

CHAPTER 08

資訊安全



8-1 資訊安全的基本原則

8-2 資料機密性

8-3 資料完整性

8-4 系統可用性

8-5 網路攻擊

8-6 網路防護

8-7 資訊倫理



8-1 資訊安全的基本原則

- 資料機密性 (confidentiality)
 - 防止未經授權的第三者取得資料
- 資料完整性 (integrity)
 - 避免資料遭到竄改
- 系統可用性 (availability)
 - 旨在確保資料和系統可以可靠即時地取得





其他額外需求

- 信賴性 (authenticity)
 - ▶ 資料本身或是資料的來源是可以被驗證
- 究責性 (accountability)
 - ▶ 確保資料的不可否認性 (nonrepudiation)
 - ▶ 提供可靠的紀錄
 - ▶ 可依據紀錄追溯出應負責的個體





8-2 資料機密性

- ➡ 通常是透過「加密」(encryption) 的方式來進行。加密會把原本可以直接讀取的資料加以處理，轉換為另外一種無法直接讀懂的方式呈現。
- ➡ 有知道「密碼」(password，或是key) 的使用者，才可以透過「解密」(decryption) 的過程，取得原始的資料。

對稱式金鑰

非對稱式金鑰





加密演算法的二大類型

- 對稱式密碼演算法 (symmetric cryptographic algorithm)
 - ▶ 加密和解密使用同一組密碼
- 非對稱式密碼演算法 (asymmetric cryptographic algorithm)
 - ▶ 一共有二組密碼
 - ▶ 一組用來加密；另一組則用來解密
 - ▶ 也稱為「公開金鑰密碼系統」 (public key crypto system)
 - 因為二組密碼中的其中一組可以公開
 - 另一組必需要小心保管





對稱式金鑰的加解密演算法

- ▶ 對稱式金鑰的加解密演算法在處理資料加密和解密時，是使用相同的密碼。在加密的過程中，我們常稱我們的原始資料為本文（plaintext），而加密出來的資料稱為密文（ciphertext）。
- ▶ 加密的演算法要做的工作，就是將本文轉換為密文。





對稱式密碼演算法

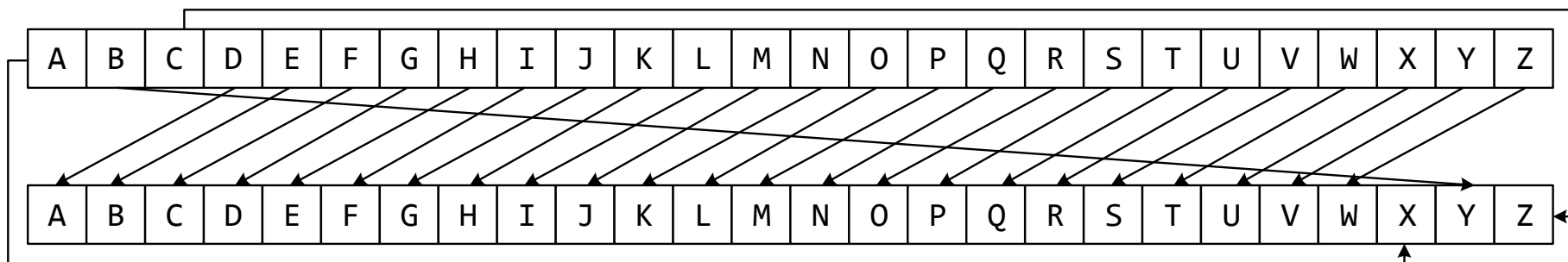
■ 對稱式加密流程示意圖





範例一：位移法

- ➡ 假設加密時將每個英文文字**往前移N位**
- ➡ 凱撒加密法 ($N=3$) : $E \rightarrow B$ 、 $D \rightarrow A$ 、 $C \rightarrow Z$ 、...
- ➡ 輸入本文：「ATTACK AT DAWN」
- ➡ 輸出密文：「XQQXZH XQ AXTK」
- ➡ 解密時，則改**往後移N位**





範例二：查表法

- ➡ 假設密碼為一個對照表
- ➡ 加密時查表 (本文→密文)
- ➡ 輸入本文：「hi, this is alice」
- ➡ 輸出密文：「fy, ofye ye tiysq.」

本文	a	b	c	d	e	f	g	h	i
密文	t	p	s	w	q	z	a	f	y
本文	j	k	l	m	n	o	p	q	r
密文	l	c	i	g	b	k	d	u	v
本文	s	t	u	v	w	x	y	z	
密文	e	o	r	j	n	h	m	x	



範例二：查表法 (續)

- ➡ 解密時可查同一張表 (密文→本文)
- ➡ 輸入密文：「fy, ofye ye tiysq.」
- ➡ 輸出本文：「hi, this is alice.」





範例三：使用XOR運算

➡ XOR運算 (以 \otimes 符號表示)，二進位運算的結果：

▶ $0 \otimes 0 = 0$

▶ $0 \otimes 1 = 1$

▶ $1 \otimes 0 = 1$

▶ $1 \otimes 1 = 0$





XOR加密運算

- ➡ 假設密碼為十進位的171
(等同於十六進位的AB，或是二進位的10101011)
- ➡ 若輸入本文為「h」，ASCII編號為十進位的104
(等同於十六進位的68，或是二進位的01101000)
- ➡ 加密時使用XOR運算來加密「h」，得到195

	0 1 1 0 1 0 0 0	(本文) 十六進位的68 (h)
⊗)	1 0 1 0 1 0 1 1	(密碼) 十六進位的AB
	1 1 0 0 0 0 1 1	(密文) 十六進位的C3 (密文) 等於十進位的195





XOR解密運算

- 假設密碼為十進位的171
(等同於十六進位的AB，或是二進位的10101011)
- 若輸入密文為十進位的195
(等同於十六進位的C3，或是二進位的11000011)
- 解密時使用XOR運算來解密195，得到「h」

	1 1 0 0 0 0 1 1	(密文) 十六進位的C3
⊗)	1 0 1 0 1 0 1 1	(密碼) 十六進位的AB
	0 1 1 0 1 0 0 0	(本文) 十六進位的68 (h)





完整的XOR加密結果

- 加密「hi, this is alice.」
- 得到密文(以ASCII編碼表示)為
「┐┘çï■┐┘÷ï┘÷ï≡┐┘ℒ÷à」

ASCII 本文	h	i	,		t	h	i	s		i	s		a	l	i	c	e	.
16 進位值	68	69	2c	20	74	68	69	73	20	69	73	20	61	6c	69	63	65	2e
加密 (⊕AB)	c3	c2	87	8b	df	c3	c2	d8	8b	c2	d8	8b	ca	c7	c2	c8	ce	85
ASCII 密文	┐	┘	ç	ï	■	┐	┘	÷	ï	┘	÷	ï	≡	┐	┘	ℒ	÷	à
16 進位值	c3	c2	87	8b	df	c3	c2	d8	8b	c2	d8	8b	ca	c7	c2	c8	ce	85
解密 (⊕AB)	68	69	2c	20	74	68	69	73	20	69	73	20	61	6c	69	63	65	2e
ASCII 本文	h	i	,		t	h	i	s		i	s		a	l	i	c	e	.

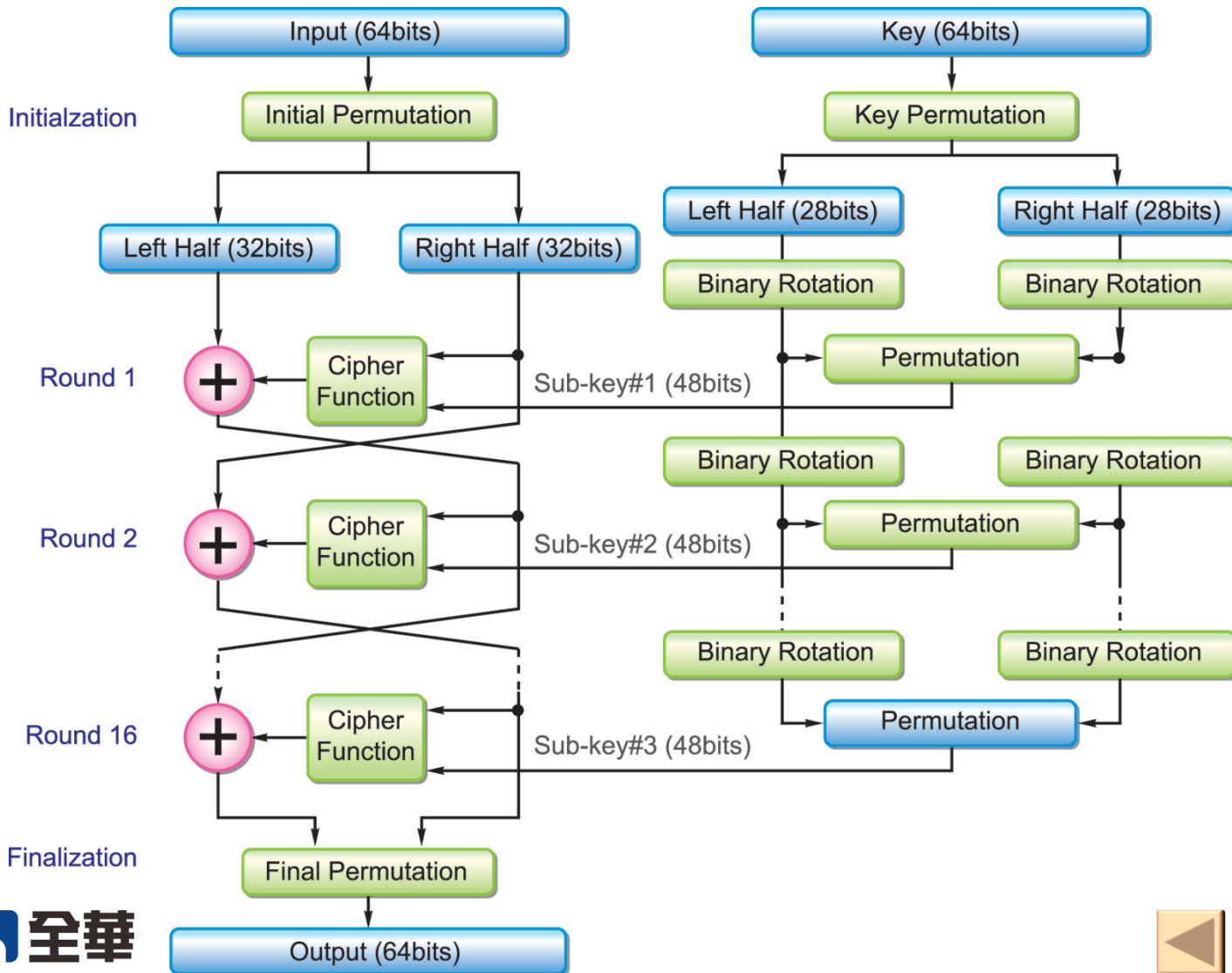




上述三個範例

- ➡ 簡單，但容易破解
 - ▶ 相同的本文得到相同的密文
 - ▶ 同時知道本文和密文時，可立刻破解
 - ▶ 或者是，累積夠多密文時，容易被破解
- ➡ 實際上的對稱式密碼演算法更為複雜
 - ▶ 查表 + XOR + 排列組合 等等複合方式
 - ▶ 密碼長度更長：56-bit、64-bit、128-bit、256-bit等
 - ▶ 範例：DES演算法的加密流程 (下頁圖)

