

Wired Equivalent Privacy

2006-08-18 By 小宗宗 <http://www.soujirou.idv.st/>

簡介:

隨著無線網路的發達,帶著自己的筆記電腦在咖啡廳裡上網已經是稀鬆平常的事了,不過大部分的人似乎都沒有發現使用無線網路上網,資料的隱密性將大大降低.資料竊聽者可以在隔壁或是對面的馬路使用適當的設備,就能輕易的竊聽到你筆記電腦傳送到網路的所有資料.這使得每個人傳送與接收的資料完全暴露在空氣中任人竊聽.試想如果有位公司的研發部員工使用無線網路收到了從公司寄來即將上市的產品設計圖,剛好這封 E-mail 被其他有心人士竊聽到了,該公司的損失將是無法估計的.

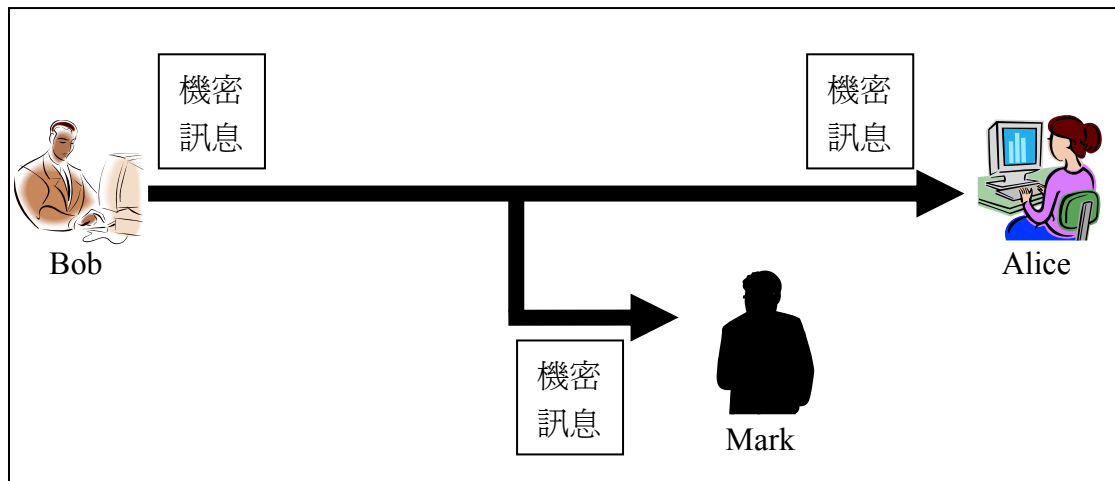
00a0	64	2c	20	31	36	20	41	75	67	20	32	30	30	36	20	31	d, 16 Au g 2006 1
00b0	38	3a	33	31	3a	30	34	20	2b	30	38	30	30	0d	0a	4d	8:31:04 +0800..M
00c0	49	4d	45	2d	56	65	72	73	69	6f	6e	3a	20	31	2e	30	IME-vers ion: 1.0
00d0	0d	0a	43	6f	6e	74	65	6e	74	2d	54	79	70	65	3a	20	..Conten t-Type:
00e0	74	65	78	74	2f	70	6c	61	69	6e	3b	0d	0a	09	63	68	text/pla in;...ch
00f0	61	72	73	65	74	3d	22	62	69	67	35	22	0d	0a	43	6f	arset="b ig5"..Co
0100	6e	74	65	6e	74	2d	54	72	61	6e	73	66	65	72	2d	45	ntent-Tr ansfer-E
0110	6e	63	6f	64	69	6e	67	3a	20	37	62	69	74	0d	0a	58	ncoding: 7bit..X
0120	2d	4d	61	69	6c	65	72	3a	20	4d	69	63	72	6f	73	6f	-Mailer: Microso
0130	66	74	20	4f	66	66	69	63	65	20	4f	75	74	6c	6f	6f	ft offic e Outloo
0140	6b	2c	20	42	75	69	6c	64	20	31	31	2e	30	2e	36	33	k, Build 11.0.63
0150	35	33	0d	0a	54	68	72	65	61	64	2d	49	6e	64	65	78	53..Thre ad-Index
0160	3a	20	41	63	62	42	48	78	42	31	37	53	63	52	4f	77	: AcbBHx B17ScRow
0170	4d	58	52	54	2b	36	43	32	6f	73	31	4b	4d	31	41	77	MXRT+6C2 os1KM1Aw
0180	3d	3d	0d	0a	58	2d	4d	69	6d	65	4f	4c	45	3a	20	50	==..X-Mi meOLE: P
0190	72	6f	64	75	63	65	64	20	42	79	20	4d	69	63	72	6f	roduced By Micro
01a0	73	6f	66	74	20	4d	69	6d	65	4f	4c	45	20	56	36	2e	soft Mim eOLE V6.
01b0	30	30	2e	32	39	30	30	2e	32	31	38	30	0d	0a	0d	0a	00.2900. 2180....
01c0	54	45	53	54	20	66	6f	72	20	74	68	65	20	43	61	70	TEST for the Cap
01d0	74	75	72	65	2e	0d	0a										ture...

使用簡單的程式就可輕易的擷取到他人的 E-mail 內容

WEP 就是防止遭人竊聽時資料外洩的一種加密協定,本篇將會介紹基礎密碼學,WEP 的加密/解密流程以及 WEP 的已知弱點.

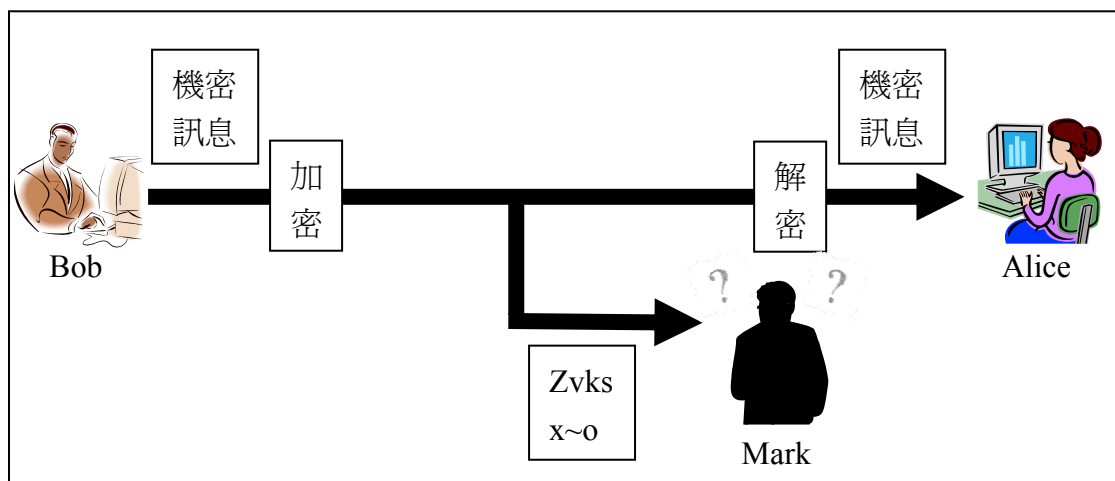
爲什麼要加密?

