# 國立交通大學

## 理學院網路學習學程

## 碩 士 論 文

遠端使用者身份驗證之研究

A Study of Authenticating Remote User

研 究 生：蔡佳倫

指導教授：李榮耀　教授

中 華 民 國 九 十 六 年 六 月

# 遠端使用者身份驗證之研究
# A Study of Authenticating remote user

研 究 生：蔡佳倫　　Student：Jia Lun Tsai

指導教授：李榮耀　　Advisor：Dr. Jong Eao Lee

國 立 交 通 大 學

理學院網路學習學程

碩 士 論 文

A Thesis

Submitted to Degree Program of E-Learning

College of Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Degree Program of E-Learning

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

# 遠端使用者身份驗證之研究

學生： 蔡 佳 倫　　指導教授： 李 榮 耀 教授

國立交通大學理學院網路學習學程碩士班

## 摘　　要

隨著網路攻擊事件的日益增多，網路安全的重要性日益增加。為了能有效的管理使用者的存取行為，如何驗證遠端的使用者變得十分的重要。在分散式的網路環境之中，許多系統都是利用傳統的帳號密碼驗證機制來驗證遠端的使用者，這樣的驗證機制比較容易實作，但是卻也造成了許多安全上的問題。因此，在本研究中我們利用智慧卡卡來強化系統的安全性以及改進使用者使用上的問題。

本論文包含四個研究。這四個研究都是以單向雜湊函數為基礎，而且它們都不用存放任何的驗證表。在前面的三個研究，我們指出有些已經發表的驗證方法有著安全性問題與缺失，為了改善這些安全性問題與缺失，我們也提出了改進的驗證機制去避免這些安全性問題與缺失。而第四個研究主要是以第三個研究為基礎，進而提出一個多伺服器的無驗證表驗證機制，這個驗證方法主要是利用一個驗證中心(Registreation Center)來提供伺服器與使用者互相驗證，使用者僅需要在註冊中心註冊一次，便可以使用這些伺服器所提供的服務，而且伺服器與註冊中心都不需要存放任何的驗證表。

關鍵字：驗證、智慧卡、單向雜湊函數

**A Study of Authenticating Remote User**

student：Jia Lun Tsai　　Advisors：Dr. Jong Eao Lee

Degree Program of E-Learning

College of Science

National Chiao Tung University

ABSTRACT

**Because** the network attack is increasing, the network security becomes more and more important. In order to manage the remote users's accesses, it is very important to authenticate the remote users. In the distributed network environment, there are many systems which use the basic authentication scheme to authenticate the remote users. It is easy to implement, but it also has many network security problems. In this study, we use smart card to strengthen the security of the system and improve the using problems of the remote users.

This paper includes four researches. They are all based on one way hash function, and they do not need to store any verification table. In the previous three researches, we demonstrate that some proposed authentication schemes have the security problems and flaws. In order to improve these security problems and flaws, we also propose improved authentication scheme to overcome the security problems and flaws in these proposed authentication scheme. In the fourth research, we propose a multi-server authentication scheme without verification table according to the third research. This authentication scheme helps the servers and the users to authenticate each other by using a registration center. The users only need to register once, and they can get the services provided by the servers. In this multi-server authentication scheme, the servers and the registration center do not need to store any verification table.

Keyword：Authentication, smart card, one-way hash function

# 誌　　謝

這本論文得以完成，需要感謝許許多多人的幫忙。首先要先感謝指導教授李榮耀教授的指導，在他熱心的指導以及鼓勵之下，我才得以完成了這本論文，如果沒他的幫忙，可能到現在還無法看到這本論文的產生，他對我的幫助，除了讓我順利生這本論文外，更讓我了解到學術研究的執著以及樂趣。還記得碩一剛踏進交大時，當時李榮耀老師擔任我們論文研討的指導老師，在課堂上李榮耀老師常常對我們講起他以前指導過一個做網路安全的學生，聽著他口沫橫飛地講著那個曾受他指導的學生、看著他那麼推崇那個學生，間接地也鼓勵了非資訊本科系的我，讓我從碩一開始便下定決心，專心一致地研究起如何利用智慧卡來驗證使用者身份的安全議題，而當時這個研究領域是網路學習專班從未有人撰寫的研究領域，現在回想起來，還對於當時能夠立下那麼大的決心，感到不可思議。當我成為他的研究生之後，老師總是熱心的鼓勵我，記得有一次在一個偶然的機緣下發現到一篇 SCI 期刊論文是有問題的，當時的我還不太能確定那個問題是否成立，於是找了李榮耀老師來共同討論這一個問題，在那次的討論中，李榮耀老師用了很多的方式來考驗我是否真的了解期刊論文的內容，也因為那次反覆的熱烈研究討論之後，我開始了解到學術研究的樂趣。因此，如果有人問我：「誰是在研究所影響你最深的教授？」我會很直接的跟他說：「除了李榮耀教授，別無他人！」

除了李榮耀老師以外，在這邊我還要感謝的是我的父母，在我做研究的時候他們總是給予我鼓勵，在我因為撰寫論文無法回家過節時，也未曾給我任何的責備與苛責，甚至當我難過失意的時候，他們總是第一時間就出現在我的面前，給我鼓勵以及打氣，如果沒有他們，或許我早已放棄了這個論文題目。再來要感謝口試委員-裘性天主任和余啟哲教授，謝謝你們

花費極大的心力來審查學生的碩士論文，有了你們的建議以及批評，才能讓這本論文更加的完美。

回想這兩年的研究生活，因為是在職專班的關係，每天早上除了要去學校教書外，晚上還要進行論文的研讀，放假時還必須要到交大上課而不能夠出去遊玩，漸漸地完成了這份研究，心裡感到十分的喜悅。在此論文完成之際，我要特別對所有一起陪我走過這個歲月的家人、李榮耀教授、專班同學、職場的同事和朋友們致上最高的敬意，沒有你們的支持與付出，我也無法順利地寫出這本論文。

# 目　　錄

# 表　目　錄



# 圖　目　錄

# 第一章　緒　論

隨著科技的日新月異，如何驗證遠端的使用者身份一直是網路安全中一個非常重要而且基本的研究領域，本論文研究主要是利用智慧卡來當作儲存工具，用來協助網路系統去驗證遠端的使用者身份，希望透過本論文之研究能夠幫助網路系統撰寫人員撰寫出更安全的使用者驗證系統，以保護網路系統的安全性。

本章針對研究背景與動機、研究目的、論文架構做探討。

## 1.1　研究背景與動機

隨著網際網路的蓬勃發展，漸漸有許許多多的系統搬移到網路上，希望利用網路的便利性可以讓遠端的使用者互相的分享資訊，因為在網路上所有的資料與資訊都可以在共通的網路上傳遞，所以任何人都可以很容易取得網路上分享的資訊，也因為如此，讓攻擊者的攻擊重心也漸漸轉移到網路上。透過網路，攻擊者可以將攻擊的範圍拓展到全世界的各個角落，所造成的危害也比傳統以檔案的方式傳送病毒要來的有破壞力，因此，近幾年來網路攻擊事件逐年增多，讓網路安全變得越來越重要。

使用者的認證(Authentication)問題一直是網路安全中一個基本而且重要的研究議題，因為如何驗證遠端的使用者一直是網路系統的第一道防線，透過系統驗證使用者的機制，可以有效地保護系統中重要的資料不被攻擊者所盜用或是竊取。以目前的網路環境來說，管理使用者最有效、最簡單的方式是以傳統帳號密碼的驗證方式來驗證遠端使用者，這樣的驗證方式實作起來雖然簡單，但是卻常因為使用者為了自身的方便，僅使用短小、具有某種意義的密碼來登入系統，讓系統管理人員大傷腦筋，雖然可以透過系統的註冊設定或是公司安全政策的訂定來要求使用者使用長串、無意義的密碼，卻也無形中造成使用者使用上及記憶上的困難，讓使用者有可能將這些密碼放置於隨手可得的地方，反而產生了新的安全性問題，此外，以傳統帳號密碼來驗證遠端使用者的驗證方式在伺服器上都必須要存放一個使用者驗證表來驗證遠端的使用者，如何保障這一個驗證表的安全性也成為了系統管理人員的一個十分頭痛的問題，如果這個驗證表因為系統的安全性問題或是內部人員外洩的關係，而讓攻擊者所竊取，將會導致系統暴露在極不安全的環境之下。

## 1.2　研究目的

有鑒於上述的研究背景與動機，著手研究如何利用智慧卡來驗證遠端的使用者身份，本論文一共分成了四個部分，前三個部份針對目前已經發表的期刊論文進行分析以及改進，第四部份則是以第三部分之研究為基礎，進而提出一個多伺服器的驗證方法，這四部份的研究說明如下：

## 1. 改善以ElGamal為基礎的使用者驗證機制

近年來，生物驗證技術(biometrics)逐漸加入到智慧卡的技術之中，如何將生物驗證技術融入到使用者身份驗證也逐漸的變成了這個領域的研究重心，西元2002，學者Lee等人[12]為了改善西元2000年學者Hwang等人[17]所提出以ElGamal為基礎的無認證表驗證機制有著使用者假冒攻擊(Impersonation Attack)的安全性問題，進而提出了一種融入指紋技術(Fingerprint)的無驗證表驗證機制，這個改進的驗證機制也是以ElGamal公開鑰匙加密演算法為基礎，而且在伺服器上不需要存放任何的使用者驗證表，很不幸的，西元2004年學者Lin等人[4]發現這個驗證機制有著使用者假冒攻擊(Impersonation Attack)的問題，他們提出了一種融入生物驗證技術(biometrics)的改進驗證機制，然而，西元2007年學者Khan等人[15]發現這樣的驗證機制並沒有提供使用者與伺服器相互驗證(僅提供伺服器驗證使用者的功能)。為了有效改善這個驗証方法的缺點，學者Khan等人提出了一種改善的驗證機制，這個改善的驗證機制提供了相互驗證的功能，讓使用者跟伺服器可以互相驗證對方的身分，不幸的是，本研究發現學者Lin等人所提出的驗證機制與學者Khan等人所提出的驗證機制都有著使用者偽造攻擊(Impersonation attack)的安全性問題存在，為了改善他們的安全性問題，本研究提出了一個新的改善驗證機制。

