GOTOP E&T 碁峰學苑

資訊安全 原理與實驗

Information Security : Principles and Experiments

人們習於透過網路的傳遞分享數位內容，例如：建立部落格、個人相簿等。數位內容之產生與傳遞是非常便利的，也相對使其遭到濫用或篡改的情況愈加嚴重。本書第一章至第六章，著眼於如何採用密碼學方法與浮水印方法進行資訊的保護，而第七章至第十四章，則側重於網路部份的防範。

碁峰 www.gotop.com.tw

Ubi qui to us 台灣優質士

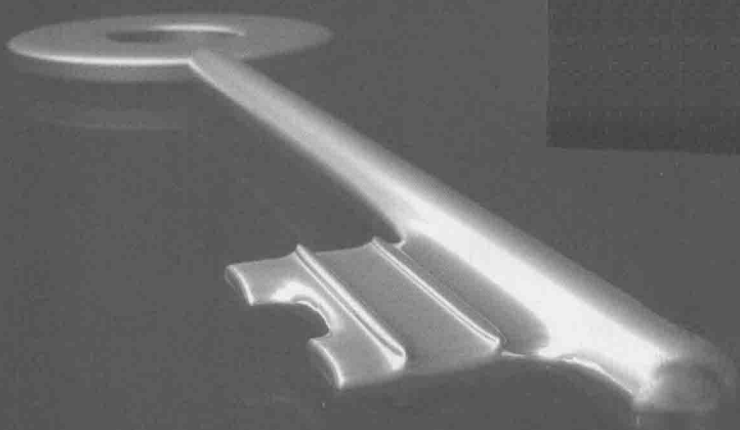
黃祥哲、吳惠麟、盧長青、張峯誠、羅浩、潘正祥

資訊安全
原理與實驗

資訊安全
原理與實驗

Information Security : Principles and Experiments

人們習於透過網路的傳遞分享數位內容，例如：建立部落格、個人相簿等。數位內容之產生與傳遞是非常便利的，也相對使其遭到濫用或篡改的情況愈加嚴重。本書第一章至第六章，著眼於如何採用密碼學方法與浮水印方法進行資訊的保護，而第七章至第十四章，則側重於網路部份的防範。



黃祥哲、吳惠麟、盧長青、張峯誠、羅浩、潘正祥

碁峰資訊

國家圖書館出版品預行編目資料

資訊安全原理與實驗 / 黃祥哲等著. -- 初版. -- 臺北市：碁峰資訊,2008.11

面； 公分

ISBN 978-986-181-553-4(平裝)

1. 資訊安全 2.電腦密碼 3.電腦網路

312.76 97018635

書　　名	資訊安全原理與實驗
書　　號	AEE031200
作　　者	黃祥哲 / 吳惠麟 / 盧長青 / 張峰誠 / 羅浩/ 潘正祥
建議售價	NT$ 400
發 行 人	廖文良
發 行 所	碁峰資訊股份有限公司
地　　址	台北市南港區三重路 66 號 7 樓之 6
電　　話	(02)2788-2408
傳　　真	(02)2788-1031
法律顧問	明貞法律事務所　胡坤佑律師
版　　次	2008 年 11 月初版

序

隨著電腦網路與消費電子產品的普及，資訊化與數位化已是當前人類最常接觸到的媒介之一。不僅新聞節目、個人檔等需要以數位方式進行存取，人們也習於在網路上設立部落格、個人相簿等，透過網路的傳遞，來分享數位內容。由於數位內容的的產生與傳遞變得非常便利，也使得數位內容遭到濫用或篡改的情況愈加嚴重了。由此可知，數位內容的智慧財產權管理與保護，為當今的一個重要課題。我們必須採用資訊安全的方法，從多媒體內容與網路兩方面著手，來防範上述的問題。

