

南 开 大 学

网络空间安全学院

密码学课程报告

|  |
| --- |
| 第二次实验报告  ——分组密码算法DES |

学号： **1611519**

姓名： 周子祎

年级： 2016 级

专业： 信息安全-法学

2018 年 12 月 13 日

**密码学第二次实验报告**

——分组密码算法DES

1. 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。

1. 实验原理
2. 整体流程

明文分组长度为64位，秘钥长度为56位，最终形成的密文长度为64位。如果明文长度不足64位，即将其扩展为64位（如补零等方法）。

具体加密过程：

1. 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；
2. 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；
3. 如此循环16轮，最后得到L16、R16；
4. 再对L16、R16实行逆初始置换IP－1，即可得到加密数据。解密过程与此类似，不同之处仅在于子密钥的使用顺序正好相反；

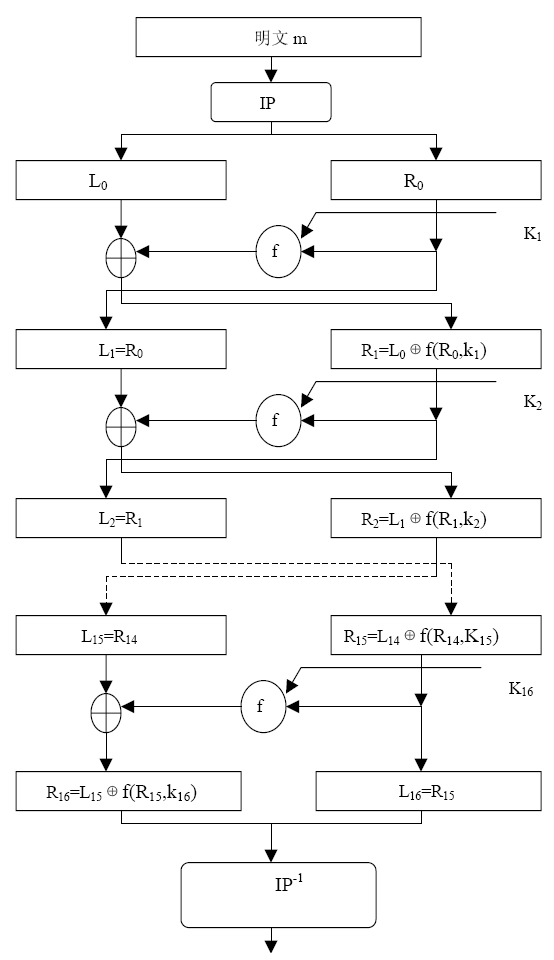


图1－1 DES加密/解密流程

1. 基本函数

DES的加密算法包括3个基本函数：

1. **初始置换IP**

它的作用是**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位换到第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。

58 50 42 34 26 18 10 02

60 52 44 36 28 20 12 04

62 54 46 38 30 22 14 06

64 56 48 40 32 24 16 08

57 49 41 33 25 17 09 01

59 51 43 35 27 19 11 03

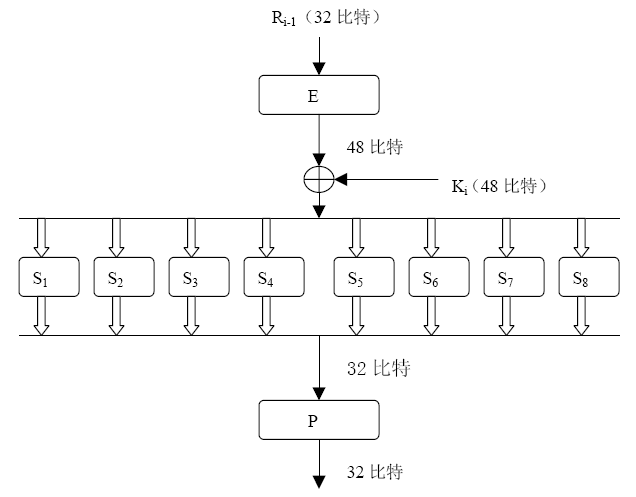
61 53 45 37 29 21 13 05

63 55 47 39 31 23 15 07

R0和K1经过f（R0，K1）变换后的输出结果，再和L0进行异或运算，输出结果位R1，R0则赋给L1。L1和R1同样再做类似运算生成L2和R2，…，**经过16次运算后生成L16和R16**。

1. **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。

****

**图1－2 F函数原理图**

1. **逆初始置换IP－1**

它将L16和R16作为输入，进行逆初始置换得到密文输出。**逆初始置换是初始**

**置换的逆运算**，置换规则如下所列：

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

1. **子秘钥生成模块**

DES的加密算法中除了上面介绍的3个基本函数，还有一个非常重要的功能模块，即子密钥的生成模块，具体子密钥的产生流程图如图1－3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**，不参与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**平分为两部分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**。

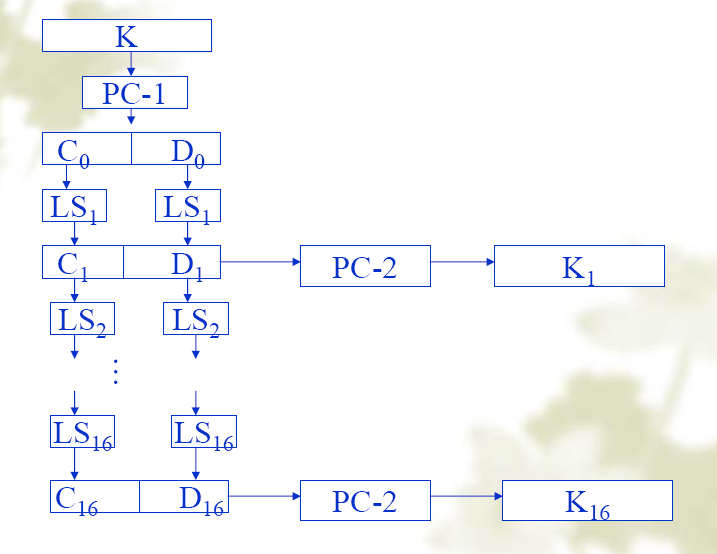


图1－3 子密钥的生成流程

表1－1 左移位数规则

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| LSi | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

