國立交通大學

資訊科學與工程研究所

碩士論文

網路安全協定之自動化驗證

Automatic verification of network security protocol

研究生：劉宸瑋

指導教授：楊武　教授

中華民國　九十五　年六月

網 路 安 全 協 定 之 自 動 化 驗 證
Automatic verification of network security protocol

研 究 生：劉宸瑋　　Student：Chen-Way Liu

指導教授：楊 武　　Advisor：Wuu Yung

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

中 華 民 國　九 十 五　年 六 月

網路安全協定之自動化驗證

學生 ： 劉宸瑋　　指導教授 ： 楊武 博士

國立交通大學資訊科學與工程研究所

摘要

在目前這個電腦網路蓬勃發展的年代，要如何驗證千里之外與我們溝通者的身分，變成一個重要的問題。因此有許許多多的研究學者提出了一個又一個網路安全協定。但不幸的，要制定一個完美的安全協定是非常的困難的，我們在不斷推出的研究中，發現安全協定一個個被找出漏洞。因此我們想要找出一個方法可以檢驗安全協定的漏洞，相信對這個領域有所幫助。有鑑於目前主要採用的方式:邏輯分析法太過於複雜艱澀，我們試圖設計一個輕巧又有力的分析方法，不但易懂又保有威力。在本論文中，我們將先對網路安全協定領域做個介紹，之後提到我們的方法，並附上幾個實例，最後在介紹我們如何實作。期許這個新提出來的方法能對網路安全分析領域有所幫助。

Automatic verification of network security protocol

Student : Chen-Way Liu Advisor : Dr. Wuu Yang

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

Security protocol domain is more and more important with the popular Internet. But there are some troubles that we find flaws of many security protocols. So how to verify the correction of security protocols is a very important topic. Now, we propose a new algorithm to verify the security protocol. The goal of the method we desire is easy but robust. In this essay, we introduce base of the security protocol topic. Then we explain our method and give some examples. At last, we will show the our method's implementation, and we give some goals that may be used to improve the method better in the future. We hope that this algorithm can be some contribution to security protocol domain.

誌謝

首先，我要感謝我的指導教授，楊武博士。老師在這兩年來不斷的對我進行督促使我在研究上不斷的能有所前進，當我有許多問題的時候，也能耐心的對我進行教導。最後階段也一再的協助我進行論文的修改，給了我許多的意見。沒有老師的協助我無法在研究所的生涯中學習到如此豐富的知識，真是讓我學到太多東西了。我想再最後的說一次，老師，謝謝您。

再來要感謝研究室的學長，同學，學弟們。你們給了我一段豐富又有趣的時光。讓我覺得在研究室的生活總是多彩多姿，不論是在課業上，生活上，娛樂上。有你們的存在，讓我撐過許許多多的難關與挫折。

最後，我要感謝我的父母，家人。你們在背後的支持是讓我順利完全學業的最大動力。你們的鼓舞，讓我每次徬徨時，能再找到方向繼續下去，沒有你們，我不會能如此順利的完成學業，我想說，我愛你們。

目錄

圖目錄

第一章 前言

1.1 研究動機

資訊科學在這短短的幾十年間快速成長，其中網路方面的技術更是飛快的突破，革新。我們生活在現今社會的人們對網路的倚賴程度也越來越高，在這種情形之下，網路安全協定的設計就變的非常重要，因為錯誤的設計會遭受有心人士的利用導致誤認遠端某台電腦的身分，更進一步有可能還會洩漏本機重要的機密資訊。

雖然網路安全協定如此的重要，然而在許多研究報告顯示，世界上許多網路安全協定仍存在著許許多多的漏洞，這樣使的我們使用網路變成非常不安全，可靠。因此，設計出一個檢驗網路安全協定是否正確的方法，變成一個相當重要的課題。利用一個優秀的檢驗方法，對現行存在的網路安全協定檢驗，確定無誤之後再行採用，那對網路安全的品性將有很大的提升。而本篇論文的目的就是試圖設計出一個有效堅固的檢驗法則。

1.2 研究目標

網路安全協定應用於網路上，為了驗証某遠端電腦的身分，更進一步可以交換密碼學上的加密金鑰，供未來要與對方進行溝通使用。而設計不良導致嚴重後果就如同我們在上一節所提到的那樣嚴重。

雖然這方面的方法不斷的演進，越來越精密進步，驗證的功效也越來越高，但也讓方法變的十分的繁複，要使用那些方法來進行驗證需要受過相當的訓練還有長時間的使用才可熟練。因此我們換個角度思考，是否有可能設計出一個簡單易懂，但又相當有效的驗證方法，而且可以交給電腦自動驗證，不像上面所敘方法還需要大量的人力介入。這就是本篇論文努力的目的。

在此我們提出本篇論文設計的驗証方法主要幾項目標 ：

1. 一個可以驗証網路安全協定的方法
2. 這個方法可以簡當學習使用來進行驗証
3. 不因方法簡單而失去了嚴謹度
4. 不因為方法簡單而削弱了方法強大的驗証能力
5. 能夠自動化，執行不需人力介入。

1.3 論文架構

本篇研究總共分成六個章節，第一章是前言，簡介我們本篇論文研究的方向目標，第二章是對網路安全協定做基本的介紹，有助於對此領域不熟練者能先有基本的認識我們的研究領域的基本知識，有助於瞭解我們所提出的方法。

第三章則講述我們在網路安全協定中找出一些特性，，藉由這些特性經過整理觀察之後，我們想出了本篇論文的方法。第四章則就是我們方法的解釋說明以及演算法，在這一個章節閱讀完畢後，就能了解我們所提出方法的原理及其如何運作。第五章則提出幾個例子，例子分別針對不同種類的漏洞，有助於瞭解我們演算法解決問題的強度，也有助於單獨看第三章可能覺得演算法的概念太過於模糊，再實例說明後能讓概念更加清晰。

最後的第六章則是結論，我們把研究的成果做個總結與分析，並且探討之後繼續研究的方向該如何繼續的改進增強。

第二章 網路安全協定概論

2.1 網路安全協定概論

雖然我們在第一章的時候有稍微簡介了網路安全協定的一些基本的概念，但為了讓對此領域不熟悉者能有個基本的認識，有助在後面的章節易於瞭解我們的方法，此章節將介紹基本的網路安全協定的觀念及和之後介紹我們方法需要先具備的知識。

首先我們先由什麼是協定(Protocol)講起，定義如下:

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) 協定是有序的過程，每一個步驟必須依序執行，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) 當協定完成之後可達到某向我們所希望的目的。

而網路安全協定，由字面上看來即是幫助我們在結束完協定的步驟可以達成在網路上溝通能享有安全的保證，但這樣講很含糊不明確，實際上安全協定的分類法由很多種，我們根據常見的分類法，由目的來看可以分成三類網路安全協定，其名稱與意義如下所見:

(1) 金鑰交換協定(Key Exchange Protocol)

透過執行完這個協定之後，所有協定參與者都會得到至少一把金鑰，用來做之後溝通使用。

(2) 認證協定(Authentication Protocol)

主要的目的是想要驗證參與協定者的身分，當執行完協定之後，可確認某參與者或某些參與者的身份，可用來決定是否還要與進行溝通，或決定能給與多少權限執行本機所提供的資源…等。這類還協定還能細分為單向認證和雙向認證，甚至是多向認証。

(3) 認証和金鑰交換協定(Authentication and Key Exchange Protocol)

這類協定則是同時包含上面的兩種功能。

2.2 網路安全協定基本的表示法

我們在圖 1.1 看過了一個網路安全協定的例子，其中採用了許多代號，要能了解網路安全協定實際運行的步驟，首先就需要先知道一般常用的符號意義，而一些較特別的符號不在此介紹，通常在參照論文時會加以附註。

