# 一個有效率的行動代理人存取控制與金鑰管理機制

林家禎　　楊蕾巧
靜宜大學資訊管理系所 教授　靜宜大學資訊管理系所 研究生
mhlin3@pu.edu.tw　　g9571004@pu.edu.tw

## 摘要

隨著資訊科技的快速發展，越來愈多資訊或是服務可以在網路來獲取。然而網路如此浩瀚，在網路上的相關資訊或是服務如此豐富。要在有限的時間內搜尋到跟自己所需相關的服務或是資訊是件相當具有挑戰力的事情。而行動代理人便是回應上述需求的一項重要技術。雖然行動代理人可以攜帶使用者事先給定的需求在網路上漫遊，協助使用者搜尋跟他有關的服務以及資訊。但是，網路上的商務主機並不是全部都是善意的。此外，當商務主機彼此之間存在著利益衝突時，惡意的主機有可能企圖修改代理人已蒐集到的相關資訊、或是竄改予其利益衝突相關主機的資訊。為了防止行動代理人遭到惡意商務主級的攻擊，如何有效透過金鑰管理以及適當的存取控制來限定各商務主機對行動代理人所需資訊的存取範圍便是件重要的議題。本篇論文嘗試提出一個以角色關係為基礎的金鑰管理和存取控制方法。我們的方法因為使用了單向雜湊函數，除了能保護行動代理人所攜帶的資料外，還能有效地降低代理人在各商務主機間移動時為商務主機推導金鑰所需耗費的運算成本以及行動代理人所攜帶的資料量。

**關鍵詞**： 行動代理人、存取控制、角色關係、單向雜湊函數、金鑰管理

## 1. 前言

在這個資訊科技的時代，每一個人都逐漸地依賴起虛擬的網路以期更有效率地完成工作。當我們都能很方便的透過網路來聯繫到對方時，也會思考個人的資訊是否有可能已洩露出去。因此，保障機密資訊的安全就變成一個很重要的議題；而加密的方式也就常被學者們提出討論，在反覆修正加密演算法的環境中，有關於加密的方式都已經具有相當的安全性。儘管如此，如果加密的方法再多難被破解，若開門的鑰匙沒有被有效的管理，就像是一道上了很多鎖的門，但鑰匙卻輕易地就讓攻擊者取得它們。在這樣的情況下，任何加密方法都會失敗。因此，本研究為了確保行動代理人的資源可以得到合理的分配和有效的保護，並且降低商務主機在執行運作時所耗費的資源，我們分析行動代理人執行的安全性，和討論如何有效的處理行動代理人的金鑰管理跟存取控制方式。故我們提出以角色階層的概念運用在行動代理人結構中，並將金鑰的運算採用單向雜湊函數運算，降低商務主機在執行運作時所耗費的資源，以期達到高效率和安全性兼具體的架構。

在本篇論文中，共分為四節。第二節主要描述有關金鑰管理和存取控制的文獻；第三節提出我們的方法。第四節說明我們對我們所提出方法的綜合分析；同時我們也將我們的方法跟 Lin-Ou-Hwang 的方法進行效率及安全性的比較。在第四節，我們則對我們所提出的方法做一個總結。

## 2. 文獻探討

由於行動代理人為了完成使用者所指派的任務，因此常需要在不同的商務主機之間移動，為了確保行動代理人的資源可以得到合理的分配和有效的保護，簡化行動代理人所攜帶的金鑰數目及適當的存取控制機制是不可或缺的。在此章節中我們將會回顧 Volker 和 Mehrdad [8]所提出行動代理人架構、金鑰管理的方法及其缺點，並探討由 Iuon-Chang Lin, Hsia-Hung Ou, and Min-Shiang Hwang 三位學者[7]針對 Volker 和 Mehrdad 方法的缺失所提出的改進方法及分析，以及 Li 等人[6]所提出的在分散式系統下基於角色關聯的金鑰管理方法。

### 2.1 Volker 和 Mehrdad 的行動代理人架構

行動代理人在開放的網路環境下完成使用者交付的任務時，保護行動代理人不受到惡意的主機或其他代理人的攻擊與威脅是很大

的挑戰。因為行動代理人在各主機執行任務時，除了無法被發送的代理人主機控制外，代理人內部的資料包括程式碼，還會攜帶許多為達成任務而用來參考的資訊，例如：路徑、憑證...等，都會曝露在商務主機的執行平台上。因此，為了保護資訊的安全，Volker 和 Mehrdad 以加密確保代理人資料的機密性，並透過數位簽章之核對去實現代理人資料的完整性跟驗證性。根據 Volker 和 Mehrdad 所提出的樹狀架構表示行動代理人所攜帶的內容，如圖 1。

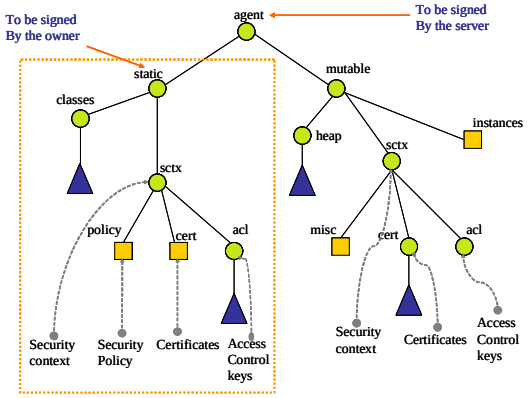



**圖 1　Volker 和 Mehrdad 所提出的行動代理人樹狀架構**

Volker 和 Mehrdad 所提出的行動代理人架構主要是將行動代理人所攜帶的內容分成不可變動(static)和可變動(mutable)兩個部份：

1. **不可變動(static)部份：**包括安全政策(security policy)、權限分級(Class)、憑證(Certificates)、存取控制金鑰(Access control keys)...等。表示在行動代理人的生命週期內都不會改變的資料，避免行動代理人被惡意的主機竄改，因此代理人主機會對行動代理人不可變動的資料使用數位簽章以保護資料的完整跟驗證。

2. **可變動(mutable)部份：**會變動的資料是因為當行動代理人拜訪各個商務主機時，代理人的狀態或收集到的資料會被拜訪的主機修改而有所變動，因此為了保持行動代理人內容的完整性，每一個被行動代理人拜訪的主機都必須對代理人(agent)做簽章，透過這個簽章代理人主機可以與拜訪的商務主機作驗證，讓主機間形成一個互信的關係。

### 2.1.1 Volker 和 Mehrdad 的存取控制與金鑰管理

有鑑於加密的安全性主要是取決於金鑰的安全，如果金鑰沒有被適當的管理，惡意的主機就有辦法取得它們，導致加密的失效。所以 Volker 和 Mehrdad 將代理人的內容以樹狀結構表示，除了可以輕易地將資料搜尋、擷取、插入、分群外。另一方面，這樣的安排也可以防止未授權的商務主機去存取行動代理人機密的資料。首先，代理人主機會替每一個要拜訪的主機建立一個專屬的目錄，並使用各商務主機的公開金鑰對目錄做加密，目錄內會存入金鑰和相關的安全管理資訊。商務主機可以根據自己的私密金鑰解開對應的目錄拿到授權的金鑰跟檔案，並利用這些金鑰解開符合自己權限的檔案(Class)。根據以上的說明，Volker 和 Mehrdad 存取控制和金鑰管理的辦法可以表示如圖 2。

