以 CryptoAPI 實做 BCB/Delphi 檔案/字串 加解密

資料整理：bruce0211@yahoo.com.tw 2003/04/27

■ 訊息加解密及其應用

對稱式加密法	常用的如 DES 加密標準，速度快；私鑰加/解密，所謂私鑰就是不能公開的鑰匙(Key)，加密解密都用同樣的 Key。 舉例來說：A、B、C 三個人，A 將檔案用私鑰加密，B 跟 C 無 A 的私鑰就無法解開文件，由於加解密是同一把 Key，所以加密者一定可以解密。 其它的對稱式加密法標準還有：AES、DESX、IDEA、RC2、RC4、RC5 等等。
非對稱式加密法	常用的如 RSA 加密標準，速度較對稱式加密法慢；公鑰加密/私鑰解密，加解密分別使用不同的鑰匙(Key)，能用公鑰加密的人不能用同一把鑰匙解密，所以公鑰可任意公開(這也是公鑰名稱由來)；但解密則需要特定的鑰匙，只有該特定鑰匙才能解開，這個鑰匙當然就不能公開，也就是所謂的「私鑰」。 舉例來說：A、B、C 三個人，皆有相同的加密公鑰，A 將檔案用公鑰加密後傳給 D(擁有解密私鑰)，B 跟 C 雖然都有相同的公鑰，但都無法解開這的檔案(包含加密者 A 本身也無法解開)，只有擁有解密私鑰的 D 可以解開這個檔案。 其它的非對稱式加密法標準還有：ElGamal、DSS 等等。
數位簽章	數位簽章為非對稱式加密法之反相運作；私鑰加密/公鑰解密，重點不在資料的保密，因為解密的公鑰是可公開的，人人都可拿到這把公鑰來解開資料；加密的資料一定是使用特定私鑰來加密，才能以與該私鑰相配對的公鑰解密，這可以用來確認加密者身分(是否擁有正確的加密私鑰)
訊息摘要 (單項雜湊函式)	與數位簽章的功能類似，但不用為了證明檔案確實由某人所發，而將檔案先加密再用加密者所給的公鑰看看可不可以解密；訊息摘要就像是檔案的指紋，而檔案本身不需要任何加密也能透過訊息摘要證明檔案是否被修改過。 常見的例子如 symantec 防毒軟體 AntiVirus 公佈於網站上的病毒定義檔下載網頁，每次發表新的病毒定義檔也同時給你該檔的 MD5 值(不管檔案大小，都隨同檔案公告一個 16 個 Bytes 長度的 MD5 數據)，若該病毒定義檔被動過手腳，哪怕只是更動一個 Byte，整個檔案的 MD5 值就會截然不同，您也沒辦法將檔案修改到 ”恰巧” 有跟原來相同的 MD5 值，要使兩個不同的檔案產生相同的 MD5 值結果，幾乎是不可能的事。 訊息摘要函式由資料或檔案內容(無論長度大小)推導出一組數據，數據長度通常在 128-256 bits 之間，這組數據就稱為訊息摘要(message digest)，訊息摘要函式具備下面的數學性質：

	1. 無論輸入的資料長度大小，訊息摘要的輸出數據爲固定長度(如 MD5 的輸出數據長度爲 16 Bytes) 2. 輸出結果的每一個 bit 與輸入的每一個 bit 有關 3. 如果輸入的資料有任一 bit 變動，輸出的每一個 bit 都有 50%的機率會改變 4. 必須很難(或無法)找到不同的輸入資料而能產生相同的訊息摘要的情況發生 訊息摘要有時也稱單項雜湊函式(one-way hash function)，因爲它產生的值無法反推回原始輸入值，難以攻擊、幾乎各不相同且數字範圍分布很廣；訊息摘要函式本身並不對資料進行加密或解密的動作，它的用途是替資料產生數位簽名、訊息認證碼(MAC)，以及由密語(passphrase)產生加密用的 Key。 常用的訊息摘要函式有：HMAC、MD2、MD4、MD5、SHA、SHA-1、SHA-256、RIPEMD-160 等等。

■ 相關名詞歸類

Checksum	CRC32, XOR32bit, XOR16bit, CRC16-CCITT, CRC16-Standard
Hash(雜湊法)	MD4, MD5, SHA (other Name SHS), SHA1, RipeMD128, RipeMD160, RipeMD256, RipeMD320, Haval (128, 160, 192, 224, 256) with Rounds, Snefru, Square, Tiger Sapphire II (128, 160, 192, 224, 256, 288, 320)
Cipher(加密法)	Gost, Cast128, Cast256, Blowfish, IDEA, Mars, Misty 1, RC2, RC4, RC5, RC6, FROG, Rijndael, SAFER, SAFER-K40, SAFER-SK40,SAFER-K64, SAFER-SK64, SAFER-K128, SAFER-SK128, TEA, TEAN, Skipjack, SCOP, Q128, 3Way, Twofish, Shark, Square, Single DES, Double DES, Triple DES, Double DES16, Triple DES16, TripleDES24, DESX, NewDES, Diamond II, Diamond II Lite, Sapphire II
RNG	Standard Random Generator, Linear Feedback Shift Register RNG with variable Period from 2^64-1 to 2^2032-1.
Text Formats	Hexadecimal, MIME Base 64, Plain, RFC1760 Six Word, UU Coding, XX Coding

■ CryptoAPI

Windows 2000 藉由 advapi32.dll 提供我們 CryptoAPI 函式庫，這個函式庫包裝了一些複雜的演算法，免除了我們自行實做上的困難；透過 CryptoAPI，我們可輕易的保護所需保護的資料；目前可使用 CryptoAPI 函式庫的平台環境為 NT4.0(或更新的版本)、Win95 OSR2 版本或是安裝過 IE 3.02 以上的版本，WinCE 尚不支援

由於編碼演算法很複雜，容易變動又常有專利權的問題，所以 CryptoAPI 通常以模組的架構來提供應用程式呼叫；支援 CryptoAPI 的編碼模組又稱之為”編碼服務提供者”(Cryptographic Service Providers；CSP)。下表是存取 CSP 的一些 API

CryptAcquireContext()	產生一個 CSP 物件及編碼用的 key
CryptGetProvParam()	檢查 CSP 物件的設定和參數
CryptReleaseContext()	釋放掉一個 CSP 物件
CryptGetDefaultProvider()	取得預設 CSP 物件的 handle
CryptEnumProvider()	檢視所有已安裝在電腦中的 CSP 模組
CryptEnumProviderTypes()	列出目前電腦中安裝好的 CSP 型別

CryptoAPI 支援多個密碼提供者型別，例如您可將某些資訊用 RSA 加密，而又將其他資訊使用資訊簽名，下表為一些 CSP 型別

CSP 型別	資料加密法	簽名加密法
PROV_RSA_FULL	RC2、RC4 (對稱式加密)	RSA (非對稱式加密)
PROV_RSA_SIG		RSA
PROV_DSS		DSS
PROV_FORTEZZA	SkipJack	DSS
PROV_SSL		RSA
PROV_MS_EXCHANGECAST		RSA

另外，值得一提的是，CryptoAPI 使用鍵值資料庫來做編碼的基礎，當建立 CryptoAPI 加密應用程式時，需要先在系統中建立鍵值資料庫，否則 CryptoAPI 加密應用程式不能正確的運作；建立鍵值資料庫最好的方法是下載 Microsoft’s Developer Network (或 MSDN)中的所有範例程式樣本，樣本中有一支名為 InitUser.c 的程式，這個程式就是用以建立使用 CryptoAPI 的基礎鍵值資料庫；在本文後續介紹的「實做 CryptoAPI 檔案/字串加解密」程式中，筆者已將 InitUser.c 改寫為 BCB/Delphi 版本，加解密範例函式也會自動引用 InitUser 這個 Function，不需單獨或特別地去執行它。

■ BCB/Delphi 加解密範例函式主要架構

□ 加密流程

1. 執行 InitUser()，這個函式在同一台電腦上其實只會執行一次，當發現作業系統中已有註冊基礎鍵值資料庫時，本函式會被 bypass
2. CryptAcquireContext() ：取得 CSP 物件的 Handle
3. 判斷使用者有否設定加密 key (範例程式中傳入的 FilePassword 變數)

if (FilePassword == NULL)

CryptGenKey()：產生一組隨機的 public/private key 來搭配 CSP 一起使用

CryptGetUserKey()：從 CSP 中取出 public key

CryptExportKey()：建立 blob 資料區塊，用來傳送 key (通常是 public key)給其它程式使用，並將此 blob 資料區塊寫到加密後的檔案檔頭，以備將來解密程式取出使用

