CamelliaKey のソースファイル

（C）宇山　靖政

2012.04.11

CamelliaKey.exe は、 Camellia 暗号ソフトの暗号化鍵を作り、そのテストもできます。

ソースファイルは、ヘッダーファイルと CPP のファイルを公開します。暗号機能について理解するには、これがあれば十分です。リソースファイルはご自由にお作りください。

このソフトウェアは、ベクターから無料でダウンロードできます。

最初は、ヘッダーファイルです。

CMLkey.h

```
/////////////////////////////////////////////////////////
// CMLkey.h : CMLkey アプリケーションのメインヘッダーファイルです。
//

#if !defined(AFX_CMLkey_H__ECE335A6_5F65_45B5_8310_4F8CC0348B80__INCLUDED_)
#define AFX_CMLkey_H__ECE335A6_5F65_45B5_8310_4F8CC0348B80__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// メインシンボル

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyApp:
// このクラスの動作の定義に関してはCMLkey.cpp ファイルを参照してください。
//
typedef unsigned char Byte;
typedef unsigned long Word;

void Camellia_Ekeygen( const int, const Byte *, Byte * );

void Camellia_Encrypt( const int, const Byte *, const Byte *, Byte * );
void Camellia_Decrypt( const int, const Byte *, const Byte *, Byte * );

class CCMLkeyApp : public CWinApp
{
public:
	CCMLkeyApp();
```

```
// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CCMLkeyApp)
        public:
        virtual BOOL InitInstance();
        //}}AFX_VIRTUAL

// インプリメンテーション

        //{{AFX_MSG(CCMLkeyApp)
                // メモ- ClassWizard はこの位置にメンバ関数を追加または削除します。
                //        この位置に生成されるコードを編集しないでください。
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_CMLCRYPT_H__ECE335A6_5F65_45B5_8310_4F8CC0348B80__INCLUDED_)
```

CMLkeyDlg.h

```
/////////////////////////////////////////////
// CMLkeyDlg.h : ヘッダーファイル
//

#if !defined(AFX_CMLkeyDLG_H__557CB8F4_1D30_461F_AADF_7C258D02B224__INCLUDED_)
#define AFX_CMLkeyDLG_H__557CB8F4_1D30_461F_AADF_7C258D02B224__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyDlg ダイアログ

class CCMLkeyDlg : public CDialog
{
// 構築
public:
        CCMLkeyDlg(CWnd* pParent = NULL); // 標準のコンストラクタ

// ダイアログデータ
```

```
        //{{AFX_DATA(CCMLkeyDlg)
        enum { IDD = IDD_CMLKEY_DIALOG };
        int madekey;
        CString keylength;
        int i_KeyLen;
        CString s_key1;
        CString s_key2;

        CString s_fname1;
        CString s_fname2;

        CString planedata_file;
        CString encryptdata_file;
        CString decryptdata_file;
        CString encryptkey_file;
        CString decryptkey_file;
        CEdit   datadisp;

    void yDisplay(const char *cp);
                // メモ: この位置にClassWizard によってデータメンバが追加されます。
        //}}AFX_DATA

        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CCMLkeyDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV のサポート
        //}}AFX_VIRTUAL

// インプリメンテーション
protected:
        HICON m_hIcon;

        // 生成されたメッセージマップ関数
        //{{AFX_MSG(CCMLkeyDlg)
        virtual void OnOK();
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        afx_msg void OnDataChange();
        afx_msg void MakeKey();
        afx_msg void TestKey();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

                // メモ: この位置にClassWizard によってデータメンバが追加されます。
        //}}AFX_DATA
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。
```

```
#endif // !defined(AFX_CMLkeyDLG_H__557CB8F4_1D30_461F_AADF_7C258D02B224__INCLUDED_)
```

Resource.h

```
//////////////////////////////////////////////////////////
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by CMLcrypt.rc
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_CMLKEY_DIALOG               102
#define IDR_MAINFRAME                   128
#define IDC_DISPLAY                     1000
#define IDC_DECRYPTDATA_FILE            1001
#define ID_MAKEKEY                      1003
#define IDC_RADIO1                      1004
#define IDC_SECRETKEY1                  1005
#define IDC_RADIO2                      1006
#define IDC_SECRETKEY2                  1007
#define ID_TESTKEY                      1008
#define IDC_PLANEDATA_FILE              1009
#define IDC_ENCRYPTDATA_FILE            1010
#define IDC_ENCRYPTKEY_FILE             1011
#define IDC_DECRYPTKEY_FILE             1012
#define IDC_RADIO3                      1013
#define IDC_RADIO6                      1016

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        129
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Stdafx.h