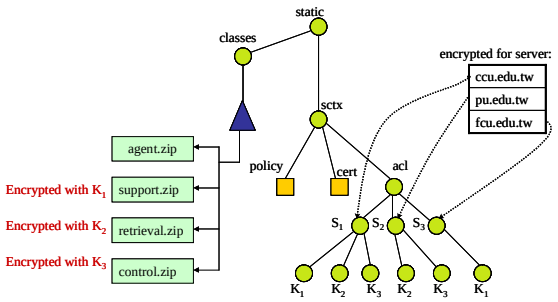



**圖 2　Volker 和 Mehrdad 的存取控制與金鑰管理範例**

由圖 2 可知，當行動代理人拜訪商務主機時，如 ccu.edu.tw，該商務主機就會用自己的私密金鑰解開專屬的目錄 $S_1$，根據裡面存放的金鑰 $K_1$、$K_2$、$K_3$ 解開對應的權限檔案 support.zip、retrieval.zip、control.zip，利用對應的金鑰及目錄進而達到對於行動代理人存取控制和金鑰管理的目的。

### 2.1.2 Volker 和 Mehrdad 方法的缺點

根據 Volker 和 Mehrdad 的存取控制與金鑰管理策略，每個行動代理人會拜訪的商務主機目錄內都會存放著依據商務主機權限而檔案數多寡不同的金鑰量。因此 Volker 和 Mehrdad 所提出的方法會產生以下兩個問題：

1. **資訊重覆：**在 Volker 和 Mehrdad 的行動代理人樹狀結構下，我們發現解開各權限檔案的金鑰重複儲存在各個主機的目錄

下產生資訊重覆。如圖 2 的例子，由於商務主機 $S_1$ 跟 $S_2$ 皆擁有 retrieval.zip 的權限，因此金鑰 $K_2$ 會重複出現在 $S_1$ 跟 $S_2$ 的目錄下，造成行動代理人程式冗長和繁雜，浪費行動代理人的資源。

2. **計算量增加：**由於代理人主機會替每一個拜訪的主機建立一個專屬的目錄，並使用各商務主機的公開金鑰對目錄做加密。因此當拜訪的商務主機數量越多，代理人主機就必須使用更多的公開金鑰加密運算，以保護各主機目錄的安全，造成計算量增加、運算的時間變長，導致效能不好。

### 2.2 Lin-Ou-Hwang 改善金鑰管理之策略

在本節我們將探討 Iuon-Chang Lin, Hsia-Hung Ou, and Min-Shiang Hwang 三位學者在 2004 年針對 Volker 和 Mehrdad 的行動代理人金鑰管理跟存取控制所提出的兩個改進方法。

#### 2.2.1 Lin-Ou-Hwang 提出的改進方法一

第一個方法主要是 Iuon-Chang Lin 等三位學者參考 Akl 和 Taylor[5]所提出由上到下的階層式觀念而改進的方法。如圖 3 所示，Akl 和 Taylor 所提出方法中，葉節點(leaf nodes)代表不同權限檔案的金鑰物件，如 $K_1$、$K_2$、$K_3$ 分別代表權限檔案 support.zip、retrieval.zip、control.zip，而中間節點(internal nodes)則表示代理人要拜訪的各個商務主機，如 $S_1$、$S_2$、$S_3$，分別代表主機 ccu.edu.tw、pu.edu.tw 和 fcu.edu.tw。

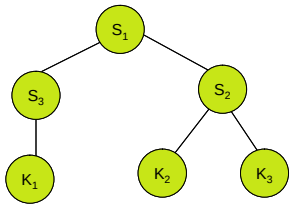



**圖 3　Akl 和 Taylor 的階層結構範例**

對於每一個被行動代理人拜訪的商務主機，代理人主機會為他們建立一個目錄，每一個商務主機的目錄中會存放一把 superkey，商務主機可利用自己的私密金鑰解開專屬的目錄取得 superkey，並根據行動代理人所攜帶的資料，推導出加密和解密代表不同權限檔案物件的金鑰。也就是說，若有一個商務主機能執行一個權限，即代表著這個商務主機能推導出物件的解密金鑰，進而存取不同權限的檔案。

**A. 金鑰產生**

代理人主機會依照以下的步驟產生相關的金鑰跟參數：

1. 代理人主機隨機選出兩個大的質數 $p$、$q$ 以計算公開的參數 $n=p*q$，代理人主機並另外選出一個安全不公開的參數 $K_0$ 用在之後金鑰產生的方程式上。

2. 代理人主機選擇一組不同的質數{$P_1$、$P_2$、…、$P_m$}給階層內所有的商務主機節點 {$N_1$、$N_2$、…、$N_m$}，然後代理人主機會根據公式(1)計算出每一個節點的參數{$t_1$、$t_2$、…、$t_m$}。

$$t_i = \prod_{N_I < N_I} P_j , \qquad (1)$$

此式中 $N_j \leqq N_i$.表示 $N_j$ 不是 $N_i$ 的子節點並且 $t_i$ 需滿足以下兩個條件:

$$t_i \mid t_j \quad \text{且} \quad N_j \leqq N_i , \qquad (1\text{-}1)$$

$$\gcd_{Ni<Nj}(t_i) \nmid t_j , \qquad (1\text{-}2)$$

3. 代理人主機並根據公式(2)計算出每一個商務主機節點的金鑰。

$$K_i = K_0^{t_i} \bmod n , \qquad (2)$$

**B. 金鑰推導**

當商務主機對應的節點為 $N_i$ 時，若節點之間的關係是 $N_j < N_i$，則商務主機 $N_i$ 能根據公式(3) 推導出 $N_j$ 的金鑰，其公式如下：

$$K_i = K_0^{\frac{t_j}{t_i}} \bmod n = K_0^{\left(\frac{t_j}{t_i}\right)} \bmod n$$

$$= K_0^{\frac{t_j}{t_i}} \bmod n , \qquad (3)$$

#### 2.2.2 Lin-Ou-Hwang 提出的改進方法二

由於第一個方法所使用的公開參數 $t_i$ 值會隨著權限資料夾和商務主機的數量增加而成長，使得行動代理人公開參數所佔用的空間提升，因此 Lin-Ou-Hwang 提出第二種方法，是

利用歐基里德演算法公式產生金鑰的參數，使得公開參數跟權限資料夾數量一樣，降低公開參數所佔用的空間。

**A. 金鑰產生**

代理人主機會依照以下的步驟產生相關的金鑰跟參數：

1. 代理人主機隨機選出兩個大的質數 $p$、$q$ 以計算公開的參數 $n=p*q$，代理人主機並另外選出一個與 $n$ 互質，安全不公開的參數 $K_0$ 用在之後金鑰產生的方程式上，$K_0$ $[2,n-1]$。
2. 原先代理人主機會選擇一組不同的質數 $\{P_1、P_2、\ldots、P_m\}$給階層內所有的節點 $\{N_1、N_2、\ldots、N_m\}$，然後代理人主機會根據公式(1)計算出每一個節點的參數$\{t_1、t_2、\ldots、t_m\}$ ，但隨著拜訪的商務主機物件越多，$t_i$ 值也會隨著增加，因此為了降低使用的參數空間，代理人主機會選擇一組不同的數值 $\{e_1、e_2、\ldots、e_u\}$ 給物件節點，且此處 $e_i$ 與 $(p-1)(q-1)$需互質。
3. 然後代理人主機根據歐基里德演算法計算出 $\{d_1、d_2、\ldots、d_r\}$，其歐基里德演算法公式(4)如下：

