江 蘇 大 學

JIANGSU UNIVERSITY

2017-2018 学年第 1 学期

《密码学原理与技术》课程设计

学院名称：计算机科学与通信工程学院

专业班级：信息安全 1501

学生姓名：沈鑫楠

学生学号：3150604028

教师姓名：张星

完成日期：2018 年 1 月 18 日

目录

1.任务要求

题目：**三重 DES 文件加解密系统的设计与实现**

1.1 问题描述

编程实现 3DES 算法。 从 DES 原理出发，设计 3DES 加解密过程；通过编程调试以实现 3DES 算法；利用由学生本人的学号姓名等信息组成若干密钥，以及明文样本进行加解密测试；最后作总结。

1.2 设计要求

（1）设计良好的交互界面，如要求用户输入密钥、明文字符串、得到相应的解密字符串等。

（2）程序设计，编写相应程序并调试。

（3）试用验证，记录每次操作过程和系统的结果。

（4）分析相应的问题。

（5）编写课程设计报告。

2.开发环境与工具

2.1 开发环境

2.1.1 硬件平台

(1)、主频 1GHz 及以上的微处理器

(2)、使用 1G 及以上内存

(3)、不少于 100MB 的可用磁盘空间

2.1.2 软件平台

(1)、Windows 7 及以上的操作系统

(2)、相关 VC 运行库已安装

2.2 开发工具

使用 Visual Studio 2013 Ultimate 版本进行开发

3.设计原理与流程图

3.1 设计原理

3.1.1 总体设计

本课题要求实现一个 3DES 的加解密软件。3DES 是实现了三重 DES 的加解密算法，如图 1、图 2 所示。

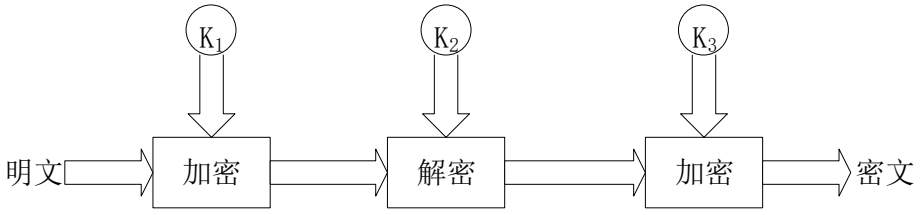

图 1 3DES 加密流程

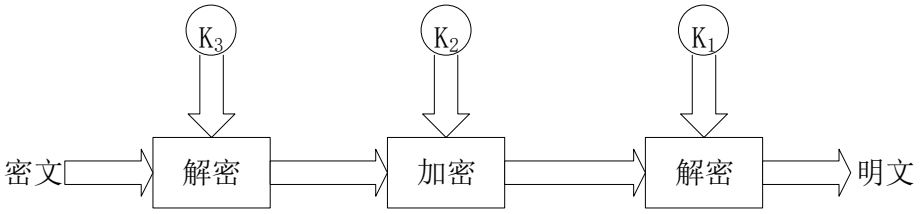

图 2 3DES 解密流程

3.1.2 详细设计

（1）DES 基本结构

DES 是一种使用 Feistel 体制的分组密码，使用 56 比特原始密钥产生 16 组轮密钥，对 64 比特的明文分组进行 16 轮变换，最终得到密文分组。而解密时使用加密的函数进行解密，但是需要将 16 组轮密钥逆向使用。DES 的整体结构如图 3 所示。

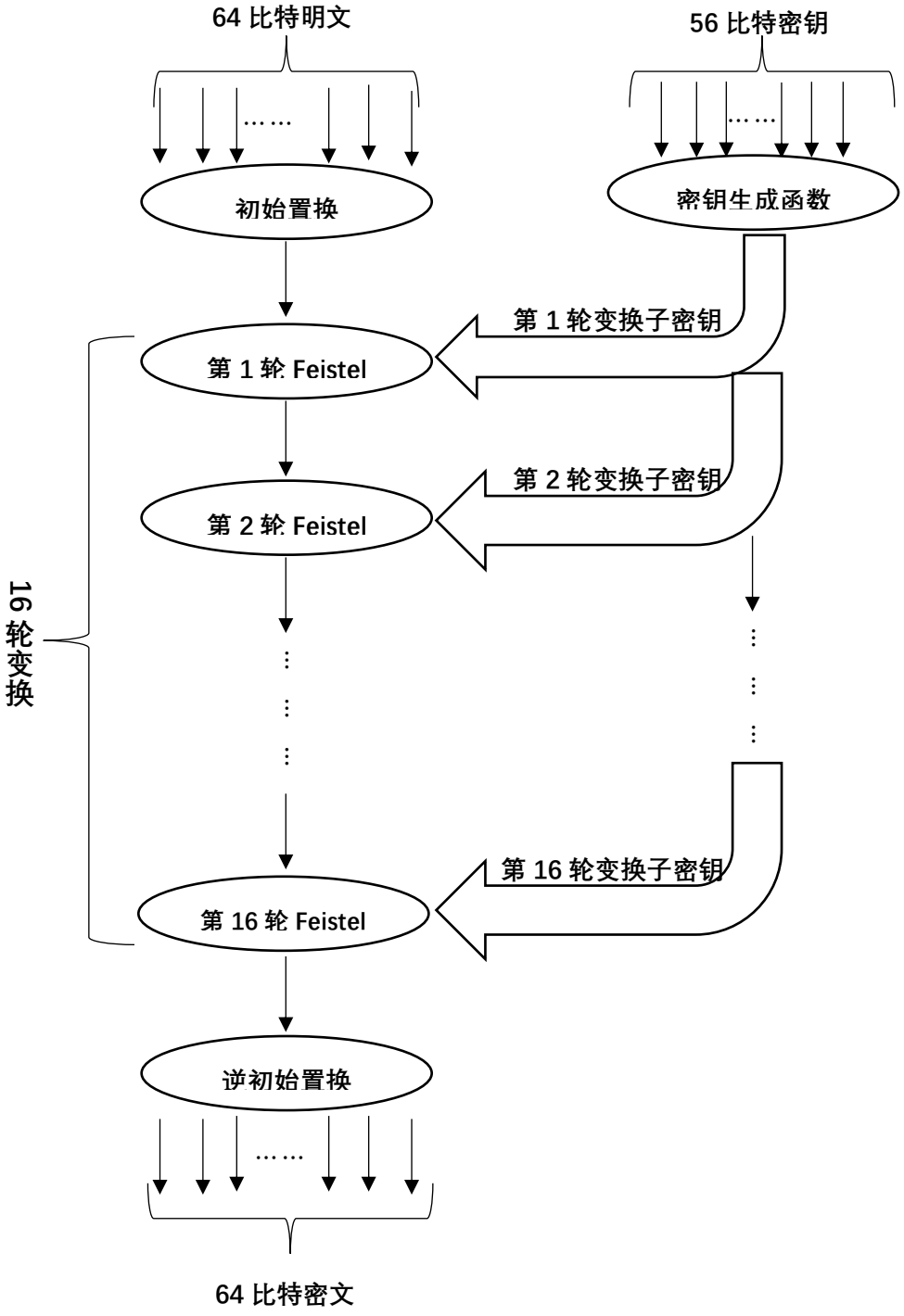

图 3 DES 基本结构图

（2）DES 的 Feistel 体制

Feistel 体制是一种多轮结构的加解密体制，每一轮的操作相同，一轮的 Feistel 体制的示意图如图 4 所示。

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

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

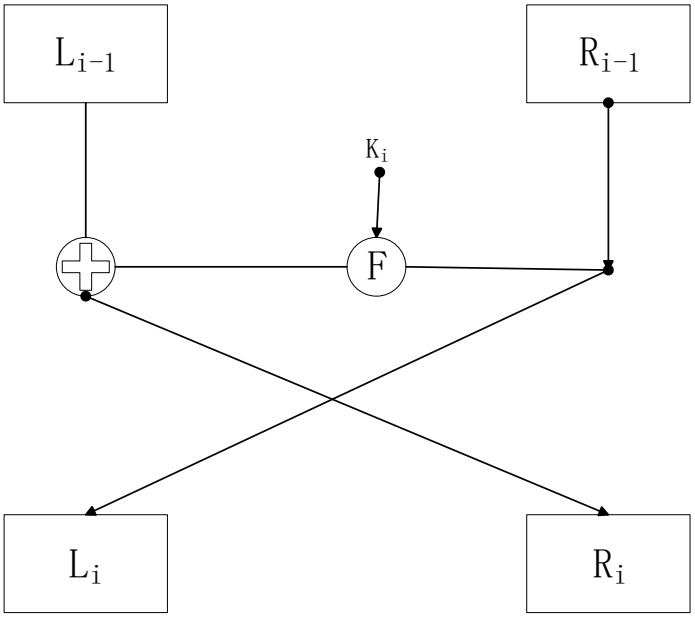

图 4 单轮 Feistel 体制

DES 的具体单轮 Feistel 体制的实现如下，主要是由拓展置换（E 置换）、异或操作、代换选择（S 盒代换）和 P 盒置换组成。整个 F 函数的流程如图 5 所示：

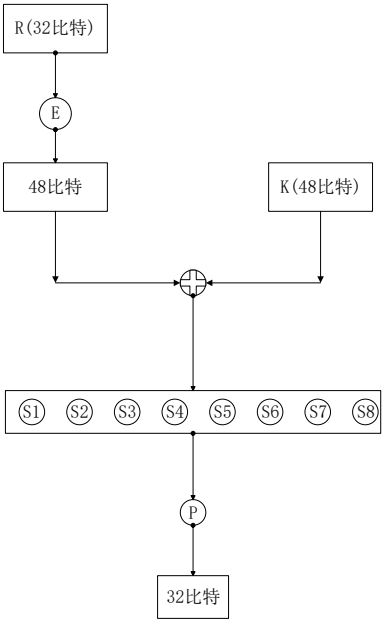

图 5 DES 的 F 函数示意图

①扩展置换（E 置换）

如表 6 所示，表中的每个元素表明了某个输入比特在输出比特的位置。

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

表 6 扩展置换（E）

②S 盒变换

每个 S 盒的输入为 6 比特，输出为 4 比特，输入的第 1 位和最后 1 位组成一个 2 比特的二进制数，用来确定 S 盒的行数；中间 4 位组成一个 4 比特的二进制数，用来确定 S 盒的列数。S 盒如表 7 所示。

盒 S_1															
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

盒 S_2															
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
盒 S_3															
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
盒 S_4															
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
盒 S_5															
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
盒 S_6															
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
盒 S_7															
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
盒 S_8															
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

表 7 DES 的 S 盒变换

③P 盒置换

取得 S 盒变换的结果后，还需要再进行 P 盒置换才能得到 F 函数的输出，表中的每个元素表明了某个输入比特在输出比特的位置。P 盒置换表如表 8 所示。

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

表 8 P 盒置换表

（3）DES 的初始置换与逆初始置换

DES 在执行 Feistel 体制的 16 轮变换运算之前，要先进行初始置换；16 轮变换运算完成后，还要进行逆初始置换。初始置换和逆初始置换数据如表 9、表 10 所示，表中的每个元素表明了某个输入比特在输出比特的位置。

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

表 9 初始置换表（IP）

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

表 10 逆初始置换表（IP^{-1}）

（4）DES 的密钥扩展算法

64 比特的密钥先进行置换选择-1 运算，去除校验位，得到 56 比特的密钥。然后，密钥被分为 2 个 28 比特的分组，进行 16 轮运算得到 16 轮运算的轮密钥。假设当前处于第 i 轮，输入为 C_{i-1}和 D_{i-1}，首先进行左移运算，然后进行置换选择-2 运算，形成子密钥 K_i。密钥扩展运算单轮结构如图 11 所示，置换选择-1 的数据如表 12 所示，移位次数关系表如表 13 所示，置换选择-2 的数据如表 14 所示。