CryptDestroyKey()：釋放 key 物件

else

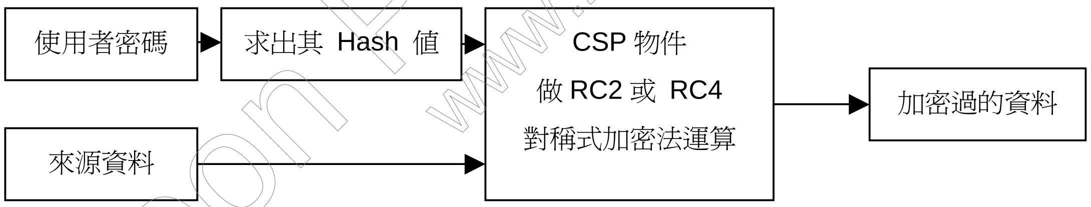
CryptCreateHash()：產生 Hash 物件

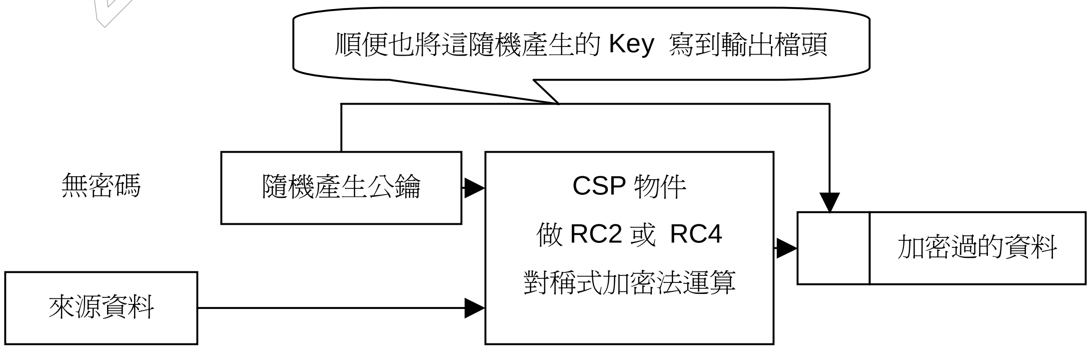
CryptHashData()：替 FilePassword 產生 Hash 值

CryptDeriveKey()：傳回經過 Hash 後的 key 值

CryptDestroyHash()：釋放 hash 物件

4. CryptEncrypt()：依據 key 值(隨機產生的 public key 或 hash 過的 FilePassword 變數)，來加密緩衝區中的資料



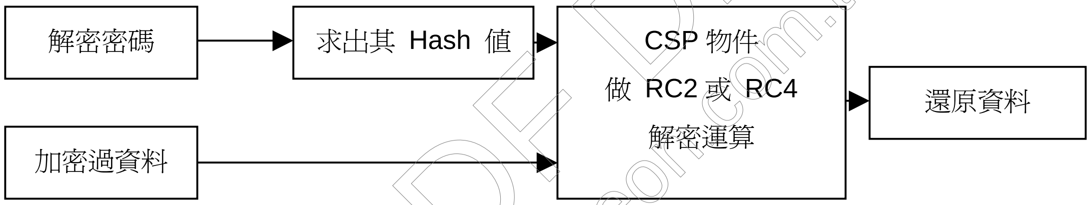


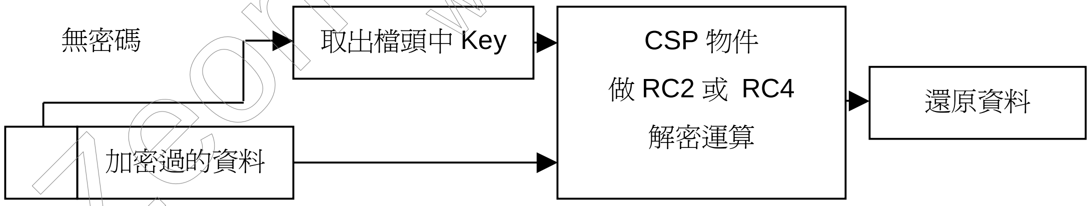
☐ 解密流程

1. 執行 InitUser()，這個函式在同一台電腦上其實只會執行一次，當發現作業系統中已有註冊基礎鍵值資料庫時，本函式會被 bypass
2. CryptAcquireContext() ：取得 CSP 物件的 Handle
3. 判斷使用者有否輸入解密 key (範例程式中傳入的 FilePassword 變數)

```
if (FilePassword == NULL)
    CryptImportKey() 由讀出的檔案檔頭 blob 中取回鍵值(接收當初加密程序中
                     CryptExportKey()產生的 key 值)
else
    CryptCreateHash()：產生 Hash 物件
    CryptHashData()：替 FilePassword 產生 Hash 值
    CryptDeriveKey()：傳回經過 Hash 後的 key 值
    CryptDestroyHash()：釋放 hash 物件
```

4. CryptDecrypt()：依據 key 值(檔案檔頭 blob 中取回的 public key 或 hash 過的 FilePassword 變數),來解碼緩衝區中的資料





■ 補充說明

1. 基本上本文的例子皆是由 MSDN 中的範例改寫,架構上有一點筆者不是很明瞭,在 User 有輸入 Key 的情況下，爲何 Key 要經過 Hash 再丟入 CSP ?? 應該可以將 Key 直接跟 Data 一起丟入 CSP 做加密運算吧 ? 或許...是寫這個 MSDN 範例的作者要展示 Hash 用法的功能罷了....
2. 在改寫 BCB/Delphi 版的過程中，檔案的處理部份我已盡量使用 TFileStream 去做，因爲筆者認爲既然要用 BCB/Delphi ，就應該用高階的 C++ 或 VCL 檔案存取類別 (Class)，如果還是用 fopen()等古早的 C 語法(雖然 BCB/Delphi 中也可使用 fopen())，那直接拿 MSDN 中範例去 COMPILE 即可，用不著多此一舉改爲 BCB/Delphi 版
3. 接下來即公開筆者研究改寫，經過兩個多禮拜才測試成功的程式碼，基本上分爲 4 大函數：「檔案加密」、「檔案解密」、「字串加密」、「字串解密」；其中，在字串加解密的函數中有比檔案加解密多一步驟，那就是在加密程序後有可能產生 ASCII 爲 0 的 Byte(NULL)，但它仍爲資料的一部份，若是直接針對檔案串流做處理那還沒問題；但若放於字串變數中時，NULL 以後的資料都會被截掉，換句話說，經過「字串加密」後若所得的加密資料中含有 NULL，我們根本沒有辦法用變數去完整保存這個加密後的全部資料，所以在字串加解密的函數會多一步驟將資料轉成 HEX 值傳回
4. CryptoAPI 還有一堆功能未發掘出來，在此僅以拋磚引玉之心，祈求各位賢達先進能多多指教並發表更新的應用主軸，如「如何以 CryptoAPI 取得檔案或字串的 MD5 值」等等

■ 以 Borland C++ Builder (5.0 版) 實做 CryptoAPI 檔案/字串加解密

C++Builder 中相關常數的宣告

```
#include <vcl.h>
#include <stdio.h>
#pragma hdrstop

#include "Unit1.h"

#ifdef USE_BLOCK_CIPHER
     #define ENCRYPT_ALGORITHM CALG_RC2
     #define ENCRYPT_BLOCK_SIZE 8
#else
     #define ENCRYPT_ALGORITHM CALG_RC4
     #define ENCRYPT_BLOCK_SIZE 1
#endif
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
          : TForm(Owner)
{
}
```

函式名稱：InitUser()

函式功能：在系統環境中建立 CryptoAPI 的基礎鍵值資料庫

```
void __fastcall InitUser()
{
    HCRYPTPROV hProv;
    HCRYPTKEY hKey;
    char szUserName[100];
    unsigned long dwUserNameLen = 100;
    char tmp_buf[81];

    // Attempt to acquire a handle to the default key container.
    if (!CryptAcquireContext(&hProv, NULL, MS_DEF_PROV, PROV_RSA_FULL, 0))
       {
         // Some sort of error occured.

         // Create default key container.
         if (!CryptAcquireContext(&hProv, NULL, MS_DEF_PROV, PROV_RSA_FULL,
CRYPT_NEWKEYSET))
            {
              sprintf(tmp_buf,"Error creating key container!\n");
              ShowMessage(tmp_buf);
              return;
            }

         // Get name of default key container.
         if (!CryptGetProvParam(hProv, PP_CONTAINER, szUserName, &dwUserNameLen, 0))
            {
              // Error getting key container name.
              szUserName[0] = 0;
            }

         sprintf(tmp_buf,"Create key container '%s'\n",szUserName);
         ShowMessage(tmp_buf);
       }

    // Attempt to get handle to signature key.
    if (!CryptGetUserKey(hProv, AT_SIGNATURE, &hKey))
       {
         if (GetLastError() == NTE_NO_KEY)
            {
              // Create signature key pair.
              sprintf(tmp_buf,"Create signature key pair\n");
              ShowMessage(tmp_buf);

              if (!CryptGenKey(hProv,AT_SIGNATURE,0,&hKey))
                 {
                   sprintf(tmp_buf,"Error %x during CryptGenKey (1)!\n", GetLastError());
                   ShowMessage(tmp_buf);
                   return;
                 }
              else
                 {
                   CryptDestroyKey(hKey);
                 }
            }
         else
            {
              sprintf(tmp_buf,"Error %x during CryptGetUserKey (1)!\n", GetLastError());
              ShowMessage(tmp_buf);
              return;
            }
       }

    // Attempt to get handle to exchange key.
    if (!CryptGetUserKey(hProv,AT_KEYEXCHANGE,&hKey))
       {
        if(GetLastError()==NTE_NO_KEY)
          {
            // Create key exchange key pair.
            sprintf(tmp_buf,"Create key exchange key pair\n");
            ShowMessage(tmp_buf);

            if (!CryptGenKey(hProv,AT_KEYEXCHANGE,0,&hKey))
               {
                 sprintf(tmp_buf,"Error %x during CryptGenKey (2)!\n", GetLastError());
                 ShowMessage(tmp_buf);
                 return;
               }
            else
               {
                 CryptDestroyKey(hKey);
               }

          }
        else
          {
            sprintf(tmp_buf,"Error %x during CryptGetUserKey (2)!\n", GetLastError());
            ShowMessage(tmp_buf);
            return;
          }
      }

    CryptReleaseContext(hProv,0);

}
```

