

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:______<u>沈鑫楠</u>____

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	3
1.1 问题描述	3
1.2 设计要求	3
2.开发环境与工具	3
2.1 开发环境	3
2.1.1 硬件平台	3
2.1.2 软件平台	4
2.2 开发工具	4
3.设计原理与流程图	4
3.1 设计原理	4
3.1.1 总体设计	4
3.1.2 详细设计	5
3.2 具体实现算法与流程图	11
3.2.1 分组密码工作模式	11
3.2.2 分组密码填充模式	14
3.2.3 流程图	15
4.程序主要编码实现	18
4.1 3DES 分组密码工作模式和填充模式的实现	18
4.2 单重 DES 加密与解密的实现	55
5.程序运行结果	60
5.1 ECB 模式加解密	60
5.2 CBC 模式加解密	61
5.3 CFB 模式加解密	61
5.4 OFB 模式加解密	61
5.5 CTR 模式加解密	62
6 单结	62

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。

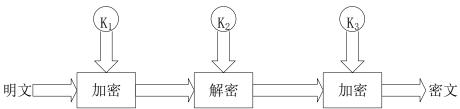


图 13DES加密流程

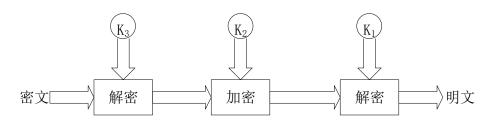
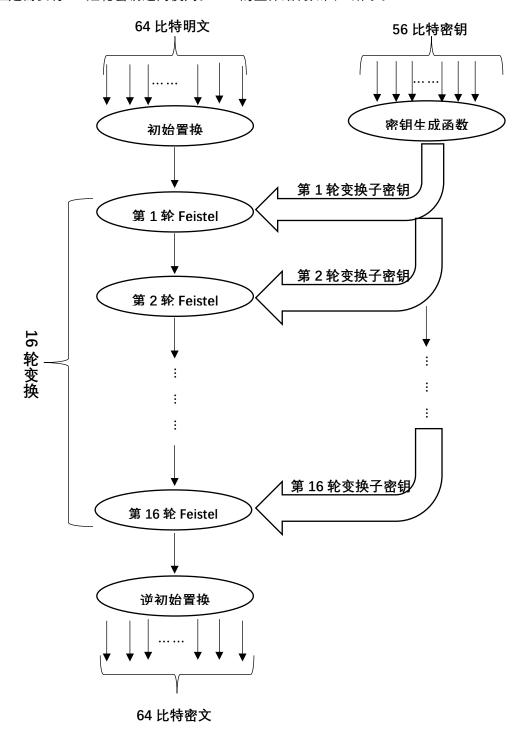


图 23DES解密流程

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。



(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。

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

$$R_{i} = L_{i-1} \oplus F(R_{i-1}, K_{i})$$

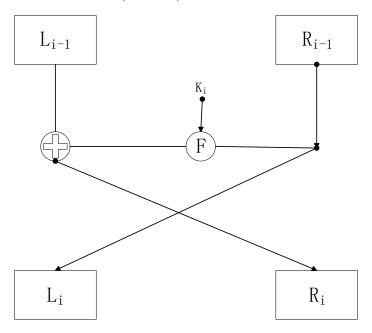


图 4 单轮 Feistel 体制

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:

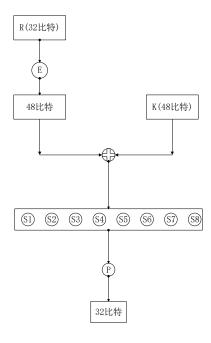


图 5 DES 的 F 函数示意图

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 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
							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
	1	1		ı	1	ı	盒	S ₄	ı	1	1	ı	1	ı	
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
	T	T		ı	T	ı	盒	S ₅	ı	T	T	ı	T	ı	
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
	1	1		ı	1	ı	盒	S ₆	ı	1	1	ı	1	ı	
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
	1	1		ı	1	I	盒	S ₇	ı	1	1	ı	1	ı	
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
	1	1		ı	1	ı	盒	S ₈	ı	1	1	ı	1	ı	
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-1和 D-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。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。

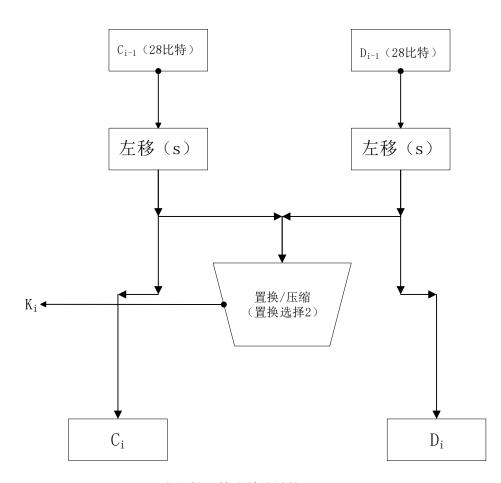


图 11 密钥扩展算法单轮结构

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。

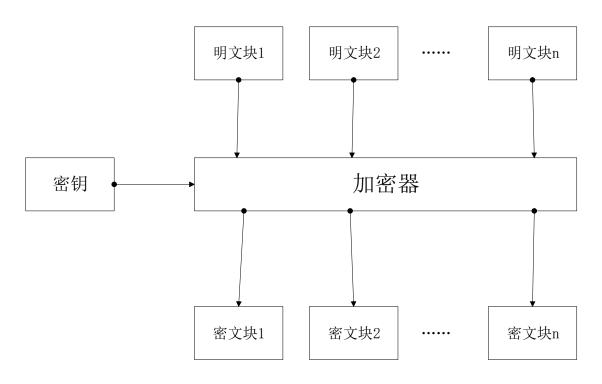


图 15 电码本模式示意图

(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。

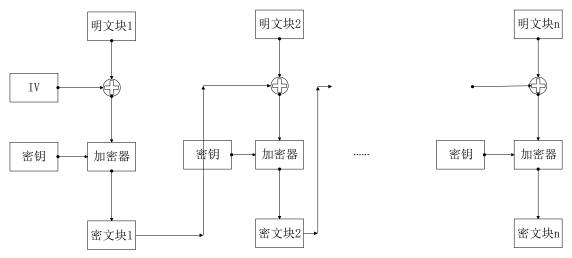


图 16 密码分组链接模式示意图

(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。

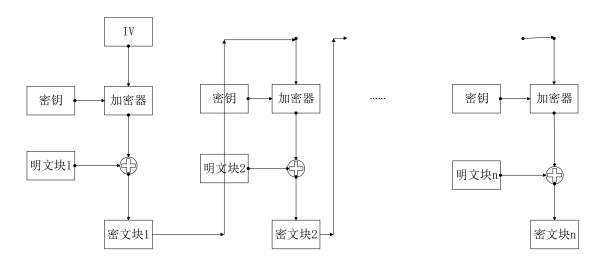


图 17 密文反馈模式示意图

(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。

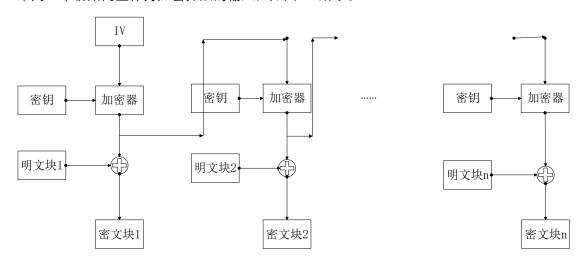


图 18 输出反馈模式示意图

(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。

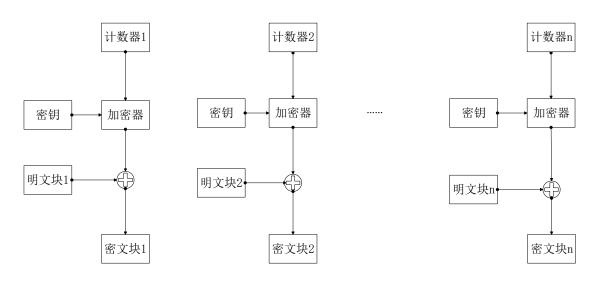


图 19 计数器模式示意图

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 0x00002;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 0x0000004,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)

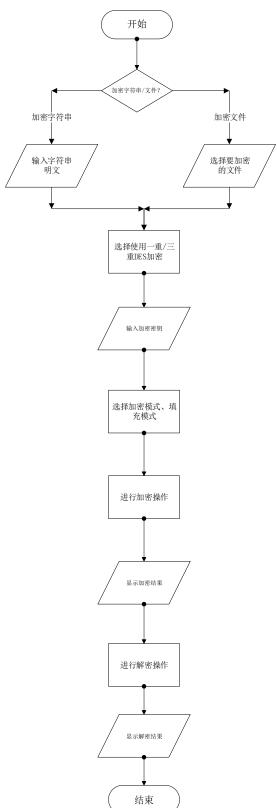


图 20 程序执行总流程图

(2) 加密模块流程图 (如图 21 所示)

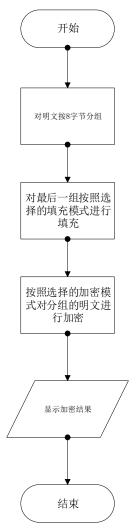


图 21 加密模块流程图

(3) 解密模块流程图 (如图 22 所示)

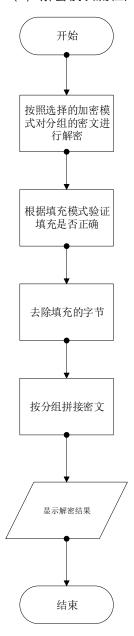


图 22 解密模块流程图

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
                                                       m_temp[i] = "";
//
                                                       c_temp[i] = "";
                                                       p_temp[i] = "";
#ifndef _CRT_SECURE_NO_WARNINGS
                                                       ctr_temp[i] = "";
#define _CRT_SECURE_NO_WARNINGS 1//
去除警告提示
                                                  }
#endif
                                                  //string 转换成 unsigned char*类型
#include "stdafx.h"
                                                  int n1 = k1.length();
#include "Encrypt_Decrypt_MFC.h"
                                                  int n2 = k2.length();
                                                  int n3 = k3.length();
#include "Encrypt Decrypt MFCDlg.h"
#include "afxdialogex.h"
                                                  unsigned char *key1 = new unsigned
#include "des.h"
                                              char[8];
#include <string>
                                                  unsigned char *key2 = new unsigned
#ifdef DEBUG
                                              char[8];
#define new DEBUG NEW
                                                  unsigned char *key3 = new unsigned
#endif
                                              char[8];
//加解密代码
                                                  strcpy((char*)key1, k1.c_str());
void
                                                  strcpy((char*)key2, k2.c_str());
CEncrypt_Decrypt_MFCDlg::_3DES_Startup(
                                                  strcpy((char*)key3, k3.c_str());
                                                  key1[7] = 0;
string k1, string k2, string k3)//3des 密钥初
始化
                                                  key2[7] = 0;
{
                                                  key3[7] = 0;
    srand((unsigned)time(NULL));
                                                  des_setup((unsigned char*)key1, 8, 0,
    for (int i = 0; i < 8; i++)
                                              &s k1);//初始化密钥 k1
         //IV[i] = (unsigned char)(rand() %
                                                  des_setup((unsigned char*)key2, 8, 0,
255 + 1);
                                              &s_k2);//初始化密钥 k2
         IV[i] = (unsigned
                                                  des_setup((unsigned char*)key3, 8, 0,
                                              &s k3);//初始化密钥 k3
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++)
    {
                                              string
```

```
CEncrypt_Decrypt_MFCDlg::_3DES_Encrypt(
string m)//3des 加密函数
                                                   unsigned char *pt = new unsigned
{
                                               char[c.length() + 1];
                                                   unsigned char *temp1 = new unsigned
    unsigned char *pt = new unsigned
                                               char[c.length() + 1];
char[m.length() + 1];
    unsigned char *temp1 = new unsigned
                                                   unsigned char *temp2 = new unsigned
char[m.length() + 1];
                                               char[c.length() + 1];
    unsigned char *temp2 = new unsigned
                                                   unsigned char *ct = new unsigned
char[m.length() + 1];
                                               char[c.length() + 1];
                                                   for (int i = 0; i < c.length() + 1; i++)
    unsigned char *ct = new unsigned
char[m.length() + 1];
                                                   {
    for (int i = 0; i < m.length() + 1; i++)
                                                        pt[i] = 0;
    {
                                                        temp1[i] = 0;
         pt[i] = 0;
                                                        temp2[i] = 0;
         temp1[i] = 0;
                                                        ct[i] = 0;
         temp2[i] = 0;
         ct[i] = 0;
                                                   //string 转换成 unsigned char *类型
    }
                                                   strcpy((char *)ct, c.c_str());
    //string 转换成 unsigned char*类型
                                                   ct[c.length()] = 0;
    strcpy((char *)pt, m.c_str());
                                                   des_ecb_decrypt(ct, temp2, &s_k3);//用
    pt[m.length()] = 0;
                                               k3 解密
    des_ecb_encrypt(pt, temp1, &s_k1);//
                                                   des_ecb_encrypt(temp2, temp1,
用 k1 加密
                                               &s k2);//用 k2 加密
                                                   des_ecb_decrypt(temp1, pt, &s_k1);//
    des_ecb_decrypt(temp1, temp2,
&s_k2);//用 k2 解密
                                               用 k1 解密
    des_ecb_encrypt(temp2, ct, &s_k3);//用
                                                   //unsigned char *转换成 string 类型
k3 加密
                                                   string m((char *)pt);
    string c((char *)ct);
                                                   return m;
    //unsigned char 转换成 string 类型
                                              }
    return c;
}
                                               void
                                               CEncrypt_Decrypt_MFCDlg::Inintialize_Encry
                                               pt_Decrypt(string s1, string s2, string
string
CEncrypt_Decrypt_MFCDlg::_3DES_Decrypt(
                                               s3)//3des 初始化
string c)//3des 解密函数
                                               {
```

