計畫編號：89shu20

行政院衛生署「二代全國醫療資訊網計畫」

各項試辦計畫之委託發展計畫成果報告

以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件

研 究 報 告

執行機構：臺北醫學大學(臺北醫學院)

計畫主持人：劉立

研究人員：

執行期間：八十九年七月十二日至九十年三月三十一日

＊＊本研究報告僅供參考，不代表本署意見＊＊

目錄

圖目錄

中文摘要

關鍵詞: 公用物件請求仲介者架構(CORBA)、健康資訊交換第七層協定(HL7)、可擴展標注語言(XML)、密碼(Cryptography)、安全(Security)、元件化(Component Based)

研究目的

醫療資訊的暢通，避免了重覆性的檢驗，並可減少許多的醫療資源浪費。同時使得醫療人員，得以根據此完整之資訊，針對病人的病情做出更精確的診斷及更適當的處置，提高醫療品質。網路化及電子化病歷具有提供快速搜尋及統計之能力，使得醫生的醫療行為更有效率。利用此一完整的資料庫資源，更能增進醫學研究的發展。

傳統的 Two-tier Client-Server 架構顯然無法再應付上述的需求，取而代之的是 Multi-tier 的觀念。新架構的觀念在於強調將企業邏輯(Business Logic)與系統所提供的服務(System Service)及使用者介面(User Interface)能確實的彼此分隔出來，企業邏輯置於中間層內(Middle-tier)、 資料放置於後端(Back-End)的資料庫內以供多人操作存取(Share)、 而前端(Front-End)僅提供輕型(Lightweight)的使用者操作介面並能輕易的部署(Deploy)至使用者端(Client)的機器上。

導入公用物件請求仲介者架構(COBRA)，可使醫療單位跨組織的活動

更具效率並可降低成本。前端的使用者介面層與中間層之間的溝通為了更適合於分散式的網路環境(尤其是要能應付於 INTERNET 的低頻寬網路)，而導入了以分散式物件為主的訊息導向(Message-oriented)方式；而企業邏輯則以元件化的架構(Component-based Architecture)來落實以保持系統的彈性及增加可重用性(Reusable)；而為了能控管交易行為，中間層的系統要能提供關於元件化的Transaction Monitor機制；同時，提供一個安全(Security)、穩定性(Scalability)、可攜性(Portability)、永續性(Persistency)、一致性及共通性(Consistence & Concurrence)的服務也是應用系統所應具有的機制。

研究方法

1. 以 Enhanced CORBA 架構為基礎。
2. 以 IDL 來描述交換格式，利用 JAVA IDL 將其轉成 JAVA 。
3. 以 JAVA ORB 作為 Object Broker 。
4. 以 JAVA Security 作為安全管理。
5. 以 JAVA Cryptography 作認證、加解密、簽章。
6. 以 JAVA XML 產生 HL7 交換文件。
7. 提供 Adaptor 以便不同技術之引用。

主要發現

以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件，整體完成後具有下列之功能:

1 Identification & authentication(認證) of principle : principal(主角) 可能是使用者或者是一個物件，他們必須透過確認來證實身分。Authorization(授權) & access control (存取控制)通過確認後，根據 principal 的 attributes (屬性)來決定他的存取控制權。

2 Security auditing(監聽) : 對於物件的動作，能做 log 紀錄，這樣可以讓使用者或管理者來觀看了解物件間的動作，這樣可以讓一連串的物件執行後仍能去檢查正確性和安全性。

3 Security of communication(通信安全) : 物件和物件間信息的傳遞需要有一個機制來作安全的保護來防止竊聽或是修改。

4 Non-repudiation(使不能否認) : 對於每一個存取動作，能夠留下一個證據，來防止事後的否認。

結論

1. 利用 http 協定，以 URL 存取資料，提供一個簡易的使用介面。
2. 經由 HL7/XML 之協定交換資訊，符合醫學資訊交換標準。
3. 透過 Web Adaptor 連接 CORBA Agent，降低各醫療院所進入障礙。

4. CORBA Agent 提供 Private/Public Key Encryption/Decryption
5. Security Component 內含 Secret Key，跨越主機之 Security Component 之間傳遞參數可利用 Secret Key 加密，以免遭受竊聽。
6. SMILE Layer 亦利用 Secret Key 加密，宛如置於 Virtual Private Network 上執行。

建議事項

Component based的Multi-tier架構已是未來開發醫學資訊系統的主流，透過本研究規格的制訂與實作更可以讓醫學資訊系統的開發人員除了專注在醫學邏輯的開發上，更不用擔心受限於單一廠商的束縛，除了可以縮短開發的時程、簡化資訊系統的部署(Deployment)及維護(Maintenance)，並且具有高度的可攜性及穩定性，更進一步享有完整的安全性。醫療機構間不限於單純的資料交換，而可以協同進行組織間的醫療活動，像是醫藥分業、轉診等。醫療機構間也可以利用呼叫對方系統物件庫內的物件來得到服務，值得推廣。

Abstract

Objectives

The facility of healthcare information exchange avoids wasteful repeated testing and makes more efficient for the utilization of medical resources. Also, supporting with better and complete information for healthcare entities to make more accurate and appropriate treatments from the diagnosis of patients. The network-based and electronic data enable the efficiency of treatment, quality measurement for doctors' medical behaviors in response to the capacity of rapid search and accurate statistics. Using this complete database resource, the development of medical research can be improved.

The concept of Multi-tier is presented as a substitute for conventional two-tier client/server architecture which is obviously no longer to meet the needs of above mentioned. This new concept emphasizes the business logic, system service and user interfaces can be segmented concretely. The multi-tier architecture consisting of business logic in middle-tier, back-end data storage tier includes the sources of data available to share and operate among numbers of end-users, front-end provides simply lightweight user interface that can be deployed easily to client side.

By the contribution of COBRA, it leverages the efficiency and cost saving for activities crossing different organizations. To enhance the communication between front-end user interface tier and middle tier in a distributed network-based environment (especially to manage the Internet works on low bandwidth), it requires the application of distributed message-oriented object. The business logic constructed with component-base architecture to keep system flexible and reusable. While managing the transaction behavior, the middle tier should be able to provide the mechanism of component-based transaction monitor. In the meantime, the application system should offer services with the mechanism of security, scalability, portability, persistency, consistence and concurrence.

Methodology

1. Based on enhanced CORBA architecture.
2. Describe the exchange format by IDL, transfer into JAVA by JAVA IDL .
3. Use JAVA ORB for Object Broker .
4. Use JAVA Security for Security management .
5. Use JAVA Cryptography for authentication, encryption/decryption, signature .
6. Use JAVA XML to provoke HL7 for document exchange.

7. Provide Adaptor for different technical quoting.

Discovery

Establish a collaborative security object for medical information exchange center by using extensive CORBA based architecture can be committed to following functionality after overall implementation:

1. Identification and authentication of principle: Principle can be a user or object and must be qualified its identification by reconfirmation. After reconfirming for authorization and access control, can decide access control by attributes of principle.
2. Security auditing: Can log record from object's operation, this ensures user or management to monitor and understand the interaction of objects also, enable to check over the accuracy and security after serial object execution.
3. Security of communication: Provide a mechanism during the transmission of messages between objects to protect security from modification or wire-tapping.
4. Non-repudiation : Leave evidence after each access to avoid future repudiation.

Conclusion

1. Using HTTP protocol, access data by URL and provide a easy user interface.
2. Through HL7/XML protocol for information exchange, compliance with the exchange standard of medical information.
3. Reduce the barrier among different medical organizations by Web Adaptor connected CORBA Agent.
4. CORBA Agent provides Private/Public Key Encryption / Decryption
5. Security Component including Secret Key for encryption of parameter transmitting between Security Component while across servers.
6. SMILE Layer can also encrypt with Secret Key, like executing over Virtual Private Network.

Suggestion

It's no doubt that component-based multi-tier architecture becoming as momentum for future development of medical information system. Through the specifications and practices in this research, it will permit the R&D staffs of medical information system to concentrate upon medical logic development and also, avoid the limitation bundled by only single vendor. Other than shorten development lead-time, simplify the deployment and maintenance of

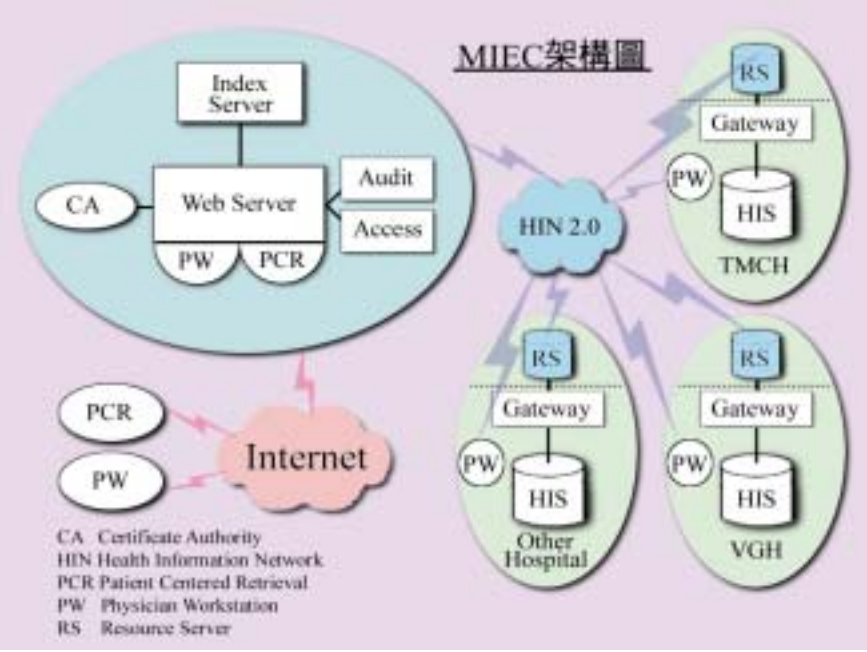
information system, this research provides system with highly portable and scalable functionality, moreover, offer an incredibly complete security. Meanwhile, information exchange is not the only application, more cooperative activities among different medical organizations like pharmacy and medical segregation, data transformation can be realized as well. Further, more services can be practiced by calling system objects from database between medial organizations for the benefit of consumers. In any case, it is indeed a valuable research worthy to be widely promoted and applied.

medical activities oriented security and access control can really resolve the security management of medical records. Beside, establishing Certificate Authority (CA) which dedicates in Key management of Encryption and carrying out in every security protection point resolves all security scruple. To ensure the access control of inseparable medical information group must base on the medical records which have completeness and unity. Medical activity oriented can control the access and lose of medical records in order to avoid the released authority can’t be revoked.

前言

研究問題之背景與現況

民國八十四年三月全民健康保險制度實施以來, 國內的醫療院所無可避免的面臨與全民健康保險局交換電子資料的問題。促成了國內基層醫療界目前全面資訊化的現狀。而醫藥分業與轉診制度實施後，基層院所與健保局以外的醫療院所、藥局也開始交換電子資料，醫療單



MIEC 架構圖

位跨組織的活動日益頻繁， 單純 EDI 之電子資料交換技術已漸不符合所需。尤其醫療資訊擁有高度的私密性，因此如何建構醫學資訊交換中心安全物件之標準，實刻不容緩。

申請人已經完成醫學資訊交換中心之醫學資訊在網路安全防範及認證策略之研究(Secure Medical Information Link Everywhere –

SMILE)，然對於與 MIEC 相連之 Gateway 設計，基於下列理由認為有改善之必要

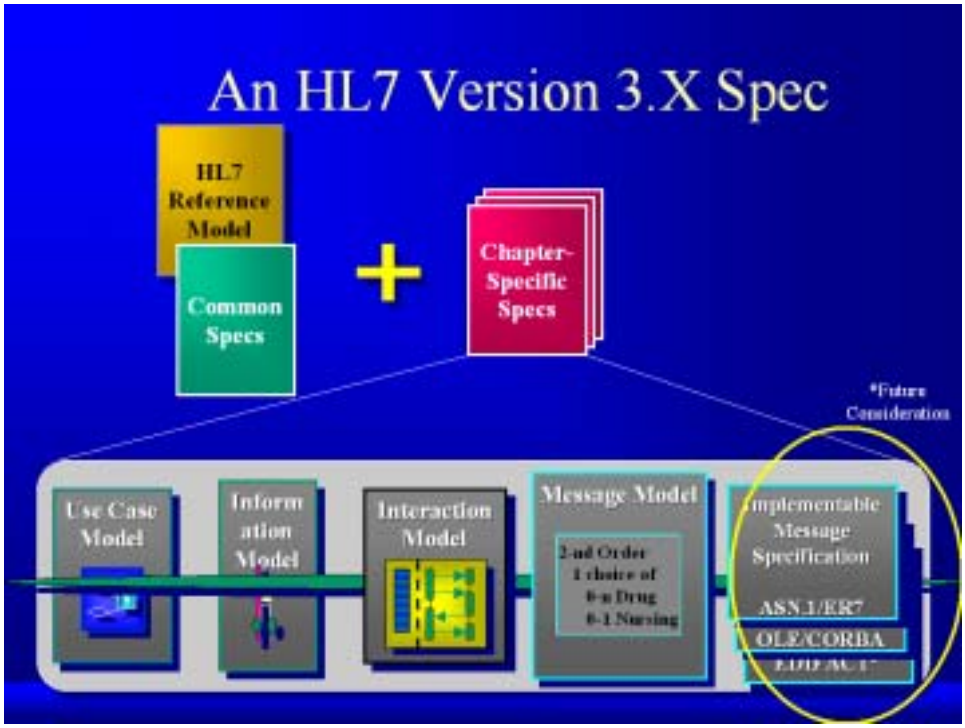
1. Security Management
2. Automatic Stub and Skeleton for Medical Exchange Format
3. Standardize Component
4. Handle Distributed Environment
5. Used for Multi-platform
6. Programming Language Independent
7. Location Transparency
8. Component Reuse
9. Server Activation
10. Vendor Independent

傳統的 Two-tier Client-Server 架構顯然無法再應付上述的需求，取而代之的是 Multi-tier 的觀念。新架構的觀念在於強調將企業邏輯(Business Logic)與系統所提供的服務(System Service)及使用者介面(User Interface)能確實的彼此分隔出來，企業邏輯置於中間層內(Middle-tier)、 資料放置於後端(Back-End)的資料庫內以供多人操作存取(Share)、 而前端(Front-End)僅提供輕型(Lightweight)的使用者操作介面並能輕易的部署(Deploy)至使用者端(Client)的機器上。