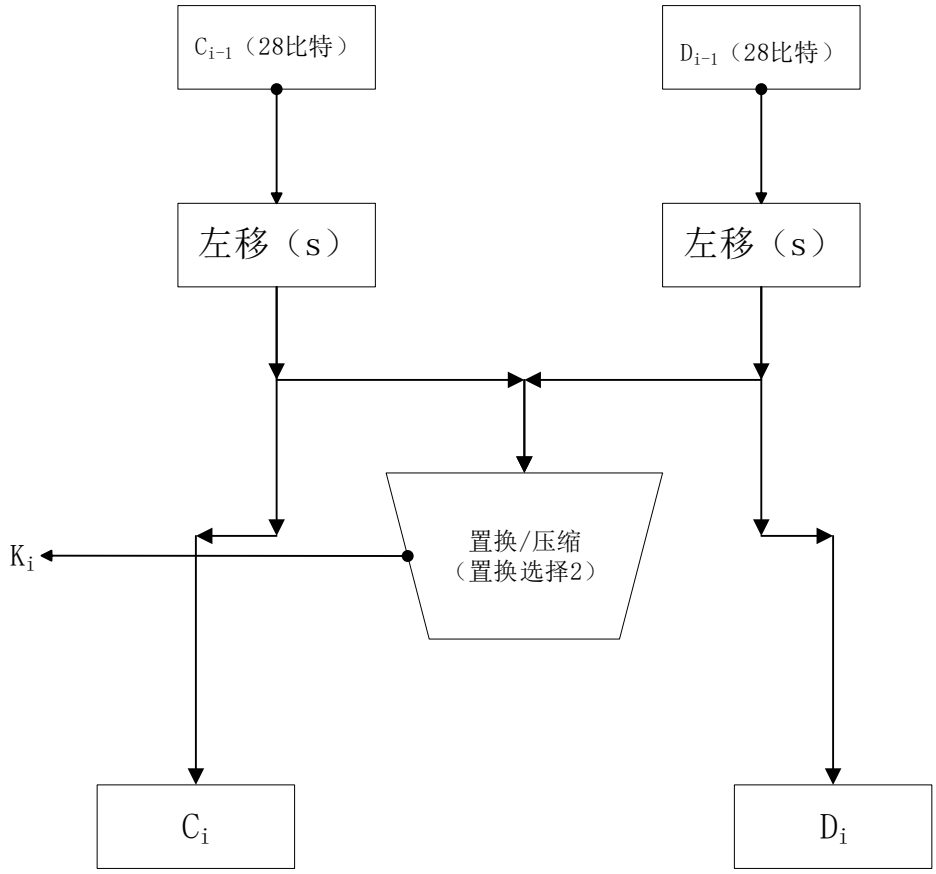

图 11 密钥扩展算法单轮结构

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 12 置换选择-1（PC-1）

迭代轮次	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
移位次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

表 13 移位次数关系表

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	27	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

表 14 置换选择-2（PC-2）

3.2 具体实现算法与流程图

3.2.1 分组密码工作模式

（1）电码本（ECB）模式

电码本模式是将明文分块后对每一块明文进行加密得到相应密文块，再拼接后得到密文，如图 15 所示。

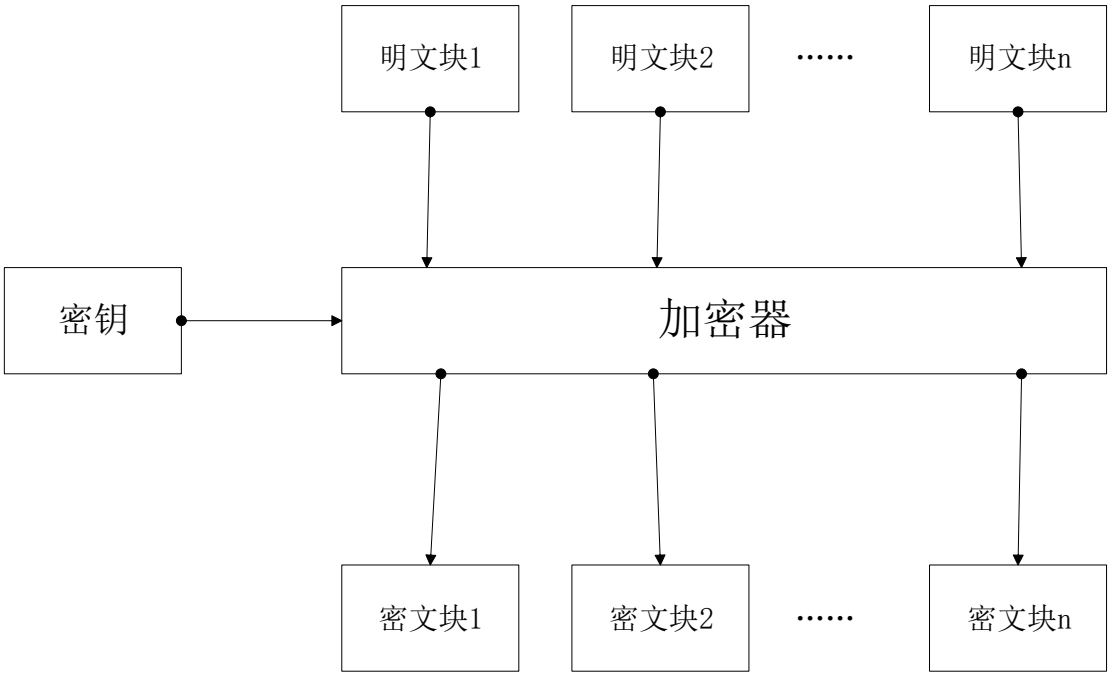

图 15 电码本模式示意图

（2）密码分组链接（CBC）模式

密码分组链接模式是将每个明文块先于前一个密文块相异或，再进行加密，而对于第一个密文块，将其与一个初始向量相异或，再进行加密，如图 16 所示。

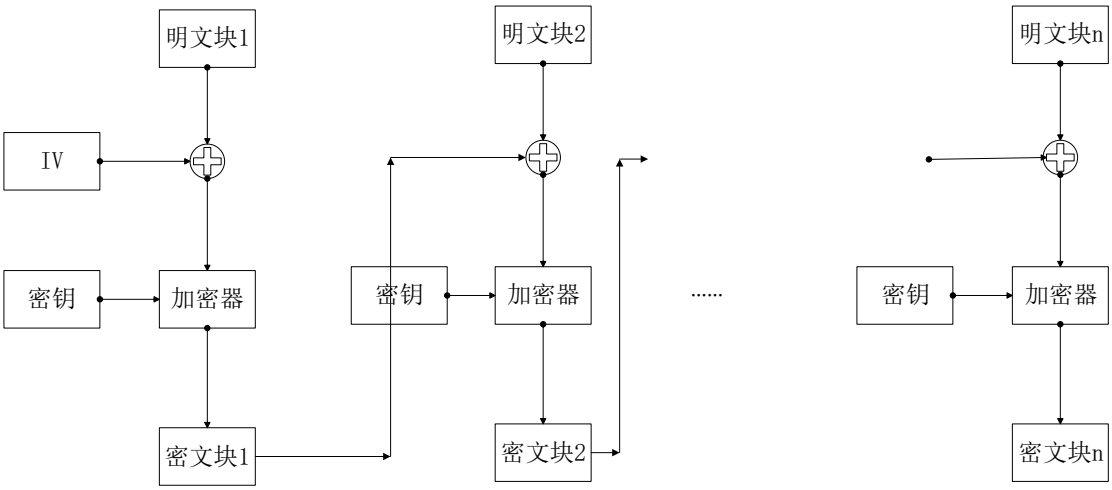

图 16 密码分组链接模式示意图

（3）密文反馈（CFB）模式

密文反馈模式是明文不进入加密算法中处理，而是与加密算法的输出异或得到密文；加密算法的输入为上次的密文以及加密用的密钥，对于第一次加密输入使用初始向量 IV，如图 17 所示。

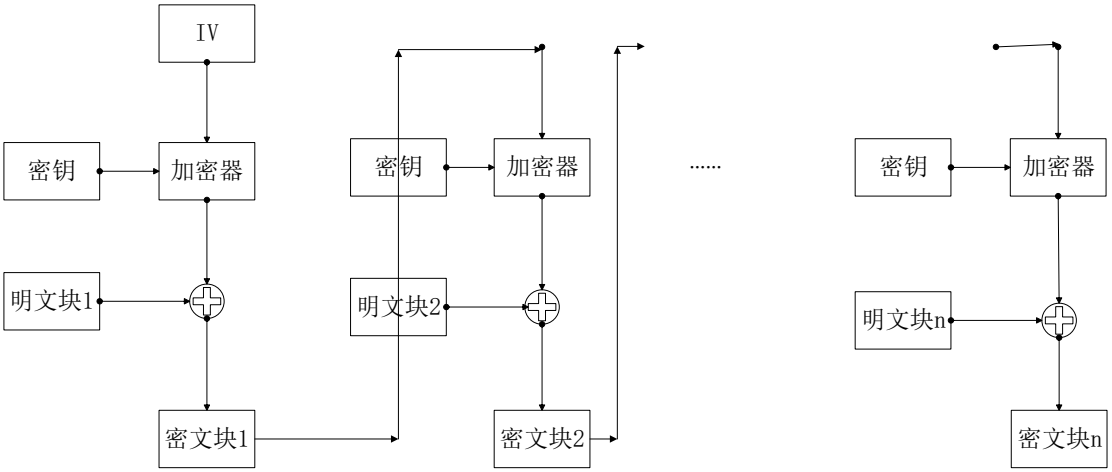

图 17 密文反馈模式示意图

（4）输出反馈（OFB）模式

输出反馈模式使用上一轮加密算法的输出作为下一轮加密算法的输入，对于第一轮加密，采用一个初始向量作为加密算法的输入，如图 18 所示。

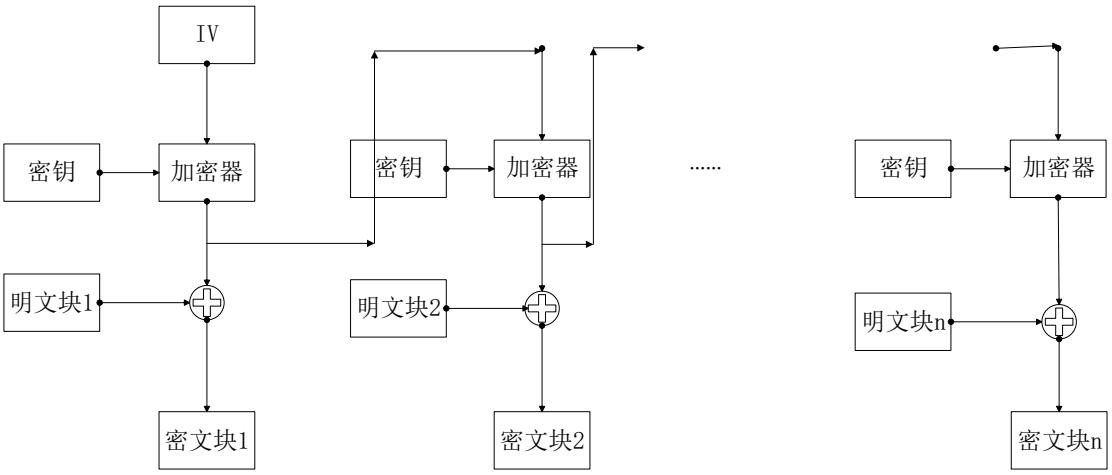

图 18 输出反馈模式示意图

（5）计数器（CTR）模式

在计数器模式中，每个分组具有自己的计数，计数器的输出为本轮加密算法的输入，加密算法的输出与明文块异或得到密文，如图 19 所示。

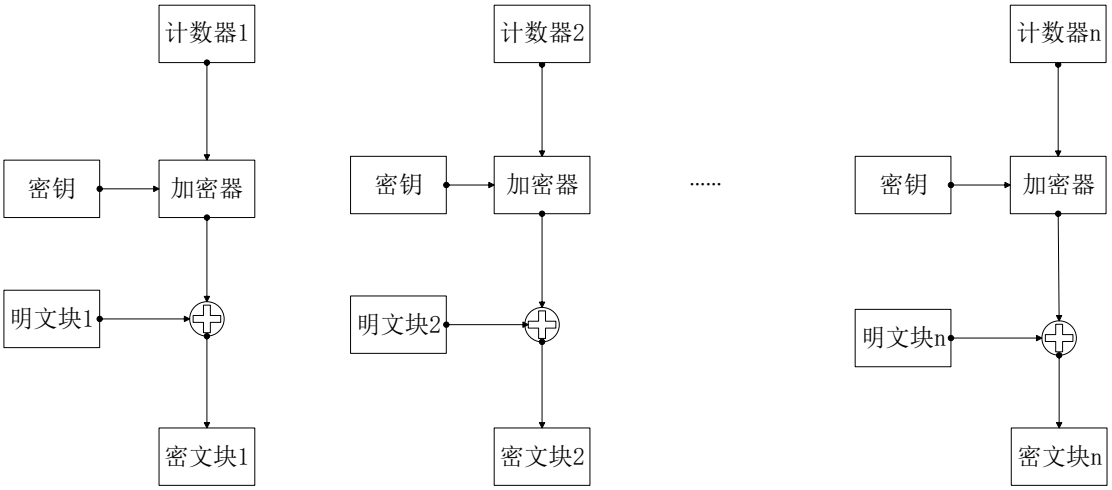

图 19 计数器模式示意图

3.2.2 分组密码填充模式

（1）零字节填充

零字节填充是发送方在需要填充字节的位置填充 0，接收方收到密文并解密后会将末尾的所有 0 删去。

（2）PKCS7

对于使用 PKCS7 填充模式，发送方根据字节数的不同进行填充。如果最后有一个字节需要填充，发送方填充上 0x01；如果最后有两个字节需要填充，发送方填充上 0x0202；如果最后有三个字节需要填充，发送方填充上 0x030303；如果最后有四个字节需要填充，发送方填充上 0x04040404，以此类推。接收方收到密文并解密后会查看最后的填充字符是否符合填充规则，然后删去填充的字符。

