Bitomaを利用した暗号ソフト

(C) 宇山　靖政

2012.04.14

暗号に楽しさを加えようと、どんなファイルもビットマップに変換するソフトを作りました。

暗号化鍵は、128,192,256 ビットから選択できます。

このソフトを利用して、メールデータを暗号化してから送信します。拡張子は、現在は変更していませんが、最後にこのソフトで暗号化されて送られてきたデータは、拡張子を bmp にすれば、画像として表示されます。ほかのメールソフトで受信すれば単に画像ファイルの拡張子が不適切なものが送られてきたということになります。

ただし、送信のときに、拡張子を bmp にすることも現在検討中です。その場合はまた、ソースコードを公開いたします。

暗号化の過程では乱数を利用して、同一データも暗号化する時間が異なれば結果も異なりますので、暗号化されたものが同一化データか否かは判定が難しくなっています。

2種類のソースコードを掲載いたします。

1. BmpEC のソースコード
2. BmpDC のソースコード

これに関しては、著作権を主張いたします。内容を明らかにして、公知の技術とするために公開いたします。

コンソールタイプの暗号で、暗号化鍵と復号化鍵は次のようなファイルです。

暗号化鍵のファイル(256 ビットのもの)は

BmpkeyEC.bin
//
1
256
421DE5372FAD740D681448B8329661CA286FE169261F13486606ADD7E93052B7

復号化鍵のファイルは

BmpkeyDC.bin
//
1
256
421DE5372FAD740D681448B8329661CA286FE169261F13486606ADD7E93052B7

で、どちらも同じテキストファイルです。

最初に、BmpEC のソースコード

ヘッダーファイルは

BmpEC.h

```
//////////////////////////////////////////
/***
 * BmpEC.h
 * Bmp1 暗号関数
 */

// 戻り値
#define NOERROR                  0                    // エラーなし
#define NOTENOUGHMEMORY  -1                   // メモリー不足
#define ACCESSERROR              -2                    // アクセスエラー
#define MATHERROR                -3                    // 数学的な誤り
#define KEYLENGTHERROR   -4                   // 鍵長不正
#define OTHERERROR               -5                    // その他のエラー

// 定数
#define MINKEYLENGTH     32                   // 最低鍵長（ビット）


typedef unsigned char uchar;
typedef unsigned short ushort;

// 暗号化　復号化
void bmp1(char* St1, char* St2, char* St3);
```

Stdafx.h

```
//////////////////////////////////////////////////////
// stdafx.h : 標準のシステムインクルードファイルの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。
//

#pragma once

#ifndef _WIN32_WINNT             // Windows XP 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0501      // これをWindows の他のバージョン向けに適切な値に変更してください。
#endif
```

```
#include <stdio.h>
#include <tchar.h>



// TODO: プログラムに必要な追加ヘッダーをここで参照してください。
```