函式名稱：EnCryptFile(char *SourFile,char *DestFile,char *FilePassword)
函式功能：將原始檔案加密並轉出成目的檔案

```
bool __fastcall EnCryptFile(char *SourFile,char *DestFile,char *FilePassword)
{
    TFileStream *hSource;
    TFileStream *hDestination;

    HCRYPTPROV hProv     = 0;
    HCRYPTKEY hKey       = 0;
    HCRYPTKEY hXchgKey = 0;
    HCRYPTHASH hHash     = 0;

    char *pbKeyBlob;
    char *pbBuffer;

    unsigned long dwKeyBlobLen;
    unsigned long dwBlockLen;
    unsigned long dwBufferLen;
    unsigned long dwCount;

    int eof = 0;
    bool status = false;
    char tmp_buf[81];

    hSource=new TFileStream(SourFile,fmOpenRead);
    if (!hSource)
       {
         sprintf(tmp_buf,"無法開啓原始檔案!\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    hDestination=new TFileStream(DestFile,fmCreate);
    if (!hDestination)
       {
         sprintf(tmp_buf,"無法開啓目的檔案\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    InitUser();

    if (!CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL, 0))
       {
         sprintf(tmp_buf,"無法取得 Handle \n", GetLastError());
         ShowMessage(tmp_buf);
         goto done;
       }

    if (FilePassword == NULL) //不輸入加密 Key 時，系統自動產生 Key 於目的檔頭，故加密後的檔案 Size
會比原來大
       {
         if (!CryptGenKey(hProv, ENCRYPT_ALGORITHM, CRYPT_EXPORTABLE,&hKey))
            {
              sprintf(tmp_buf,"Error %x 產生隨機鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

         if (!CryptGetUserKey(hProv, AT_KEYEXCHANGE, &hXchgKey))
            {
              sprintf(tmp_buf,"Error %x 交換鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }
```

```
        if (!CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, NULL, &dwKeyBlobLen))
            {
              sprintf(tmp_buf,"Error %x 由 blof 取出鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

        if ((pbKeyBlob = (char *) malloc(dwKeyBlobLen)) == NULL)
            {
              sprintf(tmp_buf,"記憶體不足！!\n");
              ShowMessage(tmp_buf);
              goto done;
            }

        if (!CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, pbKeyBlob,&dwKeyBlobLen))
            {
              sprintf(tmp_buf,"Error %x 傳回簽名時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

        CryptDestroyKey(hXchgKey);
        hXchgKey = 0;

        if (!hDestination->Write(&dwKeyBlobLen, sizeof(DWORD)))
            {
              sprintf(tmp_buf,"無法寫入檔頭！\n");
              ShowMessage(tmp_buf);
              goto done;
            }

        if (!hDestination->Write(pbKeyBlob,dwKeyBlobLen))
            {
              sprintf(tmp_buf,"無法寫入檔頭\n");
              ShowMessage(tmp_buf);
              goto done;
            }

      }
    else
      {

        if (!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash))
            {
              sprintf(tmp_buf,"Error %x 產生 Hash 物件時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

        if (!CryptHashData(hHash, FilePassword, strlen(FilePassword), 0))
            {
              sprintf(tmp_buf,"Error %x 無法產生 Hash 值！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

        if (!CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, &hKey))
            {
              sprintf(tmp_buf,"Error %x Hash 物件無法傳回鍵值\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

        CryptDestroyHash(hHash);
        hHash = 0;

      }

    dwBlockLen = 1000 - 1000 % ENCRYPT_BLOCK_SIZE;
    if (ENCRYPT_BLOCK_SIZE > 1)
        dwBufferLen = dwBlockLen + ENCRYPT_BLOCK_SIZE;
    else
        dwBufferLen = dwBlockLen;


    if ((pbBuffer = (char *) malloc(dwBufferLen)) == NULL)
      {
        sprintf(tmp_buf,"記憶體不足！\n");
        ShowMessage(tmp_buf);
        goto done;
      }

    do {
        dwCount = hSource->Read(pbBuffer,dwBlockLen);
        if (!dwCount)
           {
             sprintf(tmp_buf,"無法開啓原始檔！!\n");
             ShowMessage(tmp_buf);
             goto done;
           }

        eof = 1;

        if (!CryptEncrypt(hKey, 0, eof, 0, pbBuffer, &dwCount, dwBufferLen))
           {
             sprintf(tmp_buf,"Error %x 加密緩衝區資料發生錯誤！\n", GetLastError());
             ShowMessage(tmp_buf);
             goto done;
           }

        if (!hDestination->Write(pbBuffer,dwCount))
           {
             sprintf(tmp_buf,"無法寫入目的檔！\n");
             ShowMessage(tmp_buf);
             goto done;
           }
    } while(hSource->Position<hSource->Size);

    status = true;
    sprintf(tmp_buf,"檔案加密作業完成！\n");
    ShowMessage(tmp_buf);

 done:
    if(hSource) delete hSource;
    if(hDestination) delete hDestination;

    if(pbKeyBlob && FilePassword == NULL) free(pbKeyBlob);
    if(pbBuffer) free(pbBuffer);
    if(hKey) CryptDestroyKey(hKey);
    if(hXchgKey) CryptDestroyKey(hXchgKey);
    if(hHash) CryptDestroyHash(hHash);
    if(hProv) CryptReleaseContext(hProv, 0);
    return(status);
}
```

函式名稱：DeCryptFile(char *SourFile,char *DestFile,char *FilePassword)

函式功能：將加密檔案還原並轉出成目的檔案

```
bool __fastcall DeCryptFile(char *SourFile,char *DestFile,char *FilePassword)
{
    TFileStream *hSource;
    TFileStream *hDestination;

    HCRYPTPROV hProv     = 0;
    HCRYPTKEY hKey       = 0;
    HCRYPTHASH hHash     = 0;

    char *pbKeyBlob;
    char *pbBuffer;

    unsigned long dwKeyBlobLen;
    unsigned long dwBlockLen;
    unsigned long dwBufferLen;
    unsigned long dwCount;

    int eof = 0;
    bool status = false;
    char tmp_buf[81];

    hSource=new TFileStream(SourFile,fmOpenRead);
    if (!hSource)
       {
         sprintf(tmp_buf,"無法開啟原始檔\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    hDestination=new TFileStream(DestFile,fmCreate);
    if (!hDestination)
       {
         sprintf(tmp_buf,"無法開啟目的檔！\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    InitUser();

    if (!CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL, 0))
       {
         sprintf(tmp_buf,"無法取得 handle ！\n", GetLastError());
         ShowMessage(tmp_buf);
         goto done;
       }

    if (FilePassword == NULL)
       {

         if (!hSource->Read(&dwKeyBlobLen, sizeof(DWORD)))
            {
              sprintf(tmp_buf,"無法讀取檔頭！\n");
              ShowMessage(tmp_buf);
              goto done;
            }

         if ((pbKeyBlob = (char *) malloc(dwKeyBlobLen)) == NULL)
            {
              sprintf(tmp_buf,"記憶體不足！\n");
              ShowMessage(tmp_buf);
              goto done;
            }

         if (!hSource->Read(pbKeyBlob, dwKeyBlobLen))
```

```
          {
            sprintf(tmp_buf,"無法讀取檔頭！\n");
            ShowMessage(tmp_buf);
            goto done;
          }

       if (!CryptImportKey(hProv, pbKeyBlob, dwKeyBlobLen, 0, 0, &hKey))
          {
            sprintf(tmp_buf,"從 blob 中取得鍵值時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }
     }
  else
     {
       if (!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash))
          {
            sprintf(tmp_buf,"產生 Hash 物件時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

       if (!CryptHashData(hHash, FilePassword, strlen(FilePassword), 0))
          {
            sprintf(tmp_buf,"產生 Hash 值時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

       if (!CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, &hKey))
          {
            sprintf(tmp_buf,"取得 Hash 值時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }
       CryptDestroyHash(hHash);
       hHash = 0;
     }

  dwBlockLen = 1000 - 1000 % ENCRYPT_BLOCK_SIZE;
  dwBufferLen = dwBlockLen;

  if ((pbBuffer = (char *) malloc(dwBufferLen)) == NULL)
     {
       sprintf(tmp_buf,"記憶體不足\n");
       ShowMessage(tmp_buf);
       goto done;
     }

  do {
       dwCount = hSource->Read(pbBuffer,dwBlockLen);
       if (!dwCount)
          {
            sprintf(tmp_buf,"無法讀取原始檔！\n");
            ShowMessage(tmp_buf);
            goto done;
          }

       eof = 1;

       if (!CryptDecrypt(hKey, 0, eof, 0, pbBuffer, &dwCount))
          {
            sprintf(tmp_buf,"解密時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

       if (!hDestination->Write(pbBuffer, dwCount))
          {
                sprintf(tmp_buf,"無法寫入目的檔！\n");
                ShowMessage(tmp_buf);
                goto done;
              }
    } while(hSource->Position<hSource->Size);

    status = true;
    sprintf(tmp_buf,"檔案解密作業完成！\n");
    ShowMessage(tmp_buf);

done:
    if(hSource) delete hSource;
    if(hDestination) delete hDestination;

    if(pbKeyBlob && FilePassword == NULL) free(pbKeyBlob);
    if(pbBuffer) free(pbBuffer);
    if(hKey) CryptDestroyKey(hKey);
    if(hHash) CryptDestroyHash(hHash);
    if(hProv) CryptReleaseContext(hProv, 0);

    return(status);
}
```