Bob 和 Alice 是一家公司的研發小組成員,他們正在秘密研發一項新的產品,今天 Bob 把改好的產品設計圖透過 E-mail 傳送給 Alice.在寄送過程中被商業間諜 Mark 從中竊聽到了這份 E-mail.由於 E-mail 並未進行加密處理,因此 Mark 可以不費任何力氣就可以直接取得這個產品設計圖並轉售給其他競爭公司獲得利益.



Bob 傳送給 Alice 的 E-mail 未經加密處理,Mark 輕易的就能取得 E-mail 內容

如果這封 E-mail 有經過加密後再傳送,即使是寄送的过程中遇到有心人士從中竊聽 E-mail 時,由於 E-mail 的內容經過了加密處理,沒有解密鑰匙的竊聽者將會竊聽到一篇加密過的 E-mail.因此就無法得知傳送的資料為何.這樣就可以不用擔心 E-mail 在傳送過程遭到他人竊聽內容.



Bob 傳送給 Alice 的 E-mail 經過加密處理,Mark 無法得知 E-mail 內容

加密方式

對稱式加密法

對稱式加密碼法由於加密與解密都是使用同一個金鑰來做加密與解密的運算,因此又稱為單鑰加密法.

對稱式的加密系統有五大元素:

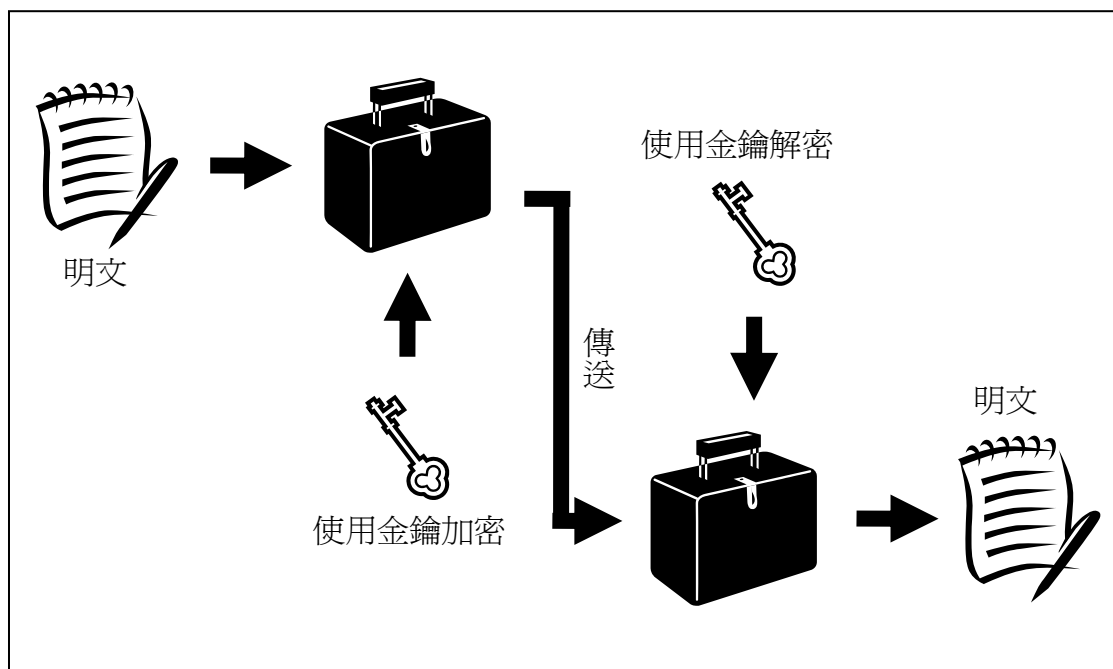
1. 明文(Plaintext): 尚未加密的訊息或資料.

2. 加密演算法(Encryption algorithm): 將明文進行加密的動作.通常使用替換(substitution)或是排列(transformation)的方式來對明文進行加密.
3. 金鑰(Secret Key): 金鑰是演算法中的重要參數之一,演算法通常依據不同的金鑰對加解密的資料進行不同的替換或是排列的運算.通常金鑰的長度越長表示安全性越高,不過也可能會降低加解密的速度.現今的演算法中最常見的金鑰長度是 128 位元.

金鑰長度 (Bits)	可能的金鑰數目	在 1 Encryption/ μ s 系統上所需的時間	在 10^6 Encryption/ μ s 系統上所需的時間
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8$ 分	2.15ms
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142$ 年	10.01小時
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24}$ 年	5.4×10^{18} 年
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36}$ 年	5.9×10^{30} 年

表 1 使用暴力破解法(窮舉法)找出金鑰的平均時間

4. 密文(Ciphertext): 明文經過加密演算法後所輸出的不規則訊息就是密文.通常不同的金鑰和明文會產生不同的密文.
5. 解密演算法(Decryption algorithm): 加密演算法的逆向程序就是解密演算法.將加密的密文和加密時所使用的金鑰輸入給解密演算法後就可以獲得明文的輸出完成解密的動作.



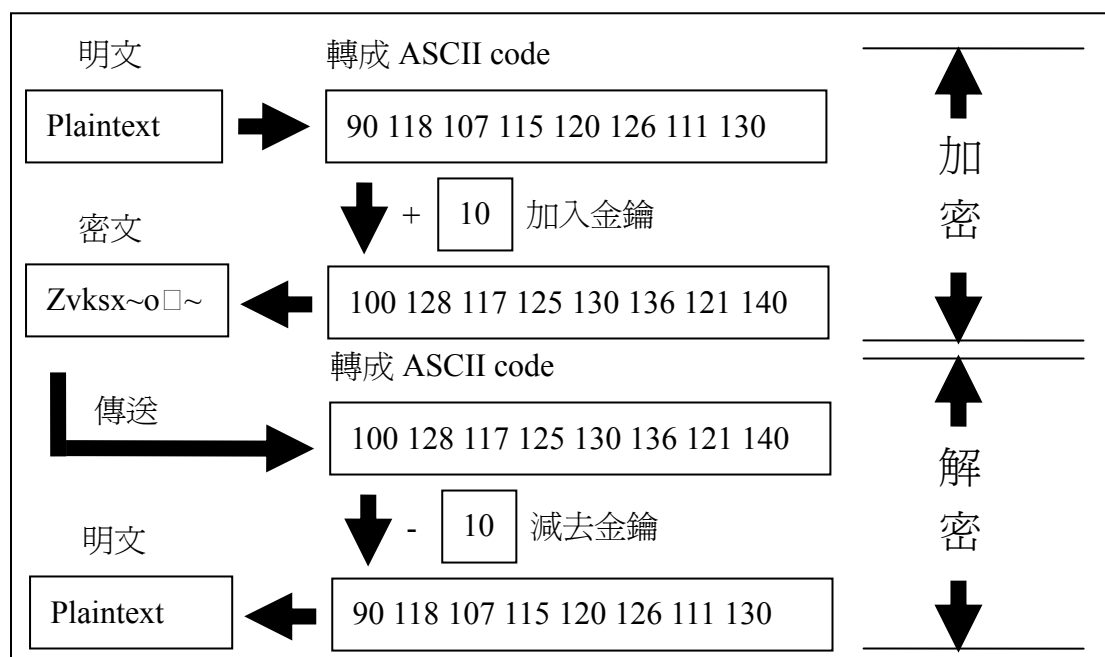
對稱式加密法

加解密的流程如下:

1. 產生一個金鑰.
2. 如果 Mark 要傳送加密的訊息給 Alan,則 Mark 需要透過安全的管道先將金鑰傳送給 Alan.
3. Mark 利用金鑰進行加密演算把明文轉換成密文後傳送給 Alan.
4. Alan 收到 Mark 傳送的密文後,利用 Mark 之前傳送的金鑰進行解密演算把密文轉換成明文.

要能使透過對稱式加密法的資料達到保密,必須要滿足下面兩項要求:

1. 加密演算法必須非常堅固,就算讓攻擊者知道加解密的演算法,而且取得許多密文樣本也無法解譯出原來的明文或是猜測出使用者所使用的加密金鑰.
2. 傳送及接收訊息的雙方都必須以極度安全的方式來傳遞並保存他們所共同使用的金鑰.如果金鑰遭到第三者竊取,則所有使用該金鑰加密的密文都可以輕易的被解密還原成明文.



簡單的對稱式加密法範例

上圖範例的程式碼: (PHP 4 by 小宗宗)

```
$plaintext = "Plaintext";  
$plaintext_2 = "";  
$ciphertext = "";  
$ascii = array();  
$text_length = strlen($plaintext);  
$secret_key = 10;
```

```

//Encryption
//convert to ASCII code
echo "Plaintext after Encryption: ".$plaintext."<br>";
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i] = ord(substr($plaintext, $i, 1));
}
//ASCII code + secret_key
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i]=$ascii[$i] + $secret_key;
    if ($ascii[$i] > 255) {$ascii[$i] = $ascii[$i] - 256;}
}
//convert to ciphertext
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i] = chr($ascii[$i]);
}
for ($i=0;$i<=($text_length - 1);$i++){
    $ciphertext = $ciphertext.$ascii[$i];
}
echo "Ciphertext: ".$ciphertext."<br>";

//decode
//convert to ASCII code
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i] = ord(substr($ciphertext, $i, 1));
}
//ASCII code - secret_key
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i]=$ascii[$i] - $secret_key;
    if ($ascii[$i] < 0) {$ascii[$i] = $ascii[$i] + 256;}
}
//convert to plaintext
for ($i=0;$i<=($text_length - 1);$i++){
    $ascii[$i] = chr($ascii[$i]);
}
for ($i=0;$i<=($text_length - 1);$i++){
    $plaintext_2 = $plaintext_2.$ascii[$i];
}
echo "Ciphertext after Decryption: ".$plaintext_2."<br>";

```

執行結果:

Plaintext after Encryption: Plaintext

Ciphertext: Zvksx~o□~

Ciphertext after Decryption: Plaintext

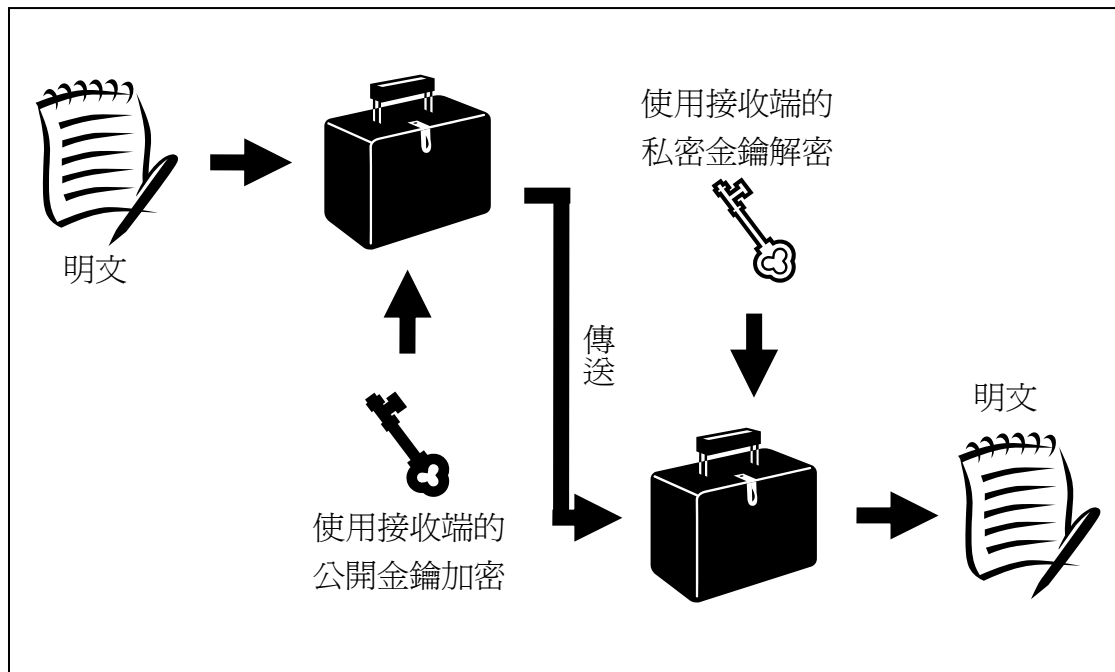
由於對稱式加密法加密與解密都是使用同一把金鑰,因此加密與解密的演算法比非對稱式簡單,常見的對稱式加密演算法: Data Encryption Standard (DES), Triple DES (3DES), Advanced Encryption Standard (AES)

非對稱式加密法

非對稱式加密法在加密與解密運算時使用兩把不同的金鑰運算.因此不需要在傳送加密訊息前先把金鑰傳送給接收端.這樣可以降低金鑰在傳送途中遭竊取的機會,因此安全性相對的比對稱式加密法更佳,但加密與解密運算較對稱式加密法複雜.

非對稱式的加密系統有五大元素:

1. 明文(Plaintext): 尚未加密的訊息或資料.
2. 加密演算法(Encryption algorithm): 將明文進行加密的轉換動作.明文和對應的金鑰輸入給加密演算法後就可以獲得密文的輸出完成加密的動作.
3. 公開金鑰(Public Key)與私密金鑰(Private Key): 這是一對金鑰,其中一把金鑰用來加密,而另一把金鑰用來解密.顧名思義私密金鑰必須由個人保存在安全的地方,而公開金鑰則是公開出來給大家自行索取的金鑰.
4. 密文(Ciphertext): 明文經過加密演算法後所輸出的不規則訊息就是密文.通常不同的金鑰和明文會產生不同的密文.
5. 解密演算法(Decryption algorithm): 將密文解譯成明文的轉換動作.加密的密文和對應的金鑰輸入給解密演算法後就可以獲得明文的輸出完成解密的動作.