DES演算法示意圖





對稱式演算法的分類

➡ 區塊式

- ▶ 加密以固定大小的區塊為單位
- ▶ 輸入的本文資料切割為區塊，每一個區塊分別加密
- ▶ 區塊大小通常由密碼的長度決定
- ▶ 資料長度不足一個區塊的部份，補0
- ▶ 常見的演算法：IDEA、DES、AES、RC5等

➡ 目前的加密標準採用AES演算法





對稱式演算法的分類 (續)

- ➡ 串流式
 - ▶ 加密以bit為單位
 - ▶ 輸入的本文長度和輸出的密文長度通常相同
 - ▶ 常見的演算法：A5、RC4





對稱式演算法：IV及運作模式

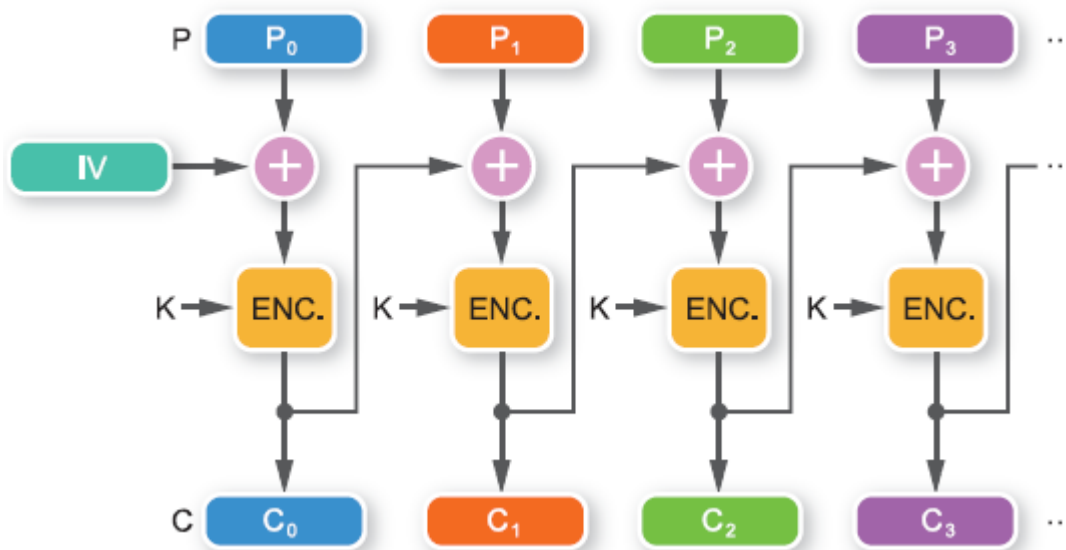
- ➡ 為了避免「相同的本文使用相同的密碼加密，得到相同的密文」的結果
- ➡ IV (initial vector)：初始向量
 - ▶ 區塊式和串流式演算法均適用
 - ▶ 將初始向量混入資料一起加密，以產生不同的結果
- ➡ 運作模式 (mode of operations)
 - ▶ 搭配區塊式演算法使用
 - ▶ 將加密的結果混入本文，以產生不同的結果
 - ▶ 常見的模式：CBC、CFB、OFB、GCM等





操作模式：以CBC為例

- ➡ 本文 P 切割為多個區塊： P_0, P_1, P_2, \dots
- ➡ IV 為初始向量
- ➡ 密碼為 K ，加密演算法為 ENC
- ➡ 輸出密文 C 同樣為多個區塊： C_0, C_1, C_2, \dots





非對稱式金鑰的加解密演算法

- 二組密碼，一組用來加密，一組用來解密
- 加密的常稱為「公鑰」；解密的常稱為「私鑰」
- 公鑰：公開給全部人知道
- 私鑰：只有自己知道





非對稱式的演算法 (續)

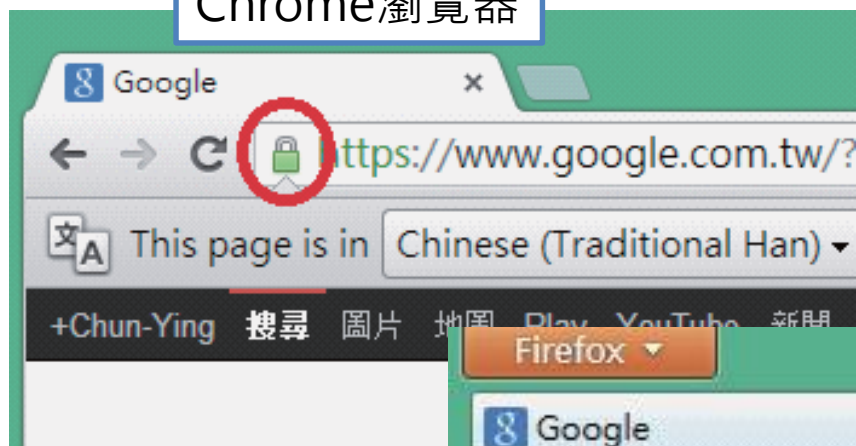
- 非對稱式演算法的常見應用
 - 加密
 - 數位簽章
- 加密時：公鑰用來加密；私鑰用來解密
- 簽章時：私鑰用來簽章；公鑰用來驗證簽章
- 注意：加密用的那一對密碼和簽章用的那一對密碼不可混用！
- 常見的演算法：RSA演算法、Diffie-Hellman演算法



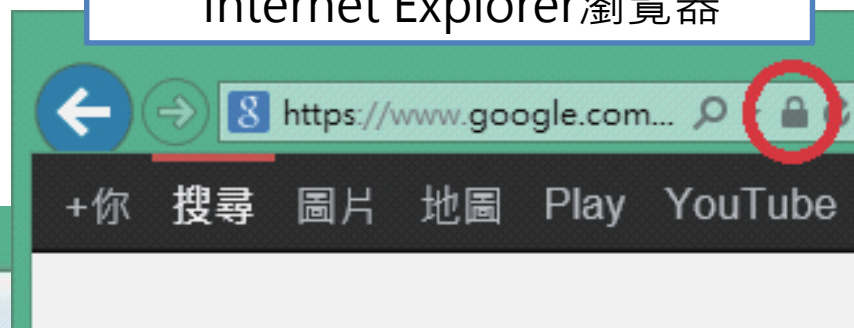
8-2-3 網路上常見的加解密演算法應用

- 日常生活隨處可見
- 安全的資料傳輸，如HTTPS

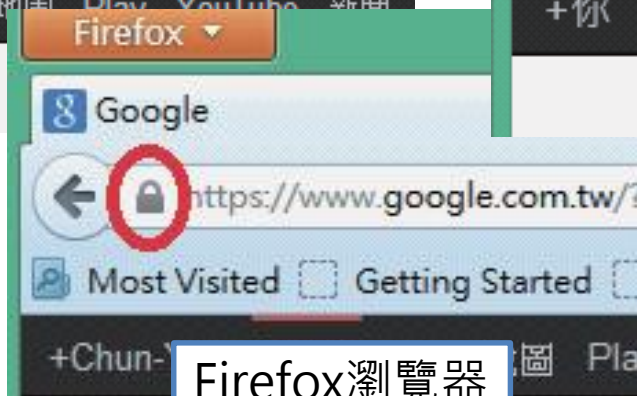
Chrome 瀏覽器



Internet Explorer 瀏覽器



Firefox



Firefox 瀏覽器





光是看到HTTPS是不夠的...

