实 验 报 告

EXPERIMENT REPORT

姓名 贺验权

学号 2021307220416

专业 计算机科学与技术

教师 徐士伟

科目 信息安全

信 息 学 院

COLLEGE OF INFORMATIC

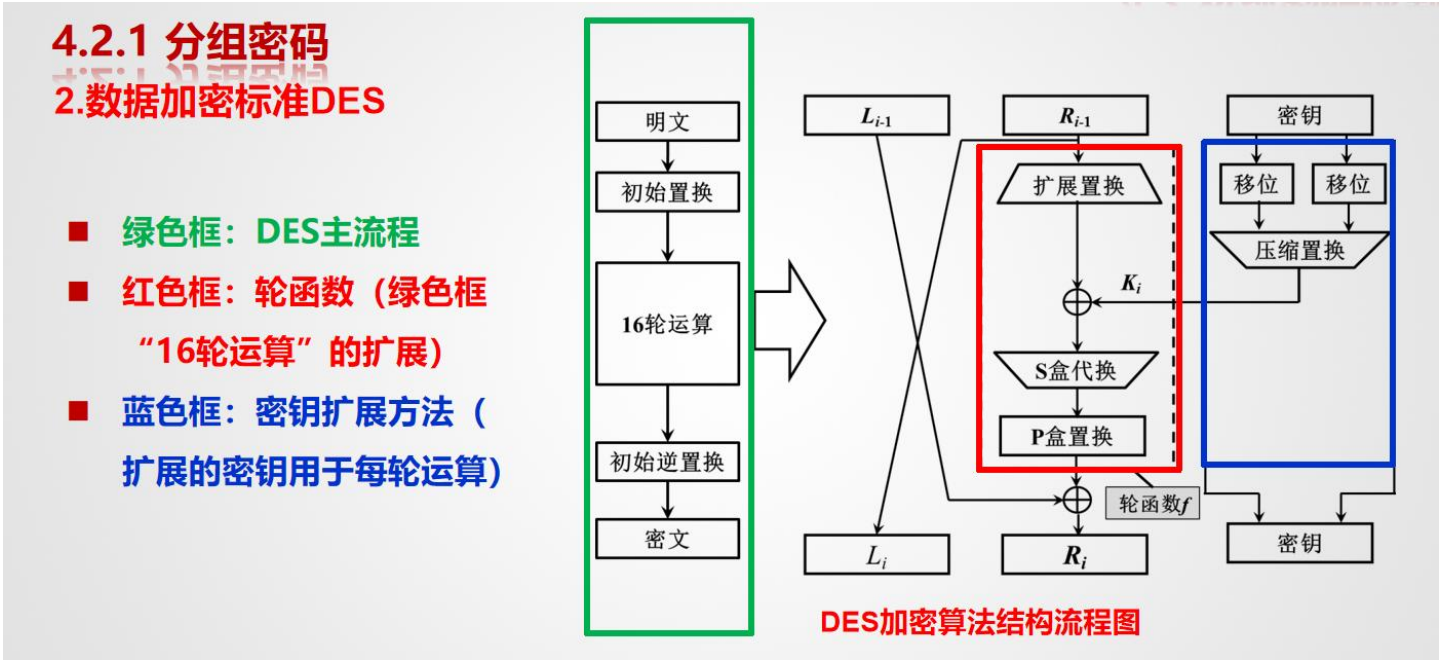
基于 3DES-CBC 模式的文件加解密小程序

一、实验内容

(1)编程实现 DES 加解密算法，并使用 DES 加解密算法实现 3DES 加解密算法
(2)选择一种填充方式，对需要加密的文件进行填充（解密要去掉填充部分）
(3)DES 的加解密的工作模式，采用密码分组链接（CBC）模式
(4)读取/写入被加密/解密文件时，采用字节流的形式进行文件读取/写入

二、实验原理

2.1 DES 加解密原理



2.1.1 DES 加密流程

（1）输入 64 位明文数据，并进行初始置换 IP；
（2）在初始置换 IP 后，明文数据再被分为左右两部分，每部分 32 位，以 L0，R0 表示；
（3）在秘钥的控制下，经过 16 轮运算(f)；
（4）16 轮后，左、右两部分交换，并连接再一起，再进行逆置换；
（5）输出 64 位密文。

2.1.2 DES 加密中子密钥生成

(1)将 64 位主钥通过置换选择 1 变为 56 位密钥，它分成左右两个 28 位半密钥
(2)将左、右两个半密钥循环左移指定次数
(3)将左右半密钥拼接成 56 位密钥，再进行置换选择 2 ，生成 48 位的子密钥
(4)重复步骤 2、步骤 3 一共 16 次，于是得到了 16 个 48 位的子密钥

2.1.3 轮函数

轮函数是 DES 加密的核心部分。在每个轮次中，它接受 32 位的数据（半个数据块），然后将其与 48 位的轮密钥进行异或操作。这个异或结果通过 S-盒置换和 P-盒置换来产生新的 32 位数据块。S-盒置换引入了非线性性质，增强了加密的安全性。

