一個支援移動式多代理者系統之安全模型
A Security Model for Mobile Multi-agent Systems

李明哲　王宗一　蔡昆樺

智慧型網路應用實驗室
國立成功大學 工程科學系
台南市大學路一號
TEL: (06)2757575-63338
E-mail: apple@linuxeagle.es.ncku.edu.tw,
wti535@mail.ncku.edu.tw

摘要

安全性在行動代理者技術的發展與應用上是一項很重要的議題。尤其近年來電子商務的風行，智慧型行動代理者的技術在電子商務型態的網路交易應用上有越來越廣的趨勢，而其中安全性更是一大考量。尤其在一個分散式的環境中，主機(host)與主機間未必建立於互信的基礎上，使安全性的問題更顯重要。目前一般對於行動代理者安全性議題的研究與討論，若非不夠全面性，便是必須透過另一公信機構，例如認證中心(Certificate Authority, CA)[1]來完成。

針對行動代理者及其運載環境在安全性上可能遭遇的問題，本論文列出以下數點加以考量:

1. 惡意代理者對伺服器主機的攻擊。
2. 惡意伺服器主機對行動代理者的攻擊。
3. 行動代理者在異質主機間的遷移及訊息傳遞上的安全性。
4. 來自網路上其他形式的攻擊。

本論文的主要目的即是在已開發出的行動代理者運載環境(Mobile Agent Carrier Environment -MACE)[10]上建立一套完整且全面性、不用透過第三認證中心便可運作的行動代理者安全防護機制。且對於上述可能的安全性問題，提供一套直接有效，且可立即解決的辦法。

一、 簡介

行動代理者是由一個主機移動遷移到另一主機的軟體模組。他們可以互相溝通而且可以在異質的網路中存取分散的資源。行動代理者只需花費很少的網路資源[4]，所以在發展分散式的應用程式上特別有用。現在已經有一些行動代理者的研究，實作及原型應用程式 [3], [5]。他們大部分都使用結合主從式計算及行動碼 (mobile code) 技術的模型，且用於永久連線的電腦上。其中行動代理者本身具有網路漫遊，離線操作及高彈性化等特點，極易應用於無線通訊、電子商務等相關領域。

一般來說，代理者(Agent)可以看成是一個獨立的模組(Module)或程式(Program)。而每個代理者可以單獨執行使用者所賦予的任務，或與其他代理者作互動(Interact)及會談(Meet)。基本上，代理者通常具有以下能力:

1. 搜尋(Search) : 代理者可至遠端伺服器要求搜尋並擷取某些資料。
2. 安排(Orchestrate) : 使用者可將一件任務分成許多工作，在分派給每個代理者，使其合作、協調來完成任務。
3. 旅行(Travel) : 代理者可一需要自動行進至遠端伺服器以執行所需的服務。
4. 會談(Meet) : 代理者可以與其他代理者作互動及會談，以達到分工合作(Collaboration)之目的。

在一個分散式多代理者(Distributed Multi-Agent System)的環境中，主機與主機間未必建立於互信的基礎上。例如對於一個網路購物車的模型而言，購物車漫遊於網路商店進行比價與購物的行為，而商店提供購物車(即代理者)一個執行的環境及其所提供的服務。在這簡單的電子商務模型中，至少存在著兩個安全性上的問題，即:

1. 商店(即 host)與商店間未必互信.
2. 購物車與商店間亦未必互信.

若仔細觀察上述代理者常見的活動，不難看出其中所隱藏的種種安全性上的隱憂。甚者

代理者本身所攜帶的除了自身的執行程式碼外，尚可能包含從某些主機所擷取的機密性資料。以一個行動代理者系統而言，若沒有一套嚴密的安全防護機制，要杜絕網路上有心人士的攻擊與破壞是非常難的。有鑑於此，本論文便針對網路上可能的攻擊逐一分析，發展出一套防護機制及解決的辦法。論文其餘章節大致如下:在第二節中將針對行動代理者的安全考量，提出常見的問題及解決辦法，並比較其中之優缺點。第三節將對行動代理者運載環境(MACE)做個簡單的介紹。而此行動代理者運載環境為行動代理者的執行平台。第四節將介紹對 MACE 系統的擴充，即安全防護機制模組的詳細運作情形。第五節針對其他行動代理者安全議題的相關研究進行比較。第六節為本篇論文之結論。

