SerpentKey のソースファイル

（C）宇山　靖政

2012.04.11

SerpentKey.exe　は、Serpent 暗号の暗号鍵作成とそのテストが可能です。

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。

Resource.h

```
//////////////////////
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by SerpentKey.rc
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_SERPENTCRYPT_DIALOG         102
#define IDR_MAINFRAME                   128
#define IDD_SERPENTKEY_DIALOG           129
#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_RADIO4                      1014
#define IDC_RADIO5                      1015
#define IDC_RADIO6                      1016
#define IDC_RADIO7                      1017
#define IDC_CIPHERINIT                  1018
#define IDC_CHGCI                       1019

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
```

```
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        130
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1020
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Serpent.h

```
//////////////////////////////
/*  aes.h  */

/*  AES Cipher header file for ANSI C Submissions
      Lawrence E. Bassham III
      Computer Security Division
      National Institute of Standards and Technology

      April 15, 1998

    This sample is to assist implementers developing to the Cryptographic
API Profile for AES Candidate Algorithm Submissions.  Please consult this
document as a cross-reference.

    ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS FILE
MUST BE DOCUMENTED.  CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED WITH
THE STRING "CHANGE POSSIBLE".  FUNCTION CALLS AND THEIR PARAMETERS CANNOT
BE CHANGED.  STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS TO INCLUDE
IMPLEMENTATION SPECIFIC INFORMATION.
*/

/*  Includes:
        Standard include files
*/

#include <stdio.h>

/*  Defines:
        Add any additional defines you need
*/

#define     DIR_ENCRYPT     0    /*  Are we encrpyting?  */
#define     DIR_DECRYPT     1    /*  Are we decrpyting?  */
#define     MODE_ECB        1    /*  Are we ciphering in ECB mode?   */
#define     MODE_CBC        2    /*  Are we ciphering in CBC mode?   */
#define     MODE_CFB1       3    /*  Are we ciphering in 1-bit CFB mode? */
#define     TRUE            1
#define     FALSE           0
```

```
/*  Error Codes - CHANGE POSSIBLE: inclusion of additional error codes  */
#define     BAD_KEY_DIR        -1  /*  Key direction is invalid, e;g;,
                                       unknown value */
#define     BAD_KEY_MAT        -2  /*  Key material not of correct
                                       length */
#define     BAD_KEY_INSTANCE   -3  /*  Key passed is not valid  */
#define     BAD_CIPHER_MODE    -4  /*  Params struct passed to
                                       cipherInit invalid */
#define     BAD_CIPHER_STATE   -5  /*  Cipher in wrong state (e.g., not
                                       initialized) */

/*  CHANGE POSSIBLE:  inclusion of algorithm specific defines  */
#define     MAX_KEY_SIZE 64  /* # of ASCII char's needed to
                                       represent a key */
#define     MAX_IV_SIZE          32  /* # of ASCII char's needed to
                                       represent an IV  */

/*  Typedefs:

        Typedef'ed data storage elements.  Add any algorithm specific
parameters at the bottom of the structs as appropriate.
*/

typedef    unsigned char    BYTE;

/*  The structure for key information */
typedef struct {
      BYTE  direction;   /*  Key used for encrypting or decrypting? */
      int   keyLen;      /*  Length of the key  */
      char  keyMaterial[MAX_KEY_SIZE+1];  /*  Raw key data in ASCII, e.g.,
                                       what the user types or KAT values)*/
      /*  The following parameters are algorithm dependent, replace or
                add as necessary  */
      unsigned long key[8];             /* The key in binary */
      unsigned long subkeys[33][4];       /* Serpent subkeys */
      } keyInstance;

/*  The structure for cipher information */
typedef struct {
      BYTE       mode;           /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
      char  IV[MAX_IV_SIZE];      /* A possible Initialization Vector for
                                       ciphering */
      /*  Add any algorithm specific parameters needed here  */
      int   blockSize;            /* Sample: Handles non-128 bit block sizes
                                       (if available) */
      } cipherInstance;


/*  Function protoypes  */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
            char *keyMaterial);
```

```
int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer);

int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer);

int serpent_convert_from_string(int, char *, unsigned long *);
void serpent_encrypt(unsigned long [],unsigned long [], unsigned long [][4]);
void serpent_decrypt(unsigned long [],unsigned long [], unsigned long [][4]);
```

Serpentkey.h

```
////////////////////////////
// SerpentKey.h : SERPENTKEY 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_SERPENTKEY_H__96FBC5E7_D67A_4056_8856_3419D423CF35__INCLUDED_)
#define AFX_SERPENTKEY_H__96FBC5E7_D67A_4056_8856_3419D423CF35__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

/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyApp:
// このクラスの動作の定義に関してはSerpentKey.cpp ファイルを参照してください。
//

class CSerpentKeyApp : public CWinApp
{
public:
        CSerpentKeyApp();

// 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(CSerpentKeyApp)
        public:
        virtual BOOL InitInstance();
```

```
        //}}AFX_VIRTUAL

// インプリメンテーション

        //{{AFX_MSG(CSerpentKeyApp)
                // 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_SERPENTKEY_H__96FBC5E7_D67A_4056_8856_3419D423CF35__INCLUDED_)
```

Serpentkeydlg.h

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

#if !defined(AFX_SERPENTKEYDLG_H__E53522C5_7B65_4B30_8B56_E53A5ACA862A__INCLUDED_)
#define AFX_SERPENTKEYDLG_H__E53522C5_7B65_4B30_8B56_E53A5ACA862A__INCLUDED_

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

/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyDlg ダイアログ
#include "serpent.h"

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

// ダイアログデータ
        //{{AFX_DATA(CSerpentKeyDlg)
        enum { IDD = IDD_SERPENTKEY_DIALOG };
        int madekey;
        CString keylength;
        int i_KeyLen;
```

```
        int i_Mode;
        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;

        CString s_cipherinit;
        CEdit   datadisp;

    void yDisplay(const char *cp);

        keyInstance keyI;
        cipherInstance cipherI;
    int rc;
    unsigned long x[4];
                // 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(CSerpentKeyDlg)
        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(CSerpentKeyDlg)
        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 void ChgCI();
        //}}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_SERPENTKEYDLG_H__E53522C5_7B65_4B30_8B56_E53A5ACA862A__INCLUDED_)
```