函式名稱：EnCryptStr(char *SourStr,char *DestStr,char *StrPassword)
函式功能：將字串加密成另一個經過加密編碼的字串

```
bool __fastcall EnCryptStr(char *SourStr,char *DestStr,char *StrPassword)
{
    TStringStream *hSource;
    TStringStream *hDestination;

    HCRYPTPROV hProv     = 0;
    HCRYPTKEY hKey       = 0;
    HCRYPTKEY hXchgKey = 0;
    HCRYPTHASH hHash     = 0;

    char *pbKeyBlob;
    char *pbBuffer;

    unsigned long dwKeyBlobLen;
    unsigned long dwBlockLen;
    unsigned long dwBufferLen;
    unsigned long dwCount;

    int eof = 0;
    bool status = false;
    char tmp_buf[81];

    hSource=new TStringStream(SourStr);
    if (!hSource)
       {
         sprintf(tmp_buf,"無法開啓原始檔案!\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    hDestination=new TStringStream("");

   InitUser();

   if (!CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL, 0))
      {
        sprintf(tmp_buf,"無法取得 Handle \n", GetLastError());
        ShowMessage(tmp_buf);
        goto done;
      }


    if (StrPassword == NULL) //不輸入加密 Key 時，系統自動產生 Key 於目的檔頭，故加密後的檔案 Size
會比原來大
       {
         if (!CryptGenKey(hProv, ENCRYPT_ALGORITHM, CRYPT_EXPORTABLE,&hKey))
            {
              sprintf(tmp_buf,"Error %x 產生隨機鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

         if (!CryptGetUserKey(hProv, AT_KEYEXCHANGE, &hXchgKey))
            {
              sprintf(tmp_buf,"Error %x 交換鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
            }

         if (!CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, NULL, &dwKeyBlobLen))
            {
              sprintf(tmp_buf,"Error %x 由 blof 取出鍵值時發生錯誤！\n", GetLastError());
              ShowMessage(tmp_buf);
              goto done;
```

```
          }

        if ((pbKeyBlob = (char *) malloc(dwKeyBlobLen)) == NULL)
          {
            sprintf(tmp_buf,"記憶體不足！!\n");
            ShowMessage(tmp_buf);
            goto done;
          }

        if (!CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, pbKeyBlob,&dwKeyBlobLen))
          {
            sprintf(tmp_buf,"Error %x 傳回簽名時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

        CryptDestroyKey(hXchgKey);
        hXchgKey = 0;

        if (!hDestination->Write(&dwKeyBlobLen, sizeof(DWORD)))
          {
            sprintf(tmp_buf,"無法寫入檔頭！\n");
            ShowMessage(tmp_buf);
            goto done;
          }

        if (!hDestination->Write(pbKeyBlob,dwKeyBlobLen))
          {
            sprintf(tmp_buf,"無法寫入檔頭\n");
            ShowMessage(tmp_buf);
            goto done;
          }

    }
  else
    {

        if (!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash))
          {
            sprintf(tmp_buf,"Error %x 產生 Hash 物件時發生錯誤！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

        if (!CryptHashData(hHash, StrPassword, strlen(StrPassword), 0))
          {
            sprintf(tmp_buf,"Error %x 無法產生 Hash 值！\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

        if (!CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, &hKey))
          {
            sprintf(tmp_buf,"Error %x Hash 物件無法傳回鍵值\n", GetLastError());
            ShowMessage(tmp_buf);
            goto done;
          }

        CryptDestroyHash(hHash);
        hHash = 0;

    }

  dwBlockLen = 1000 - 1000 % ENCRYPT_BLOCK_SIZE;

  if (ENCRYPT_BLOCK_SIZE > 1)
      dwBufferLen = dwBlockLen + ENCRYPT_BLOCK_SIZE;
  else
      dwBufferLen = dwBlockLen;
    if ((pbBuffer = (char *) malloc(dwBufferLen)) == NULL)
       {
         sprintf(tmp_buf,"記憶體不足！\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    do {
        dwCount = hSource->Read(pbBuffer,dwBlockLen);
        if (!dwCount)
           {
             sprintf(tmp_buf,"無法開啓原始檔！!\n");
             ShowMessage(tmp_buf);
             goto done;
           }

        eof = 1;

        if (!CryptEncrypt(hKey, 0, eof, 0, pbBuffer, &dwCount, dwBufferLen))
           {
             sprintf(tmp_buf,"Error %x 加密緩衝區資料發生錯誤！\n", GetLastError());
             ShowMessage(tmp_buf);
             goto done;
           }

        if (!hDestination->Write(pbBuffer,dwCount))
           {
             sprintf(tmp_buf,"無法寫入目的檔！\n");
             ShowMessage(tmp_buf);
             goto done;
           }
    } while(hSource->Position<hSource->Size);

    //直接傳回加密後結果，若是檔案串流還沒關係，字串串流會有中間有 null 的問題 ....
    //strcpy(DestStr,hDestination->DataString.c_str());

    //怕字串串流會有中間有 null 的問題，所以需要將加密結果轉成 hex 再回傳
    strcpy(DestStr,StrToHex(hDestination->DataString).c_str());

    status = true;
    sprintf(tmp_buf,"檔案加密作業完成！\n");
    ShowMessage(tmp_buf);

 done:
    if(hSource) delete hSource;
    if(hDestination) delete hDestination;

    if(pbKeyBlob && StrPassword == NULL) free(pbKeyBlob);
    if(pbBuffer) free(pbBuffer);

    if(hKey) CryptDestroyKey(hKey);
    if(hXchgKey) CryptDestroyKey(hXchgKey);
    if(hHash) CryptDestroyHash(hHash);
    if(hProv) CryptReleaseContext(hProv, 0);
    return(status);
}
```

