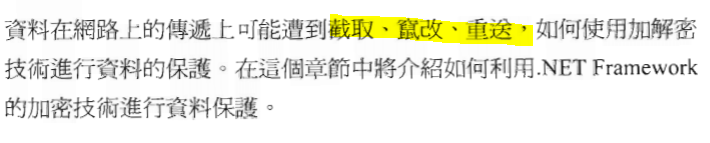
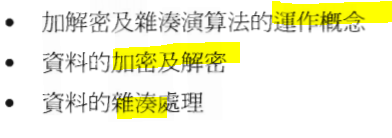
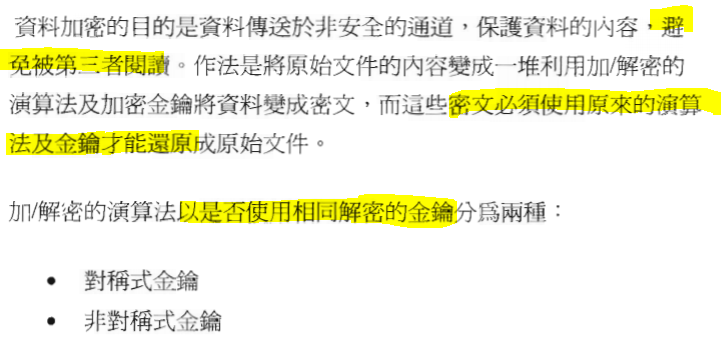
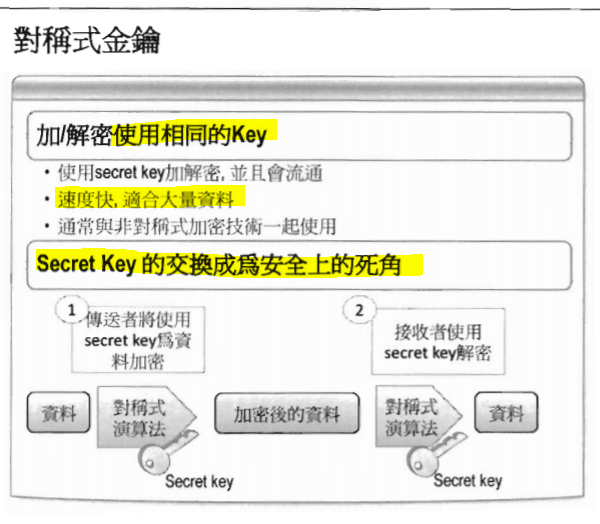
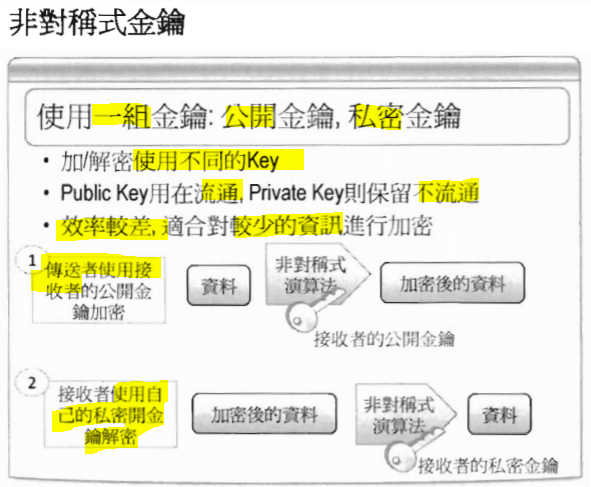
P 546