```
_3DES_Startup(s1, s2, s3);//调用密钥初
始化函数
                                                   int k = 0;
}
                                                   string p = "";
                                                   do
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                        p_{temp[k]} =
a_ECB_ZERO(string m)//ECB 模式加密,零
                                               _3DES_Decrypt(c_temp[k]);
字节填充
                                                       //比特填充
                                                        for (int i = 0; i <
{
    string c = "";
                                               p_temp[k].length(); i++)
    int k = 0;
                                                        if (p_{temp}[k][i] < 0x08)
    do
                                                            p_{temp[k]} =
    {
                                               p_temp[k].substr(0, i);
         m_{temp[k]} = m.substr(8 * k, 8);
                                                        p = p + p_{temp[k]};
         //比特填充
                                                        k++;
                                                   } while (c_temp[k] != "\0");
         if (m_{temp[k].length()} < 8)
                                                   return p;
             int count_sub = 8 -
                                              }
m_temp[k].length();
                                               string
    m_temp[k].append(count_sub, 0x00);
                                               CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
         }
                                               a_ECB_PKCS7(string m)//ECB 模式加密,
                                               PKCS7 方式填充
         c_{temp[k]} =
_3DES_Encrypt(m_temp[k]);
                                              {
         c = c + c_{temp[k]};
                                                   string c = "";
         k++;
                                                   int k = 0;
    } while (8 * k < m.length());
                                                   do
    c_{temp[k]} = "\0";
                                                   {
    return c;
                                                        m_{temp}[k] = m.substr(8 * k, 8);
}
                                                        //比特填充
                                                        if (m_temp[k].length() < 8)
string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
                                                            int count_sub = 8 -
a_ECB_ZERO(string c)//ECB 模式解密,零
                                               m_temp[k].length();
字节填充
```

```
m_temp[k].append(count_sub,
(char)count_sub);
                                               string
         }
                                               CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
         c_temp[k] =
                                               a_ECB_ANSIX923(string m)//ECB 模式加
                                               密. ANSI X.923 方式填充
_3DES_Encrypt(m_temp[k]);
         c = c + c_{temp}[k];
                                                    string c = "";
    } while (8 * k < m.length());
                                                    int k = 0;
    c_{temp[k]} = "\0";
                                                    do
                                                    {
    return c;
}
                                                         m_{temp}[k] = m.substr(8 * k, 8);
                                                         //比特填充
                                                         if (m_temp[k].length()<8)
string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_ECB_PKCS7(string c)//ECB 模式解密,
                                                             int count_sub = 8 -
PKCS7 方式填充
                                               m_temp[k].length();
{
                                                             if (count_sub - 1>0)
    int k = 0;
    string p = "";
                                                    m_temp[k].append(count_sub - 1,
    do
                                               0x00);
    {
                                                             m_temp[k].append(1,
         p_{temp[k]} =
                                               (char)count_sub);
_3DES_Decrypt(c_temp[k]);
         //比特填充
                                                         c_{temp[k]} =
         for (int i = 0; i < 0
                                                _3DES_Encrypt(m_temp[k]);
p_temp[k].length(); i++)
                                                         c = c + c_{temp[k]};
         if (p_{temp[k][i]} < 0x08)
                                                         k++;
                                                    } while (8 * k < m.length());
             p_{temp[k]} =
p_temp[k].substr(0, i);
                                                    c_{temp[k]} = "\0";
         p = p + p_{temp[k]}
                                                    return c;
                                               }
         k++;
    } while (c_temp[k] != "\0");
    return p;
                                               string
}
                                               CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
                                               a_ECB_ANSIX923(string c)//ECB 模式解密,
```

```
ANSI X.923 方式填充
                                                              if (count_sub - 1>0)
{
                                                             {
    int k = 0;
    string p = "";
                                                    srand((unsigned)time(NULL));
    do
                                                                  unsigned int fill =
    {
                                                rand() % 256;
         p_{temp[k]} =
_3DES_Decrypt(c_temp[k]);
                                                    m_temp[k].append(count_sub - 1,
         //比特填充
                                                (char)fill);
         for (int i = 0; i <
p_temp[k].length(); i++)
                                                              m_temp[k].append(1,
         if (p_{temp[k][i]} < 0x08)
                                                (char)count_sub);
                                                         }
              p_{temp[k]} =
p_temp[k].substr(0, i);
                                                         c_{temp[k]} =
                                                _3DES_Encrypt(m_temp[k]);
         p = p + p_{temp[k]}
         k++;
                                                         c = c + c_{temp}[k];
    } while (c_temp[k] != "\0");
                                                         k++;
                                                    } while (8 * k < m.length());
    return p;
}
                                                    c_{temp[k]} = "\0";
                                                    return c;
string
                                               }
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_ECB_ISO10126(string m)//ECB 模式加
                                                string
密, ISO 10126 方式填充
                                                CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
{
                                                a_ECB_ISO10126(string c)//ECB 模式解密,
                                                ISO 10126 方式填充
    string c = "";
    int k = 0;
    do
                                                    int k = 0;
                                                    string p = "";
    {
         m_{temp}[k] = m.substr(8 * k, 8);
                                                    do
         //比特填充
         if (m_temp[k].length()<8)</pre>
                                                         p_{temp[k]} =
                                                _3DES_Decrypt(c_temp[k]);
             int count_sub = 8 -
                                                         //比特填充
                                                         for (int i = 0; i <
m_temp[k].length();
```

```
p_temp[k].length(); i++)
                                                           else
         if (p_{temp}[k][i] < 0x08)
                                                           for (int i = 0; i < 8; i++)
              p_{temp[k]} =
                                                                m_{temp[k][i]} = m_{temp[k][i]}
p_temp[k].substr(0, p_temp[k].length() -
                                                  ^ c_temp[k - 1][i];
                                                           c_temp[k] =
(int)p_temp[k][i]);
         p = p + p_{temp[k]}
                                                  _3DES_Encrypt(m_temp[k]);
                                                           c = c + c_{temp[k]};
    } while (c_temp[k] != "\0");
                                                           k++;
                                                      } while (8 * k < m.length());
    return p;
}
                                                      c_{temp[k]} = "\0";
                                                      return c;
                                                 }
string
CEncrypt_Decrypt_MFCDIg::Encrypt_All_Dat
a_CBC_ZERO(string m)//CBC 模式加密,零
                                                 string
字节填充
                                                 {\sf CEncrypt\_Decrypt\_MFCDlg::Decrypt\_All\_Dat}
{
                                                 a_CBC_ZERO(string c)//CBC 模式解密,零
    string c = "";
                                                 字节填充
                                                 {
    int k = 0;
    do
                                                      int k = 0;
    {
                                                      string p = "";
         m_{temp}[k] = m.substr(8 * k, 8);
                                                      do
                                                      {
         //比特填充
                                                           p_{temp[k]} =
         if (m_temp[k].length() < 8)
                                                 _3DES_Decrypt(c_temp[k]);
                                                           //向量异或
         {
              int count_sub = 8 -
                                                           if (k == 0)
m_temp[k].length();
                                                           for (int i = 0; i < 8; i++)
                                                                p_{temp[k][i]} = p_{temp[k][i]} ^
    m_temp[k].append(count_sub, 0x00);
                                                 IV[i];
         }
                                                           else
         //向量异或
                                                           for (int i = 0; i < 8; i++)
         if (k == 0)
                                                                p_temp[k][i] = p_temp[k][i] ^
         for (int i = 0; i < 8; i++)
                                                 c_temp[k - 1][i];
                                                           //比特填充
              m_{temp[k][i]} = m_{temp[k][i]}
^ IV[i];
                                                           for (int i = 0; i <
```

```
p_temp[k].length(); i++)
                                                           else
         if (p_{temp}[k][i] < 0x08)
                                                           for (int i = 0; i < 8; i++)
              p_{temp[k]} =
                                                                m_{temp[k][i]} = m_{temp[k][i]}
                                                  ^ c_temp[k - 1][i];
p_temp[k].substr(0, i);
                                                           c_temp[k] =
         p = p + p_{temp[k]};
                                                  _3DES_Encrypt(m_temp[k]);
    } while (c_temp[k] != "\0");
                                                           c = c + c_{temp[k]};
    return p;
                                                           k++;
}
                                                      } while (8 * k < m.length());
                                                      c_{temp[k]} = "\0";
                                                      return c;
                                                 }
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CBC_PKCS7(string m)//CBC 模式加密,
                                                  string
PKCS7 方式填充
                                                  {\sf CEncrypt\_Decrypt\_MFCDlg::Decrypt\_All\_Dat}
{
                                                  a_CBC_PKCS7(string c)//CBC 模式解密,
                                                  PKCS7 方式填充
    string c = "";
    int k = 0;
                                                 {
    do
                                                      int k = 0;
    {
                                                      string p = "";
         m_{temp}[k] = m.substr(8 * k, 8);
                                                      do
         //比特填充
         if (m_{temp[k].length()} < 8)
                                                           p_{temp[k]} =
                                                  _3DES_Decrypt(c_temp[k]);
                                                           //向量异或
              int count_sub = 8 -
m_temp[k].length();
                                                           if (k == 0)
                                                           for (int i = 0; i < 8; i++)
                                                                p_{temp[k][i]} = p_{temp[k][i]} ^
    m_temp[k].append(count_sub,
(char)count_sub);
                                                  IV[i];
         }
                                                           else
         //向量异或
                                                           for (int i = 0; i < 8; i++)
         if (k == 0)
                                                                p_temp[k][i] = p_temp[k][i] ^
         for (int i = 0; i < 8; i++)
                                                  c_temp[k - 1][i];
                                                           //比特填充
              m_{temp[k][i]} = m_{temp[k][i]}
^ IV[i];
                                                           for (int i = 0; i <
```

```
p_temp[k].length(); i++)
                                                               m_{temp[k][i]} = m_{temp[k][i]}
                                                 ^ IV[i];
         if (p_{temp}[k][i] < 0x08)
              p_{temp[k]} =
                                                           else
                                                           for (int i = 0; i < 8; i++)
p_temp[k].substr(0, i);
                                                               m_{temp[k][i]} = m_{temp[k][i]}
         p = p + p_{temp[k]};
                                                 ^ c_temp[k - 1][i];
    } while (c_temp[k] != "\0");
                                                           c_{temp[k]} =
    return p;
                                                 _3DES_Encrypt(m_temp[k]);
}
                                                           c = c + c_{temp[k]};
                                                           k++;
string
                                                      } while (8 * k < m.length());
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                      c_{temp[k]} = "\0";
a_CBC_ANSIX923(string m)//CBC 模式加
                                                      return c;
密, ANSI X.923 方式填充
                                                 }
{
    string c = "";
                                                 string
    int k = 0;
                                                 CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
    do
                                                 a_CBC_ANSIX923(string c)//CBC 模式解
                                                 密, ANSI X.923 方式填充
    {
         m_{temp}[k] = m.substr(8 * k, 8);
                                                 {
         //比特填充
                                                      int k = 0;
         if (m_temp[k].length()<8)
                                                      string p = "";
         {
                                                      do
              int count_sub = 8 -
                                                      {
m_temp[k].length();
                                                           p_{temp[k]} =
                                                 _3DES_Decrypt(c_temp[k]);
              if (count_sub - 1>0)
                                                           //向量异或
    m_temp[k].append(count_sub - 1,
0x00);
                                                           if (k == 0)
                                                           for (int i = 0; i < 8; i++)
              m_temp[k].append(1,
(char)count_sub);
                                                               p_temp[k][i] = p_temp[k][i] ^
                                                 IV[i];
         //向量异或
                                                           else
         if (k == 0)
                                                           for (int i = 0; i < 8; i++)
         for (int i = 0; i < 8; i++)
                                                               p_{temp[k][i]} = p_{temp[k][i]} ^{i}
```

