以工作權責爲基礎之存取控制模式之研究
Job-Based Access Control Model

余俊德
元智大學電機暨資訊工程研究所
中壢市遠東路 135 號
yu@www.mmlab.cse.yzu.edu.tw

黃士殷
元智大學電機暨資訊工程研究所
中壢市遠東路 135 號
shyhin@cs.yzu.edu.tw

摘要

Role-Based Access Control 是近年來由美國國家標準局（NIST）提出的較新的資料存取機制，但 Role-Based Access Control 仍有實作上的困難，其中包括分派 Objects 的 permission，本論文的目的是要研究出一個以 Role-Based Access Control 爲基礎，加入一 Job 的觀念於 Role-Based Access Control 架構中，提供一個系統發展者及管理者易於發展與管理的資料存取機制模式。Job-Based Access Control Model 中，Role 不是直接與 Permission 做連結，而是經由 Job 的連結，間接跟多個 Objects 與 Permission 做連結。且 Job-Based Access Control Model 是以整體工作運作模式作分析，將 Objects 與 Permission 依據實際工作之需求加以分類形成多個 Job，以及對於 Job 之間的關係作定義，使其更接近於實際工作狀況，同時也加強 Role-Based Access Control Model 之優點，包括將 Permission 管理之複雜度更爲降低、整體架構與實際組織更爲相似、在分散管理 Permission 方面，更易於分派給適當之 Role 等等。

關鍵詞:Access control, Formal model

1. 簡介

目前已有發展出多種有關資料存取的方法（approaches）與策略（policies），例如：存取控制矩陣（The Access Matrix）、存取控制列（Access Control Lists）、Mandatory Access Control、Discretionary Access control 及 Role-Based Access Control[1,[2]等等，但目前並沒有可以證明任何資料存取機制爲最佳的，而 Role-Based Access Control 是近年來由美國國家標準局（NIST）提出的較新的資料存取機制，目前實作方式大致上有使用物件導向的方法[3]、使用資料庫的方法[4]，但 Role-Based Access Control 實作之相關文獻中，仍較著重於特定之系統，如 Web 等，對於實際組織架構之實作的問題，並未多加討論，所以如何依據實際需求，提供適當的存取控制，是一值得探討的課題。

目前相關文獻中，Role-Based Access Control Model 皆是 Role-Object 做連結，對於 Role-Object 的分派並未有一簡易的模式，所以本論文的目的是要研究出一個以 Role-Based Access Control 爲基礎，加入一 Job-Based 的觀念於 Role-Based Access Control 架構中，提供一個系統發展者及管理者易於發展的資料存取機制模式。Job-Based 若以一公司的觀點來看的話，Job 就相當於工作權責;而就 Role 而言稱爲 Job。Job 將 Permission-object 做一包裝，Role 直接與 Job 做連結，不再是與各個 Object 做連結，希望透過 Job 的中介，改善 Role-Based Access Control Model 的缺點。

2. Job-Based Access Control Model

2.1 Job-Based Access Control Model 定義分析

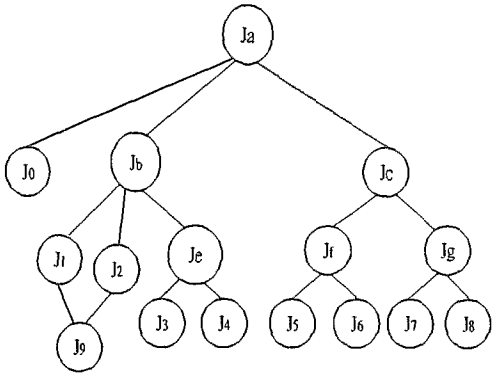
Job-Based Access Control Model 中 Role 不再直接與 Permission 作連結，而是經由 Job 跟多個 Objects 與 Permission 作連結。

我們將原本權限表現方式 a = { < R, O,P > }，(R：Role，O：Object，P：權限)，改爲 a' = { < R, J, C > }, J = { <O, P >}，（J：Job, C: Constraint），將 Object 與 Permission 包成 Job，加入一允許存取的時間。以下就上節描述功能，提出實作的方式：