Serpentsboxes.h

```
////////////////////////////////
/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *  i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */
```

```
/* S0:  3  8 15  1 10  6  5 11 14 13  4  2  7  0  9 12 */

/* depth = 5,7,4,2, Total gates=18 */
#define RND00(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t11, t12, t13, t14, t15, t17, t01;¥
        t01 = b   ^ c  ; ¥
        t02 = a   | d  ; ¥
        t03 = a   ^ b  ; ¥
        z   = t02 ^ t01; ¥
        t05 = c   | z  ; ¥
        t06 = a   ^ d  ; ¥
        t07 = b   | c  ; ¥
        t08 = d   & t05; ¥
        t09 = t03 & t07; ¥
        y   = t09 ^ t08; ¥
        t11 = t09 & y  ; ¥
        t12 = c   ^ d  ; ¥
        t13 = t07 ^ t11; ¥
        t14 = b   & t06; ¥
        t15 = t06 ^ t13; ¥
        w   =     ~ t15; ¥
        t17 = w   ^ t14; ¥
        x   = t12 ^ t17; }

/* InvS0:  13  3 11  0 10  6  5 12  1 14  4  7 15  9  8  2 */

/* depth = 8,4,3,6, Total gates=19 */
#define InvRND00(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t12, t13, t14, t15, t17, t18,
t01;¥
        t01 = c   ^ d  ; ¥
        t02 = a   | b  ; ¥
        t03 = b   | c  ; ¥
        t04 = c   & t01; ¥
        t05 = t02 ^ t01; ¥
        t06 = a   | t04; ¥
        y   =     ~ t05; ¥
        t08 = b   ^ d  ; ¥
        t09 = t03 & t08; ¥
        t10 = d   | y  ; ¥
        x   = t09 ^ t06; ¥
        t12 = a   | t05; ¥
        t13 = x   ^ t12; ¥
        t14 = t03 ^ t10; ¥
        t15 = a   ^ c  ; ¥
        z   = t14 ^ t13; ¥
        t17 = t05 & t13; ¥
        t18 = t14 | t17; ¥
        w   = t15 ^ t18; }

/* S1:  15 12  2  7  9  0  5 10  1 11 14  8  6 13  3  4 */
```

```
/* depth = 10,7,3,5, Total gates=18 */
#define RND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t10, t11, t12, t13, t16, t17, t01;¥
        t01 = a   | d  ; ¥
        t02 = c   ^ d  ; ¥
        t03 =     ~ b  ; ¥
        t04 = a   ^ c  ; ¥
        t05 = a   | t03; ¥
        t06 = d   & t04; ¥
        t07 = t01 & t02; ¥
        t08 = b   | t06; ¥
        y   = t02 ^ t05; ¥
        t10 = t07 ^ t08; ¥
        t11 = t01 ^ t10; ¥
        t12 = y   ^ t11; ¥
        t13 = b   & d  ; ¥
        z   =     ~ t10; ¥
        x   = t13 ^ t12; ¥
        t16 = t10 | x  ; ¥
        t17 = t05 & t16; ¥
        w   = c   ^ t17; }

/* InvS1:   5  8  2 14 15  6 12  3 11  4  7  9  1 13 10  0 */

/* depth = 7,4,5,3, Total gates=18 */
#define InvRND01(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t14, t15, t17, t01;¥
        t01 = a   ^ b  ; ¥
        t02 = b   | d  ; ¥
        t03 = a   & c  ; ¥
        t04 = c   ^ t02; ¥
        t05 = a   | t04; ¥
        t06 = t01 & t05; ¥
        t07 = d   | t03; ¥
        t08 = b   ^ t06; ¥
        t09 = t07 ^ t06; ¥
        t10 = t04 | t03; ¥
        t11 = d   & t08; ¥
        y   =     ~ t09; ¥
        x   = t10 ^ t11; ¥
        t14 = a   | y  ; ¥
        t15 = t06 ^ x  ; ¥
        z   = t01 ^ t04; ¥
        t17 = c   ^ t15; ¥
        w   = t14 ^ t17; }

/* S2:   8  6  7  9  3 12 10 15 13  1 14  4  0 11  5  2 */

/* depth = 3,8,11,7, Total gates=16 */
#define RND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t05, t06, t07, t08, t09, t10, t12, t13, t14, t01;¥
        t01 = a   | c  ; ¥
```

```
        t02 = a   ^ b  ; ¥
        t03 = d   ^ t01; ¥
        w   = t02 ^ t03; ¥
        t05 = c   ^ w  ; ¥
        t06 = b   ^ t05; ¥
        t07 = b   | t05; ¥
        t08 = t01 & t06; ¥
        t09 = t03 ^ t07; ¥
        t10 = t02 | t09; ¥
        x   = t10 ^ t08; ¥
        t12 = a   | d  ; ¥
        t13 = t09 ^ x  ; ¥
        t14 = b   ^ t13; ¥
        z   =     ~ t09; ¥
        y   = t12 ^ t14; }

/* InvS2:  12  9 15  4 11 14  1  2  0  3  6 13  5  8 10  7 */

/* depth = 3,6,8,3, Total gates=18 */
#define InvRND02(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t12, t15, t16, t17, t01;¥
        t01 = a   ^ d  ; ¥
        t02 = c   ^ d  ; ¥
        t03 = a   & c  ; ¥
        t04 = b   | t02; ¥
        w   = t01 ^ t04; ¥
        t06 = a   | c  ; ¥
        t07 = d   | w  ; ¥
        t08 =     ~ d  ; ¥
        t09 = b   & t06; ¥
        t10 = t08 | t03; ¥
        t11 = b   & t07; ¥
        t12 = t06 & t02; ¥
        z   = t09 ^ t10; ¥
        x   = t12 ^ t11; ¥
        t15 = c   & z  ; ¥
        t16 = w   ^ x  ; ¥
        t17 = t10 ^ t15; ¥
        y   = t16 ^ t17; }

/* S3:   0 15 11  8 12  9  6  3 13  1  2  4 10  7  5 14 */

/* depth = 8,3,5,5, Total gates=18 */
#define RND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t13, t14, t15, t01;¥
        t01 = a   ^ c  ; ¥
        t02 = a   | d  ; ¥
        t03 = a   & d  ; ¥
        t04 = t01 & t02; ¥
        t05 = b   | t03; ¥
        t06 = a   & b  ; ¥
        t07 = d   ^ t04; ¥
```

```
        t08 = c   | t06; ¥
        t09 = b   ^ t07; ¥
        t10 = d   & t05; ¥
        t11 = t02 ^ t10; ¥
        z   = t08 ^ t09; ¥
        t13 = d   | z  ; ¥
        t14 = a   | t07; ¥
        t15 = b   & t13; ¥
        y   = t08 ^ t11; ¥
        w   = t14 ^ t15; ¥
        x   = t05 ^ t04; }

/* InvS3:   0  9 10  7 11 14  6 13  3  5 12  2  4  8 15  1 */

/* depth = 3,6,4,4, Total gates=17 */