了解對稱﹑非對稱金鑰﹑雜湊演算法原理



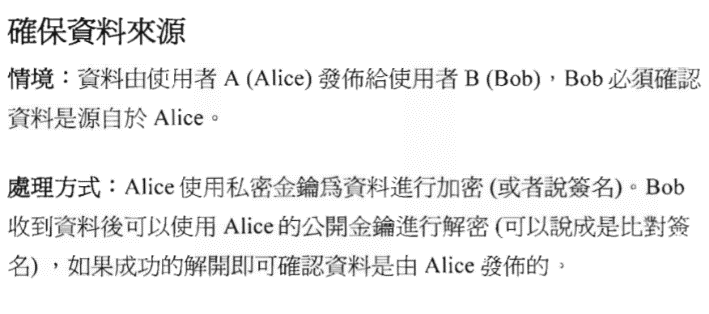
非對稱的話，比如說︰

我是要那個資料的人，我給你一個帶鎖的箱子，你把資料放入鎖好。

只有我有那個箱子的鎖的鑰匙，當箱子送回來到我手上時，我就可以看到資料了。而且我確保過程中沒有人看過，因為鑰匙一直都只有在我的手上。

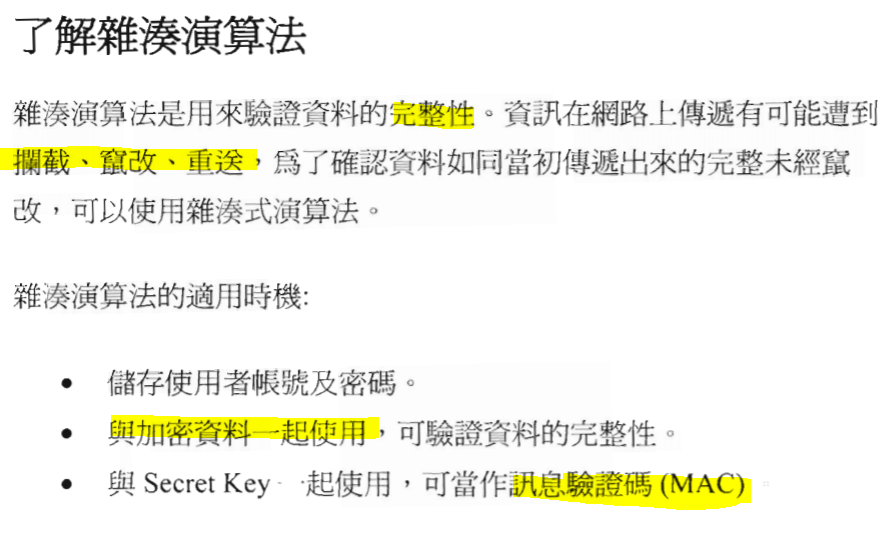
對稱的更有效率處理大量資料，但是非對稱更安全，所以對稱的「secret key」通常用非對稱的方法傳遞。