2.2 3DES 加解密原理

2.2.1 3DES 加密原理：

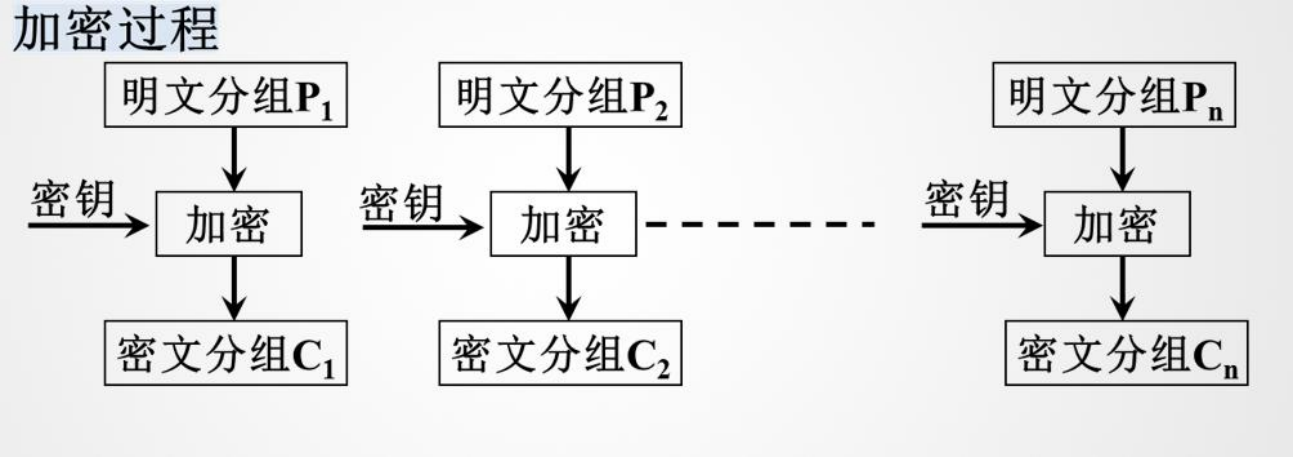
(1)密钥生成： 3DES 需要三个独立的密钥，通常称为 key1、key2 和 key3。这些密钥的长度通常为 56 位，但通常将它们扩展为 112 位或 168 位以提高安全性。
(2)初始置换： 与普通 DES 一样，3DES 首先对明文数据进行初始置换，以确保数据的混淆。
(3)轮函数的执行： 3DES 使用三个密钥对数据进行多轮加密。在每一轮中，数据分为两部分：左半部分（L）和右半部分（R）。对右半部分的操作包括以下步骤：
对右半部分应用普通 DES 加密（R = DES(key1, R)）。
对结果应用普通 DES 解密（R = DES(key2, R)）。
对结果再次应用普通 DES 加密（R = DES(key3, R)）。
(4)最后的置换： 在所有轮加密结束后，对左半部分和右半部分的数据进行最后的置换操作，以生成加密后的数据块。

2.2.2 3DES 解密原理：

3DES 的解密过程与加密过程基本相同，只是在解密时密钥的使用顺序相反。解密步骤如下：
(1)密钥生成： 与加密时相同，使用相同的三个密钥：key1、key2 和 key3。
(2)初始置换： 与加密时一样，对密文数据进行初始置换。
(3)轮函数的执行： 解密时，轮函数的执行与加密相反。对右半部分的操作包括以下步骤：
对右半部分应用普通 DES 解密（R = DES(key3, R)）。
对结果应用普通 DES 加密（R = DES(key2, R)）。
对结果再次应用普通 DES 解密（R = DES(key1, R)）。
(4)最后的置换： 在所有轮解密结束后，对左半部分和右半部分的数据进行最后

的置换操作，以生成解密后的数据块。

2.3 分组密码 CBC 加解密模式原理



2.3.1 CBC 加密原理：

(1)分组数据： 将明文数据分成固定大小的数据块（通常为 64 位或 128 位），每个数据块称为一个分组。
(2)初始向量（IV）： 选择一个随机的初始向量（IV），其大小与数据分组相同。IV 通常在每次加密过程中都不同。
(3)初始步骤： 在第一个分组上执行以下操作：
将明文分组与 IV 进行异或运算（XOR），得到异或结果。
使用密钥对异或结果进行加密（例如，使用 AES 加密），得到密文。
(4)迭代过程： 对每个后续的分组执行以下操作：
将当前分组与前一个分组的密文进行异或运算（XOR）。
使用密钥对异或结果进行加密，得到密文。

2.3.2 CBC 解密原理：

(1)初始向量（IV）： 与加密时相同，选择一个初始向量（IV）。
(2)初始步骤： 对第一个密文分组执行以下操作：
使用密钥对密文进行解密，得到解密后的数据块。
(3)将解密后的数据块与 IV 进行异或运算（XOR），得到明文分组。
(4)迭代过程： 对每个后续的密文分组执行以下操作：
使用密钥对密文进行解密，得到解密后的数据块。
将解密后的数据块与前一个密文分组进行异或运算（XOR），得到明文分组。

2.4 填充原理

2.4.1 加密

(1)最后一个分组不足 8 字节，则填充为 00 00…0X
(2)最后一个分组若正好是 8 个字节，则填充为 00 00…08

2.4.2 解密

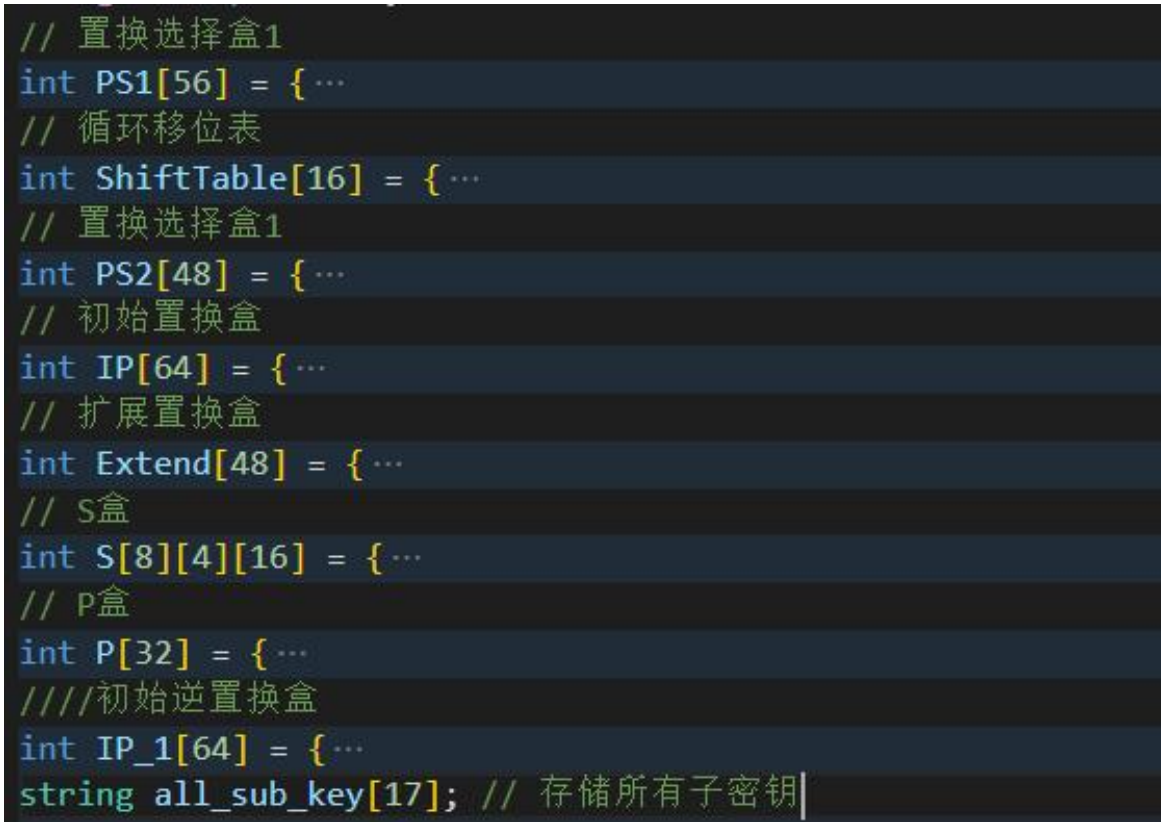
解密时先将密文解密为明文，再对明文的最后一个 8 字节分组进行解填充。
(1)最后一个分组正好是 0000…08，则去掉该分组
(2)最后一个分组满足最后一个字节是 0X，去掉的 0X 和前 X-1 对 0