#define InvRND03(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t11, t12, t13, t14, t16, t01;¥
        t01 = c   | d  ; ¥
        t02 = a   | d  ; ¥
        t03 = c   ^ t02; ¥
        t04 = b   ^ t02; ¥
        t05 = a   ^ d  ; ¥
        t06 = t04 & t03; ¥
        t07 = b   & t01; ¥
        y   = t05 ^ t06; ¥
        t09 = a   ^ t03; ¥
        w   = t07 ^ t03; ¥
        t11 = w   | t05; ¥
        t12 = t09 & t11; ¥
        t13 = a   & y  ; ¥
        t14 = t01 ^ t05; ¥
        x   = b   ^ t12; ¥
        t16 = b   | t13; ¥
        z   = t14 ^ t16; }

/* S4:   1 15  8  3 12  0 11  6  2  5  4 10  9 14  7 13 */

/* depth = 6,7,5,3, Total gates=19 */
#define RND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t12, t13, t14, t15, t16,
t01;¥
        t01 = a   | b  ; ¥
        t02 = b   | c  ; ¥
        t03 = a   ^ t02; ¥
        t04 = b   ^ d  ; ¥
        t05 = d   | t03; ¥
        t06 = d   & t01; ¥
        z   = t03 ^ t06; ¥
        t08 = z   & t04; ¥
        t09 = t04 & t05; ¥
        t10 = c   ^ t06; ¥
        t11 = b   & c  ; ¥
```

```
        t12 = t04 ^ t08; ¥
        t13 = t11 | t03; ¥
        t14 = t10 ^ t09; ¥
        t15 = a   & t05; ¥
        t16 = t11 | t12; ¥
        y   = t13 ^ t08; ¥
        x   = t15 ^ t16; ¥
        w   =     ~ t14; }

/* InvS4:   5  0  8  3 10  9  7 14  2 12 11  6  4 15 13  1 */

/* depth = 6,4,7,3, Total gates=17 */
#define InvRND04(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t09, t10, t11, t12, t13, t15, t01;¥
        t01 = b   | d  ; ¥
        t02 = c   | d  ; ¥
        t03 = a   & t01; ¥
        t04 = b   ^ t02; ¥
        t05 = c   ^ d  ; ¥
        t06 =     ~ t03; ¥
        t07 = a   & t04; ¥
        x   = t05 ^ t07; ¥
        t09 = x   | t06; ¥
        t10 = a   ^ t07; ¥
        t11 = t01 ^ t09; ¥
        t12 = d   ^ t04; ¥
        t13 = c   | t10; ¥
        z   = t03 ^ t12; ¥
        t15 = a   ^ t04; ¥
        y   = t11 ^ t13; ¥
        w   = t15 ^ t09; }

/* S5:  15  5  2 11  4 10  9 12  0  3 14  8 13  6  7  1 */

/* depth = 4,6,8,6, Total gates=17 */
#define RND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t14, t01;¥
        t01 = b   ^ d  ; ¥
        t02 = b   | d  ; ¥
        t03 = a   & t01; ¥
        t04 = c   ^ t02; ¥
        t05 = t03 ^ t04; ¥
        w   =     ~ t05; ¥
        t07 = a   ^ t01; ¥
        t08 = d   | w  ; ¥
        t09 = b   | t05; ¥
        t10 = d   ^ t08; ¥
        t11 = b   | t07; ¥
        t12 = t03 | w  ; ¥
        t13 = t07 | t10; ¥
        t14 = t01 ^ t11; ¥
        y   = t09 ^ t13; ¥
```

```
        x   = t07 ^ t08; ¥
        z   = t12 ^ t14; }

/* InvS5:   8 15  2  9  4  1 13 14 11  6  5  3  7 12 10  0 */

/* depth = 4,6,9,7, Total gates=17 */
#define InvRND05(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t12, t13, t15, t16, t01;¥
        t01 = a   & d  ; ¥
        t02 = c   ^ t01; ¥
        t03 = a   ^ d  ; ¥
        t04 = b   & t02; ¥
        t05 = a   & c  ; ¥
        w   = t03 ^ t04; ¥
        t07 = a   & w  ; ¥
        t08 = t01 ^ w  ; ¥
        t09 = b   | t05; ¥
        t10 =     ~ b  ; ¥
        x   = t08 ^ t09; ¥
        t12 = t10 | t07; ¥
        t13 = w   | x  ; ¥
        z   = t02 ^ t12; ¥
        t15 = t02 ^ t13; ¥
        t16 = b   ^ d  ; ¥
        y   = t16 ^ t15; }

/* S6:   7  2 12  5  8  4  6 11 14  9  1 15 13  3 10  0 */

/* depth = 8,3,6,3, Total gates=19 */
#define RND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t15, t17, t18,
t01;¥
        t01 = a   & d  ; ¥
        t02 = b   ^ c  ; ¥
        t03 = a   ^ d  ; ¥
        t04 = t01 ^ t02; ¥
        t05 = b   | c  ; ¥
        x   =     ~ t04; ¥
        t07 = t03 & t05; ¥
        t08 = b   & x  ; ¥
        t09 = a   | c  ; ¥
        t10 = t07 ^ t08; ¥
        t11 = b   | d  ; ¥
        t12 = c   ^ t11; ¥
        t13 = t09 ^ t10; ¥
        y   =     ~ t13; ¥
        t15 = x   & t03; ¥
        z   = t12 ^ t07; ¥
        t17 = a   ^ b  ; ¥
        t18 = y   ^ t15; ¥
        w   = t17 ^ t18; }
```

```
/* InvS6:  15 10  1 13  5  3  6  0  4  9 14  7  2 12  8 11 */

/* depth = 5,3,8,6, Total gates=19 */
#define InvRND06(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t07, t08, t09, t12, t13, t14, t15, t16, t17,
t01;¥
        t01 = a   ^ c  ; ¥
        t02 =     ~ c  ; ¥
        t03 = b   & t01; ¥
        t04 = b   | t02; ¥
        t05 = d   | t03; ¥
        t06 = b   ^ d  ; ¥
        t07 = a   & t04; ¥
        t08 = a   | t02; ¥
        t09 = t07 ^ t05; ¥
        x   = t06 ^ t08; ¥
        w   =     ~ t09; ¥
        t12 = b   & w  ; ¥
        t13 = t01 & t05; ¥
        t14 = t01 ^ t12; ¥
        t15 = t07 ^ t13; ¥
        t16 = d   | t02; ¥
        t17 = a   ^ x  ; ¥
        z   = t17 ^ t15; ¥
        y   = t16 ^ t14; }

/* S7:   1 13 15  0 14  8  2 11  7  4 12 10  9  3  5  6 */

/* depth = 10,7,10,4, Total gates=19 */
#define RND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t05, t06, t08, t09, t10, t11, t13, t14, t15, t16, t17,
t01;¥
        t01 = a   & c  ; ¥
        t02 =     ~ d  ; ¥
        t03 = a   & t02; ¥
        t04 = b   | t01; ¥
        t05 = a   & b  ; ¥
        t06 = c   ^ t04; ¥
        z   = t03 ^ t06; ¥
