曹偉駿、鄭博元、李建邦（2013），『基於情境感知之網路服務環境的高安全存取控制機制』，*資訊管理學報*，第二十卷，第二期，頁167-192。

基於情境感知之網路服務環境的高安全存取控制機制

曹偉駿*
大葉大學資訊管理學系

鄭博元
大葉大學資訊管理學系

李建邦
大葉大學資訊管理學系

摘要

以角色為基礎的存取控制（Role-based Access Control; RBAC）方式應用於網路服務中，雖然可使管理者有效率地檢視使用者目前所擁有的權限，然而隨著網路的發展與普及，網路安全問題層出不窮，現有存取控制機制是不足以確保網路服務的安全性。因此，本研究除了以情境感知機制彌補 RBAC 的不足，更進一步使用決策樹演算法探勘隱藏情境。其中，情境感知技術能隨著不同的時空與環境狀態變化，動態地調整用戶存取限制，並依照通訊裝置的特色，提供適當的服務與存取內容，使得授權機制更為彈性，故系統不論在安全性或執行效率上，皆能夠獲得較佳的改善。本機制首先整合單一登入與跨網域 RBAC，改善多系統權限不一與角色衝突的問題，並藉由情境感知技術達到彈性授權之目的，更進一步地採用決策樹演算法，使得情境的推論更加精準，以提升系統在資訊傳輸的安全性及執行效率。

關鍵詞：網路服務、角色為基礎的存取控制、資訊安全、情境感知、決策樹

* 本文通訊作者。電子郵件信箱：wjtsaur@yahoo.com.tw
2011/07/19 投稿；2012/06/14 修訂；2012/10/17 接受

Tsaur, W.J., Cheng, B.Y. and Lee, C.P. (2013), 'A Highly Secure Access Control Scheme for Web Services Based on Context-aware', *Journal of Information Management*, Vol. 20, No. 2, pp. 167-192.

A Highly Secure Access Control Scheme for Web Services Based on Context-aware

Woei-Jiunn Tsaur*
Department of Information Management, Da-Yeh University

Bo-Yuan Cheng
Department of Information Management, Da-Yeh University

Chien-Pang Lee
Department of Information Management, Da-Yeh University

Abstract

With the more and more serious networks security problems, the existing role-based access control (RBAC) mechanisms are insufficient. Therefore, our research will improve RBAC mechanism by adding the functionality of context-aware, and further analyze hidden context data using the decision tree algorithm. The context-aware technique can dynamically adjust users' access constraints with differently temporal, spatial and environmental factors, and at the same time provide adaptable access contents according to distinctive features of equipment (or devices), such that it can extremely enhance security and efficiency in the information systems. Our proposed scheme will integrate the single sign-on and cross-domain RBAC mechanism to solve inconsistent authority and role conflict problems among multi-system in web services, and further achieve the capability of flexible authorization by using context-aware technique. Moreover, in order to improve the system security and efficiency for information transmissions of web services, we further employ the decision tree algorithm to enhance the precision of context inference.

Keywords: Web services, RBAC, Information security, Context-aware, Decision tree

* Corresponding author. Email: wjtsaur@yahoo.com.tw
2011/07/19 received; 2012/06/14 revised; 2012/10/17 accepted

壹、緒論

在資訊科技日新月異的時代，世界各國為了將人類的文明永續經營及保存，無不致力於發展文化的數位化。但各資訊系統卻各自獨立運作，缺乏整合性且資訊不一致，主要原因在於資訊系統來源不一，其來源包含自行研發與許多外包系統，因此需要大量人工作業的配合，才能進行資訊交換。服務導向架構（Service Oriented Architecture; SOA）的應用，促使分布在各地的使用者，可隨時透過資訊服務提供者快速地獲得其所想要的資訊。

使用者驗證（Authentication）與授權（Authorization）一直是網路服務相關應用的重要資訊安全課題。近來隨著資料量日益增加，媒體型式的多樣化，以及網路服務範圍的不斷擴大，加上存取網際網路的方式更加多元，增加系統管理者在管理上的複雜性，同時造成資訊傳輸的安全問題。此外，若連線品質較不理想的行動用戶或發生延遲程度較為顯著的無線通訊使用者，則較易產生網路存取效率不彰的現象。現有解決方式是讓使用者根據自己的裝置平台選擇最適當的傳輸方式，用以告知服務端該如何較有效率地傳送檔案。然而，這些前置手續導致使用者在系統設定上更加繁瑣，徒增其使用困擾而已。情境感知根據使用者連線品質的優劣，藉以提供最適當的存取方式，提升通訊效率，並讓使用者省去人工設定的困擾，增進使用者使用資訊系統的意願。同時，情境的運用更可以彌補純粹 RBAC（Role-based Access Control）機制的安全不足問題（Kapsalis et al. 2006），例如可針對用戶的連線情境，動態且彈性地賦予不同的存取限制（Constraint）（Strembeck & Neumann 2004; Abdallah & Takabi 2010），以提高系統的安全性及執行效率。

本研究加入情境感知用以彌補系統的不完善，並利用網路服務分散式架構之運算優勢，降低系統負載（Park & Hwang 2003），以使得用戶取得權限後，可確保其能享用資源。然而，主從式架構下涉及個體的權限分配問題，例如每個節點（Peer）對每個使用者都將開放不同的存取權限（Palomar et al. 2008），而 RBAC 卻是以角色為最小授權單位，使得權限分配僅顧及控管便利性與效率性，卻欠缺彈性。因此，本研究將藉著情境感知的加入，並利用決策樹演算法探勘隱藏情境，動態地調整用戶存取限制，並依照通訊裝置的特性，提供適當的服務與存取內容，使得授權機制更為彈性，因而再藉由授權的角色來減少資訊傳輸上的安全問題。

本研究由第貳節開始，首先將針對與本研究相關的文獻進行探討。第參節則整合第貳節節所探討的相關技術，藉以建構網路服務環境之情境感知授權機制。第肆節則針對本研究所提出的授權機制進行各種分析與優越性比較，最後於第伍節對本研究進行總結。

貳、文獻探討

本節將先針對網路服務架構進行說明，再針對情境的搜集和分析方法加以描述，最後對決策樹的演算過程進行簡要的介紹。

一、網路服務架構

網路服務架構是將網路服務建立在傳輸規範之上（Younas et al. 2006），使得系統增加了主動式的傳輸概念，當服務來自用戶端的請求，所有的服務個體能夠相互地支援，協同呼叫用戶端所須的應用程式。

網路服務（Lim et al. 2004）是一種以 XML 為基礎，使得應用程式可以提供在網路上橫跨不同平台與不同語言，使其相互溝通的提供服務，以讓其他機器的程式呼叫使用，其相關標準包括了簡單物件存取協定（Simple Object Access Protocol; SOAP）、網路服務描述語言（Web Services Description Language; WSDL），以及通用描述、探索與整合（Universal Description Discovery and Integration; UDDI）。SOAP 訊息以 XML 作為傳輸規範，但基於 XML 訊息之通訊時間較二進位訊息來的高，故使得 Younas 等（2006）提出組合式的網路服務，藉以提升系統服務效率。此外，Han 與 Xia（2009）提出更高的容錯機制，以有效地利用企業資源，進一步提供用戶即時的服務。