```
////////////////////////////////////////////////////////
// stdafx.h : 標準のシステムインクルードファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルードファイルを記述します。
//

#if !defined(AFX_STDAFX_H__0054DD18_DBC2_4FBB_83CA_D0376E18075A__INCLUDED_)
#define AFX_STDAFX_H__0054DD18_DBC2_4FBB_83CA_D0376E18075A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Windows ヘッダーから殆ど使用されないスタッフを除外します。

#include <afxwin.h>         // MFC のコアおよび標準コンポーネント
#include <afxext.h>         // MFC の拡張部分
#include <afxdisp.h>        // MFC のオートメーションクラス
#include <afxdtctl.h>           // MFC のInternet Explorer 4 コモンコントロールサポート
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC のWindows コモンコントロールサポート
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__0054DD18_DBC2_4FBB_83CA_D0376E18075A__INCLUDED_)
```

次は、cpp ファイルです。

Camellia.cpp

```
////////////////////////////////////////////////////////////////////////////
/*****************************************************
 *                                                   *
 *      Camellia Block Encryption Algorithm          *
 *        in ANSI-C Language : Camellia.c   *
 *                                                   *
 *        Version M1.02 September 24 2001   *
 *  Copyright Mitsubishi Electric Corp 2000-2001  *
 *                                                   *
 *****************************************************/
#include "stdafx.h"
```

```
typedef unsigned char Byte;
typedef unsigned long Word;

void Camellia_Ekeygen( const int, const Byte *, Byte * );

void Camellia_Encrypt( const int, const Byte *, const Byte *, Byte * );
void Camellia_Decrypt( const int, const Byte *, const Byte *, Byte * );

void Camellia_Feistel( const Byte *, const Byte *, Byte * );
void Camellia_FLlayer( Byte *, const Byte *, const Byte * );

void ByteWord( const Byte *, Word * );
void WordByte( const Word *, Byte * );
void XorBlock( const Byte *, const Byte *, Byte * );
void SwapHalf( Byte * );
void RotBlock( const Word *, const int, Word * );

const Byte SIGMA[48] = {
0xa0,0x9e,0x66,0x7f,0x3b,0xcc,0x90,0x8b,
0xb6,0x7a,0xe8,0x58,0x4c,0xaa,0x73,0xb2,
0xc6,0xef,0x37,0x2f,0xe9,0x4f,0x82,0xbe,
0x54,0xff,0x53,0xa5,0xf1,0xd3,0x6f,0x1c,
0x10,0xe5,0x27,0xfa,0xde,0x68,0x2d,0x1d,
0xb0,0x56,0x88,0xc2,0xb3,0xe6,0xc1,0xfd};

const int KSFT1[26] = {
0,64,0,64,15,79,15,79,30,94,45,109,45,124,60,124,77,13,
94,30,94,30,111,47,111,47 };
const int KIDX1[26] = {
0,0,4,4,0,0,4,4,4,4,0,0,4,0,4,4,0,0,0,0,4,4,0,0,4,4 };
const int KSFT2[34] = {
0,64,0,64,15,79,15,79,30,94,30,94,45,109,45,109,60,124,
60,124,60,124,77,13,77,13,94,30,94,30,111,47,111,47 };
const int KIDX2[34] = {
0,0,12,12,8,8,4,4,8,8,12,12,0,0,4,4,0,0,8,8,12,12,
0,0,4,4,8,8,4,4,0,0,12,12 };

const Byte SBOX[256] = {
112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
 35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
166,225, 57,202,213, 71, 93, 61,217,  1, 90,214, 81, 86,108, 77,
139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
 20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
 16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
 82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
```

```
120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
 64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158};