        t08 = c   | z  ; ¥
        t09 = d   | t05; ¥
        t10 = a   ^ t08; ¥
        t11 = t04 & z  ; ¥
        x   = t09 ^ t10; ¥
        t13 = b   ^ x  ; ¥
        t14 = t01 ^ x  ; ¥
        t15 = c   ^ t05; ¥
        t16 = t11 | t13; ¥
        t17 = t02 | t14; ¥
        w   = t15 ^ t17; ¥
        y   = a   ^ t16; }
```

```
/* InvS7:   3  0  6 13  9 14 15  8  5 12 11  7 10  1  4  2 */

/* depth = 9,7,3,3, Total gates=18 */
#define InvRND07(a,b,c,d,w,x,y,z) ¥
        { register unsigned long t02, t03, t04, t06, t07, t08, t09, t10, t11, t13, t14, t15, t16, t01;¥
        t01 = a   & b  ; ¥
        t02 = a   | b  ; ¥
        t03 = c   | t01; ¥
        t04 = d   & t02; ¥
        z   = t03 ^ t04; ¥
        t06 = b   ^ t04; ¥
        t07 = d   ^ z  ; ¥
        t08 =     ~ t07; ¥
        t09 = t06 | t08; ¥
        t10 = b   ^ d  ; ¥
        t11 = a   | d  ; ¥
        x   = a   ^ t09; ¥
        t13 = c   ^ t06; ¥
        t14 = c   & t11; ¥
        t15 = d   | x  ; ¥
        t16 = t01 | t10; ¥
        w   = t13 ^ t15; ¥
        y   = t14 ^ t16; }

#define RND08(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND09(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND10(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND11(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND12(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND13(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND14(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND15(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND16(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND17(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND18(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND19(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND20(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND21(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND22(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND23(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)
#define RND24(a,b,c,d,e,f,g,h) RND00(a,b,c,d,e,f,g,h)
#define RND25(a,b,c,d,e,f,g,h) RND01(a,b,c,d,e,f,g,h)
#define RND26(a,b,c,d,e,f,g,h) RND02(a,b,c,d,e,f,g,h)
#define RND27(a,b,c,d,e,f,g,h) RND03(a,b,c,d,e,f,g,h)
#define RND28(a,b,c,d,e,f,g,h) RND04(a,b,c,d,e,f,g,h)
#define RND29(a,b,c,d,e,f,g,h) RND05(a,b,c,d,e,f,g,h)
#define RND30(a,b,c,d,e,f,g,h) RND06(a,b,c,d,e,f,g,h)
#define RND31(a,b,c,d,e,f,g,h) RND07(a,b,c,d,e,f,g,h)

#define InvRND08(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND09(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND10(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
```

```
#define InvRND11(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND12(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND13(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND14(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND15(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND16(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND17(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND18(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND19(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND20(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND21(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND22(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND23(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)
#define InvRND24(a,b,c,d,e,f,g,h) InvRND00(a,b,c,d,e,f,g,h)
#define InvRND25(a,b,c,d,e,f,g,h) InvRND01(a,b,c,d,e,f,g,h)
#define InvRND26(a,b,c,d,e,f,g,h) InvRND02(a,b,c,d,e,f,g,h)
#define InvRND27(a,b,c,d,e,f,g,h) InvRND03(a,b,c,d,e,f,g,h)
#define InvRND28(a,b,c,d,e,f,g,h) InvRND04(a,b,c,d,e,f,g,h)
#define InvRND29(a,b,c,d,e,f,g,h) InvRND05(a,b,c,d,e,f,g,h)
#define InvRND30(a,b,c,d,e,f,g,h) InvRND06(a,b,c,d,e,f,g,h)
#define InvRND31(a,b,c,d,e,f,g,h) InvRND07(a,b,c,d,e,f,g,h)

/* Linear transformations and key mixing: */

#define ROL(x,n) ((((unsigned long)(x))<<(n))| ¥
                  (((unsigned long)(x))>>(32-(n))))
#define ROR(x,n) ((((unsigned long)(x))<<(32-(n)))| ¥
                  (((unsigned long)(x))>>(n)))

#define transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
      y0 = ROL(x0, 13); ¥
      y2 = ROL(x2, 3); ¥
      y1 = x1 ^ y0 ^ y2; ¥
      y3 = x3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
      y1 = ROL(y1, 1); ¥
      y3 = ROL(y3, 7); ¥
      y0 = y0 ^ y1 ^ y3; ¥
      y2 = y2 ^ y3 ^ ((unsigned long)y1<<7); ¥
      y0 = ROL(y0, 5); ¥
      y2 = ROL(y2, 22)

#define inv_transform(x0, x1, x2, x3, y0, y1, y2, y3) ¥
      y2 = ROR(x2, 22);¥
      y0 = ROR(x0, 5); ¥
      y2 = y2 ^ x3 ^ ((unsigned long)x1<<7); ¥
      y0 = y0 ^ x1 ^ x3; ¥
      y3 = ROR(x3, 7); ¥
      y1 = ROR(x1, 1); ¥
      y3 = y3 ^ y2 ^ ((unsigned long)y0)<<3; ¥
      y1 = y1 ^ y0 ^ y2; ¥
      y2 = ROR(y2, 3); ¥
      y0 = ROR(y0, 13)
```

```
#define keying(x0, x1, x2, x3, subkey) ¥
                        x0^=subkey[0];x1^=subkey[1]; ¥
                        x2^=subkey[2];x3^=subkey[3]

/* PHI: Constant used in the key schedule */
#define PHI 0x9e3779b9L
```