（3）ANSI X.923

对于使用 ANSI X.923 填充模式，发送方根据字节数的不同进行填充。如果最后有一个字节需要填充，发送方填充上 0x01；如果最后有两个字节需要填充，发送方填充上 0x0002；如果最后有三个字节需要填充，发送方填充上 0x000003；如果最后有四个字节需要填充，发送方填充上 0x00000004，以此类推。接收方收到密文并解密后会查看最后的填充字符是否符合填充规则，然后删去填充的字符。

（4）ISO 10126

对于使用 ISO 10126 填充模式，发送方在最后一个字节填充总共填充的比特数，其余字节填充随机数。接收方收到密文并解密后会查看最后的填充字符是否符合填充规则，然后删去填充的字符。

3.2.3 流程图

（1）程序执行总流程图（如图 20 所示）

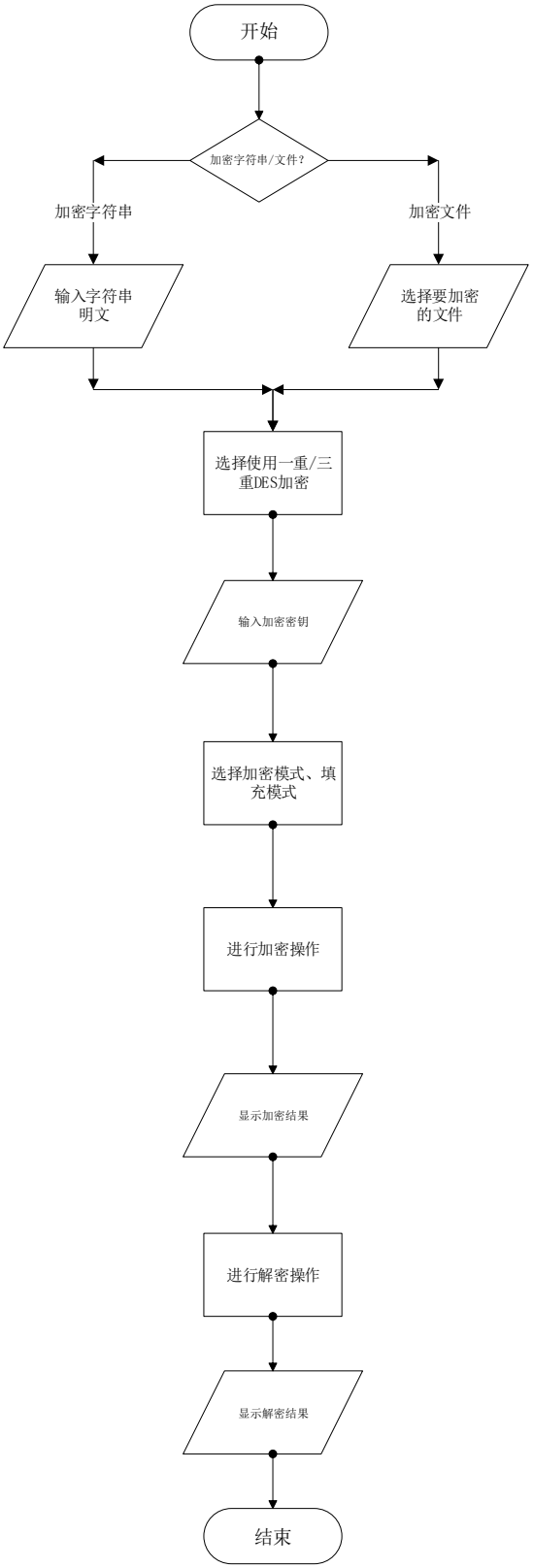

图 20 程序执行总流程图

（2）加密模块流程图（如图 21 所示）

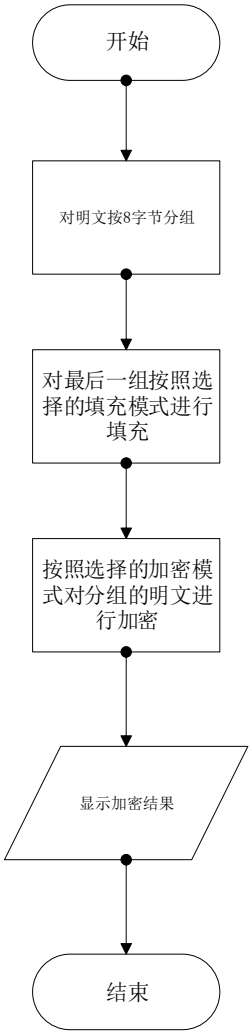

图 21 加密模块流程图

（3）解密模块流程图（如图 22 所示）

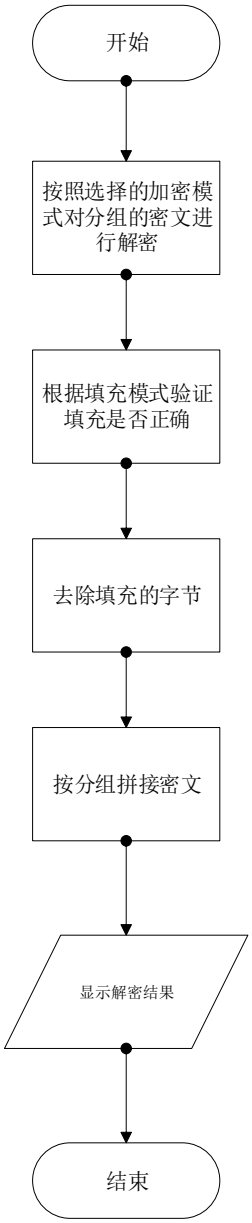

图 22 解密模块流程图

4.程序主要编码实现

4.1 3DES 分组密码工作模式和填充模式的实现

```
// Encrypt_Decrypt_MFCDlg.cpp : 实现文件
//
#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS 1//去除警告提示
#endif
#include "stdafx.h"
#include "Encrypt_Decrypt_MFC.h"
#include "Encrypt_Decrypt_MFCDlg.h"
#include "afxdialogex.h"
#include "des.h"
#include <string>
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
//加解密代码
void CEncrypt_Decrypt_MFCDlg::_3DES_Startup(string k1, string k2, string k3)//3des 密钥初始化
{
	srand((unsigned)time(NULL));
	for (int i = 0; i < 8; i++)
		//IV[i] = (unsigned char)(rand() % 255 + 1);
		IV[i] = (unsigned char)((i+2)*16+(i+5));//生成随机向量 IV
	//清空中间变量
	for (int i = 0; i < 100; i++)
	{
		m_temp[i] = "";
		c_temp[i] = "";
		p_temp[i] = "";
		ctr_temp[i] = "";
	}
	//string 转换成 unsigned char*类型
	int n1 = k1.length();
	int n2 = k2.length();
	int n3 = k3.length();
	unsigned char *key1 = new unsigned char[8];
	unsigned char *key2 = new unsigned char[8];
	unsigned char *key3 = new unsigned char[8];
	strcpy((char*)key1, k1.c_str());
	strcpy((char*)key2, k2.c_str());
	strcpy((char*)key3, k3.c_str());
	key1[7] = 0;
	key2[7] = 0;
	key3[7] = 0;
	des_setup((unsigned char*)key1, 8, 0, &s_k1);//初始化密钥 k1
	des_setup((unsigned char*)key2, 8, 0, &s_k2);//初始化密钥 k2
	des_setup((unsigned char*)key3, 8, 0, &s_k3);//初始化密钥 k3
}

string
```

```
CEncrypt_Decrypt_MFCDlg::_3DES_Encrypt(string m)//3des 加密函数
{
    unsigned char *pt = new unsigned char[m.length() + 1];
    unsigned char *temp1 = new unsigned char[m.length() + 1];
    unsigned char *temp2 = new unsigned char[m.length() + 1];
    unsigned char *ct = new unsigned char[m.length() + 1];
    for (int i = 0; i < m.length() + 1; i++)
    {
        pt[i] = 0;
        temp1[i] = 0;
        temp2[i] = 0;
        ct[i] = 0;
    }
    //string 转换成 unsigned char*类型
    strcpy((char *)pt, m.c_str());
    pt[m.length()] = 0;
    des_ecb_encrypt(pt, temp1, &s_k1);//用 k1 加密
    des_ecb_decrypt(temp1, temp2, &s_k2);//用 k2 解密
    des_ecb_encrypt(temp2, ct, &s_k3);//用 k3 加密
    string c((char *)ct);
    //unsigned char 转换成 string  类型
    return c;
}

string CEncrypt_Decrypt_MFCDlg::_3DES_Decrypt(string c)//3des 解密函数
{
    unsigned char *pt = new unsigned char[c.length() + 1];
    unsigned char *temp1 = new unsigned char[c.length() + 1];
    unsigned char *temp2 = new unsigned char[c.length() + 1];
    unsigned char *ct = new unsigned char[c.length() + 1];
    for (int i = 0; i < c.length() + 1; i++)
    {
        pt[i] = 0;
        temp1[i] = 0;
        temp2[i] = 0;
        ct[i] = 0;
    }
    //string 转换成 unsigned char *类型
    strcpy((char *)ct, c.c_str());
    ct[c.length()] = 0;
    des_ecb_decrypt(ct, temp2, &s_k3);//用 k3 解密
    des_ecb_encrypt(temp2, temp1, &s_k2);//用 k2 加密
    des_ecb_decrypt(temp1, pt, &s_k1);//用 k1 解密
    //unsigned char *转换成 string 类型
    string m((char *)pt);
    return m;
}

void CEncrypt_Decrypt_MFCDlg::Inintialize_Encrypt_Decrypt(string s1, string s2, string s3)//3des 初始化
{
```

```
    _3DES_Startup(s1, s2, s3);//调用密钥初始化函数
}

string CEncrypt_Decrypt_MFCDlg::Encrypt_All_Data_ECB_ZERO(string m)//ECB 模式加密，零字节填充
{
    string c = "";
    int k = 0;
    do
    {
        m_temp[k] = m.substr(8 * k, 8);
        //比特填充
        if (m_temp[k].length() < 8)
        {
            int count_sub = 8 - m_temp[k].length();

    m_temp[k].append(count_sub, 0x00);
        }
        c_temp[k] = _3DES_Encrypt(m_temp[k]);
        c = c + c_temp[k];
        k++;
    } while (8 * k < m.length());
    c_temp[k] = '\0';
    return c;
}

string CEncrypt_Decrypt_MFCDlg::Decrypt_All_Data_ECB_ZERO(string c)//ECB 模式解密，零字节填充
{
    int k = 0;
    string p = "";
    do
    {
        p_temp[k] = _3DES_Decrypt(c_temp[k]);
        //比特填充
        for (int i = 0; i < p_temp[k].length(); i++)
        if (p_temp[k][i] < 0x08)
            p_temp[k] = p_temp[k].substr(0, i);
        p = p + p_temp[k];
        k++;
    } while (c_temp[k] != '\0');
    return p;
}

string CEncrypt_Decrypt_MFCDlg::Encrypt_All_Data_ECB_PKCS7(string m)//ECB 模式加密，PKCS7 方式填充
{
    string c = "";
    int k = 0;
    do
    {
        m_temp[k] = m.substr(8 * k, 8);
        //比特填充
        if (m_temp[k].length() < 8)
        {
            int count_sub = 8 - m_temp[k].length();
```

```
        m_temp[k].append(count_sub,
(char)count_sub);
            }
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_ECB_PKCS7(string c)//ECB 模式解密,
PKCS7 方式填充
{
        int k = 0;
        string p = "";
        do
        {
            p_temp[k] =
_3DES_Decrypt(c_temp[k]);
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}
```

```
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_ECB_ANSIX923(string m)//ECB 模式加
密，ANSI X.923 方式填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)

        m_temp[k].append(count_sub - 1,
0x00);
                m_temp[k].append(1,
(char)count_sub);
            }
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_ECB_ANSIX923(string c)//ECB 模式解密,
```

```
ANSI X.923 方式填充
{
	int k = 0;
	string p = "";
	do
	{
		p_temp[k] =
_3DES_Decrypt(c_temp[k]);
		//比特填充
		for (int i = 0; i <
p_temp[k].length(); i++)
		if (p_temp[k][i] < 0x08)
			p_temp[k] =
p_temp[k].substr(0, i);
		p = p + p_temp[k];
		k++;
	} while (c_temp[k] != '\0');
	return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_ECB_ISO10126(string m)//ECB 模式加
密，ISO 10126 方式填充
{
	string c = "";
	int k = 0;
	do
	{
		m_temp[k] = m.substr(8 * k, 8);
		//比特填充
		if (m_temp[k].length()<8)
		{
			int count_sub = 8 -
m_temp[k].length();
			if (count_sub - 1>0)
			{

	srand((unsigned)time(NULL));
				unsigned int fill =
rand() % 256;

	m_temp[k].append(count_sub - 1,
(char)fill);
			}
			m_temp[k].append(1,
(char)count_sub);
		}
		c_temp[k] =
_3DES_Encrypt(m_temp[k]);
		c = c + c_temp[k];
		k++;
	} while (8 * k < m.length());
	c_temp[k] = '\0';
	return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_ECB_ISO10126(string c)//ECB 模式解密，
ISO 10126 方式填充
{
	int k = 0;
	string p = "";
	do
	{
		p_temp[k] =
_3DES_Decrypt(c_temp[k]);
		//比特填充
		for (int i = 0; i <
```

```
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, p_temp[k].length() -
(int)p_temp[k][i]);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CBC_ZERO(string m)//CBC 模式加密，零
字节填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);

            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub, 0x00);
            }
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ IV[i];
            else
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ c_temp[k - 1][i];
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CBC_ZERO(string c)//CBC 模式解密，零
字节填充
{
        int k = 0;
        string p = "";
        do
        {
            p_temp[k] =
_3DES_Decrypt(c_temp[k]);
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
IV[i];
            else
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
c_temp[k - 1][i];
            //比特填充
            for (int i = 0; i <
```

```
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}



string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CBC_PKCS7(string m)//CBC 模式加密，
PKCS7 方式填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub,
(char)count_sub);
            }
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ IV[i];
            else
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ c_temp[k - 1][i];
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CBC_PKCS7(string c)//CBC 模式解密，
PKCS7 方式填充
{
        int k = 0;
        string p = "";
        do
        {
            p_temp[k] =
_3DES_Decrypt(c_temp[k]);
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
IV[i];
            else
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
c_temp[k - 1][i];
            //比特填充
            for (int i = 0; i <
```

```
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CBC_ANSIX923(string m)//CBC 模式加
密，ANSI X.923 方式填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)

        m_temp[k].append(count_sub - 1,
0x00);
                m_temp[k].append(1,
(char)count_sub);
            }
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ IV[i];
            else
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ c_temp[k - 1][i];
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CBC_ANSIX923(string c)//CBC 模式解
密，ANSI X.923 方式填充
{
        int k = 0;
        string p = "";
        do
        {
            p_temp[k] =
_3DES_Decrypt(c_temp[k]);

            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
IV[i];
            else
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
```

```
c_temp[k - 1][i];
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);

            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CBC_ISO10126(string m)//CBC 模式加
密，ISO 10126 方式填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)
                {

        srand((unsigned)time(NULL));
                    unsigned int fill =
rand() % 256;
        m_temp[k].append(count_sub - 1,
(char)fill);
                }
                m_temp[k].append(1,
(char)count_sub);
            }
            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ IV[i];
            else
            for (int i = 0; i < 8; i++)
                m_temp[k][i] = m_temp[k][i]
^ c_temp[k - 1][i];
            c_temp[k] =
_3DES_Encrypt(m_temp[k]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CBC_ISO10126(string c)//CBC 模式解
密，ISO 10126 方式填充
{
        int k = 0;
        string p = "";
        do
        {
            p_temp[k] =
```

```
_3DES_Decrypt(c_temp[k]);

            //向量异或
            if (k == 0)
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
IV[i];
            else
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
c_temp[k - 1][i];
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, p_temp[k].length() -
(int)p_temp[k][i]);

            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}


string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_ZERO(string m)//CFB 模式加密，零
字节填充
{
        string c = "";
        int k = 0;
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub, 0x00);
            }
            //向量异或
            string iv((char *)IV);
            if (k == 0)
                c_temp[k] =
_3DES_Encrypt(iv);
            else
                c_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CFB_ZERO(string c)//CFB 模式解密，零
字节填充
{
        int k = 0;
        string p = "";
        do
```

```
    {
        string iv((char *)IV);
        if (k == 0)
            p_temp[k] =
_3DES_Encrypt(iv);
        else
            p_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
        p_temp[k] = p_temp[k].substr(0,
8);
        //向量异或
        for (int i = 0; i < 8; i++)
            p_temp[k][i] = p_temp[k][i] ^
c_temp[k][i];
        //比特填充
        for (int i = 0; i <
p_temp[k].length(); i++)
        if (p_temp[k][i] < 0x08)
            p_temp[k] =
p_temp[k].substr(0, i);
        p = p + p_temp[k];
        k++;
    } while (c_temp[k] != '\0');
    return p;
}



string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_PKCS7(string m)//CFB 模式加密,
PKCS7 方式填充
{
    string c = "";
    int k = 0;
    do
    {
        m_temp[k] = m.substr(8 * k, 8);
        //比特填充
        if (m_temp[k].length() < 8)
        {
            int count_sub = 8 -
m_temp[k].length();

    m_temp[k].append(count_sub,
(char)count_sub);
        }

        //向量异或
        string iv((char *)IV);
        if (k == 0)
            c_temp[k] =
_3DES_Encrypt(iv);
        else
            c_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
        for (int i = 0; i < 8; i++)
            c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
        c = c + c_temp[k];
        k++;
    } while (8 * k < m.length());
    c_temp[k] = '\0';
    return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CFB_PKCS7(string c)//CFB 模式解密,
PKCS7 方式填充
{
```

```
    int k = 0;
    string p = "";
    do
    {
        string iv((char *)IV);
        if (k == 0)
            p_temp[k] =
_3DES_Encrypt(iv);
        else
            p_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
        p_temp[k] = p_temp[k].substr(0,
8);
        //向量异或
        for (int i = 0; i < 8; i++)
