

实验二 分组密码算法 DES

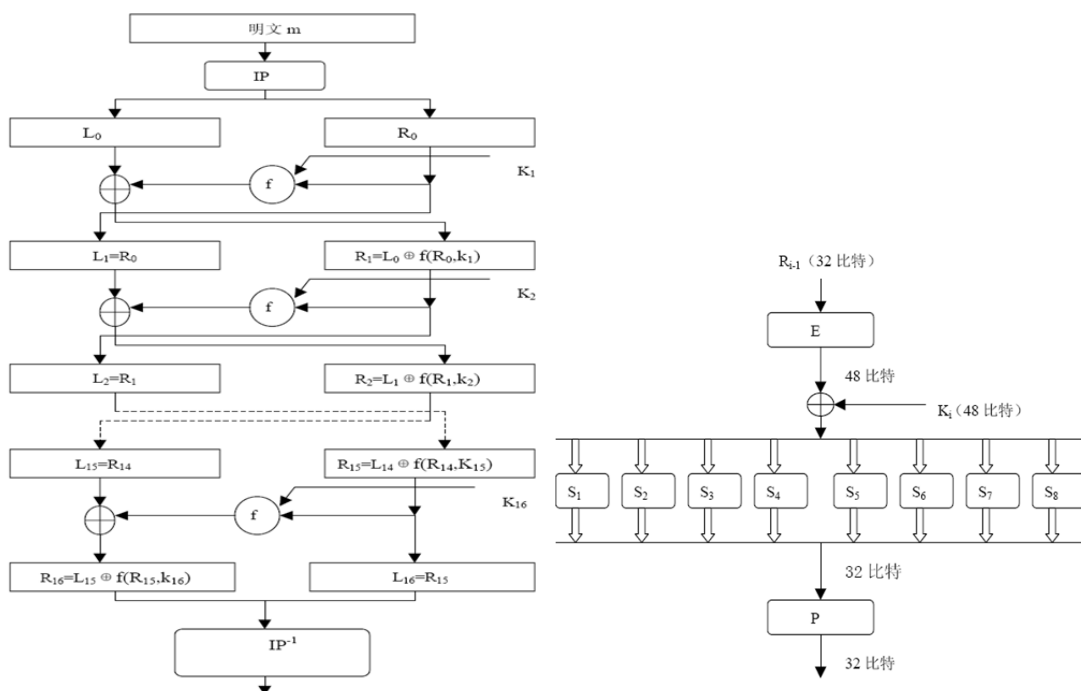
姓名：范毓哲 学号：1910378

一、 实验目的

通过用 DES 算法对实际的数据进行加密和解密来深刻了解 DES 的运行原理。

二、 设计流程

移位密码的加解密：



三、 实验环境

Windows10 操作系统+VS2019(VC17)。

同级目录下，des.cpp 为 DES 分组加解密算法的实现，密钥雪崩.cpp 为修改密钥时密文的变化情况统计，明文雪崩.cpp 为修改明文时密文的变化情况统计，对应的 exe 分别为其可执行程序。所有程序需要使用的头文件都是 tables.h