(1) Na : Nonce，隨機產生的變數，在理想的狀況下我們假定產生出來的 nonce 為完全隨機的且任兩個 nonce 的值不會相同，而 Na 代表了 A 所產生的隨機變數，通常用來做驗證使用。

(2) Kab : Shared Key，私有金鑰，在理想狀況下透過該金鑰加密的訊息只有金鑰持有人才能解開，而 Kab 代表只有 a 和 b 兩者持有該金鑰。

(3) A : Identity，用來表示身分，說明此訊息參與者…等，即參與此安全協定之使用者個人身分。

(4) {X}K : {X}K 表示訊息 X 被 K 加密，只有擁有 K 者才能解開此加密訊息得到 X。

(X 表示一連串明文訊息，K 表示金鑰，可以是公開金鑰或私有金鑰)

2.3 網路安全協定實例

我們在這一節看一下網路安全協定的格式

1. A->B: M, A, B, $\{N_a, M, A, B\}_{Kas}$
2. B->S: M, A, B, $\{N_a, M, A, B\}_{Kas}$, $\{Nb, M, A, B\}_{Kbs}$
3. S->B: M, $\{N_a, K_{ab}\}_{Kas}$, $\{Nb, Kab\}_{Kbs}$
4. B->A: M, $\{N_a, K_{ab}\}_{Kas}$

圖 2.1 Otway - Rees Protocol

這協定中共有三位參與者: A 為請求協定溝通者，B 為接受請求協定者，S 為協助進行安全協定的第三方。一個設計良好的安全協定，當傳遞完所有的訊息之後，A 可以驗証 B 的身分(單方向認證)，部分功能叫齊全的協定甚至可以達到 B 也能驗証 A 的身分(雙向認證)，除此之外視需求可能還會交換金鑰，當未來 A 和 B 驗証身分結束後，進行溝通可以加密訊息確保機密。圖 1.1 中的安全協定即是有交換金鑰(Kab)。

2.4 網路安全協定的安全性

經過基本的介紹後，我們對網路安全協定有了一定的瞭解。接著我們就來探討它有可能發生的問題，藉由知道網路安全協定遭受的問題及對應的攻擊方法，在有所認知之後，才能開始思考如何解決這些問題的辦法。

網路基本攻擊法概述

在探討網路安全協定的攻擊法之前，在網路安全，基本假設攻擊者所具有的基本攻擊能力，有以下四種 ：

1. 阻斷（Interrupt）：利用各種攻擊方法癱瘓電腦系統或者讓使用者要存取的資料無法使用。這是針對資料的可利用性加以攻擊。

2. 攔截（Interception）：一個非法的第三者在網路上竊取他人在網路上溝通傳送的資料。這是針對資料的機密性加以攻擊。

3. 修改（Modification）：一個非法的第三者不僅從網路中竊取資料，並將資料加以修改，再傳送給原本的接收者，試圖欺騙之。這是針對資料的完整性加以攻擊。

4. 捏造（Forgery）：一個非法的第三者擅自偽造資料，並假裝是某位合法使用者的身分，將資料傳給欲欺瞞的對象。這是針對資料的確實性加以攻擊。

看過了基本攻擊者的能力，接著針對網路安全協定，我們介紹到目前為止被提出來的各種漏洞。根據Ulf Carlsen 於1994年提出的論文[12]，大致可以劃分為如下面幾種漏洞:

1. 基本型漏洞 (Elementary flaws)

這類型的漏洞發生在早期網路剛開始使用時,主要是忽略了在網路上傳輸時，忘記重要的資料要利用加密法保護，而遭攻擊者擷取重要資訊，現在這一類型的錯誤已經很少發生。

如果我們一般常用的電子佈告欄系統(BBS)，當我們要傳送帳號與密碼皆是利用明文傳送，即會遭受此種攻擊。

2. 密碼猜測漏洞 (Password-guessing flaws)

這類漏洞主要算是加密方法不夠強大，抵禦能力不足，可能在現代高速的電腦計算能力之下，在不常的時間內即被破解，或者是使用者採用了長度過短或一般常見的單字來當密碼，讓攻擊者利用暴力法或字典攻擊法的方法猜測出加密金鑰，進而對整個網路安全協定產生危害，導致原本看似正確無誤的網路安全協定出現問題。

3. 時效性漏洞 (Freshless flaws)

這類型的漏洞，主要發生在某些網路安全協定，某個參與者無法驗證他所接收到訊息是否是本回合協定溝通中由對方產生且傳送給他的，還是某個惡意第三者利用其之前擷取並且儲存下來的訊息傳送給他，導致參與者誤判，對系統造成危害。我們看下面的一個例子:

1. A->S : A, B, N_a

2. S->A : $\{N_a, B, K_{ab}, \{K_{ab}, A\}K_{bs}\}K_{as}$

3. A->B : $\{K_{ab}, A\}K_{bs}$

4. B->A : $\{N_b\}K_{ab}$

5. A->B : $\{N_b\text{-}1\}K_{ab}$

圖 2.2 The Needham-Schroeder protocol

這個例子中，我們可以很明顯的發現Message3沒有任何資訊可以驗證此訊息是否為新鮮的(Fresh)，所以攻擊者可以將一封訊息三擷取並且儲存下來，再利用密碼學方法破解取得Kab，之後若A和B還有在利用此安全協定進行溝通，喬裝成A傳送訊息三給B，造成B的誤認。安全協定即沒達到原本的目的，這就是沒有注意時效性的漏洞。

4. 神諭使漏洞 (Oracle flaws)

這算是最複雜的一種漏洞，也算是所以漏洞中非常難分析的一種，變形的攻擊很多，因此很難預防，但隨著檢驗的技術不斷推陳出新，我們已經能較為有效的偵測出這種缺點。

＊本漏洞有許多別名，將他們列出來有助瞭解(僅列出英文):

Parallel session attacks, Reflection attacks, Multi-role flaws。

本漏洞主要是利用網路安全協定交換訊息的過程，攻擊者藉由扮演特定參與者身分，且利用多次的安全協定溝通，得到特定的機密資訊，即使該資訊有受到完美的加密演算法所保護。我們可以參考下面的例子:

1. A->B : $\{M\}K_a$
2. B->A : $\{\{M\}K_a\}K_b$
3. A->B : $\{M\}K_b$

圖 2.3 The simple Three pass protocol

這是一個很簡略的三步驟式協定，並且我們假設$\{\{M\}_{Ka}\}_{Kb} = \{\{M\}_{Kb}\}_{Ka}$，現在如果有位參與者Alice扮演A的腳色想要和Eve溝通(如此一來，Eve扮演著B的角色)，但攻擊者Bob冒充了Eve，則會出現以下的攻擊模式:

1. Alice->Eve_{Bob} : {M}Ka
2. Eve_{Bob}->Alice : {M}Ka
3. Alice->Eve_{Bob} : M

圖 2.4 Single-role oracle attack on the protocol

由上圖，第二步驟時Bob沒按照規則傳送訊息二，反而傳送$\{M\}_{Ka}$，而Alice接收到時，按照規則將訊息二用金鑰Ka解密傳送回去，這樣一來明顯變成傳送明文，導致原本機密的資料就此洩漏出去。

5. 類別漏洞 (Type flaws)

這類型的漏洞，現在在實做上通常都已經能夠避免了，而且已經有提出許多良好的設計方法，可以使設計出來的安全協定不會遭受到此種漏洞，但這是一個相當經典的漏洞，攻擊法也相當的漂亮，值得我們好好探討一翻，或許將來會有利用這方法為基礎，延伸出來的新攻擊法。

本方法主要利用某些網路安全協定在實做沒有詳細檢查收到訊息的類別，導致在攻擊者精心設計之下，把某類型的資訊誤認為另一種類型的資訊，導致網路安全協定出現問題。(通常是讓參與者把Nonce或Identity誤認為Key)

我們參考以下經典的例子:

1. A->B : M, A, B, $\{N_a, M, A, B\}K_{as}$
2. B->S : M, A, B, $\{N_a, M, A, B\}K_{as}$, $\{N_b, M, A, B\}K_{bs}$
3. S->B : M, $\{N_a, K_{ab}\}K_{as}$, $\{N_b, K_{ab}\}K_{bs}$
4. B->A : M, $\{N_a, K_{ab}\}K_{as}$

圖 2.5 Otway-Rees protocol

這個安全協定每個加密訊息，看似包含了許多資訊，難以利用之前所講的神諭使攻擊法攻破，但卻有個極大的類別漏洞。攻擊如下:

1. A -> E_b : M, A, B, {N_a, M, A, B}K_{as}

2. E_b -> A : M, {N_a, M, A, B}K_{as}

圖 2.6 Attacking the Otway-Rees protocol

當攻擊者收到A傳來的訊息一時，即回傳訊息四給A，而若沒有檢查訊息類別的實做情形下，A會將M, A, B 三個訊息總和當成Kab，如此一來攻擊者之後就可以把M, A, B總和當成金鑰和A溝通，這可說是非常精簡有力的攻擊法。

6. 內部漏洞 (Internal flaws)

內部漏洞主要發生在實做時，除了協定訊息的傳送外，缺乏做一些本機該做的檢查所導致。

如圖2.3，當A在傳送訊息三之前，因該要先檢查訊息三是一封加密的訊息，如此一來就不會遭受神諭使攻擊法的攻擊。設計網路安全協定，應該明確的規範每個本機端內部的行為，如此一來，可以大大減少發生各種漏洞的機會。

7. 加密演算法漏洞 (Cryptosystem-related flaws)

這點和密碼猜測漏洞很有關係，設計不良的加密演算法很容易被密碼猜測法攻破，除了之外有些安全協定會在傳送的訊息中，利用XOR之類的函式組合一些訊息，但是若設計不當很可能遭受攻擊者使用一些精妙的技巧推出原本隱密的訊息。所以在設計與加密有關的部份時需要慎重的考慮。

2.5 相關研究

許多網路安全協定因為設計不良，導致有漏洞，反到使想要利用協定來進行驗証身分者受到傷害，因此檢驗一個網路安全協定是否有漏洞，成為了網路安全協定領域中相當重要的一個課題，受重視的程度超乎之前主要的研究重心如何設計出一個優秀有效率又正確的網路安全協定。

驗證網路安全協定是否正確的這個領域中，其要驗證的目標又可以分為兩者，一是要證明網路安全協定真的能夠達成如設計者所期待的目的，令一則是要證明網路安全協定中交換的金鑰不會洩露。

這許許多多的方法先後被提出激盪之後，邏輯證明法成為了許多學者研究的方向，其中最有名的也就是Ban Logic[2]，也是這方法推出之後激起大家往此方面的研究，隨後Ban logic的缺點也被發現，最大的原因在於它設計之初就專注在於檢測網路安全協定是否能達到驗證對方身分，而忽略了網路安全協定中交換的金鑰會不會洩露[14]。之後的GNY logic, AT logic, VO logic, SVO logic接是依據Ban logic修補改良，改進了Ban Logic的缺點，但面臨了一個重要的問題是雖然功能越來越強大，但是過於複雜以致難以學習使用，而SVO logic不但成功驗証許多認証協定，又維持Ban logic的易於使用，算這方面的佼佼者。

另一許多學者採用的方式是CSP，模型檢驗技術是驗證有限狀態系統的自動化分析技術，是一種安全協議的自動驗證工具。Lowe等學者應用CSP方法的成功，促進了這一領域的發展.Schneider發表了一系列關於CSP方法應用的論文，應用CSP方法討論安全協議的安全性質、匿名等問題;分析了各種安全協議。 Clarke教授的研究小組，長期從事定理證明和自動檢驗的研究。他們提出了一種通用的模型檢驗器，提出了一種新型的模型及其代數理論，並證明瞭該模型的有效性。

Mitchell的方法是透過狀態計數工具Murphi分析安全協議，從安全協議可能到達的狀態，分析安全協議是否安全。他應用Murphi分析了一系列著名的安全協議，成功地發現了所有已知的攻擊。

最後提到Spi-Caculus模型，Spi-Caculus是由Pi-Calculus所延伸。Pi-Caculus是依個用來描述分析網路安全協定的的簡單又強力的程式語言。然而因為其沒有支援加密運算，所以延生出了Spi-Caculus的這種方法，補足了加密部分。其主要是將各個參與者視為一個程序(Process)彼此利用管道(Tunnel)傳訊訊息。而當接收到訊息的時候，可能會產生一些狀態的變化。

第三章 演算法概念

3.1 定義

一開始我們先為一些名詞做定義，如此一來在之後提到我們的方法中看見這些詞彙時，才能正確無誤的了解他們。這樣也提升了整個方法的嚴謹度，避免產生誤解。

※參與者（Participant）

參加一次網路安全協定溝通的所有端點，通常是一位參與者可想成是一台電腦，或電腦上的一個執行中的程序(Process)。如圖1.1共有三個參與者A, B, S。

※訊息（Message）

一位參與者在打算展開一次協定時會發出許多資訊，或者當某位參與者收到資訊有可能會發出一些資訊，我們稱由一參與者傳送給給另一為參與者一次的資訊總和為訊息。一個網路安全協定可以視為一群訊息的總和。如圖1.1中的協定，我們稱第一句被傳送的訊息為訊息(1)（ Message one ），第二句被傳送的訊息為訊息(2)，接下來以此類推。

※元素（Atom）

一封訊息中最小的單位，無法再被切割，如一把Key Ka，如果再切割就不能被視為一把Key。一般來說常見的atom有User Identity, Nonce, Timestamp, Key。

※攻擊者知識庫（Intruder’s Knowledge）

這是我們設計方法所設計出的獨特名詞，入侵者會將在網路協定中，他會將所有看過的訊息記憶起來，我們稱他所有記錄的資訊集合為知識庫。

Ex: 當入侵者看到 A→B: Kab, A.

他便會把 Kab 和 A 加到他的 Knowledge 之中。

這裡需要注意的是，如果入侵者的 Knowledge 中擁有 Kab，和 A 代表著，他 Knowledge 也等於擁有擁有{A}Kab, {Kab}A, {Kab, A}Kab…等資訊。

※參與者知識庫 (Participant’s knowledge)

和攻擊者的知識庫有點類似，每位參與者都會擁有屬於他自己的紀錄空間，紀錄一開始他便知道的訊息，和在協定過程中他將產生的元素。不過和攻擊者知識庫不同的是我們不讓參與者的知識庫進行成長，一直保持初始設定。主要的功能是用來進行驗證他收到的訊息，有他產生或者認識的部份。如果沒有額外假設，一開始裡面有的資訊包括: 和所有參與者共用的 Key, 所有參與者 ID，及自己的 Public & Private Key(如果是 public protocol 的話)，還有協定過程中會用到的 Nonce 與 Timestamp。

※公開金鑰關聯性 (Public key relation)

指兩個數 X, Y 滿足 : 某數 A 用 X 加密，可以用 Y 解密為 A；用 Y 加密，可以用 X 解密為 A，則稱 X, Y 有公開金鑰關聯性。

※一個有漏洞的安全協定 (A Flaw Security Protocol)

我們稱一個網路安全協定有漏洞，是當我們可以冒充進行溝通的某一方，或者是網路安全協定中交換供之後溝通用的金鑰洩漏。

3.2 演算法基本原理與目標

由上面的定義可看出，如果一個網路安全協定有漏洞，那可能造成假冒成功，或者是溝通的 Key 洩漏。因此我們的方法打算嘗試去攻擊接受我們方法檢驗的網路安全協定，看是否會產生上面的缺點。

