一、緒論

1.1 研究動機

長久以來，資訊安全（Information Security）一直是人們所重視的議題，隨著網際網路（Internet）的廣泛使用，與透過網際網路大量而蓬勃發展的各項服務，也衍生出許許多多不同於過去的資訊安全需求，而無線網路（Wireless Network）的快速發展，益加凸顯網路安全的重要性。網路安全協定是一種資訊傳輸的技術，我們常用網路安全協定來識別資料真偽，網路安全協定的種類非常的多，而這些安全協定也有許多的用途，最廣為人所知的就是在傳送重要資料或密秘文件，例如電子郵件、會議資料、銀行帳戶進出等，用途在於辨別真偽，防止他人的仿冒。

隨著科技的進步，電腦的發明及普及，人們開始將所擁有的資料數位化，並將資料傳輸於網路中，雖然，這些數位化的資料已不同於一般真實世界上的紙上作業傳送。而在許多情況上，我們也需要確保這些數位化的資料不被第三者所知，所以網路安全協定是否安全也開始被人們所重視，這是一個近年來才開始研究探討的領域，不斷地有新奇的漏洞被發現，同時人們也不斷地在思索找出各種不安全的形況。

現今的網路安全，大多依賴在金鑰的加密，且依附在健全的安全協定上，在這篇論文之中，我們專注於網路安全協定的漏洞找尋上，找出網路安全協定的漏洞更具有許多挑戰性。

1.2 研究方法

本篇論文的目標在於設計良好而有彈性的安全協定模擬器，我們安全協定模擬器的核心，主要使用漏洞法則來嵌入及偵測網路安全協定。在這篇論文中，我們利用編譯器的技術，來判斷安全協定輸入，進而加以分析，給予整個安全協定模擬器完整的架構。由於有了完整的架構，我們可以更清楚了解安全協定模擬器的性質，以及在安全協定模擬器對原始的輸入資料的修改之後，利用程式來判斷是否安全，這可以幫助程式設計師設計安全協定。

1.3 論文架構

首先，在第二章，說明了何謂安全協定，再來於第三章，介紹了安全漏洞相關研究，以及目前研究的網路安全協定的技術，使我們對網路安全協定漏洞有廣泛地了解，除此之外，也介紹評量網路安全協定漏洞的準則。然後，在第四章，提出一個良好的安全協定模擬器之設計方法，並且使用安全漏洞法則(pattern)來推導和分析，藉此了解如何控制我們模擬器，來達到我們的安全需求。在第五章，介紹了操作介面，測試本論文提出的安全協定模擬器，然後分析實驗的結果，以及和其它人所做的相關研究做比較。最後，在第六章，提出本論文的結論，並說明未來的研究方向。

二、何謂安全協定

2.1 安全協定的基本概念

所謂協定,就是兩個或者兩個以上的參與者為完成某項特定的任務而採取的一系列步驟。這個定義包含三層含義：

(1)、協定自始至終是有序的過程，每一個步驟必須執行，在前一步沒有執行完之前，後面的步驟不可能執行；

(2)、協定至少需要兩個參與者；

(3)、通過協定必須能夠完成某項任務。

而密碼協定是使用密碼學技術的協定，協定的參與者可能是可以信任的人，也可能是攻擊者(Adversary)和完全不信任的人。密碼協定包含某種密碼演算法。在網路通信中最常用的、基本的密碼協定按照其完成的功能可以分成以下三類：

(1)、金鑰交換協定(Key Exchange Protocol)：

一般情況下是在參與協定的兩個人或者多個人之間建立秘密通訊管道，通常用於建立在一次通訊中所使用的會話金鑰(Session Key)，也就是說，下一次之會話金鑰會重新的建立。協定可以採用對稱密碼體制，也可以採用非對稱密碼體制，例如 Diffie-Hellman 金鑰交換協定。

(2)、認證協定(Authentication Protocol)：

認證協定中包括實體認證（身份認證）協定、消息認證協定、資料源認證和資料目的認證協定等，用來防止假冒、篡改、否認等攻擊。

(3)、認證和金鑰交換協定(Authentication and Key Exchange Protocol)：

這類協定將認證和金鑰交換協定結合在一起，是網路通信中最普遍應用的安全協定。常見的有 Needham-Schroeder 協定、分佈認證安全服務(DASS)協定、ITU-T X.509 認證協定等等。

2.2 安全協定的安全性

安全協定是許多分散式系統安全的基礎，而確保這些協定的安全運行是極為重要的。大多數安全協定只有為數不多的幾個消息傳遞而以，其中每一個訊息都是經過了很巧妙設計，消息之間存在著複雜的相互作用和制約；同時，安全協定中使用了多種不同的密碼體制，就因為如此，所以安全協定這種複雜的情況導致目前的許多安全協定存在安全漏洞。而造成協定存在安全漏洞的原因主要有兩個：一是協定設計者誤解，或是採用了不恰當的技術；二是協定設計者對環境要求的安全需求研究不足。因此，對協定的安全性進行分析和研究是一個很重要的課題。

由於實際應用的安全協定產生漏洞的原因很多，所以很難有一種通用的分類方法將安全協定的安全缺陷進行分類。根據安全漏洞產生的原因和相應的攻擊方法，可以將安全漏洞分成以下六點：

(1)、基本協定漏洞：

是指在安全協定的設計中，沒有或者很少防範攻擊者的攻擊。

(2)、密碼/金鑰猜測漏洞：

這類缺陷產生的原因，是使用者常常會從一些常用的詞中選擇其密碼，而導致攻擊者能夠進行密碼猜測攻擊，或是利用字典攻擊；或者是利用了不安全的假亂數生成演算法構造金鑰，一一加以比對，使攻擊者能夠找出該金鑰。

(3)、不新鮮（stale）消息漏洞：

主要是指協定設計中，對訊息的新鮮性沒有充分考慮，而使攻擊者能夠進行消息重送攻擊，包括訊息源的攻擊、消息目的的攻擊等等。

(4)、並行通訊漏洞：

協定對並行通訊攻擊若缺乏防範，會導致攻擊者通過交換適當的協定訊息後，能夠獲得所需要的資訊，而造成傷害。

(5)、內部協定漏洞

協定的可達性存在問題，也就是說每一步驟都要完成，協定的參與者中至少有一方若沒有完成所有必須的動作，這會導致的漏洞而造成傷害。

(6)、密碼系統漏洞

協定中使用的密碼演算法和密碼協定，導致協定不能完全滿足所要求的機密性、認證等需求，而造成傷害。

安全協定的安全性到現在為此，都還是一個很難解決的問題，現今有許多廣泛應用的安全協定，經過一段時間後，都有被發現安全漏洞的存在。因此，從安全協定的分析和設計角度來看的話，我們都不能夠存在輕信和盲從的心理，而應當對協定的安全性作出認真的評估和分析，以確保真正的安全。

2.3 安全協定的分析

目前，對安全協定進行分析的方法主要有兩大類：一類是攻擊檢驗方法，而另一類是形式化的分析方法。

所謂攻擊檢驗方法，就是搜集使用目前的對協定的有效攻擊方法，逐一對安全協定進行攻擊，檢驗安全協定是否具有抵抗這些攻擊的能力。在分析的過程中主要使用自然語言和示意圖，對安全協定所交換的消息進行剖析。這種分析方法往往是非常有效的，關鍵在於攻擊方法的選擇。

而另一類，形式化的分析方法是採用各種形式化的語言或者模型，為安全協定建立模型，並按照規定的假設和分析、驗證方法證明協定的安全性。目前，形式化的分析方法是研究的熱點，但是就其實用性來說，還沒有什麼突破性的進展。

近幾年來，密碼學家提出了許多關於安全協定的形式化分析方法，以檢驗協定中是否存在安全缺陷。總的來說，協定的形式化分析技術可以概括為四大類：

(1)、使用通用的、不是為分析安全協定專門設計的形式化描述語言和協定校驗工具建立安全協定的模型並進行校驗。其主要思想是將安全協定看作一般的協定，並試圖證明協定的正確性。採用的工具和模型與驗證一般協定的類似，例如使用有限狀態圖、Petri 網模型、LOTOS 語言等等。這種方法的一個主要缺點是僅證明協定的正確性而不是安全性。

(2)、安全協定的設計者，設計專門的專家系統來制定協定的校驗方案，並進行協定檢驗，從而對協定的安全性作出結論。其主要思想是根據協定的設計開發專用的專家系統，使用專家系統發現協定是否能夠達到不合理的狀態（比如金鑰的洩露等等）。這種技術能夠很好的識別漏洞但是不能證明協定的安全性，也不可能發現未知的漏洞。

(3)、使用基於知識和信念的邏輯，來建立所分析的協定的安全需求模型。

(4)、基於密碼學系統的代數特性，開發協定的形式化模型。這種方法是將安全協定系統當作一個代數系統模型，表示出協定的參與者的各種狀態，然後分析某種狀態的可達性。

安全協定的形式化分析方法本身是研究的熱點，但是其應用並不是非常廣泛，主要原因是安全協定的安全性的形式化過程比較困難。需要指出的是，由於安全協定本身的複雜性，目前並沒有一種方法能夠給出安全協定安全性的充分而且必要的理論證明。上述每一類方法都有不同的側重點，或者說或多或少地存在不足之處，我們在使用上述方法分析安全協定的時候，應當仔細分析協定的特點、應用環境和需求，綜合使用這些分析方法，以得到一個比較合理的結果。

2.4 安全協定符號的介紹

(1)、Na,Nb : Nonce，一種隨機變數，Na 代表了 A 所產生的隨機變數，在每次執行一次協定時，所產生的 Nonce 值都會改變。

(2)、Kas,Ka : A 所專用的 Key，除了 Server(S)知道之外，其它人都不可能知道這把 Key 的內容。

(3)、Kab : Session Key of A and B，A 和 B 的對話金鑰，用於建立起之連線，若第二次連線的話，則 Kab 值會改變。

(4)、{X}K：X 表示一段明文訊息，K 表示金鑰，{X}K 表示明文訊息 X 用 K 加密，{X}K 除了有 K 者才解密得知 X。若 X‘ 表示另一段明文訊息，則找不到 X‘ 使得 {X‘ }K={X}K。

三、安全漏洞相關研究

網路安全協定的技術及問題近年來不斷地被研究探討，網路協定工程師面對網路安全問題，並且要尋求解決之道，因此首要步驟是去獲得及了解不同的網路安全漏洞相關資訊，而本章重點在於介紹一些網路安全漏洞的分類，並說明如何發生這些漏洞。

