第一章　緒論

1.1 研究動機與背景

信任與信賴是所有商業交易的基礎，儘管目前電子交易商業活動蓬勃發展，但是網路安全的疑慮一直是大家關心的議題及努力解決的重點。在網際網路與資訊科技發展快速的今天，政府、企業及民眾等，對於網際網路上所提供之各式各樣服務的需求，值與量均快速提升，實體社會的商業行為與服務均逐一改變成以四通八達的網際網路為基礎的數位虛擬社會。要讓使用者能夠很放心、安心的在網際網路上進行交易，必須從技術、管理及法律層面，建構一個安全及可信賴的網路交易環境，才能讓使用者有信心、有確定感，進而促進網際網路經濟活動的健全發展。

面對愈來愈龐雜而多樣化的網際網路服務，傳統的使用帳號機制管理使用者身分識別控管，已無法完全掌控。公開金鑰基礎架構（Public Key Infrastructure, PKI）[3,4]提供了公共的身分認證系統環境，透過具有公信力且可信賴的第三者，憑證中心來發行及維護使用者的數位憑證，可確保網路使用者身分。然而身分認證只是確保使用者進人某一虛擬社群中是合法的個體，也就是解決了『鑑別』的問題[22]，但是登入系統後可以允許此人執行何種程度之資源存取及內部權限調度，也就產生了『授權』的問題。

鑑別與授權兩項機制都是建立電子商務中信任與信賴所不可少的環節，ITU-T 組織於 2000 年提出 X.509 第四版中定義了授權管理基礎建設（Privilege management infrastructure, PMI）[3]及屬性憑證，運用執行權限管控之屬性憑證（Attribute certificate ,AC）實現授權、基於角色的控制及授權代理等需求的有效方法，由於屬性憑證支援授權代理，故可適用於以網際網路為基礎構連之跨國企業，而且經由權限控管機制更可以防範企業內部員工對非自己工作職權範圍內的

企業資產不正當之存取。

例如以聯電員工利用電子郵件將內部機密文件傳送給競爭對手，造成公司營業秘密的外洩事件，而為公司帶來實質與潛在的損失。所謂的營業秘密，依營業秘密法第二條規定，必須符合下列三個要件：一、「新穎性」，就是該方法、技術、製程…並非一般涉及該類資訊之人所知者。二、「價值性」，就是因其秘密性而具有實際或潛在之經濟價值者，而該價值不限於積極之價值，負面失敗之資訊亦包括。三、「秘密性」，就是秘密之所有人已採取合理之保密措施者。而所謂「所有人已採取合理之保密措施」，係指營業秘密之所有人，在客觀上已經為一定之行為，使人了解其有將該資訊當成秘密加以保守之意思。依前述營業秘密法之立法意旨，若公司對此份資料沒有採取「合理之保密措施」，縱然公司認為這份資料很重要，但仍然不屬於營業秘密。

上述案例由法律的角度來看，可清楚了解企業內部資源控管機制若不當，企業依賴競爭優勢及創造利潤之機密技術文件，一但外洩，在法律上將不獲得保障；故以網際網路為基礎下，企業應對外部或內部資源控管建立一致性的身分鑑別及一致性的資產執行權限控管，以保障資產不被非法的竊取或盜賣。

基於此問題，本論文主要是以授權管理基礎建設為籃圖，設計出以屬性憑證為使用者授權管理為目的之企業內部權限控管存取架構，使 PKI 架構更完整及更高的安全性，是為本論文的研究主題重心。

1.2 研究目的

公開金鑰基礎架構的本質在於解決網際網路環境裡參與個體的不信任問題，現行公開金鑰基礎架構環境僅以公開金鑰憑證實施使用者身分鑑別，以確保網路作業之安全性，惟對於持有合法身分憑證之使用者依其角色定位所付予之可執行作業權限，並無加以規範及控管，亦表示只要是通過系統身分憑證鑑別無誤，

即可執行或存取該網路所提供之各式服務或資源，此點容易造成資源存取控管之盲點。

本研究主要目的在公開金鑰基礎架構環境下，利用屬性憑證之優點，規劃、設計出授權管理基礎架構，其具有安全性更高、更完整及更有效率，在權限管理上更有彈性；此架構使企業在網路運作上不僅有效保障數位資產，也使得使用者對企業所提供之認證服務有信心，提高參與電子商務作業。

1.3 論文章節概述

本論文共分為六章，第一章為緒論，主要闡述研究的動機及背景及研究目的，及引導出整個論文的架構。

第二章是文獻探討，探討的議題為本論文所應用之技術，包含四大部份：（1）公開金鑰基礎架構基本組成元件及運作流程（2）X.509 憑證介紹（3）目錄服務簡介（4）輕量目錄存取協定運作機制。

第三章是探討與本論有關之授權管理理論（任意性的存取控制、強制性的存取控制、以職位為基礎之授權模式）及運作原理，並針對各種授權理論依不同構面比較分析其優缺點。

第四章是探討本論文核心之授權管理基礎架構，本章詳細整理國外 ITU-T 組織公布之 X.509v4 授權管理基礎架構組成元件、運作模型及各類相關官方與產業報告，並將以屬性憑證為授權模式概念之授權管理基礎架構與公開金鑰基礎架構之關係作一分析比較，作為本論文第五章設計空軍總部授權管理基礎架構之理論基礎。

第五章是本論文理論架構之應用設計，並以空軍總部為範例，提出適用於機密資料保管特性的組織授權架構，並設計以「業務」為導向之授權管理模式，主要目的是減少最高屬性憑證管理中心（SOA）之管理作業負荷，並精減授權體系之層級，以提昇作業效率。

第六章為結論及建議，提出本論文設計授權管理架構之心得及後續研究方向，供爾後研究之參考。

第二章　文獻探討

2.1 公開金鑰基礎架構簡介

PKI 是一種資訊安全機制，係運用公開金鑰及電子憑證以確保網路交易的安全性及確認交易對方身分之機制。公開金鑰基礎建設係以網路認證之信任機制為基礎，交易雙方相互地信任其認證機構，搭配金鑰對之產製及數位簽章等功能，即可經由其認證機構核發之電子憑證確認彼此的身分，能在電子訊息傳遞過程中，提供含身分鑑別（Authentication）、不可否認性（Non Repudiation）、資料完整（Integrity）、機密性（Confidentiality）等的資訊安全保障；例如，在網路下單買賣股票時，在 PKI 機制下配合電子簽章所給予之法律效力，能確保交易記錄的存在（不可否認性）、買賣的完成（資料完整）及雙方身分（身分鑑別）。

公開金鑰基礎架構由下列幾個主要元件所組成(圖2-1)[5]：

- **終端實體(End Entity)**：使用憑證的主要對象，終端實體可視為一般使用者、團體、公司行號或是系統等。
- **憑證中心(CA)**：可信任的公正第三者，負責憑證、憑證註銷清冊之發行與管理等工作。
- **註冊中心(RA)**：位於憑證中心之前端，終端實體可透過註冊中心申請憑證。
- **憑證/憑證註銷清冊資料庫(Certificate/CRL Repository)**：公開金鑰基礎架構中的資料儲存體，負責儲放憑證、憑證註銷清冊之資料。

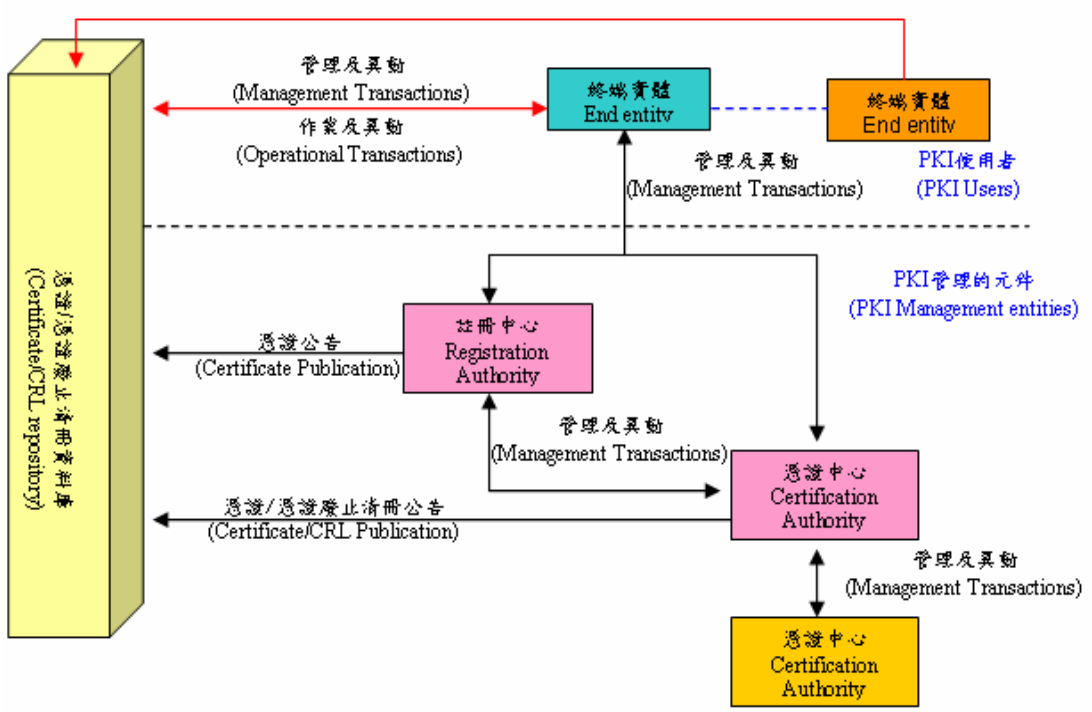


圖 2-1 公開金鑰基礎架構模型

資料來源：IETF," Internet X.509 Public Key Infrastructure Certificate and CRL Profile," RFC 2459, January 1999.

2.2 X.509 憑證

所謂憑證是指一組固定格式的資料，將使用者的身分（Identity）與其公開金鑰連結（Binding）在一起。憑證可以是 PKI 的基礎元件，而每個憑證至少包含了一把公開金鑰（Public key）及相對應的私密金鑰（Private key），這組金鑰對是由一個大家所共同位賴的認證中心（CA）簽署，認明公開金鑰與持有人的真實性。

目前有關 X.509 憑證的標準發展整理如下：

（1）ITU-T

1.Version 1（V1）：1988 年出版 ITU-T X.509。

2. Version 2（V2）：1993 年增加「使用主體單位唯一 ID」和「簽發單位唯一 ID」兩個欄位，稱為 ITU-T X.509[10]。

3. Version 3（V3）：1997 年出版，增加擴充欄位（Extensions），讓 X.509 便具彈性。

4.Version 4(V4)：2000 年出版，提出屬性憑證（Attribute certificate）概念。

（2）ISO 9594-8

1.將 ITU-T X.509 列為 ISO 的標準，稱為 ISO/IEC 9594-8[11]。

2.將 X.509（v3）列為 ISO 的標準，稱為 Amendment 1 to ISO/IEC 9594-8[12]。

2.2.1 憑證格式

在網路的虛擬世界中，基本上是一個不安全的環境，要如何達成雙方秘密通連與相互鑑別身分，以 X.509 憑證架構之 PKI 下，必須在系統中建立一個公開及可信賴的第三者（Trust Third Party, TTP），我們稱為憑證中心（Certification Authorith, CA），負責產製及管理維護終端實體的金鑰憑證。憑證申請是由 CA 核對身分及所提出之相關所需認證資料後，CA 以自己的私鑰對申請者的身分資料和相對應的公鑰做簽署的動作，產生一個數位簽章。而所謂的相關資料包括了申請者的名稱、使用者 ID、使用者對應的公開金鑰及發行者資料，此資料結合數位簽章，就是一張憑證。

憑證中以數位簽章簽發憑證，憑證中包含實體（Entity）之相關資訊，如公開金鑰、憑證唯一序號、實體名稱及憑證持有者資訊等。X.509 定義憑證格式並由 CA 發行，以下說明憑證符號格式，並解釋其意義：

$$\mathbf{CA<<A>>=CA\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}}$$

V 憑證版本代號。

SN 憑證序號。

AI 憑證簽章所使用之演算法。

CA 憑證中心識別名稱。

UCA 憑證中心唯一識別碼（選項）。

A 憑證主體唯一名稱。

UA 憑證主體唯一識別碼（選項）。

Ap 憑證主體 A 之公開金鑰資訊。

T^A 憑證之有效期限。

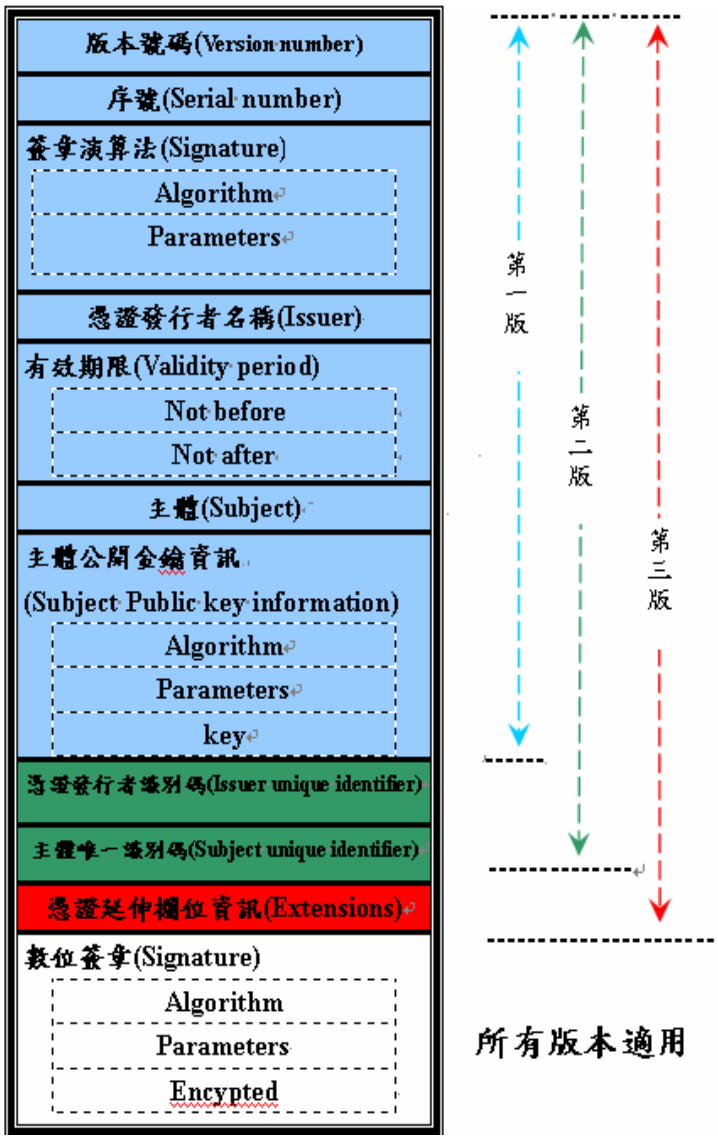


圖 2-2 X.509 V1、V2 及 V3 憑證格式

X.509 憑證如圖 2-2，標準欄位定義說明如下：

- **版本號碼（Version number）**：發行憑證之版本代號。X.509 憑證最後一版本本為 Version3。
- **序號（Serial number）**：憑證中心發行此憑證所付予之唯一序號。
- **簽章演算法（Signature）**：憑證中心發行此憑證所使用之雜湊函數（hash function）及簽章演算法。

- **憑證發行者名稱（Issuer）：**簽發此憑證之憑證中心名稱。
- **有效期限（Validity）：**定義憑證之有效期限，包含開始日期及結束日期。
- **主體（Subject）：**憑證主體資訊，可以是個人（Individual）或實體裝置（Entity），可以經由憑證中的公開金鑰相對應。
- **主體公開金鑰資訊（SubjectPublicKeyInfo）：**憑證主體之公開金鑰資訊。
- **憑證發行者識別碼（IssuerUniqueIdentifier）：**憑證發行者識別碼，供憑證中心簽發憑證時驗証發行者名稱是否重複使用，本欄位為選擇性使用，並只適用於 Version2 及 Version3 憑證版本。
- **主體唯一識別碼（SubjectUniqueIdentifier）：**憑證主體唯一識別碼，供憑證主體驗証主體名稱是否重複使用，本欄位為選擇性使用，並只適用於 Version2 及 Version3 憑證版本。
- **憑證延伸欄位資訊（Extensions）：**提供記載憑證額外資訊，而不須要修改憑證格式。

X.509 中定義上述資料欄位，並使用 ASN.1[5]規範來描述憑證資料格式，內容如下：

```
Certificate   ::=   SEQUENCE   {
        tbsCertificate          TBSCertificate,
        signatureAlgorithm      AlgorithmIdentifier,
        signatureValue          BIT STRING   }

   TBSCertificate   ::=   SEQUENCE   {
        version             [0]   EXPLICIT Version DEFAULT v1,
        serialNumber              CertificateSerialNumber,
        signature                 AlgorithmIdentifier,
        issuer                    Name,
        validity                  Validity,
        subject                   Name,
        subjectPublicKeyInfo SubjectPublicKeyInfo,
        issuerUniqueID   [1]   IMPLICIT UniqueIdentifier OPTIONAL,
                                  -- If present, version shall be v2 or v3
        subjectUniqueID [2]   IMPLICIT UniqueIdentifier OPTIONAL,
                                  -- If present, version shall be v2 or v3
        extensions          [3]   EXPLICIT Extensions OPTIONAL
```

```
                                    -- If present, version shall be v3
        }
    Version   ::=   INTEGER   {   v1(0), v2(1), v3(2)   }
    CertificateSerialNumber   ::=   INTEGER
    Validity ::= SEQUENCE {
            notBefore           Time,
            notAfter            Time }
    Time ::= CHOICE {
            utcTime              UTCTime,
            generalTime         GeneralizedTime }
    UniqueIdentifier   ::=   BIT STRING
    SubjectPublicKeyInfo   ::=   SEQUENCE   {
            algorithm                AlgorithmIdentifier,
            subjectPublicKey         BIT STRING   }
    Extensions   ::=   SEQUENCE SIZE (1..MAX) OF Extension
    Extension   ::=   SEQUENCE   {
            extnID          OBJECT IDENTIFIER,
            critical        BOOLEAN DEFAULT FALSE,
            extnValue       OCTET STRING   }
```

2.2.2 憑證型態（Certificate Types）

目前憑證型態主要可區分為兩種，一種是終端實體憑證（End-entity certificates），是由 CA 發行給實體（entity），此憑證不可以再往下發行給另一實體；另一種是憑證機構憑證（CA certificates），是由 CA 發行給終端實體，但可以再往下發行至另一實體，此實體可以是憑證機構或是一個終端實體。

憑證機構憑證又可區分為下列三種型態：

- 自我簽發憑證（Self-issued certificates）：自我簽發憑證格式中發行者名稱（issuer）及憑證主體名稱（Subject ）與發行憑證機構（CA）相同。此憑證可應用於特殊驗證作業，例如當 CA 產製新公開金鑰對時憑證中包含新公開金鑰，並使用舊金鑰對憑證簽章時，亦允許與新的公開金鑰建立信任。

- 自我簽章憑證（Self-signed certificates）：自我簽章憑證格式中的公開金鑰與憑證的數位簽章相同，主要為建立信任模式。

- 相互認證憑證（Cross-certificates）：在交互認證憑證格式中憑證主體名稱（Subject ）及發行者名稱（issuer）是來自不同的憑證機構，主要是應用於 CA 之間相互認證，但前提是必須先在 CA 之間建立信賴關係（Trust Relationship）。

2.2.3 憑證路徑

建置 PKI 的系統環境中 CA 不可能只有一個，若兩個終端實體由不同 CA 所發的憑證要通訊，但相互並不信任，此刻立即產生相互認證問題。在 X.509 中提出憑證路徑（Certification path）概念，即憑證的驗證方可逐步的重複追蹤，直到接收方可以信賴的 CA 為止。如此 CA 與另一 CA 間就是以階層式架構所構成，即為一個憑證鏈（Certification Chain），如此便可解決溝通分屬不同 PKI 之 CA 通訊問題，驗證憑證的正確性[24]。

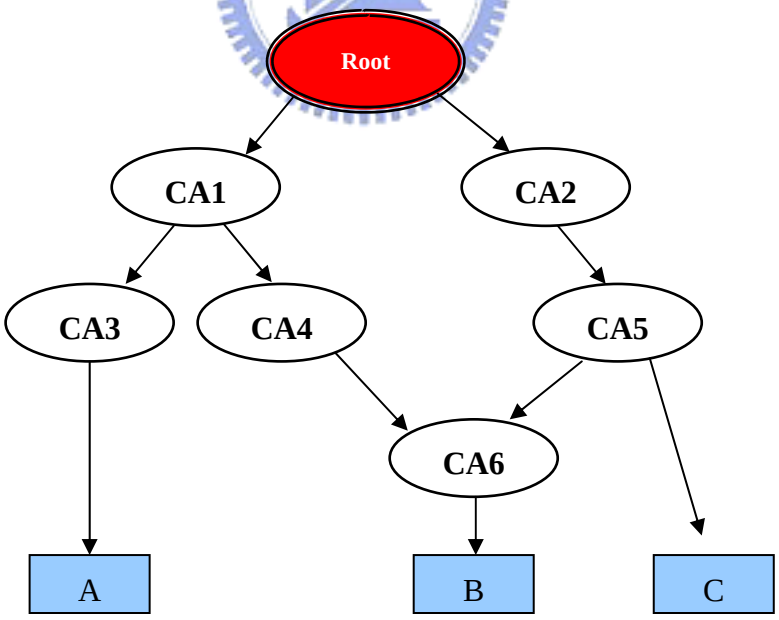


圖 2-3 憑證路徑

資料來源：吳宗成，”資通安全技術資源簡介-資通安全專輯之三，”行政院國家科學委員會科學技術資料中心，2002.

圖 2-3 所示，若使用者 A 的信任 CA 為 CA3，而使用者 B 的憑證為 CA6 所簽發，當使用者 A 由目錄服務中獲得以下憑證路徑時，CA3<<CA1>>, CA1<<CA4>>, CA4<<CA6>>, CA6<<B>>，則使用者就可憑這一串憑證路徑，利用使用者 A 所信任 CA3 公鑰，驗證 CA3 對 CA1 所發的公鑰憑證之簽章是否正確（CA3<<CA1>>），若是正確，即可驗證得知 CA1 之公鑰是可信賴的公鑰；接著 CA1 的公鑰驗證 CA4 公鑰的正確性，如此重複，最後可驗證得到 CA6 公鑰的正確性，然後再以 CA6 的公鑰驗證 B 的憑證（CA6<<B>>），便可知道 B 的公鑰是否可以信賴。

X.509 亦提出交互認證的概念，也就是各 CA 間是可以互相認證，以圖 2-3 為例若 B、C 要相互認證那使用者 B 所得到的憑證路徑為 CA6<<CA5>>, CA5<<C>>，如此認證路徑縮短，使得目錄服務也都減少許多負擔，提昇作業效率，此認證屬於非階層式認證中心架構。除了交互認證外，為了降低使用者或是目錄服務的負擔，X.509 提出下列認證原則：

（1）若使用者 A 與 B 的憑證屬同一個 CA，則 A 只要對 CA<<B>>直接驗證即可，B 亦只要驗證 CA<<A>>。

（2）若為階層式認證中心架構，使用者可先儲存本身到 Root CA 的順向或反向憑證，則後來就只要取得對方到共同信任 CA 路徑即可。（如圖 3-3 所示當使用者 A 欲與 B 認證，則只要取得 CA4、CA6 之憑證即可。）

（3）若使用者 A 經常和某一 CA 所簽發的使用者通訊，則 A 可事先儲存本身與該 CA 之間的順向或反向憑證。

（4）若使用者 A 與 B 經常通訊，則可將對方的憑證儲存下來，即可免向目錄服務要求服務。

2.3 目錄服務（Directory Service）簡介

廣義的來說，目錄服務是一種儲存資料的結構，而在電腦系統目錄服務是為在網路上，快速擷取資訊而設計的，它們針對經常讀取且很少寫入做最佳化，並且所有的目錄服務一定含有一個命名服務，目錄服務是建立在命名服務之上，命名服務能夠把一個物件視為獨一無二，目錄服務除了擁有命名服務的功能外，還提供了查詢物件，並且讓物件與屬性產生關聯。目錄服務提供我們需要查詢資料的時候，可由一些基本的關鍵字進行資料的查詢，例如從電話簿裡，我們可以直接依照資料的名稱進行查詢，當然，在電腦的系統上，也可以提供相關的關鍵字，例如我可以直接以印表機的名稱查到該印表機的相關資料，假設無法提供關鍵字，我們也可以透過分類的方式進行查詢，舉例來說，也許要查詢一家位於自己主住家附近的五金行，當然事先不知道名稱，所以我們可以透過電話簿的分類，從五金行的分類找到一家在自己住家附近的五金行。電腦上的目錄服務也是如此，但是比起傳統的目錄服務來說，又更為有彈性，因為在電腦上，我們不僅可以用分類進行資料查詢，也可以透過一些特別的資料屬性進行查詢。

今天在網路上執行了許多程式 ，每一種環境都需要相關的使用者與資源的資訊，因此在維護與多個系統同步化的考量上，有效的儲存與管理資訊儼然變成在企業電子化的過程中一個最重要的環節。目錄服務的目的就在一個分散式網路環境裡，展現一個安全、具延展性並且易於維護的單一資訊來源的方式。

2.4 輕量目錄存取協定(Lightweight Directory Access Protocol,LDAP)

LDAP[1,2]為美國密西根大學根據 X.500[8,9]精簡後提出較為精簡目錄存取協定，主要目的是能在 TCP/IP 上實作出目錄服務系統，並簡化用戶與 X.500 目錄伺服器之間的協定，因此，LDAP 伺服器提供用戶端標準的目錄服務，同時具有與 X.500 目錄伺服器溝通的能力。

LDAP 可讓使用者找到網路中的組織、個人、檔案或裝置等其他資源的一種軟體協定，不論是公共的網際網路或是內部網路。LDAP 的目錄為一層一層分支出去的樹狀圖，從根目錄以下，細分國家、地區、組織、小組織及個人。整個目錄分布在許多伺服器中每個伺服器都複製了一份整體分支圖，定期同步化資料。一個 LDAP 伺服器被稱作 Directory System Agent（DSA），接收使用者要求指令，並在必要時傳給其他 DSA，並確保有單一伺服器可真正執行任務。因此，經由 LDAP Protocol 你可以很容易的跟任何一台支援 LDAP 的 directory server 存取你想要的資料，而不用去關心 LDAP server 後面的資料庫是那一家公司的，使用何種作業系統，而能真正感受到目錄服務所帶來的便利。

LDAP 目錄服務定義了四種物件 Person、Organizational Unit、Group、Domain，六種可能的操作 Binding to the server、Searching the server、Compare entries、Adding an entry to the server、Modifying existing entries、Removing an entry from the server，與三種服務 Referral、Replication、Encryption/Security 藉由物件與相關的操作，達成企業體中目錄服務所需提供的功能。

2.4.1 LDAP 協定模型（**Protocol Model**）

LDAP 的協定模型(圖2-4)，是定義在TCP/IP 的網路環境中，用戶端與伺服器端間，一連串的服務要求與回應之訊息的交換過程。一個標準的LDAP 連線服務流程如下：