## 2.改善學者Chang和學者Chang所提出的驗證機制

西元2000年，學者Peyavian和學者Zunic[16]提出了一種如何在不安全的網路環境下如何只利用單向雜湊函數去驗證遠端使用者的驗證機制，然而，西元2002年學者Hwang和學者Yeh[10]卻指出學者Peyravian和學者Zunic所提出的驗證機制有著許許多多的安全性問題存在，為了改善這個問題，他們也提出了一種改善的驗證機制，可惜的是，西元2003年學者Lin和Hwang[6]指出學者Hwang和Yeh的驗證機制也有著許許多多的安全性問題，因此，他們也提出了一種改善的驗證機制來改善這個驗證機制的安全性問題，不幸的是，西元2005年學者Chang和學者Chang[24]指出學者Lin和Hwang所提出的驗證機制有使用者驗證表被竊(Stolen-verifier Attack)的安全性問題，為了改善這個問題，學者Chang和學者Chang也提出了兩種不同的驗證機制去避免驗證表被竊攻擊。然而，本研究發現學者Chang和學者Chang所提出的兩個驗證機制中，有一個驗證機制有著使用者偽裝攻擊(Impersonation Attack)的安全性問題，為了改善這個安全性問題，本研究提出一個新的改善驗證機制。

## 3.提出一個以隨機亂數為基礎(Nonce-based)的無驗證表驗證機制

傳統的帳號密碼驗證方式在伺服器上必須要存放一個使用者驗證表，可是如果這個使用者驗證表因為某些安全的因素而外洩，將會導致許許多多安全性的問題發生。為了避免伺服器的驗證表被竊的問題，西元2005年學者Lee等人[18]與學者Chen等人[23]分別提出了以隨機亂數(nonce)為基礎的無認證表驗證機制，本研究仔細分析這兩個驗證機制，發現他們有著一些缺陷與安全性問題存在，並且提出了一個以隨機亂數為基礎(nonce-based)的無驗證表驗證機制。

### 4. 提出一個多伺服器的無認證表驗證機制

上述以隨機亂數(nonce)為基礎的無驗證表驗證機制雖然解決了使用者驗證表被竊的問題，但是，這些的機制如果在使用者想要使用多伺服器的服務時，卻有個明顯的缺陷，那就是使用者必須要在這些伺服器上一一註冊，因此，本研究利用研究三所提出以隨機亂數(nonce)為基礎的無驗證表認證機制，進而提出一個多伺服器的無驗證表驗證機制，在這個驗證機制中，伺服器跟註冊中心都不需要存放任何的驗證表，而且使用者僅需要在註冊中心註冊一次，便可以使用多伺服器的網路服務，大大降低了使用者重複註冊的問題。

## 1.3 論文架構

本論文共分為七個章節，各章節的內容說明如下：

第一章節說明本研究的背景、動機與研究目的。

第二章節說明本研究所需的相關文獻探討。

第四章節分析以生物驗證技術(biometrics)為基礎的兩個驗證機制之安全性問題，並提出改進的驗證機制。

第四章節分析學者 Chang 和學者 Chang 所提出的驗證機制之安全性問題，並提出改進的驗證機制。

第五章節分析目前以隨機亂數(nonce)為基礎的無驗證表驗證機制之缺失，並且提出自己的驗證機制。

第六章節利用第五章節所提出的驗證機制，進而提出一種多伺服器的無驗證表驗證機制。

第七章節為本研究的結論與未來發展。

# 第二章 文獻探討

## 2.1 驗證(Authentication)

如何在網路上確認對方的身分一直是網路安全一個十分重要的研究議題，當使用者希望得到某項服務而需要登入到某台伺服器時，首先他必須要先通過伺服器的驗證程序。目前網路上驗證使用者身份通常有下面三種模式[39]：

### 2.1.1 單向驗證(One-Way Authentication)

這是目前網路上最常見的驗證方式，大部分的系統都是利用這樣的方式來驗證遠端使用者的身分，因為這樣的驗證方式簡單、容易實作，在這個模式中使用者必須事先在伺服器上註冊使用者的帳號密碼，當使用者想要登入時，使用者必須先提供使用者的帳號與密碼給伺服器，方可跟伺服器作驗證確認動作，進而登入系統或是使用該伺服器所提供之服務，不過，這樣的驗證方式並沒有辦法驗證伺服器的身份是否合法，因此，當攻擊者偽裝成一台合法的伺服器時，使用者無法察覺這一台偽裝的伺服器是否合法，所以有著伺服器偽裝攻擊(Server Spoofing Attack)的安全性問題存在。

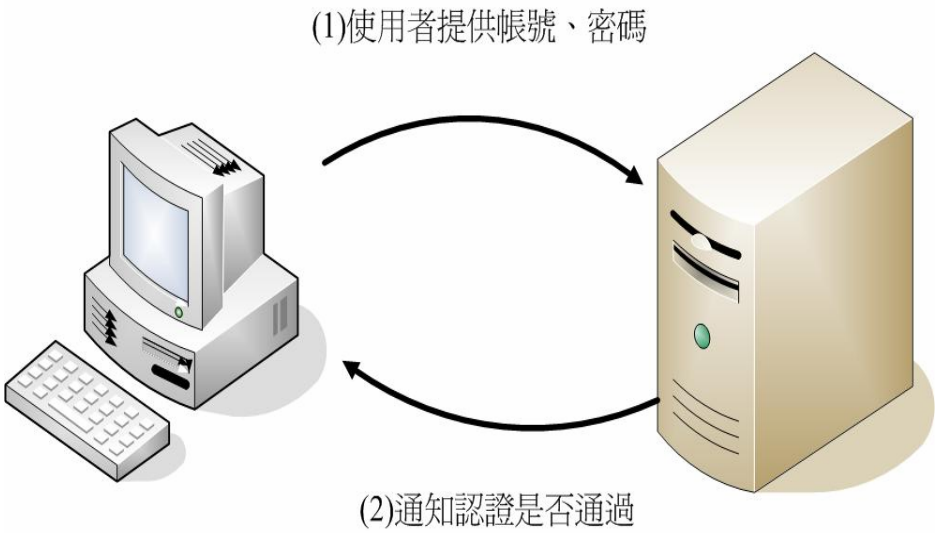


**圖 2.1：單向認證**

### 2.1.2 雙向驗證(Mutual Authentication)

前面所介紹的單向認證在安全上並不是很安全，因為它僅僅提供伺服器如何去驗證遠端的使用者身分，卻未提供使用者去驗證伺服器的身分是否合法，如果一個

攻擊者假冒伺服器來欺騙遠端的使用者，將會發生使用者被攻擊者所假冒的伺服器所欺騙導致使用者的帳號密碼或重要資料等被攻擊者所盜取的問題，因此，為了避免這樣的問題，通常我們會希望使用者與伺服器都必須互相提供帳號密碼給對方，以確認驗證的雙方身分是否合法，不過這類型的驗證方式有個嚴重的安全性問題，那就是使用者要如何保有伺服器的帳號密碼，因為萬一這組伺服器的帳號密碼遺失或是被合法使用者所盜用時，反而可以讓攻擊者可以假冒合法的伺服器去欺騙其他的使用者，因此，常見的保護方式是利用智慧卡來存放伺服器的帳號密碼，以避免一個攻擊者在擁有一個合法使用者的智慧卡時，盜用這組帳號密碼來假冒伺服器，而且最好每個使用者所擁有的伺服器帳號密碼皆不同，。

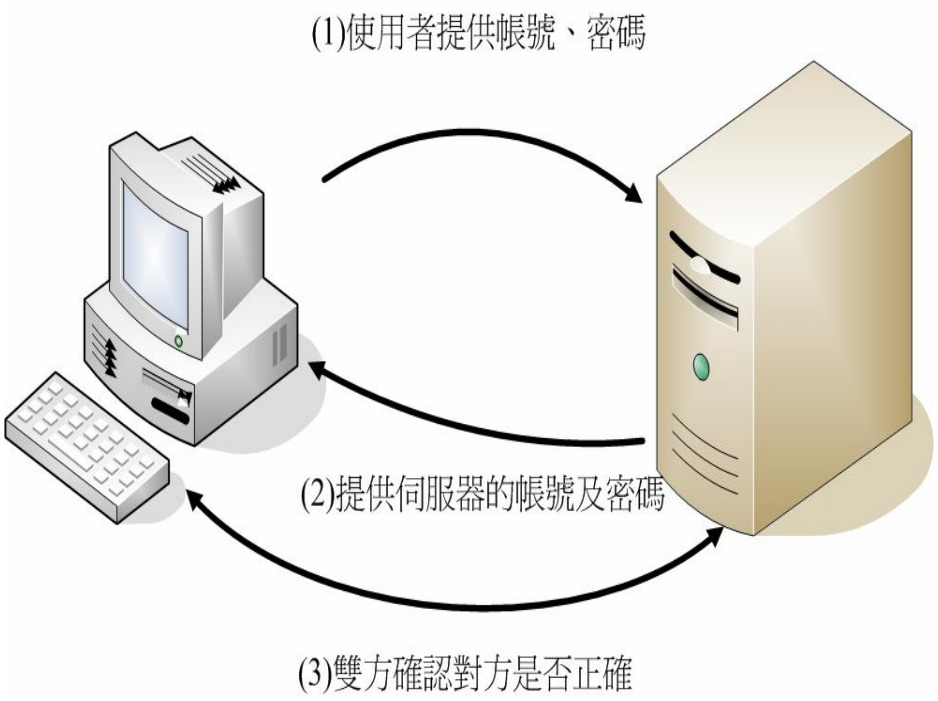


**圖 2.2：雙向認證**

## 2.1.3 使用公正的第三者驗證 (Trusted Third-party Authentication)

這個驗證方法與前面的驗證方式一樣，都是希望能夠讓伺服器與使用者去驗證對方彼此的身分是否合法，但是在方法上卻不是十分相同，其特色主要是在驗證的過程中加入了一個彼此都相信的第三公正單位，透過這個第三公正單位來幫忙驗證彼此的身分。因此，當使用者要登入時，使用者與伺服器都必須先通過這一個第三公正單位的確認。當第三公正單位確認過需要確認者的身分是合法時，第三公正單位會頒發一個通過認證資訊給需要確認者，當使用者跟伺服器拿到這個認證資訊後，能夠利用它來讓通訊的對方知道它的身分是合法的，不過，這樣的驗證方法有個明顯的缺點，那就是在這個驗證方法中第三公正單位的安全性變得十分的重要，如果這個第三公正單位被攻擊者所入侵，將會導致所有的伺服器及使用者暴露在極不安全的環境之下。