導入公用物件請求仲介者架構(COBRA)，可使醫療單位跨組織的活動更具效率並可降低成本。前端的使用者介面層與中間層之間的溝通

為了更適合於分散式的網路環境(尤其是要能應付於 INTERNET 的低頻寬網路)，而導入了以分散式物件為主的訊息導向(Message-oriented)方式；而企業邏輯則以元件化的架構(Component-based Architecture)來落實以保持系統的彈性及增加可重用性(Reusable)；而為了能控管交易行為，中間層的系統要能提供關於元件化的 Transaction Monitor 機制；同時，提供一個安全(Security)、穩定性(Scalability)、可攜性(Portability)、永續性(Persistency)、一致性及共通性(Consistence & Concurrence)的服務也是應用系統所應具有的機制。

Component based 的 Multi-tier 架構已是未來開發醫學資訊系統的主流，透過本研究規格的制訂與實作更可以讓醫學資訊系統的開發人員除了專注在醫學邏輯的開發上，更不用擔心受限於單一廠商的束縛，除了可以縮短開發的時程、簡化資訊系統的部署(Deployment)及維護(Maintenance)，並且具有高度的可攜性及穩定性，更進一步享有完整的安全性。醫療機構間不限於單純的資料交換，而可以協同進行組織間的醫療活動，像是醫藥分業、轉診等。醫療機構間也可以利用呼叫對方系統物件庫內的物件來得到服務。醫學資訊交換規則改變時，只需要更換物件庫內物件即可更新軟體版本。HL7 3.X 更將 CORBA 納入其中。



HL7 Version 3.X Spec

研究目的

以技術的觀點而言，傳統的 Client/Server 的系統已不敷使用，必須利用分散式物件方得以解決，而在分散式的環境下有下列問題須要注意

1. 分散式物件可能同時扮演 Client 和 Server 的角色:在傳統的 Client / Server 架構下，誰的角色是 Server、誰的角色是 Client 是很清楚的。我們可以完全信賴 Server，而不可信任的是 Client，可是在分散式物件環境的架構下，一個物件可能同時扮演 Client 和 Server 的角色，我們不能清楚的分出誰是 Client、誰是 Server。

2. 分散式物件會不斷援用新的物件:在分散式物件環境的架構下，使用了一個物件，但實際上我們可能只看到冰山的一角，因為這個物件可能會再援用新的物件在執行的時候，而且物件的 implementaion 也可能會隨時間而改變。
3. 分散式物件間的互相影響很難切確的去了解:因為 encapsulation 的關係，使用了一個物件我們很難完整的掌握這個物件可能會在援用新物件時會發生什麼互相影響。
4. 分散式物件間的互相影響不好預測:因為分散式物件環境很有彈性，物件間的互相影響有更多的可能性，所以很難預測。
5. 分散式物件是有多形態的(polymophic):分散式物件是很有彈性的，在 polymophism 的設計下，物件有相同的 interface，在這樣的情況下，很有可能別人會設計特洛伊木馬。
6. 分散式物件是非常動態的:分散式物件時時可以產生新的物件，也會時時刪除一些物件，所以難以掌握，也成了 Security 的夢魘。
7. 避免成為單一廠商的奴隸:在 Multi-tier 的環境下，廠商所開發的 Application Server 擔任著企業資訊系統中不可或缺的角色，然而，系統所提供的服務(System-level Service)若沒有一定的標準介面，企業資訊系統的開發勢必迫於現實環境的考量而決定採用單一廠商所提供的服務，相對的也就限制開發人員開發的能力而屈就於產品面

的學習，而後該資訊系統也就鎖死(Lock-in)在該廠商的束縛下了。

本計劃之主要目的乃是利用已經發展之現有基礎, 以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件，整體完成後具有下列之功能:

1. Identification & authentication(認證) of principle : principal(主角) 可能是使用者或者是一個物件，他們必須透過確認來證實身分。Authorization(授權) & access control (存取控制)通過確認後，根據 principal 的 attributes (屬性)來決定他的存取控制權。
2. Security auditing(監聽) : 對於物件的動作，能做 log 紀錄，這樣可以讓使用者或管理者來觀看了解物件間的動作，這樣可以讓一連串的物件執行後仍能去檢查正確性和安全性。
3. Security of communication(通信安全) : 物件和物件間信息的傳遞需要有一個機制來作安全的保護來防止竊聽或是修改。
4. Non-repudiation(使不能否認) : 對於每一個存取動作，能夠留下一個證據，來防止事後的否認。

材料與方法

本研究進行的步驟為:

1. 整理系統安全之顧慮
2. 整理醫療資源之保密策略方式
3. 制定安全性策略
4. 分析系統之需求
5. 架設 Java ORB 相關環境與伺機器
6. 訂定之 Component 規格
7. 以 Java Cryptography 及 Java Security API 實作相關 Component
8. 研究 Prototype 使用之對象及格式
9. 實作 Prototype 以測試 Component
10. 各單元測試
11. 整合測試

本研究將以 JAVA Platform 來實作，並以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件，其主要方法為:

1. 以 Enhanced CORBA 架構為基礎。
2. 以 IDL 來描述交換格式，利用 JAVA IDL 將其轉成 JAVA 。
3. 以 JAVA ORB 作為 Object Broker 。
4. 以 JAVA Security 作為安全管理。

5. 以 JAVA Cryptography 作認證、加解密、簽章。

6. 以 JAVA XML 產生 HL7 交換文件。

7. 提供 Adaptor 以便不同技術之引用。

結果

安全目標

安全目標在於保護資料及系統資源的機密性(confidentiality)、整體性(integrity)、可取用性(availability)。機密性主要的對象是資料，機密性的目標在醫學上除了一般秘密性(secrecy)與隱私性(privacy)的要求以外，有著較強烈的要求，由於醫療記錄的擁有者是病患，但其產生者卻是醫師，而非醫師病患本人，因此 confidentiality 的要求又多了一層困難。

整體性(integrity)要求資料只能以可接受的方式更改、只能被已授權人員或處理更改、相關資料間必須保持一致性(consistent)以及有意義的正確結果。由於醫學資料豐富性及完整性的要求，擁有 Video，Image 及 Text 等不同型態的媒體格式常結合於同類別的醫學資料，例如一個肝癌病患可能在不同的醫院中做過 CT、MRI 及腹部超音波等檢查，其影像資料如無文字報告敘述，此資料是難以成為一份完整的醫學資料，就權限之設定與描述亦有困難，因此整體性的考量在醫學上就更加重要。

醫學資訊乃是生命攸關的資訊，可取用性的要求自不待言，醫師看診的立即性以及服務身心靈受創之患者的急迫性，資料與系統必須能夠及時回應(timely response)、公平配置(fair allocation)並且具有容錯

(fault tolerance)的能力，因此可取用性的要求更是一般系統所無法比擬

安全目標的考慮首重實體安全，實體若不安全其他的安全皆屬虛妄，一但實體安全得以確保，資料的安全乃接踵而至，資料乃是現今社會最有價值的財務，筆者相信緊接著實體的保險外，資料定是 21 世紀最有潛力的保險對象。資料要安全，處理資料的程式安全以及所處的系統環境安全亦是不容忽視，因此安全目標必須從四個方向來考慮:

1. 實體安全：門禁管制、消防設備、媒體出入管制、資訊線路之管制及災害應變計劃、設備定期維護。
2. 資料安全：檔案的備份、檔案保管人與維護人清單、訂定擷取檔案資料權責、檔案機密等級分類、研討檔案遭損壞之風險接受程度、資料的加密及解密。
3. 程式安全：一切已正式啟用的程式、模組的變更管理、問題管理、個人電腦硬式磁碟使用管理、網路監控管理、線上傳輸異動代號管理、終端機使用權責管理、特定人員之專用線路及路線接頭控制、密碼的規則與變更期限都包括在內。
4. 系統安全：網路作業系統為整個作業環境的中樞，若系統遭人蓄意破壞，將無法產生有用的資訊，因此事先必須詳細規畫並設定好網路使用者所能使用的資源及帳號和密碼，之後時常監控網路環境的變化。

系統基礎

密碼學

在一般人的觀念中，談到資訊安全時最常想到的是保護資料不被他人竊取。因為電腦網路是個開放的環境，在網路上流通的資訊都可能遭他人竊取。既然無法防止他人竊取，唯一的方法就是先將資料加密，再於網路上傳送，等傳到目的地後再加以解密。若他人在網路上取得傳送中的資訊，因為是加密過的資料，竊取的人無法解讀其內容，這樣就達到了資料保密的目的，也就是所謂「機密性」的服務。

早在古希臘羅馬時代，人們就使用了密碼的技術。當時使用的方法很簡單，將原始的文件資料 (明文，plaintext) 中每個字母向右移動 (或迴轉) 三個位置，也就是 A 變成 D，B 變成 E，而 X 變成 A。經過這樣的轉換後資料就成了密文 (ciphertext)，接收資料的人再將密文反向運算，即可得出原始的文件內容。

像這樣的密碼系統其關鍵在於所使用的演算法只有發文與收文的兩方知道，如果其他人也知道的話，機密性將不復存在。這種將機密性建立在密碼演算法的密碼系統，只適用於封閉性的環境，但現今網路通訊是開放性的，因此必須使用共通的密碼系統，也就是說密碼演算法必須公開，而任何人都可以使用。

為了因應這樣的需要，現代的密碼系統使用了鑰匙 (key) 的概

念。資料的加密與解密需要配合鑰匙，由特定的密碼演算法進行。使用的演算法是公開的，需要保護的則是鑰匙。因為演算法公開了，因此必須確保演算法本身是安全的、無法破解的。而所謂的鑰匙是一連串 0 與 1 的組合，若使用暴力入侵，得嘗試各種不同 0 與 1 的組合。假設鑰匙的長度為 N 位元，就可能有 2N 種不同的鑰匙。因此鑰匙的長度 (位元數) 越長，理論上安全性越佳。

密碼學技術主要透過包括明文之混淆(confusion)及擴散(diffusion)兩種方式而成為密文，其所謂之安全並非不可由密文回復成為明文，而是根據計算理論的 NP-Complete，計算複雜度上不可行，例如給定可用的處理速度及記憶體，還是很難因數分解 (factorization)或計算離散對數(discrete logarithm)。密碼系統破密複雜度可以分為四類：資料複雜度(data complexity)、計算複雜度(computation complexity)、記憶體複雜度(memory complexity)、成本效益考量(cost effective)，透過上述四類複雜度而達到計算上安全。

密碼系統被破解之程度可以分為完全破解(total break)、全域推論(global inference)、局部推論(local inference)及資訊推論(information inference)。完全破解乃是獲得加解密之金鑰對或計算出另一金鑰對，而全域推論就是找到另一個替代的密碼演算法而可以得到原先演算法所產生的任何相同結果，局部推論係指由某一個密文去推導出所對應

的明文，資訊推論即為由一些密文中去推論出有關明文或金鑰的資訊。

因此破密的方式可能由給定一些密文，推導所有的明文或、密鑰，或從某一個密文推導所對應的明文(ciphertext-only attack)，此外可能由某些明文及密文對推導密鑰(known-plaintext attack)，亦可能預先選擇的明文及其相對應的密文推導密鑰(chosen-plaintext attack)， 另外在未知金鑰的前提下，可以計算出合法的明文/密文對或可被驗證成功的訊息/簽章(chosen-ciphertext attack)。除了上述由明文/密文之攻擊方式以外，亦可能從某一些金鑰中獲得金鑰產生關係，進而得之對應於某一把公鑰之私鑰(chosen-key attack)，根據預測目前(公元 2000 年)至五年內可以達到安全的私密金鑰長度為 128 位元公開金鑰長度為 1024 至 2048 位元。

SSL(Secure Socket Layer)乃是常用的網路安全協定，是由網景公司開發，用來保護網上使用瀏覽器交易安全的規格，因為各家瀏覽器軟體都支援它的功能，因此是目前在網路上最受到廣泛採納的一種。SSL 傳輸的資料也是經過鑰匙加密的處理，雖然有可能被第三者截取，卻很難讀取資料內容，而且經過加密的資料可以保持完整，不會受到竄改或破壞。

至於 SET(Secure Electronic Transaction)安全電子交易標準則由萬事達與威士兩個信用卡組織主導，結合 IBM、微軟、網景等國際資訊

廠商共同推廣的網路電子商務交易安全標準，商家可以利用 SET 確認消費者身份，但不會看見消費者信用卡的號碼，因此消費者在網上沒有被盜刷的危險，不過 SET 的系統太過複雜，建置的成本稍高，所以目前電子商務上的保密協定，還是以 RSA、SSL 系統為主。

醫療資訊交換是十分依賴密碼技術的，在網路上傳送的資料可採用密碼技術加以保護，加密過的訊息在網路上流通時就算被駭客竊取，也無法將訊息解密，因此金鑰的管理相當重要:

1. 金鑰產製 -- 憑證管理中心之初始金鑰對產製，須由多位經授權之憑證管理人員所持有之 Key 驗證後才可啟動產製流程。用戶之金鑰產製有兩種方式，第一種由用戶自行產生，再將公開金鑰交給憑證管理中心簽發憑證；第二種由具有公信力的第三者代為產生金鑰對後，將秘密金鑰對交給用戶保管使用，並將與產生該金鑰有關的所有相關資料銷毀，然後把公開金鑰傳送憑證管理中心簽發憑證。
2. 金鑰存取 -- 憑證管理中心之私密金鑰存取時須由多位經授權之憑證管理人員所持有之智慧卡驗證後才可存取。
3. 金鑰備援 -- 憑證管理中心之私密金鑰分割為三張智慧卡備援，此備援存放在一安全獨立的環境，存取時須由多位經授權之憑證管理人員所持有之智慧卡驗證後才可啟動。
4. 金鑰對更新替換 -- 憑證管理中心之金鑰對定期更新，在金鑰的使用期限到達之前，系統會依照金鑰產製方式重新產生金鑰對，簽發新的公開金鑰憑證

5. 金鑰危害處理 -- 本憑證管理中心的私密金鑰有危害之疑慮時，重新產生金鑰對，重新簽發一份新的憑證，同時通知其他憑證機構及用戶更換憑證，以避免該可疑金鑰之使用。

醫療資訊除了密碼技術的，電子簽章(Electronic Signature)的重要性亦不容忽視。顧名思義，電子簽章的功能與傳統的簽章是一樣的，只不過是以數位資料的形式出現。傳統醫療行為是以書面文件、簽名、蓋章來確定相關權利與義務，在網路環境中，必依賴電子文件及電子簽章作為通信及交易基礎，唯現有法令並未明確規範電子文件及電子簽章法律地位。