1.用戶端向伺服器端發送連線要求(BindRequest)，要求建立連線。

2.伺服器端回應給用戶端連線回應(BindResponse)，連線建立或錯誤訊息。

3.用戶端發送操作要求(OpertionRequest)給伺服器端。

4.伺服器端接收到操作要求後，對LDBM(LDAP DatabaseManagement) 執行用戶端的操作服務。

5.伺服器端將操作的結果回傳給用戶端，若是操作發生錯誤，則回傳錯誤訊息。

6.用戶端向伺服器端發送終止連線要求(UnbindRequest)，終止LDAP連線。

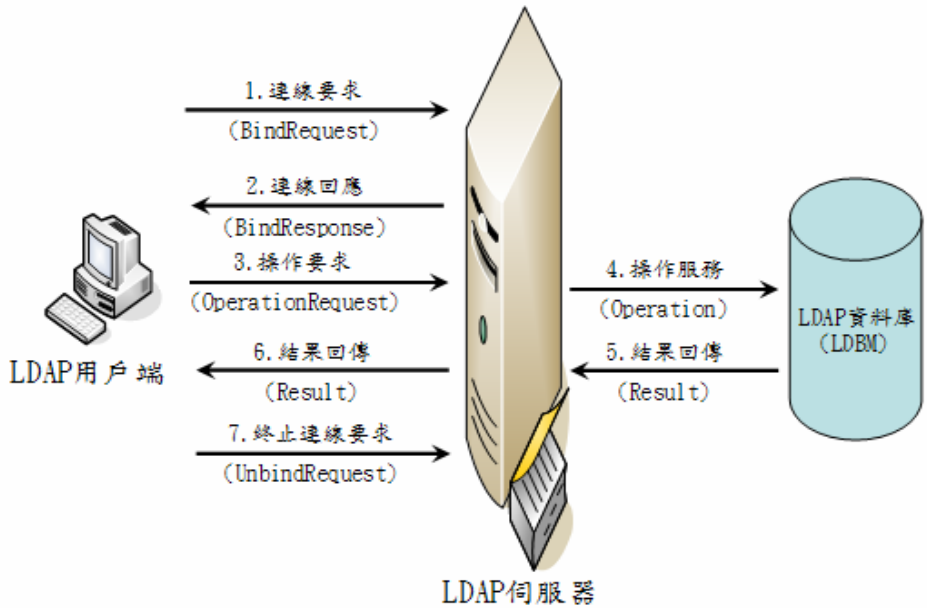


圖2-4 LDAP 協定模型

2.4.2 LDAP 資料模型（**Data Model**）

LDAP 的資料儲存方式，它是將資料依樹的型式儲存，一個一個的 Entry 構成一個完整的資料目錄樹(Directory Information Tree,DIT)，每個 Entry 我們可以想像成是關聯式資料庫的橫列(Row)，裡面存放著我們的資料，而每個 Entry 都會有一個 DN(Distinguished Name)，這個 DN 是由命名服務而來，並且由數個元件組成，作為一個獨一無二的識別，例如，DN: cn=WCH, ou=IIM, o=NCTU, c=TW。而每一個 DN 元件的命名，會與上層的 RDN(Relative Distinguished Name)有關係。由於 LDAP 是根據 X.500 而制定，所定義的資料模型之資訊也相容於 X.500 定義的資料模型。LDAP 伺服器所提供的目錄服務，需要資料庫 LDBM(LDAP Database Management)來負責儲存資料。LDAP 的資料模型的組成元素包含下列幾項（如圖 2-5）：

- **DIT(Directory Information Tree)**：為了提供LDAP 快速存取資料的特性，LDBM 的資料結構為階層性樹狀結構(hierarchical tree-like structure)，而非用一般關聯式資料庫。LDBM 的樹狀資料結構，稱為DIT。在邏輯上，一個以上具有從屬關係的DIT，可以接合成更大型的DIT。
- **Entry**：LDAP 的資料項，也是組成DIT 的基本單元；在實際應用上，一個 entry 可代表一個使用者、一個組織單位、一部電腦、一台伺服器或一個週邊裝置的識別資料。 Entry是儲存資料的地方，一個Entry是由許多屬性組合起來，而每個屬性都是由一個Type與一個數值(Value)組成，就好比一筆關聯式資料庫的row的資料一樣，一筆row裡面有許多的column，每個column會有一個型態(Type)也會有一個數值(Value)存在。

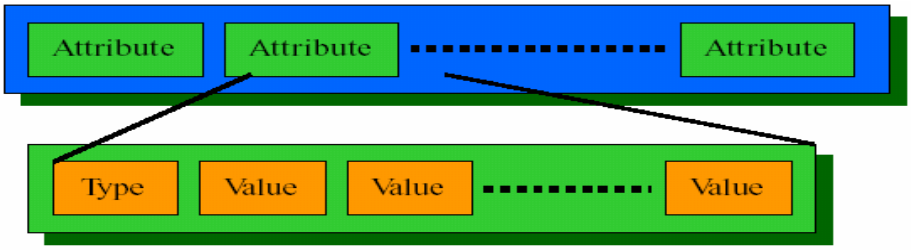


圖2-5 LDAP 資料模型

- **物件類別(Object Class)**：物件類別為一種特殊的屬性，主要目的在將一般屬性做分類歸納。每一個entry 必須包含一個以上的物件類別，這個特殊屬性的屬性值是可以更改的，但是屬性型態objectClass是不容許變更。
- **屬性(Attribute)**：一個entry 是由一個以上的屬性所組成，每一個屬性包含一個屬性型態(attribute type)及一個以上的屬性值(attribute value)。
- **Schema**：schema 為屬性型態與物件類別定義的集合，主要在定義LDAP 伺服器可允許使用的屬性及其規格資訊。LDAP伺服器可根據schema 的內容來做搜尋資料的過濾規則，或規範目錄資料庫內的entry 可擁有的屬性資料。

為了能夠順利的在不同的LDAP伺服器中交換物件資料，故必須使用具備跨平台且統一的格式，來儲存LDAP物件的資料。LDIF（LDAP Data Interchange Format）是LDAP資料的標準格式，LDIF是以純文字的格式來儲存LDAP 屬性資料，並使得LDAP伺服器間的資料交換更方便，圖2-6 是一個範例。

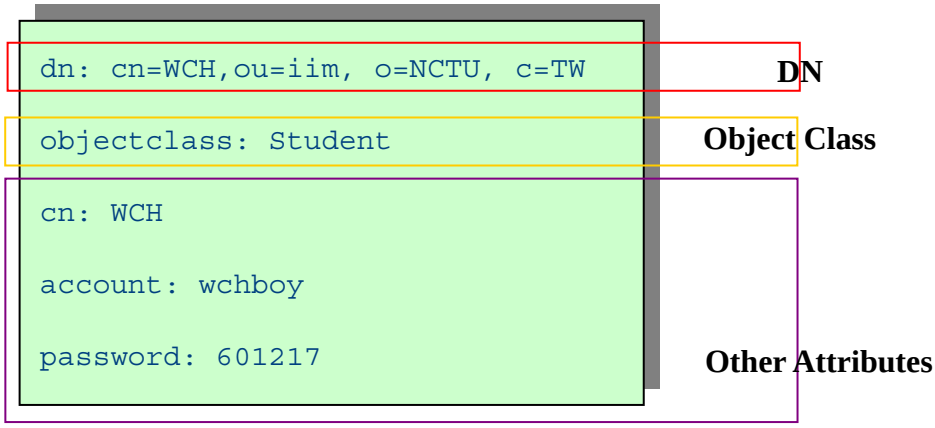


圖2-6 LDAP 屬性資料交換格式

2.4.3 LDAP 命名方式（**Naming Rule**）

由於LDAP 的目錄資料庫為階層式樹狀結構，一般而言，這樣的結構設計可以對應地理關係或國家組織這類具有從屬關係的資料。目錄資料庫中的每一個資料項都有其唯一的識別名稱DN，DN 的命名原則有下列兩種：

- **傳統命名(Traditional Naming)**：傳統命名原則是根據X.500對DN 命名原則修訂[18]，如圖2-7 所示，是依照國家組織的從屬關係建立目錄資料項，以右邊的樹狀結構來看，最上層的資料項為國家—台灣(c=TW)，其次為省份—台灣省(st=Taiwan)，再下一層為組織—交通大學(o=NCTU)，再下一層為組織內的單位—資管所(ou=IIM)，最底層為人(cn=WCH)。

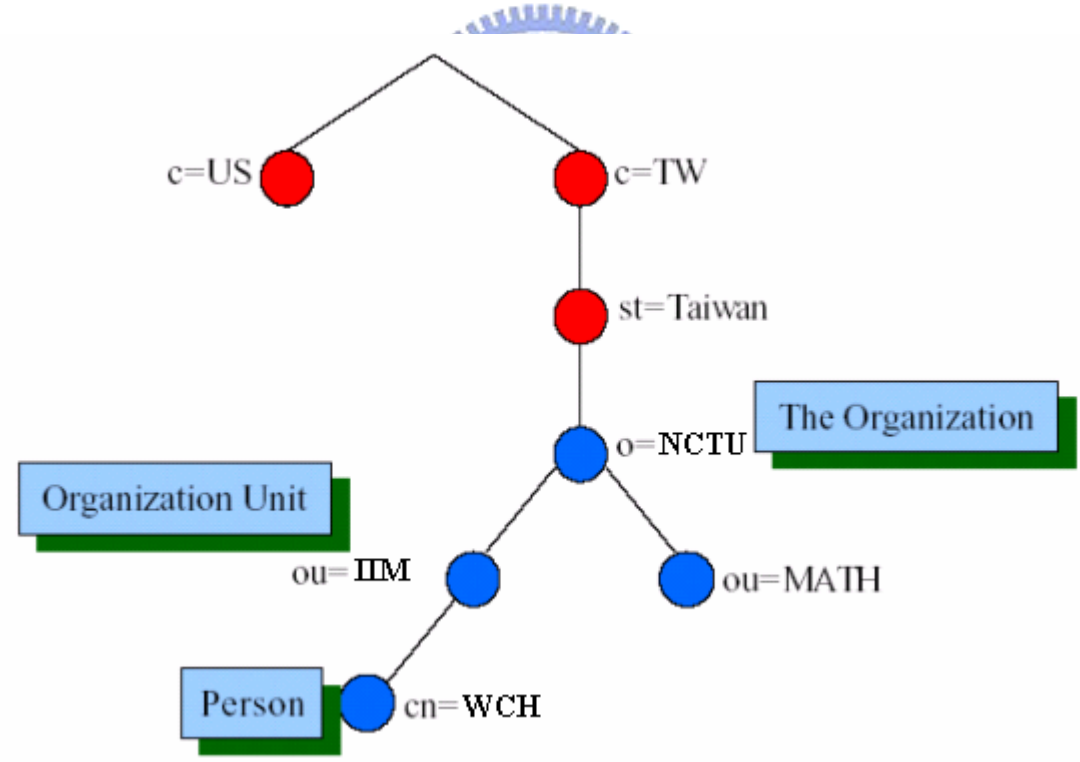


圖2-7 LDAP 傳統命名法則

網際網路命名(Internet Naming)：以目前LDAP 應用在網際網路的情況看來，傳統的命名原則不全然可以提供較適宜的命名方式，為了實際的需求及方便性，根據TCP/IP 網域域名為基礎而訂定了新的命名原則。如圖2-8 所示，使用新的屬性值dc 來代表TCP/IP 域名，中間的子樹的根為dc=edu.tw，再其次為dc=nctu，此一資料項的DN 為dc=nctu,dc=edu.tw，對應域名為nctu.edu.tw，再往下層命名原則與傳統命名原則相同（圖3-17）。

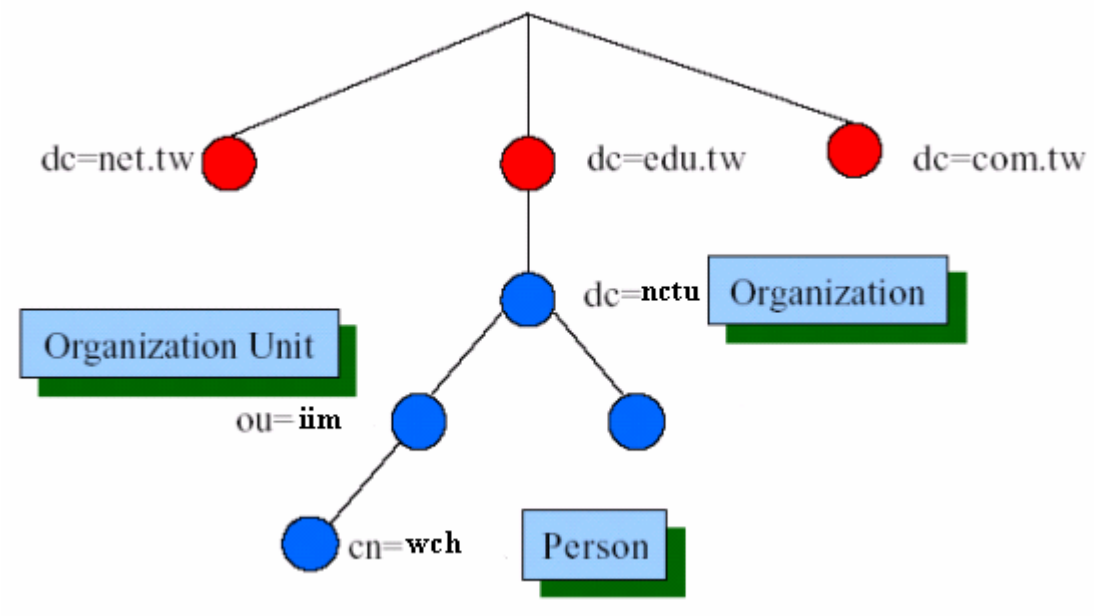


圖2-8 LDAP 網際網路命名法則

2.4.4 LDAP伺服器類型及運作機制

一般而言，LDAP 伺服器共可分為四種類型：本地目錄服務(Local Directory Service)、具參照功能之本地目錄服務(Local Directory Service with Referrals)、複製目錄服務(Replicated Directory Service)及分散式本地目錄服務(Distributed Directory Service)。使用者可依自己的需求將LDAP 設定為其中一種類型的LDAP 伺服器。運作機制分述如下：

1.本地目錄服務(Local Directory Service)

只為伺服器所屬的網域提供目錄服務，伺服器並不會與其他的目錄伺服器進行溝通。因此，如果是剛開始使用或是只打算提供本地端的服務且不想和其他伺服器做溝通，則此類型較適合，因為任何時候都可以容易地變更為其他類型的LDAP伺服器（圖2-9）。

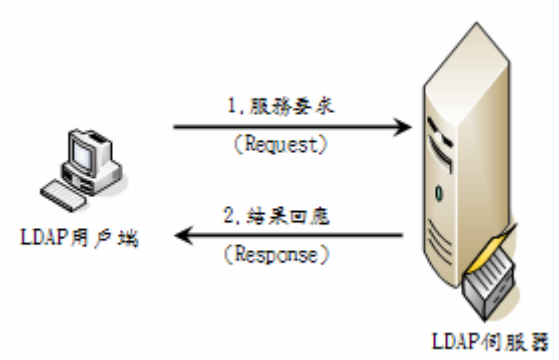


圖2-9 本地目錄服務

2.具參照功能之本地目錄服務(Local Directory Service with Referrals)

此類型的伺服器可以提供目錄服務給該伺服器的網域用戶以及設定並且傳送位於本地網域以外的上層伺服器的參照值。當用戶端提出服務要求，LDAP 伺服器回應結果，若LDAP 伺服器本身無此資料，則會回應給用戶端參照的訊息，讓用戶端可向所指引的其他LDAP伺服器提出服務要求。由圖2-10可以發現，若想要提供本地端服務並可與全球目錄互通的話，這個類型為最佳選擇。

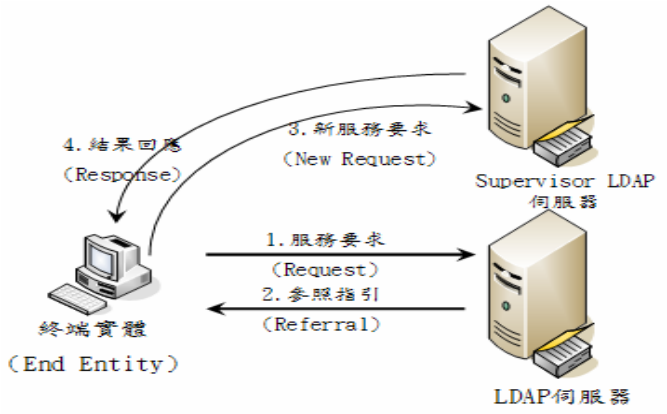


圖2-10 具參照功能之本地目錄服務

3.複製目錄服務(Replicated Directory Service)

伺服器將有異動的資料從某一主要伺服器傳送到一個或多個從屬伺服器。當單一LDAP 伺服器無法提供服務所需的可靠性或有效性時，複製目錄服務類型的服務可以和前述兩種服務之一的設定合併使用（圖2-11）。

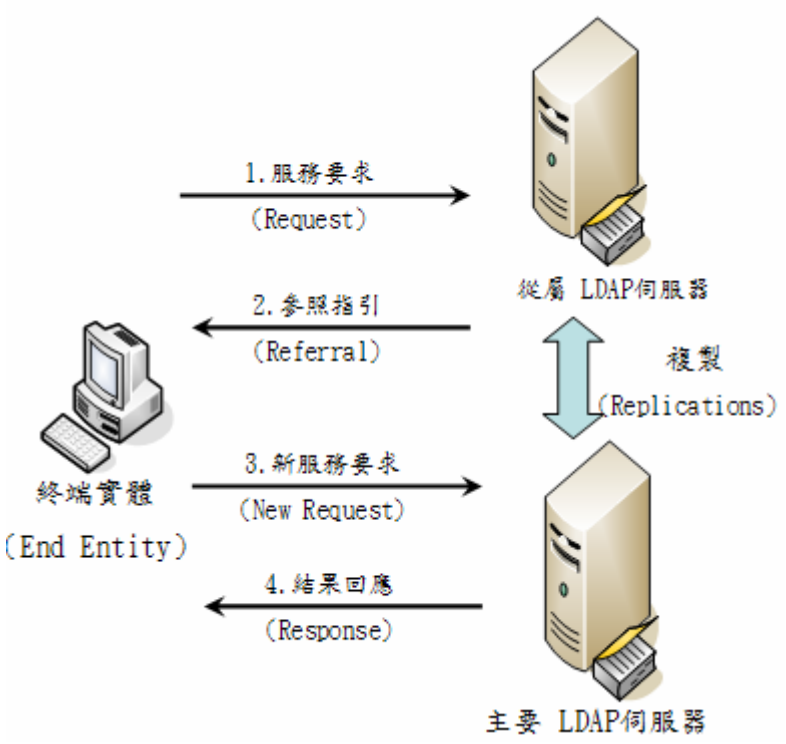


圖2-11 複製目錄服務

4.分散式目錄服務(Distributed Directory Service)

本地服務將被細分為更小的多個服務，各個服務可能會重複並且與上層或下層LDAP 伺服器結合在一起。

2.4.5 LDAP協定操作功能(Operation)

LDAP 定義出前述的兩個模型：協定模型、資料模型，在TCP/IP的網路環境中，用戶端與LDAP 伺服器端之間對於LDAP 內部資料之存取操作，依LDAPv3所定義出來的操作功能，可區分為三大類型：

（1）查詢(Interrogation)：提供用戶端對LDAP 伺服器端的資料進行搜尋、比對。

1.Search：允許用戶端向LDAP 伺服器端提出搜尋資料的要求。

2.Compare：允許用戶端對LDAP 目錄資料庫中的一個entry的內容做比較。

（2）異動(Modification)：提供用戶端對LDAP 伺服器端的資料新增、修改等管理功能。

1.Add：允許用戶端向LDAP 伺服器端要求新增一筆資料之操作服務。

2.Delete：允許用戶端向LDAP伺服器端要求刪除一筆資料之操作服務。

3.Modify：允許用戶端向LDAP 伺服器端要求修改一筆資料內容之操作服務。

4.Modify DN：允許用戶端向LDAP 伺服器端要求修改一筆資料的DN，或是將一整個子樹下所有的資料移至目錄的另一個新的位置。

（3）鑑別(Authentication)：提供用戶端與伺服器端之間連線的建立、終止，以及建立連線需要的鑑別機制：

1.Bind：允許用戶端與伺服器端之間交換鑑別資訊。

2.UnBind：終止LDAP 協定的連線。

3.Abandon：允許用戶端在操作意外時提出中斷交談的要求。

2.5 ISO/IEC 10181-3 開放式系統存取控制（Access Control Framework）架構簡介

ISO/IEC 10181-3 開放式系統存取控制（Access Control Framework）架構[13]是由國際標準組織（International Standard Organization, ISO）及國際電子工程委員會（International Electrotechnical Commission, IEC）所制定的標準，主要定義在開放式的電腦運作環境下的存取控制機制標準，全文共有九章及附錄 A-G。

所謂存取控制（Access Control）依 ISO/IEC 10181-3 定義：「在開放式的環境中鑑定對使用資源的過程，允許以適當的方法預防未經授權的存取」。而整個安全架構所提出的論點如下：

（1）定義存取控制的基本觀念。

（2）說明基本存取控制的觀念能夠專業的支援一般存取控制服務及存取控制機制。

（3）定義存取控制的服務及相對應的存取控制機制。

（4）確認支援存取控制服務及存取控制機制的功能需求協定。

（5）確認支援存取控制服務及存取控制機制的管理需求。

（6）說明存取控制服務及存取控制機制與其它安全服務機制的交互運作方式。

整個安全架構的設計主要目的是防範電腦系統或是通訊系統，在運作時遭受未經授權的存取威脅，這些威脅可以區分為下列五類：

（1）未經授權的使用（unauthorized use）

（2）未允許的揭露（disclosure）

（3）未允許的修改（modification）

（4）惡意的毀壞（destruction）

（5）拒絕服務（denial of service）

2.5.1存取控制功能元件

存取控制功能元件（Access Control Function）定義出基本存取控制系統所需的組成單元，並可提供系統實作時的設計參考標準，其主要的元件包含（1）起始主體（Initiator）、（2）存取控制執行功能（Access Control Enforcement Function, AEF）、（3）存取控制決策功能（Access Control Decision Function, ADF）、（4）目的受體（Target）等四部份，整體架構如圖 2-12。

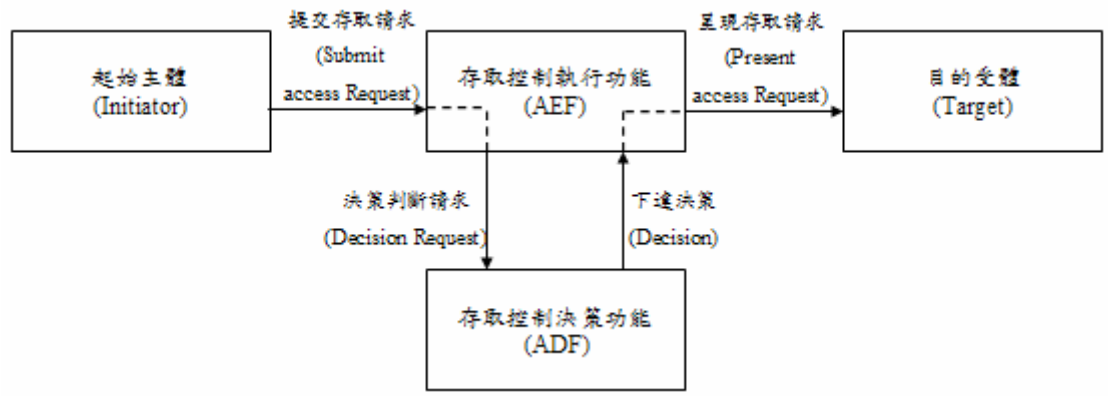


圖2-12 基本存取控制功能元件

資料來源：ISO/IEC,” ISO/IEC 10181-3 Information Technology – Open System Interconnection – Security Framework for Open System: Access Control Framework,”1996.

起始主體是存取控制功能的輸入端，將存取控制的請求輸入AEF，起始主體可以是使用者或是電腦為基礎的實體（computer-based entity）等；AEF負責接收起始主體所提出的存取請求，並檢查存取請求訊息內容所使

用的加解密演算法是否相同及訊息是否來自於同一安全領域內，然後將此請求訊息傳送給ADF，AEF是建立起始主體及目的受體間的存取路徑的溝通介面；ADF依組織所定訂的存取政策規則及存取控制決策資訊（Access Control Dicision Information, ADI），判斷是否允許起始主體存取目的受體的請求，並將決策後資訊交由AEF執行。

2.5.2存取控制決策功能

整個存取控制決策功能中最重要的單元就是ADF，當ADF執行起始主體所提出的存取請求決策判斷時，必須輔以相關ADI，這些資訊是決定資源管理存取控制的依據。圖2-13是ADF各種不同的進入資訊說明，可區分為如下：

（1）起始主體的存取控制決策資訊（Initiator ADI）：由起始主體提供存取控制資訊（Access Control Information, ACI），也就是相關存取資源的授權資訊。

（2）目的受體的存取控制決策資訊（Target ADI）：提供目的受體所擁有的資源可被存取權限。

（3）存取請求決策資訊（Access Request ADI）：執行存取請求時所需之ADI。

（4）存取控制政策規則（Access Control Policy Rules）：是由ADF的安全領域授權機構制訂規則，提供ADF提以執行存取決策判斷。

（5）其他控制資訊（Contextual Information）：其他控制資訊是ADF讀取ADI或存取控制政策時可提供更詳細的解釋，以做出正確的決策判斷，這些資訊包含起始主體的位置、系統存取的時間及使用

特定的通訊路徑等資訊。

（6）被保留的存取控制決資訊（Retained ADI）：此區是提供ADF的安全領域授權機構對內部管理存取控制工作使用。

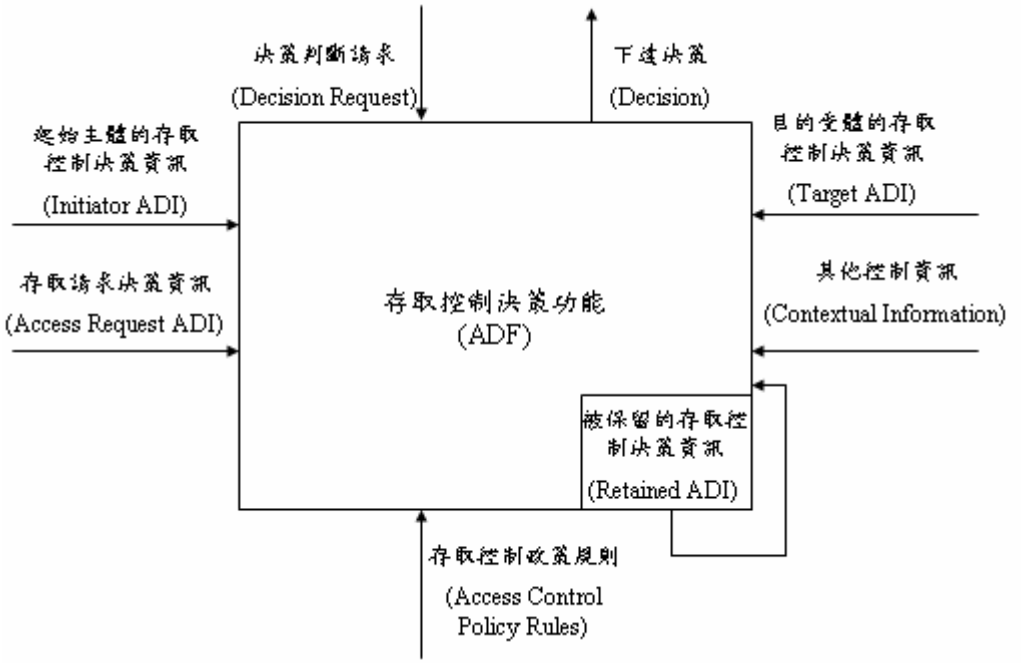


圖2-13 存取控制決策功能所需資訊輸入示意圖

資料來源：ISO/IEC,” ISO/IEC 10181-3 Information Technology – Open System Interconnection – Security Framework for Open System: Access Control Framework,”1996.