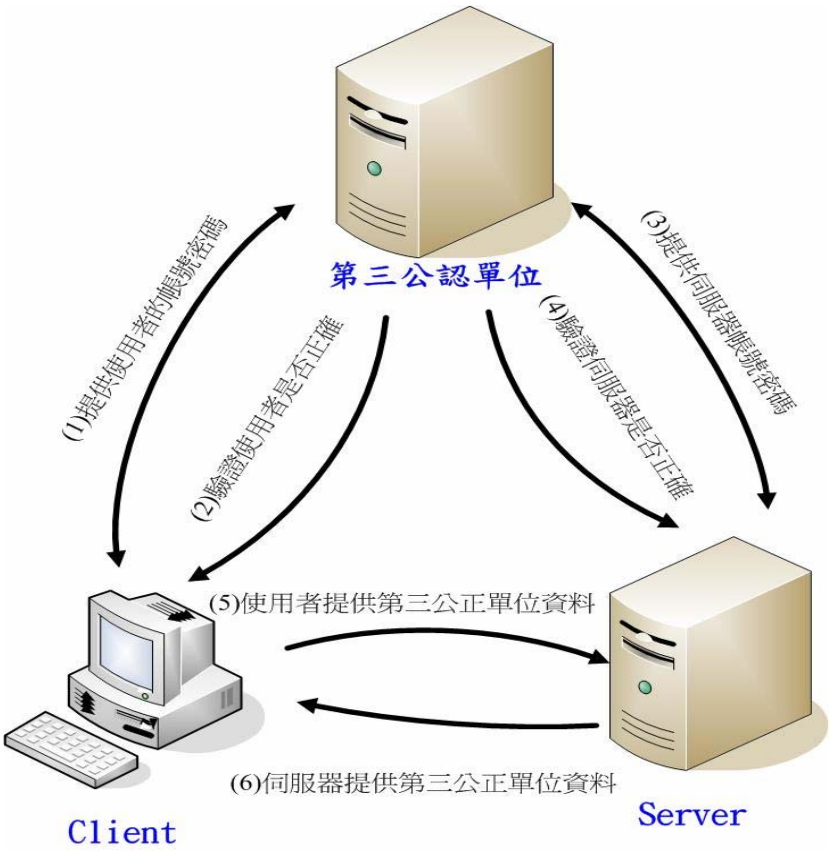


**圖 2.3：使用公正的第三者認證**

## 2.2 單向雜湊函數(One-way hash function)

單向雜湊函數是一種可以將任意長度的訊息加以濃縮、轉換，使其成為一個固定長度的訊息的加密函數，所產生的訊息通常叫做雜湊值(Hash Value)[25]，而我們通常利用雜湊值來驗證某項資訊是否正確無誤。單向雜湊函數本身具備不可逆的單向〔One-Way〕特性，亦即如果將一段訊息經過單向雜湊函數之後所產生的雜湊值，是無法將其還原成原本的訊息。此外，單向雜湊函數還必須具備唯一性，每份文件計算出來的雜湊值必須要是唯一值，欲找出另外一份文件可以輸出相同的雜湊值是不可能的。目前常見的單向雜湊函數主要有MD5及SHA[25][39]。

### 2.2.1 MD5(Message Digest Algorithm version 5)

MD 系列的雜湊函數皆是由學者 Rivest 所發展，MD5 是在西元 1992 年由 MD4 改良而成，運算速度比 MD4 慢了一點，壓縮過程也從 MD4 的三個回合，增加為四個回合，其他都與 MD4 相同，其 C 語言實作基礎被定義在 RFC1321 之中。加密時首先會把訊息長度 mod 2 後，將產生訊息摘要放置於訊息的最後，再把訊息切割為數個 512 位元的區塊，並且檢查最後一個區塊的長度是否為 448 位元，若不是的話，會填補明文，讓最後一個區塊的長度為 448 位元，然後以 512 位元區塊為單位，進行運算，最後產生一個長度為 128 位元的摘要[26][37]。

原本 MD5 被認為是一種很安全的雜湊函數，並且廣泛的使用在各種電腦系統之中，不幸的是，在西元 2004 年 8 月 17 日，美國加州聖巴巴拉所召開的國際密碼學會議中安排了三場關於雜湊函數的特別報告。在國際著名密碼學家 Eli Biham 和 Antoine Joux 相繼做了對 SHA-1 的分析與給出 SHA-0 的一個碰撞之後，來自山東大學的王小雲教授做了破譯 MD5、HAVAL-128、 MD4 和 RIPEMD 算法的報告。也因為這樣讓美國已經考慮再制定新的雜湊函數技術，因為隨著它們的發現將使駭客更能在電腦程式碼中放置後門或是捏造電子簽章。美國國家技術與標準局（NIST）更於西元 2004 年 8 月 24 日針對王小雲等人所提出的報告發表了專門評論，評論的大致內容：「在最近的國際密碼學會議（Crypto 2004）上，有研究人員宣佈他們發現了破解數種 HASH 算法的方法，其中包括 MD4、MD5、HAVAL-128、RIPEMD 還有 SHA-0。美國國家技術與標準局表明，於西元 1994 年替代 SHA-0 成為聯邦信息處理標準的 SHA-1 的減弱條件的變種算法能夠被破解；但完整的 SHA-1 並沒有被破解，也沒有找到 SHA-1 的碰撞。研究結果說明 SHA-1 的安全性暫時沒有問題，但隨著技術的發展，技術與標準局計劃在西元 2010 年之前逐步淘汰 SHA-1，換用其他更長更安全的算法（如 SHA-224、SHA-256、SHA-384 和 SHA-512）來替代。」[26][27][37]

## 2.2.2 SHA(Secure Hash Algorithm, SHA)

SHA(Secure Hash Algorithm)是由美國「國家安全局(National Security Agency)」與「國家技術與標準局(National Institute of Standard and Technology, NIST)」於西元 1993 所公佈的，並成為美國聯邦資料處理標準(Federal Information Processing Standard，FIPS)的 FIPS 180。SHA 的設計主要是以 MD4 為其主要依據，在西元 1995 年修正部份的缺失，制定了 FIPS-1，亦即 SHA-1，其 C 語言實作基礎被定義在 RFC3174 之中。SHA-1 的加密方式與 MD5 差不多，只是區塊處理長度以及輸出的摘要長度為 160 位元。然而為了因應電腦速度越來越快的影響以及避免碰撞的發生，美國國家標準與技術中心於西元 2002 年八月也發表 SHA-224、SHA-256、SHA-384 與 SHA-512 等雜湊函數，並列入聯邦資料處理標準，以增加其安全性[26]。

## 2.3 ElGamal 公開加密演算法

公開加密演算法是密碼學的一大突破，開創了密碼學一個新的章程，第一個有非對稱性加密構想的是 Diffie 與 Hellman 這兩位學者，他們是在西元 1976 年提出了這樣的想法，但是他們卻不知可不可行，直到西元 1978 年美國麻省理工學院的 Rivest、Shamir 及 Adleman 提出了第一個公開加密演算法—RSA[35][37]。

ElGamal 公開加密演算法是一種建立於離散對數問題上的非對稱加密演算法，它是在西元 1985 年被提出的，其特色在於它將隨機加密(Radomized encryption)這樣的方法加入到加密的過程中，因此，一個相同的明文在經過每次的加密後，都會因為一個隨機產生的亂數而產生不同的密文，但是這些密文都可以解密回原來的明文，公開加密演算法 ElGamal 加解密過程如下：

**密鑰建立：**

在使用 ElGamal 非對稱加密演算法時，雙方必須事先共有一個非常大的質數 p 與 p 的原根 G，並且在通訊前傳送者必須要事先取得接受者的公開鑰匙，這把公開鑰匙的計算方法是先讓每個使用者隨機產生一個正整數 X，$p-2 \geq X \geq 2$，然後計算：

$$Y = G^X \bmod p$$

**加密：**

當得到了這一把公開鑰匙後，使用者與傳送者之間便可以進行通訊，明文 M 在此被視為一個整數，$p-2 \geq M \geq 0$，通訊時，傳送者先隨機產生一個亂數 K，$p-2 \geq K \geq 1$，之後計算：

$$C_1 = G^K \bmod p$$
$$C_2 = MY^K \bmod p$$

**解密：**

當計算出$C_1$和$C_2$後，傳送者將$C_1$、$C_2$傳送到接受者手中，當接受者收到$C_1$、$C_2$，接受者便可以利用私密鑰匙X去計算出M，靠著計算[35][37][39]：

$$M = C_2 C_1^{-X} \bmod p$$

## 2.4 網路攻擊法

### 2.4.1 竊聽攻擊(Eavesdrop Attack)

竊聽攻擊(Eavesdrop Attack)是攻擊者常使用的攻擊方法，造成這個攻擊方式的主要原因是因為網路上傳遞的訊息都是公開的，因此，攻擊者可以藉由監聽網路的封包進而取得有用的登入資訊。這樣的攻擊法對於以傳統帳號密碼來驗證遠端使用者身份的系統特別有效，因此，通常在傳送的過程中我們會要求傳送的資訊都必須先經過加密之後才能夠傳送到對方的手上，以避免造成重要資訊被攻擊者所竊取[29]，不過，如果加密的帳號密碼沒有經過時間或隨機亂數來改變，在傳送時還是會保持一個固定的值，所以攻擊者還是可以在竊聽後，以重送攻擊(Reply Attack)的方式來入侵系統，所以為了避免重送攻擊的危害，一般常見的改善方法都是在傳送的過程中加入隨機亂數(nonce)或時間戳記(timestamp)來改變傳送的資訊。

### 2.4.2 密碼猜測攻擊(Password Guessing Attack)

以傳統帳號密碼來驗證使用者身份的機制是目前網路上常見的驗證機制，但是這樣的驗證方式有個很嚴重的問題─使用者記憶的問題。通常人腦的記憶力有限，要求使用者記憶長串、無意義的密碼是十分的困難，這時，密碼猜測攻擊(Password Guessing Attack)就變成了攻擊者一個很有效的攻擊方式。密碼猜測攻擊主要是利用猜測的方式來獲取使用者的密碼。利用密碼猜測攻擊所採用的攻擊方式可以將密碼猜測攻擊區分為在網路上直接進行密碼猜測的線上式密碼猜測攻擊(On-Line Password Guessing Attack)，以及先盜取加密的密碼後再以離線的方式來猜測密碼猜測的離線式密碼猜測攻擊(Off-Line Password Guessing Attack)兩種。此外，密碼猜測攻擊又可以以如何進行猜測來做區分，區分為以某些常用的密碼或是字句來進行猜測的字典攻擊(Dictionary Attack)，以及窮盡各種組合所產生的字詞進行破解的暴力攻擊(Brute-Force Attack)兩種。以字典攻擊的方式來攻擊系統比較有效率，因為使用者通常習慣以自身相關的相關資訊來當作密碼，例如：身分證字號、生日、電話號碼等等，而且往往沒有定期更改密碼的習慣[29][31]。