三、实验过程

3.1 变量说明

3.1.1 主函数变量说明

```
int op; // 选项
string key1, key2, key3; // 密钥 1,2,3
int paddingBytes;                // 填充的字节
const int bufferSize = 8; // 一次读取的字节
char buffer[bufferSize];   // 存储字节的数组
string IV = "9876543211472583";//初始向量
bool p8 = false; // 判断是否读到文件尾
string data = "";//当前这组存储 16 进制的数据
string result = encoder(decoder(encoder(data, key1), key2), key3);//加密后的结果
stringstream ss;//字符串流读取 16 进制
```



3.1.2 其他重要变量说明

3.2 函数功能说明

3.2.1 主函数说明

(1)输出提示语句，实现与用户的交互功能，通过用户的输入选择,来确认是加密还是解密还是退出当前小程序.
(2)如果选择加密,则读取需要加密的文件,输入三个密钥,先进行填充,然后进行3DES+CBC 加密,把加密文件保存
(3)如果选择解密,则读取需要解密的文件,输入三个密钥,然后解密,然后去掉填充,保存解密的文件
(4)加密或者解密完成后提升加密或解密是否成功

3.2.2 其他重要函数说明

(1)string h_to_b(string h)
作用：16 进制转 2 进制
参数：需要转换成 2 进制的 16 进制字符串
返回：转换后的字符串

(2)string b_to_h(string b)

作用：2 进制转 16 进制
参数：需要转换成 16 进制的 2 进制字符串
返回：转换后的字符串

(3)void get_all_ki(string key)
作用：获取所有子密钥
参数：初始密钥
返回：无

(4)string yihuo(string s1, string s2)
作用：将两个字符串异或
参数：需要进行异或运算的字符串
返回：异或运算后的字符串

(5)int string_to_int(string s1)
作用：字符串变整形
参数：需要转换成整形的字符串
返回：转换后的整形

(6)string int_to_string(int i)
作用：整形变字符串
参数：需要转换成字符串的整形
返回：转换后的字符串

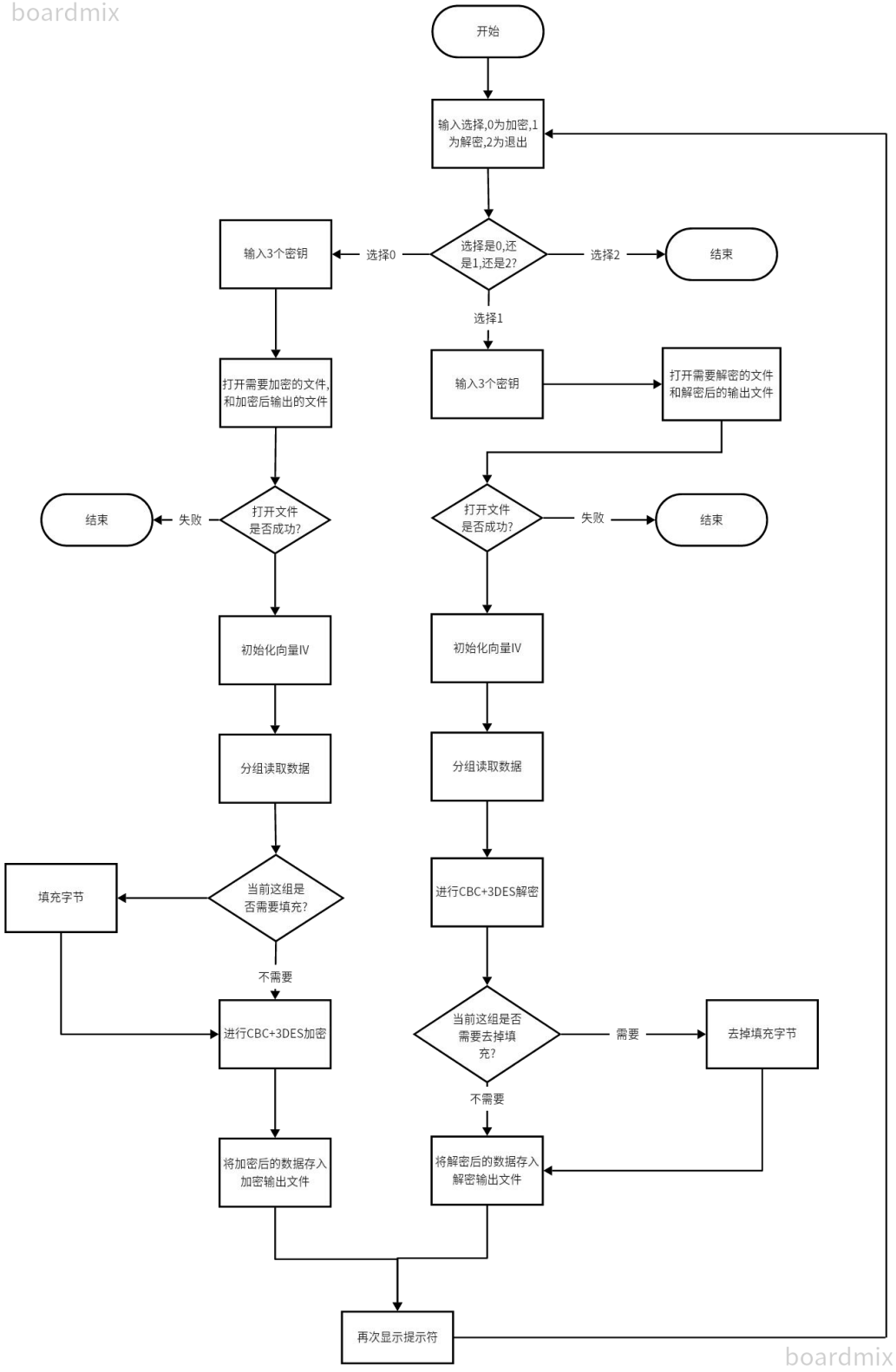
(7)string f(string r, string k)
作用：轮函数 f
参数：字符串 r 和当前的轮数 k
返回：转换后的字符串