#define SBOX1(n) SBOX[(n)]
#define SBOX2(n) (Byte)((SBOX[(n)]>>7^SBOX[(n)]<<1)&0xff)
#define SBOX3(n) (Byte)((SBOX[(n)]>>1^SBOX[(n)]<<7)&0xff)
#define SBOX4(n) SBOX[((n)<<1^(n)>>7)&0xff]

void Camellia_Ekeygen( const int n, const Byte *k, Byte *e )
{
        Byte t[64];
        Word u[20];
        int  i;

        if( n == 128 ){
                for( i=0 ; i<16; i++ ) t[i] = k[i];
                for( i=16; i<32; i++ ) t[i] = 0;
        }
        else if( n == 192 ){
                for( i=0 ; i<24; i++ ) t[i] = k[i];
                for( i=24; i<32; i++ ) t[i] = k[i-8]^0xff;
        }
        else if( n == 256 ){
                for( i=0 ; i<32; i++ ) t[i] = k[i];
        }

        XorBlock( t+0, t+16, t+32 );

        Camellia_Feistel( t+32, SIGMA+0, t+40 );
        Camellia_Feistel( t+40, SIGMA+8, t+32 );

        XorBlock( t+32, t+0, t+32 );

        Camellia_Feistel( t+32, SIGMA+16, t+40 );
        Camellia_Feistel( t+40, SIGMA+24, t+32 );

        ByteWord( t+0,  u+0 );
        ByteWord( t+32, u+4 );

        if( n == 128 ){
                for( i=0; i<26; i+=2 ){
                        RotBlock( u+KIDX1[i+0], KSFT1[i+0], u+16 );
                        RotBlock( u+KIDX1[i+1], KSFT1[i+1], u+18 );
                        WordByte( u+16, e+i*8 );
                }
        }
        else{
                XorBlock( t+32, t+16, t+48 );

                Camellia_Feistel( t+48, SIGMA+32, t+56 );
                Camellia_Feistel( t+56, SIGMA+40, t+48 );
```

```
                ByteWord( t+16, u+8  );
                ByteWord( t+48, u+12 );

                for( i=0; i<34; i+=2 ){
                        RotBlock( u+KIDX2[i+0], KSFT2[i+0], u+16 );
                        RotBlock( u+KIDX2[i+1], KSFT2[i+1], u+18 );
                        WordByte( u+16, e+(i<<3) );
                }
        }
}

void Camellia_Encrypt( const int n, const Byte *p, const Byte *e, Byte *c )
{//n=鍵長   p=平文   e=拡張鍵   c=暗号文
        int i;

        XorBlock( p, e+0, c );

        for( i=0; i<3; i++ ){
                Camellia_Feistel( c+0, e+16+(i<<4), c+8 );
                Camellia_Feistel( c+8, e+24+(i<<4), c+0 );
        }

        Camellia_FLlayer( c, e+64, e+72 );

        for( i=0; i<3; i++ ){
                Camellia_Feistel( c+0, e+80+(i<<4), c+8 );
                Camellia_Feistel( c+8, e+88+(i<<4), c+0 );
        }

        Camellia_FLlayer( c, e+128, e+136 );

        for( i=0; i<3; i++ ){
                Camellia_Feistel( c+0, e+144+(i<<4), c+8 );
                Camellia_Feistel( c+8, e+152+(i<<4), c+0 );
        }

        if( n == 128 ){
                SwapHalf( c );
                XorBlock( c, e+192, c );
        }
        else{
                Camellia_FLlayer( c, e+192, e+200 );

                for( i=0; i<3; i++ ){
                        Camellia_Feistel( c+0, e+208+(i<<4), c+8 );
                        Camellia_Feistel( c+8, e+216+(i<<4), c+0 );
                }

                SwapHalf( c );
                XorBlock( c, e+256, c );
        }
```

```
}

void Camellia_Decrypt( const int n, const Byte *c, const Byte *e, Byte *p )
{
        int i;

        if( n == 128 ){
                XorBlock( c, e+192, p );
        }
        else{
                XorBlock( c, e+256, p );

                for( i=2; i>=0; i-- ){
                        Camellia_Feistel( p+0, e+216+(i<<4), p+8 );
                        Camellia_Feistel( p+8, e+208+(i<<4), p+0 );
                }

                Camellia_FLlayer( p, e+200, e+192 );
        }

        for( i=2; i>=0; i-- ){
                Camellia_Feistel( p+0, e+152+(i<<4), p+8 );
                Camellia_Feistel( p+8, e+144+(i<<4), p+0 );
        }

        Camellia_FLlayer( p, e+136, e+128 );

        for( i=2; i>=0; i-- ){
                Camellia_Feistel( p+0, e+88+(i<<4), p+8 );
                Camellia_Feistel( p+8, e+80+(i<<4), p+0 );
        }