第三章　授權管理模式分析

3.1 傳統授權模式

傳統的授權模式主要是以存取控制（Access control）為主要方法，也就是預設已通過身分鑑別的合法系統使用者，在合法的範圍內執行管理者所授予的資源的存取，主要的目的是限制合法系統使用者在系統操作活動中對資訊物件的管制。

在執行存取控制的方法上，通常是由安全管制程式向授權資料庫(Authorization Database)查詢，以取得使用者所授予的最高與最低存取權的類別，據以判定通行權的依據。而授權資料庫是由安全管理者（Security Administrator）依據組織內的安全政策所制訂，並且負責管理與維護內部資源的機密性（Confidentiality）、完整性（Integrity）及可用性（Availability）。

一般使用權限的設計管制，在文獻中已提出的方法可分為三大類[19]：

(1)任意性的存取控制(Discretionary Access Control, DAC）。

(2)強制性的存取控制(Mandatory Access Control, MAC）。

(3)基於角色的存取控制(Role-based Access Control, RBAC)。

以下分別說明(1)任意性的存取控制及(2)強制性的存取控制的特色及基本設計概念，(3)基於角色的存取控制於 3-2 節說明。

3.1.1 任意性的存取控制(Discretionary Access Control, DAC)

又稱為自由裁量存取控制或開放式存取控制，DAC 是對資訊或物件(Object)屬個人使用者所擁有，對於該資訊或物件的授予使用權，完全取決於個人的自由裁量方式管理，讓使用者依自主性訂定對資源或物件的實行要求，系統管理者並不會採強制性的約束。簡單的說就是使用者可以按自己的意願對系統的參數做適當修改，以決定將自給所擁有之資源存取權限授權予其他使用者，亦即一個使用者可以有選擇地與其他使用者共用他的資源，使用者有自主的決定權。例如：學生所擁有的個人資訊設備、文件、各類書籍及物品等，可依個人的喜好、心情及交情的深淺授予不同人不同的使用權限及使用期限。

一個安全的作業系統需要具備存取控制機制，它基於對主體及主體所屬的群組的識別，來限制對受體的存取，並且校驗主體對受體的存取請求是否符合存取控制規定來決定對受體存取的執行與否。任意性的存取控制是指主體可以自主地（也可能是單位方式）將資源存取權，或資源存取的某個子集合授予其他主體。

任意性的存取控制系統，存取資訊儲存於存取控制矩陣（Access Control Matrix）中。控制矩陣的每行表示一個主體，每一列則表示一個受保護的受體，而矩陣中的元素，則表示主體可以對受體的存取模式，例如從表 3-1 中 USER-A 對 FILE 1 有寫入（Write）的權限，為一合法的行為，若 USER-A 所發出的請求為讀取（Read），則此為一不合法的請求行為，系統將予以拒絕。

表 3-1 存取控制矩陣

	FILE 1	FILE 2	FILE 3	FILE 4
USER-A	Write	Read	Append	Read,Write
USER -B	Read	Read	Append	
USER -C	Write,Modify			Append

在實際運作上當主體使用的資源不夠完整時，常會造成稀疏矩陣（Sparse Matrix），形成記憶體的浪費，也就是存取控制矩陣，並不是完整地儲存於記憶體，因為矩陣中的許多元素常常為空，空元素將會造成存儲空間的浪費，而且搜尋某個元素會耗費很多時間。所以實際上常常是以存取控制清單（Access Control Lists, ACLs）及能力清單（Capability Lists, CLs）兩種機制來實施[23]。

存取控制清單（ACLs）與存取控制矩陣類似，不同的是將存取控制矩陣以「行」的形式儲存，整個存取控制的型式如同一個個串列一般，在這方法中控制串列是以受體為主軸，即每一個受體對應一串列的主體，該串列所記錄的是此物件進行存取所有主體及每個主體所擁有資訊存取權限，即每一個受體皆能對應至一個列向量，該向量記錄著所有能存取該檔案之使用者及其存取權；此種以受體為主的設計機制，是屬於資源導向（Resource-oriented）的應用方式。此法最主要的缺點為在主體列中，尋找某一特定的使用者時相當浪費時間，或者若主體的權限更動頻繁，則易造成系統管理的負擔；其優點則是對於受體權限管理維護較容易。

以表 3-1 為例，擁有存取 File 1 權限的對象包含 USER-A、USER-B 及 USER-C。在串列中，USER-A 有寫入(Write)的權限，USER-B 有讀取(Read)

，而 USER-C 有寫入(Write)及修改(Modify)的權限。每一個主體列有一鏈結(Link)串列各個主體，形成對 File 1 的關聯，如圖 3-1。

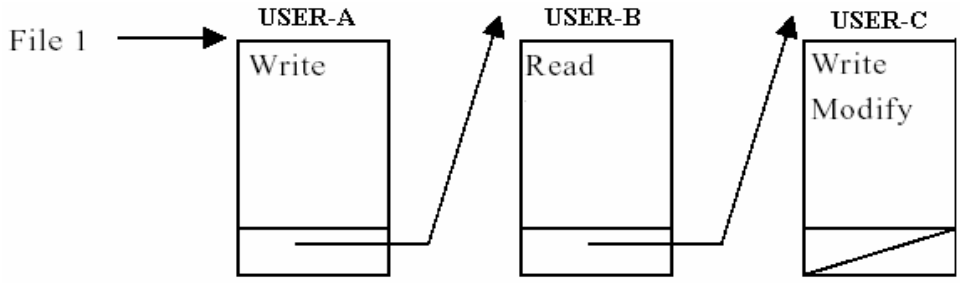


圖 3-1 存取控制清單表示法

能力清單（CLs）和 ACLs 不同的是二者的控制角色對象互換。CLs 是將存取控制矩陣以「列」的形式儲存，即每一個主體皆能對應至一個權限列，該列記錄著此使用者所能存取的檔案以及對於檔案的存取權，即對應一連串的受體所作的存取控制權限，該串列記錄著此主體所能存取的受體以及對於受體的存取權力，在每一個串列之後會有一鏈結，連結所有的檔案以表達對 USER-A 權限資格的關聯，這種以主體為主的管理機制是屬於以使用者為導向（User-oriented）的應用方式。

以圖 3-2 所示，表達 USER-A 所擁有的權限，包含有 FILE 1、FILE 2、FILE 3 及 FILE 4 等，在每一個串列中包含對此檔案的權限資格，比如 USER-A 對 FILE 1 有寫入(Write)的權限，對 FILE 2 有讀取(Read)的權限，對 FILE 3 有新增(Append)的權限，FILE 4 有讀取(Read)及寫入(Write)的權限。CLs 的缺點為修改受體資格的權限，如增加(Propagation)、撤銷(Revocation)、回顧(Review)及更新權利串列時，相當耗時費事，易形成系統管理的負擔；優點則是主體使用者只需讀取權力清單一次，就可執行受體資訊的權限，對於使用者權限管理維護較容易。

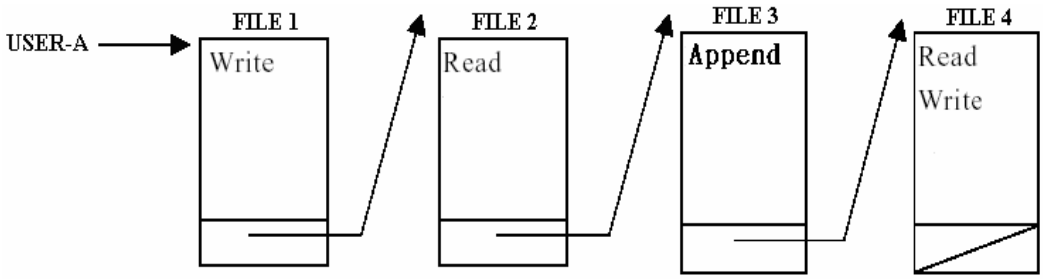


圖 3-2 能力清單表示法

DAC 是採取對個別的使用者主體執行受體資訊的權限管制，所以 DAC 又可被稱為基於個體識別（identity-base）的存取控制管制政策[13]，受管制的權限包括了讀、寫、執行、擁有等基本操作。使用者主體每一次的執行操作要求，必須被系統設定的授權資訊所管制，執行權管制系統根據預設的授權模式，查核操作授權的合法性。DAC 並不會牽涉到其他共同擁有者資訊，而且權限易於轉移，適用於自主性運用權限的環境。優點是授權管理符合組織的需求且可以彈性的運用於不同的資訊系統，缺點是一旦主體取得某一物件擁有者的許可權後，便無法對後續的行為約束，因此可能引發一些無法事先控制的情形[23]，例如：某一合法使用者可以任意執行一程式來修改他擁有的檔案存取控制資訊，而作業系統無法區分這種修改是使用者自己的操作，還是惡意攻擊的特洛伊木馬的非法操作。

3.1.2 強制性的存取控制（Mandatory Access Control, MAC）

強制性的執行管制，對於被使用的資訊或物件執行要求，是以使用者或程式（主體）與該資訊或物件(檔案，目錄或周邊設備)（受體）之間由系統角度給予主體及受體一個安全等級作為依據為基礎，並不可因個人因素而隨意的改變，以達到系統安全政策的規定。安全等級是強制性的規定，它是由系統管理者，或者是作業系統根據限定的規則確定的，使用者或使用者的程式不能加以修改。如果系統認為具有某一個安全等級的使用者不適存取某類資源，那麼任何人（包括資源的擁有者）都無法授予該使用者具有存取該資源的權限。

經由加強不可逾越的存取限制機制，系統可以防止某一些類型的特洛伊木馬的攻擊。MAC 適用於機密性需求較高的單位，其運作方式是將組織所屬的受體資訊，根據重要性的不同給予分類，並賦予不同的安全標籤（Security Label）來標示分類結果，最後系統根據之前對受體資訊的分類結果提供不同的管制與保護的服務。因為 MAC 是以資訊的機密等級作為存取控制的決策依據，又可稱之為基於規則（rule-based）的存取控制管制政策。MAC 在安全標籤的分類上共區分為五種：(1)極機密（Top Secrt、簡稱為 TS）、(2)機密（Secrt、簡稱為 S）、(3)密（Confidential、簡稱為 C）、(4)限制（Restricted、簡稱為 R）(5)普通（Unclassified、簡稱為 U），圖 3-3 描述了這五種安全標籤的次序關係，其中 TS＞S＞C＞R＞U。因為存在著這種次序關係，因此，被貼上安全標籤的受體資訊在某一程度上代表著所在的機密等級位置[17,23]。

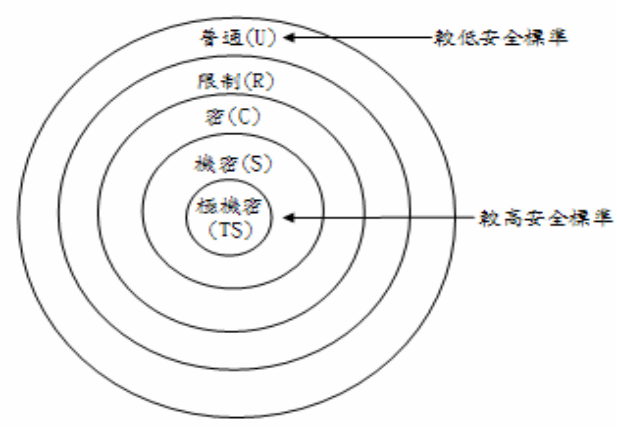


圖 3-3 安全標籤等級的次序關係

資料來源:劉興華,”執行權管制系統的理論性架構設計,”國立交通大學資訊管理研究所, 博士論文 1999.

強制性存取控制方式對使用者執行嚴格的限制，但是，系統為了防範特洛伊木馬，必須要這麼做。即便是不在存在特洛伊木馬，強制性存取控制也可以防止使用者無意或蓄意的操作時，洩露機密資訊。一般強制性存取控制採用以下幾種方法：

(1)限制存取控制:由於任意性的存取控制方式允許使用者執行程式來修改自己擁有資源的存取控制表，為非法者帶來可乘之機。因此，DAC 系統不提供此功能，在 DAC 系統中，使用者要修改存取控制表的唯一方式是由中央控制系統決定。

(2)程序控制：在通常的電腦系統中，只要系統允許執行使用者程式，就沒辦法杜絕特洛伊木馬攻擊。但可以對其存取過程中採取某些預防措施，這種方法稱為程序控制。例如：警告使用者不要執行系統目錄以外的任何程式等等。

(3)系統限制：由系統依資源安全等級，自動完成存取功能限制。

3.2 以職位為基礎之授權模式(Role-based Access Control, RBAC)

美國國家標準技術局提出了以「職位」(Role)為基礎之授權管理模式，簡稱 RBAC[14]，此模式最大的特色，就是能與企業組織現行架構及相關內部管理政策所設計的資源管理模組。與傳統授權方式相異之處，不再是以使用者為主體(Subject)，而是符合企業組織階層之職位為主體，如此設計之授權模式，不但符合企業的實際架構，便於內部授權管理系統之設計，更增加實務上對組織管理的彈性，降低了管理的複雜性。

RBAC 模式主要的管制對象是以「職位」及被授予職務之「權限」而設計，組織依員工之專長、工作層級、部門等不同會授予員工不同的職位，依不同的職位，會有不同的工作職權，執行該職務之工作；而且同一員工能在同一時間，同時擁有多個不同的職務，在不同的場合扮演不同的身分，此模式可以彈性配合企業的管理政策，以達到資訊系統的存取控制[15]。

R. Sandhu 等學者[16]於 1996 年發表於 IEEE Computer 期刊的“Role-Based Access Control Models”文章將RBAC 以模組的方式來表達。文章中將RBAC分成$RBAC_0$到$RBAC_3$四個模組，其中$RBAC_0$是基本模組，透過職位來串連使用者與權限之間的關係；$RBAC_1$ 表達的是職位之間的層級關係，稱為階層式RBAC，一個職位可同時與多個職位存在多個層級關係；而$RBAC_1$及$RBAC_2$為基本模組的衍生模組，$RBAC_3$是集合$RBAC_0$、$RBAC_1$及$RBAC_2$元件的綜合模組，如圖 3-4 所示。

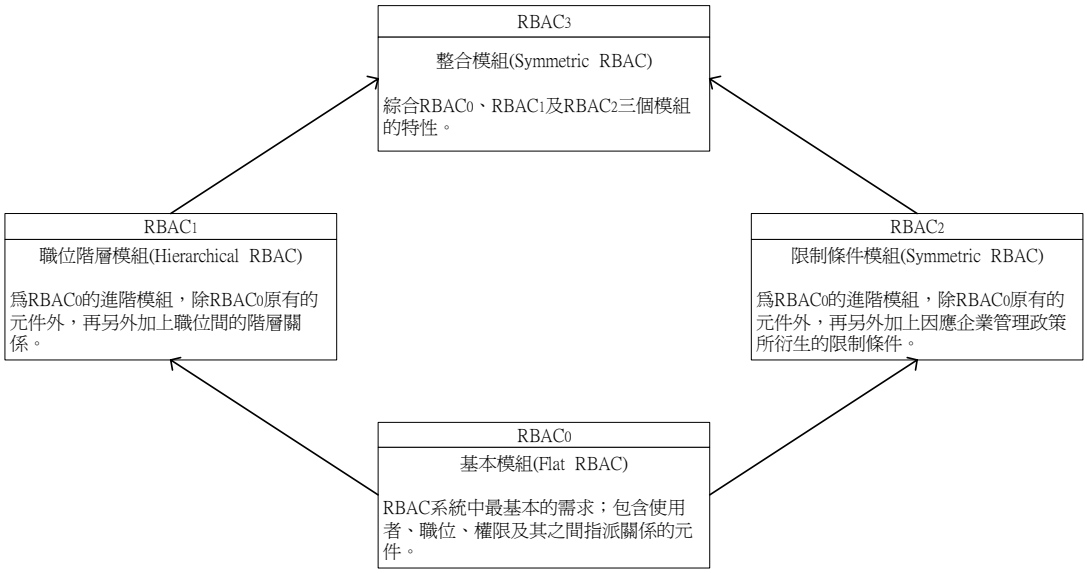


圖 3-4 RBAC 模組架構

資料來源：R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” *IEEE Computer*, Vol. 29, No. 2, pp. 38-47, February 1996.

3.2.1 基本模組（Base model）－$RBAC_0$

$RBAC_0$是由「使用者」、「職位」及「權限」三項元件所組成的（如圖 3-5）。$RBAC_0$是使用者透過職位的指派來使用權限(permissions)，並且使用者與職位、職位與權限都為多對多的關係，這表示同一使用者可以擔任多個職位，例如：採購部經理可兼職採購人員，而採購人員可以由多人擔任。且同一權限也可為多個職位來使用這樣的做法和過去以使用者為主體的方式很不一樣，RBAC是利用職位來結合(binding)權限，和過去用使用者來結合權限有很大的不同，這樣的做法不會因使用者的變動而造成授權的變動。

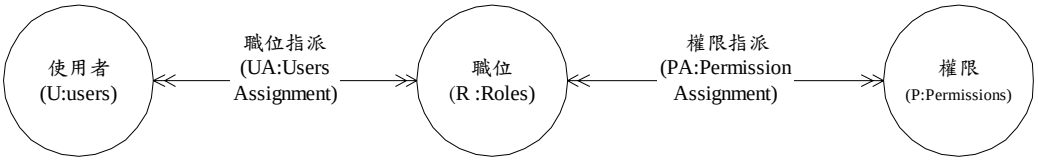


圖 3-5 基本模組–$RBAC_0$

資料來源：R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” IEEE Computer, Vol. 29, No. 2, pp. 38-47, February 1996.

3.2.2 職位階層模組（Role hierarchies）－$RBAC_1$

$RBAC_1$模組是將$RBAC_0$基本模組再加上「職位階層」的概念，這部分是與企業組織中職位階層觀念相同；在此模組中較高層的職位可利用繼承的特性，來行使較低層職位所擁有的權限，如此可簡化職位與權限間的指派關係（如圖 3-6）。

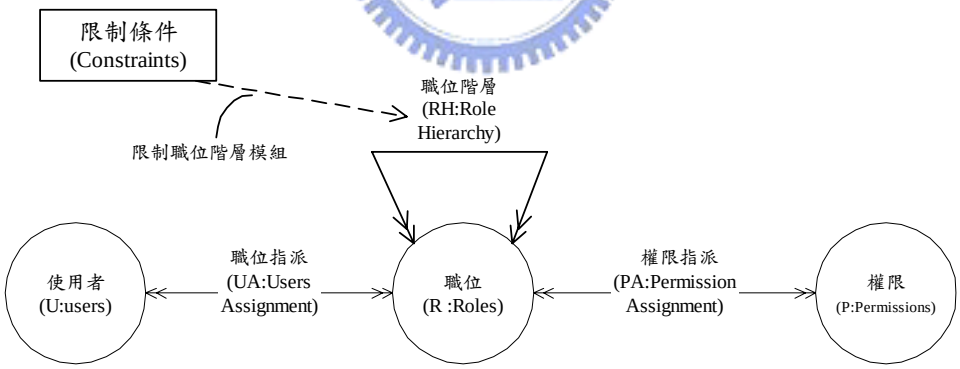


圖 3-6 職位階層模組–$RBAC_1$

資料來源：R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” *IEEE Computer*, Vol. 29, No. 2, pp. 38-47, February 1996.

在職位階級模組中，除了基本模組的組成元件，還包括「職位與職位間為多對多關係」元件，即一個職位可同時與多個職位存在多個層級關係，例：襄理與一般職員存有上對下的關係，而與經理有下對上的關係。另外在此模組中還將職位階層的模式分為以下兩種：

(1)一般職位階層模式(General hierarchical RBAC)：上級主管(senior role)可以繼承下屬(junior role)的所有的權限，沒有任何限制條件。

(2)限制職位階層模式(Limited hierarchical RBAC)：限制上級主管所可以繼承的權限範圍，當管理政策需要調整時可以加以限制，例如：會計部門的主任可以繼承出納人員的職權來執行銀行的領款，但是為防止會計主任自行簽發傳票造成舞弊的情事，為避免此種狀況，此時其對權限的繼承應有所限制，僅給予會計主任對出納的業務有審查的權限。

3.2.3 限制條件模組（Constraints model）－$RBAC_2$

$RBAC_2$是將$RBAC_1$再加上「職位指派」或「行使權限時」的限制條件，主要是使企業內部的管理政策藉由此模組實現；此時由於牽涉到使用者啟動職位的限制，相較於前面的模組，多了「職位期間」(Session)這一個元件，而使用者啟動職務的關係為一對多，表示同一時間使用者可啟動多個非互斥性職位（如圖 3-7）。

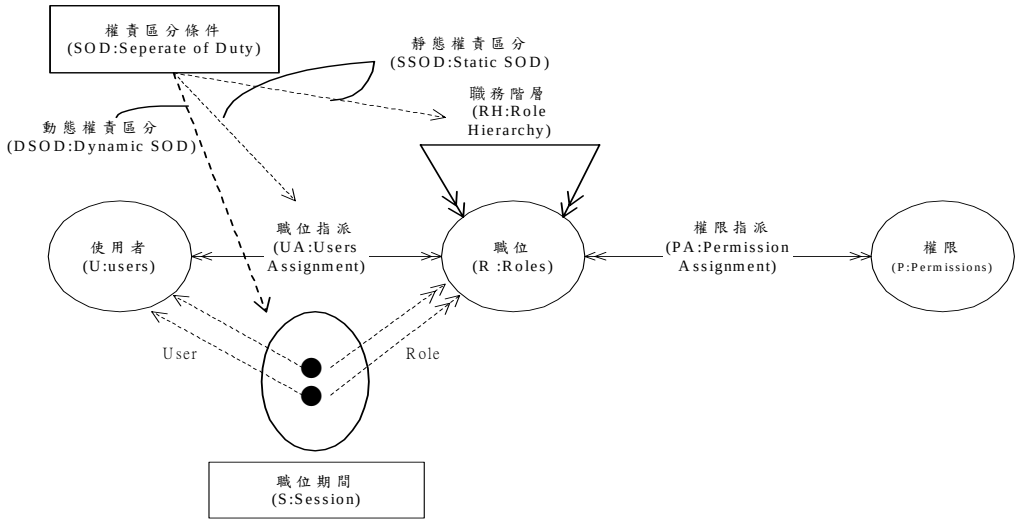


圖 3-7 限制條件模組–$RBAC_2$

資料來源：R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” *IEEE Computer*, Vol. 29, No. 2, pp. 38-47, February 1996.

根據 RBAC 所支援的安全原則，此限制條件最主要是根據權責區分(Separation of duty, SOD)及最小權限(Least Privilege)原則，其目的是要規範擔任職位所需的條件或根據職位所需完成的工作來給予適當的權限，以避免職位不當的指派或使用者有過多的權限而越權，以下分別說明：

(1)權責區分：依據不同的執行時機可分為靜態及動態兩種。

1.靜態權責區分(Static Separation of Duty, SSOD)：針對職位指派給使用者的限制條件，當職位間的關係為強互斥(Strong exclusion)時，表示這些職位不可指派給同一個使用者，要分別將這些職位指派給不同的使用者例如：會計人員與出納人員不得為同一人擔任，以避免舞弊。

2.動態權責區分(Dynamic Separation of Duty, DSOD)：這是指使用者啟動職位的限制條件，當職位間的關係為弱互斥(Weak Exclusion)，表示這些職位可以由同一人擔任，但不可同時啟動職位，故動態權責區分可以讓企業的政策有較大的彈性，例如：Peter 同時被指派擔任會計及出納職

位，若目前 Peter 已啟動會計職位，在這段期間，Peter 不可在啟動出納職位，除非 Peter 結束會計職位的使用。

(2)先決職位(Prerequisite Roles)：在擔任此職位前必須先擔任過的職位；例如：在擔任系統分析師前要先擔任程式設計師，此時程式設計師就是先決職位。

(3)職位擔任人數限制(Cardinality)：職位所需人數的最大及最小值，可視情況需要而定，例如：會計室主任職位只可有一人擔任，而會計人員職位可以有 8 人以上。

3.2.4 整合模組（Consolidated model）－$RBAC_3$

$RBAC_3$是綜合$RBAC_0$到$RBAC_2$的元件，還必須包括對權限指派給職位方面(permission-role assignment)作再檢查，例如：有時因法律或企業政策上的修改，可能對企業職位的權責範圍有所更動，此時職位所賦予的權限也應該有所調整，如圖 3-8 所示。

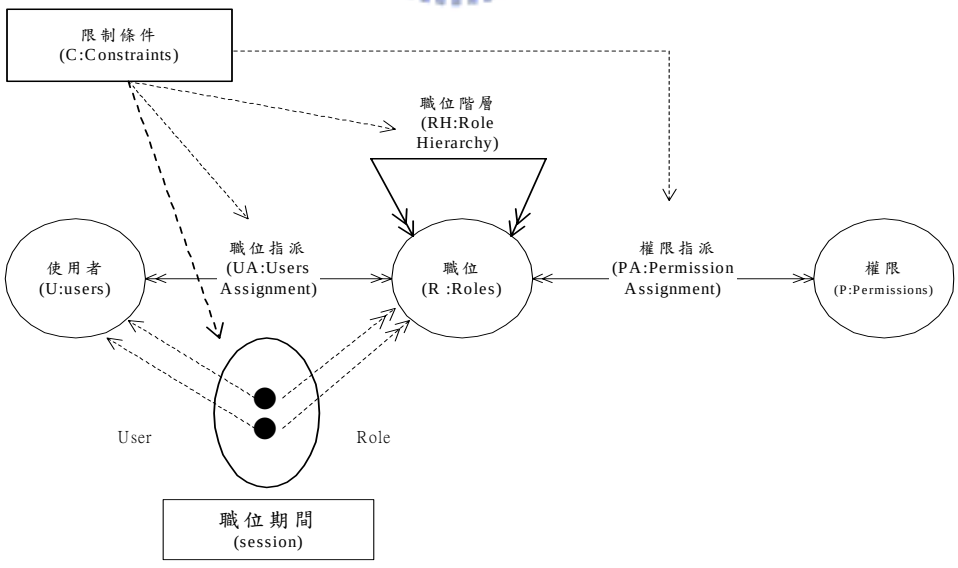


圖 3-8 整合模組–$RBAC_3$

資料來源：R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” *IEEE Computer*, Vol. 29, No. 2, pp. 38-47, February 1996.