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__278D2248_8894_4A99_A969_4147E7DFC130__INCLUDED_)
#define AFX_STDAFX_H__278D2248_8894_4A99_A969_4147E7DFC130__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__278D2248_8894_4A99_A969_4147E7DFC130__INCLUDED_)
```

つぎは、CPPファイルです。

Serpent.cpp

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

```
/* Copyright (C) 1998 Ross Anderson, Eli Biham, Lars Knudsen
 * All rights reserved.
 *
 * This code is freely distributed for AES selection process.
 * No other use is allowed.
 *
 * Copyright remains of the copyright holders, and as such any Copyright
 * notices in the code are not to be removed.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted only for the AES selection process, provided
 * that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed without the authors permission.
 *  i.e. this code cannot simply be copied and put under another distribution
 * licence [including the GNU Public Licence.]
 */

#include "stdafx.h"
#include "serpent.h"
#include "serpentsboxes.h"

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define BLOCK_SIZE 128
```

```
/*  The functions  */
int serpent_makeKey(keyInstance *key, BYTE direction, int keyLen,
                        char *keyMaterial)
{
  unsigned long i,j;
  unsigned long w[132],k[132];
  int rc;

  if(direction != DIR_ENCRYPT &&
     direction != DIR_DECRYPT)
    return BAD_KEY_DIR;

  if(keyLen>256 || keyLen<1)
    return BAD_KEY_MAT;

  key->direction=direction;
  key->keyLen=keyLen;
  strncpy(key->keyMaterial, keyMaterial, MAX_KEY_SIZE+1);

  rc=serpent_convert_from_string(keyLen, keyMaterial,key->key);
  if(rc<=0)
    return BAD_KEY_MAT;

  for(i=0; i<keyLen/32; i++)
    w[i]=key->key[i];
  if(keyLen<256)
    w[i]=(key->key[i]&((1L<<((keyLen&31)))-1))|(1L<<((keyLen&31)));
  for(i++; i<8; i++)
    w[i]=0;
  for(i=8; i<16; i++)
    w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^(i-8),11);
  for(i=0; i<8; i++)
    w[i]=w[i+8];
  for(i=8; i<132; i++)
    w[i]=ROL(w[i-8]^w[i-5]^w[i-3]^w[i-1]^PHI^i,11);

  RND03(w[  0], w[  1], w[  2], w[  3], k[  0], k[  1], k[  2], k[  3]);
  RND02(w[  4], w[  5], w[  6], w[  7], k[  4], k[  5], k[  6], k[  7]);
  RND01(w[  8], w[  9], w[ 10], w[ 11], k[  8], k[  9], k[ 10], k[ 11]);
  RND00(w[ 12], w[ 13], w[ 14], w[ 15], k[ 12], k[ 13], k[ 14], k[ 15]);
  RND31(w[ 16], w[ 17], w[ 18], w[ 19], k[ 16], k[ 17], k[ 18], k[ 19]);
  RND30(w[ 20], w[ 21], w[ 22], w[ 23], k[ 20], k[ 21], k[ 22], k[ 23]);
  RND29(w[ 24], w[ 25], w[ 26], w[ 27], k[ 24], k[ 25], k[ 26], k[ 27]);
  RND28(w[ 28], w[ 29], w[ 30], w[ 31], k[ 28], k[ 29], k[ 30], k[ 31]);
  RND27(w[ 32], w[ 33], w[ 34], w[ 35], k[ 32], k[ 33], k[ 34], k[ 35]);
  RND26(w[ 36], w[ 37], w[ 38], w[ 39], k[ 36], k[ 37], k[ 38], k[ 39]);
  RND25(w[ 40], w[ 41], w[ 42], w[ 43], k[ 40], k[ 41], k[ 42], k[ 43]);
  RND24(w[ 44], w[ 45], w[ 46], w[ 47], k[ 44], k[ 45], k[ 46], k[ 47]);
  RND23(w[ 48], w[ 49], w[ 50], w[ 51], k[ 48], k[ 49], k[ 50], k[ 51]);
  RND22(w[ 52], w[ 53], w[ 54], w[ 55], k[ 52], k[ 53], k[ 54], k[ 55]);
  RND21(w[ 56], w[ 57], w[ 58], w[ 59], k[ 56], k[ 57], k[ 58], k[ 59]);
  RND20(w[ 60], w[ 61], w[ 62], w[ 63], k[ 60], k[ 61], k[ 62], k[ 63]);
```