            p_temp[k][i] = p_temp[k][i] ^
c_temp[k][i];
        //比特填充
        for (int i = 0; i <
p_temp[k].length(); i++)
        if (p_temp[k][i] < 0x08)
            p_temp[k] =
p_temp[k].substr(0, i);

        p = p + p_temp[k];
        k++;
    } while (c_temp[k] != '\0');
    return p;
}


string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_ANSIX923(string m)//CFB 模式加
密，ANSI X.923 方式填充
{
    string c = "";
    int k = 0;
    do
    {
        m_temp[k] = m.substr(8 * k, 8);

        //比特填充
        if (m_temp[k].length()<8)
        {
            int count_sub = 8 -
m_temp[k].length();
            if (count_sub - 1>0)

    m_temp[k].append(count_sub - 1,
0x00);
            m_temp[k].append(1,
(char)count_sub);
        }
        //向量异或
        string iv((char *)IV);
        if (k == 0)
            c_temp[k] =
_3DES_Encrypt(iv);
        else
            c_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
        for (int i = 0; i < 8; i++)
            c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
        c = c + c_temp[k];
        k++;
    } while (8 * k < m.length());
    c_temp[k] = '\0';
    return c;
```

```
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CFB_ANSIX923(string c)//CFB 模式解密，
ANSI X.923 方式填充
{
	int k = 0;
	string p = "";
	do
	{
		string iv((char *)IV);
		if (k == 0)
			p_temp[k] =
_3DES_Encrypt(iv);
		else
			p_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
		p_temp[k] = p_temp[k].substr(0,
8);
		//向量异或
		for (int i = 0; i < 8; i++)
			p_temp[k][i] = p_temp[k][i] ^
c_temp[k][i];
		//比特填充
		for (int i = 0; i <
p_temp[k].length(); i++)
		if (p_temp[k][i] < 0x08)
			p_temp[k] =
p_temp[k].substr(0, i);
		p = p + p_temp[k];
		k++;
	} while (c_temp[k] != '\0');
	return p;
}
```

```
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_ISO10126(string m)//CFB 模式加
密，ISO 10126 方式填充
{
	string c = "";
	int k = 0;
	do
	{
		m_temp[k] = m.substr(8 * k, 8);

		//比特填充
		if (m_temp[k].length()<8)
		{
			int count_sub = 8 -
m_temp[k].length();
			if (count_sub - 1>0)
			{

	srand((unsigned)time(NULL));
				unsigned int fill =
rand() % 256;

	m_temp[k].append(count_sub - 1,
(char)fill);
			}
			m_temp[k].append(1,
(char)count_sub);
		}
		//向量异或
		string iv((char *)IV);
		if (k == 0)
			c_temp[k] =
```

```
_3DES_Encrypt(iv);
            else
                c_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);

            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CFB_ISO10126(string c)//CFB 模式解密，
ISO 10126 方式填充
{
        int k = 0;
        string p = "";
        do
        {
            string iv((char *)IV);
            if (k == 0)
                p_temp[k] =
_3DES_Encrypt(iv);
            else
                p_temp[k] =
_3DES_Encrypt(c_temp[k - 1]);
            p_temp[k] = p_temp[k].substr(0,
8);
            //向量异或
            for (int i = 0; i < 8; i++)
                p_temp[k][i] = p_temp[k][i] ^
c_temp[k][i];
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, p_temp[k].length() -
(int)p_temp[k][i]);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_OFB_ZERO(string m)//OFB 模式加密，零
字节填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            c_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);

            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub, 0x00);
```

```
            }
            //向量异或
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);

            for (int i = 0; i < 8; i++)
                c_temp[k].append(1,
(char)key[i] ^ m_temp[k][i]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_OFB_ZERO(string c)//OFB 模式解密，零
字节填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            key = key.substr(0, 8);
            //向量异或
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1, key[i] ^
c_temp[k][i]);
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_OFB_PKCS7(string m)//OFB 模式加密，
PKCS7 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            c_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);

            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub,
(char)count_sub);
```

```
            }
            //向量异或
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            for (int i = 0; i < 8; i++)
                c_temp[k].append(1,
(char)key[i] ^ m_temp[k][i]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_OFB_PKCS7(string c)//OFB 模式解密,
PKCS7 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            key = key.substr(0, 8);
            //向量异或
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1, key[i] ^
c_temp[k][i]);
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_OFB_ANSIX923(string m)//OFB 模式加
密，ANSI X.923 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            c_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)

        m_temp[k].append(count_sub - 1,
0x00);
                m_temp[k].append(1,
```

```
(char)count_sub);
            }

            //向量异或
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            for (int i = 0; i < 8; i++)
                c_temp[k].append(1,
(char)key[i] ^ m_temp[k][i]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_OFB_ANSIX923(string c)//OFB 模式解
密，ANSI X.923 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            key = key.substr(0, 8);
            //向量异或
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1, key[i] ^
c_temp[k][i]);
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_OFB_ISO10126(string m)//OFB 模式加
密，ISO 10126 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            c_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)
                {
```

```
        srand((unsigned)time(NULL));
                        unsigned int fill =
rand() % 256;

        m_temp[k].append(count_sub - 1,
(char)fill);
                }
                m_temp[k].append(1,
(char)count_sub);
            }
            //向量异或
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            for (int i = 0; i < 8; i++)
                c_temp[k].append(1,
(char)key[i] ^ m_temp[k][i]);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = '\0';
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_OFB_ISO10126(string c)//OFB 模式解
密，ISO 10126 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            string iv((char *)IV);
            string key = iv;
            for (int i = 0; i < k + 1; i++)
                key = _3DES_Encrypt(key);
            key = key.substr(0, 8);
            //向量异或
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1, key[i] ^
c_temp[k][i]);
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, p_temp[k].length() -
(int)p_temp[k][i]);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != '\0');
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CTR_ZERO(string m)//CTR 模式加密，零
字节填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            ctr_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
```

```
            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();

        m_temp[k].append(count_sub, 0x00);
            }
            //向量异或
            ctr_temp[k].append(1, (char)k);
            while (ctr_temp[k].length() < 8)
                ctr_temp[k].append(1, '0');
            c_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c_temp[k] = c_temp[k].substr(0, 8);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CTR_ZERO(string c)//CTR 模式解密，零
字节填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            m_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1,
(char)(m_temp[k][i] ^ c_temp[k][i]));
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CTR_PKCS7(string m)//CTR 模式加密，
PKCS7 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            ctr_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length() < 8)
            {
                int count_sub = 8 -
m_temp[k].length();
```

```
        m_temp[k].append(count_sub,
(char)count_sub);
            }
            //向量异或
            ctr_temp[k].append(1, (char)k);
            while (ctr_temp[k].length() < 8)
                ctr_temp[k].append(1, '0');
            c_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c_temp[k] = c_temp[k].substr(0, 8);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CTR_PKCS7(string c)//CTR 模式解密,
PKCS7 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            m_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1,
(char)(m_temp[k][i] ^ c_temp[k][i]));
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}



string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CTR_ANSIX923(string m)//CTR 模式加
密，ANSI X.923 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            ctr_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)

        m_temp[k].append(count_sub - 1,
```

```
0x00);
                m_temp[k].append(1,
(char)count_sub);
            }
            //向量异或
            ctr_temp[k].append(1, (char)k);
            while (ctr_temp[k].length() < 8)
                ctr_temp[k].append(1, '0');
            c_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c_temp[k] = c_temp[k].substr(0, 8);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CTR_ANSIX923(string c)//CTR 模式解
密，ANSI X.923 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            m_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1,
(char)(m_temp[k][i] ^ c_temp[k][i]));
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, i);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}

string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CTR_ISO10126(string m)//CTR 模式加
密，ISO 10126 方式填充
{
        string c = "";
        int k = 0;
        for (int i = 0; i < 100; i++)
            ctr_temp[i] = "";
        do
        {
            m_temp[k] = m.substr(8 * k, 8);
            //比特填充
            if (m_temp[k].length()<8)
            {
                int count_sub = 8 -
m_temp[k].length();
                if (count_sub - 1>0)
                {

        srand((unsigned)time(NULL));
```

```
                    unsigned int fill =
rand() % 256;

        m_temp[k].append(count_sub - 1,
(char)fill);
                }
                m_temp[k].append(1,
(char)count_sub);
            }
            //向量异或
            ctr_temp[k].append(1, (char)k);
            while (ctr_temp[k].length() < 8)
                ctr_temp[k].append(1, '0');
            c_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                c_temp[k][i] = c_temp[k][i] ^
m_temp[k][i];
            c_temp[k] = c_temp[k].substr(0, 8);
            c = c + c_temp[k];
            k++;
        } while (8 * k < m.length());
        c_temp[k] = "\0";
        return c;
}

string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_CTR_ISO10126(string c)//CTR 模式解密，
ISO 10126 方式填充
{
        int k = 0;
        string p = "";
        for (int i = 0; i < 100; i++)
            p_temp[i] = "";
        do
        {
            m_temp[k] =
_3DES_Encrypt(ctr_temp[k]);
            for (int i = 0; i < 8; i++)
                p_temp[k].append(1,
(char)(m_temp[k][i] ^ c_temp[k][i]));
            //比特填充
            for (int i = 0; i <
p_temp[k].length(); i++)
            if (p_temp[k][i] < 0x08)
                p_temp[k] =
p_temp[k].substr(0, p_temp[k].length() -
(int)p_temp[k][i]);
            p = p + p_temp[k];
            k++;
        } while (c_temp[k] != "\0");
        return p;
}
// CEncrypt_Decrypt_MFCDlg 对话框
CEncrypt_Decrypt_MFCDlg::CEncrypt_Decry
pt_MFCDlg(CWnd* pParent /*=NULL*/):
CDialogEx(CEncrypt_Decrypt_MFCDlg::IDD,
pParent), m_encryptway(0), m_fillway(0),
m_enchoose(0)
{
        m_hIcon =
AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void
CEncrypt_Decrypt_MFCDlg::DoDataExchang
e(CDataExchange* pDX)
{
        CDialogEx::DoDataExchange(pDX);
```

```
    DDX_Control(pDX,
IDC_COMBO_KEYBYTES, m_cb_keybytes);
}

BEGIN_MESSAGE_MAP(CEncrypt_Decrypt_
MFCDlg, CDialogEx)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDOK,
&CEncrypt_Decrypt_MFCDlg::OnBnClicked
Ok)
    ON_BN_CLICKED(IDCANCEL,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedC
ancel)
    ON_BN_CLICKED(IDSETKEY,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedS
etkey)
    ON_BN_CLICKED(IDENCRYPT,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedE
ncrypt)
    ON_BN_CLICKED(IDDECRYPT,
&CEncrypt_Decrypt_MFCDlg::OnBnClicked
Decrypt)
    ON_BN_CLICKED(IDC_RADIO11,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio11)
    ON_BN_CLICKED(IDC_RADIO12,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio12)
    ON_BN_CLICKED(IDC_RADIO13,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio13)
    ON_BN_CLICKED(IDC_RADIO14,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio14)
    ON_BN_CLICKED(IDC_RADIO15,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio15)
