MARSKey のソースファイル

(C) 宇山　靖政

2012.04.08

MRAScrypt.exe は　　MARSKey.exe のソースコードに次のものを追加したものです。

追加ソースコード 開始:

```
void CMarsKeyDlg::DCKRead()
{
	int i;

	CString strNewFileName;
	CString sFName;
	CWnd* pwnd;


	strNewFileName = "*.*";
	CFileDialog fileDlg(TRUE, NULL,"*.bin", OFN_HIDEREADONLY, strNewFileName);
	if(fileDlg.DoModal() == IDOK){
		strNewFileName = fileDlg.GetPathName();

		// If file doesn't already exist, then create it.
		CFile file;
		CFileStatus status;
		if (!file.GetStatus(strNewFileName, status)){
			CString strMessage;
//			AfxFormatString1(strMessage, IDS_MAKENEWFILE,
//				strNewFileName);
			if(AfxMessageBox(strMessage, MB_YESNO) == IDNO)	{
				return;
			}
			if (!file.Open(strNewFileName, CFile::modeCreate)){
				CString strMessage;
//				AfxFormatString1(strMessage, IDS_NOADBOOKMAILBOXTEMPLATE,
//					strNewFileName);
				AfxMessageBox(strMessage);
				return;
			}
			file.Close();
		}
	}

	FILE *fkey = 0;
	char c_klen[8], c_mode[8], c_skey[128];

	if((fkey = fopen(strNewFileName, "rt")) == NULL){
		MessageBox( "Can not find key file for encryption. ¥n");
		return ;
```

```
}

fgets( c_mode, 8, fkey );
fgets( c_klen, 8, fkey );
fgets( c_skey, 128, fkey );

if(fkey){ fclose(fkey);  }

i = atoi(c_klen);

if(i == 128 ){
         keylength = "128";
        CheckRadioButton(IDC_RADIO1, IDC_RADIO4, IDC_RADIO1);
}else{
        if(i==192){
                 keylength = "192";
                CheckRadioButton(IDC_RADIO1, IDC_RADIO4, IDC_RADIO2);
        }else{
                if(i==256){
                         keylength = "256";
                        CheckRadioButton(IDC_RADIO1, IDC_RADIO4, IDC_RADIO3);
                }else{
                        if(i==448){
                                 keylength = "448";
                                CheckRadioButton(IDC_RADIO1, IDC_RADIO4, IDC_RADIO4);
                        }else{
                                MessageBox( "鍵の長さが不正です。¥n");
                                return ;
                        }
                }
        }
}

i = atoi(c_mode);
if(i == 1 ){
        CheckRadioButton(IDC_RADIO5, IDC_RADIO7, IDC_RADIO5);
}else{
        if(i==2){
                CheckRadioButton(IDC_RADIO5, IDC_RADIO7, IDC_RADIO6);
        }else{
                if(i==3){
                        CheckRadioButton(IDC_RADIO5, IDC_RADIO7, IDC_RADIO7);
                }else{
                        MessageBox( "モードが不正です。¥n");
                        return ;
                }
        }
}

pwnd = GetDlgItem(IDC_SECRETKEY1);
pwnd->SetWindowText(c_skey);
pwnd = GetDlgItem(IDC_SECRETKEY2);
```

```
        pwnd->SetWindowText(c_skey);

        sFName = strNewFileName;
        int jj = sFName.ReverseFind( '¥¥' );
        sFName.Delete(0,jj+1);
        pwnd = GetDlgItem(IDC_DECRYPTKEY_FILE);
        pwnd->SetWindowText(sFName);
        pwnd = GetDlgItem(IDC_ENCRYPTKEY_FILE);
        pwnd->SetWindowText(sFName);

        madekey = 1;

        return ;

        // Open the file now that it has been created.
//      strcpy(tmppath, strNewFileName);
//      OpenDocumentFile(strNewFileName);


}



void CMarsKeyDlg::Search2(){
        CString strNewFileName;

        strNewFileName.LoadString(IDS_BSEARCH);
        CFileDialog fileDlg(TRUE, NULL, NULL, OFN_HIDEREADONLY, strNewFileName);
        if(fileDlg.DoModal() == IDOK){
                strNewFileName = fileDlg.GetPathName();
            l_bin.AddString(strNewFileName);
        }
}

void CMarsKeyDlg::AddList2(){
        CWnd* pwnd;
        pwnd = GetDlgItem(IDC_ADDBIN2);
        pwnd->GetWindowText(s_addbin);
    l_bin.AddString(s_addbin);
        s_addbin = "";
        pwnd->SetWindowText(s_addbin);
}

void CMarsKeyDlg:: DelList2(){
        int index = l_bin.GetCurSel();
        l_bin.DeleteString((UINT)index);
}

void CMarsKeyDlg::OnBnClickedButton5(){//暗号化（連続）
        unsigned long filelen, mesLength; // 平文長（バイト）
        char* bufp;
        unsigned char* bufc;
```

```
int numclosed;
int rc;
char Key[256/*128+2*/];
unsigned char key2[128+2];
char c_ci[65];
int  cnt, c, block, i;
int n;


CWnd* pwnd;
CFileStatus fs, fsec, fsdc;

if(madekey==0){
        MessageBox("鍵を作成してください。");
        return;
}

datadisp.SetLimitText(INT_MAX);

if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
{
        n= 16;
        keylength = "128";
}
if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
{
        n= 24;
        keylength = "192";
}
if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
{
        n= 32;
        keylength = "256";
}
if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
{
        n= 56;
        keylength = "448";
}

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(&encipher, MODE_ECB, "");
}
if(IDC_RADIO6 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
        rc=cipherInit(&encipher, MODE_CBC, c_ci);//"0123456789abcdef0123456789abcdef");
```

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

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


	pwnd = GetDlgItem(IDC_SECRETKEY1);
	pwnd->GetWindowText(s_key1);
	strcpy_s(Key, s_key1);

////////////////////////////////////////////////////////////////
		yDisplay("¥r¥n");
		yDisplay("鍵長    = "); yDisplay(keylength); yDisplay(" bit");
		yDisplay("¥r¥n");
		yDisplay("¥r¥n");

	SetCurrentDirectory(bufpath);

	char buf[256];
	int i2, n2;
	n2 = l_bin.GetCount();

	for(i2=0; i2<n2; i2++){
	l_bin.SetCurSel(i2);
		l_bin.GetText(i2,buf);
////////////////////////////////////////////////////////////
	planedata_file = buf;

 /* ファイルの名前*/
	CString sFName = planedata_file;
	int j = sFName.ReverseFind( '¥¥' );
	sFName.Delete(0,j+1);
	encryptdata_file = ".¥¥Encrypted¥¥";
	encryptdata_file += sFName;

   /* 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）
    */
	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*/
```

```
        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");}


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

    // 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[256];
        CString  Ssd;

        pwnd = GetDlgItem(IDC_SECRETKEY1);
        pwnd->GetWindowText(Ssd);
```

```
        strcpy_s(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(&encipher, &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" );

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

        delete[] bufp;
        delete[] bufc;

        ////////////////////////////////////////////////////////////////////
        }
////////////////////////////////////////////////////////////////////////////
        yDisplay("¥r¥n");
        yDisplay("暗号化終了。¥n");
        yDisplay("¥r¥n");
```

```
	return;
}


void CMarsKeyDlg::OnBnClickedButton6(){//復号化（連続）
	unsigned long filelen, mesLength; // 平文長（バイト）
	unsigned char* bufc;
	int numclosed;
	int rc;
	char* bufdc;
	int n;
	unsigned char key2[128+2];
	char c_ci[65];
	int c, block, i;