表1－2 压缩置换PC－1

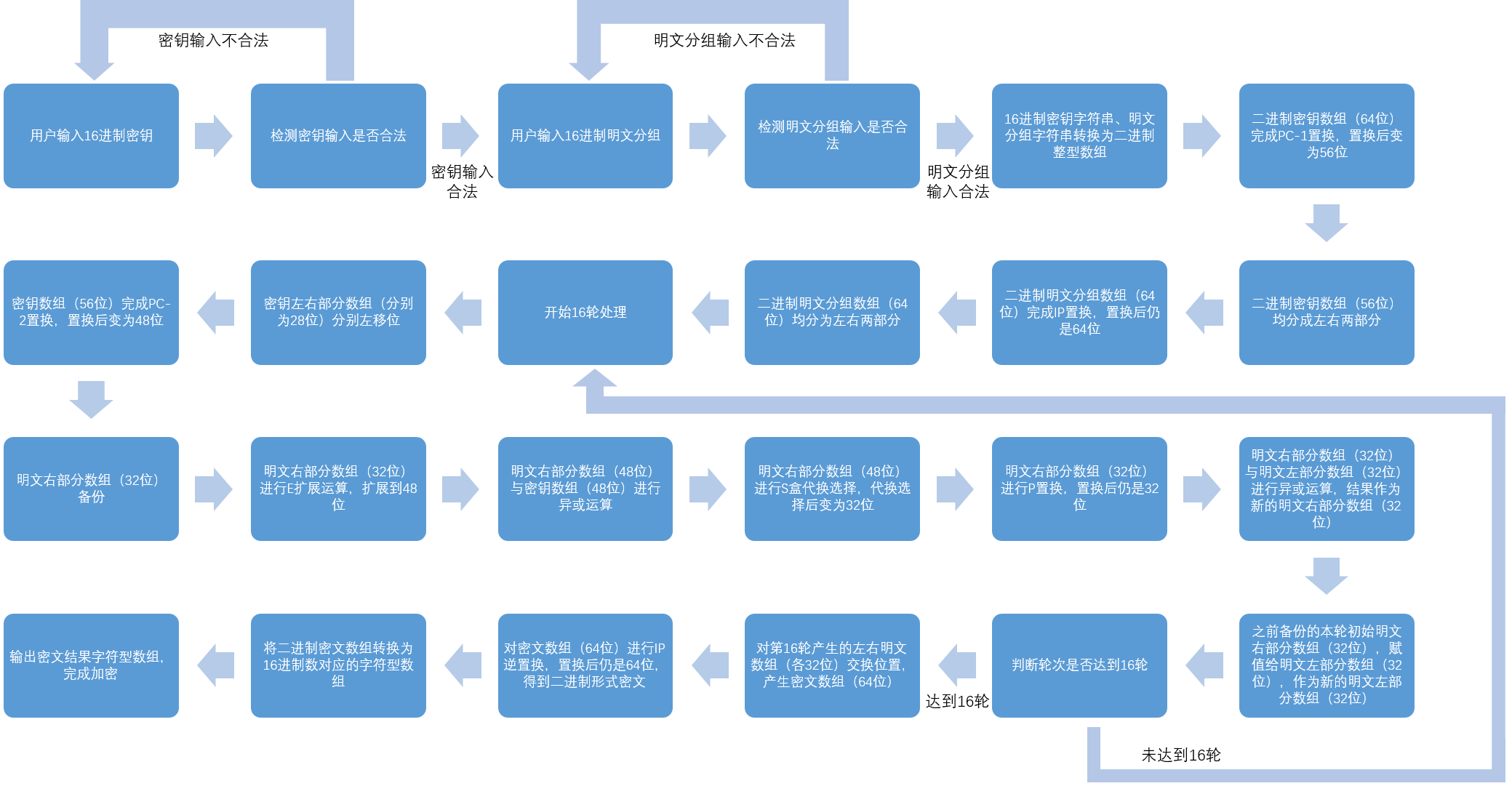
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

表1－3 压缩置换PC－2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | 3 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

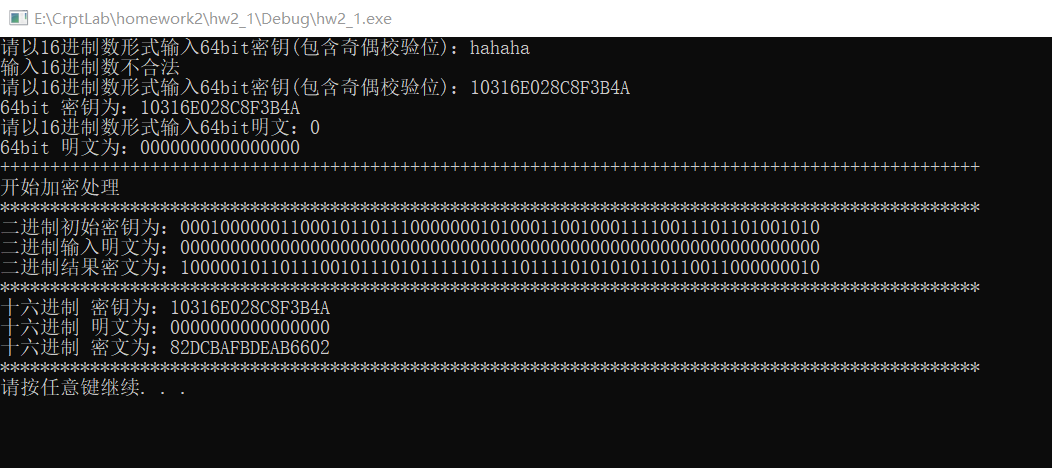
1. 实验要求
2. DES实现程序的总体设计：在第一步的基础上，对整个**DES加密函数**的实现进行**总体设计**，考虑数据的存储格式，参数的传递格式，程序实现的总体层次等，**画出程序实现的流程图**。
3. 在总体设计完成后，开始**具体的编码**，在编码过程中，注意要尽量使用高效的编码方式。
4. 对DES的密文进行雪崩效应检验。即固定密钥，仅**改变明文中的一位**，统计密文改变的位数；固定明文，仅**改变密钥中的一位**，统计密文改变的位数。
5. 实验内容
6. **DES加密程序整体流程图**