        Camellia_FLlayer( p, e+72, e+64 );

        for( i=2; i>=0; i-- ){
                Camellia_Feistel( p+0, e+24+(i<<4), p+8 );
                Camellia_Feistel( p+8, e+16+(i<<4), p+0 );
        }

        SwapHalf( p );
        XorBlock( p, e+0, p );
}

void Camellia_Feistel( const Byte *x, const Byte *k, Byte *y )
{
        Byte t[8];

        t[0] = SBOX1(x[0]^k[0]);
        t[1] = SBOX2(x[1]^k[1]);
        t[2] = SBOX3(x[2]^k[2]);
        t[3] = SBOX4(x[3]^k[3]);
        t[4] = SBOX2(x[4]^k[4]);
```

```
        t[5] = SBOX3(x[5]^k[5]);
        t[6] = SBOX4(x[6]^k[6]);
        t[7] = SBOX1(x[7]^k[7]);

        y[0] ^= t[0]^t[2]^t[3]^t[5]^t[6]^t[7];
        y[1] ^= t[0]^t[1]^t[3]^t[4]^t[6]^t[7];
        y[2] ^= t[0]^t[1]^t[2]^t[4]^t[5]^t[7];
        y[3] ^= t[1]^t[2]^t[3]^t[4]^t[5]^t[6];
        y[4] ^= t[0]^t[1]^t[5]^t[6]^t[7];
        y[5] ^= t[1]^t[2]^t[4]^t[6]^t[7];
        y[6] ^= t[2]^t[3]^t[4]^t[5]^t[7];
        y[7] ^= t[0]^t[3]^t[4]^t[5]^t[6];
}

void Camellia_FLlayer( Byte *x, const Byte *kl, const Byte *kr )
{
        Word t[4],u[4],v[4];

        ByteWord( x, t );
        ByteWord( kl, u );
        ByteWord( kr, v );

        t[1] ^= (t[0]&u[0])<<1^(t[0]&u[0])>>31;
        t[0] ^= t[1]|u[1];
        t[2] ^= t[3]|v[1];
        t[3] ^= (t[2]&v[0])<<1^(t[2]&v[0])>>31;

        WordByte( t, x );
}

void ByteWord( const Byte *x, Word *y )
{
        int i;
        for( i=0; i<4; i++ ){
                y[i] = ((Word)x[(i<<2)+0]<<24) + ((Word)x[(i<<2)+1]<<16)
                     + ((Word)x[(i<<2)+2]<<8 ) + ((Word)x[(i<<2)+3]<<0 );
        }
}

void WordByte( const Word *x, Byte *y )
{
        int i;
        for( i=0; i<4; i++ ){
                y[(i<<2)+0] = (Byte)(x[i]>>24&0xff);
                y[(i<<2)+1] = (Byte)(x[i]>>16&0xff);
                y[(i<<2)+2] = (Byte)(x[i]>> 8&0xff);
                y[(i<<2)+3] = (Byte)(x[i]>> 0&0xff);
        }
}

void RotBlock( const Word *x, const int n, Word *y )
{
```

```
        int r;
        if( r = (n & 31) ){
                y[0] = x[((n>>5)+0)&3]<<r^x[((n>>5)+1)&3]>>(32-r);
                y[1] = x[((n>>5)+1)&3]<<r^x[((n>>5)+2)&3]>>(32-r);
        }
        else{
                y[0] = x[((n>>5)+0)&3];
                y[1] = x[((n>>5)+1)&3];
        }
}

void SwapHalf( Byte *x )
{
        Byte t;
        int  i;
        for( i=0; i<8; i++ ){
                t = x[i];
                x[i] = x[8+i];
                x[8+i] = t;
        }
}

void XorBlock( const Byte *x, const Byte *y, Byte *z )
{
        int i;
        for( i=0; i<16; i++ ) z[i] = x[i] ^ y[i];
}
```

CMLkey.cpp

```
////////////////////////////////////////////////////
// CMLkey.cpp : アプリケーション用クラスの定義を行います。
//

#include "stdafx.h"
#include "CMLkey.h"
#include "CMLkeyDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyApp
```