	CWnd* pwnd;
	CFileStatus fs, fsec, fsdc;

	if(madekey==0){
		MessageBox("鍵を作成してください。");
		return;
	}

	datadisp.SetLimitText(INT_MAX);

	if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 16;
		keylength = "128";
	}
	if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 24;
		keylength = "192";
	}
	if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 32;
		keylength = "256";
	}
	if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 56;
		keylength = "448";
	}

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

        makeKey(&keyI, DIR_DECRYPT,  n*8, (char*)key2 );


////////////////////////////////////////////////////////////////
                yDisplay("¥r¥n");
                yDisplay("鍵長    = "); yDisplay(keylength); yDisplay(" bit");
                yDisplay("¥r¥n");
                yDisplay("¥r¥n");

        SetCurrentDirectory(bufpath);

        char buf[256];
        int i2, n2;
        n2 = l_bin.GetCount();

        for(i2=0; i2<n2; i2++){
        l_bin.SetCurSel(i2);
                l_bin.GetText(i2,buf);
////////////////////////////////////////////////////////////
        encryptdata_file = buf;

        CString sFName = encryptdata_file;
        int j = sFName.ReverseFind( '¥¥' );
        sFName.Delete(0,j+1);
        decryptdata_file = ".¥¥Decrypted¥¥";
        decryptdata_file += sFName;

   /* 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）
    */
   /* 暗号文を書き込むファイルを開く*/
        if( (stream1 = fopen( encryptdata_file, "rb" )) == 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*/
        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");}

////////////////////////////////////
        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]);

    rc=blockDecrypt(&decipher, &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( stream1 ) )
      MessageBox( "暗号化ファイルは閉じられませんでした。¥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[] bufdc;
        delete[] bufc;
        ////////////////////////////////////////////////////////////////////
        }
////////////////////////////////////////////////////////////////////////
        yDisplay("¥r¥n");
        yDisplay("復号化終了。¥n");
        yDisplay("¥r¥n");

        return;
}




void CMarsKeyDlg::OnBnClickedButton1(){//暗号化（連続　非表示）
        unsigned long filelen, mesLength; // 平文長（バイト）
        char* bufp;
        unsigned char* bufc;
        int numclosed;
        int rc;
        char Key[256/*128+2*/];
        unsigned char key2[128+2];
        char c_ci[65];
        int  cnt, c, block, i;
        int n;


        CWnd* pwnd;
        CFileStatus fs, fsec, fsdc;

        if(madekey==0){
                MessageBox("鍵を作成してください。");
                return;
        }

        datadisp.SetLimitText(INT_MAX);

        if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 16;
                keylength = "128";
        }
        if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
```

```
                n= 24;
                keylength = "192";
        }
        if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 32;
                keylength = "256";
        }
        if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 56;
                keylength = "448";
        }

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

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


        pwnd = GetDlgItem(IDC_SECRETKEY1);
        pwnd->GetWindowText(s_key1);
        strcpy_s(Key, s_key1);

////////////////////////////////////////////////////////////////
                yDisplay("¥r¥n");
                yDisplay("鍵長    = "); yDisplay(keylength); yDisplay(" bit");
                yDisplay("¥r¥n");
                yDisplay("¥r¥n");

        SetCurrentDirectory(bufpath);

        char buf[256];
        int i2, n2;
```

```
        n2 = l_bin.GetCount();

        for(i2=0; i2<n2; i2++){
        l_bin.SetCurSel(i2);
                l_bin.GetText(i2,buf);
////////////////////////////////////////////////////////
        planedata_file = buf;

 /* ファイルの名前*/
        CString sFName = planedata_file;
        int j = sFName.ReverseFind( '¥¥' );
        sFName.Delete(0,j+1);
        encryptdata_file = ".¥¥Encrypted¥¥";
        encryptdata_file += sFName;

   /* 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）
    */
        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*/
        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");}


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

    // 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[256];
        CString  Ssd;

        pwnd = GetDlgItem(IDC_SECRETKEY1);
        pwnd->GetWindowText(Ssd);
        strcpy_s(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(&encipher, &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" );

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

        delete[] bufp;
        delete[] bufc;

        ////////////////////////////////////////////////////////////////////
        }
////////////////////////////////////////////////////////////////////////////
        yDisplay("¥r¥n");
        yDisplay("暗号化終了。¥n");
        yDisplay("¥r¥n");

        return;
}


void CMarsKeyDlg::OnBnClickedButton2(){//復号化（連続　非表示）
        unsigned long filelen, mesLength; // 平文長（バイト）
        unsigned char* bufc;
        int numclosed;
        int rc;
        char* bufdc;
        int n;
        unsigned char key2[128+2];
        char c_ci[65];
        int   block, i;

        CWnd* pwnd;
        CFileStatus fs, fsec, fsdc;

        if(madekey==0){
                MessageBox("鍵を作成してください。");
                return;
        }

        datadisp.SetLimitText(INT_MAX);
```

```
	if(IDC_RADIO1 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 16;
		keylength = "128";
	}
	if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 24;
		keylength = "192";
	}
	if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 32;
		keylength = "256";
	}
	if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
	{
		n= 56;
		keylength = "448";
	}

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

	makeKey(&keyI, DIR_DECRYPT,  n*8, (char*)key2 );


////////////////////////////////////////////////////////////////
		yDisplay("¥r¥n");
		yDisplay("鍵長    = "); yDisplay(keylength); yDisplay(" bit");
		yDisplay("¥r¥n");
		yDisplay("¥r¥n");

	SetCurrentDirectory(bufpath);
```

```
        char buf[256];
        int i2, n2;
        n2 = l_bin.GetCount();

        for(i2=0; i2<n2; i2++){
        l_bin.SetCurSel(i2);
                l_bin.GetText(i2,buf);
////////////////////////////////////////////////////////
        encryptdata_file = buf;

        CString sFName = encryptdata_file;
        int j = sFName.ReverseFind( '¥¥' );
        sFName.Delete(0,j+1);
        decryptdata_file = ".¥¥Decrypted¥¥";
        decryptdata_file += sFName;

   /* 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）
    */
   /* 暗号文を書き込むファイルを開く*/
        if( (stream1 = fopen( encryptdata_file, "rb" )) == 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*/
        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");}

////////////////////////////////////
        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]);

    rc=blockDecrypt(&decipher, &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( stream1 ) )
      MessageBox( "暗号化ファイルは閉じられませんでした。¥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[] bufdc;
        delete[] bufc;
        ////////////////////////////////////////////////////////////////////
        }
//////////////////////////////////////////////////////////////////////////
        yDisplay("¥r¥n");
        yDisplay("復号化終了。¥n");
        yDisplay("¥r¥n");