（**图片png文件已经单独附在压缩包中上交**，如查看本图，建议放大查看）

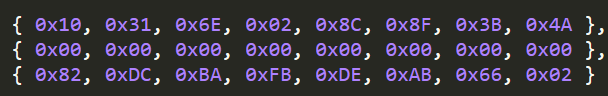


1. **DES程序实现结果**
2. **加密过程程序实现结果**

（相关程序文件见工程文件hw2\_1）

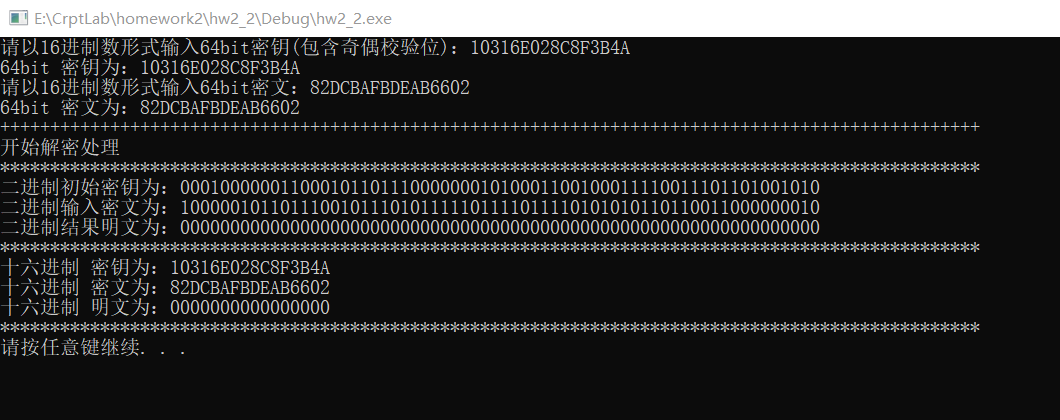


与所给定的测试数据结果相吻合：

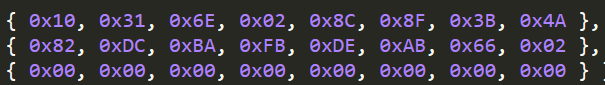


1. **解密过程程序实现结果**

（相关程序文件见工程文件hw2\_2）



与所给定的测试数据结果相吻合：



1. **DES程序实现代码**
2. **程序所使用的常量数据表**

相关代码见hw2\_1和hw2\_2工程文件中的hw2\_1\_data.h和hw2\_2\_data.h 头文件，此处不再展示。

其中定义了程序中要使用的**明文初始置换IP表**、**明文逆初始置换IP-1表**、**明文E扩展选择表**、**明文P换位表**、**密钥PC1选位表**、**密钥PC2选位表**、**左循环移位位数表**以及**S盒**。

1. **相关功能函数实现**

相关代码见hw2\_1和hw2\_2工程文件中的hw2\_1.h和hw2\_2.h 头文件。

**以hw2\_1工程文件中的hw2\_1.h头文件为例：**

#include <iostream>

#include <string>

#include "hw2\_1\_data.h"

using namespace std;

char \* get\_Key(string String\_Key\_Input); //将用户输入String 转换为 char数组Char\_Key

char \* get\_PlainText(string String\_PlainText\_Input); //将用户输入String 转换为 char数组Char\_PlainText

bool is\_input\_valid(char \* str);//检查用户输入16进制数是否合法

void Hex\_to\_Bin(char \* Hex, int \* Bin);//十六进制char数组转换为二进制int数组

void Bin\_to\_Hex(int \* Bin, char \* Hex);//二进制int数组转换为十六进制char数组

//将用户输入String 转换为 char数组Char\_Key

char \* get\_Key(string String\_Key\_Input)

{

    //截取前16位十六进制数作为密钥

    string String\_Key = String\_Key\_Input.substr(0,16);

    //不足16位十六进制数则末尾补0至16位十六进制数

    if(String\_Key.size()<16)

    {

        int temp\_String\_Key\_size = String\_Key.size();

        for(int i=1;i<=16 - temp\_String\_Key\_size;i++)

        {

            String\_Key = String\_Key + '0';

        }

    }

    char\* Char\_Key = new char[17];

    String\_Key.copy(Char\_Key, 16, 0);

    Char\_Key[16] = '\0';

    return Char\_Key;

}

//将用户输入String 转换为 char数组Char\_PlainText

char \* get\_PlainText(string String\_PlainText\_Input)

{

    //截取前16位十六进制数作为明文

    string String\_PlainText = String\_PlainText\_Input.substr(0,16);

    //不足16位十六进制数则末尾补0至16位十六进制数

    if(String\_PlainText.size()<16)

    {

        int temp\_String\_PlainText\_size = String\_PlainText.size();

        for(int i=1;i<=16 - temp\_String\_PlainText\_size;i++)

        {

            String\_PlainText = String\_PlainText + '0';

        }

    }

    char\* Char\_PlainText = new char[17];

    String\_PlainText.copy(Char\_PlainText, 16, 0);

    Char\_PlainText[16] = '\0';

    return Char\_PlainText;

}

//检查用户输入16进制数是否合法

bool is\_input\_valid(char \* str)

{

    for(int i=0;i<=15;i++)

    {

        if(!( (str[i]>='A'&&str[i]<='F') || (str[i]>='a'&&str[i]<='f') ||(str[i]>='0'&&str[i]<='9') ))

        {

            return 0;

        }

    }

    return 1;

}

//十六进制char数组转换为二进制int数组

void Hex\_to\_Bin(char \* Hex, int \* Bin)

{

    int i=0,k=0,j=0;

    for(i=0;i<16;i++)

    {

        int val = 0;

        if(Hex[i] >= '0' && Hex[i] <= '9')

            val = Hex[i]-'0';

        else if (Hex[i] >= 'A' && Hex[i] <= 'F')

            val = Hex[i]-'A'+10;

        else if (Hex[i] >= 'a' && Hex[i] <= 'f')

            val = Hex[i]-'a'+10;

        for(int k=4\*i+3;k>=4\*i;k--)

        {

            Bin[k] = val % 2;

            val/=2;

        }

    }

}

//二进制int数组转换为十六进制char数组

void Bin\_to\_Hex(int \* Bin, char \* Hex)

{

    int i=0,k=0,j=0;

    for(i=0;i<16;i++)

    {

        int val = 0;

        for(int k=4\*i;k<=4\*i+3;k++)

        {

            val \*= 2;

            val += Bin[k];

        }

        if( val<10 )

            Hex[i] = val+'0';

        else if (val >=10 && val <=15 )

            Hex[i] = val-10+'A';

    }

    Hex[16]='\0';

}

1. **加密过程函数实现**

相关代码见hw2\_1工程文件中的hw2\_1.cpp 源文件。

#include "hw2\_1.h"

int main()

{

//用户输入密钥

    string String\_Key\_Input;

cout <<"请以16进制数形式输入64bit密钥(包含奇偶校验位)：" ;

cin >> String\_Key\_Input;

    //根据用户输入的string获得char数组

    char \* Char\_Key = get\_Key(String\_Key\_Input);

    //检查密钥输入是否合法

    while(true)

    {

        if(is\_input\_valid(Char\_Key))

        {

            cout<<"64bit 密钥为："<< Char\_Key<<endl;

            break;

        }

        else

        {

            cout<<"输入16进制数不合法"<<endl;

            cout <<"请以16进制数形式输入64bit密钥(包含奇偶校验位)：" ;

            cin >> String\_Key\_Input;

            Char\_Key = get\_Key(String\_Key\_Input);