(8)string encoder(string mingwen_b, string key)
作用：加密
参数：二进制表示的明文字符串和密钥
返回：加密后的 64 位字符串

(9string decoder(string miweng_b, string key)
作用：解密
参数：二进制表示的密文字符串和密钥
返回：解密后的 64 位字符串

3.3 流程图

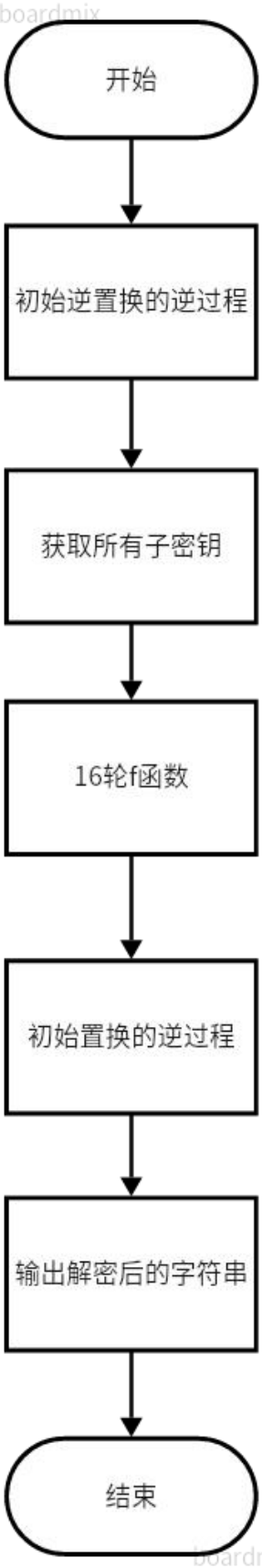
3.3.1 主函数流程图



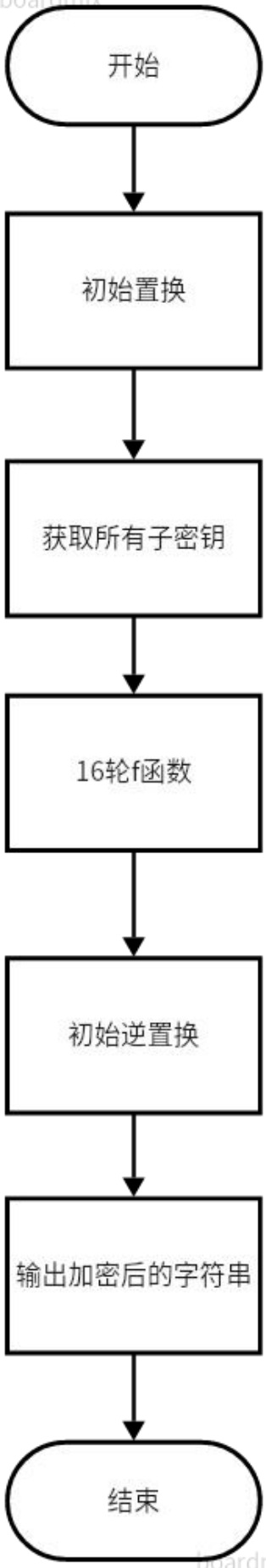
.

3.3.2 其他重要函数流程图

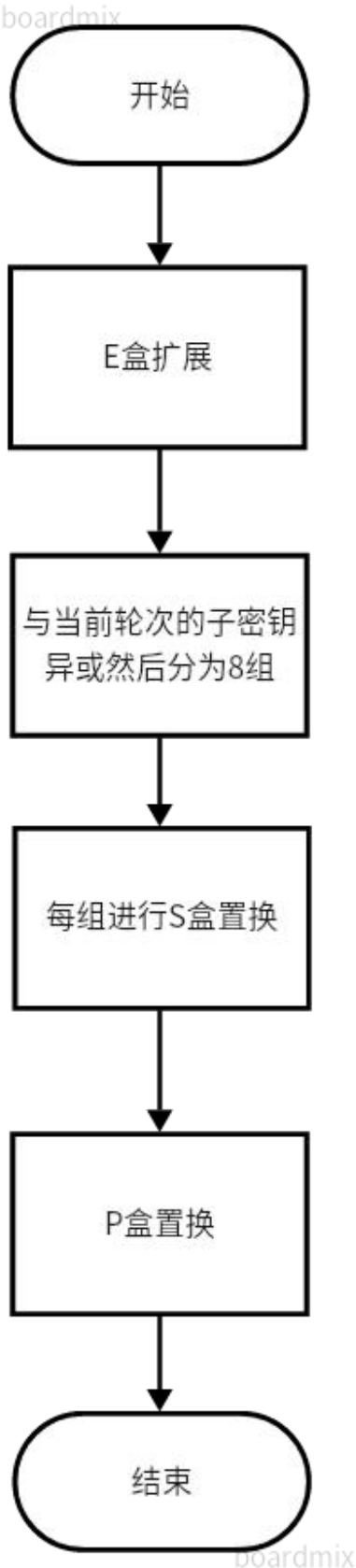
3.3.2.1 string decoder(string miweng_b, string key)流程图