非對稱式加密法

加解密的流程如下:

1. 每個使用者會產生一對加密和解密用的金鑰,分別是公開金鑰以及私密金鑰.
2. 每個使用者將其中一把金鑰當作是公開金鑰,放在大家都可以取得的地方.而另一把私密金鑰則存放在安全不公開的地方.
3. 如果 Mark 要傳送加密的訊息給 Alan,則 Mark 需要使用 Alan 的公開金鑰來進行加密演算的動作,然後再將密文傳送給 Alan.
4. Alan 收到了 Mark 傳來的訊息後,使用自己的私密金鑰進行解密演算的動作後取得原始未加密的明文.由於只有 Alan 擁有私密金鑰,因此可以確保這則訊息只有 Alan 可以順利解密轉換成未加密的明文.

非對稱式加密法由於加密與解密的金鑰不同,所以理論上只要能確保私密金鑰的安全就可以保證往來的訊息可以受到保護,不用像對稱式加密法需要擔心金鑰在傳送過程中被竊取的問題.變更金鑰相對的也比較容易.常見的非對稱式加密演算法: RSA, Elliptic Curve.

什麼是 WEP

WEP, Wired Equivalent Privacy.是一種被納入 IEEE 802.11b 的加密方式.採用的是 RC4 對稱式加密法將傳輸的資料封包進行加密與解密,是一種效率高且可靠的加密演算法.

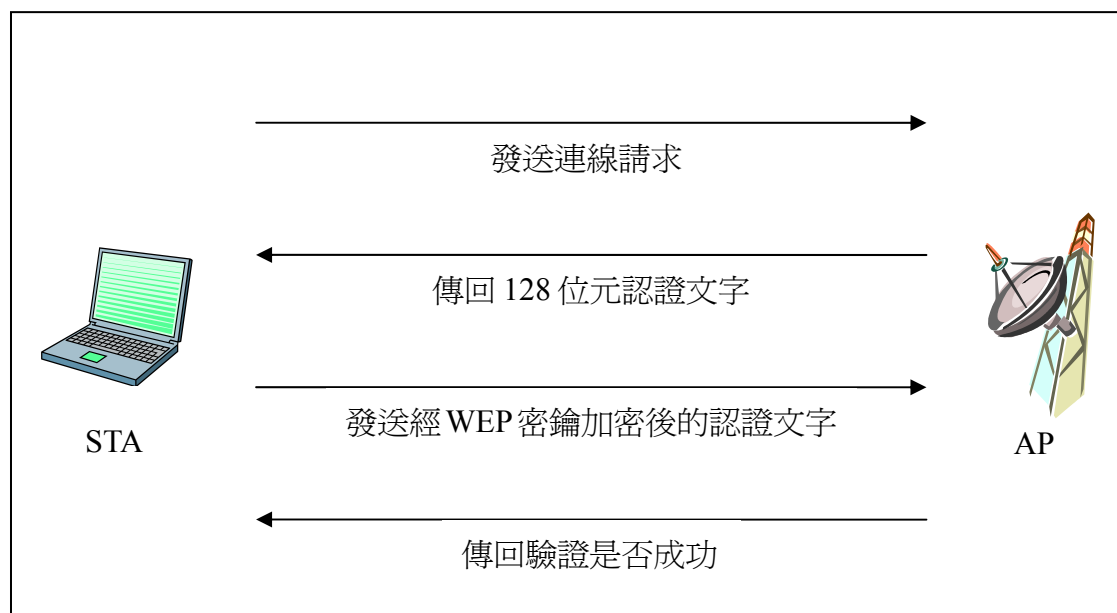
WEP 具有下列特色:

- 有 64 位元以及 128 位元兩種版本可供選擇,可視需要選擇使用的版本.
- WEP 演算法計算效率高,對於攜帶式系統的電力需求不大.
- 當通訊不慎中斷時,可自動重新進行同步連結.
- 可以根據使用者的需求開啓或關閉 WEP 功能.

由於無線網路的電磁波會散佈於開放的空間中,竊聽者只需要在無線網路基地台涵蓋的範圍內都可以對傳送的封包進行監聽.使用 WEP 加密可以讓使用者在使用無線網路時不需要擔心傳送的資料封包會遭到竊聽,因為即使資料遭到竊聽也無法輕易的解密成原始的明文資料.

WEP 的連線基本流程:

1. 無線網路使用者 (Wireless Station, 或稱為 STA) 對存取點 (Access Point, AP) 提出認證的請求.
2. AP 收到請求後產生一個 128 位元長度的認證文字 (Challenge Text)並傳送給 STA.
3. STA 收到認證文字後,使用 WEP 密鑰進行 RC4 加密演算並傳回給 AP.
4. AP 收到加密後的密文,使用 WEP 密鑰及 RC4 演算法進行解密,將其與原本傳送的明文進行比對.如果解密後的結果與認證文字相符則表示通過驗證,並發送接受連線請求給 STA.如不符合則表示對方的 WEP 密鑰不正確,無法通過驗證,將會拒絕 STA 的連線.



RC4 演算法

RC4 是一種同步的串流加密,使用 XOR 來結合金鑰串流產生器的輸出和訊息的明文.這種演算法是由 RSA Security 的 Ron Rivest 於 1987 年設計的.RC4 原本是 RSA 的商業機密,並未公佈其演算法詳細內容.在 1994 年九月時有匿名網友將 RC4 演

算法公佈於新聞群組上,但 RSA 並不承認新聞群組上發佈的演算法就是真正的 RC4.由於 RC4 是 RSA 註冊的名字,爲了防止侵權的問題,網路上發佈出來的 RC4 演算法通常稱爲 ARC4(Alleged-RC4).而 WEP 主要就是採用 RC4 加密解密演算法.

RC4 加密流程

RC4 會產生一個用來加密的虛擬隨機串流(pseudorandom stream, 簡稱 keystream)然後使用 XOR 與明文做運算.而解密部分則是使用同樣的方式逆向操作完成解密的動作.