$$e_i \times d_i \equiv 1 \bmod \phi(n), \qquad (4)$$

$\phi(n)$ 表示尤拉商數，定義為小於 $n$ 且 $n$ 與互質的所有整數的個數，例如

$$\phi(n)=\phi(15)$$
$$=(1,2,4,6,7,8,11,13,14)=9$$

換言之 $d_i = e_i^{-1} \bmod \phi(n)$

4. 因此代理人主機能根據公式(5)推導出加密各個物件的金鑰

$$DK_i = K_0^{d_i} \bmod n,\ 1<i<u \qquad (5)$$

5. 而代理人主機推導各商務主機的金鑰則為公式(6)

$$SK_i = K_0^{\Pi_{N_j<N_i} d_j} \bmod n,\ 1<i<r, \qquad (6)$$

**B. 金鑰推導**

當商務主機對應的節點為 $N_i$ 時，若物件和商務主機之間的關係是 $N_j < N_i$ ，則商務主機 $N_i$ 能根據公式(7) 推導出物件 $N_j$ 的金鑰，其公式如下：

$$DK_j = SK_i^{\prod_{N_k<N_i \text{ and } N_k \neq N_j} e_k} \bmod n,$$
$$= \left(K_0^{\prod_{N_k<N_i} d_k}\right)^{\prod_{N_k<N_i \text{ and } N_k \neq N_j} e_k} \bmod n,$$
$$= K_0^{d_j} \bmod n, \qquad (7)$$

## 2.3 Li-Yang-Cheung 基於角色關聯的金鑰管理

在分散式系統內，有無數的權限和數以千計的使用者位於不同的網域中，導致控管使用者對電腦內資源許許可權的存取控制串列(ACL)，會因為不同的使用者需要各類型檔案或資料的存取權限而增加存取控制串列的數量。為了改善存取控制串列對於系統所形成的負擔，以角色為基礎的存取控制(RBAC)被 Sandhu [9]於 1996 年時提出。RBAC 意指存取的決定是基於角色擁有的權限，每個角色代表可執行特定工作或任務的集合，定義物件，如檔案或資料可以被操作。系統管理者會根據使用者的責任來指派合適的角色，這樣使用者就可以依照被賦予的角色去執行系統管理者所授權的存取活動。但在實際情況上，角色之間往往會有重疊的責任和權限，擁有不同角色的使用者常會執行相同的權限，而一些基本的權限也通常能被大部份的使用者執行，造成系統效能和管理上的麻煩。因此 RBAC 的基本概念還包含了角色的階層性，階層高的角色可以擁有階層低的角色的權限。舉例說明：假若角色 A 在階層關係中是角色 B 直接或間接的母節點，則角色 A 可以繼承所有角色 B 的權限。Li-Yang-Cheung[6]三位學者將 RBAC 的角色階層概念延伸運用在金鑰管理上，建構的階層關係中分為角色跟物件兩種，角色可以根據產生金鑰的規則推導出金鑰，進而存取代表特定權限的物件，整個架構稱作以角色為基礎的金鑰管理(RBEM)方法。

在這個方法中 Li-Yang-Cheung 利用單向的雜湊函數來產生金鑰，使得計算上更有效率和擁有不可更改的安全性。根據 Li-Yang-Cheung 以角色為基礎的金鑰管理方法，產生角色金鑰的規則分為三點：

規則 1. 沒有母節點的角色，網路安全管理者

會分配給此角色一個任意的金鑰。

規則 2. 假若某一個角色只有一個直接的母節點角色，網路安全管理者會依照雜湊函數 $H_i(K)$ 來產生金鑰。$i$ 表示這個角色是母節點自左往右邊數第 $i$ 個子節點；$K$ 則代表此母節點角色的金鑰。

規則 3. 假若某一個角色有超過一個以上的母節點，網路安全管理者會依照組合的雜湊函數 $H_i\left(H_i(K_x),H_j(K_y),...\right)$ 來產生金鑰。$K_x,K_y,...$ 表示此角色自左往右邊數每一個母節點的金鑰；$i$ 代表是有金鑰 $K_x$ 的母節點從左往右邊數第 $i$ 個子節點；$j$ 是代表有金鑰 $K_y$ 的母節點從左往右邊數第 $j$ 個子節點角色，以此類推。

在這個以角色為基礎的金鑰管理方法中，角色的金鑰都是根據上述的三項規則來產生，而加密或解密物件的金鑰，則會根據物件與角色之間的關係而有所不同，若物件被超過一個以上的角色直接存取，則物件的金鑰的產生方式會與角色金鑰的規則 3 相同；若是物件只能被一個角色直接存取，則物件的金鑰會與此角色的金鑰相同，以圖 4 來說明。

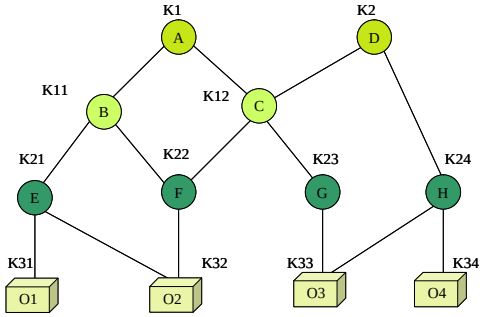



**圖 4　推導在階層關係中角色和物件的金鑰**

在角色階層關係中，每個母節點最大的子節點數量是 2。因此網路安全管理者會選擇兩個雜湊函數 $H_1$ 和 $H_2$，並且分配金鑰 $K1$ 和 $K2$ 給沒有母節點的角色 A 和 D，其中角色跟物件的金鑰產生說明如下：

**A. 角色**

1. 當角色只有唯一的母節點時。如角色 B，管理者會根據金鑰產生的規則 2 推導出金鑰 $K11$，也就是 $K11=H_1(K1)$。$K1$ 是角色 A 的金鑰，且角色 A 是角色 B 唯一的母節點。

2. 當角色有超過一個以上的母節點時。如角色 C，管理者會根據金鑰產生的規則 3 推導金鑰 $K12$，也就是 $K12=H_2\left(H_2(K1),H_1(K2)\right)$，這種有超過一個以上的母節點所產生的金鑰，當代表母節點的角色想要推導出其子節點的金鑰時，母節點必須需要其他母節點的參數值。以圖 4 所示，角色 D 跟角色 A 是角色 C 的母節點，因此角色 D 必須要有角色 A 的金鑰參數 $H_2(K1)$，配合角色 D 的金鑰 $K2$ 才能推導出來角色 C 的金鑰 $K12=H_2\left(H_2(K1),H_1(K2)\right)$；而角色 A 則要擁有角色 D 的金鑰 $H_1(K2)$，才能推導出來角色 C 的金鑰 $K12=H_2\left(H_2(K1),H_1(K2)\right)$。

**B. 物件**

1. 當物件只有被一個角色直接存取時。如物件 O1 只能被角色 E 直接存取，則物件的金鑰值就跟角色 E 的金鑰相同。

2. 當物件被超過一個以上的角色直接存取時，物件的金鑰產生方式就會跟角色的金鑰產生方式相同。以物件 O2 來說，O2 能被角色 E 和角色 F 直接存取，根據金鑰產品的規則 3，則物件 O2 的金鑰為 $K32=H_2\left(H_2(K21),H_1(K22)\right)$。