3.1 攻擊法概述

無論是針對電腦系統或者網路系統，現行的攻擊手法種類繁多。我們可以利用攻擊的手段來對攻擊法分為阻礙、攔截、修改及捏造等四種最基本的分類。

1. 阻礙（Interrupt on）：利用種種攻擊手法攤瘓電腦系統或者讓資料無法使用。這是針對資料的可利用性加以攻擊。

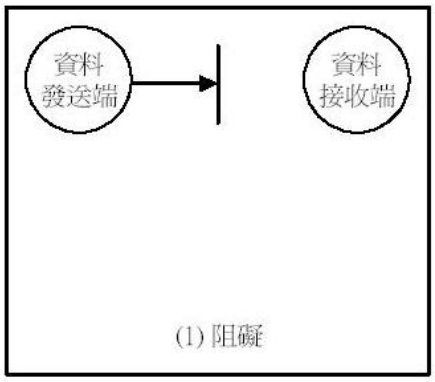


圖 3-1：攻擊手段之阻礙（Interrupt on）

2. 攔截（Interception）：一個未經過合法授權（Unauthorized）的第三者從系統中欄截資料。這是針對資料的機密性加以攻擊。

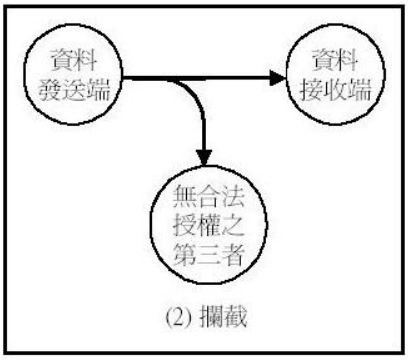


圖 3-2：攻擊手段之攔截（Interception）

3. 修改（Modification）：一個未經過合法授權的第三者不僅從系統中欄截資料，並將資料加以竄改。這是針對資料的完整性加以攻擊。

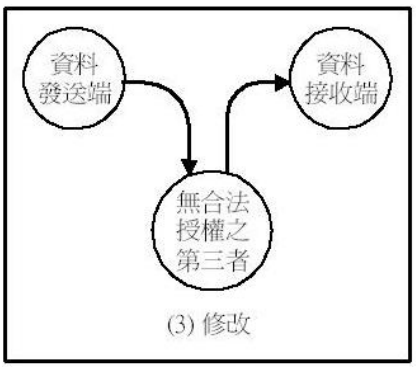


圖 3-3：攻擊手段之修改（Modification）

4. 捏造（Forgery）：一個未經過合法授權的第三者擅自偽造系統中欄資料。這是針對資料的確實性加以攻擊。

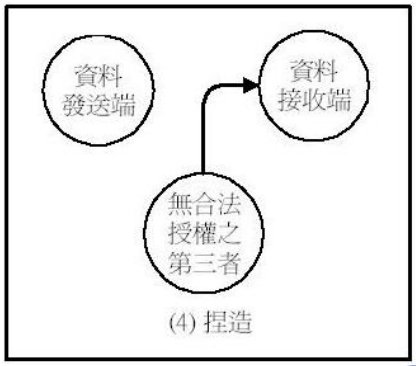


圖 3-4：攻擊手段之捏造（Forgery）

此外，攻擊也可以利用攻擊的目的而加以分類成主動式攻擊（Active Attack）與被動式攻擊（Passive Attack）兩種。被動式攻擊法的目的，只是為了竊取通訊中的資訊內容，並不破壞通訊本身，常見的被動式攻擊如竊聽、流量分析等手法皆是如此，由於被動式攻擊法並不破壞通訊本身，因此相對而言，參與通訊的雙方難以檢測通訊是否遭到被動式攻擊。另一方面，主動式攻擊的目的就不像被動式攻擊那般單純了，主動式的攻擊法常常造成通訊系統的破壞，無論是對於通訊者，或是通訊中的資訊等等，皆有可能遭受損壞，常見的主動式攻擊手法如阻礙、修改、偽裝(Masquerade)、重播(Replay)、DoS(Denial of Service)等皆是如此。

3.2 類型漏洞(Type Flaw)的分析

類型漏洞是近代一個常見的漏洞，它利用了加密訊息內，沒有資料型態宣告的情況下，加以混亂視聽，而攻擊者可以利用其它已知的訊息，取代重要的訊息，例如利用密文內，且在明文有出現過的訊息，利用這段密文來取代有重要訊息的密文，例如 Session Key(兩者通訊時所建立之金鑰)，可以使別人誤認假的訊息是真的 Session Key，進而利用這段假的的 Session Key 和別人通訊。

例如以下例子(圖 3-5)：它是一個任何人與任何人要建立起安全通訊管道的通訊協定，這協定先假設有共享金鑰的密碼系統，每一個使用者有他專用的金鑰，而這把金鑰和證明伺服器 S 共享

1. A->B:M,A,B,{Na,M,A,B}Kas
2. B->S:M,A,B,{Na,M,A,B}Kas,{Nb,M,A,B}Kbs
3. S->B:M,{Na,Kab}Kas,{Nb,kab}Kbs
4. B->A:M,{Na,Kab}Kas

圖 3-5：The Otway-Rees protocol

在這個例子中，A 先傳給 B 訊息 M, A, B, 及密文{Na, M, A, B}Kas，告訴 B 說我要和你做通訊;然後 B 再傳給 S 訊息 M, A, B, 密文{Na, M, A, B}kas 及密文 {Nb, M, A, B}kbs，告訴 S 說 A 和 B 要做通訊，請製造出一把讓 A 和 B 通訊的 Session Key;再來 S 回傳給 B 訊息 M,密文{Na, Kab}kas,及密文{Nb, kab}kbs，告訴 B 你可以拿到一把 Session Key，並且要 B 將 Session Key 也傳給 A;最後 B 傳給 A 訊息 M,及密文{Na, Kab}kas，如此一來，A 就可以拿到和 B 做通訊的 Session Key，之後便利用此 Session Key 加密所有要和 B 通訊的訊息。

而這個例子有很嚴重的類型漏洞，在第一行中，A 送出訊息給 B，到最後第四行，A 收到了送回來的訊息，送出的訊息 M,A,B,{Na,M,A,B}kas，可以被攻擊者截取，然後在第四步驟，攻擊者假冒 B 送回給 A 訊息 M,{Na,M,A,B}kas，如此

一來，這個通訊協定使得 A 誤以為 M,A,B 是 Kab，因此攻擊者可以利用 M,A,B 當成 Session Key 和 A 做通訊，使得原本 A 要和 B 做通訊，變成了 A 和攻擊者做通訊，告成了嚴重的漏洞。

3.3 新鮮度漏洞(Freshness Flaw)的分析

新鮮度漏洞也是近代常見的一個網路安全問題，這個漏洞加密訊息中缺乏資料的新鮮度，攻擊者可以利用這個漏洞截取到有用的訊息，若沒有新鮮度性質的訊息，攻擊者可以理論上的無限制地利用這段訊息，也許可以用密碼翻譯法推出密文中的明文，這將造成嚴重的傷害。

在文中提到一個相關例子(圖 3-6)[13]，Needham-Schroeder protocol，它是被 Denning 和 Sacco 所發現[14]，以及被 Bauer, Berson 和 Feiertag 所討論[3]，它是一個任何人與任何人要建立起安全通訊管道的通訊協定，這協定先假設了有共享金鑰的密碼系統，每一個使用者有他專用的金鑰，而這把金鑰和證明伺服器 S 共享。

1. A->S: A,B,Na
2. S->A:{Na,B,Kab,{Kab,A}Kbs}Kas
3. A->B:{Kab,A}Kbs
4. B->A:{Nb}Kab
5. A->B:{Nb-1}Kab

圖 3-6：The Needham-Schroeder protocol

這個訊息 {Kab,A}Kbs 在第 3 步驟從 A 傳送給 B，沒有包含任何它的生命週期，攻擊者可以截取這段訊息，從容不迫的利用密碼翻譯法找尋 Session Key Kab，或者利用其它管道得到了舊的 Kab，之後便可以利用{Kab,A}Kbs 及 Kab ，來進行騙 B 的動作。

3.4 第三者漏洞(Third-Part Flaw)分析

第三者漏洞是一種在 A 與 B 要做通訊之前，A 先和 C 做了通訊，且 A 和 C 的 Session Key 被 C 給公佈了出來，造成了 A 與其它人通訊產生漏洞。以下我們用 Otway-Rees protocol(by Burrows)[15]來做說明(圖 3-7)，它是一個任何人與任何人要建立起安全通訊管道的通訊協定，這協定先假設了有共享金鑰的密碼系統，每一個使用者有他專用的金鑰，而這把金鑰和證明伺服器 S 共享。

1. A->B: Na,A,B,{Na,A,B}Ka

2. B->S: Na,A,B,{Na,A,B}Ka,Nb,{Na,A,B}Kb

3. S->B: Na,{Na,Kab}Ka,{Nb,Kab}Kb

4. B->A: Na,{Na,Kab}Ka

圖 3-7：Otway-Ress protocol

1. A->Cb: Na,A,B,{Na,A,B}Ka

A->Cb 表示 A 傳送訊息給 B，同時 C 也攔截到此訊息

1‘.C->A: Nc,C,A,{Nc,C,A}Kc

A 試著和 B 通訊，然而，C 截取了這訊息並且重新開始一次協定的運作，和 A 作通訊

2‘. A->Cs: Nc,C,A,{Nc,C,A}Kc,Na‘,{Nc,C,A}Ka

A->Cs 表示 A 傳送訊息給 S，同時 C 也攔截到此訊息

2‘‘. Ca->S: Nc,C,A,{Nc,C,A}Kc,Na,{Nc,C,A}Ka

A 回應了 C’ 的訊息向 Server 要求訊息，但是 C 改變了他的訊息，將變數 Na’ 改成了 Na。

3‘. S->Ca: Nc,{Nc,Kca}Kc,{Na,Kca}Ka

S->Ca 表示 S 傳送訊息給 A，同時 C 也攔截到此訊息

在回應修改過後的訊息，S 回應了 A 一把和 C 做通訊用的 Session Key

4. Cb->A: Na,{Na,Kca}Ka

C 捉到了兩個認證且傳給 A 錯的一個，密文內有 Na,所以 A 接受了 Kca 當成了和 B 第一次跑協定的 Session Key Kab，但是這把 Key 是和 C 共享的。

四、網路安全協定模擬器之設計

通訊協定設計師在進行實驗及設計一個通訊協定時,大多對安全協定模擬器有相當的興趣，這篇論文主要是設計及實現一個安全協定模擬器，以方便通訊協定設計師可以設計模擬安全協定之各個步驟,以下是這篇論文程式的主架構流程圖。

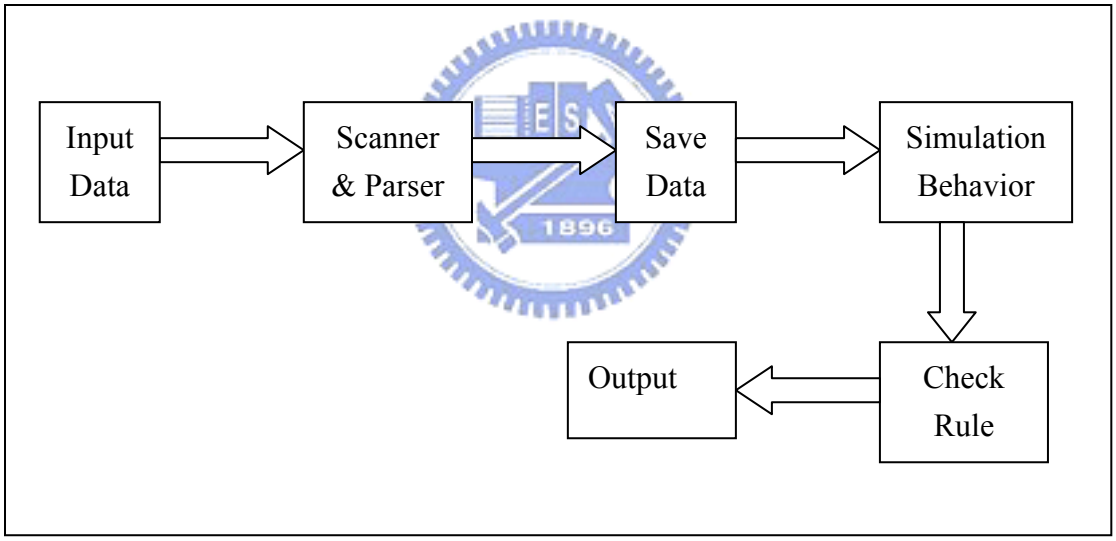


圖 4-1：系統架構流程圖

首先將協定當成資料輸入，檔案格式如下，存成文字檔，再來利用編譯器技術中的 Scanner 及 Parser，來判斷所輸入的資料是否合乎協定的格式，例如下面的例子，經過了 Scanner 及 Parser 後，將它存入串列這種資料結構，再來模擬這協定的行為，及加入檢查法則來試試看這協定是否會出現漏洞。

