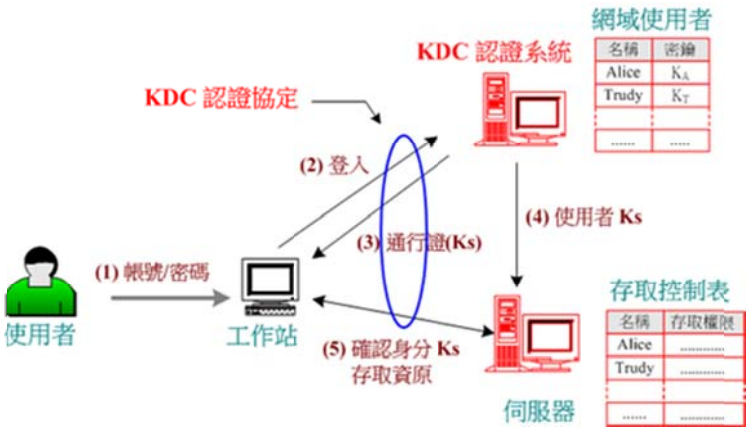
第十四章 Kerberos 認證系統

傳說中『地獄三頭神犬』 Kerberos 神威無比，看看祂如何阻擋外侵進入!!

14-1 認證協定與認證系統

14-1-1 認證協定與系統簡介

前面已介紹『鑰匙分配中心』(KDC)的基本觀念，接下來介紹幾種較常用的『KDC 認證協定』(KDC Authentication Protocol) 。到目前為止，我們瞭解在一個 KDC 管轄範圍之內，使用者與 KDC 之間都擁有一把相同的『共享密鑰』，便利用此密鑰互相確定身份，然後 KDC 再分配一把『會議鑰匙』(Session Key, K_S) 給使用者與伺服器 (使用者要求服務的)雙方，它們完全利用此會議鑰匙來互相通訊。以下分別介紹 KDC 認證協定的幾種演算法。



14-1-2 KDC 基本認證協定

我們以圖 14-1 (a) 說明 KDC 的基本構想。發起者 (客戶端，如 Alice) 向 KDC 提出希望與回應者 (伺服器，如 Bob) 通訊。首先 Alice 向 KDC 發出請求通往 Bob 的會議鑰匙 (訊號 (1)) ，其中 ID_A 與 ID_B 表示雙方身份，待 KDC 收到請求之後，分別以 Alice 與 Bob 的共享密鑰(K_A 與 K_B)傳送會議鑰匙給雙方(訊號 (2) 與 (3)，$E_{Ka}[ID_B \| K_S]$、$E_{Kb}[ID_A \| K_S]$) ，接下來，便可以利用該會議鑰匙來通訊，但其運作程序存在下列缺點：

◈ 回應者也許不存在，但發起者可能誤認是 KDC 鑰匙分配不當。

◈ 如果發起者重複要求的話，KDC 會連續送給回應者多把鑰匙，一旦回應者與發起者通訊時，可能會使用到舊的鑰匙。

◈ 容易遭受重播攻擊(Replay Attack)。攻擊者如欲破壞發起者與回應者之間的通訊，只要偽裝成發起者身份 (利用 IP 位址) ，連續提出多個請求，回應者就擁有多把鑰匙了。

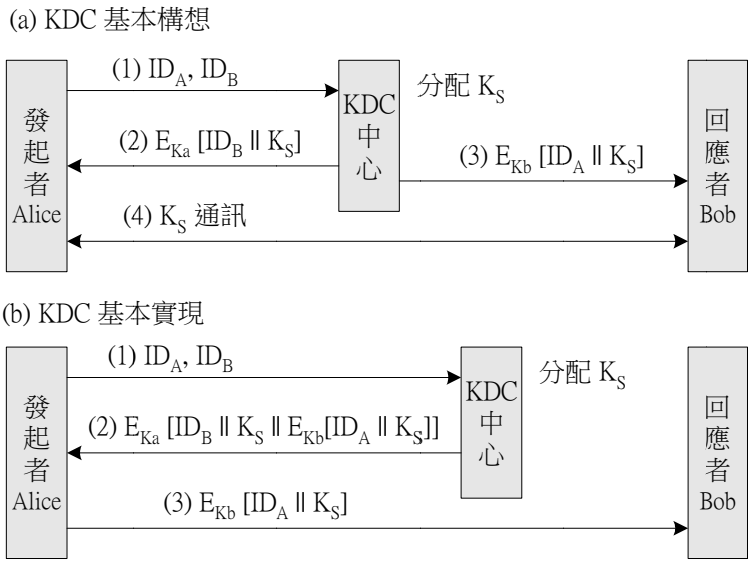


圖 14-1 KDC 的基本構想與實現

比較可行的實現方式，是由發起者攜帶會議鑰匙給回應者，如圖 14-1 (b) 所示。當 KDC 收到發起者請求之後，祇傳遞會議鑰匙給發起者 (訊號 (2)) ，但其中包含一把通往回應者 (如 Bob) 的『門票』(Ticket) ，其格式如下：

$$Ticket = E_{Kb} [ID_A \| K_S]$$

然而，此門票是利用回應者的共享密鑰加密著，其中包含著發起者的身份識別與會議鑰匙，當然只有回應者可以解密得到其中訊息。發起者將此門票傳送給回應者之後，兩者便可以利用該會議鑰匙 (K_S) 來通訊。由此可以看出，發起者與回應者之間一定採用相同的會議鑰匙通訊，而且回應者也不至於使用舊的鑰匙來通訊。

圖 14-1 (b) 認證協定還是有缺陷的地方，如果攻擊者可以攔截到門票(訊號 (3))，甚至不需要破解它，只要連續向回應者發出重複攻擊，也可以得到利用會議鑰匙 (K_S) 加密的密文。如此一來，攻擊者只要稍加努力，便可得到許多明文與密文配對，欲找出

會議鑰匙也不是困難的事；或是攻擊者偽裝成發起者身份，發送門票給回應者，回應者也很難辨別對方是冒充的。因此，並非只利用 KDC 來分配鑰匙就可以，通訊雙方還是需要確認所持有的鑰匙是否相同，如此一來，便牽涉到『相互認證』（Mutual Authentication）的問題。

14-2 Needham-Schroeder 認證協定

14-2-1 N-S 協定簡介

Needham 與 Schroeder 於 1978 年提出一種『多重盤問與回應』的認證協定 [110]，它不但可以避免重播攻擊，也可以解決相互認證的問題。此認證協定大多承襲圖 14-1 (b) 的運作方式，但它利用隨機產生的亂數（Nonce，64 bits）標示每一個通訊連線，並作為測試對方所持有的會議鑰匙（盤問與回應）使用。圖 14-2 為 Needham-Schroeder 認證協定的運作程序，其執行步驟如下：（以訊號發送次序說明）

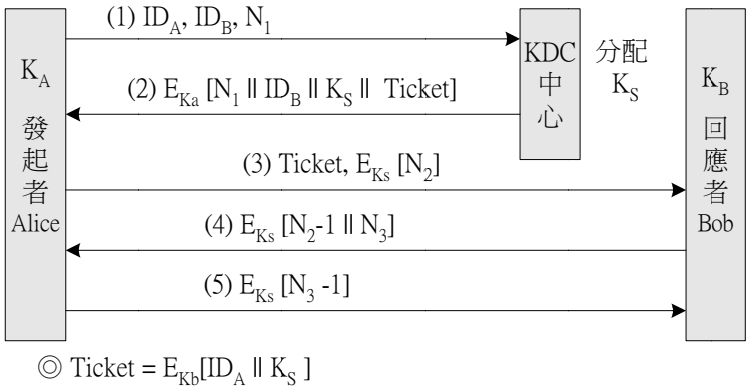


圖 14-2 Needham-Schroeder 認證協定

1. Alice 想要和 Bob 作安全性通訊，Alice 送出自己與對方的身份識別 ID_A 與 ID_B 給 KDC 中心，其中加入了盤問亂數 N_1，為該連線的識別。

2. KDC 利用 Alice 的共享密鑰，傳送會議鑰匙（Ks）與通往 Bob 的門票（Ticket = E_{Kb} [ID_A || K_S]）。訊息中包含著原來 Alice 所產生的亂數 N_1，如此一來，Alice 便可由此亂數來判別 KDC 所回應的是那一個請求（盤問與回應功能）。

3. Alice 得到通往 Bob 的門票（E_{Kb} [ID_A || K_S]）之後，便可以利用它向 Bob 要求通訊，並且以另一個亂數（E_{Ks} [N_2]）來盤問 Bob，而該亂數係利用雙方的會議鑰匙加密的。

4. Bob 利用自己的共享密鑰解開門票的內容，從中瞭解要求通訊的對象是 Alice（ID_A），並得到雙方的會議鑰匙（K_S），同時利用會議鑰匙解開 Alice 的盤問亂數（N_2），然後 Bob 將亂數 N_2 減一（N_2-1），並產生另一個亂數 N_3，使用會議鑰匙加密後，一併回傳給 Alice，其中 N_2-1 表示對 Alice 的回應，而 N_3 則是作為再盤問 Alice 使用。