```
c_temp[k - 1][i];
         //比特填充
                                                     m_temp[k].append(count_sub - 1,
         for (int i = 0; i < 0
                                                 (char)fill);
p_temp[k].length(); i++)
         if (p_{temp}[k][i] < 0x08)
                                                               m_temp[k].append(1,
              p_{temp[k]} =
                                                 (char)count_sub);
p_temp[k].substr(0, i);
                                                          }
                                                          //向量异或
                                                          if (k == 0)
         p = p + p_{temp[k]};
                                                          for (int i = 0; i < 8; i++)
         k++;
    } while (c_temp[k] != "\0");
                                                               m_{temp[k][i]} = m_{temp[k][i]}
                                                 ^ IV[i];
    return p;
}
                                                          else
                                                          for (int i = 0; i < 8; i++)
string
                                                               m_temp[k][i] = m_temp[k][i]
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                 ^ c_temp[k - 1][i];
a_CBC_ISO10126(string m)//CBC 模式加
                                                          c_{temp[k]} =
密, ISO 10126 方式填充
                                                 _3DES_Encrypt(m_temp[k]);
{
                                                          c = c + c_{temp[k]};
    string c = "";
                                                          k++;
    int k = 0;
                                                     } while (8 * k < m.length());
    do
                                                     c_{temp[k]} = "\0";
    {
                                                     return c;
         m_{temp}[k] = m.substr(8 * k, 8);
                                                }
         //比特填充
         if (m_temp[k].length()<8)
                                                 string
                                                 CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
         {
                                                 a_CBC_ISO10126(string c)//CBC 模式解
              int count_sub = 8 -
                                                 密, ISO 10126 方式填充
m_temp[k].length();
              if (count_sub - 1>0)
              {
                                                     int k = 0;
                                                     string p = "";
    srand((unsigned)time(NULL));
                                                     do
                  unsigned int fill =
                                                     {
rand() % 256;
                                                          p_{temp[k]} =
```

```
_3DES_Decrypt(c_temp[k]);
                                                           m_{temp}[k] = m.substr(8 * k, 8);
                                                           //比特填充
         //向量异或
                                                           if (m_{temp}[k].length() < 8)
         if (k == 0)
         for (int i = 0; i < 8; i++)
                                                               int count_sub = 8 -
              p_{temp[k][i]} = p_{temp[k][i]} ^{i}
                                                 m_temp[k].length();
IV[i];
         else
                                                      m_temp[k].append(count_sub, 0x00);
         for (int i = 0; i < 8; i++)
                                                          }
                                                          //向量异或
              p_temp[k][i] = p_temp[k][i] ^
c_temp[k - 1][i];
                                                           string iv((char *)IV);
         //比特填充
                                                           if (k == 0)
         for (int i = 0; i <
                                                               c_{temp[k]} =
p_temp[k].length(); i++)
                                                 _3DES_Encrypt(iv);
         if (p_{temp}[k][i] < 0x08)
                                                           else
              p_{temp[k]} =
                                                               c_{temp[k]} =
p_temp[k].substr(0, p_temp[k].length() -
                                                 _3DES_Encrypt(c_temp[k - 1]);
(int)p_temp[k][i]);
                                                           for (int i = 0; i < 8; i++)
                                                               c_{temp[k][i]} = c_{temp[k][i]} ^{i}
         p = p + p_temp[k];
                                                 m_temp[k][i];
         k++;
                                                           c = c + c_{temp[k]};
    } while (c_temp[k] != "\0");
                                                           k++;
                                                      } while (8 * k < m.length());
    return p;
}
                                                      c_{temp[k]} = "\0";
                                                      return c;
                                                }
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                 string
a_CFB_ZERO(string m)//CFB 模式加密,零
                                                 CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
字节填充
                                                 a_CFB_ZERO(string c)//CFB 模式解密,零
                                                 字节填充
    string c = "";
    int k = 0;
                                                      int k = 0;
    do
                                                      string p = "";
    {
                                                      do
```

```
{
                                                      {
         string iv((char *)IV);
                                                           m_{temp}[k] = m.substr(8 * k, 8);
                                                           //比特填充
         if (k == 0)
                                                           if (m_temp[k].length() < 8)
              p_{temp[k]} =
_3DES_Encrypt(iv);
         else
                                                                int count_sub = 8 -
              p_{temp[k]} =
                                                 m_temp[k].length();
_3DES_Encrypt(c_temp[k - 1]);
         p_temp[k] = p_temp[k].substr(0,
                                                      m_temp[k].append(count_sub,
8);
                                                 (char)count_sub);
         //向量异或
                                                           }
         for (int i = 0; i < 8; i++)
                                                           //向量异或
              p_{temp[k][i]} = p_{temp[k][i]} 
                                                           string iv((char *)IV);
c_temp[k][i];
         //比特填充
                                                           if (k == 0)
         for (int i = 0; i <
                                                               c_{temp[k]} =
p_temp[k].length(); i++)
                                                 _3DES_Encrypt(iv);
         if (p_{temp}[k][i] < 0x08)
                                                           else
              p_{temp[k]} =
                                                                c_{temp[k]} =
p_temp[k].substr(0, i);
                                                 _3DES_Encrypt(c_temp[k - 1]);
         p = p + p_{temp[k]};
                                                           for (int i = 0; i < 8; i++)
                                                                c_{temp[k][i]} = c_{temp[k][i]} ^{i}
         k++;
    } while (c_temp[k] != "\0");
                                                 m_temp[k][i];
                                                           c = c + c_{temp}[k];
    return p;
}
                                                           k++;
                                                      } while (8 * k < m.length());
                                                      c_{temp[k]} = "\0";
string
                                                      return c;
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                 }
a_CFB_PKCS7(string m)//CFB 模式加密,
PKCS7 方式填充
                                                 {\tt CEncrypt\_Decrypt\_MFCDlg::Decrypt\_All\_Dat}
                                                 a_CFB_PKCS7(string c)//CFB 模式解密,
    string c = "";
                                                 PKCS7 方式填充
    int k = 0;
    do
```

```
int k = 0;
                                                  {
    string p = "";
                                                      string c = "";
    do
                                                      int k = 0;
    {
                                                       do
         string iv((char *)IV);
         if (k == 0)
                                                           m_{temp}[k] = m.substr(8 * k, 8);
              p_{temp[k]} =
                                                           //比特填充
_3DES_Encrypt(iv);
                                                           if (m_temp[k].length()<8)
         else
              p_{temp[k]} =
_3DES_Encrypt(c_temp[k - 1]);
                                                                int count_sub = 8 -
         p_{temp[k]} = p_{temp[k].substr(0,
                                                  m_temp[k].length();
8);
                                                                if (count_sub - 1>0)
         //向量异或
         for (int i = 0; i < 8; i++)
                                                       m_temp[k].append(count_sub - 1,
              p_{temp[k][i]} = p_{temp[k][i]} ^{i}
                                                  0x00);
c_temp[k][i];
                                                                m_temp[k].append(1,
         //比特填充
                                                  (char)count_sub);
         for (int i = 0; i <
                                                           }
p_temp[k].length(); i++)
                                                           //向量异或
         if (p_temp[k][i] < 0x08)
                                                           string iv((char *)IV);
                                                           if (k == 0)
              p_{temp[k]} =
p_temp[k].substr(0, i);
                                                                c_{temp[k]} =
                                                  _3DES_Encrypt(iv);
         p = p + p_{temp[k]}
                                                           else
                                                                c_{temp[k]} =
         k++;
    } while (c_temp[k] != "\0");
                                                  _3DES_Encrypt(c_temp[k - 1]);
                                                           for (int i = 0; i < 8; i++)
    return p;
}
                                                                c_{temp[k][i]} = c_{temp[k][i]} ^{i}
                                                  m_temp[k][i];
                                                           c = c + c_{temp[k]};
                                                           k++;
string
CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                      } while (8 * k < m.length());
a_CFB_ANSIX923(string m)//CFB 模式加
                                                       c_{temp[k]} = "\0";
密, ANSI X.923 方式填充
                                                       return c;
```

```
}
string
                                                 string
{\tt CEncrypt\_Decrypt\_MFCDlg::Decrypt\_All\_Dat}
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_ANSIX923(string c)//CFB 模式解密,
                                                 a_CFB_ISO10126(string m)//CFB 模式加
                                                 密, ISO 10126 方式填充
ANSI X.923 方式填充
{
                                                 {
    int k = 0;
                                                     string c = "";
    string p = "";
                                                     int k = 0;
    do
                                                      do
    {
                                                     {
         string iv((char *)IV);
                                                          m_{temp}[k] = m.substr(8 * k, 8);
         if (k == 0)
                                                          //比特填充
              p_{temp[k]} =
                                                          if (m_temp[k].length()<8)
_3DES_Encrypt(iv);
         else
                                                               int count_sub = 8 -
              p_{temp[k]} =
_3DES_Encrypt(c_temp[k - 1]);
                                                 m_temp[k].length();
         p_temp[k] = p_temp[k].substr(0,
                                                               if (count_sub - 1>0)
8);
                                                               {
         //向量异或
         for (int i = 0; i < 8; i++)
                                                     srand((unsigned)time(NULL));
              p_{temp[k][i]} = p_{temp[k][i]} 
                                                                    unsigned int fill =
c_temp[k][i];
                                                 rand() % 256;
         //比特填充
         for (int i = 0; i <
                                                     m_temp[k].append(count_sub - 1,
p_temp[k].length(); i++)
                                                 (char)fill);
         if (p_{temp}[k][i] < 0x08)
              p_{temp[k]} =
                                                               m_temp[k].append(1,
p_temp[k].substr(0, i);
                                                 (char)count_sub);
         p = p + p_{temp[k]};
                                                          //向量异或
         k++;
    } while (c_temp[k] != "\0");
                                                          string iv((char *)IV);
                                                          if (k == 0)
    return p;
}
                                                               c_{temp[k]} =
```

```
_3DES_Encrypt(iv);
                                                                p_{temp[k][i]} = p_{temp[k][i]} ^{i}
                                                 c_temp[k][i];
         else
                                                           //比特填充
              c_{temp[k]} =
_3DES_Encrypt(c_temp[k - 1]);
                                                           for (int i = 0; i <
                                                 p_temp[k].length(); i++)
         for (int i = 0; i < 8; i++)
                                                           if (p_{temp}[k][i] < 0x08)
              c_{temp[k][i]} = c_{temp[k][i]} ^
                                                                p_{temp[k]} =
                                                 p_temp[k].substr(0, p_temp[k].length() -
m_temp[k][i];
         c = c + c_{temp[k]};
                                                 (int)p_temp[k][i]);
         k++;
                                                           p = p + p_{temp[k]};
    } while (8 * k < m.length());
                                                           k++;
    c_{temp[k]} = "\0";
                                                      } while (c_temp[k] != "\0");
    return c;
                                                      return p;
}
                                                 }
string
                                                 string
CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
a_CFB_ISO10126(string c)//CFB 模式解密,
                                                 a_OFB_ZERO(string m)//OFB 模式加密,零
ISO 10126 方式填充
                                                 字节填充
{
                                                 {
    int k = 0;
                                                      string c = "";
    string p = "";
                                                      int k = 0;
                                                      for (int i = 0; i < 100; i++)
    do
                                                           c_temp[i] = "";
    {
         string iv((char *)IV);
                                                      do
         if (k == 0)
                                                      {
                                                           m_{temp[k]} = m.substr(8 * k, 8);
              p_{temp[k]} =
_3DES_Encrypt(iv);
         else
                                                           //比特填充
                                                           if (m_{temp[k].length()} < 8)
              p_{temp[k]} =
_3DES_Encrypt(c_temp[k - 1]);
         p_{temp[k]} = p_{temp[k].substr(0,
                                                                int count_sub = 8 -
8);
                                                 m_temp[k].length();
         //向量异或
         for (int i = 0; i < 8; i++)
                                                      m_temp[k].append(count_sub, 0x00);
```

```
}
                                                               p_temp[k].append(1, key[i] ^
         //向量异或
                                                 c_temp[k][i]);
                                                          //比特填充
         string iv((char *)IV);
                                                          for (int i = 0; i <
         string key = iv;
         for (int i = 0; i < k + 1; i++)
                                                 p_temp[k].length(); i++)
              key = _3DES_Encrypt(key);
                                                          if (p_{temp}[k][i] < 0x08)
                                                               p_{temp[k]} =
         for (int i = 0; i < 8; i++)
                                                 p_temp[k].substr(0, i);
              c_temp[k].append(1,
                                                          p = p + p_{temp[k]}
(char)key[i] ^ m_temp[k][i]);
                                                          k++;
         c = c + c_{temp}[k];
                                                     } while (c_temp[k] != "\0");
         k++;
                                                      return p;
    } while (8 * k < m.length());
                                                 }
    c_{temp[k]} = "\0";
    return c;
                                                 string
}
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                 a_OFB_PKCS7(string m)//OFB 模式加密,
                                                 PKCS7 方式填充
string
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
                                                 {
                                                      string c = "";
a_OFB_ZERO(string c)//OFB 模式解密,零
字节填充
                                                     int k = 0;
                                                      for (int i = 0; i < 100; i++)
                                                          c_temp[i] = "";
    int k = 0;
    string p = "";
                                                      do
    for (int i = 0; i < 100; i++)
                                                      {
         p_temp[i] = "";
                                                          m_{temp}[k] = m.substr(8 * k, 8);
    do
    {
                                                          //比特填充
         string iv((char *)IV);
                                                          if (m_temp[k].length() < 8)
         string key = iv;
         for (int i = 0; i < k + 1; i++)
                                                               int count_sub = 8 -
              key = _3DES_Encrypt(key);
                                                 m_temp[k].length();
         key = key.substr(0, 8);
         //向量异或
                                                      m_temp[k].append(count_sub,
         for (int i = 0; i < 8; i++)
                                                 (char)count_sub);
```