### 2.4.3 重送攻擊(Replay Attack)

重送攻擊(Reply Attack)主要是利用竊聽技術來觀察或紀錄使用者和伺服器之間通訊的資訊，從中找出有關帳號密碼的通訊資訊，並在合法的時間內利用這些相關的帳號密碼通訊資料，重新傳送到伺服器，以假冒合法使用者入侵系統，進而取得系統內部資訊。為了防止重送攻擊，一般系統會使用時間戳記(Timestamp)或是用隨機亂數(Nonce)的方式來隨機地改變傳送的資訊，讓每次傳送的資訊都不一樣，這樣一來，攻擊者便無法藉由重新將驗證資訊傳送給伺服器，進而入侵系統。通常，利用時間戳記(timestamp)的方式的驗證方式比較不太適合分散式的網路環境，因為在實際的網路環境中，要求遠端的伺服器與遠端的使用者在時間的頻率上必須保持同步是有相當程度的困難，因此，以時間戳記(timestamp)為主的驗證機制大多使用於一些能夠要求時間同步的系統之中。而以隨機亂數(Nonce)為主的系統則比較適合目前分散式的網路環境，因為它並不需要要求使用者與伺服器在時間上必須同步，而改以隨機亂數(Nonce)來動態地改變傳送的資訊[35]，目前網路上最常見的是挑戰/回答(challenge-response)驗證機制。

### 2.4.4 使用者假冒攻擊(Impersonation Attack)

使用者假冒攻擊(Impersonation Attack)是一種可以讓攻擊者以假造的方式假冒成其他合法使用者入侵系統的攻擊方式。這樣的攻擊方式對於驗證機制來說，是十分的具有殺傷力的。因為大部分的假冒攻擊都不容易被系統管理人員察覺出目前正有攻擊者在假冒其他的合法使用者登入系統，因此，當一個系統具有假冒攻擊的危險性時，將會導致重要的資料輕易的被攻擊者所竊取[30][31][36]。

### 2.4.5 認證表被竊攻擊(Stolen-Verifier Attack)

認證表被竊取攻擊(Stolen-verifier Attack)是目前以智慧卡做為存放工具的驗證方式最想改善的地方，這是因為傳統的帳號密碼驗證機制通常在伺服器中都需要存放一個使用者驗證表(Verifier table)，這一個驗證表上儲存了系統驗證使用者身份時所需的相關資料。如果攻擊者透過入侵或是其他方式竊取到系統內部的認證表時，攻擊者便可以從中取得使用者的帳號密碼，進而假冒使用者入侵系統，此外，攻擊者還可以以修改這個驗證表的方式，假造一組新的使用者帳號密碼來入侵系統。西元1981年學者Lamport[14]提出了一種改善方法，這個方法是先將這個驗證表的使用者密碼經過雜湊函數加密後，再存放於驗證表中，雖然可以讓攻擊者無法在第一時間得知使用者的密碼，但是可惜的是攻擊者還是可以利用離線密碼破解的方式去破解出使用者所擁有的密碼，進而假冒使用者登入系統，或是假造一個合法的使用者帳號密碼存放於這個使用者驗證表，再利用這個假造的使用者帳號密碼登入系統，進而入侵系統[30][36]。

### 2.4.6 伺服器偽裝攻擊(Server Spoofing Attack)

伺服器偽裝攻擊(Server Spoofing Attack)主要發生的原因通常是因為使用者無法確認伺服器的身份是否合法或是因為驗證機制的缺陷，所以導致攻擊者可以偽裝成一台合法的伺服器來欺騙使用者，這個攻擊方法更是以傳統帳號密碼驗證使用者之系統的最大弱點，因為傳統帳號密碼驗證機制通常不提供使用者驗證伺服器的功能。因此，當一個攻擊者偽裝成一台合法的伺服器，而使用者在不知道此伺服器是偽裝的情況下，傳送使用者相關的秘密資訊給偽裝的伺服器，將會導致使用者驗證的資料被攻擊者所竊取、利用 [36]。如果要有效避免這樣的攻擊方法，在驗證方法上就必須要提供驗證伺服器身分的機制，通常改善的方法是使用雙向驗證或是第三公信單位驗證的方式來讓使用者驗證伺服器的身分，以避免使用者遭受偽裝伺服器攻擊(Server Spoofing Attack)的攻擊。

## 2.5 IC卡簡介

IC卡又稱為智慧卡、晶片卡。它主要是在西元1968年由德國Jurgen Dethloff和Helmut Grotrupp所提出，當時他們提出將所謂的積體電路加在卡片中的想法，並獲得了專利權，而最早應用它的是法國電信公司Postal and Telecommunications Services的電話卡產品。因為IC卡兼具方便、易攜、記憶能力高等優點，所以廣為大家所接受，至此，IC卡就慢慢的進入到我們的生活之中，漸漸成為我們生活的一部分。IC卡的規格主要是遵循ISO7816系列的標準，其外觀是一個長寬為 85.6mm x 53.98mm ，厚度是0.8mm的卡片，隨著卡片的構造不同而有不同的功能以及應用方式[28][38]。

### 2.5.1 IC卡的分類

IC卡如果依照內部構造來分類的話，可以分為記憶卡(Memory Card)及智慧卡(Smart Card)，記憶卡比起智慧卡要來的低廉，其記憶體容量大約為256Bytes-64Kbytes，因為本身沒有CPU，所以其上的晶片只具有儲存資料的功能而不能作邏輯以及運算等功能，不過，某些記憶卡為了保護資料的安全性，大多具備了安全邏輯機制，使用者必須輸入密碼才能夠存取卡片上的資訊，在幾次輸入錯誤之後，便將卡片資料鎖死，記憶卡最常見的應用是在儲存資料及門禁上，例如儲值卡、電話卡等安全性不高的應用。而智慧卡因為本身包含有CPU、記憶體及其他運算處理單位，因此卡片可以直接作一些加密的運算。智慧卡上的CPU一般為8位元單晶片處理器，新一代的智慧卡晶片有16位元或是32位元的處理器。因為智慧卡可以直接在卡片上進行直接進行密碼運算(如DES、3DES、AES、RSA)，所以智慧卡比起記憶卡來說，對於存放資料的保護性更高，常應用於一些像是金融等重要資料的存放。一般在習慣性上我們叫具有CPU運算能力的為智慧卡，但有時也有人會稱記憶卡為智慧卡[28][32][33]。

IC卡如果依照讀取的方式來分類的話，主要可以分為接觸式卡(Contact Card)、非接觸式卡(Contactless Card)以及接觸式與非接觸式的混合卡(Combi-Card)三種。接觸式智慧卡是指卡片上的晶片必須接觸讀卡機的讀寫頭才能進行資料的存取，所以具有較高的安全性及正確性，在其上通常有8個金屬接觸點C1~C8，用來與讀卡機進行各種資料的存取，不過，接觸式智慧卡也不是沒有缺點，缺點是因為需要與讀卡機的讀寫頭接觸，因此常因為卡片的抽換動作過於頻繁而發生磨損，所以通常接觸式的卡片的壽命比起非接觸式的卡片壽命要來的短[28][33][38][39]。

非接觸式的卡片主要是利用 RFID(Radio Frequency Identification；射頻辨識系統)技術來達到不需要插卡的功能，RFID 包括 RFID Reader(射頻辨識讀取器)、RFID Tag(射頻辨識標籤)以及電腦系統三個主要因素所組成，這個技術最早出現在第二次世界大戰，當時敵我飛機利用這項技術來發射無線電電波，互相辨識對方身分[38]。一個 RFID Tag 具備儲存資訊與修改資訊的能力，內部結構是由一個積體電路與一個天線組合而成，當標籤接收到無線電頻率之後，便會產生能量來讓整個 RFID Tag 運作，因此他並不需要另外提供電源來供給電量，除此之外，RFID Tag 可以做成各種不同的形狀、大小，讓人感覺不到他的存在，日本甚至利用這樣的技術，將 RFID Tag 作成一塊小晶片置入手機中，讓大學生可以透過這樣的技術來列印成績單或是上課點名等用途。一般人常會將 RFID 與非接觸式智慧卡混為一談，因為部分僅提供射頻辨識器在外型上看起來也非常像 ISO7810 卡片形式，不過，非接觸式智慧卡除了具有 RFID 的技術外，它更具備了 CUP 智慧卡的安全特性，在卡片上可以執行加解密及保護資料的功能，一般熟悉的非接觸式智慧卡如台北的捷運悠遊卡。因為非接觸式的卡片不像接觸式的卡片那樣需要接觸讀卡機，因此使用上來說壽命比起接觸式的卡片來的要長，不過它也不是沒有缺點的，他的主要缺點是沒有多種的加密機制，所以安全性比起接觸式的智慧卡要來的低[32][38]。

## 2.5.2 IC卡的安全機制

大部分的IC卡對於使用者的認證是利用個人驗證碼(Personal Identification Number，簡稱PIN)的系統協定，當使用者想要使用IC卡的時候，通常都必須要先輸入該卡片的個人驗證碼給卡片進行確認工作，為了卡片避免遭受非法的使用者進行猜測密碼，IC卡會在經過連續的幾次錯誤之後，即判斷此使用者為非法使用者，自動的鎖住卡片資料，導致卡片無法繼續使用。另外，智慧卡比起記憶卡更具有安全性，主要原因是因為智慧卡本身具有CPU，所以可以實現各種想要的安全演算法(視該卡片提供的功能而定)，以達到使用的安全性。而且智慧卡為了避免個人認證碼的外洩，在傳輸的過程中通常會先被加密起來，當卡片收到該名使用者的認證碼之後，再經由解密以供計算、比對內部的個人驗證碼，當確認無誤之後即可以使用該卡片。智慧卡上的特殊作業系統(Card OS)，可以對卡內儲存的資料進行控管動作，並且依照不同的使用者權限做不同的存取控制。另外，智慧卡卡片上因為具有密碼運算的功能，所以當使用者需要執行加解密動作或簽章動作時，可以將資料傳送到智慧卡中，在智慧卡中執行運算，大大降低鑰匙因為在電腦上運算時，可能會洩漏、被盜取的風險[28][33][39]。不過，智慧卡也不是不能夠被破解的，破解智慧卡技術可以區分為破壞性攻擊與非破壞性攻擊兩種類型，破壞性攻擊技術通常以反向工程技術為起點，破壞智慧卡的結構進而找出秘密金鑰存放於何處。非破壞性攻擊技術通常需要微處理器和軟體的專業知識，目前常見的非破壞性攻擊技術為電力分析攻擊技術、時間分析攻擊技術、故障分析攻擊技術和暫時性故障分析攻擊技術等四種[34]。