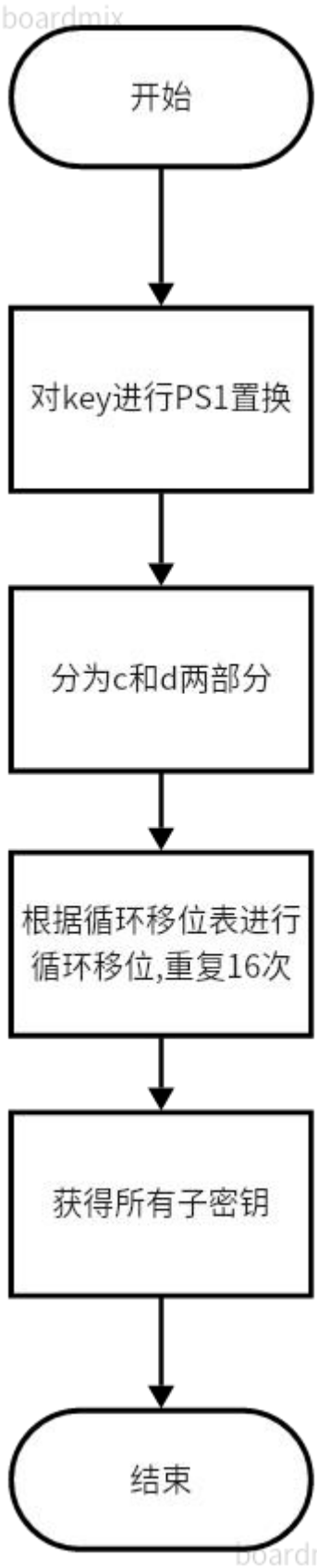
3.3.2.2 string encoder(string mingwen_b, string key)流程图