電子簽章的產生與資料的加密是反向的運作。舉例來說，如果 A 要傳送一份文件給 B，A 以自己的私密鑰匙對文件進行運算，即可產生電子簽章。然後 A 將文件與簽章同時傳給 B，B 利用 A 的公開鑰匙對 A 的電子簽章進行運算，將結果與傳送的文件進行比對，如果兩者相同，就表示該文件確實是由 A 所簽發，因為除了 A 之外，沒有人擁有 A 的私密鑰匙。

電子簽章是由文件的數位指紋產生，所以電子簽章也可用於提供資料真確性的服務。除此之外，透過電子簽章也可以檢驗資料的來源，而提供不可否認性的服務。也就是說，如果一份資料上的電子簽章經驗證為某人的私密鑰匙所產生的，就表示該文件資料由某人發出，而

他無法否認。如何防止其間流通的資料遭到非法的竊取、竄改，以及如何確認資料的來源，除了仰賴密碼學的技術外，相關法規的制訂也很重要。日前行政院通過了「電子簽章法草案」，便是要規範電子簽章的相關作業，確立電子簽章的法律效力，以做為電子商務與電子資料交換的作業依據，並以「技術中立」、「契約自由」及「市場導向」三大原則為基礎。

技術中立原則是指，任何可確保資料在傳輸或儲存過程中的完整性及鑑別使用身份的技術，皆可用來製作電子簽章，不特別指定某類型加密技術，以免阻礙其他技術應用發展。該立法原則是採聯合國及歐盟等國際組織倡議的「電子簽章」(Electronic Signature)為立法基礎，而不以「數位簽章」(Digital Signature)為限，以因應今後諸如生物科技等電子鑑別技術的創新發展。任何利用電子技術製作的電子簽章及電子文件，只要功能與書面文件及簽名、蓋章相當，只要確保該項電腦科技可以證實身份的認證，皆可使用。

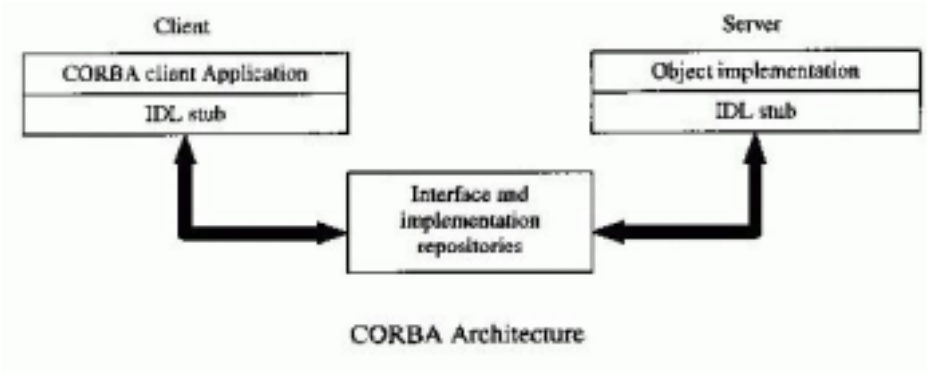
至於契約自由原則，是對民間電子交易行為，在契約自由原則下，由交易雙方當事人自行定採行共同信守的安全技術、程序及方法作成的電子簽章或電子文件，作為雙方共同信賴及遵守的依據，並作為事後相關法律責任的基礎，政府不以公權力介入交易雙方的契約原則。政府不強制建立專有的認證制度，讓這些認證方式可以在市場並行不

悖，並以契約自由原則讓憑證機構與其使用者之間，可以契約方式規範雙方的權利及義務。

公用物件請求仲介者架構

OMG (Object Management Group) 是一個由會員贊助而成立的非營利組織，其目的在推廣物件導向的觀念及使用 並致力於加強軟體的可攜性、再利用性、以及互通性。該組織會員包括了廠商、學術單位及用戶。 目前全球大約有三百八十家重要廠商為其會員。

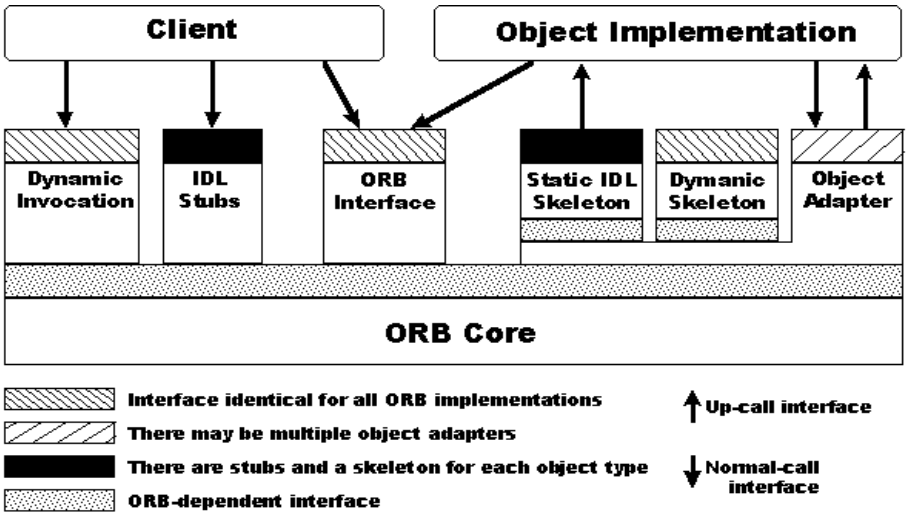
公用物件請求仲介者架構(CORBA:Common Object Request Broker Architecture) 為 OMG 在一九九一年十二月提出之物件導向分散式工作環境規格，一九九三年十二月一九九四年九月此規格多次被修訂，目前最新之版本為一九九五年七月之 CORBA 2.0 版，互通性及 C++對應之標準，均於新版之標準中更明確地被製訂。 CORBA 規格對一個物件請求仲介者之標準介面作了詳細的規定。根據 OMG 的定義，一個物件請求仲介者為一可提供分散式環境上各個物件透明化的請求服務與回應接收功能的應用程式建構工具。



CORBA Architecture

Object Request Broker(ORB) :物件請求仲介者，能夠讓物件透通的在分散式的環境下發出請求與接收回應，其為一種以分散式物件建立應用系統，以及讓系統在同質或異質的環境下相互協同運作的基礎設施。CORBA 將物件管理架構中的 ORB 機制各單元定義如下:

1. Object Services :物件服務是用來實作或使用物件的一組物件服務項目集合，每一個服務都是一個物件或是介面，這些服務備用來建構一個分散式應用系統，而且通常來說物件服務不侷限在每一特定的應用領域。
2. Common Facilities :讓許多應用系統使用的集合，但是沒有物件服務那麼的基礎。
3. Application Object :應用系統物件為可以控制應用程式介面的產品，應用系統物件由應用系統開發廠商所規範，不受 OMG 的標準化所規範。在實際應用時由於應用系統平台與開發所使用的程式語言各有其特性，所以一般都是以定義介面的形式來實作。也就是 IDL 語法。



ORB Core Component

第一個被 OMG 所認定的規格是 CORBA 規格，它詳細的制定了 OMA(Object Management Architecture)元件的介面及特徵。目前最後的修改版本是 OMG 於 1995 年中期提出的 CORBA 2.0 。

ORB(Object Request Broker)是 CORBA 的核心(Core)。ORB 負責傳送要求(request)到物件並傳回物件之回應到提出該要求的客戶端(client)，而此一客戶端所欲要求服務的物件稱之為目標物件(target object)。這種讓客戶端與物件溝通更輕鬆的透通性便是 ORB 的主要特色。一般來說，ORB 隱藏了下列幾點：

- 物件位置—客戶端並不知道目標物件位於何處，它可能位於網路上另一部機器上的某行程(*process*)，或是同機器中的不同行程，亦有可能是在同一行程中。
- 物件實作—客戶端並不知道目標物件是如何實作，像是以何種語

言，何種作業系統在那種硬體上實作等。

- 物件執行狀態—當客戶端要對目標物件要發出一個要求時，客戶端並不需要知道物件目前是否已啟動及準備接受要求，如目標物件尚未啟動ORB會去啟動它。
- 物件通訊機制—客戶端並不知道ORB 所採用的通訊機制。這些ORB 特色允許應用程式開發者將注意力放在應用程式領域主題，而較少關注在低階的分散式系統程式設計主題。欲產生一個要求客戶端必須利用物件參考*(object reference*)，當一個CORBA 物件建立時會同時產生一物件參考並指派給它。當物件被一個客戶端使用時，只要物件存在物件參考便會一直參考到該物件。客戶端可以由下列幾種方法得到物件參考：
- *Object Creation* —客戶端為了得到一個物件參考可以建立一個新的物件，透過叫用*(invoke*)建立要求*(creation request*)來啟動一物件，這類物件叫作工廠物件*(factory objects*)，一個建立要求會傳回一個最近建立的物件參考到客戶端。
- *Directory Service* —為了得到物件參考，客戶端可以叫用某些種類

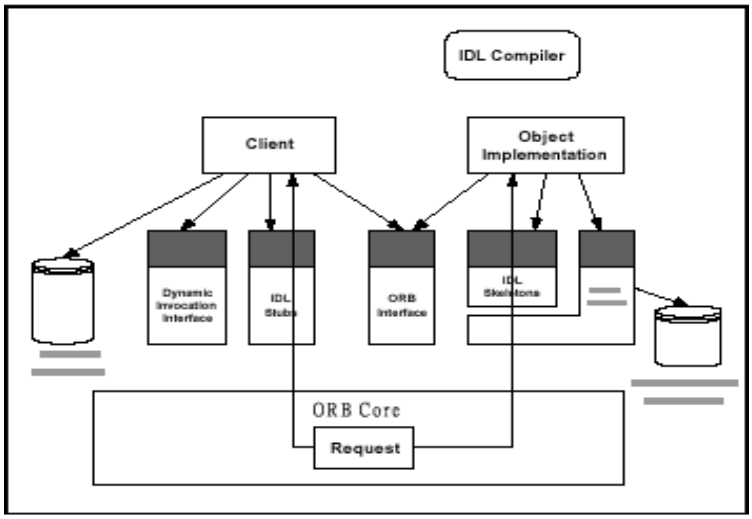
的服務如*Naming Service* 及*Trader Service* ，允許客戶端根據物件名稱或性質得到物件參考。不像工廠物件，那些服務不會建立一個新物件，它們為已存在的物件儲存物件參考和聯結資訊。

. *Convert to String and Back* —一個應用程式可以要求ORB 傳回一個由物件參考轉成的字串，且這個字串可以儲存在檔案或資料庫，稍後這個字串可以從永久的儲存體取出並被ORB 轉回物件參考。

在客戶端對一個物件作出要求前必須知道物件所支援的操作(operation)型態(type)，這些都在物件的介面裏詳細規定。OMG 為物件的介面定義了一套介面定義語言 Interface Definition Language(OMG IDL)。OMG IDL 的一個重要特性是“與語言無關”(language independent)，因為 OMG IDL 是純宣告(declarative)語言而不是可程式語言，它強迫介面的定義從物件實作中抽離。

OMG IDL 提供了類似一些程式語言的資料型態集合，它們用來指定參數型態和操作傳回的型態，此外，OMG IDL 亦提供例外處理(exceptions)的定義。最後，OMG IDL 有一個重要的特性就是它們可以從一個以上的介面繼承而來。當每一個 CORBA-based 的應用程式在執行時皆需取用 OMG IDL 型態系統，CORBA 的介面儲存庫(Interface

Repository, IR)允許 OMG IDL 型態系統在執行時期(runtime)被可程式的存取。除了產生可程式的語言型態，OMG IDL 語言編譯器及轉譯器亦會產生客戶端的 stubs 和伺服端的 skeletons 。Stub 是在客戶端有效的建立及發出要求的一種機制，而 skeleton 是在 CORBA 物件實作中接受要求的機制，透過 stubs 及 skeletons 的要求分派(Dispatch)通常叫做靜態叫用(static invocation)。CORBA 的語言對映(Language Mapping)能力會對映操作叫用到等效的程式語言函式呼叫(function call)。Stub 直接在客戶端的 ORB 做包裝(marshal)要求的工作，換言之，就是幫助要求從程式語言表示法轉成適合在網路上轉輸的型式。一旦要求抵達目標物件，ORB 及 skeleton 合作拆封(unmarshal)要求，將它從轉輸型式轉成程式語言型式並分派到物件。一旦物件完成要求，若有回應則順著原來的路線送回。



CORBA Request Module

除了籍由靜態叫用，CORBA 還支援動態叫用(Dynamic Invocation)的方法，並提供兩種動態叫用介面：

. *Dynamic Invocation Interface (DI*I) —使用 DII ，客戶端應用程式可以叫用沒有編譯時期的物件介面庫資訊的物件, DII 也可用來做程式間的互動。

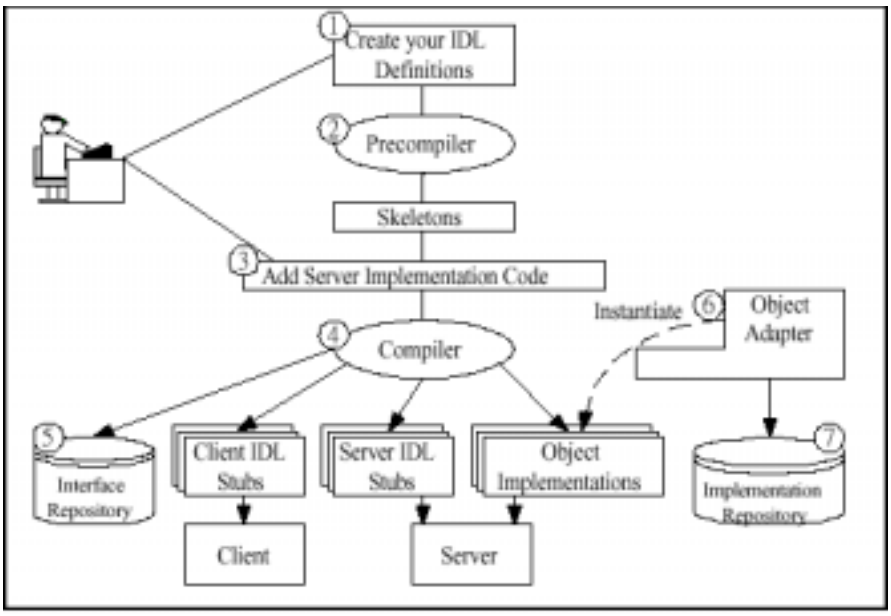
. *Dynamic Skeleton Interface (DS*I) — DII 類比到伺服端就是 DSI ，只是 DII 允許客戶端不用存取靜態 stub 便可叫用要求, 而 DSI 允許伺服端不用將叫用靜態地編譯進程式。至於 *Object Adapte*r(OA)的功能就像把 CORBA 物件實作與 ORB 黏起來。OA 是將另一個物件介面插入*(adap*t)到呼叫者所期望介面的物件，目前 CORBA 僅提供一個 *Basic Object Adapte*r(BOA)。整個 CORBA 架構圖下。