函式名稱：DeCryptStr(char *SourStr,char *DestStr,char *StrPassword)
函式功能：將加密字串還原

```
bool __fastcall DeCryptStr(char *SourStr,char *DestStr,char *StrPassword)
{
    bool status = false;

    TStringStream *hSource;
    TStringStream *hDestination;

    HCRYPTPROV hProv     = 0;
    HCRYPTKEY hKey       = 0;
    HCRYPTHASH hHash     = 0;

    char *pbKeyBlob;
    char *pbBuffer;

    unsigned long dwKeyBlobLen;
    unsigned long dwBlockLen;
    unsigned long dwBufferLen;
    unsigned long dwCount;

    int eof = 0;
    char tmp_buf[81];

    //將 HEX 解碼
    String ss=HexToStr(SourStr);

    hSource=new TStringStream(ss);
    if (!hSource)
       {
         sprintf(tmp_buf,"無法開啓原始檔\n");
         ShowMessage(tmp_buf);
         goto done;
       }

    hDestination=new TStringStream("");

    InitUser();

    if (!CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL, 0))
       {
         sprintf(tmp_buf,"無法取得 handle ！\n", GetLastError());
         ShowMessage(tmp_buf);
         goto done;
       }


    if (StrPassword == NULL)
       {
         if (!hSource->Read(&dwKeyBlobLen, sizeof(DWORD)))
            {
              sprintf(tmp_buf,"無法讀取檔頭！\n");
              ShowMessage(tmp_buf);
              goto done;
            }

         if ((pbKeyBlob = (char *) malloc(dwKeyBlobLen)) == NULL)
            {
              sprintf(tmp_buf,"記憶體不足！\n");
              ShowMessage(tmp_buf);
              goto done;
            }

         if (!hSource->Read(pbKeyBlob, dwKeyBlobLen))
            {
              sprintf(tmp_buf,"無法讀取檔頭！\n");
```

```
                ShowMessage(tmp_buf);
                goto done;
              }

          if (!CryptImportKey(hProv, pbKeyBlob, dwKeyBlobLen, 0, 0, &hKey))
              {
                sprintf(tmp_buf,"從 blob 中取得鍵值時發生錯誤！\n", GetLastError());
                ShowMessage(tmp_buf);
                goto done;
              }
        }
    else
        {
          if (!CryptCreateHash(hProv, CALG_MD5, 0, 0, &hHash))
              {
                sprintf(tmp_buf,"產生 Hash 物件時發生錯誤！\n", GetLastError());
                ShowMessage(tmp_buf);
                goto done;
              }

          if (!CryptHashData(hHash, StrPassword, strlen(StrPassword), 0))
              {
                sprintf(tmp_buf,"產生 Hash 值時發生錯誤！\n", GetLastError());
                ShowMessage(tmp_buf);
                goto done;
              }

          if (!CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, &hKey))
              {
                sprintf(tmp_buf,"取得 Hash 值時發生錯誤！\n", GetLastError());
                ShowMessage(tmp_buf);
                goto done;
              }
          CryptDestroyHash(hHash);
          hHash = 0;
        }

    dwBlockLen = 1000 - 1000 % ENCRYPT_BLOCK_SIZE;
    dwBufferLen = dwBlockLen;

    if ((pbBuffer = (char *) malloc(dwBufferLen)) == NULL)
        {
          sprintf(tmp_buf,"記憶體不足\n");
          ShowMessage(tmp_buf);
          goto done;
        }

    do {
          dwCount = hSource->Read(pbBuffer,dwBlockLen);
          if (!dwCount)
              {
                sprintf(tmp_buf,"無法讀取原始檔！\n");
                ShowMessage(tmp_buf);
                goto done;
              }

          eof = 1;

          if (!CryptDecrypt(hKey, 0, eof, 0, pbBuffer, &dwCount))
              {
                sprintf(tmp_buf,"解密時發生錯誤！\n", GetLastError());
                ShowMessage(tmp_buf);
                goto done;
              }

          if (!hDestination->Write(pbBuffer, dwCount))
              {
                sprintf(tmp_buf,"無法寫入目的檔！\n");
                ShowMessage(tmp_buf);
                goto done;
              }
    } while(hSource->Position<hSource->Size);


    strcpy(DestStr,hDestination->DataString.c_str());

    status = true;
    sprintf(tmp_buf,"檔案解密作業完成！\n");
    ShowMessage(tmp_buf);

done:
    if(hSource) delete hSource;
    if(hDestination) delete hDestination;

    if (pbKeyBlob && StrPassword == NULL) free(pbKeyBlob);
    if(pbBuffer) free(pbBuffer);
    if(hKey) CryptDestroyKey(hKey);
    if(hHash) CryptDestroyHash(hHash);
    if(hProv) CryptReleaseContext(hProv, 0);

    return(status);


}
```

函式名稱：字串加解密處理中相關的函式
函式功能：將字串轉成/還原 16 進位 ASCII 碼以便變數儲存應用

```
String __fastcall StrToHex(String sour)
{
   String d="";
   for (int i=1;i<=sour.Length();i++)
       {
          d=d+IntToHex((Byte)sour[i],2);
       }
   return (d);
}
//-------------------------------------------------------------------------
String __fastcall HexToStr(String sour)
{
   String tmp;
   char c;

   String d="";
   for (int i=1;i<=sour.Length()-1;i=i+2)
       {
         tmp=sour.SubString(i,2);
         d=d+char(StrToInt("0x"+tmp));
       }
   return (d);
}
```

函式使用範例：檔案加解密

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   String Source;
   String EnCrypt;
   String DeCrypt;
   if (OpenDialog1->Execute())
      {
        Source=OpenDialog1->FileName; //來源檔名
        EnCrypt=Source+".EnCrypt"; //加密後的檔名
        DeCrypt=Source+".DeCrypt"; //加密再解密後的檔名

        //NULL 的部分可填入密碼，但加解密都需同一個密碼
        EnCryptFile(Source.c_str(),EnCrypt.c_str(),NULL);
        DeCryptFile(EnCrypt.c_str(),DeCrypt.c_str(),NULL);
        ShowMessage(DeCrypt+" 檔案還原產生");
      }
}
```

函式使用範例：字串加解密

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   //將 Edit1 中的字串加密後放到 Edit2 中
   char tmp_buf[65535];
   EnCryptStr(Edit1->Text.c_str(),tmp_buf,NULL); // NULL 的部分可填入密碼
   Edit2->Text=tmp_buf;

   //將 Edit2 中的字串還原後放到 Edit3 中
   DeCryptStr(Edit2->Text.c_str(),tmp_buf,NULL); // NULL 的部分可填入密碼
   Edit3->Text=tmp_buf;
}
```

■ 以 Borland Delphi (5.0 版) 實做 CryptoAPI 檔案/字串加解密

Delphi 中相關的宣告 (較 C++Builder 繁雜，需自行使用靜態鏈結 'advapi32.dll' 中的函式)

```
……………..
……………..
var
  Form1: TForm1;

implementation

//--------------------------------------
//Crypto API 2.0 宣告區段
//--------------------------------------
const
  ALG_CLASS_DATA_ENCRYPT = (3 shl 13);
  ALG_TYPE_BLOCK             = (3 shl   9);
  ALG_TYPE_STREAM            = (4 shl   9);
  ALG_CLASS_HASH             = (4 shl 13);
  ALG_SID_RC2                =   2;
  ALG_SID_RC4                =   1;
  ALG_TYPE_ANY               =   0;
  ALG_SID_MD5                =   3;
  CALG_RC2 = (ALG_CLASS_DATA_ENCRYPT or ALG_TYPE_BLOCK or ALG_SID_RC2);
  CALG_RC4 = (ALG_CLASS_DATA_ENCRYPT or ALG_TYPE_STREAM or ALG_SID_RC4);
  CALG_MD5 = (ALG_CLASS_HASH or ALG_TYPE_ANY or ALG_SID_MD5);
  PROV_RSA_FULL              = 1;
  CRYPT_EXPORTABLE           = $00000001;
  CRYPT_NEWKEYSET            = $00000008;
  AT_KEYEXCHANGE             = 1;
  AT_SIGNATURE               = 2;
  SIMPLEBLOB                 = $1;
  PP_CONTAINER               = 6;

  {$IFDEF USE_BLOCK_CIPHER}
  ENCRYPT_ALGORITHM = CALG_RC2;
  ENCRYPT_BLOCK_SIZE = 8;
  {$ELSE}
  ENCRYPT_ALGORITHM = CALG_RC4;
  ENCRYPT_BLOCK_SIZE = 1;
  {$ENDIF}

  MS_DEF_PROV_A = 'Microsoft Base Cryptographic Provider v1.0';
  {$IFNDEF VER90}
  MS_DEF_PROV_W = WideString( 'Microsoft Base Cryptographic Provider v1.0');
  {$ELSE}
  MS_DEF_PROV_W = ( 'Microsoft Base Cryptographic Provider v1.0');
  {$ENDIF}

  {$IFDEF UNICODE}
  MS_DEF_PROV     = MS_DEF_PROV_W;
  {$ELSE}
  MS_DEF_PROV     = MS_DEF_PROV_A;
  {$ENDIF}

type HCRYPTPROV = ULONG;
     HCRYPTKEY   = ULONG;
     HCRYPTHASH = ULONG;
     ALG_ID       = ULONG;

     PHCRYPTPROV = ^HCRYPTPROV;
     PHCRYPTKEY   = ^HCRYPTKEY;
     PHCRYPTHASH = ^HCRYPTHASH;

     {$IFDEF UNICODE}
     LPAWSTR = PWideChar;
     {$ELSE}
```

```
    LPAWSTR = PAnsiChar;
    {$ENDIF}


{$IFDEF UNICODE}
function CryptAcquireContext(phProv       :PHCRYPTPROV;
                            pszContainer :LPAWSTR;
                            pszProvider  :LPAWSTR;
                            dwProvType   :DWORD;
                            dwFlags      :DWORD
                            ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptAcquireContextW';
{$ELSE}
function CryptAcquireContext(phProv       :PHCRYPTPROV;
                            pszContainer :LPAWSTR;
                            pszProvider  :LPAWSTR;
                            dwProvType   :DWORD;
                            dwFlags      :DWORD
                            ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptAcquireContextA';
{$ENDIF}


function CryptReleaseContext(hProv   :HCRYPTPROV;
                            dwFlags :DWORD
                            ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptReleaseContext';

function CryptGenKey(hProv   :HCRYPTPROV;
                     Algid   :ALG_ID;
                     dwFlags :DWORD;
                     phKey   :PHCRYPTKEY
                     ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptGenKey';

function CryptGetUserKey(hProv      :HCRYPTPROV;
                         dwKeySpec :DWORD;
                         phUserKey :PHCRYPTKEY
                         ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptGetUserKey';

function CryptExportKey(hKey        :HCRYPTKEY;
                        hExpKey     :HCRYPTKEY;
                        dwBlobType :DWORD;
                        dwFlags     :DWORD;
                        pbData      :PBYTE;
                        pdwDataLen :PDWORD
                        ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptExportKey';

function CryptDestroyKey(hKey   :HCRYPTKEY
                        ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptDestroyKey';

function CryptCreateHash(hProv   :HCRYPTPROV;
                         Algid   :ALG_ID;
                         hKey    :HCRYPTKEY;
                         dwFlags :DWORD;
                         phHash  :PHCRYPTHASH
                         ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptCreateHash';

function CryptHashData(hHash        :HCRYPTHASH;
                 const  pbData       :PBYTE;
                        dwDataLen    :DWORD;
                        dwFlags      :DWORD
                        ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptHashData';

function CryptDeriveKey(hProv      :HCRYPTPROV;
                        Algid      :ALG_ID;
                        hBaseData :HCRYPTHASH;
                        dwFlags    :DWORD;
                        phKey      :PHCRYPTKEY
                        ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptDeriveKey';

function CryptDestroyHash(hHash :HCRYPTHASH
                         ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptDestroyHash';
```

```
function CryptEncrypt(hKey          :HCRYPTKEY;
                      hHash         :HCRYPTHASH;
                      Final         :BOOL;
                      dwFlags       :DWORD;
                      pbData        :PBYTE;
                      pdwDataLen :PDWORD;
                      dwBufLen      :DWORD
                      ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptEncrypt';

function CryptImportKey(hProv       :HCRYPTPROV;
                        pbData      :PBYTE;
                        dwDataLen :DWORD;
                        hPubKey     :HCRYPTKEY;
                        dwFlags     :DWORD;
                        phKey       :PHCRYPTKEY
                        ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptImportKey';

function CryptDecrypt(hKey          :HCRYPTKEY;
                      hHash         :HCRYPTHASH;
                      Final         :BOOL;
                      dwFlags       :DWORD;
                      pbData        :PBYTE;
                      pdwDataLen :PDWORD
                      ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptDecrypt';

function CryptGetProvParam(hProv        :HCRYPTPROV;
                           dwParam      :DWORD;
                           pbData       :PBYTE;
                           pdwDataLen :PDWORD;
                           dwFlags      :DWORD
                           ) :BOOL; stdcall; external 'advapi32.dll' name 'CryptGetProvParam';

{$R *.DFM}
```