原始的想法很簡單，看以下的 Protocol，假設我們要攻擊 B，也即是我們要成功冒充成 A 和 B 溝通。

1. $A \rightarrow B : A, Na.$
2. $B \rightarrow S : B, \{A, Na, Tb\}Kbs, Nb.$
3. $S \rightarrow A : \{B, Na, Kab, Tb\}Kas, \{A, Kab, Tb\}Kbs, Nb.$
4. $A \rightarrow B : \{A, Kab, Tb\}Kbs, \{Nb\}Kab$

圖 3.1 Neuman Stubblebine Protocol

那我們只要能產生出 Message1, Message4，那就會讓 B 相信我們是和他溝通的人，這也是網安安全協定攻擊中所謂的: 冒充身分。但是除此之外我們還需要考慮一個問題，有的網路安全協定有所謂交換之後要溝通使用的 Key，雖然我們成功的冒充身分，但若是沒有取得 Key，那就像是介於兩個溝通者網路中間傳送的一個端點而已，雖然冒充了身分，但是沒有 Key，收到了一堆密文也徒勞，也無法於之後繼續和我們攻擊的目標進行溝通。

經由上面的分析歸納，我們定義出我們所設計的演算法，認定成功的攻擊，共有兩種格式:

1. 若是沒有在安全協定中交換 Key

成功冒充身分，就算是成功的攻擊。

2. 若是有在安全協定中交換 Key，除了冒充身分，

還要能夠取得交換的 Key，才算是成功的攻擊。

那麼我們要怎麼樣產生我們需要的Message呢?我們打算利用推演法則來產生我們所需要的訊息，這是什麼意思呢?

我們假定攻擊者有個知識庫，裡面存放著他所知道的資訊。一開始他可能只知道一些基本的資訊，但隨著他藏匿在網路背後觀察，他所知道的資訊就會越來越多。我們所要產生的訊息，就看是否攻擊者的知識庫可以產生該訊息，如果不行則嘗試傳遞訊息給某些參與者，看是否他會幫我們產生某些訊息。我們再檢察這是否是我們所需要的訊息，如果不是，則在傳送給另一些參與者看看。藉由每個參與者收到特定訊息會產生對應訊息的這種方法，這即是我們所講的推演法則。

此外我們採用由後面往前推演的法則: 檢查每一句訊息看是否能攻擊者的知識庫能夠產生，若不行看是缺少了什麼資訊。然後嘗試藉由推演法則先設法產生前一步訊息，因為如果能得到前一步訊息，及能得到對應的此步訊息，如果前一步訊息不能產生，再往前推一步訊息，這樣不斷的推演，利用訊息間關聯，來探討整個安全協定是否有所疏失，可以在我們多方測試下製造出安全協定設計者意想不到的訊息，導致系統的危害。

我們可根據過去期刊中所提到的許多網路安全協定，要將之攻破是複雜的，需要經過累積執行好幾次安全協定才能破解。這也相對增加了我們要設計出一個自動驗證方法的難度，因此設計的方法要可以自動進行好幾次安全協定溝通嘗試破解協定，有如人眼觀察法一樣。所以我們在推演法的過程中，當我們試圖製造訊息，我們會嘗試用各種身分去嘗試進行許多回合的溝通，根據我們觀察整理設計之後定出來的規則，這樣有助於能偵測出需要許多回合的溝通才會發生的漏洞。

第四章 自動化網路安全協定演算法

4.1 基本假設

在介紹我們方法前，我們先根據我們假設一些我們所存在的環境，即常用到的字彙的簡稱，有助於更快熟悉我們的方法。

※角色代號

我們採用英文字母大寫來代表在網路安全協定中所扮演的各個角色，我們假定我們要該要攻擊的該網路安全協定，其中:

A : 請求者 B : 接受請求者 S : 驗證伺服器 P : 入侵者

C, D, E… : 且已經被入侵者攻破(入侵者擁有它所有的金鑰)

※入侵者能力

我們假設入侵者具有以下能力

A. 具有控制整個網路的能力
   - 接收任何在此網路傳送的訊息。
   - 阻斷一位使用者，傳送給另一使用者的訊息。
   - 假造訊息傳給任意使用者。

B. 入侵者也為此網域的一位合法使用者，即他和S共用一把金鑰

C. 入侵者可以利用它所擁有的金鑰將任何message根據網路安全協定底層所規定的加密演算法進行加解密。

D. 入侵者無法用暴力法之類的任何方法將一封他沒有擁有金鑰的message進行解密。

4.2 步驟流程

我們在第三章提到許多我們觀察網路安全協定的心得與想法，現在我們就來看看根據那些想法，我們設計出來的網路安全協定攻擊演算法。在這一節我們將詳細完整的介紹演算法的所有步驟，其所採取的行為以及它的目的。

1. 設定初始環境

一開始必須提供一些基本環境設定給系統，包括

A. 此Protocol是否有交換Key，其代號為何，如key Ka。

B. 此網域共有多少合法使用者(2 或 >2，不包含驗證伺服器），若合法使用者為 2 表示我們不能假冒其他人的身份嘗試產生訊息。

C. 設定各參與者知識庫中的資料，

並且設定一開始入侵者的初始Knowledge所擁有的資料。

EX:入侵者和驗証Server共用一把對稱金鑰K_{PS}。

D. 將網路安全協定文字檔轉換成程式資料Protocol_msg[NUMBER]，

NUMBER是該安全協定的訊息數，假定由 1 開始，除此之外還保存此訊息的傳送者，此訊息的接收者，傳送的訊息內容。

2. 選擇攻擊目標

將要攻擊目標所收到的訊息全部列出，我們稱那些攻擊目標所該收到的訊息為攻擊訊息。我們可以選擇攻擊 A, 或 B(攻擊 A 表示我們想冒充 B 和 A 溝通)，一般來講我們不會攻擊 S，也未曾在論文中看過攻擊 S 的例子，因為一般而言我們僅僅是把 S 當作幫我產生特定模式訊息的中間人，有可能我們會欺騙 S 幫我們產生特定訊息，但是絕非我們要攻擊的終極目標。

Ex. 如圖 2.5，若我們想攻擊 B，此步驟就列出訊息一與訊息三。

3. 進行攻擊訊息改寫

針對每一則攻擊訊息，如果攻擊目標無法驗證此訊息中的某個元素，將其取代成元素變數(我們稱為 atom variable)。改寫攻擊訊息的目的是，如之前所提，如果我們能產生所有攻擊訊息則代表攻擊成功，改寫之後的攻擊訊息，被取代成元素變數的地方，其意義代表可以置入任意訊息，可使我們更有可能產生攻擊訊息。

不過有例外一點例外，因為對參與者 ID 是我們要冒充的目標，對其進行取代沒有意義，所以不對其進行取代。

◆何謂無法被驗證的元素

在我們所設計的演算法中我們採用一個非常直觀的規定，當一個端點收到一則訊息，其中的某個元素若不在該端點的知識庫之中(各端點的知識庫在步驟 1.C 設定完成，一般設定為該端點在該網路安全協定中所產生的資訊，及一開始便了解的初始資訊)，即稱之為無法被驗證的元素。可取代成 Atom variable。

◆EX: 假設 Message A -> B: M, A, B, Ka, Kab, Kas, {M, A, B, Na}Kb，而 B 的知識庫為 : [B, S, Kab, Kb, Kbs] (一般性的假設，B 知道自己和驗證伺服器的名稱，以及和他有關的金鑰)，那麼取代後的訊息為 : X, A, B, Y, Kab, Kas, {X, A, B, Z}Kb，其中 X, Y, Z 分別為三個 atom variable 分別取代 B 所不能驗證的 M, Ka 和 Na。值得注意的是，因為我們不取代參與者 ID，所以 A 我們不到其取代，且我們不取代與驗證伺服器共有的金鑰，所以我們對 Kas 也不進行取代。