5. Alice 收到 N_2-1 之後，可以確定對方所持有的會議鑰匙與自己的相同(Bob 沒錯)，一樣將亂數 N_3 減一（N_3-1）作為回應（利用 K_S 加密）。

6. Bob 收到 Alice 的回應（K_S[N_3-1]）之後，便可以確定對方所持有的會議鑰匙與自己的相同，同時確定對方身份（是 Alice 沒錯）。

7. 雙方都得到了會議鑰匙，也彼此確認所持有的鑰匙；接下來，就可以利用該會議鑰匙進行通訊。

由以上的敘述可以發現，訊號 (1) 與 (2) 是 KDC 分配鑰匙的運作程序；然而，訊號 (3) 到 (5) 是執行相互認證的程序。此協定最大的特點是，每一個訊號都在執行『盤問與回應』的機制，因此，又稱為『多重盤問與回應』協定。

Needham-Schroeder 認證協定的弱點，還是在發起者傳送門票給回應者的時候(訊號 (3))。假設攻擊者攔截到門票（訊號 (3)），它可以重複傳送此訊號給回應者（重播攻擊法）；縱使該門票內含舊的鑰匙，回應者並無法分辨到底哪一把才是新的鑰匙，如此一來，攻擊者就可以阻斷回應者與發起者之間的通訊。

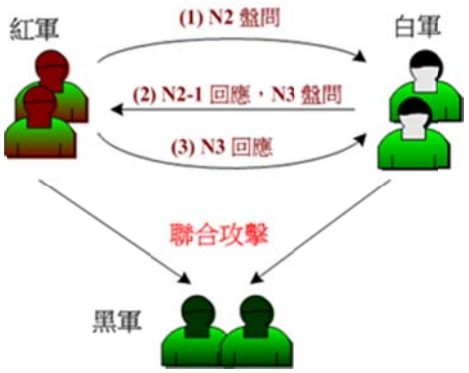


圖 14-3-0 三向式詢問法

14-2-2 N-S 協定- 加入時間戳記

Needham-Schroeder 認證協定的問題是出現在遭受重播攻擊時，回應者無法分辨出所收到的門票是否過期，也就是無法辨別新舊鑰匙。有一個簡單的構想，是在門票上加入時間戳記 [49]，當回應者收到門票時，便可利用時間戳記來判斷該門票是否過期，甚至由時間戳記可以看出所傳送的鑰匙是新的或舊的。加入時間戳記的 Needham-Schroeder 協定之運作程序如圖 14-3 所示，門票為：

$$Ticket = E_{Kb}[ID_A \| K_S \| T]$$

其中 T 是時間戳記。

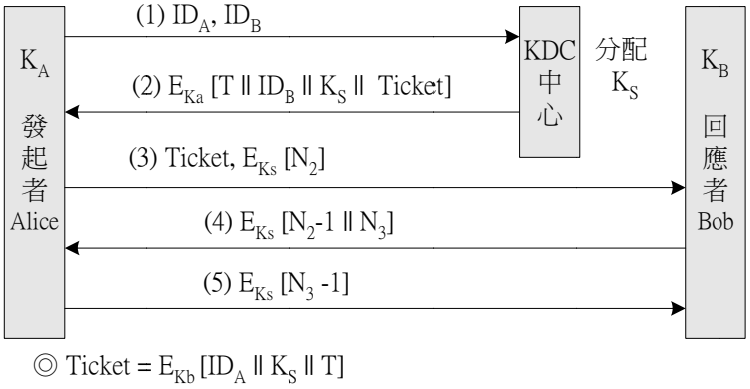


圖 14-3 加入時間戳記的 Needham-Schroeder 認證協定

雖然加入時間戳記可以解決一部份的問題，但亦衍生新的問題產生。為使時間戳記保持有效，通常參與通訊的設備之間的時序必須保持同步。如圖 14-3 中，發起者、KDC 中心與回應者之間的時序必須保持同步，縱然無法達到完全同步，最起碼保持在可容許範圍內。就封閉型的分散式系統而言，尚可達成，至於開放式的分散式系統，可就困難重重。

14-2-3 擴展型 N-S 協定

我們回到原來 Needham-Schroeder 協定的問題，關鍵在於回應者遭受重播攻擊時，無法辨別新舊門票。究其原因是回應者並不知道有人欲與他進行通訊，所以一旦回應者收到門票時，便立即與對方通訊。如果發起者能事先通知他，讓他有點準備，或許就能

辨別新舊鑰匙。擴展型 Needham-Schroeder 認證協定，就是利用這種觀念發展出來的[110]，其運作程序如圖 14-4 所示。

我們可以比較圖 14-4 與圖 14-2 兩者之間的差異，圖 14-4 中訊號 (1) 與 (2) 是 Alice 事先通知 Bob，並由 Bob 得到一個亂數 N_B（由 Bob 的主密鑰加密），而且 KDC 也將該亂數植入門票內（訊號 (4)），如下：

$$Ticket = E_{Kb}\,[ID_A \,||\, K_S \,||\, N_B]$$

當 Alice 利用此門票向 Bob 要求通訊時（訊號 (5)），Bob 便能由亂數 N_B 中知曉到底是哪一張門票。假設，攻擊者攔截到門票（訊號 (5)），並重複發送給 Bob，Bob 由門票內的 N_B 便可以分辨出新舊會議鑰匙。

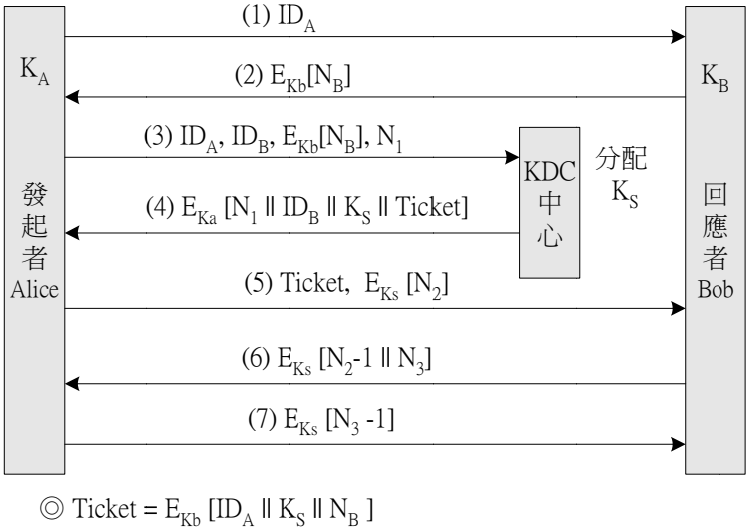


圖 14-4 擴展型 Needham-Schroeder 認證協定

14-3 公開鑰匙認證協定

14-3-1 公開鑰匙認證協定簡介

上述所介紹的認證協定（如 Needham-Schroeder 協定）大多是使用於封閉型的分散式系統；在這些協定之中，使用者必須事先在 KDC 中心建立有帳戶，並且雙方利用使用者密碼所產生的『共享密鑰』（或稱使用者的主密鑰）來達成相互認證。但在開放式分散式系統中，使用者可能來自四面八方，且不可能事先在 KDC 中心建立帳戶，當然也不會有共享密鑰；在這種情況之下，如何來達成認證問題？以及如何分配會議鑰匙

（Session Key）？關於公開鑰匙的認證協定，本書第四章已詳細介紹過，這裡僅介紹其鑰匙分配方法。

簡單的說，與陌生人之間的認證則必須採用公開鑰匙系統，但公開鑰匙系統的加密和解密時間大約是秘密鑰匙系統的 100 到 1000 倍（鑰匙較長），所以利用公開鑰匙系統加密傳遞資料，效率將是非常的低。因此，必須結合兩套系統，在身份認證方面採用公開鑰匙系統，而在資料傳遞方面則使用秘密鑰匙（會議鑰匙）系統，成為『混合型密碼系統』（Hybrid Cryptosystem）。

14-3-2 AS 分配公開鑰匙

在系統之中建立一個『認證伺服器』(Authentication Server, AS)是最簡單的做法，由該伺服器負責認證客戶的公開鑰匙（或數位憑證）。AS 伺服器必須是一個較堅固的系統，至少不容易讓攻擊者入侵修改內部儲存的公開鑰匙，並且參與通訊者都能相信它。當然參與通訊者都必須持有自己的鑰匙配對，並且能夠安全地將公開鑰匙交與 AS 伺服器保管。圖 14-5 為簡單的認證程序，為了預防重播攻擊、或避免攻擊者使用已破解的會議鑰匙，因此，加入時間戳記 [49]，其中發起者(假設為 Alice)的鑰匙配對是 $\{K_{Ra}, K_{Ua}\}$(前者為私有鑰匙，後者為公開鑰匙)，AS 為 $\{K_{Rs}, K_{Us}\}$，回應者(如 Bob)是$\{K_{Rb}, K_{Ub}\}$。

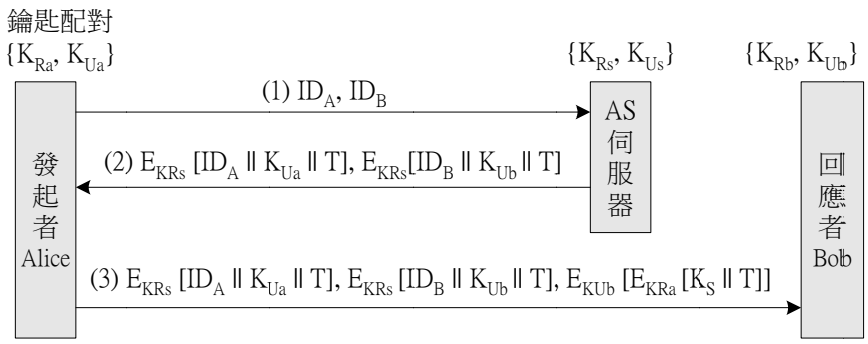


圖 14-5 簡單的公開鑰匙認證

圖 14-5 的基本構想是，AS 分配可信賴的公開鑰匙，然而發起者確認對方公開鑰匙無誤之後，再傳送會議鑰匙給回應者；既然會議鑰匙是由發起者直接傳送給回應者，就沒有會議鑰匙洩漏的問題，其運作說明如下：