函式名稱：InitUser
函式功能：在系統環境中建立 CryptoAPI 的基礎鍵值資料庫

```
procedure InitUser;
var
  hProv : HCRYPTPROV;
  hKey   : HCRYPTKEY;
  dwUserNameLen : DWORD;
  szUserName : array [0..100] of char;
  tmp_buf : array [0..80] of char;

begin

    hProv:=0;
    hKey:=0;
    dwUserNameLen := 100;

    // Attempt to acquire a handle to the default key container.
    if not(CryptAcquireContext(@hProv, nil, MS_DEF_PROV, PROV_RSA_FULL, 0))   then
       begin
         // Some sort of error occured.

         // Create default key container.
         if not(CryptAcquireContext(@hProv, nil, MS_DEF_PROV, PROV_RSA_FULL,
CRYPT_NEWKEYSET)) then
            begin
              ShowMessage('Error creating key container!');
              Exit;
            end;

         // Get name of default key container.
         if not(CryptGetProvParam(hProv, PP_CONTAINER, (@szUserName), @dwUserNameLen, 0))
then
            begin
              // Error getting key container name.
              szUserName[0] := Char(0);
            end;

         ShowMessage('Create key container '+String(szUserName));

       end;

    // Attempt to get handle to signature key.
    if not CryptGetUserKey(hProv, AT_SIGNATURE, @hKey) then
       begin
         if (GetLastError()=DWORD(NTE_NO_KEY)) then  //不加 DWORD() 永遠也比不出來...Delphi 5
之 bug ??
            begin
              // Create signature key pair.
              ShowMessage('Create signature key pair');

              if not(CryptGenKey(hProv,AT_SIGNATURE,0,@hKey)) then
                 begin
                   ShowMessage('Error during CryptGenKey (1)!');
                   Exit;
                 end
              else
                 begin
                   CryptDestroyKey(hKey);
                 end;

            end
         else
            begin
              ShowMessage('Error during CryptGetUserKey (1)!');
              Exit;
            end;
       end;

    // Attempt to get handle to exchange key.
    if not(CryptGetUserKey(hProv,AT_KEYEXCHANGE,@hKey)) then
       begin

         if(GetLastError()=DWORD(NTE_NO_KEY)) then  //不加 DWORD() 永遠也比不出來...Delphi 5 之
bug ??
           begin
             // Create key exchange key pair.
             ShowMessage('Create key exchange key pair');

             if not(CryptGenKey(hProv,AT_KEYEXCHANGE,0,@hKey)) then
                begin
                  ShowMessage('Error during CryptGenKey (2)!');
                  Exit;
                end
             else
                begin
                  CryptDestroyKey(hKey);
                end;

           end
         else
           begin
             ShowMessage('Error during CryptGetUserKey (2)!');
             Exit;
           end;

       end;

    CryptReleaseContext(hProv,0);

end;
```

函式名稱：EnCryptFile(SourFile:String ; DestFile:String ; FilePassword:String)
函式功能：將原始檔案加密並轉出成目的檔案

```
function EnCryptFile(SourFile:String ; DestFile:String ; FilePassword:String):Boolean;
var
  hSource : TFileStream;
  hDestination : TFileStream;
  hProv : HCRYPTPROV;
  hKey   : HCRYPTKEY;
  hXchgKey : HCRYPTKEY;
  hHash : HCRYPTHASH;

  pbKeyBlob : PByte ;
  pbBuffer   : PByte;

  dwKeyBlobLen : DWORD;
  dwBlockLen      : DWORD;
  dwBufferLen    : DWORD;
  dwCount          : DWORD;

  eof : Boolean;
  status : Boolean;
  tmp_buf : array [0..80] of char;

label done;

begin

    hProv     := 0;
    hKey        := 0;
    hXchgKey := 0;
    hHash     := 0;

    eof:=false;
    status:=false;

    hSource := TFileStream.Create(SourFile, fmOpenRead);
    if (hSource=nil) then
       begin
         ShowMessage('無法開啓原始檔案!');
         goto done;
       end;

    hDestination := TFileStream.Create(DestFile, fmCreate);
    if (hDestination=nil) then
       begin
         ShowMessage('無法開啓目的檔案!');
         goto done;
       end;


    InitUser;

    if not CryptAcquireContext(@hProv,   nil, nil, PROV_RSA_FULL, 0) then
       begin
         ShowMessage('無法取得 Handle ');
         goto done;
       end;


    if (FilePassword = '') then //不輸入加密 Key 時，系統自動產生 Key 於目的檔頭，故加密後的檔案 Size
會比原來大
       begin

         if not(CryptGenKey(hProv, ENCRYPT_ALGORITHM, CRYPT_EXPORTABLE,@hKey)) then
            begin
              ShowMessage('產生隨機鍵值時發生錯誤！');
              goto done;
            end;

         if not(CryptGetUserKey(hProv, AT_KEYEXCHANGE, @hXchgKey)) then
            begin
              ShowMessage('交換鍵値時發生錯誤！');
              goto done;
            end;

         if not(CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, nil, @dwKeyBlobLen)) then
            begin
              ShowMessage('由 blof 取出鍵値時發生錯誤！');
              goto done;
            end;

         GetMem(pbKeyBlob, dwKeyBlobLen);
         if (pbKeyBlob=nil) then
            begin
              ShowMessage('記憶體不足');
              goto done;
            end;

         if not(CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, pbKeyBlob,@dwKeyBlobLen)) then
            begin
              ShowMessage('傳回簽名時發生錯誤！');
              goto done;
            end;

         CryptDestroyKey(hXchgKey);
         hXchgKey := 0;

         if (hDestination.Write(dwKeyBlobLen, sizeof(DWORD))=0) then
            begin
              ShowMessage('無法寫入檔頭！');
              goto done;
            end;

         if (hDestination.Write(pbKeyBlob^,dwKeyBlobLen)=0) then
            begin
              ShowMessage('無法寫入檔頭！');
              goto done;
            end;

       end
    else
       begin

         if not(CryptCreateHash(hProv, CALG_MD5, 0, 0, @hHash)) then
            begin
              ShowMessage('產生 Hash 物件時發生錯誤！');
              goto done;
            end;

         if not(CryptHashData(hHash, PByte(FilePassword), Length(FilePassword), 0)) then
            begin
              ShowMessage('無法產生 Hash 値！');
              goto done;
            end;

         if not(CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, @hKey)) then
            begin
              ShowMessage('Hash 物件無法傳回鍵値');
              goto done;
            end;

         CryptDestroyHash(hHash);
         hHash := 0;

       end;
    dwBlockLen := (1000 - (1000 mod ENCRYPT_BLOCK_SIZE));

    if (ENCRYPT_BLOCK_SIZE > 1) then
        dwBufferLen := (dwBlockLen + ENCRYPT_BLOCK_SIZE)
    else
        dwBufferLen := dwBlockLen;

    GetMem(pbBuffer, dwBufferLen);
    if (pbBuffer=nil) then
       begin
         ShowMessage('記憶體不足');
         goto done;
       end;


    repeat

        dwCount := hSource.Read(pbBuffer^,dwBlockLen); //光是 這個 ^ 符號 , 就 error & try 了 3,4
天 - 2003/04/22
        if (dwCount=0) then
           begin
             ShowMessage('無法開啓原始檔！');
             goto done;
           end;

        eof:=true;

        if not CryptEncrypt(hKey, 0, eof , 0, pbBuffer, @dwCount, dwBufferLen) then
           begin
             ShowMessage('加密緩衝區資料發生錯誤！');
             goto done;
           end;

        if hDestination.Write(pbBuffer^,dwCount)=0 then
           begin
             ShowMessage('無法寫入目的檔！');
             goto done;
           end;

    until not(hSource.Position<hSource.Size);

    status := true;
    ShowMessage('檔案加密作業完成！');

 done:

    if(hSource<>nil) then hSource.free;
    if(hDestination<>nil) then hDestination.free;

    if((pbKeyBlob<>nil) and (FilePassword = '')) then FreeMem(pbKeyBlob);
    if(pbBuffer<>nil) then FreeMem(pbBuffer);
    if(hKey<>0) then CryptDestroyKey(hKey);
    if(hXchgKey<>0) then CryptDestroyKey(hXchgKey);
    if(hHash<>0) then CryptDestroyHash(hHash);
    if(hProv<>0) then CryptReleaseContext(hProv, 0);

    Result := status;

end;
```