在分散式環境下，每個節點（Node）皆具有相同地位之存取權利，若應用上缺乏系統管理員來集中控管，則容易遭受阻絕服務攻擊（Denial of Service; DoS）（Maniatis et al. 2004），因此 Palomar 等（2008）提出了授權機制來改善這個問題。網路服務架構中依照集中程度區分為三種（Stephanos & Diomidis 2004）：(1)完全分散式（Purely Decentralized），網路上每一個節點同時兼扮演用戶端與服務端的角色，但缺乏伺服器集中控管，在安全上也較容易出現隱憂；(2)部份集中式（Partially Centralized），與完全分散式架構較相近，但是不同的地方是少數節點定義了一些規則，具有修復或是避免惡意破壞的功能，用以防止傳輸失敗，我們稱這些節點為超節點（Super Node）；(3)混雜分散式（Hybrid Decentralized）架構中，節點之間佈署一台伺服器，用以集中處理檔案的捷徑，並告知使用者所要求的檔案位置，因而可使檔案搜尋更有效率。

基於上述，部分集中式架構除了具有集中控管的能力，尚能夠保護系統安全。因此，本研究將整合網路服務技術於此部份集中式的架構中，使得服務可以得到更高的穩定性及安全性。

二、結合 RBAC 與情境感知的授權機制

RBAC 在 1992 年由 Ferraiolo 與 Kuhn （1992）所提出，用以有效地控制人員存取權限的一種模式。在 RBAC 模型中，資源的存取權限是被指派到角色，而不是直接授權給使用者，故設定使用者權限時，我們只需將角色分配給使用者，使用者就可經由角色所對應的權限而間接獲得授權。

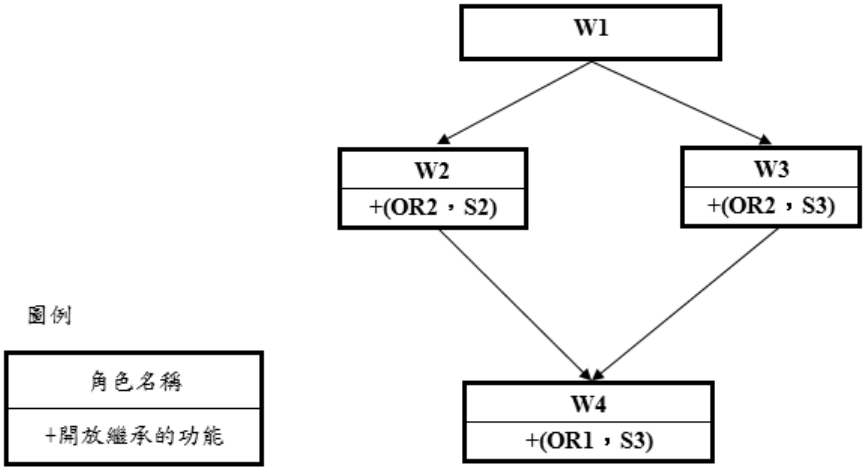

圖 1：角色階層架構

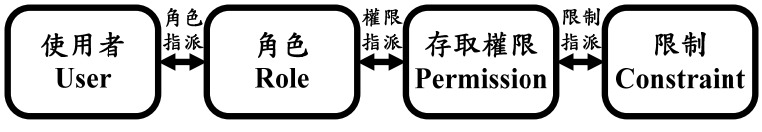

圖 2：結合角色與情境感知的存取控制模型

在角色權限的設計上，為了能與身份驗證做一結合，加速處理上的效能，Chang、Lou 與 Wu（1997）提出基於「將一整數分解成質因數的乘積是唯一」的特性，計算角色所含存取控制範圍。其將每個存取集合中的元素，對應一個質數值，且每個質數值都是唯一且彼此互質。而在存取控制的設計上，只要將每個角色所擁有的存取集合所對應的質數相乘，即可產生每個角色所具有的存取資訊。如此便可透過存取資訊與授權配對所對應乘積相除，若能整除，則代表使用者具有該存取權限。舉例來說，使用者 U_1 具有角色 W_1 的權限、使用者 U_2 具有角色 W_3 的權限、使用者 U_3 具有角色 W_4 的權限、使用者 U_4 具有角色 W_2 的權限。可用資源與存取權限的關係如表 1 所示，並將此關係表中的元素各自配予一個質數值，而每個角色與可用資源的授權關係則如表 2 所示。從表 1 及表 2 可以得知上述每

個角色所擁有的可用資源與存取權限為何。然而以目前企業組織的架構來說，必定存在著一個職位階層的概念，假設角色繼承關係如圖 1 所示，因此便可以計算出每個角色的存取（Role Control; RC）如表 3。當使用者 U_2 想要執行 OR_1 物件時，只要將 U_2 所代表角色 W_3 的存取值 11305 除以代表「執行 OR_1」的質數值 3，若能整除則代表 U_2 具有該功能的存取權限；反之，若餘數不為零，則代表 U_2 不具有該功能的存取權。

表 1：可用資源與存取權限對應關係

可用資源	存取權限		
	執行（S1）	讀取（S2）	寫入（S3）
OR_1	3	5	7
OR_2	11	13	17
OR_3	19	23	29
OR_4	31	37	41

表 2：角色與可用資源授權的對應關係

角色	可用資源授權集合
W_1	（3, 19, 41）
W_2	（13, 23, 31, 1）
W_3	（5, 17, 19）
W_4	（7, 23, 1）

表 3：角色具有的存取資訊

角色	角色存取值（RC）	可用資源與權限
W_1	3×19×41×13×17×7=3615339	OR_1 執行、OR_3 執行、OR_4 寫入、OR_2 讀取、OR_2 寫入、OR_1 寫入
W_2	13×23×31×7=64883	OR_2 讀取、OR_3 讀取、OR_4 執行、OR_1 寫入
W_3	5×17×19×7=11305	OR_1 讀取、OR_2 寫入、OR_3 執行、OR_1 寫入
W_4	7×23=161	OR_1 寫入、OR_3 讀取

情境感知的存取控制和 RBAC 模型不盡相同，但卻是 RBAC 最原始觀念的延伸。具情境感知的 RBAC 模型是由四個元素所組成，除了使用者、角色與存取權限三個元素外，還必須以情境限制（Constraint）來決定最後權限的依據（Strembeck & Neumann 2004），如圖 2 所示。此外，情境限制元素亦能夠加入 RBAC 模型中作一應用，並依照當時條件狀態的改變，適時調整所提供的存取權限。情境感知是用來彌補系統安全性以及效率上的不足，因此本研究首先利用推拉（Pull, Push）模式取得情境資料，進一步再以決策樹演算法加以分析情境所擁有的隱藏特徵。推拉模式常用於 Web-based 應用系統管理模式，主要用於兩個異質點中交換資料。在行銷角度當中，Push 模式較為主動，而 Pull 模式較為被動。Push 模式可以稱為一種指派訂閱與公佈措施，例如：發送 E-mail、訂閱 RSS、廣播通知，而 Pull 模式則是一種呼叫和回應（request and response）的基礎，由用戶端發送服務請求到伺服端。網路搜尋引擎的運用就是屬於 Pull 模式的應用。