要產生這個 keystream 需要有一個初始狀態值,這狀態值包含兩個部份:

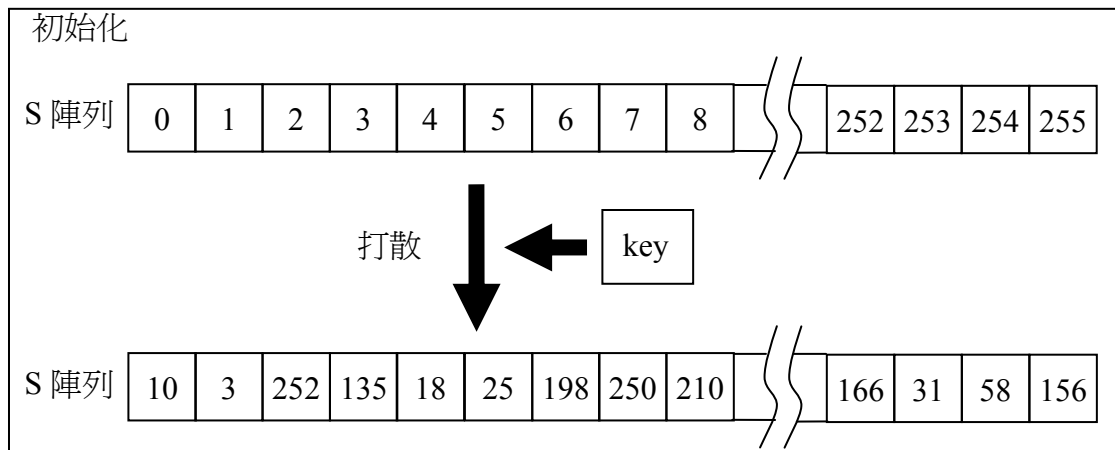
1. 一個含有 256 個索引值的陣列.(以 S 表示)
2. 兩個 8 位元指標器(以 i 和 j 表示)

金鑰排程演算法 The key-scheduling algorithm (KSA)

在產生虛擬隨機串流(pseudorandom stream)前,必須先建立演算法要使用的索引值陣列 S.下面程式中"keylength"指的是使用多少位元組的金鑰.範圍則是介於 1~256.一般來說是使用 5~16 個位元組.執行 KSA 時,首先 S 陣列會被依序填入 0~255 的數值,接著加入金鑰變數把這整個陣列給打散.

KSA 程式碼:

```
//初始化
for i from 0 to 255 {
    S[i] = i
}
j = 0
//打散 S 陣列
for i from 0 to 255 {
    j = (j + S[i] + K[i mod keylength]) mod 256
    swap(S[i],S[j])
}
```



虛擬隨機產生演算法 The pseudo-random generation algorithm (PRGA)

當完成 KSA 演算後,接著我們利用 PRGA 演算來產生加密用的 keystream. 透過迴圈會為每個封包的位元組建立串流.最後的結果就是要傳送給接收端的密文.

PRGA 程式碼:

```

初始化
i = 0
j = 0
產生迴圈
while GeneratingOutput {
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap(S[i],S[j])
output = S[(S[i] + S[j]) mod 256]
}

```

RC4 加密範例

由於真正的 RC4 加密使用的索引值是 256,但因為 256 個索引值計算非常繁瑣,因此我們這邊只使用索引值為 4 做為 RC4 加密範例.

使用的初始變數如下:

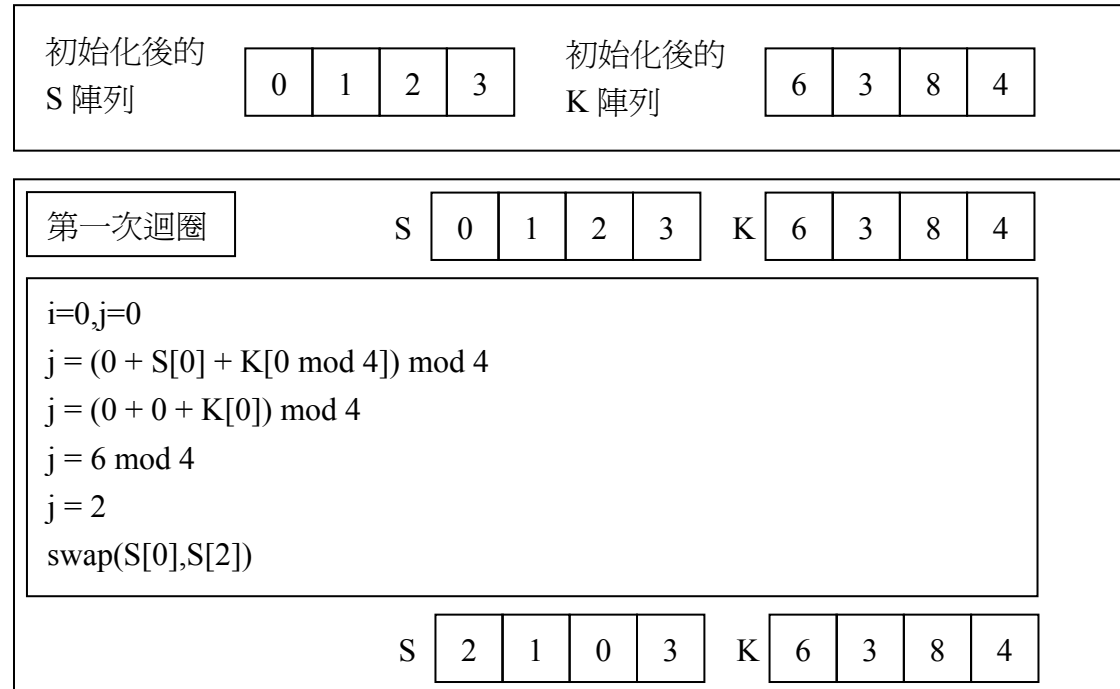
金鑰: 6384, keylength: 4

KSA

KSA 程式碼

```
初始化
for i from 0 to 3 {
    S[i] = i
}
j = 0
打散 S 陣列
for i from 0 to 3 {
    j = (j + S[i] + K[i mod 4]) mod 4
    swap(S[i],S[j])
}
```

KSA 程式執行過程



第二次迴圈	S	2	1	0	3	K	6	3	8	4										
$i=1, j=2$ $j = (2 + S[1] + K[1 \bmod 4]) \bmod 4$ $j = (2 + 1 + K[1]) \bmod 4$ $j = 6 \bmod 4$ $j = 2$ swap(S[1],S[2])																				
<table style="width: 100%; border: none;"> <tr> <td style="width: 25%; padding: 5px;">S</td> <td style="width: 20%; padding: 5px;">2</td> <td style="width: 20%; padding: 5px;">0</td> <td style="width: 20%; padding: 5px;">1</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 25%; padding: 5px;">K</td> <td style="width: 20%; padding: 5px;">6</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 20%; padding: 5px;">8</td> <td style="width: 20%; padding: 5px;">4</td> </tr> </table>											S	2	0	1	3	K	6	3	8	4
S	2	0	1	3	K	6	3	8	4											