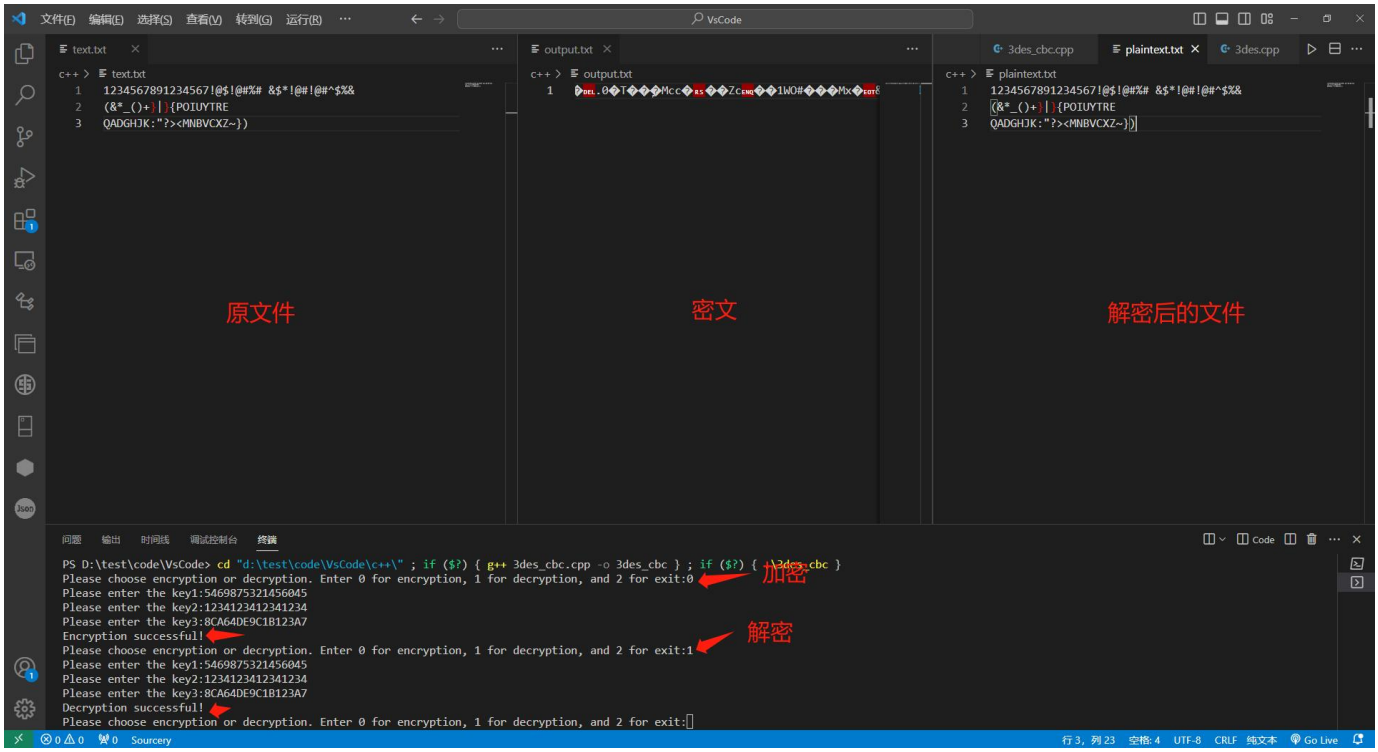
3.3.2.3 string f(string r, string k)流程图



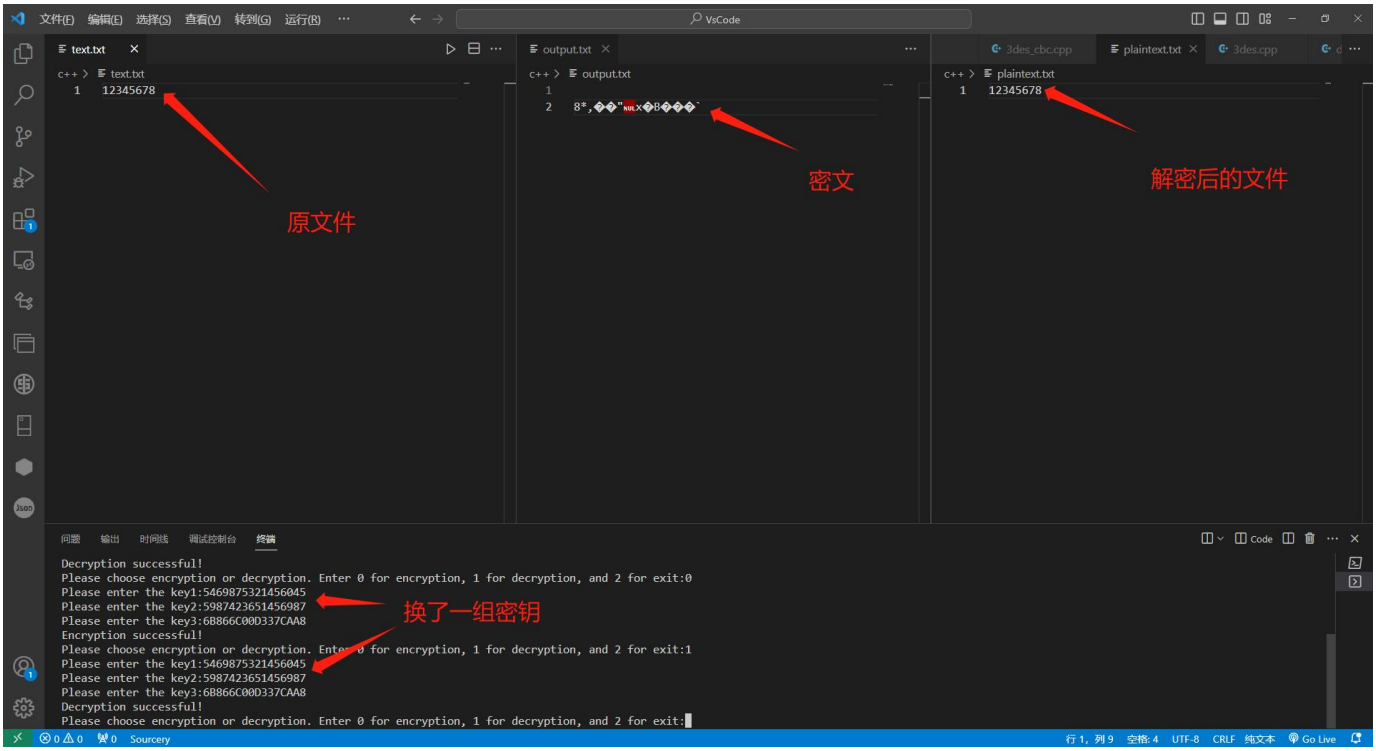
3.3.2.4 void get_all_ki(string key)流程图



四、实验结果与截图



如上图所示,左边是需要加密的文件,中间是加密后的文件,右边是解密后的文件



如上图所示,考虑8字节的特殊情况,换了和上面那个示例相比,同时换了一组密钥,此时填充和去掉填充也表示正常

五、实验总结

这次实验给了我宝贵的经验。这次实验从总体来看确实较为复杂,但把每一个功能单独抽象出来具体一步一步的实现,配合老师的讲解,写出代码就非常的容易了,在实验的一开始,只是要我们搭建一个单独的 DES,后面再搭建 3DES,到最后搭建 CBC+3DES 进行文件的加密和解密,每次都是循序渐进,由易到难,此次实验极大的提高了我对密码学的理解和自己的编码能力,同时非常感谢在实验期间徐老师耐心的教导,每次在 QQ 联系徐老师,徐老师都能很快的回复并解答我的疑惑,所以我也能非常顺利的完成这次课程设计,学习完信息安全这么课后,我不仅编写代码的能力得到了提升了,更重要的是,我是我学习到了信息安全在现今社会的重要性,无论是金融贸易,国防,公司文件,通信还是其他方面,都与信息安全密不可分,所以这么课对我来说很有意义,让我明白了以后设计代码不仅要完成这个代码本身的功能,还要考虑代码的安全性,特别是面对恶意攻击,如何设计程序抵抗,也让我明白了代码的设计不是孤立的,而是多学科交叉的,所以,非常感谢信息安全这门课刷新了我对代码和程序的认知,也非常非常感谢徐老师耐心细致的教导