◈ 步驟 1：Alice 向 AS 伺服器請求 Bob 的公開鑰匙，其中以 ID_A 與 ID_B 表示 Alice 與 Bob 的身份識別；訊號 (1)。

◈ 步驟 2：AS 伺服器利用自己的私有鑰匙（K_{Rs}）回應 Alice 與 Bob 的公開鑰匙（K_{Ua} 與 K_{Ub}）、與身份識別（ID_A 與 ID_B）給 Alice，其中順便加入時間戳記（T）；訊號 (2)。即是：

Alice 公開鑰匙：$E_{KRs}[ID_A \| K_{Ua} \| T]$

Bob 公開鑰匙：$E_{KRs}[ID_B \| K_{Ub} \| T]$

◈ 步驟 3：Alice 利用 AS 的公開鑰匙（K_{Us}）解開 AS 伺服器所回傳的訊息，並比較 ID_A 與 K_{Ua} 是否屬於自己的，如果正確的話，表示 AS 確實回應自己的請求。另外，也比較 ID_B 是否是自己想要通訊的對象，如果是的話，則表示已取得對方的公開鑰匙（K_{Ub}）。上述都正常的話，則 Alice 產一個通訊鑰匙（K_S），並利用自己的私有鑰匙與對方的公開鑰匙加密，連帶著 AS 所回傳的訊息，一起發送給 Bob，其中每一筆訊號亦加入時間戳記（T）；訊號 (3)。會議鑰匙為：

Alice 產生會議鑰匙：$E_{KUb}[E_{KRa}[K_S \| T]]$

◈ 步驟 4：Bob 利用 AS 的公開鑰匙解開前面兩個訊息，可以得到對方和自己的身份識別與公開鑰匙，並比對該身份識別（ID_B）是否屬於自己。再利用自己的私有鑰匙（K_{Rb}）與對方的公開鑰匙（K_{Ub}）解開會議鑰匙（K_S）的加密，並比較時間戳記（T）是否在可容許的範圍內。

◈ 步驟 5：之後發起者與回應者之間便利用該會議鑰匙（K_S）來通訊。

圖 14-25 認證協定還是出現時序同步的老問題，然而，因為在開放性的分散式處理當中，欲達成時序同步並不容易，下面是較完整的認證協定介紹。

14-3-3 KDC 分配會議鑰匙

圖 14-6 為比較完整的公開鑰匙認證協定 [145]，主要是利用亂數（Nonce）來取代時間戳記，並且由 KDC 分配會議鑰匙，因此 KDC 同時肩負公開鑰匙的分配與會議鑰匙的產生。說明如下：

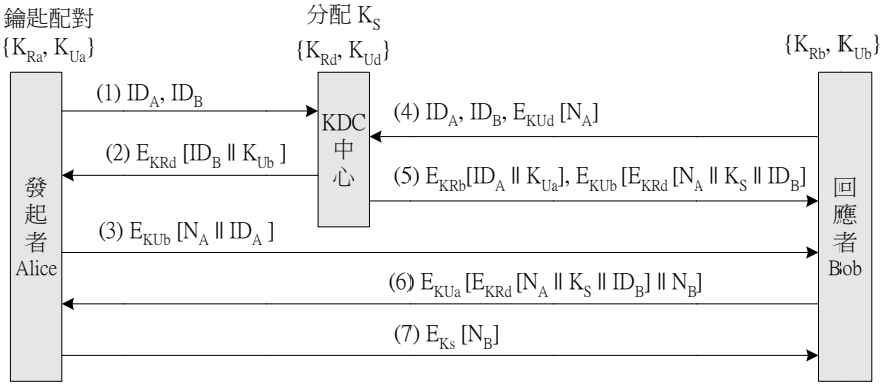


圖 14-6 公開鑰匙認證協定

◈ 步驟 1：發起者（假設 Alice）向 KDC 要求通往回應者（假設 Bob）的公開鑰匙，並傳送兩方的身分證明（或數位憑證，ID_A 與 ID_B）給 KDC；訊號 (1)。

◈ 步驟 2：KDC 收到請求之後，利用自己的私有鑰匙（K_{Rd}）傳送 Bob 的憑證與公開鑰匙鑰匙給 Alice；訊號 (2)。其中：

Bob 公開鑰匙：E_{KRd} [ID_B || K_{Ub}]

◈ 步驟 3：Alice 用 KDC 的公開鑰匙向訊號 (2) 解密，取得 Bob 的憑證與公開鑰匙，再利用 Bob 的公開鑰匙傳送自己的憑證與一個亂數 N_A 給 Bob；亂數 N_A 是作為此次通訊的識別，並作為防禦重播攻擊使用；訊號 (3)。其中：

Alice 向 Bob 表示身分：E_{KUb} [N_A || ID_A]

◈ 步驟 4：Bob 收到 Alice 的身份識別之後，必須確認 Alice 的身份是否正確。因此，Bob 也傳送兩方的憑證（ID_A 與 ID_B），並希望索取 Alice 的公開鑰匙；另外，也利用 KDC 的公開鑰匙傳送 Alice 所產生的亂數 N_A，希望 KDC 能將此亂數加入雙方通訊鑰匙的識別之中；訊號 (4)。

◈ 步驟 5：KDC 收到訊號 (4) 請求之後，傳送兩個訊息給 Bob。一者是利用 KDC 私有鑰匙加密之 Alice 的憑證與公開鑰匙（E_{KRd}[ID_A || K_{Ua}]）；另一者是利用 Bob 公開鑰匙與 KDC 私有鑰匙加密的會議鑰匙資料，該資料包含 Alice 所產生的亂數 N_A、會議鑰匙（K_S）與 Alice 的憑證（ID_A）；如此，Bob 便取得了 Alice 的公開鑰匙、與通往 Alice 的會議鑰匙；訊號 (5)。其中：

Alice 公開鑰匙：E_{KRd} [ID_A || K_{Ua}]

KDC 產生會議鑰匙：$E_{KUb}[E_{KRd}[N_A \| K_S \| ID_B]]$

◈ 步驟 6：雖然 Bob 取得了會議鑰匙，然而它也必須採用一種可以讓 Alice 相信的方法，將會議鑰匙傳送給他。因此，它將原來 KDC 送過來的訊息（$E_{KRd}[N_A \| K_S \| ID_B]$）再加上一個亂數 N_B，以 Alice 的公開鑰匙（K_{Ua}）加密後，傳送給 Alice；訊號 (6)。其中：

Bob 傳送會議鑰匙給 Alice：$E_{KUa}[E_{KRd}[N_A \| K_S \| ID_B] \| N_B]$

◈ 步驟 7：Alice 利用自己的私有鑰匙與 KDC 的公開鑰匙解開訊號 (6)，證明對方的身份是 Bob（ID_B），並取得會議鑰匙（K_S）。Alice 再利用會議鑰匙傳送 N_B 給 Bob（訊號 (7)）；接下來，Bob 利用會議鑰匙解開訊號 (7)，並確認 N_B 是自己所產生的亂數，如此可確定雙方所持有的會議鑰匙是相同的。

我們可以由 Alice 的觀點來看，當它利用 Bob 的公開鑰匙（K_{Ub}）傳送身份憑證 ID_A 與亂數 N_A（訊號 (3)），再由 Bob 方面收到訊號 (6)；接著，它將比較兩者 N_A 是否相同，如果相同，則可確認是哪一個請求訊號所回應的。如此一來，雖然攻擊者攔截到訊號 (6)，再重播給 Alice，Alice 也可以分辨出它是舊的會議鑰匙。

由以上的介紹，我們很難說出哪一種認證協定是安全的，各種認證協定都是使用之後發現缺點，再不斷的改進。但可以確定的是安全性越高的認證協定，它的運作程序越複雜，至於應該採用哪一層次的認證協定，那就見仁見智了。接下來，將介紹使用非常普遍，安全性又高，但運作程序頗複雜的 Kerberos 協定。

14-4 Kerberos V4 認證系統

14-4-1 Kerberos 認證系統簡介

『Kerberos 認證系統』（Kerberos Authentication System）是由麻省理工學院（M. I. T）發展出來的，其命名係來自希臘神話中看守地獄入口那條三頭狗 Kerberos。設計 Kerberos 最主要的目的是要讓使用者能安全地存取網路上資源，也是目前網路使用最廣泛的技術。到目前為止，Kerberos 較常用的有兩個版本，版本 4（V4）[26, 49, 111] 目