採用靜態方法叫用*(Static Method Invocatio*n)方式的CORBA

程式設計流程如後，其步驟分述如下：

1. 使用IDL 定義你的物件類別：這個IDL 用來告訴潛在客戶端有哪些操作可利用及如何叫用。IDL定義了物件型態、屬性、它們所要顯露的方法及方法參數。
2. 執行CORBA IDL 程式語言前置編譯器(precompiler) ：依你想要實作的程式語言選擇對應的CORBA IDL 程式語言前置編譯器。在編譯之後會產生用來支援實作伺服類別(server classes)的skeleton 及客戶端的IDL stub 。
3. 填入物件實作碼至skeleton：替skeleton 加入方法的實作碼。
4. 編譯程式：將客戶端的stub 與客戶端原始碼一起編譯成客戶端應用程式，而伺服端的stub(即skeleton)與物件實作碼一起編譯成伺服物件程式。
5. 繫結(bind)類別定義到介面儲存器(Interface Repository)：使用工具程式繫結IDL 資訊到IR ，如此程式才可在執行時期(run time)存取IR。
6. 在伺服器上例元化(instantiate)一個物件：在啟動時期，OA 會建立伺服物件例元(instance)以服務遠端客戶的方法叫用。
7. 註冊執行時期物件到實作儲存器(Implementation Repository)：OA 會記錄任何伺服器上物件例元的物件參考及

型態到實作儲存器。



CORBA 程式設計流程

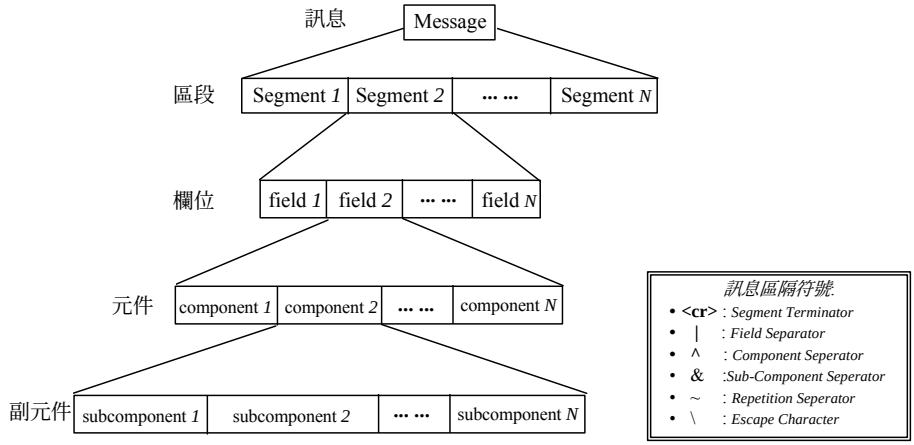
HL7

HL7 發展主要的之目的為經由對醫療保健應用系統間資料交換標準之定義，減少界面間程式製作及維護之負擔，以簡化應用系統之間整合的複雜度，降低系統開發的成本。

在 HL7 中，應用系統之間是透過「訊息」來交換資料。訊息交換分成兩種類型，一種是非請求性更新(Unsolicited Update)，另一種是查詢 (Query)。非請求性更新對應到真實世界一個觸發事件 (Trigger Event) (如病患住院)，由一個系統傳送到另一個系統，目的是通知對方本身狀態已發生改變。當接收端接收該訊息後，不需傳回任何資料，只須傳回一個認可 (Acknowledgment) 的訊息，告知發送端該訊息已經收到。

若一個應用系統必須向另一個系統要求資訊時 (如某一個應用系統可能傳送一個包含病患號碼的訊息給住院系統，以請求獲得該病患相關的資訊)，所使用的訊息交換稱為「查詢」。在這類型訊息交換中，接收端收到該查詢的訊息之後，必須回應相關的資訊，而不是傳回「認可」訊息。

HL7 標準所定義的這兩類訊息交換：非請求性更新-認可訊息，查詢-回應訊息，可適用於主從架構 (Client-Server) 的操作模式。在 HL7 中，每個訊息屬於一種訊息型態 (Message Type)，以區分其種類，如 ADT (Admission, Discharge, and Transfer) 訊息型態是運用於住院出院與轉診應用中。



HL7 Message Structure

每個訊息由數個區段(Segment)所組成，它是邏輯上相關資料的集合，例如 PID(Patient IDentification) 區段包含了所有病患的基本資料，OBR(OBservation Request) 區段則包含了病患觀察檢驗的資料，相同的

區段可以使用在不同的訊息中。

每個區段包含許多欄位 (Field)，例如 PID 的區段中包含了病患的姓名、住址和聯絡電話，每個欄位又由許多的元件(Component) 組合而成，例如病患的姓名欄位包含了姓氏、名稱和職銜等元件，元件中可以再包含次元件 (Sub-component)。

HL7 標準之中另外定義了編碼規則 (Encoding Rule)，說明了不同的資料型態如何在一個欄位中編碼，以及何時個別的欄位可能重複等，因此，發送端和接收端可以正確的解釋訊息的意義。

HL7 標準文件，涵蓋了下列醫療保健應用系統間資料的交換：

(1) 患者住院出院轉診

在 HL7 中，住院、出院及轉診相關的訊息，稱為 ADT 交易集。此交易集提供病患相關的看診以及統計資訊之傳輸。因為大部份子系統都需要病患相關資訊，所以 ADT 交易集經常被使用。

(2) 醫囑登錄，包括藥房 (Pharmacy)、檢驗 (Laboratory) 及用膳 (Diet)、供應品 (Supply)。

在 HL7 標準中，醫囑或醫囑相關資訊之傳輸是由醫囑登錄交易集所提供，醫囑登錄交易集包括:臨床觀察及診斷性檢查 (Clinical Observation And Diagnostic Studies) (例如：護理服務之臨床觀察、檢驗室之檢驗等)、治療 (例如：藥物治療、化學

治療等)、飲食 (例如:配膳房之飲食供應等)、供應物 (Supplies) (例如:內務管理之衣物床單供應、中央供應站之供應等) 及其它醫囑。發出醫囑之個人或實體稱為發出者 (Placer),實現醫囑之個人或實體則稱為執行者 (Filler)。

(3) 患者會計交易,包括收費、付款、病患基本資料、保險及其他患者帳務資訊。

在 HL7 中,與病患財務、會計相關之訊息總稱為病患會計交易集。此交易集提供了收費、付款、調整、病患統計分析、保險、其他相關的病患帳務與應收帳款資訊的登錄與操作。在不同的應用系統間,財務交易可以用批次或連線的方式來傳送,透過編碼規則,可以將多重交易集合在一起,用檔案傳輸媒體或程式來傳送。

(4) 臨床觀察結果,包括檢驗結果、影像判讀結果、心電圖肺功能研判結果、病患狀況衡量等。

HL7 定義了醫院應用系統之間,傳送病患導向、結構化臨床資料之交易集。這些交易集可用來傳輸任何臨床觀察之結果,例如:臨床檢驗結果、影像判讀結果 (不含影像)、心電圖肺功能研判結果、體檢結果等。這些交易傳送的資訊包含文字、數字與類別數值,但是這些訊息並不包含影像與追蹤資料,而觀察

結果可以是許多資料型態中的一種，其主要的資料型態為文字、數字與代碼。

(5) 定義開放架構之醫療保健環境中應用系統所共同參考之檔案，包括 doctor 主檔、系統用戶主檔、位置設備主檔、患者狀態種類主檔、收費檔等，其主要的目的在於使不同的系統間所共同參考的檔案資料，能夠維持同步的更動。

XML

網際網路的方便讓使用者能簡易且經濟地分享文件或資訊，然而隨著網路的廣泛興起，網路上的文件越來越龐大且複雜，資訊提供者漸漸體驗到這個媒介並不提供延伸性、結構化以及資料檢查等大型商業出版所需求的特性。網路上大部分的文件都是以 HTML 撰寫，但是 HTML 卻不具有下列特性：

1. Extensibility：HTML 不提供使用者自訂標籤。
2. Structure：HTML 不支援可敘述資料庫 schema 或是物件導向階層的結構。
3. Validation：HTML 不提供 application 能驗證文件結構正確與否的方式。

上述的缺憾在 SGML 都有詳盡的支援，但是 SGML 卻包含了許多

Web 非必須的特性。SGML 乃是一九八六年國際標準組織(International Standards Organization, ISO)公佈的「標準通用標示語言」(Standard Generalized Markup Language, SGML)。SGML 太過於覆雜，不適用 Web 上，因此 1998 年 2 月，美國 W3C 組織正式公佈 SGML 的精簡版，即 XML 的 Recommendation 1.0 版語法標準。XML 掌握了 SGML 其延展性、文件自我描述特性、以及其強大的文件結構化功能，但 XML 卻摒除了 SGML 過於龐大複雜以及不易普及化的缺點。字面上來看 XML 是一種標示語言，但嚴格來說它和 SGML 一樣是一種「元語言」(meta-language)。換言之，XML 是一種用來定義其它語言的語法系統。這正是 XML 功能強大的主因。

目前全球以 XML 為基礎之垂直產業電子化標準計有 RosettaNet(資訊與電子)、AIA(航太業)、SWIFT(金融業)、ACORD(保險業)、OTA(旅遊業)、以及 HL7(醫藥業)等。XML 將快速成為網路上文件交換的標準。不同於 HTML 僅僅對文件如何呈現加以描述；XML 可顯示它所描述的資料“是什麼”，而不只是如何表示這資料。它可促進各專業機構、不同產業界、學術界和特定應用領域發展各自標準的文件和訊息，以利資訊的交換、處理和相關衍生性資料加值服務。XML 相較於 HTML 有下列幾項優點：

1. 資訊提供者可自行定義自己的標籤(tag)以及屬性(attribute)名稱。

2. 文件結構可巢狀化，不論複雜到何種程度。

3. 任何的 XML 文件可額外提供有關文件文法的敘述，以供軟體對於其文件結構正確性的檢查。

XML 文件和訊息的主要特色在於它是結構以及資訊內容導向。結構化文件和訊息編碼方法的主要精神在於它可供其它電子資料傳遞、文件出版系統、電腦輔助設計或製造、資料庫管理等系統，在處理重複和共享的資料時，能有效提升其效率和效能，節制資訊系統的開發建置和管理營運成本。這種方法將資訊內容、結構和格式等不相同的文件要素予以區分。它保存了文件的資料和結構(有助於原始資料的回溯)，可是卻不指出文件的呈現格式，如是格式的解析應在資料最後傳遞時，才依據用戶需求進行最佳化之處理。XML 技術本質上的優勢和特色，使商務資訊流電子化產生根本上的改變，並在應用上提供更多維的可能性。

伴隨著 XML 的標準，各種相關技術組成了一個家族。Xlink 將超連結(hyperlink)以及文件間的關係的描述加入 XML 檔案中。XPointer &XFragement 是一種指向文件某一部份的指標描述。XSL 能將 XML 文件轉換成其他格式以供列印及其他處理。CSS 可供描述 XML 呈現的外觀。DOM 是一群標準的函式庫供其他程式語言來呼叫。

一個完整的 XML 檔案包含了一個 declaration(or prolog)用來指定

這是一個 XML 檔案，此標籤類似於 HTML 的檔案標頭 <html>，但 XML 利用<?xml …?>作為宣告，並可擁有下列三個 attributes ：

1. version 指出所使用 XML 標註語言的版本, 這個屬性必須存在。
2. encoding 指出此份資料所使用的編碼, 此屬性可有可無。(一般預設值是 compressed Unicode: UTF-8) 。
3. standalone 告知此份文件是否參考外部連結的「資料型態規格(data type specification)」(也就是此文件是否包含或是外部參考一份 Document Type Definition, DTD)，如果未參考那麼 standalone=“yes”。

以下乃是一個完整的 XML 檔案標頭範例：

<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”yes”?>

在 XML 中屬性與屬性間以空白符號來區隔，不像 HTML 檔案可利用逗號(,)來區隔，屬性值必須介於雙引號中(” … ”)，而且 XML 是 case-sensitive 的，不像 HTML 是 case-insensitive 。一份 XML 文件可包含此文件資料型態的定義 Document Type Definition, DTD(如同於 SGML 所定義般)DTD 的宣告有點類似一個指標指到此 DTD 檔案存在的位置。

DTD 的英文全名是 Document Type Definition。中文的字面意義

為「文件格式定義」。其作用在於定義和規範特定 XML 文件的內容架構。它通常是一個含有某一種特定格式正式定義的檔案文件。換句話說，透過 DTD 檔案的描述，XML 文件檔案的格式結構就可以成形了。以商業交易為例，在交易的流程中常會需要用到訂單、訂單變更要求、未結案訂單報告、發貨單、收貨驗收單、請款對帳單、付款明細表等商業表單，為降低交易雙方的商業糾紛並提高交易的準確性和使用效率。這些表單的內容和結構必須一致且嚴僅。在 XML 的領域中，DTD 的作用便是在定義 XML 表單應如何撰寫安排，它就類似於文章的寫作文法和語意表達方式。

DTD 是文件的正式定義，它指定了可在此文件中使用的 element tag ，以及伴隨這些 tag 的 attribute ，以及這些 element 之間所形成的結構。舉例來說，一份 DTD 可能指定<Company>元素只能在<Author>元素之中。DTD 可讓 XML parser 來檢查這份文件結構、型態是否正確。XML 文件不一定要有 DTD(也就是所謂的 DTDless)，但必須一定是 well-formed ，也就是必須遵守一些簡單的規則，以確保文件在 parse 時能正確，舉例來說，一個 element 必須要有起始標籤(start tag)與結束標籤(end tag)而且要正確的巢狀，不像 HTML 有類似
、<P>這類型的單獨標籤存在。如果 XML parser 在處理 XML 文件，發生錯誤時，必須提供使用者錯誤訊息，不同於 HTML 會將錯誤的部分一併處理。

讀取 XML 的方法有二， 一是 DOM， 一是 SAX， 都是 W3C 推薦的， 不過不一定是 W3C 做出來的. DOM 是 W3C 做出來的， 所以其 API 在 org.x3c.dom 之下. 而 SAX 是 xml-dev mailing list 做出來的， 所以在 org.xml.sax 之下， 這兩個都是標準， Apache Xerces 與 Sun ProjectX 都有支援.