        return;
}
```

追加ソースコード　終了：

MARSKey.exe は、 MARS 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。

Aes.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
 *
 *    Modified for IBM submission
 *    David Safford
 *    4/16/1998
 */

#include <stdio.h>

/*****************************************************************************
 *
 *    NIST High Level C API, with some IBM additions
 *
 *****************************************************************************/

#define TRUE            1
#define FALSE           0

#define DIR_ENCRYPT     0    /*  Are we encrypting?                     */
#define DIR_DECRYPT     1    /*  Are we decrypting?                     */
#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 BAD_KEY_DIR      -1  /*  Key direction is invalid               */
#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                      */

typedef unsigned char BYTE;

/* IBM Addition - a WORD must be 32 bits for this implementation */
typedef unsigned long DWORD;

/* IBM specific defines: these parameters can be changed              */
#define NUM_MIX 8           /* number of mixing rounds per stage      */
#define NUM_ROUNDS 16       /* number of full core rounds             */
#define NUM_SETUP 7         /* number of key setup mixing rounds      */

/* IBM specific defines: these parameters are fixed for this implementation */
#define W          32       /* number of bits in a word               */
#define NUM_DATA    4       /* data block size in words               */
#define EKEY_DWORDS (2*(NUM_DATA+NUM_ROUNDS))    /* number of subkey words   */

/* IBM modified values                                                */
#define MAX_KEY_SIZE (EKEY_DWORDS*8)  /* max ASCII char's needed for a key   */
#define MAX_IV_SIZE (NUM_DATA*4)     /* max bytes's needed for an IV        */

/*  The structure for key information */
typedef struct {
    BYTE  direction;        /*  Key used for encrypting or decrypting?      */
    int   keyLen;           /*  Length of the key in BITS                   */
    char  keyMaterial[MAX_KEY_SIZE+1];  /*  Raw key data in ASCII           */
    DWORD  E[EKEY_DWORDS];   /* IBM addition for mars expanded key           */
} keyInstance;

/*  The structure for cipher information */
typedef struct {
    BYTE mode;              /*  MODE_ECB, MODE_CBC, or MODE_CFB1           */
    BYTE IV[MAX_IV_SIZE];  /*  initial binary IV BYTE for chaining         */
    DWORD CIV[NUM_DATA];    /*  IBM addition: current IV in binary WORDs    */
} cipherInstance;

/*  NIST High level function protoypes  */
int 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);

/*****************************************************************************
 *
 *    IBM Low Level (WORD Oriented) API
 *
```

```
 ****************************************************************************/

/* setup a mars expanded key
 *
 * k  (input) is the number of words in the key
 * kp (input) is a pointer to the array of k key words
 * ep (output) is a pointer to an array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int k, WORD *kp, WORD *ep);

/* The basic mars encryption of one block (of NUM_DATA WORDS) */
void mars_encrypt(WORD *in, WORD *out, WORD *ep);

/* mars decryption is simply encryption in reverse */
void mars_decrypt(WORD *in, WORD *out, WORD *ep);
```

Marskey.h

```
////////////////////////////////////////////////////
// MarsKey.h : MARSKEY 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_MARSKEY_H__C5D6E195_3CF2_4052_9EA0_432C5003938C__INCLUDED_)
#define AFX_MARSKEY_H__C5D6E195_3CF2_4052_9EA0_432C5003938C__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

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

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

// 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(CMarsKeyApp)
        public:
        virtual BOOL InitInstance();
        //}}AFX_VIRTUAL

// インプリメンテーション

        //{{AFX_MSG(CMarsKeyApp)
                // 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_MARSKEY_H__C5D6E195_3CF2_4052_9EA0_432C5003938C__INCLUDED_)
```

marskeyDlg.h

```
//////////////////////////////////////////////////
// MarsKeyDlg.h : ヘッダーファイル
//
#if !defined(AFX_MARSKEYDLG_H__CF8BFFCD_3843_4D15_A3F8_8572AFAE7C1C__INCLUDED_)
#define AFX_MARSKEYDLG_H__CF8BFFCD_3843_4D15_A3F8_8572AFAE7C1C__INCLUDED_

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


/////////////////////////////////////////////////////////////////////////////
// CMarsKeyDlg ダイアログ
#include "MarsKey.h"
#include "aes.h"

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

```
// ダイアログデータ
        //{{AFX_DATA(CMarsKeyDlg)
        enum { IDD = IDD_MARSKEY_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 encipher,decipher;
    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(CMarsKeyDlg)
        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(CMarsKeyDlg)
        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_MarsKEYDLG_H__E53522C5_7B65_4B30_8B56_E53A5ACA862A__INCLUDED_)
```

Resource.h

```
//////////////////////////////////////////////////////////////
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by MarsKey.rc
//
#define IDOK2                           3
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_MARSKEY_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_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        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__FC8849F8_9494_42C6_B9A4_8E4245658A87__INCLUDED_)
#define AFX_STDAFX_H__FC8849F8_9494_42C6_B9A4_8E4245658A87__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__FC8849F8_9494_42C6_B9A4_8E4245658A87__INCLUDED_)
```

次は、CPP ファイルです。

Mars-opt.cpp

```
//////////////////////////////////////////////
/* mars.c optimized C code - copyright(c) 1998 IBM
 *
 * This code implements both the NIST high level C API (version 5)
 * and an underlying IBM defined low level (DWORD oriented) C API.
 */

/* Revisions log:
 *
 *   Aug 1999, Shai - the "tweaked" key schedule
 *   Apr 1998, Dave && Shai - new key scheduling, new sbox, NIST API
 *   Mar 1998, Shai Halevi  - adapted to the latest variant of mars
 *   Feb 1998, Dave Safford - created
 */

/* Compilation using pcg's version of gcc:
 *    gcc -Wall -pedantic -c -O6 -fomit-frame-pointer -mcpu=pentiumpro
 *        -DINTEL_GCC tests.c mars-opt.c
 *
 * Compilation using xlc on AIX:
 *    xlc -c -O3 -DAIX_XLC mars-opt.c
 *
 * Compilation using Borland C++ 5.0 from a DOS command line:
 *    bcc32 -Oi -6 -v -A -a4 -O2 -DKAT tests.c mars-opt.c
 *
 * Useful compilation defines:
 *    NO_MIX     - do not execute the mixing phases (core only)
 *    INTEL_GCC  - optimized for pentiumpro and djgpp/gcc compiler
 *    AIX_XLC    - optimized for powerpc/AIX with xlc compiler
 *    SWAP_BYTES - force endian conversion (eg __BYTE_ORDER not supported)
 *    IVT        - add intermediate values test outputs (this slows
 *                 down encryption and decryption significantly)
 */
 #include "stdafx.h"
#include "aes.h"

/* The low level mars routines are completely DWORD oriented, and
 * endian neutral. The high level NIST routines provide BYTE oriented
 * inputs and outputs, thus raising the endian issue when converting
 * between BYTEs and DWORDs. For these conversions, mars assumes
 * little endian order. On a big endian machine, we define BSWAP()
 * to do the conversions. BSWAP() is about the best you can do in C.
 * Real implementations will undoubtedly use inline ASM, as most risc
 * machines can do this in one instruction.
 *
 * Make a best guess on platform endianness; This works on linux, AIX,
 * and W95. For other platforms, you may have to manually define SWAP_BYTES
 * on a big endian machine if this guessing doesn't work.
 */
```

```
#ifdef _AIX
#define SWAP_BYTES
#else
#ifdef __linux__
#include <endian.h>
#ifdef __BYTE_ORDER
#if __BYTE_ORDER == __BIG_ENDIAN
#define SWAP_BYTES
#endif
#endif
#endif
#endif
#ifdef SWAP_BYTES
#define BSWAP(x) ¥
    ( (( x          ) << 24) | ¥
      (((x)&0xff00  ) << 8 ) | ¥
      (((x)&0xff0000) >> 8 ) | ¥
      (( x          ) >> 24)  )
#else
#define BSWAP(x) (x)
#endif

/* Some compiler optimizers will recognize certain rotation idioms,
 * and reduce them to native rotation instructions. These idioms
 * have been found to work for GCC on Intel, and XLC on RS6000.
 * Nothing seems to work on Borland C++ 5.0, although we can leave
 * out the mask on 'b'.
 */
#ifdef INTEL_GCC
#define LROTATE(a,b) (((a)<<(int)(b)) | ((a)>>(W - (int)(b))))
#define RROTATE(a,b) (((a)>>(int)(b)) | ((a)<<(W - (int)(b))))
#else
#ifdef AIX_XLC
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#ifdef __BORLANDC__
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#define LROTATE(a,b) (((a)<<(int)(b&31)) | ((a)>>(W - (int)(b&31))))
#define RROTATE(a,b) (((a)>>(int)(b&31)) | ((a)<<(W - (int)(b&31))))
#endif
#endif
#endif