二、 行動代理者系統之安全考量

網路上有心人士或駭客的攻擊模式可謂千變萬化，但可將之歸納危以下幾種最常見的攻擊行為進行分析與討論。

(一)惡意的的代理者(Malicious Agent)對代理者執行平台(Agent Execution Platform)的攻擊:

由於行動代理者所攜帶的除了一般性的資料外，還包含可執行的程式碼。因此對於一個代理者執行平台而言，其本身允許外來的行動代理者在平台執行的特質便具有極大的安全性上的隱憂。一般常被提及討論的代理者對代理者執行平台的攻擊模式主要有以下三種:

1.偽裝攻擊(Masquerade Attack)

2.未經授權的存取(Unauthorized Access)

3.拒絕服務(Denial of Service，DoS)

現就這三種可能的攻擊模式分述如下:

1. 偽裝攻擊(Masquerade Attack)

所謂的偽裝攻擊，是指未經授權的代理者假冒成一已經授權的代理者，向代理者執行平台提出執行其程式碼或其他系統服務的要求。一未經授權的代理者可能經由偽造、複製、或竊取其他合法的代理者的身分代碼(Identity Code)、或認證密碼進行偽裝。未經授權的代理者除了可經由偽裝攻擊獲得未經授權的系統服務或系統資源外，尚可能破壞原合法代理者與伺服主機之間已建立之信賴關係。

2. 未授權的存取(Unauthorized Access)

一般的行動代理者執行平台都會有存取控制的機制，但若此機制建構的不完善或設計者意外的疏忽，則可能會有未經授權的存取漏洞。例如系統中公開的快取空間(cache space)可能存有之前執行過的代理者的資訊或其所攜帶的資料，若系統內的存取控制機制建構的不甚完善，則此資料或資訊便可能被外界或有心人士設計的代理者所竊取，進而危及系統本身或其他行動代理者的安全。

3. 拒絕服務(Denial of Service，DoS)

拒絕服務模式的攻擊是現今最常見的攻擊形式之一。拒絕服務的攻擊通常是藉由持續的送出大量無用的訊息(例如 ICMP 訊息[12])至某特定電腦或主機(host)中，嚴重拖慢電腦速度，讓電腦無法進行某項動作(例如網頁服務)，嚴重者甚至可導致電腦當機等。拒絕服務的攻擊模式並不會讓電腦內部的資料遭竊取或修改，而是以癱瘓主機為其主要目的。

近年來對於代理者執行平台及一般網路交易的安全防護措施已有為數不少的技術與機制被提出討論與研究。現就較常見的安全防護措施分述如下:

1.JVM(Java Virtual Machine)的應用

Java Language 是目前被廣泛應用於發展行動代理者技術的程式語言之一。其具有下列之先天性的優點:

・Java code 的執行環境為 JVM，因其具有跨平台的特性，程式發展者可以輕易地在異質環境下設計應用程式。

・ Java 本身已具有相當嚴密的安全防護機制，包括記憶體的配置、I/O 存取及系統呼叫程序都受到 JVM 的規範。

2.數位簽章之電子認證

數位簽章為現今電子商務應用上最常見的安全防護機制。數位簽章之電子認證機制具有下列的安全保證:

・可鑑別對方身分。

・防止資料內容被竄改或偽造，確保資料的完整性與私密性。

・易於進行事後之追蹤。

以上所述之防護機制如欲直接採用為行動代理者執行平台最為普遍的安全防護措施，經由相關的研究分析與測試發現，上述兩項防護措施依然有其不足之處。前者 Java 本身的安全防護機制對於代理者對伺服主機上的系統呼

叫部分可以有周詳嚴密的規範，例如系統時間，記憶體配置與清除或其它較為底層的系統呼叫等。但對於前述三種可能之攻擊模式中的1、3點尚無法完全預防，它的安全機制可以說是較為被動的。而後者的數位簽章認證則需額外透過一認證中心進行數位簽章的認證與發送。對於一般行動代理者的應用程式而言，若不透過此一認證機構便無法使用數位簽章的防護機制。