```
}
                                                 c_temp[k][i]);
         //向量异或
                                                           //比特填充
         string iv((char *)IV);
                                                           for (int i = 0; i < 0
                                                 p_temp[k].length(); i++)
         string key = iv;
         for (int i = 0; i < k + 1; i++)
                                                           if (p_{temp}[k][i] < 0x08)
              key = _3DES_Encrypt(key);
                                                               p_{temp[k]} =
         for (int i = 0; i < 8; i++)
                                                 p_temp[k].substr(0, i);
              c_temp[k].append(1,
                                                           p = p + p_{temp[k]};
(char)key[i] ^ m_temp[k][i]);
         c = c + c_{temp[k]};
                                                      } while (c_temp[k] != "\0");
         k++;
                                                      return p;
    } while (8 * k < m.length());
                                                 }
    c_{temp[k]} = "\0";
    return c;
                                                 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
CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
                                                 {
a_OFB_PKCS7(string c)//OFB 模式解密,
                                                      string c = "";
PKCS7 方式填充
                                                      int k = 0;
{
                                                      for (int i = 0; i < 100; i++)
    int k = 0;
                                                           c_temp[i] = "";
    string p = "";
                                                      do
    for (int i = 0; i < 100; i++)
                                                      {
         p_temp[i] = "";
                                                           m_{temp}[k] = m.substr(8 * k, 8);
    do
                                                           //比特填充
    {
                                                           if (m_temp[k].length()<8)
         string iv((char *)IV);
                                                           {
         string key = iv;
                                                               int count_sub = 8 -
         for (int i = 0; i < k + 1; i++)
                                                 m_temp[k].length();
              key = _3DES_Encrypt(key);
                                                               if (count_sub - 1>0)
         key = key.substr(0, 8);
         //向量异或
                                                      m_temp[k].append(count_sub - 1,
         for (int i = 0; i < 8; i++)
                                                 0x00);
              p_temp[k].append(1, key[i] ^
                                                               m_temp[k].append(1,
```

```
(char)count_sub);
                                                          for (int i = 0; i < 8; i++)
         }
                                                               p_temp[k].append(1, key[i] ^
                                                 c_temp[k][i]);
                                                          //比特填充
         //向量异或
                                                          for (int i = 0; i <
         string iv((char *)IV);
         string key = iv;
                                                 p_temp[k].length(); i++)
         for (int i = 0; i < k + 1; i++)
                                                          if (p_{temp}[k][i] < 0x08)
              key = _3DES_Encrypt(key);
                                                               p_{temp[k]} =
         for (int i = 0; i < 8; i++)
                                                 p_temp[k].substr(0, i);
              c_temp[k].append(1,
                                                          p = p + p_{temp[k]}
(char)key[i] ^ m_temp[k][i]);
                                                          k++;
                                                      } while (c_temp[k] != "\0");
         c = c + c_{temp[k]};
         k++;
                                                      return p;
    } while (8 * k < m.length());
                                                 }
    c_{temp[k]} = "\0";
    return c;
                                                 string
}
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
                                                 a_OFB_ISO10126(string m)//OFB 模式加
                                                 密, ISO 10126 方式填充
string
CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
                                                 {
a_OFB_ANSIX923(string c)//OFB 模式解
                                                      string c = "";
密, ANSI X.923 方式填充
                                                      int k = 0;
                                                      for (int i = 0; i < 100; i++)
{
                                                          c_temp[i] = "";
    int k = 0;
    string p = "";
                                                      do
    for (int i = 0; i < 100; i++)
                                                      {
         p_temp[i] = "";
                                                          m_{temp[k]} = m.substr(8 * k, 8);
                                                          //比特填充
    do
    {
                                                          if (m_temp[k].length()<8)
         string iv((char *)IV);
         string key = iv;
                                                               int count_sub = 8 -
         for (int i = 0; i < k + 1; i++)
                                                 m_temp[k].length();
              key = _3DES_Encrypt(key);
                                                               if (count_sub - 1>0)
         key = key.substr(0, 8);
         //向量异或
```

```
srand((unsigned)time(NULL));
                                                      {
                   unsigned int fill =
                                                           string iv((char *)IV);
rand() % 256;
                                                           string key = iv;
                                                           for (int i = 0; i < k + 1; i++)
    m_temp[k].append(count_sub - 1,
                                                               key = _3DES_Encrypt(key);
(char)fill);
                                                           key = key.substr(0, 8);
                                                           //向量异或
                                                           for (int i = 0; i < 8; i++)
              m_temp[k].append(1,
                                                               p_temp[k].append(1, key[i] ^
(char)count_sub);
         }
                                                 c_temp[k][i]);
         //向量异或
                                                           //比特填充
         string iv((char *)IV);
                                                           for (int i = 0; i <
                                                 p_temp[k].length(); i++)
         string key = iv;
         for (int i = 0; i < k + 1; i++)
                                                           if (p_{temp[k][i]} < 0x08)
              key = _3DES_Encrypt(key);
                                                               p_{temp[k]} =
         for (int i = 0; i < 8; i++)
                                                 p_temp[k].substr(0, p_temp[k].length() -
              c_temp[k].append(1,
                                                 (int)p_temp[k][i]);
(char)key[i] ^ m_temp[k][i]);
                                                           p = p + p_{temp[k]}
                                                           k++;
         c = c + c_{temp}[k];
         k++;
                                                      } while (c_temp[k] != "\0");
    } while (8 * k < m.length());
                                                      return p;
    c_{temp[k]} = "\0";
                                                 }
    return c;
}
                                                 string
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
string
                                                 a_CTR_ZERO(string m)//CTR 模式加密,零
                                                 字节填充
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
a_OFB_ISO10126(string c)//OFB 模式解
                                                 {
密, ISO 10126 方式填充
                                                      string c = "";
{
                                                      int k = 0;
                                                      for (int i = 0; i < 100; i++)
    int k = 0;
    string p = "";
                                                           ctr_temp[i] = "";
    for (int i = 0; i < 100; i++)
                                                      do
         p_temp[i] = "";
                                                      {
    do
                                                           m_{temp}[k] = m.substr(8 * k, 8);
```

```
//比特填充
                                                      {
         if (m_{temp}[k].length() < 8)
                                                          m_{temp[k]} =
         {
                                                 _3DES_Encrypt(ctr_temp[k]);
                                                          for (int i = 0; i < 8; i++)
              int count_sub = 8 -
m_temp[k].length();
                                                               p_temp[k].append(1,
                                                 (char)(m_temp[k][i] ^ c_temp[k][i]));
                                                          //比特填充
    m_temp[k].append(count_sub, 0x00);
                                                          for (int i = 0; i <
         //向量异或
                                                 p_temp[k].length(); i++)
         ctr_temp[k].append(1, (char)k);
                                                          if (p_{temp}[k][i] < 0x08)
         while (ctr_temp[k].length() < 8)
                                                               p_{temp[k]} =
              ctr_temp[k].append(1, '0');
                                                 p_temp[k].substr(0, i);
         c_{temp[k]} =
                                                          p = p + p_{temp[k]}
_3DES_Encrypt(ctr_temp[k]);
                                                          k++;
         for (int i = 0; i < 8; i++)
                                                      } while (c_temp[k] != "\0");
              c_{temp[k][i]} = c_{temp[k][i]} ^
                                                      return p;
                                                }
m_temp[k][i];
         c_{temp[k]} = c_{temp[k].substr(0, 8);}
         c = c + c_{temp}[k];
                                                 string
         k++;
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
    } while (8 * k < m.length());
                                                 a_CTR_PKCS7(string m)//CTR 模式加密,
    c_{temp[k]} = "\0";
                                                 PKCS7 方式填充
    return c;
}
                                                      string c = "";
                                                      int k = 0;
string
                                                      for (int i = 0; i < 100; i++)
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
                                                          ctr_temp[i] = "";
a_CTR_ZERO(string c)//CTR 模式解密,零
                                                      do
字节填充
                                                      {
                                                          m_{temp}[k] = m.substr(8 * k, 8);
                                                          //比特填充
    int k = 0;
    string p = "";
                                                          if (m_temp[k].length() < 8)
    for (int i = 0; i < 100; i++)
         p_temp[i] = "";
                                                               int count_sub = 8 -
                                                 m_temp[k].length();
    do
```

```
p_temp[k].append(1,
    m_temp[k].append(count_sub,
                                                 (char)(m_temp[k][i] ^ c_temp[k][i]));
                                                          //比特填充
(char)count_sub);
                                                          for (int i = 0; i <
         }
         //向量异或
                                                 p_temp[k].length(); i++)
         ctr_temp[k].append(1, (char)k);
                                                          if (p_{temp}[k][i] < 0x08)
         while (ctr_temp[k].length() < 8)
                                                               p_{temp[k]} =
              ctr_temp[k].append(1, '0');
                                                 p_temp[k].substr(0, i);
         c_{temp[k]} =
                                                          p = p + p_{temp[k]}
_3DES_Encrypt(ctr_temp[k]);
                                                          k++;
         for (int i = 0; i < 8; i++)
                                                     } while (c_temp[k] != "\0");
              c_{temp[k][i]} = c_{temp[k][i]} ^
                                                      return p;
                                                }
m_temp[k][i];
         c_{temp[k]} = c_{temp[k].substr(0, 8)};
         c = c + c_{temp}[k];
         k++;
                                                 string
    } while (8 * k < m.length());
                                                 CEncrypt_Decrypt_MFCDlg::Encrypt_All_Dat
    c_{temp[k]} = "\0";
                                                 a_CTR_ANSIX923(string m)//CTR 模式加
                                                 密, ANSI X.923 方式填充
    return c;
}
                                                 {
                                                      string c = "";
                                                      int k = 0;
string
                                                      for (int i = 0; i < 100; i++)
CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
                                                          ctr_temp[i] = "";
a_CTR_PKCS7(string c)//CTR 模式解密,
PKCS7 方式填充
                                                      do
{
                                                      {
    int k = 0;
                                                          m_{temp}[k] = m.substr(8 * k, 8);
                                                          //比特填充
    string p = "";
    for (int i = 0; i < 100; i++)
                                                          if (m_temp[k].length()<8)
         p_temp[i] = "";
                                                               int count_sub = 8 -
    do
    {
                                                 m_temp[k].length();
         m_{temp[k]} =
                                                               if (count_sub - 1>0)
_3DES_Encrypt(ctr_temp[k]);
         for (int i = 0; i < 8; i++)
                                                      m_temp[k].append(count_sub - 1,
```

```
0x00);
                                                               p_temp[k].append(1,
              m_temp[k].append(1,
                                                 (char)(m_temp[k][i] ^ c_temp[k][i]));
                                                          //比特填充
(char)count_sub);
                                                          for (int i = 0; i <
         }
         //向量异或
                                                 p_temp[k].length(); i++)
         ctr_temp[k].append(1, (char)k);
                                                          if (p_{temp}[k][i] < 0x08)
         while (ctr_temp[k].length() < 8)
                                                               p_{temp[k]} =
              ctr_temp[k].append(1, '0');
                                                 p_temp[k].substr(0, i);
         c_{temp[k]} =
                                                          p = p + p_{temp[k]};
_3DES_Encrypt(ctr_temp[k]);
                                                          k++;
         for (int i = 0; i < 8; i++)
                                                      } while (c_temp[k] != "\0");
              c_{temp[k][i]} = c_{temp[k][i]} ^
                                                      return p;
                                                 }
m_temp[k][i];
         c_{temp[k]} = c_{temp[k].substr(0, 8)};
         c = c + c_{temp}[k];
                                                 string
         k++;
                                                 CEncrypt_Decrypt_MFCDIg::Encrypt_All_Dat
    } while (8 * k < m.length());
                                                 a_CTR_ISO10126(string m)//CTR 模式加
    c_{temp[k]} = "\0";
                                                 密, ISO 10126 方式填充
                                                 {
    return c;
}
                                                      string c = "";
                                                      int k = 0;
                                                      for (int i = 0; i < 100; i++)
string
                                                          ctr_temp[i] = "";
CEncrypt_Decrypt_MFCDIg::Decrypt_All_Dat
a_CTR_ANSIX923(string c)//CTR 模式解
                                                      do
密, ANSI X.923 方式填充
                                                      {
{
                                                          m_{temp}[k] = m.substr(8 * k, 8);
    int k = 0;
                                                          //比特填充
                                                          if (m_temp[k].length()<8)
    string p = "";
    for (int i = 0; i < 100; i++)
                                                          {
         p_temp[i] = "";
                                                               int count_sub = 8 -
    do
                                                 m_temp[k].length();
    {
                                                               if (count_sub - 1>0)
         m_{temp[k]} =
_3DES_Encrypt(ctr_temp[k]);
         for (int i = 0; i < 8; i++)
                                                      srand((unsigned)time(NULL));
```