////////////////////////////////////////
#define BLOCK_SIZE 128
  unsigned long t[4];
  int  b, n, i;
  DWORD x[BLOCK_SIZE/32];
  BYTE bit, bit0, ctBit, carry;
```

```
/* two 8 x 32 sboxes - stored together to save a pointer
 * these sboxes were generated with buf[3] = 0x02917d59, as
 * chosen by sbox.c, which had the following output:
 *   After 54817140, (38023 fail) new min j = 43089241 (0x2917d59)
 *   test 0 eval 0.044922 (single bit correlation)
 *   test 1 eval 0.033203 (single bit bias)
 *   test 2 eval 0.031250 (consecutive bit bias)
 *   test 3 eval 0.007813 (parity bias)
 *   test 4 eval 0.148438 (avalanche)
 */
static DWORD S[512] = {
  0x09d0c479, 0x28c8ffe0, 0x84aa6c39, 0x9dad7287,
  0x7dff9be3, 0xd4268361, 0xc96da1d4, 0x7974cc93,
  0x85d0582e, 0x2a4b5705, 0x1ca16a62, 0xc3bd279d,
  0x0f1f25e5, 0x5160372f, 0xc695c1fb, 0x4d7ff1e4,
  0xae5f6bf4, 0x0d72ee46, 0xff23de8a, 0xb1cf8e83,
  0xf14902e2, 0x3e981e42, 0x8bf53eb6, 0x7f4bf8ac,
  0x83631f83, 0x25970205, 0x76afe784, 0x3a7931d4,
  0x4f846450, 0x5c64c3f6, 0x210a5f18, 0xc6986a26,
  0x28f4e826, 0x3a60a81c, 0xd340a664, 0x7ea820c4,
  0x526687c5, 0x7eddd12b, 0x32a11d1d, 0x9c9ef086,
  0x80f6e831, 0xab6f04ad, 0x56fb9b53, 0x8b2e095c,
  0xb68556ae, 0xd2250b0d, 0x294a7721, 0xe21fb253,
  0xae136749, 0xe82aae86, 0x93365104, 0x99404a66,
  0x78a784dc, 0xb69ba84b, 0x04046793, 0x23db5c1e,
  0x46cae1d6, 0x2fe28134, 0x5a223942, 0x1863cd5b,
  0xc190c6e3, 0x07dfb846, 0x6eb88816, 0x2d0dcc4a,
  0xa4ccae59, 0x3798670d, 0xcbfa9493, 0x4f481d45,
  0xeafc8ca8, 0xdb1129d6, 0xb0449e20, 0x0f5407fb,
  0x6167d9a8, 0xd1f45763, 0x4daa96c3, 0x3bec5958,
  0xababa014, 0xb6ccd201, 0x38d6279f, 0x02682215,
  0x8f376cd5, 0x092c237e, 0xbfc56593, 0x32889d2c,
  0x854b3e95, 0x05bb9b43, 0x7dcd5dcd, 0xa02e926c,
  0xfae527e5, 0x36a1c330, 0x3412e1ae, 0xf257f462,
  0x3c4f1d71, 0x30a2e809, 0x68e5f551, 0x9c61ba44,
  0x5ded0ab8, 0x75ce09c8, 0x9654f93e, 0x698c0cca,
  0x243cb3e4, 0x2b062b97, 0x0f3b8d9e, 0x00e050df,
  0xfc5d6166, 0xe35f9288, 0xc079550d, 0x0591aee8,
  0x8e531e74, 0x75fe3578, 0x2f6d829a, 0xf60b21ae,
  0x95e8eb8d, 0x6699486b, 0x901d7d9b, 0xfd6d6e31,
  0x1090acef, 0xe0670dd8, 0xdab2e692, 0xcd6d4365,
  0xe5393514, 0x3af345f0, 0x6241fc4d, 0x460da3a3,
  0x7bcf3729, 0x8bf1d1e0, 0x14aac070, 0x1587ed55,
  0x3afd7d3e, 0xd2f29e01, 0x29a9d1f6, 0xefb10c53,
  0xcf3b870f, 0xb414935c, 0x664465ed, 0x024acac7,
  0x59a744c1, 0x1d2936a7, 0xdc580aa6, 0xcf574ca8,
  0x040a7a10, 0x6cd81807, 0x8a98be4c, 0xaccea063,
  0xc33e92b5, 0xd1e0e03d, 0xb322517e, 0x2092bd13,
  0x386b2c4a, 0x52e8dd58, 0x58656dfb, 0x50820371,
  0x41811896, 0xe337ef7e, 0xd39fb119, 0xc97f0df6,
  0x68fea01b, 0xa150a6e5, 0x55258962, 0xeb6ff41b,
  0xd7c9cd7a, 0xa619cd9e, 0xbcf09576, 0x2672c073,
```

0xf003fb3c, 0x4ab7a50b, 0x1484126a, 0x487ba9b1,
0xa64fc9c6, 0xf6957d49, 0x38b06a75, 0xdd805fcd,
0x63d094cf, 0xf51c999e, 0x1aa4d343, 0xb8495294,
0xce9f8e99, 0xbffcd770, 0xc7c275cc, 0x378453a7,
0x7b21be33, 0x397f41bd, 0x4e94d131, 0x92cc1f98,
0x5915ea51, 0x99f861b7, 0xc9980a88, 0x1d74fd5f,
0xb0a495f8, 0x614deed0, 0xb5778eea, 0x5941792d,
0xfa90c1f8, 0x33f824b4, 0xc4965372, 0x3ff6d550,
0x4ca5fec0, 0x8630e964, 0x5b3fbbd6, 0x7da26a48,
0xb203231a, 0x04297514, 0x2d639306, 0x2eb13149,
0x16a45272, 0x532459a0, 0x8e5f4872, 0xf966c7d9,
0x07128dc0, 0x0d44db62, 0xafc8d52d, 0x06316131,
0xd838e7ce, 0x1bc41d00, 0x3a2e8c0f, 0xea83837e,
0xb984737d, 0x13ba4891, 0xc4f8b949, 0xa6d6acb3,
0xa215cdce, 0x8359838b, 0x6bd1aa31, 0xf579dd52,
0x21b93f93, 0xf5176781, 0x187dfdde, 0xe94aeb76,
0x2b38fd54, 0x431de1da, 0xab394825, 0x9ad3048f,
0xdfea32aa, 0x659473e3, 0x623f7863, 0xf3346c59,
0xab3ab685, 0x3346a90b, 0x6b56443e, 0xc6de01f8,
0x8d421fc0, 0x9b0ed10c, 0x88f1a1e9, 0x54c1f029,
0x7dead57b, 0x8d7ba426, 0x4cf5178a, 0x551a7cca,
0x1a9a5f08, 0xfcd651b9, 0x25605182, 0xe11fc6c3,
0xb6fd9676, 0x337b3027, 0xb7c8eb14, 0x9e5fd030,
0x6b57e354, 0xad913cf7, 0x7e16688d, 0x58872a69,
0x2c2fc7df, 0xe389ccc6, 0x30738df1, 0x0824a734,
0xe1797a8b, 0xa4a8d57b, 0x5b5d193b, 0xc8a8309b,
0x73f9a978, 0x73398d32, 0x0f59573e, 0xe9df2b03,
0xe8a5b6c8, 0x848d0704, 0x98df93c2, 0x720a1dc3,
0x684f259a, 0x943ba848, 0xa6370152, 0x863b5ea3,
0xd17b978b, 0x6d9b58ef, 0x0a700dd4, 0xa73d36bf,
0x8e6a0829, 0x8695bc14, 0xe35b3447, 0x933ac568,
0x8894b022, 0x2f511c27, 0xddfbcc3c, 0x006662b6,
0x117c83fe, 0x4e12b414, 0xc2bca766, 0x3a2fec10,
0xf4562420, 0x55792e2a, 0x46f5d857, 0xceda25ce,
0xc3601d3b, 0x6c00ab46, 0xefac9c28, 0xb3c35047,
0x611dfee3, 0x257c3207, 0xfdd58482, 0x3b14d84f,
0x23becb64, 0xa075f3a3, 0x088f8ead, 0x07adf158,
0x7796943c, 0xfacabf3d, 0xc09730cd, 0xf7679969,
0xda44e9ed, 0x2c854c12, 0x35935fa3, 0x2f057d9f,
0x690624f8, 0x1cb0bafd, 0x7b0dbdc6, 0x810f23bb,
0xfa929a1a, 0x6d969a17, 0x6742979b, 0x74ac7d05,
0x010e65c4, 0x86a3d963, 0xf907b5a0, 0xd0042bd3,
0x158d7d03, 0x287a8255, 0xbba8366f, 0x096edc33,
0x21916a7b, 0x77b56b86, 0x951622f9, 0xa6c5e650,
0x8cea17d1, 0xcd8c62bc, 0xa3d63433, 0x358a68fd,
0x0f9b9d3c, 0xd6aa295b, 0xfe33384a, 0xc000738e,
0xcd67eb2f, 0xe2eb6dc2, 0x97338b02, 0x06c9f246,
0x419cf1ad, 0x2b83c045, 0x3723f18a, 0xcb5b3089,
0x160bead7, 0x5d494656, 0x35f8a74b, 0x1e4e6c9e,
0x000399bd, 0x67466880, 0xb4174831, 0xacf423b2,
0xca815ab3, 0x5a6395e7, 0x302a67c5, 0x8bdb446b,
0x108f8fa4, 0x10223eda, 0x92b8b48b, 0x7f38d0ee,