```
  RND19(w[ 64], w[ 65], w[ 66], w[ 67], k[ 64], k[ 65], k[ 66], k[ 67]);
  RND18(w[ 68], w[ 69], w[ 70], w[ 71], k[ 68], k[ 69], k[ 70], k[ 71]);
  RND17(w[ 72], w[ 73], w[ 74], w[ 75], k[ 72], k[ 73], k[ 74], k[ 75]);
  RND16(w[ 76], w[ 77], w[ 78], w[ 79], k[ 76], k[ 77], k[ 78], k[ 79]);
  RND15(w[ 80], w[ 81], w[ 82], w[ 83], k[ 80], k[ 81], k[ 82], k[ 83]);
  RND14(w[ 84], w[ 85], w[ 86], w[ 87], k[ 84], k[ 85], k[ 86], k[ 87]);
  RND13(w[ 88], w[ 89], w[ 90], w[ 91], k[ 88], k[ 89], k[ 90], k[ 91]);
  RND12(w[ 92], w[ 93], w[ 94], w[ 95], k[ 92], k[ 93], k[ 94], k[ 95]);
  RND11(w[ 96], w[ 97], w[ 98], w[ 99], k[ 96], k[ 97], k[ 98], k[ 99]);
  RND10(w[100], w[101], w[102], w[103], k[100], k[101], k[102], k[103]);
  RND09(w[104], w[105], w[106], w[107], k[104], k[105], k[106], k[107]);
  RND08(w[108], w[109], w[110], w[111], k[108], k[109], k[110], k[111]);
  RND07(w[112], w[113], w[114], w[115], k[112], k[113], k[114], k[115]);
  RND06(w[116], w[117], w[118], w[119], k[116], k[117], k[118], k[119]);
  RND05(w[120], w[121], w[122], w[123], k[120], k[121], k[122], k[123]);
  RND04(w[124], w[125], w[126], w[127], k[124], k[125], k[126], k[127]);
  RND03(w[128], w[129], w[130], w[131], k[128], k[129], k[130], k[131]);

  for(i=0; i<=32; i++)
    for(j=0; j<4; j++)
      key->subkeys[i][j] = k[4*i+j];

  return TRUE;
}

int cipherInit(cipherInstance *cipher, BYTE mode, char *IV)
{
//  int i;
  int rc;

  if((mode != MODE_ECB) &&
     (mode != MODE_CBC) &&
     (mode != MODE_CFB1))
    return BAD_CIPHER_MODE;

  cipher->mode = mode;            /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
  cipher->blockSize=128;
  if(mode != MODE_ECB)
    {
      rc=serpent_convert_from_string(cipher->blockSize, IV, (unsigned long*)cipher->IV);
      if(rc<=0)
        return BAD_CIPHER_STATE;
    }

  return TRUE;
}

int blockEncrypt(cipherInstance *cipher,
                 keyInstance *key,
                 BYTE *input,
                 int inputLen,
                 BYTE *outBuffer)
```

```
{
  unsigned long t[4];
  int  b, n, i;
  DWORD x[BLOCK_SIZE/32];
  BYTE bit, bit0, ctBit, carry;

  /*
   * Note about optimization: the code becomes slower of the calls to
   * serpent_encrypt and serpent_decrypt are replaced by inlined code.
   * (tested on Pentium 133MMX)
   */

  switch(cipher->mode)
    {
    case MODE_ECB:
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
        serpent_encrypt((unsigned long*)input,(unsigned long*) outBuffer, key->subkeys);
      return inputLen;

    case MODE_CBC:
      t[0] = ((unsigned long*)cipher->IV)[0];
      t[1] = ((unsigned long*)cipher->IV)[1];
      t[2] = ((unsigned long*)cipher->IV)[2];
      t[3] = ((unsigned long*)cipher->IV)[3];
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
        {
          t[0] ^= ((unsigned long*)input)[0];
          t[1] ^= ((unsigned long*)input)[1];
          t[2] ^= ((unsigned long*)input)[2];
          t[3] ^= ((unsigned long*)input)[3];
          serpent_encrypt(t, t, key->subkeys);
          ((unsigned long*)outBuffer)[0] = t[0];
          ((unsigned long*)outBuffer)[1] = t[1];
          ((unsigned long*)outBuffer)[2] = t[2];
          ((unsigned long*)outBuffer)[3] = t[3];
        }
      ((unsigned long*)cipher->IV)[0] = t[0];
      ((unsigned long*)cipher->IV)[1] = t[1];
      ((unsigned long*)cipher->IV)[2] = t[2];
      ((unsigned long*)cipher->IV)[3] = t[3];

      return inputLen;

    case MODE_CFB1:
                cipher->mode = MODE_ECB; /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,(BYTE *)cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0  = 0x80 >> (n & 7);/* which bit position in byte */
                        ctBit = (input[n/8] & bit0) ^ ((((BYTE *) x)[0] & 0x80) >> (n&7));
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) | ctBit;
                        carry = ctBit >> (7 - (n&7));
```

```
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;/* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;/* restore mode for next time */
                return inputLen;

    default:
      return BAD_CIPHER_STATE;
    }
}

int blockDecrypt(cipherInstance *cipher,
                 keyInstance *key,
                 BYTE *input,
                 int inputLen,
                 BYTE *outBuffer)
{
  unsigned long t[4];
  int  b, n, i;
  DWORD x[BLOCK_SIZE/32];
  BYTE bit, bit0, ctBit, carry;

  switch(cipher->mode)
    {
    case MODE_ECB:
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
          serpent_decrypt( (unsigned long*)input, (unsigned long*)outBuffer, key->subkeys);
      return inputLen;

    case MODE_CBC:
      t[0] = ((unsigned long*)cipher->IV)[0];
      t[1] = ((unsigned long*)cipher->IV)[1];
      t[2] = ((unsigned long*)cipher->IV)[2];
      t[3] = ((unsigned long*)cipher->IV)[3];
      for(b=0; b<inputLen; b+=128, input+=16, outBuffer+=16)
        {
          serpent_decrypt((unsigned long*)input, (unsigned long*)outBuffer,key->subkeys);
          ((unsigned long*)outBuffer)[0] ^= t[0];
          ((unsigned long*)outBuffer)[1] ^= t[1];
          ((unsigned long*)outBuffer)[2] ^= t[2];
          ((unsigned long*)outBuffer)[3] ^= t[3];
          t[0] = ((unsigned long*)input)[0];
          t[1] = ((unsigned long*)input)[1];
          t[2] = ((unsigned long*)input)[2];
          t[3] = ((unsigned long*)input)[3];
        }
      ((unsigned long*)cipher->IV)[0] = t[0];
      ((unsigned long*)cipher->IV)[1] = t[1];
```