        }

    }

    //用户输入明文

    string String\_PlainText\_Input;

cout <<"请以16进制数形式输入64bit明文：" ;

cin >> String\_PlainText\_Input;

    //根据用户输入的string获得char数组

    char \* Char\_PlainText = get\_PlainText(String\_PlainText\_Input);

    //检查明文输入是否合法

    while(true)

    {

        if(is\_input\_valid(Char\_PlainText))

        {

            cout<<"64bit 明文为："<< Char\_PlainText<<endl;

            break;

        }

        else

        {

            cout<<"输入16进制数不合法"<<endl;

            cout <<"请以16进制数形式输入64bit明文：" ;

            cin >> String\_PlainText\_Input;

            Char\_PlainText = get\_PlainText(String\_PlainText\_Input);

        }

    }

    cout<<"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"<<endl;

    cout<<"开始加密处理"<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    //用于保存用户输入的64bit二进制密钥

    int Bit\_Key\_Init[64];

    //将输入的十六进制密钥char数组转换为二进制int数组

    Hex\_to\_Bin(Char\_Key, Bit\_Key\_Init);

    cout<<"二进制初始密钥为：";

    for(int i=0;i<64;i++)

        cout<<Bit\_Key\_Init[i];

    cout<<endl;

    //用于保存用户输入的64bit二进制明文

    int Bit\_PlainText\_Init[64];

    //将输入的十六进制明文char数组转换为二进制int数组

    Hex\_to\_Bin(Char\_PlainText, Bit\_PlainText\_Init);

    cout<<"二进制输入明文为：";

    for(int i=0;i<64;i++)

        cout<<Bit\_PlainText\_Init[i];

    cout<<endl;

    //PC-1置换后的二进制密钥数组(56bit)

    int Bit\_Key\_aft\_PC1[56];

    //密钥完成初始置换PC-1

    for(int m=0;m<56;m++)

    {

        Bit\_Key\_aft\_PC1[m]=Bit\_Key\_Init[PC1\_Table[m]-1];

    }

    //把密钥分为左右两半

    int Bit\_Key\_l1[28],Bit\_Key\_r1[28];

    for(int m=0;m<56;m++)

    {

        if(m<28)

            Bit\_Key\_l1[m]=Bit\_Key\_aft\_PC1[m];

        else

            Bit\_Key\_r1[m-28]=Bit\_Key\_aft\_PC1[m];

    }

    //IP置换后的二进制明文数组(64bit)

    int Bit\_PlainText\_aft\_IP[64];

    //明文完成初始IP置换

    for(int m=0;m<64;m++)

    {

        Bit\_PlainText\_aft\_IP[m]=Bit\_PlainText\_Init[IP\_Table[m]-1];

    }

    //把明文分为左右两半

    int Bit\_PlainText\_l1[32],Bit\_PlainText\_r1[32];

    for(int m=0;m<64;m++)

    {

        if(m<32)

            Bit\_PlainText\_l1[m]=Bit\_PlainText\_aft\_IP[m];

        else

            Bit\_PlainText\_r1[m-32]=Bit\_PlainText\_aft\_IP[m];

    }

    //存储每轮子密钥用于解密

    int Round\_Key\_Store[16][48];

    //存储每轮移位后的左右密钥

    int Bit\_Key\_l2[28],Bit\_Key\_r2[28];

    //存储左右明文

    int Bit\_PlainText\_l2[32],Bit\_PlainText\_r2[32];

    int Bit\_PlainText\_l3[32],Bit\_PlainText\_r3[48];

    //16轮次结构

    for(int round=0;round<16;round++)

    {

        //密钥左移位

        for(int m=0;m<28;m++)

        {

            Bit\_Key\_l2[m]=Bit\_Key\_l1[(m+LOOP\_Table[round])%28];

            Bit\_Key\_r2[m]=Bit\_Key\_r1[(m+LOOP\_Table[round])%28];

        }

        //更新左右密钥，用于下次循环使用

        for(int m=0;m<28;m++)

        {

            Bit\_Key\_l1[m]=Bit\_Key\_l2[m];

            Bit\_Key\_r1[m]=Bit\_Key\_r2[m];

        }

        //密钥完成PC-2置换

        for(int m=0;m<48;m++)

        {

            if(PC2\_Table[m]<29)

            {

                Round\_Key\_Store[round][m]=Bit\_Key\_l2[PC2\_Table[m]-1];

            }

            else

            {

                Round\_Key\_Store[round][m]=Bit\_Key\_r2[PC2\_Table[m]-29];

            }

        }

        //对右明文备份

        for(int m=0;m<32;m++)

        {

            Bit\_PlainText\_r2[m]=Bit\_PlainText\_r1[m];

        }

        //对右明文进行E扩展运算

        for(int m=0;m<48;m++)

        {

            Bit\_PlainText\_r3[m]=Bit\_PlainText\_r1[E\_Table[m]-1];

        }

        //右明文与密钥做异或运算

        for(int m=0;m<48;m++)

        {

            Bit\_PlainText\_r3[m]=Bit\_PlainText\_r3[m]^Round\_Key\_Store[round][m];

        }

        //右明文S盒代换

        for(int m=0;m<8;m++)

        {

            //根据右明文的值从S\_Box中选出值

            int s;

            s=S\_Box[(Bit\_PlainText\_r3[m\*6]\*2+Bit\_PlainText\_r3[5+m\*6])+m\*4][Bit\_PlainText\_r3[1+m\*6]\*8+Bit\_PlainText\_r3[2+m\*6]\*4+Bit\_PlainText\_r3[3+m\*6]\*2+Bit\_PlainText\_r3[4+m\*6]];

            //将S\_Box中选中的值表示为二进制 并 赋值到Bit\_PlainText\_r3中

            for(int n=1;n<=4;n++)

            {

                Bit\_PlainText\_r3[(4-n)+4\*m]=s%2;

                s=s/2;

            }

        }

        //右明文完成P置换,与左明文异或,并将备份的本轮原始右明文传给左明文，用于下次循环使用

        for(int m=0;m<32;m++)

        {

            Bit\_PlainText\_r1[m]=Bit\_PlainText\_r3[P\_Table[m]-1]^Bit\_PlainText\_l1[m];

            Bit\_PlainText\_l1[m]=Bit\_PlainText\_r2[m];

        }

    }

    //逆初始置换IPR置换前的二进制密文数组(64bit)

    int Bit\_CipherText\_bfr\_IPR[64];

    //16轮后明文左右交换形成密文

    for(int m=0;m<64;m++)

    {

        if(m<32)

            Bit\_CipherText\_bfr\_IPR[m]=Bit\_PlainText\_r1[m];

        else

            Bit\_CipherText\_bfr\_IPR[m]=Bit\_PlainText\_l1[m-32];