```
  0xab2701d4, 0x0262d415, 0xaf224a30, 0xb3d88aba,
  0xf8b2c3af, 0xdaf7ef70, 0xcc97d3b7, 0xe9614b6c,
  0x2baebff4, 0x70f687cf, 0x386c9156, 0xce092ee5,
  0x01e87da6, 0x6ce91e6a, 0xbb7bcc84, 0xc7922c20,
  0x9d3b71fd, 0x060e41c6, 0xd7590f15, 0x4e03bb47,
  0x183c198e, 0x63eeb240, 0x2ddbf49a, 0x6d5cba54,
  0x923750af, 0xf9e14236, 0x7838162b, 0x59726c72,
  0x81b66760, 0xbb2926c1, 0x48a0ce0d, 0xa6c0496d,
  0xad43507b, 0x718d496a, 0x9df057af, 0x44b1bde6,
  0x054356dc, 0xde7ced35, 0xd51a138b, 0x62088cc9,
  0x35830311, 0xc96efca2, 0x686f86ec, 0x8e77cb68,
  0x63e1d6b8, 0xc80f9778, 0x79c491fd, 0x1b4c67f2,
  0x72698d7d, 0x5e368c31, 0xf7d95e2e, 0xa1d3493f,
  0xdcd9433e, 0x896f1552, 0x4bc4ca7a, 0xa6d1baf4,
  0xa5a96dcc, 0x0bef8b46, 0xa169fda7, 0x74df40b7,
  0x4e208804, 0x9a756607, 0x038e87c8, 0x20211e44,
  0x8b7ad4bf, 0xc6403f35, 0x1848e36d, 0x80bdb038,
  0x1e62891c, 0x643d2107, 0xbf04d6f8, 0x21092c8c,
  0xf644f389, 0x0778404e, 0x7b78adb8, 0xa2c52d53,
  0x42157abe, 0xa2253e2e, 0x7bf3f4ae, 0x80f594f9,
  0x953194e7, 0x77eb92ed, 0xb3816930, 0xda8d9336,
  0xbf447469, 0xf26d9483, 0xee6faed5, 0x71371235,
  0xde425f73, 0xb4e59f43, 0x7dbe2d4e, 0x2d37b185,
  0x49dc9a63, 0x98c39d98, 0x1301c9a2, 0x389b1bbf,
  0x0c18588d, 0xa421c1ba, 0x7aa3865c, 0x71e08558,
  0x3c5cfcaa, 0x7d239ca4, 0x0297d9dd, 0xd7dc2830,
  0x4b37802b, 0x7428ab54, 0xaeee0347, 0x4b3fbb85,
  0x692f2f08, 0x134e578e, 0x36d9e0bf, 0xae8b5fcf,
  0xedb93ecf, 0x2b27248e, 0x170eb1ef, 0x7dc57fd6,
  0x1e760f16, 0xb1136601, 0x864e1b9b, 0xd7ea7319,
  0x3ab871bd, 0xcfa4d76f, 0xe31bd782, 0x0dbeb469,
  0xabb96061, 0x5370f85d, 0xffb07e37, 0xda30d0fb,
  0xebc977b6, 0x0b98b40f, 0x3a4d0fe6, 0xdf4fc26b,
  0x159cf22a, 0xc298d6e2, 0x2b78ef6a, 0x61a94ac0,
  0xab561187, 0x14eea0f0, 0xdf0d4164, 0x19af70ee
};

/*
 * The following implements Intermediate Values Tests Macros.
 * Since internal words are always kept little-endian, always
 * swap bytes before displaying.
 */
#ifdef IVT
int ivt_debug = 0;
FILE *ivt_fp;
int ivt_l = 0;
#define IVTSWAP(x) ¥
    ( (( x          ) << 24) | ¥
      (((x)&0xff00  ) << 8 ) | ¥
      (((x)&0xff0000) >> 8 ) | ¥
      (( x          ) >> 24)  )
#define IVT_DEBUG(a,b,c,d) ¥
```

```
    if (ivt_debug) ¥
        fprintf(ivt_fp,"IV%d=%8.8lx %8.8lx %8.8lx %8.8lx¥n",ivt_l++,¥
        IVTSWAP(a),IVTSWAP(b),IVTSWAP(c),IVTSWAP(d));
#else
#define IVT_DEBUG(a,b,c,d)
#endif

/*****************************************************************************
 *
 *   Low Level key setup, block encrypt and decrypt routines.
 *   For efficiency, these are WORD oriented. The high level NIST
 *   routines provide BYTE oriented interfaces, with ENDIAN conversion.
 *
 *****************************************************************************/
/* if multiplication subkey k has 10 0's or 10 1's, mask in a fixing value */
static DWORD fix_subkey(DWORD k, DWORD r)
{
    /* the mask words come from S[265]..S[268], as chosen by index.c */
    DWORD *B = &S[265];
    DWORD m1, m2;
    int i;

    i = k & 3;            /* store the least two bits of k */
    k |= 3;               /* and then mask them away       */