此外，因為情境資料須依靠監督式學習來分析隱藏特徵，故本研究採用決策樹的分類方法，其依照特定規則或方法分類，進一步找出影響結果的主要因子（Polat & Gunes 2009）。決策樹最具代表性的種類為 ID3（Iterative Dichotomizer 3）演算法，經過多次改良，先後出現了 C4.5 與目前的 C5.0 版本，自 C4.5 版本推出之後，連續型資料已經可經由離散化程序，再進行分析任務。每個情境元素皆可對應一組存取限制，而用戶最終的存取權限將受到情境組合的影響（Mengelberg 2005）。情境的組合方式可以是由許多單一情境所組成，亦能夠由許多複合情境摻雜單一情境組合產生，而使用完畢的用戶情境，最終將保存在歷史情境資料庫中。

三、決策樹

在資料探勘的領域中，決策樹是利用一種樹狀結構的規則進行分類的探勘方法（Quinlan 1987; Yuan & Shaw 1995）。決策樹在應用上的優點有：(1)決策樹利用樹狀結構規則進行分類的探勘，故可以明顯的了解每一條規則裡面每一個屬性（變數）所代表的意義；(2)在進行探勘時，決策樹可以自動的決定哪些屬性（變數）是重要的，若不重要則會被決策樹捨棄，不會出現在決策樹的樹狀結構規則裡。決策樹演算法種類繁多，底下將針對 C5.0 演算法建立決策樹的過程進行簡要的介紹 （Gu & Guo 2010）。

首先計算決策樹在分類上的資訊含量 I，假設 S 是決策樹 m 個分類結果所組成的集合，p_i 代表 i 類別在整個分類結果的佔有率，則 S 集合的分類資訊量可由公式(1)表示：

$$I(S) = I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{1}$$

公式(2)中，s_{1j} 到 s_{mj} 代表決策樹 m 個分類結果當中擁有 j 特徵所形成的集合。在擁有 j 特徵的前提下，則 S 集合的分類資訊量可表示為：

$$H(s_{1j}, s_{2j}, \ldots, s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}) \tag{2}$$

在測試某個屬性影響分類結果的誤判率方面，$E(A)$代表熵值，其值越小表示該屬性影響分類結果越顯著。假設這個屬性是由 v 個特徵所組成，則公式(3)為針對該屬性給予 S 集合的分類資訊量：

$$E(A) = \sum_{j=1}^{v} \frac{s_j}{s} H(s_{1j}, s_{2j}, \ldots, s_{mj}) \tag{3}$$

資訊量越高代表雜訊越多，因此由公式(1)與(3)相減即能得出資訊量減少的多寡（或稱為資訊獲利），其以公式(4)表示：

$$Gain(A) = I(s_1 + \ldots + s_m) - E(A) \tag{4}$$

最後選擇能得出較大資訊獲利的屬性（或熵值最小的屬性）作為情境特徵判別的依據，亦即該屬性可以當作子樹分裂的節點，進而降低決策樹的複雜度。

參、建構情境感知授權機制

本研究已於第貳節中論述 RBAC 不僅在維護上提供較便利及彈性的管理外，還能對權限給予合理的分配。然而，RBAC 在複雜的資訊環境下，其安全度是為不足。因此，為了更謹慎提供用戶存取許可，將藉由情境感知輔助授權。本研究首先建立之 SOAP，進而整合跨網域 RBAC 與情境感知授權機制（Kapsalis et al. 2006），其中情境資料的取得將採用推拉模式篩選，並將取得的情境資料以決策樹演算法進行探勘，最後再將情境進行組合以得出存取限制。以下將詳細介紹本機制之運作方式。

一、建置簡單物件存取協定

圖 3 為網路服務環境下之存取控制機制架構，其中情境感知為本機制將加入的核心項目，而網路服務是以簡單物件存取協定（Simple Object Access Protocol; SOAP）作為傳輸規範，並以同儕運算技術，用以架構協同運算時所需之服務呼叫機制。在網路服務環境下，使用者可以視為一般主從式架構的用戶端，而服務端即以各個節點與超節點表示。

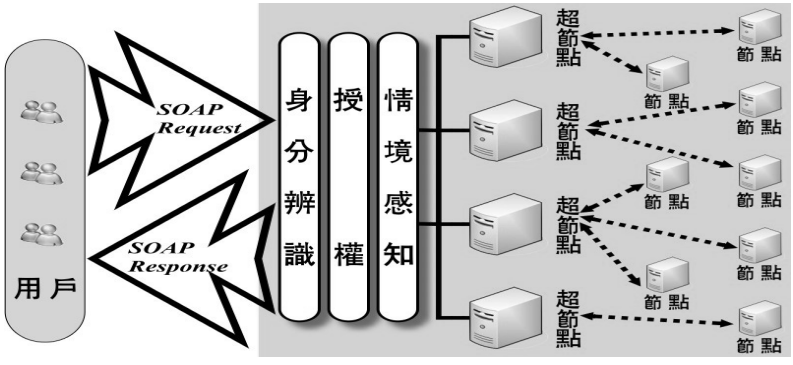

圖 3：網路服務環境之存取控制架構

使用者取得服務必須先通過身分辨識、授權與情境感知機制，再藉由超節點之間的溝通，用戶得以發送連線請求至任一超節點來完成驗證。當用戶取得存取限制的指派，超節點進而向其所負責控管的節點查詢用戶所要求的資源，使得用戶和節點之間的傳輸動作得以成功建立。以下利用演算法說明資源傳輸機制，首先進行各項變數定義：

resourceRn：資源。

superNode[S]：超節點 *S*。

node[S][b]：由超節點 *S* 所控管的第 *b* 節點。

reqRn：用戶要求的資源。

資源傳輸演算法

input: *Resource request: (superNode)*
output: *deliver Resource*
User send request to superNode via SOAP message;
superNode:{
receive request;
process request;
if (request.authorization.equals("allow"))
service initialized;
else
return deny;
endif
for (b=1; b<=n; b++)

```
    if (node[S][b].resource != reqRn || node[S][b].equals("dead"))
        continue;
    else
        send resourceRn = node[S][b].resource;
    endif
  endfor
  }
```