Cppファイルは

Bmp1.cpp

```
////////////////////////////////////////
// Misty1.cpp : コンソールアプリケーション用のエントリポイントの定義
//

#include "stdafx.h"

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>


typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned long DWORD;

        FILE* keyfile;
        FILE* srcfile;
        FILE* dstfile;

void bmp1(char* keyfn, char* pt, char* ct)  // 暗号化
{
        int mode,klen; //,blen,rc;
        char c_mode[4], c_klen[8], c_key[64];

int nsize, fsize, sidesize;
char FName[256];
unsigned int j, k, l, jj;
int i, mn;
unsigned char tmpch[16];
unsigned char tmprbuf[16];
char key[64];//32];

unsigned char bmpHeader[54] = {
'B', 'M', /* [ 0] ファイルタイプ*/
54, 4, 0, 0, /* [ 2] ファイルサイズ 54+4*16*16=1078*/
0, 0, 0, 0, /* [ 6] 予約*/
54, 0, 0, 0, /* [10] ビットマップデータのシーク位置*/
```

```
40, 0, 0, 0, /* [14] ここから始まるヘッダの高さ*/
16, 0, 0, 0, /* [18] ビットマップの幅*/
16, 0, 0, 0, /* [22] ビットマップの高さ*/
0x01, 0, /* [26] プレーン数*/
32, 0, /* [28] 1ピクセルあたりのビット数（課題がバイト指定されていたのでbitに変更）*/
0, 0, 0, 0, /* [30] 圧縮タイプ*/
0, 1, 0, 0, /* [34] ビットマップデータの長さ16*16=256*/
0, 0, 0, 0, /* [38] 水平解像度(px/m) */
0, 0, 0, 0, /* [42] 垂直解像度(px/m) */
0, 0, 0, 0, /* [46] カラーインデックス数*/
0, 0, 0, 0, /* [50] 重要なカラーインデックス数*/
};

        if((keyfile = fopen(keyfn, "rt")) == NULL){
//              printf("Can not find key file for encryption. ¥n");
                return;
        }
        fgets( c_mode, 4, keyfile );
        fgets( c_klen, 8, keyfile );
        fgets( c_key, 64, keyfile );

        fclose(keyfile);

        strcpy(key, c_key);

        mode = atoi(c_mode);
        klen = atoi(c_klen);

        mn = 8;
        if(klen == 128){ mn = 16; }
        if(klen == 192){ mn = 24; }
        if(klen == 256){ mn = 32; }

        if((srcfile = fopen(pt, "rb")) == NULL){
                printf("Can not open plane file. ¥n");
                return;
        }
/*
        strcpy(FName, ".¥¥Encrypt¥¥");
        strcat(FName, ct);
        if((dstfile = fopen(FName, "wb")) == NULL){
            if(0 == _mkdir(".¥¥Encrypt")){
                        if((dstfile = fopen(FName, "wb")) == NULL){
                                return;
                        }
                }
                else{
                        return;
                }
        }
*/
        if((dstfile = fopen(ct, "wb")) == NULL){
```

```
            printf("Can not open crypted file. ¥n");
            return;
        }

        srand( (unsigned)time(NULL) );  //in constracter

        strcpy(FName, pt);// = srcfile.GetFileName();
        nsize = strlen(FName); //.GetLength();// length of file name.
// 平文
        fseek(srcfile, 0, SEEK_END);
        long filelen = ftell(srcfile);
        fseek(srcfile,0,0);
        fsize = filelen; //fsize = srcfile.GetLength();// as char 8 bit
        sidesize = 1 + (int)sqrt((double)(4+1+(fsize/2)+1+(nsize/2)));// as short int 16 bit

        // Write the header of bitmap file.
        *((DWORD*)(&(bmpHeader[2]))) = 54+4*sidesize*sidesize;
        *((DWORD*)(&(bmpHeader[18]))) = sidesize;
        *((DWORD*)(&(bmpHeader[22]))) = sidesize;
        *((DWORD*)(&(bmpHeader[34]))) = sidesize*sidesize;

        fwrite( bmpHeader,sizeof(char),54, dstfile); //dstfile.Write(bmpHeader, 54);

        j = rand(); // 16 bits
        k = nsize & 0x0000ffff;
        jj = (j<<16) + (j^k);
        *((unsigned int*)&tmpch) = jj;//32 bit
        fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
        j = rand(); // 16 bits
        k = nsize & 0xffff0000;
        jj = (j<<16) + (j^(k>>16));
        *((unsigned int*)&tmpch) = jj;//32 bit
        fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit

        j = rand(); // 16 bits
        k = fsize & 0x0000ffff;
        jj = (j<<16) + (j^k);
        *((unsigned int*)&tmpch) = jj;//32 bit
        fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
        j = rand(); // 16 bits
        k = fsize & 0xffff0000;
        jj = (j<<16) + (j^(k>>16));
        *((unsigned int*)&tmpch) = jj;//32 bit
        fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit

        for(i=0; i<nsize/2 ; i++){
            j = rand(); // 16 bits
            k = (unsigned char)FName[2*i];
            l = (unsigned char)FName[2*i+1];
            jj = (j<<16) + (j^((k<<8)+ l));
            *((unsigned int*)&tmpch) = jj;//32 bit
            fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
```

```
	}
	if(nsize%2 == 1){
		j = rand(); // 16 bits
		k = (unsigned char)FName[nsize-1];
		jj = (j<<16) + (j^((k<<8)+ 0));
		*((unsigned int*)&tmpch) = jj;//32 bit
		fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
	}
	if(nsize%2 == 0){
		j = rand(); // 16 bits
		k = 0;
		jj = (j<<16) + (j^((k<<8)+ 0));
		*((unsigned int*)&tmpch) = jj;//32 bit
		fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
	}

	for(i=4+1+nsize/2; i<sidesize*sidesize ; i++){
		j = rand(); // 16 bits
		if(1 == fread(tmprbuf, sizeof(char), 1, srcfile)){//1 == srcfile.Read(tmprbuf,1)){
			k = (unsigned char)tmprbuf[0];
			k ^= (unsigned char)key[(i-(4+1+nsize/2))%mn];
		}
		else{k = 0;}
		if(1 == fread(tmprbuf, sizeof(char), 1, srcfile)){//1 == srcfile.Read(tmprbuf,1)){
			l = (unsigned char)tmprbuf[0];
			l ^= (unsigned char)key[(i-(4+1+nsize/2))%mn];
			jj = (j<<16) + (j^((k<<8)+ l));
		}
		else{jj = (j<<16) + (j^((k<<8)+ 0));}
		*((unsigned int*)&tmpch) = jj;//32 bit
		fwrite( tmpch,sizeof(char),4, dstfile); //dstfile.Write(tmpch, 4);//32 bit
	}

	fclose(srcfile);//.Close();
	fclose(dstfile);//.Close();

}
```