    /* we look for 9 consequtive 1's in m1 */
    m1 = (~k) ^ (k<<1);   /* for i > 1, m1_i = 1 iff k_i = k_{i-1} */
    m2 = m1 & (m1 << 1);  /* m2_i = AND (m1_i, m1_{i-1})   */
    m2 &= m2 << 2;        /* m2_i = AND (m1_i...m1_{i-3})  */
    m2 &= m2 << 4;        /* m2_i = AND (m1_i...m1_{i-7})  */
    m2 &= m1 << 8;        /* m2_i = AND (m1_i...m1_{i-8})  */
    m2 &= 0xfffffe00;     /* mask out the low 9 bits of m2 */
    /* for i = 9...31, m2_i = 1 iff k_i = ... = k_{i-9} */

    /* if m2 is zero, k was good, so return */
    if (!m2)
        return(k);

    /* need to fix k: we copy each 1 in m2 to the nine bits to its right */
    m1 = m2 | (m2 >> 1);  /* m1_i = AND (m2_i, m2_{i+1})   */
    m1 |= m1 >> 2;        /* m1_i = AND (m2_i...m2_{i+3})  */
    m1 |= m2 >> 4;        /* m1_i = AND (m2_i...m2_{i+4})  */
    m1 |= m1 >> 5;        /* m1_i = AND (m2_i...m2_{i+9})  */
    /* m1_i = 1 iff k_i belongs to a sequence of ten 0's or ten 1's */

    /* we turn off the two lowest bits of M, and also every bit
     * M_i such that k_i is not equal to both k_{i-1} and k_{i+1}
     */
    m1 &= ((~k)^(k<<1)) & ((~k)^(k>>1)) & 0x7ffffffc;

    /* and finally pick a pattern, rotate it,
     * and xor it into k under the control of the mask m1
```

```
     */
     k ^= LROTATE(B[i], r) & m1;

     return(k);
}


/* setup a mars key schedule
 *
 * n  (input) is the number of words in the key
 * kp (input) is a pointer to the array of key words
 * ep (output) is a pointer to the array of EKEY_WORDS expanded subkey WORDs
 */
int mars_setup(int n, DWORD *kp, DWORD *ep)
{
    DWORD T[15] = {0};
    int i,j,t;

    /* check key length */
    if ((n<4)||(n>14))
        return(BAD_KEY_MAT);

    /* initialize the T[] array with key data */
    for (i=0; i<n; i++)
        T[i] = kp[i];
    T[n] = n;
    for (i=n+1; i<15; i++)
        T[i] = 0;

    /* Four iterations, each one computing 10 words of the array */
    for (j=0; j<4; j++) {
        DWORD w;

        /* Linear transformation */
        w = T[8] ^ T[13];         T[0] ^= LROTATE(w,3) ^ j;
        w = T[9] ^ T[14];         T[1] ^= LROTATE(w,3) ^ (4+j);
        for (i=2; i<7; i++) {
            w = T[i+8] ^ T[i-2];
            T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
        }
        for (i=7; i<15; i++) {
            w = T[i-7] ^ T[i-2];
            T[i] ^= LROTATE(w,3) ^ ((i<<2)+j);
        }

        /* Four stirring rounds */
        for (t=0; t<4; t++){
            /* stir with full type-1 s-box rounds */
            T[0] += S[ T[14]&511 ];
            T[0] = LROTATE(T[0],9);
            for (i=1; i<15; i++) {
                T[i] += S[ T[i-1]&511 ];
```

```
                T[i] = LROTATE(T[i],9);
            }
        }

        /* copy subkeys to mars_ctx, with swapping around */

#define SWAP(i) (ep[(10*j)+i] = T[(i*4)%15])

        SWAP(0); SWAP(1); SWAP(2); SWAP(3); SWAP(4);
        SWAP(5); SWAP(6); SWAP(7); SWAP(8); SWAP(9);
    }

    /* check and fix all multiplication subkeys */
    for (i=NUM_DATA+1;i<(EKEY_DWORDS-NUM_DATA);i+=2)
        ep[i] = fix_subkey(ep[i], ep[i-1]);

    return(TRUE);
}


#define MixForwardRound(d1,d2,d3,d4) ¥
    d2 ^= sp[d1&255];       ¥
    y = RROTATE(d1,8);      ¥
    z = RROTATE(d1,16);     ¥
    d1 = RROTATE(d1,24);    ¥
    d2 += sp[(y&255)+256]; ¥
    d3 += sp[z&255];        ¥
    d4 ^= sp[(d1&255)+256];

#define MixBackwardsRound(d1,d2,d3,d4) ¥
    d2 ^= sp[(d1&255)+256];¥
    y = LROTATE(d1,8);      ¥
    z = LROTATE(d1,16);     ¥
    d1 = LROTATE(d1,24);    ¥
    d3 -= sp[y&255];        ¥
    d4 -= sp[(z&255)+256]; ¥
    d4 ^= sp[d1&255];

#define CoreRound(d1,d2,d3,d4,i) ¥
    y = d1;             ¥
    d1 += ep[i];        ¥
    y = LROTATE(y,13); ¥
    z = d1;             ¥
    tmp = y;            ¥
    y *= ep[(i)+1];     ¥
    z &= 511;           ¥
    z = sp[z];          ¥
    y = LROTATE(y,5);  ¥
    z ^= y;             ¥
    d1 = LROTATE(d1,y);¥
    y = LROTATE(y,5);  ¥
    d3 += d1;           ¥
```

```
    z ^= y;             ¥
    z = LROTATE(z,y);   ¥
    d2 += z;            ¥
    d4 ^= y;            ¥
    d1 = tmp;

#define InvCoreRound(d1, d2, d3, d4, i) ¥
    y = d1;             ¥
    d1 = RROTATE(d1,13);¥
    y *= ep[(i)+1];     ¥
    tmp = d1;           ¥
    d1 += ep[i];        ¥
    z = d1;             ¥
    y = LROTATE(y,5);   ¥
    z &= 511;           ¥
    z = sp[z];          ¥
    d1 = LROTATE(d1,y); ¥
    z ^= y;             ¥
    y = LROTATE(y,5);   ¥
    d3 -= d1;           ¥
    z ^= y;             ¥
    d1 = tmp;           ¥
    z = LROTATE(z,y);   ¥
    d4 ^= y;            ¥
    d2 -= z;

/* The basic mars encryption: (ep is the expanded key array) */
void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep)
{
    register DWORD a,b,c,d,y,z;
    DWORD tmp;
    DWORD *sp = S;

    a = in[0];
    b = in[1];
    c = in[2];
    d = in[3];
    IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
    /* first, add subkeys to all input data words */
    a += ep[0];
    b += ep[1];
    c += ep[2];
    d += ep[3];
    IVT_DEBUG(a,b,c,d);

    /* then do eight mixing rounds */
    MixForwardRound(a,b,c,d);
    a += d;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(b,c,d,a);
```

```
    b += c;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);

    MixForwardRound(a,b,c,d);
    a += d;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(b,c,d,a);
    b += c;
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(a,b,c,d);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);
#endif

    /* then sixteen mars encrypting rounds             *
     * (eight in forward- and eight in backwards-mode) */

    CoreRound(a,b,c,d,4);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,c,d,a,6);
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,d,a,b,8);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,a,b,c,10);
    IVT_DEBUG(a,b,c,d);

    CoreRound(a,b,c,d,12);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,c,d,a,14);
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,d,a,b,16);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,a,b,c,18);
    IVT_DEBUG(a,b,c,d);

    CoreRound(a,d,c,b,20);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,a,d,c,22);
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,b,a,d,24);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,c,b,a,26);
    IVT_DEBUG(a,b,c,d);