此方法是藉助不能反轉推算的單向雜湊函數以達到安全性。根據 Li-Yang-Cheung 的金鑰產生規則，即使是代表某一個角色的使用者也無法推算出其母節點的金鑰。也就是說，此使用者無法存取其母節點被授權的資源或權力。除此之外，如果某一角色能被兩個母節點角色直接存取，這兩個母節點角色也無法經由產生角色的金鑰規則或角色內儲存的資訊而去推導出彼此的金鑰，進而存取彼此的資源或權力。以圖 5 舉例說明，根據產生角色金鑰的規則，代表角色 D 的使用者會擁有參數值 $K2$ 和 $H_2(K1)$，進而能推算出角色 C 的金鑰 $K12=H_2\left(H_2(K1),H_1(K2)\right)$；同樣的，代表角色 A 的使用者也能根據所擁有的參數值 $K1$ 和 $H_1(K2)$ 推算出角色 C 的金鑰。有鑑於這方法角色 D 只能知道將角色 D 的金鑰經過雜湊函數

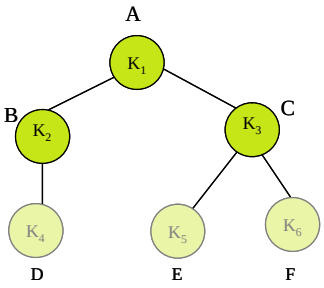
計算的參數值 $H_2(K1)$，因此角色 D 無法根據此參數值推導出角色 D 的金鑰。同理類推，很顯然地，角色 A 也無法得知角色 D 的金鑰，因此雙方都無法存取彼此的資源或權力。

## 3. 我們的方法

本節中，我們結合了 Lin-Ou-Hwang[7]和 Li-Yang-Cheung[6]的方法，形成一個以角色關聯性為基礎的樹狀結構，在金鑰的運算上使用單向的雜湊函數以減少先前 Lin-Ou-Hwang 方法中指數運算的複雜性，降低系統耗費的計算時間，簡化金鑰的儲存空間。首先我們會將商務主機或權限檔案用節點來表示，並定義金鑰產生的方法且舉例說明，接著會介紹行動代理人在執行任務過程中所需要攜帶的資料，最後並推導行動代理人抵達要拜訪的商務主機時，商務主機解密金鑰的步驟。

### 3.1 金鑰管理跟存取控制

在描述我們的方法前，我們必須針對節點的身份，也就是商務主機和代表權限檔案的物件來表示在我們的方法中代表的意義，如圖 5 所示。




註：D、E、F是虛擬的節點，分別用來代表三個不同權限檔案的金鑰物件 - support.zip、retrieval.zip 和 control.zip，並非代理人實際會攜帶的資料。

**圖 5　商務主機和權限物件的關係圖**

節點 A、B、C 代表三個商務主機的專屬目錄；D、E、F 則表示三個不同權限檔案的虛擬物件，如 support.zip、retrieval.zip 和 control.zip。每個專屬的目錄都會被代理人主機用商務主機的公開金鑰加密，而目錄內所存放的是根據我們的方法所產生的參數值，參數值的產生會在後續章節說明，在此並不贅述。在本研究中，必須要強調的是物件 D、E、F 是虛擬的節點，並非像商務主機的專屬目錄是代理人實際上會攜帶的資料，而物件所代表的意義是當某一商務主機用該主機的私密金鑰解開所屬的目錄時，可以根據目錄內的金鑰和相關參數去推導出解密物件的金鑰。也就是說，商務主機就可以根據這些金鑰，解開被代理人主機所授權的權限檔案，如 support.zip、retrieval.zip 或 control.zip。以圖 5 來說明，商務主機 A 利用自己的私密金鑰解開目錄 A，取得金鑰 $K_1$ 和相關的參數值後，可根據我們的存取規則推導出解密虛擬物件 D、E、F 的金鑰 $K_4$、$K_5$、$K_6$，也就表示可以解開被代理人主機授權的權限檔案 support.zip、retrieval.zip 和 control.zip。

#### 3.1.1 金鑰產生

代理人主機產生相關的金鑰跟參數的方法如下：

1. 代理人主機會分配給每一個節點(在此指商務主機和物件)一把金鑰($K_{xy}$)。$xy$ 表示節點的階層關係，$x$ 表示由上到下階層的順序，$0 \le x \le i$，$i$ 為最大的階層數；$y$ 代表從左到右商務主機或物件節點的順序，$1 \le y \le j$，$j$ 為從左到右最大的節點數。以圖 6 來說明，$i$=2、$j$=3。

2. 每一把金鑰的設計是根據節點之間的階層關係和單向的雜湊函數( $H$ )來產生各個商務主機或物件的金鑰 ($K_{xy}$)。

   假若某一節點沒有母節點時，則代理人主機會隨機產生一把金鑰分給此節點，以圖 6 為例。節點 A 沒有母節點，因此代理人主機會隨機產生一把金鑰($K_{01}$)分配給節點 A，01 表示節點 A 在角色階層內的關係。

   當子節點只有一個直接的母節點時，子節點的金鑰會根據母節點的金鑰($K_{ij}$)和節點的階層關係($xy$)，利用單向的雜湊函數($H$)推算產生，如圖 6，節點 C 只有一個母節點 A，則節點 C 會根據公式(8)產生金鑰。

   $$K_{xy} = H(xy, K_{ij}), \qquad (8)$$

   因此節點 C 金鑰($K_{12}$)的推算過程為：$K_{12} = H(12, K_{01})$。

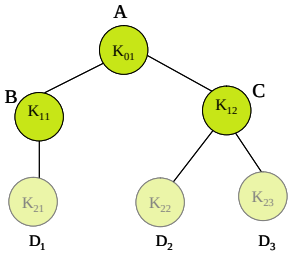



**圖 6　節點間的階層關係圖**

假若當子節點有超過一個直接存取的母節點時，節點的金鑰要利用組合的雜湊函數公式(9)來產生。

$$K_{xy} = H(xy, H(xy, K_{ij}), ..., H(xy, K_{ij})), (9)$$

以圖 7 為例，節點 C 有兩個母節點 A 和 D，根據步驟 1 推導出節點 D 的金鑰為 $K_{02}$；節點 A 的金鑰為 $K_{01}$，因此節點 C 的金鑰產生表示為：$K_{12} = H(12, H(12, K_{01}), H(12, K_{02}))$

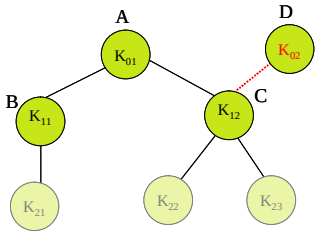



圖 7　改變階層關係

### 3.1.2　金鑰產生範例

為了說明上述 3.1.1 的定義，我們舉例圖 8 如下：

**範例一：**假設有四個商務主機 A、B、C、D，和四個代表不同權限檔案的金鑰物件 $D_1$、$D_2$、$D_3$、$D_4$ 共 8 個節點。根據圖 7 節點的階層關係，代理人主機推導出各節點的金鑰計算步驟表示如下：

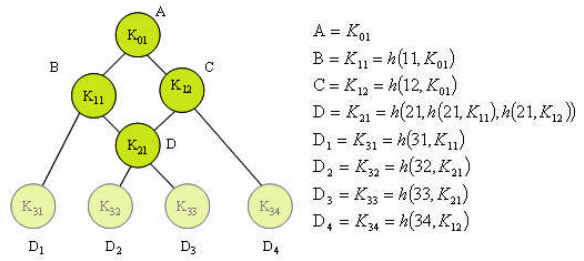



**圖 8　金鑰產生範例一**

步驟 1：由於節點 A 沒有母節點，因此代理人主機會隨機產生一把金鑰($K_{01}$)分配給節點 A，01 表示節點 A 的階層關係。

步驟 2：而節點 B 跟 C 的階層關係為 11 和 12，且節點 B 跟 C 只有一個母節點 A，則節點 B 跟 C 會根據公式(8)產生金鑰為 $B = K_{11} = h(11, K_{01})$ 和 $C = K_{12} = h(12, K_{01})$。