(二) 不友善的執行主機(malicious hosts)對代理者的攻擊：

一個不友善的執行平台對於代理者的影響非常大，就像是比賽中有了不公正的裁判，如果一個代理者的安全性沒有做好的話，遇到了不友善的執行平台就只能任人宰割了，例如在執行一項交易時不友善的執行平台就可以修改代理者的資料，使代理者判斷錯誤，甚至替代理者做決定使代理者做出對特定的執行平台較有利的決定，或是一代理者在收集某項資料時，惡意的執行平台就可能會傳送錯誤的資料給代理者，進而影響到使用者的判斷。因此一個公正的執行平台對於網路上的安全是非常重要的。一個代理者(agent)可能會受到不友善的執行主機的攻擊可分為下面幾種：

- **偵測程式碼與資料**

  執行主機可藉由直接讀取或傾印記憶體(Dump Memory)等方式偵測出代理者的程式碼或由其它主機攜帶而來的資料。
- **修改程式碼與資料**

  若伺服主機擁有代理者執行時期的控制權，則要修改代理者的程式碼亦不是難事。而代理者的資料遭竄改後則可能危及其他主機的權益，甚或對代理者植入病毒等。
- **修改執行程序**

  在一多行程的系統中，惡意的主機可能修改代理者程式的執行程序，而使代理者無法完成任務或得到錯誤的結果。
- **拒絕執行**

  代理者執行主機可能拒絕提供代理者要求的服務等。
- **偵測與其他代理者的互動**

  在一代理者執行平台上，代理者與其他代理者之間的私密會談可能被執行主機偵測出，進而獲得其它主機的機密資訊等。
- **修改與其他代理者的互動**

  藉由偵測代理者與其他代理者之間的互動，惡意的主機可能修改其內容，藉此從中獲取利益。
- **對系統呼叫傳回錯誤值**

  代理者執行平台對於代理者的系統呼叫可能不正確的執行或傳回錯誤值，這個原因未必是惡意的主機造成，亦有可能是未知的系統錯誤。

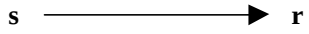
對於上述惡意的執行平台對代理者的攻擊，一般的研究與討論較少提及，但近年來也有一些預防的辦法被提出，如建構在信賴法則上的組織解決辦法(Organizational Solution)[7],運用編碼規則的黑盒子(BlackBox)編碼法, 或參考狀態(Reference State)檢查法等。而這些方法一各有其優缺點。

(三) 網路上其他形式的攻擊

在行動代理者的架構當中，除了代理者與伺服主機的相互偽裝攻擊外，還有其他型式的網路攻擊，便是第三者對在 client 與 server 間來回傳送的代理者做非法的修改與偽裝，而可能的攻擊形式大約有下列數種：

正常情況：

s：傳送端　　r：接收端

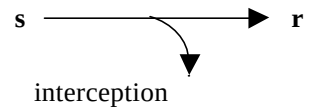


遭遇攻擊：

1. **中斷**（Interruption）：

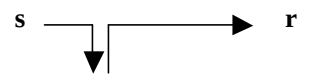
此種情況是在 server 跟 client 間，所連結的硬體線路破壞，除了是人為，也有可能是天災，這種破壞只能單靠改善硬體架構，或者是 server 與 client 間有多條路徑的連接來改善，如此一來就不會因為單一路徑的毀損而導致系統無效。

2. **竊聽**（Interception）：



這類的情形發生在未經授權的第三者，透過擷取網路封包取得 agent 內容，藉此得知 agent 內部由別處攜帶而來的機密資料，或姐由此手段進行更進一步的攻擊等。

3. **擅改**（Modification）：



Modification

這類的情形發生在蓄意破壞的第三者，透過擷取網路封包取得 agent 內容，而再加以修改 agent 的內容，進而從中獲利或破壞。

4. 偽冒（Fabrication）：

這類的情形發生在第三者蓄意假造 agent 內容，再傳送到接受端取的資料，進而從中獲利。

針對上述網路上常見的攻擊手法，由公開金鑰與私密金鑰架構而成的認證編碼法是最直接有效，同時也是目前最普遍使用的防護辦法。

四、 MACE: 行動代理者運載環境之介紹 (Mobile Agent Carrier Environment)