    ON_BN_CLICKED(IDC_RADIO21,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio21)
    ON_BN_CLICKED(IDC_RADIO22,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio22)
    ON_BN_CLICKED(IDC_RADIO23,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio23)
    ON_BN_CLICKED(IDC_RADIO24,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio24)
    ON_BN_CLICKED(IDC_RADIO31,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio31)
    ON_BN_CLICKED(IDC_RADIO32,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedR
adio32)
    ON_BN_CLICKED(IDCHOOSEFILE,
&CEncrypt_Decrypt_MFCDlg::OnBnClickedC
hoosefile)
    ON_CBN_SELCHANGE(IDC_COMBO_K
EYBYTES,
&CEncrypt_Decrypt_MFCDlg::OnCbnSelcha
ngeComboKeybytes)
ON_WM_NCHITTEST()
ON_WM_CTLCOLOR()
ON_WM_TIMER()
END_MESSAGE_MAP()
// CEncrypt_Decrypt_MFCDlg 消息处理程
序
BOOL
```

```
CEncrypt_Decrypt_MFCDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();
    // 设置此对话框的图标。  当应用程序主窗口不是对话框时，框架将自动
    //  执行此操作
    SetIcon(m_hIcon, TRUE);            // 设置大图标
    SetIcon(m_hIcon, FALSE);        // 设置小图标
    // TODO:  在此添加额外的初始化代码
    SetTimer(1, 1000, NULL);
    m_cb_keybytes.AddString(L"8 字节");
    m_cb_keybytes.AddString(L"24 字节");
    GetDlgItem(IDENCRYPT)->ShowWindow(FALSE);
    GetDlgItem(IDDECRYPT)->ShowWindow(FALSE);
    GetDlgItem(IDCHOOSEFILE)->ShowWindow(FALSE);
    GetDlgItem(IDC_BITSTRING_EDIT)->ShowWindow(FALSE);//隐藏比特流框
    GetDlgItem(IDC_STATIC7)->ShowWindow(FALSE);
    GetDlgItem(IDC_PLAIN_EDIT)->EnableWindow(FALSE);
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->EnableWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY1)->ShowWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY2)->ShowWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY3)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC3)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC4)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC5)->ShowWindow(FALSE);
    GetDlgItem(IDC_PLAIN_EDIT)->ShowWindow(FALSE);//显示明文输入框
    GetDlgItem(IDC_STATIC1)->ShowWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY)->ShowWindow(FALSE);//隐藏密钥输入框
    GetDlgItem(IDSETKEY)->ShowWindow(FALSE);//隐藏设置密钥按钮
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->ShowWindow(FALSE);//隐藏密文框
    GetDlgItem(IDC_BITSTRING_EDIT)->ShowWindow(FALSE);//隐藏比特流框
    GetDlgItem(IDC_ORIGINAL_EDIT)->ShowWindow(FALSE);//隐藏原文框
    GetDlgItem(IDC_STATIC6)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC7)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC8)->ShowWindow(FALSE);
    GetDlgItem(IDC_BITSTRING_EDIT)->EnableWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY1)->EnableWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY2)->EnableWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY3)->EnableWindow(FALSE);
    GetDlgItem(IDC_RADIO11)->ShowWin
```

```
dow(FALSE);
	GetDlgItem(IDC_RADIO12)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO13)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO14)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO15)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO21)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO22)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO23)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_RADIO24)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_STATIC9)->ShowWin
dow(FALSE);
	GetDlgItem(IDC_STATIC10)->ShowWi
ndow(FALSE);
	return TRUE;  // 除非将焦点设置到控
件，否则返回 TRUE
}
// 如果向对话框添加最小化按钮，则需要
下面的代码
//  来绘制该图标。  对于使用文档/视图
模型的 MFC 应用程序，
//  这将由框架自动完成。

void CEncrypt_Decrypt_MFCDlg::OnPaint()
{
	CRect	rect;
	CPaintDC	dc(this);
	GetClientRect(rect);
	dc.FillSolidRect(rect,
RGB(186,226,234));	//背景
	if (IsIconic())
	{
		CPaintDC dc(this); // 用于绘制的
设备上下文

		SendMessage(WM_ICONERASEBKGND
,
reinterpret_cast<WPARAM>(dc.GetSafeHdc
()), 0);
		// 使图标在工作区矩形中居中
		int cxIcon =
GetSystemMetrics(SM_CXICON);
		int cyIcon =
GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1)
/ 2;
		int y = (rect.Height() - cyIcon + 1)
/ 2;
		// 绘制图标
		dc.DrawIcon(x, y, m_hIcon);
	}
	else CDialogEx::OnPaint();
}

//当用户拖动最小化窗口时系统调用此函数
取得光标
//显示。
HCURSOR
CEncrypt_Decrypt_MFCDlg::OnQueryDragIc
on()
{
```

```
    return
static_cast<HCURSOR>(m_hIcon);
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedOk
()//确认响应
{
}



void
CEncrypt_Decrypt_MFCDlg::OnBnClickedCa
ncel()//取消响应
{
    CDialogEx::OnCancel();
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedSet
key()//按下设置密钥按钮
{
        //显示 3 个密钥框
    GetDlgItem(IDC_EDIT_KEY1)->ShowWi
ndow(TRUE);
    GetDlgItem(IDC_EDIT_KEY2)->ShowWi
ndow(TRUE);
    GetDlgItem(IDC_EDIT_KEY3)->ShowWi
ndow(TRUE);
    GetDlgItem(IDC_STATIC3)->ShowWin
dow(TRUE);
    GetDlgItem(IDC_STATIC4)->ShowWin
dow(TRUE);
    GetDlgItem(IDC_STATIC5)->ShowWin
dow(TRUE);
    GetDlgItem(IDSETKEY)->ShowWindow
(FALSE);//禁用设置密钥按钮
    CString CKey;
    GetDlgItem(IDC_EDIT_KEY)->GetWind
owText(CKey);//获取密钥
    string skey;
    CStringA temp(CKey.GetBuffer(0));
    CKey.ReleaseBuffer();
    skey = temp.GetBuffer(0);
    temp.ReleaseBuffer();
    string key1, key2, key3;
    int nIndex =
m_cb_keybytes.GetCurSel();
    CString indexText;
    m_cb_keybytes.GetLBText(nIndex,
indexText);
    if (indexText==L"8 字节")//选择 8 字节
密钥
    {
        //相关提示信息
        if
(skey.length()<8)::MessageBox(NULL, L"密
钥少于 8 字节，系统将补零！", L"提示
",MB_OK|MB_ICONINFORMATION);
        if (skey.length()>8)
    ::MessageBox(NULL, L"密钥多于 8 字
节，系统去除多余部分！", L"提示", MB_OK
| MB_ICONINFORMATION);
        while (skey.length()<8)
skey.append(1, '0');//少于 8 字节补零
        skey = skey.substr(0, 8);//多于 8
字节截断
        //3 个密钥均为上述生成的 8 字节
密钥
        key1 = skey;
```

```
        key2 = skey;
        key3 = skey;
    }
    else if (indexText == L"24 字节")//选择24 字节密钥
    {
        //相关提示信息
        if (skey.length()<24)::MessageBox(NULL, L"密钥少于 24 字节，系统将补零！", L"提示", MB_OK | MB_ICONINFORMATION);
        if (skey.length()>24)::MessageBox(NULL, L"密钥多于 24 字节，系统去除多余部分！", L"提示", MB_OK | MB_ICONINFORMATION);
        while (skey.length() < 24)//少于24 字节补零
            skey.append(1, '0');
        skey = skey.substr(0, 24);//多于 24 字节截断
        //密钥 1 为 0-7 字节
        key1 = skey.substr(0, 8);
        //密钥 2 为 8-15 字节
        key2 = skey.substr(8, 8);
        //密钥 3 为 16-23 字节
        key3 = skey.substr(16, 8);
    }
    //将 3 个密钥转换为 CString 类型
    CString k1,k2,k3;
    k1 = CA2T(key1.c_str());
    k2 = CA2T(key2.c_str());
    k3 = CA2T(key3.c_str());
    Inintialize_Encrypt_Decrypt(key1, key2, key3);//初始化密钥
    //将三个密钥显示出来
    GetDlgItem(IDC_EDIT_KEY1)->SetWindowText(k1);
    GetDlgItem(IDC_EDIT_KEY2)->SetWindowText(k2);
    GetDlgItem(IDC_EDIT_KEY3)->SetWindowText(k3);
    GetDlgItem(IDSETKEY)->EnableWindow(FALSE);//禁用设置密钥按钮
    GetDlgItem(IDC_COMBO_KEYBYTES)->EnableWindow(FALSE);//禁用选择字节数选项
    GetDlgItem(IDC_EDIT_KEY)->EnableWindow(FALSE);//禁用密钥输入框
    GetDlgItem(IDENCRYPT)->ShowWindow(TRUE);//启用加密按钮
    //启用加密模式和填充模式选择键
    GetDlgItem(IDC_RADIO11)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO12)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO13)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO14)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO15)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO21)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO22)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO23)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO24)->ShowWindow(TRUE);
    GetDlgItem(IDC_STATIC9)->ShowWin
```

```
dow(TRUE);
    GetDlgItem(IDC_STATIC10)->ShowWindow(TRUE);
}

void CEncrypt_Decrypt_MFCDlg::OnBnClickedEncrypt()//按下加密按钮
{
    GetDlgItem(IDSETKEY)->EnableWindow(FALSE);//禁用设置密钥按钮
    GetDlgItem(IDC_COMBO_KEYBYTES)->EnableWindow(FALSE);//禁用密钥选择框
    GetDlgItem(IDC_EDIT_KEY)->EnableWindow(FALSE);//禁用密钥输入框
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->ShowWindow(TRUE);//显示密文框
    GetDlgItem(IDC_STATIC6)->ShowWindow(TRUE);
    GetDlgItem(IDSETKEY)->ShowWindow(FALSE);//禁用设置密钥按钮
    GetDlgItem(IDC_EDIT_KEY1)->ShowWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY2)->ShowWindow(FALSE);
    GetDlgItem(IDC_EDIT_KEY3)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC3)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC4)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC5)->ShowWindow(FALSE);
    if (m_enchoose == 0)
    {
        GetDlgItem(IDC_BITSTRING_EDIT)->ShowWindow(TRUE);//显示比特流框

        GetDlgItem(IDC_STATIC7)->ShowWindow(TRUE);
    }
    GetDlgItem(IDC_ORIGINAL_EDIT)->ShowWindow(FALSE);//隐藏明文框
    GetDlgItem(IDC_STATIC8)->ShowWindow(FALSE);
    //禁用加密模式和填充模式选择键
    GetDlgItem(IDC_PLAIN_EDIT)->EnableWindow(FALSE);
    GetDlgItem(IDC_RADIO11)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO12)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO13)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO14)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO15)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO21)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO22)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO23)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO24)->ShowWindow(FALSE);
    GetDlgItem(IDC_RADIO31)->EnableWindow(FALSE);
    GetDlgItem(IDC_RADIO32)->EnableWi
```

```
ndow(FALSE);
    GetDlgItem(IDC_STATIC9)->ShowWin
dow(FALSE);
    GetDlgItem(IDC_STATIC10)->ShowWi
ndow(FALSE);
    UpdateData(TRUE);
    CString CPlainText;
    if (m_enchoose==1)//选择加密文件
        CPlainText = cfileplain;//明文为文
件内容
    else

    GetDlgItem(IDC_PLAIN_EDIT)->GetWi
ndowText(CPlainText);//获取明文输入框信
息
    //CString 转换为 string 类型
    CStringA t(CPlainText.GetBuffer(0));
    CPlainText.ReleaseBuffer();
    string m_t = t.GetBuffer(0);//m_t 为明
文
    t.ReleaseBuffer();
    string c_t;
    //根据选择的加密模式和填充方式加密
    if (m_encryptway == 0 && m_fillway
== 0)c_t =
Encrypt_All_Data_ECB_ZERO(m_t);
    else if (m_encryptway == 0 &&
m_fillway == 1)c_t =
Encrypt_All_Data_ECB_PKCS7(m_t);
    else if (m_encryptway == 0 &&
m_fillway == 2)c_t =
Encrypt_All_Data_ECB_ANSIX923(m_t);
    else if (m_encryptway == 0 &&
m_fillway == 3)c_t =
Encrypt_All_Data_ECB_ISO10126(m_t);
    else if (m_encryptway == 1 &&
m_fillway == 0)c_t =
Encrypt_All_Data_CBC_ZERO(m_t);
    else if (m_encryptway == 1 &&
m_fillway == 1)c_t =
Encrypt_All_Data_CBC_PKCS7(m_t);
    else if (m_encryptway == 1 &&
m_fillway == 2)c_t =
Encrypt_All_Data_CBC_ANSIX923(m_t);
    else if (m_encryptway == 1 &&
m_fillway == 3)c_t =
Encrypt_All_Data_CBC_ISO10126(m_t);
    else if (m_encryptway == 2 &&
m_fillway == 0)c_t =
Encrypt_All_Data_CFB_ZERO(m_t);
    else if (m_encryptway == 2 &&
m_fillway == 1)c_t =
Encrypt_All_Data_CFB_PKCS7(m_t);
    else if (m_encryptway == 2 &&
m_fillway == 2)c_t =
Encrypt_All_Data_CFB_ANSIX923(m_t);
    else if (m_encryptway == 2 &&
m_fillway == 3)c_t =
Encrypt_All_Data_CFB_ISO10126(m_t);
    else if (m_encryptway == 3 &&
m_fillway == 0)c_t =
Encrypt_All_Data_OFB_ZERO(m_t);
    else if (m_encryptway == 3 &&
m_fillway == 1)c_t =
Encrypt_All_Data_OFB_PKCS7(m_t);
    else if (m_encryptway == 3 &&
m_fillway == 2)c_t =
Encrypt_All_Data_OFB_ANSIX923(m_t);
    else if (m_encryptway == 3 &&
m_fillway == 3)c_t =
```

```
Encrypt_All_Data_OFB_ISO10126(m_t);
        else if (m_encryptway == 4 &&
m_fillway == 0)c_t =
Encrypt_All_Data_CTR_ZERO(m_t);
        else if (m_encryptway == 4 &&
m_fillway == 1)c_t =
Encrypt_All_Data_CTR_PKCS7(m_t);
        else if (m_encryptway == 4 &&
m_fillway == 2)c_t =
Encrypt_All_Data_CTR_ANSIX923(m_t);
        else if (m_encryptway == 4 &&
m_fillway == 3)c_t =
Encrypt_All_Data_CTR_ISO10126(m_t);
        string temp0 = "";//比特流（16 进制）
        for (int i = 0; i < c_t.length(); i++)
        {
            int t0 = (int)c_t[i];//记录字符串的
某一位字符
            if (t0 < 0) t0 = 256 + t0;//负数处
理
            char c0, c1;
            c0 = (t0 % 16 < 10) ? (t0 % 16 +
'0') : (t0 % 16 - 10 + 'A');//16 进制的低位
            c1 = (t0 / 16 < 10) ? (t0 / 16 +
'0') : (t0 / 16 - 10 + 'A');//16 进制的高位
            //字符串追加字符
            temp0.append("0x");
            temp0.append(1, c1);
            temp0.append(1, c0);
            temp0.append(" ");
            if ((i + 1) % 7 == 0)
temp0.append("\r\n");//换行
        }
        CString CConText;
        CConText = CA2T(c_t.c_str());
        if (m_enchoose == 1)//文件加密
        {
            //将加密结果保存到文件中
            // 设置过滤器
            TCHAR szFilter[] = _T("文本文件
(*.txt)|*.txt|所有文件(*.*)|*.*||");
            // 构造保存文件对话框
            CFileDialog fileDlg(FALSE,
_T("txt"), NULL, OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT, szFilter, this);
            fileDlg.m_ofn.lpstrTitle = L"保存密
文文件";
            CString strFilePath;
            if (IDOK == fileDlg.DoModal())
            {
            // 如果点击了文件对话框上的“保