步驟 3：若當子節點有超過一個的母節點時，節點的金鑰則要利用組合的雜湊函數公式(9)來產生。本範例中的節點 D 就有兩個母節點 B 跟 C，因此節點 C 的金鑰產生表示為 $D = K_{21} = h(21, h(21, K_{11}), h(21, K_{12}))$

步驟 4：因此物件 $D_1$、$D_2$、$D_3$、$D_4$ 可根據節點 B、C、D 的金鑰和公式(8)產生金鑰為 $D_1 = K_{31} = h(31, K_{11})$、$D_2 = K_{32} = h(32, K_{21})$、$D_3 = K_{33} = h(33, K_{21})$、$D_4 = K_{34} = h(34, K_{12})$

### 3.2　行動代理人攜帶的資料

根據 Volker 和 Mehrdad 的所提出的行動代理人樹狀結構，是將行動代理人所攜帶的資料以樹狀結構來表示，架構可分為不可變動和可變動兩部份。而本研究主要是為了確保行動代理人的資源可以得到合理的分配和有效的保護，因此主要是針對行動代理人所攜帶的金鑰數目及存取控制機制來探討，所以我們在此僅討論有關不可變動的部份，可變動的部份就不再贅述。在我們的方法中，行動代理人為了讓商務主機能推導出所被授權的權限檔案所需要攜帶的資料包括：

1. **權限檔案(classes)：**代表不同權限的檔案，如 agent.zip、support.zip、retrieval.zip 和 control.zip。

2. **安全相關資訊(security context)：**行動代理人在主機之間運行，執行任務時所需要的資訊，如雜湊函數(hash)、安全政策(policy)、推導金鑰所需的數學公式(program)、憑證(certificate)、存取控制金鑰(access control keys)。

根據 Volker 和 Mehrdad 的行動代理人樹狀結構，我們將以角色關聯性為基礎的存取控制與金鑰管理方法，以圖 9 表示如下：

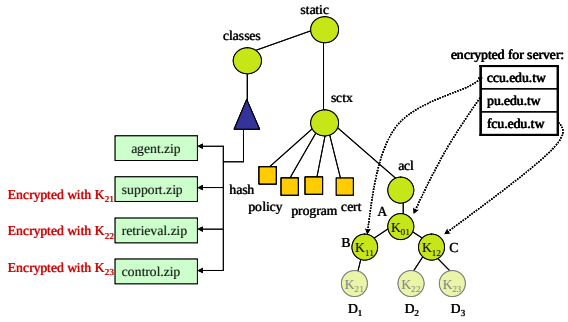



**圖 9 行動代理人的存取控制與金鑰管理方法**

代理人要拜訪的商務主機分別為 ccu.edu.tw、pu.edu.tw、fcu.edu.tw 三個主機。每一個代理人要拜訪的商務主機，代理人主機都會替主機建立一個專屬的目錄，以圖 9 來說，商務主機 ccu.edu.tw 的目錄為 A、商務主機 pu.edu.tw 的目錄為 B、商務主機 fcu.edu.tw 的目錄為 C。這些目錄代理人主機分別會用商務主機的公開金鑰加密，目錄內會存入根據我們的方法所產生的金鑰和相關的參數。這樣，商務主機就可以根據自己的私密金鑰解開對應的目錄拿到代理人主機授權的金鑰，該主機並利用所被授權的金鑰根據我們所提出的方法推導出符合自己權限的檔案(classes)；在圖 9 中，權限(classes)分為 agent.zip、support.zip、retrieval.zip 和 control.zip 四個檔案。其中 agent.zip 表示代理人基本的運作執行，不牽涉機密性，故不需要加密，但因為其他的權限檔案代表各角色間具有不同的權限，因而能執行不同的功能，故檔案必須是機密的。所以，這些權限檔案分別被代理人主機用金鑰 $K_{21}$、$K_{22}$、$K_{23}$ 加密。因此當商務主機要去存取特定的權限檔案時，只要商務主機能推導出對應的解密金鑰 $K_{21}$、$K_{22}$ 或 $K_{23}$，代理人就可以執行符合自己權限的檔案，利用這樣的金鑰管理跟存取控制機制保護各個商務主機需要執行不同類型的權限檔案時的安全。

### 3.3 金鑰推導

當行動代理人抵達一商務主機時，商務主機便能先用自己的私密金鑰解開專屬的目錄，根據目錄內的金鑰和其它相關資訊，商務主機推算權限檔案的解密金鑰過程可分成兩大步驟：

**步驟一：**被代理人拜訪的商務主機會先用自己的私密金鑰解開專屬的目錄，取得代理人主機授權的金鑰。

**步驟二：**商務主機根據代理人所攜帶的雜湊函數(hash)、數學公式(program)及以角色關聯性為基礎的存取控制金鑰(access control keys)所形成的樹狀架構來產生金鑰。

以圖 9 來說明，代理人主機首先根據我們 3.1.1 所提出的金鑰產生方式，將金鑰 $K_{01}$、$K_{11}$、$K_{12}$ 分別放在專屬的主機目錄下，並使用各商務主機的公開金鑰加密該主機的目錄，因此商務主機 A 的目錄裡存放了金鑰 $K_{01}$；主機 B 的目錄裡存放了金鑰 $K_{11}$；主機 C 的目錄裡存放了金鑰 $K_{12}$。當代理人拜訪到商務主機 A 時，商務主機 A 可根據 3.1 金鑰產生的公式，計算出子節點主機 B 的金鑰為 $K_{11} = h(11, K_{01})$、主機 C 的金鑰為 $K_{12} = h(12, K_{01})$，然後推算出節點 $D_1$、$D_2$、$D_3$ 所代表的金鑰 $K_{21} = h(21, K_{11})$、$K_{22} = h(22, K_{11})$、$K_{23} = h(23,, K_{11})$，因此商務主機 A 能推導出解密 $D_1$、$D_2$、$D_3$ 的金鑰，取得代理人主機許可的權限檔案 support.zip、retrieval.zip 和 control.zip；而主機 B 能取得解密 $D_1$ 的金鑰，取得權限檔案 support.zip；主機 C 能取得解密 $D_2$、$D_3$ 的金鑰，得到權限檔案 retrieval.zip 和 control.zip；其他，代表權限檔案的金鑰物件 $D_1$、$D_2$、$D_3$、$D_4$，因為都只有一個母節點，且階層關係分別為 31、32、33、34。因此金鑰物件 $D_1$、$D_2$、$D_3$、$D_4$ 的金鑰根據公式(8)可表示為 $D_1 = K_{31} = h(31, K_{11})$、$D_2 = K_{32} = h(32, K_{21})$、$D_3 = K_{33} = h(33, K_{21})$、$D_4 = K_{34} = h(34, K_{12})$。

## 4. 分析與討論

在本章節中，我們會藉由範例來比較本研究和 Lin-Ou-Hwang 所提出的存取控制與金鑰管理方法在資料量和運算量的差異，並藉由改變藉節點數的多寡或增加新的聯結來觀察兩個方法在資料量上的影響，最後會針對我們的方法做安全性的分析。