# 第三章　改善以ElGamal為基礎的使用者驗證機制

## 3.1前言

使用者身份驗證是網路系統用來保護系統內部資料的一個保護措施，其中以密碼為主的使用者驗證機制更是廣泛的使用在網路系統之中，因為它具備簡單、容易實作的特性。西元 1981 年，學者 Lamport[14]提出了一種使用驗證表的使用者驗證機制，它可以讓伺服器在不安全的環境之下也能夠驗證遠端的使用者身份，在 Lamport 所提出的驗證機制裡，密碼會先經由單向雜湊函數的加密後再存放到這個使用者驗證表中，不過，如果有攻擊者成功入侵了這一台伺服器，並且盜取出這個使用者驗證表，他就可以利用離線密碼破解的方式去取得使用者的帳號密碼，為了有效避免這個安全性問題的發生，許許多多不同的無驗證表驗證機制被提出。西元 2000 年，學者 Hwang 等人[17]指出了學者 Lamport 所提出的驗證機制容易遭受攻擊者以密碼表被竊攻擊(Stolen-Verifier Attack)所攻擊，提出了一種以 ElGamal 公開鑰匙加密演算法為基礎的無驗證表驗證機制，隨後，學者 Sun 等人[8]也在西元 2000 年提出了一種以單向雜湊函數為基礎的無認證表驗證機制，這些驗證機制因為都不需要在伺服器上存放任何的驗證表，所以可以有效的抵擋驗證表被竊攻擊的發生，不過，這些驗証機制還是存在許許多多的安全性問題，為了讓使用者驗證機制更加的安全，不斷的有學者從事這方面的研究，並且陸陸續續提出了許許多多的改善驗證機制。

西元 2000 年，學者 Chan 等人[5]分析了學者 Hwang 等人所提出以 ElGamal 為基礎的無認證表驗證機制，發現該驗證機制有著使用者假冒攻擊(Impersonation Attack)的安全性問題，透過這個攻擊方法，攻擊者可以在擁有一個合法使用者身份之後，利用這個合法身份假冒成其他的合法使用者入侵系統。西元 2002 年學者 Lee 等人為了改善這個缺點，提出了一個改善的無驗證表驗證機制，這個驗證機制也是以 ElGamal 公開鑰匙加密演算法為基礎，並且融合了指紋技術(fingerprint)，很不幸的是，西元 2004 年學者 Lin 等人[4]發現這個驗證機制還是有使用者假冒攻擊的問題，為了改善這個使用者假冒攻擊的安全性問題，他們提出了一個新的改善驗證機制，這個改善的驗證機制融合了生物驗證技術(biometrics)，並且也是以 ElGamal 公開鑰匙加密演算法為基礎，而且在這個驗證機制中，使用者可以很輕易的選擇及改變他的使用者密碼。可惜的是，西元 2007 年學者 Khan 等人[15]指出 Lin 等人所提出的驗證機制還是有安全性問題，這個安全性問題主要是因為這個驗證方法只提供伺服器如何去驗證遠端使用者的身分，但是卻未提供使用者如何去驗證伺服器身分，因此，攻擊者還是可以偽裝成一台伺服器去欺騙使用者，所以它有著伺服器偽裝攻擊(Server Spoofing Attack)的安全性問題存在，為了改善這個問題，學者 Khan 等人也提出了一個改善的驗證機制去改善這樣的安全性問題。

對於以智慧卡為存放工具的驗證機制來說，密碼可以分為使用者記憶的使用者密碼以及存放於智慧卡中的驗證密碼，在本研究中，本研究發現西元 2004 年學者 Lin 等人[4]所提出的驗證機制和西元 2007 年學者 Khan 等人[15]所提出的驗證機制

都有著使用者假冒攻擊(Impersonation Attack)的安全性問題，任何一個合法的註冊者都可以使用他所擁有智慧卡中的重要資訊，創造出其他合法使用者的重要資訊，進而偽裝成其他的使用者入侵系統，為了有效改善這個安全性問題，本研究也提出了一個新的改善驗證機制去防範使用者偽裝攻擊的發生。

## 3.2 學者 Lin 等人與學者 Khan 等人所提出的驗證機制

這個小節，將會簡單的介紹一下這兩個有安全性問題之驗證機制，這兩個驗證機制都是以 ElGamal 公開鑰匙加密演算法為基礎的驗證機制，第一個驗證機制是由學者 Lin 和學者 Lai 於西元 2004 所提出，另外一個驗證機制是由學者 Khan 和學者 Zhang 於西元 2007 年所提出。在介紹完之後，本研究會針對這兩個驗證機制提出攻擊的方法，不過，在檢視這兩個驗證機制之前，先簡單的定義驗證過程中所需要的符號，在驗證過程中需要用到符號如下表所示：

**表 3.1：符號說明表**

| 符號 | 說明 |
|---|---|
| h() | 單向雜湊函數 |
| XS | 伺服器的私密鑰匙 |
| ⊕ | Xor 運算 |
| ID, PW | 使用者的帳號密碼 |
| \|\| | 連結 |
| N | 伺服器與使用者所產生的亂數 |
| X->Y:M | X 傳送 M 給 Y |
| U, S | 代表使用者與伺服器 |
| $S_i$ | 使用者的指紋 |
| P | 一個非常大的質數 |
| $Y_A$’ | 使用者與伺服器互相驗證用之網路密碼 |

### 3.2.1 學者 Lin 等人所提出的驗證機制

西元 2004，學者 Lin 和學者 Lai[4]提出了一個以 ElGamal 公開鑰匙加密演算法為基礎的驗證機制，這個驗證機制融合了生物驗證技術(biometrics)，並且使用智慧卡來當作存放工具，整個驗證流程可以簡單的分成註冊期(Registration Phasee)、登入期(Login Phase)、驗證期(Authentication Phase)、密碼改變期(Password Change Phase)等四個時期，下面將會簡單的介紹他們的驗證機制流程。

### 3.2.1.1 註冊期(Registration Phase)

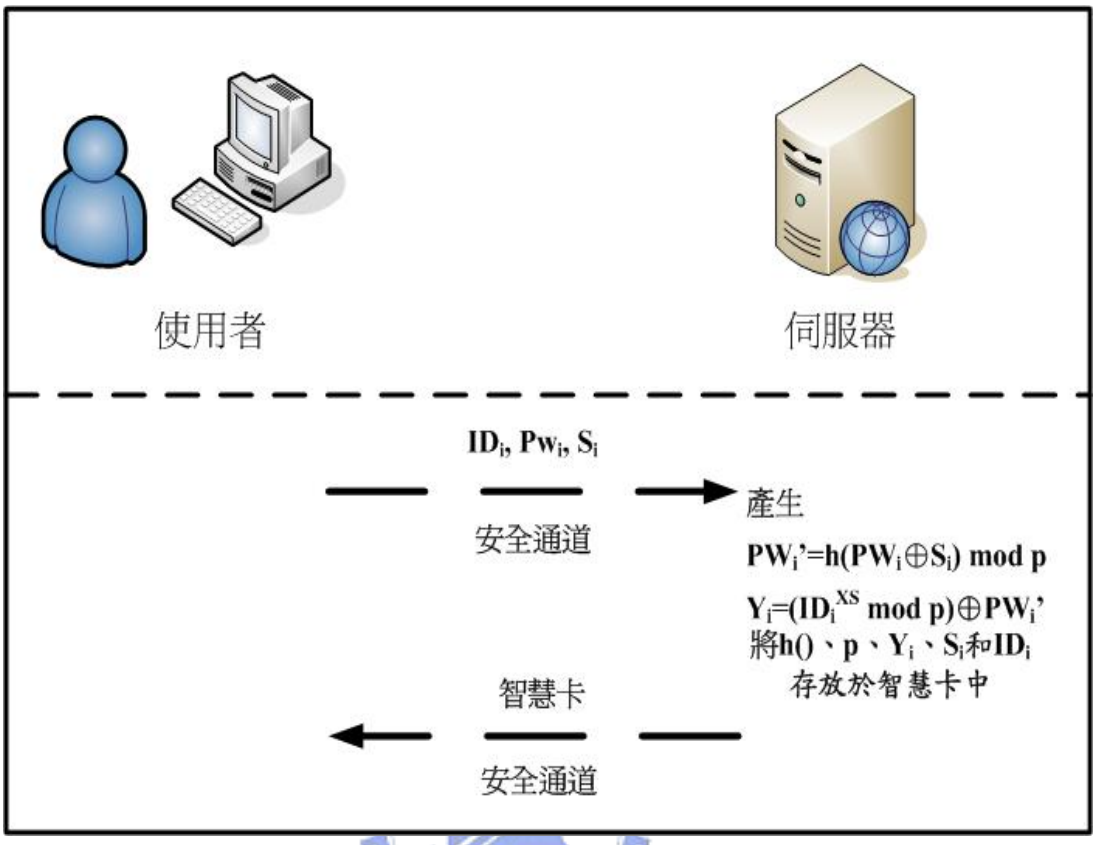


**圖 3.1：學者Lin等人所提出驗證機制的註冊期**

整個註冊期主要是在伺服器端執行，令XS是由伺服器所保管的祕密鑰匙，當使用者想要註冊及成為一個合法的使用者時，這個使用者必須先將自己的指紋按在輸入的機器上，並且將他/她的指紋($S_i$)、帳號($ID_i$)以及密碼($PW_i$)以安全通道的方式送到伺服器，整個註冊期執行步驟如下：

Step1: U -> S : $S_i$, $ID_i$、$PW_i$、$S_i$

使用者將$ID_i$和$PW_i$、$S_i$以安全通道的方式送到伺服器。

Step2: 伺服器使用$PW_i$、$S_i$、$ID_i$、XS和p去計算$PW_i'=h(PW_i \oplus S_i) \bmod p$以及$Y_i=(ID_i^{XS} \bmod p) \oplus PW_i'$