存”按钮，则将选择的文件路径显示到编辑
框里
                strFilePath =
fileDlg.GetPathName();
                // 判断文件是否存在,如果存
在则去掉只读属性
                if (PathFileExists(strFilePath)
&& !PathIsDirectory(strFilePath))
                {
                    DWORD dwAttrs =
GetFileAttributes(strFilePath);
                    if (dwAttrs !=
INVALID_FILE_ATTRIBUTES&& (dwAttrs &
FILE_ATTRIBUTE_READONLY))
                    {
                        dwAttrs &=
~FILE_ATTRIBUTE_READONLY;

    SetFileAttributes(strFilePath, dwAttrs);
```

```
                }
            }
            // 打开文件
            CStdioFile file;
            BOOL ret =
file.Open(strFilePath,CFile::modeCreate   |
CFile::modeWrite | CFile::shareDenyWrite);
            if (!ret)
            {
                ::AfxMessageBox(L"打开
文件失败");
                return;
            }
            file.SeekToEnd();
            file.WriteString(CConText);//
写入文件
            file.Close();//关闭文件
            strFilePath = L"<文件>" +
strFilePath;//显示文件路径

    GetDlgItem(IDC_CONFIDENTIAL_EDIT)
->SetWindowText(strFilePath);
        }

    }
    else//字符串加密
    {
GetDlgItem(IDC_CONFIDENTIAL_EDIT)->Se
tWindowText(CConText);//显示加密后的字
符串
        CString t1;
        t1 = CA2T(temp0.c_str());

    GetDlgItem(IDC_BITSTRING_EDIT)->Se
tWindowText(t1);//显示加密后的比特串
    }
    GetDlgItem(IDENCRYPT)->ShowWind
ow(FALSE);//禁用加密按钮
    GetDlgItem(IDDECRYPT)->ShowWind
ow(TRUE);//启用解密按钮
    GetDlgItem(IDC_ORIGINAL_EDIT)->Set
WindowText(L"");//清空解密框内容
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedDe
crypt()//按下解密按钮
{
    GetDlgItem(IDENCRYPT)->ShowWind
ow(TRUE);//启用加密按钮
    GetDlgItem(IDDECRYPT)->ShowWind
ow(FALSE);//禁用解密按钮
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)
->ShowWindow(FALSE);//隐藏密文框
    GetDlgItem(IDC_BITSTRING_EDIT)->Sh
owWindow(FALSE);//隐藏比特流框
    GetDlgItem(IDC_ORIGINAL_EDIT)->Sh
owWindow(TRUE);//显示原文框
    GetDlgItem(IDC_STATIC6)->ShowWin
dow(FALSE);
    GetDlgItem(IDC_STATIC7)->ShowWin
dow(FALSE);
    GetDlgItem(IDC_STATIC8)->ShowWin
dow(TRUE);
    UpdateData(TRUE);
    CString CConText;
    if (m_enchoose == 1)//解密文件
    {
        CString strFilePath;
GetDlgItem(IDC_CONFIDENTIAL_EDIT)->G
```

```
etWindowText(strFilePath);//获取文件路径
            strFilePath = strFilePath.Mid(4);//
去除密文框前面的<文件>标识符
            CStdioFile file;
            CString strText;
            //打开文件
            if (!file.Open(strFilePath,
CFile::modeRead))
            {
                ::AfxMessageBox(_T("文件打
开失败。"));
                return;
            }
            //读文件
            strText = _T("");
            CConText = L"";
            while (file.ReadString(strText))
                CConText = CConText +
strText;//追加到密文中
            //关闭文件
            file.Close();
        }
        else

        GetDlgItem(IDC_PLAIN_EDIT)->GetWi
ndowText(CConText);//获取密文框信息
        //CString 转换成 string 类型
        CStringA t(CConText.GetBuffer(0));
        CConText.ReleaseBuffer();
        string c_t = t.GetBuffer(0);//c_t 为密文
        t.ReleaseBuffer();
        string m_t;
        //根据选择的加密方式和填充模式解密
        if (m_encryptway == 0 && m_fillway
== 0)m_t =
Decrypt_All_Data_ECB_ZERO(c_t);
        else if (m_encryptway == 0 &&
m_fillway == 1)m_t =
Decrypt_All_Data_ECB_PKCS7(c_t);
        else if (m_encryptway == 0 &&
m_fillway == 2)m_t =
Decrypt_All_Data_ECB_ANSIX923(c_t);
        else if (m_encryptway == 0 &&
m_fillway == 3)m_t =
Decrypt_All_Data_ECB_ISO10126(c_t);
        else if (m_encryptway == 1 &&
m_fillway == 0)m_t =
Decrypt_All_Data_CBC_ZERO(c_t);
        else if (m_encryptway == 1 &&
m_fillway == 1)m_t =
Decrypt_All_Data_CBC_PKCS7(c_t);
        else if (m_encryptway == 1 &&
m_fillway == 2)m_t =
Decrypt_All_Data_CBC_ANSIX923(c_t);
        else if (m_encryptway == 1 &&
m_fillway == 3)m_t =
Decrypt_All_Data_CBC_ISO10126(c_t);
        else if (m_encryptway == 2 &&
m_fillway == 0)m_t =
Decrypt_All_Data_CFB_ZERO(c_t);
        else if (m_encryptway == 2 &&
m_fillway == 1)m_t =
Decrypt_All_Data_CFB_PKCS7(c_t);
        else if (m_encryptway == 2 &&
m_fillway == 2)m_t =
Decrypt_All_Data_CFB_ANSIX923(c_t);
        else if (m_encryptway == 2 &&
m_fillway == 3)m_t =
Decrypt_All_Data_CFB_ISO10126(c_t);
        else if (m_encryptway == 3 &&
```

```
m_fillway == 0)m_t =
Decrypt_All_Data_OFB_ZERO(c_t);
        else if (m_encryptway == 3 &&
m_fillway == 1)m_t =
Decrypt_All_Data_OFB_PKCS7(c_t);
        else if (m_encryptway == 3 &&
m_fillway == 2)m_t =
Decrypt_All_Data_OFB_ANSIX923(c_t);
        else if (m_encryptway == 3 &&
m_fillway == 3)m_t =
Decrypt_All_Data_OFB_ISO10126(c_t);
        else if (m_encryptway == 4 &&
m_fillway == 0)m_t =
Decrypt_All_Data_CTR_ZERO(c_t);
        else if (m_encryptway == 4 &&
m_fillway == 1)m_t =
Decrypt_All_Data_CTR_PKCS7(c_t);
        else if (m_encryptway == 4 &&
m_fillway == 2)m_t =
Decrypt_All_Data_CTR_ANSIX923(c_t);
        else if (m_encryptway == 4 &&
m_fillway == 3)m_t =
Decrypt_All_Data_CTR_ISO10126(c_t);
        //string 转换为 cstring 类型
        CString COriText;
        COriText = CA2T(m_t.c_str());
        if (m_enchoose == 1)//解密文件
        {
            // 设置过滤器
            TCHAR szFilter[] = _T("文本文件
(*.txt)|*.txt|所有文件(*.*)|*.*||");
            // 构造保存文件对话框
            CFileDialog fileDlg(FALSE,
_T("txt"), NULL, OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT, szFilter, this);
            fileDlg.m_ofn.lpstrTitle = L"保存原
文文件";
            CString strFilePath;
            if (IDOK == fileDlg.DoModal())
            {
            // 如果点击了文件对话框上的“保
存”按钮，则将选择的文件路径显示到编辑
框里
                strFilePath =
fileDlg.GetPathName();
                // 判断文件是否存在,如果存
在则去掉只读属性
                if (PathFileExists(strFilePath)
&& !PathIsDirectory(strFilePath))
                {
                    DWORD dwAttrs =
GetFileAttributes(strFilePath);
                    if (dwAttrs !=
INVALID_FILE_ATTRIBUTES
                        && (dwAttrs &
FILE_ATTRIBUTE_READONLY))
                    {
                        dwAttrs &=
~FILE_ATTRIBUTE_READONLY;

    SetFileAttributes(strFilePath, dwAttrs);
                    }
                }
                // 打开文件
                CStdioFile file;
                BOOL ret =
file.Open(strFilePath, CFile::modeCreate |
CFile::modeWrite | CFile::shareDenyWrite);
                if (!ret)
                {
```

```
                    ::AfxMessageBox(L"打开文件失败");
                    return;
                }
                file.SeekToEnd();
                file.WriteString(COriText);//写入文件
                file.Close();
                strFilePath = L"<文件>" + strFilePath;//记录文件路径

    GetDlgItem(IDC_ORIGINAL_EDIT)->SetWindowText(strFilePath);//显示文件路径
            }
    }
    else//解密字符串

    GetDlgItem(IDC_ORIGINAL_EDIT)->SetWindowText(COriText);//显示解密后的原文
    GetDlgItem(IDSETKEY)->EnableWindow(TRUE);//启用设置密钥按钮
    GetDlgItem(IDC_COMBO_KEYBYTES)->EnableWindow(TRUE);//启用设置密钥位数框
    GetDlgItem(IDC_EDIT_KEY)->EnableWindow(TRUE);//启用密钥输入框
    //启用加密模式和填充模式选择键
    GetDlgItem(IDC_RADIO11)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO12)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO13)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO14)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO15)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO21)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO22)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO23)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO24)->ShowWindow(TRUE);
    GetDlgItem(IDC_RADIO31)->EnableWindow(TRUE);
    GetDlgItem(IDC_RADIO32)->EnableWindow(TRUE);
    GetDlgItem(IDC_STATIC9)->ShowWindow(TRUE);
    GetDlgItem(IDC_STATIC10)->ShowWindow(TRUE);
    GetDlgItem(IDC_PLAIN_EDIT)->EnableWindow(TRUE);//启用明文输入框
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->SetWindowText(L"");//清空密文框
    GetDlgItem(IDC_BITSTRING_EDIT)->SetWindowText(L"");//清空比特流框
    GetDlgItem(IDSETKEY)->ShowWindow(TRUE);//启用设置密钥按钮
}

void CEncrypt_Decrypt_MFCDlg::OnBnClickedRadio11()//ECB 模式
{
    m_encryptway = 0;
}
```

```
void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio12()//CBC 模式
{
	m_encryptway = 1;
}


void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio13()//CFB 模式
{
	m_encryptway = 2;
}


void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio14()//OFB 模式
{
	m_encryptway = 3;
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio15()//CTR 模式
{
		m_encryptway = 4;
}


void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio21()//ZERO 模式
{
	m_fillway = 0;
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio22()//PKCS7 模式
{
		m_fillway = 1;
}

void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio23()//ANSI X.923 模式
{
	m_fillway = 2;
}


void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio24()//ISO 10126 模式
{
	m_fillway = 3;
}


void
CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
dio31()//选择加密字符串方式
{
	m_enchoose = 0;
	GetDlgItem(IDC_PLAIN_EDIT)->Show
Window(TRUE);//显示明文输入框
	GetDlgItem(IDC_STATIC1)->ShowWin
dow(TRUE);
```

```
    GetDlgItem(IDCHOOSEFILE)->ShowWindow(FALSE);//禁用选择文件按钮
    GetDlgItem(IDC_PLAIN_EDIT)->EnableWindow(TRUE);//启用明文输入框
    GetDlgItem(IDC_PLAIN_EDIT)->SetWindowText(L"");//清空明文输入框
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->SetWindowText(L"");//清空密文框
    GetDlgItem(IDC_ORIGINAL_EDIT)->SetWindowText(L"");//清空原文框
}


void CEncrypt_Decrypt_MFCDlg::OnBnClickedRadio32()//选择加密文件方式
{
    m_enchoose = 1;
    GetDlgItem(IDC_STATIC7)->ShowWindow(FALSE);
    GetDlgItem(IDC_STATIC1)->ShowWindow(TRUE);
    GetDlgItem(IDC_PLAIN_EDIT)->ShowWindow(FALSE);//隐藏明文输入框
    GetDlgItem(IDCHOOSEFILE)->ShowWindow(TRUE);//启用选择文件按钮
    GetDlgItem(IDC_PLAIN_EDIT)->EnableWindow(FALSE);//禁用明文输入框
    GetDlgItem(IDC_PLAIN_EDIT)->SetWindowText(L"");//清空明文框
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)->SetWindowText(L"");//清空密文框
    GetDlgItem(IDC_ORIGINAL_EDIT)->SetWindowText(L"");//清空原文框
}


void CEncrypt_Decrypt_MFCDlg::OnBnClickedChoosefile()
{
    // 设置过滤器
    TCHAR szFilter[] = _T("文本文件(*.txt)|*.txt|所有文件(*.*)|*.*||");
    // 构造打开文件对话框
    CFileDialog fileDlg(TRUE, _T("txt"), NULL, 0, szFilter, this);
    fileDlg.m_ofn.lpstrTitle = L"打开明文文件";
    CString strFilePath;
    if (IDOK == fileDlg.DoModal())
    {
        // 如果点击了文件对话框上的“打开”按钮，则将选择的文件路径显示到编辑框里
        strFilePath = fileDlg.GetPathName();
        CStdioFile file;
        CString strText;
        //打开文件
        if (!file.Open(strFilePath, CFile::modeRead))
        {
            ::AfxMessageBox(_T("文件打开失败。"));
            return;
        }
        //读文件
        strText = _T("");
        cfileplain = L"";
        while (file.ReadString(strText))
```

```
                cfileplain = cfileplain +
strText;
            //关闭文件
            file.Close();

    GetDlgItem(IDC_PLAIN_EDIT)->Show
Window(TRUE);//显示明文输入框
            strFilePath = L"<文件>" +
strFilePath;

    GetDlgItem(IDC_PLAIN_EDIT)->SetWin
dowText(strFilePath);
    }
}

void
CEncrypt_Decrypt_MFCDlg::OnCbnSelchan
geComboKeybytes()
{
    GetDlgItem(IDC_EDIT_KEY)->ShowWin
dow(TRUE);//显示密钥输入框
    GetDlgItem(IDSETKEY)->ShowWindow
(TRUE);//显示设置密钥按钮
    GetDlgItem(IDC_EDIT_KEY)->SetWind
owText(L"");//密钥输入框清空
}

LRESULT
CEncrypt_Decrypt_MFCDlg::OnNcHitTest(C
Point point)
{
    int ret = CDialog::OnNcHitTest(point);
    //禁止改变窗口大小
    if (HTTOP == ret || HTBOTTOM == ret
|| HTLEFT == ret || HTRIGHT == ret ||
HTBOTTOMLEFT == ret ||
HTBOTTOMRIGHT == ret || HTTOPLEFT ==
ret || HTTOPRIGHT == ret || HTCAPTION
== ret)
        return HTCLIENT;
    return ret;
}

HBRUSH
CEncrypt_Decrypt_MFCDlg::OnCtlColor(CD
C* pDC, CWnd* pWnd, UINT nCtlColor)
{
    HBRUSH hbr =
CDialog::OnCtlColor(pDC, pWnd,
nCtlColor);
    // TODO:  Change any attributes of
the DC here
    if (nCtlColor == CTLCOLOR_BTN)
//更改按钮颜色
    {       pDC->SetTextColor(RGB(254,
1, 1));
        pDC->SetBkColor(RGB(117, 31,
111));
        HBRUSH b =
CreateSolidBrush(RGB(186, 226, 234));
        return b;
    }
    else if (nCtlColor == CTLCOLOR_EDIT)
//更改编辑框
    {

        pDC->SetTextColor(RGB(24, 6,
102));
        pDC->SetBkColor(RGB(254, 198,
103));
```

```
            HBRUSH b =
CreateSolidBrush(RGB(186, 226, 234));
            return b;
        }
        if (nCtlColor == CTLCOLOR_STATIC)
//更改静态文本
        {
            pDC->SetTextColor(RGB(254, 1,
1));
            pDC->SetBkColor(RGB(186, 226,
234));
            HBRUSH b =
CreateSolidBrush(RGB(186, 226, 234));
            return b;
        }
        return hbr;
}

void
CEncrypt_Decrypt_MFCDlg::OnTimer(UINT_
PTR nIDEvent)
{
        CString strTime;
        CTime tm;
        tm = CTime::GetCurrentTime();
        strTime =
tm.Format("%Y-%m-%d %H:%M:%S");
        SetDlgItemText(IDC_ShowTime,
strTime);              //显示系统时间
        CDialogEx::OnTimer(nIDEvent);
}
```