第三次迴圈	S	2	0	1	3	K	6	3	8	4										
$i=2, j=2$ $j = (2 + S[2] + K[2 \bmod 4]) \bmod 4$ $j = (2 + 1 + K[2]) \bmod 4$ $j = 11 \bmod 4$ $j = 3$ swap(S[2],S[3])																				
<table style="width: 100%; border: none;"> <tr> <td style="width: 25%; padding: 5px;">S</td> <td style="width: 20%; padding: 5px;">2</td> <td style="width: 20%; padding: 5px;">0</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 20%; padding: 5px;">1</td> <td style="width: 25%; padding: 5px;">K</td> <td style="width: 20%; padding: 5px;">6</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 20%; padding: 5px;">8</td> <td style="width: 20%; padding: 5px;">4</td> </tr> </table>											S	2	0	3	1	K	6	3	8	4
S	2	0	3	1	K	6	3	8	4											

第四次迴圈	S	2	0	3	1	K	6	3	8	4										
$i=3, j=3$ $j = (3 + S[3] + K[3 \bmod 4]) \bmod 4$ $j = (3 + 1 + K[3]) \bmod 4$ $j = 8 \bmod 4$ $j = 0$ swap(S[3],S[0])																				
<table style="width: 100%; border: none;"> <tr> <td style="width: 25%; padding: 5px;">S</td> <td style="width: 20%; padding: 5px;">1</td> <td style="width: 20%; padding: 5px;">0</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 20%; padding: 5px;">2</td> <td style="width: 25%; padding: 5px;">K</td> <td style="width: 20%; padding: 5px;">6</td> <td style="width: 20%; padding: 5px;">3</td> <td style="width: 20%; padding: 5px;">8</td> <td style="width: 20%; padding: 5px;">4</td> </tr> </table>											S	1	0	3	2	K	6	3	8	4
S	1	0	3	2	K	6	3	8	4											

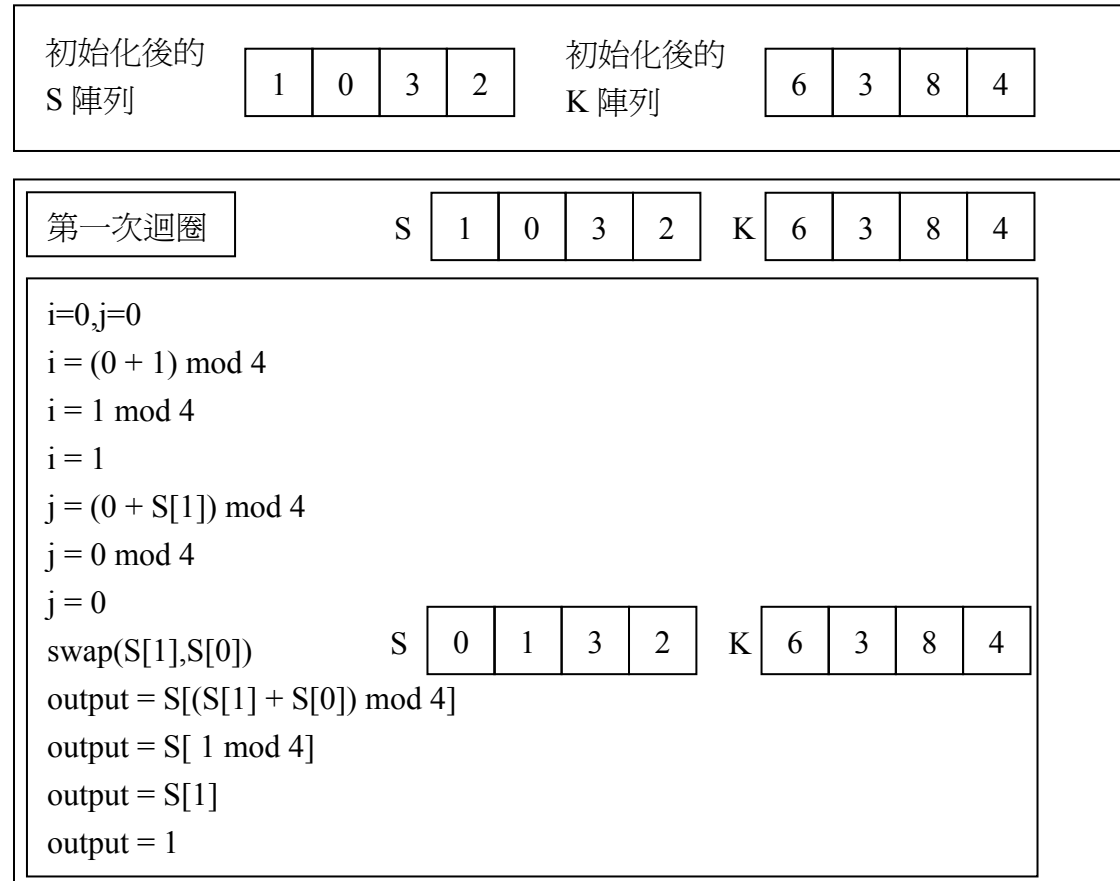
PRGA

假設我們要加密的明文為 pas 共三個字.因此我們要產生三組 keystream 也就是說要執行三次產生迴圈

PRGA 程式碼

```
初始化
i = 0
j = 0
產生迴圈
while GeneratingOutput {
i = (i + 1) mod 4
j = (j + S[i]) mod 4
swap(S[i],S[j])
output = S[(S[i] + S[j]) mod 4]
}
```

程式執行過程



第二次迴圈	S	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">0</td><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">2</td></tr> </table>	0	1	3	2	K	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">6</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">8</td><td style="padding: 2px 10px;">4</td></tr> </table>	6	3	8	4
0	1	3	2									
6	3	8	4									
$i=1, j=0$ $i = (1 + 1) \bmod 4$ $i = 2 \bmod 4$ $i = 2$ $j = (0 + S[2]) \bmod 4$ $j = 3 \bmod 4$ $j = 3$ <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;">swap(S[2],S[3])</div> <div style="margin-right: 10px;">S</div> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">0</td><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">2</td><td style="padding: 2px 10px;">3</td></tr> </table> <div style="margin-right: 10px;">K</div> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">6</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">8</td><td style="padding: 2px 10px;">4</td></tr> </table> </div> $output = S[(S[2] + S[3]) \bmod 4]$ $output = S[5 \bmod 4]$ $output = S[1]$ $output = 1$					0	1	2	3	6	3	8	4
0	1	2	3									
6	3	8	4									

第三次迴圈	S	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">0</td><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">2</td><td style="padding: 2px 10px;">3</td></tr> </table>	0	1	2	3	K	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">6</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">8</td><td style="padding: 2px 10px;">4</td></tr> </table>	6	3	8	4
0	1	2	3									
6	3	8	4									
$i=2, j=3$ $i = (2 + 1) \bmod 4$ $i = 3 \bmod 4$ $i = 3$ $j = (3 + S[3]) \bmod 4$ $j = 6 \bmod 4$ $j = 2$ <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;">swap(S[3],S[2])</div> <div style="margin-right: 10px;">S</div> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">0</td><td style="padding: 2px 10px;">1</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">2</td></tr> </table> <div style="margin-right: 10px;">K</div> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="padding: 2px 10px;">6</td><td style="padding: 2px 10px;">3</td><td style="padding: 2px 10px;">8</td><td style="padding: 2px 10px;">4</td></tr> </table> </div> $output = S[(S[3] + S[2]) \bmod 4]$ $output = S[5 \bmod 4]$ $output = S[1]$ $output = 1$					0	1	3	2	6	3	8	4
0	1	3	2									
6	3	8	4									

因此我們獲得了 3 個 PRGA 產生的 keystream 轉成二進位後分別是 00000001, 00000001, 00000001.