例一、(符合要求)

A->B:M,A,B,{Na,M,A,B}kas

B->S:M,A,B,{Na,M,A,B}kas,{Nb,M,A,B}kbs

S->B:M,{Na,Kab}kas,{Nb,kab}kbs

B->A:M,{Na,Kab}kas

例二、(不符合要求)

A-->>B::M,A,B,{Na,M,A,B}kas

B->S:{M,A,B,{Na,M,A,B}kas,{Nb,M,A,B}kbs}

S->B:M,{{Na,Kab}kas,{Nb,kab}kbs

B->A:M,{Na,Kab}}kas

因為 A-->>B::M,A,B,{Na,M,A,B}kas 中，應該是->符號，而不是 -->> 符號。B->S:{M,A,B,{Na,M,A,B}kas,{Nb,M,A,B}kbs}不符合要求，因為左括號個數必須和右括號個數一致。S->B:M,{{Na,Kab}kas,{Nb,kab}kbs 和 B->A:M,{Na,Kab}}kas 也是相同問題，必須左括號個數必須和右括號個數一致。

4.1 利用 Lex 及 Yacc 來掃描及剖析輸入資料

Lex 與 Yacc 是給編譯器與直譯器程式設計師用的工具，也可以用在其他與編譯器無關的程式上。任何需要在輸入資料尋找特定樣式或是輸入的資料是某種語言的程式，都可以考慮使用 Lex 與 Yacc。此外，使用這些工具可以讓你快速的寫出原型程式，在修改與維護上都相當方便。

Lex 與 Yacc 都是 1970 年代在貝爾實驗室發展出來的。MS-DOS 與 OS/2 上有兩個 Lex 與 Yacc 的版本，一個是由 MKS（Mortice Kern Systems 公司），也是 MKS Toolkit 的發行者所提供的的版本。這是一個獨立的產品，而且支援許多 PC 上的 C 語言編譯器; MKS 版的 Lex 與 Yacc 還附有完整的技術手冊。Abraxas 公司發行的 PCYACC 附有許多目前通行的程式語言的簡單解析器。

我們使用 Parser Generation(Pargen) 這個工具來發展我們的論文，且使用 Lex 和 Yacc 來定義此論文輸入資料的記號及句法。

記號如下(Token):

```
[a-z,A-Z][0-9,a-z,A-Z]*  {return ID;}
"\n"                     { return NL;}
"->"                     {return ARROW;}
":"                      {return COLON;}
"{"                      {return ML;}
"}"                      {return MR;}
","                      {return MID;}
```

圖 4-2：系統 Scanner 之記號

句法如下(Syntax):

```
input    : line | input line  ;
line     : ID ARROW ID COLON messageK
         | ID ARROW ID COLON messageK NL;
messageK: message
         | message ML messageK MR ID
         | message ML messageK MR ID MID
         | ML messageK MR ID
```

```
        | message MID ML messageK MR ID
        | ML messageK MR ID MID ML messageK MR ID
        | message MID ML messageK MR ID MID ML messageK MR ID;
message : ID
        | message MID ID;
```

圖 4-3：系統 Parser 之句法

4.2 使用一般串列儲存資料(Generalized List)

每一個點的資料型類宣告如下:

```
    char sender;                sender
    char receiver;              receiver
    int counter;                recode the number of messages in the node
    AnsiString message[10];     the context of each node
    AnsiString key;             recode key value
    int line_no;                Protocol 的第幾行
    struct protocol *link;      link to next node
    struct protocol *sublink;   link to a node with ciphertext
```

圖 4-4：系統 Node 之 Data Type

例一:

A->S: A, B, Na

S->A: Na, B, {Kab, A, B}Kas

A->B: {Kab, A}Kbs

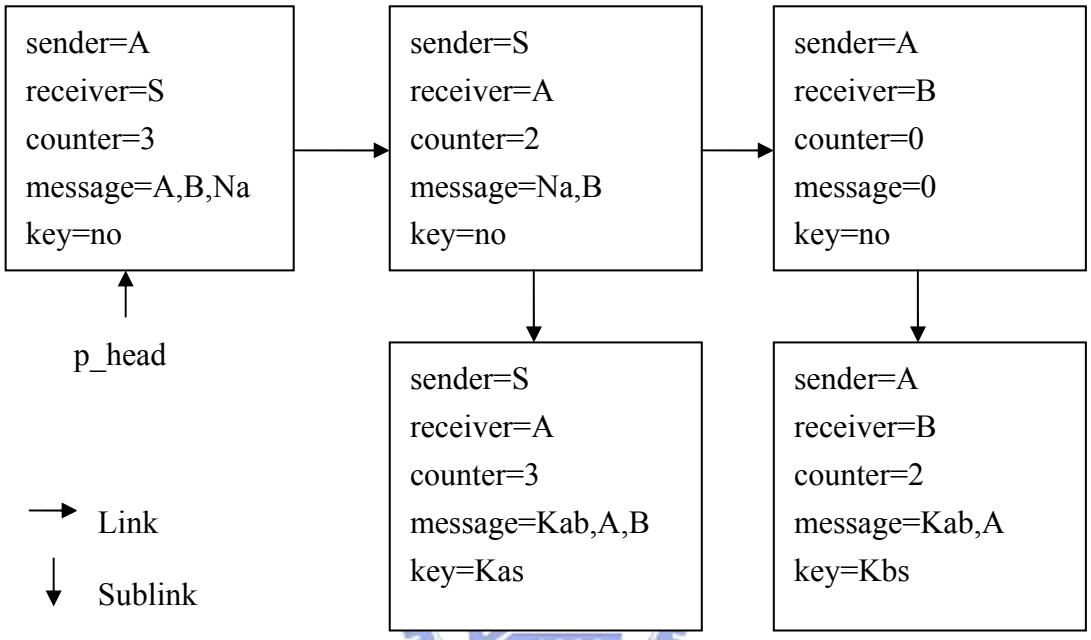


圖 4-5：資料結構使用一般串列例 1

例二:

A->S: A, B, Na

S->A: Na, B, {Kab, A, B, {Kab, A}Kbs }Kas

A->B: Nb, A

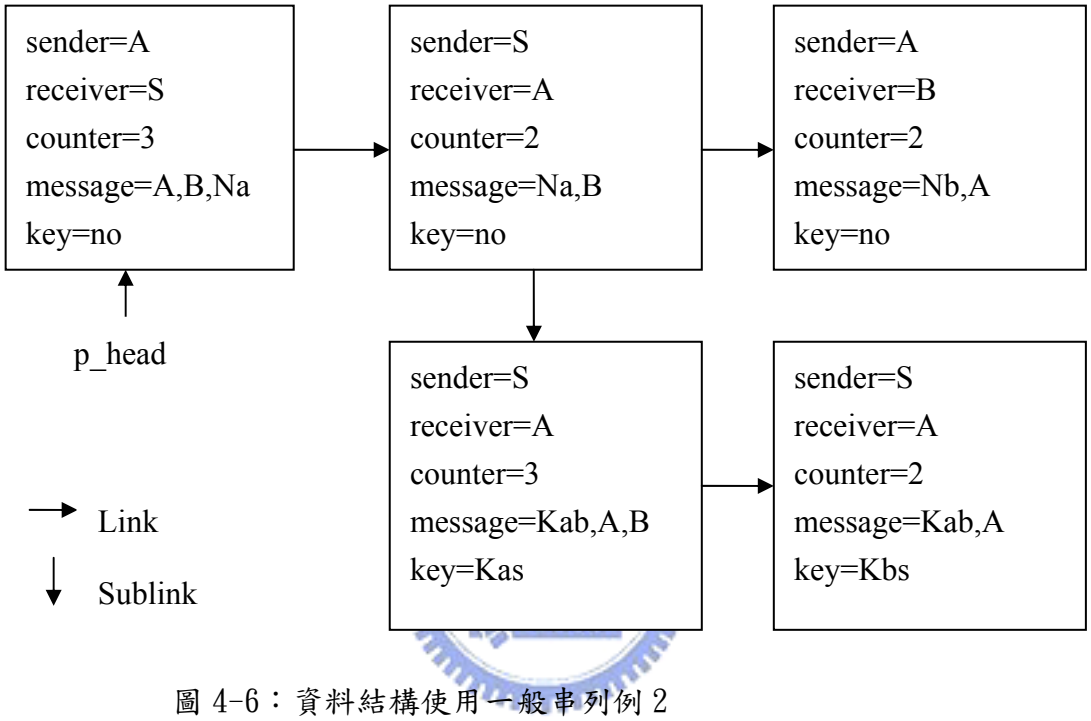


圖 4-6：資料結構使用一般串列例 2

例三:

A->S: A, B, Na

S->A: Na, B, {Kab, A, B, {Kab, A}Kbs }Kas, {Kab, A, B}Kcs

A->B: Nb, A

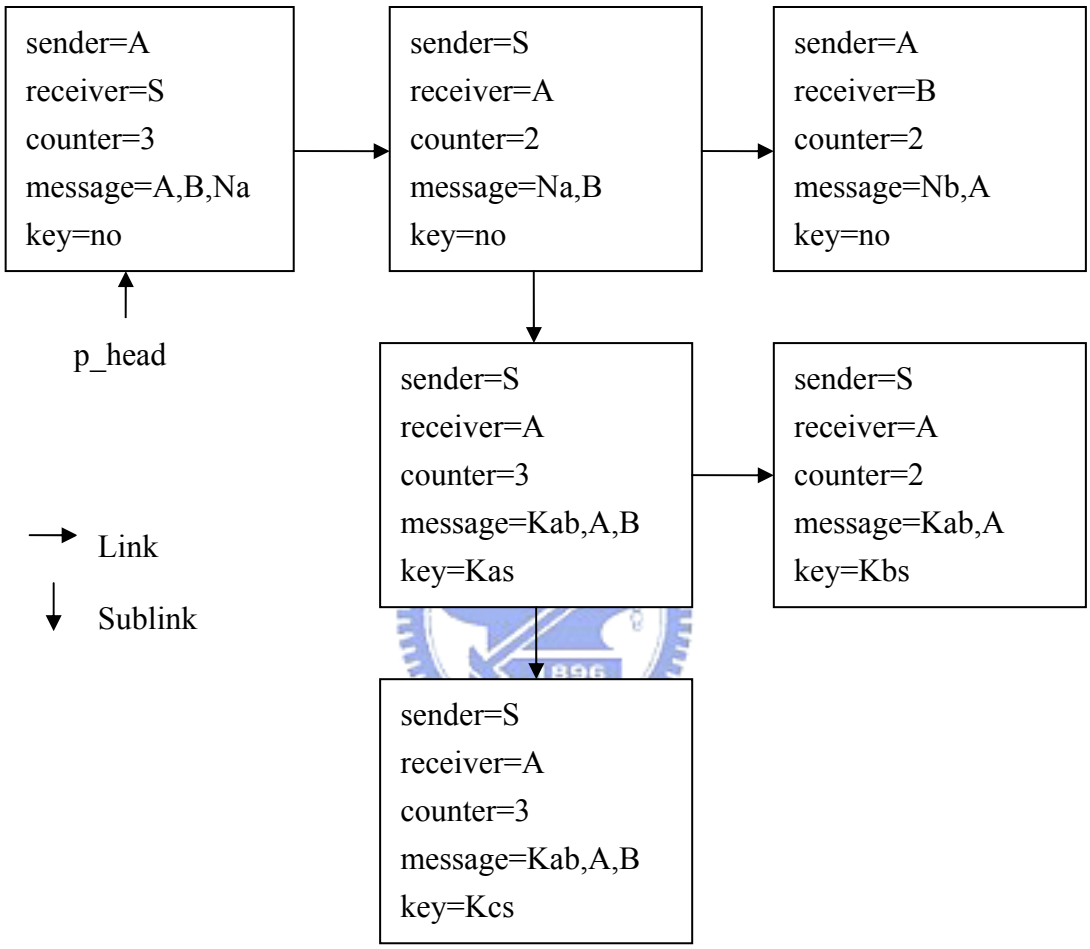


圖 4-7：資料結構使用一般串列例 3

4.3 模擬安全協定的行為

在網路中，通訊協定的格式固定了，但是畢竟他也只是一些文字的描述而已，例如：

1. A->B:M,A,B,{Na,M,A,B}kas
2. B->S:M,A,B,{Na,M,A,B}kas,{Nb,M,A,B}kbs
3. S->B:M,{Na,Kab}kas,{Nb,kab}kbs
4. B->A:M,{Na,Kab}kas