4.2 单重 DES 加密与解密的实现

```
//G_des.c
//des 加密与解密的实现
#include "des.h"
#ifdef DES
#define EN0 0
#define DE1 1
#endif
static void cookey(const ulong32 *raw1,
ulong32 *keyout);
#ifdef CLEAN_STACK
void _deskey(const unsigned char *key,
short edf, ulong32 *keyout)
#else
void deskey(const unsigned char *key,
short edf, ulong32 *keyout)
#endif
{
        ulong32 i, j, l, m, n, kn[32];
        unsigned char pc1m[56], pcr[56];
        for (j = 0; j < 56; j++) {
                l = (ulong32)pc1[j];
                m = l & 7;
                pc1m[j] = (unsigned
char)((key[l >> 3U] & bytebit[m]) ==
bytebit[m] ? 1 : 0);
        }
        for (i = 0; i < 16; i++)
{                       /*生成加密或者解密用
的 16 轮子密钥*/
                if (edf == DE1) m = (15 - i) << 1;
                else m = i << 1;
                n = m + 1;
                kn[m] = kn[n] = 0L;
                for (j = 0; j < 28; j++)
```

```
{                   /*子密钥的前半部分循环移位*/
                    l = j + (ulong32)totrot[i];
                    if (l < 28) pcr[j] = pc1m[l];
                    else pcr[j] = pc1m[l - 28];
                }
                for (/*j = 28*/; j < 56; j++)
{           /*子密钥的后半部分循环移位*/
                    l = j + (ulong32)totrot[i];
                    if (l < 56) pcr[j] = pc1m[l];
                    else pcr[j] = pc1m[l - 28];
                }
                for (j = 0; j < 24; j++)
{                   /*对 48bit 密钥进行置换*/
                    if ((int)pcr[(int)pc2[j]] != 0)
kn[m] |= bigbyte[j];
                    if ((int)pcr[(int)pc2[j + 24]] !=
0) kn[n] |= bigbyte[j];
                }
        }
        cookey(kn, keyout);
}
#ifdef CLEAN_STACK
void deskey(const unsigned char *key,
short edf, ulong32 *keyout)
{
        _deskey(key, edf, keyout);
        burn_stack(sizeof(int)* 5 +
sizeof(ulong32)* 32 + sizeof(unsigned
char)* 112);
}
#endif
#ifdef CLEAN_STACK
static void _cookey(const ulong32 *raw1,
ulong32 *keyout)
#else
static void cookey(const ulong32 *raw1,
ulong32 *keyout)
#endif
{
        ulong32 *cook;
        const ulong32 *raw0;
        ulong32 dough[32];
        int i;
        cook = dough;
for (i = 0; i < 16; i++, raw1++)   /*把子密钥
按平均分成 8 组，重新按 1、3、5、7、
2、4、6、8 排序*/
        {
                raw0 = raw1++;
                *cook = (*raw0 & 0x00fc0000L)
<< 6;
                *cook |= (*raw0 & 0x00000fc0L)
<< 10;
                *cook |= (*raw1 &
0x00fc0000L) >> 10;
                *cook++ |= (*raw1 &
0x00000fc0L) >> 6;
                *cook = (*raw0 & 0x0003f000L)
<< 12;
                *cook |= (*raw0 & 0x0000003fL)
<< 16;
                *cook |= (*raw1 &
0x0003f000L) >> 4;
                *cook++ |= (*raw1 &
0x0000003fL);
        }
        memcpy(keyout, dough, sizeof
dough);
}
```

```
#ifdef CLEAN_STACK
static void cookey(const ulong32 *raw1,
ulong32 *keyout)
{
    _cookey(raw1, keyout);
    burn_stack(sizeof(ulong32 *)* 2 +
sizeof(ulong32)* 32 + sizeof(int));
}
#endif
#ifndef CLEAN_STACK
static void desfunc(ulong32 *block, const
ulong32 *keys)
#else
static void _desfunc(ulong32 *block, const
ulong32 *keys)
#endif
{
    ulong32 work, right, leftt;
    int round;
    leftt = block[0];
    right = block[1];
#ifdef SMALL_CODE
    work = ((leftt >> 4) ^ right) &
0x0f0f0f0fL;
    right ^= work;
    leftt ^= (work << 4);
    work = ((leftt >> 16) ^ right) &
0x0000ffffL;
    right ^= work;
    leftt ^= (work << 16);
    work = ((right >> 2) ^ leftt) &
0x33333333L;
    leftt ^= work;
    right ^= (work << 2);
    work = ((right >> 8) ^ leftt) &
0x00ff00ffL;
    leftt ^= work;
    right ^= (work << 8);
    right = ROL(right, 1);
    work = (leftt ^ right) & 0xaaaaaaaaL;
    leftt ^= work;
    right ^= work;
    leftt = ROL(leftt, 1);
#else
    {
        ulong64 tmp;
/*加密或解密时进行初始置换*/
        tmp = des_ip[0][byte(leftt, 0)] ^
            des_ip[1][byte(leftt, 1)] ^
            des_ip[2][byte(leftt, 2)] ^
            des_ip[3][byte(leftt, 3)] ^
            des_ip[4][byte(right, 0)] ^
            des_ip[5][byte(right, 1)] ^
            des_ip[6][byte(right, 2)] ^
            des_ip[7][byte(right, 3)];
        leftt = (ulong32)(tmp >> 32);
        right = (ulong32)(tmp &
0xFFFFFFFFUL);
    }
#endif
for (round = 0; round < 8; round++) {   /*
对置换后的信息进行运算，主要包括信息
扩展和查 s 盒*/
work = ROR(right, 4) ^ *keys++;     /*一共
8 轮循环，每轮进行加密或解密的两轮运
算*/
        leftt ^= SP7[work & 0x3fL]
            ^ SP5[(work >> 8) & 0x3fL]
            ^ SP3[(work >> 16) & 0x3fL]
            ^ SP1[(work >> 24) & 0x3fL];
```

```
            work = right ^ *keys++;
            leftt ^= SP8[work & 0x3fL]
                  ^ SP6[(work >> 8) & 0x3fL]
                  ^ SP4[(work >> 16) & 0x3fL]
                  ^ SP2[(work >> 24) & 0x3fL];
            work = ROR(leftt, 4) ^ *keys++;
            right ^= SP7[work & 0x3fL]
                  ^ SP5[(work >> 8) & 0x3fL]
                  ^ SP3[(work >> 16) & 0x3fL]
                  ^ SP1[(work >> 24) & 0x3fL];
            work = leftt ^ *keys++;
            right ^= SP8[work & 0x3fL]
                  ^ SP6[(work >> 8) & 0x3fL]
                  ^ SP4[(work >> 16) & 0x3fL]
                  ^ SP2[(work >> 24) & 0x3fL];
      }
#ifdef SMALL_CODE
      right = ROR(right, 1);
      work = (leftt ^ right) & 0xaaaaaaaaL;
      leftt ^= work;
      right ^= work;
      leftt = ROR(leftt, 1);
      work = ((leftt >> 8) ^ right) &
0x00ff00ffL;
      right ^= work;
      leftt ^= (work << 8);
      work = ((leftt >> 2) ^ right) &
0x33333333L;
      right ^= work;
      leftt ^= (work << 2);
      work = ((right >> 16) ^ leftt) &
0x0000ffffL;
      leftt ^= work;
      right ^= (work << 16);
      work = ((right >> 4) ^ leftt) &
0x0f0f0f0fL;
      leftt ^= work;
      right ^= (work << 4);
#else
      {
            ulong64 tmp;
            tmp = des_fp[0][byte(leftt, 0)] ^
                  des_fp[1][byte(leftt, 1)] ^
                  des_fp[2][byte(leftt, 2)] ^
                  des_fp[3][byte(leftt, 3)] ^
                  des_fp[4][byte(right, 0)] ^
                  des_fp[5][byte(right, 1)] ^
                  des_fp[6][byte(right, 2)] ^
                  des_fp[7][byte(right, 3)];
            leftt = (ulong32)(tmp >> 32);
            right = (ulong32)(tmp &
0xFFFFFFFFUL);
      }
#endif
      block[0] = right;
      block[1] = leftt;
}
#ifdef CLEAN_STACK
static void desfunc(ulong32 *block, const
ulong32 *keys)
{
      _desfunc(block, keys);
      burn_stack(sizeof(ulong32)* 4 +
sizeof(int));
}
#endif
int des_setup(const unsigned char *key, int
keylen, int num_rounds, symmetric_key
*skey)
{
```

```
    _ARGCHK(key != NULL);
    _ARGCHK(skey != NULL);
    if (num_rounds != 0 &&
num_rounds != 16)          return
CRYPT_INVALID_ROUNDS;
    if (keylen != 8)           return
CRYPT_INVALID_KEYSIZE;
    deskey(key, EN0, skey->ek);
    deskey(key, DE1, skey->dk);
    return CRYPT_OK;
}

void des_ecb_encrypt(const unsigned char
*pt, unsigned char *ct, symmetric_key *key)
{
    ulong32 work[2];
    _ARGCHK(pt != NULL);
    _ARGCHK(ct != NULL);
    _ARGCHK(key != NULL);
    LOAD32H(work[0], pt + 0);
    LOAD32H(work[1], pt + 4);
    desfunc(work, key->ek);
    STORE32H(work[0], ct + 0);
    STORE32H(work[1], ct + 4);
}

void des_ecb_decrypt(const unsigned char
*ct, unsigned char *pt, symmetric_key *key)
{
    ulong32 work[2];
    _ARGCHK(pt != NULL);
    _ARGCHK(ct != NULL);
    _ARGCHK(key != NULL);
    LOAD32H(work[0], ct + 0);
    LOAD32H(work[1], ct + 4);
    desfunc(work, key->dk);
    STORE32H(work[0], pt + 0);
    STORE32H(work[1], pt + 4);
}

int des_test(void)
{
#ifndef LTC_TEST
    return CRYPT_NOP;
#else
    int err;
    };
    int i, y;
    unsigned char tmp[8];
    symmetric_key des;
    for (i = 0; i < (int)(sizeof(cases) /
sizeof(cases[0])); i++)
    {
        if ((err = des_setup(cases[i].key, 8,
0, &des)) != CRYPT_OK) return err;
        if (cases[i].mode != 0)
des_ecb_encrypt(cases[i].txt, tmp, &des);
        else des_ecb_decrypt(cases[i].txt,
tmp, &des);
        if (memcmp(cases[i].out, tmp,
sizeof(tmp)) != 0)return
CRYPT_FAIL_TESTVECTOR;
        for (y = 0; y < 8; y++) tmp[y] = 0;
        for (y = 0; y < 1000; y++)
des_ecb_encrypt(tmp, tmp, &des);
        for (y = 0; y < 1000; y++)
des_ecb_decrypt(tmp, tmp, &des);
        for (y = 0; y < 8; y++) if (tmp[y] !=
0) return CRYPT_FAIL_TESTVECTOR;
    }
```

```
        return CRYPT_OK;
#endif
}

int des_keysize(int *desired_keysize)
{
        _ARGCHK(desired_keysize != NULL);
        if (*desired_keysize < 8) {
                return CRYPT_INVALID_KEYSIZE;
        }
        *desired_keysize = 8;
        return CRYPT_OK;
}

void zeromem(void *dst, size_t len)
{
        unsigned char *mem = (unsigned char
*)dst;
        _ARGCHK(dst != NULL);
        while (len-- > 0)
                *mem++ = 0;
}
void burn_stack(unsigned long len)
{
        unsigned char buf[32];
        zeromem(buf, sizeof(buf));
        if (len > (unsigned long)sizeof(buf))
                burn_stack(len - sizeof(buf));
}
#include <signal.h>
#if (ARGTYPE == 0)
void crypt_argchk(char *v, char *s, int d)
{
        fprintf(stderr, "_ARGCHK '%s' failure on
line %d of file %s\n",
                v, d, s);
        (void)raise(SIGABRT);
}
#endif
#endif
```