由於執行一個行動代理者必須架構在一個執行環境上，所以本論文將使用一個已經建立的行動代理器運載環境（MACE）作為建置平台，在這章節中則將對行動代理器運載環境（MACE）做個簡單的介紹，若要深入探討請參閱參考文獻[8][9]。

(一) MACE 的架構:

行動代理器運載環境（MACE）的設計的主要考量有以下各點：

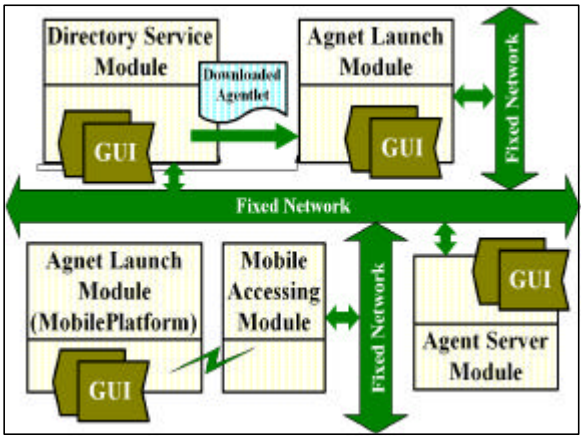
1、提供使用者一良好之使用者界面，可讓使用者方便及快速地獲取遠端之服務擷取資料。

2、讓行動平台使用者不須保持長久的連線狀態。

3、系統必須在一異質分散式的網路環境下執行。

4、具有良好的擴充性並且易於維護。

整個 MACE 系統分為四個主要的部分(如圖一所示)。代理者發送模組(Agent Launch Module)就如同字面意思是創造及發送代理者到網路上的模組。被送出去的代理者漫遊在網路上然後將一些服務項目攜帶到代理者伺服器模組 (Agent Server Module)。在服務協定(Service Protocol)中，代理者發送模組是代理者產生器(Agent Creator)的實作。當需要使用某些服務時，使用者可以查閱目錄服務模組(Directory Service Module) 來找出較適合或較喜愛的服務且下載相關的 Agentlets。而行動存取模組 (Mobile Accessing Module) 就是行動平台將代理者發送到網路或是從網路上接收代理者的一個橋樑。



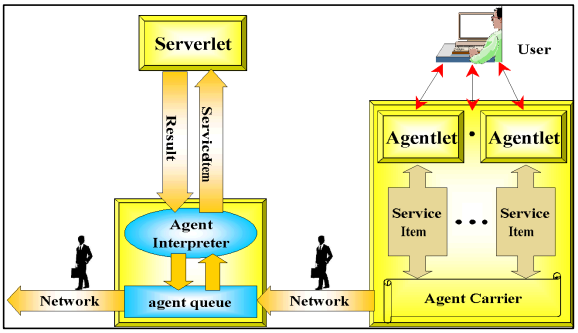
圖一、行動代理器運載環境（MACE）

(二) MACE 採用之策略 - 服務協定

MACE 採取了弱遷移（weak migration）的想法，也就是代理者在執行的狀態下不會被轉移。事實上，它甚至使用了更弱（weaker migration）的機制。服務協定的建立就是用來實現隨選服務的概念，也就是將分散式資源的存取和管理統一抽象化為服務（Service）。在服務協定中，完成一個服務是藉由執行兩個緊密相關的元件，一個是 Agentlet，另一個則是 Serverlet。