■ 以Google Chrome為例

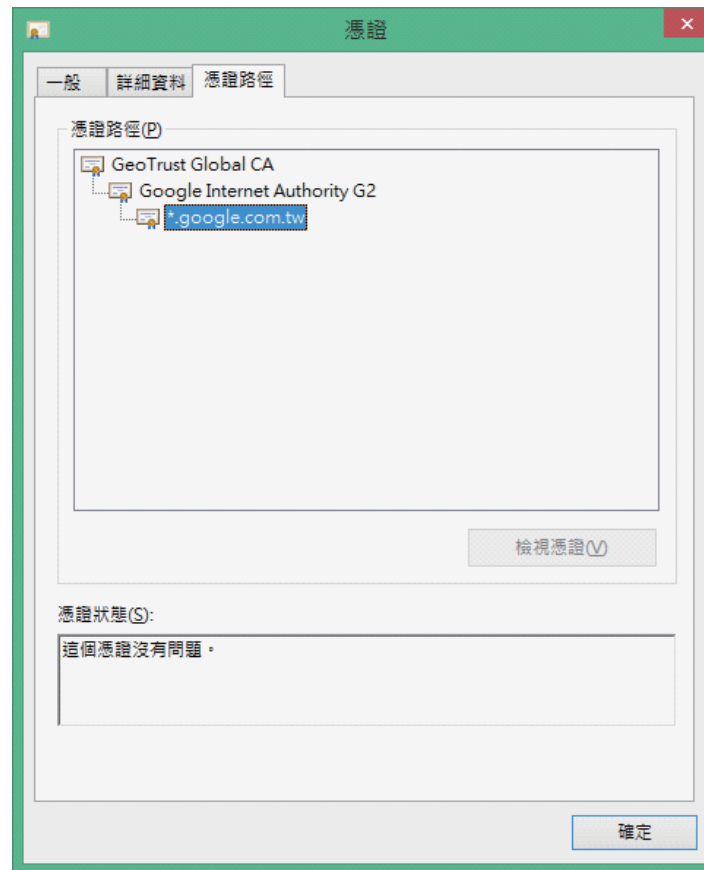
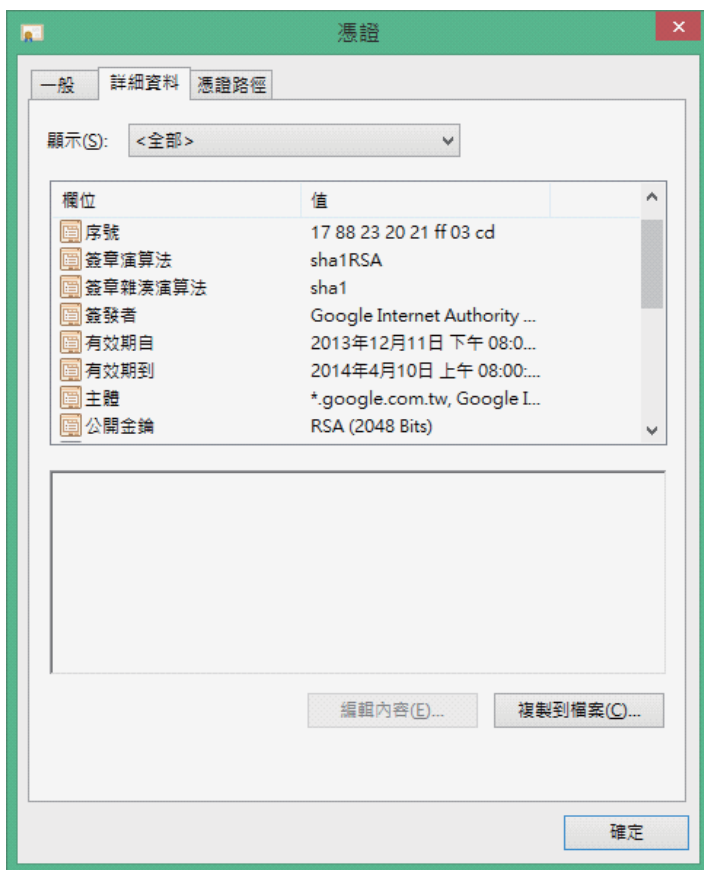
■ 左：沒問題的網站；右：憑證有問題的網站





檢視憑證的詳細資訊

■ Google網站的憑證





非對稱式演算法的基本原理**

- ➡ 使用數學上難解的問題
- ➡ 離散對數：給定二個數字 x 和 y
 - ▶ 計算 x 的 y 次方的結果 (x^y) 很容易
 - ▶ 反過來說，只知道 x^y ，要得到 x 和 y 很困難
 - ▶ 例： $x=17; y=21;$
 $x^y=69091933913008732880827217$
- ➡ 因數分解：給定二個質數 p 和 q
 - ▶ 計算 p 乘上 q 的結果 (pq) 很容易
 - ▶ 反過來說，只知道 pq ，要得到 p 和 q 很困難
 - ▶ 例： $p=15289; q=25903; pq=396030967$



RSA演算法及其流程

➡ 基於質數的因數分解，用來做資料加解密

1. 先選二個質數 p 和 q ，計算 $N=pq$
 2. 計算 $\varphi(N) = (p-1) * (q-1)$
 3. 從 1 到 $\varphi(N)$ 中挑選一個整數 e ， e 和 $\varphi(N)$ 需互質
 4. 計算出 e 的乘法反原素 d ，即 $d * e$ 除以 $\varphi(N)$ 的餘數等於1。通常以「 $d * e \equiv 1 \pmod{\varphi(N)}$ 」表示
 5. 產生出來的二組密碼為 (e, N) 以及 (d, N)
- ➡ 任選一組為公鑰，如 (e, N) ；另一組為私鑰，如 (d, N)





RSA演算法及其流程 (續)

- ➡ 產生好二組密碼(e , N)及(d , N)後
- ➡ 加密時
 - ▶ 若本文為 n
 - ▶ 計算密文 c 為 n^e 除以 N 的餘數，即 $n^e \equiv c \pmod{N}$
- ➡ 解密時
 - ▶ 若密文為 c
 - ▶ 計算本文 n 為 c^d 除以 N 的餘數，即 $c^d \equiv n \pmod{N}$





RSA範例一

- ➡ 選定 $p=3$; $q=11$
- ➡ $N = 3 * 11 = 33$; $\varphi(N) = (3-1) * (11-1) = 20$
- ➡ 選 $e = 3$, 算出 $d = 7$ ($3 * 7$ 除以 20 的餘數為 1)
- ➡ 公鑰為 $(3, 33)$; 私鑰為 $(7, 33)$
- ➡ 加密時 : 本文為 29
 - ▶ 計算 29^3 除以 33 , 餘數為 2(密文)
- ➡ 解密時 : 密文為 2
 - ▶ 計算 2^7 除以 33 , 餘數為 29(本文)





RSA範例二

- ➡ 選定 $p=7$; $q=11$
- ➡ $N = 7 * 11 = 77$; $\varphi(N) = (7-1) * (11-1) = 60$
- ➡ 選 $e = 17$, 算出 $d = 53$ ($17 * 53$ 除以 60 的餘數為 1)
- ➡ 公鑰為(17 , 77) ; 私鑰為(53 , 77)
- ➡ 加密時 : 本文為45
 - ▶ 計算 45^{17} 除以 77 , 餘數為 12(密文)
- ➡ 解密時 : 密文為12
 - ▶ 計算 12^{53} 除以 77 , 餘數為 45(本文)





實務上的RSA

- ➡ 選的 p 和 q 是長達 1024-bit 甚至 2048-bit 的質數
 - ▶ 約 30 至 60 位數的數字
 - ▶ 需要大數運算
 - ▶ 運算非常耗時
 - ▶ 通常不會用來加密大量的資料
 - ▶ 可配合對稱式密碼使用
 - 加密臨時產生、長度有限的對稱式密碼
 - 使用對稱式密碼進行大量的資料加密





Diffie-Hellman演算法**

- ➡ 基於離散對數，用來做密碼交換
- ➡ 簡稱為DH演算法
- ➡ 使用情境
 - ▶ 在公開的場合，二個人大聲的交換密碼
 - ▶ 旁邊可能有人在偷聽
 - ▶ 密碼可以交換成功
 - ▶ 但偷聽的人還是猜不出密碼是什麼





DH演算法的流程

- ➡ 假設甲和乙二個人要交換密碼
- ➡ 挑選合適的數字 g 和 p ，這二個數字大家都可以知道
- ➡ 甲和乙各自選一個數字 a 和 b ：甲知道 a ；而乙知道 b
- ➡ 甲計算 g^a 除以 p 的餘數，然後傳給乙
- ➡ 乙計算 g^b 除以 p 的餘數，然後傳給甲
- ➡ 甲可以計算 $(g^b)^a$ ，得到共同的密碼 g^{ab} 除以 p 的餘數
- ➡ 乙可以計算 $(g^a)^b$ ，得到共同的密碼 g^{ab} 除以 p 的餘數
- ➡ 偷聽的第三者無法算出共同的密碼 g^{ab} 除以 p 的餘數





DH的例子

- ➡ 假設公開的資訊 $g=5; p=23$
- ➡ 甲選了 $a=6$
- ➡ 乙選了 $b=15$
- ➡ 甲傳送 $g^a \pmod{p} = 5^6 \pmod{23} = 8$
- ➡ 乙傳送 $g^b \pmod{p} = 5^{15} \pmod{23} = 19$
- ➡ 甲可算出共用密碼 $(g^b)^a \pmod{p} = 19^6 \pmod{23} = 2$
- ➡ 乙可算出共用密碼 $(g^a)^b \pmod{p} = 8^{15} \pmod{23} = 2$





實務上的DH

- ➡ 同樣是使用非常大的數值來進行運算
- ➡ 算出來的密碼通常只用一次
- ➡ 用完就丟了，所以就算被破解了，也沒有太大的意義
- ➡ DH演算法是目前公開交換密碼演算法的基礎





8-3 資料完整性

- ➡ 驗證資料是否遭到竄改或破壞
- ➡ 密碼學的雜湊函數
cryptographic hash function
- ➡ 數位簽章
digital signature





8-3-1 密碼學的雜湊函數

- ➡ Cryptographic hash function，簡稱為 hash 函數
- ➡ 將任意長度的字串進行運算後，得到一固定長度的雜湊值
- ➡ 常見密碼學的雜湊函數
 - ▶ MD5 – message digest 5：產生128-bit的雜湊值
 - ▶ SHA-1 – secure hash algorithm 1：產生160-bit的雜湊值
 - ▶ SHA-256 – secure hash algorithm 2 with 256-bit digest sizes：產生256-bit的雜湊值