DOM 是 Document Object Model 的縮寫. 其概念是將 XML 轉成一般資料結構上的 Tree 來使用. 因為 XML 在結構上有關連， 很適合建構成一個 Tree， 而這個 Tree 就是 DOM. 換句話說， 一旦 XML 被 parser 讀取並轉換成 DOM 之後，便可以在這個 Tree 上任意遊走， 取得所需的資料. 這樣一個轉換由 parser 來處理， 完全不用動手， 我們要處理的是在 Tree 上走動並讀取資料的問題. 建構一個 Tree 不難， 但是如果 XML 太大的話， 是相當花時間並占空間的， 因此如果只是要取得 XML 中少許的資料， 這是相當不划算的。

SAX 稱為 Simple API for XML， 顧名思意， 就是想要用簡單的方法來處理 XML. SAX 的概念是， 一路從頭到尾讀取 XML， 讀到什麼就輸出什麼來， 讀完了也就輸出完了， 完全不著痕跡， 想要再用一次 XML 的資料， 那就再讀一次吧. 從這裡可以發現， 利用 SAX 可以建一個 DOM 出來， Apache Xerces 與 Sun ProjectX 都是採用這種做法. 因此 SAX 在時間及空間上的使用較 DOM 小得多， 一來是

不用建一個 Tree， 二來是讀出的資料不用記錄成 Tree， 但是要反覆利用 XML 的資料時， SAX 反而不方便就是了， 因為用一次要讀一次. SAX 針對讀到不同的內容， 會去呼叫相對應的函式， 因此我們要實作 (implement) 這些函式， 來達成我們的目的， 這些函數稱為 Event Handler. 要注意這些 Event Handler 與 AWT/Swing 用的 Event 是不一樣的， 只是個函數而已。當 SAX 讀到一個 element， 例如 <table> 時， 就會呼叫 startElement() 函數， 並把 element 的名稱及屬性傳入， 當 SAX 讀到一個 </table> 時， 就會呼叫 endElement()，並傳入 element 的名稱. 因此我們就在實做這些函數， 來處理不同 element 的需要。如果 XML 的文件中有機器所產生的資料，利用 SAX 較好；如果 XML 文件中含有人類可閱讀的資料的話，利用 DOM 較方便。

Java 不受限於平台環境的特性，改變了程式語言的演進史，XML 則更進一步將資料交換的問題解決了，讓 Java 可以更盡情自在的發揮。Java 和 XML 有許多共同的特徵，而這些特徵是用在以網站為基本單位的企業應用軟體上，最重要的功能，例如：平台獨立、可擴充性、程式碼再使用、支授全球共同語言（Unicode）... 等等。

Java 和 XML 的組合，是 21 世紀軟體開發人員的聖杯。沒有任何程式語言比 Java 更具吸引力，而 XML 提供了絕佳的資料表達方

式，Java 和 XML 個別來看都是很有用的工具，而兩者結合起來的威力更是讓程式員興奮。

有了 Sun 所設計的 JAXP（全名是 Java API for XML Parsing），Java 版本的 XML API 終於開始正視到一個重要的事實：許多 Java 程式員只希望讓程式相容於 XML，但不想把 XML 的規格讀到滾瓜爛熟。JAXP 讓你可以利用簡單的方式來獲得一個 DOM Document 或 SAX 剖析器，因此程式員不用再為不同廠商的不同版本大傷腦筋。JAXP 也意圖讓你的程式只需小修改就可換成其它廠商的實作版本。這才是程式員習慣的方式。然而，JAXP 只是讓底下的實作可以交換，整個 JAXP 還是建構在 DOM 和 SAX 這兩個不太好用的 API 上。因為它支援舊版的 API（SAX 1.0 與 DOM Level 1），所以造成一些不方便的限制。

JDOM（Java Document Object Model）是一個新的 API，用來提供 Java 程式處理 XML 的能力。更重要的是：它還是第一個明確地針對 Java 程式員所設計用來處理 XML 的 API，這也意味著它符合一般 Java 程式員的期待、習慣、和希望。此 API 不要求程式員一板一演地操作樹狀結構，而是，比方說，直接的操作某 Element。比較起來，傳統的方式可就麻煩了：首先你得逛到樹中的 Element 節點的子節點，判斷其是否為文字節點，然後才能取得其值。

XSL (Extensible Stylesheet Language)，全名為延伸式樣規語言，早期也曾被稱為「Extensible Style Language」；XSL 係用來描述資料該如何被展示在使用者眼前的語言，XSL 為用來表現樣規(stylesheets)之語言，由兩個部分組成：

1. XSL Transformations (XSLT)：轉換 XML 文件之語言
2. XSL Formatting Objects (FO)：XSL 格式化物件，用來述明格式化語意之 XML 字彙。

XSL 樣規以描述文件如何藉由格式化字彙(即 XSL 格式化物件，XSL FO)轉換成 XML 文件，進而詳述 XML 文件呈現方式。XSL 提供程式開發者一個工具，敘述 XML 頁面中哪一欄位的資料應該被顯示、其顯示的確切位置，以及如何顯示的方式。XSL 也可重複利用在多個 XML 文件上。以呈現到一般網頁為例，藉著 XSL 的使用，使用者可以控制原始 XML 文件中的資料呈現在瀏覽器上的外觀，例如字的大小與樣式、擺放的位置等。

XML 和 EDI 之比較

富含特色的商務互動行為必定包含了大量的資訊，傳統上在論及電子商務資訊流時，言必出「電子資料交換」(Electronic Data Interchange， EDI)。EDI 是一種快速可靠的文件資料交換方式，它主要被用於不同公司間不同電腦系統的商業文件交換，特別是上下游工

廠(供應鏈)或是交易企業間的資料交換。它藉由電腦的資料處理及通訊功能，傳達一標準格式的電子資料檔案，將交易往來的商業文件，如訂單、訂單回覆、請款對帳單或付款明細表等，透過相關轉換機制和系統，傳達至對方的資料庫或 MIS 系統，以便進一步處理。早期的 EDI 屬於專屬封閉的系統，建置成本高，因此造成一般中小企業的進入障礙。此外，早期的 EDI 系統僅能改善和處理片段的作業流程，但網際網路世代的來臨，卻改變和衝擊傳統的 EDI 生態。相較於 EDI，XML 的主要優勢在於：

1. 只要資料結構、語意和資料值能夠統一，XML 的文件對應用程式來說具有自我定義（self-defining）的特性，亦即 XML 文件不必像 EDI 訊息一樣需要預先設定的特殊格式和結構。
2. XML 文件內容的標籤元素基本上與通訊協定獨立。因此，XML 文件特別適合在網際網路和全球資訊網的環境中流通傳輸。
3. 相較於 EDI，XML 在編輯器、中介軟體以及應用工具上擁有更多的選擇。這些差異性，將使 XML 的標準化和導入歷程不會像 EDI 走得那樣艱辛。

XML 和 HTML 之比較

現今的 HTML-based 全球資訊網是、呈現導向、(presentation-oriented)，換句話說，HTML 語法是用來指定文件在瀏

覽器上的呈現方式，這意味了人類可輕易地瞭解 HTML 的檔案內容，但電腦軟體本身卻無法了解 HTML 檔案資料的內容和意義為何。雖然 HTML 的簡單輕便，助長了全球資訊網的迅速普及，但隨著全球資訊網平台上之多媒體及編排上的多樣化殷切需求，以及強調效率和精準的電子商務的興起，HTML 語法已逐漸顯露其捉襟見肘的窘態。雖然許多程式設計人員利用自定的 HTML 標籤以及專屬的軟體來擷取網頁中的資訊內容，但此法卻無法滿足普及化的需求，且造成各行其事的紛亂局面。若資訊本身未經過語意化和結構化來表達，許多的軟體以及搜尋引擎將無法更有效地善用這些資訊。在 XML 的架構下，結構化的資料以及具有意義的資料標籤，將使電腦和軟體得以理解和利用網頁或文件和訊息內的資訊，再透過代理程式以及其他自動化程序，電子商務資訊流的自動化將可有效地提升，並從本質上轉變電子商務的環境。

系統設計

醫療資訊交換系統在規劃與設計前必須先被清楚的描述解決方案的範圍。因此本研究提出了醫療資訊交換系統模式 RACE 的概念來描述醫療資訊交換系統。RACE 模式是以層次的方式來表現，其次序由上往下分別是權力關係(Relation)、支援活動(Activity)、流程控制

(Control)、訊息交換(Exchange)層。RACE 描述了醫療組織間的權力關係來用以規範出跨醫療組織間的支援活動類型、流程控制的嚴謹程度，而這些都在訊息交換的基礎上。

	RACE層次	RACE層次類型
4	Relation 組織間關係	平行單位關係
		階層(含集權)單位關係
3	Activeity 組織間活動	One To Many(一對多活動)
		Many To Many(多對多活動)
		Chat(一對一串接活動)
2	Control 流程控制	Procedure(商業程序級控制)
		Program(程式級控制)
		Data(資料共享級控制)
1	Exchange 訊息交換	Message Exchange(訊息交換、通訊協定)
		Access Control(存取控制)

醫療資訊交換系統模式

權力關係層: 權力關係層是界定醫療組織間的管理經營權關係，例如平行的診所與藥局、階層的衛生局與衛生所。在設計系統的規範上當然不相同；其類型有階層(集權)式與平行式兩種，可使用 Extranet 或價值鏈資訊系統來支援該層次。

支援活動層: 支援活動層的目的在於描述醫療組織間活動的類型，有 One to Many、Many to Many、Chat，而此層次資訊技術為支援”

溝通”的技術，例如 CALS、EDI、Net Meeting、Groupware 等.將訊息內容變更就可以用在不同的活動，像是訂購衛材、釋出處方籤。

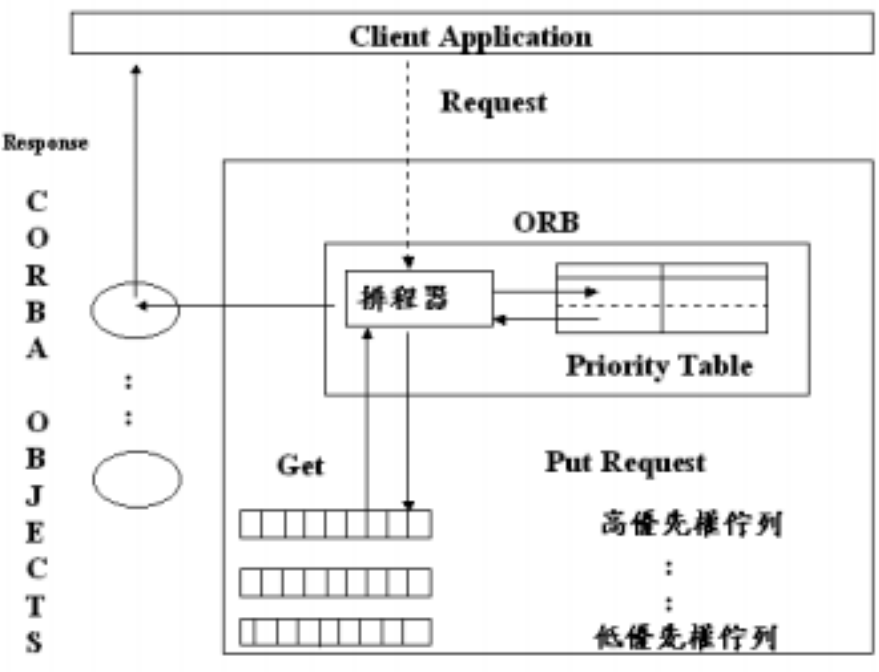
流程控制層: 流程控制層將作業活動分為 Data、Program、Procedure。依活動嚴謹度不同而使用不同的資訊技術，例如轉診後轉出的醫院要回覆單以了解病患的後續情形，但是醫藥分業的診所釋出處方籤後就不需要了解處方籤調劑的情形。Data 控制只要訊息送達即可，以 DDBMS 為支援技術。Program 控制需將訊息作即時的處理，以 CGI 為支援技術。Procedure 控制除了即時外，發訊息的組織還要得到處理結果，並且決定下一步流程該如何進行，所以支援的資訊技術為 Transaction Server、DCOM、RPC 等。

訊息交換層: 訊息交換層就是負責醫療組織間的訊息交換，進行跨組織活動時，勢必要有一個醫療組織先發出訊息，其他的醫療組織才能進行下一步流程。其功能有存取控制與訊息交換，使用 FTP、Email、Socket Application、WWW 技術。

採用共通物件仲介者架構建立醫藥分業策略性資訊系統，醫學資訊之交換將被轉換成網際網路上的服務，每一個細部活動都會有一個對應的服務，而每個組織則在網際網路上公開自己提供的物件給需要與該組織進行互動的其他組織呼叫。而組織間訊息的交換目前有許多成熟的標準如 ODBC、JDBC、DICOM、HL7 等..。由於醫療活動的特殊

性，擴充公用物件請求仲介者架構在原有的公用物件請求仲介者架構上另外提供醫療活動優先權與醫療活動流程控制功能。醫學資訊交換之問題，可以利用擴充式公用物件請求仲介者架構與標準的資料交換協定解決活動的不確定性，要優先處理的醫療活動事件，將獲得高優先權即時被處理。由於擴充式公用物件請求仲介者架構是在網際網路上讓多台電腦協同執行，所以可以分散使用有關醫學資訊交換中心內部資訊系統的負荷，並且可以利用分散式服務與物件來減低醫療組織單位之間的相互依賴程度。

為了使擴充式共通物件仲介者架構以應用在跨組織醫療系統上，在原有的公用物件請求仲介者架構上加上優先權排程與流程控制的服務。CORBA 架構中的物件在執行方法(Method)前，必須先經過優先權物件排程，我們採用了多重佇列排程法來作優先權排程，當優先權物件被要求服務時，會依據呼叫物件的身分與要進行動作查表決定排進哪一個佇列當中，接下來排程器會會依據多重佇列排程法依序執行程序，物件在接收回應後則執行欲執行的方法。流程控制功能分成兩個部分，一為當活動流程由 A 轉移至 B 狀態時，查詢轉移狀態要實行的物件名稱並執行之。二為物件執行時會查詢活動目前屬性或狀態，而採取對應的處理。



