Protected Communication System（Serpent）Ver. 3

ソフトウエアの名称　：　Protected Communication System（Serpent）

１．ソフトの概要

Serpent　は、アメリカで　DES　の代わりに採用するために、新しい暗号（AES）を公募したときに最終選考まで残った暗号化方式の一つです。　著作権 → http://en.wikipedia.org/wiki/Serpent_(cipher)

本来の鍵の長さはもっと長いのですが、このソフトでは、貿易管理令との関連で、鍵の長さを56ビットにしてあります。処理速度は鍵の長さとはあまり関係なく高速で暗号化されます。

初期ベクトルについては、鍵の長さとの関連で固定した値になっています。

```
/*Set mode*/
        if(IDC_RADIO5 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_ECB, "");
        }
        if(IDC_RADIO6 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_CBC, "0123456789abcdef0123456789abcdef");
        }
        if(IDC_RADIO7 == GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO7) ){
                rc=cipherInit(&cipherI, MODE_CFB1, "0123456789abcdef0123456789abcdef");
        }
```

この暗号ソフトは、Serpent のソースコードを参考にして作りました。
一部変更してありますのでご自分で信頼度は確認してください。ソースコードの変更部分については、plane.txt　で確認できます。

このソフトでは、ファイルの暗号化、復号化がまとめて出来ます。さらに非表示での変換で処理速度が確認できます。電子メールの添付ファイルの暗号化にご利用ください。

メールアドレスの管理をしながら、暗号メールで利用する場合は、“メールもビトマ”、または、“Cipher Web Mail”　をご利用ください。Serpent　や　AES、RSA　などでの５段階の多重暗号化が可能です。暗号メールソフトとして機能するのはもちろんですが、暗号化、復号化ルーツとしても機能します。

“メールもビトマ”、“Cipher Web Mail”　についている　Serpent　では、初期ベクトルの値も自由に設定できます。

２．　作者への連絡先(メールアドレス、ホームページ)

メールアドレス：uyama33@yahoo.co.jp　　（宇山　靖政）
ホームページ　：http://uyama22.pa.land.to/　（ソースコード）

３．　取り扱い種別(フリーウエア)