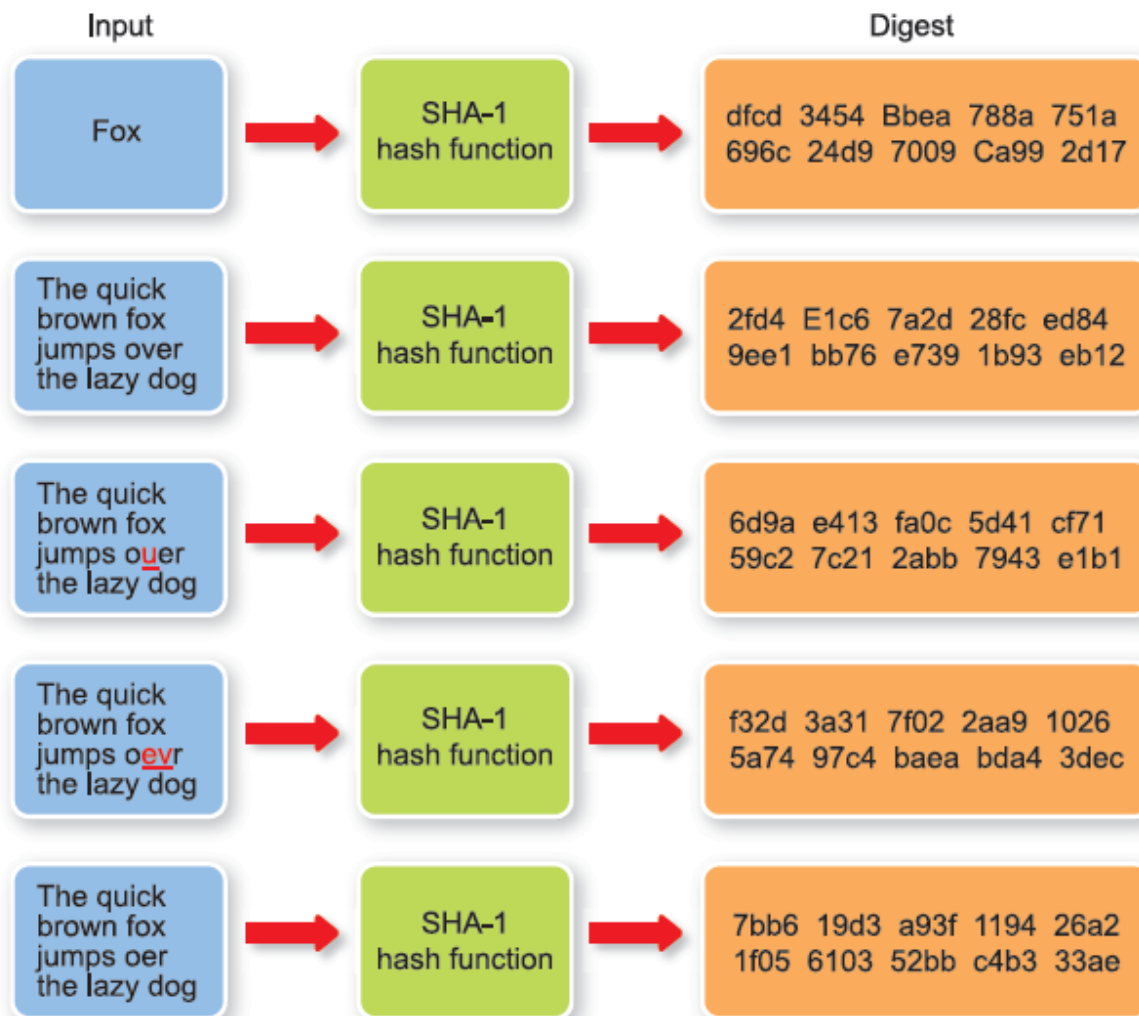




密碼學的雜湊函數 (續)

- ➡ 輸出的長度固定
 - ▶ 一般而言，輸出愈長的演算法愈安全
- ➡ 任一微小變動，其結果就會完全不同
- ➡ 單向函數
 - ▶ 無法從結果回推原始輸入
- ➡ 不易發生碰撞 (collision)
 - ▶ 不同的輸入，不易得到相同的結果

利用SHA-1函數對相似字串計算雜湊函數得到的結果





雜湊函數的應用

- ➡ 驗證資料傳輸是否無誤
 - ▶ 傳送資料 m 以及雜湊值 $h = \text{hash}(m)$
 - ▶ 接收端收到 m' 後，計算 $h' = \text{hash}(m')$ ，並比較 h 與 h'
 - ▶ h 和 h' 結果相同時，表示傳輸內容無誤
- ➡ 訊息驗證碼 (message authentication code, MAC)
 - ▶ 確認資料是由受信賴的使用者傳輸
 - ▶ 使用 hash 函數來實作時，常簡寫為 HMAC





HMAC的基本概念

- ➡ 假設傳送端和接收端共享一組密碼 c
- ➡ 傳送資料 m 時，計算 $h = \text{hash}(m \parallel c)$ ，然後傳送 m 以及 h
- ➡ 接收端收到 m' 時，計算 $h' = \text{hash}(m' \parallel c)$ ，然後比較 h 以及 h'
- ➡ 如果 h 和 h' 相同，表示資料是來自受信賴的使用者，而非由他人偽造
- ➡ 當然實際上還需要再配合其他參數，以避免其他不同類型的網路攻擊





8-3-2 數位簽章



- 確認資料是來自特定的使用者
 - 假設私鑰只有擁有者自己知道，且妥善保護
 - 需經過特定使用者，使用其私鑰進行簽章
 - 文件和簽章資料一同送出，接收端可使用簽章者的公鑰進行驗證





數位簽章

- ➡ 以RSA為例，簽章的動作和加解密的運算相同
- ➡ 加解密時
 - ▶ 用公鑰加密，傳送出去後，用私鑰解密
- ➡ 簽章時
 - ▶ 用私鑰簽章，傳送出去後，用公鑰驗證
- ➡ 簽章和加密的動作相同；解密和驗證的方式相同
- ➡ 再次強調：加密和簽章**不可**使用同一對密碼
 - ▶ 產生一對加解密專用的金鑰
 - ▶ 再產生另一對簽章專用的金鑰



實務上數位簽章的做法

- ➡ 公開金鑰(如RSA)的運算非常耗時
- ➡ 搭配雜湊函數使用
- ➡ 不論原始資料的長度為何
- ➡ 先將要簽章的資料，用雜湊函數加以計算
 - ▶ 有效將資料長度縮短為128至512-bit
- ➡ 將雜湊出來的值進行簽章保護
 - ▶ 只需要針對16至64位元組的資料進行簽章





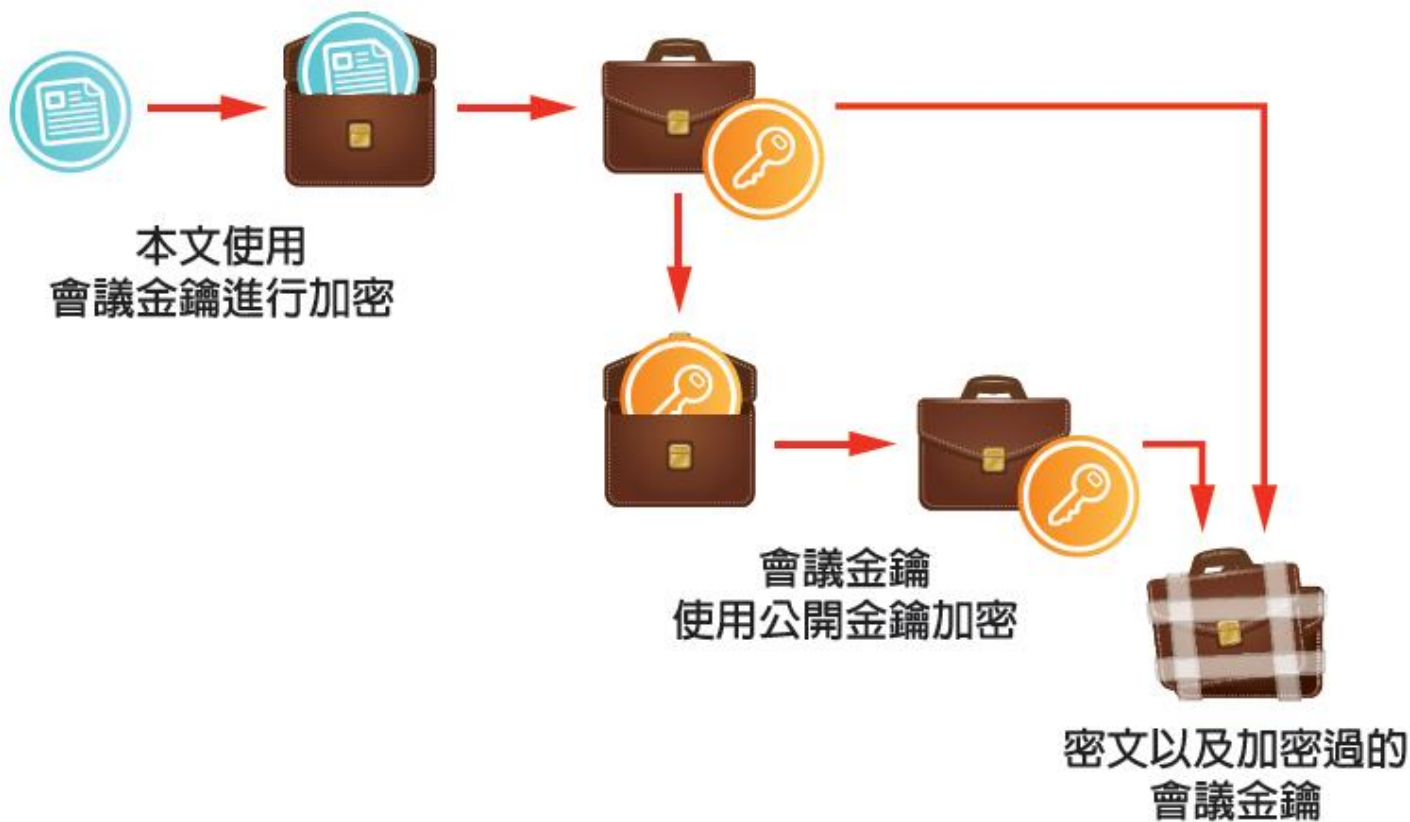
數位簽章的應用 – PGP

- ➡ PGP , pretty good privacy
- ➡ 安全的Email傳輸：安全傳輸 + 數位簽章
- ➡ 安全傳輸
 - ▶ 使用一次性密碼，配合對稱式演算法加密信件內容
 - ▶ 一次性密碼，也稱為「會鑰金鑰」(session key)
 - ▶ 一次性密碼透過公開金鑰演算法傳輸
- ➡ 數位簽章
 - ▶ 計算信件內容雜湊值後，再用公開金鑰演算法簽章