3.3 各授權模式優缺點分析

前述所介紹的三種權限存取控管模式中，DAC 及 MAC 是屬於較早期的控制方法，是應用於區域型及中央控制管理系統的作業環境，以解決資源控管問題。而隨著資訊基礎作業環境的不斷改變，網際網路的普及，企業組織的競爭架構無不以全球化為目標，組織營運部門以分散式的架構建置於各國家，以方便管理各項運作業務，這種架構下的跨國組織，若仍是以傳統的資源控管模式設計，一定無法滿足組織的需求。

RBAC 的設計以「職位」為核心，職位對應著相對的工作權限，使用者經由組織分派職位，同時也賦予職位相對各種有限度的資源存取權限，此種資訊架構下，使用者與資源存取權限，是間接的關係，使用者與權限分別維護，如此可以彈性的授予使用者權限及取消權限；而且以使用者而言，一般的組織階均以階層式的設計，如此以職位的編制而言就有高低之分，不同職位也賦予不同權限，這種從屬的關係也是繼承表現，RBAC 支援職位的權限由下至上繼承（例如：員工＞部門主管＞副總經理＞總經理），這是傳統授權模式所不支援的，高階的職位可以擁有下屬職位的權限，這在授權的彈性上是非常便利的。而且對於臨時的指派的任務均可以立即分配權限，任務執行完畢後，即可解除職位，此種控制方式並不是以使用者為主，這是因為人的變動性較頻繁，以職位的管理授權較為快速且安全。在快速異動的競爭環境中　，組織常會有員工身兼數職，此種情形只要賦予員工多個職位，就等於是快速的授權，此種系統管理模式所帶給組織最大的優點是組織的資源管理成本相對降低。

表 3-2 列出各種權限存取控管模式，依各種構面相互比較。

表 3-2 權限存取控管模式分析比較

授權模式 / 分析構面	任意性存取控制（DAC）	強制性存取控制（MAC）	以職位為基礎存取控制（RBAC）
主體	使用者（主體）及物件（受體）	使用者（主體）及物件（受體）	職位
語言介面	指令	指令	語意（Semantic）
權限控制及取得	直接	直接	間接
權限管理	主體自行定義	中央系統一管控	由職位定義
實作方式	ACLs、CLs	安全標籤	$RBAC_0$ ~ $RBAC_3$
分散式授權	無	無	無
權限繼承	無	無	職位繼承
系統管理成本	較高	較高	較低
系統設計導向	資源、使用者	基於規則	基於職位
適用組織類型	使用者自行控管權限需求單位	安全性需求較高單位	架構彈性較高，可適用各類型組織
優點	(1)架構簡單 (2)使用者自行管理權限	(1)集中權限管理安全性高且容易管理 (2)以安全等級件為授權限基礎	(1)權限管理簡化 (2)有效的授權 (3)權限分工明確 (4)符合最小權限
缺點	(1)浪費系統空間 (2)主體與受體異動頻繁會使系統管理成本提高	(1)權限修改耗時 (2)無法因應動態或臨時性權限變更(彈性較差)	維護職位與權限之關係，必須建置於不同資料庫

第四章　授權管理基礎架構分析（Privilege Management Infrastructure, PMI）

4.1 PMI 基本概念

以屬性憑證為授權管理之架構稱為「授權管理基礎建設」。ITU-I X.509v4 中對 PMI 定義為「此基礎建設是能夠支援權限管理，而用以支持廣泛的授權服務，並與公開金鑰基礎建設相關連」[3]。PMI 的管理雖是可以獨立於公開金鑰基礎建設之外，然因屬性憑證仍須 PKI 來鑑別其簽發人與持有人的身分，因此在建置 PMI 之前必須先行建置 PKI。

PMI 以授權管理為核心，對資源的存取控制權統一交由授權中心統一處理，PMI 提供使用者和應用程式間授權管理服務，也就是使用者身份到授權的對映功能。PMI 架構元件主要是運用屬性憑證、授權中心及憑證資料庫等用來實現授權憑證的產生、管理、存儲、分發和撤銷等功能。PMI 使用屬性憑證表示授權資訊，經由管理憑證的生命週期實現對授權生命週期的管理。屬性憑證的申請、簽發、登出、驗證流程對應著授權的申請，發放，撤銷，使用和驗證的過程。而且，使用屬性憑證進行授權管理方式使得授權的管理不必依賴某個授權中心，而且利於授權的分散應用。

PMI 中最頂層的單位稱為「最高授權中心」(Source of Authority, SOA)，SOA 的性質與 ROOT CA 類似，為 PMI 中所有權限的最終總管理單位，所有的權限均由 SOA 下放分配到下層的屬性憑證管理中心(Attribute Authority,AA）或終端用戶（End Entity）。AA 必須是權限的持有者（Holder of Privilege），AA 可再授權給下一層 AA 或終端用戶，但 AA 所授與的權限不可比自己所被允許的權限更大（如圖 4-1）。

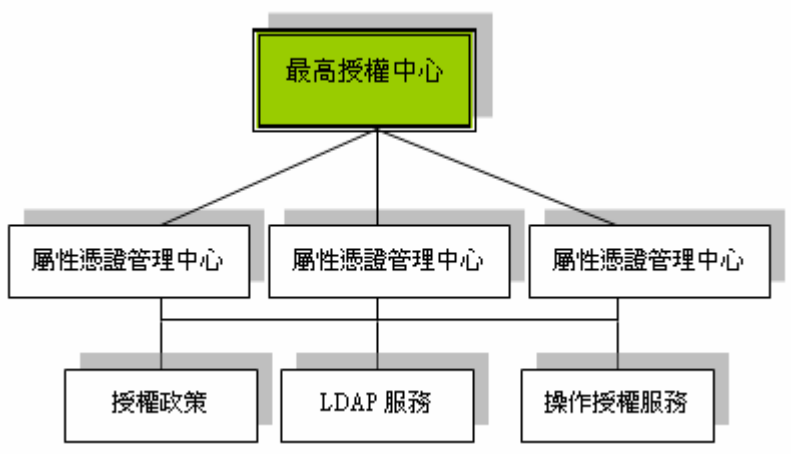


圖 4-1 授權管理基礎建設的基本架構

4.2 PMI 元件（Components）

PMI 的基本架構是由下列元件所組成：

- **屬性憑證管理中心（Attribute Authorities,AAs）**：此機構負責發行及註銷屬性憑證等管理作業，又稱為屬性憑證發行者。
- **屬性憑證（Attribute Certificate,AC）**。
- **屬性憑證使用者（Attribute Certificate Users）**：可以是任何一種實體（Entity）持有或處理屬性憑證者。
- **屬性憑證驗證者（Attribute Certificate Verifier）**：可以是任何一種實體，負責驗證屬性憑證有效性。
- **屬性憑證持有者（Attribute Certificate holder）**：記載於屬性憑證中的持有者（holder）欄位的實體。
- **用戶端（Client）**：發出授權檢查要求的實體
- **憑證儲存所（Repositories）**：儲存憑證及憑證註銷清單之資料庫。

4.3 PMI 運作模型

PMI 的模型在 ITU-T X.509v4 中定義四種，分別是一般模型（General model）、控制模型（Control model）、授權模型（Delegation model）及角色模型（Role model），以下逐一說明各模型基本組成元件及運作方式。

4.3.1 一般模型（General model）

一般模型是 PMI 中架構較簡單的模型，由下列三個組成元件：

- SOA 或 AA
- 授權認證者
- 終端實體權限持有者

運作方式是由 SOA 或 AA 負責指派權限給終端實體權限持有者，終端實體可能是使用者、伺服器、防火牆或其他實體，資源存取認證由授權認證者依授權政策執行，各元件之間的關係如圖 4-2

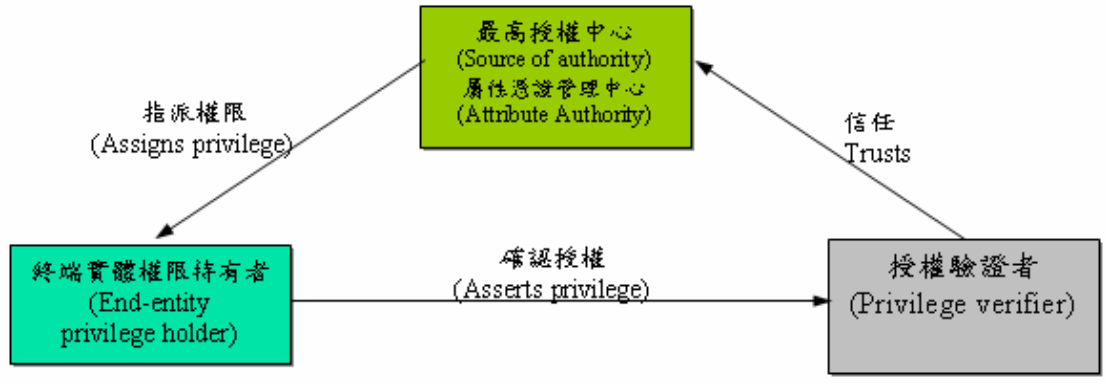


圖 4-2 一般模型

資料來源：ITU-T Recommendation X.509: The Directory-Public-key and Certificate frameworks,”March 2000.

4.3.2 控制模型（Control model）

控制模型是授權管理基礎架構中一種「存取控制」方式的模型，其共有五個組成元件：

- 權限提出者
- 權限驗證者
- 物件的存取方式
- 環境變數
- 相關授權政策

五個組成元件之間的關係如圖 4-3，授權認證者結合各種元件為輸入端並執行授權認證作業，決定使用者是否通過驗證存取資源，是結合所有輸入端的安全政策條件而定，包含授權政策及系統環境變數等條件。

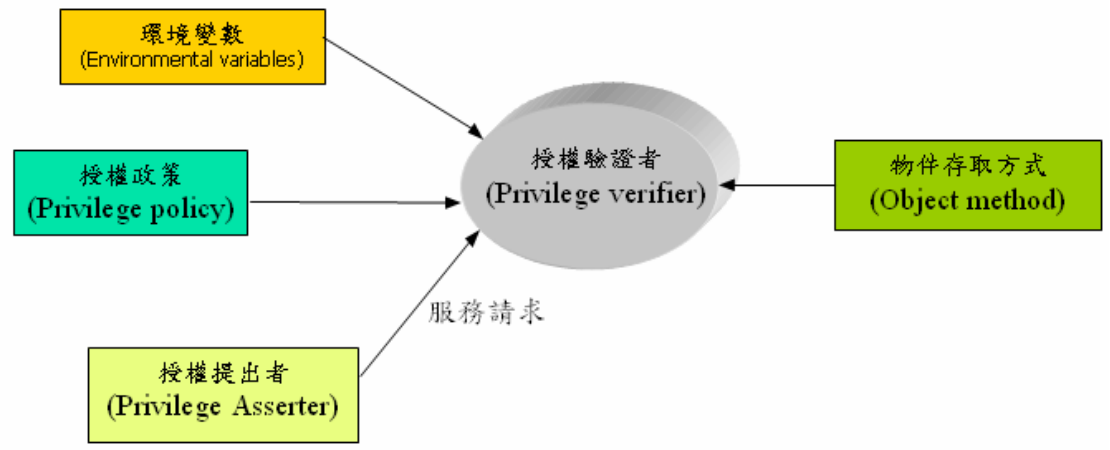


圖 4-3 控制模型

資料來源：ITU-T Recommendation X.509: The Directory-Public-key and Certificate frameworks,”March 2000.

4.3.3 授權模型（Delegation model）

授權模型主要是敘述在授權管理基礎架構中「權限」授予是如何運作的，本模型中有四個組成元件（如圖 4-4）。

- 授權認證者
- 最高授權中心
- 屬性憑證管理中心
- 終端實體權限持有者

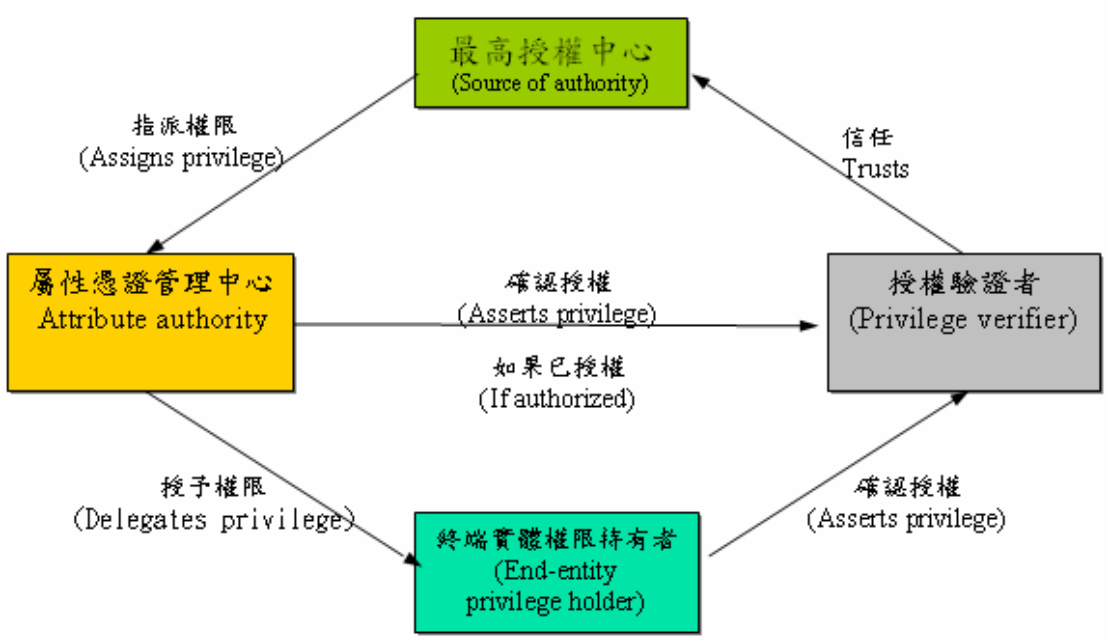


圖 4-4 授權模型

資料來源：Nash, A., Duane, W., Joseph, C., & Brink, D.,"PKI: Implementing and Managing E-Security," Mc-Graw Hill2001.

SOA 是整個架構中簽發屬性憑證並授權予終端實體的初始單位；由於憑證支援代理模式，故 SOA 可將屬性憑證之發放授予同一授權管理區域的 AA，執行代理授權作業，AA 發放屬性憑證必須遵循 SOA 所訂定之規則與

限制，AA 下授之權限也不能大於本身所持有之權限，AA 亦可將授權作業授予另一 AA 執行。

授權認證者必需對 SOA 已授權之資源執行驗證，故必須建立憑證授權路徑（如圖 4-5）完整路徑是從終端實體憑證持有者至 SOA，授權路徑可由屬性憑證或是公開金鑰憑證來建立。

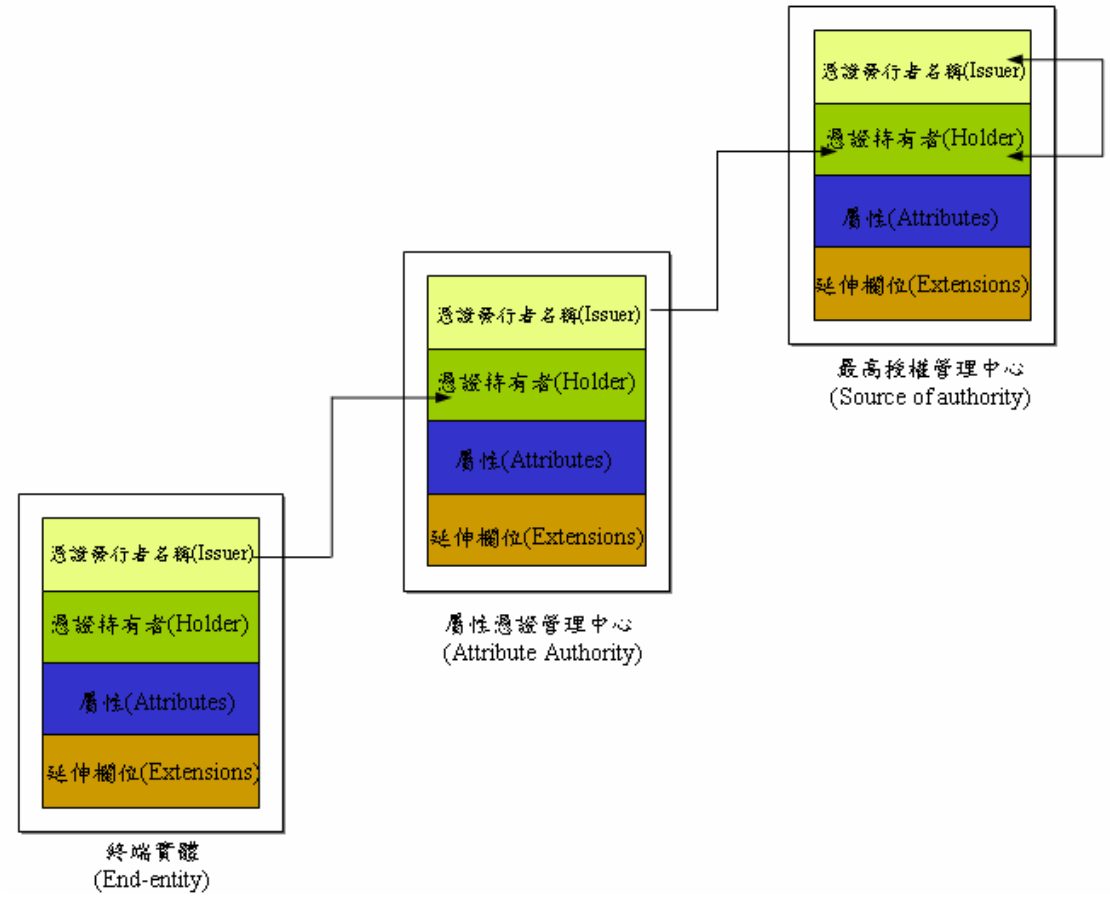


圖 4-5 授權路徑

資料來源：Nash, A., Duane, W., Joseph, C., & Brink, D.,"PKI: Implementing and Managing E-Security," Mc-Graw Hill2001.

4.3.4 角色模型（Roles model）

角色模型並不直接指派權限給個體（Individuals），而是先由 SOA 或 AAs 發行角色指派憑證（Role assignment certificates）（如圖 4-6），個體可以同時擁有一個以上的角色，角色的屬性是定義在憑證中；另外角色的權限則是定義於角色規格憑證中（Role specification certificates），角色規格憑證是一屬性憑證，其權限定義於憑證的屬性（Attribute）欄位，憑證的持有者（Holder）是角色的名稱，而不是個體的名稱，角色指派憑證及角色規格憑證關係如圖 4-7。SOA 發行角色指派憑證時可以使用屬性憑證或公開金鑰憑證，但發行角色規格憑證時只能使用屬性憑證，而且角色規格憑證不行再授權給其他實體（Entity）；發行角色規格憑證及角色指派憑證可以是由不同的 SOA 或 AAs 執行作業[3]。

角色指派憑證中角色的屬性（role attribute）資訊是紀錄於 *roleName* 及 *roleAuthority* 欄位表示，*roleName* 欄位是識別已指派憑證持有者的角色，*roleAuthority* 欄位是識別發行角色規格憑證的屬性憑證管理機構。

終端使用者在提出確認授權時可能提出角色指派憑證，而不是屬性憑證，故授權認證者必須能識別角色所表達的姓名並且知道這角色所分配的權限。

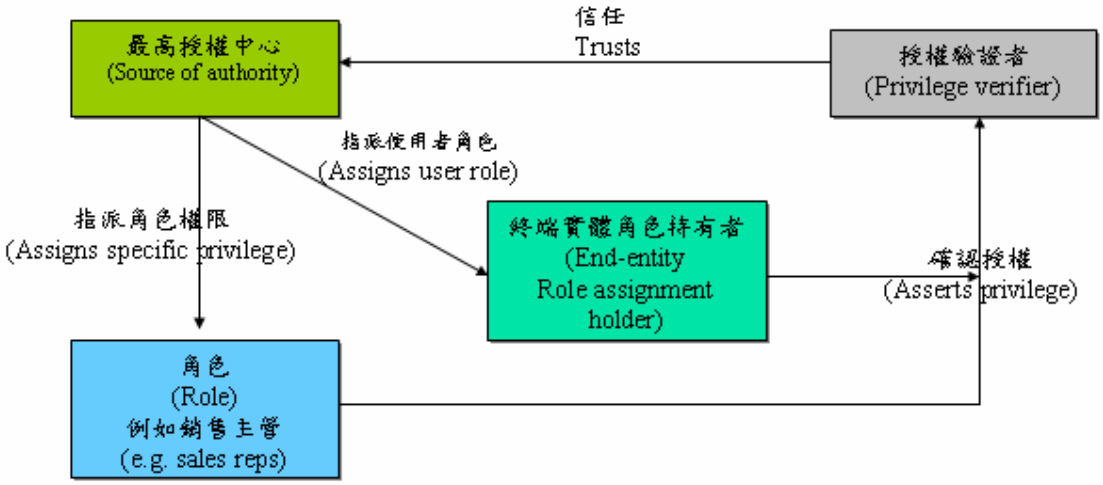


圖 4-6 角色模型

資料來源：ITU-T Recommendation X.509: The Directory-Public-key and Certificate frameworks,"March 2000.

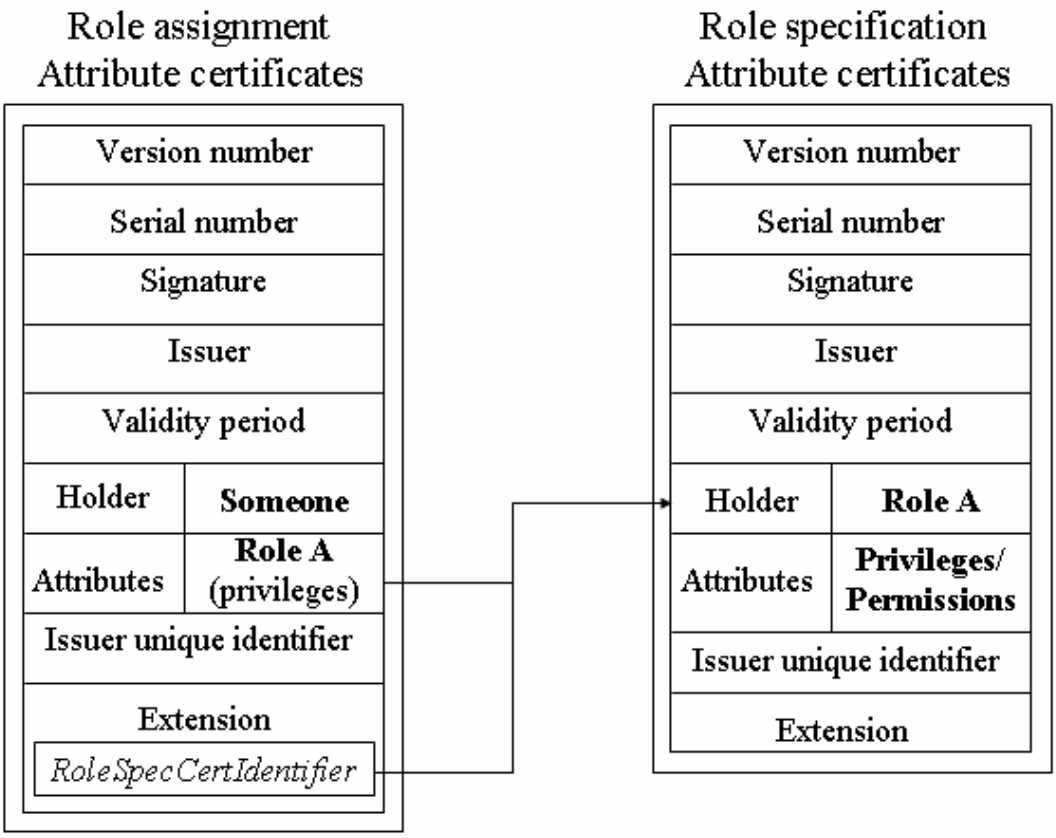


圖 4-7 角色分配屬性憑證及角色規格屬性憑證關係

資料來源：David W. Chadwick , Alexander Otenko, "The PERMIS X.509 Role Based Privilege Management Infrastructrue," Seventh ACM Symposium on Access Control Models and Technologies, pp.135-140, June 2002.

4.4 屬性憑證

屬性憑證是授權管理基礎架構中管控終端實體權限的重要憑證，其經由數位簽章資料架構連結實體所被授予的權限，而另一種達成的方法是運用公開金鑰憑證，於其延伸欄位中定義實體的權限，這兩種定義方式均可實現資源存取控制的目的。

依 ITU-T X.509v4 中定義，屬性憑證乃是「經由屬性憑證管理中心以數位方式簽署，而連結持有人屬性與身分資訊的資料架構」[3]。而「屬性」(Attribute)，在 X.509v4 中並無說明屬性的意義，因此從授權管理的方向來解釋，應係指憑證持有人所擁有之職權、資格、權力或限制，故憑證持有者所附予的資格、下授之權限、或被禁止執行某項行為之限制，都是可以列入本論文所探討的屬性範圍內。

4.4.1 屬性憑證格式

屬性憑證是以連結公開金鑰憑證主體（Subject）的分散結構[3]，單一主體可以擁有多個屬性憑證連結，屬性憑證可以由 CA 發行，亦可由 AA 發行，端視不同的架構需求而定。運用屬性憑證作為權限管理可以提昇整體安全控管服務效能，必定是未來憑證的運用方向。以下解釋說明屬性憑證中各欄位資訊意義，屬性憑證格式（如圖 4-8）：

- 版本（Version）：說明屬性憑證的版本。
- 序號（Serial number）：屬性憑證中心發行此屬性憑證的唯一序號。
- 簽章演算法（Signature）：屬性憑證中心簽發此屬性憑證之演算法。
- 憑證發行者名稱（Issuer）：屬性憑證中心之識別名稱。
- 有效日期（Validity period）：屬性憑證起始日期及終止日期。

- 憑證持有者（Holder）：包含公鑰憑證發行者或公鑰憑證之序號、主體名稱與物件摘要。
- 屬性（Attributes）：包含使用者之屬性或授權資訊，X.509 並無指定授權屬性形態，屬性之定義由使用屬性憑證之系統或應用程式定義。
- 發行者識別（Issuer unique Identifier）：屬性憑證中心之識別號碼。
- 屬性憑證擴充欄位（Extensions）：允許加入屬性憑證附加之資訊，而不需變更憑證架構。

圖 4-8 屬性憑證格式

資料來源：Nash, A., Duane, W., Joseph, C., & Brink, D.,"PKI: Implementing and Managing E-Security," Mc-Graw Hill2001.

4.4.2 屬性憑證 V.S 公開金鑰憑證

公開金鑰憑證主要是在一個安全服務的環境中提供實體身分鑑別機制，而屬性憑證則是紀錄持有者所授予的權限內容，兩者所執行的目的不同，ITU-T X.509v4 建議將身分鑑別與授權資訊區分運作,但在某些情況下是可以結合兩者合一，惟對運用此憑證之整體運作有所限制，表 4-1 整理如下：

表 4-1 結合 PKC 與 AC 授權資訊記載之區別

憑證名稱	憑證發行者	授權資訊	憑證內容	生命週期	憑證發行頻率
公開金鑰憑證(PKC)	憑證管理中心（CA）	紀錄於 subjectDirectoryAttributes 延伸欄位	結合主體(subject)及屬性等資訊	PKC 與屬性相同	發行次數較少
屬性憑證（AC）	屬性憑證管理中心（AA）	紀錄於 Attribute 欄位	屬性	較短	發行次數較頻繁

一般的運作模式公開金鑰憑證是由憑證管理中心簽發（PKI），而屬性憑證則是由屬性憑證管理中心簽發（PMI）。在精簡的架構中可能由結合憑證管理中心與屬性憑證管理中心，但亦可能是分別獨立運作的單位。由上表之區分可明顯的了解結合公開金鑰憑證與屬性憑證合一方法是將屬性資訊記載於公開金鑰憑證的 subjectDirectoryAttributes 延伸欄位中，此情況適用於同一憑證中心發行公開金鑰憑證及也發行屬性憑證，而且公開金鑰憑證與屬性憑證擁有相同的生命期。但在實際的運作情況下通常終端實體的屬性憑證生命週期均比公開金鑰憑證為短，發行的次數也比公開金鑰憑證為頻繁，而且對單一終端實體而言，有可能其擁有屬性憑證是來自於多個不同屬性憑證管理中心授予的，也就是授權的主體與鑑別身分的主體常常是由不同機構執行,因此,結合 PKI 與 PMI 運作模式，乃是在組織中設立憑證管理中心來簽發其組織的公開金鑰憑證，而公開金鑰憑證持有者之權限則由組織中各單位成立屬性憑證管理中心負責簽發

屬性憑證，以作權限的管理。基於上述的原因，區分公開金鑰憑證及屬性憑證各別負責主要的任務是有其必要的。

公開金鑰憑證與屬性憑證雖然都可以執行權限的管理，然與公開金鑰憑證不同的是，屬性憑證上並無公開金鑰的記載。屬性憑證乃是利用身分與權限的結合，來表達證明屬性憑證持有人所被授予的權限。屬性憑證中所記載的屬性雖可達到存取控制的功能，然因未利用公開金鑰作為區別的因素，無法用來鑑別資源存取者的身分，所以仍必須與公開金鑰憑證配合使用[3]（如圖 4-9）。

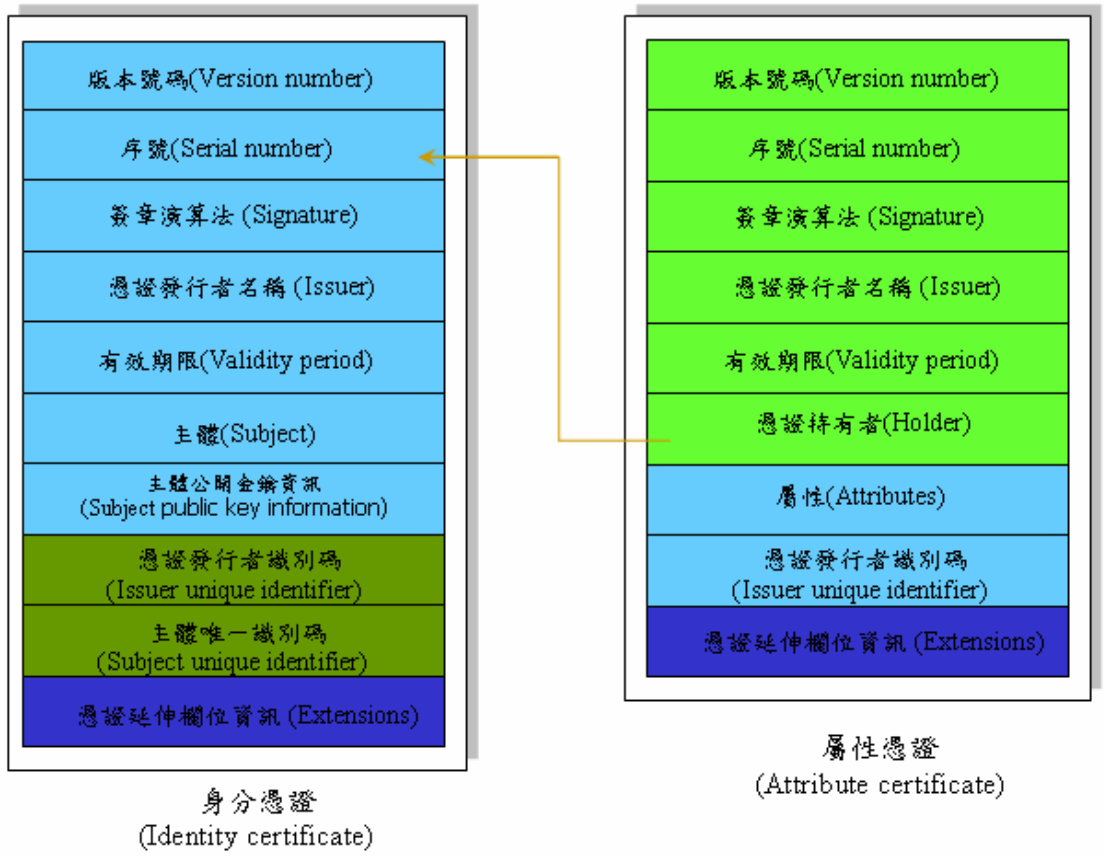


圖 4-9 身分憑證與屬性憑證關係

資料來源：Nash, A., Duane, W., Joseph, C., & Brink, D.,"PKI: Implementing and Managing E-Security," Mc-Graw Hill2001.