因此我們在此將輸入格式，做一些網路上行為模擬，首先利用 IP_mapping()這個函數，將 A、B、S 等目標物轉換成網路上的獨特 ID，也就是 IP 位址，例如見到了 A,便將 A 改成 www.aaa.com.tw,這個對映表可以依程式設計師設定做修改。

message_mapping()這個函數是將 A、B、S 等傳送的訊息做一對映，例如 Na 即傳回一個變數的值，而此變數在同一次協定內是同一值，但在另一次執行協定則會變成另一值，達到隨機變數的真正目的。而 key_maker()是產生 key 鍵值的一個函數，若是 Ka,Kb 等則給定一個固定值，若是 Kab,Kbc 等，則隨機產生 key 鍵值。random_maker()則是用亂數產生鍵值的函數。

例如：

A->S:A,B,Na

S->A:{Na,B,Kab,{Kab,A}Kbs}Kas

A->B:{Kab,A}Kbs

B->A:{Nb}Kab

A->B:{Nb-1}Kab

以下是我們安全協定模擬器真正執行時的畫面，稱之為模擬視窗。在模擬視窗內之 A、B、S 分別代表了協定內 sender A、receiver B、server S 之各個運作，而 Hacker 部皆記錄了 Adversary 可以收集到的資訊。

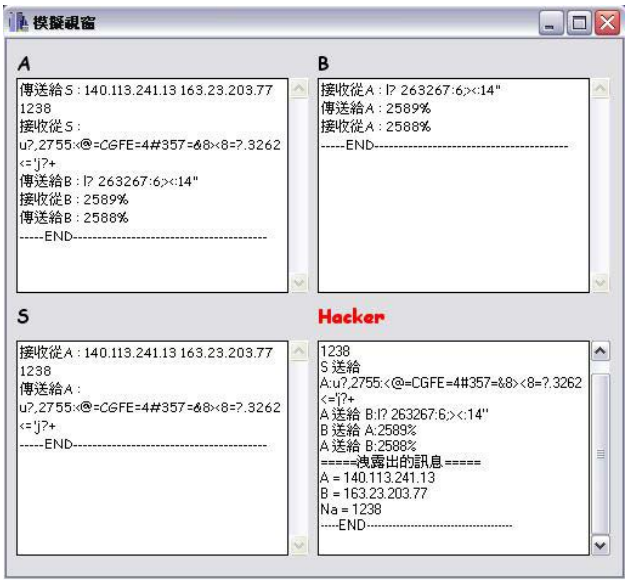


圖 4-8：系統模擬視窗

4.4 定義檢測法則的格式

在定義檢測法則的格式中，我們引入了物件導向技術為基礎，所以在日後新增或刪除檢測法則時，不用改寫程式的主體，只須改寫程式中的一小部分，在這篇論文中，建議 designer 只須改寫 RULE.CPP 這個檔案，提供 designer 可以隨時新增 pattern，現在分別介紹這篇論文中主要的三大檢測法則(pattern)。

4.4.1 類型漏洞的分析

從之前的例子中，A 先傳給 B 訊息 M, A, B, 及密文{Na, M, A, B}Kas，然後 B 再傳給 S 訊息 M, A, B, 密文{Na, M, A, B}kas 及密文 {Nb, M, A, B}kbs，再來 S 回傳給 B 訊息 M, 密文{Na, Kab}kas, 及密文{Nb, kab}kbs，最後 B 傳給 A M, 及密文{Na, Kab}kas，如此一來，A 就可以拿到和 B 做通訊的 Session Key，之後便利用此 Session Key 加密所有要和 B 通訊的訊息，而這個例子有類型漏洞，在第一行中，A 送出訊息給 B，到最後第四行，A 收到了送回來的訊息，送出的訊息 M,A,B,{Na,M,A,B}kas，可以被攻擊者截取，然後在第四步驟，攻擊者假冒 B 送回給 A 訊息 M,{Na,M,A,B}kas，如此一來，這個通訊協定使得 A 誤以為 M,A,B 是 Kab，因此攻擊者可以利用 M,A,B 當成 Session Key 和 A 做通訊，使得原本 A 要和 B 做通訊，變成了 A 和攻擊者做通訊，造成了嚴重的漏洞。

1. A->B:M,A,B,{Na,M,A,B}Kas

2. B->S:M,A,B,{Na,M,A,B}Kas,{Nb,M,A,B}Kbs

3. S->B:M,{Na,Kab}Kas,{Nb,kab}Kbs

4. B->A:M,{Na,Kab}Kas

我們發現類型漏洞有一些共同點，例如

1.有兩段在加密的密文中，用來加密的金鑰相同(如{Na,M,A,B}Kas 和{Na,Kab}Kas 之加密的金鑰相同)

2.且其中一段密文中必須要有 Session Key 的存在(如{Na,Kab}Kas 內有存在 Session Key Kab)

3.且兩段加密密文中有重複的出現(如{Na,M,A,B}kas 和{Na,Kab}kas 之 Na 相同)

4.且兩段加密密文中沒有重複出現的，必須在明文內有出現(如{Na,M,A,B}kas 內之 M,A,B 在明文中有出現)

當我們將協定當成輸入，經過程式的判斷，如果有符合以上四點，可以肯定這個協定可能會發生類型漏洞。

4.4.2 新鮮度漏洞的分析

在這類型協定分析中，{Kab,A}Kbs 在第 3 步驟從 A 傳送給 B，沒有包含任何 NONCE 或 Timestap，攻擊者可以截取這段訊息，從容不迫的利用密碼翻譯法找尋 Session Key-Kab，或者利用其它管道得到了舊的 Kab，之後便可以利用 {Kab,A}Kbs 及 Kab ，來進行騙 B 的動作。

1. A->S: A,B,Na
2. S->A:{Na,B,Kab,{Kab,A}Kbs}Kas
3. A->B:{Kab,A}Kbs
4. B->A:{Nb}Kab
5. A->B:{Nb-1}Kab

我們發現新鮮度漏洞有一些共同點，例如

1. 在加密的密文中，密文中的內容沒有 NONCE(Na,Nb,…)
2. 且在此密文中，有出現 Session Key

當我們將協定當成輸入，經過程式的判斷，如果有符合以上二點，可以肯定這個協定可能會發生新鮮度漏洞。

4.4.3 第三者漏洞分析

第三者漏洞我們用 Otway-Rees protocol 來做說明，協定如下：

1. A->B: Na,A,B,{Na,A,B}Ka

2. B->S: Na,A,B,{Na,A,B}Ka,Nb,{Na,A,B}Kb

3. S->B: Na,{Na,Kab}Ka,{Nb,Kab}Kb

4. B->A: Na,{Na,Kab}Ka

漏洞如下：

1. A->Cb: Na,A,B,{Na,A,B}Ka

1‘.C->A: Nc,C,A,{Nc,C,A}Kc

2‘. A->Cs: Nc,C,A,{Nc,C,A}Kc,Na‘,{Nc,C,A}Ka

2‘‘. Ca->S: Nc,C,A,{Nc,C,A}Kc,Na,{Nc,C,A}Ka

3‘. S->Ca: Nc,{Nc,Kca}Kc,{Na,Kca}Ka

4.Cb->A: Na,{Na,Kca}Ka

我們發現第三者漏洞有一些共同點，例如

1. 在所有的密文中，將所有的 a 改成 c，b 改成 a，和原來的密文比較，是否有密文會全部相同，除了 Kab 及 Kca 等 Session Key 不同，若有的話，這樣的訊息可能會被和第三者通訊時，第三者洩漏 Session Key 時產生危機。

五、實驗分析

5.1 操作介面介紹

我們使用 Borland C++ Builder 為發此系統的平台，首先先建立出一個方便的使用者介面，在左上角部分提供使用者選取安全協定資料的來源(圖 5-1)，一旦選取之後，所選取之資料將會顯示在左方的區域(圖 5-2)，我們先將『Parser』按鈕按下，系統將會開始 Scanner 及 Parser 所輸入的安全協定，並且判斷是否有錯，如果沒有錯的話，接下來將『Simulation』按鈕按下，系統將會自動的將網路安全協定儲存並顯示其內容(圖 5-3)，然後系統將模擬整個安全協定的運作(圖 5-4)及找出這個安全協定是否有漏洞的存在(圖 5-5)，若有發生漏洞的話，為了更詳細說明此漏洞的發生及其它例子，我們建立了一個網頁(圖 5-6)，以便更詳細的說明，更清楚的使用說明，詳見附錄一。