四、 程序实现

1. DES 加解密

如下的代码实现了 DES 分组加解密（tables.h 当中只列了已经给出的置换表等基

础数据，没有需要进行程序设计的地方，故报告中不再分析）。

```
#include "tables.h"
using namespace std;
//定义全局变量：置换前的明文，置换后的明文，操作码，密钥，56位密钥
int plaintext_64[64], plaintext_64_IP[64], mode = 0, key[64], key_56[56];
//将指定字段转换为二进制数组
void hex2bin(int x, int txt[], int tag)
{
    for (int i = 0; i < 8; i++)
    {
        int a = cases[x].txt[i], j = 0;
        if (tag == 1) a = cases[x].key[i];
        while (a > 0) {txt[8*i+7-j]=a%2; a/=2; j++;}
    }
}
//生成子密钥
int mkidkey[16][56], tp[4]={26,27,54,55}, kidkey[16][48];
void shiftkey()
{
    for (int j = 0; j < 27; j++)
    {
        mkidkey[0][j] = key_56[j + 1];
        mkidkey[0][j + 28] = key_56[j + 29];
    }
    mkidkey[0][27] = key_56[0]; mkidkey[0][55] = key_56[28];
    for (int i = 1; i < 16; i++)
    {
        if (shift[i] == 1) //shift 是密钥的移位位数
        {
            for (int j = 0; j < 27; j++)
            {
                mkidkey[i][j] = mkidkey[i - 1][j + 1];
                mkidkey[i][j + 28] = mkidkey[i - 1][j + 29];
            }
            mkidkey[i][27] = mkidkey[i - 1][0];
            mkidkey[i][55] = mkidkey[i - 1][28];
        }
        if (shift[i] == 2)
        {
            for (int j = 0; j < 26; j++)
            {
                mkidkey[i][j] = mkidkey[i - 1][j + 2];
                mkidkey[i][j + 28] = mkidkey[i - 1][j + 30];
            }
        }
    }
}
```

```

        for (int j = 0; j < 4; j++) mkidkey[i][tp[j]] = mkidkey[i-1][tp[j] - 26];
    }
}

void lun(int temp[]) //DES 的核心轮操作
{
    int sum = 0, Lturn[17][64], Eres[16][48], S_res[16][32], P_res[16][32];
    memset(S_res, 0, sizeof(S_res));
    for (int i = 0; i < 64; i++) Lturn[0][i] = plaintext_64_IP[i];
    for (int i = 1; i < 17; i++)
    {
        for (int j = 0; j < 32; j++) Lturn[i][j] = Lturn[i - 1][j + 32];
        //E 扩展+异或
        for (int j = 0; j < 48; j++)
        {
            int te = Lturn[i - 1][E[j] - 1 + 32];
            if (mode == 0) Eres[i - 1][j] = te ^ kidkey[i - 1][j];
            if (mode == 1) Eres[i - 1][j] = te ^ kidkey[16 - i][j];
        }
        for (int j = 0; j < 8; j++) //S 盒置换
        {
            int a = Eres[i - 1][6 * j] * 2 + Eres[i - 1][6 * j + 5];
            int b = Eres[i - 1][6 * j + 1] * 8 + Eres[i - 1][6 * j + 2] * 4 + Eres[
i - 1][6 * j + 3] * 2 + Eres[i - 1][6 * j + 4];
            int c = S[j][a][b]; int ji = 0;
            while (c > 0) { S_res[i - 1][4 * j + 3 - ji] = c % 2; c /= 2; ji++; }
        }
        //P 置换
        for (int j = 0; j < 32; j++) P_res[i - 1][j] = S_res[i - 1][P[j] - 1];
        for (int j = 0; j < 32; j++) Lturn[i][j + 32] = Lturn[i - 1][j] ^ P_res[i -
1][j];
    } //最后一次对换
    for (int i = 0; i < 32; i++) { temp[i] = Lturn[16][i + 32]; temp[i + 32] = Ltur
n[16][i]; }
}

int main()
{
    int idx = 0, cipher[64], temp[64];
    cout << "请输入加解密的 mode 和测试组别, 用空格分开(mode 为 0 表示加密, 1 解密)." << endl;
    cout << "(输负数则可直接退出程序)\n"; int ii = 0;
    //cout << "二十组加解密测试结果如下: \n";
    while (ii < 20)

```

```

{
    cout << "输入 mode 和组别: "; cin >> mode >> idx;
    //idx = ii; ii++; if (idx > 9) mode = 1;
    memset(plaintext_64,0,sizeof(plaintext_64));memset(plaintext_64_IP,0,sizeof
(plaintext_64_IP));
    memset(key, 0, sizeof(key)); memset(key_56, 0, sizeof(key_56));
    idx=10 * mode+idx; if(mode<0||idx<0) {cout<<"【退出】程序结束\n";return 0; }
    hex2bin(idx, plaintext_64, 0);//得到 IP 置换后的文本
    for (int i = 0; i < 64; i++) plaintext_64_IP[i] = plaintext_64[IP[i] - 1];
    hex2bin(idx, key, 1); //得到密钥 PC-1 置换后的文本
    for (int i = 0; i < 56; i++) key_56[i] = key[PC_1[i] - 1];
    shiftkey();//子密钥移位
    for (int i = 0; i < 16; i++) for (int j = 0; j < 48; j++) kidkey[i][j] = mk
idkey[i][PC_2[j] - 1];
    lun(temp);
    for (int i = 0; i < 64; i++) cipher[i] = temp[IP_1[i] - 1];//逆置换
    for (int i = 0; i < 16; i++) //输出十六进制形式的结果
    {
        int resi = cipher[4*i]*8+cipher[4*i+1]*4+cipher[4*i+2]*2+cipher[4*i+3];
        if (i % 2 == 0) { printf("0x%X", resi); continue; }
        printf("%X", resi); if (i != 15) cout << ",";
    }
    cout << endl;
}
}