4.4.3 屬性憑證的請求（Acquisition）模式

屬性憑證提供一個安全的機制在用戶端（client）與伺服器端（sever）之間交換授權資訊，在不同的執行環境中適用不同的交換模型，IETF 公佈之 RFC3281 提出屬性憑證的請求有「PUSH」及「PULL」兩種模式[4,20]：

- PUSH 模式：假定用戶端傳送屬性憑證給伺服器並要求執行物件存取，當用戶端存取資源的時候，他可以選擇將自己的屬性憑證「推」向伺服器一方，這樣伺服器便可以直接讀取用戶的授權資訊，來決定下一步的操作。這種方式減輕了伺服器的負擔，提高了執行效率。而且，此時伺服器只被告知了它應該知道的資訊，用戶端不必暴露自己持有的其他特權。這種方式適合於用戶端的屬性憑證不是由存取目標授權領域伺服器發行的情況。
- PULL 模式：用戶端存取資源的時候只簡單的向伺服器端證明身份，並不主動出示其屬性憑證。伺服器端自行決定是否需要判斷該用戶的授權，如果需要，則主動向為該用戶端發行屬性憑證的授權機構「拉」回其屬性憑證。這種方式的好處是，在實際運作的時候可以不用考慮用戶端和伺服器端的服務協定（client-server protocol）的不同。該方式適合於用戶端的屬性憑證由要求存取的伺服器端的網域發行的情況。

屬性憑證交換架構包含用戶端、伺服器端、屬性憑證發行機構及憑證目錄資料庫，其相互關係架構如圖 4-10。

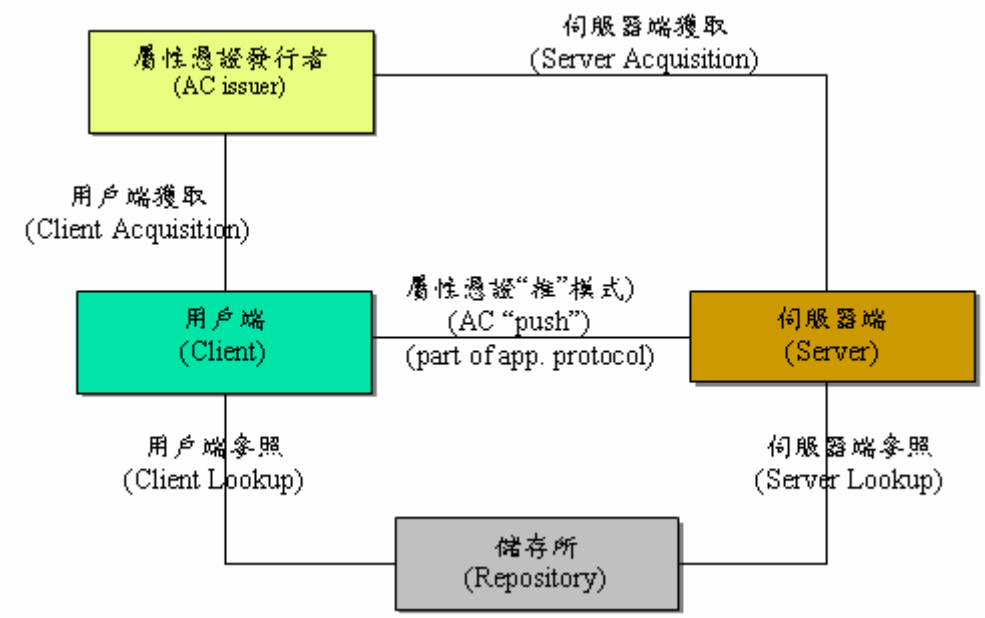


圖 4-10 屬性憑證的交換模型

資料來源：IETF," An Internet Attribute Certificate Profile for Authorization," RFC 3281, April 2002.

4.5 PMI 與 PKI 實體比較

ITU-T X.509 | ISO/IEC 9594-8 提出了在屬性憑證的基礎上構建的 PMI 主要機制是在於授權（Authorization）而 PKI 主要機制是在使用者的身分鑑別（Authentication），因此兩種架構在觀念及建置之實體上有許多類似處，比較如下表 4-2[7]。

表 4-2 PKI 與 PMI 組成元件之比較

觀念 (Concept)	PKI 實體 (PKI entity)	PMI 實體 (PMI entity)
憑證 (Certificate)	公開金鑰憑證 (Public Key Certificate,PKC)	屬性憑證 (Attribute Certificate,AC)
憑證發行者 (Certificate issuer)	憑證管理中心 (Certification Authority (CA)	屬性憑證管理中心 (Attribute Authority,AA)
憑證使用者 (Certificate user)	主體 (Subject)	持有者 (Holder)
憑證連結 (Certificate binding)	主體姓名至公開金鑰 (Subject's Name to Public Key)	持有者姓名至權限屬性 (Holder's Name to Privilege Attribute(s))
憑證註銷 (Revocation)	憑證註銷清冊 (Certificate Revocation List,CRL)	屬性憑證註銷清冊 (Attribute Certificate Revocation List,ACRL)
最高信任機構 (Root of trust)	最高憑證管理中心 (Root Certification Authority or Trust Anchor)	最高授權中心 (Source of Authority,SOA)
從屬機構 (Subordinate authority)	從屬憑證管理中心 (Subordinate Certification Authority)	屬性憑證管理中心 (Attribute Authority,AA)

資料來源：David W. Chadwick , Alexander Otenko, "The PERMIS X.509 Role Based Privilege Management Infrastructrue," Seventh ACM Symposium on Access Control Models and Technologies, pp.135-140, June 2002

公開金鑰憑證與屬性憑證主要的不同點在於屬性憑證中並沒有公開金鑰（Public key），屬性憑證所包含的屬性（例：會員、角色、安全證書及其他授權資訊）運用"Holder"欄位連結。不同的屬性憑證管理中心可以為同一個使用者發行屬性憑證，這只是表示該用戶同時歸屬於這些授權管理範圍內，若使用者為了控制自己內部資源的存取時，甚至也可以自行建立屬性憑證管理中心的角色，向資源存取者頒發屬性憑證。

公開金鑰憑證和屬性憑證的概念如此接近，如何區分它們二者呢？例如：公開金鑰憑證可以理解成護照，它用來證明持有者的身份，一般有效期很長；而屬性憑證是入境簽證，它由不同的管理單位頒發，有效期一般不是很長。為了申請簽證，通常要出示護照，以證明身分，而且其過程相對簡單一些。

4.6 PMI 的優點

與傳統的授權管理模式相比較，基於 PMI 技術的授權管理模式主要有下列三面的優勢：

（1）靈活的授權管理：

PMI 的授權管理模式可以經由屬性憑證的有效期限及委託授權機制，靈活的進行授權管理，竟而實現 DAC 及 MAC 的結合及支援 RBAC 的職位授權，此點是傳統授權管理無法相比的。且可以配合組織的架構彈性建立 AA，可以更符合組織現況需求及未來延展的需求。

（2）區分授權管理與資訊業務管理作業

PMI 管理模式明確的責任分工授權管理者與一般資訊維護者的業務，避免因資訊業務管理人員參與授權管理作業所引發的問題。而且 PMI 可以透過屬性憑證的審核機制來提供對操作授權過程的審核，進一步加強了授權管理的可信度。

（3）多種授權模式的運用

基於 PMI 技術授權管理模式將整個授權管理體系從應用系統管理中獨立出來，授權管理的操作與應用系統相關連，因此，可在不影響原有應用系統正常運作下，實現多種授權模式的運用。

4.7 PMI 與傳統授權模式的關係

傳統的授權模式主要是以存取控制為主要方法，執行管理者所授予的資源的存取，其使用的管理方法就 DAC 而言是採取使用者自行控管機制，系統管理者不會加以強迫管制；而 MAC 是採取中央系統集中控管資源存取的方式，使用者完全遵循組織的安全政策。而 PMI 所採取的授權管理方式與 DAC 及 MAC 就管理功能面而言，有相同的意義，PMI 的架構可以是單獨建置，也可以依組織的架構配合建置，集中式單一 AA 的授權管理型態，其實就是 DAC 所採用的授權管理方式，經由 AA 的授權政策管理組織的資源，使用者依屬性憑證所授予的權限，執行相對的資源存取，就有如 DAC 所採取的管理方式一樣；而 PMI 的架構可以依不同的單位建置下屬的 AA 將授權管理的權限下授，由各授權領域之 AA 依單位特性自行制定安全政策，此種授權管理的方式與 MAC 所採行的方式相同。

由上述的分析本研究可以得到一個結論，就 PMI 的管理面、系統面及技術面來看，傳統的授權管理模式，PMI 已完全支援，而且可以取代，並提供更彈性的操作模式。企業在以網際網路為基礎的電子商務時代中，策略部門的建置規劃均朝向全球化部署，相對的企業的資源也是分散在各地，在如此的經營環境中，對於企業賴以競爭生存的資源，若仍然使用傳統的授權管理方式，是絕對無法有效管理，因此，PMI 模式是一種可以快速導入及適應變動的環境所產生對資源管理的一種系統面技術。

4.8 PMI 與 RBAC 的關係

RBAC 是一種從管理面所制定的資源存取控管模式，其最大的特色，就是能與企業組織現行架構及相關內部管理政策相配合。與傳統授權方式相異之處，不再是以使用者為主體，而是符合企業組織階層之職位（Role）為主體，此授權模式，不但可以依企業的實際架構設計資源控管模式，更可提昇內部組織管理上的彈性，降低了管理成本。

經由上述的分析，我們可以對 PMI 與 RBAC 的關係下一結論：「PMI 是一種實現 RBAC 的系統面的技術」。經由 RBAC 的理論指導，運用 PMI 所提供的職位指派屬性憑證（Role assignment attribute certificates）及職位規格屬性憑證（Role specification attribute certificates），表達出組織中每一成員所應有的工作職權，實現 RBAC 的資源管理精神，而相關 RBAC 所提供的管理機制，則運用憑證各欄位所提供資訊執行管理。而且 PMI 支援分散式的授權管理，此管理模式可以支援各部門依業務特性設計相對的權限控管機制，整體的管制則交由 SOA 負責將分散於各地的部門授權管理結合，使得整體企業得以完全掌握資源的運用情況。

RBAC 的理論提出，使得授權管理的方式更彈性化，而且應用系統的開發亦支援此管理模式。PMI 理論的提出也是 RBAC 的實現技術之一，而且提供比傳統授權管理模式所無法達成的需求，由此得知運用 PMI 所建置的授權管理模式，可依組織的需求及業務特性，設計不同的權限管理方式，靈活管理資源的存取控制。

第五章 授權管理基礎架構之設計 -以空軍總部為例

5.1 組織架構與網路現況分析

網路是目前資訊傳送最快速且最普及的平台，規畫完善的架構可以建構出穩定且高效率的作業環境。有別於網際網路（Internet），國軍基於安全及軍事機密的考量，已自行佈建國軍網路（MINET），提供各級單位作為傳輸的主幹平台，而且是一完全封閉的管理架構，與網際網路完全隔離，各單位廣域網路連接國軍網路主幹，內部則依單位特性及需求不同，分別建置區域網路。

國軍目前已發展成現代化科技兵種，各項資訊連結無不以網路為傳輸介質，因國軍各單位之所保存之機密資訊依不同性質及單位區分，並不是任何人均可接觸的，雖然國軍網路是一封閉架構，但並不保證網路傳輸是絕對安全的，因此，各類資訊依機密等級不同及性質不同，必需區分不同的存取權限及儲存資料庫，唯有授予權限的使用者才可在授權範圍內執行資料存取，而不是以使用者的階級及職位高低為存取的條件，這是基於對機密資訊分類保存所建立的控管政制。

以空軍總部為例，目前所建置之系統授權管控方式多半是提供密碼為基礎（Password base）存取架構，此方式並非最安全的管理，在人員異動頻繁情況下，往往容易造成密碼外洩，而且一種系統就有一組帳號密碼，在管理上也造成使用者不方便，基於整體考量，以建置授權管理架構是最佳的解決方案。

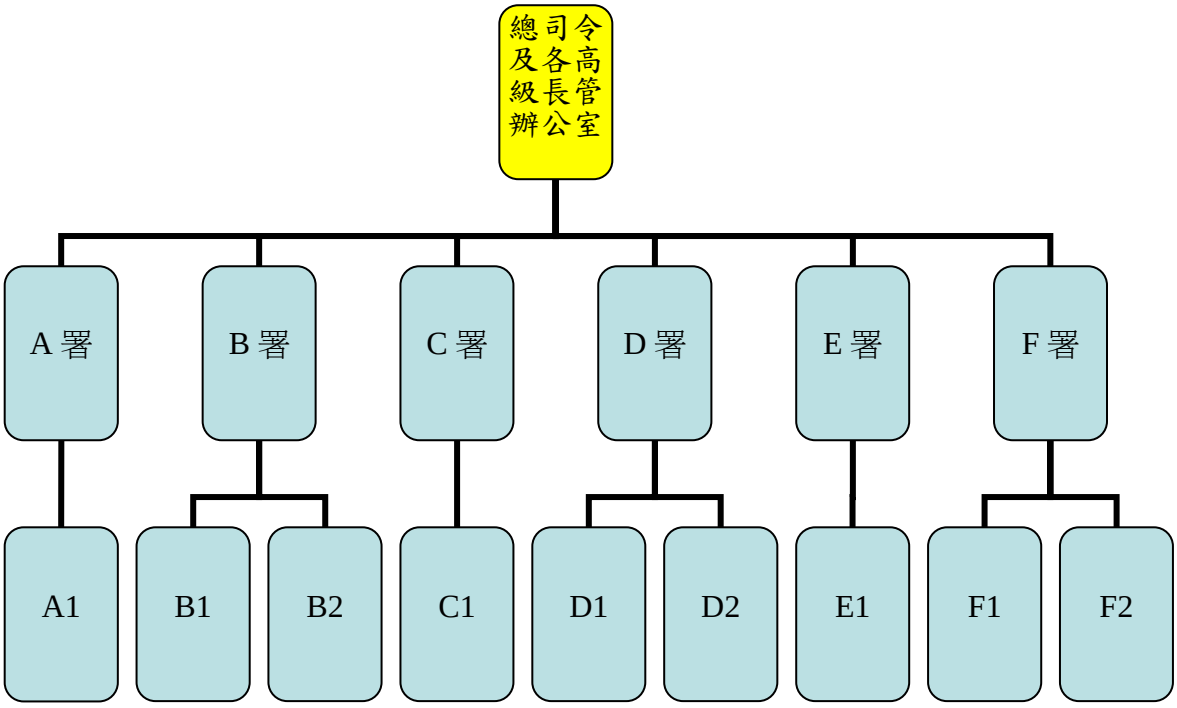


圖 5-1 空軍總部組織架構

圖 5-1 空軍總部組織架構，是一典型階層式（Hierarchical）架構，最高層級指揮為總司令等高級長官辦公室，往下依不同業務功能區分設有各署部門，各署內部又下設有各個組等小門部，分別管理全空軍各項業務。所以總部內部可劃分為三層級，第一層級為決策階層、第二層級為業務管理部門，以及第三級為署內各組部門。

本論文的授權管理架構設計是選擇於空軍總部資訊部門中建立最高授權管理中心（CAFHQ_SOA），管理憑證申請、發放及驗證等各項業務，並於各署部門內建置業務憑證管理中心（Independent Business Certificates Unit, IBCU），由各署自行管理內部或授權其他部門之屬性憑證申請業務、屬性憑證伺服器維護、屬性資料庫維護及授權政策管理等事項，CAFHQ_SOA 負

責督導及協助各 IBCU 完成總部各部門權限控管作業，詳細架構及運作流程我們將於其他章節中說明。

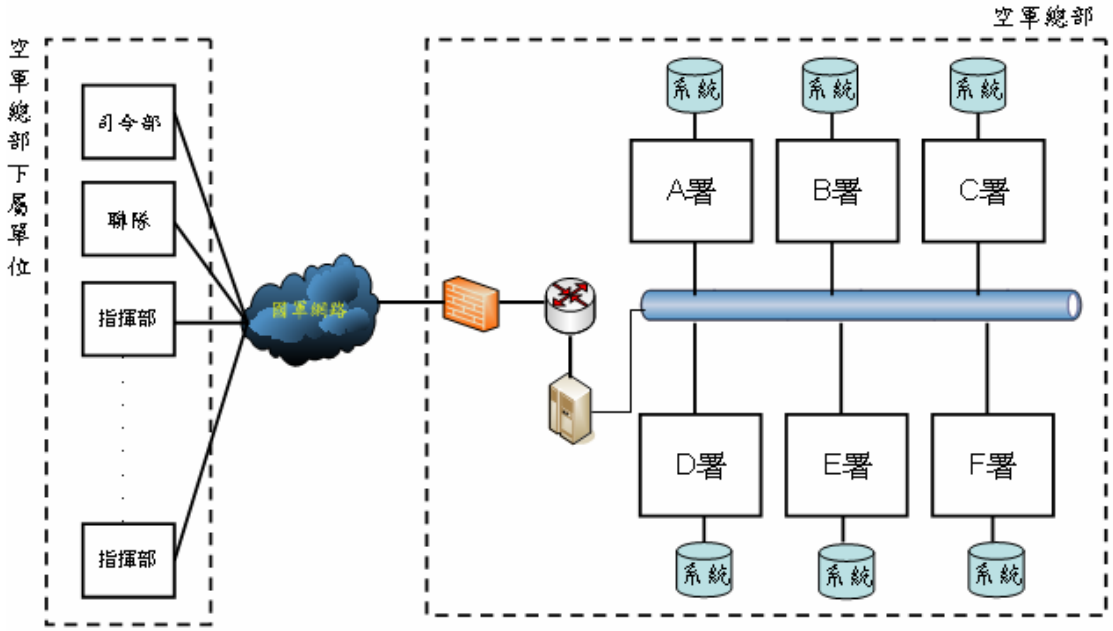


圖 5-2 空軍總部內部網路及系統資源架構

圖 5-2 為空軍總部內部網路及系統資源架構示意圖。其中廣域網路部份以國軍網路與部外各單位構連，總部內部網路以光纖主幹連接至各業務單位，內部各業務單位分別針所管轄工作事項及業管部外單位業務，建立資訊系統，經由國軍網路提供部內及部外使用者操作，各單因業務性質不同，故系統資源並不相互分享連結，另屬於共通性管理資訊需求則是由總部資訊單位負責發展及維護。

5.2 授權管理架構規畫與配置

依 ITU-T X.509v4 中定義授權運作基礎模式，可規歸納出授權基礎架構建立流程原則如下：

（1）配合組織建置之 PKI 架構建立 PMI。

（2）於組織內建立 SOA 統籌管理組織屬性憑證各項業務，並定義組織授權政策。

（3）依組織層級架構及需求建置 AA，並依授權政策管理屬性憑證申、發放及註銷等業務。

（4）下屬 AA 的建立，適組織層級及需求而定。

上述原則必須配合組織內部層級架構及外部所面臨的競爭環境，保有靈活調整架構是非常重要的。此原則非常適合階層式組織架構導入 PMI 模式，以組織架構配合設計 PMI 也是一般快速建立的方法。

5.2.1 授權管理架構規畫

有別於 ITU-T X.509v4 定義架構，本論文提出之 PMI 是以「業務」為授權領域（Privilege Domain）單位，相同業務性質規畫為一「授權連結」，於總部設立 SOA 管理授權各項作業，下屬 AA 的設計並不以單位分支機構為設立點，而是以總部各署其所管理之業務直接對映下屬單位相同業務執行單位為建立點，例如：空軍總部是管理全空軍戰備督導及規畫單位，依所轄業務設立管理部門可區分為 A、B、C、D、E、F 等業務管理部門，部外各單位分別為一獨主子單位（司令部、聯隊及指揮部等），依業務區分設立各部

門，負責執行與總部對映之業務，例如：總部 A 署業務直接管理下屬 A 單位之 A 部門。

此授權架構是以總部設置 CAFHQ_SOA，部內各業務單位設置「業務憑證管理中心」(Independent Business Certificates Unit, IBCU)，部外獨立子單位並不設置單一下屬 AA 管理內部授權作業，而是以總部各署之 IBCU 直接對映各部外子單位內各分項業務部門之下屬 IBCU，整體架構如圖 5-3。

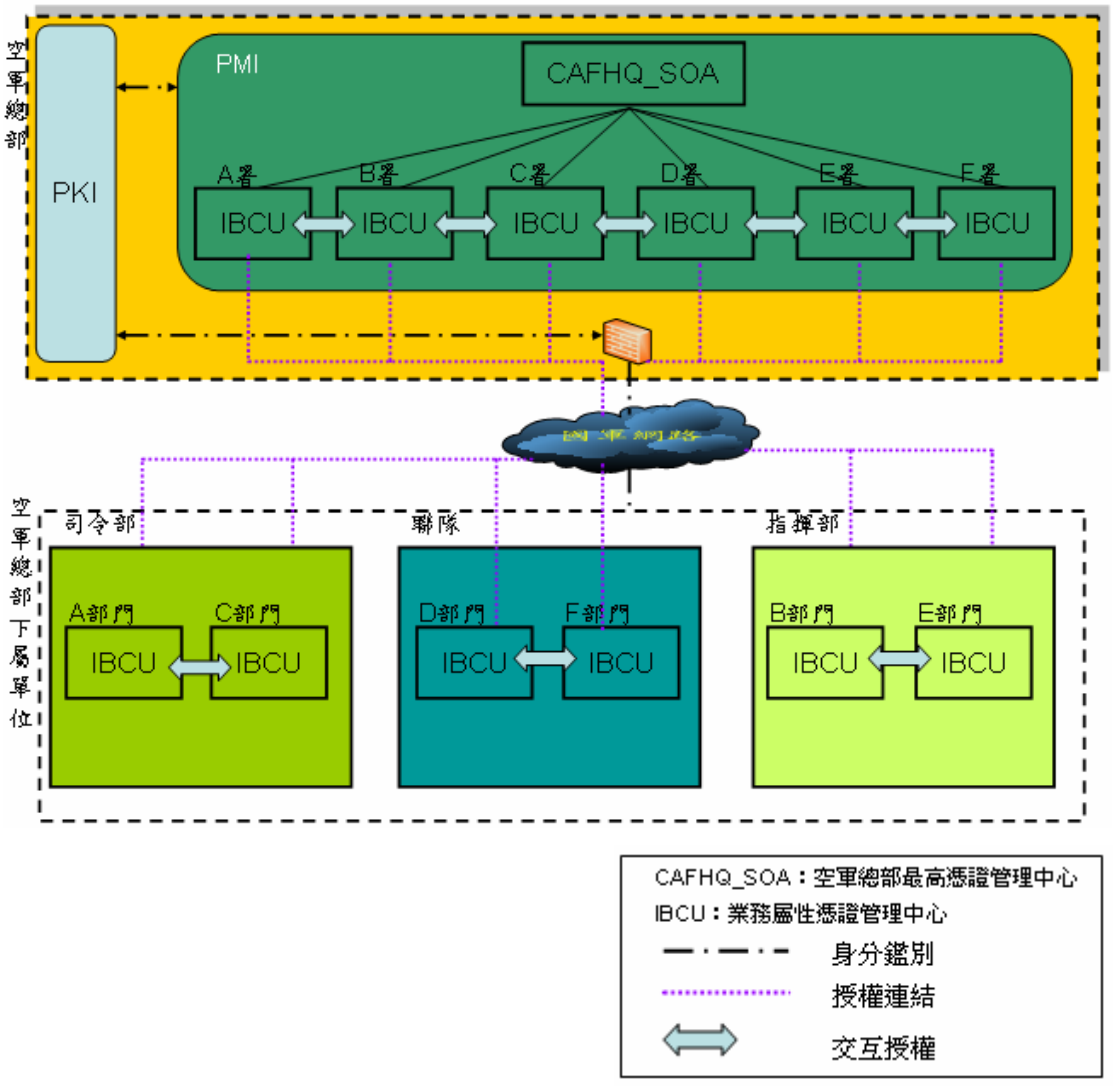


圖 5-3 空軍總部 PMI 架構

圖 5-3 空軍總部 PMI 架構與現行 PMI 最主要之不同在於 CAFHQ_SOA 不並直接以階層方式設置屬性憑證管理中心（AA）而是以「業務」對映相對之下屬單位，例如：空軍總部下屬司令部內有 A 部門及 C 部門，此兩部門業務直屬管轄為空軍總部 A 署與 C 署，故授權連結由 A 署與 C 署直接下授權限於 A 部門及 C 部門，而不是於司令部內設置一屬性憑證管理中心（AA），負責司令部內部授權管理作業，直接由總部之 IBCU 負責管理。此設計觀點是基於不同業務部門所管理之資源具有機密性及獨特性，授權存取此部門資源的對象判斷，只由該部門才可清楚區分出，在此情況下若由單一屬性憑證管理中心決定授權方式與對象，則必定無法符合最小授權原則。而且在授權架構中減少中間層轉換，可以減少下屬單位授權管理作業的負擔，以「業務」為授權連結亦可減少 SOA 的線上授權認證，因為 SOA 只須負責管理內部下授業務授權結點（IBCU），而不必管理部外單位所設置之 IBCU，改由部內各 IBCU 直接管理下屬單位之 IBCU。

與 ITU-T X.509v4 所提出之 PMI 架構，本論文設計之 PMI 架構有下列之優點：

（1）由業務單位直接管控 IBCU 運作，更能掌控屬性憑證之申請、發行及註銷等作業，提昇整體 PMI 運作效率。

（2）有效減輕 SOA 管理工作。

（3）正確授權並符合最小授權原則。

（4）減少 PMI 架構階層，且分層獨立管理，可降低整體建置及管理成本。

5.2.2 授權流程規畫

授權流程是規範本論文所提出之授權架構，使用者（主體）於存取資源前，所必須執行之授權驗證流程（圖 5-4），也就是說明存取系統資源前的每一驗證步驟，惟有通過授權服務請求，系統才會允許進入下一階段，應用系統之存取服務。