```
      ((unsigned long*)cipher->IV)[2] = t[2];
      ((unsigned long*)cipher->IV)[3] = t[3];
      return inputLen;

    case MODE_CFB1://blockDecrypt
                cipher->mode = MODE_ECB; /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,(BYTE *)cipher->IV,BLOCK_SIZE,(BYTE *)x);
                        bit0  = 0x80 >> (n & 7);
                        ctBit = input[n/8] & bit0;
                        outBuffer[n/8] = (outBuffer[n/8] & ~ bit0) |
                                                        (ctBit ^ ((((BYTE *) x)[0] & 0x80) >> (n&7)));
                        carry = ctBit >> (7 - (n&7));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = cipher->IV[i] >> 7;/* save next "carry" from shift */
                                cipher->IV[i] = (cipher->IV[i] << 1) ^ carry;
                                carry = bit;
                                }
                        }
                cipher->mode = MODE_CFB1;/* restore mode for next time */
                return inputLen;

    default:
      return BAD_CIPHER_STATE;
    }
}

void serpent_encrypt(unsigned long plaintext[4],
                unsigned long ciphertext[4],
                unsigned long subkeys[33][4])
{
  register unsigned long x0, x1, x2, x3;
  register unsigned long y0, y1, y2, y3;

  x0=plaintext[0];
  x1=plaintext[1];
  x2=plaintext[2];
  x3=plaintext[3];

  /* Start to encrypt the plaintext x */
  keying(x0, x1, x2, x3, subkeys[ 0]);
  RND00(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[ 1]);
  RND01(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[ 2]);
  RND02(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[ 3]);
```

```
RND03(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 4]);
RND04(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 5]);
RND05(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 6]);
RND06(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 7]);
RND07(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 8]);
RND08(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[ 9]);
RND09(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[10]);
RND10(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[11]);
RND11(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[12]);
RND12(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[13]);
RND13(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[14]);
RND14(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[15]);
RND15(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[16]);
RND16(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[17]);
RND17(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[18]);
RND18(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[19]);
RND19(x0, x1, x2, x3, y0, y1, y2, y3);
transform(y0, y1, y2, y3, x0, x1, x2, x3);
keying(x0, x1, x2, x3, subkeys[20]);
RND20(x0, x1, x2, x3, y0, y1, y2, y3);
```

```
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[21]);
  RND21(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[22]);
  RND22(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[23]);
  RND23(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[24]);
  RND24(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[25]);
  RND25(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[26]);
  RND26(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[27]);
  RND27(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[28]);
  RND28(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[29]);
  RND29(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[30]);
  RND30(x0, x1, x2, x3, y0, y1, y2, y3);
  transform(y0, y1, y2, y3, x0, x1, x2, x3);
  keying(x0, x1, x2, x3, subkeys[31]);
  RND31(x0, x1, x2, x3, y0, y1, y2, y3);
  x0 = y0; x1 = y1; x2 = y2; x3 = y3;
  keying(x0, x1, x2, x3, subkeys[32]);
  /* The ciphertext is now in x */

  ciphertext[0] = x0;
  ciphertext[1] = x1;
  ciphertext[2] = x2;
  ciphertext[3] = x3;
}

void serpent_decrypt(unsigned long ciphertext[4],
                     unsigned long plaintext[4],
                     unsigned long subkeys[33][4])
{
  register unsigned long x0, x1, x2, x3;
  register unsigned long y0, y1, y2, y3;

  x0=ciphertext[0];
  x1=ciphertext[1];
```

```
x2=ciphertext[2];
x3=ciphertext[3];

/* Start to decrypt the ciphertext x */
keying(x0, x1, x2, x3, subkeys[32]);
InvRND31(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[31]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND30(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[30]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND29(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[29]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND28(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[28]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND27(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[27]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND26(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[26]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND25(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[25]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND24(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[24]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND23(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[23]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND22(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[22]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND21(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[21]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND20(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[20]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND19(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[19]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND18(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[18]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND17(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[17]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND16(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[16]);
```

```
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND15(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[15]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND14(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[14]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND13(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[13]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND12(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[12]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND11(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[11]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND10(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[10]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND09(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 9]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND08(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 8]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND07(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 7]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND06(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 6]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND05(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 5]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND04(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 4]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND03(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 3]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND02(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 2]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND01(x0, x1, x2, x3, y0, y1, y2, y3);
keying(y0, y1, y2, y3, subkeys[ 1]);
inv_transform(y0, y1, y2, y3, x0, x1, x2, x3);
InvRND00(x0, x1, x2, x3, y0, y1, y2, y3);
x0 = y0; x1 = y1; x2 = y2; x3 = y3;
keying(x0, x1, x2, x3, subkeys[ 0]);
/* The plaintext is now in x */

plaintext[0] = x0;
```

```
  plaintext[1] = x1;
  plaintext[2] = x2;
  plaintext[3] = x3;
}

//#define min(x,y) (((x)<(y))?(x):(y))

int serpent_convert_from_string(int len, char *str, unsigned long *val)
/* the size of val must be at least the next multiple of 32 */
/* bits after len bits */
{
  int is, iv;
  int slen=min((int)strlen(str), (len+3)/4);

  if(len<0)
    return -1;          /* Error!!! */

  if(len>slen*4 || len<slen*4-3)
    return -1;          /* Error!!! */

  for(is=0; is<slen; is++)
    if(((str[is]<'0')||(str[is]>'9')) &&
       ((str[is]<'A')||(str[is]>'F')) &&
       ((str[is]<'a')||(str[is]>'f')))
      return -1; /* Error!!! */

  for(is=slen, iv=0; is>=8; is-=8, iv++)
    {
      unsigned long t;
      sscanf(&str[is-8], "%08lX", &t);
      val[iv] = t;
    }
  if(is>0)
    {
      char tmp[10];
      unsigned long t;
      strncpy(tmp, str, is);
      tmp[is] = 0;
      sscanf(tmp, "%08lX", &t);
      val[iv++] = t;
    }
  for(; iv<(len+31)/32; iv++)
    val[iv] = 0;
  return iv;
}

char *serpent_convert_to_string(int len, unsigned long val[8], char *str)
/* str must have at least (len+3)/4+1 bytes. */
{
  int i;

  if(len<0)
```

```
    return (char *)-1;            /* Error!!! */

  str[0] = 0;
  i=len/32;
  if((len&31)>0)
    {
      char tmp[10];
      sprintf(tmp, "%08lX", val[i]&(((len&31)<<1)-1));
      strcat(str, &tmp[8-(((len&31)+3)/4)]);
    }
  for(i--; i>=0; i--)
    {
      char tmp[10];
      sprintf(tmp, "%08lX", val[i]);
      strcat(str, tmp);
    }
  return str;
}
```

Serpentkey.cpp