```

2. 明文雪崩

程序的整体和原先的加解密没有任何区别，只是添加了对比密文变化的函数：

```

int sum = 0; //记录每一组的总位数变化
void compare(int x,int cipher[])
{
    int standard[64]; memset(standard, 0, sizeof(standard));
    for (int i = 0; i < 8; i++)
    {
        int a = cases[x].out[i], j = 0;
        while (a > 0) //取出原结果
        { standard[8 * i + 7 - j] = a % 2; a /= 2; j++; }
    }
    for (int i = 0; i < 64; i++) //进行逐位比较
    {if (cipher[i] != standard[i]) sum++;}
}

```

然后是 main 函数当中：

```

int ii = 0; srand(time(0));
int num[10]; for (int i = 0; i < 8; i++) num[i]=rand() % 64;

```

```

while (ii<10)
{
    idx = ii; ii++; if (idx > 9) mode = 1;
    sum = 0;
    for (int k = 0; k < 8; k++) //进行八次明文变换
    {
        memset(plaintext_64, 0, sizeof(plaintext_64)); memset(plaintext_64_IP,
0, sizeof(plaintext_64_IP)); memset(key, 0, sizeof(key)); memset(key_56, 0, siz
eof(key_56)); memset(temp, 0, sizeof(temp)); memset(kidkey, 0, sizeof(kidkey));
memset(mkidkey, 0, sizeof(mkidkey));
        hex2bin(idx, plaintext_64, 0);//得到 IP 置换后的文本
        plaintext_64[num[k]] = 1 - plaintext_64[num[k]];
        cout <<"第"<<idx<<"组的明文第" << num[k] << "位取反。" << endl;
        //其余内容不变，此处省略
    }
    cout << "第" << idx << "组密文改变的平均位数是" << sum / 8; cout << "\n\n";
}
getchar();
}

```

3. 密钥雪崩

只要把明文雪崩效应测试代码当中对明文的修改，改成对密钥的修改即可，其他内容不变：


```

key[num[k]] = 1 - key[num[k]];
cout <<"第"<<idx<<"组的密钥第" << num[k] << "位取反。" << endl;

```

五、 实验结果

移位密码的加解密结果如下，首先是单组测试的结果：

 Microsoft Visual Studio 调试控制台

```

请输入加解密的mode和测试组别，用空格分开(mode为0表示加密，1解密).
(输负数则可直接退出程序)
输入mode和组别: 0 0
0x82, 0xDC, 0xBA, 0xFB, 0xDE, 0xAB, 0x66, 0x02
输入mode和组别: 0 9
0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
输入mode和组别: 1 0
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
输入mode和组别: 1 9
0xD9, 0x03, 0x1B, 0x02, 0x71, 0xBD, 0x5A, 0x0A
输入mode和组别: -1 -1
【退出】程序结束

```

在注释中也提供了输出全部结果的代码，结果如下：

Microsoft Visual Studio 调试控制台

```
二十组加解密测试结果如下：
0x82, 0xDC, 0xBA, 0xFB, 0xDE, 0xAB, 0x66, 0x02
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
0x95, 0xF8, 0xA5, 0xE5, 0xDD, 0x31, 0xD9, 0x00
0xDD, 0x7F, 0x12, 0x1C, 0xA5, 0x01, 0x56, 0x19
0x2E, 0x86, 0x53, 0x10, 0x4F, 0x38, 0x34, 0xEA
0x4B, 0xD3, 0x88, 0xFF, 0x6C, 0xD8, 0x1D, 0x4F
0x20, 0xB9, 0xE7, 0x67, 0xB2, 0xFB, 0x14, 0x56
0x55, 0x57, 0x93, 0x80, 0xD7, 0x71, 0x38, 0xEF
0x6C, 0xC5, 0xDE, 0xFA, 0xAF, 0x04, 0x51, 0x2F
0x0D, 0x9F, 0x27, 0x9B, 0xA5, 0xD8, 0x72, 0x60
0xD9, 0x03, 0x1B, 0x02, 0x71, 0xBD, 0x5A, 0x0A
```

直接将它与实验已经提供的 out 进行对比，可以看到结果一致，算法实现正确：

```
{ 0x82, 0xDC, 0xBA, 0xFB, 0xDE, 0xAB, 0x66, 0x02 } },
{ 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } },
{ 0x95, 0xF8, 0xA5, 0xE5, 0xDD, 0x31, 0xD9, 0x00 } },
{ 0xDD, 0x7F, 0x12, 0x1C, 0xA5, 0x01, 0x56, 0x19 } },
{ 0x2E, 0x86, 0x53, 0x10, 0x4F, 0x38, 0x34, 0xEA } },
{ 0x4B, 0xD3, 0x88, 0xFF, 0x6C, 0xD8, 0x1D, 0x4F } },
{ 0x20, 0xB9, 0xE7, 0x67, 0xB2, 0xFB, 0x14, 0x56 } },
{ 0x55, 0x57, 0x93, 0x80, 0xD7, 0x71, 0x38, 0xEF } },
{ 0x6C, 0xC5, 0xDE, 0xFA, 0xAF, 0x04, 0x51, 0x2F } },
{ 0x0D, 0x9F, 0x27, 0x9B, 0xA5, 0xD8, 0x72, 0x60 } },
{ 0xD9, 0x03, 0x1B, 0x02, 0x71, 0xBD, 0x5A, 0x0A } }
```