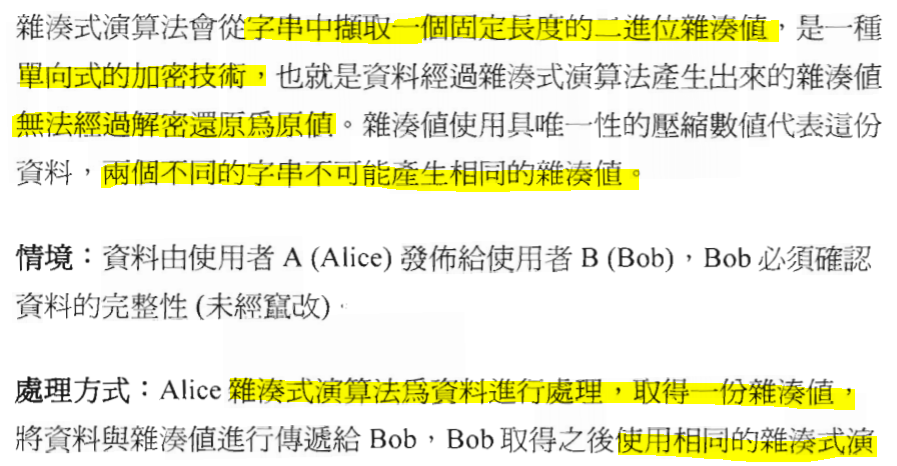
非對稱的另一個用法，比如說︰

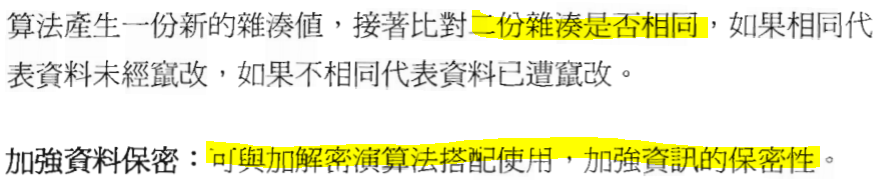
B 收到A寄來的箱子。

他用A發給所有人的鑰匙打開。所以他確定這是A寄給他的箱子。

因為只有A會做A鑰匙可以打開的鎖。







比如說︰

A 寄三個箱子給 B，B知道是A寄來的，因為B用A的公鑰打開了。

但是他懷疑箱子少了一個。

但以他拿出A在第一個箱子中放入的物品清單，點算一下。

發現A原本應該寄出了四個箱子，還有一個不見了。

這個「物品清單」

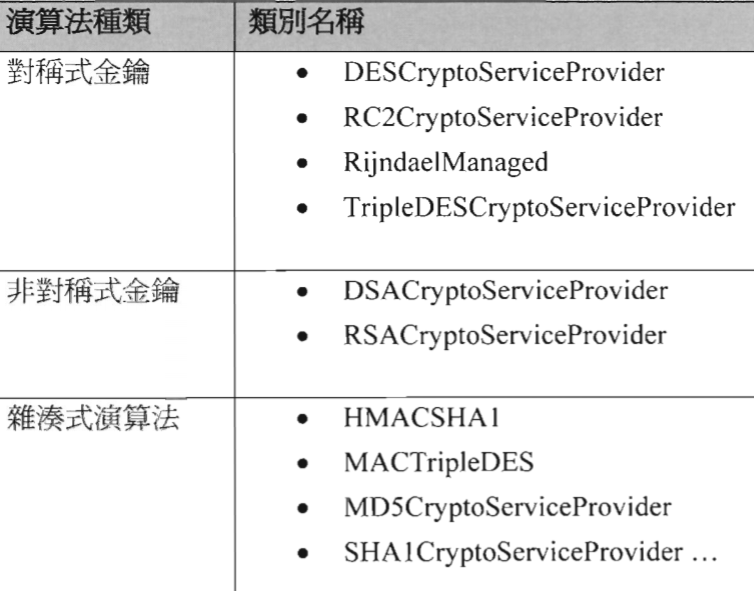
是 A 用演算法從「所有物件的加總」產生的，解不開的雜湊碼。( 列清單 )

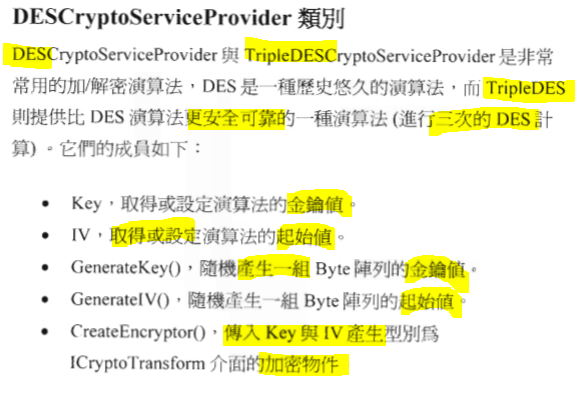
所以「物品清單」是不可能被修改的。

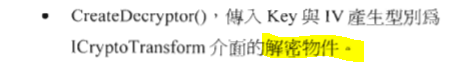
B 是用相同的演算法，從他收到的「所有物件的加總」中產生雜湊碼。(點算)

所以當B看到兩個雜湊碼不一樣，就知道「物品清單」在送出和收到時不同。

P 557 如何實現

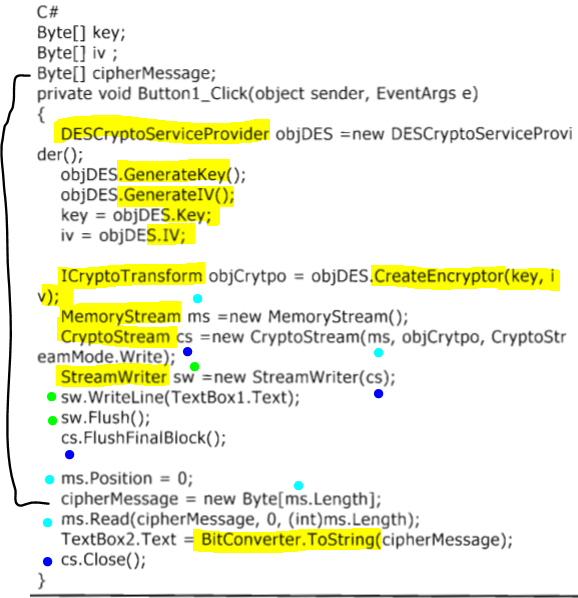




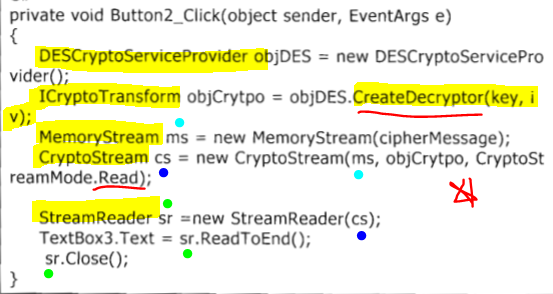




對稱加密︰

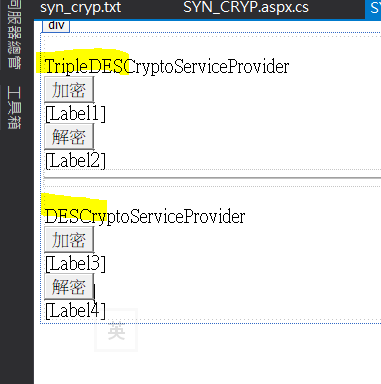


對稱解密︰



P 563 實作對稱 note3\_ex3.SYN\_CRYP 後改名 CRYP\_1.aspx (因為下方的練習是它的延續)