    CoreRound(a,d,c,b,28);
    IVT_DEBUG(a,b,c,d);
    CoreRound(b,a,d,c,30);
```

```
    IVT_DEBUG(a,b,c,d);
    CoreRound(c,b,a,d,32);
    IVT_DEBUG(a,b,c,d);
    CoreRound(d,c,b,a,34);
    IVT_DEBUG(a,b,c,d);

#ifndef NO_MIX
    /* then do eight inverse-mixing rounds */
    MixBackwardsRound(a,b,c,d);
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(b,c,d,a);
    c -= b;
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(c,d,a,b);
    d -= a;
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);

    MixBackwardsRound(a,b,c,d);
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(b,c,d,a);
    c -= b;
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(c,d,a,b);
    d -= a;
    IVT_DEBUG(a,b,c,d);
    MixBackwardsRound(d,a,b,c);
    IVT_DEBUG(a,b,c,d);

    /* subtract final subkeys */
    a -= ep[2*NUM_ROUNDS+4];
    b -= ep[2*NUM_ROUNDS+5];
    c -= ep[2*NUM_ROUNDS+6];
    d -= ep[2*NUM_ROUNDS+7];
    IVT_DEBUG(a,b,c,d);
#endif

    out[0] = a;
    out[1] = b;
    out[2] = c;
    out[3] = d;
}

/* mars decryption is simply encryption in reverse */
void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep)
{
    register DWORD a,b,c,d,y,z;
    DWORD tmp;
    DWORD *sp = S;

    d = in[0];
```

```
    c = in[1];
    b = in[2];
    a = in[3];
    IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
    /* first, add subkeys to all input data DWORDs */
    a += ep[2*NUM_ROUNDS+7];
    b += ep[2*NUM_ROUNDS+6];
    c += ep[2*NUM_ROUNDS+5];
    d += ep[2*NUM_ROUNDS+4];
    IVT_DEBUG(d,c,b,a);

    /* then do eight mixing rounds */
    MixForwardRound(a,b,c,d);
    IVT_DEBUG(d,c,b,a);
    a += d;
    MixForwardRound(b,c,d,a);
    IVT_DEBUG(d,c,b,a);
    b += c;
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(d,c,b,a);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(d,c,b,a);

    MixForwardRound(a,b,c,d);
    IVT_DEBUG(d,c,b,a);
    a += d;
    MixForwardRound(b,c,d,a);
    IVT_DEBUG(d,c,b,a);
    b += c;
    MixForwardRound(c,d,a,b);
    IVT_DEBUG(d,c,b,a);
    MixForwardRound(d,a,b,c);
    IVT_DEBUG(d,c,b,a);
#endif

    /* then sixteen mars decrypting rounds             *
     * (eight in forward- and eight in backwards-mode) */

    InvCoreRound(a,b,c,d,34);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(b,c,d,a,32);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(c,d,a,b,30);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(d,a,b,c,28);
    IVT_DEBUG(d,c,b,a);

    InvCoreRound(a,b,c,d,26);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(b,c,d,a,24);
```

```
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(c,d,a,b,22);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(d,a,b,c,20);
    IVT_DEBUG(d,c,b,a);

    InvCoreRound(a,d,c,b,18);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(b,a,d,c,16);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(c,b,a,d,14);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(d,c,b,a,12);
    IVT_DEBUG(d,c,b,a);

    InvCoreRound(a,d,c,b,10);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(b,a,d,c,8);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(c,b,a,d,6);
    IVT_DEBUG(d,c,b,a);
    InvCoreRound(d,c,b,a,4);
    IVT_DEBUG(d,c,b,a);

#ifndef NO_MIX
    /* then do eight inverse-mixing rounds */
    MixBackwardsRound(a,b,c,d);
    IVT_DEBUG(d,c,b,a);
    MixBackwardsRound(b,c,d,a);
    IVT_DEBUG(d,c,b,a);
    c -= b;
    MixBackwardsRound(c,d,a,b);
    IVT_DEBUG(d,c,b,a);
    d -= a;
    MixBackwardsRound(d,a,b,c);
    IVT_DEBUG(d,c,b,a);

    MixBackwardsRound(a,b,c,d);
    IVT_DEBUG(d,c,b,a);
    MixBackwardsRound(b,c,d,a);
    IVT_DEBUG(d,c,b,a);
    c -= b;
    MixBackwardsRound(c,d,a,b);
    IVT_DEBUG(d,c,b,a);
    d -= a;
    MixBackwardsRound(d,a,b,c);
    IVT_DEBUG(d,c,b,a);

    /* subtract final subkeys */
    a -= ep[3];
    b -= ep[2];
    c -= ep[1];
```

```c
    d -= ep[0];
    IVT_DEBUG(d,c,b,a);
#endif

    out[0] = d;
    out[1] = c;
    out[2] = b;
    out[3] = a;
}

/*************************************************************************
 *
 *   NIST High Level key setup, block encrypt and decrypt routines
 *
 ************************************************************************/

/* table for rapid, case insensitive hex conversion */
BYTE hex[256] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0, 0, 0, 0, 0,
    0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

/* NIST defined high level key setup */
int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial)
{
    DWORD tmpkey[EKEY_DWORDS];
    int i,j;

    /* sanity check pointers */
    if (key == NULL)
        return(BAD_KEY_INSTANCE);
    if (keyMaterial == NULL)
        return(BAD_KEY_MAT);

    /* save parameters into keyInstance */
    key->direction = direction;
    key->keyLen = keyLen;
    for (i=0;i<keyLen/4;i++)
        key->keyMaterial[i] = keyMaterial[i];
```

```
    key->keyMaterial[MAX_KEY_SIZE] = '¥0';

    /* convert ascii keyMaterial to BYTEs */
    for(i=0,j=0;i<keyLen/4;i+=2,j++)
        ((BYTE *)tmpkey)[j] = (BYTE)((hex[(int)keyMaterial[i]]<<4) |
                                     hex[(int)keyMaterial[i+1]]);
#   ifdef SWAP_BYTES
        /* BSWAP the input key DWORDs */
        for (i=0;i<keyLen/W;i++)
            tmpkey[i] = BSWAP(tmpkey[i]);
#   endif

    /* call low level mars setup routine */
    return(mars_setup(keyLen/32,tmpkey,key->E));
}

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

    /* sanity check pointers */
    if (cipher == NULL)
        return(BAD_CIPHER_MODE);

    /* save cipher parameters */
    cipher->mode = mode;

    /* handle IV */
    if((mode == MODE_CBC)||(mode == MODE_CFB1)) {
        if(IV == NULL)
            return(BAD_CIPHER_MODE);
        /* convert ascii IV to BYTEs */
        for(i=0,j=0;j<MAX_IV_SIZE;i+=2,j++)
            cipher->IV[j] = (BYTE)((hex[(int)IV[i]]<<4) | hex[(int)IV[i+1]]);
        /* copy BYTE IV to DWORD CIV, with conversion if necessary */
        for(i=0;i<NUM_DATA;i++)
            cipher->CIV[i] = BSWAP(((DWORD *)&cipher->IV)[i]);
    }
    return(TRUE);
}

/* this assumes the input length is a multiple of 128 bits */
int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    DWORD tmp[4];
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
```