Inhanced CORBA Architecture

本研究將以 JAVA Platform 來實作，並以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件。並以 IDL 來描述醫學資訊之交換格式，利用 JAVA IDL 將其轉成 JAVA，JAVA ORB 作為 Broker，並利用醫學資訊交換中心之共用安全物件來交換資訊，共用安全物件包括下去功能:

1. Authentication(認證)

在 Secure ORB 裡，每一個物件實體必須先建立一個自己的身份證，然後才能對其它的物件做存取的動作。我們在 CORBA 裡稱這個物件叫做 principal (主角)，它得先和 ORB 拿到 credential(資格證明書)，credential 可能不只証明你是屬於哪一個 role(角色)，可能是很多 role，一個 principal 只須要拿一次 credential，便可以對一個或多個其

它物件作存取。 Authentication 的結果是拿到一個獨一無二的 authenticated ID，這是其它物件都不能加以改變的，除了 authentication Server。認證的程序如下

1.1 Authenticate the Principal：要有一個 User Sponsor (使用者的保證人)，它是一段程式碼負責和 User 做溝通來得到 ID 和 Password，然後呼叫 authenticate 來將 ID 和 Password 傳入 Principal Authenticator。

1.2 Create the Credential object：Principal Authenticator 會傳回一個 Credential (資格證明書)物件，這個物件可以就是算是門票了，裡面說明 Principal 所代表的角色和擁有的權限。

1.3 Set the credential of the execution environment : User Sponser(使用者的保證人)將 Credential 傳入 Current 物件。Current 物件是在 Transaction Service 的 pseudo object，用來表示 Client / Server 間互動的內容。

1.4 The client invoke a secure method on the server：所有的 Security 的資訊是經由 Current 這個 pseudo object 來做交流。

1.5 The server executes the secure method：Server 用 get_attributes method 從 Current 物件得到 client 的屬性，知道 client 的 ID 和權限後就可以決定 client 是否可以存取。

2. Privilege Delegation(權限代表)

在分散式物件的系統環境下，物件會 call 新的物件，所以該送哪一個 credential 給下一個 object 就是 privilege Delegation 要討論的。假設有三個 object，分別是 Client----Intermediate----Target，Client 送 credential 給 Intermediate，而 Intermediate 送 credential 給 Target 的選擇有以下三種

2.1 No delegation : Intermediate object 不代表 Client，所以送 Intermediate object 的 credential 給 Target Object。

2.2 Simple delegation：Intermediate object 代表 Client，所以送 Client object 的 credential 給 Target Object。

2.3 Composite delegation : Intermediate object 複合代表 Client 和自己，所以送 Intermediate object 和 client object 的 credentials 給 Target Object。

3. 存取決定(Access Decision)

CORBA 的 Access Model 是 based on 一個 trusted ORB Model。也就是說你必須要相信你的 ORB 會在 Server Resource 上強迫你執行特定的 Access Policy。Access Policy 將會決定 ORB 是否要呼叫所欲使用物件的 method。Access Decision 是根據呼叫者目前的權限。呼叫者的權限是定義在 Credentials 裡的，這些權限定義了：Access

Principal 的 Access identity、Security Clearance(安全許可)及一個以上的 Role，這些 Role 是有關於 Access Principal 的 Job Function。

4. Audit Trail(監聽)

就像 Unix 對 login 或 User 所做的特別動作有做記錄(log)一般，CORBA Security Service 也有設計相似的功能，就是 Audit Trail。Audit 是旁聽，Trail 是線索。Audit service 讓系統管理者可以監聽 ORB 的 event，主要的用意在防治侵入者。我們可以選擇想監聽的部份做監聽，這樣 log 的大小才不會爆炸式的成長。除此之外，Audit Service 也可以制定 policy 來方便的監聽要聽的部份。

5. Authorization/Access Control(授權/存取控制)

既然 Client 已經通過認證，Server 要決定的就是確定 Client 可以擁有哪些權限，Server 利用 ACLs 和 Capability Lists 來控制 User 的 Access。ACLs 可以存取 Server 上任何的資源。它包含了一長串的 Principal Names 和 Operation Types(Operation 只可以控制某些被允許操作的資源)。Client 也可以在同一個 ORB 上採取不同的 ACL Policy。有關 ACL Policy，ORB 上可以 implement 某些 Policy，當然你也可以自己在 Server 端 implement 其他的 Policy。

6. Non-Repudiation(使不能否認、拒絕支付)

ORB 提供證據說明某個動作確實有發生，這樣使 Client 不能辯駁

說 request 不是他發的或者是沒得到 data 或 invoke method ，這一點在電子商務界是很重要的。為了要提供 Non-Repudiation Service，ORB 要提供 electric equivalent of sealed envelope(電子密封信封)，來確保資料的安全性，使資料不會被改變或竊聽。還有 ORB 會要 Sender 提出 delivery 的證明，Receiver 也要提出 Sender identity 的證明。CORBA Security Service 定義的 non-repudiation framework 是根據 ISO non-repudiation model，有下列的 Service：

6.1 Evidence of message creation：Sender 送一個 proof-of-origin 證明，這個證明將會傳給 Receiver 當證據，如果有爭議就可以作證明。

6.2 Evidence of message receipt：Receiver 送一個 proof-of-receipt 證明，這個證明將會傳給 Sender 當證據，如果有爭議就可以作證明。

6.3 An action timestamp：是 non-repudiation 產生的證據，用來記錄動作或事件發生的日期和時間。

6.4 The evidence long-term storage facility：用來保存證據，如果發生爭論，adjudicator(法官)可以用這些 facility 來取回證據。

7. Non-Tampering and Encryption(加密)

所有的 Message 是根據所指定的 quality of protection(QOP)，

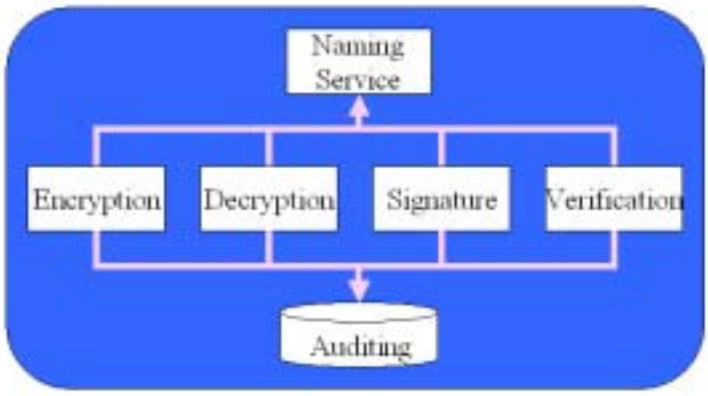
request 和 response 可能使用不同的保密方式。Encryption 可以使用許多工業的標準，例如 RSA's public/private key，OSF's Kerberos，Internet's SSL，NetWare 4.X，和 NIS+。而 ORB 並沒有定以任何 Cryptographic 的 interface，原因是因為要讓 ORB 在私下做 cryptographic 的動作，這些是不讓其他的物件看到的。ORB 可以提供 Encryption : 使兩個 principal 有安全的通信，並提供 Cryptographic checksums : 保證通信中傳輸的資料不被更改。。

8. Secure Domain(安全領域)

policy 就是物件間資料傳遞或處理的策略，而 policy 應用的範圍乃是以 Domain 為單位。Policy 是由(Administrator)管理者來管理控制，有 access control，authentication，privilege delegation，non-repudiation，和 auditability 等 policy。 一個 Secure Domain 可以有 Sub-Domain 來反映組織的分類，例如一個部門可以是一個公司裡的一個 Sub-domain。我們希望不同的 domains 能夠彼此可以使用對方的的一些權限，我們就可以把這些 domain 設為 federation domain，不過要注意的是在 federation 必須要能處理不同 domain 的 policy，例如必須能夠對不同 domain 的 role 能夠作 mapping。

本計劃具有 Secure Channel 之設計，Security Component 內含 Secret Key。跨越主機之 Security Component 之間傳遞參數可利用 Secret

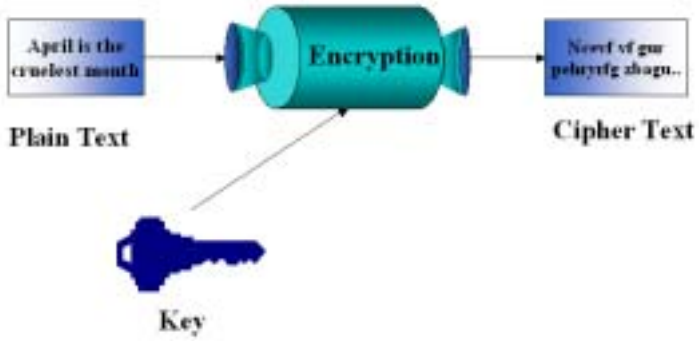
Key 加密，以免遭受竊聽。Security Component 之間宛如置於 Virtual Private Network 上執行，其架構如下：



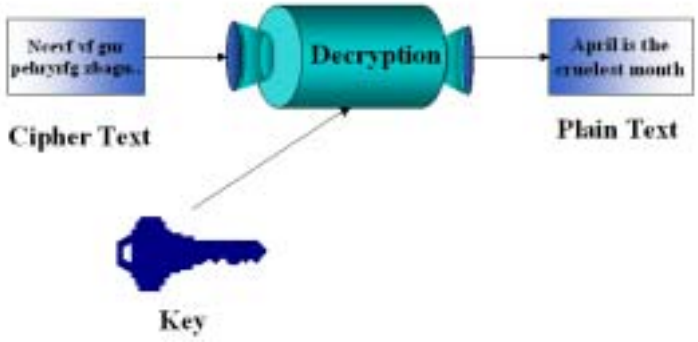
Security Component Architecture

系統模組

本計劃包括 Encryption Service、Decryption Service、Signature Service、Verification Service 及 Auditing Service，Auditing Service 對外部不開放，只提供其他 Service 使用。



Encryption Service



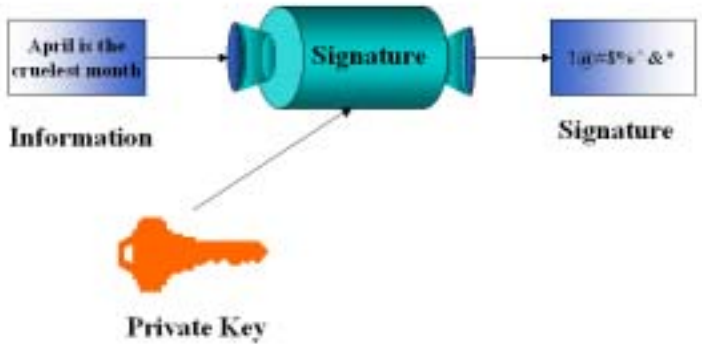
Decryption Service

以鑰匙為基礎的加密技術分為對稱性 (symmetric) 與非對稱性 (asymmetric) 兩種，主要的功能是在文件透過網路傳輸的過程中，利用軟體將傳輸文件編碼，收件者在接獲文件時，再以解密的技術將文件還原。對稱性與非對稱性主要的差異是對稱性保密系統採用的加密金鑰與解密金鑰都是相同的，在對稱性密碼系統中，加密與解密使用的鑰匙是相同的，雙方在通信前必須以安全的方式，互通加解密的運算方式，因此使用此系統必須對此金鑰嚴格保密，在非對稱性密碼系統中，加密與解密使用的鑰匙則是不同的。

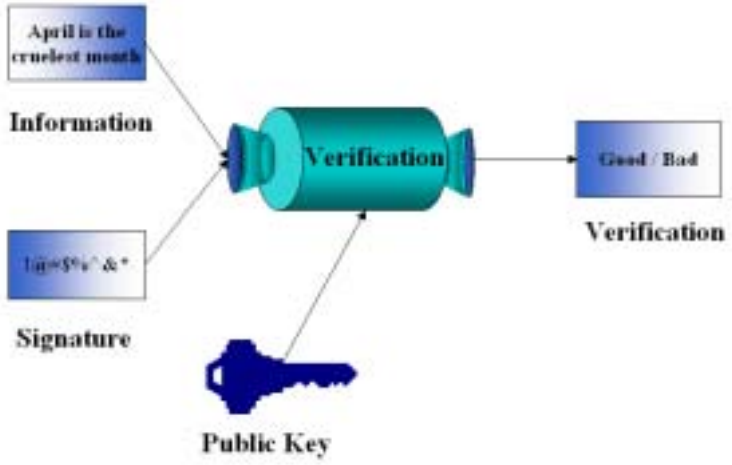
最常見的對稱性密碼系統是於 1976 年提出的 DES (Data Encryption Standard)，1977 年被美國政府採用為國家標準。對稱性密碼系統有個缺點，就是鑰匙的交換問題。發文者與收文者必須擁有相同的鑰匙，而此鑰匙不能為其他人知道。為了提升安全性，每次使用的鑰匙最好不同，而鑰匙如利用網路傳送，還是可能為他人竊取，所以最好透過其他的管道交換，這增加了實用上的困擾。

美國麻省理工學院 Rivest、Shamir 與 Adlemam 三位數學家在 1977 年共同發表 RSA 密碼演算法，解決了對稱性密碼系統中鑰匙交換的問題，這種演算法主要以兩個質數(prime number)作為加密與解密的兩個鑰匙(key)，加密與解密所用的鑰匙不同，使用者先透過其演算法產生一對鑰匙，一把稱為私密鑰匙 (private key)，另一把稱為公開鑰匙(public key)，私密鑰匙只有使用者自己知道，而公開鑰匙則可公開讓大眾知道，兩把鑰匙有一定的依存關係，但從公開鑰匙無法推得私密鑰匙。資料的加密是使用 RSA 的演算法，配合公開鑰匙來處理，而資料的解密則必須使用對應的私密鑰匙才能完成，鑰匙的長度(位元數)決定了加密編碼的複雜度，只要 RSA 鑰匙長度增加，要破解 RSA 得大費周章，因此它算是相當安全的保密系統。

非對稱密碼系統不只 RSA 一種，具有相當高的安全性。但與對稱性密碼系統相比，其運算所需的時間較長，因此實作上並不用於直接對明文加密。常見的作法還是先使用對稱性密碼法 (如 DES) 對明文加密，再使用非對稱性密碼法將所用的 DES 鑰匙加密，然後將密文與加密後的鑰匙一同傳送。收文的一方則先用非對稱性密碼法解開 DES 鑰匙，再用解開後的鑰匙將密文解密。如此兼顧了運算的速度與鑰匙交換的問題。