Step3: 伺服器將存放有h()、p、$Y_i$、$S_i$和$ID_i$的智慧卡交給使用者。

### 3.2.1.2 登入期(Login phase)

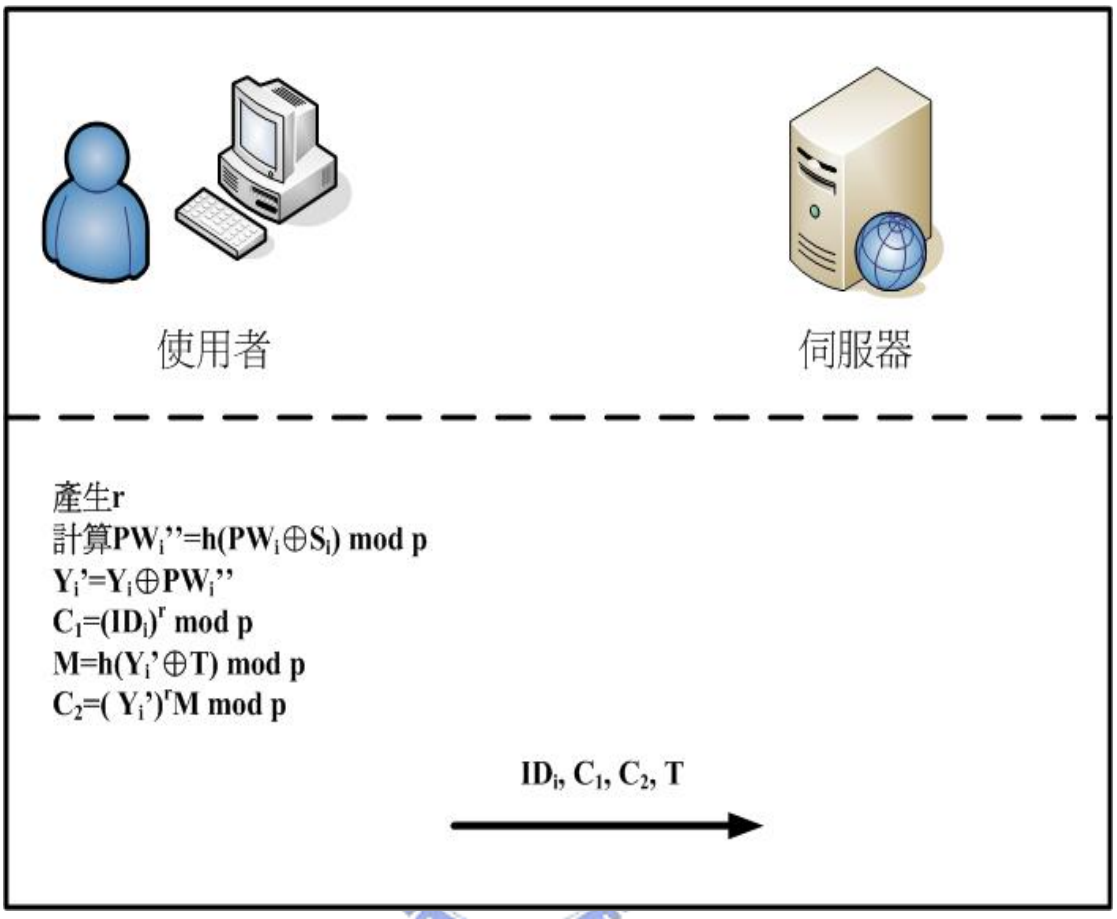


**圖 3.2：學者Lin等人所提出驗證機制的登入期**

當一個合法的使用者想要進入系統時，他必須先將智慧卡插入讀卡機中，然後在指紋讀取機器上按下他的指紋，並且輸入他個人的密碼，整個登入期的執行步驟如下所示：

Step1: 使用者從指模樣板(fingerprint template)中的使用指紋特徵點(minutiae)產生一個隨機亂數 r 。

Step2: 使用者使用$PW_i$、$S_i$、$Y_i$和p去計算$PW_i''=h(PW_i \oplus S_i) \bmod p$以及$Y_i'=Y_i \oplus PW_i''$。

Step3: 使用者使用$ID_i$、r、$Y_i'$、T和p去計算$C_1=(ID_i)^r \bmod p$,以及$M=h(Y_i' \oplus T) \bmod p$，T代表目前登入設備的時間。

Step4: 使用者使用$Y_i'$、M和p去計算$C_2=(Y_i')^r M \bmod p$。

Step5: U -> S : $ID_i, C_1, C_2, T$

使用者將$ID_i$、$C_1$、$C_2$和T送給伺服器。

### 3.2.1.3 驗證期(Authentication Phase)

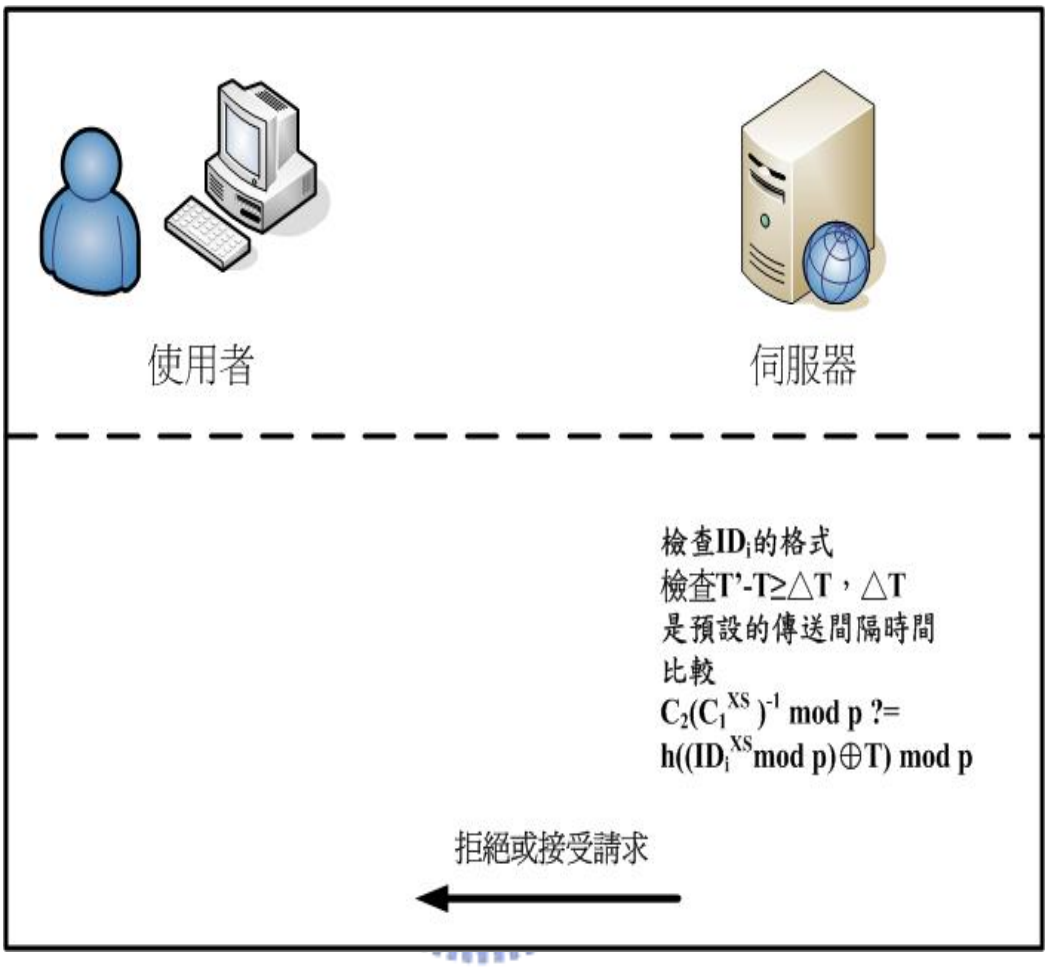


**圖 3.3：學者Lin等人所提出驗證機制的驗證期**

當伺服器接收了使用者登入的訊息之後，伺服器將會先存放收到訊息的時間T'，之後，伺服器會繼續下面的步驟：

Step1: 伺服器檢查$ID_i$的格式是否正確與否，如果格式不正確的話，系統便會拒絕使用者的登入要求。如果正確的話，便進行下面的步驟。

Step2: 伺服器會計算 T'-T，假如 T'-T≥△T 的話，伺服器拒絕使用者的登入要求，否則的話，伺服器接受使用者的登入要求並且繼續下面的步驟，△T 是預設的傳送間隔時間。

Step3: 伺服器比較$C_2(C_1^{XS})^{-1}$ mod p ?= $h((ID_i^{XS} \bmod p) \oplus T) \bmod p$ 是否一樣，如果他們相同，伺服器接受登入要求，否則，伺服器便拒絕了使用者的登入請求。

### 3.2.1.4 密碼改變期(Password Change Phase)

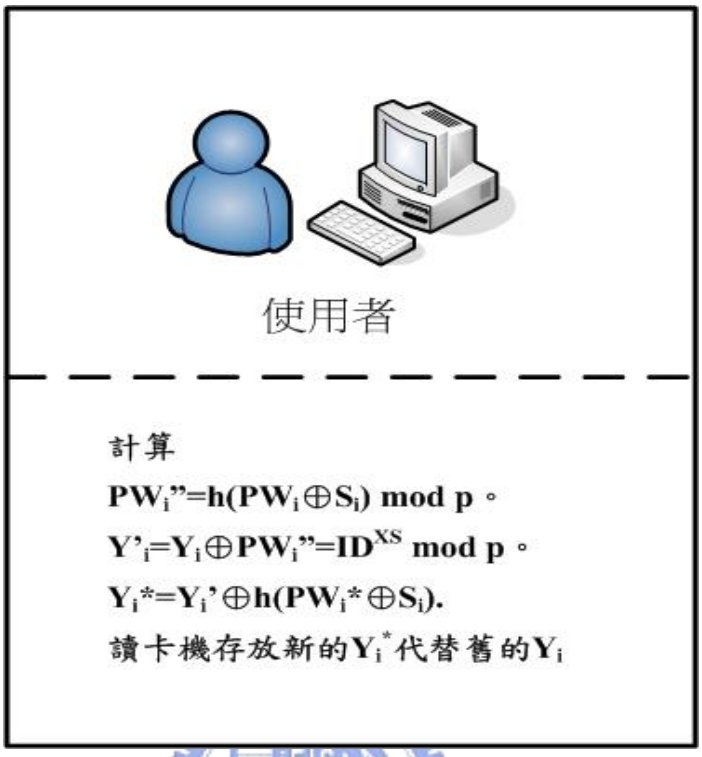


**圖 3.4：學者Lin等人所提出驗證機制的密碼改變期**

當使用者想要將他的舊密碼$PW_i$改成新密碼$PW_i^*$時，這個使用者必須要插入他智慧卡進入到讀卡機中。然後使用者必須要將他的指紋輸入到機器中，這時，讀卡機會比對使用者的指紋，如果使用者通過了指紋的比對，使用者便需要輸入舊密碼$PW_i$以及新密碼$PW_i^*$，讀卡機會進行下面的步驟：

Step1: 讀卡機使用$PW_i$、$S_i$和p去計算$PW_i'' = h(PW_i \oplus S_i) \bmod p$。

Step2: 讀卡機使用$PW_i''$、$Y_i$、p去計算$Y_i' = Y_i \oplus PW_i'' = ID^{XS} \bmod p$。

Step3: 讀卡機使用$Y_i'$、$PW_i^*$、$S_i$ 去計算$Y_i^* = Y_i' \oplus h(PW_i^* \oplus S_i)$。

Step4: 讀卡機存放新的$Y_i^*$取代舊的$Y_i$.