有鑒於上述問題的重要性，作者群精心整理多媒體與網路資訊安全的基本原理與技術，設計相關實驗，並補充心得以編撰此書。全書共分為十四章，略述如下。

- 第一章至第六章，著眼於採用密碼學方法與浮水印方法進行資訊保護，包含金鑰交換、對稱式金鑰系統、非對稱式金鑰系統、影像認證、可見型浮水印方法、以及半色調浮水印技術等章節。
- 第七章至第十四章，側重於網路部份，包含 TCP/IP 通訊協定、入侵偵測系統、垃圾郵件防止、阻絕服務攻擊、防火牆、網頁過濾、電腦病毒、弱點偵測等章節。

本書適合大學院校相關科系作為一學期的教材使用。撰寫本書時，我們儘量採用淺顯的文字與原理說明，並引用大量的範例做介紹。藉此，希望能引發讀者興趣並進一步自行驗證與深入學習。

本書的編撰雖力求完善，但疏漏之處難免，至望海內外專家、學者不吝予以指教。

黃祥哲、吳惠麟、盧長青、張肇誠、羅浩、潘正祥

目錄

第 1 章 金鑰交換

第 2 章 對稱式金鑰系統

第 3 章 非對稱式金鑰系統

第 4 章 植基於數位易碎浮水印技術的影像認證

第 5 章 適應於原始影像之可見型浮水印方法

第 6 章 適用於半色調影像之可見浮水印技術

第 7 章 IP 偵測與通訊埠掃描

第 8 章 入侵偵測系統

第 9 章 垃圾郵件過濾

第 10 章 阻斷服務攻擊

第 11 章 防火牆

第 12 章 網頁過濾

第 13 章 電腦病毒種類及原理

第 14 章 系統弱點偵測

1 金鑰交換

1.1 實驗目的

通訊或任何類型的資訊交換，必須有傳遞的介質。就物理意義而言，這些媒介把代表資訊的能量，以某種形式從某一端傳達至另一端。而傳遞的過程中，只要能實際接觸介質，理論上是可以擷取傳送中的資訊。舉例來說，郵差遞送信件的過程，如果能讓郵差停下，就有機會取得信件並獲取內容。這類問題在以電磁訊號為主的現代通訊主流中更為嚴重，這是因為電磁訊號不只可以擷取，還可以近乎完美的複製，所以對於通訊兩端以及遞送機制而言，甚至不會察覺資訊已被竊取。

由前述可知，若對於傳遞媒介沒有特定的保護，任何有辦法接觸到媒介的人，皆有機會取得傳輸中的資訊。因為環境形同對所有人開放，所以有時我們稱其為開放式環境。就安全性而言，開放式環境的確不是理想的資訊交換環境，但基於建置與維護的成本考量，開放式架構仍為主流。既然無法避免開放環境中的實體資料竊取問題，保密的手段便改為即使遭竊取也無法解讀。因此，密碼方法應運而生，其主要目的就是在開放式環境中建立一虛擬的保密傳遞路徑，讓竊取資訊的人無法解讀獲得的資料。密碼方法多半仰賴金鑰作為加密或解密的參數，因此，通訊雙方如何安全地建立共通的金鑰便是首要工作。

本實驗的目的，在於了解如何於開放式環境中，互相通訊的兩端動態建立共通的金鑰，作為後續保密通訊之用。

1.2 原理說明

本實驗將操作 Diffie-Hellman 金鑰交換(Diffie-Hellman key exchange)演算法。該方法為 Whitfield Diffie 與 Martin Hellman 於 1976 年發表，後來成為密碼系統中相當知名且典型的理論，亦有其他名稱如 Diffie-Hellman

key agreement、Diffie-Hellman key establishment、及 Diffie-Hellman key negotiation。這個方法雖然最初只是作為建立共同金鑰的通訊協定，且缺乏認證(authentication)的能力，但其理論基礎對後來的密碼方法造成深遠影響，例如知名的 RSA 公開金鑰方法。

Diffie-Hellman 金鑰交換的理論基礎，是建立在數論中指數運算取餘數後不易逆向推導，因此又有一個名稱為指數密鑰交換(exponential key exchange)。以下定義本方法中用到的參數與運算：