5.程序运行结果

5.1 ECB 模式加解密

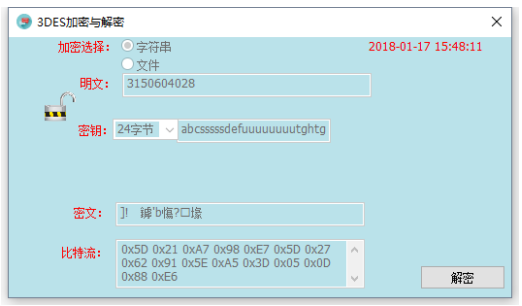

图 23 ECB 模式加密运行结果

图 24 ECB 模式解密运行结果

5.2 CBC 模式加解密

图 25 CBC 模式加密运行结果

图 26 CBC 模式解密运行结果

5.3 CFB 模式加解密

图 27 CFB 模式加密运行结果

图 28 CFB 模式解密运行结果

5.4 OFB 模式加解密

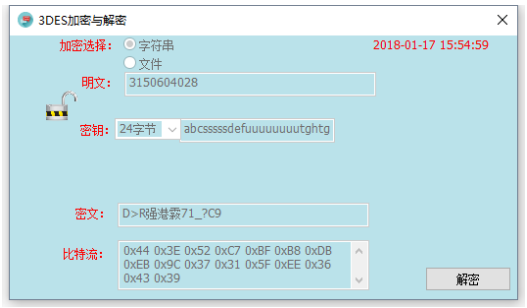

图 29 OFB 模式加密运行结果

图 30 OFB 模式解密运行结果

5.5 CTR 模式加解密

图 31 CTR 模式加密运行结果

图 32 CTR 模式解密运行结果

6.总结

通过这次课程设计，我对基本的 3des 的加密和解密方法有了一个更熟练的掌握，能把学习到的理论知识更好地运用到实际的操作中来。此外，这次课程设计也让我的逻辑思维能力得到一定的提升，让我更好地更有条理地处理一系列的问题。在课程设计中，我难免会遇到各种各样的问题，一开始加密失败，后来解密又无法恢复明文。面对这些问题，我冷静分析，在程序中仔细耐心地调试，寻找出错的原因并加以改进，最终得到了满意的运行结果。

通过一个星期的课程设计，虽然过程并不是一帆风顺的，但是最终程序能够完美地运行出正确的结果。我懂得了要把书本上的理论知识运用到实际的动手操作中并在设计中发现和解决问题，这样才能有所提高。