Signature Service



Verification Service

同樣地，產生電子簽章的演算過程也很花時間，所以實作上並不直接對文件進行運算，而是將文件先以一個單向赫序函數 (one-way hash function) 處理。所謂單向函數是指從函數的計算結果無法 (非常困難) 推算出輸入給函數的值。而赫序函數則可將輸入的資料濃縮成較短且為特定長度的結果。任意的文件資料經過一個單向赫序函數計算後，可以產生一串固定長度的資料，因為不太可能設計另一份文件資料而在同一函數運算後產生相同的結果，所以該結果可視為原始文件資料

的特徵值，稱為數位指紋 (digital fingerprint) 或訊息摘要 (message digest)。電子簽章在實作上便是針對文件的數位指紋進行運算而產生的。因為不同的文件資料其數位指紋不太可能相同，因此收文者可對收到的文件以相同的單向赫序函數計算，然後比對計算的結果與其數位指紋，如果兩者不同，就表示文件遭到了竄改。透過這種方式，可以提供資料的真確性服務。

確認使用者身份時，Sender 可利用 Encryption Service，以 Private Key 將 (ID，Password) 加密而成為密文後，由 Sender 將 ID 與加密後之(ID，Password) 送到 Receiver，Receiver 將收到的 ID 與加密後之(ID，Password)利用 Decryption Service 解密。

Communication 時，Sender Summit 一個 Encryption Service Request，Encryption Service 利用 Sender 之 Private Key 來 Encrypt，然後由 Sender 送出 Cipher 給 Receiver，Receiver 接到後 Submit 一個 Decryption Service Request，Decryption Service 利用 Sender's Private Key 來 Decrypt。

Non-Repudiation 可以由 Sender 以欲簽署之文件提出 Signature Service Request，Signature Service 利用 Sender 之 Private Key 來製作 Signature，然後由 Sender 送出文件及 Private Key 製作之 Signature 給 Receiver，Receiver 接到後 Submit 文件及 Private Key 製作之 Signature

給 Verification Service ，Verification Service 利用 Sender’s Public Key 來 Verify。

系統整合實作

整合對象

醫學資訊交換中心(Medical Information Exchange Center， MIEC)的構想乃是為了建立病歷資訊共享架構。其功能類似國內龐大的金融資訊 EDI 網路；MIEC 所扮演的角色則與金資中心在金融資訊網路中所扮演的角色相同。MIEC 可連接國內各級醫療院所、保險機構、衛生研究機構及衛生主管機關；各級醫療院所的病歷資訊皆可在此龐大的醫療資訊網路內流通。此資訊可提供衛生主管機關即時地監控國內醫療服務現況，亦可提供研究機構教學、統計及疾病研究之用。未來 MIEC 龐大的資訊交通量如果非單一資訊交換中心所能負荷。可以仿照全國醫療網之區域資訊中心(RC) 設置區域 MIEC，建構起完整的網路架構，提供醫療相關機構連線傳輸資料的管道，再將這些區域 MIEC 彼此串連形成醫學資訊網路。

除了醫學資訊交換中心以外，國家病歷庫的構想也是一個可以考慮的方向。建立醫學資訊交換中心乃是消極的提供病歷資訊流通管道，而國家病歷庫乃是更積極、更有計劃地將病歷資訊由國家來統一管理。醫學資訊交換有地方自制的精神，相對地，國家病歷庫乃是中央集權的管理模式，各具有不同的優劣。各醫療院所儲存的僅是國家病歷庫的一部份，不同醫院之電子病歷可互相整合而成為一個完整且統

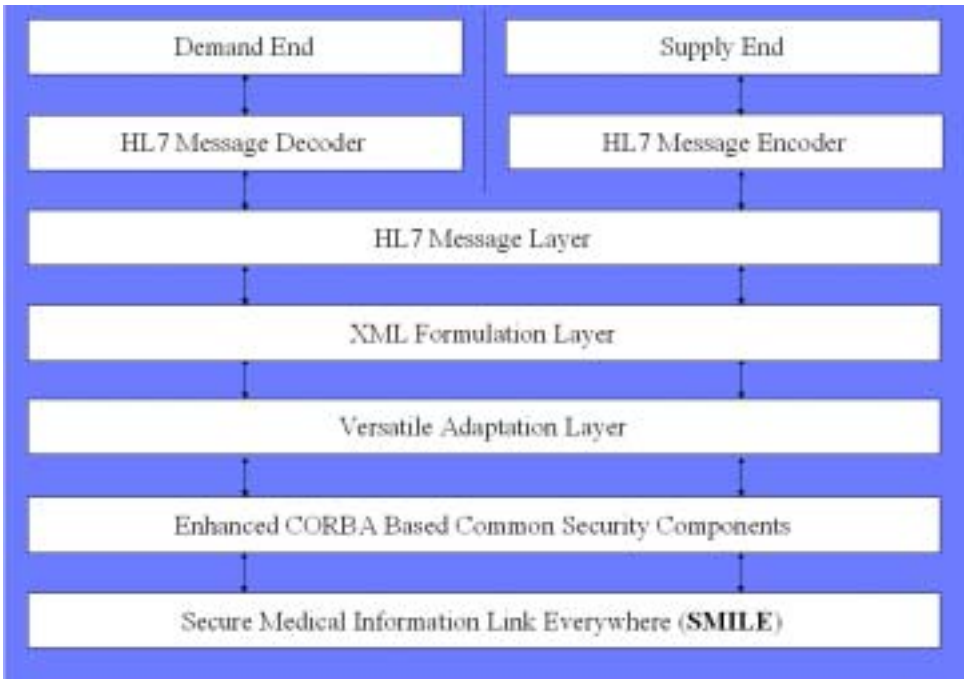
一的資料庫，達成病歷跟病人一起轉診的目標，促進病歷透明化及電子病歷交換及分享的目地，並透過國家病歷庫保管全國的病歷，落實病歷為病人所有。

由於各醫院的 HIS 系統異質性大, 因此台北榮民總醫院提出病歷信託中心之概念，病歷信託中心的構想乃是由企業界經營，向委託的民眾收費(政府可補貼若干費用)，病歷信託中心可使民眾掌握個人健康資訊，達到更有效率之治療及預防保健；就衛生機關而言，病歷信託中心可促進醫療資源共享，減少浪費；就醫院本身而言，亦可透過病歷信託中心增進服務速度與品質。具有高度的適應性病歷信託中心可透過網路接收具 HL7 和 DICOM 通訊能力，但能上網的醫院客戶可利用包含文字及影像的 HTML 經由網路做文字及影像病歷的交換，病患可自行上網查詢，亦可同意其診療醫院經網路查詢放在病歷信託中心的電子病歷，而無法上網的醫院客戶則以傳真機交換文字性的病歷至於個人, 客戶則經由網路提供 On-Line 的病歷或以光碟提供 Off-Line 的病歷使病歷隨身化，本計劃的整合對象即是台北榮民總醫院。

整合內容

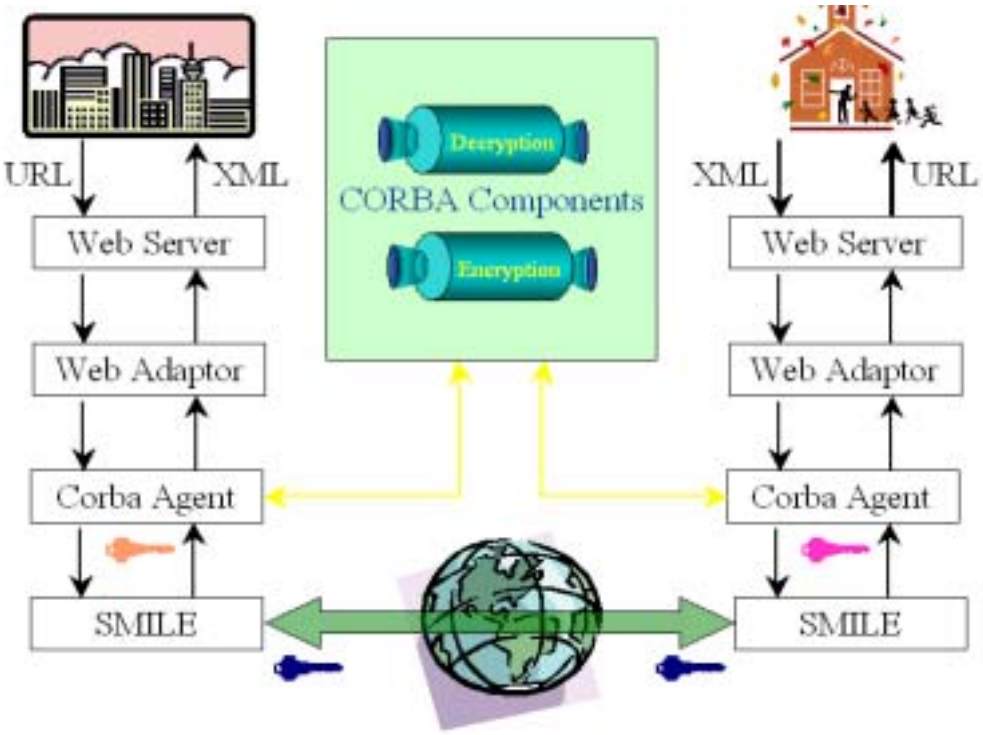
Example XML With HL7 Message Format

整合架構



Architecture For Secure Message Exchange

整合方法



Integration Methodology Beteen TMUH and VGHTP

整合效益

1. 利用 http 協定，以 URL 存取資料，提供一個簡易的使用介面。
2. 經由 HL7/XML 之協定交換資訊，符合醫學資訊交換標準。
3. 透過 Web Adaptor 連接 CORBA Agent，降低各醫療院所進入障礙。
4. CORBA Agent 提供 Private/Public Key Encryption/Decryption
5. Security Component 內含 Secret Key，跨越主機之 Security Component 之間傳遞參數可利用 Secret Key 加密，以免遭受竊聽。
6. SMILE Layer 亦利用 Secret Key 加密，宛如置於 Virtual Private Network 上執行。

討論

本計劃利用已經發展之現有基礎，以擴充式公用物件請求仲介者架構為基礎建構醫學資訊交換中心之共用安全物件，本計劃選擇 CORBA 乃是由於 CORBA 是跨平台的協定。

公用物件請求仲介者架構的制定者 OMG (Object Management Group) 乃是一個由會員贊助而成立的非營利組織， 其目的在推廣物件導向的觀念及使用 並致力於加強軟體的可攜性、再利用性以及互通性。該組織會員包括了廠商、學術單位及用戶。目前全球大約有三百八十家重要廠商為其會員。CORBA 乃是 OMG 在一九九一年十二月提出之物件導向分散式工作環境規格，一九九三年十二月一九九四年九月此規格多次被修訂，目前最新之版本為一九九五年七月之 CORBA 2.0 版，互通性及 C++對應之標準，均於新版之標準中更明確地被製訂。 CORBA 規格對一個物件請求仲介者之標準介面作了詳細的規定。根據 OMG 的定義，一個物件請求仲介者為一可提供分散式環境上各個物件透明化的請求服務與回應接收功能的應用程式建構工具。

就 Security 的觀點，跨平台亦是相當重要的，因此本計劃安全機制則以 JAVA 研發，為了考慮各醫院的 HIS 系統異質性大且各醫療院所使用的開發工具不同之問題，本計劃除開發 CORBA 之服務以外，本計劃更提出 Versatile Adaptation Layer 的觀念，保留介面讓其他方式使用，使整個 MIEC

infrastructure 有一致之方向。為證實其適應性，本計劃已先與台北榮民總醫院病歷信託中心之計劃，與提供一個簡易的使用介面，利用 http 協定，以 URL 存取 XML/HL7 之資料，並利用本計劃之加密機制，將資料在收發間加密。

結論與建議

在醫學資訊發展的過程中，醫學資料高度數位化已經成為主流，並且為因應醫學資料豐富性及完整性的要求，如 Video，Image 及 Text 等不同型態的媒體格式常結合於同類別的醫學資料。而近幾年來由於全球網際網路(Internet)之快速成長，使的各種醫學資料透過 Internet 的形式提供給廣大的群眾。醫療資訊系統日新月異，對於病患就醫流程之便利性具有正面效益，但也由於醫療資訊的引進造成病患就醫資料之安全性所受到挑戰相對地提高，醫院對於病歷之保管權責困難度相對提高，其重要性也隨著臨床資訊之數位化程度與日俱增。

本計劃經整理系統安全之顧慮、整理醫療資源之保密策略方式、分析系統之需求等步驟訂定之 Component 規格。並架設 Java ORB 相關環境與伺機器，以 Java Cryptography 及 Java Security API 實作相關 Component。為測試 Component，業已利用 http 協定，以 URL 存取資料，提供一個簡易的使用介面，經由符合醫學資訊交換標準之 HL7/XML 協定與臺北榮民總醫院病歷中心交換資訊，透過 Web Adapter 連接 CORBA Agent，降低進入障礙。其中 CORBA Agent 提供使用 Encryption/Decryption Component 內含，跨越主機之 Security Component 之間傳遞參數可利用 Secret Key 加密，以免遭受竊聽，SMILE Layer 亦利用 Secret Key 加密，宛如置於 Virtual Private Network 上執行。

Component based 的 Multi-tier 架構已是未來開發醫學資訊系統的主流，透過本研究規格的制訂與實作更可以讓醫學資訊系統的開發人員除了專注在醫學邏輯的開發上，更不用擔心受限於單一廠商的束縛，除了可以縮短開發的時程、簡化資訊系統的部署(Deployment)及維護(Maintenance)，並且具有高度的可攜性及穩定性，更進一步享有完整的安全性。醫療機構間不限於單純的資料交換，而可以協同進行組織間的醫療活動，像是醫藥分業、轉診等。醫療機構間也可以利用呼叫對方系統物件庫內的物件來得到服務，值得推廣。

針對醫學資訊交換之安全議題，本研究有下列建議:

醫學資訊交換系統的標準若能透過本研究規格, 則醫學資訊交換邏輯是由 MIEC 提供位於伺服端(Server-Side)的元件化(Component-based)標準模型，而透過其標準，該邏輯模型可以落實在任一作業系統平台的應用軟體系統下而具有安全交易處理能力、高度的安全性、穩定性、分散式網路的多層式架構的資訊系統。如此重要的醫學資訊交換工作實不應放任各醫療機構與廠商，僅僅靠著標準來連結，不論是人力、財力、能力或是安全性皆非上選，而由衛生署來為醫療機構提供 Gateway 亦非可行，因此本研究極力希望能透過這個研究，取得折衷之道，由衛生署委由研究機構開發醫學資訊交換之物件供各醫療機構與廠商分享，透過共用醫學資訊交換物件

之管控，標準與安全之期望可待，“**MIEC Write Once， Run Anywhere**”的願景(Vision)不遠矣。

參考文獻

[1]. M.T.Kwan et al ， ”NCSA’s World Wide Web Server：Design and Performance ”， IEEEE Computer，pp.68-74，1995。

[2]. D.Andresen et .al .，”SWEB: Toward s Scalable World Wide Web Server on Multicomputers”， Department of Computer Science Tech Report-TRCS95-17，U.C.Santa Barbara。

[3]. Damani P.-Y chung， Huang，C .Kintala， Y. –M . Wang，”ONE-IP：Techniques for Hoting a Service on a Cluster of Machines” in Sixth International World Wide Conference， Santa Clara， Apr. 1997 .(URL:http://www6.nttlabs.com/HyperNews/get/Paper196.html)。

[4]. Dias et .al .，”A Scalable and Highly Available Server”， COMPCON’96，pp.85-92，1996。

[5]. J . Gwertzman and M. Selter， “The Case for Geographical Push-Caching”， HotOS’95， 1996。

[6]. Bestavaros ， “Speculative Data Dissemination and Service to Reduce Server Load . Network Traffic and Service Time in Distributed Information System “ ， in Proceedings of the International Conference on Data Engineering March 1996。

[7]. Heddays and S. Mirdad ，”WebWave：Globally Load Balance Fully Distributed Caching of Hot Published Documents”， in Proceedings of

the International Conference on Distributed Computing Systems, ICDCS'97 , May 1997.