4. 建立訊息關聯表

這個表格由幾句關聯法則(Related rules)所組成。主要是搭建起每步訊息間的關係，藉由這個表格我們能夠知道，如果需要什麼訊息，因該要傳入什麼訊息。表格總共會有比網路安全協定少一行，其格式如下:

(傳入訊息編號, 送出訊息編號, 訊息接收者, 變數(參與者1), 變數(參與者2)):
修改後傳入訊息//修改後送出訊息

其中修改後傳入與送出訊息的格式，將安全協定請求者相關訊息用變數X取代，和接受者相關訊息用變數Y取代。為了幫助了解，我們舉例說明:

$$
\begin{aligned}
&(1)\ A \rightarrow S \ : \ A, B, Na \\
&(2)\ S \rightarrow A \ : \ E(Kas : Na, B, Kab, E(Kbs : Kab, A)) \\
&(3)\ A \rightarrow B \ : \ E(Kbs : Kab, A) \\
&(4)\ B \rightarrow A \ : \ E(Kab : Nb) \\
&(5)\ A \rightarrow B \ : \ E(Kab : Nb - 1)
\end{aligned}
$$

圖4.1 Needham Schroeder Protocol

我們圖4.1的安全協定製作訊息關聯表如下(共有4句關聯法則):

(1, 2, S, X, Y) : X, Y, Nx // { Nx, y, Z, {Z, x}Kys }Kxs
(2, 3, A, X, Y) : { Nx, Y, Z, {Z, X}Kys }Kxs // {Z, X}Kys
(3, 4, B, X, Y) : {Z, X}Kys // {Nb}Z
(4, 5, A, X, Y) : {Ny}Z // {Ny-1}Z

建立關聯表之後，就可以對其中的關聯法則進行實體化，其方法是是將X如Y帶入實際的參數，則產生對應句子。

如我們對(1, 2, S, X, Y)帶入X值為C, Y值為D，可得當我們傳入訊息C, D, Nc，可得到對應訊息{ Nc, d, Z, {Z, c}Kds }Kcs。對這句關聯法則實體化的意義是我們傳送訊息給S並告知此訊息是C和D溝通進行使用的，他會檢索之後產生送出訊息。

5.本階段為攻擊演算法主體，是一個迴圈，由第一個攻擊訊息開始檢查，看是否每封攻擊的訊息我們都能夠順利產生。

A. 選擇第一封尚未檢查的攻擊訊息，將在此訊息前的所有訊息其中的所有元素加入攻擊者的知識庫。

B. 檢查攻擊者知識庫是否能產生該攻擊訊息，如果可以則檢視下一封攻擊訊息重複步驟 4。

如果無法產生則查閱訊息關係表，取得到本訊息的前一句訊息 M。藉由訊息關聯表用各種身分去嘗試帶入 M，取得對應的輸入句。然後將 M 的接收者和我們嘗試丟入的身分兩者當參數丟入對話函式中直到 M 這句為止，並把所有過程添到攻擊者知識庫中。檢查看是否能夠產生了，若不能則換一個身份帶入，如果全部身分都失敗，代表攻擊失敗

◆這裡有項規定，若我們能產生某個加密元素，因為知識庫中擁有該元素，但其中若包含交換金鑰，而且我們並不擁有該金鑰，我們視為我們無法產生該加密元素。

◆對話函式：

一組對話函式會有兩個參與者，我們用(A, B)表示，其中 A 為發出此次 Protocol 溝通的請求者，B 為接受請求者。按圖 2.2 範例，將(A, B)帶入，就是如圖中所寫，如果我們換成(A, P)則是

1. A → P : A, Na
2. P → S : P, {A, Na, Tp}Kps, Np
3. S → A : {P, Na, Kap, Tp}Kas, {A, Kap, Tp}Kps, Np
4. A → P : {A, Kap, Tp}Kps, {Np}Kap

不過在此在這裡我們面臨了一個問題:

我們要用各種身分帶入關聯表，是什麼樣的身分呢?

這裡我們打算採用窮舉法，試驗所有的可能性如下:

A:訊息請求者, B:訊息接受者, C:此網域被攻擊者攻破的合法使用者。

＊本步驟比較複雜，我們在此繪製一下流程圖幫助了解

◆攻擊某句訊息(試圖假造)，檢視攻擊者知識庫

1. 能成功製造，攻擊下一句訊息
2. 失敗
   - A. 由關係表取得對應的輸入訊息
   - B. 將攻擊訊息的接收者 a 和一參數 b 帶入關係對應表
     取得對應訊息的實際內容 M
   - C. 執行對話函式(a, b), (b, a)，並將所有的訊息加入攻擊者知識。
   - D. 檢視攻擊者知識庫是否能產生 M
     - a. Yes, 檢視下一句攻擊訊息
     - b. No,換一組參數帶入。若所有參數都不能幫助我們得到
       目標訊息，則攻擊失敗。

6. 若所有 Message 皆能產生，根據是否有交換 Key 做之後的溝通，可以得出兩種結果:

   A. 此安全協定沒有進行 Key 的交換，即顯示此協定有漏洞。

   B. 若有 Key 的交換，則需檢查我們的 Knowledge 中是否有該 Key（不一定是真的有，可能是該 Key 被之前推導過程改為 atom variable，而我們有辦法產生該格式即可），如果有 Key，則表示該協定有漏洞。

4.3 演算法

0. 設定初始環境。

// 1. A 為訊息請求者，B 為接受請求者，S 為驗証第三方，P 為入侵者

// 2. Format Intruder's Knowledge: string Knowledge [space]

// space is size of knowledge and each column contains a atom which type

// is string.

//3. 設定 Protocol 有沒有交換 Key，該 Key 名稱，一般假設為 Kab

//4. 將網路安全協定文字檔讀入 Global variable‘ Protocol_msg[NUMBER] ’

// Protocol_msg[NUMBER]，NUMBER 是安全協定的訊息數目，假定由 1 開始

// Protocol_msg's Data structure

// { char sender; //此訊息的傳送者

// char receiver; //此訊息的接收者

// string content; //傳送的訊息內容

// }

1. char attack_goal = 攻擊目標; //選擇攻擊目標

2. 建立 global variable‘ desiremsg’， desiremsg 為一個 Queue，其中 element 為 int。

   call procedure get_goalmsg(attack_goal)