五、附录(源码)

```
#include <bits/stdc++.h>
using namespace std;
int PS1[56] = {
    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};
// 循环移位表
int ShiftTable[16] = {
    1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
int PS2[48] = {
    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};
```

```
int IP[64] = {
    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};
int Extend[48] = {
    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};
// S 盒
int S[8][4][16] = {
    // 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}}};
int P[32] = {
    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};
int IP_1[64] = {
    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 h_to_b(string h) // 十六进制变二进制
{
    string b = "";
    // cout << "h.len:" << h.length() << endl;
    for (int i = 0; i < h.length(); i++)
    {
        if (h[i] == '0')
            b.append("0000");
        if (h[i] == '1')
            b.append("0001");
```

```
        if (h[i] == '2')
            b.append("0010");
        if (h[i] == '3')
            b.append("0011");
        if (h[i] == '4')
            b.append("0100");
        if (h[i] == '5')
            b.append("0101");
        if (h[i] == '6')
            b.append("0110");
        if (h[i] == '7')
            b.append("0111");
        if (h[i] == '8')
            b.append("1000");
        if (h[i] == '9')
            b.append("1001");
        if (h[i] == 'A')
            b.append("1010");
        if (h[i] == 'B')
            b.append("1011");
        if (h[i] == 'C')
            b.append("1100");
        if (h[i] == 'D')
            b.append("1101");
        if (h[i] == 'E')
            b.append("1110");
        if (h[i] == 'F')
            b.append("1111");
    }
    return b;
}
string b_to_h(string b) // 二进制变十六进制
{
    string h = "";
    string t = "";
    for (int i = 0; i < b.length(); i++)
    {
        t += (b[i]);
        if (t.length() == 4)
        {
            if (t == "0000")
                h += "0";
            if (t == "0001")
                h += "1";
```

```
            if (t == "0010")
                h += "2";
            if (t == "0011")
                h += "3";
            if (t == "0100")
                h += "4";
            if (t == "0101")
                h += "5";
            if (t == "0110")
                h += "6";
            if (t == "0111")
                h += "7";
            if (t == "1000")
                h += "8";
            if (t == "1001")
                h += "9";
            if (t == "1010")
                h += "A";
            if (t == "1011")
                h += "B";
            if (t == "1100")
                h += "C";
            if (t == "1101")
                h += "D";
            if (t == "1110")
                h += "E";
            if (t == "1111")
                h += "F";
            t = "";
        }
    }
    return h;
}
string initial_place(string s) // 初始置换
{
    string ip = "";
    for (int i = 0; i < 64; i++)
        ip += s[IP[i] - 1];
    return ip;
}
string all_sub_key[17]; // 存储所有子密钥
string get_ki(int epoch)
{
    return all_sub_key[epoch];
```

```
}
void get_all_ki(string key) // 获取所有子密钥
{
    string temp = "";
    key = h_to_b(key);
    for (int i = 0; i < 56; ++i)
        temp += key[PS1[i] - 1];
    string c[17], d[17];
    c[0] = temp.substr(0, 28);
    d[0] = temp.substr(28, 28);
    for (int i = 1; i <= 16; ++i)
    {
        c[i] = c[i - 1].substr(ShiftTable[i - 1], 28 - ShiftTable[i - 1]) + c[i - 1].substr(0,
ShiftTable[i - 1]);
        d[i] = d[i - 1].substr(ShiftTable[i - 1], 28 - ShiftTable[i - 1]) + d[i - 1].substr(0,
ShiftTable[i - 1]);
        temp = c[i] + d[i];
        string t2 = "";
        for (int j = 0; j < 48; ++j)
        {
            t2 += temp[PS2[j] - 1];
        }
        all_sub_key[i] = t2;
    }
}
string yihuo(string s1, string s2) // 异或
{
    string yh = "";
    for (int i = 0; i < s1.length(); i++)
    {
        if (s1[i] != s2[i])
            yh += "1";
        else
            yh += "0";
    }
    return yh;
}
int string_to_int(string s1) // 字符串变整形
{
    int ans = 0;
    int base = 2;
    int e = 0;
    for (int i = s1.length() - 1; i >= 0; i--)
    {
```

```
        ans += (s1[i] - '0') * pow(base, e);
        e++;
    }
    return ans;
}
string int_to_string(int i) // 整形变字符串
{
    string ans = "";
    while (i)
    {
        ans += ((i % 2) + '0');
        i /= 2;
    }
    while (ans.length() != 4)
    {
        ans += "0";
    }

    reverse(ans.begin(), ans.end());
    return ans;
}
string f(string r, string k) // f 函数
{
    string temp = "";
    // E 盒扩展
    for (int i = 0; i < 48; ++i)
    {
        temp += r[Extend[i] - 1];
    }
    // 异或
    temp = yihuo(temp, k);
    string b[9];
    for (int i = 0; i < 8; ++i)
    {
        b[i] = temp.substr(6 * i, 6);
    }
    string ans = "";
    for (int i = 0; i < 8; ++i)
    {
        string t = "";
        t += b[i][0];
        t += b[i][5];
        // 变为整形
        int row = string_to_int(t);
```

```
        int col = string_to_int(b[i].substr(1, 4));
        int ts = S[i][row][col];
        string t1 = int_to_string(ts);
        ans += t1;
    }
    string t2 = "";
    // P 盒置换
    for (int j = 0; j < 32; ++j)
    {
        t2 += ans[P[j] - 1];
    }
    return t2;
}
```

```
string encoder(string mingwen_b, string key) // 加密,八字节一组
{
    // 初始置换
    string ip = initial_place(mingwen_b);
    string l[30], r[30];
    l[0] = ip.substr(0, 32);
    r[0] = ip.substr(32, 32);
    // 获取所有子密钥
    get_all_ki(key);
    // 16 轮 f 函数
    for (int i = 1; i <= 16; i++)
    {
        string ki = get_ki(i);
        string t = f(r[i - 1], ki);
        r[i] = yihuo(t, l[i - 1]);
        l[i] = r[i - 1];
    }
    string temp = r[16] + l[16];
    string result = "";
    // 逆置换
    for (int i = 0; i < 64; ++i)
    {
        result += temp[IP_1[i] - 1];
    }
    return result;
}
string decoder(string miweng_b, string key) // 解密
{
    // 输入密文
    string ip_1(64, ' ');
```

```
    // 逆置换的逆过程
    for (int i = 0; i < 64; ++i)
    {
        ip_1[IP_1[i] - 1] = miweng_b[i];
    }
    // 获取子密钥 k1-k16
    get_all_ki(key);
    string l[30], r[30];
    l[16] = ip_1.substr(32, 32);
    r[16] = ip_1.substr(0, 32);
    // f 函数,16 轮
    for (int i = 16; i >= 1; i--)
    {
        r[i - 1] = l[i];
        string ki = get_ki(i);
        string t = f(r[i - 1], ki);
        l[i - 1] = yihuo(t, r[i]);
    }
```

```
    string ip = l[0] + r[0];
    string mingwen_b(64, ' ');
    // 初始置换的逆过程
    for (int i = 0; i < 64; ++i)
    {
        mingwen_b[IP[i] - 1] = ip[i];
    }
    return mingwen_b;
}
int main()
{
    int op; // 选项
    while (1)
    {
        string key1, key2, key3;
        printf("Please choose encryption or decryption. Enter 0 for encryption, 1 for decryption,
and 2 for exit:");
        cin >> op;
        if (op == 0) // 加密
        {
            while (1) // 输入密钥
            {
                cout << "Please enter the key1:";