    }

    //逆初始置换IPR置换后的二进制密文数组(64bit)

    int Bit\_CipherText\_aft\_IPR[64];

    //密文完成逆初始置换

    for(int m=0;m<64;m++)

    {

        Bit\_CipherText\_aft\_IPR[m]=Bit\_CipherText\_bfr\_IPR[IPR\_Table[m]-1];

    }

    cout<<"二进制结果密文为：";

    for(int m=0;m<64;m++)

    {

        cout<<Bit\_CipherText\_aft\_IPR[m];

    }

    cout<<endl;

    //存储密文十六进制char数组

    char Char\_CipherText[17];

    //将得到的二进制密文int数组转化为十六进制char数组

    Bin\_to\_Hex(Bit\_CipherText\_aft\_IPR, Char\_CipherText);

    //输出16进制密钥、明文、密文，方便对比结果

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    //输出密钥

    cout<<"十六进制 密钥为："<<Char\_Key<<endl;

    //输出明文

    cout<<"十六进制 明文为："<<Char\_PlainText<<endl;

    //输出密文

    cout<<"十六进制 密文为："<<Char\_CipherText<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

system("pause");

return 0;

}

1. **解密过程函数实现**

相关代码见hw2\_2工程文件中的hw2\_2.cpp 源文件。

#include "hw2\_2.h"

int main()

{

//用户输入密钥

    string String\_Key\_Input;

cout <<"请以16进制数形式输入64bit密钥(包含奇偶校验位)：" ;

cin >> String\_Key\_Input;

    //根据用户输入的string获得char数组

    char \* Char\_Key = get\_Key(String\_Key\_Input);

    //检查密钥输入是否合法

    while(true)

    {

        if(is\_input\_valid(Char\_Key))

        {

            cout<<"64bit 密钥为："<< Char\_Key<<endl;

            break;

        }

        else

        {

            cout<<"输入16进制数不合法"<<endl;

            cout <<"请以16进制数形式输入64bit密钥(包含奇偶校验位)：" ;

            cin >> String\_Key\_Input;

            Char\_Key = get\_Key(String\_Key\_Input);

        }

    }

    //用户输入密文

    string String\_CipherText\_Input;

cout <<"请以16进制数形式输入64bit密文：" ;

cin >> String\_CipherText\_Input;

    //根据用户输入的string获得char数组

    char \* Char\_CipherText = get\_CipherText(String\_CipherText\_Input);

    //检查密文输入是否合法

    while(true)

    {

        if(is\_input\_valid(Char\_CipherText))

        {

            cout<<"64bit 密文为："<< Char\_CipherText<<endl;

            break;

        }

        else

        {

            cout<<"输入16进制数不合法"<<endl;

            cout <<"请以16进制数形式输入64bit密文：" ;

            cin >> String\_CipherText\_Input;

            Char\_CipherText = get\_CipherText(String\_CipherText\_Input);

        }

    }

    cout<<"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"<<endl;

    cout<<"开始解密处理"<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    //用于保存用户输入的64bit二进制密钥

    int Bit\_Key\_Init[64];

    //将输入的十六进制密钥char数组转换为二进制int数组

    Hex\_to\_Bin(Char\_Key, Bit\_Key\_Init);

    cout<<"二进制初始密钥为：";

    for(int i=0;i<64;i++)

        cout<<Bit\_Key\_Init[i];

    cout<<endl;

    //用于保存用户输入的64bit二进制密文

    int Bit\_CipherText\_Init[64];

    //将输入的十六进制密文char数组转换为二进制int数组

    Hex\_to\_Bin(Char\_CipherText, Bit\_CipherText\_Init);

    cout<<"二进制输入密文为：";

    for(int i=0;i<64;i++)

        cout<<Bit\_CipherText\_Init[i];

    cout<<endl;

    //-----------------子密钥获取------------------------------

    //先要获得16轮子密钥

    //PC-1置换后的二进制密钥数组(56bit)

    int Bit\_Key\_aft\_PC1[56];

    //密钥完成初始置换PC-1

    for(int m=0;m<56;m++)

    {

        Bit\_Key\_aft\_PC1[m]=Bit\_Key\_Init[PC1\_Table[m]-1];

    }

    //把密钥分为左右两半

    int Bit\_Key\_l1[28],Bit\_Key\_r1[28];

    for(int m=0;m<56;m++)

    {

        if(m<28)

            Bit\_Key\_l1[m]=Bit\_Key\_aft\_PC1[m];

        else

            Bit\_Key\_r1[m-28]=Bit\_Key\_aft\_PC1[m];

    }

    //存储每轮子密钥用于解密

    int Round\_Key\_Store[16][48];

    //存储每轮移位后的左右密钥

    int Bit\_Key\_l2[28],Bit\_Key\_r2[28];

    //16轮次结构

    for(int round=0;round<16;round++)

    {

        //密钥左移位

        for(int m=0;m<28;m++)

        {

            Bit\_Key\_l2[m]=Bit\_Key\_l1[(m+LOOP\_Table[round])%28];

            Bit\_Key\_r2[m]=Bit\_Key\_r1[(m+LOOP\_Table[round])%28];

        }

        //更新左右密钥，用于下次循环使用

        for(int m=0;m<28;m++)

        {

            Bit\_Key\_l1[m]=Bit\_Key\_l2[m];

            Bit\_Key\_r1[m]=Bit\_Key\_r2[m];

        }

        //密钥完成PC-2置换

        for(int m=0;m<48;m++)

        {

            if(PC2\_Table[m]<29)

            {

                Round\_Key\_Store[round][m]=Bit\_Key\_l2[PC2\_Table[m]-1];

            }

            else

            {

                Round\_Key\_Store[round][m]=Bit\_Key\_r2[PC2\_Table[m]-29];

            }

        }

    }

    //-----------------子密钥获取结束--------------------------

    //IP置换后的二进制密文数组(64bit)

    int Bit\_CipherText\_aft\_IP[64];

    //密文完成初始IP置换

    for(int m=0;m<64;m++)

    {

        Bit\_CipherText\_aft\_IP[m]=Bit\_CipherText\_Init[IP\_Table[m]-1];

    }

    //把密文分为左右两半

    int Bit\_CipherText\_l1[32],Bit\_CipherText\_r1[32];

    for(int m=0;m<64;m++)

    {

        if(m<32)

            Bit\_CipherText\_l1[m]=Bit\_CipherText\_aft\_IP[m];

        else

            Bit\_CipherText\_r1[m-32]=Bit\_CipherText\_aft\_IP[m];