感覺奇怪的地方，是TripleDESCryptoServiceProvider 把 KEY 和 IV 直接寫在被加密檔案的開頭，所以是不是知道了「加密」的方法，檔案就可以被破解了？

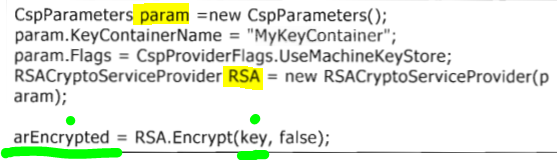


這是加密檔案

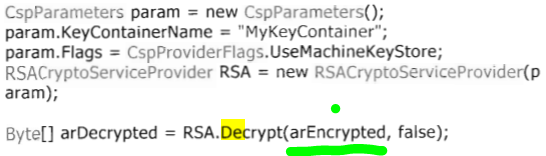
我連之前示範的都實作出來。

這個是加密句子

P 570 實作非對稱



Key 是從「對稱」的那一邊提供的。



P 572 練習︰ note3\_ex3.ASYN\_CRYP 後改名 CRYP\_2.aspx

沿上一個練習，只改加密和解密方法的一小段 (紅字)

byte[] ProcessEncrypto(Stream sourceStream)

{

// 加密物件

Byte[] key, iv;

key = objDES.Key;

iv = objDES.IV;

ICryptoTransform objCrytpo = objDES.CreateEncryptor(key, iv);

// 把加密物件 加入 Straem 的處理

MemoryStream ms = new MemoryStream();

BinaryWriter writer = new BinaryWriter(ms);

writer.Write(ProtectKey(key)); // 由「非對稱」再將 它們加密一次

writer.Write(ProtectKey(iv));

// 把來源的 Stream 寫到新的加密 Stream

using (CryptoStream cs = new CryptoStream(ms, objCrytpo, CryptoStreamMode.Write))

{

Byte[] buffer = new Byte[1024];

int readCount;

do

{

readCount = sourceStream.Read(buffer, 0, 1024);

cs.Write(buffer, 0, readCount); // <-- 寫入

Array.Clear(buffer, 0, 1024);

}

while (readCount == 1024); // 因為寫到最後一個的時候，就不是 == buffersize了

cs.FlushFinalBlock();

return ms.ToArray(); // 回傳 已透過 cs 處理好的 ms 給召喚的方法

}

}

byte[] ProcessDecrypto(Stream sourceStream)

{

// 讀取來源

byte[] key, iv, data;

BinaryReader reader = new BinaryReader(sourceStream);

// 對稱時 24 和 8 :

//key = reader.ReadBytes(24);// 前 24 節是 Key 的意思？//<-- 反正他們這樣反而拿得到和加密時一樣的 KEY

//iv = reader.ReadBytes(8);// 24 後的 8 節是 IV 的意思？

////key = objDES.Key; // <-- 這樣是不對的，因為每次拿出的Key 都不一樣？我也不知道為什麼會這樣。

////iv = objDES.IV;//<---- 拿錯的 KEY 和 IV 去解密，會在 cs.Read 那裡出問題。

// 不對稱時 128, 128 ?︰

key = UnprotectKey(reader.ReadBytes(128));

iv = UnprotectKey(reader.ReadBytes(128));

data = reader.ReadBytes((int)sourceStream.Length);

// 準備物件

MemoryStream encMS = new MemoryStream(data);

MemoryStream decMS = new MemoryStream();

objDES = new TripleDESCryptoServiceProvider();

ICryptoTransform objCrytpo = objDES.CreateDecryptor(key, iv);

// 讀取及解密

using (CryptoStream cs = new CryptoStream(encMS, objCrytpo, CryptoStreamMode.Read))

{

Byte[] buffer = new byte[1024];

int readCount;

do

{

readCount = cs.Read(buffer, 0, 1024);//<-- 以解密讀取

decMS.Write(buffer, 0, readCount);//<-- 寫到另一個 Stream

Array.Clear(buffer, 0, 1024);// 清空暫存

}

while (readCount == 1024);

decMS.Flush();

return decMS.ToArray();