一般來說，一個服務的 Agentlet 及 Serverlet 都是由同一個服務提供者所開發出來。Agentlet 可被複製多份，然後分散放置到一些提供目錄服務給所有使用者的專用目錄伺服器上，而 Serverlet 則被存在一服務站台(service station)裡，此服務站台便是此一服務實際被執行的地方。一些經常使用到的 Agentlet 會被下載下來暫存(cache)在使用者機器內的當地目錄（Local Directory）內。服務提供者常會因為某一服務有新的功能而更新其 Agentlet。藉由服務提供者和 MACE 的目錄服務機制的保證，一個使用者將可隨時取得一服務的最新版本的 Agentlet。使用者經由一個代理者產生器 (Agent Creator)來叫用一 Agentlet。一個 Agentlet 在被叫用之後通常會產生一個服務項目 (Service

item)，一個代理者則可包含一個或一個以上的服務項目，而每一個服務項目則在服務站台中對應呼叫一個 Serverlet。因此，一個代理者是運載服務項目而不是程式碼，而一個服務項目是一個 Agentlet 和一個 Serverlet 之間的連接(Connection)。圖二所視為 MACE 服務協定關係之觀念圖



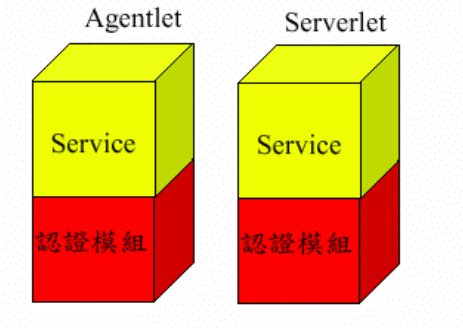
圖二、MACE 服務協定關係圖

採用運載服務項目的優點就是它的內容範圍可以從簡單到只是一些內容提要或是參數值，到非常的複雜、像是一個區塊那麼大且由顧客定義的腳本語言 (Script language) 所寫的程式碼。在後者中，這樣的服務項目所串聯的 Serverlet 會是一個直譯程式(Interpreter) 。而所需之服務則隱含在腳本的直譯過程中完成了。MACE 實際上就是因此而取 Mobile Agent Carrier Environment 這個名字的。因此，此一服務協定簡單但卻強而有力，而主要規範則是 Agentlet， Agent Creator 和 Serverlet 這三者間訊息傳遞的協定。

MACE 已成功地應用在多種應用領域、包括分散式資料庫資料擷取及分散式交易處理[8][9]，並推廣至移動式平台上來支援行動中資料擷取。

四、MACE 系統安全防護機制之擴充

在 MACE 系統之模組中，提供服務的廠商可經由設計或購買一組 Agentlet 及 Serverlet 以提供服務。而本論文所發展的安全防護機制主要也是針對這兩個元件來作設計。



圖三. Agentlet 與 Serverlet 安全模組

如圖三所示，Agentlet 與 Serverlet 除本身所提供之 service 外，尚包含一認證模組。
其認證模組之內容分別如下:

Agentlet : 包含 Serverlet 的公開金鑰(public key)及其本身的私密金鑰(private key)。

Serverlet : 包含 Agentlet 的公開金鑰及其本身的公開金鑰及私密金鑰。

而此一雙向認證之運作流程如下:

一．Agentlet 將 Agent 以 Serverlet 的公開金鑰編碼後發射至伺服端。

二．伺服端接收到 Agent 後，用其本身的私密金鑰將 Agent 解碼，若失敗，則回應錯誤訊息，否則執行 Agent 要求的服務。

三．若伺服端主機為 Agent 網路漫遊之終 點站，則以 Agentlet 的公開金鑰將 Agent 編碼後送回使用者端，並進行步驟五。否則進行步驟四。

四．伺服端將屬於此主機的資料以 Agentlet 的公開金鑰加密,再將 Agent 以其 Serverlet 自身的公開金鑰編碼後送至下一目的地。回到步驟二。

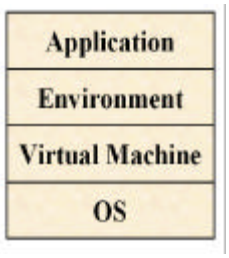
五． Agent 由網路漫遊回原發送站後，原發送站之 Agentlet 利用其自身的私密金鑰將 Agent 解碼。Agent 完成任務。