## 3.2.2 學者 Khan 等人所提出的驗證機制

西元 2007，學者 Khan 和學者 Zhang[15]指出學者 Lin 等人所提出的驗證機制只有提供伺服器去驗證使用者，而未提供使用者驗證伺服器，容易遭受攻擊者以伺服器偽裝攻擊(Server Spoofing Attack)來攻擊整個驗証機制，為了要改善這個缺失，他們也提出了一個改進的驗證機制，整個驗證機制可以簡單的分為註冊期(Registration Phase)、登入期(Login Phase)、驗證期(Authentication Phase)、密碼改變期(Password Change Phase)，下面將會介紹學者 Khan 所提出的驗證機制。

### 3.2.2.1 註冊期(Registration Phase)

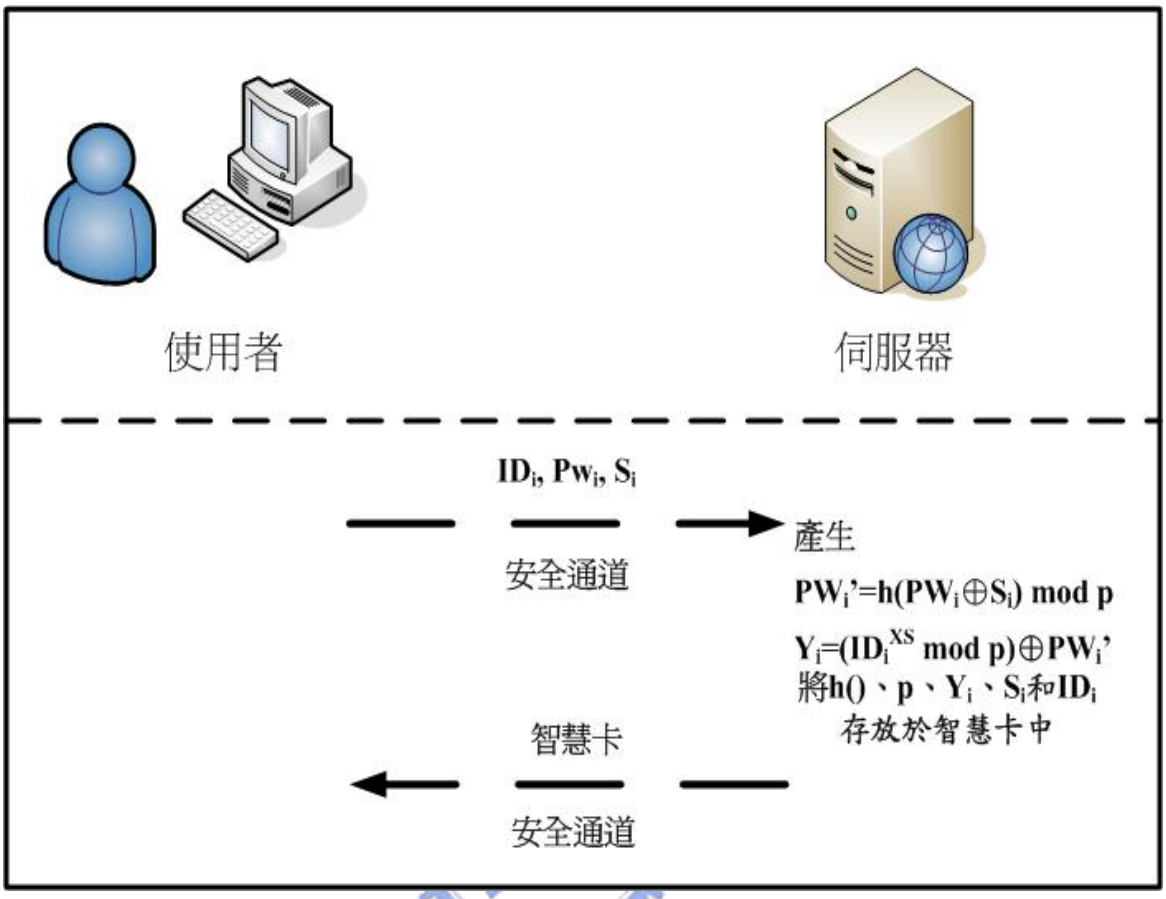


**圖 3.5：學者Khan等人所提出驗證機制的註冊期**

整個註冊期主要是在伺服器端執行，令XS是由伺服器所保管的祕密鑰匙，當使用者想要註冊及成為一個合法的使用者時，這個使用者必須先將自己的指紋按在輸入的機器上，並且將個人的指紋($S_i$)、帳號($ID_i$)以及密碼($PW_i$)以安全通道的方式送到伺服器，整個註冊期執行步驟如下：

Step1: U -> S : $S_i$, $ID_i$, $PW_i$

使用者將$ID_i$、$PW_i$、$S_i$傳送到伺服器。

Step2: 伺服器使用$PW_i$、$S_i$、$ID_i$、XS、p去計算$PW_i'=h(PW_i \oplus S_i) \bmod p$和$Y_i=(ID_i^{XS} \bmod p) \oplus PW_i'$。

Step3: 伺服器將存放有h()、p、$Y_i$、$S_i$、$ID_i$的智慧卡交給使用者。

### 3.2.2.2 登入期(Login phase)

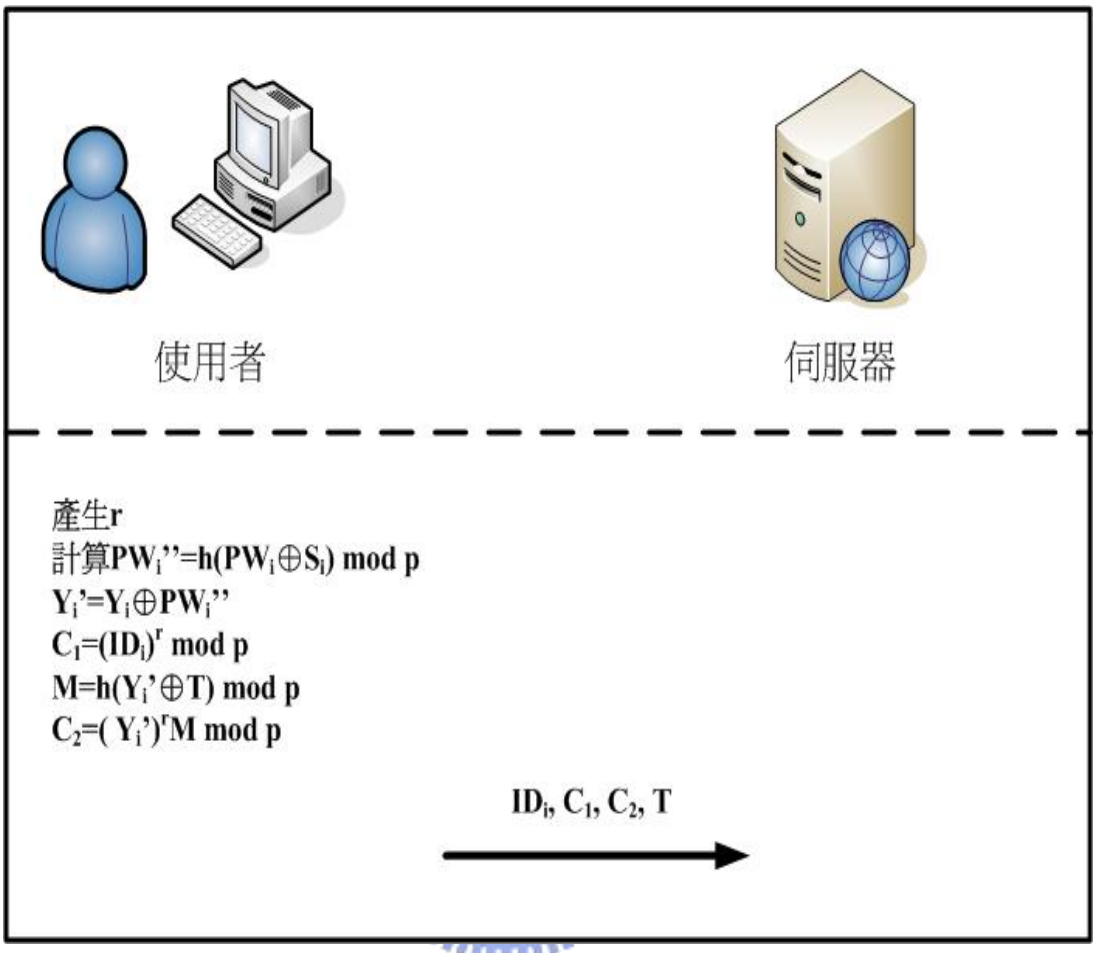


**圖 3.6：學者Khan等人所提出驗證機制的登入期**

當一個合法的使用者想要進入系統時，他必須先將智慧卡插入讀卡機中，然後在機器上按下他的指紋，並且輸入他個人的密碼，整個登入期的執行步驟如下所示：

Step1: 使用者從指模樣板(fingerprint template)中的使用指紋特徵點(minutiae)產生一個隨機亂數 r 。

Step2: 使用者使用$PW_i$、$S_i$和$Y_i$去計算$PW_i'' = h(PW_i \oplus S_i) \bmod p$以及$Y_i'=Y_i \oplus PW_i''=ID^{XS} \bmod p$.

Step3: 使用者使用$ID_i$、r、T和p去計算$C_1=(ID_i)^r \bmod p$和$M=h(Y' \oplus T) \bmod p$，T是目前使用者的時間。

Step4: 使用者使用$Y_i'$、r、M和p去計算$C_2=(Y_i')^r M \bmod p$.