    }

    //存储左右明文

    int Bit\_CipherText\_l2[32],Bit\_CipherText\_r2[32];

    int Bit\_CipherText\_l3[32],Bit\_CipherText\_r3[48];

    //16轮次结构

    for(int round=0;round<16;round++)

    {

        //对右明文备份

        for(int m=0;m<32;m++)

        {

            Bit\_CipherText\_r2[m]=Bit\_CipherText\_r1[m];

        }

        //对右明文进行E扩展运算

        for(int m=0;m<48;m++)

        {

            Bit\_CipherText\_r3[m]=Bit\_CipherText\_r1[E\_Table[m]-1];

        }

        //右明文与对应的轮密钥做异或运算

        for(int m=0;m<48;m++)

        {

            Bit\_CipherText\_r3[m]=Bit\_CipherText\_r3[m]^Round\_Key\_Store[15-round][m];

        }

        //右明文S盒代换

        for(int m=0;m<8;m++)

        {

            //根据右明文的值从S\_Box中选出值

            int s;

            s=S\_Box[(Bit\_CipherText\_r3[m\*6]\*2+Bit\_CipherText\_r3[5+m\*6])+m\*4][Bit\_CipherText\_r3[1+m\*6]\*8+Bit\_CipherText\_r3[2+m\*6]\*4+Bit\_CipherText\_r3[3+m\*6]\*2+Bit\_CipherText\_r3[4+m\*6]];

            //将S\_Box中选中的值表示为二进制 并 赋值到Bit\_PlainText\_r3中

            for(int n=1;n<=4;n++)

            {

                Bit\_CipherText\_r3[(4-n)+4\*m]=s%2;

                s=s/2;

            }

        }

        //右明文完成P置换,与左明文异或,并将备份的本轮原始右明文传给左明文，用于下次循环使用

        for(int m=0;m<32;m++)

        {

            Bit\_CipherText\_r1[m]=Bit\_CipherText\_r3[P\_Table[m]-1]^Bit\_CipherText\_l1[m];

            Bit\_CipherText\_l1[m]=Bit\_CipherText\_r2[m];

        }

    }

    //逆初始置换IPR置换前的二进制明文数组(64bit)

    int Bit\_PlainText\_bfr\_IPR[64];

    //16轮后密文左右交换得到明文

    for(int m=0;m<64;m++)

    {

        if(m<32)

            Bit\_PlainText\_bfr\_IPR[m]=Bit\_CipherText\_r1[m];

        else

            Bit\_PlainText\_bfr\_IPR[m]=Bit\_CipherText\_l1[m-32];

    }

    //逆初始置换IPR置换后的二进制明文数组(64bit)

    int Bit\_PlainText\_aft\_IPR[64];

    //明文完成逆初始置换

    for(int m=0;m<64;m++)

    {

        Bit\_PlainText\_aft\_IPR[m]=Bit\_PlainText\_bfr\_IPR[IPR\_Table[m]-1];

    }

    cout<<"二进制结果明文为：";

    for(int m=0;m<64;m++)

    {

        cout<<Bit\_PlainText\_aft\_IPR[m];

    }

    cout<<endl;

    //存储明文十六进制char数组

    char Char\_PlainText[17];

    //将得到的二进制明文int数组转化为十六进制char数组

    Bin\_to\_Hex(Bit\_PlainText\_aft\_IPR, Char\_PlainText);

    //输出16进制密钥、明文、密文，方便对比结果

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    //输出密钥

    cout<<"十六进制 密钥为："<<Char\_Key<<endl;

    //输出密文

    cout<<"十六进制 密文为："<<Char\_CipherText<<endl;

    //输出明文

    cout<<"十六进制 明文为："<<Char\_PlainText<<endl;

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

system("pause");

return 0;

}

//将用户输入String 转换为 char数组Char\_Key

char \* get\_Key(string String\_Key\_Input)

{

    //截取前16位十六进制数作为密钥

    string String\_Key = String\_Key\_Input.substr(0,16);

    //不足16位十六进制数则末尾补0至16位十六进制数

    if(String\_Key.size()<16)

    {

        int temp\_String\_Key\_size = String\_Key.size();

        for(int i=1;i<=16 - temp\_String\_Key\_size;i++)

        {

            String\_Key = String\_Key + '0';

        }

    }

    char\* Char\_Key = new char[17];

    String\_Key.copy(Char\_Key, 16, 0);

    Char\_Key[16] = '\0';

    return Char\_Key;

}

//将用户输入String 转换为 char数组Char\_CipherText

char \* get\_CipherText(string String\_CipherText\_Input)

{

    //截取前16位十六进制数作为明文

    string String\_CipherText = String\_CipherText\_Input.substr(0,16);

    //不足16位十六进制数则末尾补0至16位十六进制数

    if(String\_CipherText.size()<16)

    {

        int temp\_String\_CipherText\_size = String\_CipherText.size();

        for(int i=1;i<=16 - temp\_String\_CipherText\_size;i++)

        {

            String\_CipherText = String\_CipherText + '0';

        }

    }

    char\* Char\_CipherText = new char[17];

    String\_CipherText.copy(Char\_CipherText, 16, 0);

    Char\_CipherText[16] = '\0';

    return Char\_CipherText;

}

//检查用户输入16进制数是否合法

bool is\_input\_valid(char \* str)

{

    for(int i=0;i<=15;i++)

    {

        if(!( (str[i]>='A'&&str[i]<='F') || (str[i]>='a'&&str[i]<='f') ||(str[i]>='0'&&str[i]<='9') ))

        {

            return 0;

        }

    }

    return 1;

}

//十六进制char数组转换为二进制int数组

void Hex\_to\_Bin(char \* Hex, int \* Bin)

{

    int i=0,k=0,j=0;

    for(i=0;i<16;i++)

    {

        int val = 0;

        if(Hex[i] >= '0' && Hex[i] <= '9')

            val = Hex[i]-'0';

        else if (Hex[i] >= 'A' && Hex[i] <= 'F')

            val = Hex[i]-'A'+10;

        else if (Hex[i] >= 'a' && Hex[i] <= 'f')

            val = Hex[i]-'a'+10;

        for(int k=4\*i+3;k>=4\*i;k--)

        {

            Bin[k] = val % 2;

            val/=2;

        }

    }

}

//二进制int数组转换为十六进制char数组

void Bin\_to\_Hex(int \* Bin, char \* Hex)

{

    int i=0,k=0,j=0;

    for(i=0;i<16;i++)

    {

        int val = 0;

        for(int k=4\*i;k<=4\*i+3;k++)

        {

            val \*= 2;

            val += Bin[k];

        }

        if( val<10 )

            Hex[i] = val+'0';

        else if (val >=10 && val <=15 )

            Hex[i] = val-10+'A';