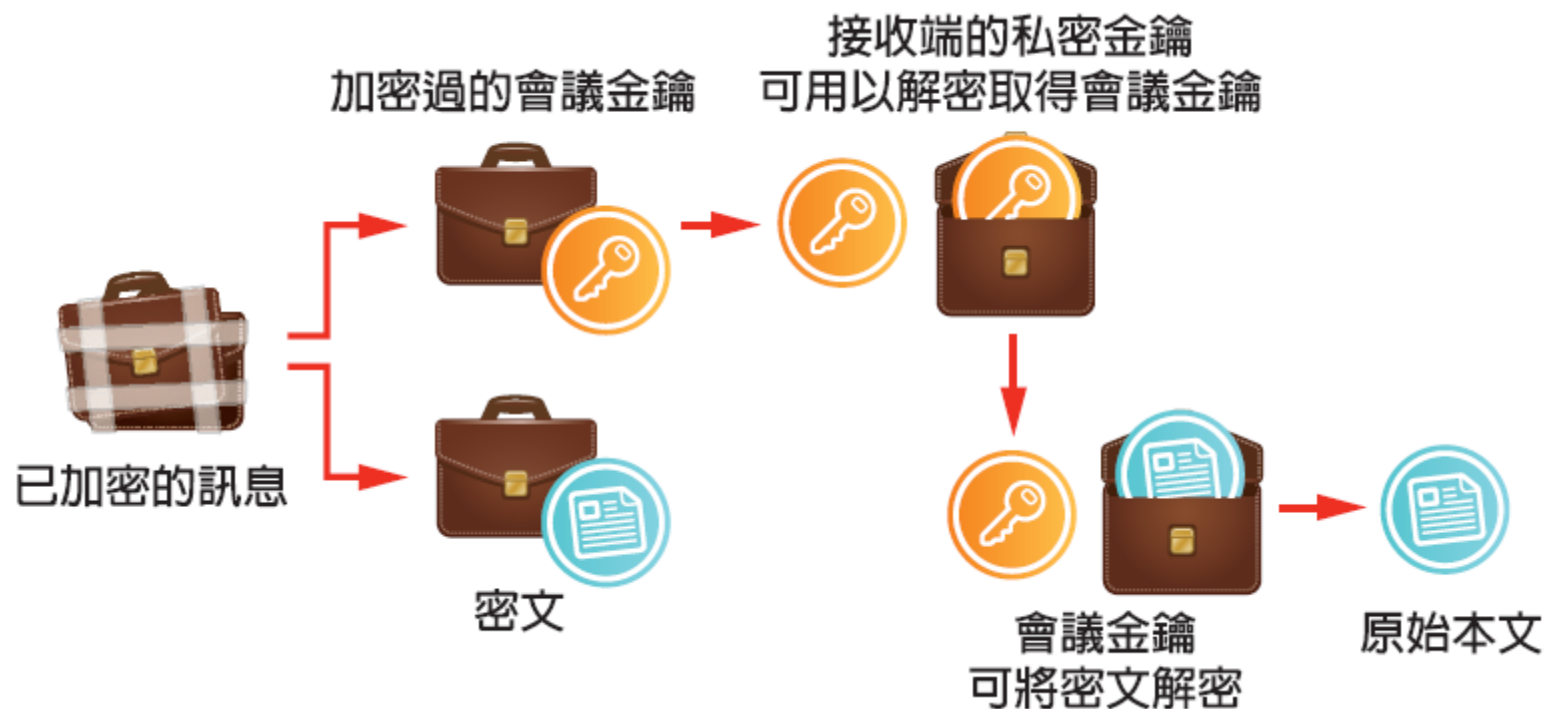


PGP的加密流程示意圖

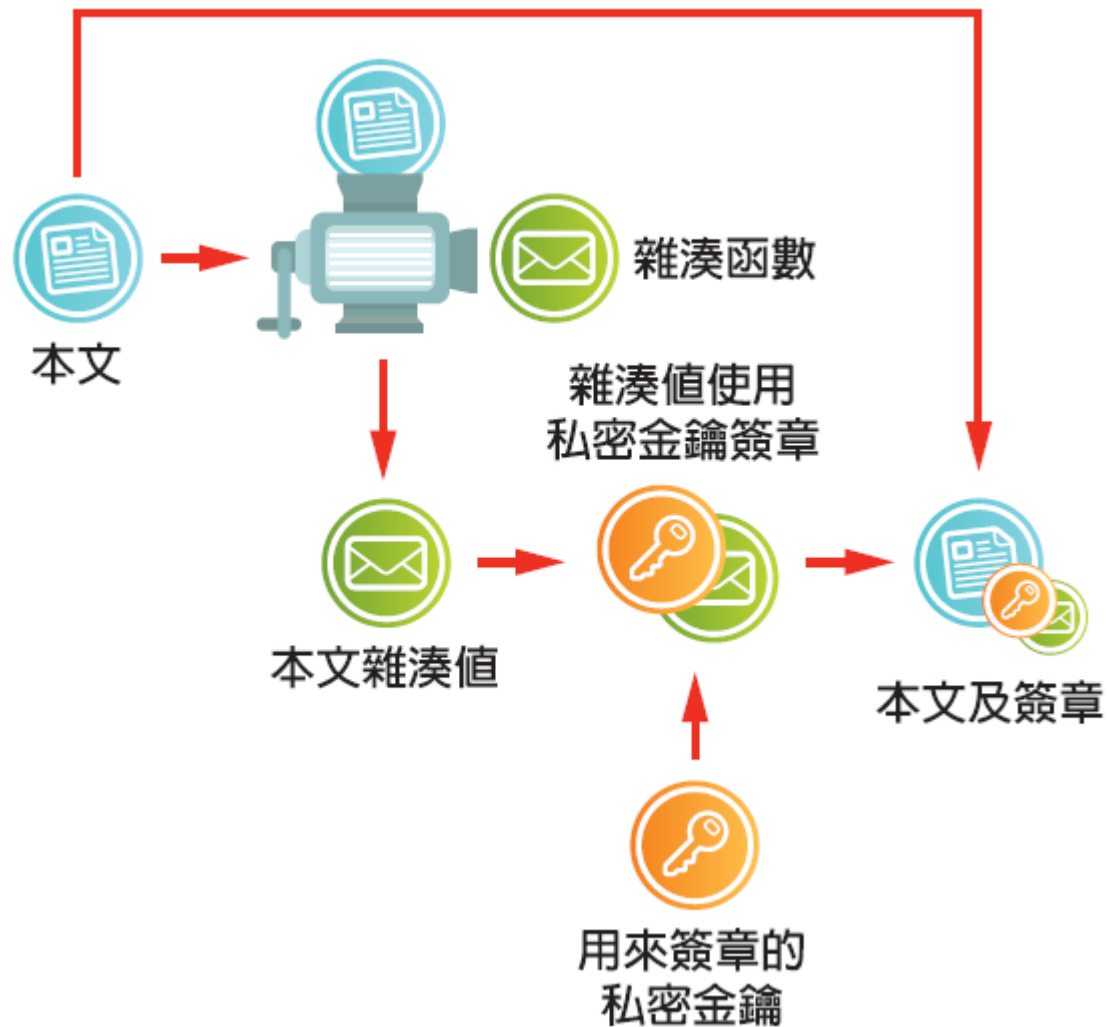




PGP的解密流程示意圖



PGP的數位簽章示意圖





公開金鑰的管理

- ➡ 使用公開金鑰的演算法，都會用到公鑰的部份
 - ▶ 加解密時用來「加密」
 - ▶ 簽章時用來「驗證」
- ➡ 如何驗證「公鑰」沒有問題？
- ➡ 集中式的管理：使用「公開金鑰基礎建設」
 - ▶ Public key infrastructure, PKI
 - ▶ 由第三方的公證機構(CA)認證公鑰
- ➡ 分散式的管理：Web of Trust





透過CA進行認證

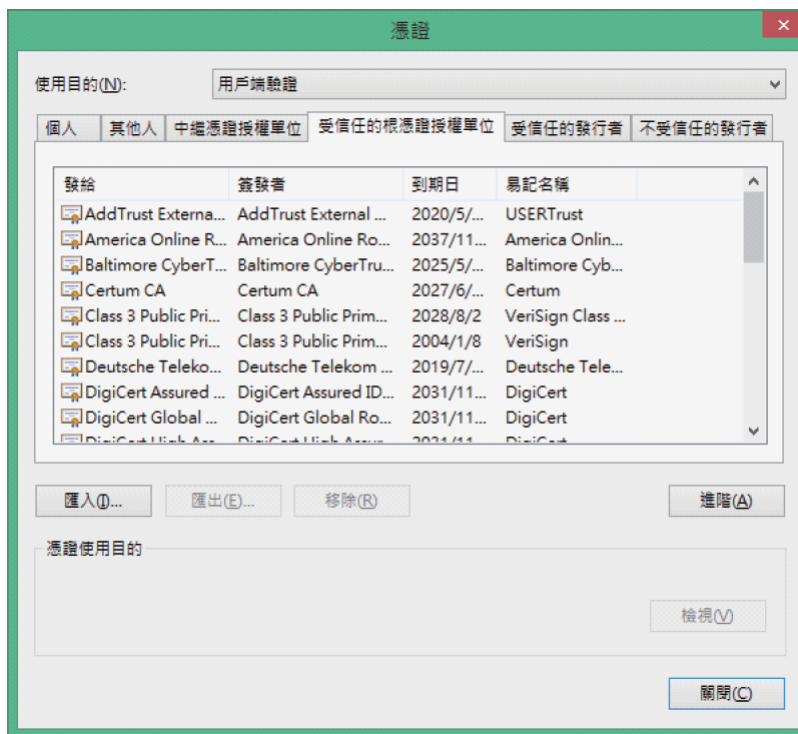
- ➡ CA：確保「公鑰」和使用單位的關聯性
- ➡ 產生「憑證」(certificate)證明這個關係
 - ▶ 範例：Google使用公開金鑰提供網站安全連線的服務
 - ▶ Google產生一對金鑰：公鑰 Ku 以及私鑰 Kr
 - ▶ 將 Ku 送給認證機構，由認證機構對 Ku 進行簽章，將得到簽章資訊 $\text{Sign}(\text{Ku})$
 - ▶ Google的網站上提供 $\text{Ku} + \text{Sign}(\text{Ku})$
 - ▶ 使用者可以用認證機構的公鑰，驗證Google的 Ku





上一頁的例子有什麼問題？

- ➡ 如何知道認證機構的公鑰沒有問題？
- ➡ 大部份的作業系統或瀏覽器裡有**內建**認證機構的公鑰
- ➡ 由認證機構來對公鑰簽章通常是**需要付費的**
- ➡ 以Windows為例





分散式的管理

- ➡ 不用付費
- ➡ 透過使用者之間，相互背書
- ➡ 也稱為Web of Trust，人與人之間的信賴關係網
- ➡ PGP常用這種方式
- ➡ 假設有甲、乙、丙三人
 - ▶ 甲、乙互相信任
 - ▶ 若丙的公鑰由乙來簽章(背書)
 - ▶ 那麼甲就可以直接相信丙的公鑰