1. 就 Job 分配方面：依據上節所提之原則，將 Object 做一分類，分別形成多個 Job 的集合，屬同一 Job 的 Object 做一整合，也就是產生多個{<O, P>}的集合。在經由 Job Assignment(JA)將 Job 與 Role 作連結，即可完成所需要求。若 Job 中有 Permission 需要修正或者 Objects 加入時，僅需要對該 Job 的{<O, P>}作修改，則可以達到 Object 的 Permission 管理。使用者若對 Object 的 Permission 作檢視，可以以 Job 爲主，檢視各個 Job 中對各個 Object 的 Permission，而非以往將屬於 Role 的整體 Objects 的 Permission 展開作檢視。經由 Job 分配之後，對於 Objects 與 Permission 的管理與執行，由 Job 整體實際之需求修改與運作，能更符合實際的工作狀況。

2. 就 Job Hierarchy 方面：J={<O, P>}視爲一 Object Record，所以可屬於另一個 Job 中，而此一 Job 可對其提出要求，獲得所需的結果[8]。例如：如圖 1，J b 爲 J a 的下層 Job，則在< J b， P> 屬於 J a 之 Permission 集合中的一個 Object Record，則 J a 可以對 J b 提出要求，J b 經過運算傳回給 J a。

圖 1 Job-Based Access Control 階層圖



3. 就 Job 與 Job Administrator 分派方面：Root Administrator 依據 Security Policy 指定一些 Job Administrators，而 Job Administrator 各自依據其權限產生與管理 Jobs，並且將 Job 給適當的 Role，在一般情形下，Role 其被賦予的 Job，執行其工作。

4. 其他：由於在存取控制中加入 Constraint 的觀點，能有以下兩點功能：

- 在某一些情況下，必須要上層的 Job 或者 Role 存在，下層的 Job 才能運作，也就是在上層的 Job 或者 Role 能夠監視下，下層的 Job 才可執行。
- 另一種情形爲上層 Job 賦予下層 Job 某些權限時，下層 Job 才能運作，但是上層的 Job 並不能執行此權限。

2.1.1 Job-Based Access Control Model 之定義

Job-Based access control 依據各個 security policy 的需求，可彈性的分爲三或四個部分，主要部分有三：

1) Level 1 : Normal Job：用以規範一般 Role 的 Job，定義 Job 內 Objects 的 Permission 與 Constraints，以及與其他 Job 之間的 Permission 和 Constraints。

2) Level 2 : Job of Administrators：規範 Administrator Role 的 Job，其職責爲 Normal Job 的權限之賦予及 Permission 更動，所以定義用以管理與更動 Normal Job Permission 與權限賦予的兩種 Job。

3) Level 3 : Job of Root Administrator (RA)：規範 Root Administrator Role 的 Job，其職責爲 Job of Administrators 的權限之賦予及 Permission 更動，所以定義用以管理與更動 Job of Administrators Permission 與權限賦予的兩種 Job。

較具彈性部分爲 Level 4 : Assignment Job of Root Administrator ：規範 Root Administrator Role 的分派 Root Administrator 的 Job，其職責爲 Job of Root Administrators 的權限之賦予及 Permission 更動與 Root Administrator 的產生與賦予，所以定義用以管理與更動 Job of Root Administrators Permission 與權限賦予及與 Root Administrator 的產生與賦予的三種 Job。

以下爲各個 Level 之定義以及彼此之間的 Relation：[5,6,7,8]

2.1.1.1 Normal Job 之定義

用以規範一般 Role 的 Job，定義 Job 內 Objects 的 Permission 與 Constraints，以及與其他 Job 之間的 Permission 和 Constraints。其定義如下：

Roles: The set of roles
Job: The set of jobs
C: Constraints
Access: the set of access control rules

$A_i^1 \in Access, R_i \in Roles, J_i \in Jobs, C_i \in C$

$A_i^1 = < R_i, J_i', C_i >$

$J_i' = \prod_m (O_{i,m}, P_{i,m}, C_{i,m}) \cup \prod_n (X_{i,n}, P_{i,n}, C_{i,n})$