    }

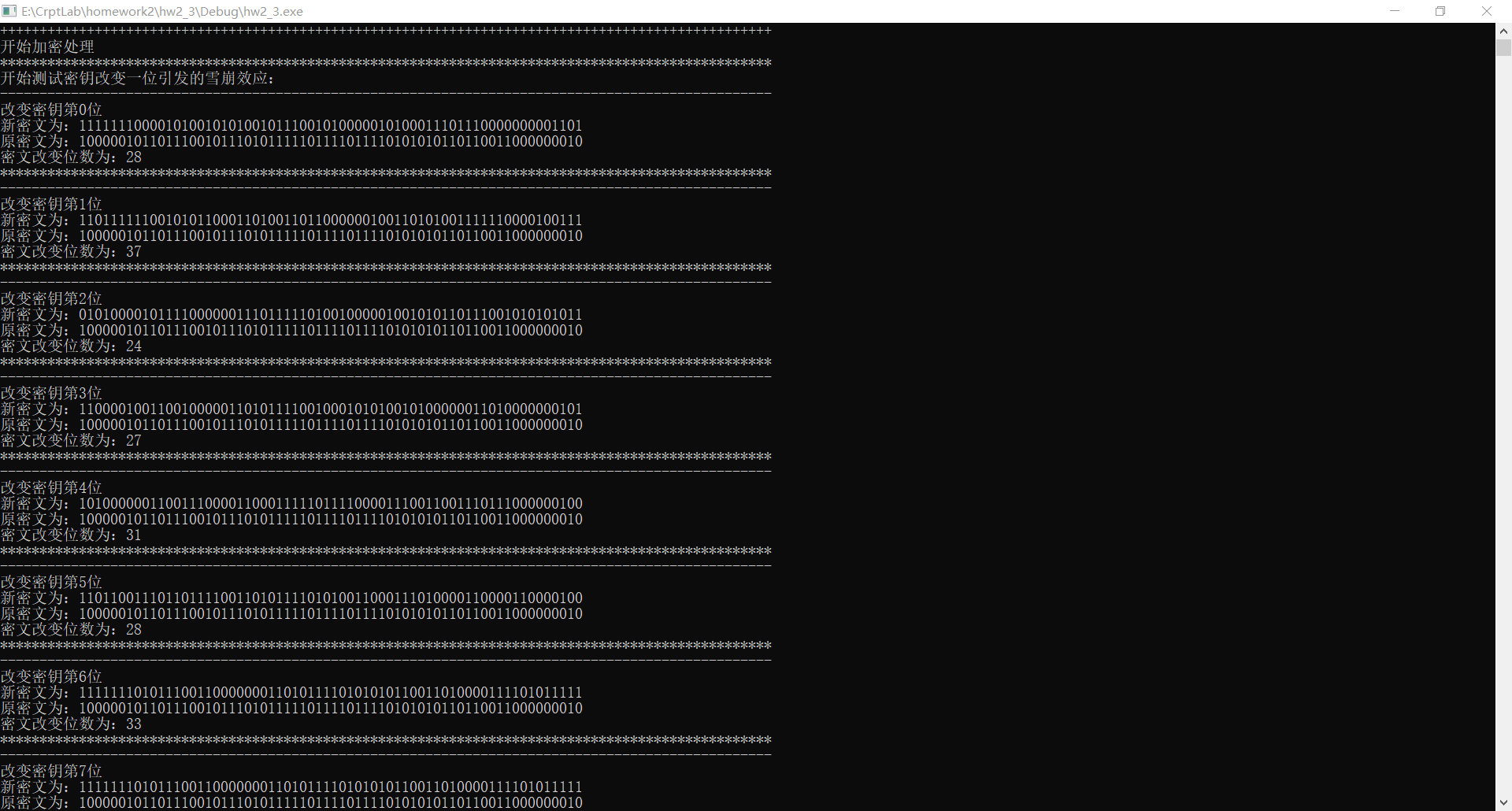
    Hex[16]='\0';

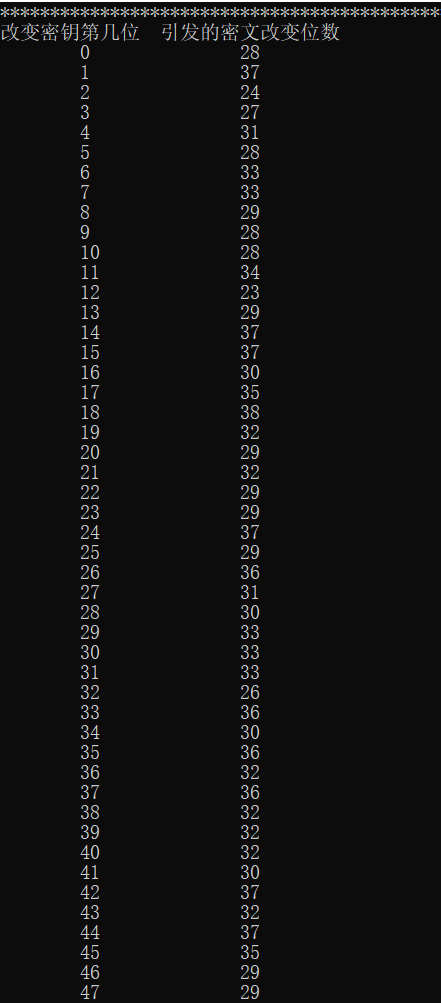
}

1. **雪崩效应测试代码及结果**
2. **测试密钥改变一位引发的密文改变的位数**

（相关程序文件及代码见工程文件hw2\_3）

测试程序运行结果如下：(可以放大查看)