```
unsigned int fill =
                                                     do
rand() % 256;
                                                     {
                                                         m_{temp[k]} =
    m_temp[k].append(count_sub - 1,
                                                _3DES_Encrypt(ctr_temp[k]);
                                                         for (int i = 0; i < 8; i++)
(char)fill);
                                                              p_temp[k].append(1,
                                                (char)(m_temp[k][i] ^ c_temp[k][i]));
              m_temp[k].append(1,
                                                         //比特填充
(char)count_sub);
         }
                                                         for (int i = 0; i <
         //向量异或
                                                p_temp[k].length(); i++)
         ctr_temp[k].append(1, (char)k);
                                                         if (p_{temp}[k][i] < 0x08)
         while (ctr_temp[k].length() < 8)
                                                              p_{temp[k]} =
                                                p_temp[k].substr(0, p_temp[k].length() -
              ctr_temp[k].append(1, '0');
         c_{temp[k]} =
                                                (int)p_temp[k][i]);
_3DES_Encrypt(ctr_temp[k]);
                                                         p = p + p_{temp[k]};
         for (int i = 0; i < 8; i++)
                                                         k++;
              c_{temp[k][i]} = c_{temp[k][i]} ^
                                                     } while (c_temp[k] != "\0");
m_temp[k][i];
                                                     return p;
         c_{temp[k]} = c_{temp[k].substr(0, 8)};
                                                }
         c = c + c_{temp}[k];
                                                // CEncrypt_Decrypt_MFCDIg 对话框
         k++;
                                                CEncrypt_Decrypt_MFCDlg::CEncrypt_Decry
    } while (8 * k < m.length());
                                                pt_MFCDlg(CWnd* pParent /*=NULL*/):
    c_{temp[k]} = "\0";
                                                CDialogEx(CEncrypt_Decrypt_MFCDlg::IDD,
    return c;
                                                pParent), m_encryptway(0), m_fillway(0),
}
                                                m_enchoose(0)
                                                {
string
                                                     m_hlcon =
CEncrypt_Decrypt_MFCDlg::Decrypt_All_Dat
                                                AfxGetApp()->LoadIcon(IDR_MAINFRAME);
a_CTR_ISO10126(string c)//CTR 模式解密,
                                                }
ISO 10126 方式填充
                                                void
    int k = 0;
                                                CEncrypt_Decrypt_MFCDlg::DoDataExchang
    string p = "";
                                                e(CDataExchange* pDX)
    for (int i = 0; i < 100; i++)
                                                {
         p_temp[i] = "";
                                                     CDialogEx::DoDataExchange(pDX);
```

DDX_Control(pDX, ON_BN_CLICKED(IDC_RADIO15, IDC_COMBO_KEYBYTES, m_cb_keybytes); &CEncrypt_Decrypt_MFCDlg::OnBnClickedR } adio15) ON_BN_CLICKED(IDC_RADIO21, BEGIN_MESSAGE_MAP(CEncrypt_Decrypt_ &CEncrypt_Decrypt_MFCDlg::OnBnClickedR MFCDIg, CDialogEx) adio21) ON_WM_PAINT() ON_BN_CLICKED(IDC_RADIO22, ON_WM_QUERYDRAGICON() &CEncrypt_Decrypt_MFCDlg::OnBnClickedR ON BN CLICKED(IDOK, adio22) &CEncrypt_Decrypt_MFCDlg::OnBnClicked ON_BN_CLICKED(IDC_RADIO23, Ok) &CEncrypt_Decrypt_MFCDlg::OnBnClickedR ON_BN_CLICKED(IDCANCEL, adio23) ON BN CLICKED(IDC RADIO24, &CEncrypt_Decrypt_MFCDlg::OnBnClickedC $\& CEncrypt_Decrypt_MFCDlg::OnBnClickedR$ ancel) ON_BN_CLICKED(IDSETKEY, adio24) &CEncrypt_Decrypt_MFCDlg::OnBnClickedS ON_BN_CLICKED(IDC_RADIO31, etkey) &CEncrypt_Decrypt_MFCDlg::OnBnClickedR ON BN CLICKED(IDENCRYPT, adio31) ON BN CLICKED(IDC RADIO32, &CEncrypt Decrypt MFCDlg::OnBnClickedE ncrypt) &CEncrypt_Decrypt_MFCDlg::OnBnClickedR ON BN CLICKED(IDDECRYPT, adio32) &CEncrypt_Decrypt_MFCDlg::OnBnClicked ON BN CLICKED(IDCHOOSEFILE, Decrypt) &CEncrypt_Decrypt_MFCDlg::OnBnClickedC ON_BN_CLICKED(IDC_RADIO11, hoosefile) ON_CBN_SELCHANGE(IDC_COMBO_K &CEncrypt_Decrypt_MFCDlg::OnBnClickedR adio11) EYBYTES, ON BN CLICKED(IDC RADIO12, &CEncrypt_Decrypt_MFCDlg::OnCbnSelcha &CEncrypt Decrypt MFCDlg::OnBnClickedR ngeComboKeybytes) adio12) ON_WM_NCHITTEST() ON_BN_CLICKED(IDC_RADIO13, ON_WM_CTLCOLOR() &CEncrypt_Decrypt_MFCDlg::OnBnClickedR ON_WM_TIMER() adio13) END MESSAGE MAP() ON BN CLICKED(IDC RADIO14, // CEncrypt Decrypt MFCDIg 消息处理程 &CEncrypt_Decrypt_MFCDlg::OnBnClickedR 序

adio14)

BOOL

// 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");
GetDlgltem(IDENCRYPT)->ShowWind
ow(FALSE);

GetDlgltem(IDDECRYPT)->ShowWind ow(FALSE):

GetDlgItem(IDCHOOSEFILE)->ShowWindow(FALSE);

GetDlgItem(IDC_BITSTRING_EDIT)->Sh owWindow(FALSE);//隐藏比特流框

GetDlgItem(IDC_STATIC7)->ShowWin
dow(FALSE);

GetDlgItem(IDC_PLAIN_EDIT)->Enable Window(FALSE);

GetDlgItem(IDC_CONFIDENTIAL_EDIT)
->EnableWindow(FALSE);

GetDlgItem(IDC_EDIT_KEY1)->ShowWindow(FALSE);

GetDlgItem(IDC_EDIT_KEY2)->ShowWindow(FALSE);

GetDlgItem(IDC_EDIT_KEY3)->ShowWi
ndow(FALSE);

GetDlgItem(IDC_STATIC3)->ShowWin
dow(FALSE);

GetDlgItem(IDC_STATIC4)->ShowWin dow(FALSE);

GetDlgItem(IDC_STATIC5)->ShowWin
dow(FALSE);

GetDlgItem(IDC_PLAIN_EDIT)->Show Window(FALSE);//显示明文输入框

GetDlgItem(IDC_STATIC1)->ShowWin
dow(FALSE);

GetDlgItem(IDC_EDIT_KEY)->ShowWin dow(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

GetDlgltem(IDC_BITSTRING_EDIT)->Sh

owWindow(FALSE);//隐藏比特流框

GetDlgItem(IDC_ORIGINAL_EDIT)->Sh owWindow(FALSE);//隐藏原文框

GetDlgItem(IDC_STATIC6)->ShowWin dow(FALSE);

GetDlgItem(IDC_STATIC7)->ShowWin dow(FALSE);

GetDlgItem(IDC_STATIC8)->ShowWin dow(FALSE);

GetDlgItem(IDC_BITSTRING_EDIT)->En ableWindow(FALSE);

GetDlgItem(IDC_EDIT_KEY1)->Enable
Window(FALSE);

GetDlgltem(IDC_EDIT_KEY2)->Enable Window(FALSE);

GetDlgItem(IDC_EDIT_KEY3)->Enable
Window(FALSE);

GetDlgltem(IDC_RADIO11)->ShowWin

```
dow(FALSE);
                                            dc.FillSolidRect(rect,
    GetDlgltem(IDC_RADIO12)->ShowWin
                                        RGB(186,226,234));
                                                          //背景
dow(FALSE);
                                            if (Islconic())
    GetDlgltem(IDC_RADIO13)->ShowWin
                                            {
                                                CPaintDC dc(this); // 用于绘制的
dow(FALSE);
                                        设备上下文
   GetDlgltem(IDC_RADIO14)->ShowWin
dow(FALSE);
   GetDlgltem(IDC_RADIO15)->ShowWin
                                            SendMessage(WM_ICONERASEBKGND
dow(FALSE);
   GetDlgltem(IDC_RADIO21)->ShowWin
                                        reinterpret_cast<WPARAM>(dc.GetSafeHdc
dow(FALSE);
                                        ()), 0);
    GetDlgltem(IDC_RADIO22)->ShowWin
                                               // 使图标在工作区矩形中居中
dow(FALSE);
                                                int cxlcon =
   GetDlgltem(IDC_RADIO23)->ShowWin
                                        GetSystemMetrics(SM_CXICON);
dow(FALSE);
                                                int cylcon =
   GetDlgltem(IDC_RADIO24)->ShowWin
                                        GetSystemMetrics(SM_CYICON);
dow(FALSE);
                                                CRect rect;
   GetDlgltem(IDC_STATIC9)->ShowWin
                                                GetClientRect(&rect);
dow(FALSE);
                                                int x = (rect.Width() - cxlcon + 1)
   GetDlgItem(IDC_STATIC10)->ShowWi
                                        / 2;
ndow(FALSE);
                                                int y = (rect.Height() - cylcon + 1)
   return TRUE; // 除非将焦点设置到控
                                        / 2;
件, 否则返回 TRUE
                                                // 绘制图标
}
                                                dc.Drawlcon(x, y, m_hlcon);
// 如果向对话框添加最小化按钮, 则需要
                                            }
下面的代码
                                            else CDialogEx::OnPaint();
// 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。
                                        //当用户拖动最小化窗口时系统调用此函数
                                        取得光标
void CEncrypt_Decrypt_MFCDlg::OnPaint()
                                        //显示。
                                        HCURSOR
{
                                        CEncrypt_Decrypt_MFCDlg::OnQueryDraglc
    CRect
           rect;
   CPaintDC
              dc(this);
                                        on()
   GetClientRect(rect);
```

```
GetDlgItem(IDSETKEY)->ShowWindow
    return
static_cast<HCURSOR>(m_hlcon);
                                         (FALSE);//禁用设置密钥按钮
}
                                             CString CKey;
                                             GetDlgItem(IDC_EDIT_KEY)->GetWind
void
                                         owText(CKey);//获取密钥
CEncrypt_Decrypt_MFCDlg::OnBnClickedOk
                                             string skey;
()//确认响应
                                             CStringA temp(CKey.GetBuffer(0));
                                             CKey.ReleaseBuffer();
}
                                             skey = temp.GetBuffer(0);
                                             temp.ReleaseBuffer();
                                             string key1, key2, key3;
void
                                             int nIndex =
CEncrypt_Decrypt_MFCDlg::OnBnClickedCa
                                         m_cb_keybytes.GetCurSel();
ncel()//取消响应
                                             CString indexText;
                                             m_cb_keybytes.GetLBText(nIndex,
{
    CDialogEx::OnCancel();
                                         indexText);
}
                                             if (indexText==L"8 字节")//选择 8 字节
                                         密钥
void
                                             {
CEncrypt_Decrypt_MFCDlg::OnBnClickedSet
                                                 //相关提示信息
key()//按下设置密钥按钮
{
                                         (skey.length()<8)::MessageBox(NULL, L"密
        //显示 3 个密钥框
                                         钥少于8字节,系统将补零!",L"提示
    GetDlgltem(IDC_EDIT_KEY1)->ShowWi
                                         ",MB_OK|MB_ICONINFORMATION);
ndow(TRUE);
                                                 if (skey.length()>8)
                                             ::MessageBox(NULL, L"密钥多于8字
    GetDlgltem(IDC_EDIT_KEY2)->ShowWi
ndow(TRUE);
                                         节,系统去除多余部分!",L"提示",MB_OK
                                         | MB ICONINFORMATION);
    GetDlgltem(IDC EDIT KEY3)->ShowWi
ndow(TRUE);
                                                 while (skey.length()<8)
                                         skey.append(1, '0');//少于 8 字节补零
    GetDlgltem(IDC_STATIC3)->ShowWin
                                                 skey = skey.substr(0, 8);//多于 8
dow(TRUE);
    GetDlgltem(IDC_STATIC4)->ShowWin
                                         字节截断
dow(TRUE);
                                                 //3 个密钥均为上述生成的 8 字节
    GetDlgltem(IDC_STATIC5)->ShowWin
                                         密钥
dow(TRUE);
                                                 key1 = skey;
```