   // Queue 提供三個個函式 insert，delete 與 isempty

   //此三個函式如同一般資料架構書籍所定義

3. 對在 Desiremsg 中的訊息進行改寫，將其中攻擊目標無法檢驗的 atom 改成 atom variable，參考前一節 3.攻擊訊息改寫。

4. 建立訊息關聯表 RelatedTable[NUMBER] // NUMBER 是協定的訊息數目

```
//RelatedTable ’s Data structure
//{
//    int inNum;          //傳入訊息編號
//    int outNum;         //輸出訊息編號
//    char role;          //訊息接收者
//    string content;     //輸入輸出訊息預設內容，格式: xxxx // xxxx
//    char* getInMsg(char X, char Y);   //取得輸入訊息
//    char* getOutMsg(char X, char Y)  //取得輸出訊息
//}
```

```
while ( isempty(desiremsg) == false ) do {
        int no =delete (desiremsg);          //no 為目前想要產生之 message 編號
        for i from 1 to ( no - 1 )
            add all atoms of Protocol_msg [i].content into Knowledge[space]

    If (Check_Knowledge (Protocol_msg [no].content) == false) {
        int goal = searchOutNo(no);
        //假設此函式會找出 RelatedTable 中 outNum 為 no 的法則編號，
        //可用 For 迴圈實做。
```

```
//以下步驟將查閱訊息關聯表取得對應訊息並帶入參數以進行實體化，
//參數為我們的攻擊目標，和 A, B, C 其中一者
//A 為訊息初始者，B 為訊息接受者，
//C 為該網域合法使用者且為入侵者攻破，並因攻擊目標的身分，
//改變帶入參數的位置。
//若攻擊目標為協定初始者，假設為 A
if(attack_goal == A){
        String goalmsg0 = RelateTable[goal].getInMsg(A, B);
        String goalmsg1 = RelateTable[goal].getInMsg(A, C);

        runConversation(A, B);   runConversation(B, A);
        runConversation(A, C);   runConversation(C, A);
}

//若攻擊目標為協定接受者，假設為 B
if(attack_goal == B){
        String goalmsg0 = RelateTable[goal].getInMsg(A, B);
        String goalmsg1 = RelateTable[goal].getInMsg(C, B);

        runConversation(A, B);   runConversation(B, A);
        runConversation(B, C);   runConversation(C, B);
}
```

```
            If (Check_Knowledge (goalmsg0) == false &&
                Check_Knowledge (goalmsg1) == false){
                 //輸出產生訊息失敗!
                 Exit(0);
            }
        }

} //End of while

if 此 Protocol 沒有交換 Key
//這裡指的有漏洞就是我們可以冒充該使用者，但至於攻擊途徑發展至的演
//演算法目前仍無法列出來
    Protocol 有漏洞 END
else{ //有交換 session key Kab
      if(Check_Knowledge (Kab) ||
          intruder's Knowledge 有 atom 和 Protocol 中交換 Key 的 Message
          有相同格式，且 Kab 部分有其餘 atom 取代)
              Protocol 有漏洞 END
      else
         攻擊此 Protocol 失敗 END
}
```

```
//將所有我們攻擊目標的訊息之編號加入 desiremsg
Void get_goalmsg (char attack_goal) {
    For i from 1 to NUMBER {
        If (Protocol_msg [i]. receiver == attack_goal)
            Insert i to desiremsg;
    }
}

//檢查是否 Message 中所有 atom 都存在於 Knowledge 中，
//如果是回傳 True ，否則回傳 false。
Bool Check_Knowledge (Message)
{
    If( all atoms of message are in the knowledge)
        return true;
    return false;
}

//執行對話函式，且把所有產生的訊息丟入攻擊者知識庫
//傳入兩個參數，分別代表訊息請求者和訊息接收者
void runConversation(char roleA, char roleB)
{
    將網路安全協定取代，如前一節 5.附註對話函式，採用(roleA, roleB)。
    將所有訊息加入攻擊者知識庫。
}
```

第五章實例解說

在這個章節裡面，我們將舉幾個例子來說明我的演算法，這些例子中的安全協定分別屬於常見的網路安全協定漏洞(神諭使漏洞，和類型漏洞)，也包含了公開金鑰與私密金鑰的類型，看我們的方法如何找出這些漏洞。

範例一 Neuman Stubblebine Protocol

Neuman Stubblebine Protocol 犯了常見的類型錯誤，因為類型錯誤比較好偵查，所以我們在此把它當成我們第一個例子。雖然這個例子不能看到我們安全協定的精髓，但再玼當第一個範例，看我們的安全協定如何偵測出類型漏洞。

1. $A \rightarrow B : A, Na.$
2. $B \rightarrow S : B, \{A, Na, Tb\}Kbs, Nb.$
3. $S \rightarrow A : \{B, Na, Kab, Tb\}Kas, \{A, Kab, Tb\}Kbs, Nb.$
4. $A \rightarrow B : \{A, Kab, Tb\}Kbs, \{Nb\}Kab$

圖 5.1 Neuman Stubblebine Protocol

演算法步驟:

1. 設定初始環境，如各個參與者的知識庫…等，最重要為要設定此安全協定有交換 Key 叫 Kab。
2. 選擇攻擊目標 B，列出所有攻擊訊息，此安全協定只需要能產生，訊息(1)和訊息(4)即可。
3. 重新改寫所有攻擊訊息，如下:

   (1)A->B : A, X

   (4) A -> B : {A, Y, tb}Kbs, {Nb}Y(要注意取元素變數 Y 對應的交換 Key Kab)
4. 建立關聯表，因為檢測類型漏洞主要是類型遭攻擊者惡意誤用，不需要檢查

關聯表，我們在此略過此步驟不詳細寫出關聯表。實際上偵測我們不曉得安全協定可能有什麼問題，此步驟不能省略。

5. 攻擊目標為訊息(1)，可以從資訊庫輕鬆產生出來。

6. 攻擊目標為訊息(4)，所以將訊息(2)~(3)加入攻擊者知識庫。

7. 檢測知識庫，發現我們有加密元素{A, Na, tb}Kbs 與{A, Y, tb}Kbs 格式相同，

且我們知道 Na 的值(因為 Y 是交換金鑰，需要知道其值才能算成功產生)。

後面{Nb}Y 也能輕鬆產生，所以我們能成功產生訊息(4)。

8. 我們成功產生所有攻擊訊息，且此安全協定有交換金鑰，我們也能夠拿到，代表此安全協定有漏洞。

範例二　SSH protocol

這是眾多 SSH 協定其中的一個版本，這版本有非常明顯的錯誤，藉由我們的演算法可以很容易的識別。除此之外本安全協定有使用的公開金鑰的加密技術，能提供解說我們的方法如何處理公開金鑰的問題。下列是列出此安全協定用到的一些特殊符號:

H() : Hash Function

K'　is derived from K, Na, and Nb

previous msg : 即訊息(1) - (3)所有的內容

1. A -> B : N_A

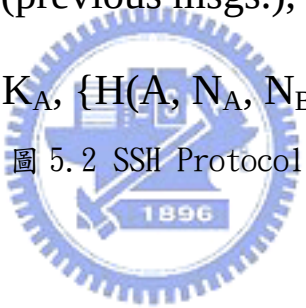
2. B -> A : N_B

3. B -> A : K_{Bh}, K_{Bs}

4. A -> B : {{H(previous msgs.), K}K_{Bs}}K_{Bh}

5. A -> B : {A, K_A, {H(A, N_A, N_B)}K_{Apr}}K'

圖 5.2 SSH Protocol



演算法步驟:

1. 基本初始設定。
2. 攻擊目標 B，列出必要產生訊息:訊息(1), 訊息(4), 訊息(5)
3. 改寫所有的攻擊訊息
   (1) A -> B :　X
   (4) A ->B :　{{ H (X, N_B, K_{Bh}, K_{Bs}), Y }K_{Bs} }K_{Bh}
   (5) A -> B :　{A, Z, {H (A, X, Nb) }W}k'
   注意:Z 和 W 有 Public key relation
4. 建立訊息關聯表，此安全協定的檢查也不需要用到，在此省略。
5. 檢視攻擊訊息(1)，我們可以輕鬆的產生，所以將 N_B, K_{Bh}, K_{Ba}加入Intruder's knowledge。(即是加入訊息(2)和(3))
6. 檢視攻擊訊息(4)，我們的知識庫擁有 Nb, KBh, KBs，所以能順利的產生訊息(4)。
7. 檢視攻擊訊息(5)，我們所擁有的知識庫有 A, Nb，也能利用知識庫所擁有的元素生成 k' ，所以能生成 Message 5。
8. 此 Protocol 有交換 Key K，且已經被我們用元素變數 Y 取代。檢查知識庫發現我們有此金鑰，又已將所有的訊息都成功產生，所以此 Protocol 有漏洞。