分散式的管理

➡ 小世界理論

- ▶ 透過六、七層的人際網路，可以認識全世界的人
- ▶ 只要互相信任的使用者夠多，就可以透過使用者背書的方式，驗證大多數的公開金鑰

➡ 若信賴網的含蓋範圍不夠大，可能會發生金鑰無法驗證的情況





8-4 系統可用性

- ➡ 資訊系統可以在需要的時候正確地存取
- ➡ 可用性降低的原因
 - ▶ 系統本身的穩定度
 - 硬體故障
 - 軟體問題
 - ▶ 來自外部的攻擊



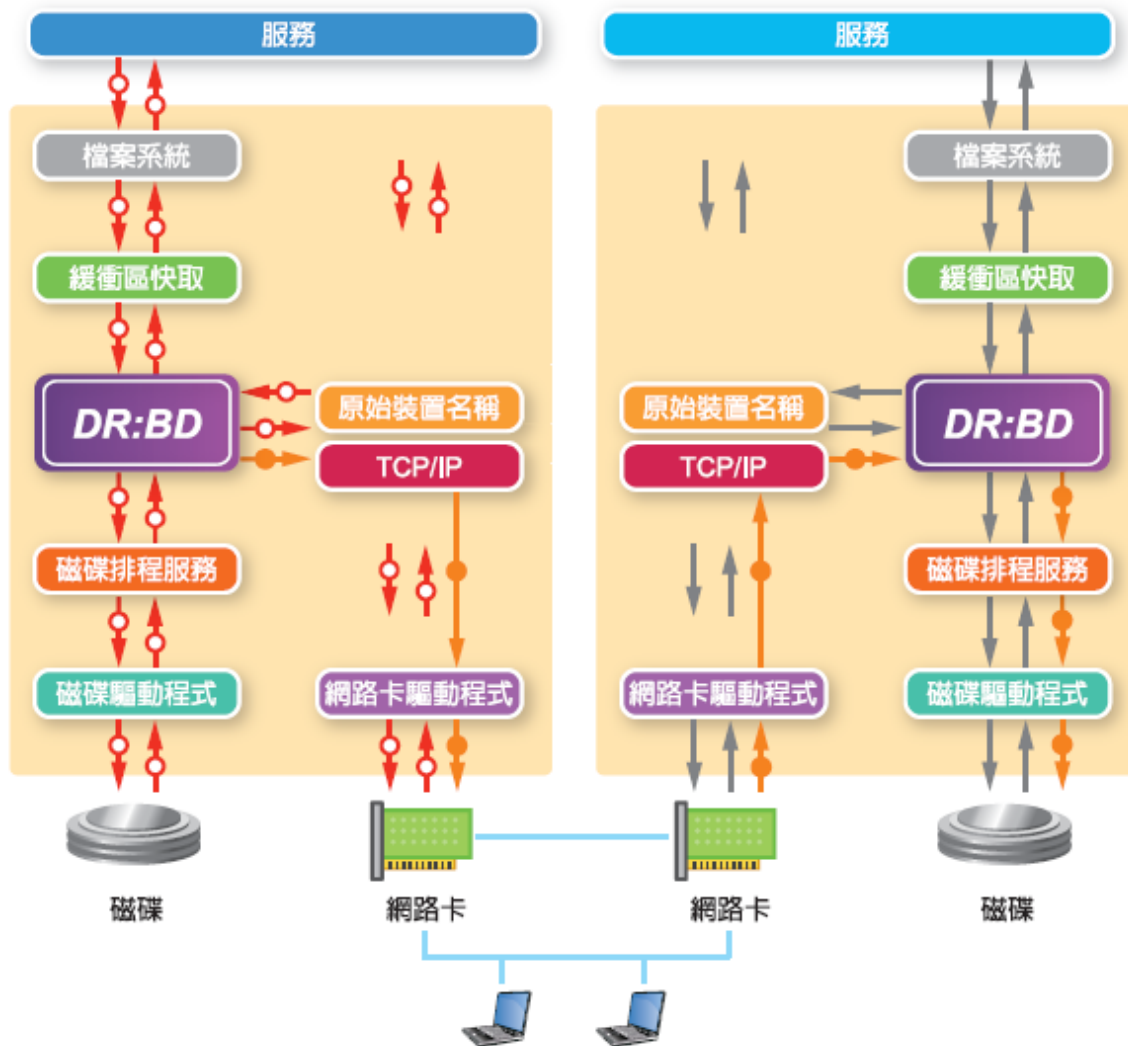


提高系統可用性

- ➡ 高可用性，high availability
- ➡ 建置備援系統
- ➡ 以網路系統而言 ...
 - ▶ 網路連線線路 x 2
 - ▶ 伺服器 x 2
 - ▶ 儲存設備 x 2
 - ▶ 故障偵測機制，常用的是「心跳機制」(heartbeat)



■ 儲存系統的高可用性





8-5 網路攻擊

- ➡ 目的
 - ▶ 取得未經授權的存取
 - ▶ 中斷網路服務的正常運作
- ➡ 所謂的「駭客」
 - ▶ Hacker：廣義來說，指對某項技術專精的人士
 - 不見得是壞人，有時還帶有稱讚的味道
 - ▶ Cracker：利用技術進行攻擊、破壞的人士





常見的網路攻擊

阻斷服務
攻擊

主機入侵

電腦病毒

網路監聽





阻斷服務攻擊

- ➡ 英文為denial of service，DoS
- ➡ 常見的方式為阻斷網路連線
 - ▶ 攻擊者產生大量網路流量，耗盡目標的網路頻寬
 - ▶ 產生大量的ping封包、TCP連線要求、或是UDP封包
- ➡ 分散式阻斷服務攻擊：distributed denial of service，DDoS
 - ▶ 透過一大群機器進行阻斷服務攻擊
 - ▶ 來自四面八方的機器，更難進行抵擋



ping指令的執行結果。用ping指令探測本機 IP 127.0.0.1

```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\huangant>ping 127.0.0.1

Ping 127.0.0.1 <使用 32 位元組的資料>:
回覆自 127.0.0.1: 位元組=32 時間<1ms TTL=128
回覆自 127.0.0.1: 位元組=32 時間<1ms TTL=128
回覆自 127.0.0.1: 位元組=32 時間<1ms TTL=128
回覆自 127.0.0.1: 位元組=32 時間<1ms TTL=128

127.0.0.1 的 Ping 統計資料:
    封包: 已傳送 = 4, 已收到 = 4, 已遺失 = 0 (0% 遺失),
    大約的來回時間 (毫秒):
        最小值 = 0ms, 最大值 = 0ms, 平均 = 0ms

C:\Users\huangant>
```





主機入侵

- ➡ 通常是利用系統上的漏洞
 - ▶ 所以要常常更新系統，以確保系統沒有安全漏洞
- ➡ 入侵後可再嘗試取得系統操作權限
- ➡ 常見的「網頁置換」攻擊
 - ▶ 駭客入侵網站後，將首頁換成指定的圖案





幾個網站「置換首頁」的範例畫面。
駭客們好像比較喜歡黑色底的介面呢！





主機入侵

- ➡ 竊取主機內的各種資料
 - ▶ 尤其是入侵大型商業網站、銀行網站、政府網站等
- ➡ 建置「釣魚網站」
 - ▶ 詐騙使用者的個人資訊
- ➡ 但，主機入侵技術也可以用在正途
 - ▶ 檢測網路服務的漏洞
 - ▶ 確保網路產品安全無虞





電腦病毒

- ➡ 帶有惡意的程式碼
 - ▶ 病毒：會不斷自我複製及感染其他檔案
 - ▶ 蠕蟲：蠕蟲可以透過網路散播
 - ▶ 特洛伊木馬程式：植入後門進而竊取資訊





巨集型病毒

- ➡ 隱藏在可夾帶巨集的文件檔案裡
 - ▶ 如Office Word、Excel、Outlook等等
- ➡ 文件檔案開啟時，巨集一併被執行
- ➡ 2000年的「ILOVEYOU」病毒
 - ▶ 使用者只要用Outlook收Email，一開始信件就中毒





檔案型病毒

- ➡ 寄生在執行檔裡的惡意程式
- ➡ 執行檔被執行時，就觸發藏在裡面的病毒
- ➡ 有時會再嘗試感染其他正常的程式
- ➡ 不要使用來路不明的軟體
 - ▶ 破解版、序號產生器





蠕蟲

- ▶ 透過網路散播
- ➡ 2003年的Blaster
 - ▶ 透過「網路上的芳鄰」進行散佈
 - ▶ 掃描網路上，安裝Windows作業系統的電腦
 - ▶ 進行感染，然後重覆掃描和感染的動作
 - ▶ 嚴重時，甚至影響區域網路運作





特洛伊木馬

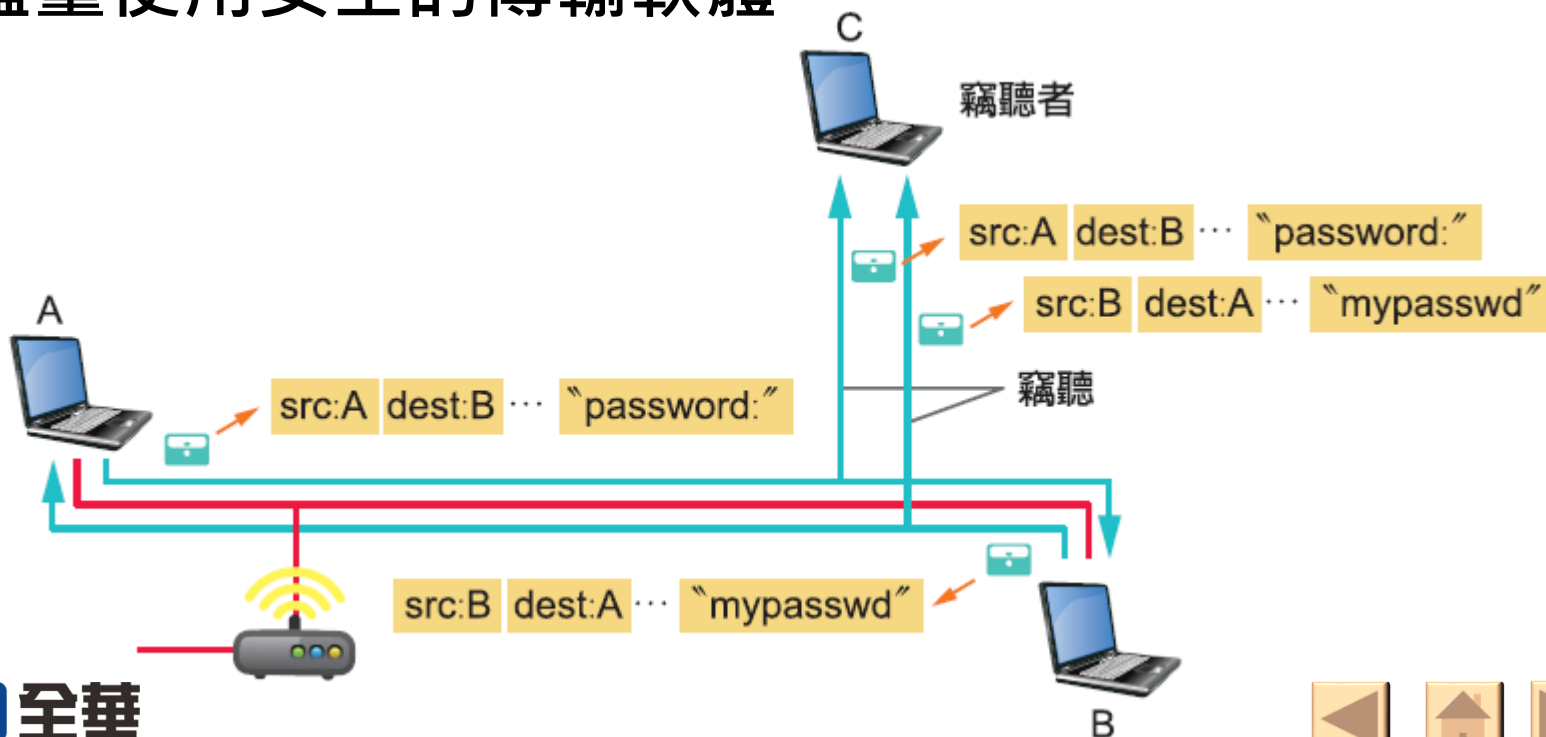
- ▶ 植入後門進而竊取資訊
 - 帳號、密碼、郵件等各種資料
- ➡ 除了竊取資料外，還可以遠端遙控
- ▶ 成為駭客的跳板
- ➡ 殭屍網路
 - ▶ 駭客控制一大群木馬
 - ▶ 發送垃圾郵件、發動分散式網路攻擊、進行網路釣魚等各種大規模的攻擊





