0x20 0xB9 0xE7 0x67 0xB2 0xFB 0x14 0x56
15:  true
第16组解密结果为:0x55 0x57 0x93 0x80 0xD7 0x71 0x38 0xEF
16:  true
第17组解密结果为:0x6C 0xC5 0xDE 0xFA 0xAF 0x4 0x51 0x2F
17:  true
第18组解密结果为:0xD 0x9F 0x27 0x9B 0xA5 0xD8 0x72 0x60
18:  true
第19组解密结果为:0xD9 0x3 0x1B 0x2 0x71 0xBD 0x5A 0xA
19:  true
```

图 4: 解密结果

五、 雪崩效应检验

1. 检验函数代码

根据雪崩效应的规则，编写了雪崩效应检验函数 avalancheTest，输入为 des_test_case, 分别改变明文位数 64 次，统计改变之后密文的变动 bit 数；以及改变密钥次数 56 次，统计密文的变动 bit 数，求出其平均值分析

雪崩效应检验

```
void avalancheTest(des\_test\_case c,int m){//雪崩效应检查
        cout << "m =0 change plaintext ; m=1, change key" << endl;
        cout << "改变位置   " << "密文改变bit数   "<<"mode   " <<"avg"<<endl;
        bitset<64> txt = strTobitset((char*)c.txt);
        bitset<64> key = strTobitset((char*)c.key);
        DES des;
        int ss=0;
        if (!m)
        {
                bitset<64> cipher0;
                cipher0 = des.encrypt(txt, key);
                int sum;
                for (int i = 0; i < 64; i++)
                {
                        sum = 0;
                        bitset<64> cipher1;
                        txt[i] = ~txt[i];
                        cipher1 = des.encrypt(txt, key);
                        //统计比较
                        for (int j = 0; j < 64; j++)
                        {
                                if (cipher0[j] != cipher1[j])
                                        sum++;
                        }
                        cout <<i << "            " << sum <<"            "
                            <<m<< endl;
                        ss += sum;
                        txt[i] = ~txt[i];
                }
                cout << "                                        " << ss
                     / 64 << endl;
                return;
        }
        bitset<64> cipher0;
        cipher0 = des.encrypt(txt, key);
        int sum;
        for (int i = 0; i < 64; i++)
        {
                if ((i + 1) % 8 == 0)
                        continue;
```

```
                sum = 0;
                bitset<64> cipher1;
                key[i] = ~key[i];
                cipher1 = des.encrypt(txt, key);
                //统计比较
                for (int j = 0; j < 64; j++)
                {
                        if (cipher0[j] != cipher1[j])
                                sum++;
                }
                cout << "改变明文第" << i << "位后, 密文改变" << sum << endl;
                ss += sum;
                key[i] = ~key[i];
        }
        cout << "经过56次统计, 平均值为" << ss / 64 << endl;
        return;
}
```

2. 检验结果

由于图片大小限制，这里只检验了 8 次

```
m =0 change plaintext ; m=1, change key
改变位置    密文改变bit数    mode    avg
0             43              0
1             34              0
2             33              0
3             29              0
4             29              0
5             31              0
6             29              0
7             31              0
                                      32
```

图 5: 改变明文 8 次测试结果

```
m =0 change plaintext ; m=1, change key
改变位置    密文改变bit数    mode    avg
0             33              1
1             34              1
2             40              1
3             40              1
4             37              1
5             34              1
6             35              1
8             39              1
                                      36
```

图 6: 改变密钥 8 次测试结果

参考文献

[1] 吴世忠、宋晓龙、郭涛等译 Paul Garrett 著. *An Introduction to Cryptology*. 机械工业出版社, 2003.