$P_{i,m} = \{read, write, excute, ...\}$

$P_{i,n} = \{read, write, accredite\}$

$\prod_m (O_{i,m}, P_{i,m}, C_{i,m})$ 原爲 $\{(O_{i,0}, P_{i,0}, C_{i,0}), (O_{i,1}, P_{i,1}, C_{i,1}), ..\}$，爲方便於表示，故將其縮寫以 $\prod_m (O_{i,m}, P_{i,m}, C_{i,m})$ 表示之。

$\prod_m (O_{i,m}, P_{i,m}, C_{i,m})$ 表示之意義爲 Job J_i^I 中可以 Access 的 Objects $O_{i,m}$，以及在限制條件 Constraint $C_{i,m}$ 下對各個 object 的 Permission $P_{i,m}$ 之集合。

$\prod_n (X_{i,n}, P_{i,n}, C_{i,n})$ 亦是爲方便於表示而予以縮寫，其表示之意義爲 Job J_i^I 在 Constraint $C_{i,n}$ 成立時，可透過 $X_{i,n}$ 對於下層 Job J_n^I，依據規定的 Permission $P_{i,n}$ 要求下層 Job 執行後回應之集合。

所以 J_i^I 是 $\prod_m (O_{i,m}, P_{i,m}, C_{i,m})$ 與 $\prod_n (X_{i,n}, P_{i,n}, C_{i,n})$ 聯集所形成的集合，記錄此一 Job J_i^I 可以 Access 的 Objects 與下層 Job 的 Permission。

若將下層 Job J_n^I 視爲上層 Job J_i^I 的一個物件 $X_{i,n}$，上層 J_i^I 可在符合 Constraint $C_{i,n}$ 情況下，經由 Permission $P_{i,n}$ 對下層 J_n^I 提出要求。

上層 Job 與下層 Job 之間的關係有以下幾種情形：

1. 上層 Job J_i^I 對於下層 Job J_n^I 提出要求，同時透過 $X_{i,n}$ 傳遞資料給 J_n^I，J_n^I 在 $P_{i,n}$ 與 $C_{i,n}$ 的規則下，完成 J_i^I 所提之要求，再經由 $X_{i,n}$ 回

應給 J_i^I 。

2. 另一種情形爲上層 Job J_i^I 對於下層 Job J_n^I 僅提出要求，並沒有傳遞資料給 J_n^I ，而 J_n^I 在 $P_{i,n}$ 與 $C_{i,n}$ 的規則下，完成 J_i^I 所提之要求，在經由 $X_{i,n}$ 回應給 J_i^I 。
3. 上層 Job J_i^I 對於下層 Job J_n^I 提出要求，同時透過 $X_{i,n}$ 傳遞資料給 J_n^I ，J_n^I 在 $P_{i,n}$ 與 $C_{i,n}$ 的規則下，完成 J_i^I 所提之要求，並不經由 $X_{i,n}$ 回應給 J_i^I 。
4. 上層 Job J_i^I 對於下層 Job J_n^I 僅提出要求，並沒有傳遞資料給 J_n^I ，而 J_n^I 在 $P_{i,n}$ 與 $C_{i,n}$ 的規則下，完成 J_i^I 所提之要求，並不經由 $X_{i,n}$ 回應給 J_i^I 。

各個非 Administrators 之 Role 可根據 Level 1 之 access control rules $A_i^1 =< R_i, J_i^I, C_i >$ 中的規則，完成其 Job。

2.1.1.2 Job of Administrators 之定義

規範 Administrator Role 的 Job，其職責爲 Normal Job 的權限之賦予及 Permission 更動，所以定義用以管理與更動 Normal Job Permission 與權限賦予的兩種 Job。其定義如下：