４．　動作環境

Windows Vista Home Premium　32 ビット
Windows 7 Home Premium　64 ビット
の上で動きます。

５．　別途必要なソフト：特になし（暗号ソフトとして単独で動きます。メール機能はありません。）

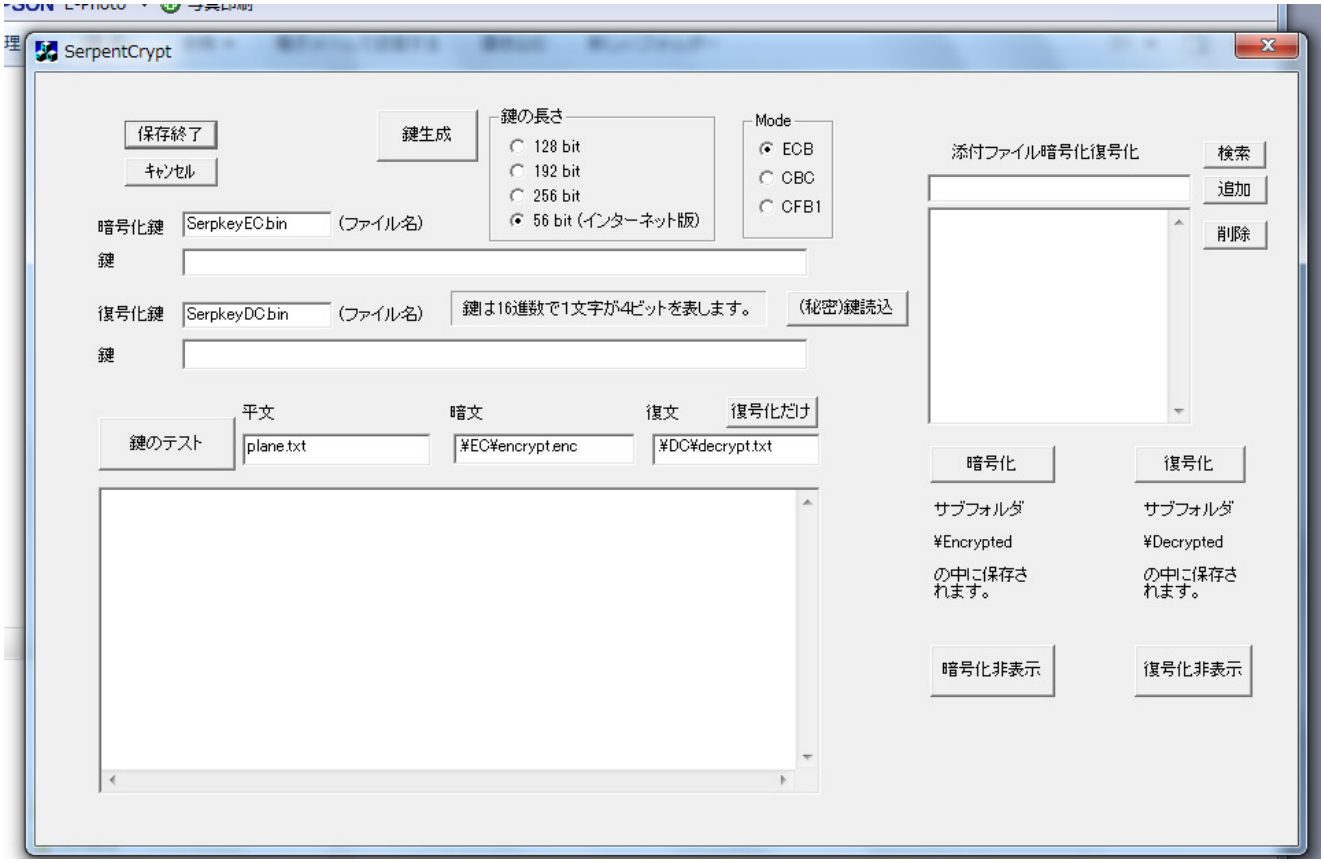
６．　インストール・アンインストール方法

インストール：SerpentPac.zip を解凍すると、このマニュアルの他に、SerpentSys.zip　（暗号ソフトと関連するフォルダ、ファイル）が現れます。

さらに、SerpentSys.zip を解凍すると、“SerpentSys”フォルダが出来ます。このフォルダをデスクトップ等の適当な場所に置き、その中にある
“SerpentCrypt.exe”へのショートカットを作成してください。

７．アンインストール：作成したショートカットと、フォルダを削除してください。

使い方：



１．適当なフォルダをつくり、Serpentcrypt.exe を置く。
２．暗号化したい添付ファイルを同じフォルダに入れる。
３．Serpentcrypt.exe　を起動して、平文の所に暗号化したいファイル名を記入する。
　　暗号化されたものの所に暗号化されたファイルの名前を記入する。
　　復号化されたものの所に平文のファイル名とは別の名前を記入する。
４．＜鍵を作成＞のボタンをクリックする。
　　必要なら、５６ビット鍵（１４文字の所２カ所には同じ文字列を入れる）の所を好きな１６進数にする。
５．＜鍵のテスト＞ボタンをクリックして正常に暗号化、復号化出来るか確認
６．暗号化されたものを添付ファイルとして送信。
７．相手にもこのソフトをダウンロードしてもらって、鍵を教える。
８．相手は、＜鍵を作成＞のボタンをクリックする。
　　５６ビット鍵（2 箇所の１４文字は同じ文字列を入れる）の所を教えられた１６進数にする。
９．暗号化されたファイル名を正しく入力する。
１０．＜復号化のみ＞をクリックする。
１１．相手は添付ファイルを復号化できる。
となります。

暗号化鍵、復号化鍵を読み込んでから、暗号化するものを検索して、リストボックスにその一覧を表示しておけば、連続して暗号化、復号化が行えます。
暗号化したものが置かれるフォルダをしっかり確認してください。この暗号化されたものを復号化することになります。

非表示で、操作すれば Serpent 暗号の高速処理が確認できます。この処理速度ならば電子メールを送信する途中で本文や添付ファイルを暗号化できます。
電子メールでの扱いには、メールアドレスとの関連が問題になります。“メールもビトマ”、または“Cipher Web Mail” を利用すれば、送信アドレスごとに暗号化鍵、復号化鍵、暗号化アルゴリズムを 5 段階で設定できます。