函式名稱：DeCryptFile(SourFile:String ; DestFile:String ; FilePassword:String)
函式功能：將加密檔案還原並轉出成目的檔案

```
function DeCryptFile(SourFile:String ; DestFile:String ; FilePassword:String):Boolean;
var
   hSource, hDestination: TFileStream;
   hProv : HCRYPTPROV;
   hKey   : HCRYPTKEY;
   hXchgKey : HCRYPTKEY;
   hHash : HCRYPTHASH;

   pbKeyBlob : PByte;
   pbBuffer   : PByte;

   dwKeyBlobLen : DWORD;
   dwBlockLen     : DWORD;
   dwBufferLen    : DWORD;
   dwCount         : DWORD;

   eof : Boolean;
   status : Boolean;
   tmp_buf : array [0..80] of char;

label done;

begin

      hProv     := 0;
      hKey       := 0;
      hXchgKey := 0;
      hHash     := 0;

      eof:=false;
      status:=false;

      hSource := TFileStream.Create(SourFile, fmOpenRead);
      if (hSource=nil) then
          begin
            ShowMessage('無法開啓原始檔案!');
            goto done;
          end;

      hDestination := TFileStream.Create(DestFile, fmCreate);
      if (hDestination=nil) then
          begin
            ShowMessage('無法開啓目的檔案!');
            goto done;
          end;



      InitUser;

      if not(CryptAcquireContext(@hProv,   nil, nil, PROV_RSA_FULL, 0)) then
          begin
            ShowMessage('無法取得 Handle ');
            goto done;
          end;


      if (FilePassword = '') then
          begin

            if (hSource.Read(dwKeyBlobLen, sizeof(DWORD))=0) then
                begin
                  ShowMessage('無法讀取檔頭！');
                  goto done;
                end;

            GetMem(pbKeyBlob, dwKeyBlobLen);
            if (pbKeyBlob=nil) then
                begin
                  ShowMessage('記憶體不足');
                  goto done;
                end;

            if (hSource.Read(pbKeyBlob^, dwKeyBlobLen)=0) then
                begin
                  ShowMessage('無法讀取檔頭！');
                  goto done;
                end;

            if not(CryptImportKey(hProv, pbKeyBlob, dwKeyBlobLen, 0, 0, @hKey)) then
                begin
                  ShowMessage('從 blob 中取得鍵值時發生錯誤！');
                  goto done;
                end;
          end
      else
          begin

            if not(CryptCreateHash(hProv, CALG_MD5, 0, 0, @hHash)) then
                begin
                  ShowMessage('產生 Hash 物件時發生錯誤！');
                  goto done;
                end;

            if not(CryptHashData(hHash, PByte(FilePassword), Length(FilePassword), 0)) then
                begin
                  ShowMessage('產生 Hash 值時發生錯誤！');
                  goto done;
                end;

            if not(CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, @hKey)) then
                begin
                  ShowMessage('取得 Hash 值時發生錯誤！');
                  goto done;
                end;

            CryptDestroyHash(hHash);
            hHash := 0;

          end;

      dwBlockLen := 1000 - (1000 mod ENCRYPT_BLOCK_SIZE);
      dwBufferLen := dwBlockLen;

      GetMem(pbBuffer, dwBufferLen);
      if (pbBuffer=nil) then
          begin
            ShowMessage('記憶體不足');
            goto done;
          end;


      repeat

            dwCount:=hSource.Read(pbBuffer^,dwBlockLen);
            if (dwCount=0) then
                begin
                  ShowMessage('無法讀取原始檔！');
                  goto done;
                end;

            eof := true;
```

```
          if not(CryptDecrypt(hKey, 0, eof, 0, pbBuffer, @dwCount)) then
             begin
               ShowMessage('解密時發生錯誤！');
               goto done;
             end;

          if (hDestination.Write(pbBuffer^, dwCount)=0) then
             begin
               ShowMessage('無法寫入目的檔！');
               goto done;
             end;

    until not(hSource.Position<hSource.Size);

    status := true;
    ShowMessage('檔案解密作業完成！');

done:

    if(hSource<>nil) then hSource.free;
    if(hDestination<>nil) then hDestination.free;

    if((pbKeyBlob<>nil) and (FilePassword = '')) then FreeMem(pbKeyBlob);
    if(pbBuffer<>nil) then FreeMem(pbBuffer);
    if(hKey<>0) then CryptDestroyKey(hKey);
    if(hXchgKey<>0) then CryptDestroyKey(hXchgKey);
    if(hHash<>0) then CryptDestroyHash(hHash);
    if(hProv<>0) then CryptReleaseContext(hProv, 0);

    Result := status;

end;
```

函式名稱：EnCryptStr(SourStr:String ; var DestStr:String ; StrPassword:String)
函式功能：將字串加密成另一個經過加密編碼的字串

```
function EnCryptStr(SourStr:String ; var DestStr:String ; StrPassword:String):Boolean;
var
  hSource : TStringStream;
  hDestination : TStringStream;
  hProv : HCRYPTPROV;
  hKey   : HCRYPTKEY;
  hXchgKey : HCRYPTKEY;
  hHash : HCRYPTHASH;

  pbKeyBlob : PByte ;
  pbBuffer   : PByte;

  dwKeyBlobLen : DWORD;
  dwBlockLen     : DWORD;
  dwBufferLen   : DWORD;
  dwCount         : DWORD;

  eof : Boolean;
  status : Boolean;
  tmp_buf : array [0..80] of char;

label done;

begin

    hProv     := 0;
    hKey       := 0;
    hXchgKey := 0;
    hHash     := 0;

    eof:=false;
    status:=false;

    hSource := TStringStream.Create(SourStr);
    if (hSource=nil) then
       begin
         ShowMessage('無法開啓原始檔案!');
         goto done;
       end;

    hDestination := TStringStream.Create('');

    InitUser;

    if not CryptAcquireContext(@hProv,   nil, nil, PROV_RSA_FULL, 0) then
       begin
         ShowMessage('無法取得 Handle ');
         goto done;
       end;


    if (StrPassword = '') then //不輸入加密 Key 時，系統自動產生 Key 於目的檔頭，故加密後的檔案 Size
會比原來大
       begin

         if not(CryptGenKey(hProv, ENCRYPT_ALGORITHM, CRYPT_EXPORTABLE,@hKey)) then
            begin
              ShowMessage('產生隨機鍵值時發生錯誤！');
              goto done;
            end;

         if not(CryptGetUserKey(hProv, AT_KEYEXCHANGE, @hXchgKey)) then
            begin
              ShowMessage('交換鍵值時發生錯誤！');
              goto done;
            end;

         if not(CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, nil, @dwKeyBlobLen)) then
            begin
              ShowMessage('由 blof 取出鍵值時發生錯誤！');
              goto done;
            end;

         GetMem(pbKeyBlob, dwKeyBlobLen);
         if (pbKeyBlob=nil) then
            begin
              ShowMessage('記憶體不足');
              goto done;
            end;

         if not(CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, pbKeyBlob,@dwKeyBlobLen)) then
            begin
              ShowMessage('傳回簽名時發生錯誤！');
              goto done;
            end;

         CryptDestroyKey(hXchgKey);
         hXchgKey := 0;

         if (hDestination.Write(dwKeyBlobLen, sizeof(DWORD))=0) then
            begin
              ShowMessage('無法寫入檔頭！');
              goto done;
            end;

         //ShowMessage(IntToStr(Length(String(pbKeyBlob))));  //好像固定都爲 82

         if (hDestination.Write(pbKeyBlob^,dwKeyBlobLen)=0) then
            begin
              ShowMessage('無法寫入檔頭！');
              goto done;
            end;

       end
    else
       begin

         if not(CryptCreateHash(hProv, CALG_MD5, 0, 0, @hHash)) then
            begin
              ShowMessage('產生 Hash 物件時發生錯誤！');
              goto done;
            end;

         if not(CryptHashData(hHash, PByte(StrPassword), Length(StrPassword), 0)) then
            begin
              ShowMessage('無法產生 Hash 値！');
              goto done;
            end;

         if not(CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, @hKey)) then
            begin
              ShowMessage('Hash 物件無法傳回鍵值');
              goto done;
            end;

         CryptDestroyHash(hHash);
         hHash := 0;

       end;

    dwBlockLen := (1000 - (1000 mod ENCRYPT_BLOCK_SIZE));

    if (ENCRYPT_BLOCK_SIZE > 1) then
```

```
        dwBufferLen := (dwBlockLen + ENCRYPT_BLOCK_SIZE)
    else
        dwBufferLen := dwBlockLen;

    GetMem(pbBuffer, dwBufferLen);
    if (pbBuffer=nil) then
       begin
         ShowMessage('記憶體不足');
         goto done;
       end;


    repeat

        dwCount := hSource.Read(pbBuffer^,dwBlockLen); //光是 這個 ^ 符號 , 就 error & try 了 3,4
天 - 2003/04/22
        if (dwCount=0) then
           begin
             ShowMessage('無法開啓原始檔！');
             goto done;
           end;

        eof:=true;

        if not CryptEncrypt(hKey, 0, eof , 0, pbBuffer, @dwCount, dwBufferLen) then
           begin
             ShowMessage('加密緩衝區資料發生錯誤！');
             goto done;
           end;

        if hDestination.Write(pbBuffer^,dwCount)=0 then
           begin
             ShowMessage('無法寫入目的檔！');
             goto done;
           end;

    until not(hSource.Position<hSource.Size);

    //怕字串串流會有中間有 null 的問題 , 所以需要將加密結果轉成 hex 再回傳
    DestStr:=StrToHex(hDestination.DataString);

    status := true;
    ShowMessage('檔案加密作業完成！');

 done:

    if(hSource<>nil) then hSource.free;
    if(hDestination<>nil) then hDestination.free;

    if((pbKeyBlob<>nil) and (StrPassword = '')) then FreeMem(pbKeyBlob);
    if(pbBuffer<>nil) then FreeMem(pbBuffer);
    if(hKey<>0) then CryptDestroyKey(hKey);
    if(hXchgKey<>0) then CryptDestroyKey(hXchgKey);
    if(hHash<>0) then CryptDestroyHash(hHash);
    if(hProv<>0) then CryptReleaseContext(hProv, 0);

    Result := status;

end;
```