用戶通過身分驗證後，使得服務得以初始化並傳送服務清單，接著便能向系統要求所需服務。超節點負責向節點搜尋用戶所需的資源，若是可得來源越多傳輸效率越佳。然而，並非每個節點都具可得性，若該服務常常被要求，系統則根據情境利用備份機制擴增檔案來源，以提升傳輸效率。關於備份機制之首要步驟是先讓超節點確認備份清單的來源。以下為「以資源清單完成備份動作」之演算法，首先定義該演算法所新增的變數：

listRn：資源清單 Rn。

superNodeRn：成功查詢資源「清單 Rn」的超節點。

nodeRn：擁有資源「檔案 Rn」的節點。

以資源清單完成備份動作之演算法

```
input: resource list
output: backup resource
superNode[0]=superNodeRn;
//搜尋其他超節點是否可查詢資源『清單 Rn』。若搜尋成功，將其標記為
//『superNodeRn』，否則該超節點向標記『superNodeRn』提出資源查詢請求
  foreach (a=1; a<m; a++)
    if (superNode[a]==listRn)
        superNode[a]=superNodeRn;
    else
        SEARCH listRn from superNodeRn;
    endif
  endfor
  foreach (a=0; a<m; a++)
     foreach (b=0; b<n; b++)
//該節點若擁有資源『檔案 Rn』，將其標記為『nodeRn』
```

```
            if (node[a][b]==resourceRn)
                node[a][b]=nodeRn;
//每個超節點 a 所負責控管的節點 b 數量都不同，表示 a 對應不到節點了
      else if (node[a][b]==null)
                break;
//該節點若缺乏資源『檔案 Rn』，則依照剛才查詢結果呼叫『nodeRn』進行備
//份，備份完畢將其標記為『nodeRn』
         else
                BACKUP resourceRn from nodeRn;
                node[a][b]=nodeRn;
            endif
      endfor
  endfor
```

在「以資源清單完成備份動作」之演算法中，「超節點 0」假設為成功查詢資源清單 Rn，因此我們得知欲備份的來源最少有一個，再利用其他能查詢到相同資源清單的超節點，命令其向自己負責的節點要求資源檔案 Rn，最後利用完成備份或是其他可得來源進行剩下的備份工作。

二、整合 RBAC 與情境感知的存取控制

在系統安全的部份，本機制一共分為「單一登入」、「RBAC 授權」與「情境感知」三項運作流程如圖 4 所示，每個運作流程皆涵蓋著相關細節，並列述如表 4：

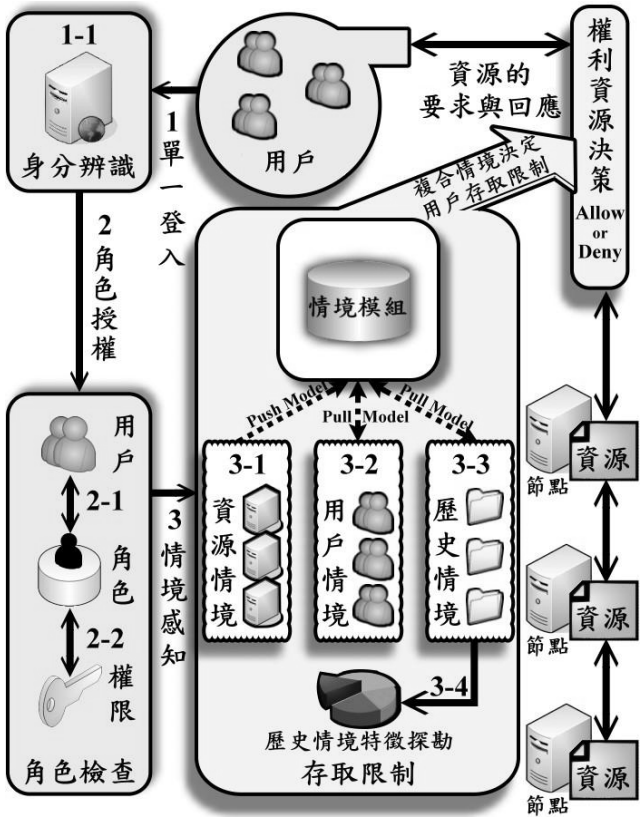

圖 4：情境感知存取控制流程

表 4：情境感知存取控制步驟說明

單一登入（Single Sign-On）階段	
1-1	以帳號密碼驗證使用者身份。
RBAC 跨網域角色授權階段	
2-1	角色對應。
2-2	權限指派，依據不同角色授予不同的存取權限。
情境感知階段	
3-1	以 Push 模式更新系統各項資源的情境。
3-2	以 Pull 模式要求用戶目前情境，同時將用戶情境新增至歷史情境資料庫。
3-3	以 Pull 模式要求相關歷史情境。
3-4	對歷史情境進行資料探勘，包括針對現有的情境資料進行學習，使得授權的過程中提供更大的權限調適空間。

（一）單一登入階段

為了對用戶執行授權，首先勢必提供一套完善的登入機制。此外，應該避免讓用戶執行每個服務時，皆必須重覆地進行身分驗證，本機制將採用單一登入對用戶進行身分辨識（Tsaur & Lin 2009），以防止訊息在頻繁的驗證交談過程中產生不必要的風險。

（二）RBAC 跨網域角色授權階段

在角色權限的設計上，為了能與身份驗證作一結合，進而加速處理效能，本機制利用「將一整數分解成質因數的乘積是唯一」的特性，計算角色的存取範圍，進而做到網路服務下跨網域整合的目標。換言之，一個角色所具有的存取權限，是將根據該角色值所包含的質因數而定，只要將每個角色所擁有的存取集合所對應的質數相乘，即可產生每個角色所具有的存取資訊。如此便可透過存取資訊與授權配對所對應乘積相除，若能整除，則代表使用者具有該存取權限（Chang et al. 1997）。

假設 DA 為可用服務（Object）所形成的集合，表示為：$DA=\{DA_1, DA_2,\ldots,DA_n\}$；而 OP 為所有存取方式（Operation）所形成的集合，表示為 $OP=\{OP_1, OP_2,\ldots,OP_n\}$；並且利用 DA 和 OP 可形成一項存取權限（Permission），針對此權限給定一個權限值 P，則對於本機制提出以下定義如式(5)所示：

$$\begin{aligned} &\forall DA \in object,\ \forall OP \in operation \\ &\exists (object, operation) = P_n \subseteq Permission \\ &\Rightarrow lcm[P_1, P_2,\ldots,P_n] = \prod_{i=1}^{n} P_i \ ,\ gcd(P_1, P_2,\ldots,P_n) = 1 \end{aligned} \tag{5}$$

根據式(5)，管理人員可以將不同的權限分配給不同角色（Role），並且將每個角色賦予一個角色值（RoleID），則對於用戶（User）個體所擁有的存取權限描述如式(6)所示：

$$\exists User \in RoleID_m, P_n \mid RoleID_m \qquad \Rightarrow User\ can\ access\ to\ P_n \tag{6}$$

RBAC 是由兩個運作細節所構成，包含角色指派以及針對角色賦予權限，各個角色所擁有的存取權限可利用質因數分解演算法來表示，首先對各項新增的變數進行定義：

role_id：角色值。

auth_id：權限值。

以角色值取得存取權限之演算法

```
input: role_id
output: authorization()
foreach (auth_id=2; auth_id<=role_id; auth_id++)
      times[auth_id]=0;
      while (role_id%auth_id==0) {
                role_id/=auth_id;
                times[auth_id]++;
                }
       if (times[auth_id]!=0)
            out.print(auth_id+" ");
       else
            continue;
       endif
endfor
```

本機制將相同權限分配特定的角色值，藉由質因數分解演算法可以解決多系統之角色授權的問題，並且改善傳統 RBAC 之角色衝突的發生可能。然而，RBAC 機制在授權方面尚不夠完善，本機制將利用情境感知機制加以補強，藉著週遭環境狀態的改變，動態調整用戶的存取權限。