```
BEGIN_MESSAGE_MAP(CCMLkeyApp, CWinApp)
	//{{AFX_MSG_MAP(CCMLkeyApp)
		// メモ- ClassWizard はこの位置にマッピング用のマクロを追加または削除します。
		//		この位置に生成されるコードを編集しないでください。
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyApp クラスの構築

CCMLkeyApp::CCMLkeyApp()
{
	// TODO: この位置に構築用のコードを追加してください。
	// ここにInitInstance 中の重要な初期化処理をすべて記述してください。
}

/////////////////////////////////////////////////////////////////////////////
// 唯一のCCMLkeyApp オブジェクト

CCMLkeyApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyApp クラスの初期化

BOOL CCMLkeyApp::InitInstance()
{
	AfxEnableControlContainer();

	// 標準的な初期化処理
	// もしこれらの機能を使用せず、実行ファイルのサイズを小さくしたけ
	//  れば以下の特定の初期化ルーチンの中から不必要なものを削除して
	//  ください。

#ifdef _AFXDLL
	Enable3dControls();			// 共有DLL 内でMFC を使う場合はここをコールしてくだ
さい。
#else
	Enable3dControlsStatic();// MFC と静的にリンクする場合はここをコールしてください。
#endif

	CCMLkeyDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: ダイアログが<OK> で消された時のコードを
		//	記述してください。
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: ダイアログが<ｷｬﾝｾﾙ> で消された時のコードを
```

```
		//		記述してください。
	}

	// ダイアログが閉じられてからアプリケーションのメッセージポンプを開始するよりは、
	// アプリケーションを終了するためにFALSE を返してください。
	return FALSE;
}
```

CMLkeyDlg.cpp

```
////////////////////////////////////////////////////////////
// CMLkeyDlg.cpp : インプリメンテーションファイル
//

#include "stdafx.h"
#include "CMLkey.h"
#include "CMLkeyDlg.h"
#include "fstream"
#include <iostream>

using namespace std;

FILE *stream0;//plane
FILE *stream1;//encrypt
FILE *stream2;//decrypt
FILE *stream3;//eckey
FILE *stream4;//dckey

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

unsigned char k[128];
unsigned char e[256];
//typedef unsigned long Word;


// 暗号文のHEX表示用
char toChar( int c )
{
	if( c >= 0 && c <= 9 )					// 0～ならば
		return char( c + 0x30 ); // ASCIIに変換して返す
	else if( c >= 10 && c <= 15 )	// 10～ならば
```

```
                return char( c + 0x37 ); // A～FのASCIIを返す
        else
                return ' ';
}


/////////////////////////////////////////////////////////////////////////////
// アプリケーションのバージョン情報で使われているCAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// ダイアログデータ
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard は仮想関数のオーバーライドを生成します
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV のサポート
        //}}AFX_VIRTUAL

// インプリメンテーション
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // メッセージハンドラがありません。
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
```

```
// CCMLkeyDlg ダイアログ

CCMLkeyDlg::CCMLkeyDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CCMLkeyDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCMLkeyDlg)
	i_KeyLen = 0;
	s_fname1 = "CMLkeyEC1.bin";
	s_fname2 = "CMLkeyDC1.bin";
	srand( (unsigned)time(NULL) );
	madekey = 0;
	planedata_file = "plane.txt";
	encryptdata_file = "encrypt.enc";
	decryptdata_file = "decrypt.txt";
		// メモ: この位置にClassWizard によってメンバの初期化が追加されます。
	//}}AFX_DATA_INIT
	// メモ: LoadIcon はWin32 のDestroyIcon のサブシーケンスを要求しません。
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CCMLkeyDlg::OnOK()
{
	CDialog::OnOK();

	ofstream ofs(s_fname1, ios::out);
	if(i_KeyLen == 0) ofs << 128 << endl;
	if(i_KeyLen == 1) ofs << 192 << endl;
	if(i_KeyLen == 2) ofs << 256 << endl;
	ofs << s_key1 << endl;
	ofs.close();

	ofstream ofs2(s_fname2, ios::out);
	if(i_KeyLen == 0) ofs2 << 128 << endl;
	if(i_KeyLen == 1) ofs2 << 192 << endl;
	if(i_KeyLen == 2) ofs2 << 256 << endl;
	ofs2 << s_key2 << endl;
	ofs2.close();
}

void CCMLkeyDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCMLkeyDlg)
	DDX_Radio(pDX, IDC_RADIO1, i_KeyLen);
	DDX_Text(pDX, IDC_SECRETKEY1, s_key1);
	DDX_Text(pDX, IDC_SECRETKEY2, s_key2);
	DDX_Text(pDX, IDC_ENCRYPTKEY_FILE, s_fname1);
	DDX_Text(pDX, IDC_DECRYPTKEY_FILE, s_fname2);
	DDX_Text(pDX, IDC_PLANEDATA_FILE, planedata_file);
    DDV_MaxChars(pDX, planedata_file, 128);
    DDX_Text(pDX, IDC_ENCRYPTDATA_FILE, encryptdata_file);
    DDV_MaxChars(pDX, encryptdata_file, 128);
```

```
    DDX_Text(pDX, IDC_DECRYPTDATA_FILE, decryptdata_file);
    DDV_MaxChars(pDX, decryptdata_file, 128);
        DDX_Control(pDX, IDC_DISPLAY, datadisp);
                // メモ: この場所にはClassWizard によってDDX とDDV の呼び出しが追加されます。
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CCMLkeyDlg, CDialog)
        //{{AFX_MSG_MAP(CCMLkeyDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_COMMAND(ID_MAKEKEY, MakeKey)
        ON_COMMAND(ID_TESTKEY, TestKey)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CCMLkeyDlg::yDisplay(const char *cp)
{
  int nEditLength = datadisp.GetWindowTextLength();
  datadisp.SetSel(nEditLength, nEditLength); //テキストの終端にカーソルを移動する
  datadisp.ReplaceSel(cp); //新しいテキストを追加する
}

void CCMLkeyDlg::MakeKey()
{
        int i, n;
        char j , k;
        char a[64];
        char b[128];
        for(i=0; i<128; i++){ b[i] = 0;}

        if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 16;}
        if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 24;}
        if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 32;}

        for(i=0; i<n ; i++){
                *(a+i) = (char)rand();
        }
        a[n] = NULL;

        for(i=0; i<n; i++){
                j = *(a+i);
                k = (j>>4)&0x0f;
                if(k>=0 && k<=9) b[2*i] = k + 0x30;
                else if(k>=10 && k<=15) b[2*i] = k + 0x37;
                k = j & 0x0f;
                if(k>=0 && k<=9) b[2*i+1] = k + 0x30;
                else if(k>=10 && k<=15) b[2*i+1] = k + 0x37;
        }
        b[2*n] = NULL;
```

```
        s_key1 = b;
        s_key2 = b;

        CWnd * pwnd = GetDlgItem(IDC_SECRETKEY1);
        pwnd->SetWindowText(b);
        pwnd = GetDlgItem(IDC_SECRETKEY2);
        pwnd->SetWindowText(b);
        madekey = 1;
}

//int/*long*/ mesLength; // 平文長（バイト）

void CCMLkeyDlg::TestKey()
{
        unsigned char key2[128+2],key[64+2];
        unsigned char exkey[512];
        int  cnt, c, block, i, j;
        char cj, ck;
        int filelen, mesLength; // 平文長（バイト）
        char* bufp;
        unsigned char* bufc;
        char* bufdc;
        int numclosed;
        int n;
        CWnd* pwnd;
        CFileStatus fs;

        datadisp.SetLimitText(INT_MAX);

        if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 16; keylength = "128"; }
        if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 24; keylength = "192"; }
        if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) ){ n= 32; keylength = "256"; }
   /* 読み出すファイルを開く
    *（ファイルが存在しないときは、呼び出しが失敗）
    */
        pwnd = GetDlgItem(IDC_PLANEDATA_FILE);
        pwnd->GetWindowText(planedata_file);
        if( (stream0  = fopen( planedata_file, "rb" )) == NULL ){
                yDisplay("ファイル"); yDisplay(planedata_file); yDisplay(" は開けませんでした。¥r¥n");
                return;//(-1);
                }
   else
                {yDisplay("ファイル"); yDisplay(planedata_file); yDisplay(" は開けました。¥r¥n");}

   /* 暗号文を書き込むファイルを開く*/
        pwnd = GetDlgItem(IDC_ENCRYPTDATA_FILE);
        pwnd->GetWindowText(encryptdata_file);
        if( (stream1 = fopen( encryptdata_file, "w+b" )) == NULL ){
                yDisplay("ファイル"); yDisplay(encryptdata_file); yDisplay(" は開けませんでした。
¥r¥n");
                return;//(-1);
                }
```