### 4.1 效能分析

#### 4.1.1 行動代理人的資料量的差異

為了說明我們的方法和 Lin-Ou-Hwang 方法所產生的資料量不同，我們假設行動代理人要拜訪四個商務主機 A、B、C、D 以執行使用者交付的任務。代理人出發前，代理人主機首先會將商務主機能夠執行代理人的權限區分為 $D_1$、$D_2$、$D_3$ 和 $D_4$ 四個檔案，並且根據商務主機之間的繼承關係建立階層結構，如圖 10 所示。

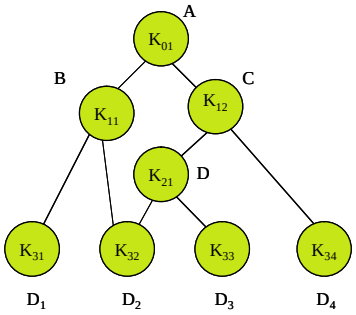



**圖 10 物件和商務主機階層範例**

地位較高階層的商務主機能繼承地位較低的商務主機所能執行的權限，因此商務主機 A 會繼承主機 B 和 C 的權限，並由於主機 C 會繼承主機 D 的權限，因此商務主機 A 能執行 $D_1$、$D_2$、$D_3$ 和 $D_4$ 全部的權限。在此根據圖 10 的範例，我們將 Lin-Ou-Hwang 和本研究的兩個方法，透過圖 11 和圖 12 來說明在行動代理人不可變動的資料中，為了讓商務主機推導出所能解開權限檔案的金鑰，行動代理人必須攜帶的資料。兩個方法分別如下所示：

#### A. Lin-Ou-Hwang 的存取控制與金鑰管理

根據 Lin-Ou-Hwang(2004)所存取控制與金鑰管理方法[7]，各商務主機可根據自己所持有的金鑰、產生金鑰的公式和公開參數值，進行指數運算以求得解密權限檔案的金鑰值，如圖 11 所示。

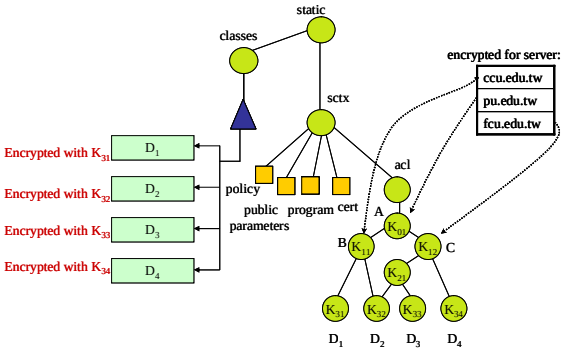



**圖 11 Lin-Ou-Hwang 的存取控制與金鑰管理**

行動代理人為了讓商務主機能推導出可解開權限檔案的金鑰，代理人所攜帶的資料為：憑證(certificate)、數學公式(program)、安全政策(security policy)、公開參數(public parameters )、存取控制金鑰(access control keys)。而 Lin-Ou-Hwang 所提出的推導金鑰的方法有兩種，以圖 10 的範例來表示，

(1) Lin-Ou-Hwang 的方法一：根據 2.2.1 小節，每一個節點都會有一個公開參數 $t_i$ 值，$t_i$ 值的大小會隨著權限資料夾和商務主機的數量增加而成長，使得行動代理人公開參數所佔用的空間提升，我們將 Lin-Ou-Hwang 方法一的 $t_i$ 值計算表示如圖 12。

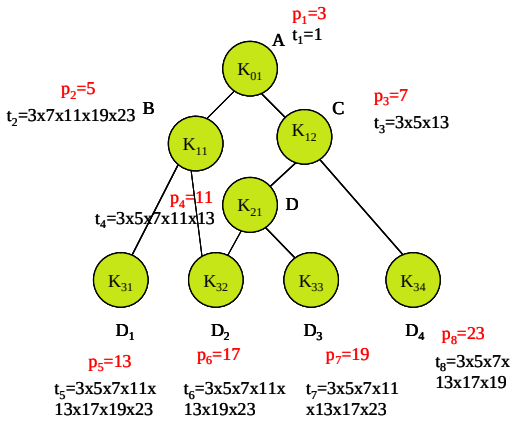



**圖 12 Lin-Ou-Hwang 的 $t_i$ 值計算表示**

若要從節點 D 推算出 $D_3$ 的金鑰，根據 Lin-Ou-Hwang 方法一的公式 $K_i = K_0^{t_i} \bmod n$，節點 D 的金鑰為 $D = K_0^{t_4} \bmod n$，則節點 D 可根據自己所持有的金鑰 $K_{21}$、推導金鑰的公式

$$K_j = K_i^{\frac{t_j}{t_i}} \bmod n$$

$$= K_0^{t_i(\frac{t_j}{t_i})} \bmod n = K_0^{t_j} \bmod n$$

，及公開參數值 $t_1$~$t_8$ 以推算出

$$D_3 = D^{\frac{t_7}{t_4}} \bmod n = \left(K_0^{t_4}\right)^{\frac{t_7}{t_4}} \bmod n$$

。

$$= K_0^{t_7} \bmod n$$

由此可知， Lin-Ou-Hwang 的方法一，就金鑰的數量來說，若代理人所要拜訪的商務主機有 n 個，則必須儲存 n 把經由指數運算產生的金鑰；就參數的數量而言，行動代理人需要儲存 $i$ 個節點的 $t_i$ 值，以圖 10 的範例來說也就是 4 把金鑰及 $t_1$~$t_8$ 八個參數值。

(2) Lin-Ou-Hwang 的方法二：Lin-Ou-Hwang 所提出第二種方法，是利用歐基里德演算法公式 $e_i \times d_i \equiv 1 \mod \phi(n)$，來做為產生金鑰的參數值 $e_1$、$e_2$、…、$e_r$，因此根據圖 10 的範例，我們將 Lin-Ou-Hwang 方法一的金鑰計算表示如圖 13。

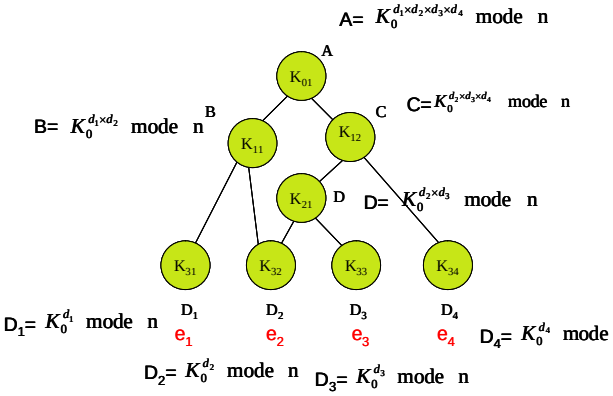



**圖 13　Lin-Ou-Hwang 方法二的金鑰計算表示**

若要從節點 D 推算出 $D_3$ 的金鑰，根據 Lin-Ou-Hwang 方法二的公式

$SK_i = K_0^{\prod_{N_j<N_i} d_j} \bmod n,\ 1<i<r$ ，節點 D 的金鑰為 $D = K_0^{d_2\times d_3} \bmod n$ ，則節點 D 可根據自己所持有的金鑰 $K_{21}$、推導金鑰的公式