（三）RBAC 情境感知階段

情境資料由情境模組取得，再根據資料探勘技術加以分析。本機制利用推拉模式來取得情境，如圖 4 所示。其中，用戶情境和歷史情境採用 Pull 模式來獲得，因為資訊的取得是經過篩選動作，除了防止非必要的廣告式訊息產生，並且利用 Pull 來取得用戶情境可以避免過度隱私的資訊傳入系統當中。另外，資源情境則利用 Push 模式來進行內部即時更新。情境取得模組包含檢測環境的應用程式或是資訊感測裝置。本機制採用應用程式介面（Application Programming Interface; API）來取得這些情境變數，並將情境組合比對。

1. 資源端情境更新

   資源端情境描述其目前服務狀況，基於產生方式是屬於系統內部的處理程序，因而利用 Push 模式來省下一道要求與回應的呼叫程序。其中，資源端情境亦包括各項服務被使用的情形，或是服務被請求的次數，進而發現服務趨勢。

2. 用戶情境要求與回應

   我們蒐集用戶的情境包含網路位址、連線延遲狀況、作業系統、視訊解析度與時間狀態等。不同裝置作業系統、視訊解析度都不盡相同，例如行動設備解析度與頻寬較小，不應該讓其接受過於龐大的服務，同時替服務端

省下系統資源。

連線的延遲可以幫助系統判斷網路是否擁塞，或是用戶與服務端連線之順暢與否，連線不良往往是 ISP 的問題，若能提供給頻寬較小的用戶適當瀏覽環境，其使用價值將再次提升。以下使用 Java 演算法來進行說明情境的取得方式。

取得用戶情境之演算法

```
input: Context request ()
output: user context(os, resolution, ip_address, delay, current_time)
//作業系統擷取模組
String os_name=request.getHeader("User-Agent").toLowerCase();
            if (os_name.indexOf("win")>0)    os="Windows";
       else if (os_name.indexOf("nux")>0)    os="Linux";
       else if (os_name.indexOf("x11")>0)    os="Unix";
       else                                  os="others";
       endif
//視訊解析度擷取模組
import    java.awt.Toolkit
var toolkit = java.awt.Toolkit.getDefaultToolkit();
var screen_size = toolkit.getScreenSize();
var screen_width = screen_size.width;
var screen_height = screen_size.height;
var resolution = toolkit.getScreenResolution();
//網路位址擷取模組
String ip_address=request.getRemoteAddr();
//延遲檢測擷取模組
//(ICMP 檢測：Windows 系統檢測用，因 Windows 不支援 TCP/IP 檢測)
String p=Runtime.getRuntime().exec("ping –n 1 ip_address");
int ping_start=p.indexOf("me=", 0);
int ping_end=p.indexOf("ms", 0);
String Str_ping=p.Substring(ping_start+1 ,ping_end);
int ping=Integer.parseInt(Str_ping);
   if (ping<80)            delay1="no";
   else                    delay1="yes";
   endif
//延遲檢測擷取模組
//(TCP/IP 檢測：非 Windows 系統檢測用，判斷作業系統用 if else 條件式)
InetAddress ia=InetAddress.getByName("ip_address");
boolean delay_tcpip=ia.isReachable(80000);
   if (delay_tcpip==True)     delay2="no";
   else                       delay2="yes";
```

```
  endif
  if (os==Windows)
        delay=delay1;        //視窗 OS 變數來自 ping
    else
        delay=delay2;        //其他 OS 變數來自 delay_tcpip
  endif
//時間狀態擷取模組
long current_time = System.currentTimeMillis()
```

3. 歷史情境要求與回應

歷史情境經過系統漫長的搜集過程，所有使用完畢的用戶情境最後皆保存在資料庫中，需要時以 Pull 模式取出，系統每次皆要求相關的情境記錄。歷史情境除了可用來探勘無法判別的隱藏情境，尚可得知用戶個體對系統有無造成威脅的可能。歷史情境還可用來分析無法驗證的情境，以下將介紹情境探勘方法。

三、情境探勘

情境探勘的首要步驟是資料前處理，以下根據管理者的需求決定資料轉換後的型態，例如時間資料係根據 24 小時區分早上（00：01～12：00）、下午（12：01～17：00）與晚上（17：01～24：00）三個時段，表 5 為尚未經過處理的 8 位使用者情境。

表 5：未處理的歷史情境資料

用戶 ID	網路位址	延遲	作業系統	解析度	時間
001	163.23.10.87	NO	Windows	1920x1080	2009 年 04 月 21 日 14:10
002	192.168.3.10	N/A	others	240x320	2009 年 04 月 22 日 22:50
003	220.115.17.40	NO	Windows	1280x720	2009 年 04 月 24 日 00:31
004	140.14.9.47	YES	Windows	1024x768	2009 年 04 月 24 日 20:40
005	210.175.43.100	NO	Linux	1024x768	2009 年 04 月 25 日 16:12
006	140.24.6.212	YES	Windows	1600x1200	2009 年 04 月 25 日 19:31
007	220.134.39.41	NO	Windows	1440x900	2009 年 04 月 28 日 17:44
008	125.233.171.15	YES	Linux	800x600	2009 年 05 月 01 日 21:22

根據表 5，假設欲找出網路延遲的主要原因，當下則毫無規則可循。因此，本研究針對表 5 進行前處理，處理方式可以由管理人員自訂，方法說明如下：

網路位址：將 163 與 140 開頭的 IP 設為學網網域，IP 位址開頭為 192.168 則設為驗證錯誤，其餘 IP 位址則定義為其他網域。

延遲與否：此情境已經簡化，不需處理。

作業系統：此情境已經簡化，不需處理。

視訊大小：以順序尺度分為 high、normal、low 三種類型。

記錄時間：分為早上(00:01~12:00)、下午(12:01~17:00)與晚上(17:01~24:00)。

除了時間可直接利用資料庫語法來處理，其他如網路位址與視訊大小係根據以上條件或是管理需求將情境資料進行「前處理」，其演算法如下，並將處理完成的情境資料顯示於表 6 中。

歷史情境資料的前處理演算法

```
input: context data
output: transformed context data
//情境前處理：IP 資料轉換
    if (ip_address.indexOf("163")==0 || ip_address.indexOf("140")==0)   ISP="學術網域";
else if (ip_address.indexOf("192.168")==0)                              ISP="驗證錯誤";
    else                                                                ISP="其他網域";
    endif
//情境前處理：視訊解析度
x/=10;   y/=10;   pixel=x*y;
    if (pixel>=0 && pixel<=4800)          screen="low";
    else if (pixel>4800 && pixel<12690)   screen="normal";
    else                                  screen="high";
    endif
```