```
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_encrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                tmp[0] = BSWAP(*(DWORD *)(input+i+0)) ^ cipher->CIV[0];
                tmp[1] = BSWAP(*(DWORD *)(input+i+4)) ^ cipher->CIV[1];
                tmp[2] = BSWAP(*(DWORD *)(input+i+8)) ^ cipher->CIV[2];
                tmp[3] = BSWAP(*(DWORD *)(input+i+12)) ^ cipher->CIV[3];
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                tmp[0] = *(DWORD *)(input+i+0) ^ cipher->CIV[0];
                tmp[1] = *(DWORD *)(input+i+4) ^ cipher->CIV[1];
                tmp[2] = *(DWORD *)(input+i+8) ^ cipher->CIV[2];
                tmp[3] = *(DWORD *)(input+i+12) ^ cipher->CIV[3];
                mars_encrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                cipher->CIV[0] = *(DWORD *)(outBuffer+i+0);
                cipher->CIV[1] = *(DWORD *)(outBuffer+i+4);
                cipher->CIV[2] = *(DWORD *)(outBuffer+i+8);
                cipher->CIV[3] = *(DWORD *)(outBuffer+i+12);
#           endif
        }
    }
    else if(cipher->mode == MODE_CFB1) {
                cipher->mode = MODE_ECB; /* do encryption in ECB */
                for (int 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;
        /* Input is one bit (lsb of first byte).
         * Encrypt IV with key, xor input with msb of encrypted IV,
         * and then feed output cipher bit into lsb of IV.
         */
                /*
        DWORD ECIV[4];

        if(inputLen != 1)
            return(BAD_CIPHER_MODE);

        mars_encrypt(cipher->CIV, ECIV, key->E);
        outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
        cipher->CIV[0] = (cipher->CIV[0]<<1)|(cipher->CIV[1] & 0x80000000);
        cipher->CIV[1] = (cipher->CIV[1]<<1)|(cipher->CIV[2] & 0x80000000);
        cipher->CIV[2] = (cipher->CIV[2]<<1)|(cipher->CIV[3] & 0x80000000);
        cipher->CIV[3] = (cipher->CIV[3]<<1)|(DWORD)outBuffer[0];*/
    }
    else
        return(BAD_CIPHER_MODE);

    return(inputLen);
}

/* this assumes the input length is a multiple of 128 bits */
int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
                 int inputLen, BYTE *outBuffer)
{
    int i;

    if (cipher->mode == MODE_ECB) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0));
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4));
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8));
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12));
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
```

```
#           endif
        }
    }
    else if(cipher->mode == MODE_CBC) {
        for(i=0;i<inputLen/8;i+=16){
#           ifdef SWAP_BYTES
                DWORD tmp[4];
                tmp[0] = BSWAP(*(DWORD *)(input+i+0));
                tmp[1] = BSWAP(*(DWORD *)(input+i+4));
                tmp[2] = BSWAP(*(DWORD *)(input+i+8));
                tmp[3] = BSWAP(*(DWORD *)(input+i+12));
                mars_decrypt(tmp,(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) = BSWAP(*(DWORD *)(outBuffer+i+0)
                    ^ cipher->CIV[0]);
                *(DWORD *)(outBuffer+i+4) = BSWAP(*(DWORD *)(outBuffer+i+4)
                    ^ cipher->CIV[1]);
                *(DWORD *)(outBuffer+i+8) = BSWAP(*(DWORD *)(outBuffer+i+8)
                    ^ cipher->CIV[2]);
                *(DWORD *)(outBuffer+i+12) = BSWAP(*(DWORD *)(outBuffer+i+12)
                    ^ cipher->CIV[3]);
                cipher->CIV[0] = tmp[0];
                cipher->CIV[1] = tmp[1];
                cipher->CIV[2] = tmp[2];
                cipher->CIV[3] = tmp[3];
#           else
                mars_decrypt((DWORD *)(input+i),(DWORD *)(outBuffer+i),key->E);
                *(DWORD *)(outBuffer+i+0) ^= cipher->CIV[0];
                *(DWORD *)(outBuffer+i+4) ^= cipher->CIV[1];
                *(DWORD *)(outBuffer+i+8) ^= cipher->CIV[2];
                *(DWORD *)(outBuffer+i+12) ^= cipher->CIV[3];
                cipher->CIV[0] = *(DWORD *)(input+i+0);
                cipher->CIV[1] = *(DWORD *)(input+i+4);
                cipher->CIV[2] = *(DWORD *)(input+i+8);
                cipher->CIV[3] = *(DWORD *)(input+i+12);
#           endif
        }
    }
    else if(cipher->mode == 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);
                        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;


        /* Input is one bit (lsb of first byte).
         * Encrypt IV with key, xor input with msb of encrypted IV,
         * and then feed input cipher bit into lsb of IV.
         */
                /*
        DWORD ECIV[4];

        if(inputLen != 1)
            return(BAD_CIPHER_MODE);

        mars_encrypt(cipher->CIV, ECIV, key->E);
        outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
        cipher->CIV[0] = (cipher->CIV[0]<<1)|(cipher->CIV[1] & 0x80000000);
        cipher->CIV[1] = (cipher->CIV[1]<<1)|(cipher->CIV[2] & 0x80000000);
        cipher->CIV[2] = (cipher->CIV[2]<<1)|(cipher->CIV[3] & 0x80000000);
        cipher->CIV[3] = (cipher->CIV[3]<<1)|(DWORD)(input[0]&1);
*/
    }
    else
        return(BAD_CIPHER_MODE);

    return(inputLen);
}
```

Marskey.cpp

```
////////////////////////////////////////////////
// MarsKey.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 "MarsKey.h"
#include "MarsKeyDlg.h"

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

```
/////////////////////////////////////////////////////////////////////////////
// CMarsKeyApp

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

/////////////////////////////////////////////////////////////////////////////
// CMarsKeyApp クラスの構築

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

CMarsKeyApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CMarsKeyApp クラスの初期化

BOOL CMarsKeyApp::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

	CMarsKeyDlg 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;
}
```

marskeyDlg.cpp

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

#include "stdafx.h"
#include "MarsKey.h"
#include "MarsKeyDlg.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()

/////////////////////////////////////////////////////////////////////////////
// CMarsKeyDlg ダイアログ
/////////////////////////////////////////////////////////////////////////////
// CMarsKeyDlg ダイアログ
```

```
CMarsKeyDlg::CMarsKeyDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CMarsKeyDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CMarsKeyDlg)
	i_KeyLen = 3;
	i_Mode = 0;
	s_fname1 = "MarskeyEC.bin";
	s_fname2 = "MarskeyDC.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 CMarsKeyDlg::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;
	if(i_KeyLen == 3) ofs << 448 << 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;
	if(i_KeyLen == 3) ofs2 << 448 << endl;
	ofs2 << s_key2 << endl;
```

```
        ofs2 << s_cipherinit << endl;
        ofs2.close();
}

void CMarsKeyDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CMarsKeyDlg)
        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, decryptdata_file, 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(CMarsKeyDlg, CDialog)
        //{{AFX_MSG_MAP(CMarsKeyDlg)
        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 CMarsKeyDlg::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 CMarsKeyDlg::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 CMarsKeyDlg::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_RADIO4) ){
                n= 16;
        }
        if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) ){
                n= 24;
        }
        if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) ){
                n= 32;
        }
        if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) ){
                n= 56;
        }

        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 CMarsKeyDlg::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_RADIO4) )
        {
                n= 16;
                keylength = "128";
        }
        if(IDC_RADIO2 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 24;
                keylength = "192";
        }
        if(IDC_RADIO3 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 32;
                keylength = "256";
```

```
        }
        if(IDC_RADIO4 == GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO4) )
        {
                n= 56;
                keylength = "448";
        }

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

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

```
        }

        makeKey(&keyI, DIR_DECRYPT,  n*8, (char*)key2 );

    rc=blockDecrypt(&decipher, &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;
}


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

BOOL CMarsKeyDlg::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 CMarsKeyDlg::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 CMarsKeyDlg::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 CMarsKeyDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

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

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
//              MarsKey.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。