網路監聽

- 網路上傳輸的資料若未加密，就可能遭到監聽
- 可取得使用者的帳號、密碼以及傳輸的資料
- 儘量使用安全的傳輸軟體





8-6 網路防護

防毒軟體

網路加密

防火牆與入
侵偵測系統

無線網路
安全





8-6-1 防毒軟體

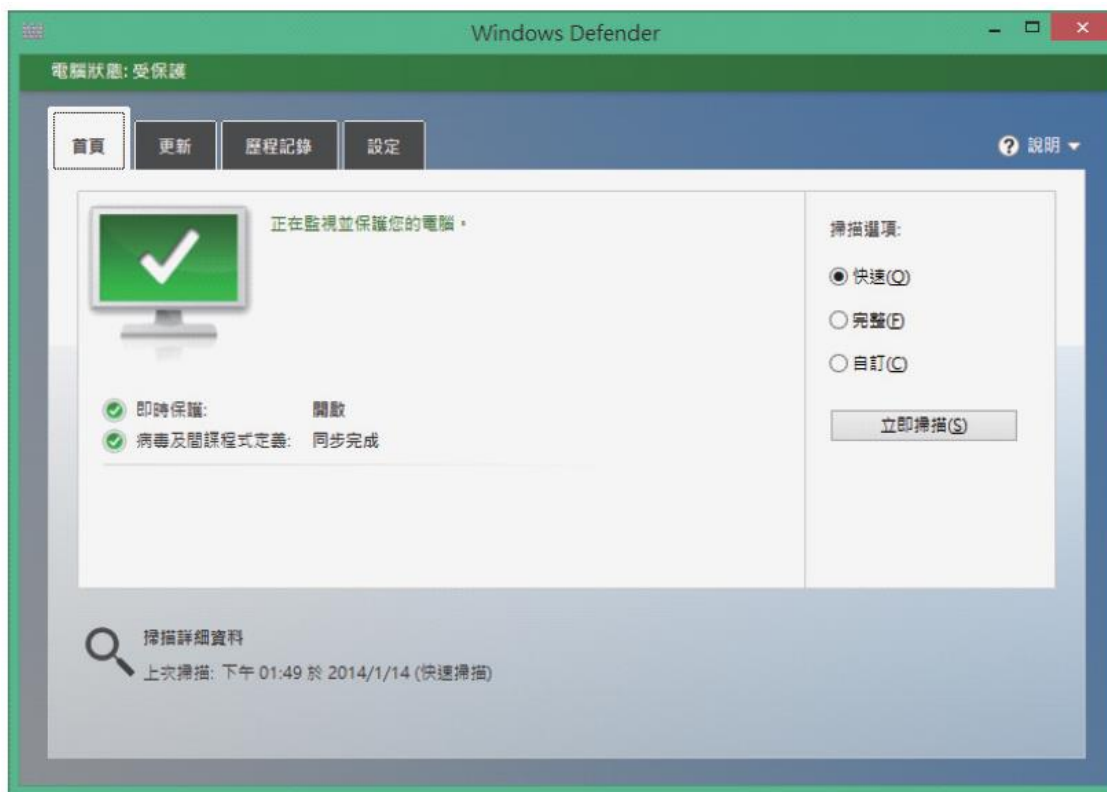
- ➡ 選擇多樣，有付費版，也有免費版
- ➡ 防毒軟體的偵測方式
 - ▶ 病毒的定義檔
 - ▶ 「啟發式」的偵測方式
- ➡ 使用防毒軟體之後並非一勞永逸，而是需要正確的使用習慣
 - ▶ 定期修補作業系統的漏洞
 - ▶ 不要使用來路不明的程式





防毒軟體 (續)

- ➡ 病毒很常見，甚至連作業系統都內建防毒軟體
- ➡ 圖為Windows內建的Windows Defender





偵測病毒

- ➡ 使用病毒定義檔進行偵測
 - ▶ 將病毒樣本的特徵碼字串取出來，建立資料庫
 - ▶ 透過字串比對，找出病毒
 - ▶ 要定義更新，確保可以偵測出最新的病毒
- ➡ 「啟發式」的偵測方式
 - ▶ 依據病毒的「行為模式」，如修改系統檔案、修改其他執行檔等等
- ➡ 目標：高偵測率、低誤判率、低漏判率



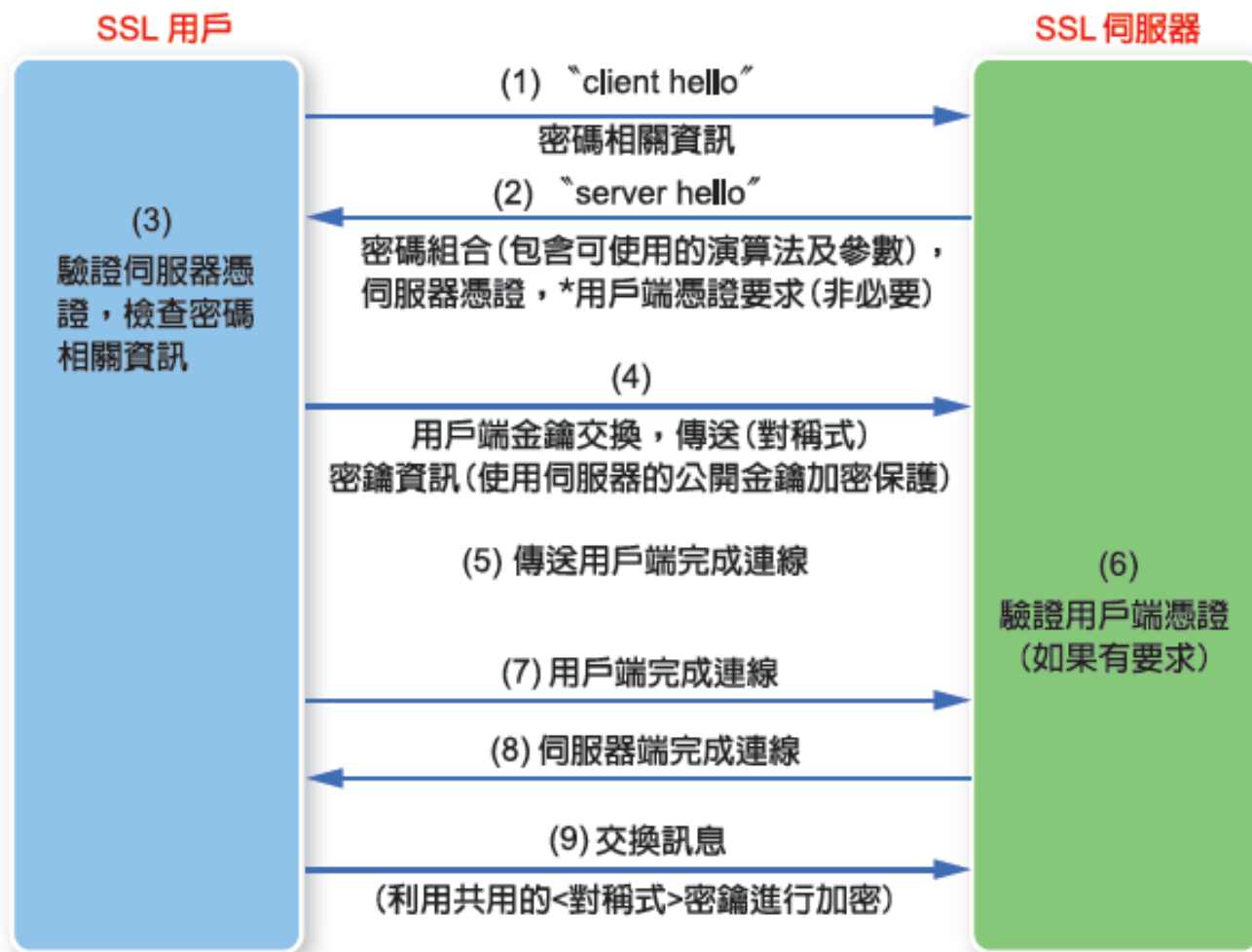


SSL與TLS

- ➡ 網路上常用的安全傳輸協定是SSL以及TLS
- ➡ SSL於1995年被提出
- ➡ TLS則於1999年被提出，是SSL的後繼者
- ➡ 提供應用層協定一個透通(transparent)的加密連線保護
 - ▶ 應用層協定可以不用修改
 - ▶ SSL/TLS安全連線建立後，應用層協定可在受保護的連線內進行資料傳輸