[8]. Luotonen and K. Altis " Word-Wide Web Proxies "Computer Networks and ISDN Systems , 1994。

[9]. Bestavaors, "Using Speculation to Reduce Server Load and Service Time on the WWW" , Computer Science Technical Report , BUCS-TR-95-006, Boston University , April 1995。

[10]. Y-M. Wang, P. –Y chung , C.-M. Lin, Y .Huang, "HAWA： A Client-side Approach to High Availability Web Access" in Sixth International World Wide Web Conference, Poster session , Santa Clara Apr.1997.(URL://http://poster.www6conf.org/poster/736/F.html)

[11]. M. Garland et .al , "Implement Distributed Server Groups for the Worlds Wide Web " , Technical Report CMU-CS-95-114, School of Computer Science , Carnegie Mellon University , January 1995。

[12]. Yoshikawa et .al , "Using Smart Clients to Build Scalable Service ", in Proceed of USENIX'97。

[13]. J. Gwwertaman and M. seltzer of Geographical Push-Caching " manuscript。

[14]. M.E. Crovella and R.L. Carter, "Dynamic Server Selection in the Internet", Proc. of the Third IEEE Workshop on the Architecture and

Implementation of High Performance Communication Subsystem, HPCS'95, 1995.

[15]. J.D. Guyton and M.F. Schwartz, "Locating Nearby Copies of Replicated

[16]. Internet Servers", Technical Reprot CU-CS-762-95, University of Colorado at Boulder, 1995.

[17]. T.D.C Little and D. Venkatesh, "Popularity-Based Assignment of Movies to

[18]. Storage Devices in a Video-on-Demand System", ACM/Springer Multimedia Systems, 1994.

[19]. L. Kleinrock, "Queuing Systems", Vol. 1, John Wiley and Sons, 1975.

[20]. D.N. Serpanos et. al, "A load and Storage Balancing Algorithm for Distributed

[21]. Multimedia Servers", Proc. F the IEEE Conference on Computer Design (ICCD'96), pp.170-174, October 1996.

[22]. R.K. Ahuja et. al., "Network Flows: Theory, Algorithms and Applications",

[23]. Prentice Hall, 1993.

[24]. R. Fourer et. al., "AMPL: A Modeling Language For Mathematical

[25]. Programming", The Scientific Press, 1993.

[26]. 林錦鴻 因應醫藥分業策略性資訊系統可行性評估 Feasibility Evaluation of Prescription Information System 碩士論文 國立台灣大學醫療機構管理研究所 (86 年)

[27]. 方智惠 全民健保特約醫療機構門診處方釋出情形之探討 A Study on Prescriptions Released from NHI Contract Hospitals and Clinics 碩士論文國立成功大學 臨床藥學研究所 (85 年)

[28]. 宋居定 全民健保特約藥局接受門診處方箋之探討 A Study on Prescriptions Released to NHI Contract Pharmacies 碩士論文 國立成功大學 臨床藥學研究所 (85 年)

[29]. 趙俊人 全民健康保險「公辦民營」之研究 Research on National Health Insurance Adopting “Public Program Running by the Local People”碩士論文 東吳大學 政治學系研究所 (86 年)

[30]. 陳登旺 全民健保特約診所處方箋釋出之現況分析與探討 An Analytical Study on the Prescription Released by Primary Clinics in Taiwan 碩士論文 中國醫藥學院 醫務管理研究所 (86 年)

[31]. 彭啟釗 台灣診所採用資訊科技之關鍵因素研究 An Empirical Study on the Critical Factors Influencing the Use of Information Technology in Clinics in Taiwan 碩士論文 國立成功大學 企業管理學系(84 年)

[32]. 李旻穎 UML 技術於物流中心資訊系統分析與設計之應用研究

Applied Research of UML Technology on the Information System Analysis and Design of a Distribution Center 碩士論文 國立中山大學資訊管理學系(87 年)

[33]. 楊仙維 整合關聯式資料庫於 CORBA 架構之封裝實作 An Implementation of Wrapper for Integrating Relational Database with CORBA 碩士論文 國立中興大學應用數學系 (87 年)

[34]. Simon Bennett, Steve McRobb and Ray Farmer Object-Oriented Systems Analysis and Design using UML 1999 McGraw-Hill International Editions

[35]. 李建民 CORBA 的即時與容錯架構 Real-Time and Fault-Tolerant Enhancements to CORBA 碩士論文 國立台灣大學 電機工程學系研究所 (87 年)

[36]. M.T.Kwan et al ，”NCSA’s World Wide Web Server：Design and Performance ”， IEEEE Computer，pp.68-74，1995。

[37]. Andresen et .al .，”SWEB: Toward s Scalable World Wide Web Server on Multicomputers”， Department of Computer Science Tech Report TRCS95-17， U.C.Santa Barbara。

[38]. Damani P.-Y chung， Huang，C .Kintala， Y. –M . Wang，”ONE-IP：Techniques for Hoting a Service on a Cluster of Machines” in Sixth International World Wide Conference， Santa Clara， Apr. 1997

[39]. Dias et .al .，”A Scalable and Highly Available Server”， COMPCON’96， pp.85-92， 1996。

[40]. J . Gwertzman and M. Selter， “The Case for Geographical Push Caching”， HotOS’95， 1996。

[41]. Bestavaros ， “Speculative Data Dissemination and Service to Reduce Server Load . Network Traffic and Service Time in Distributed Information System “ ， in Proceedings of the International Conference on Data Engineering March 1996。

[42]. Heddays and S. Mirdad ，”WebWave：Globally Load Balance Fully Distributed Caching of Hot Published Documents”， in Proceedings of the International Conference on Distributed Computing Systems, ICDCS’97 ， May 1997.

[43]. Luotonen and K. Altis “ Word-Wide Web Proxies ”Computer Networks and ISDN Systems ， 1994。

[44]. Bestavaors， “Using Speculation to Reduce Server Load and Service Time on the WWW”， Computer Science Technical Report ， BUCS-TR-95-006，Boston University ， April 1995。

[45]. Y-M. Wang，P. –Y chung ， C.-M. Lin， Y .Huang， “HAWA： A Client-side Approach to High Availability Web Access” in Sixth International World Wide Web Conference， Poster session ，Santa Clara

Apr.1997.

[46]. M. Garland et .al ， “Implement Distributed Server Groups for the Worlds Wide Web “ ， Technical Report CMU-CS-95-114， School of Computer Science ， Carnegie Mellon University ， January 1995。

[47]. Yoshikawa et .al .， “Using Smart Clients to Build Scalable Service “， in Proceed of USENIX’97。

[48]. J. Gwwertaman and M. seltzer of Geographical Push-Caching “ manuscript。

[49]. M.E. Crovella and R.L. Carter， “Dynamic Server Selection in the Internet”， Proc. of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystem, HPCS’95， 1995.

[50]. J.D. Guyton and M.F. Schwartz， “Locating Nearby Copies of Replicated Internet Servers”， Technical Reprot CU-CS-762-95， University of Colorado at Boulder， 1995.

[51]. T.D.C Little and D. Venkatesh， “Popularity-Based Assignment of Movies to Storage Devices in a Video-on-Demand System”， ACM/Springer Multimedia Systems， 1994.

[52]. L. Kleinrock， “Queuing Systems”， Vol. 1， John Wiley and Sons,

1975.

[53]. D.N. Serpanos et. al, "A load and Storage Balancing Algorithm for Distributed Multimedia Servers", Proc. F the IEEE Conference on Computer Design (ICCD'96), pp.170-174, October 1996.

[54]. R.K. Ahuja et. al., "Network Flows: Theory, Algorithms and Applications", Prentice Hall, 1993.

[55]. R. Fourer et. al., "AMPL: A Modeling Language For Mathematical Programming", The Scientific Press, 1993.

名詞解釋

1. **ASN.1** Abstract Syntax Notation One. A data definition language.
2. **CA** Certificate Authority. An entity that issues， manages and revokes certificates.
3. **CA certificate** Identifies the Certificate Authority (CA) that issues server and/or client authentication certificates to the servers and clients that request these certificates. Because it contains a public key used in digital signatures, it is also referred to as a signature certificate. If the CA is a root authority， the CA certificate may be referred to as a root certificate. Also sometimes known as a site certificate.
4. **CA hierarchy** A Certificate Authority (CA) hierarchy contains multiple CAs. It is organized such that each CA is certified by another CA in a higher level of the hierarchy until the top of the hierarchy， also known as the root authority， is reached.
5. **Client certificate** Refers to a certificate used for client authentication， such as authenticating a Web browser on a Web server. When a Web browser client attempts to access a secured Web server， the client sends its certificate to the server to allow it to verify the client's identity.
6. **CRL** Certificate revocation list. A document maintained and published by a CA that lists certificates that have been revoked by the CA.
7. **CSP** Cryptographic Service Provider. The code that actually performs authentication， encoding and encryption services accessed by Win32-based

applications through the CryptoAPI.

8. **Key exchange certificate** Certificate used to encrypt information sent to another party. The Certificate Authority (CA) key exchange certificate can be used by a client to encrypt information sent to the CA.
9. **LDAP** Lightweight Directory Access Protocol. A more easily implemented subset of the X.500 DAP standard for directory services.
10. **MD2** Message digest algorithm.
11. **MD4** Message digest algorithm.
12. **MD5** Message digest algorithm.
13. **Message digest** A short， fixed-length digital string derived from a longer， variable-length message using a computational algorithm.
14. **PKCS** Public-Key Cryptography Standards.
15. **Root authority** The Certificate Authority (CA) at the top of a CA hierarchy. Certifies CAs in the next level of the hierarchy.
16. **Root certificate** A self-signed Certificate Authority (CA) certificate that identifies a CA. It is called a root certificate because it is the certificate for the root CA. The root CA must sign its own CA certificate because by definition there is no higher certifying authority to sign its CA certificate.
17. **Server certificate** Refers to a certificate used for server authentication， such as authenticating a Web server to a Web browser. When a Web browser client attempts to access a secured Web server， the server sends its certificate to the browser to allow it to verify the server's identity.
18. **SET** Secure Electronic Transaction. A protocol for secure electronic transactions over the Internet.

19. **SHA** Secure Hash Algorithm. A message digest algorithm.
20. **Signature certificate** A certificate containing a public key that is used to verify digital signatures.
21. **Site certificate** Both server certificates and Certificate Authority (CA) certificates are sometimes called site certificates. When referring to a server certificate, the certificate identifies the Web server presenting the certificate. When referring to a CA certificate, the certificate identifies the CA that issues server and/or client authentication certificates to the servers and clients that request these certificates.
22. **S/MIME** Secure/Multipurpose Internet Mail Extensions. A protocol for secure electronic mail over the Internet.
23. **SSL** Secure Sockets Layer. A protocol for secure network communications using a combination of public and secret key technology.
24. **URL** Uniform Resource Locator. An encoding scheme used by the World Wide Web (WWW) for accessing data on the Internet. The general format is service://hostname/pathname.
25. **X.509** Standard certificate format supported by Certificate Server.
26. **Public Key** A number associated with a particular entity (for example, an individual or an organization). A public key is intended to be known to everyone who needs to have trusted interactions with that entity.
27. **Private Key** A number that is supposed to be known only to a particular entity. That is, private keys are always meant to be kept secret. A private key is always associated with a single public key.
28. **Digital Signature** A string of bits that is computed from some data (the data

being "signed") and the private key of an entity. The signature can be used to verify that the data came from the entity.

29. Cryptography Algorithm

An algorithm used to help ensure one or more of the following:

1.the confidentiality of data

2.authentication of the data sender

3.integrity of the data sent

4.nonrepudiation; a sender cannot deny having sent a particular message

A digital signature algorithm provides some of these characteristics. Also see message digest algorithms. Digital signature and message digest algorithms are available in JDK 1.1.

30. Encryption The process of taking data (called cleartext) and a short string (a key) and producing ciphertext， which is data meaningless to a third-party who does not know the key.

31. Decryption The inverse of encryption; the process of taking ciphertext and a short key string， and producing cleartext.

32. Certificate A digitally signed statement from one entity， saying that the public key of some other entity has some particular value. If you trust the entity that signed a certificate， you trust that the association in the certificate between the specified public key and another particular entity is authentic.

33. Message Digest Algorithm (or One-Way Hash Function) A function that takes arbitrary-sized input data (referred to as a message) and generates a fixed-size output， called a digest (or hash).

34. Engine Class The term engine class is used in the Java Security API to refer to a class that provides the functionality of a type of cryptography algorithm. The Security API defines a Java class for each engine class. For example, there is a MessageDigest class, a Signature class, and a KeyPairGenerator class. Users of the API request and utilize instances of these engine classes to carry out corresponding operations. A Signature instance is used to sign and verify digital signatures, a MessageDigest instance is used to calculate the message digest of specified data, and a KeyPairGenerator is used to generate pairs of public and private keys suitable for a specified algorithm. An engine class provides the interface to the functionality of a specific type of algorithm, while its actual implementations (from one or more providers) are those for specific algorithms. The Signature engine class, for example, provides access to the functionality of a digital signature algorithm. The actual implementation supplied in a Signature subclass could be that for any kind of signature algorithm, such as SHA-1 with DSA, SHA-1 with RSA, or MD5 with RSA.