Bmpec.cpp

```
////////////////////////////////////////////
// BmpEC.cpp : コンソールアプリケーションのエントリポイントを定義します。
//


#include "stdafx.h"

#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

```
#include "BmpEC.h"

using namespace std;


// メイン
int main(                                                   // 正常終了時:0、異常時:1
                int argc,                                   // 引数の数
                char** argv )                           // 引数へのポインタ
{
        printf( "Start!¥n" );

          // 引数チェック
        if( argc != 4 ){                            // 使い方の誤り
                printf( "引数の数が不正ですから異常終了します。¥n" );
                exit(1);
        }

    // 暗号化実行
            // 鍵file    平文    暗文
        bmp1( argv[1], argv[2], argv[3] );

        printf( "End!¥n" );
        return 0;
}
```

Stdafx.cpp

```
////////////////////////////////////////////////
// stdafx.cpp : 標準インクルードBmpEC.pch のみを
// 含むソースファイルは、プリコンパイル済みヘッダーになります。
// stdafx.obj にはプリコンパイル済み型情報が含まれます。

#include "stdafx.h"

// TODO: このファイルではなく、STDAFX.H で必要な
// 追加ヘッダーを参照してください。
```

次に、BmpDC のソースコード

BmpDC.h

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

```
/***
 * BmpEC.h
 * Bmp 暗号関数
 */

// 戻り値
#define NOERROR                  0                      // エラーなし
#define NOTENOUGHMEMORY  -1                     // メモリー不足
#define ACCESSERROR              -2                     // アクセスエラー
#define MATHERROR                -3                     // 数学的な誤り
#define KEYLENGTHERROR   -4                     // 鍵長不正
#define OTHERERROR               -5                     // その他のエラー

// 定数
#define MINKEYLENGTH     32                     // 最低鍵長（ビット）


typedef unsigned char uchar;
typedef unsigned short ushort;

// 暗号化　復号化
void bmp2(char* St1, char* St2, char* St3);
```

Stdafx.h

```
////////////////////////////////////////////////////////
// stdafx.h : 標準のシステムインクルードファイルの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。
//

#pragma once

#ifndef _WIN32_WINNT             // Windows XP 以降のバージョンに固有の機能の使用を許可します。
#define _WIN32_WINNT 0x0501      // これをWindows の他のバージョン向けに適切な値に変更してください。
#endif