Keystream 與明文進行 XOR 運算

接著我們要把 keystream 和明文做 XOR 運算取得最終的密文。

明文	明文 ASCII 十進位碼	明文 ASCII 二進位碼
p	112	01110000
a	97	01100001
s	115	01110011

XOR 的邏輯運算

	0	1
0	0	1
1	1	0

明文 ASCII 二進位碼	keystream	XOR 運算	密文
01110000	00000001	01110001	q
01100001	00000001	01100000	`
01110011	00000001	01110010	r

最後 pas 轉換成密文是 q`r

WEP 的加解密流程

之前介紹了 RC4 的加密流程,現在我們來談本篇文章的主題-WEP 加密.WEP 加密包含了 RC4,IV(Initiation Vector 初始向量),金鑰和 CRC 四部份。

IV (Initiation Vector 初始向量)

在加密過程中如果只使用固定金鑰資料將容易被破解,因此 WEP 加密過程加入了一個 24 bits 長度的初始向量,藉此打亂加密金鑰的組合。

金鑰

金鑰是加解密時重要的參數之一,WEP 可以使用 5 或 13 個字元的密碼,也就是 40 bits 或 104 bits 長度的金鑰.金鑰將與 IV 結合成 64bits 或 128 bits 作為 RC4 的 PRGA 種子以產生 keystream.

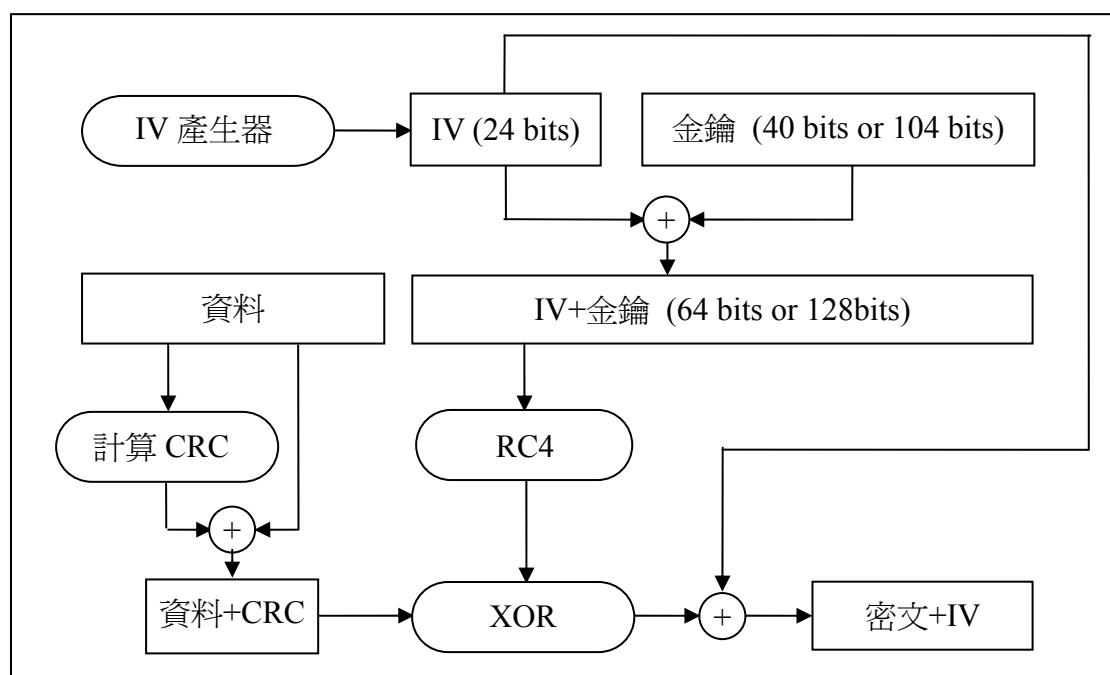
CRC(Cyclic Redundancy Checks 循環冗餘檢查)

無線資料傳輸時需要將傳送的資料切割成許多小的封包傳送.而在傳送的過程中由於受到大氣,環境等影響會產生干擾而使得資料封包內容損壞.試想如果傳送的是一份有化學反應方程式的 E-mail,在傳送中遭到干擾而使得方程式產生錯誤而

接收端的人並未發現就進行實驗.那很可能會發生爆炸也說不定.

爲了要能確保收到的封包資料正確性,我們使用 CRC 來驗證封包資料是否正確.CRC 是種簡易的演算法,依據資料封包的內容會算出一個唯一值.在 802.11 中,這個值是一個長度 32 bits 的數值.