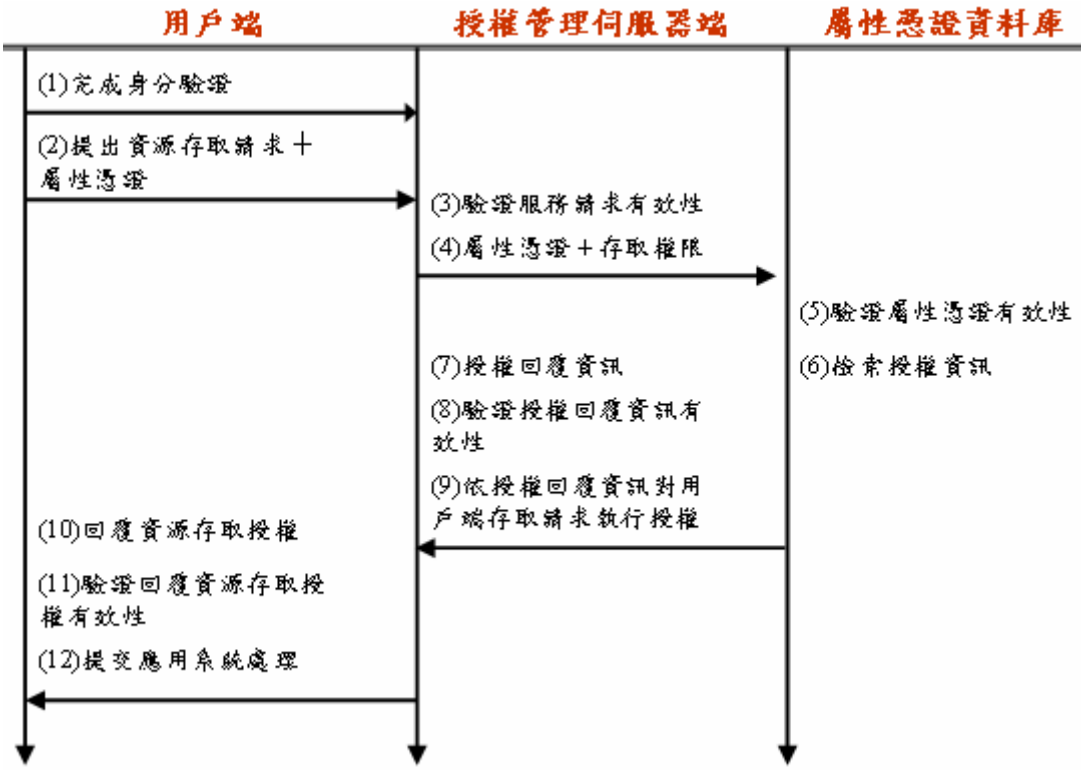


圖 5-4 授權管理流程

授權流程詳細說明如下：

（1）用戶端（Client）執行授權請求前已完成身分鑑別。

（2）用戶端提出資訊存取請求（已完成簽章）及屬性憑證。

$$Message2=Env\{Reg, S_{Client}\}||AC_{Client}$$

（3）授權管理伺服器驗證用戶端服務請求有效性。

（4）授權管理伺服器將用戶端屬性憑證及請求服務的存取授權傳送至屬性憑證資料庫。

（5）屬性憑證資料庫驗證用戶端屬性憑證的有效性。

（6）由屬性憑證中讀取屬性資訊，並檢索相對映授權資訊。

（7）屬性憑證資料庫回傳授權回覆資訊。

Message7=Env{R, $S_{verifier}$}

（8）授權管理伺服器驗證授權回覆資訊的有效性。

（9）依授權回覆資訊對用戶端存取請求執行授權。

（10）授權管理伺服器回覆資源存取授權。

Message12=Env{S_{Res}, S_{server}}

（11）用戶端驗證回覆資源存取授權的有效性。

（12）用戶端將存取資源請求回覆結果提交應用系統處理。

5.3 業務憑證管理中心之設計

5.3.1 授權領域內部與外部屬性憑證設計

本研究針對屬性憑證應用服務架構設計區分為授權領域內部及外部兩部份，對於內部使用者運用 RBAC 理論設計以職位為基礎之授權存取架構，以職位指派屬性憑證(Role assignment attribute certificates)及職位規格屬性憑證(Role specification attribute certificates)記載內部每一成員職位及其職位相對映之權限資訊，以作為內部使用者資源存取控管依據。

外部使用者因無法預先定義其適當的職位(Role)及預判每次提出之資源存取請求項目，故設計以屬性憑證記載外部使用者授權資訊。因各業務單位所保管之資源均具有機密特性，故外部使用者之屬性憑證申請，須同時完成

系統提供之線上屬性憑證申請註冊及離線申請審查機制，以有效管理平行單位、上層級單位及下層級單位之資源存取請求。

本研究提出組織的授權管理模式，在 IBCU 內部使用職位屬性憑證，外部則使用屬性憑證，原因在於組織授權架構全面使用職位屬性憑證定義授權資訊，會增加授權路徑驗證處理的複雜度，而使用者的職位屬性憑證的職位指派（Role assignment）可能會來自多個不同授權領域的 IBCU 賦予，使用者在請求資源存取時，授權驗證的程序須經過多道手續的審核判斷，此種設計在規模較大的組織中使用，整體運作效能必定會降低[3]，故將屬性憑證之運用區分內部及外部之設計可提供下列優點：

(1)運用 RBAC 權限管理原則，符合組織架構設計，而且可以彈性的修改職位與權限間的對映關係，有效降低授權管理成本。

(2)依組織任職人令指派工作職權，可符合最小授權原則，可提昇資源管理之安全性。

(3)區分職位的指派、職位權限定義及資源管理三項作業，明確的分工可提昇整體授權管理效率。

5.3.2 業務憑證管理中心架構

本論文所設計之業務憑證管理中心（Independent Business Certificates Unit, IBCU）如圖 5-5，是整合屬性憑證及相關授權管理功能而設計，對內部授權方式是由各業務單位人事部門負責賦予每一職員所佔的「編制職位」所應有的工作職權，也就是設計以職位屬性憑證來實現 RBAC 的權限控管機制；對外部單位的授權方式並不使用 RBAC 的權限控制機制，而是單純的使用屬性憑證來執行權限控管作業。區分內、外部不同的權限控管，可以

有效的提昇資源維護的安全性，區分授權管理與職位管理，更可以防止因人為的介入，造成不當的授權情形發生。

IBCU 設置原則是以一個「業務單位」為一授權管理領域（Privilege Management Domain）。以空軍總部為例，共區分 A、B、C、D、E、F 等六個不同業務授權管理領域，每一個 IBCU 對上接受 CAFHQ_SOA 管轄，對下負責往下授權並管理相同業務性質之子部門授權作業。

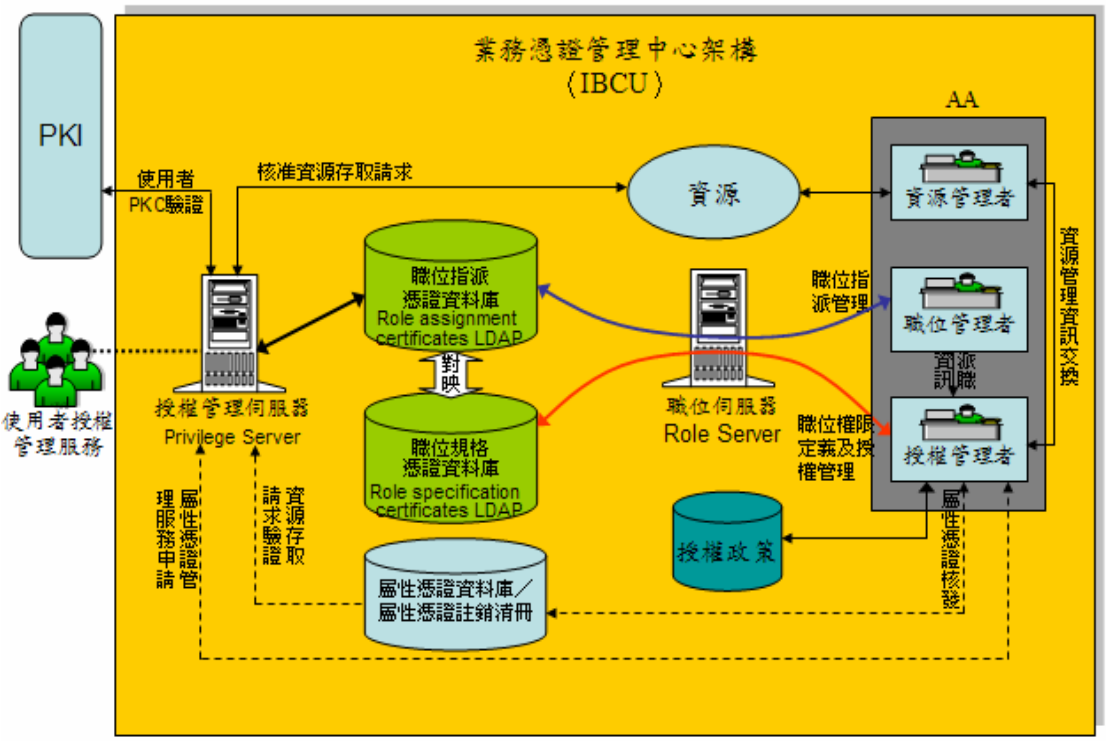


圖 5-5 業務憑證管理中心架構

IBCU 組成元件及內部作業流程說明如下：

一、組成元件介紹：

(一)資源管理者（Resource administrator）：負責管理單位內部各項資源，並定義各項資源使用單位或對象；並將各項資源管理資訊傳送給授權管理者據以定義資源存取規則。

(二)職位管理者（Role administrator）：負責單位職缺管理及指派組織成員系統職位，並執行職位指派憑證資料庫維護作業。職位管理者在本系統所設計的授權管理模式中是一位負有重責的管理者，其是每一組織成員指派職位的負責人，故依 RBAC 的權限管理原則本研究定義出下列工作負責項目：

1.組織成員指派職位資格管理(包含新進人員、離職人員、先決職位管制及職位代理等)。

2.職位的靜態責任區分及動態責任區分。

3.職位階層管理(包含一般職位階層及限制職位階層設定)。

4.職位指派人數的管理。

5. 職位啓動管理。

6.職位生效期間管理。

7.職權調整管理。

(三)授權管理者（Privilege administrator）：負責定義單位內各職位之權限(含權限調整)及職位權限與單位內資源之間存取控制，並執行職位規格憑證資料庫維護作業；依授權政策執行組織外部授權領域之用戶端屬性憑證申請、發行與註銷等各項作業。

(四)職位伺服器（Role server）：負責提供給職位管理者及授權管理者之系統操作介面，並作為職位指派憑證資料庫及職位規格憑證資料庫之管理主機。

(五)授權管理伺服器（Privilege server）：提供用戶端執行內部授權領域或外部授權領域的各項授權管理服務作業（屬性憑證申請、發行、驗證及查詢等作業），作為用戶端執行資源存取請求的作業窗口。

(六)屬性憑證資料庫及屬性憑證註銷清冊（Attribute certificates LDAP/ACRL）：儲存外部使用者之屬性憑證申請及註銷相關資料，此資料庫由授權管理者負責維護，由於屬性憑證之修改或註銷等作業只在 IBCU 中運作，故不會影響其他系統之運作。

(七)職位指派憑證資料庫（Role assignment certificates LDAP）：儲存單位內每一使用者被指派之職位資料，也就是賦予使用者在單位的角色定位，此資料庫由職位管理者負責維護。

(八)職位規格憑證資料庫（Role specification certificates LDAP）：儲存單位內每一職位（缺）屬性資料，也就是此職位所對映的執行工作權限，此資料庫由授權管理者負責維護。

(九)授權政策（Privilege policy）：定義本單位資源存取控制的安全政策，並接受上一層的安全政策指導，所有使用者均須遵守。

二、作業流程簡介：

(一)授權領域內部使用者：

1.每一位使用者經由職位管理者依派職命令賦予系統職位，並儲存於職位指派憑證資料庫，相關資訊傳送給授權管理者。

2.授權管理者依組織授權政策完成職位規格權限定義，並儲存於職位規格憑證資料庫，使用者經職位管理者賦予職位即間接擁有對映的職權。

3.使用者執行資源存取請求時，提示 PKC 及職位指派屬性憑證，由授權管理伺服器鑑別無誤後，於職位規格屬性憑證資料庫中檢索出使用者相對映之職位權限，符合授權者，即同意資源存取請求，反之則拒絕。

(二) 授權領域外部使用者：

1.使用者經由授權管理伺服器線上執行申請屬性憑證作業，輸入相關註冊資訊後完成線上申請程序。

2.使用者離線提出資源存取說明及單位核准公文等相關文件，由授權管理者審查，符合組織授權政策之要求後，核發屬性憑證，並儲存於屬性憑證資料庫中。

3.使用者執行資源存取請求時，提示 PKC 及屬性憑證，由授權管理伺服器鑑別使用者之身分及權限的合法性及有效性，符合授權者，即同意資源存取請求，反之則拒絕。

5.4 授權領域內部職位屬性憑證管理服務設計

IBCU 的運作對授權領域內部使用者主要提供的職位屬性憑證的管理服務如下列所示：

（1）職位屬性憑證的申請

（2）職位屬性憑證的註銷

（3）職位屬性憑證的委託

（4）職位屬性憑證的存取控制驗證

5.4.1 職位屬性憑證申請

職位屬性憑證的申請僅提供單位內部人員，申請過程可分為二階段，第一階段由職位管理者依使用者任職命令賦予系統職位，並將任職資訊線上通知授權管理者設定職位權限，只要職位啟動後，指派該職位的使用者即擁有相對映的權限。第二階段由使用者線上執行基本資料註冊，經授權管理者比對審核職位指派憑證資料庫及職位規格憑證資料庫無誤後，立即啟動權限，使用者完成職位權限申請，由系統發給使用者職位指派屬性憑證，作業流程圖如圖 5-6 所示。

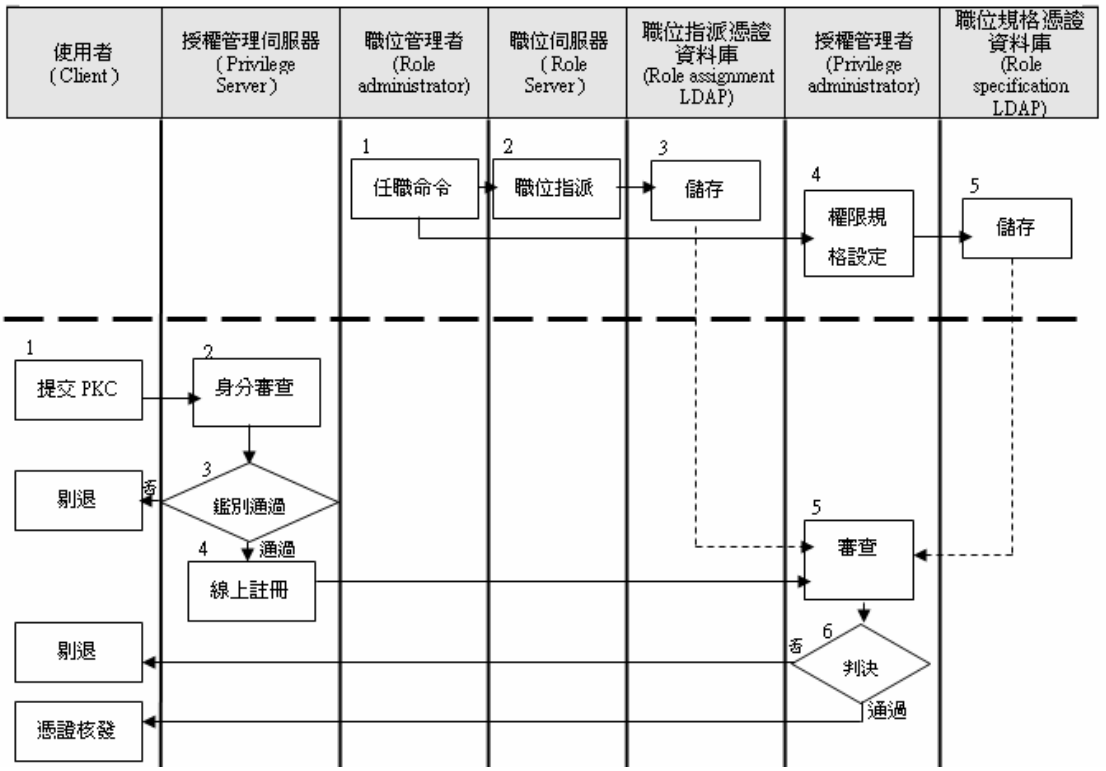


圖 5-6 職位屬性憑證的申請流程

5.4.2 職位屬性憑證的註銷

職位屬性憑證的註銷申請時機為，當使用者離職或組織內部職位異動等情況發生時，由職位管理者依相關人事異動命令解除所賦予之職位，原職位屬性憑證資訊儲存於資料庫中，提供後續管制查詢記錄，並將憑證註銷資訊通知授權管理者終止該職位對映之權限功能，使用者完全不須上線執行申請程序，減化作業步驟。當職位屬性憑證註銷程序完成後即不提供回覆的機制，作業流程圖如圖 5-7。

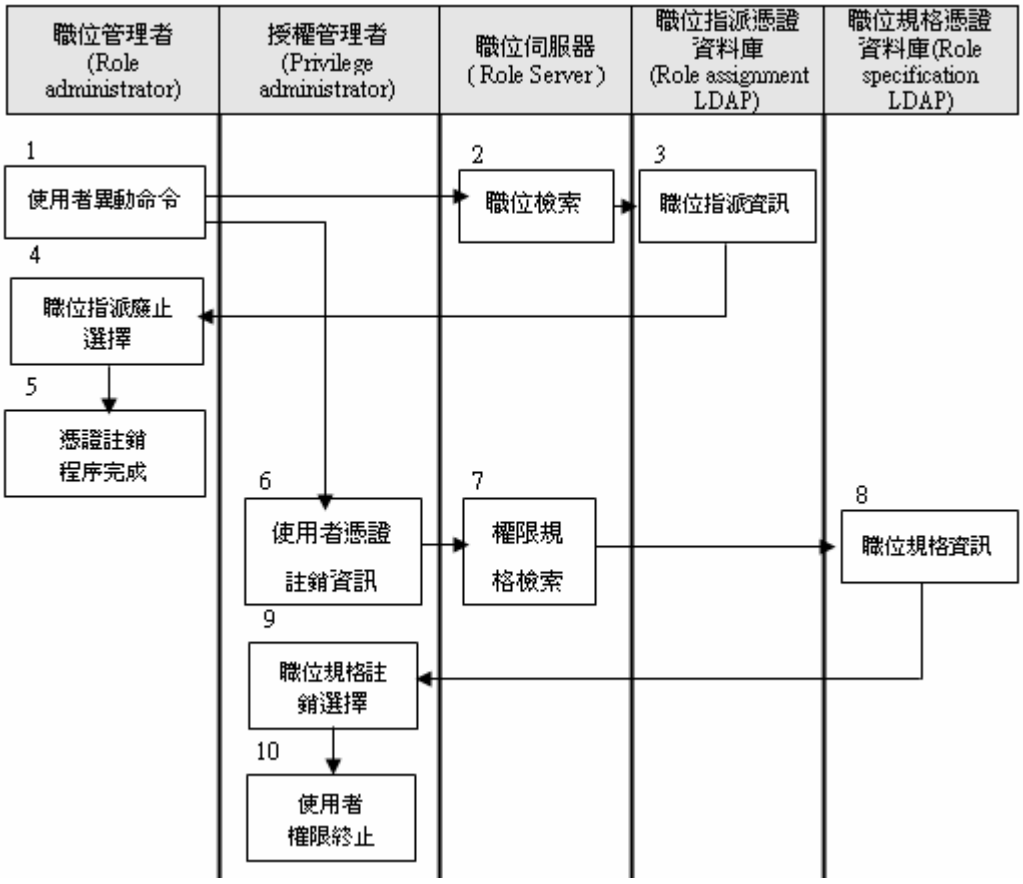


圖 5-7 職位屬性憑證的註銷流程

5.4.3 職位屬性憑證委託

職位屬性憑證的委託是針對「業務代理」而設計之機制，組織職位的代理依內部的政策規定執行，主官（管）或副主官（管）的職務代理必須有行政命令為依據，而一般參謀的職務代理由單位內部挑選代理人，並由主官裁示即可。本作業流程是由委託者提出申請並出示經權責長官核准的代理職位命令，由職位管理者執行設定，將委託者的職位指派給代理者，代理者同時擁有本身的職位權限及委託者的權限。當職位委託期限到期，代理者的代理職權，即自動解除，作業流程圖如圖 5-8。

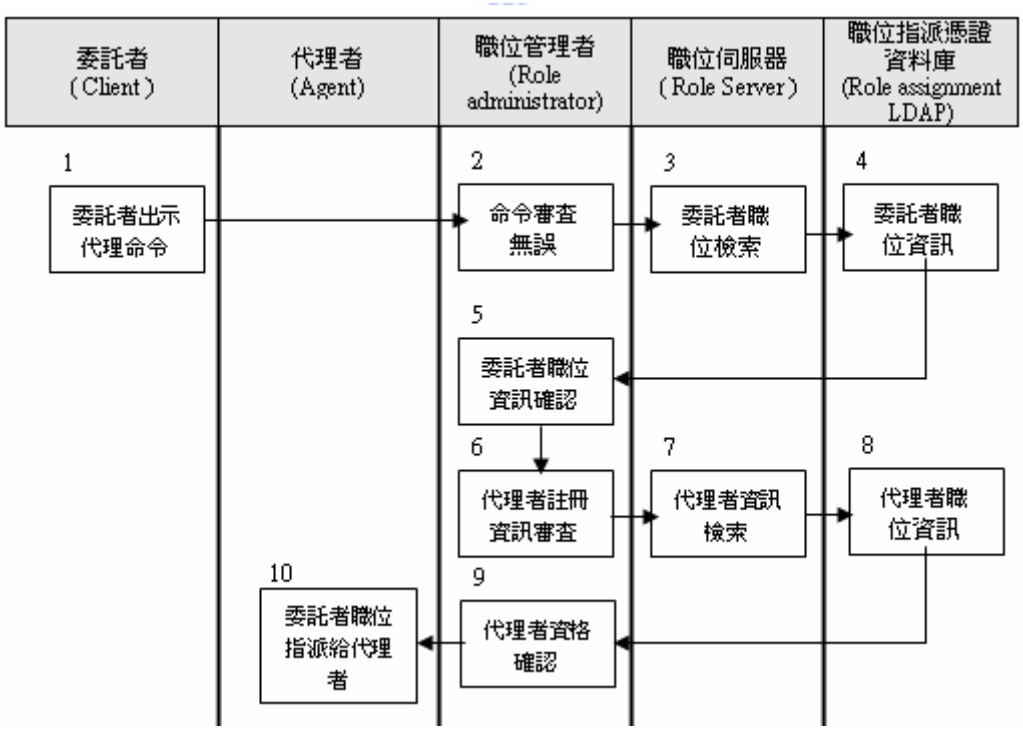


圖 5-8 職位屬性憑證的委託流程

5.4.4 職位屬性憑證的存取控制驗證

職位屬性憑證的驗證是當使用者提出資源存取請求時，為了鑑別使用者的身分及合法的資源存取權限，而執行的一連串過程，可區分為三階段。

（1）第一階段使用者提交公開金鑰憑證（PKC）及職位指派屬性憑證，PKC 驗證使用者的身分是否為本授權領域的 IBCU 註冊合法使用者（只有本授權領域的使用者才會准許使用職位指派屬性憑證，非本授權領域的使用者只適用組織外部屬性憑證）。

（2）第二階段驗證使用者提交的職位指派屬性憑證與 PKC 的關連，運用職位指派屬性憑證的持有者（Holder）欄位之 *baseCertificateID* 驗證與 PKC 的序號（Serial number）欄位是否相同，此驗證手續可以保證職位指派屬性憑證確實屬於 PKC 的擁有者。

（3）第三階段使用職位指派屬性憑證檢索出該使用者職位相對映的權限與使用者提出的資源存取請求是否有超越職權，審核通過，即可同意使用者執行資源存取，作業流程圖如圖 5-9。

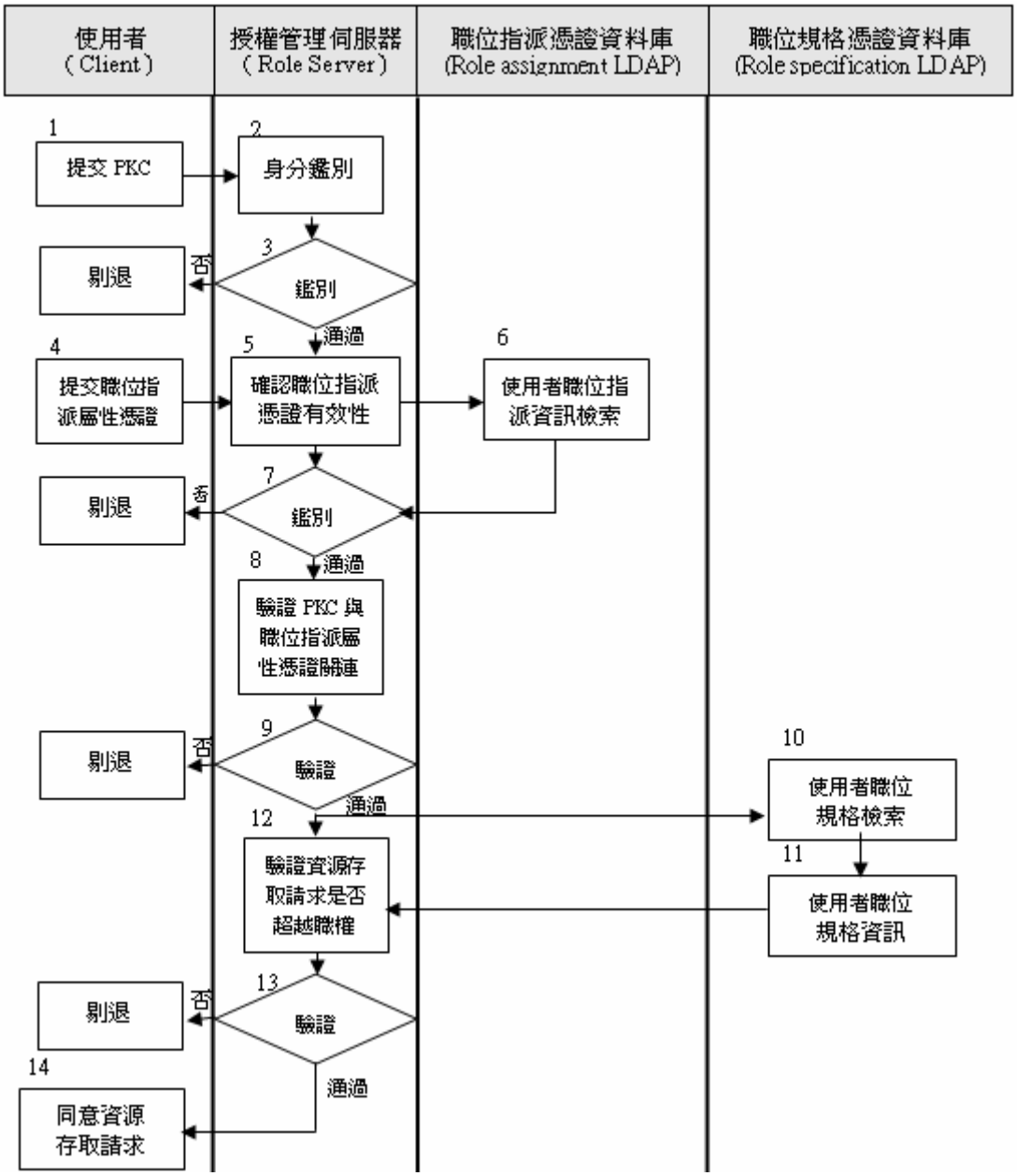


圖 5-9 職位屬性憑證的存取控制驗證流程

5.5 授權領域外部屬性憑證管理服務設計

IBCU 的運作對授權領域外部使用者主要提供的屬性憑證管理服務如下列所示：

（1）屬性憑證的申請與發行

（2）屬性憑證的註銷

（3）屬性憑證的委託

（4）屬性憑證的請求（Request）及驗證（Verification）

（5）屬性憑證的存取控制

屬性憑證的申請主要是經由線上及離線等程序完成各項服務流程作業，使用者向 IBCU 提出申請屬性憑證(線上申請)，並出具相關文件提供 IBCU 審查申請授權之適用性及合法性（離線審查），由 IBCU 主官核准申請後，簽發屬性憑證，並將屬性憑證儲存於屬性憑證資料庫。

當屬性憑證的使用期限已過、或遭非法盜用、或不合法的屬性篡改及使用者主動提出註銷等情況發生，該屬性憑證即由發行之 IBCU 自動列入屬性憑證註銷清冊中（Attribute Certificates Revocation List, ACRL），並線上通知屬性憑證原持有者，且供各使用者查詢。屬性憑證中所記載之資訊為存取資源的授權，通常是具機密性質的，故一旦列入 ACRL 中，就不提供回覆的機制，當使用者提出存取資源請求，並出具屬性憑證時，授權管理伺服器會自動檢查 ACRL，來判斷使用者的屬性憑證是否已被註銷，這種方法在實作上是可行的而且是必要的。

屬性憑證的委託是針對「業務代理」而設計之機制，當屬性憑證持有者短期內無法執行某項業務，可以經由本機制將持有之權限委託給業務代理人，代為執行工作，屬性憑證的委託由屬性憑證持有者提出，經由業務單位主

官核准後，送交 IBCU 執行屬性憑證委託作業，委託期間可視情況限制代理人授權項目，也就是並不一定所有權限全部委託，而只是部份職權委託，委託期限期滿或是委託者主動提出委託終止，可立即取消代理人之授權，此設計可滿足組織業務代理運作現況作業的需求。

屬性憑證的請求及驗證是使用者執行資源存取前重要的控管機制，當外部的使用者提出資源存取的請求時，是為了確保資源存取的合法性，若使用者未提示屬性憑證時，本地的 IBCU 會向該使用者授權領域之授權管理伺服器提出屬性憑證的請求，並驗證該使用者是否具有相對應之存取權限，若通過驗證即可存取資源，否則拒絕該使用者。