$A_i^{II} = (R_i, J_i^{II}, C_i)$

$A_j^{II} = (R_j, J_j^{II}, C_j)$

$J_i^{II} = \prod_m (O(J_m^I), P_{i,m}, C_{i,m})$

$O(J_m^I)$: Object related with the record of the job J_m^I

$P_{i,m}$ ={Read, Write}

$J_j^{II} = \prod_m (O(A_m^I), P_{j,m}, C_{j,m})$

$O(A_m^I)$: Object related with the record of the access control rule A_m^I 。$O(A_m^I)$ 視爲 Level 1 Job 之靜態記錄，在 Level 1 Job 動態執行時，系統必須保證所有存取依照此紀錄執行。

$P_{j,m}$ ={Read, Write}

$O(J_m^I)$ 表示將 J_i^I 視爲 record 包含於物件中，J_i^{II} 透過編輯 $O(J_m^I)$ 對 Level 1 Job J_i^I 做管理，所以 J_i^{II} 視爲可以管理 Level 1 Job J_i^I 的 Level 2 Job，只有擁有其管理權限的 Role 且在 Constraint 符合要求時，就能對 J_i^I 做管理的動作。例如：一行政秘書原本負責整理採購文具報帳，此一 Job 訂爲 J_a^I ，而行政主任對於 J_a 內容有管理的權限，當行政主任認爲 J_a^I 內工作項目需要有所修改時，行政主任則執行管理 J_a^I 的管理 Job J_a^{II} ，如將電腦採購也交由行政秘書執行，則 J_a^I 內工作項目有採購文具報帳與採購電腦，而行政秘書能對 J_a^I 的工作項目作動作，如：產生財務報表等行爲，而行政主任對於 J_a^I 工作項目有管理權限，但他並不去參與其執行過程，僅要行政秘書向其報告工作進度即可。在 Constraint 部分，我們可以設定行政主任只有在每個月初或者每個月只能修改 J_a^I 工作項目，可避免行政秘書工作及行政責任的混亂。

$O(A_m^I)$ 表示將 A_m^I 視爲 record 包含於物件中，J_j^{II} 透過編輯 $O(A_m^I)$ 對 Level 1 之 access control rule 做管理，所以 J_j^{II} 視爲可以管理 Level 1 的 access control rule 的 Level 2 Job，也就是 J_j^{II} 負責將經由 J_i^{II} 管理後的 J_i^I ，指定給適當的 Role 以及 Constraint。

負責執行 Level 2 Job J_i^{II} 與 J_j^{II} 的 administrators 可以爲同 Role，也可以是不同的 Role，如何賦予由 security policy 決定。

2.1.1.3 Job of Root Administrator (RA) 之定義

規範 Root Administrator Role 的 Job，其職責爲 Job of Administrators 的權限之賦予及 Permission 更動，所以定義用以管理與更動 Job of Administrators Permission 與權限賦予的兩種 Job。其定義如下：

$A_{RA} = (RA, J_i^{III}, C_i)$

$A_{RA} = (RA, J_j^{III}, C_j)$

$J_i^{III} = \prod_m (O(J_i^{II}), P_{i,m}, C_{i,m})$

$J_j^{III} = \prod_m (O(A_m^{II}), P_{j,m}, C_{j,m})$

$O(J_i^{II})$: Object related with the record of the administrator job J_i^{II}

$P_{i,m}$ ={Read, Write}

$O(A_m^{II})$: Object related with the record of the access control rule A_m^{II}

$P_{j,m}$ ={Read, Write}

在 Level 3 的 Job 分爲兩種，第一種 Job 爲 J_i^{III} ，$O(J_i^{II})$ 表示將 J_i^{II} 視爲 record 包含於物件中，則 J_i^{III} 負責編輯 $O(J_i^{II})$ ，可對於 Level 2 的 Job J_i^{II} 做管理，也就是 Root Administrator 可以經由此 Job 對 Administrators 的權限及 Constraints 做管理；另一種 Job 爲 J_j^{III} ，$O(A_m^{II})$ 表示將 A_m^{II} 視爲 record 包含於物件中，負責經由 $O(A_m^{II})$ 管理，將 Job of Administrator 分配給適當的 Administrators。