```
key2 = skey;
        key3 = skey;
                                         owText(k1);
   }
    else if (indexText == L"24 字节")//选择
                                          owText(k2);
24 字节密钥
   {
                                         owText(k3);
        //相关提示信息
(skey.length()<24)::MessageBox(NULL, L"密
钥少于 24 字节, 系统将补零!", L"提示",
                                         选项
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
                                         dow(TRUE);
               skey.append(1, '0');
        skey = skey.substr(0, 24);//多于 24
字节截断
                                         dow(TRUE);
        //密钥1为0-7字节
        key1 = skey.substr(0, 8);
                                         dow(TRUE);
        //密钥 2 为 8-15 字节
        key2 = skey.substr(8, 8);
                                         dow(TRUE);
        //密钥 3 为 16-23 字节
        key3 = skey.substr(16, 8);
                                         dow(TRUE);
   }
   //将 3 个密钥转换为 CString 类型
                                         dow(TRUE);
    CString k1,k2,k3;
    k1 = CA2T(key1.c_str());
                                         dow(TRUE);
    k2 = CA2T(key2.c_str());
    k3 = CA2T(key3.c_str());
                                         dow(TRUE);
    Inintialize_Encrypt_Decrypt(key1, key2,
                                              GetDlgltem(IDC RADIO24)->ShowWin
key3);//初始化密钥
                                         dow(TRUE);
   //将三个密钥显示出来
                                              GetDlgltem(IDC_STATIC9)->ShowWin
```

GetDlgItem(IDC_EDIT_KEY1)->SetWind GetDlgltem(IDC_EDIT_KEY2)->SetWind GetDlgltem(IDC_EDIT_KEY3)->SetWind GetDlgItem(IDSETKEY)->EnableWindo w(FALSE);//禁用设置密钥按钮 GetDlgltem(IDC COMBO KEYBYTES)->EnableWindow(FALSE);//禁用选择字节数 GetDlgItem(IDC_EDIT_KEY)->EnableWi ndow(FALSE);//禁用密钥输入框 GetDlgltem(IDENCRYPT)->ShowWind ow(TRUE);//启用加密按钮 //启用加密模式和填充模式选择键 GetDlgltem(IDC_RADIO11)->ShowWin GetDlgltem(IDC RADIO12)->ShowWin GetDlgltem(IDC_RADIO13)->ShowWin GetDlgltem(IDC RADIO14)->ShowWin GetDlgItem(IDC_RADIO15)->ShowWin GetDlgltem(IDC RADIO21)->ShowWin GetDlgltem(IDC_RADIO22)->ShowWin GetDlgltem(IDC_RADIO23)->ShowWin

```
dow(TRUE);
    GetDlgltem(IDC_STATIC10)->ShowWi
                                            GetDlgltem(IDC_BITSTRING_EDIT)->Sh
                                        owWindow(TRUE);//显示比特流框
ndow(TRUE);
}
                                            GetDlgItem(IDC_STATIC7)->ShowWin
void
                                        dow(TRUE);
CEncrypt_Decrypt_MFCDlg::OnBnClickedEn
                                            }
crypt()//按下加密按钮
                                            GetDlgItem(IDC_ORIGINAL_EDIT)->Sh
                                        owWindow(FALSE);//隐藏明文框
{
                                            GetDlgItem(IDC_STATIC8)->ShowWin
    GetDlgItem(IDSETKEY)->EnableWindo
w(FALSE);//禁用设置密钥按钮
                                        dow(FALSE);
    GetDlgltem(IDC_COMBO_KEYBYTES)-
                                            //禁用加密模式和填充模式选择键
>EnableWindow(FALSE);//禁用密钥选择框
                                            GetDlgltem(IDC_PLAIN_EDIT)->Enable
    GetDlgltem(IDC_EDIT_KEY)->EnableWi
                                        Window(FALSE);
ndow(FALSE);//禁用密钥输入框
                                            GetDlgltem(IDC_RADIO11)->ShowWin
    GetDlgItem(IDC_CONFIDENTIAL_EDIT)
                                        dow(FALSE);
->ShowWindow(TRUE);//显示密文框
                                            GetDlgltem(IDC_RADIO12)->ShowWin
    GetDlgltem(IDC_STATIC6)->ShowWin
                                        dow(FALSE);
dow(TRUE);
                                            GetDlgltem(IDC RADIO13)->ShowWin
    GetDlgItem(IDSETKEY)->ShowWindow
                                        dow(FALSE);
(FALSE);//禁用设置密钥按钮
                                            GetDlgItem(IDC_RADIO14)->ShowWin
    GetDlgltem(IDC_EDIT_KEY1)->ShowWi
                                        dow(FALSE);
ndow(FALSE);
                                            GetDlgltem(IDC RADIO15)->ShowWin
   GetDlgltem(IDC_EDIT_KEY2)->ShowWi
                                        dow(FALSE);
ndow(FALSE);
                                            GetDlgItem(IDC_RADIO21)->ShowWin
    GetDlgItem(IDC_EDIT_KEY3)->ShowWi
                                        dow(FALSE);
ndow(FALSE);
                                            GetDlgltem(IDC RADIO22)->ShowWin
                                        dow(FALSE);
   GetDlgltem(IDC STATIC3)->ShowWin
dow(FALSE);
                                            GetDlgItem(IDC_RADIO23)->ShowWin
   GetDlgltem(IDC_STATIC4)->ShowWin
                                        dow(FALSE);
                                            GetDlgltem(IDC_RADIO24)->ShowWin
dow(FALSE);
   GetDlgltem(IDC_STATIC5)->ShowWin
                                        dow(FALSE);
dow(FALSE);
                                            GetDlgltem(IDC RADIO31)->EnableWi
   if (m_enchoose == 0)
                                        ndow(FALSE);
                                            GetDlgItem(IDC_RADIO32)->EnableWi
```

```
ndow(FALSE);
                                                 else if (m_encryptway == 1 &&
    GetDlgltem(IDC_STATIC9)->ShowWin
                                             m_fillway == 0)c_t =
dow(FALSE);
                                             Encrypt_All_Data_CBC_ZERO(m_t);
    GetDlgItem(IDC_STATIC10)->ShowWi
                                                 else if (m_encryptway == 1 &&
ndow(FALSE);
                                             m_fillway == 1)c_t =
    UpdateData(TRUE);
                                             Encrypt_All_Data_CBC_PKCS7(m_t);
    CString CPlainText;
                                                 else if (m_encryptway == 1 &&
    if (m_enchoose==1)//选择加密文件
                                             m_fillway == 2)c_t =
        CPlainText = cfileplain;//明文为文
                                             Encrypt All Data CBC ANSIX923(m t);
件内容
                                                 else if (m_encryptway == 1 &&
    else
                                             m_fillway == 3)c_t =
                                             Encrypt_All_Data_CBC_ISO10126(m_t);
    GetDlgltem(IDC_PLAIN_EDIT)->GetWi
                                                 else if (m_encryptway == 2 &&
ndowText(CPlainText);//获取明文输入框信
                                             m fillway == 0)c t =
息
                                             Encrypt_All_Data_CFB_ZERO(m_t);
    //CString 转换为 string 类型
                                                 else if (m_encryptway == 2 &&
    CStringA t(CPlainText.GetBuffer(0));
                                             m_fillway == 1)c_t =
    CPlainText.ReleaseBuffer();
                                             Encrypt_All_Data_CFB_PKCS7(m_t);
    string m t = t.GetBuffer(0);//m t 为明
                                                 else if (m_encryptway == 2 &&
文
                                             m_fillway == 2)c_t =
    t.ReleaseBuffer();
                                             Encrypt_All_Data_CFB_ANSIX923(m_t);
                                                 else if (m_encryptway == 2 &&
    string c_t;
    //根据选择的加密模式和填充方式加密
                                             m fillway == 3)c t =
    if (m_encryptway == 0 && m_fillway
                                             Encrypt_All_Data_CFB_ISO10126(m_t);
== 0)c_t =
                                                 else if (m_encryptway == 3 &&
                                             m_fillway == 0)c_t =
Encrypt_All_Data_ECB_ZERO(m_t);
    else if (m encryptway == 0 &&
                                             Encrypt_All_Data_OFB_ZERO(m_t);
m fillway == 1)c t =
                                                 else if (m encryptway == 3 &&
Encrypt_All_Data_ECB_PKCS7(m_t);
                                             m_fillway == 1)c_t =
    else if (m_encryptway == 0 &&
                                             Encrypt_All_Data_OFB_PKCS7(m_t);
m_fillway == 2)c_t =
                                                 else if (m_encryptway == 3 &&
Encrypt_All_Data_ECB_ANSIX923(m_t);
                                             m_fillway == 2)c_t =
    else if (m_encryptway == 0 &&
                                             Encrypt_All_Data_OFB_ANSIX923(m_t);
m_fillway == 3)c_t =
                                                 else if (m_encryptway == 3 &&
Encrypt_All_Data_ECB_ISO10126(m_t);
                                             m_fillway == 3)c_t =
```

```
Encrypt_All_Data_OFB_ISO10126(m_t);
                                               if (m_enchoose == 1)//文件加密
    else if (m_encryptway == 4 &&
                                               {
                                                    //将加密结果保存到文件中
m_fillway == 0)c_t =
Encrypt_All_Data_CTR_ZERO(m_t);
                                                    // 设置过滤器
                                                    TCHAR szFilter[] = _T("文本文件
    else if (m_encryptway == 4 &&
m_fillway == 1)c_t =
                                           (*.txt)|*.txt|所有文件(*.*)|*.*||");
Encrypt_All_Data_CTR_PKCS7(m_t);
                                                    // 构造保存文件对话框
                                                    CFileDialog fileDlg(FALSE,
    else if (m_encryptway == 4 &&
                                           T("txt"), NULL, OFN HIDEREADONLY I
m fillway == 2)c t =
                                           OFN_OVERWRITEPROMPT, szFilter, this);
Encrypt_All_Data_CTR_ANSIX923(m_t);
    else if (m_encryptway == 4 &&
                                                    fileDlg.m_ofn.lpstrTitle = L"保存密
m_fillway == 3)c_t =
                                           文文件":
Encrypt_AII_Data_CTR_ISO10126(m_t);
                                                    CString strFilePath;
    string temp0 = "";//比特流 (16 进制)
                                                    if (IDOK == fileDlg.DoModal())
    for (int i = 0; i < c_t.length(); i++)
                                                    // 如果点击了文件对话框上的"保
        int t0 = (int)c_t[i];//记录字符串的
                                           存"按钮,则将选择的文件路径显示到编辑
某一位字符
                                           框里
        if (t0 < 0) t0 = 256 + t0;//负数处
                                                        strFilePath =
玾
                                           fileDlg.GetPathName();
        char c0, c1;
                                                        // 判断文件是否存在,如果存
        c0 = (t0 \% 16 < 10) ? (t0 \% 16 +
                                           在则去掉只读属性
'0'):(t0 % 16 - 10 + 'A');//16 进制的低位
                                                        if (PathFileExists(strFilePath)
        c1 = (t0 / 16 < 10) ? (t0 / 16 +
                                           && !PathIsDirectory(strFilePath))
'0'):(t0 / 16 - 10 + 'A');//16 进制的高位
        //字符串追加字符
                                                            DWORD dwAttrs =
        temp0.append("0x");
                                           GetFileAttributes(strFilePath);
        temp0.append(1, c1);
                                                            if (dwAttrs !=
        temp0.append(1, c0);
                                           INVALID_FILE_ATTRIBUTES&& (dwAttrs &
        temp0.append(" ");
                                           FILE_ATTRIBUTE_READONLY))
        if ((i + 1) \% 7 == 0)
temp0.append("\r\n");//换行
                                                                dwAttrs &=
   }
                                           ~FILE ATTRIBUTE READONLY;
    CString CConText;
    CConText = CA2T(c_t.c_str());
                                                SetFileAttributes(strFilePath, dwAttrs);
```

```
}
                                             }
           }
                                             GetDlgItem(IDENCRYPT)->ShowWind
           // 打开文件
                                         ow(FALSE);//禁用加密按钮
            CStdioFile file:
                                             GetDlgItem(IDDECRYPT)->ShowWind
            BOOL ret =
                                         ow(TRUE);//启用解密按钮
                                             GetDlgltem(IDC_ORIGINAL_EDIT)->Set
file.Open(strFilePath,CFile::modeCreate |
CFile::modeWrite | CFile::shareDenyWrite);
                                         WindowText(L"");//清空解密框内容
           if (!ret)
                                         }
           {
               ::AfxMessageBox(L"打开
                                         void
文件失败");
                                         CEncrypt_Decrypt_MFCDlg::OnBnClickedDe
                                         crypt()//按下解密按钮
               return;
           }
                                         {
           file.SeekToEnd();
                                             GetDlgItem(IDENCRYPT)->ShowWind
           file.WriteString(CConText);//
                                         ow(TRUE);//启用加密按钮
写入文件
                                             GetDlgItem(IDDECRYPT)->ShowWind
           file.Close();//关闭文件
                                         ow(FALSE);//禁用解密按钮
           strFilePath = L"<文件>"+
                                             GetDlgltem(IDC_CONFIDENTIAL_EDIT)
strFilePath;//显示文件路径
                                         ->ShowWindow(FALSE);//隐藏密文框
                                             GetDlgltem(IDC_BITSTRING_EDIT)->Sh
   GetDlgItem(IDC_CONFIDENTIAL_EDIT)
                                         owWindow(FALSE);//隐藏比特流框
->SetWindowText(strFilePath);
                                             GetDlgItem(IDC_ORIGINAL_EDIT)->Sh
                                         owWindow(TRUE);//显示原文框
       }
                                             GetDlgltem(IDC_STATIC6)->ShowWin
                                         dow(FALSE);
   else//字符串加密
                                             GetDlgltem(IDC_STATIC7)->ShowWin
                                         dow(FALSE);
GetDlgItem(IDC_CONFIDENTIAL_EDIT)->Se
                                             GetDlgltem(IDC STATIC8)->ShowWin
tWindowText(CConText);//显示加密后的字
                                         dow(TRUE);
符串
                                             UpdateData(TRUE);
       CString t1;
                                             CString CConText;
       t1 = CA2T(temp0.c_str());
                                             if (m_enchoose == 1)//解密文件
    GetDlgItem(IDC_BITSTRING_EDIT)->Se
                                                 CString strFilePath;
tWindowText(t1);//显示加密后的比特串
                                         GetDlgltem(IDC_CONFIDENTIAL_EDIT)->G
```