表 6：處理完畢的歷史情境資料

用戶 ID	網路位址	延遲	作業系統	解析度	時間
001	學術網域	NO	Windows	high	2009 年 04 月 21 日　下午
002	驗證錯誤	N/A	others	low	2009 年 04 月 22 日　晚上
003	其他網域	NO	Windows	normal	2009 年 04 月 24 日　早上
004	學術網域	YES	Windows	normal	2009 年 04 月 24 日　晚上
005	其他網域	NO	Linux	normal	2009 年 04 月 25 日　下午
006	學術網域	YES	Windows	high	2009 年 04 月 25 日　晚上
007	其他網域	NO	Windows	normal	2009 年 04 月 28 日　下午
008	其他網域	YES	Linux	low	2009 年 05 月 01 日　晚上

另外，我們可以發現用戶 002 產生情境取得不完全的現象，這是由於各個裝置平台所包含的關鍵特徵都不盡相同，因而導致極少數的情境在判斷上出現錯誤，這時僅須以其他情境來判斷，並且利用決策樹分析法進行交叉比對即可。

承表 5 所衍生的問題：「找出網路延遲的主要原因」，此時利用完成前處理的資料來簡化問題的複雜度，則「網路延遲屬性」稱為問題的「分類結果」，則其他屬性為「影響因子」，欲找出影響因子則利用決策樹演算法中的熵值（Entropy）探討影響結果最甚的屬性。

根據公式(1)至(3)，利用決策樹 C5.0 分析程式得出網路位址、作業系統、解析度與時間四個屬性的 Entropy 分別為 0.752，0.857，0.908 與 0.408。由於若某屬性的 Entropy 最小，則代表該屬性為影響分類結果最重要的因素，故由上述例子可知，時間屬性為影響分類結果最甚之因素。由時間對網路延遲所繪製的簡化決策樹如圖 5 所示，在圖 5 中，若用戶標號被標記底線，則代表網路延遲情境被分類為「延遲」，即 class[0]；若用戶標號被以矩形標記，則代表網路延遲情境被分類為「無法判斷」，即 class[2] ；反之，則代表網路延遲情境被分類為「順暢」，即 class[1]。由於本研究利用決策樹進行探勘，為了增加探勘結果的準確性，應定期更新探勘結果，但由於用戶使用特性在短期內不常更改，因此本研究建議將更新決策樹模式的時間定為每隔一個月或一季。

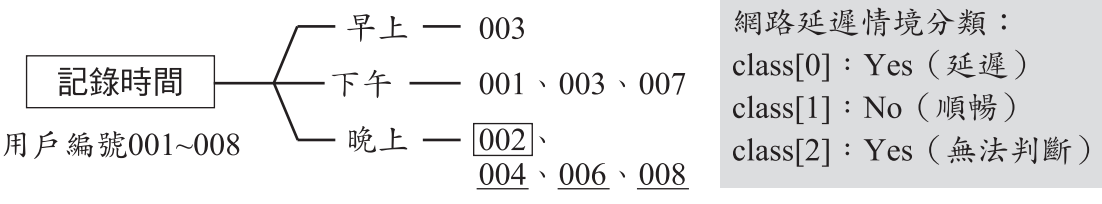

圖 5、用戶網路狀況與時間分配之決策樹

圖 5 顯示網路延遲情境的產生之主要原因為晚上時段，因此管理員可以檢查這個時段系統是否也有回應過長的現象產生，若有即為服務超載，則管理者可以在晚上時段關閉解析度較大的視訊資源，或是另外提供壓縮過的多媒體資源開放在晚上時段取代佔用較大頻寬的服務。總括上述，並非所有情境都需要進行探勘，本機制加入決策樹演算法使得情境取得更精準，讓授權更加謹慎。通過 RBAC 與情境感知機制也獲得了相關的情境，因此以下將介紹情境的組合與權限關係。

四、情境組合與用戶之存取限制

剛加入會員的用戶是沒有任何情境記錄的，因此僅利用 RBAC 與目前情境來

授權。隨著使用次數的增加，用戶的情境記錄將會逐漸保留在系統當中，此時存取權限就受到各種情境記錄的組合，進而影響存取範圍。假設角色值與角色名稱對應如表 7 所示，系統所提供的服務與存取的權限如表 8 所示，而情境與存取限制對應關係則如表 9 所示。以下將由表 7 至表 9 資料的顯示，舉例說明授權結果。

表 7：角色值與角色分配

表角色名稱	角色值	角色值可分解之質因數
VIP 會員	105	3, 5, 7
管理員	1896289395	3, 5, 7, 11, 13, 17, 19, 23

表 8：服務與權限需求

資源編號	資源格式	解析度	使用權限值	管理權限值
R01	JPG	1024x768	3	11
R02	WMV	320x240	3	13
R03	WMF	ALL	5	17
R04	CD/ISO	ALL	7	17
R05	EXE	800x600	5	19
R06	AVI	720P	7	23
R07	MPEG2 （單位超過 200MB）	640x480	3	13

表 9：情境與存取限制關係

情境編號	情境內容	情境限制
C01	Time(H)<6	管理權限全數封鎖
C02	Resolution: W<1280 ∩ H<720	720P 影音資源無法使用
C03	Resolution:W ≥ 320 ∩ H ≥ 240 ∩ 手持設備	影音格式提供 320*240 以下使用
C04	IP ∉ 學術網路	R07 無法使用
C05	OS ≠ Windows	EXE 執行檔無法使用
C06	Device: ~PC ∩ ~NB	光碟映像檔 ISO 無法下載
C07	延遲時段	單位超過 200MB 之資源無法使用

假設今有兩位使用者，其角色資料與情境如下：

1. 角色為 VIP 會員，網路位址為學術網域，無延遲狀況，OS=MAC，解析度

為 1024x768，時間為下午 6 時。

2. 角色為管理員，網路位址屬其他網域，無延遲狀況，OS=WinXP，解析度為 1600x1200，時間為凌晨 3 時。

根據上述，前者為 VIP 會員，原本應可使用所有的服務，但我們將依據其情境開放其可使用的資源，作法是使用刪除法來刪除無法執行的服務，由於 OS=MAC 故 R05 被刪除，解析度為 1024x768 故 R06 被刪除，時間為下午 6 時被視為延遲時段故 R07 被刪除，所以可使用的資源為 R01 至 R04；同理可證，後者為管理員，故可使用的部份為 R01 至 R06，而 R07 因受到區網使用限制故無法進行存取。此外，管理權限則受到時段限制，如此的考量是希望非上班時段取消管理員的權限，降低身分偽冒的事件發生機率，進而提升機制的安全性。

經由本節所介紹的研究架構，本機制依照人、事、地、物、時間、空間等狀態的改變，動態調整用戶存取權限，使得系統安全與服務效率皆可獲得極大的提升，並且搭配網路服務跨平台以及協同運算的特性，使得服務更加穩定，以下第肆節則針對本研究進行各項分析與比較。

肆、機制綜合分析與比較

為了能安全又有效率地對用戶授權，本節將針對所提之存取控制機制進行正規驗證分析（Formal Verification）（Ferraiolo et al. 1999）、安全性分析與效益探討，同時與現有機制進行優越性比較。

一、正規驗證分析

所謂正規驗證法（Formal Verification）係採用數學邏輯進行推論，並以集合之加法與乘法觀念來說明之，目的是算出所有元素的可能排列組合。在存取控制領域中，其更能夠找出授權階段的各個元素對應關係。同時，本驗證法亦可被用來證明存取控制的機制嚴謹度與彈性（Ferraiolo et al. 1999）。