2.1.1.4 Assignment Job of Root Administrator 之定義

規範 Root Administrator Role 的分派 Root Administrator 的 Job，其職責爲 Job of Root Administrators 的權限之賦予及 Permission 更動與 Root Administrator 的產生與賦予，所以定義用以管理與更動 Job of Root Administrators Permission 與權限賦予及與 Root Administrator 的產生與賦予的三種 Job。其定義如下：

$A_{aRA(i)} = (RA_i, J_i^{III}, C_i)$

$A_{aRA(j)} = (RA_j, J_j^{III}, C_j)$

$A_{aRA(k)} = (RA_k, J_k^{III}, C_k)$

$J_i^{IV} = \prod_m (O(J_i^{III}), P_{i,m}, C_{i,m})$

$J_j^{IV} = \prod_m (O(A_{RA(m)}^J), P_{j,m}, C_{j,m})$

$J_k^{IV} = \prod_m (O(A_{aRA(m)}), P_{k,m}, C_{k,m})$

$P_{i,m}$ ={Read, Write}

$P_{j,m}$ ={Read, Write}

$P_{k,m}$ ={Read, Write}

$O(J_i^{III})$: Object related with the record of the child root administrators job J_i^{III}

$O(A_{RA(m)})$: Object related with the record of the access control rule $A_{RA(m)}$

$O(A_{aRA(m)})$: Object related with the record of the access control rule $A_{aRA(m)}$

Assignment Job of Root Administrator 主要有三：

1) $O(J_i^{III})$ 表示將 J_i^{III} 視爲 Record 包含於物件中，則 J_i^{IV} 對於 Child Root Administrators，可透過 $O(J_i^{III})$ 管理，也就是 Parent Root Administrator 可以經由此 Job 對 Child Root Administrator 的權限及 Constraints 做管理；
2) $O(A_{RA(m)})$ 表示將 $A_{RA(m)}$ 視爲 Record 包含於物件中，J_j^{IV} 經由對於 $O(A_{RA(m)})$ 管理，將 Job of Root Administrator，賦予適當的 Child Root Administrator。
3) $O(A_{aRA(m)})$ 表示將 $A_{aRA(m)}$ 視爲 Record 包含於物件中，J_j^{IV} 可經由對於 $O(A_{aRA(m)})$ 管理，賦予 Child Root Administrator 有分派 Child Root Administrator 的權限。

此一 Level 之 Job 是在將 Root Administrator 的權限作一分配，主要在於當整體 Job 架構很大時，避免 Root Administrator 過於集中，而產生效能降低與 Root Administrator 賦予 Administrators 權限不易，也就是 Root Administrator 的權限作適當的分散，形成 Root Administrator tree，各個 Root Administrator 負責其責任範圍 Administrator 的管理即可，如圖 1。採用四層的 Job-Based Access Control，雖然整體系統複雜度會增加，所負擔的 Security 風險卻相對減低；此外，若一個整體 Job 架構不大，如僅十餘人參與的 Job 架構，仍採用四層架構，會因系統的複雜度，而增加不必要的資源浪費。所以是否採用四層架構，應考慮實際狀況以及 Security Policy 之規定，彈性處理之，例如：具有多種性質分公司的集團最爲適合四層架構的 Job-Based Access Control Model，因爲集團總公司可以指定個別分公司的 Root Administrator，各個 Root Administrator 根據所屬分公司之狀況，訂定最佳的存取控制規則。

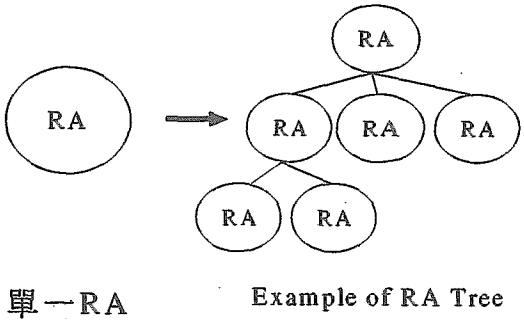


圖 2 Example of Root Administrator Tree

2.1.1.5 Relation of Job Level

如圖 2，首先先對 Job-Based Access Control Model 中主要三 Job Level(Level 1- Level 3)之間的關係作一說明。