```
etWindowText(strFilePath);//获取文件路径
                                             Decrypt_All_Data_ECB_ZERO(c_t);
        strFilePath = strFilePath.Mid(4);//
                                                 else if (m_encryptway == 0 &&
去除密文框前面的<文件>标识符
                                            m_fillway == 1)m_t =
        CStdioFile file:
                                             Decrypt_All_Data_ECB_PKCS7(c_t);
        CString strText;
                                                 else if (m_encryptway == 0 &&
        //打开文件
                                            m_fillway == 2)m_t =
        if (!file.Open(strFilePath,
                                            Decrypt_All_Data_ECB_ANSIX923(c_t);
CFile::modeRead))
                                                 else if (m_encryptway == 0 &&
        {
                                            m fillway == 3)m t =
            ::AfxMessageBox(_T("文件打
                                            Decrypt_All_Data_ECB_ISO10126(c_t);
开失败。"));
                                                 else if (m_encryptway == 1 &&
                                            m_fillway == 0)m_t =
            return;
        }
                                            Decrypt_All_Data_CBC_ZERO(c_t);
        //读文件
                                                 else if (m_encryptway == 1 &&
        strText = _T("");
                                            m_fillway == 1)m_t =
        CConText = L"";
                                            Decrypt_All_Data_CBC_PKCS7(c_t);
        while (file.ReadString(strText))
                                                 else if (m_encryptway == 1 &&
             CConText = CConText +
                                            m_fillway == 2)m_t =
strText;//追加到密文中
                                            Decrypt_All_Data_CBC_ANSIX923(c_t);
        //关闭文件
                                                 else if (m_encryptway == 1 &&
        file.Close();
                                            m_fillway == 3)m_t =
    }
                                            Decrypt_All_Data_CBC_ISO10126(c_t);
    else
                                                 else if (m encryptway == 2 &&
                                            m_fillway == 0)m_t =
    GetDlgItem(IDC_PLAIN_EDIT)->GetWi
                                             Decrypt_All_Data_CFB_ZERO(c_t);
ndowText(CConText);//获取密文框信息
                                                 else if (m_encryptway == 2 &&
    //CString 转换成 string 类型
                                            m fillway == 1)m t =
    CStringA t(CConText.GetBuffer(0));
                                             Decrypt_All_Data_CFB_PKCS7(c_t);
    CConText.ReleaseBuffer();
                                                 else if (m_encryptway == 2 &&
    string c_t = t.GetBuffer(0);//c_t 为密文
                                            m_fillway == 2)m_t =
    t.ReleaseBuffer();
                                            Decrypt_All_Data_CFB_ANSIX923(c_t);
                                                 else if (m_encryptway == 2 &&
    string m_t;
    //根据选择的加密方式和填充模式解密
                                            m fillway == 3)m t =
    if (m_encryptway == 0 && m_fillway
                                            Decrypt_All_Data_CFB_ISO10126(c_t);
== 0)m t =
                                                 else if (m_encryptway == 3 &&
```

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

```
::AfxMessageBox(L"打开
                                            GetDlgItem(IDC_RADIO15)->ShowWin
文件失败");
                                        dow(TRUE);
                                            GetDlgltem(IDC_RADIO21)->ShowWin
               return:
                                        dow(TRUE);
           }
           file.SeekToEnd();
                                            GetDlgItem(IDC_RADIO22)->ShowWin
                                        dow(TRUE);
           file.WriteString(COriText);//写
入文件
                                            GetDlgItem(IDC_RADIO23)->ShowWin
                                        dow(TRUE);
           file.Close();
           strFilePath = L"<文件>"+
                                            GetDlgltem(IDC RADIO24)->ShowWin
strFilePath;//记录文件路径
                                        dow(TRUE);
                                            GetDlgltem(IDC_RADIO31)->EnableWi
   GetDlgItem(IDC_ORIGINAL_EDIT)->Set
                                        ndow(TRUE);
WindowText(strFilePath);//显示文件路径
                                            GetDlgItem(IDC_RADIO32)->EnableWi
       }
                                        ndow(TRUE);
   }
                                            GetDlgltem(IDC_STATIC9)->ShowWin
   else//解密字符串
                                        dow(TRUE);
                                            GetDlgltem(IDC_STATIC10)->ShowWi
    GetDlgltem(IDC_ORIGINAL_EDIT)->Set
                                        ndow(TRUE);
WindowText(COriText);//显示解密后的原文
                                            GetDlgltem(IDC PLAIN EDIT)->Enable
    GetDlgItem(IDSETKEY)->EnableWindo
                                        Window(TRUE);//启用明文输入框
w(TRUE);//启用设置密钥按钮
                                            GetDlgItem(IDC_CONFIDENTIAL_EDIT)
    GetDlgItem(IDC_COMBO_KEYBYTES)-
                                        ->SetWindowText(L"");//清空密文框
>EnableWindow(TRUE);//启用设置密钥位
                                            GetDlgItem(IDC_BITSTRING_EDIT)->Se
数框
                                        tWindowText(L"");//清空比特流框
    GetDlgltem(IDC_EDIT_KEY)->EnableWi
                                            GetDlgItem(IDSETKEY)->ShowWindow
ndow(TRUE);//启用密钥输入框
                                        (TRUE);//启用设置密钥按钮
   //启用加密模式和填充模式选择键
                                        }
   GetDlgltem(IDC RADIO11)->ShowWin
dow(TRUE);
                                        void
   GetDlgltem(IDC_RADIO12)->ShowWin
                                        CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
                                        dio11()//ECB 模式
dow(TRUE);
   GetDlgltem(IDC_RADIO13)->ShowWin
                                        {
dow(TRUE);
                                            m_{encryptway} = 0;
    GetDlgltem(IDC_RADIO14)->ShowWin
                                        }
```

dow(TRUE);

```
void
                                               m_fillway = 0;
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
                                           }
dio12()//CBC 模式
{
                                           void
                                           CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
    m_{encryptway} = 1;
                                           dio22()//PKCS7 模式
}
                                           {
                                                   m_fillway = 1;
void
                                           }
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
dio13()//CFB 模式
                                           void
{
                                           CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
                                           dio23()//ANSI X.923 模式
    m_encryptway = 2;
}
                                           {
                                               m_fillway = 2;
                                           }
void
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
dio14()//OFB 模式
                                           void
{
                                           CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
                                           dio24()//ISO 10126 模式
    m_encryptway = 3;
}
                                           {
                                               m_fillway = 3;
void
                                           }
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
dio15()//CTR 模式
{
                                           void
                                           CEncrypt_Decrypt_MFCDlg::OnBnClickedRa
        m_encryptway = 4;
                                           dio31()//选择加密字符串方式
}
                                           {
                                               m_{enchoose} = 0;
void
                                               GetDlgltem(IDC_PLAIN_EDIT)->Show
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
                                           Window(TRUE);//显示明文输入框
dio21()//ZERO 模式
                                               GetDlgItem(IDC_STATIC1)->ShowWin
                                           dow(TRUE);
{
```

```
GetDlgltem(IDCHOOSEFILE)->ShowWi
ndow(FALSE);//禁用选择文件按钮
                                         void
    GetDlgltem(IDC_PLAIN_EDIT)->Enable
                                         CEncrypt_Decrypt_MFCDlg::OnBnClickedCh
Window(TRUE);//启用明文输入框
                                         oosefile()
    GetDlgltem(IDC_PLAIN_EDIT)->SetWin
                                         {
dowText(L"");//清空明文输入框
                                             // 设置过滤器
    GetDlgltem(IDC_CONFIDENTIAL_EDIT)
                                             TCHAR szFilter[] = _T("文本文件
->SetWindowText(L"");//清空密文框
                                         (*.txt)|*.txt|所有文件(*.*)|*.*||");
    GetDlgltem(IDC ORIGINAL EDIT)->Set
                                             // 构造打开文件对话框
WindowText(L"");//清空原文框
                                             CFileDialog fileDlg(TRUE, _T("txt"),
}
                                         NULL, 0, szFilter, this);
                                             fileDlg.m_ofn.lpstrTitle = L"打开明文文
                                         件";
                                             CString strFilePath;
void
CEncrypt_Decrypt_MFCDIg::OnBnClickedRa
                                             if (IDOK == fileDlg.DoModal())
dio32()//选择加密文件方式
                                                 // 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
    m enchoose = 1;
    GetDlgltem(IDC_STATIC7)->ShowWin
                                         框里
dow(FALSE);
                                                 strFilePath =
    GetDlgltem(IDC_STATIC1)->ShowWin
                                         fileDlg.GetPathName();
dow(TRUE);
                                                 CStdioFile file:
    GetDlgItem(IDC_PLAIN_EDIT)->Show
                                                 CString strText;
Window(FALSE);//隐藏明文输入框
                                                 //打开文件
    GetDlgItem(IDCHOOSEFILE)->ShowWi
                                                 if (!file.Open(strFilePath,
                                         CFile::modeRead))
ndow(TRUE);//启用选择文件按钮
    GetDlgItem(IDC_PLAIN_EDIT)->Enable
                                                 {
Window(FALSE);//禁用明文输入框
                                                     ::AfxMessageBox( T("文件打
                                         开失败。"));
    GetDlgltem(IDC_PLAIN_EDIT)->SetWin
dowText(L"");//清空明文框
                                                     return;
    GetDlgltem(IDC_CONFIDENTIAL_EDIT)
                                                 }
->SetWindowText(L"");//清空密文框
                                                 //读文件
    GetDlgltem(IDC_ORIGINAL_EDIT)->Set
                                                 strText = _T("");
WindowText(L"");//清空原文框
                                                 cfileplain = L"";
                                                 while (file.ReadString(strText))
}
```

```
cfileplain = cfileplain +
                                           HTBOTTOMLEFT == ret ||
strText:
                                           HTBOTTOMRIGHT == ret || HTTOPLEFT ==
        //关闭文件
                                           ret || HTTOPRIGHT == ret || HTCAPTION
        file.Close();
                                           == ret)
                                                   return HTCLIENT:
    GetDlgItem(IDC_PLAIN_EDIT)->Show
                                               return ret;
Window(TRUE);//显示明文输入框
                                           }
        strFilePath = L"<文件>"+
strFilePath;
                                           HBRUSH
                                           CEncrypt_Decrypt_MFCDlg::OnCtlColor(CD
    GetDlgltem(IDC_PLAIN_EDIT)->SetWin
                                           C* pDC, CWnd* pWnd, UINT nCtlColor)
dowText(strFilePath);
                                           {
                                               HBRUSH hbr =
    }
}
                                           CDialog::OnCtlColor(pDC, pWnd,
                                           nCtlColor);
void
                                               // TODO: Change any attributes of
CEncrypt_Decrypt_MFCDlg::OnCbnSelchan
                                           the DC here
geComboKeybytes()
                                               if (nCtlColor == CTLCOLOR_BTN)
                                           //更改按钮颜色
{
    GetDlgItem(IDC_EDIT_KEY)->ShowWin
                                               {
                                                       pDC->SetTextColor(RGB(254,
dow(TRUE);//显示密钥输入框
                                           1, 1));
    GetDlgItem(IDSETKEY)->ShowWindow
                                                   pDC->SetBkColor(RGB(117, 31,
(TRUE);//显示设置密钥按钮
                                           111));
    GetDlgltem(IDC_EDIT_KEY)->SetWind
                                                   HBRUSH b =
owText(L"");//密钥输入框清空
                                           CreateSolidBrush(RGB(186, 226, 234));
}
                                                   return b;
                                               }
                                               else if (nCtlColor == CTLCOLOR_EDIT)
LRESULT
                                           //更改编辑框
CEncrypt_Decrypt_MFCDlg::OnNcHitTest(C
Point point)
                                               {
    int ret = CDialog::OnNcHitTest(point);
                                                   pDC->SetTextColor(RGB(24, 6,
    //禁止改变窗口大小
                                           102));
    if (HTTOP == ret || HTBOTTOM == ret
                                                   pDC->SetBkColor(RGB(254, 198,
|| HTLEFT == ret || HTRIGHT == ret ||
                                           103));
```