- p：用於定義數群的質數。
- g：用來產生數群的 primitive root。
- a：通訊 A 端(Alice 端)選定的秘密正整數值。
- b：通訊 B 端(Bob 端)選定的秘密正整數值。
- K：共同金鑰。
- mod：取餘數運算。

而理論的基礎簡述如下：

根據數論：

$$\left(g^{a} \mod p\right)^{b} \mod p = \left(g^{b} \mod p\right)^{a} \mod p$$

因此我們可以讓通訊 Alice、Bob 兩端選取各自的秘密正整數值 a 與 b，配合公開的資訊 p 與 g，則

通訊 Alice 端

$$A = g^{a} \mod p$$

$$K = B^{a} \mod p$$

通訊 Bob 端

$$B = g^b \mod p$$

$$K = A^b \mod p$$

而假設 Alice 端為金鑰交換的發起端，且網路傳送過程中只可能發生竊取資訊的情況，則整個金鑰建立過程則如下表所示：

時間順序	通訊 Alice 端	網路連結	通訊 Bob 端	兩端共同資訊
1	選定秘密正整數 a 計算 $A=g^a \mod p$			
2		傳送 A, g, p 至 B 端		
3			選定秘密正整數 b 計算 $B=g^b \mod p$ 計算 $K=A^b \mod p$	A, g, p
4		傳送 B, g, p 至 A 端		A, g, p
5	計算 $K=B^a \mod p$			A, B, g, p, K

到了第 5 個步驟，通訊兩端皆擁有資訊 K，即交換取得的共同金鑰，供後續加解密使用。

在開放式環境中，整個金鑰交換過程可能被竊取資訊的地方，只會發生在網路連結。因此，假設有個 E 端(Eve 端)可以竊聽 Alice、Bob 端之間的通訊，則 Eve 端能截獲的資訊只有 A、B、g、p。由於共同金鑰 K 的計算必須知道秘密值 a 或 b，對於 Eve 而言，只能設法由 A、g、p 逆推 a，

或是由 B、g、p 逆推 b。但只要 g 與 p 選擇適當(一般而言 p 的值相當大)，就理論上是無法在有限時間內解出 a 或 b。以大部分應用而言，傳送的資料多半有其時效性，只要讓 Eve 無法在有限時間內破解，就已經達到 Alice 與 Bob 間秘密通訊的目的了。

1.3 實驗設備材料及軟體

在實作上，金鑰交換的過程牽涉到許多高位數整數運算，其中高次方運算多半無法直接使用一般 CPU 的整數計算，因此實驗過程將以 C 語言搭配 GNU Multiple Precision arithmetic library (GMP)進行。為了方便檢視實驗結果，步驟中所使用的例子相當簡單，但我們仍使用 GMP 實作，以便將結果擴展至實務狀況。

以下列出本實驗所需的材料及設備：

- 個人電腦搭載 GNU 執行環境
  - 如搭載 Linux 或 FreeBSD 等作業系統，則已符合條件
  - 如搭載 Windows 系列作業系統，請安裝 Cygwin 環境
- GNU C 開發環境
  - gcc compiler 與標準 C library
  - 視個人需求，可選擇安裝 GNU Make 方便編譯程式
- GMP 開發套件

1.4 實驗步驟

我們將用以下實驗步驟，逐步引導使用 GMP 函式庫以及 Diffie-Hellman 金鑰交換的演算法實作。過程中會列出重要程式片段以及部份執行結果，至於完整程式的編寫、編譯、連結、及執行，則不在本實驗教學範圍內。

1.4.1 GMP 的整數資料結構

在 GMP 中是以 mpz_t 這個資料結構代表一個整數，其實際佔用的記憶體會隨著運算而改變，因此，GMP 整數在使用前必須先初始化，所使用的函式為：

```
void mpz_init (mpz_t integer)
```

而使用結束後，則呼叫清除函式，將佔用的資源釋放：

```
void mpz_clear (mpz_t integer)
```

至於指定 GMP 整數的值，則有下列幾個函式：

```
void mpz_set (mpz_t rop, mpz_t op)
void mpz_set_ui (mpz_t rop, unsigned long int op)
void mpz_set_si (mpz_t rop, signed long int op)
```