#include <stdio.h>
#include <tchar.h>



// TODO: プログラムに必要な追加ヘッダーをここで参照してください。
```

Cppファイルは

Bmp2.cpp

```
//////////////////////////////////////
// Misty1.cpp : コンソールアプリケーション用のエントリポイントの定義
//

#include "stdafx.h"

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
#include <direct.h>


typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned long DWORD;


        FILE* keyfile;
        FILE* srcfile;
        FILE* dstfile;


void bmp2(char* keyfn, char* srcfn, char* dstfn)  // 復号化
{
        int mode,klen; //,blen,rc;
        char c_mode[4], c_klen[8], c_key[64], folderfile[256];

int nsize, fsize, sidesize;
char FName[256];
unsigned int j, k, jj;

int i, mn;
unsigned char tmpch[16];
unsigned char tmprbuf[16];
char key[64];//32];
char* pfName;
int q;


unsigned char bmpHeader[54] = {
'B', 'M', /* [ 0] ファイルタイプ*/
54, 4, 0, 0, /* [ 2] ファイルサイズ 54+4*16*16=1078*/
0, 0, 0, 0, /* [ 6] 予約*/
54, 0, 0, 0, /* [10] ビットマップデータのシーク位置*/
40, 0, 0, 0, /* [14] ここから始まるヘッダの高さ*/
16, 0, 0, 0, /* [18] ビットマップの幅*/
16, 0, 0, 0, /* [22] ビットマップの高さ*/
0x01, 0, /* [26] プレーン数*/
32, 0, /* [28] 1ピクセルあたりのビット数（課題がバイト指定されていたのでbitに変更）*/
0, 0, 0, 0, /* [30] 圧縮タイプ*/
```

```
0, 1, 0, 0, /* [34] ビットマップデータの長さ16*16=256*/
0, 0, 0, 0, /* [38] 水平解像度(px/m) */
0, 0, 0, 0, /* [42] 垂直解像度(px/m) */
0, 0, 0, 0, /* [46] カラーインデックス数*/
0, 0, 0, 0, /* [50] 重要なカラーインデックス数*/
};



        if((keyfile = fopen(keyfn, "rt")) == NULL){
//              printf("Can not find key file for encryption. ¥n");
                return;
        }
        fgets( c_mode, 4, keyfile );
        fgets( c_klen, 8, keyfile );
        fgets( c_key, 64, keyfile );
        fclose(keyfile);

        strcpy(key, c_key);
        mode = atoi(c_mode);
        klen = atoi(c_klen);

        mn = 8;
        if(klen == 128){ mn = 16; }
        if(klen == 192){ mn = 24; }
        if(klen == 256){ mn = 32; }

        /*
        strcpy(folderfile, ".¥¥Encrypt¥¥");
        strcat(folderfile, srcfn);
        if( fopen_s(&srcfile, folderfile, "rb") != 0){
//              printf("Can not open plane file. ¥n");
                return;
        }
        */
        if( fopen_s(&srcfile, srcfn, "rb") != 0){
                printf("Can not open source file. ¥n");
                return;
        }

        fseek(srcfile, 0, SEEK_END);
        long filelen = ftell(srcfile);
        fseek(srcfile,0,0);

        fread(bmpHeader, sizeof(char), 54, srcfile);//srcfile.Read(bmpHeader, 54);
        fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
        jj = *((unsigned int*)&tmpch);
        j = jj^(jj>>16);
        nsize = j & 0x0000ffff;
        fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
        jj = *((unsigned int*)&tmpch);
        j = jj^(jj>>16);
        k = j & 0x0000ffff;
```