```
    else
                {yDisplay("ファイル"); yDisplay(encryptdata_file); yDisplay(" は開けました。¥r¥n");}

    /* 復号文を書き込むファイルを開く*/
          pwnd = GetDlgItem(IDC_DECRYPTDATA_FILE);
          pwnd->GetWindowText(decryptdata_file);
          if( (stream2 = fopen( decryptdata_file, "w+b" )) == NULL ){
                  yDisplay("ファイル"); yDisplay(decryptdata_file); yDisplay(" は開けませんでした。
¥r¥n");
                  return;//(-1);
                  }
          else
                  {yDisplay("ファイル"); yDisplay(decryptdata_file); yDisplay(" は開けました。¥r¥n");}


////////////////////////////////////////////////////////////////
          if(madekey == 0){                  // 鍵の生成
                  yDisplay("¥r¥n");
                  yDisplay("鍵を生成中¥r¥n");
                  yDisplay("鍵長     = "); yDisplay(keylength); yDisplay(" bit");
                  yDisplay("¥r¥n");
                  yDisplay("¥r¥n");
         MakeKey();
          }

          pwnd = GetDlgItem(IDC_SECRETKEY1);
          pwnd->GetWindowText(s_key1);
          strcpy((char*)key2, s_key1);

          for(i=0; i<n; i++){
                  cj = *(key2+2*i);
                  ck = *(key2+2*i+1);
                  if(cj>=0x30 && cj<=0x39) cj = cj-0x30;
                  else{
                          if(cj>=0x41 && cj<=0x46) cj = cj-0x41+0x0A;
                  }
                  if(ck>=0x30 && ck<=0x39) ck = ck-0x30;
                  else{
                          if(ck>=0x41 && ck<=0x46) ck = ck-0x41+0x0A;
                  }
                  key[i] = cj*0x10 + ck;
          }
          key[n] = NULL;

          Camellia_Ekeygen( n*8, key, exkey );

     // 平文
          CFile::GetStatus(planedata_file, fs);
          filelen = fs.m_size;
          mesLength = filelen + sizeof(long);
          block = mesLength/16 + ((mesLength%16)?1:0);
          bufp = (char*)new(char[block*16 + 1]);
```

```
	if(bufp == NULL){
		yDisplay("メモリ不足¥r¥n");
		return;//(-1);
		}
	for(j=0; j<(block*16 + 1); j++){
		bufp[j] = 0;
	}
	*(long*)bufp = filelen;

	bufc = (unsigned char*)new(unsigned char[block*16 + 1]);
	if(bufc == NULL){
		yDisplay("メモリ不足¥r¥n");
		return;//(-1);
		}
	for(j=0; j<(block*16 + 1); j++){
		bufc[j] = 0;
	}


// 平文
	fseek(stream0,0,0);
	i = sizeof(long);
	do{
		c = fgetc(stream0);
		bufp[i]=c;
		i=i+1;
	}while(c!=EOF);
	bufp[i-1]=NULL;


	mesLength = i-1; // 平文長（バイト）

	char  *sd;
	CString  Ssd;

	sd = (char*)new(char[256]);
	pwnd = GetDlgItem(IDC_SECRETKEY1);
	pwnd->GetWindowText(Ssd);
	strcpy(sd, Ssd);

	yDisplay("秘密鍵  = "); yDisplay(sd); yDisplay("¥r¥n");
	yDisplay("¥r¥n");

	char buff[16];
	itoa(mesLength, buff, 10);
  yDisplay("平文長  = "); yDisplay(buff); yDisplay(" byte¥r¥n");
	yDisplay("¥r¥n");
	yDisplay("¥r¥n");

  yDisplay("平文    = ¥r¥n");
	yDisplay(bufp+sizeof(long));
	yDisplay("¥r¥n");
```

```
        yDisplay("¥r¥n");


        for(i=0;i<block;i++){
            Camellia_Encrypt( n*8,  (unsigned char*)(bufp+i*16), (unsigned char*)exkey, bufc+i*16);
                    //n=鍵長   bufp=平文   exkey=拡張鍵   ct=暗号文
        }