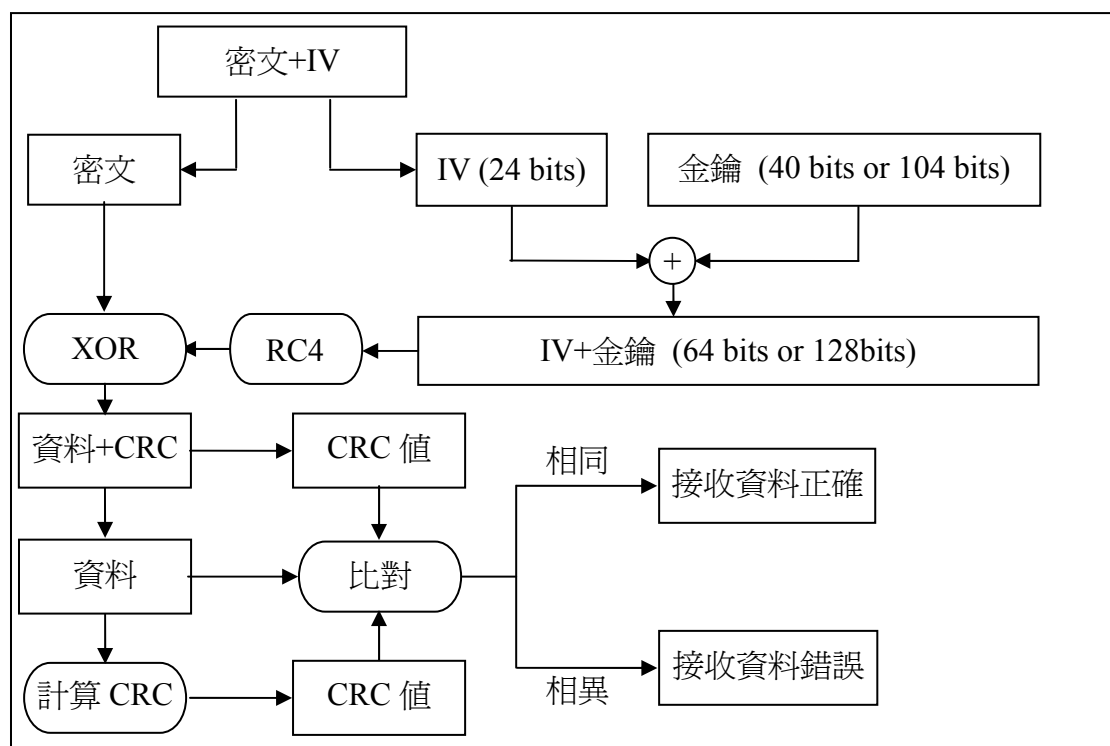
WEP 加密流程



WEP 加密流程圖

1. 將要傳送的資料進行 CRC 運算產生 CRC 檢查碼,並與傳送資料結合.
2. 使用 IV 產生器產生一個 IV.
3. 將 IV 以及金鑰進行 RC4 運算取得一個加密用 keystream.
4. 利用此 keystream 將步驟 1 所得到的封包進行 XOR 運算進行加密,獲得一密文資料,並將 24 bits 長的 IV 與密文資料結合.
5. 將含有 IV 的密文資料透過無線網路傳送出去.

WEP 解密流程



WEP 解密流程圖

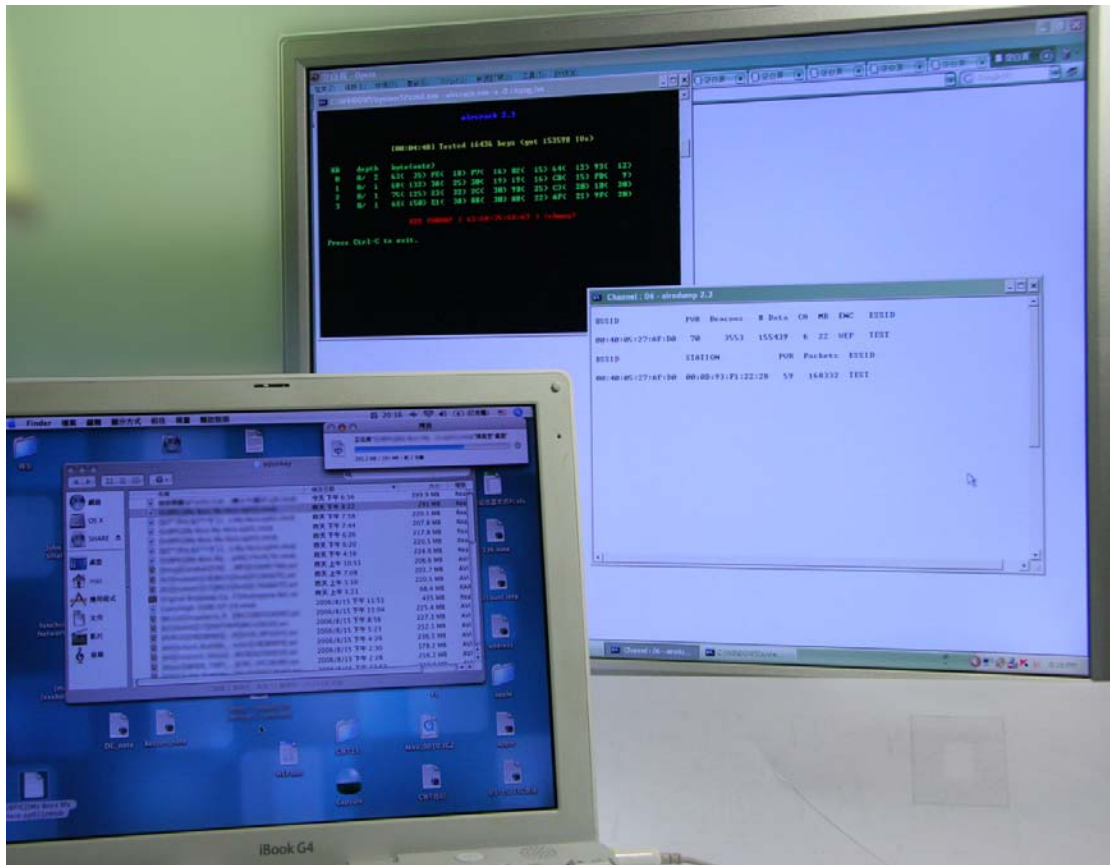
1. 將收到的封包把密文與 IV 分解.
2. 將金鑰與接收到的 IV 進行 RC4 運算取得一個解密用 keystream.
3. 利用此 keystream 將步驟 1 所分離的密文進行 XOR 運算進行解密.
4. 分離資料與 CRC 值.
5. 將資料進行 CRC 運算產生 CRC 檢查碼,並與收到的 CRC 檢查碼比較.
6. 如果兩個 CRC 值相同表示資料正確.不相同表示資料發生錯誤.

WEP 的弱點

大家看到這邊一定認為只要使用了 WEP 加密後就可以高枕無憂,不用再擔心傳送的資料會被竊聽了.事實上則不然.接下來我們來做一個小實驗來證明 WEP 加密仍不夠安全.

使用設備:

- P4 2.6c 桌上電腦搭配 Cisco Aironet 350 Series 無線網路卡當作監聽電腦.
- iBook G4 筆記電腦搭配 Apple AirPort Extreme Card 無線網卡當做 STA.
- AP 使用的是 D-Link DWL-900AP+



把 AP 的 WEP 密碼設定成 40 bits 後,打開 WEP 金鑰破解程式,接著用 iBook 透過無線網路開始抓取 NAS(Network-attached storage)的資料(不抓取 P4 電腦的資料是因為會佔用 P4 的 CPU 使用率,會影響破解效率),這樣可以讓網路滿載,破解程式可以在最短時間內竊聽到最多的資料封包。

只見竊聽的程式飛快的跑動著,這時小宗宗去樓下沖了一壺茶後回來看了一下螢幕.什麼!只見程式已經停止跑動,而加密金鑰已經顯示在螢幕的下方.看了看時間前後不到 5 分鐘.這實在是出乎小宗宗意料之外。



只花了 04:40 就找到了 40 bits 金鑰

接著小宗宗把 WEP 金鑰設定成較長的 104 bits 後重新再做一次實驗.翻了幾頁書

3. Charles R. Elden, Tara M. Swaminatha , Wireless Security and Privacy: Best Practices and Design Techniques, 2002
4. Cyrus Peikari, Seth Fogie , Maximum Wireless Security, 2002
5. William Stallings, Network Security Essentials (2nd Edition), 2002
6. Russell Dean Vines , Wireless Security Essentials: Defending Mobile Systems from Data Piracy, 2002