**测试结果数据如下：**

改变密钥中的一位，引发密文改变的位数为：

改变密钥第几位 引发的密文改变位数

0 28

1 37

2 24

3 27

4 31

5 28

6 33

7 33

8 29

9 28

10 28

11 34

12 23

13 29

14 37

15 37

16 30

17 35

18 38

19 32

20 29

21 32

22 29

23 29

24 37

25 29

26 36

27 31

28 30

29 33

30 33

31 33

改变密钥第几位 引发的密文改变位数

32 26

33 36

34 30

35 36

36 32

37 36

38 32

39 32

40 32

41 30

42 37

43 32

44 37

45 35

46 29

47 29

48 31

49 38

50 26

51 34

52 35

53 37

54 32

55 32

56 34

57 28

58 29

59 31

60 30

61 34

62 30

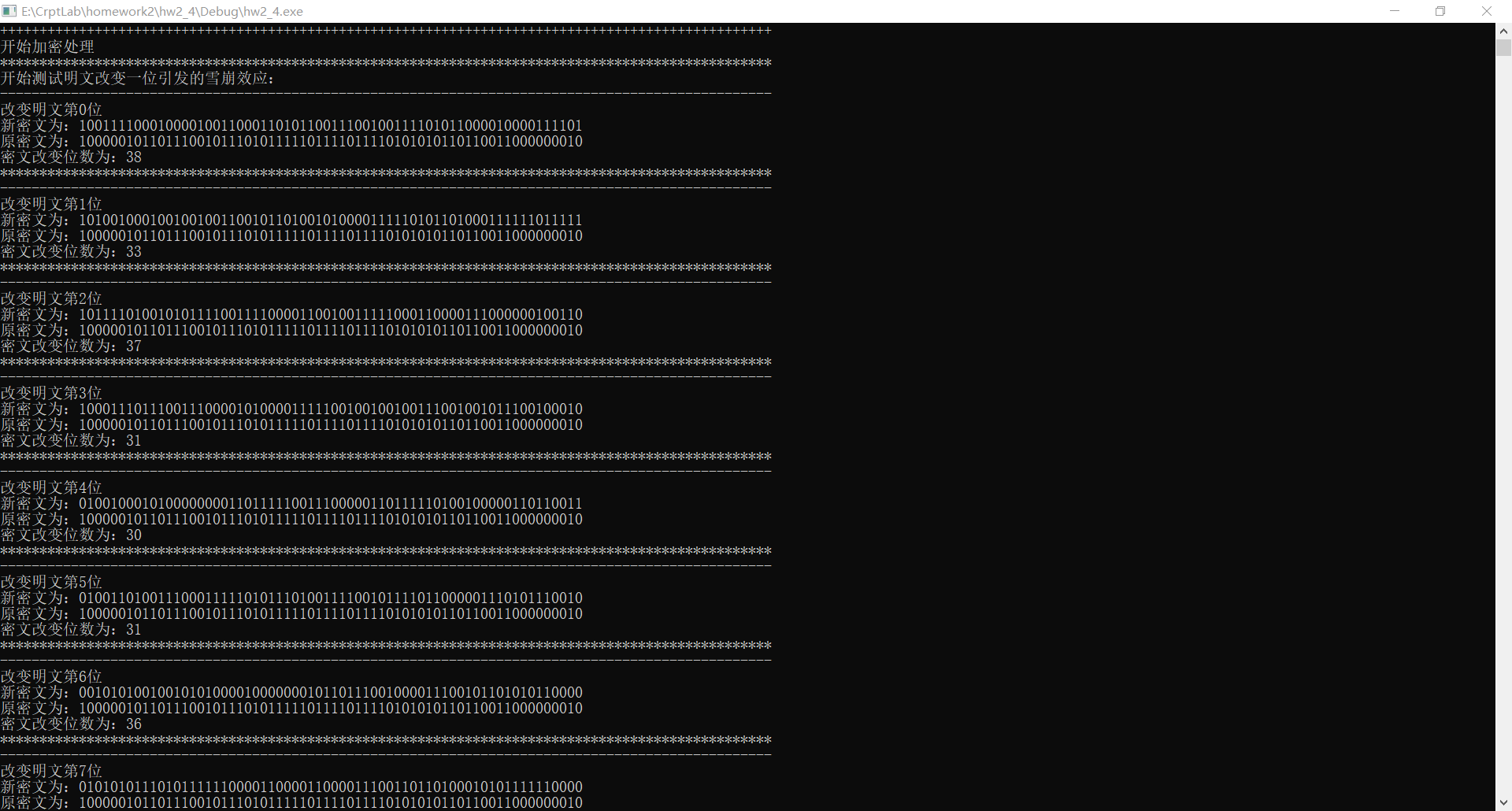
63 30

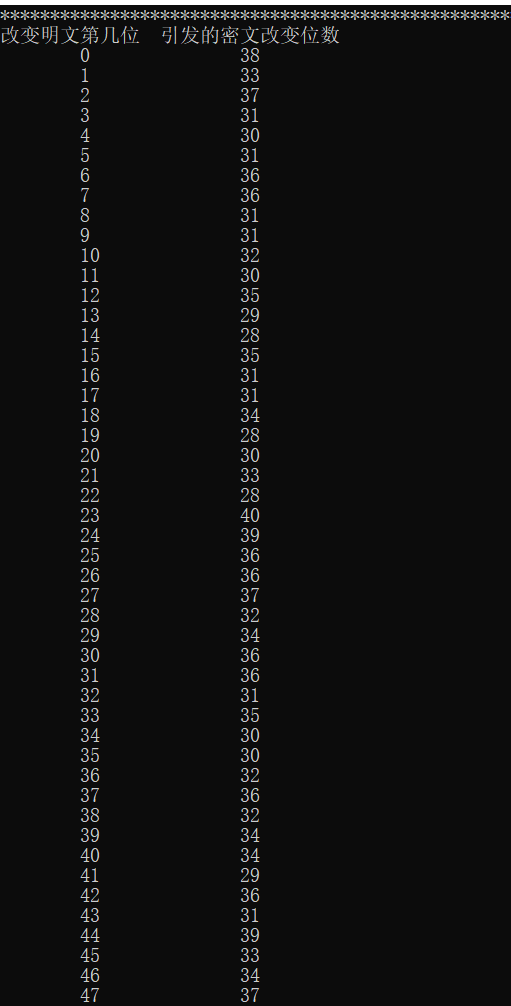
计算平均值，得到密钥改变一位，引发的密文改变的平均位数为：31.7813 位；该值接近32，说明**密钥中改变一位，将使密文中约一半的位的值发生改变**。

1. **测试明文改变一位引发的密文改变的位数**

（相关程序文件及代码见工程文件hw2\_4）

测试程序运行结果如下：(可以放大查看)





**测试结果数据如下：**

改变明文中的一位，引发明文改变的位数为：

改变明文第几位 引发的密文改变位数

0 38

1 33

2 37

3 31

4 30

5 31

6 36

7 36

8 31

9 31

10 32

11 30

12 35

13 29

14 28

15 35

16 31

17 31

18 34

19 28

20 30

21 33

22 28

23 40

24 39

25 36

26 36

27 37

28 32

29 34

30 36

31 36

改变明文第几位 引发的密文改变位数

32 31

33 35

34 30

35 30

36 32

37 36

38 32

39 34

40 34

41 29

42 36

43 31

44 39

45 33

46 34

47 37

48 35

49 32

50 33

51 36

52 27

53 35

54 25

55 29

56 27

57 36

58 25

59 34

60 35

61 39

62 31

63 34

计算平均值，得到明文改变一位，引发的密文改变的平均位数为：32.9688位；该值接近32，说明**明文中改变一位，将使密文中约一半的位的值发生改变**。