SSL/TLS連線示意圖





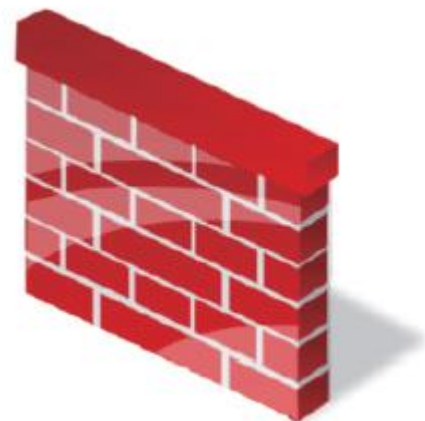
常見的SSL/TLS應用服務

- ➡ 安全的網頁瀏覽 HTTPS
- ➡ 安全的寄電子郵件 SMTPS
- ➡ 安全的下載電子郵件 POP3S/IMAPS
- ➡ 建立安全連線時，同樣需要檢查公鑰是否正確無誤





防火牆



- ➡ 避免網路型的攻擊
- ➡ 防火牆上可設定封包過濾規則
- ➡ 使用規則比對封包，以按照相對應的動作處理
 - ▶ 允許封包通過、丟棄封包，或是修改封包內容等
- ➡ 過濾規格可針對OSI協定的各層設定
- ➡ 防火牆通常置於區域網路連往廣域網路的位置
- ➡ 防火牆的規則大多從OSI的第二層到第四層





防火牆的分區

➡ WAN

- ▶ 廣域網路，用以連接外部網路

➡ LAN

- ▶ 區域網路，用以連接內部網路，即受保護的使用者

➡ DMZ

- ▶ 非軍事區 (demilitarized zone)
- ▶ 通常放置會對外服務的伺服器



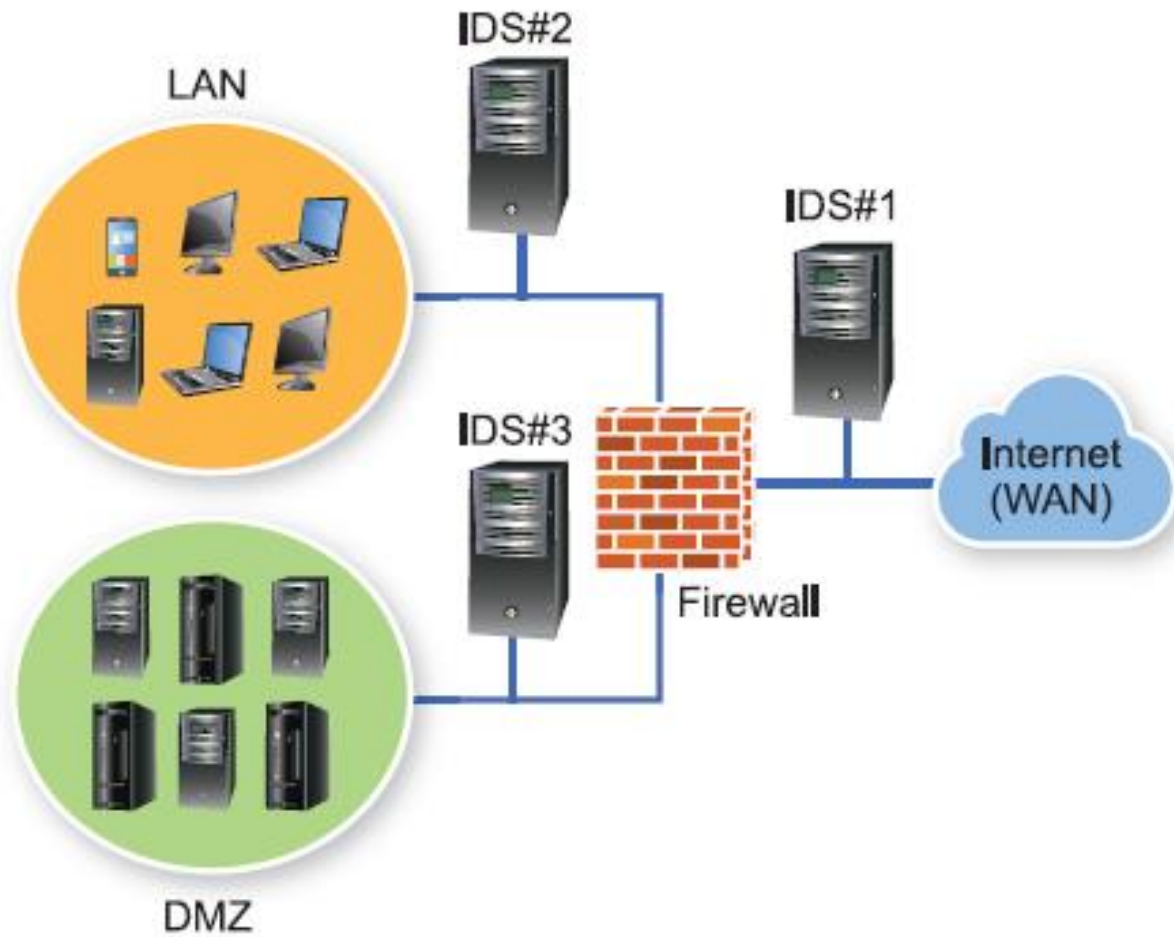


非軍事區的目的

- ▶ 通常放置會對外服務的伺服器
- ➡ 外部的使用者可以連線到這一區的伺服器
- ➡ 但這些伺服器不一定是安全的
- ➡ 萬一伺服器有漏洞被入侵成功了，避免駭客透過被攻陷的伺服器進入區域網路
- ➡ 可以達到隔離的效果



防火牆與入侵偵測系統示意圖





常見的基本防火牆規則

- ➡ 假設每一個封包均由規則編號小至大依序比對
- ➡ 比對符合的規則即執行，並乎略後面的規則
- 1. 允許LAN的主機透過任何協定建立新連線至任意位置
- 2. 允許DMZ的主機透過任何協定建立新連線至WAN
- 3. 允許WAN的主機透過TCP連接埠80 建立新連線至DMZ
(瀏覽網頁)
- 4. 允許所有已建立的連線通過防火牆
- 5. 禁止其他未定義的所有網路連線





入侵偵測系統

- ➡ 提供更完整的封包檢測功能
 - ▶ 從OSI第二層到第七層都可以檢測
 - ▶ 因此檢測時間較長、效率較差
 - ▶ 通常只用被動的方式進行檢測，不會攔截封包
- ➡ 可紀錄偵測到的入侵事件，並通知網路管理人員
- ➡ 入侵偵測系統擺放的位置非常有彈性
 - ▶ 防火牆前：可監測到所有可能的攻擊
 - ▶ 防火牆後：可監測到穿過防火牆的攻擊





入侵偵測系統的偵測方式

- ➡ 類似掃毒軟體
- ➡ 可使用特徵碼
 - ▶ 如，紀錄應用層的攻擊行為特徵碼
- ➡ 或是使用「啟發式」的規則
 - ▶ 如，統計特定主機發送封包的頻率、觀察主機是否存取列於黑名單的網站等等
- ➡ 同樣也有偵測率、誤判率、漏判率的問題





8-6-4 無線網路安全



- ➡ 無線網路愈來愈普及
- ➡ 無線網路更容易被監聽
- ➡ 無線網路的安全主要透過加密和認證來保護
- ➡ 一般的做法
 - ▶ SSID – 用來識別無線網路存取點
 - ▶ 密碼加密 – 確保資料傳輸的過程中不會被監聽
 - ▶ 帳號認證 – 通常只有企業無線網路採用



基本的保護

► 隱藏SSID

- 只有知道SSID的使用者可以連接
- 但仍有機會透過無線監聽的方式取得

► 利用無線網路卡的網路卡卡號過濾

- 可設定存取點僅允許特定的卡號連線
- 非常有效的方式，但使用前需要先進行設定
- 卡號也可能被盜用

► 一定要使用強度夠強的加密方式





無線網路的認證

- 一般家用的存取點，通常只會設定加密
 - 目前最新的標準WPA2
- 企業用的存取點，還會要求使用者輸入帳號密碼後，才可上網
- 常見的做法
 - 先連上特定網頁，輸入帳號密碼
 - 或是，透過802.1X協定，進行帳號密碼認證





相關標準整理

標準	認證方式	加密方式	金鑰長度	說明
IEEE 802.1x	增強型認證	無	無	僅提供認證功能
WEP	無	RC4	40-bit 或 104-bit	較不安全，建議不要使用
WPA-Personal (或 WPA-PSK)	無	TKIP	128-bit 或 256-bit	較佳的安全認證功能，適用於個人網路
WPA-Enterprise	802.1x	TKIP	128-bit 或 256-bit	較佳的安全認證功能，適用於中大型企業網路
WPA2-Personal (或 WPA2-PSK)	無	TKIP 或 AES	128-bit 或 256-bit	較佳的安全認證功能，適用於個人網路
WPA2-Enterprise	802.1x	TKIP 或 AES	128-bit 或 256-bit	較佳的安全認證功能，適用於中大型企業網路





8-7 資訊倫理

➡ 資訊隱私權

- ▶ 網路的便利使得資訊的交換與流通十分容易，因此必須規範個人擁有隱私的權利及防止侵犯別人隱私，以確保資訊在傳播過程中能保護個人隱私而不受侵犯。

➡ 資訊正確權

- ▶ 網路上的資訊垂手可得，難以分辨這些資訊是否正確，因此資訊提供者需負起確保提供正確資訊的責任，而資訊使用者則擁有使用正確資訊的權利。





8-7 資訊倫理

➡ 資訊財產權

- ▶ 資訊的再製和分享他人成果是相當容易的，所以應維護資訊或軟體製造者之所有權，並立法規範不法盜用者之法律責任，以保護他人的智慧成果。

➡ 資訊存取權

- ▶ 是指每個人都可以擁有以合法管道存取資訊的權利。例如：合法付費下載電子書閱讀；依創用CC授權標章原則，合法且合理使用他人作品等。





8-7 資訊倫理

- 除了上述的議題外，今日的資訊倫理還包含了提高使用者的倫理道德或社會使命感、建立正確價值觀、建立自律自重的守法美德等。這些議題可參考美國電腦倫理協會（Computer Ethics Institute）於1997年提出的電腦倫理的十大戒律（Ten Commandments of Computer Ethics）。





電腦倫理的十大戒律

不可使用電腦傷害他人。

不可干擾他人在電腦上的工作。

不可偷看他人的檔案。

不可利用電腦偷竊財務。

不可使用電腦造假。

不可拷貝或使用未付費的軟體。

未經授權，不可使用他人的電腦資源。

不可侵佔他人的智慧成果。

在設計程式之前，先衡量其對社會的影響。

使用電腦時必須表現出對他人的尊重與體諒。