其中步驟一的動作可確保發送出去的代理者不至受到網路上有心人士的竊聽、複製、與修改。步驟二則是對 Agent 與 Serverlet 的身分作雙向確認，除了伺服主機可確認接收來的 Agent 為合法之 Agent 外，亦可避免伺服主機偽造 Serverlet 對 Agent 進行非法的讀取動作。步

驟四是確保伺服主機的資料除了 Agent 擁有者之外，其他人均無法透過任何方法得知。

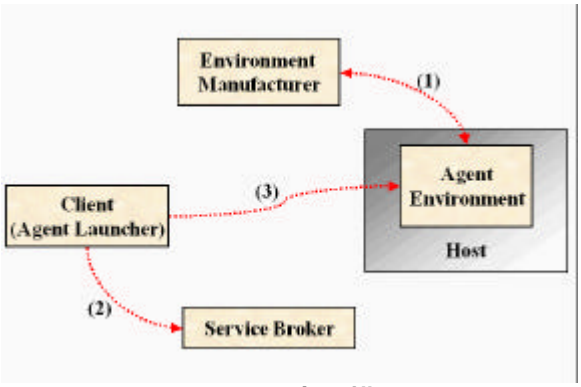
五、 深入探討與相關研究

本節將對本研究的優缺點及可解決的問題進行更深入的探討，並對目前常見的行動代理者安全架構進行比較。



圖四. Normal Agent System Architecture

目前常見的行動代理者系統(如圖四所示)，一般是建構在 Java Virtual Machine 上。Java Language 本身提供了對記憶體存取上的限制，而 VM(Virtual Machine)對於 OS 的操作上也有其限制，例如不正常的 System Call 便會被 VM 給制止。若是使用 VM 的系統，在 2.1 節中第二個未經授權的存取問題便可較輕易的解決。因此一般對於 Agent Security 的探討，均著重於如何防範惡意的代理者及惡意的主機的攻擊。



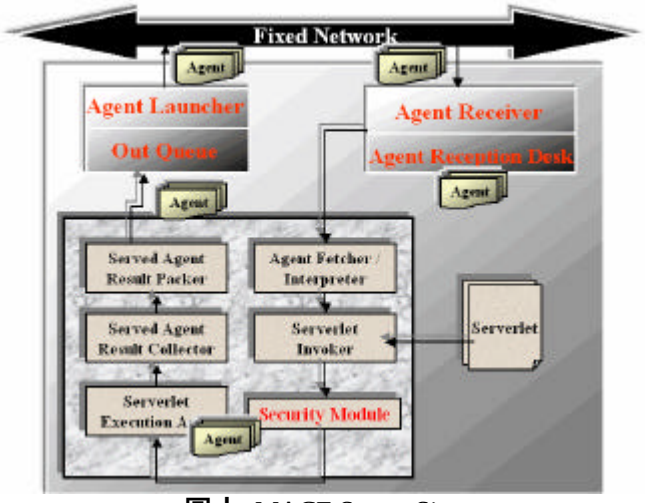
圖五. 交易模型

圖五則是目前行動代理者系統較為普及的交易模型。在此圖中 Client 是 Agent 的擁有者及發送者、Host 則提供 Agent 執行的 Site、Service Broker 紀錄著哪些提供服務的 Host、Agent Environment 則是 Agent 執行的平台、此外 Environment Manufacturer 乃是 Agent Environment 製造商。而其交易流程如下:

1. Environment Manufacturer 提供 Agent Environment 給 Host。
2. Client 向 Service Broker 查詢可用的 Host，並得到 Agent Environment 之 Public Key。
3. Client 發送 Agent 至 Host，Host 將其送至 Agent Environment。
4. Agent Environment 將 Agent 解密並執行。
5. 若下一站為 Client，則用 Client 之 Public Key 將 Agent 加密。若下一站為其他主機，則用 Agent Environment 之 Public Key 加密，再將 Agent 送出。

由上述的交易流程可以看出，此架構至少具有幾個問題:

1. 需由一額外的機構提供加密金鑰。
2. 對主機與主機間的機密資料無法提供完善的保護。
3. 主機對 Agent 具有幾乎完全的控制權。
4. 加解密金鑰取得問題。