電子メールの添付ファイルを暗号化するときは、復号化鍵を相手の方に前もって届けて置いてください。そのためには、RSA 暗号をご利用ください。フリーソフト Protected Communication System(AFCrypt) が利用できます。

参考：
Protected Communication System に関しては、ヨーロッパ、アメリカでの特許を取得いたしました。

　このデモ版は、
鍵の作成と暗号化復号化が出来ます。鍵の長さには制限がつきます。

　注意：
これは、次に示すように　Serpent のもとのソースコードを一部変更してあります。
変更が正しいか否かは、自分で判断してください。

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

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

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

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

      return inputLen;

    case MODE_CFB1:
                cipher->mode = MODE_ECB;        /* do encryption in ECB */
```

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

                    /*
        t[0] = ((unsigned long*)cipher->IV)[0];
        t[1] = ((unsigned long*)cipher->IV)[1];
        t[2] = ((unsigned long*)cipher->IV)[2];
        t[3] = ((unsigned long*)cipher->IV)[3];
        for(b=0; b<inputLen; b+=8, input++, outBuffer++)
          {
            int bit;
            int bytedata = (input[0])&0xFF;

            for(bit=0; bit<8; bit++)
              {
                unsigned long tt[4];

                serpent_encrypt(tt, tt, key->subkeys);

                bytedata ^= (tt[0]&1);

                tt[0] = ((tt[0]>>1)&0x7FFFFFFF) | ((tt[1]&1)<<31);
                tt[1] = ((tt[1]>>1)&0x7FFFFFFF) | ((tt[2]&1)<<31);
                tt[2] = ((tt[2]>>1)&0x7FFFFFFF) | ((tt[3]&1)<<31);
                tt[3] = ((tt[3]>>1)&0x7FFFFFFF) | ((bytedata&1)<<31);
                bytedata = bytedata>>1;

              }
            outBuffer[0] = (unsigned char)((tt[3]>>24)&0xFF);
          }
        ((unsigned long*)cipher->IV)[0] = t[0];
        ((unsigned long*)cipher->IV)[1] = t[1];
        ((unsigned long*)cipher->IV)[2] = t[2];
        ((unsigned long*)cipher->IV)[3] = t[3];
        return inputLen;
*/
      default:
        return BAD_CIPHER_STATE;
      }
}

int blockDecrypt(cipherInstance *cipher,
                        keyInstance *key,
                        BYTE *input,
                        int inputLen,
```

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

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

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

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

                /*
```

```
        t[0] = ((unsigned long*)cipher->IV)[0];
        t[1] = ((unsigned long*)cipher->IV)[1];
        t[2] = ((unsigned long*)cipher->IV)[2];
        t[3] = ((unsigned long*)cipher->IV)[3];
        for(b=0; b<inputLen; b+=8, input++, outBuffer++)
          {
            int bit;
            int bytedata = (input[0])&0xFF;
            int outdata=0;

            for(bit=0; bit<8; bit++)
              {
                unsigned long tt[4];
                serpent_encrypt(t, tt, key->subkeys);
                outdata |= ((bytedata^tt[0])&1)<<bit;
                tt[0] = ((tt[0]>>1)&0x7FFFFFFF) | ((tt[1]&1)<<31);
                tt[1] = ((tt[1]>>1)&0x7FFFFFFF) | ((tt[2]&1)<<31);
                tt[2] = ((tt[2]>>1)&0x7FFFFFFF) | ((tt[3]&1)<<31);
                tt[3] = ((tt[3]>>1)&0x7FFFFFFF) | ((bytedata&1)<<31);
                bytedata = bytedata>>1;
              }
            outBuffer[0] = outdata;
          }
        ((unsigned long*)cipher->IV)[0] = t[0];
        ((unsigned long*)cipher->IV)[1] = t[1];
        ((unsigned long*)cipher->IV)[2] = t[2];
        ((unsigned long*)cipher->IV)[3] = t[3];
        return inputLen;
*/
      default:
        return BAD_CIPHER_STATE;
      }
}
```