5.5.1 屬性憑證的申請與發行

（1）屬性憑證申請：

屬性憑證的申請主要是經由線上使用者申請註冊資訊及離線出示核准資訊存取文件，經由 IBCU 授權管理者審查無誤後，完成憑證申請作業，整個作業流程可區分「申請者身分確認階段」及「屬性憑證審查階段」，如圖 5-10。

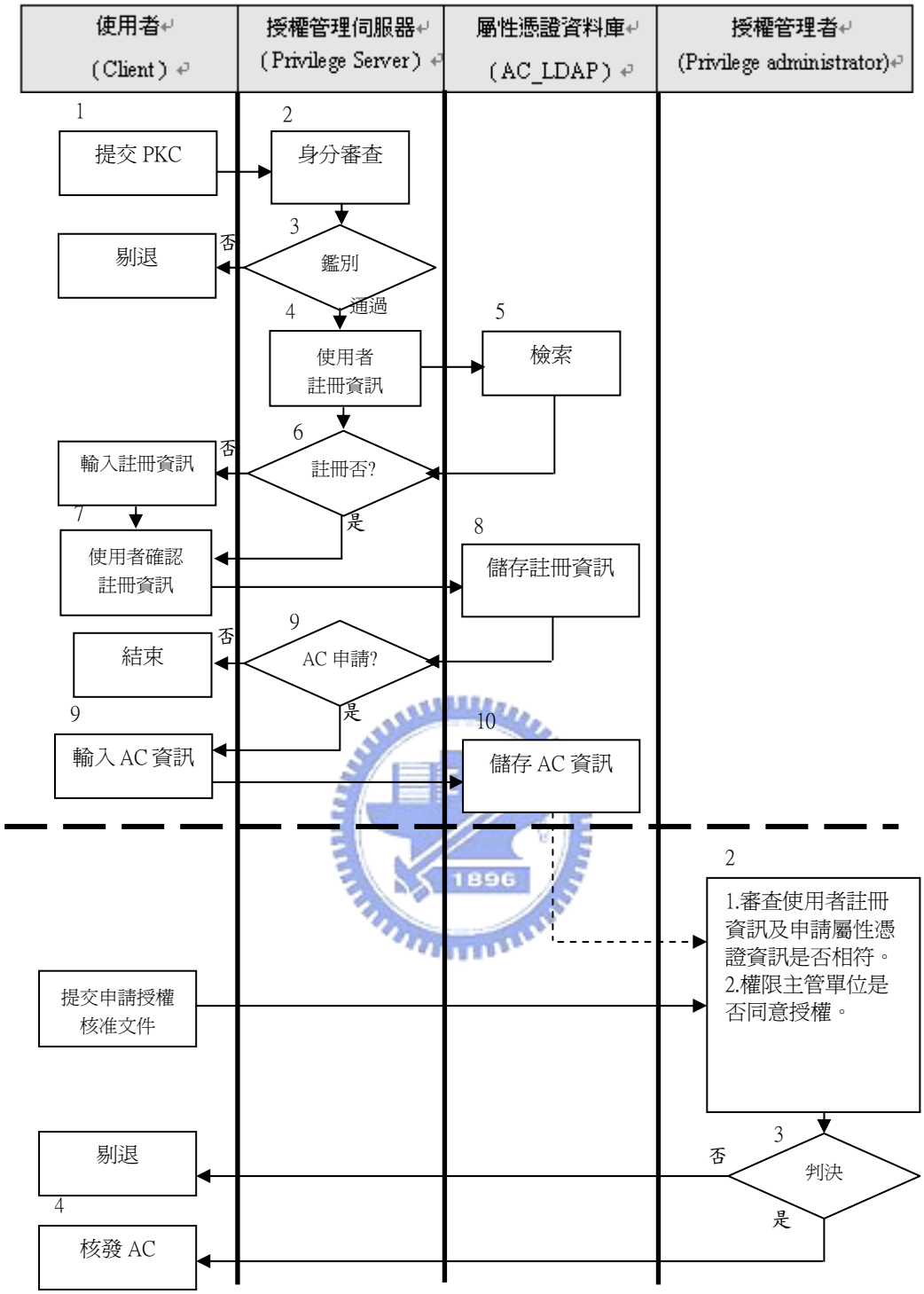


圖 5-10 屬性憑證的申請流程

屬性憑證申請流程說明如下：

1.申請者身分確認階段（線上作業）：

A.申請者提交 PKC 憑證由授權管理者驗證身分是否為合法的空軍總部用戶，或某下屬授權領域使用者。

B.通過驗證之使用者，確認是否已於本授權領域辦理註冊手續，註冊程序主要是完整保留使用者正確之資訊，以便日後各項資訊查核。

C.申請者輸入相關屬性資訊，並儲存於屬性憑證資料庫中，完成本階段作業。

2.屬性憑證審查階段（離線作業）：

A.授權管理者審查申請者身分資訊與屬性資訊是否相符及授權資訊是否經相關主官同意等查核。

B.通過審查即由屬性憑證管理中心發行屬性憑證。

（2）屬性憑證發行（如：圖 5-11）：

屬性憑證的發行機制理論已於 4.4.3 節中介紹過，可分為二種模式，第一種是「推」（PUSH）模式，當用戶要求存取資源時，由用戶直接提交其屬性憑證，即用戶將自己的屬性憑證”推”給目的地授權管理伺服器檢查，此模式在用戶與伺服器之間不需要建立新的連接，由於目的地伺服器不須執行屬性憑證的請求，故可以提昇目的地伺服器作業效能。

第二種模式是「拉」（PULL）模式，屬性憑證管理中心發行屬性憑證儲存於屬性憑證資料庫中，當用戶需要屬性憑證的時候，由授權管理伺服器從屬性憑證發行單位”拉”回屬性憑證。

此二種模式運用對使用者而言具有透通性，也就是使用者在進行驗證的過程是感覺不到且不會有實體的變更。存取的控制主要是由授權管理伺服器驗證用戶之 PKC，對其身分進行確認後，再驗證是否具有存取資源的權限，即驗證用戶的屬性憑證有效性，包含是否為可信任的 IBCU 發行的屬性憑證及簽章的有效性，通過驗證，用戶就可以執行屬性憑證中授予的權限，對資源進行存取。

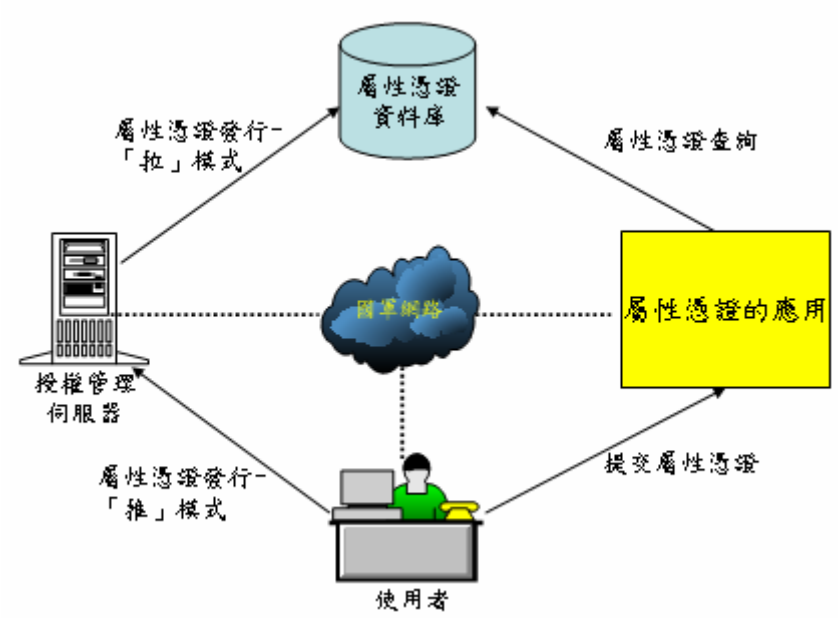


圖 5-11 兩種屬性憑證發行的模式

5.5.2 屬性憑證的註銷

屬性憑證的註銷作業與 PKC 的註銷作業大致相同，是經由屬性憑證註銷清冊（ACRL）的發布，提供使用者查詢屬性憑證的有效性。但對於廣泛使用的屬性憑證、單一次使用的屬性憑證及有效期限非常短的屬性憑證（由授權政策中律訂），是無須執行註銷作業，屬性憑證的註銷條件可視屬性憑

證的種類，律定不同的註銷條件，以供 IBCU 據以執行。屬性憑證註銷作業流程如圖 5-12。

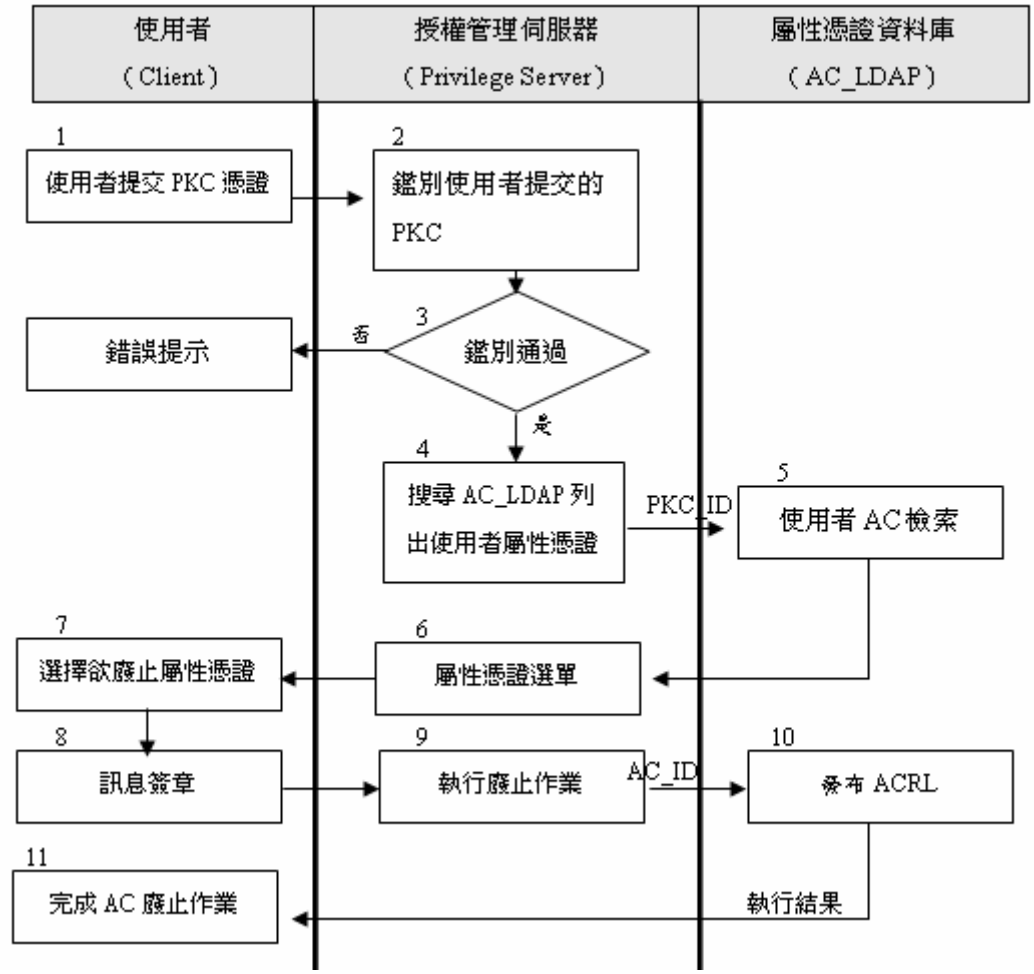


圖 5-12 屬性憑證註銷作業流程

5.5.3 屬性憑證的委託

屬性憑證的委託是由委託者發起向授權管理者提出委託申請，並提交自己及代理者的 PKC 由授權管理者驗證身分後再審查委託人的屬性憑證，審查委託人是否有權對其所申請的委託權限進行委託。在完成所有的審查後，授權管理者

向代理者核發屬性憑證，將委託的權限授予代理者，而完成整個流程，屬性憑證的委託作業流程如圖 5-13。

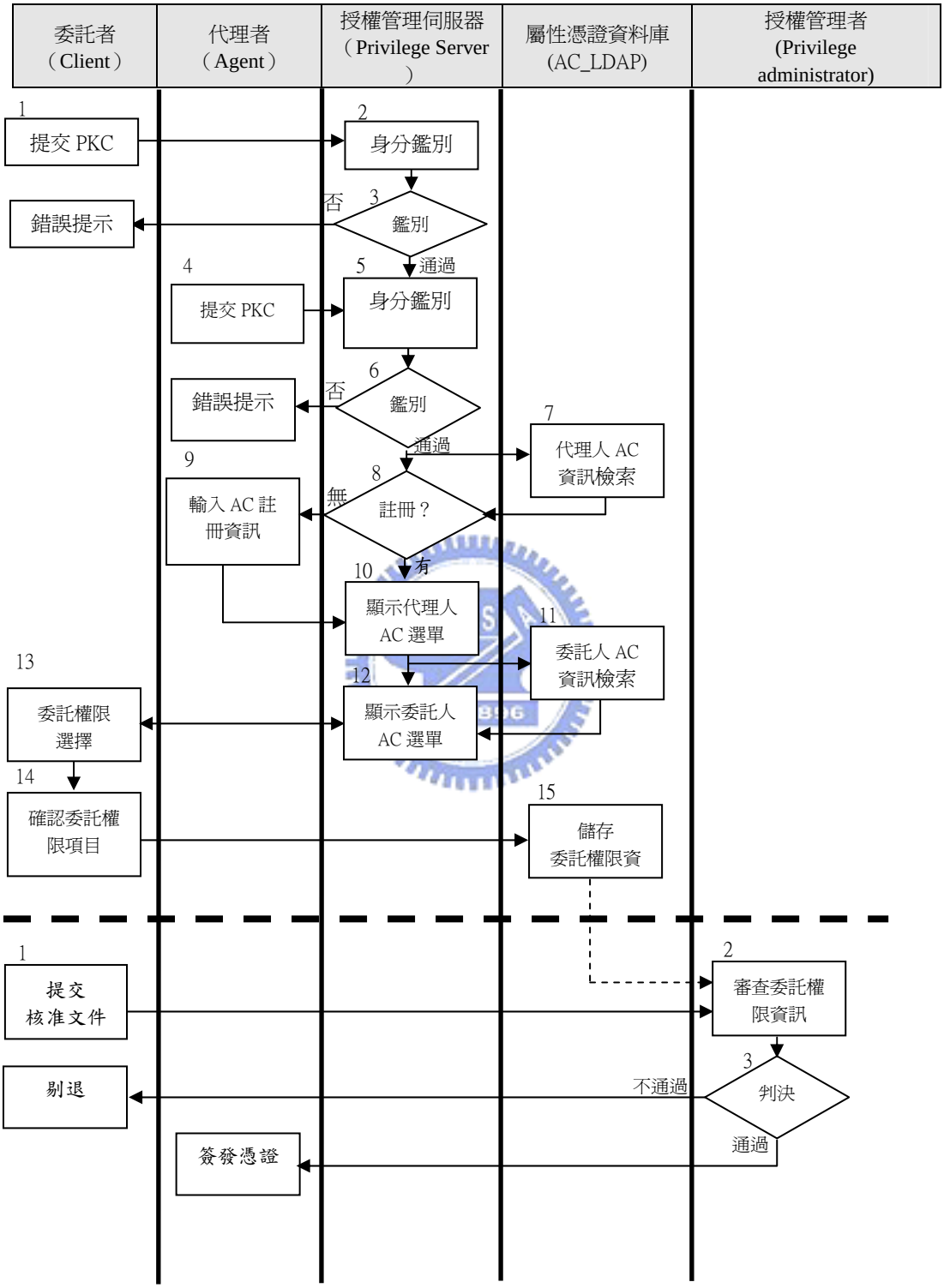


圖 5-13 屬性憑證的委託作業流程

5.5.4 屬性憑證的請求及驗證

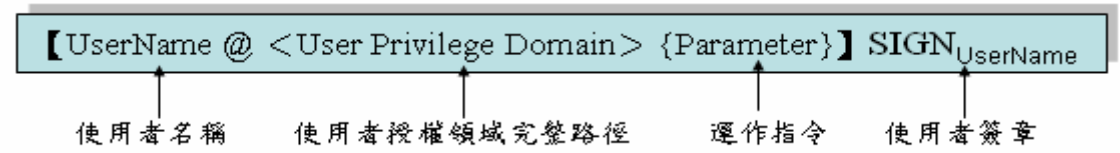
屬性憑證的請求是當使用者執行資源存取時授權管理伺服器為了驗證使用者是否有適當的權限，而要求使用者授權領域之授權管理伺服器提供使用者之屬性憑證，以驗證使用者是否具有存取資源的授權。

5.5.4.1 授權連結定義

本研究提出之以業務為授權的 PMI 中為了簡化屬性憑證的請求作業，並且提供快速查詢每一個使用者之授權資訊，本研究設計以屬性憑證的 *Holder* 欄位中以 ***baseCertificateID*** 欄位連結身分憑證(PKC)，以 ***entityName*** 欄位連結使用者的授權資訊，以 ASN.1 定義 *Holder* 欄位如下所示。

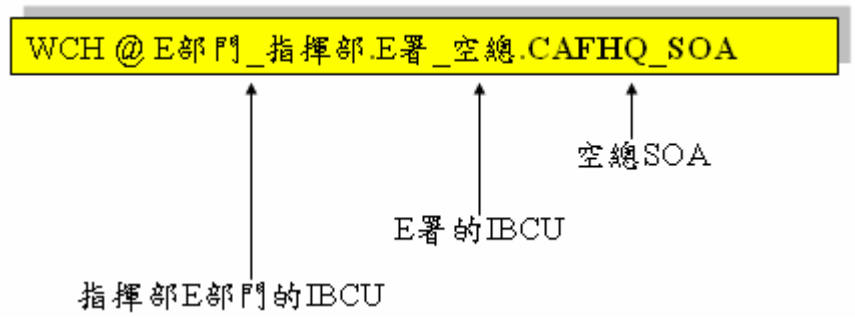
```
Holder  ::= SEQUENCE
        {
         baseCertificateID   [0] IssuerSerial          OPTIONAL,
         -- the issuer and serial number of the holder's Public Key Certificate
         entityName          [1] GeneralNames          OPTIONAL,
         -- the name of the entity or role
         objectDigestInfo    [2] ObjectDigestInfo      OPTIONAL
          --used to directly authenticate the holder, e.g. an executable
        }
```

以 ***entityName*** 連結使用者的授權資訊，設計每一位使用者的識別名稱如下[21]：



此識別名稱可以清楚的定義使用者的授權領域及欲執行的請求指令，並於傳輸過程以使用者之簽章保護，防止不法駭客截收篡改。

以空軍總部組織架構為例（參考圖 5-3），使用者：WCH 的授權領域是空軍總部下屬指揮部的 E 門的 IBCU，其識別名稱即定義如下：



識別代號的設計是以完整的業務屬性授權連結為使用者的識別，如此可以提供授權管理伺服器快速的在 PMI 中搜尋出使用者的授權資訊，範例中各單位的代號是以中文說明，實作上可以將單位以英文定義區別。

5.5.4.2 屬性憑證的請求

屬性憑證的請求可分為兩階段執行：

（1）授權管理伺服器判斷使用者之授權領域。

（2）發出屬性憑證請求。

屬性憑證的請求流程如圖 5-14，執行流程說明如下：

（1）當指揮部 E 部門的 IBCU 接收到使用者：Peter 的資源存取請求，此時授權管理伺服器必須要有 Peter 的公開金鑰憑證及屬性憑證，以驗證是否准許 Peter 執行存取作業。

（2）指揮部 E 部門的 IBCU 啟動憑證請求程序，向 Peter 的授權領域之 IBCU 發出憑證請求。

（3）Peter 授權領域之 IBCU 接收到指揮部 E 部門的請求後，回傳所需的憑證。

（4）指揮部 E 部門的 IBCU 接收到 Peter 的憑證後，為了要確保憑證於傳輸過程中未被篡改，故必須啟動憑證驗證機制。

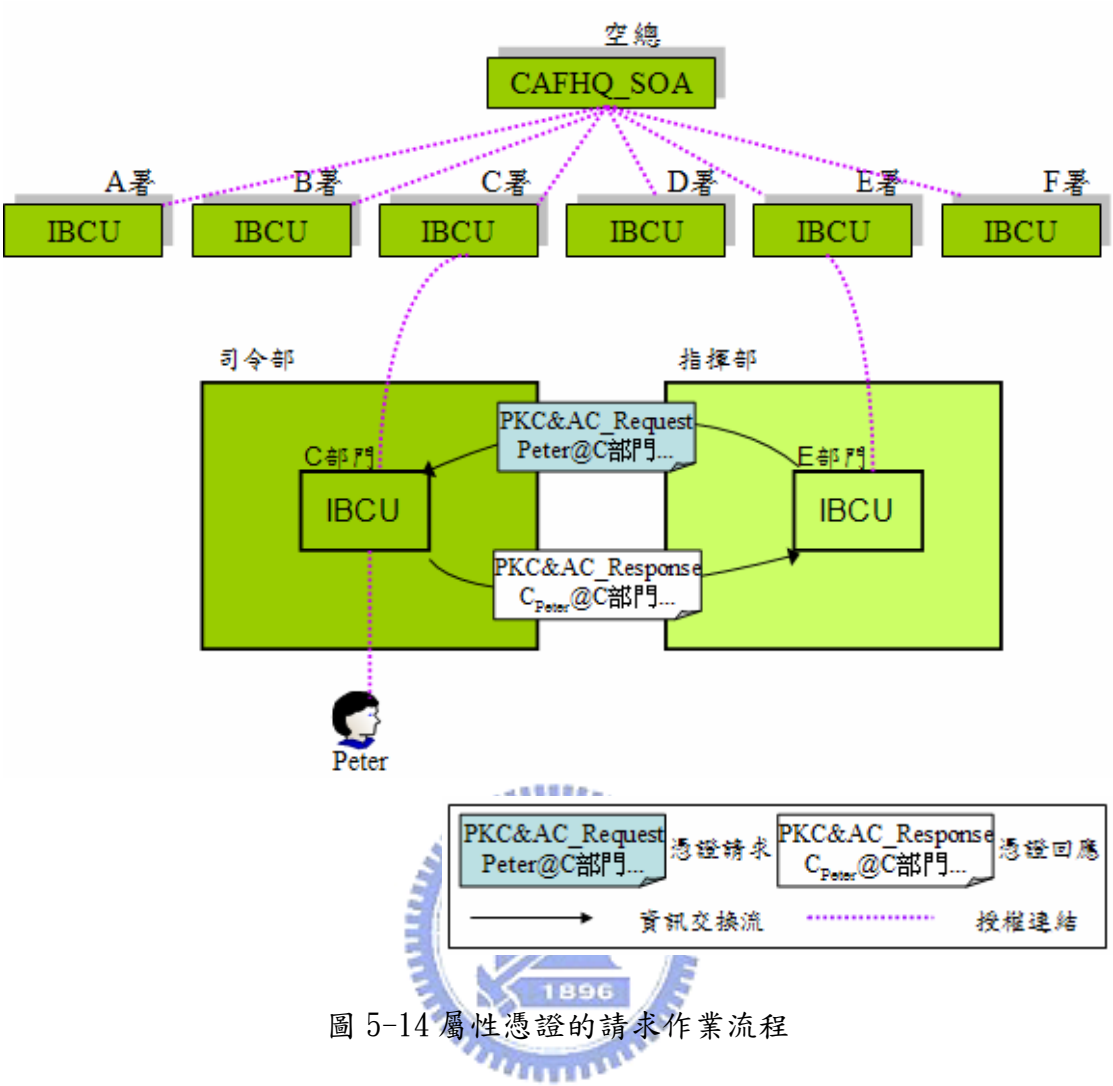


圖 5-14 屬性憑證的請求作業流程

5.5.4.3 屬性憑證的驗證

為了達成有效的使用者驗證作業，在整個架構中對每一個 IBCU 中的 AA 設計了一個特殊的憑證，其識別代碼為 AA@＜Privilege Domain＞，範例定義如下：

此憑證儲存於授權連結的上一層（parent）資料庫中，主要的功能為提供驗證方的授權管理伺服器確認接收到的屬性憑證的真確性，保證發行此屬性憑證的機構（IBCU）未遭受駭客模仿攻擊，例如：指揮部 E 部門（AA 識別代碼：AA@ E 部門_指揮部.E 署_空總.CAFHQ_SOA）的 AA 憑證，是儲存於其上一層授權連結「空軍總部 E 署」的資料庫中，但除了最上層的 CAFHQ_SOA 例外。

屬性憑證的驗證流程如圖 5-15，執行流程說明如下：

（1）為了確認 Peter 傳送的屬性憑證，指揮部 E 部門的 IBCU 向 Peter 的父結點請求 Peter 的 IBCU 的 AA 憑證。

（2）此程驗證程序可保證 Peter 的 IBCU 未被篡改，驗證的過程可以一直到屬性憑證連結的最高單位 CAFHQ_SOA，但須驗證至何階層才停止，是由指揮部 E 部門的 IBCU 決定。

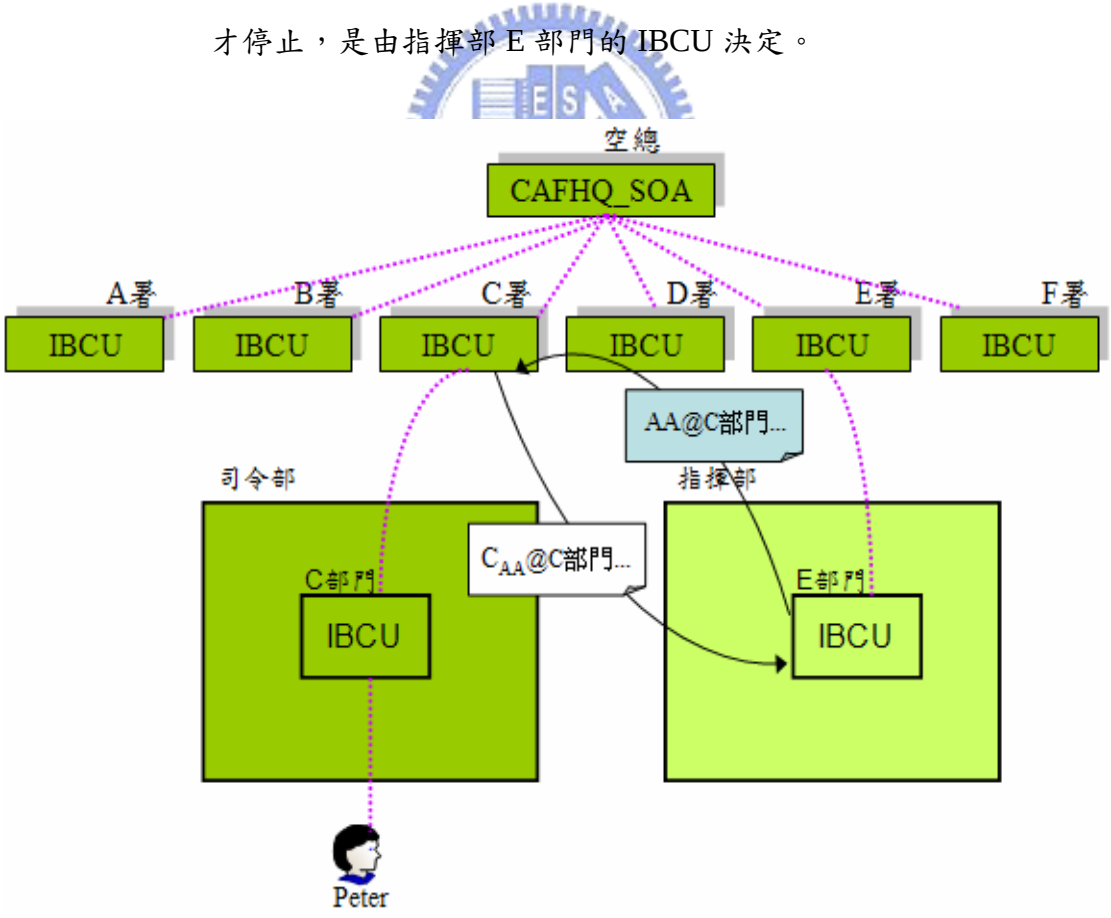


圖 5-15 屬性憑證的驗證作業流程

5.5.5 屬性憑證的存取控制

屬性憑證的存取控制是由使用者提出對資源的存取請求，首先使用者提交 PKC 給授權管理伺服器執行身分驗證，通過身分驗證後檢查使用者是否提交屬性憑證，如果使用者未提交屬性憑證，則至使用者授權領域的 IBCU 拉回（PULL）屬性憑證。而授權決策的判斷是依據使用者屬性憑證所記載之資訊及組織授權政策規則，做成存取決策，整個流程如圖 5-16。