$DK_j = SK_i^{\prod_{N_k<N_i \text{ and } N_k\neq N_j} e_k} \bmod n$ ，及公開參數值 $e_1$、$e_2$、$e_3$、$e_4$，以推算出

$$D_3 = D^{e_2} \bmod n$$

$$= \left(K_0^{d_2\times d_3}\right)^{e_2} \bmod n = K_0^{d_3} \bmod n$$

。

由此可知，Lin-Ou-Hwang 的方法二，就金鑰的數量來說，若代理人所要拜訪的商務主機有 n 個，則必須儲存 n 把經由指數運算產生的金鑰；就參數的數量而言，行動代理人只需要儲存 $r$ 個 $e_i$ 值，r 表示權限檔案的個數，以圖 10 的範例來說也就是四把金鑰及 $e_1$、$e_2$、$e_3$、$e_4$ 四個參數值，比起方法一，降低公開參數的數量，減少參數所佔用行動代理人的儲存空間。

## B.　我們方法的存取控制與金鑰管理

本研究所提出的的存取控制與金鑰管理架構，是利用角色之間的繼承關係進行單向的雜湊函數運算，進而推導出解密授權檔案的金鑰。如圖 14 所示。

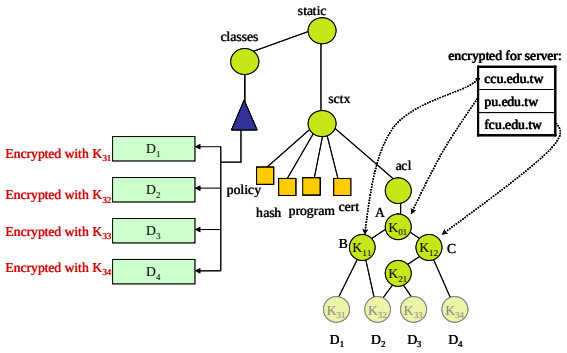



**圖 14　本研究所提出的存取控制與金鑰管理**

行動代理人為了讓商務主機能推導出可解開權限檔案的金鑰，代理人所攜帶的資料為：憑證(certificate)、數學公式(program)、安全政策(security policy)、雜湊函數(hash)、存取控制金鑰(access control keys)。因此以圖 10 的範例來說，本研究的金鑰計算表示如圖 15。

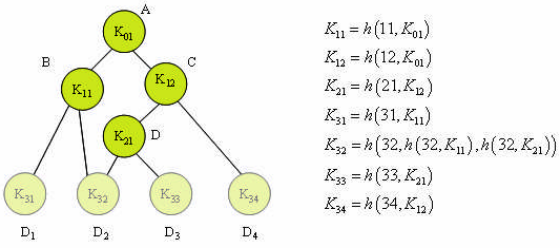



**圖 15　本研究的金鑰產生範例**

因此，在我們所提出的方法，若要能讓各個商務主機推導出所屬的金鑰，則商務主機 A、B、C、D 的目錄中必須有 $K_{01}$；商務主機

B 的目錄中必須有 $K_{11}, h(32, K_{21})$；商務主機 C 的目錄中必須有 $h(32, K_{11}), K_{21}\ K_{01}$；商務主機 D 的目錄中必須有 $K_{12}$，也就是行動代理人必須儲存四把金鑰 $K_{01}$、$K_{11}$、$K_{12}$、$K_{21}$ 及兩個雜湊函數值 $h(32, K_{11})$ 和 $h(32, K_{21})$。

此外，除了圖 10 的範例外，我們也就改變節點數的多寡及增加聯結來探討行動代理人所攜帶資料量上的差異。

(1) 當新增節點時，如圖 16 所示。

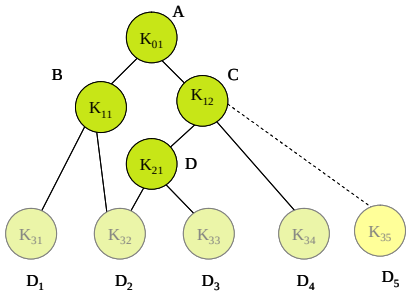



**圖 16　新增節點的範例**

| | 我們的方法 | Lin-Ou-Hwang 的方法一 | Lin-Ou-Hwang 的方法二 |
|---|---|---|---|
| 儲存資料 | $A-K_{01}$<br>$B-K_{11}, h(32, K_{21})$<br>$C-K_{12}$<br>$D-K_{21}, h(32, K_{11})$ | $A-K_{01}$、$B-K_{11}$<br>$C-K_{12}$、$D-K_{21}$<br>$t_1$~$t_4$ 九個公開參數 | $A-K_{01}$、$B-K_{11}$<br>$C-K_{12}$、$D-K_{21}$<br>$e_1 \sim e_5$ 五個參數值 |

(2) 當改變聯結時，如圖 17 所示。

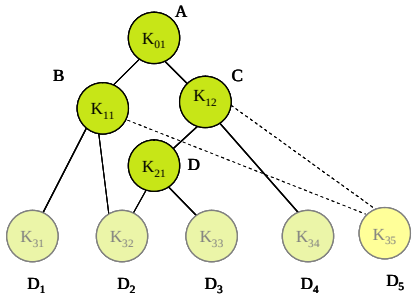



**圖 17　改變聯結的範例**

| | 我們的方法 | Lin-Ou-Hwang 的方法一 | Lin-Ou-Hwang 的方法二 |
|---|---|---|---|
| 儲存資料 | $A-K_{01}$<br>$B-K_{11}, h(32, K_{21})$<br>$, h(35, K_{12})$<br>$C-K_{12}, h(35, K_{11})$<br>$D-K_{21}, h(32, K_{11})$ | $A-K_{01}$、$B-K_{11}$<br>$C-K_{12}$、$D-K_{21}$<br>$t_1$~$t_4$ 九個公開參數 | $A-K_{01}$、$B-K_{11}$<br>$C-K_{12}$、$D-K_{21}$<br>$e_1 \sim e_5$ 五個參數值 |

經過上述的討論與分析，為方便我們與 Lin-Ou-Hwang 的方法做所需攜帶資料量上的比較，茲定義下列符號：

$f_1$：表示節點被 $n$ 個商務主機共同存取。
$f_2$：表示節點只被單一個商務主機存取。
↑：表示某種情況增加時。
如"$f_1$↑"表示當節點被 $n$ 個商務主機共同存取的情況增加時，也就是指改變聯結的關係；"$f_2$↑" 表示節點只被單一個商務主機存取的情況增加時，也就是新增節點。
$t$：代表根據 Lin-Ou-Hwang 的方法一所產生的參數
$e$：代表根據 Lin-Ou-Hwang 的方法二所產生的參數

表 1　本研究與 Lin-Ou-Hwang 方法的資料量比較表

| | 我們的方法 | Lin-Ou-Hwang 的方法一 | Lin-Ou-Hwang 的方法二 |
|---|---|---|---|
| $f_1$↑ | 增加 $2n$ 個的雜湊函數值 | 資料量雖然不會改變，但 $t$ 值都必須全部重新計算 | 資料量雖然不會改變，但 $e$ 值都必須全部重新計算 |
| $f_2$↑ | 資料量不會增加 | 資料量雖然不會改變，但 $t$ 值都必須全部重新計算 | 資料量雖然不會改變，但 $e$ 值都必須全部重新計算 |