        // 暗号文を進数で表示 0xa4  は  A4  と表示される
    yDisplay("暗号文(HEX)= ");
        for( cnt = 0; cnt<(block*16); cnt++ ){
                char c1, c2;
                if(cnt%30 ==0){yDisplay("¥r¥n");}

                c1 = toChar((int(bufc[cnt]) >> 4) & 0xf);
                c2 = toChar(int(bufc[cnt]) & 0xf);
                CString sc = c1;
                sc += c2;
                yDisplay(sc);
                fwrite( &(bufc[cnt]),sizeof(char),1, stream1);
        }
        yDisplay("¥r¥n");
        yDisplay("¥r¥n");

        if( fclose( stream0 ) )
      MessageBox( "ファイル'p-data' は閉じられませんでした。¥n" );

        if( fclose( stream1 ) )
      MessageBox( "ファイル'c-data' は閉じられませんでした。¥n" );

   /* 暗号化したデータのファイルを読み込むために開く*/
        if( (stream1 = fopen( encryptdata_file, "rb" )) == NULL ){
                MessageBox( "暗文ファイルは開けませんでした。¥n" );
                numclosed = _fcloseall( );
                return;//(-1);
        }

        delete[] bufc;

////////////////////////////////////
        CFile::GetStatus(encryptdata_file, fs);
        filelen = fs.m_size;

        bufc = (unsigned char*)new(unsigned char[filelen + 2]);
        if(bufc == 0){
                yDisplay("メモリ不足¥r¥n");
                return;//(-1);
                }
        fseek(stream1,0,0);
        int cc;
        i=0;
        do{
```

```
                cc = fgetc(stream1);
                bufc[i]=cc;
                i=i+1;
        }while(cc!=EOF);
        bufc[i-1]=NULL;

        mesLength = filelen;
        block = mesLength/16 + ((mesLength%16)?1:0);
        bufdc = (char*)new(char[block*16 + 1]);

        for(i=0;i<block;i++){
                Camellia_Decrypt(  n*8, (unsigned char*)(bufc+i*16), exkey, (unsigned
char*)(bufdc+i*16) );
        }

        // 復号文出力
        filelen = *((long*)bufdc);
        for( int ont = sizeof(long); ont<(int)(filelen+sizeof(long)); ont++ ){
                fwrite( &(bufdc[ont]),sizeof(char),1, stream2);
        }
        bufdc[filelen+sizeof(long)] = '¥0';

    yDisplay("復号文  = ¥r¥n");
        yDisplay(bufdc+sizeof(long));
        yDisplay("¥r¥n");

        if( fclose( stream2 ) )
      MessageBox( "復号化ファイルは閉じられませんでした。¥n" );


   /* 他のすべてのファイルを閉じる*/
        numclosed = _fcloseall( );

        delete[] bufp;
        delete[] bufdc;

        return;// 0;
}

/////////////////////////////////////////////////////////////////////////////
// CCMLkeyDlg メッセージハンドラ

BOOL CCMLkeyDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // "バージョン情報..." メニュー項目をシステムメニューへ追加します。

        // IDM_ABOUTBOX はコマンドメニューの範囲でなければなりません。
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);
```

```
	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// このダイアログ用のアイコンを設定します。フレームワークはアプリケーションのメイン
	// ウィンドウがダイアログでない時は自動的に設定しません。
	SetIcon(m_hIcon, TRUE);			// 大きいアイコンを設定
	SetIcon(m_hIcon, FALSE);		// 小さいアイコンを設定

	// TODO: 特別な初期化を行う時はこの場所に追加してください。

	return TRUE;  // TRUE を返すとコントロールに設定したフォーカスは失われません。
}

void CCMLkeyDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// もしダイアログボックスに最小化ボタンを追加するならば、アイコンを描画する
// コードを以下に記述する必要があります。MFC アプリケーションはdocument/view
// モデルを使っているので、この処理はフレームワークにより自動的に処理されます。

void CCMLkeyDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // 描画用のデバイスコンテキスト

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// クライアントの矩形領域内の中央
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
```

```
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // アイコンを描画します。
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// システムは、ユーザーが最小化ウィンドウをドラッグしている間、
// カーソルを表示するためにここを呼び出します。
HCURSOR CCMLkeyDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}
```

Stdafx.cpp

```
////////////////////////////////////////////////////////////
// stdafx.cpp : 標準インクルードファイルを含むソースファイル
//              CMLcrypt.pch : 生成されるプリコンパイル済ヘッダー
//              stdafx.obj : 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"
```

おわり。