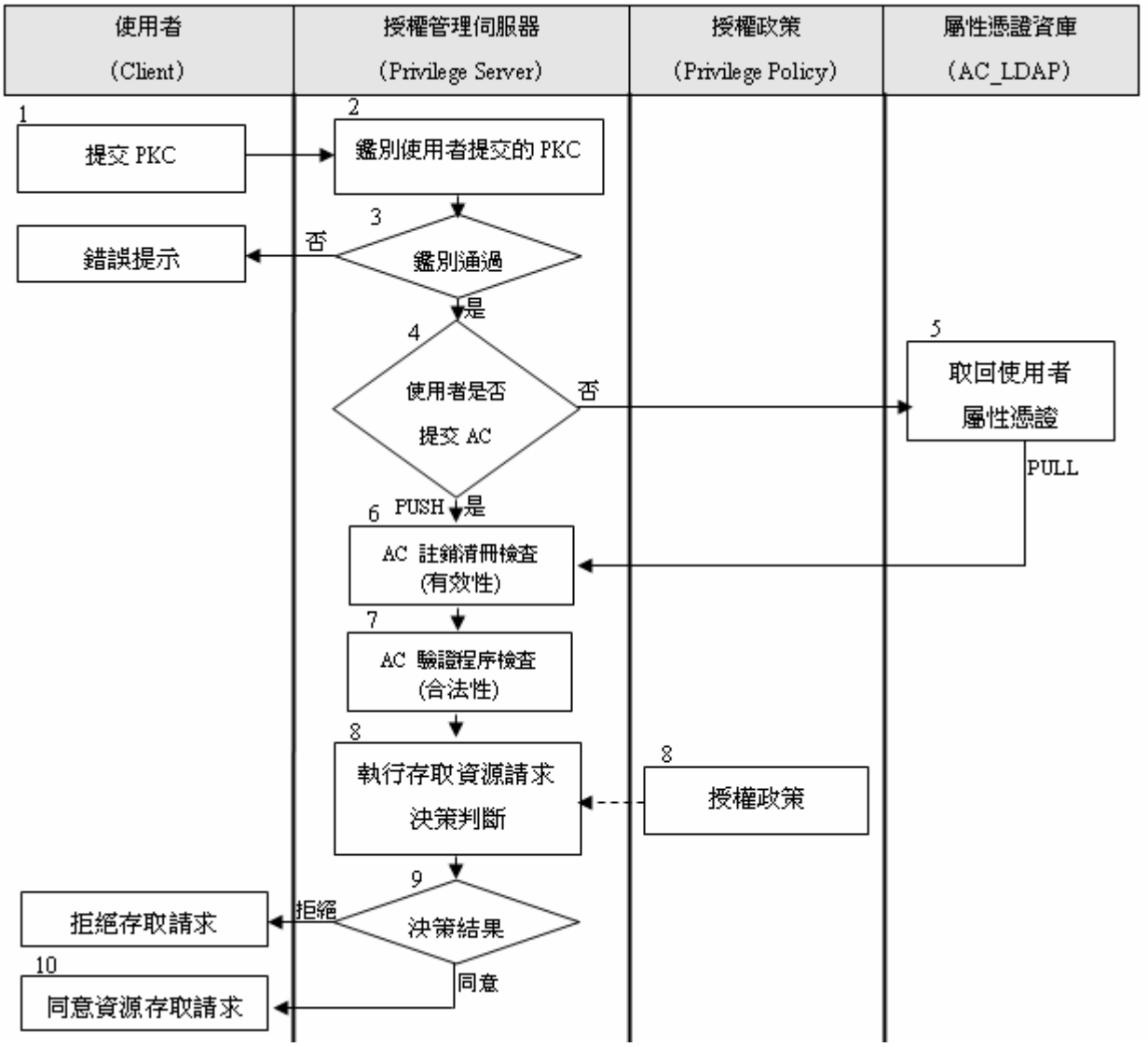


圖 5-16 屬性憑證的存取控制流程

第六章　結論與未來研究方向

6.1 結論

本論文運用 ITU-T X.509v4 文件所提供的屬性憑證概念，規畫及設計空軍總部的授權管理基礎建設，運用屬性憑證的特性使得授權管理得以分散式的建置，在管理層面，提出以「業務」性質為基礎之授權管理基礎建設，本架構模型具有彈性且兼具安全性之特性；在作業層面，以「業務」為授權領域所建置之業務憑證管理中心（IBCU）可以彈性自主的依業務特性自行定訂安全政策，使用者的授權是以職位（Role）為主體，而非傳統模式是以人及資源為主體，將 RBAC 的管理模式以屬性憑證實現，區分職位的指派及授權的分工管理，有效的降低管理成本並提昇作業效能；在安全層面，結合 PKI 的身分驗證功能使得使用者憑證不易遭受網路駭客的篡改，無形的提高組織的 PMI 安全性，這是傳統授權管理模式所無法達成的，也是本論文最主要的貢獻。

在虛擬的網路社會中存在太多無法預期的人、事、物，所以安全的顧慮一向是阻礙電子交易行為的首要問題，本論文提出的授權管理架構將組織的資源管理結合使用者授權，提供使用者一套取得權限及存取資源的規則，而且可以彈性的依組織架構修改授權管理模式，此架構可應用於下列類型的組織：

（1）政府機關或軍方單位資源存取控管。

（2）具有機密性質的數位資產管理。

（3）組織部門的階層設計及作業流程是以「業務」為主要性質的資源存取控管。

6.2 未來研究方向

本次的研究主要是以屬性憑證為主體，運用其特性建置出組織的授權管理基礎架構，而其中公開金鑰憑證是鑑別使用者的身分，屬性憑證是提供使用者授權資訊的記載，此二者具有關連性。雖然可以使用公開金鑰憑證提供的延伸欄位記載使用者的授權資訊，但確有非常多的限制，因此如何有效的設計出單一憑證即可完成上述的功能，也就是具有智慧型的憑證，此論點是本論文日後研究的方向。

專家系統的應用可以結合多種領域，經由專家提供的知識，可以解決出特定的問題，以本論文而言，在執行授權的判斷決策過程中，是必須提供多項資訊供授權管理者參考的，這也是一個嚴謹的系統必須有的審查手續，此過程是否可以結合專家系統，將各種決策資訊導入知識庫，提供系統存取控制判斷，使系統自動查核及鑑別出有非法的授權請求，也是本論文後續研究的方向。

參考文獻

[1]IETF, ”Lightweight Directory Access protocol, ”RFC 1777, March 1995.

[2]IETF, ”Lightweight Directory Access protocol（v3）, ”RFC 2251, December 1997.

[3]ITU-T,” ITU-T Recommendation X.509: The Directory-Public-key and Certificate frameworks,” March 2000.

[4]IETF,” An Internet Attribute Certificate Profile for Authorization,” RFC 3281, April 2002.

[5]IETF,” Internet X.509 Public Key Infrastructure Certificate and CRL Profile,” RFC 2459, January 1999.

[6]Nash, A., Duane, W., Joseph, C., & Brink, D.,”PKI: Implementing and Managing E-Security,” Mc-Graw Hill2001.

[7]David W. Chadwick , Alexander Otenko, ”The PERMIS X.509 Role Based Privilege Management Infrastructrue,” Seventh ACM Symposium on Access Control Models and Technologies, pp.135-140, June 2002.

[8]CCITT,”Recommendation X.500: The Directory-overview of concepts models and Services,” 1988.

[9]D.W Chadwick,”Understanding X.500（The Directory）,” Chapman and Hall, June 1994.

[10] ITU-T,” ITU-T Recommendation X.509: The Authentication frameworks,” 1993.

[11]ISO/IEC,” ISO/IEC 9594-8 Information Technology – Open System Interconnection - The Directory: Authentication Framework,”1993.

[12]Amendment 1 to ISO/IEC 9594-8,”Certificate extensions,”1995.

[13]ISO/IEC,” ISO/IEC 10181-3 Information Technology – Open System

Interconnection – Security Framework for Open System: Access Control Framework,”1996.

[14]D. Ferraiolo and D.R. Kuhn, “Role based Access Control,” 15th NIST-NCSC National Computer Security Conference, pp. 554-563, 1992.

[15]Jing-Jang Hwang, Kou-Chen Wu, Duen-Ren Liu, “Access control with role attribute certificates,” Computer Standards & Interfaces, vol.22, pp.43–53 2000.

[16]R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based Access Control Models,” IEEE Computer, Vol. 29, No. 2, pp. 38-47, February 1996.

[17]D.R. Kuhn, “Role Based Access Control on MLS Systems Without Kernel Changes,” Third ACM Workshop on Role Based Access Control, pp.22-23, October 1998.

[18]M. Wahl, “A Summary of the X.500(96) User Schema for use withLDAPv3,” RFC 2256, December 1997.

[19]R. Sandhu, P. Samarati,” Access Control：Principles and Practices,”IEEE Communication magazine, 32(9), pp.40-48, 1994.

[20]R. Oppliger, G. Pernul, and Ch. Strauss, “Using Attribute Certificates to Implement Role-based Authorization and Access Control,” Proceedings of the 4. Fachtagung Sicherheit in Informations systemen(SIS 2000), pp. 169-184, October 2000.

[21]D. Crocker, “Standard for the format of Arpa Internet Text Messages,” RFC822, August 1982.

[22]黃景彰,”資訊安全-電子商務之基礎 ,” 華泰文化事業公司, 2001.

[23]劉興華,”執行權管制系統的理論性架構設計,” 國立交通大學資訊管理研究所, 博士論文, 1999.

[24]吳宗成,”資通安全技術資源簡介–資通安全專輯之三,” 行政院國家科學委員會科學技術資料中心, 2002.

附錄

Attribute Certificate Frameworks in ASN.1

```
AttributeCertificateDefinitions {joint-iso-itu-t ds(5) module(1) attributeCertificateDefinitions(32) 4}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
-- EXPORTS ALL --
IMPORTS
       id-at, id-ce, id-mr, informationFramework, authenticationFramework,
            selectedAttributeTypes, upperBounds, id-oc, certificateExtensions
            FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1)
            usefulDefinitions(0) 4}

Name, RelativeDistinguishedName, ATTRIBUTE, Attribute,
       MATCHING-RULE, AttributeType, OBJECT-CLASS, top
       FROM InformationFramework informationFramework

CertificateSerialNumber, CertificateList, AlgorithmIdentifier,
       EXTENSION, SIGNED, InfoSyntax, PolicySyntax, Extensions, Certificate
       FROM AuthenticationFramework authenticationFramework

DirectoryString, TimeSpecification, UniqueIdentifier
       FROM SelectedAttributeTypes selectedAttributeTypes

GeneralName, GeneralNames, NameConstraintsSyntax, certificateListExactMatch
       FROM CertificateExtensions certificateExtensions

ub-name
       FROM UpperBounds upperBounds

UserNotice
       FROM PKIX1Implicit93 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
       pkix(7) id-mod(0) id-pkix1-implicit-93(4)}

ORAddress
       FROM MTSAbstractService {joint-iso-itu-t mhs(6) mts(3)
       modules(0) mts-abstract-service(1) version-1999 (1) } ;

-- Unless explicitly noted otherwise, there is no significance to the ordering
-- of components of a SEQUENCE OF construct in this specification.
-- attribute certificate constructs -

AttributeCertificate ::= SIGNED {AttributeCertificateInfo}
AttributeCertificateInfo ::= SEQUENCE
    {
    Version                 AttCertVersion, --version is v2
    Holder                  Holder,
    Issuer                  AttCertIssuer,
    Signature               AlgorithmIdentifier,
    serialNumber            CertificateSerialNumber,
    attrCertValidityPeriod  AttCertValidityPeriod,
    attributes              SEQUENCE OF Attribute,
    issuerUniqueID          UniqueIdentifier OPTIONAL,
    extensions              Extensions OPTIONAL
    }
```

```
AttCertVersion ::= INTEGER { v2(1) }

Holder    ::=     SEQUENCE
          {
           baseCertificateID      [0] IssuerSerial       OPTIONAL,
               -- the issuer and serial number of the holder's Public Key Certificate
           entityName             [1] GeneralNames    OPTIONAL,
               -- the name of the entity or role
           objectDigestInfo       [2] ObjectDigestInfo   OPTIONAL
              --used to directly authenticate the holder, e.g. an executable
--at least one of baseCertificateID, entityName or objectDigestInfo shall be present--}

ObjectDigestInfo       ::= SEQUENCE {
      digestedObjectType ENUMERATED {
          publicKey           (0),
          publicKeyCert       (1),
          otherObjectTypes    (2)},
      otherObjectTypeID       OBJECT IDENTIFIER OPTIONAL,
      digestAlgorithm          AlgorithmIdentifier,
      objectDigest             BIT STRING}

AttCertIssuer   ::=         [0]      SEQUENCE {
   issuerName                   GeneralNames OPTIONAL,
   baseCertificateID        [0] IssuerSerial OPTIONAL,
   objectDigestInfo         [1] ObjectDigestInfo OPTIONAL }
-- At least one component shall be present
( WITH COMPONENTS { ..., issuerName PRESENT } |
WITH COMPONENTS { ..., baseCertificateID PRESENT } |
WITH COMPONENTS { ..., objectDigestInfo PRESENT } )

IssuerSerial ::= SEQUENCE {
      issuer          GeneralNames,
      serial          CertificateSerialNumber,
      issuerUID       UniqueIdentifier OPTIONAL }

AttCertValidityPeriod   ::=  SEQUENCE {
      notBeforeTime          GeneralizedTime,
      notAfterTim            GeneralizedTime }

AttributeCertificationPath    ::=  SEQUENCE {
      attributeCertificate          AttributeCertificate,
      acPath                        SEQUENCE OF ACPathData OPTIONAL }

ACPathData   ::=  SEQUENCE {
Certificate             [0] Certificate OPTIONAL,
attributeCertificate    [1] AttributeCertificate OPTIONAL }

PrivilegePolicy   ::=  OBJECT IDENTIFIER

--privilege attributes—

role   ATTRIBUTE ::= {
       WITH SYNTAX         RoleSyntax
       ID                  id-at-role }
RoleSyntax ::= SEQUENCE {
       roleAuthority       [0] GeneralNames OPTIONAL,
       roleName            [1] GeneralName }
-- PMI object classes --
```

```
pmiUser OBJECT-CLASS ::= {
      SUBCLASS OF        {top}
      KIND               auxiliary
      MAY CONTAIN        {attributeCertificateAttribute}
      ID                 id-oc-pmiUser
      }

pmiAA OBJECT-CLASS ::= {
-- a PMI AA
     SUBCLASS OF          {top}
     KIND                 auxiliary
     MAY CONTAIN          {aACertificate |
                           attributeCertificateRevocationList |
                           attributeAuthorityRevocationList}
     ID                   id-oc-pmiAA
     }

pmiSOA OBJECT-CLASS ::= { -- a PMI Source of Authority
   SUBCLASS OF             {top}
   KIND                    auxiliary
   MAY CONTAIN             {attributeCertificateRevocationList |
                           attributeAuthorityRevocationList |
                           attributeDescriptorCertificate}
   ID                      id-oc-pmiSOA
   }

attCertCRLDistributionPtOBJECT-CLASS ::= {
   SUBCLASS OF           {top}
   KIND                  auxiliary
   MAY CONTAIN           { attributeCertificateRevocationList |
                         attributeAuthorityRevocationList }
   ID                    id-oc-attCertCRLDistributionPts
   }

pmiDelegationPath OBJECT-CLASS ::= {
   SUBCLASS OF         {top}
   KIND                auxiliary
   MAY CONTAIN         { delegationPath }
   ID                  d-oc-pmiDelegationPath }

privilegePolicy OBJECT-CLASS ::= {
   SUBCLASS OF         {top}
   KIND                auxiliary
   MAY CONTAIN         {privPolicy }
   ID                  id-oc-privilegePolicy }
```

- *PMI directory attributes* -
-

```
attributeCertificateAttribute ATTRIBUTE ::= {
   WITH SYNTAX                    AttributeCertificate
   EQUALITY MATCHING RULE         attributeCertificateExactMatch
   ID                             id-at-attributeCertificate }
```

```
aACertificate         ATTRIBUTE      ::=      {
      WITH SYNTAX                             AttributeCertificate
      EQUALITY MATCHING RULE                  attributeCertificateExactMatch
      ID                                      id-at-aACertificate }

attributeDescriptorCertificate ATTRIBUTE ::= {
      WITH SYNTAX                             AttributeCertificate
      EQUALITY MATCHING RULE                  attributeCertificateExactMatch
      ID                                      id-at-attributeDescriptorCertificate }

attributeCertificateRevocationList ATTRIBUTE ::= {
      WITH SYNTAX                             CertificateList
      EQUALITY MATCHING RULE                  certificateListExactMatch
      ID                                      id-at-attributeCertificateRevocationList}

attributeAuthorityRevocationList ATTRIBUTE ::= {
      WITH SYNTAX                             CertificateList
      EQUALITY MATCHING RULE                  certificateListExactMatch
      ID                                      id-at-attributeAuthorityRevocationList }

delegationPath   ATTRIBUTE     ::= {
      WITH SYNTAX      AttCertPath
      ID               id-at-delegationPath }

AttCertPath::=       SEQUENCE OF AttributeCertificate

privPolicy  ATTRIBUTE   ::= {
      WITH SYNTAX      PolicySyntax
      ID               id-at-privPolicy }


--Attribute certificate extensions and matching rules --

attributeCertificateExactMatch MATCHING-RULE ::= {
      SYNTAX            AttributeCertificateExactAssertion
      ID                id-mr-attributeCertificateExactMatch }

AttributeCertificateExactAssertion ::= SEQUENCE {
      serialNumber       CertificateSerialNumber OPTIONAL,
      issuer             IssuerSerial
      }

attributeCertificateMatch MATCHING-RULE ::= {
      SYNTAX             AttributeCertificateAssertion
      ID                 id-mr-attributeCertificateMatch }

AttributeCertificateAssertion ::= SEQUENCE {
      Holder                  [0]  CHOICE {
      baseCertificateID       [0] IssuerSerial,
      holderName              [1] GeneralNames} OPTIONAL,
     issuer                   [1] GeneralNames OPTIONAL,
     attCertValidity          [2] GeneralizedTime OPTIONAL,
     attType                  [3] SET OF AttributeType OPTIONAL}
--At least one component of the sequence shall be present
```

```
holderIssuerMatch MATCHING-RULE ::= {
        SYNTAX          HolderIssuerAssertion
        ID              id-mr-holderIssuerMatch }

HolderIssuerAssertion ::= SEQUENCE {
        Holder          [0] Holder  OPTIONAL,
        Issuer          [1] AttCertIssue r OPTIONAL
        }

delegationPathMatch MATCHING-RULE ::= {
        SYNTAX          DelMatchSyntax
        ID              id-mr-delegationPathMatch }

DelMatchSyntax ::= SEQUENCE {
        firstIssuer     AttCertIssuer,
        lastHolder      Holder }

sOAIdentifier EXTENSION ::= {
        SYNTAX              NULL
        IDENTIFIED BY       id-ce-sOAIdentifier }

authorityAttributeIdentifier EXTENSION ::=
        { SYNTAX AuthorityAttributeIdentifierSyntax
        IDENTIFIED BY { id-ce-authorityAttributeIdentifier }}

AuthorityAttributeIdentifierSyntax    ::=    SEQUENCE SIZE (1..MAX) OF AuthAttId

AuthAttId    ::=    IssuerSerial

authAttIdMatch MATCHING-RULE ::= {
        SYNTAX              AuthorityAttributeIdentifierSyntax
        ID                  id-mr-authAttIdMatch }

roleSpecCertIdentifier EXTENSION ::=
      { SYNTAX              RoleSpecCertIdentifierSyntax
      IDENTIFIED BY         { id-ce-roleSpecCertIdentifier }}

RoleSpecCertIdentifierSyntax ::= SEQUENCE SIZE (1..MAX) OF RoleSpecCertIdentifier

RoleSpecCertIdentifier ::= SEQUENCE {
      roleName                [0]    GeneralName,
      roleCertIssuer          [1]     GeneralName,
      roleCertSerialNumber    [2]     CertificateSerialNumber OPTIONAL,
      roleCertLocator         [3]     GeneralNames OPTIONAL}


roleSpecCertIdMatch MATCHING-RULE ::= {
      SYNTAX        RoleSpecCertIdentifierSyntax
      ID            id-mr-roleSpecCertIdMatch }

basicAttConstraints EXTENSION ::=
      { SYNTAX          BasicAttConstraintsSyntax
      IDENTIFIED BY     { id-ce-basicAttConstraints }}

BasicAttConstraintsSyntax ::= SEQUENCE
      { authority             BOOLEAN DEFAULT FALSE,
      pathLenConstraint       INTEGER (0..MAX) OPTIONAL}
```

```
basicAttConstraintsMatch MATCHING-RULE ::= {
        SYNTAX           BasicAttConstraintsSyntax
        ID              id-mr-basicAttConstraintsMatch }

delegatedNameConstraints EXTENSION ::= {
        SYNTAX          NameConstraintsSyntax
        IDENTIFIED BY   id-ce-delegatedNameConstraints }

delegatedNameConstraintsMatch MATCHING-RULE ::= {
        SYNTAX           NameConstraintsSyntax
        ID               id-mr-delegatedNameConstraintsMatch}

timeSpecification EXTENSION ::= {
        SYNTAX           TimeSpecification
        IDENTIFIED BY    id-ce-timeSpecification }

timeSpecificationMatch MATCHING-RULE ::= {
        SYNTAX           TimeSpecification
        ID               id-mr-timeSpecMatch }

acceptableCertPolicies EXTENSION ::= {
        SYNTAX            AcceptableCertPoliciesSyntax
        IDENTIFIED BY     id-ce-acceptableCertPolicies }

AcceptableCertPoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER

acceptableCertPoliciesMatch MATCHING-RULE ::= {
        SYNTAX            AcceptableCertPoliciesSyntax
        ID                id-mr-acceptableCertPoliciesMatch }

attributeDescriptor EXTENSION ::= {
        SYNTAX            AttributeDescriptorSyntax
        IDENTIFIED BY     {id-ce-attributeDescriptor } }

AttributeDescriptorSyntax ::= SEQUENCE {
        identifier                AttributeIdentifier,
        attributeSyntax           OCTET STRING (SIZE(1..MAX)),
        name              [0]     AttributeName OPTIONAL,
        description       [1]     AttributeDescription OPTIONAL,
        dominationRule            PrivilegePolicyIdentifier}

AttributeIdentifier ::= ATTRIBUTE.&id({AttributeIDs})

AttributeIDs ATTRIBUTE ::= {...}

AttributeName ::= UTF8String(SIZE(1..MAX))

AttributeDescription ::= UTF8String(SIZE(1..MAX))

PrivilegePolicyIdentifier ::= SEQUENCE {
        privilegePolicy      PrivilegePolicy,
        privPolSyntax        InfoSyntax }

attDescriptor MATCHING-RULE ::= {
        SYNTAX        AttributeDescriptorSyntax
        ID           id-mr-attDescriptorMatch }
```

```
userNotice EXTENSION ::= {
        SYNTAX                SEQUENCE SIZE (1..MAX) OF UserNotice
        IDENTIFIED BY         id-ce-userNotice }

targetingInformation EXTENSION ::= {
        SYNTAX                SEQUENCE SIZE (1..MAX) OF Targets
        IDENTIFIED BY         id-ce-targetInformation }

Targets ::= SEQUENCE SIZE (1..MAX) OF Target

Target ::= CHOICE {
        targetName           [0] GeneralName,
        targetGroup          [1] GeneralName,
        targetCert           [2] TargetCert }

TargetCert ::= SEQUENCE {
        targetCertificate      IssuerSerial,
        targetName             GeneralName OPTIONAL,
        certDigestInfo         ObjectDigestInfo OPTIONAL }

noRevAvail EXTENSION ::= {
        SYNTAX           NULL
        IDENTIFIED BY    id-ce-noRevAvail }

acceptablePrivilegePolicies EXTENSION ::= {
        SYNTAX            AcceptablePrivilegePoliciesSyntax
        IDENTIFIED BY     id-ce-acceptablePrivilegePolicies }

AcceptablePrivilegePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PrivilegePolicy

--object identifier assignments—
--object classes—

id-oc-pmiUser                           OBJECT IDENTIFIER ::= {id-oc 24}
id-oc-pmiAA                             OBJECT IDENTIFIER ::= {id-oc 25}
id-oc-pmiSOA                            OBJECT IDENTIFIER ::= {id-oc 26}
id-oc-attCertCRLDistributionPts         OBJECT IDENTIFIER ::= {id-oc 27}
id-oc-privilegePolicy                   OBJECT IDENTIFIER ::= {id-oc 32}
id-oc-pmiDelegationPath                 OBJECT IDENTIFIER ::= {id-oc 33}

--directory attributes—

id-at-attributeCertificate              OBJECT IDENTIFIER ::= {id-at 58}
id-at-attributeCertificateRevocationList  OBJECT IDENTIFIER ::= {id-at 59}
id-at-aACertificate                     OBJECT IDENTIFIER ::= {id-at 61}
id-at-attributeDescriptorCertificate    OBJECT IDENTIFIER ::= {id-at 62}
id-at-attributeAuthorityRevocationList  OBJECT IDENTIFIER ::= {id-at 63}
id-at-privPolicy                        OBJECT IDENTIFIER ::= {id-at 71}
id-at-role                              OBJECT IDENTIFIER ::= {id-at 72}
id-at-delegationPath                    OBJECT IDENTIFIER ::= {id-at 73}
```

```
--attribute certificate extensions—

id-ce-authorityAttributeIdentifier          OBJECT IDENTIFIER ::= {id-ce 38}
id-ce-roleSpecCertIdentifier                OBJECT IDENTIFIER ::= {id-ce 39}
id-ce-basicAttConstraints                   OBJECT IDENTIFIER ::= {id-ce 41}
id-ce-delegatedNameConstraints              OBJECT IDENTIFIER ::= {id-ce 42}
id-ce-timeSpecification                     OBJECT IDENTIFIER ::= {id-ce 43}
id-ce-attributeDescriptor                   OBJECT IDENTIFIER ::= {id-ce 48}
id-ce-userNotice                            OBJECT IDENTIFIER ::= {id-ce 49}
id-ce-sOAIdentifier                         OBJECT IDENTIFIER ::= {id-ce 50}
id-ce-acceptableCertPolicies                OBJECT IDENTIFIER ::= {id-ce 52}
id-ce-targetInformation                     OBJECT IDENTIFIER ::= {id-ce 55}
id-ce-noRevAvail                            OBJECT IDENTIFIER ::= {id-ce 56}
id-ce-acceptablePrivilegePolicies           OBJECT IDENTIFIER ::= {id-ce 57}

--PMI matching rules—

id-mr-attributeCertificateMatch             OBJECT IDENTIFIER ::= {id-mr 42}
id-mr-attributeCertificateExactMatch        OBJECT IDENTIFIER ::= {id-mr 45}
id-mr-holderIssuerMatch                     OBJECT IDENTIFIER ::= {id-mr 46}
id-mr-authAttIdMatch                        OBJECT IDENTIFIER ::= {id-mr 53}
id-mr-roleSpecCertIdMatch                   OBJECT IDENTIFIER ::= {id-mr 54}
id-mr-basicAttConstraintsMatch              OBJECT IDENTIFIER ::= {id-mr 55}
id-mr-delegatedNameConstraintsMatch         OBJECT IDENTIFIER ::= {id-mr 56}
id-mr-timeSpecMatch                         OBJECT IDENTIFIER ::= {id-mr 57}
id-mr-attDescriptorMatch                    OBJECT IDENTIFIER ::= {id-mr 58}
id-mr-acceptableCertPoliciesMatch           OBJECT IDENTIFIER ::= {id-mr 59}
id-mr-delegationPathMatch                   OBJECT IDENTIFIER ::= {id-mr 61}
END
```