接下来进行雪崩效应的测试验证。

首先是固定密钥、修改明文的密文雪崩效应：

Microsoft Visual Studio 调试控制台

```
第0组的明文第4位取反。
0x9B, 0xDA, 0x67, 0x3F, 0x14, 0xF6, 0x65, 0xAF
第0组的明文第26位取反。
0x0D, 0xD2, 0xF9, 0xF8, 0x8A, 0x5E, 0xAA, 0x82
第0组的明文第5位取反。
0x02, 0x8B, 0xD8, 0x1D, 0x9C, 0x49, 0x67, 0x88
第0组的明文第48位取反。
0xD9, 0x7F, 0x59, 0x04, 0xCD, 0x24, 0xD2, 0x12
第0组的明文第30位取反。
0xBE, 0xF3, 0xD8, 0x74, 0x65, 0x9D, 0x0C, 0xB8
第0组的明文第55位取反。
0xCE, 0x37, 0x7A, 0x9B, 0x04, 0xC1, 0x32, 0x6D
第0组的明文第48位取反。
0xD9, 0x7F, 0x59, 0x04, 0xCD, 0x24, 0xD2, 0x12
第0组的明文第40位取反。
0x3E, 0xB0, 0x95, 0x95, 0xDF, 0x8F, 0x21, 0x73
第0组密文改变的平均位数是： 31
```

可以看到，这一组在 8 次测试当中的密文平均改变位数是 31。

然后对整个实验提供的所有十个加密组别都进行明文变换测试，一次性输出最终的变化结果：

```
第0组密文改变的平均位数是： 30
第1组密文改变的平均位数是： 33
第2组密文改变的平均位数是： 32
第3组密文改变的平均位数是： 32
第4组密文改变的平均位数是： 32
第5组密文改变的平均位数是： 29
第6组密文改变的平均位数是： 30
第7组密文改变的平均位数是： 32
第8组密文改变的平均位数是： 31
第9组密文改变的平均位数是： 33
```

密文改变的平均位数是 $(30+33+32+32+32+29+30+32+31+33)/10 = 31.4$ 位

然后是修改密钥的雪崩效应：

C:\Users\Lenovo\source\repos\cipher\Debug\cipher.exe

```
第0组的密钥第25位取反。
0x67, 0x54, 0xA5, 0x5A, 0xD8, 0x77, 0x34, 0x59
第0组的密钥第38位取反。
0x1C, 0xD6, 0x21, 0x45, 0x98, 0x17, 0xCA, 0x4D
第0组的密钥第27位取反。
0x46, 0xF6, 0xE1, 0x37, 0x41, 0x83, 0xED, 0xBB
第0组的密钥第14位取反。
0x34, 0x56, 0x5D, 0x51, 0xA1, 0xE2, 0x1A, 0x15
第0组的密钥第40位取反。
0x5A, 0x6C, 0x35, 0x0B, 0x2F, 0x05, 0xD8, 0x93
第0组的密钥第11位取反。
0x01, 0x26, 0x92, 0x26, 0x27, 0xB2, 0x0D, 0x1D
第0组的密钥第23位取反。
0x82, 0xDC, 0xBA, 0xFB, 0xDE, 0xAB, 0x66, 0x02
第0组的密钥第21位取反。
0xE1, 0xD4, 0xC9, 0xDC, 0x49, 0x32, 0x0E, 0xD4
第0组密文改变的平均位数是： 29
```

然后对全部组别进行验证：

Microsoft Visual Studio 调试控制台

```
第0组密文改变的平均位数是： 32
第1组密文改变的平均位数是： 31
第2组密文改变的平均位数是： 33
第3组密文改变的平均位数是： 29
第4组密文改变的平均位数是： 30
第5组密文改变的平均位数是： 31
第6组密文改变的平均位数是： 30
第7组密文改变的平均位数是： 31
第8组密文改变的平均位数是： 31
第9组密文改变的平均位数是： 29
```

最终的平均密文改变位数是 $(32+31+33+29+30+31+30+31+31+29)/10 = 30.7$ 位。

从上述实验结果我们可以看到，对明文或密钥只改动一位，就可以使得密文改动 32 位左右，占其总长度的一半，验证了雪崩效应。