圖六. MACE Server Site

圖六為 MACE 系統 Server 端的運作流程。由圖中可看出，當系統中的 Serverlet 被喚起時，便開始執行 Agent 與 Serverlet 的雙向驗證步驟，雙方均驗證成功後，Agent 才能進一步執行並漫遊，否則便會被系統拒絕。現就上述一般交易模型的三項缺點進行探討並說明 MACE 的安全架構如何足以解決這些問題。

(一) 問題一

關於第一點，在一般模型中，使用者須向 Environment Manufacturer 取得 Agent Environment 的 Public key，或經由 Environment Manufacturer 將 Public key 交給 Host，而當 Host 向某些 Service Broker 註冊後，當使用者向其查詢資訊時，才獲得此 Public key。若 Host 向 Service Broker 註冊假的資料，例如 Host 宣稱使用某一公正的 Agent Environment，但實際卻使用另一個惡意 Agent Environment 並給予假的 Public key，將造成 Agent 被攻擊。

而在 MACE 上 Agent 的服務是藉由一組 Agentlet 及 Serverlet 程式所達成，這一組程式是由廠商所自行製造，而且由 Serverlet 產生一 Private key，並產生一相對應的 Public key 給 Agentlet，使用者若要對 Agent 內容加密，只要使用 Agent 所要求服務項目相對的 Agentlet，便可以完成加密動作，而且 Serverlet 可以放置在多個 Site 上，使用者僅需下載相對應的 Agentlet 便可以在多個 Site 上享有相同型態的服務。

(二) 問題二

關於第二點，在一般模型中，Agent 可在不同主機間漫遊，例如網路購物車的設計，Agent 可能漫遊於不同的網路商店進行購物或比價的動作。若每個主機使用相同的 Agent Environment，則主機 A 交予 Agent 的資料(可能是其商品的價格)在當 Agent 漫遊至主機 B 時，便完全不具保密性了，因為兩者是使用相同的加解密金鑰。若每個主機使用不同的 Agent Environment，則 Key 的取得也是一個問題。

而在 MACE 的設計中， Agent 到達 Site 後，由 MACE 提供 Agent 停留的環境並等待被執行，當 Agent 可以被執行時，由 MACE 呼叫其相對應的 Serverlet 來進行服務，若此 Site 惡意呼叫錯誤的 Serverlet 起來執行，由於 Agent 是由正確 Serverlet 所提供的 Key 加密，就算錯誤的 Serverlet 欲加以解密，也無法獲得 Agent 正確的資料，相反地，不合法的 Agent 也無法獲得正確 Serverlet 的服務(Serverlet 可以檢查此 Agent 是否是由其相對應的 Agentlet 所加密)，藉由此方式可以保證 Agent 不會遭受 Host 惡意的攻擊，不合法的 Agent 也不會被服務。

(三) 問題三

關於第三點，在一般模型中，最主要的問題是在於服務與執行環境無法獨立作業。雖然 Agent Environment 是由 Environment Manufacturer 提供，但 Host 與 Agent Environment 之間的相依性依然是非常的重。假設此 Agent Environment 是 Java-Based 系統，便需 Host 提供虛擬機器，且 Agent Environment 的功能亦依 Host 所提供的服務(Service)而變。Host 可能藉由提供一套偽裝的服務環境來竊取 Agent 所攜帶的機密資料。

在 MACE 的設計中，MACE 本身只擔任 Agent 的運載環境，而服務的提供者是 Serverlet，不同的服務便對應不同的 Serverlet。且 Serverlet 可由獨立的單位提供，未必須要跟 MACE 相關。如此將服務提供者由執行環境獨立出來，除了可給予主機對於服務的提供上有相當大的彈性外，更可確保執行環境的公正性。