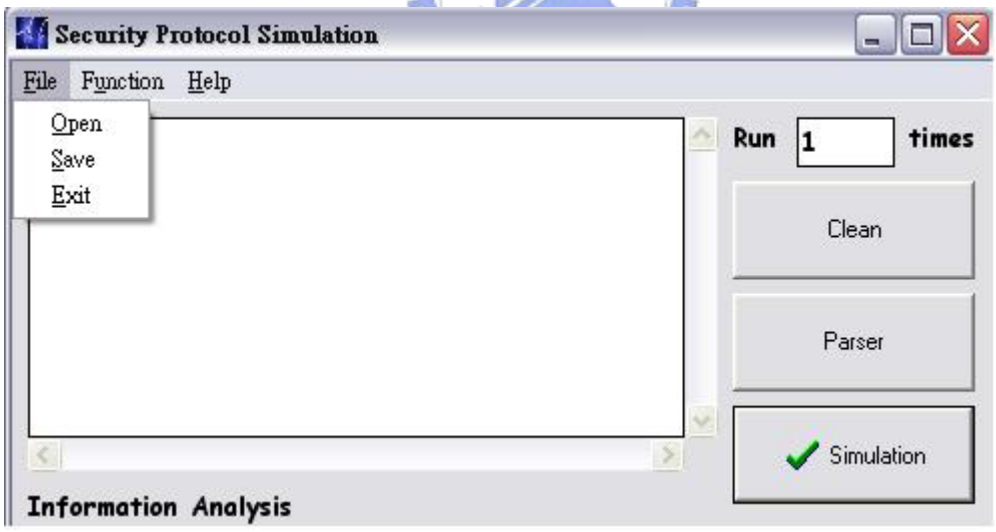


圖 5-1：系統初始介面

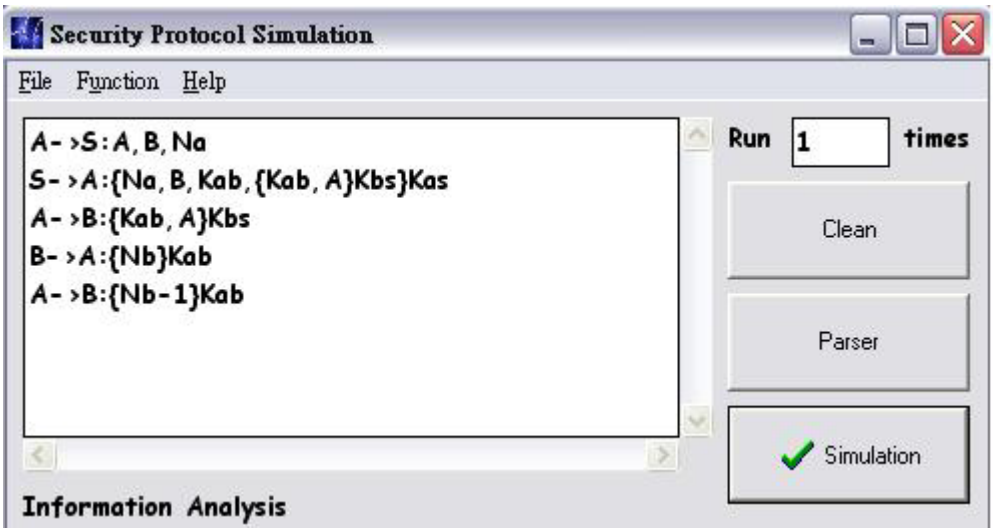


圖 5-2：系統資料輸入介面

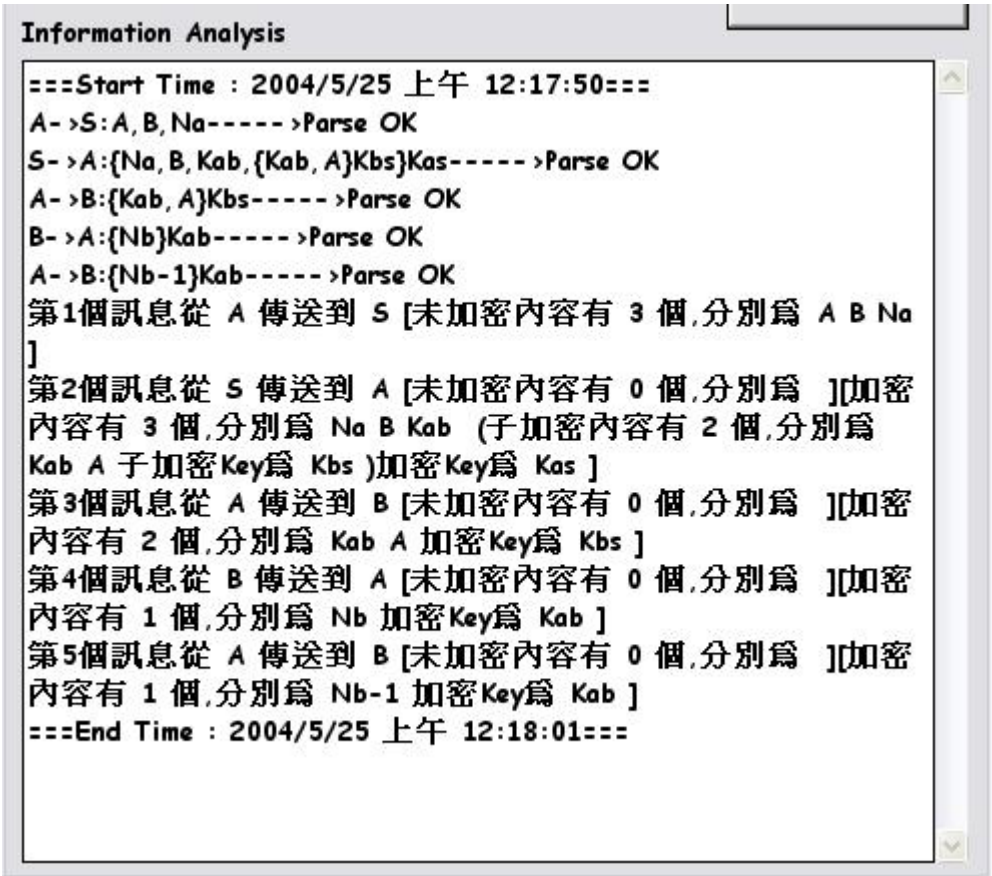


圖 5-3：系統資訊分析介面

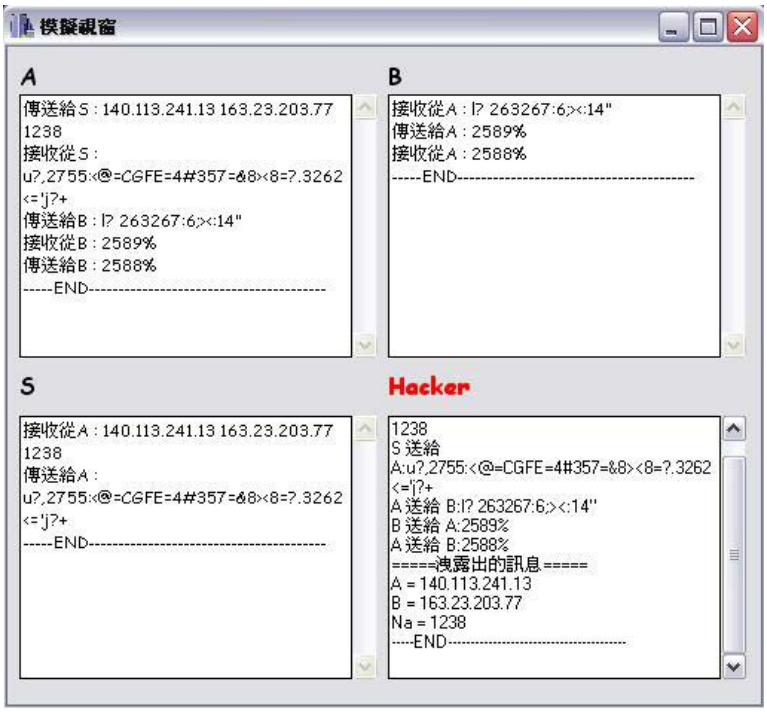


圖 5-4：系統資訊模擬介面

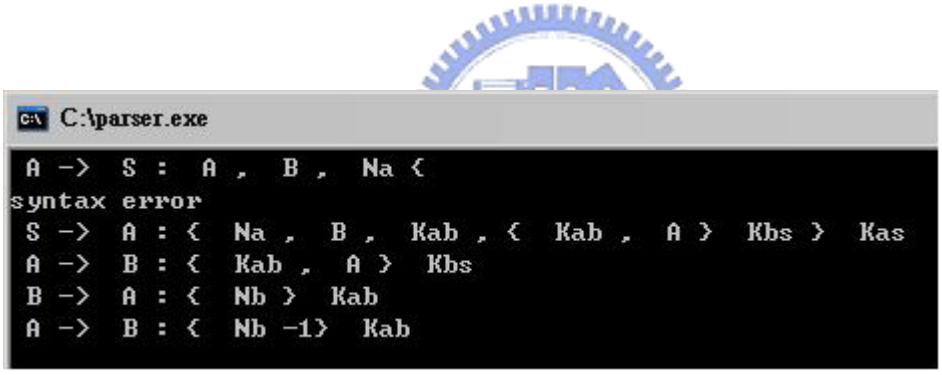


圖 5-5：系統 Parser 錯誤警告介面

圖 5-6：系統資訊警告介面

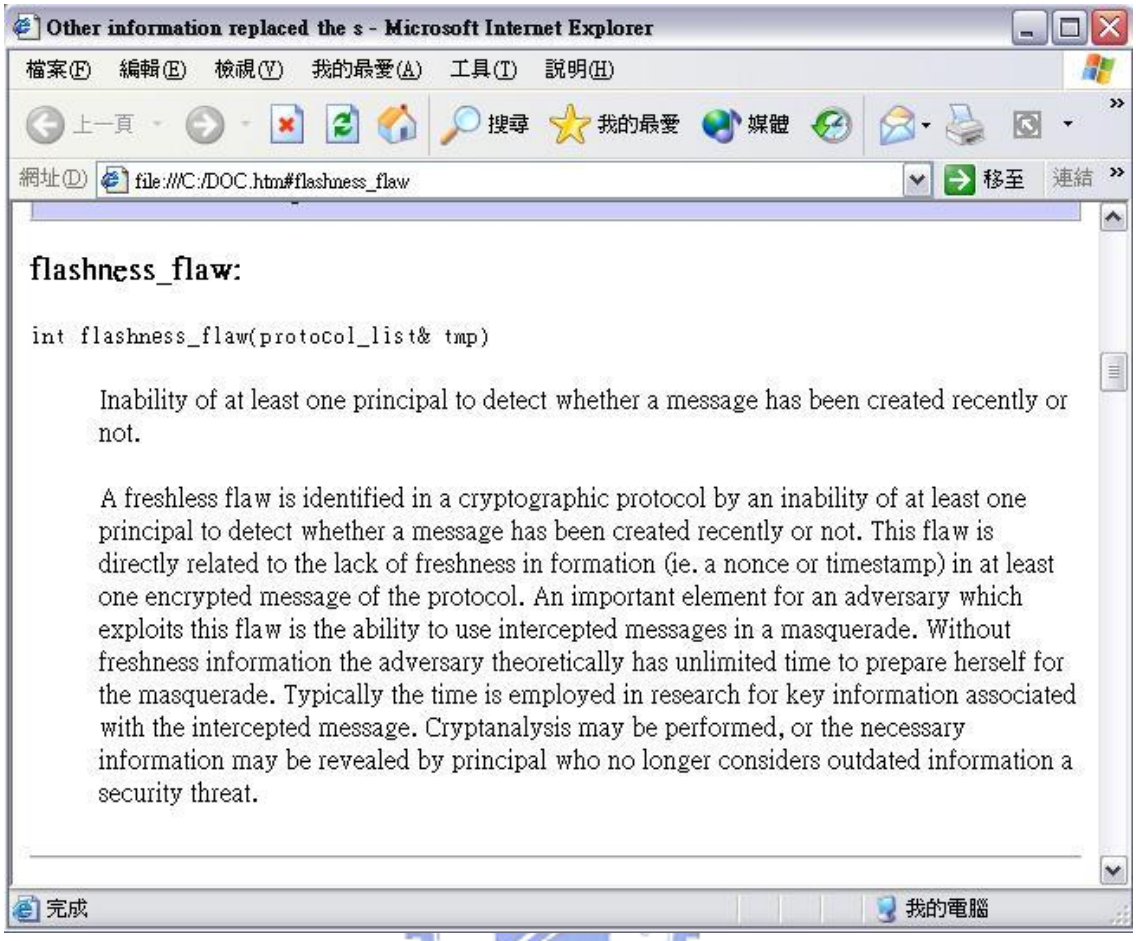


圖 5-7：系統資訊協助網頁

5.2 實驗結果

首先，我們收集了許多的網路安全協定，如附錄二，系統可以處理網路安全模型中(圖5-8)的所有網路安全協定，在這個模型中，參與安全通訊的雙方，我們稱之為參與者（Principal），而由一雙方都信認的、公證的第三者（Trusted Third Party）居中，負責安全通訊的一些必要及相關事宜。而資訊的發送端與接收端之間，我們以一條邏輯的（Logical）、概念性的通路，來表示雙方溝通所運用的管道。而網路安全所要面對的，即是攻擊者（adversary）於雙方通訊間所做的各種攻擊，並採取各種必要的解決措施。結驗結果顯示，整個系統跑完一個網路安全協定，大多在2秒以內完成。