```
        nsize = nsize + (k<<16);

        fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
        jj = *((unsigned int*)&tmpch);
        j = jj^(jj>>16);
        fsize = j & 0x0000ffff;
        fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
        jj = *((unsigned int*)&tmpch);
        j = jj^(jj>>16);
        k = j & 0x0000ffff;
        fsize = fsize + (k<<16);

        pfName = new(char[nsize+8]);
        if(pfName == NULL){
//              printf("メモリー不足です。¥n");
                return;
        }

        for(i=0; i<nsize/2 ; i++){
                fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
                jj = *((unsigned int*)&tmpch);
                j = jj^(jj>>16);
                k = j & 0x0000ffff;
                pfName[2*i] = (char)(k>>8);
                pfName[2*i+1] = (char)(k&0x000000ff);
        }
        for(i=nsize/2; i<(nsize+5)/2 ; i++){
                pfName[2*i] = NULL;
                pfName[2*i+1] = NULL;
        }
        if(nsize%2 == 0){
                fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit , pointer  を
進める
        }
        if(nsize%2 == 1){
                fread(tmpch, sizeof(char), 4, srcfile);//srcfile.Read(tmpch, 4);//32 bit
                jj = *((unsigned int*)&tmpch);
                j = jj^(jj>>16);
                k = j & 0x0000ffff;
                pfName[nsize-1] = (char)(k>>8);
        }
/*
        strcpy(FName, ".¥¥Decrypt¥¥");
        strcat(FName, pfName);
        if((dstfile = fopen(FName, "wb")) == NULL){
            if(0 == _mkdir(".¥¥Decrypt")){
//                      printf("サブフォルダーDecrypt を作りました。¥n");
                        if((dstfile = fopen(FName, "wb")) == NULL){
                                return;
                        }
                }
                else{
```

```
                        return;
                }
        }
*/

        if((dstfile = fopen(dstfn, "wb")) == NULL){
                return;
        }


        for(i=0 ; i<fsize ; i+=2){
                q = fsize - i - 2;
                if(q >= 0){
                        if(4 == fread(tmpch, sizeof(char), 4, srcfile)){//.Read(tmpch, 4)){
                                jj = *((unsigned int*)&tmpch);//32 bit
                                j = jj^(jj>>16);
                                k = j & 0x0000ffff;
                                tmprbuf[0] = (unsigned char)(k>>8);
                                tmprbuf[0] ^= (unsigned char)key[(i/2)%mn];
                                tmprbuf[1] = (unsigned char)(k&0x000000ff);
                                tmprbuf[1] ^= (unsigned char)key[(i/2)%mn];
                                fwrite( tmprbuf,sizeof(char),2, dstfile);//dstfile.Write(tmprbuf, 2);
                        }
                }
                if(q < 0){ // q==-1
                        if(4 == fread(tmpch, sizeof(char), 4, srcfile)){//srcfile.Read(tmpch, 4)){
                                jj = *((unsigned int*)&tmpch);//32 bit
                                j = jj^(jj>>16);
                                k = j & 0x0000ffff;
                                tmprbuf[0] = (unsigned char)(k>>8);
                                tmprbuf[0] ^= (unsigned char)key[(i/2)%mn];
                                fwrite( tmprbuf,sizeof(char),1, dstfile);//dstfile.Write(tmprbuf, 1);
                        }
                }
        }

        fclose(srcfile);//.Close();
        fclose(dstfile);//.Close();

}
```

BmpDC.cpp

```
////////////////////////////////////////
// BmpDC.cpp : コンソールアプリケーションのエントリポイントを定義します。
//


#include "stdafx.h"
```

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "BmpDC.h"

using namespace std;


// メイン
int main(                                                   // 正常終了時:0、異常時:1
                int argc,                                   // 引数の数
                char** argv )                           // 引数へのポインタ
{
        printf( "Start!¥n" );

          // 引数チェック
        if( argc != 4 ){                            // 使い方の誤り
                printf( "引数の数が不正ですから異常終了します。¥n" );
                exit(1);
        }

    // 復号化実行
            // 鍵file    暗文     平文
        bmp2( argv[1], argv[2], argv[3] );

        printf( "End!¥n" );
        return 0;
}
```

Stdafx.cpp

```
////////////////////////////////////////////////
// stdafx.cpp : 標準インクルードBmpDC.pch のみを
// 含むソースファイルは、プリコンパイル済みヘッダーになります。
// stdafx.obj にはプリコンパイル済み型情報が含まれます。

#include "stdafx.h"

// TODO: このファイルではなく、STDAFX.H で必要な
// 追加ヘッダーを参照してください。
```

おわり。