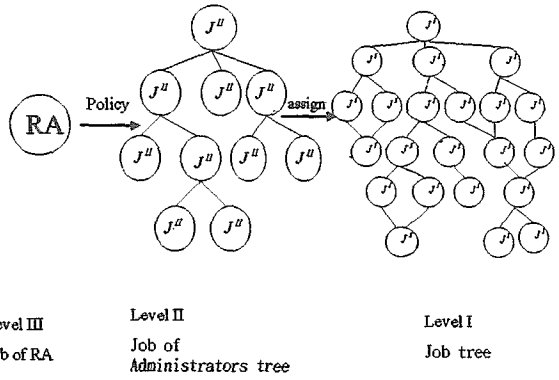


圖 3 Example I : Relation of Job Level

1) Root Administrator（RA） 依據 Security Policy 之規定，透過 J_i^{III} 將各個 administrators 的 Job 做一訂定，再經由 J_j^{III} 將先前訂定 administrator Job 分配給適當的 administrators，進而形成 Job of Administrator Tree；當 Security Policy 修正，或者 Root Administrator 在 Job of Administrator 執行過程，有所需要修改時，Root Administrator 也是經由 J_i^{III} 與 J_j^{III} 的執行，對 Administrator Job 做適當的修正。
2) Administrators 依據實際 Job 之需求，透過 J_i^{II} 將各個 Normal Job 中可以 access 的 objects 與對 objects 的 Permission，以及對下層 Job 的 Permission，再經由 J_j^{II} 將先前訂定 Normal Job 分派適當的 Roles，而 Normal Job 也可形成 Job Tree，Job of Administrators 與 Normal Job of Roles 之間有分派的關係，Administrators 分派 Normal Job 給 Roles ；相同的，Administrators 也是經由 J_i^{III} 與 J_j^{III} 的執行，對 Normal Job

以及分派的 Roles 做修正。

若將 Level 4：Assignment Job of Root Administrator 也加入，如圖 3 所示，Level 1 至 Level 3 之間的關係並沒有很大的變化，但由於 Level 4 的執行，而產生 Root Administrator Tree，與一個到多個的 Job of Administrators tree 以及 Normal Job Tree。

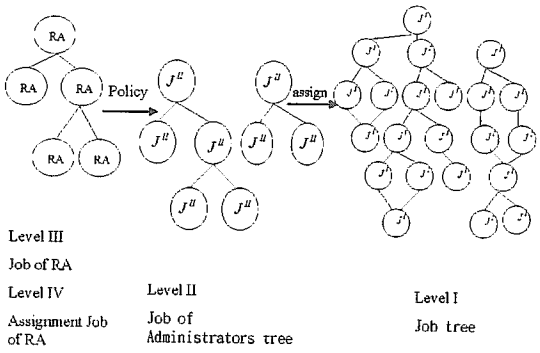


圖 4 Example II : Relation of Job Level

2.1.1.6 Example of Job-Based Access Control Model

Job-Based Access Control Model 可適用的範圍可大可小，如圖 4,圖 5，假設其爲一個公司的 Job Tree，此公司可分爲總公司與工廠，首先應對於 Job 應做一分類，故我們將 Job Tree 分爲 3 大部分，Section A 爲出納與採購部分，由總公司統籌管理，所以其 Job of Administrator 由總公司的 Role 來擔任，如：總務經理；Section B 爲工廠製造，由工廠的負責之 Role 來擔任較爲恰當，如：廠長；而 Section C 爲業務部分，則由業務負責之 Role 來管理，如：業務經理，而三個 Section 的 Administrator 負責的 Administrator 應爲總經理。在上述分析後，Root Administrator 由總經理擔任，經由 Job of Root Administrator 的執行後，將各個 Section 的 Administrators 的 Role 與權限做好設定管理，形成 Administrator tree；而後各個 Administrator 在負責的 Section 依據實際需求，將 Job 予以管理與分派，例如：Section A 中總務經理負責管理 Section A 的 Job 之管理與分派，也同時在星期一至星期六之間可執行 J1 負責所有的採購與出納資料的簽核，J4 負責審定採購資料與採購資料之提報，可由總公司採購主任與工廠的採購主任負責執行，在 Section A 中雖然在 J4 的執行 Role 處於不同單位，但是其負責的事情及權限相同，所以應當統一制訂管理之。