                cin >> key1;
                cout << "Please enter the key2:";
```

```
            cin >> key2;
            cout << "Please enter the key3:";
            cin >> key3;
            if ((key1.length() != 16) || (key2.length() != 16) || (key3.length() != 16))
            {
                cout << "Key format error" << endl;
            }
            else
                break;
        }
        // 打开文件并设置为二进制模式
        ifstream inputfile("D:\\test\\code\\VsCode\\c++\\text.txt", ios::binary);
        if (!inputfile)
        {
            cerr << "Unable to open input file" << endl;
            return 1;
        }
        ofstream outputfile("D:\\test\\code\\VsCode\\c++\\output.txt", ios::binary);
        if (!outputfile)
        {
            cerr << "Unable to open output file." << endl;
            return 1;
        }
        inputfile.seekg(0, std::ios::end); // 将文件指针移动到文件末尾
        if (inputfile.tellg() == 0)
        {
            std::cout << "File is empty" << std::endl;
            return 0;
        }
        else
        {
            inputfile.seekg(0, std::ios::beg);
        }
        int paddingBytes;
        const int bufferSize = 8;
        char buffer[bufferSize];
        string IV = "9876543211472583";
        IV = h_to_b(IV);
        while (!inputfile.eof())
        {
            inputfile.read(buffer, bufferSize);
            string data = "";
            paddingBytes = 8 - (static_cast<int>(inputfile.gcount()) % 8);
            for (int i = 0; i < inputfile.gcount(); i++)
```

```
                {
                    stringstream ss;
                    ss << hex << setw(2) << setfill('0') <<
static_cast<unsigned>(static_cast<unsigned char>(buffer[i]));
                    data += ss.str();
                }
                bool p8 = false;
                if (inputfile.peek() == EOF)
                {
                    if (paddingBytes < 8)
                    {
                        for (int i = 1; i < paddingBytes; i++)
                        {
                            data += "00";
                        }
                        data += "0";
                        data += paddingBytes + '0';
                    }
                    else
                        p8 = true;
                }
                // work
                for (char &c : data)
                {
                    if (c >= 'a' && c <= 'z')
                        c = std::toupper(c);
                }
                data = h_to_b(data);
                data = yihuo(data, IV);
                string result = encoder(decoder(encoder(data, key1), key2), key3);
                IV = result;
                result = b_to_h(result);
                for (int i = 0; i < result.length(); i += 2)
                {
                    std::string byteStr = result.substr(i, 2);
                    unsigned char byte = static_cast<unsigned char>(std::stoi(byteStr, 0, 16));
                    outputfile.write(reinterpret_cast<const char *>(&byte), 1);
                }
                if (p8)
                {
                    data = "";
                    for (int i = 1; i < paddingBytes; i++)
                    {
                        data += "00";
```

```
                }
                data += "0";
                data += paddingBytes + '0';
                data = h_to_b(data);
                data = yihuo(data, IV);
                string result = encoder(decoder(encoder(data, key1), key2), key3);
                IV = result;
                result = b_to_h(result);
                for (int i = 0; i < result.length(); i += 2)
                {
                    std::string byteStr = result.substr(i, 2);
                    unsigned char byte = static_cast<unsigned char>(std::stoi(byteStr, 0, 16));
                    outputfile.write(reinterpret_cast<const char *>(&byte), 1);
                }
            }
        }
        inputfile.close();
        outputfile.close();
        cout << "Encryption successful!" << endl;
    }
    else if (op == 1) // 解密
    {
        string miweng, key1, key2, key3;
        while (1) // 输入密钥
        {
            cout << "Please enter the key1:";
            cin >> key1;
            cout << "Please enter the key2:";
            cin >> key2;
            cout << "Please enter the key3:";
            cin >> key3;
            if ((key1.length() != 16) || (key2.length() != 16) || (key3.length() != 16))
            {
                cout << "Key format error" << endl;
            }
            else
                break;
        }
        ifstream inputfile("D:\\test\\code\\VsCode\\c++\\output.txt", ios::binary);
        if (!inputfile)
        {
            cerr << "Unable to open input file" << endl;
            return 1;
        }
```

```
            ofstream outputfile("D:\\test\\code\\VsCode\\c++\\plaintext.txt", ios::binary);
            if (!outputfile)
            {
                cerr << "Unable to open output file." << endl;
                return 1;
            }
            int paddingBytes;
            const int bufferSize = 8;
            char buffer[bufferSize];
            string IV = "9876543211472583";
            IV = h_to_b(IV);
            bool isLastBlock = false;
            string data = "";
            while (inputfile.read(buffer, bufferSize))
            {
                data = "";
                for (int i = 0; i < inputfile.gcount(); i++)
                {
                    stringstream ss;
                    ss << hex << setw(2) << setfill('0') <<
static_cast<unsigned>(static_cast<unsigned char>(buffer[i]));
                    data += ss.str();
                }
                // 大写转换
                for (char &c : data)
                {
                    if (c >= 'a' && c <= 'z')
                        c = std::toupper(c);
                }
                data = h_to_b(data);
                string result = decoder(encoder(decoder(data, key3), key2), key1);
                result = yihuo(result, IV);
                result = b_to_h(result);
                IV = data;
                if (inputfile.peek() == EOF)
                {
                    paddingBytes = result[result.length() - 1] - '0';
                    result = result.substr(0, result.length() - 2 * paddingBytes);
                }
                for (size_t i = 0; i < result.length(); i += 2)
                {
                    string byteStr = result.substr(i, 2);
                    unsigned char byte = static_cast<unsigned char>(stoi(byteStr, 0, 16));
                    outputfile.write(reinterpret_cast<const char *>(&byte), 1);
```

```
                }
            }
            inputfile.close();
            outputfile.close();
            cout << "Decryption successful!" << endl;
        }
        else if (op == 2) // 退出
        {
            break;
        }
        else
            cout << "input error" << endl;
    }
}
```