}

}

// 被引用的方法︰

/// <summary>

/// 以私Key加密

/// </summary>

/// <param name="key"></param>

/// <returns></returns>

byte[] ProtectKey(Byte[] key)

{

// 設定參數 (都是公Key?因為沒有另一台電腦？)

CspParameters param = new CspParameters();

param.KeyContainerName = "MyKeyContainer";

param.Flags = CspProviderFlags.UseMachineKeyStore;

// 產生非對稱加密

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(param);

return RSA.Encrypt(key, false);

}

/// <summary>

/// 以公Key解密

/// </summary>

/// <param name="key"></param>

/// <returns></returns>

byte[] UnprotectKey(byte[] key)

{

// 設定參數 (都是公Key?因為沒有另一台電腦？)

CspParameters param = new CspParameters();

param.KeyContainerName = "MyKeyContainer";

param.Flags = CspProviderFlags.UseMachineKeyStore;

// 產生非對稱解密

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(param);

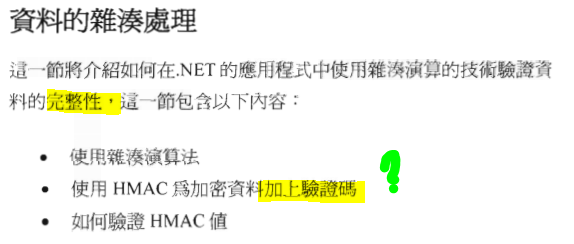
return RSA.Decrypt(key, false);

}

上述是檔案，下面是文字︰

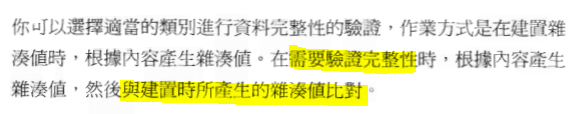


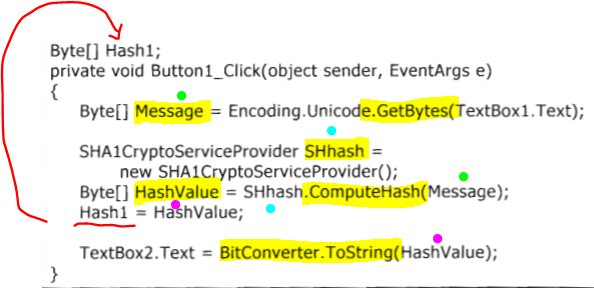
P 575 雜湊

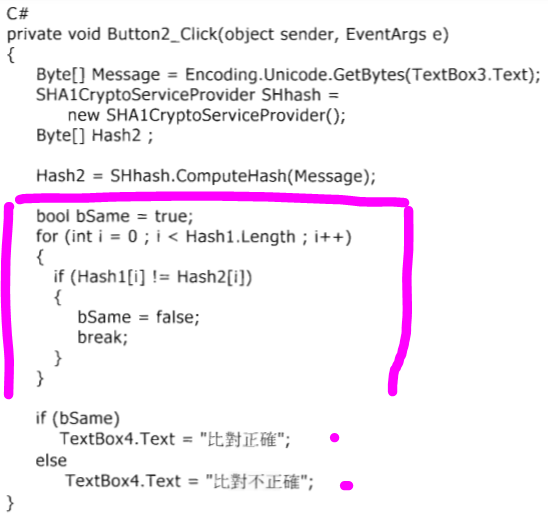


驗證碼就是這個「清單」的內容吧？

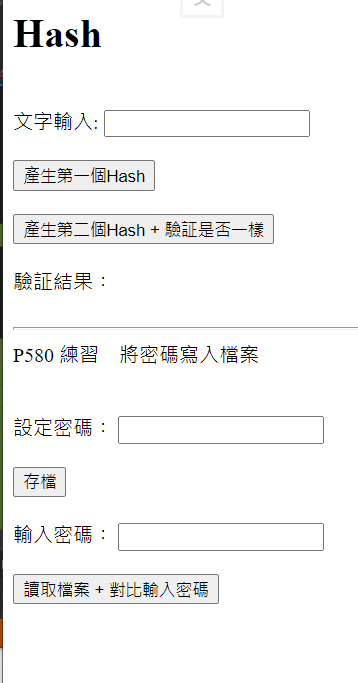








實作︰ P 579 note3\_ex3.Hash\_ex.aspx



P 584 完結