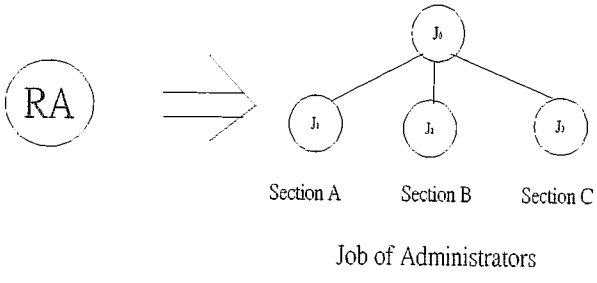


圖 5 Example of Job-Based Access Control Model（I）

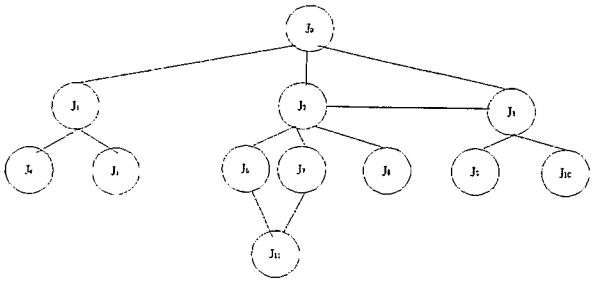


圖 6 Example of Job-Based Access Control Model （II）

在 Section B 中，假設爲工廠之 Job，其中之 Job 不同於 Section C 總公司之 Job，其主要的 Job of Administrator 的執行應當由工廠的負責人負責，如廠長。在 Section B 中，可以發現 J6、J7 與 J11 之間的關係比較特別，假設 J11 是負責模組外殼之製造，J6 是負責製造 PC 外殼，J7 負責製造螢幕，所以雖然 J6 與 J7 皆是要求模組外殼供給，但是所獲得的模組外殼會有所不同。

在各個 Section 之間的關係方面，假設 J0 是總經理之 Job，當 Section C 取得大量業務訂單，在上傳由 J3 的業務經理給 J0，J0 處理確認後下傳給 J2 的廠長，J2 在收到 J0 要求後，依據 J0 要求將工作做分配與執行，最後在工作完成後回應給 J0；另一種情形爲 J2 與 J3 爲平行的 Job，不經由 J0，彼此互爲上下層 Job，可互相提出要求與回應，例如：J3 在 Section C 中，由下層 Job 取得業務訂單，在其職權範圍內，也就是在其 Constraints 規定下，如：必須在總經理 Role 存在，且將業務訂單資料上傳給總經理的情況下，J3 將訂單交給 J2 要求完成，此時 J2 是 J3 的下層 Job。

Job-Based Access Control Model 可以依據實際上 Security Policy 做彈性的調整，如上例其使用於一個公司架構，其實我們也可視爲一個部門的 Job Tree，而每個部門的 Job Tree 又可結合爲更大的 Job Tree，也可個別形成單一部門之 Job Tree 彼此互不影響；也因爲 Job-Based Access Control Model 可大可小，Administrators 可經由 Job of Administrators 的執行對於屬於自己權限內的 Sub -Job Tree 做最佳的管理，以達到較嚴密的 Access Control。

Job-Based Access Control Model 由於能依據實際狀況彈性完成其 Access Control，故擁有以下幾點優點：

1. Permission 分派與修改 Job 容易：
   對於 Objects 與 Job 有所分類後，管理者們只需要對自己負責 Job 中的 Objects 與下層 Job 進行 Permission 管理，不再是集中於上層少數的管理者，且需要面對繁多的 Objects，而對 Permission 分派與管理感到困難。此外，一個 Role 可能因爲實際之需要，需要負責執行多個不同 Job，而管理者對於每個 Job 可各自定義與管理，彼此不受影響。
2. Job 轉移容易：
   Job 可動態被指定給適當的 Role，當 Role 因爲某些因素，對於不再參與執行某些 Job，或者需要加入某些 Job，只需要改變 Role-Job 的連結即可，也可避免需要重新邊修大量的