```
  ////////////////////////////////
// SerpentKey.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 "SerpentKey.h"
#include "SerpentKeyDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyApp

BEGIN_MESSAGE_MAP(CSerpentKeyApp, CWinApp)
        //{{AFX_MSG_MAP(CSerpentKeyApp)
                // 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()

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

```
// CSerpentKeyApp クラスの構築

CSerpentKeyApp::CSerpentKeyApp()
{
	// 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CSerpentKeyApp オブジェクト

CSerpentKeyApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyApp クラスの初期化

BOOL CSerpentKeyApp::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

	CSerpentKeyDlg 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;
}
```

Serpentkeydlg.cpp

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

#include "stdafx.h"
#include "SerpentKey.h"
#include "SerpentKeyDlg.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];
/////////////////////////////////////////////////////////////////////////////
// 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
// 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 ' ';
}


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()

/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyDlg ダイアログ
/////////////////////////////////////////////////////////////////////////////
// CSerpentKeyDlg ダイアログ

CSerpentKeyDlg::CSerpentKeyDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSerpentKeyDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSerpentKeyDlg)
	i_KeyLen = 0;
	i_Mode = 0;
	s_fname1 = "SerpkeyEC.bin";
	s_fname2 = "SerpkeyDC.bin";
	srand( (unsigned)time(NULL) );//乱数初期化
```

```
	madekey = 0;
	planedata_file = "plane.txt";
	encryptdata_file = "encrypt.enc";
	decryptdata_file = "decrypt.txt";
	s_cipherinit = "0123456789abcdef0123456789abcdef";
		// 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 CSerpentKeyDlg::OnOK()
{
	if(madekey==0){
		MessageBox("鍵を作成してください。");
		return;
	}

	CDialog::OnOK();

	ofstream ofs(s_fname1, ios::out);
	if(i_Mode == 0) ofs << 1 << endl;
	if(i_Mode == 1) ofs << 2 << endl;
	if(i_Mode == 2) ofs << 3 << endl;
	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 << s_cipherinit << endl;
	ofs.close();

	ofstream ofs2(s_fname2, ios::out);
	if(i_Mode == 0) ofs2 << 1 << endl;
	if(i_Mode == 1) ofs2 << 2 << endl;
	if(i_Mode == 2) ofs2 << 3 << endl;
	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 << s_cipherinit << endl;
	ofs2.close();
}

void CSerpentKeyDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSerpentKeyDlg)
	DDX_Radio(pDX, IDC_RADIO1, i_KeyLen);
	DDX_Radio(pDX, IDC_RADIO5, i_Mode);
	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_Text(pDX, IDC_CIPHERINIT, s_cipherinit);
    DDV_MaxChars(pDX, s_cipherinit, 32);
        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(CSerpentKeyDlg, CDialog)
        //{{AFX_MSG_MAP(CSerpentKeyDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_COMMAND(ID_MAKEKEY, MakeKey)
        ON_COMMAND(ID_TESTKEY, TestKey)
        ON_COMMAND(IDC_CHGCI, ChgCI)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

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

        n=16;

        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_cipherinit = b;

        CWnd * pwnd = GetDlgItem(IDC_CIPHERINIT);
        pwnd->SetWindowText(b);

        return;// 0;
}

void CSerpentKeyDlg::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;

        CWnd * pwnd = GetDlgItem(IDC_SECRETKEY1);
```

```
        pwnd->SetWindowText(b);
        pwnd = GetDlgItem(IDC_SECRETKEY2);
        pwnd->SetWindowText(b);
        madekey = 1;
}


void CSerpentKeyDlg::TestKey()
{
        unsigned char key2[128+2];
        char c_ci[65];
        int  cnt, c, block, i, j;
        long 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);

        strcpy((char *)c_ci, s_cipherinit);

        /*Set mode*/
        if(IDC_RADIO5 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_ECB, "");
        }
        if(IDC_RADIO6 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_CBC, c_ci);//"0123456789abcdef0123456789abcdef");
        }
        if(IDC_RADIO7 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_CFB1, c_ci);//"0123456789abcdef0123456789abcdef");
        }
        if(rc<=0){
                MessageBox("モード設定が出来ません。");
                return;//(-2);
        }
```

```
        serpent_makeKey(&keyI, DIR_ENCRYPT,  n*8, (char*)key2 );

    // 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）+ sizeof(long)

        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-4, 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");


        rc=blockEncrypt(&cipherI, &keyI,(unsigned char*)(bufp),8*mesLength,(unsigned char*)(bufc));

        // 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]);

    strcpy((char *)c_ci, s_cipherinit);

    /*Set mode*/
    if(IDC_RADIO5 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
            rc=cipherInit(&cipherI, MODE_ECB, "");
    }
    if(IDC_RADIO6 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
            rc=cipherInit(&cipherI, MODE_CBC, c_ci);//"0123456789abcdef0123456789abcdef");
    }
    if(IDC_RADIO7 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
            rc=cipherInit(&cipherI, MODE_CFB1, c_ci);//"0123456789abcdef0123456789abcdef");
    }
    if(rc<=0){
            MessageBox("モード設定が出来ません。");
            return;//(-2);
    }
serpent_makeKey(&keyI, DIR_DECRYPT,  n*8, (char*)key2 );

rc=blockDecrypt(&cipherI, &keyI,(unsigned char*)(bufc), 8*mesLength, (unsigned char*)(bufdc));

    // 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;
        delete[] bufc;

        return;// 0;
}


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

BOOL CSerpentKeyDlg::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 CSerpentKeyDlg::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 CSerpentKeyDlg::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 CSerpentKeyDlg::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
```

```
//                SerpentCrypt.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。