前還被廣泛使用中，但僅能使用於 TCP/IP 網路；版本 5（V5）已作了許多修正，使它更適合其他網路環境，並已制定成 RFC 1510 標準規範。

基本上，Kerberos 祇採用秘密鑰匙系統認證用戶端，所以每一個使用者都必須在 Kerberos 伺服器上建立帳號名稱，並利用使用者密碼所建立的『主密鑰』，來確認使用者與伺服器之間的身份。因此屬於較封閉的分散式系統應用，但目前已有許多應用系統，將公開鑰匙系統植入 Kerberos 之中，但僅限於客戶端身份認證而已。從認證技術而言，Kerberos 主要延伸許多 Needham-Schroeder 協定的技術，並加入時間戳記作為防止重播攻擊。正因如此，在 Kerberos 系統上，參與認證的工作站或伺服器之間都必須保持時序的同步。

之前我們所介紹的認證程序，大多侷限於使用者（透過工作站）與伺服器之間身份的認證問題、以及會議鑰匙的分配工作。然而，在 Kerberos 認證協定上並沒有刻意劃分使用者(人)或機器(伺服器或工作站)，它對每一個參與認證的實體(人或機器)都一視同仁，稱之為『主角』（Principal），這種觀念通常比較適合分散式處理。

我們可意識得到，Kerberos 可能是未來封閉型分散式系統(譬如，電子化辦公室)認證的主流；至於開放型的分散式系統（如 Internet 網路應用），則較傾向於 PKI 認證系統（利用數位憑證）；但也未必如此，目前許多封閉式系統（如 Windows 2000）已將 Kerberos 與 PKI 系統整合使用。基本上，Kerberos 的認證程序算是複雜的，本章之前介紹了許多認證協定的運作程序，主要為 Kerberos 鋪路；本節以介紹 Kerberos V4 為主，有了第四版本的概念之後，下一節再介紹第五版本可就容易多了。

14-4-2 Kerberos 動機

無論採用何種認證協定，主要的目標是確認通訊雙方的身份、與分配雙方通訊所需的會議鑰匙。我們採用了極複雜的運作程序，是為了要防止攻擊者從中破壞正常交易，至於如何達成上述兩個目標，其實到目前為止，也很難找出十全十美的方法。首先將攻擊者的破壞技巧歸類如下：

◈ 也許攻擊者合法取得某一工作站權限，再利用此工作站冒充另一個使用者，取得其他工作站的非法權限。

◈ 也許攻擊者會更改工作站的網路位址，來偽裝成另一個工作站，再發出偽裝訊息。

◈ 也許攻擊者由網路上竊聽到相關訊息，再重送攻擊或入侵相關伺服器，或是破壞該伺服器的正常運作。

之前所介紹的幾種認證協定，大多著重於防止重播攻擊。假設，先撇開重播攻擊的問題，我們將圖 14-1 (b) 的基本認證程序加入了網路位址，便如圖 14-7 所示。其中 AD_A 表示客戶工作站的位址，我們來討論一下可能出現的問題：

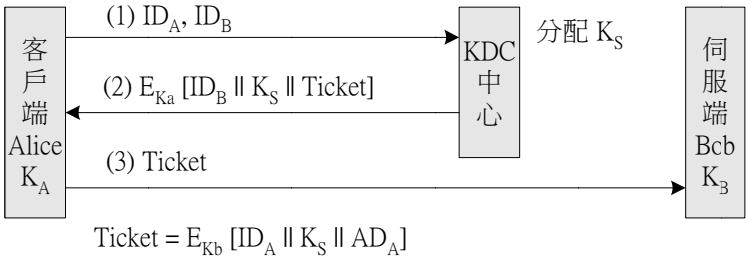


圖 14-7 附加網路位址的門票

◈ 客戶需要任何一部伺服器資源時，都需要向 KDC 申請門票；每次申請門票時，客戶都需要輸入密碼確認身份（產生主密鑰）。在目前分散式處理環境之下，每一客戶同時存取多個伺服器是非常普遍的現象；如此一來，增加客戶端輸入密碼的次數，也增加了暴露密碼的危機。最好的構想是，在同一個應用系統中，客戶只要輸入一次密碼即可。

◈ 雖然在門票中加入了客戶工作站的網路位址，但當攻擊者攔截到門票之後（訊號 (3)），還是可以等待到該客戶登出之後，再偽裝成它的網路位址；如此一來，偽裝工作站傳送門票給伺服端，伺服端也相信對方的身份了。因此，僅加入網路位址並無法完全避免他人的攻擊；或許在門票上加入使用期限，可以減少被冒充的機率。

第二個問題較簡單，我們先來討論第一個問題。記得我們在 14-5 節介紹過（如圖 14-20 所示），KDC 的概念是將客戶認證（與建立會議鑰匙）的工作分配給 KDC 伺服器管理；如果確定身份無誤之後，便可以發給客戶通往伺服器的門票，然而客戶是否有權利存取某一伺服器，仍需仰賴原伺服器自行管理（如建立 ACL 表）。如果我們另外建立一個伺服器，作為過濾客戶是否可以使用某一個伺服器的權利，此伺服器稱為『門票核准伺服器』（Ticket-Granting Server, TGS）；如果該客戶有權利存取某一伺服器，

則 TGS 便發給予通往該伺服器的門票。至於客戶存取該伺服器的權限如何，還是保存在原伺服器上管理。如此一來，原來 KDC 的工作便僅限於客戶身份的認證而已，因此，又稱為『認證伺服器』（Authentication Server, AS）。也就是說，我們將 KDC 的工作分配給 AS 與 TGS 兩個伺服器來承擔，如此一來，便可以解決客戶只要輸入一次密碼的問題。

14-4-3 Kerberos 基本構想

圖 14-8 為 Kerberos 的基本構想，運作程序的簡單程序是：客戶利用自己的主密鑰向 AS 伺服器申請到一張通往 TGS 伺服器的門票（$Ticket_{TGS}$），再利用此門票向 TGS 伺服器申請一張通往某伺服器的門票（$Ticket_B$），客戶端再利用伺服器的門票向該伺服器要求服務。兩張門票分別為：

TGS 門票：$Ticket_{TGS} = E_{Ktgs} [ID_A \| AD_A \| ID_{TGS} \| TS_1 \| Lifetime_1]$

伺服器門票：$Ticket_B = E_{Kb} [ID_A \| AD_A \| ID_B \| TS_2 \| Lifetime_2]$

其中 ID 與 AD 分別為身分識別與網路位址，TS 是時間戳記，Lifetime 為有效時間。接下來，我們來討論利用兩個伺服器的認證有何特性：（如圖 14-8 所示）

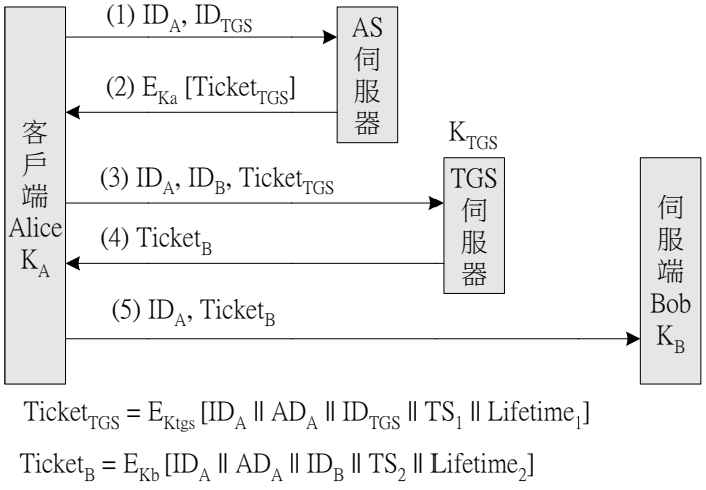


圖 14-8 Kerberos 的基本構想

◈ 主密鑰分配：AS 伺服器除了必須擁有客戶的主密鑰之外，還必須擁有 TGS 的主密鑰；另外，TGS 伺服器也需要擁有所有伺服器的主密鑰。這就是 Kerberos 將所有參與者都稱為 Principal 的主要原因。

◈ 客戶密碼只要輸入一次：客戶端取得通往 TGS 的門票（$Ticket_{TGS}$）之後，在該票的有效期限之內，都可以請求服務，而不需要再輸入密碼來索取門票。

◈ 防禦偽裝攻擊：門票（$Ticket_{TGS}$ 與 $Ticket_B$）上有登錄該票的使用者識別（ID）、工作站位址（AD）、時間戳記（TS）與有效期間（Lifetime）。攻擊者攔截到門票之後，不易在在有效期內偽裝成合法客戶。

◈ 防止重播攻擊：門票有註明時間戳記（T），當攻擊者重播門票時，接收端可以利用時間戳記辨別門票的新舊。

門票都利用接收端的主密鑰加密著，譬如，客戶傳送給 TGS 門票便利用 TGS 的主密鑰（K_{TGS}）加密著；另外。客戶傳送給伺服器的門票，也利用該伺服器（假設 Bob）的主密鑰（K_B）加密著。客戶與攻擊者都無法窺視到門票的內容，當然無法冒充或竄改它。接下來的問題是有效期間要多長？如果太短的話，當門票逾時之後，客戶端必須再重新輸入密碼取得新的門票。但如果有效期間過長，客戶登出之後，攻擊者可以冒充它的帳戶來使用門票，之間取捨難定，不過一般大多定在 8 個小時左右。

我們瞭解圖 14-8 的基本構想之後，大略可以完成身份認證的問題。我們再將會議鑰匙分配的程序加入圖 14-8 中，並規範標準的加密演算法與門票的格式，便能制定出一個完整的認證協定，這就是建構 Kerberos 的基本構想。