圖 6 為本機制的存取控制模型，其說明使用者、系統角色、存取權限與存取限制之間的指派關係。在圖 6 中，其各項符號定義如下：

$Supernode_i \subseteq RBAC$，其中 $i \in N$，使用者授權是在超節點進行。

RBAC = {User, Role, Permission, Constraint, Session, Role Hierarchy}。

User：服務的用戶個體。

Role：服務之角色設定。

Permission = OB ∩ OP：對於任一項服務（Object）所能行使的操作權責（Operation）。

RC：資源端情境（Resource Context）。

UDC：用戶裝置情境（User Device Context），如作業系統、解析度等。

UNC：用戶通訊環境（User Network Context），如網路位址、網路延遲狀況等。

CF，$\exists CF \subseteq (RC, UDC, UNC)$：特徵情境（Context Feature），以決策樹探勘法從各種情境進行探勘。

Constraint→power set of (RC∩UDC∩UNC)∪CF：情境限制受到上述情境的組合所影響。

Session：用戶以一個角色行使權責時，所形成的時效連結。

Role Hierarchy：角色階層，上層角色所擁有的權責能夠繼承下層角色的權責。

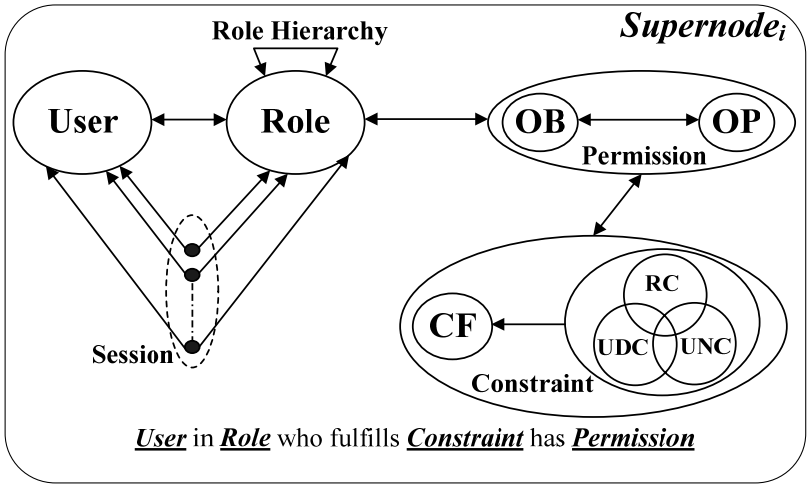

圖 6：RBAC 模型

利用圖 6 得出角色對應（Role Mapping; RM）、權限指派（Permission Assignment; PA）與限制指派關係（Constraint Assignment; CA）：

$RM \rightarrow 2^{User}$：角色對應到用戶個體的所有組合。

$PA \rightarrow 2^{Permission}$：將權責指派給角色之所有可能情形。

$CA \rightarrow 2^{Constraint}$：以限制模式對權責造成的影響之排列組合。

經由以上數學邏輯加以推導並計算唯一的權限組合如式(7)所示，同時和現有 RBAC 進行比較，包含傳統 RBAC 如式(8)所示，以及將本機制利用 Kapsalis 等（2006）學者所提出的方法來描述，如式(9)所示。

$\forall$ *User*, *OB*, *OP* 所具有的存取權限表示為 *Access*（*User*, *OB*, *OP*）$\Rightarrow$

$\exists$ *Role*, *Permission*, *Constraint* 所決定的存取限制：式(7)至(9)

$$[RM] \cap [PA \mid OB \cap OP] \cap [CA \mid (RC \cap UDC \cap UNC) \cup CF] \quad (7)$$

$$[RM] \cap [PA | OB \cap OP] \quad (8)$$

$$[RM] \cap [PA | OB \cap OP] \cap [CA | (RC \cap UDC \cap UNC)] \quad (9)$$

在式(9)中，使用者的存取權限雖然經過多重考量，但是並未從歷史情境考慮到會同步發生的因子，使得授權的動機依據不夠強烈，因此彈性較差，在安全上也較容易產生漏洞。此外，在式(8)中，傳統的 RBAC 僅對角色來授權，在安全方面僅依靠資料庫規則進行比對。因此，光靠 RBAC 是完全不夠的。

在式(7)中，本機制除了針對各個環境變化動態調整存取權限，更進一步地使用了決策樹演算法來探勘隱藏情境特徵，使得情境在辨別上更加精準，比起現有的 RBAC 機制更加安全可靠，同時在系統控管與授權方面也有極佳的彈性。

二、安全性分析

（一）隱私權

一般研究大多採用 DRM 來授權，其強制植入用戶端程式，並且蒐集用戶端裝置 ID、硬體、使用者名稱，以及使用記錄等，並於伺服端進行行為分析，破壞了用戶隱私。另外一種則結合 GPS 或 RFID，採用位置感知的方式來擷取使用者位置資訊，雖然作法上並無惡意，倘若伺服端遭受入侵，除了破壞隱私外，用戶自身安全亦出現問題。

本機制採用 Pull 模式來蒐集用戶情境，Pull 模式具有篩選資訊的能力，告知系統何者資訊是需要的，何者為非必要，故本機制僅需以少量的情境即可分配適當地權限，因而降低對隱私的危害。

（二）使用權利的控管

基於使用者付費的原則必須保障合法用戶使用資源的權利，同時禁止非經授權的使用者企圖存取服務，本研究是依據用戶的使用情形動態調整其存取權限。基於分散式架構在維護上的困難，因此所產生的風險將藉著情境感知來加以補強，節省管理上的手續。本研究亦搭配資料探勘技術，可從簡單的情境當中分配最適當的權限，比起盲目地搜集使用者資訊能夠讓用戶更安心地享受服務。

三、效益分析

以下將本機制所帶來的效益作一說明：

（一）穩定性

網路服務環境中，多媒體資料量位居系統之冠，使得網路負擔日益劇增，本機制利用網路服務分散式架構，其使用 port 80 埠號的優勢，在跨平台服務整合方

面較不容易受到設備內建防火牆的干擾。

（二）提供適當裝置存取服務

本機制根據通訊設備的特性差異來提供適當的服務，例如當行動設備運算量不及 PC 或其顯示面積的不足時，系統則不應該浪費頻寬以提供高畫質的視訊資源給此類用戶。此外，我們亦根據裝置所使用的作業系統，提供不同的存取限制，如 MAC OS 之用戶則不具備.EXE 資源格式的使用權利。