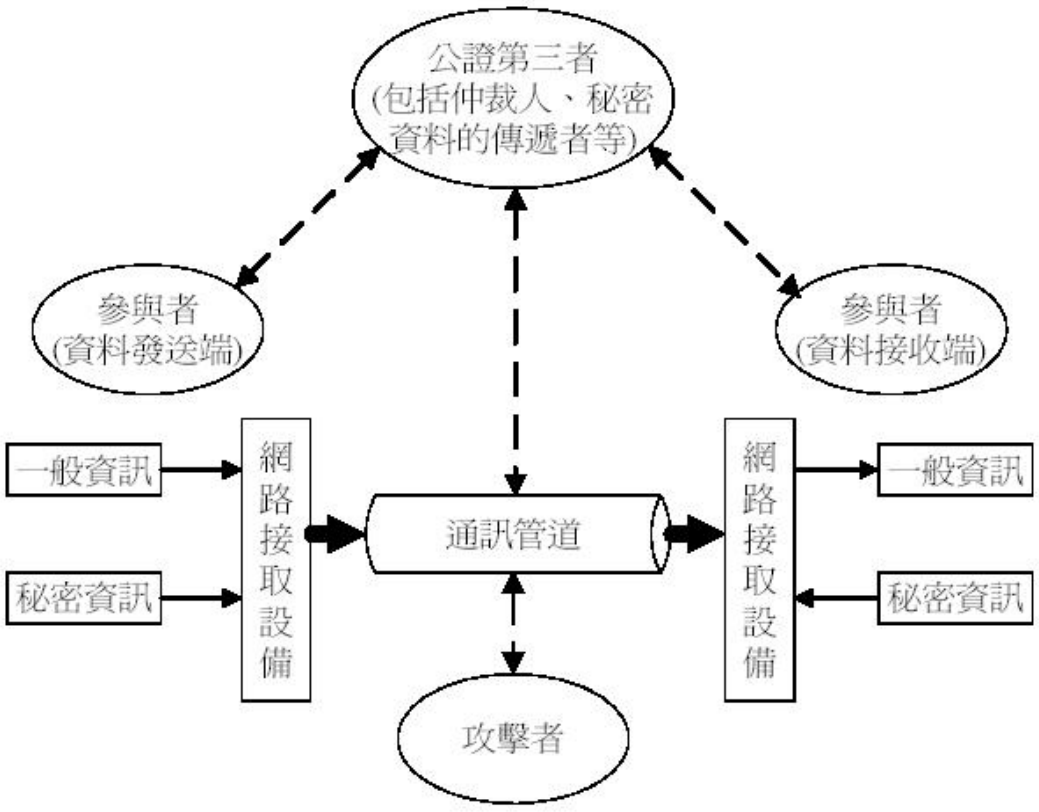


圖 5-8：網路安全示意圖

5.3 實驗比較

隨著科技的發達，時間的流逝，愈來愈多的新型的攻擊不斷的產生，當然在網路安全漏洞也愈來愈多。然而在這個判斷網路安全協定是否有漏洞，以及漏洞在哪裡這領域中，曾有許多人做過相關的研究，有人利用了安全協定的Inductive Model[10]來證明網路安全協定的正確性，也有人利用了數學邏輯的方法[11]，來驗證和測試網路安全協定，更有人利用了Strand Space[8]，來說明了為什麼這個網路安全協定是正確的，其中還有人針對了單一個類型漏洞(Type Flaw)做出了解決方案[9]及驗證方法[16]，以上提出的方法重於理論的證明，而我們所提出的論文，則是用於程式語言的角度來著手，建立了一個有彈性(extensible)的安全協定模擬器。

六、結論及未來展望

在這篇論文裡，提出了一個強韌而有彈性的安全協定偵測方法，在安全協定經過 Scanner 及 Parser 處理之後，若沒有錯誤的話，且再藉由安全檢測法則(pattern)的分析之後，我們認定為較安全的協定。我們研究了安全協定中，各種型態產生漏洞的可能性，並且利用程式語言幫我們找出可能的漏洞，因此，能很方便的協助安全協定設計師設計較無錯誤的安全協定。

在安全檢測法則(pattern)的分析之後，我們認定若符合 pattern 的條件之下，很有可能產生 pattern 下所描述的 Flaw，但若不符合 pattern 的條件，就一定不可能產生 pattern 下所描述的 Flaw 。

我們的安全協定模擬的安全性方面，是建立在安全檢測法則上，因為網路安全協定上的漏洞千奇百怪，我們必須要利用文獻的記載或經驗的累積，建立許多的檢測法則，就像是掃毒軟體一樣，每一年所產生的病毒都不同，掃毒軟體必需要不斷的增加一些病毒碼的道理一樣，因此我們建立了一個 extensible 之 Security Protocol Simulator。

在未來研究方面，有幾點是我們要繼續加強的，第一點，可以陸續研究新的安全協定漏洞並加入系統內，使整個系統更為著壯。第二點，若有人持續做此方面的論文的話，我們估計一年內可以增加 3~5 個安全檢測法則(patterns)，也許更多。

參考文獻

[1] An improved e-mail security protocol

Schneier, B.; Hall, C. Computer Security Applications Conference, 1997. Proceedings., 13th Annual , 1997

Page(s): 227 -230

[2] Security protocols in the Internet new framework

Sierra, J.M.; Ribagorda, A.; Munoz, A.; Jayaram, N.

Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on , 1999

Page(s): 311 -317

[3] R.K. Bauer, T.A. Berson, and R.J. Feiertag. A Key Distribution Protocol using Event Markers. ACM Transactions on Computer Systems,1(3):249-255,August 1983.

[4] An active security protocol against DoS attacks

Cotroneo, D.; Peluso, L.; Romano, S.P.; Ventre, G.;

Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on , 1-4 July 2002

Pages:496 - 501

[5] Controlling applets' behavior in a browser

Hassler, V.; Then, O.;

Computer Security Applications Conference, 1998, Proceedings., 14th Annual , 7-11 Dec 1998

Page(s): 120 -125

[6] Blocking Java applets at the firewall

Martin, D.M., Jr.; Rajagopalan, S.; Rubin, A.D.;

Network and Distributed System Security, 1997. Proceedings., 1997 Symposium on , 10-11 Feb 1997 Page(s): 16 -26

[7] Cryptographic protocol flaws: know your enemy

*Carlsen, U.;*Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings , 14-16 June 1994 Pages:192 - 200

[8] Strand spaces: why is a security protocol correct?

Fabrega, F.J.T.; Herzog, J.C.; Guttman, J.D.;

Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on , 3-6 May 1998 Pages:160 - 171

[9] How to Prevent Type Flaw Attacks on Security Protocols

Heather, J.; Lowe, G.; Schneider, S.;

Computer Security Foundations Workshop, 2000. CSFW-13. Proceedings. 13th IEEE , 3-5 July 2000

Page(s): 255 -268

[10] Proving security protocols correct

*Paulson, L.C.;*Logic in Computer Science, 1999. Proceedings. 14th Symposium on , 2-5 July 1999 Pages:370 - 381

[11] A logic approach to the verification and testing of security protocols,''IASTED International Conference on Commuincations and Computer Networks (CCN 2002)(Cambridge, MA, November 4-6), 140-145, 2002.

[12] Development of authentication protocols: some misconceptions and a new approach

*Wenbo Mao; Boyd, C.;*Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings , 14-16 June 1994 Pages:178 - 186

[13] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentiaction in Large Networks of Computers. Communications of the ACM, 21(12):993-999, December 1978.

[14] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. Communications of the ACM, 24(8):533-536, August 1981.

[15] M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. Preceedings of the Royal Society of London, 426:233-271,1989.

[16] Formal verification of type flaw attacks in security protocols

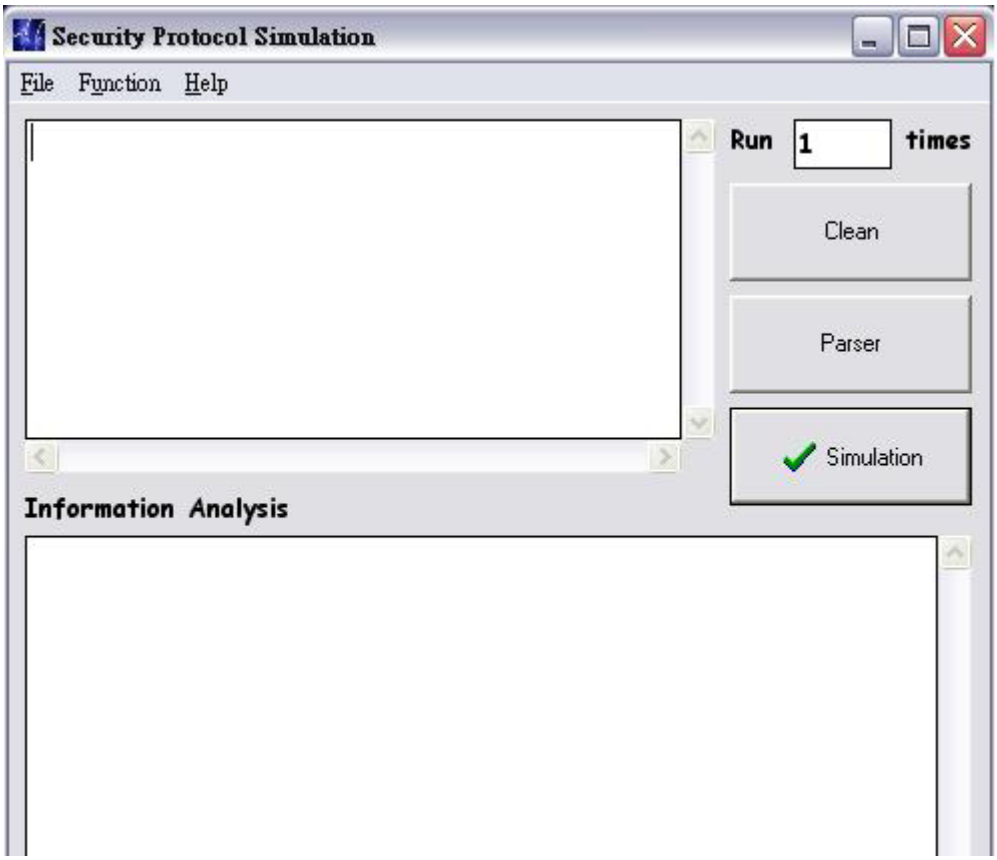
Long, B.W.;