14-4-4 Kerberos V4 認證程序

Kerberos 將參與工作的成員稱呼為『主角』（Principal），每個成員都擁有自己的『主密鑰』（Master Secret）。但基本上，它還是屬於主從式（Client/Server）架構的應用系統，其成員主要由下列四種實體（使用者或設備）所扮演而成：

1. 客戶端（Client）：在網路欲存取資源的使用者。
2. 服務伺服器（Service Server）：在網路上提供資源服務者。對每一個使用者要求服務時，必須能夠確定是當事人而非他人冒充，傳遞資源時必須作相當的加密動作，避免他人偷竊。這個加密用的秘密鑰匙只有和使用者之間擁有（由 TGS 伺服器提供），而且用完一次便拋棄，下次連線時再建立新鑰匙。

3. 認證伺服器（Authentication Server, AS）：相當於『鑰匙分配中心』（KDC）。管理每一個使用者的主密鑰（或稱共享密鑰）。欲加入的使用者都必須向 AS 伺服器申請帳戶並取得主密鑰。網路上任何一個使用者登入系統時，都會向認證中心取得通行證（Pass Book 或 TGT 票），有了通行證便可以在網路上索取所要的資源。

4. 門票核准伺服器（Ticket-Granting Server, TGS）：TGS 伺服器管理網路上所有服務伺服器，並紀錄所有服務伺服器的秘密鑰匙。當有新的伺服器加入或退出時，都必須向 TGS 伺服器申請。而且某一秘密鑰匙也只有 TGS 伺服器和該服務伺服器兩者所擁有，因此 TGS 伺服器必須管理網路上所有伺服器的秘密鑰匙。使用者如要存取服務伺服器上資源時，必須向 TGS 提出它的身份證明，即 TGT 門票。TGT 門票是經過 TGS 伺服器的秘密鑰匙加密，因此只有 TGS 伺服器能觀察門票的內容，別人無法仿冒。TGS 伺服器驗證完使用者的通行證後，再發給使用者有關使用者本身和所欲要求的服務伺服器的『服務門票』（Service Ticket, ST）。有了 ST 門票之後，使用者才可以到服務伺服器上存取資源，ST 門票同時包含了會議鑰匙。

就門票的類別可區分為下列兩大類：

◈ 門票核准票（Ticket-Granting Ticket, TGT）：此票是由 AS 伺服器發給客戶端的身分證明使用；使用者可以持此票向 TGS 伺服器申請通往某一伺服器的請求。

◈ 服務門票（Service Ticket, ST）：當使用者出示 TGT 門票，向 TGS 伺服器要求前往某一伺服器；如果 TGS 伺服器同意其要求時，則發給所要求伺服器的 ST 門票給使用者，使用者持此票即可要求該伺服器提供服務。

瞭解 Kerberos 的元件之後，再來探討 Kerberos 認證協定可就容易多了。它的運作程序如圖 14-9 所示，我們將它分為登錄、取票、要求服務等三個步驟來討論，如下：

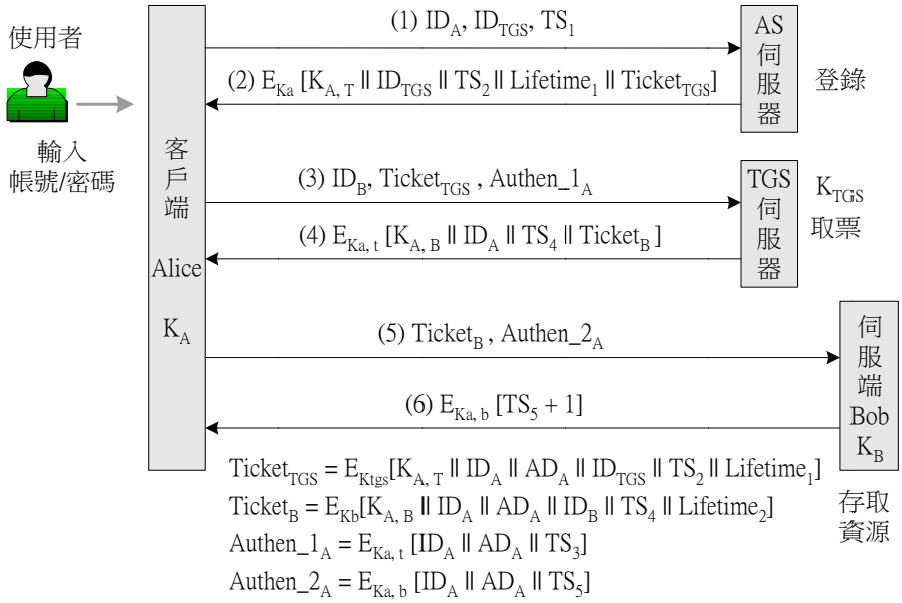


圖 14-9 Kerberos V4 協定之運作程序

【（A）登錄】

當使用者（假設 Alice）想要進入系統參與工作時，它必須進入 Kerberos 所管轄的範圍內，並由 Kerberos 認證身份。首先，Alice 傳送本身識別碼（或憑證、ID_A）與希望取得 TGS 伺服器門票的識別碼（ID_{TGS}）給 AS 伺服器，其中已加入預防重播攻擊的時間戳記（TS_1）（訊號 (1)）。AS 確認 Alice 的身份識別無誤後，便利用 Alice 主密鑰傳送通往 TGS 的 TGT 門票（$Ticket_{TGS}$）、以及 Alice 與 TGS 伺服器之間的會議鑰匙（$K_{A,T}$）給 Alice（訊號 (2)），訊號格式如下：

$$E_{Ka}\ [K_{A,T} \| ID_{TGS} \| TS_2 \| Lifetime_1 \| Ticket_{TGS}]$$

TGT 門票格式如下：

$$Ticket_{TGS} = E_{Ktgs}\ [K_{A,T} \| ID_A \| AD_A \| ID_{TGS} \| TS_2 \| Lifetime_1]$$

其中 $Lifetime_1$ 表示該門票的有效期限，$K_{A,T}$ 是 AS 伺服器分配給 Alice 與 TGS 伺服器之間的會議鑰匙，AD_A 為 Alice 工作站的網路位址（IP 位址），AS 伺服器是由 Alice 所傳送過來的 IP 封包中取得。

當訊號 (2) 到達 Alice 工作站時，工作站會要求 Alice 輸入密碼，並由輸入的密碼產生 Alice 的主密鑰（K_A），再利用主密鑰解開訊號 (2) 的加密，同時取得相關訊息。由此可見，TGT 門票是利用 Alice 的主密鑰加密著，他人不易觀察裡面的內容。

【（B）取票】

Alice 經由 AS 確認身份，並取得通往 TGS 的 TGT 門票之後，它便成為 Kerberos 系統下的成員之一。在該門票的有效期間內，它都可以向 TGS 伺服器要求其它伺服器的門票，其要求步驟如下：首先 Alice 傳送自己的簽署碼（Authentication，$Authen\_1_A$）、身份識別（ID_B）與 TGT 門票（$Ticket_{TGS}$）給 TGS 伺服器，其中身份識別（ID_B）表示欲前往的服務伺服器（假設 Bob）（訊號 (3)），簽署碼表示 Alice 自己簽署的身分證明，格式如下：

$$Authen\_1_A = E_{Ka,\,t}\,[ID_A \,||\, AD_A \,||\, TS_3]$$

由上可知簽署碼是使用 Alice 與 TGS 的會議鑰匙（$K_{A,\,T}$）加密，其中包含 Alice 的身份識別（ID_A、AD_A）與時間戳記（TS_3）。

TGS 伺服器檢視 $Ticket_{TGS}$ 門票中有效期限是否過期，如在有效期間內，則取出會議鑰匙（$K_{A,\,T}$），並解開 Alice 的認證碼（$Authen\_1_A$），再核對認證碼與門票內有關 Alice 的身份識別是否正常。如果都正常的話，便發給 Alice 通往服務伺服器（Bob）的 ST 門票（訊號 (4)），ST 門票格式如下：

$$Ticket_B = E_{Kb}\,[K_{A,\,B} \,||\, ID_A \,||\, AD_A \,||\, ID_B \,||\, TS_4 \,||\, Lifetime_2\,]$$

由上可知 ST 門票係利用 Bob 的主密鑰加密，他人無法窺視其內容，更何況要去竄改它。ST 門票中也包含著 TGS 發給 Alice 與 Bob 之間的會議鑰匙（$K_{A,\,B}$），並包含著 Alice 與 Bob 的身份識別、該門票的有效期限（$Lifetime_2$）與時間戳記（TS_4）。

【（C）要求服務】

Alice 由 TGS 伺服器上取得通往 Bob 的 ST 門票之後，在該門票的有效期間內都可以向 Bob 要求服務，要求服務的步驟如下：首先 Alice 向 Bob 出示門票（$Ticket_B$）與自己的認證碼（$Authen\_2_A$）（訊號 (5)），認證碼的格式如下：

$$Authen\_2_A = E_{Ka,\,b}\,[ID_A \,||\, AD_A \,||\, TS_5]$$

認證碼也是使用 Alice 與 Bob 的會議鑰匙（$K_{A,\,B}$）加密。Bob 收到訊號 (5) 之後，再利用自己的主密鑰解開門票的加密，並核對是否在有效期間內，如果該門票尚未逾期，則取出會議鑰匙（$K_{A,\,B}$），並利用該鑰匙解開認證碼的加密。接下來，Bob 比較 ST 門