範例三　AKA Protocol

這是一個神諭使攻擊法的例子，且又有用到公開金鑰，雖然這例子在我們檢查之下用不到訊息關聯表，我們依舊把他提出來，當一個例子。

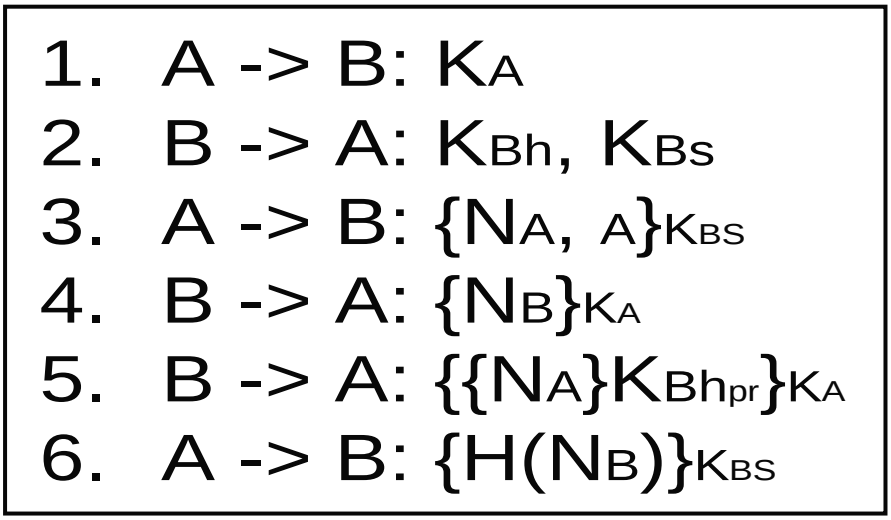


圖 5.3 AKA Protocol

根據 Abadi 提出此協定的攻擊法[6]，看起來這漏洞頗為隱密，但我們可以方法卻很直觀的檢測出協定有問題。下面是 Abadi 提出的攻擊路徑:

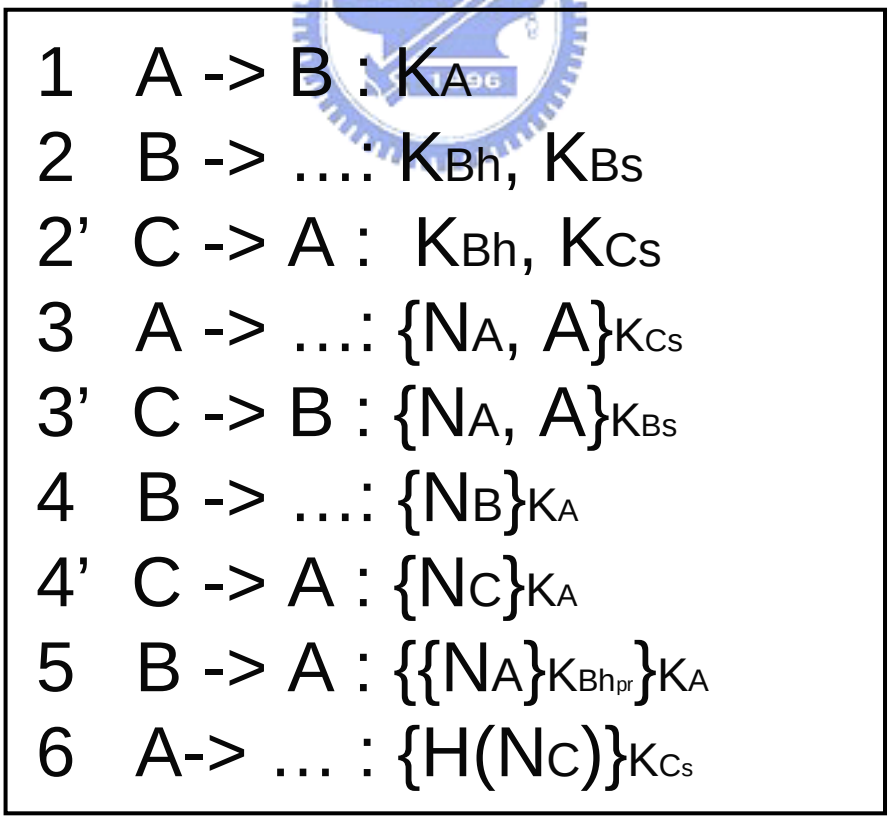


圖 5.4 An attack on AKA Protocol

演算法步驟:

1. 初始設定。
2. 選擇攻擊目標 A，列出攻擊訊息為訊息(2), (4), (5)。
3. 重新改寫攻擊訊息如下:
   (2) B -> A : X, Y
   (4) B -> A : {Z }K_A
   (5) B -> A : { { NA }$_W$ }K_A
   PS: X和W需有public key relation
4. 建立訊息關聯表，因此驗證暫時用不到，故在此省略。
5. 攻擊訊息(2)，將訊息(1)加入攻擊者知識庫中。
6. 因訊息(2)皆為元素變數，我們可以輕鬆產生。
7. 攻擊訊息(4)，將訊息(3)加入攻擊者知識庫。
8. 檢察知識庫，我們曾在訊息(1)取得Ka，所以我們能產生訊息(4)。
9. 攻擊訊息(5)，檢查攻擊者知識庫，可以成功產生{ { NA }$_W$ }K_A，其中 W為任意數只需與X滿足public key relation。
10. 我們產生所有需產生的訊息，進入最後判斷。
11. 此 Protocol 沒有交換 Key，所以此安全協定有漏洞。

在看過前面幾個例子之後，相信已經對我們的演算法有了基本的認識，除了訊息關聯表部分，最後我們舉了兩個代表性的例子，需要用到訊息關聯表的方法。相信在看過這兩個例子之後能對我們的演算法有充分的認識。

範例四 Otway-Rees Protocol

Otway-Rees Protocol 除了有通常我們見到類型漏洞，還有神諭使漏洞。我們現在假定沒有類型漏洞的情況下(在初始條件下設定)，我們來看我們的演算法如何找出一條攻擊途徑。(註:若沒有設定不會有類型漏洞的話，本演算法會先檢驗出類型漏洞)

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 5.5 Otway-Rees Protocol

1. 設定初始環境，不允許類型漏洞。即每個元素有自己的類別，不會發生如把 Nonce 當成 Key 的情形。
2. 我們選擇攻擊目標為 A。
3. 因為我們攻擊 A，觀察協定可以發現我們僅需要產生訊息 4。
4. 改寫攻擊訊息如下:

   4. Na, {Na, X}Ka

5. 建立訊息關聯表:

(1, 2, B, X, Y) : Nx, X, Y, { Nx, X, Y }Kx // Nx, X, Y, { Nx, X, Y }Kx, Ny, { Nx, X, Y }Ky
(2, 3, S, X, Y) : Nx, X, Y, { Nx, X, Y }Kx, Ny, { Nx, X, Y }Ky // Nx, { Nx, Kab}Kx, {Ny, Kab}Ky
(3, 4, B, X, Y) : Nx, { Nx, Kxy }Kx, {Ny, Kxy}Ky // Nx, { Nx, Kxy}Kx

6. 攻擊訊息(4)，將訊息(1) ~ (3)都加入攻擊者知識庫。
7. 嘗試製造訊息 4，雖然我們擁有加密元素{Na, Kab}Ka，但 Kab 為交換金鑰不在知識庫中，所以視為無法產生該訊息。

   (注: 此外又因為我們不允許 Type Flaw， 否則可以使用{Na, A, B}Ka，視為成功產生。)
8. 查詢訊息關聯表送出為(4)的法則: (3, 4, B, X, Y)，當輸出為 Na, {Na, X}Ka，

   輸入為 Na, {Na, X}Ka, {Ny, X}Ky。
9. 若我們 y 用參數 c 帶入: Na, {Na, X}Ka, {Nc, X}Kc。

   (C 為此網域合法的任意第三者，且為攻擊者攻破)。
10. 進行對話函式(A,C)，(C, A)並將所有的訊息加入攻擊者知識庫。
11. 檢視攻擊者知識庫，我們可以成功產生 Na, {Na, X}Ka, {Nc, X}Kc。
12. 我們成功產生所有訊息，進行最後判斷
13. 我們可以產生所有 Message，而且也能取得 Key(因為有{Nc, X}Kc，且我們擁有金鑰 Kc)，所以此協定有漏洞。