Object-Permission 而發生錯誤。

3. Role 能擁有適當 Permission：
   Role 執行不同 Job 時，對於相同的 Objects 會被賦予不同的 Permission，所以在每個 Job 中，Role 總能以適當的 Permission 存取 Objects，也達到 Least Privilege 的原則。

3. 結論與未來展望

本論文探討從傳統的資料存取控制到到最新的 Role-Based Access Control Model，進而提出一個以 Role-Based Access Control 爲基礎，加入一 Job 的觀念於 Role-Based Access Control 架構中，以提供一個系統發展者及管理者易於發展與管理的資料存取機制模式。

Job-Based Access Control Model 中，經由 Job 的連結，間接跟多個 Objects 與 Permission 做連結。且 Job-Based Access Control Model 是以整體工作運作模式作分析，將 Objects 與 Permission 依據實際工作之需求加以分類形成多個 Job，以及對於 Job 之間的關係作定義，使其更接近於實際工作狀況，同時也加強 Role-Based Access Control Model 之優點，包括將 Permission 管理之複雜度更爲降低、整體架構與實際組織更爲相似、在分散管理 Permission 方面，更易於分派給適當之 Role 等等。

Job-Based Access Control Model 中，是以 Jobs 一對一對應於 Roles 作討論與分析，尚未將單一的 Job 以及 Job Hierarchy 如何對應 Role Hierarchy 明確的討論與定義，所以在未來 Job-Based Access Control 如何結合 Role-Based Access Control Model 仍有發展的空間。

本 Access Control Model 在未來仍有進一步的發展，首先如何在 Role-Based Access Control Model 爲基礎，將本 Access Control Model 在與 Role-Based Access Control Model 及 Task-Based Authenthorization[7,9]結合不論在資料存取控制及資訊傳遞流程的控制發展更爲完備;此外，可經由實作的方式，對於本 Access Control Model 做更進一步的驗證。

參考文獻

[1]R. S. Sandhu, E. J. Coyne, "Role-Based Access Control Models", IEEE Computer, pp. 38-47 Feb, 1996,

[2]D. Ferraiolo, J. Cugini, and D. R. Kuhn. "Role Based Access Control: Features and Motivations", In Annual Computer Security Applications Conference. IEEE Compter Society Press, 1995.

[3] John Barkley, "Implementing Role Based Access Control using Object Tecnology", First ACM Workshop on Role Based Access Control, November 1995 (http://hissa.ncsl.nist.gov/rbac/rbacot/titlewkshp.html).

[4]John F. Barkley, Anthony V. Cincotta, David F. Ferraiolo, Serban Gavrilla, and D. Richard Kuhn, "Role Based Access Control for the World Wide Web", In 20th National Information System Security Conference. NIST/NSA, 1997.

[5] L. Giuri, "A new model for Role-Based access control", Proceedings of IEEE 11th Annual Computer Security Application Conference, New Orleans, Louisiana, p. 249-255, December 11-15, 1995

[6]Luigi Giuri, Pietro Iglio, Fondazione Ugo Bordoni, "A Formal Model For Role-Based Access Control with Constraints", Proceedings 9yh IEEE Computer Security Foundations Workshop, June 1996.

[7]Roshan Thomas & Ravi Sandhu ,"Conceptual Foundations for a Model of Task-based Authorizations", Proceedings of the 7th IEEE Computer Security Foundations Workshop, Franconia, NH, pages 66-79, June 1994

[8] Yasuda, M.; Tachikawa, T.; Takizawa, M., "A purpose-oriented access control model for object-based system", Object-Oriented Real-time Distributed Computing, 1998.(ISORC 98). Proceedings. 1998 First International Symposium on, pages:146-147

[9]George Coulouris, Jean Dollimore, Marcus Roberts, "Role and task-based access control in the PerDis groupware platform", Proceeding of the third ACM workshop on Role-Based Access Control, p. 115-121, 1998