票內所註明的身份識別是否與認證碼相同，如果相同的話，則可確定發送該門票者確實是 Alice 沒錯。但 Bob 必須確認該會議鑰匙是否與 Alice 共享的，因此，Bob 可利用會議鑰匙回送時間戳記減一的值給 Alice(訊號 (6))，作為達到『相互認證』(Mutual Authentication) 的功能。

由上述的介紹，可以發現每一筆訊號都加入了時間戳記，以防止重播攻擊。門票也註明了有效期限，如此更能減低使用者退出後，攻擊者仿冒的機會。當然攻擊者欲攔截門票來仿冒也有其困難，譬如，攻擊者攔截訊號 (3) (或訊號 (5))，取得門票之後，再偽裝成 Alice 傳送給 TGS 伺服器 (或 Bob)，但它沒 Alice 的認證碼也是枉然。

14-4-5 多重 Kerberos 領域

Kerberos 將每一系統所管轄的範圍稱之為『領域』(Realm)；每一領域內，至少包含一部 AS 伺服器、若干個 TGS 伺服器、以及所管轄的使用者與服務伺服器。基本上，使用者都必須向 AS 伺服器登錄帳戶，才正式成為該領域下的使用者，又稱為『領域使用者』；另一方面，服務伺服器必須向 AS 與 TGS 登錄，領域使用者才可以存取道該伺服器的資源，所有領域之內的成員又通稱為『主角』(Principle)。因此，針對任何一個使用者或伺服器 (或工作站) 都標示為『Realm/Principle』，譬如，『CIS/User_1』，則表示該成員是 CIS 領域下的 User_1 帳戶。

就領域的概念而言，各個成員可構成一個獨立的環境，由 AS 與 TGS 伺服器 (兩者的組合又稱為 Kerberos 伺服器) 來管轄領域內所有資源的分配；一般情況，我們都會將組織單位內工作性相同的人員與設備規劃成一個領域，也大多以組織內的子單位來劃分，如此一來，管理者就可清楚分配資源的使用權限。但如果組織內存在著多個領域，則領域之間如何來達成資源共享的問題，則有賴『多重 Kerberos』(Kerberi) 機制來達成，如圖 14-10 所示。增加了 Kerberi 的關係，我們可將 Kerberos 的特性歸類如下：

◈ 某一領域下使用者都必須向他所屬的 Kerberos 伺服器註冊，並與該伺服器共享一把鑰匙，而此共享鑰匙又稱為該使用者的『主密鑰』(Master Secret)。

◈ 某一領域下的應用伺服器必須向他所屬的 Kerberos 伺服器註冊，並與 Kerberos 伺服之間共享一把秘密鑰匙，即應用伺服器的主密鑰。

◈ 兩個領域之間，Kerberos 伺服器可依照信任關係互相註冊；被信任的 Kerberos 伺服器，與信任一方的 TGS 伺服器（亦是 Kerberos）之間共享一把鑰匙。譬如，Realm_A 領域信任 Realm_B，則表示 Realm_B 的使用者可以存取 Realm_A 領域的資源；因此，Realm_B 的 Kerberos 伺服器與 Realm_A 的 TGS 伺服器之間共享一把鑰匙。

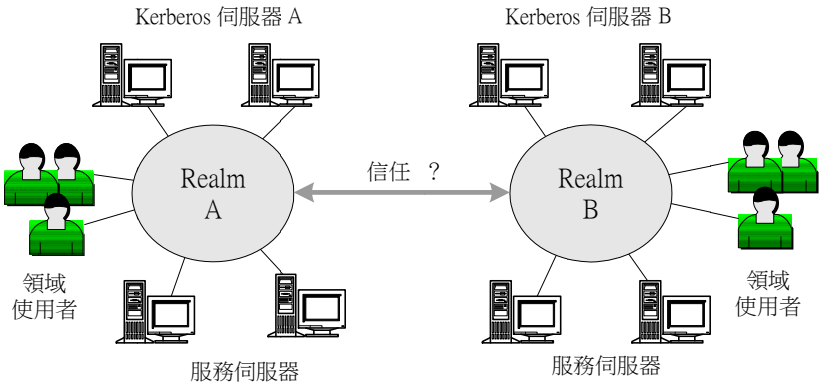


圖 14-10 多重 Kerberos 領域的關係

利用信任關係，使 Kerberos 伺服器與對方 TGS 伺服器之間共享秘密鑰匙，如此一來，領域之間的存取問題就簡單多了。然而，應用伺服器與 Kerberos 之間的共享鑰匙是由 TGS 伺服器負責，因此，與原來領域內存取的不同點，僅在於 TGS 伺服器的分配門票上。我們用圖 14-11 說明領域之間存取的運作程序，並假設領域雙方都互相信任，其運作程序與圖 14-9 大致上相同，僅就跨領域之間的程序說明如下：

◈ 訊號 (3)：客戶端向所屬領域（領域_A）索取通往領域_B（$ID_{TGS\_R\_B}$）的門票。

◈ 訊號 (4)：所屬 TGS 伺服器發給客戶通往領域_B 的門票（$Ticket_{TGS\_R\_B}$）及共享鑰匙（$K_{A,\ TGS\_R\_B}$）。值得注意的是，該門票係利用對方 TGS 伺服器的主密鑰（與領域_A 的共享鑰匙）加密著，圖中並未顯示出來。

◈ 訊號 (5)：客戶向領域_B 的 TGS 伺服器要求服務，傳送門票與所欲存取應用伺服器的識別資料（ID_{B_S}）。

◈ 訊號 (6)：領域_B 的 TGS 伺服器傳送通往應用伺服器的門票（$Ticket_{B\_S}$）給客戶端。

◈ 訊號 (7)：客戶出示門票（$Ticket_{B\_S}$）向應用伺服器要求服務。

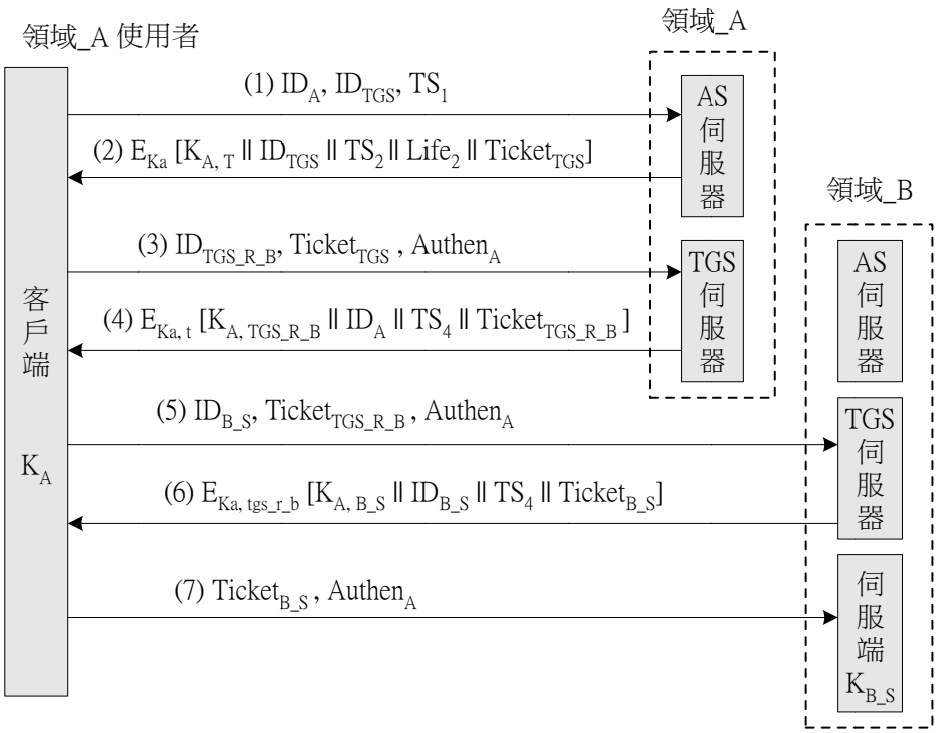


圖 14-11 跨越領域的運作程序

14-5 Kerberos V5 認證系統

隨著 Kerberos 漸漸普遍，系統亦不斷的擴張下，第四版本已漸不符新環境的需求，Kerberos V5 因而被發展出來，且已被發佈為 RFC 1510，許多應用系統也已率先安裝該版本，用來處理客戶與系統之間的認證問題。

14-5-1 Kerberos V5 系統簡介

制定 V5 的主要目的是為了解決 V4 兩個缺點：環境與技術上的缺點 [86]，以下分別說明之：

◈ 加密系統：版本 V4 係使用非標準化的 DES 加密系統，也沒有讓客戶選擇的餘地；既然 DES 加密系統的安全性已漸不符所需，V5 可以讓客戶依安全性的考量，選擇不同的加密系統。