(四)問題四

關於第四點，在一般模型中，當 Agent 在 Agent Environment 上執行完畢後，也是利用 Agent 本身的 Public Key 將結果做加密， 但接著若還要漫遊至其他的 Site，可能需在使用其他 Agent Environment 的 Public key 來做加密(因為不能保證每一個 Site 上的 Agent Environment 都是同一家廠商製造)，這可能造成一個問題是 Agent 一開始必須攜帶所有可能到達的 Site 其 Agent Environment 的 Public key，或著由目前的 Site 去取得下一個 Agent 要到達的 Site 其 Agent Environment 的 Public key，依照上述的問題，不是會造成 Agent 須帶許多 Key 漫遊(在這種狀況下，Agent 還必須確定下一個目標 Site 其 Agent Environment 的 Public key 是那一個)，就是造成效率變差(假設由目前的 Site 去獲得下一個 Site 上 Agent Environment 的 Key)。

而在 MACE 上，當 Serverlet 執行完 Agent 要求的服務後，Serverlet 利用 Agent 本身帶來的 Public key(此 Key 是由使用者提供)將寫入至 Agent 資料做加密，然後在使用 Serverlet 的 Public key 將 Agent 整個內容做加密，接著交由 MACE 送至其他 Site 去(由於能服務 Agent 的 Serverlet 具有相同的 Key，故 Agent 本身並不需在攜帶 Serverlet 所提供的 Public key，除了使用者的 Public key 外，也不需在攜帶其他多餘的 Key)。

六、 結論

在一個多代理者且分散度高的環境中,行動代理者的安全問題可畏層出不窮。其中主機與主機間的不信任，代理者與主機間的不信任，以及在一個分散的環境下，其他來自於網路上第

三者的竊聽，攻擊等，更加深了安全性的複雜度與困難度。本論文以之前所建置的行動代理者運載環境(MACE)[10]為平台，配合 MACE 中獨特的 Agentlet 與 Serverlet 之設計，提出一套適合在分散式環境及多代理者系統中運作的安全機制。對於一般討論行動代理者安全性的論文中常遇到的困難點，例如代理者在主機內的運作，及代理者攜帶機密性的資料，在不具信賴基礎的不同主機間的遷移及執行等，本研究均有詳細的探討並提供整體的解決方案。藉由此架構，代理者與代理者、代理者與主機、及主機與主機間未必須要建立互信的基礎，且不需透過第三認證中心便可在一安全的環境下正常的運作。

七、參考文獻

[1] Government Certification Authority，URL = http://www.pki.gov.tw/

[2] S. Covaci,; Zhang Tianning; I. Busse, "Java-based intelligent mobile agents for open system management", In Proceedings of Ninth IEEE International Conference on Tools with Artificial Intelligence, Page(s): 492 -501, 1997.

[3] Gunter Karjoth, Danny B. Lange, and Mitsutu Oshima, "A Security Model for Aglets,", IEEE Internet Computing, JULY,AUGUST 1997.

[4] Keith D. Kotay and David Kotz, "Transportable Agents ". In Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management, Gaithersburg, Maryland, December 1994

[5] D. Kotz; R. Gray; S. Nog; D. Rus; S. Chawla; G. Cybenko, "AGENT TCL: targeting the needs of mobile computers", IEEE Internet Computing Volume: 1 4 , Page(s): 58 -67, 1997

[6] 李明哲, 王宗一,蔡昆樺,黃源龍,顏仲偉, "行動代理者系統之安全防護機制", Workshop on the 21st Century Digital Life and Internet Technologies, 17-18, May, 200, Tainan, Taiwan

[7] General Magic Company，URL = http://www.generalmagic.com/indexflash.html

[8] Wang T. I. "A Mobile Agent Carrier Environment with Mobile Computing Facilities", IIP: International Conference on Intelligent Information Processing, The 16th IFIP World Computer Congress.21~25/08, 2000, Beijin.

[9] Wang T. I. "A Mobile Agent Carrier Environment for Mobile Information Retrieval", 11-th International Conference on Database and Expert Systems Applications - DEXA 2000, 05~08/09, 2000, Greenwich, London.

[10] Wang, T.I. "A Mobile Agent Carrier Environment", ICS2000, 6-8 December, 2000, Chiayi, Taiwan, R.O.C.

[11] Wang, T.I, "以行動代理者支援行動資訊擷取", 第五屆人工智慧與應用研討, Taipei, 17/Nov. 2000

[12] RFC2521