範例五 Woo and Lam Authentication Protocol

這也是一個有名的安全協定，他遭受了神諭使攻擊法，我們將它當成我們最後的例子，讓大家更加熟練訊息關係表的使用。

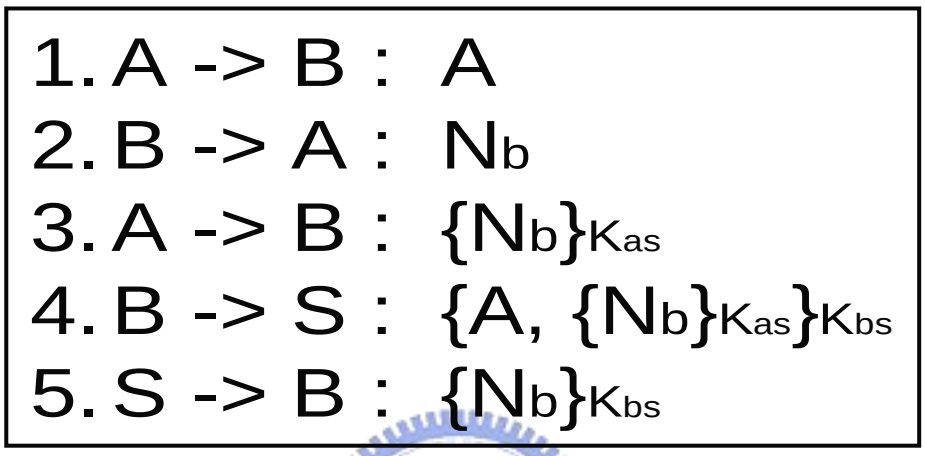


圖 5.6 Woo and Lam Authentication Protocol

演算法則:

1. 設定初始環境。
2. 我們選擇攻擊目標為 B。
3. 因為我們攻擊 B，我們需要產生訊息(1), (3), (5)。
4. 改寫攻擊訊息如下:
    1. A
    3. {Nb}X
    5. {Nb}Kbs

5. 建立訊息關聯表:

(1, 2, B, X, Y) : X // Ny

(2, 3, A, X, Y) : Ny // {Ny}Kxs

(3, 4, B, X, Y) : X, {Ny}Kxs // {X, {Ny}Kxs}Kys

(4, 5, S, X, Y) : {X, {Ny}Kxs}Kys // {Ny}Kys

6. 攻擊訊息(1)，成功產生訊息。

7. 攻擊訊息(3)，將訊息(2)加入攻擊者知識庫中。

8. 查詢攻擊者知識庫可以成功產生。

9. 攻擊訊息(5)，將訊息(4)加入攻擊者知識庫之中。

10. 檢察知識庫，仍無法產生該訊息。

11. 查詢訊息關聯表送出為(5)的法則: (4, 5, S, X, Y)，當輸出為{Nb}Kbs 輸入為{X, {Nb}Kxs}Kbs

12. 若 X 用參數 C 帶入{C, {Nb}Kcs}Kbs。

13. 進行(C, B), (B, C)對話函式，並將所有的資訊加入攻擊者知識庫。

14. 檢視知識庫，可以成功產生{C, {Nb}Kcs}Kbs。

15. 我們可以產生所有 Message 且此安全協定沒有交換金鑰，所以此協定有漏洞。

第六章 結論

6.1 結論

看完了前面章節的方法還有範例之後，可以知道藉由我們所設計的方法，試著根據不同情況用適當的腳色來進行收尋，可以有效的找出潛藏在訊息彼此之間的關連性，利用這種關連性，我們可以很方便又迅速的確認是否安全協定會被惡意的製造出一些具有危害的訊息。

除此之外，在提供了一開始能進行環境之本設定的功能，使的整個方法更具有彈性，能適用的情形更加的廣泛。隨著不同情況下前提環境有所差異，可以僅修改一開始的初始環境設定，而不需修改到演算法核心，可以有效的適應各種類型的環境，相當具有彈性，不需要花費太多餘的時間當一些狀況改變需要對方法進行通盤的修改。

6.2 未來展望

我們提出的方法雖然看起來簡單又實用，但是仍希望能在不違反簡單易懂又有效的前提之下提升他的能力，解決更多最好是所有的問題，在完成這篇論文時，有些所面對的問題尚未解決，在最後的這個部份，提出來做為之後更進一步發展的目標。

隨著各式各樣的安全協定的出現，其中有的採用許多特殊的方法，如用到 Hashing 函式，數位簽章，Exclusive or 運算…等，這對我們的方法來講，尚無法對此類安全協定進行驗證。因為我們目前只能針對基本的對稱和非對稱加密演算法。但為了應付推陳出新的安全協定，這方面的延伸是必須的。

除此之外，我們的方法雖然當可以成功假造出訊息，得知安全協定有漏洞，但卻無法列出攻擊的步驟。這方面的修改是必須的，能夠讓人們了解如果有漏洞，攻擊的步驟為何。更進一步的思考，是否當一個安全協定有多個漏洞的時候，我們能夠將它全部都列出。

再來提到初始設定部份，到底哪些範圍的需求和設定，要放到一開始的設定中，哪些的補強就需要改變到演算法整體，這也是很重要的問題。如何有效的分類規劃初始設定，能有助於演算法核心更有效的讀取那些值以供利用。

本篇論文雖然能藉由程式的撰寫，讓電腦自動化的檢查，但是出始設定，關聯表的建立都仍需要藉由手動的設定，如何能在不削弱能力之下，讓自動化的程度更高，也是極需探討的。

最後也希望看過這篇論文的各位，如果對這方面有所想法與興趣，能提供給我們一些意見，或是針對這方法進行改良，讓他越來越好，能在網路安全協定領域有所幫助。

參考文獻

[1] J. Clark and J. Jacob, “A Survey of Authentication Protocol Literature: Version 1.0”, 1997.

[2] M. Burrows, M. Abadi, and R. Needham, “A logic of authentication”, ACM Trans. Computer Systems, Vol. 8, No. 1, pp. 18-36, 1990.

[3] W. Yang, “Uncovering Attacks On Security Protocols”, in Proceeding of International Conf. Information Technology and Applications, Sydney, Australia, pp. 4-7, July 2005.

[4] M. Debbabi, M. Mejri, N. Tawbi, and I. Yahmadi “Formal automatic verification of authentication cryptographic protocols” In 1st IEEE International Conference on Formal Engineering Methods (ICFEM'97). IEEE, 1997.

[5] C. A. R. Hoare. An axiomatic basis for computer programming. Communications of the ACM, 12:576--583, 1969.

[6] Abadi, M. Explicit Communication Revisited: Two New Attacks on Authentication Protocols. IEEE Transactions on Software Engineering, 23 (3) (1997), 185-186.

[7] D.M.Nessett. "A critique of the Burrows, Abadi and Needham logic". Operating Systems Review 24 (1990), 35-38.

[8] S. Schneider. Security properties and CSP. In Proc. 17th IEEE Symposium on Security & Privacy, pages 174--187, 1996.

[9] J.C. Mitchell, M. Mitchell, U Stern. Automated Analysis of Cryptographic Protocols Using Murj.
In Proc. of Symp. Security and Privacy, IEEE Computer Society Press, 1997.

[10] K. Adi, L. Pene, "Secrecy Correctness for Security Protocols," dfma, pp. 22-29, First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05), 2005.

[11] I-Lung Kao and Randy Chow. "An Efficient and Secure Authentication Protocol Using Uncertified Keys." Operating System Review, 29(3):14-21, July 1995.

[12] Ulf Carlsen. Cryptographic protocol flaws. In Proceedings 7th IEEE Computer Security Foundations Workshop, pages 192-200, 1994.

[13] Qing SH. Twenty years development of security protocols research.
Journal of Sofeware, 2003, 14(10), 1740~1752

[14] D.M.Nessett. "A critique of the Burrows, Abadi and Needham logic".
Operating Systems Review 24 (1990), 35-38.