- ◈ 網路協定：版本 V4 僅適用於 TCP/IP 網路上製作，而 V5 允許 Kerberos 安裝在不同的網路系統之中，譬如 OSI 網路。
- ◈ 訊息格式：版本 V4 僅利用一般編碼方式來製作訊息格式，有些電腦系統為了合乎這種格式，而必須修改內部製作程序；然而，V5 利用標準化的 ASN.1 編碼方式來製作訊息，如此便可不受電腦系統的限制。
- ◈ 門票壽命：版本 V4 係利用 8 個位元來表示門票的壽命，而計算單位是 5 分鐘，所以最長時間為 5×2^8 分鐘（大約 21 小時）；在某些應用上，這個時間顯然不足夠。V5 則是依設定的起始和結束時間表示門票的壽命，所以使用者可任意設定。
- ◈ 認證移轉：在版本 V4 上，服務伺服器的門票不能移轉，也就是說，欲要求不同伺服器服務時，需重新認證身份。V5 則具有認證轉移的功能，只需認證一次，便能在不同伺服器要求服務。
- ◈ 領域之間確認：版本 V4 係利用 Kerberos 伺服器之間的信任關係來建立領域之間的認證。而 V5 則是利用樹狀結構的階層從屬關係建立領域之間的確認，此架構較適合與目前『網域名稱系統』（DNS）相結合。

14-5-2 Kerberos V5 運作程序

大致上，版本 V5 的運作程序與 V4 相同，祇不過在訊息方面增加一些參數；V5 的運作程序如圖 14-12 所示（請先參考 V4 的運作程序，圖 14-9），在此先討論增加的那些參數：

- ◈ 領域（Realm）：版本 V4 僅利用身份識別碼（ID）辨別使用者身份，V5 則增加領域的標示，如某一個使用者的識別為『Realm/Principal』（圖中表示為 Realm、ID）。
- ◈ 選項（Options）：為了讓使用者有不同演算法與門票功能的選擇，V5 增加選項欄位供通訊雙方協議。

◈ 時間參數（Times）：作為設定或表示門票的有效期限。發起者可利用此參數來要求所需的有效期限；門票發行者（AS 或 TGS）可將此參數設定於門票之內，以限制門票的使用期間。其中包含三個參數值：

- From：門票起始時間。
- Till：門票到期時間。
- Rtime：門票需重新設定的時間。

◈ 亂數（Nonce）：作為預防重播使用。發起者可隨意設定 Nonce 的值，回應者再依照發起者的值回傳。如此，回應者可以判斷該 Nonce 值的訊息是否回應過，發起者也可瞭解回應者是針對哪一個訊息作回應。譬如，訊息 (1) 中加入了 Nonce，則訊息 (2) 便依照該數值回應給使用者。

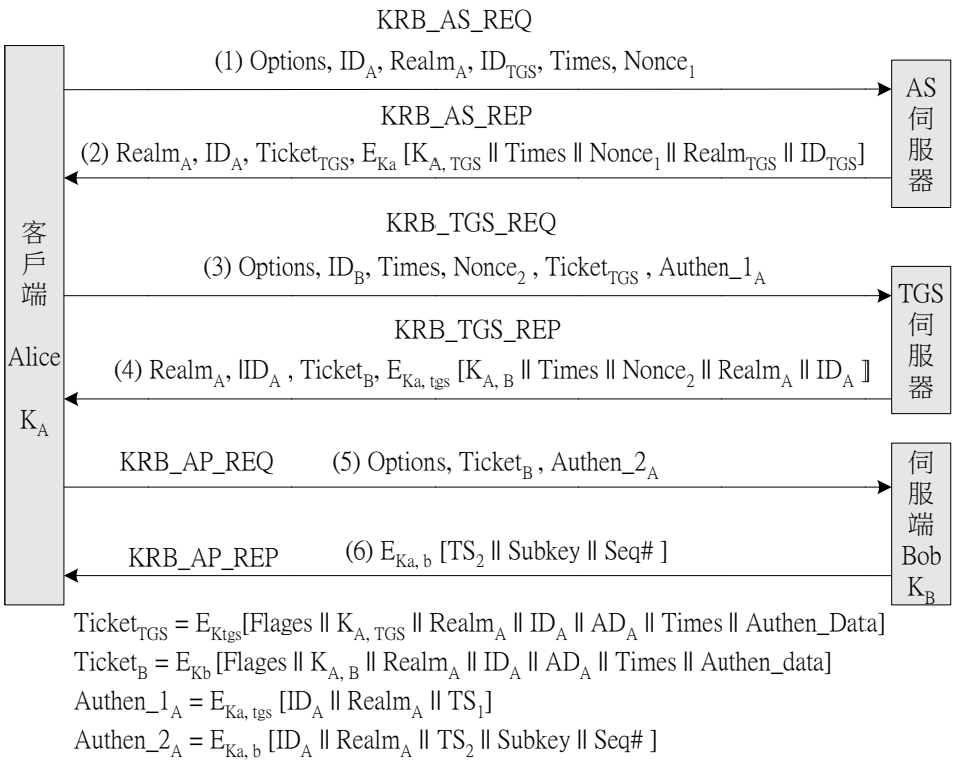


圖 14-12 Kerberos V5 協定之運作程序

版本 V5 將各種訊息都給予一個特殊的名稱（如圖 14-12 所示），並把『認證伺服器』(Authentication Server, AS)所授與通往『門票核准伺服器』(Ticket- Granting Server, TGS）的門票，稱之為『門票核准票』（Ticket-Granting Ticket, TGT），亦將 TGS 所

發行通往應用伺服器的門票，稱之為『服務門票』(Service Ticket, ST)。我們以圖 14-12 來說明其特殊功能：

◈ 認證身份與取得 TGT 門票：此運作程序是使用者與 AS 之間利用 KRB_AS_REQ 與 KRB_AS_REP 兩個訊息來達成。訊息 (1) 中，使用者的要求身份確認 (ID、Realm)、協議門票的功能 (Option)、以及希望所申請 TGT 票中的有效期限 (Times)；在訊息 (2) 中，如果 AS 伺服器同意對方的要求，則給予 TGT 門票，並載入該票的有效期間 (Times)，其中 Options 欄位可協議門票是『代理門票』(Proxy Ticket)、『前向轉送門票』(Forward Ticket)、或一般 TGT 門票。門票格式為：

$$Ticket_{TGS} = E_{Ktgs} [Flages \| K_{A, TGS} \| Realm_A \| ID_A \| Times \| Authen\_Data]$$

◈ 索取服務門票：使用者在 TGT 門票的有效期間內，都可以向 TGS 伺服器要求前往應用伺服器的服務門票 (ST)；該運作程序是由 KRB_TGS_REQ 與 KRB_TGS_RSP 兩個訊息來完成。訊息 (3) 內有包含著協商安全機制的選項 (Options)、有效期間(Times)、以及 TGT 門票。TGT 伺服器由認證碼($Authen\_1_A$) 與 TGT 票內所描述的使用者身份相符之後，並檢視使用者所提出的安全機制 (Options) 與所要求的應用伺服器 (ID_B) 相符時，便發給前往該伺服器的服務門票 (ST、訊號 (4))。其中：

$$Ticket_B = E_{Kb} [Flages \| K_{A, B} \| Realm_A \| ID_A \| Times \| Authen\_Data]$$

$$Authen\_1_A = E_{Ka, tgs} [ID_A \| Realm_A \| TS_1]$$

◈ 要求服務：當使用者持有某一應用伺服器的服務門票 (ST) 之後，則在該門票的有效期間，都可以向該伺服器要求服務 (訊號 (5))，並攜帶本身的認證訊息 ($Authen\_2_A$); 該運作程序是由 KRB_AP_REQ 與 KRB_AP_RSP 兩訊號來完成。應用伺服器回應使用者要求時(訊號 (6))，可能會指明使用哪一把子鑰匙(Subkey)，這必須伺服器內有儲存該使用者的鑰匙版本，一般應用上，大多使用 TGS 所分配的會議鑰匙 ($K_{A, B}$)。另外序號 (Seq#) 表示每一通訊連線的序號，是被用來防禦重播攻擊的。其中：

$$Authen\_2_A = E_{Ka, b} [ID_A \| Realm_A \| TS_2 \| Subkey \| Seq\#]$$

14-4-3 Ticket 旗號

無論是 TGT 或 ST 門票都如圖 14-13 的格式，在 RFC 1510 中是利用 ASN.1 語法所描述；我們用圖形來描述它，或許能讓讀者更容易接受。未加密的部分包含有：版本（Version）、領域（Realm）、持有者名稱（Principal Name）；其它加密的參數已在前面介紹過，這裡不再重複。至於『門票旗號』（Ticket Flags）是作為表示該門票的屬性，以位元設定（True 或 False）表示某一功能的啟動否。較重要的旗號有：

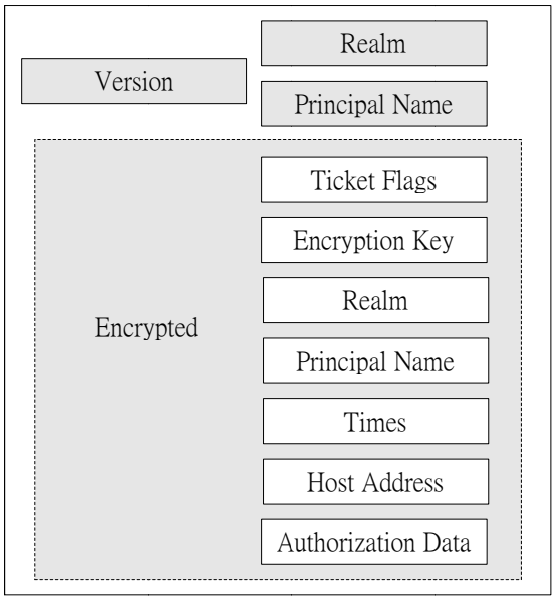


圖 14-13 門票的格式

- INITIAL：此旗號被設定，表示是起始門票，亦即是直接由 AS 伺服器所發出（TGT 門票），而非 TGS 伺服器所發出的。
- PRE-AUTHENT：預先認證。此旗號如設定，表示在起始認證當中，使用者已在發佈門票之前取得 KDC 認證完成。
- HW-AUTHENT：硬體認證。客戶端於起始認證時，會使用到硬體設備；某些客戶會利用硬體設備登入系統，如 IC 卡等，此時則需要設定此旗號。
- RENEWABLE：可重新的。此旗號如設定，表示可以向 TGS 伺服器，申請更新另一個有效日期較長的門票。
- MY-POSTSATE：可延長日期。可以根據此門票向 TGS 要求延長有效日期。