Software Engineering Conference, 2003. Tenth Asia-Pacific , 2003

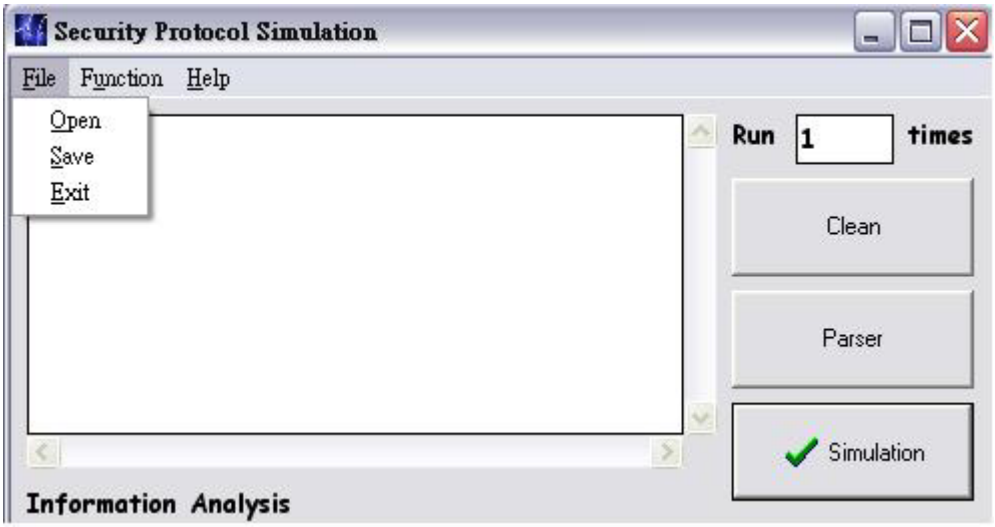
Pages:415 - 424

附錄一(SPS 操作方法指引及範例)

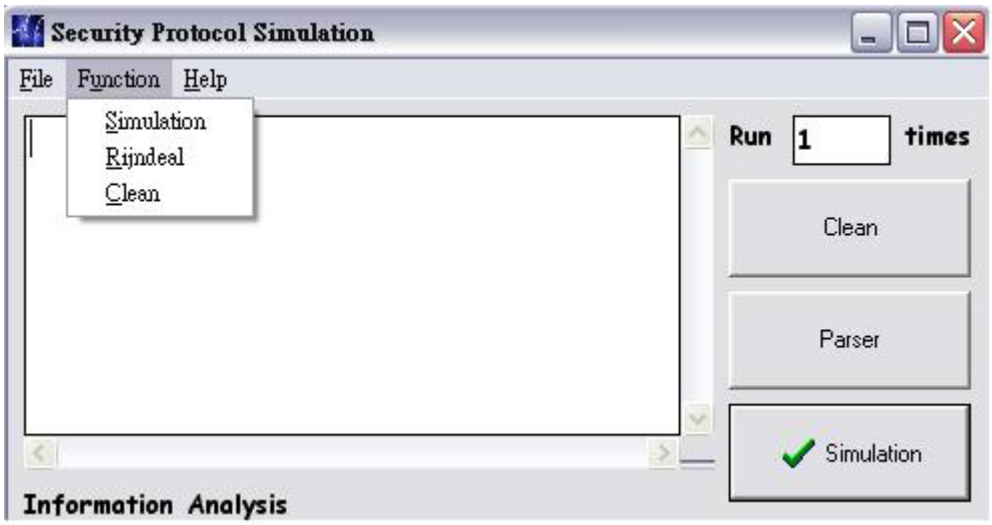
主程式畫面：



功能列上之 File 內有『Open』、『Save』、『Exit』，其中 Open 是開始檔案，Save 是儲存檔，而 Exit 是離開主程式。



功能列上之 Function 內有『Simulation』、『Rijndeal』、『Clean』，其中 Simulation 是開始模擬程式，Rijndeal 是測試加密函數，而 Clean 是清除主程式上的模擬畫面。



右邊有可以改變模擬次數的 Edit 欄位以及三個 Button，其中『Clean』是清除主程式上的模擬畫面，『Parser』是 Scanner 及 Parser 主程式所接收的輸入資料，而『Simulation』是開始模擬程式。