四、優越性比較

本研究與現有存取控制機制進行比較，如表 10 所示。在表 10 中，Kapsalis 等（2006）所提出的存取機制是在情境限制的情況下，以 RBAC 為基礎的方法，其中情境將以動態的方法進行更新，針對各個環境變化動態調整存取權限，但是因為授權彈性較差，在安全上較容易產生漏洞。表 10 中 Tomur 與 Ertrn （2006）所提出的機制是將時間與空間的概念加入 RBAC 裡，降低企業在佈署無線網路時的風險，雖然此方法解決了角色衝突的問題，但並無考慮情境的狀況，因此在使用上較無彈性。而 Mundt（2006）是以全球衛星定位系統（Global Positioning System; GPS）與存取控制來輔助授權，然而 GPS 容易產生延遲誤判（Sheu et al. 2008; Al & McNair 2008），例如公司內各部門的管理員於行政大樓皆具備其專屬的控管權限，而地區的劃分過於細緻將對權限的分配造成極大的困難。此外，在表 10 中我們可清楚看到絕大多數研究僅利用網路服務跨平台的優勢進行分散式呼叫與存取，但其卻無法解決多網域間的權限衝突問題，如此將增加系統整合的複雜性。反觀本研究改進了 Kapsaliset 等（2006）所提出的授權機制，並輔以決策樹演算法以提升安全性。

表 10：本機制與現有機制功能之比較

比較項目	Kapsalis et al.	Tomur and Erten	Mundt	本機制
授權策略	RBAC 情境感知	RBAC 時間與空間	空間	RBAC 情境感知
跨平台存取	有	有	無	有
角色衝突解決	無	有	無	有
情境參考	無	無	無	有，決策樹演算法輔助

伍、結論

藉由網路服務跨平台的優勢以及有效的授權機制設計，使得多系統環境下，可徹底根除 RBAC 角色衝突的問題，同時可以對使用者進行用以達到跨網域授權的目的。此外，本研究改良現有的情境授權機制，以決策樹演算法探勘情境當中的隱藏特徵與規則，規劃出較為嚴謹且彈性的限制模型。為了確實掌控隨時不斷改變狀態的系統用戶，在授權策略上運用情境感知來取得裝置的時空資訊，適時修正使用權利。然而，在極少數情境的蒐集上，無法被系統正確取得與驗證，本研究更進一步地提出「以決策樹演算法來探勘隱藏特徵」，使得情境知識獲取上更加多元。因此，情境感知不需要依賴其他軟硬體設備來完成，依舊能夠精準地取得情境，以使得管理者能夠省下控管會員之繁瑣程序。同時，本研究可運用範圍廣泛，舉凡在權限需要依照不同環境做臨時性的調整與校正時，本機制亦能夠彌補改善之，並且非常適合搭配現有 Youtube 影音社群平台等，推出更豐富的加值服務。

誌謝

本研究接受國科會研究計畫案 NSC 101-2219-E-212-001 、 NSC 101-2221-E-212-006-MY3 經費補助，特此致謝。

參考文獻

Abdallah, A.E. and Takabi, H. (2010), 'Formalizing delegation and integrating it into role-based access control models', *Journal of Information Assurance and Security*, Vol. 5, No. 1, pp. 21-30.

Al, A.D. and McNair, J. (2008), 'On the interaction between localization and location verification for wireless sensor networks', *Computer Networks*, Vol. 52, No. 14, pp. 2713-2727.

Chang, C.C., Lou, D.C. and Wu, T.C. (1997), 'A binary access control method using prime factorization', *Information Sciences*, Vol. 96, No. 1-2, pp. 15-26.

Ferraiolo, D. and Kuhn, R. (1992), 'Role-based access controls', *Proceedings of the Fifteenth National Computer Security Conference*, Baltimore, pp. 554- 563.

Ferraiolo, D., Barkley, F. and Kuhn, R. (1999), 'A role based access control model and reference implementation within a corporate intranet', *ACM Transactions on Information and System Security*, Vol. 2, No. 1, pp. 34-64.

Gu, Y. and Guo, W. (2010), 'Decision tree method in financial analysis of listed logistics companies', *Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation*, Changsha, China, May 11-12, pp. 1101-1106.

Han, S.C. and Xia, Y. (2009), 'Optimal node-selection algorithm for parallel download in overlay content-distribution networks', *Computer Networks*, Vol. 53, No. 9, pp. 1480-1496.

Kapsalis, V., Hadellis, L., Karelis, D. and Koubias, S. (2006), 'A dynamic context-aware access control architecture for e-services', *Computers & Security*, Vol. 25, No. 7, pp. 507-521.

Lim, B.B.L., Sun, Y. and Vila, J. (2004), 'Incorporating WS-Security into a web services-based portal', *Information Management & Computer Security*, Vol. 12, No. 3, pp. 206-217.

Maniatis, P., Giuli, T., Roussopoulos, M., Rosenthal, D.S.H. and Baker, M. (2004), 'Impeding attrition attacks in P2P systems', *Proceedings of the 11th ACM SIGOPS European Workshop*, Leuven, Belgium, September 19-22, pp. 1-5.

Mengelberg, J.B. (2005), 'Teaching system access control', *Journal of Issues in Informing Science and Information Technology*, Vol. 2, pp. 139-158.

Mundt, T. (2006), 'Two methods of authenticated positioning', Proceedings of the Second ACM International Workshop on Quality of Service & Security for Wireless and Mobile Networks, New York, USA, October 2, pp. 25-32.

Palomar, E., Tapiador, J.M.E., Hernandez-Castro, J.C. and Ribagorda, A. (2008), 'Secure content access and replication in pure P2P networks', *Computer Communications*, Vol. 31, No. 2, pp. 266-279.

Park, J.S. and Hwang, J. (2003), 'Role-based access control for collaborative enterprise in peer-to-peer computing environments', *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, New York, USA, June 2-3, pp. 93-99.

Polat, K. and Gunes, S. (2009), 'A novel hybrid intelligent method based on c4.5 decision tree classifier and one-against-all approach for multi-class classification problems', *Expert Systems with Applications*, Vol. 36, No. 3, pp. 1587-1592.

Quinlan, J.R. (1987), 'Simplifying decision trees', *International Journal of Man-Machine Studies*, Vol. 27, No. 3, pp. 221-234.

Sheu, J.P., Tu, S.C. and Hsu, C.H. (2008), 'Location-free topology control protocol in wireless ad hoc networks', *Computer Communications*, Vol. 31, No. 14, pp.

3410-3419.

Stephanos, A.T. and Diomidis, S. (2004), 'A survey of peer-to-peer content distribution technologies', *ACM Computing Surveys*, Vol. 36, No. 4, pp. 335-371.

Strembeck, M. and Neumann, G. (2004), 'An integrated approach to engineer and enforce context constraints in RBAC environments', *ACM Transactions on Information and System Security*, Vol. 7, No. 3, pp. 392-427.

Tomur, E. and Erten, Y.M. (2006), 'Application of temporal and spatial role based access control in 802.11 wireless networks', *Computers & Security*, Vol. 25, No. 6, pp. 452-458.

Tsaur, W.J. and Lin, Y.M. (2009), 'An agent-based single sign-on scheme for web services environments', *Proceedings of the International Conference on Security and Management*, Las Vegas Nevada, USA, July 13-16, pp. 220-226.

Younas, M., Awan, I. and Duce, D. (2006), 'An efficient composition of web services with active network support', *Expert Systems with Applications*, Vol. 31, No. 4, pp. 859-869.

Yuan, Y. and Shaw, M.J. (1995), 'Induction of fuzzy decision trees', *Fuzzy Sets and Systems*, Vol. 69, No. 2, pp. 125-139.