- POSTDATE：已延長。此旗號如設定，表示此門票已延長過有效日期了。
- PROXIABLE：可代理的。可依據此門票向 TGS 要求換發另一張其他網路位址的服務門票（Service Ticket）。
- PROXY：此門票是屬於代理門票。
- FORWARDABLE：可前向轉送的。可依據此門票向 TGS 要求換發另一張其他網路位址的門票核准票（TGT）。
- FORWARDED：此門票已被前向轉送的。

14-5-4 安全機制選項

Kerberos V4 的加密演算法係採用非標準的 DES 模式 —『傳導式密文區塊串接』（Propagating Cipher Block Chaining, PCBC）模式，並已證明該模式不夠安全 [84]。Kerberos V5 提供選項欄位，可讓客戶依照傳輸資料安全性的需求，協議不同的密碼系統，並且這些密碼都屬於標準規範，較容易與其他應用系統銜接。Kerberos 將安全性區分為：『檢查集』（Checksum）與『加密系統』（Encryption）等兩個層次，其中檢查集僅包含完整性功能，加密系統則包含隱密性與完整性功能。

【（A）檢查集】

目前 Kerberos V5（RFC 1510，1993 年）並沒有將較新的演算法加入選項之中，規範內的完整性檢查的標準規範有：

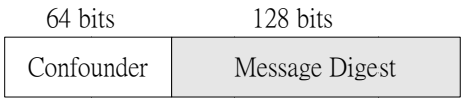
- CRC-32
- RSA-MD4
- RSA-MD5-DES
- RSA-MD5
- RSA-MD5-DES
- DES-MAC
- RSA-MD4-DES-k
- DES-MAC-k

由此可見，Kerberos V5 也不例外，大多以 MAC（Message Authentication Code） 來製作完整性檢查的認證碼（除了 CRC-32 之外）。但為了增加複雜度，V5 將每一筆訊息之前加入一個『混亂碼』（Confounder），再經過雜湊演算法計算，同時將此混亂碼一併傳送給對方。

我們以 RSA-MD5-DES 為範例，來說明產生完整性檢查碼的方法與認證步驟。MAC 產生方法如下：

1. 選擇一個 64 位元亂數作為『混亂碼』（Cofounder）。
2. 將混亂碼放置於訊息前面，格式如下：

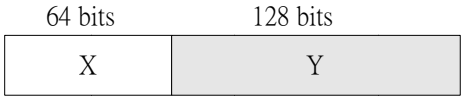
3. 將上述的訊息格式，經由 MD5 演算法計算後，得到一個 128 個位元的『訊息摘要』（Message Digest）。
4. 依照步驟 1 的方法，將訊息摘要附加在混亂碼之後，格式如下（192 個位元）：



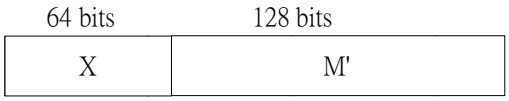
5. 將通訊者所儲存於 KDC 的共享密鑰，與資料 $F0F0F0F0F0F0F0F0_{16}$ 之間作位元 XOR 運算，得到另一把新的鑰匙 K'。
6. 利用鑰匙 K' 與起始向量為 0，將步驟 4 的訊息經由 DES 演算法的 CBC 模式計算(DES-CBC，請參考 2-7-2 節介紹)，計算後得到一個 192 位元的 MAC 值，並將 MAC 值附加在訊息（M）後面一併傳送給對方（KDC 或使用者）。

接收者收到訊息之後，如何利用訊息中的 MAC 來確認訊息完整性，步驟如下：

1. 將使用者的共享密鑰與資料 $F0F0F0F0F0F0F0F0_{16}$ 之間作位元 XOR 運算，得到另一把新的鑰匙 K'（如同上述步驟 5）。
2. 利用鑰匙 K' 向 MAC 解密，得到兩個欄位的訊息，其中 X 表示混亂碼、Y 表示訊息摘要。



3. 再將上述的 X 欄位的內容，附加在所收到訊息（M’）的前面，格式如下：



4. 接著，再將上述的訊息經過 MD5 演算法計算後，得到一個 128 個位元的 MD’。如果 MD’ = Y，則訊息的完整性檢查是正確的；否則（MD’≠ Y）表示訊息可能遭受竄改或共享密鑰不正確（身份不對）。

由上述的範例可以看出，V5 的安全性比 V4 的 PCBC 安全性高許多。

【（B）加密系統】

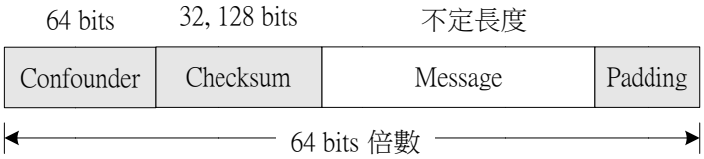
Kerberos V5 的加密系統（Encryption）亦包含檢查碼（Checksum），所以除了提供隱密性功能，亦包含完整性檢查的功能；在 RFC 1510 上規範有下列加密系統：

- Null
- DES-CBC-CRC
- DES-CBC-MD4
- DES-CBC-MD5

基本上，還是以 DES 加密系統的 CBC 操作模式為基礎（請參考 2-7-2 節介紹）；另外，加入完整性檢查有 CRC（Cyclic Redundancy Check）[3]、MD4 與 MD5 等演算法。

接下來，我們來探討 Kerberos 加密系統的製作方法，步驟如下：

1. 選擇一個 64 位元亂數作為『混亂碼』（Cofounder）。
2. 利用該混亂碼與使用者的共享鑰匙，計算出完整性檢查碼（K5 稱為 Checksum），然而演算法可以是 CRC-32、RSA-MD4、RSA-MD5-DES 等等 V5 規範中的 Checksum 演算法。得到檢查碼之後，將訊息組合如下：



其中 Checksum 欄位可能是 32 bits (CRC-32) 或 128 bits (MD4、MD5)；Padding 欄位是將整個訊息補滿 64 bits 倍數的亂數。

3. 再將上述的訊息經過 DES-CBC 演算法加密，則可得到隱密性的密文，再傳送給對方 (使用者或 Kerberos 系統) 。

雖然目前 Kerberos V5 並未將安全性較高的加密系統 (如 ASE) 或 MAC 演算法 (如 SHA-1) 加入規範之中，但我們相信這些系統應該漸漸會被植入 V5 系統之內。

14-6 結論

用戶認證算是比較複雜的運作系統，主要目的是要確認對方的身份、以及分配會議鑰匙；亦是，經過認證身份之後，再依照所分配的會議鑰匙來通訊。因此，一般安全性的通訊，可將通訊連線分為兩個階段，如圖 14-14 所示。第一階段係利用使用者主密鑰（K_A、或稱共享密鑰）或公開鑰匙來確認身份。參與認證工作的系統可以是一個專屬系統（如 Kerberos）、或是將認證協定植入應用系統內（如 IKE 協定，第十七章介紹），由應用系統自行處理。經過第一階段處理無誤之後，使用者與應用伺服器兩者已取得會議鑰匙（K_S），再進入第二階段處理。然而，第二階段主要採秘密鑰匙系統作傳輸資料，如此才可以提高傳輸效率，既然所分配的會議鑰匙可能只使用一次（或短時間使用），攻擊者欲於短時間內破解會議鑰匙，談何容易。此外，第二階段也許會採用不同的密碼系統（如 DES、ASE 或 RSA），各種密碼系統的鑰匙長度也不一樣，因此，認證系統在分配會議鑰匙時，應該會考慮到鑰匙的長度。但為了能合乎不同系統的需求，一般認證系統所分配鑰匙的長度都取較高位元（如 128 bits）；至於如何由此較長的鑰匙中，擷取出所需要會議鑰匙的長度（如 64 bits），可依照各應用系統的協定來規劃。

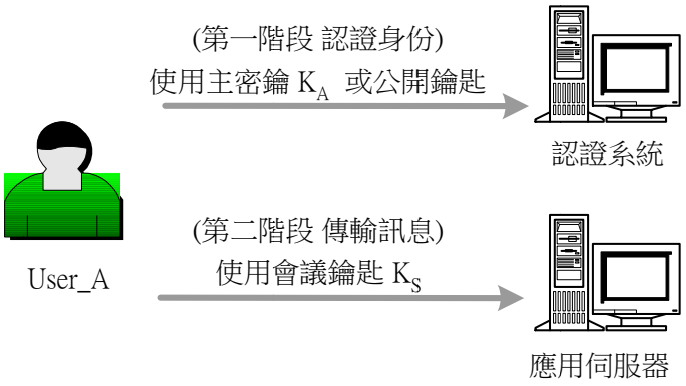


圖 14-14 安全性的通訊