例如：

輸入是 Needham-Schroeder protocol，內容如下

A->S:A,B,Na

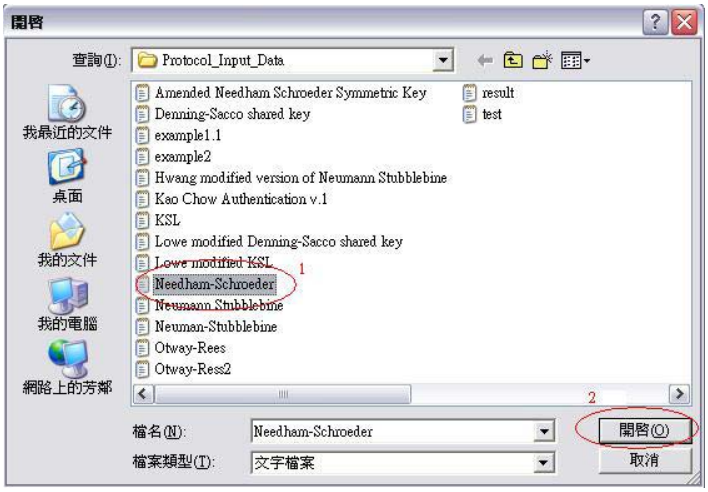
S->A:{Na,B,Kab,{Kab,A}Kbs}Kas

A->B:{Kab,A}Kbs

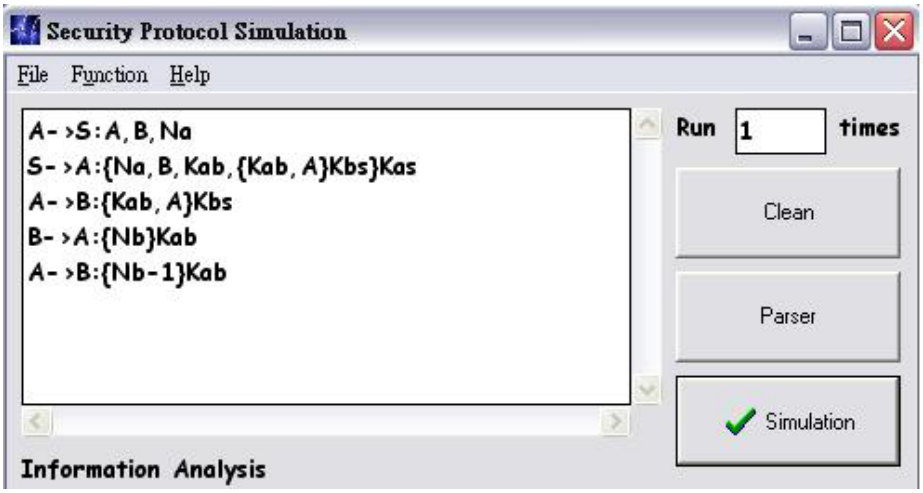
B->A:{Nb}Kab

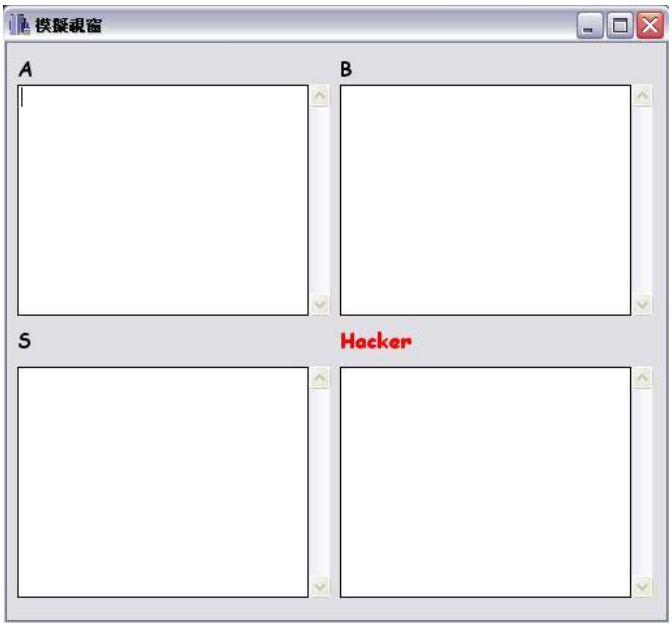
A->B:{Nb-1}Kab

則我們必須開啟檔案，按『File』『Open』，則出現以下畫面，並點選所要的檔案，接下來按開啟，即可出現「模擬視窗」。

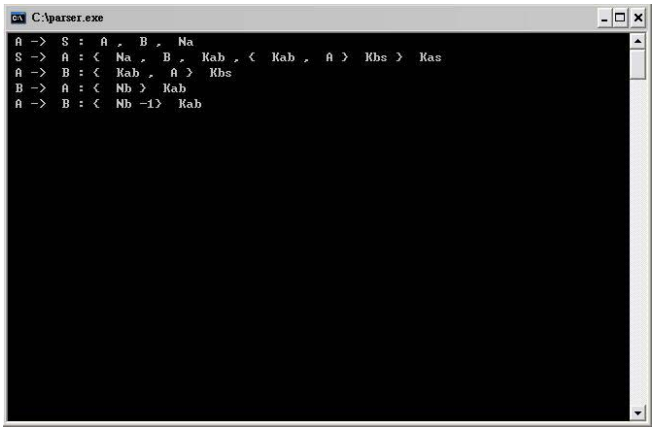


再來主程出現了要模擬的協定內容及模擬視窗，接下來我們先 Parser 這個輸入的協定是否有錯，則我們按下了 Parser 這個 Button。

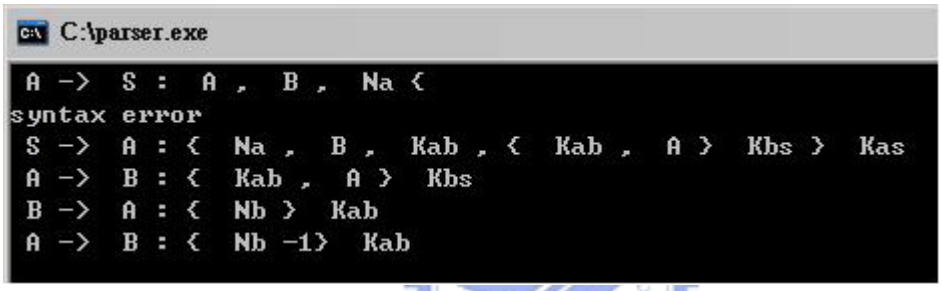




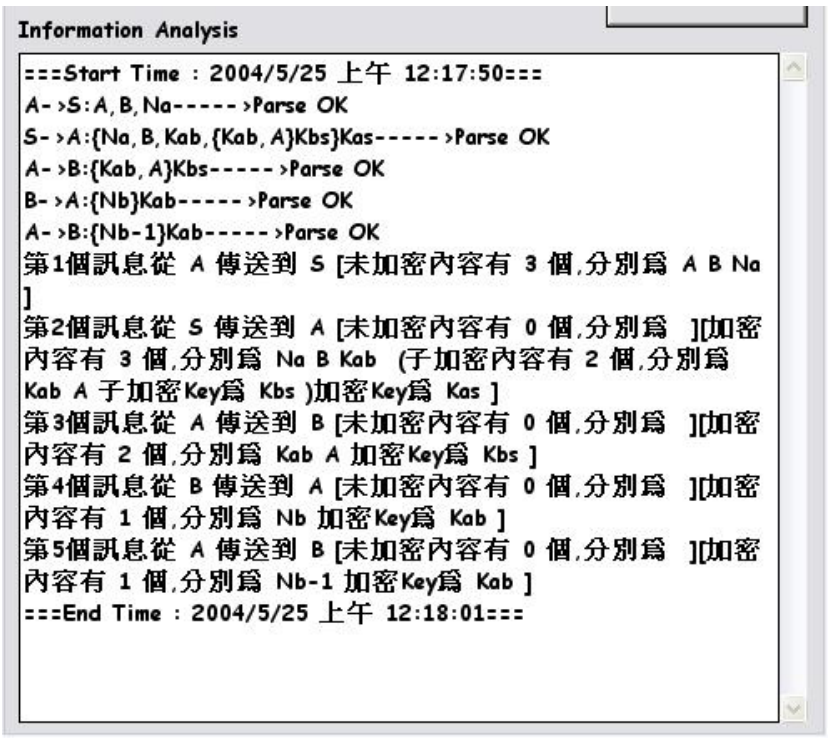
畫面會出現完整的安全協定的 DOS 畫面，表示這個輸入的語法沒有錯誤。



若在第一行輸入加一個錯誤的輸入，例如有左括號而沒有右括號，則會出現的畫面如下，會有出現 syntax error。

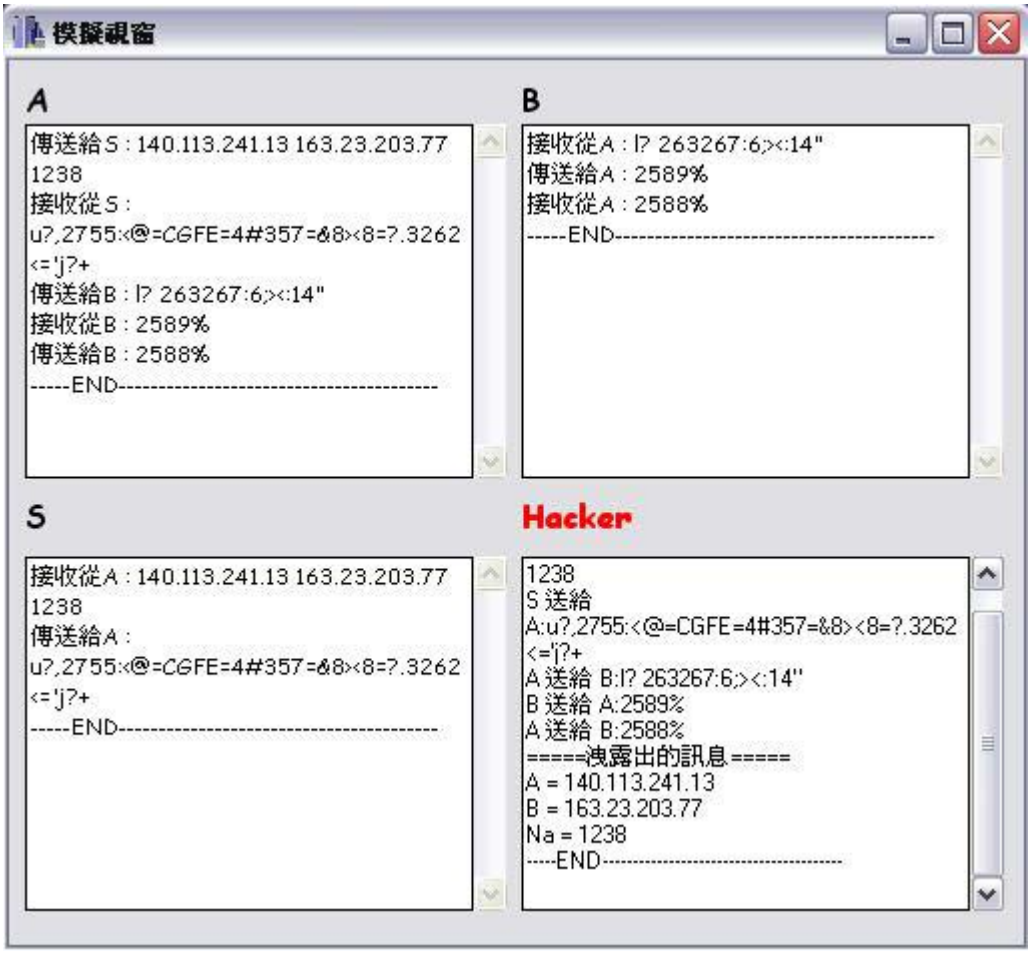


再來就直接按下『Simulation』這個 Button，則主程式開始一連串的模擬行為。可以看到資訊分析(Information Analysis)及模擬視窗的畫面，分別如下。



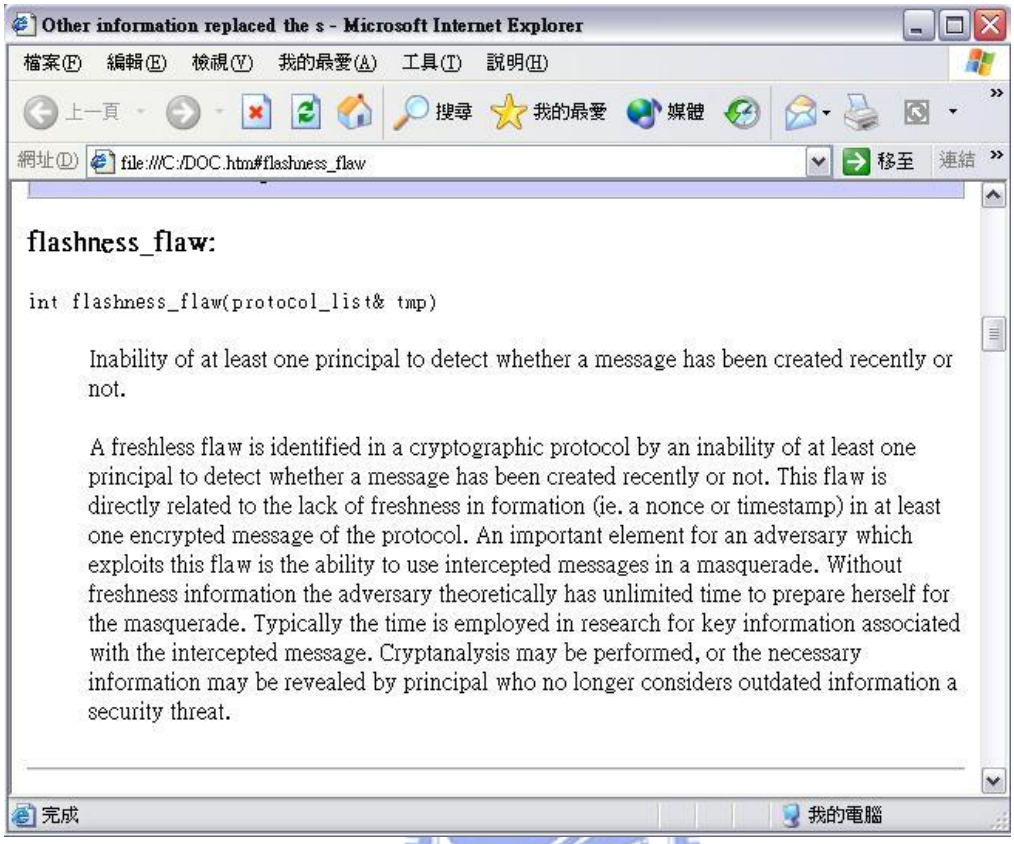
Information Analysis 這部分分析的了模擬的起始時間及終止時間，每一步的 Parser 是否完成，以及將輸入的資料完全的存入我們的資料結構中。

而模擬視窗這部分，簡單的敘述了整個安全協定的模擬行為，以及在網路上完全讓所以人知道的訊息，這當然也會讓攻擊者得到這些訊息。



最後主程式內部會自動的檢測所有的安全法則，若有一個沒有通過，則會出現警告視窗，如下

並且會出現說明網頁，更清楚的說明這個漏洞的相關訊息，如下



附錄二(測試用的資料)

Amended Needham Schroeder Symmetric Key

A->B:A

B->A:{A,Nb}Kbs

A->S:A,B,Na,{A,Nb}Kbs

S->A:{Na,B,Kab,{Kab,Nb,A}Kbs}Kas

A->B:{Kab,Nb,A}Kbs

B->A:{Nb}Kab

A->B:{Nb-1}Kab

Denning-Sacco shared key

A->S:A,B

S->A:{B,Kab,T,{Kab,A,T}Kbs}Kas

A->B:{Kab,A,T}Kbs

Otway-Ress2

A->B:Na,A,B,{Na,A,B}Ka

B->S:Na,A,B,{Na,A,B}Ka,Nb,{Na,A,B}Kb

S->B:Na,{Na,Kab}Ka,{Nb,Kab}Kb

B->A:Na,{Na,Kab}Ka

Hwang modified version of Neumann Stubblebine

A->B:A,Na

B->S:B,{A,Na,Tb,Nb}Kbs

S->A:{B,Na,Kab,Tb}Kas,{A,Kab,Tb}Kbs,Nb

A->B:{A,Kab,Tb}Kbs,{Nb}Kab

A->B:Ma,{A,Kab,Tb}Kbs

B->A:Mb,{Mb}Kab

A->B:{Mb}Kab

Kao Chow Authentication v.1

A->S:A,B,Na

S->B:{A,B,Na,Kab}Kas,{A,B,Na,Kab}Kbs

B->A:{A,B,Na,Kab}Kas,{Na}Kab,Nb

A->B:{Nb}Kab

KSL

A->B:Na,A

B->S:Na,A,Nb,B

S->B:{Nb,A,Kab}Kbs,{Na,B,Kab}Kas

B->A:{Na,B,Kab}Kas,{Tb,A,Kab}Kbb,Nc,{Na}Kab

A->B:{Nc}Kab

A->B:Ma,{Tb,A,Kab}Kbb

B->A:Mb,{Ma}Kab

A->B:{Mb}Kab

Lowe modified Denning-Sacco shared key

A->S:A,B

S->A:{B,Kab,T,{Kab,A,T}Kbs}Kas

A->B:{Kab,A,T}Kbs

B->A:{Nb}Kab

A->B:{Nb-1}Kab

Lowe modified KSL

A->B:Na,A

B->S:Na,A,Nb,B

S->B:{A,Nb,Kab}Kbs,{Na,B,Kab}Kas

B->A:{Na,B,Kab}Kas,{Tb,A,Kab}Kbb,Nc,{B,Na}Kab

A->B:{Nc}Kab

A->B:Ma,{Tb,A,Kab}Kbb

B->A:Mb,{Ma,B}Kab

A->B:{A,Mb}Kab

Needham-Schroeder

A->S:A,B,Na

S->A:{Na,B,Kab,{Kab,A}Kbs}Kas

A->B:{Kab,A}Kbs

B->A:{Nb}Kab

A->B:{Nb-1}Kab

Neumann Stubblebine

A->B:A,Na

B->S:B,{A,Na,Tb}Kbs,Nb

S->A:{B,Na,Kab,Tb}Kas,{A,Kab,Tb}Kbs,Nb

A->B:{A,Kab,Tb}Kbs,{Nb}Kab

A->B:Ma,{A,Kab,Tb}Kbs

B->A:Mb,{Ma}Kab

A->B:{Mb}Kab

Neuman-Stubblebine

A->B:A,Na

B->S:B,{A,Na,Tb}kbs,Nb

S->A:{B,Na,Kab,Tb}kas,{A,Kab,Tb}kbs,Nb

A->B:{A,Kab,Tb}kbs,{Nb}Kab

Otway-Rees

A->B:M,A,B,{Na,M,A,B}kas

B->S:M,A,B,{Na,M,A,B}kas,{Nb,M,A,B}kbs

S->B:M,{Na,Kab}kas,{Nb,kab}kbs

B->A:M,{Na,Kab}kas

索引

Interrupt on	阻礙
Interception	攔截
Modification	修改
Forgery	捏造
Active Attack	主動式攻擊
Passive Attack	被動式攻擊
Masquerade	偽裝
Replay	重播
Denial of Service	拒絕服務攻擊
NONCE	變數
Token	記法
Syntax	語法