統整兩個方法的差異，Lin-Ou-Hwang 的方法若要推導出金鑰時必須使用指數運算方式，且當改變節點或聯結的變化時，更必須每一個節點的公開參數值重新計算，加上公開參數間必須是互質的，且若是考慮到安全性，參數的選擇可能是很大的數值，因此假若當行動代理人所攜帶的權限檔案數量很多時，商務主機目錄內的金鑰所需的儲存量及推導權限檔案的運算量都會影響到代理人儲存空間和商務主機存取權限檔案時所耗費的系統資源。相對來說，雖然我們的方法在節點被 $n$ 個商務主機共同存取的情況增加時會增加 $2n$ 個的單向雜湊函數值，但在我們金鑰的產生方式是用單向的雜湊函數來做運算，且當節點只被單一個商務主機存取的情況增加時，並不會影響其他節點的金鑰值，對於行動代理人和商務主機的負擔都比較不大。

### 4.1.2　效率分析

以圖 10 為例，為方便與本研究與 Lin-Ou-Hwang 的方法做運算效率之分析，茲定義下列符號：

T( )：一次運算所要花費的時間。
h：雜湊函數運算。
r：隨機產生亂數。
D：非對稱加解密運算。
A：對稱加解密運算。
EA：乘法反元素之運算。
E：指數運算。
M：模數運算。
MM：乘法運算。
ME：指數模數運算

表 2　本研究與 Lin-Ou-Hwang 方法的運算效率分析

| | | 我們的方法 | Lin-Ou-Hwang | |
|---|---|---|---|---|
| | | | 方法一 | 方法二 |
| 金鑰產生 | | T(r)+8T(h)+4T(D)+4T(A) | 11T(r)+9T(MM)+8T(ME)+4T(D)+4T(A) | 7T(r)+ T(MM)+4T(EA)+8T(ME)4T(D)+4T(A) |
| 金鑰推導 | 主機 A | T(D)+7T(h) + 4T(A) | T(D)+4T(ME)+4T(A) | T(D)+4T(ME)+4T(A) |
| | 主機 B | T(D)+2T(h)+2T(A) | T(D)+2T(ME)+2T(A) | T(D)+2T(ME)+2T(A) |
| | 主機 C | T(D)+4T(h)+3T(A) | T(D)+3T(ME)+3T(A) | T(D)+3T(ME)+3T(A) |
| | 主機 D | T(D)+2T(h)+2T(A) | T(D)+2T(ME)+2T(A) | T(D)+2T(ME)+2T(A) |
| 總運算成本 | | T(r)+23 T(h)+8T(D)+15T(A) | 11T(r)+9T(MM)+19T(ME)+8T(D)+15T(A) | 7T(r)+T(MM)+4T(EA)+19T(ME)+ 8T(D)+15T(A) |

由表 2 可看出，Lin-Ou-Hwang 的方法大部份都為指數模數的運算，在時間上的花費比單向雜湊函數高出很多，因此可知本研究之金鑰管理與存取控制架構具有相當的效率性。

本研究將計算行動代理人運算的過程分為金鑰產生跟金鑰推導兩部份。根據圖 10 的範例，我們推算代理人主機派遣行動代理人出去時，產生各商務主機目錄內的金鑰，及各物件的加解密金鑰，最後並針對每一個專屬目錄，使用各商務主機的公開金鑰加密，以保護資料的安全的整體運算時間。另一方面，將各個商務主機收到欲執行任務的行動代理人時所執行的運算耗時表示如表 2，代表當商務主機接收到行動代理人後，首先使用自己的私密金鑰解開專屬的目錄，並根據其目錄及其他相關的資訊，如數學公式...等，推導金鑰，最後利用推算出的金鑰物件解開其可使用的權限檔案的執行過程。

### 4.2 安全性分析

如果金鑰沒有被適當的管理，惡意的主機就有辦法取得它們，導致加密的失效，因此針對金鑰的安排，我們沿用 Volker 和 Mehrdad 的架構，同時我們將 Lin-Ou-Hwang 的階層式架構進行改良，進而提出由上到下的階層式金鑰推算概念。透過我們的方法，代理人主機可以將金鑰置於各商務主機專屬的目錄底下，並使用各商務主機的公開金鑰將其目錄加密。因此，如果惡意的商務主機要取得不屬於自己的金鑰，則必須要有其他商務主機的私密金鑰方可解開目錄，基於公開金鑰密碼系統的特性，私密金鑰是必須保持不公開的，故惡意的主機很難竊取其他主機的私密金鑰，也就無法解開其它主機的目錄。另外，我們利用角色之間具有階層性的概念，使用單向的雜湊函數做計算，使得金鑰無法推算回前一個的狀態，讓階層高的角色能繼承角色階層低的所有權限，保護不同權責的角色來達成存取不同類型權現檔案的方法，控管商務主機可執行的範圍。以圖 10 的範例來看，商務主機 B 的目錄中會有 $K_{11}, h(32, K_{21})$，因此可以推算出 $D_1=K_{31}=h(31,K_{11})$ 和 $D_2=K_{32}=h(32,h(32,K_{11}),h(32,K_{21}))$，但是商務主機 B 無法經由 $h(32, K_{21})$ 去推算出主機 D 的金鑰為 $K_{21}$，以藉此達到安全性。

## 5. 結論

在本研究中，我們提出一個利用角色之間的繼承關係，以進行單向的雜湊函數運算完成的金鑰管理和存取控制架構。優點上，本方法除了具備以樹狀表示行動代理人結構，可以輕易地將資料搜尋、擷取、插入、分群外，還加入了角色繼承的觀念，讓層級高的角色能繼承低層級角色的所有權限。從效能分析的資料量和運算效率上，我們可以發現本研究之金鑰管理與存取控制架構具有相當的效率性。而本研究使用單向的雜湊函數，主要是希望藉助其不可逆的特性來防止商務主機間惡意的行為，並簡化過去學者所提出利用指數運算的複雜性和計算時間。因此，本方法不僅可以有效降低行動代理人所攜帶的資料量，以及商務主機運作行動代理人時所耗費的資源，同時也可以增加行動代理人在執行上的效能和安全性。

## 參考文獻


[1] 廖鴻圖、劉世勤,“植基於雜湊函數之電子旅行支票系統,”***電子商務學報***，世新大學資訊管理研究所，7 卷 3 期，2005，pp. 259-274

[2] 楊璿融,“一個行動代理人多元化行程的保護機制,”***碩士論文***，靜宜大學資訊管理研究所，2004 年。

[3] 黃君旭,“有效率地提供行動代理人存取控制與金鑰管理的方法,”***碩士論文***，逢甲大學資訊電機工程碩士在職專班，2006 年。

[4] 林千代,“可攜性 RBAC 資訊系統架構之研究,”***碩士論文***，朝陽科技大學資訊管理研究所，2003 年。

[5] S. G. Akl and P. D. Taylor, “Cryptographic Solution to a. Problem of Access Control in a Hierarchy,” ***ACM.***

***Transactions on Computer Systems***, Vol.1, No.3,1983, pp.239– 248

[6] C. Li, C. Yang and R. Cheung ,"Key management for role hierarchy in distributed systems," ***Journal of Network and Computer Applications***, Vol. 30, No. 3. August 2007, pp. 920-936.

[7] I.C. Lin, H.H. Ou, M.S. Hwang, "Efficient access control and key management schemes for mobile agents," ***Computer Standards and Interfaces***, Vol. 26, No. 5, 2004, pp. 423-433.

[8] R. Volker and J.S. Mehrdad, "Access Control and Key Management for Mobile Agents," ***Computers Graphics***, Vol. 22, No. 4, 1998, pp. 457-461.

[9] Ravi Sandhu, Edward J. Coyne, Hal L. Feinstein and C. E. Youman, "Role-based Access Control Models," ***IEEE Computer***, Vol. 29, No. 2, February 1996, pp. 38-47.