這些函式都是將 op 變數所代表的數值轉換成 mpz_t 型態，並儲存於 rop 變數中。而最具彈性的數值指定方式則是：

```
int mpz_set_str (mpz_t rop, char *str, int base)
```

這個函式中，str 參數是一個字串，可用整數基底 2 到 62 為 base 表示；如果 base 為 0，則字串開頭 0x 或 0X 代表基底 base=16，0b 或 0B 代表

base=2，0 代表 base=8，其他字元開頭則代表 base=10。這裡要注意的是數字的表示可為 0...9、A..Z、以及 a...z，當 base 為 2 到 36 時，英文字母大小寫皆代表相同數值，例如 A=a=10；當 base 為 37 到 62 時，英文字母的大小寫就有區別了，A...Z 代表 10...35，a...z 代表 36...61。此函式的返回值若是 0，代表該字串可被轉換成 mpz_t 型態，否則代表轉換失敗。

為了方便使用，GMP 另外提供一組函式，結合初始化與指定數值：

```
void mpz_init_set (mpz_t rop, mpz_t op)
void mpz_init_set_ui (mpz_t rop, unsigned long int op)
void mpz_init_set_si (mpz_t rop, signed long int op)
int mpz_init_set_str (mpz_t rop, char *str, int base)
```

各參數的意義同上，此處就不贅述。

有時要想將 mpz_t 整數轉換型態作為其他用途(例如印在螢幕上)時，可以利用轉換函式，以下介紹其中三個：

```
unsigned long int mpz_get_ui (mpz_t op)
signed long int mpz_get_si (mpz_t op)
char * mpz_get_str (char *str, int base, mpz_t op)
```

前兩個函式比較單純，就是將 op 轉換成 unsigned long int 或 signed long int，不過必須注意當 op 的值超過 unsigned long int 或 signed long int 能表示的範圍時，會發生截斷(truncation)的問題。而第三個函式將 op 轉換成以 base 為基底的字串，基本上不會發生無法表示的問題，但須注意參數 str。當 str 為 NULL 時，函式會自動配置字串記憶體，此時要注意的是字串使用完畢後必須記得釋放記憶體；當 str 不為 NULL 時，其回傳值就是 str 本身，但此時需注意 str 所指向的記憶區塊必須足夠容納轉換後的字串，否則可能發生記憶體溢位的狀況。

在後續的實驗步驟中，假設 Alice 與 Bob 皆同意 p=23 與 g=5，則模擬程式中可能看到類似下面片段：

```
mpz_t p, g;
mpz_init_set_ui(p, 23);
mpz_init_set_ui(g, 5);
...
mpz_clear(g);
mpz_clear(p);
```

1.4.2 測試是否為質數

這個步驟與模擬金鑰交換沒有太大關聯，但對於設計金鑰交換的參數而言，卻是相當重要。回到前面說的，金鑰交換的安全基礎建立於不易由截獲的資訊逆推雙方選定的秘密參數，而之所以不易逆推是因為選取的 p 值很大。當我們進行金鑰交換前，首先面對的一個難題就是如何找尋一個很大的質數 p。

尋找大質數是個數學難題，目前並沒有發現快速的尋找方法，因此實務上，這個問題變成：給定一個大整數，利用一些機率性的分解的方法測試，若無法分解的次數越多，代表其為質數的機會越高。在 GMP 中，用的是 Miller-Rabin 機率式質數測試法(Miller-Rabin probabilistic primality tests)，我們可以利用下面這個函式測試一個整數是否為質數：

int mpz_probab_prime_p (*mpz_t n, int reps*)

當函式的回傳值為 2 時，代表整數 n 可確認為一個質數；當回傳值為 1 時，代表 n 可能是個質數；而當回傳值為 0 時，則可確認 n 不是質數。至於 reps 參數，代表的是測試的次數，一般而言給 5 到 10 次，當然，如果給的次數越多，非質數被判定為可能是質數的機率就越低。