```
HBRUSH b =
                                            }
CreateSolidBrush(RGB(186, 226, 234));
        return b:
                                            void
                                            CEncrypt_Decrypt_MFCDlg::OnTimer(UINT_
    }
     if (nCtlColor == CTLCOLOR_STATIC)
                                            PTR nIDEvent)
//更改静态文本
                                            {
                                                 CString strTime;
    {
        pDC->SetTextColor(RGB(254, 1,
                                                 CTime tm;
1));
                                                 tm = CTime::GetCurrentTime();
                                                 strTime =
        pDC->SetBkColor(RGB(186, 226,
234));
                                            tm.Format("%Y-%m-%d %H:%M:%S");
        HBRUSH b =
                                                 SetDlgItemText(IDC_ShowTime,
CreateSolidBrush(RGB(186, 226, 234));
                                                             //显示系统时间
                                            strTime);
                                                 CDialogEx::OnTimer(nIDEvent);
        return b;
    }
                                            }
    return hbr;
```

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++) {
         I = (ulong32)pc1[j];
         m = 1 \& 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
                                              #else
位*/
                                              static void cookey(const ulong32 *raw1,
             I = j + (ulong32)totrot[i];
                                              ulong32 *keyout)
             if (1 < 28) pcr[i] = pc1m[l];
                                              #endif
             else pcr[i] = pc1m[I - 28];
        }
                                                   ulong32 *cook;
         for (/*j = 28*/; j < 56; j++)
                                                  const ulong32 *raw0;
{
       /*子密钥的后半部分循环移位*/
                                                  ulong32 dough[32];
             I = j + (ulong32)totrot[i];
                                                  int i;
             if (I < 56) pcr[j] = pc1m[l];
                                                  cook = dough;
             else pcr[j] = pc1m[l - 28];
                                              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、
        }
         for (j = 0; j < 24; j++)
                                              2、4、6、8排序*/
{
             /*对 48bit 密钥进行置换*/
                                                  {
             if ((int)pcr[(int)pc2[j]] != 0)
                                                       raw0 = raw1++;
kn[m] |= bigbyte[j];
                                                       *cook = (*raw0 & 0x00fc0000L)
             if ((int)pcr[(int)pc2[j + 24]] !=
                                              << 6;
0) kn[n] |= bigbyte[j];
                                                       *cook |= (*raw0 & 0x00000fc0L)
                                              << 10;
         }
    }
                                                       *cook |= (*raw1 &
                                              0x00fc0000L) >> 10;
    cookey(kn, keyout);
}
                                                       *cook++ |= (*raw1 &
                                              0x00000fc0L) >> 6;
#ifdef CLEAN STACK
void deskey(const unsigned char *key,
                                                       *cook = (*raw0 & 0x0003f000L)
short edf, ulong32 *keyout)
                                              << 12;
{
                                                       *cook |= (*raw0 & 0x0000003fL)
    _deskey(key, edf, keyout);
                                              << 16;
    burn stack(sizeof(int)* 5 +
                                                       *cook |= (*raw1 &
sizeof(ulong32)* 32 + sizeof(unsigned
                                              0x0003f000L) >> 4;
                                                       *cook++ |= (*raw1 &
char)* 112);
                                              0x000003fL);
#endif
                                                  }
#ifdef CLEAN STACK
                                                  memcpy(keyout, dough, sizeof
static void _cookey(const ulong32 *raw1,
                                              dough);
ulong32 *keyout)
                                              }
```

```
#ifdef CLEAN_STACK
                                               0x00ff00ffL:
static void cookey(const ulong32 *raw1,
                                                    leftt ^= work;
ulong32 *keyout)
                                                    right ^= (work << 8);
                                                    right = ROL(right, 1);
{
                                                    work = (leftt ^ right) & 0xaaaaaaaaaL;
    _cookey(raw1, keyout);
    burn_stack(sizeof(ulong32 *)* 2 +
                                                    leftt ^= work;
sizeof(ulong32)* 32 + sizeof(int));
                                                    right ^= work;
}
                                                    leftt = ROL(leftt, 1);
#endif
                                               #else
#ifndef CLEAN_STACK
                                                    {
static void desfunc(ulong32 *block, const
                                                         ulong64 tmp;
ulong32 *keys)
                                               /*加密或解密时进行初始置换*/
                                                         tmp = des_ip[0][byte(leftt, 0)] ^
#else
static void _desfunc(ulong32 *block, const
                                                             des_ip[1][byte(leftt, 1)] ^
ulong32 *keys)
                                                             des_ip[2][byte(leftt, 2)] ^
#endif
                                                             des_ip[3][byte(leftt, 3)] ^
{
                                                             des_ip[4][byte(right, 0)] ^
    ulong32 work, right, leftt;
                                                             des_ip[5][byte(right, 1)] ^
    int round;
                                                             des_ip[6][byte(right, 2)] ^
    leftt = block[0];
                                                             des_ip[7][byte(right, 3)];
    right = block[1];
                                                         leftt = (ulong32)(tmp >> 32);
#ifdef SMALL CODE
                                                         right = (ulong32)(tmp &
                                               0xFFFFFFFUL):
    work = ((leftt >> 4) \land right) &
0x0f0f0f0fL;
                                                    }
                                               #endif
    right ^= work;
    leftt ^= (work << 4);
                                               for (round = 0; round < 8; round++) \{ /* \}
    work = ((leftt >> 16) ^ right) &
                                               对置换后的信息进行运算,主要包括信息
0x0000ffffL;
                                               扩展和查 s 盒*/
    right ^= work;
                                               work = ROR(right, 4) ^ *keys++; /*一共
                                               8 轮循环, 每轮进行加密或解密的两轮运
    leftt ^= (work << 16);
    work = ((right >> 2) \land leftt) &
                                               算*/
                                                         leftt ^= SP7[work & 0x3fL]
0x3333333L;
    leftt ^= work;
                                                             ^ SP5[(work >> 8) \& 0x3fL]
                                                             ^ SP3[(work >> 16) \& 0x3fL]
    right ^= (work << 2);
                                                             ^ SP1[(work >> 24) \& 0x3fL];
    work = ((right >> 8) \land leftt) &
```

```
work = right ^ *keys++;
                                                  0x0f0f0f0fL;
         leftt ^= SP8[work & 0x3fL]
                                                       leftt ^= work;
              ^ SP6[(work >> 8) \& 0x3fL]
                                                       right ^= (work << 4);
              ^ SP4[(work >> 16) \& 0x3fL]
                                                  #else
              ^ SP2[(work >> 24) \& 0x3fL];
                                                       {
         work = ROR(leftt, 4) \land *keys++;
                                                            ulong64 tmp;
         right ^= SP7[work & 0x3fL]
                                                            tmp = des_fp[0][byte(leftt, 0)] ^
              ^ SP5[(work >> 8) \& 0x3fL]
                                                                 des_fp[1][byte(leftt, 1)] ^
              ^ SP3[(work >> 16) & 0x3fL]
                                                                 des_fp[2][byte(leftt, 2)] ^
              ^ SP1[(work >> 24) \& 0x3fL];
                                                                 des_fp[3][byte(leftt, 3)] ^
         work = leftt ^ *keys++;
                                                                 des_{fp}[4][byte(right, 0)] ^
         right ^= SP8[work & 0x3fL]
                                                                 des_fp[5][byte(right, 1)] ^
              ^ SP6[(work >> 8) \& 0x3fL]
                                                                 des_fp[6][byte(right, 2)] ^
              ^ SP4[(work >> 16) \& 0x3fL]
                                                                 des_fp[7][byte(right, 3)];
              ^ SP2[(work >> 24) \& 0x3fL];
                                                            leftt = (ulong32)(tmp >> 32);
    }
                                                            right = (ulong32)(tmp &
#ifdef SMALL_CODE
                                                  0xFFFFFFFUL);
    right = ROR(right, 1);
                                                       }
    work = (leftt ^ right) & 0xaaaaaaaaaL;
                                                  #endif
    leftt ^= work;
                                                       block[0] = right;
    right ^= work;
                                                       block[1] = leftt;
    leftt = ROR(leftt, 1);
                                                  }
    work = ((leftt >> 8) \land right) &
                                                  #ifdef CLEAN STACK
0x00ff00ffL;
                                                  static void desfunc(ulong32 *block, const
                                                  ulong32 *keys)
    right ^= work;
    leftt ^= (work << 8);
    work = ((leftt >> 2) \land right) &
                                                       _desfunc(block, keys);
0x3333333L;
                                                       burn stack(sizeof(ulong32)* 4 +
    right ^= work;
                                                  sizeof(int));
    leftt ^= (work << 2);
                                                  }
    work = ((right >> 16) ^{\land} leftt) &
                                                  #endif
0x0000ffffL;
                                                  int des_setup(const unsigned char *key, int
    leftt ^= work;
                                                  keylen, int num_rounds, symmetric_key
    right ^= (work << 16);
                                                  *skey)
    work = ((right >> 4) \land leftt) &
```

```
_ARGCHK(key != NULL);
                                                    desfunc(work, key->dk);
    _ARGCHK(skey != NULL);
                                                    STORE32H(work[0], pt + 0);
    if (num_rounds != 0 &&
                                                    STORE32H(work[1], pt + 4);
num_rounds != 16)
                           return
                                               }
CRYPT_INVALID_ROUNDS;
    if (keylen != 8)
                           return
                                               int des_test(void)
CRYPT_INVALID_KEYSIZE;
                                               #ifndef LTC_TEST
    deskey(key, EN0, skey->ek);
    deskey(key, DE1, skey->dk);
                                                    return CRYPT NOP;
    return CRYPT_OK;
                                               #else
}
                                                    int err;
                                                    };
void des_ecb_encrypt(const unsigned char
                                                    int i, y;
*pt, unsigned char *ct, symmetric_key *key)
                                                    unsigned char tmp[8];
{
                                                    symmetric_key des;
    ulong32 work[2];
                                                    for (i = 0; i < (int)(sizeof(cases) / 
    _ARGCHK(pt != NULL);
                                               sizeof(cases[0])); i++)
    _ARGCHK(ct != NULL);
                                                    {
    _ARGCHK(key != NULL);
                                                         if ((err = des_setup(cases[i].key, 8,
    LOAD32H(work[0], pt + 0);
                                               0, &des)) != CRYPT_OK) return err;
    LOAD32H(work[1], pt + 4);
                                                         if (cases[i].mode != 0)
    desfunc(work, key->ek);
                                               des_ecb_encrypt(cases[i].txt, tmp, &des);
    STORE32H(work[0], ct + 0);
                                                         else des_ecb_decrypt(cases[i].txt,
    STORE32H(work[1], ct + 4);
                                               tmp, &des);
}
                                                         if (memcmp(cases[i].out, tmp,
                                               sizeof(tmp)) != 0)return
void des_ecb_decrypt(const unsigned char
                                               CRYPT FAIL TESTVECTOR;
*ct, unsigned char *pt, symmetric key *key)
                                                         for (y = 0; y < 8; y++) \text{ tmp}[y] = 0;
{
                                                         for (y = 0; y < 1000; y++)
                                               des_ecb_encrypt(tmp, tmp, &des);
    ulong32 work[2];
                                                         for (y = 0; y < 1000; y++)
    _ARGCHK(pt != NULL);
    _ARGCHK(ct != NULL);
                                               des_ecb_decrypt(tmp, tmp, &des);
    ARGCHK(key != NULL);
                                                         for (y = 0; y < 8; y++) if (tmp[y] !=
    LOAD32H(work[0], ct + 0);
                                               0) return CRYPT_FAIL_TESTVECTOR;
    LOAD32H(work[1], ct + 4);
                                                    }
```

```
return CRYPT_OK;
                                                          *mem++ = 0;
#endif
                                                }
}
                                                void burn_stack(unsigned long len)
int des_keysize(int *desired_keysize)
                                                     unsigned char buf[32];
                                                     zeromem(buf, sizeof(buf));
    _ARGCHK(desired_keysize != NULL);
                                                     if (len > (unsigned long)sizeof(buf))
    if (*desired_keysize < 8) {
                                                          burn_stack(len - sizeof(buf));
         return CRYPT INVALID KEYSIZE;
                                                }
    }
                                                 #include <signal.h>
    *desired_keysize = 8;
                                                 #if (ARGTYPE == 0)
    return CRYPT_OK;
                                                 void crypt_argchk(char *v, char *s, int d)
}
                                                {
                                                     fprintf(stderr, "_ARGCHK '%s' failure on
void zeromem(void *dst, size_t len)
                                                 line %d of file %s\n",
                                                          v, d, s);
    unsigned char *mem = (unsigned char
                                                     (void)raise(SIGABRT);
                                                }
*)dst;
    _ARGCHK(dst != NULL);
                                                 #endif
    while (len-->0)
                                                 #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。