函式名稱：DeCryptStr(SourStr:String ; var DestStr:String ; StrPassword:String)
函式功能：將加密字串還原

```
function DeCryptStr(SourStr:String ; var DestStr:String ; StrPassword:String):Boolean;
var
  hSource, hDestination: TStringStream;
  hProv : HCRYPTPROV;
  hKey   : HCRYPTKEY;
  hXchgKey : HCRYPTKEY;
  hHash : HCRYPTHASH;

  pbKeyBlob : PByte;
  pbBuffer   : PByte;

  dwKeyBlobLen : DWORD;
  dwBlockLen     : DWORD;
  dwBufferLen   : DWORD;
  dwCount         : DWORD;

  eof : Boolean;
  status : Boolean;
  tmp_buf : array [0..80] of char;

  ss : String;


label done;

begin

    hProv     := 0;
    hKey       := 0;
    hXchgKey := 0;
    hHash     := 0;

    eof:=false;
    status:=false;

    //將 HEX 解碼
    ss:=HexToStr(SourStr);

    hSource := TStringStream.Create(ss);
    if (hSource=nil) then
       begin
         ShowMessage('無法開啓原始檔案!');
         goto done;
       end;

    hDestination := TStringStream.Create('');

    InitUser;

    if not(CryptAcquireContext(@hProv,   nil, nil, PROV_RSA_FULL, 0)) then
       begin
         ShowMessage('無法取得 Handle ');
         goto done;
       end;


    if (StrPassword = '') then
       begin

         if (hSource.Read(dwKeyBlobLen, sizeof(DWORD))=0) then
            begin
              ShowMessage('無法讀取檔頭！');
              goto done;
            end;
```

```
        GetMem(pbKeyBlob, dwKeyBlobLen);
        if (pbKeyBlob=nil) then
            begin
              ShowMessage('記憶體不足');
              goto done;
            end;

        if (hSource.Read(pbKeyBlob^, dwKeyBlobLen)=0) then
            begin
              ShowMessage('無法讀取檔頭！');
              goto done;
            end;

        if not(CryptImportKey(hProv, pbKeyBlob, dwKeyBlobLen, 0, 0, @hKey)) then
            begin
              ShowMessage('從 blob 中取得鍵值時發生錯誤！');
              goto done;
            end;
      end
    else
      begin

        if not(CryptCreateHash(hProv, CALG_MD5, 0, 0, @hHash)) then
            begin
              ShowMessage('產生 Hash 物件時發生錯誤！');
              goto done;
            end;

        if not(CryptHashData(hHash, PByte(StrPassword), Length(StrPassword), 0)) then
            begin
              ShowMessage('產生 Hash 值時發生錯誤！');
              goto done;
            end;

        if not(CryptDeriveKey(hProv, ENCRYPT_ALGORITHM, hHash, 0, @hKey)) then
            begin
              ShowMessage('取得 Hash 值時發生錯誤！');
              goto done;
            end;

        CryptDestroyHash(hHash);
        hHash := 0;

      end;

    dwBlockLen := 1000 - (1000 mod ENCRYPT_BLOCK_SIZE);
    dwBufferLen := dwBlockLen;

    GetMem(pbBuffer, dwBufferLen);
    if (pbBuffer=nil) then
        begin
          ShowMessage('記憶體不足');
          goto done;
        end;


    repeat

        dwCount:=hSource.Read(pbBuffer^,dwBlockLen);
        if (dwCount=0) then
            begin
              ShowMessage('無法讀取原始檔！');
              goto done;
            end;

        eof := true;

        if not(CryptDecrypt(hKey, 0, eof, 0, pbBuffer, @dwCount)) then
```

```
            begin
              ShowMessage('解密時發生錯誤！');
              goto done;
            end;

         if (hDestination.Write(pbBuffer^, dwCount)=0) then
            begin
              ShowMessage('無法寫入目的檔！');
              goto done;
            end;

    until not(hSource.Position<hSource.Size);

    DestStr:=hDestination.DataString;

    status := true;
    ShowMessage('檔案解密作業完成！');

done:

    if(hSource<>nil) then hSource.free;
    if(hDestination<>nil) then hDestination.free;

    if((pbKeyBlob<>nil) and (StrPassword = '')) then FreeMem(pbKeyBlob);
    if(pbBuffer<>nil) then FreeMem(pbBuffer);
    if(hKey<>0) then CryptDestroyKey(hKey);
    if(hXchgKey<>0) then CryptDestroyKey(hXchgKey);
    if(hHash<>0) then CryptDestroyHash(hHash);
    if(hProv<>0) then CryptReleaseContext(hProv, 0);

    Result := status;

end;
```

函式名稱：字串加解密處理中相關的函式

函式功能：將字串轉成/還原 16 進位 ASCII 碼以便變數儲存應用

```
function StrToHex(S: string): string;
var
  i: integer;
begin
  Result := '';
  for i := 1 to Length( S ) do
      Result := Result + IntToHex( Ord( S[i] ), 2 );
end;
//--------------------------------------------------------------------------
function HexToStr(S: string): string;
var
  i: integer;
begin
  Result := '';
  for i := 1 to Length( S ) do
  begin
    if ((i mod 2) = 1) then
            Result := Result + Chr( StrToInt( '0x' + Copy( S, i, 2 )));
  end;
end;
```

函式使用範例：檔案加解密

```
procedure TForm1.Button1Click(Sender: TObject);
var Source,EnCrypt,DeCrypt : String;
begin
  if (OpenDialog1.Execute()) then
      begin
        Source:=OpenDialog1.FileName;
        EnCrypt:=Source+'.EnCrypt';
        DeCrypt:=Source+'.DeCrypt';

        EnCryptFile(Source,EnCrypt,'');
        DeCryptFile(EnCrypt,DeCrypt,'');
        ShowMessage(DeCrypt+' 檔案還原產生');
      end;
end;
```

函式使用範例：字串加解密

```
procedure TForm1.Button2Click(Sender: TObject);
var tmp_buf : String;
begin
  //將 Edit1 中的字串加密後放到 Edit2 中
  EnCryptStr(Edit1.Text,tmp_buf,'');
  Edit2.Text:=tmp_buf;

  //將 Edit2 中的字串還原後放到 Edit3 中
  DeCryptStr(Edit2.Text,tmp_buf,'');
  Edit3.Text:=tmp_buf;
end;
```