Step5: U -> S : $ID_i, C_1, C_2, T$

使用者將$ID_i$、$C_1$、$C_2$、T傳送給伺服器。

### 3.2.2.3 驗證期(Authentication Phase)

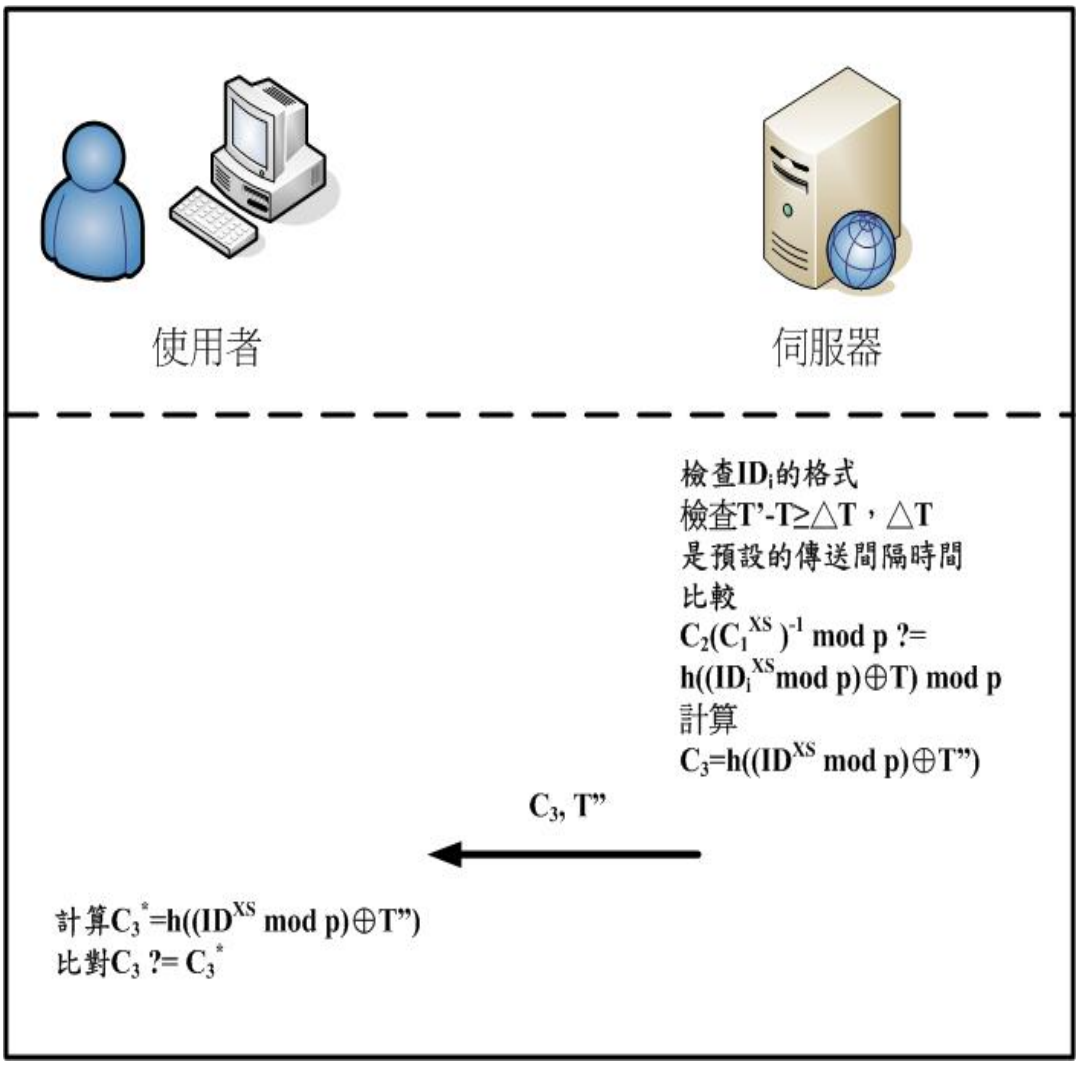


**圖 3.7：學者Khan等人所提出驗證機制的驗證期**

當伺服器收到使用者的登入要求之後，伺服器首先會計算出它收到訊息的時間T'，再來執行以下步驟：

Step1: 伺服器檢查$ID_i$的格式是否正確與否，如果格式不正確的話，系統便會拒絕使用者的登入要求。如果正確的話，便進行下面的步驟。

Step2: 伺服器計算 T'-T，假如 T'-T≥△T，伺服器拒絕使用者的登入請求，否則，伺服器接受了使用者的登入請求並且繼續下面的步驟，△T 是系統預設的時間間隔。

Step3: 伺服器比對$C_2(C_1^{XS})^{-1} \bmod p$ ?= $h((ID_i^{XS} \bmod p) \oplus T) \bmod p$ 是否相同，如果他們相同，伺服器接受了使用者的登入請求並且繼續了下面的執行步驟，否則，伺服器拒絕了使用者的登入請求。

Step4: S->U: $C_3$, T”

伺服器計算了目前的時間T”並且計算了$C_3$=h((ID$^{XS}$ mod p)⊕T”)，並且將$C_3$, T”傳送給使用者。

Step5: 當使用者接收了伺服器傳送過來的訊息，使用者計算 $C_3^*$=h((ID$^{XS}$ mod p)⊕T”)並且比對$C_3$ ?= $C_3^*$是否相同，如果他們相同的話，使用者相信伺服器是正確無誤並且接受了他的連線，否則，使用者中斷了這個連線。

## 3.2.2.4 密碼改變期(Password Change Phase)

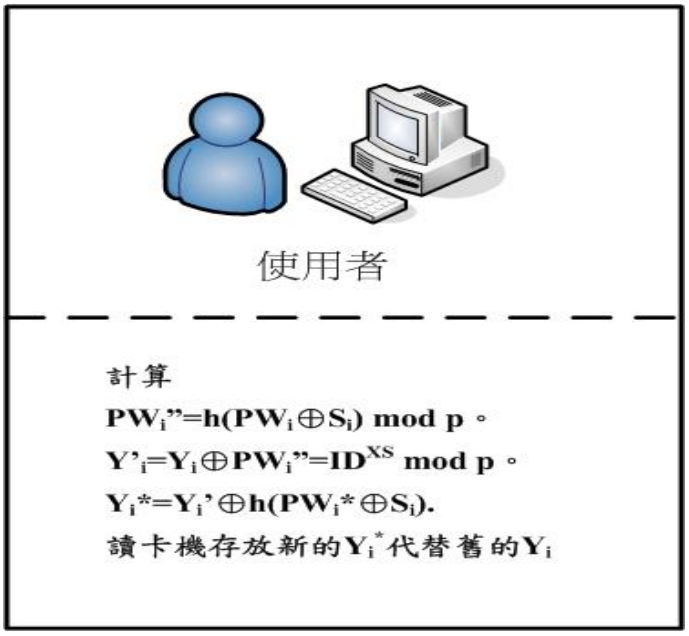


**圖 3.8：學者Khan等人所提出驗證機制的密碼改變期**

當使用者想要將他的舊密碼$PW_i$改成新密碼$PW_i^*$時，這個使用者必須先要插入個人的智慧卡進入到讀卡機中。然後使用者必須要將個人的指紋輸入到機器中，這時，讀卡機會比對使用者的指紋，如果使用者通過了指紋的比對，使用者便需要輸入舊密碼$PW_i$以及新密碼$PW_i^*$，讀卡機會進行下面的步驟：

Step1: 讀卡機使用$PW_i$、$S_i$、p去計算$PW_i$”=h($PW_i$⊕$S_i$) mod p。

Step2: 讀卡機使用$Y_i$、$PW_i$’’去計算$Y’_i$=$Y_i$⊕$PW_i$”=ID$^{XS}$ mod p。

Step3: 讀卡機使用$Y_i$’、$PW_i$、$S_i$去計算$Y_i$*=$Y_i$’⊕h($PW_i$*⊕$S_i$).

Step4: 讀卡機存放新的$Y_i^*$代替舊的$Y_i$。

### 3.2.3 本研究所提出的攻擊方法

西元 2007 年學者Khan等人[15]發現西元 2004 年學者Lin等人[4]的驗證機制只提供伺服器如何驗證使用者的身分，而未提供使用者如何驗證伺服器的身分，所以有著伺服器假冒攻擊(Server Spoofing Attack)的安全性問題，為了改進這樣的安全性問題，學者Khan等人提出了一個改善的驗證機制。然而，本研究發現到這兩個驗證機制都有著使用者偽裝攻擊(Impersonation Attack)的安全性問題，本研究提出的攻擊方法與西元 2000 年學者Chan和學者Cheng[5]以及西元 2003 年學者Chang與學者Hwang[3]所提出的使用者偽裝攻擊(Impersonation Attack)雷同，這種使用者偽裝攻擊方法最早是由學者Chan與學者Cheng[5]於西元 2000 年所提出，不過，學者Chan和學者Cheng所提出的攻擊方法卻不一定保證會成功，其原因主要是因為該驗證機制在驗證期時伺服器會去檢查$ID_i$的格式是否正確與否，因此，攻擊者所產生的使用者驗證資訊並不一定能夠通過伺服器的檢驗，所以，西元 2003 年學者Chang和學者Hwang[3]擴充了這個攻擊方法，讓攻擊者可以去產生多組的使用者驗證資訊，如此一來，大大提升攻擊者通過伺服器驗證的機會，讓攻擊者可以成功的入侵系統，下面開始介紹本研究所提出攻擊方法：

**攻擊方法一：**

假設一個攻擊者想要偽裝成其他的合法使用者，他首先必須要在這個系統上註冊成一個合法的使用者。如果一個攻擊者成功的在該系統註冊成一個合法使用者時，這個攻擊者便可以得到系統所提供的智慧卡，此時，他便可以執行下面的步驟來製造出其他合法的使用者$(ID_A, Y_A')$去入侵系統($Y_A'$必須滿足$Y_A'=ID_A^{XS} \bmod p$)：

Step1: 首先，攻擊者者必須要先將自己的智慧卡插入讀卡機，並且將自身的帳號($ID_i$)、密碼($PW_i$)和指紋($S_i$)輸入到系統之中，以計算$PW''=h(PW_i \oplus S_i) \bmod p$.

Step2: 攻擊者使用$PW_i''$、$Y_i$去計算$Y_i'=Y_i \oplus PW''=(ID_i^{XS} \bmod p)$。

Step3: 攻擊者計算

$$ID_A = ID_i^2 \bmod p$$
$$Y_A' = ((ID_i^{XS} \bmod p)* (ID_i^{XS} \bmod p) \bmod p)$$
$$= (ID_i^{2XS} \bmod p)$$
$$= ((ID_i^2)^{XS} \bmod p)$$

Step4: 這時，這個攻擊者便可以使用$ID_A$和$Y_A'$來偽裝成使用者$ID_A$入侵遠端伺服器。

因為該驗證機制在驗證期時伺服器會去檢查$ID_i$的格式是否正確與否，因此，攻擊者利用攻擊方法一所產生的使用者驗證資訊並不一定能夠通過伺服器的檢驗，但是如果擴充了攻擊方法一，來產生出多組的使用者驗證資訊，便可以增加入侵成功的機率。

## 攻擊方法二：

延伸攻擊方法一的概念，可以產生出多組使用者驗證資訊，讓攻擊者可以利用這些偽造的資訊入侵系統，這時攻擊者要將攻擊方法一的步驟三之

$$ID_A = ID_i^2 \bmod p$$

修正為

$$ID_B = ID_i^n \bmod p$$，n為與p互質的正整數。

將攻擊方法一的步驟三之

$$Y_A' = (Y_i' * Y_i') \bmod p$$
$$= (ID_i^2)^{XS} \bmod p$$

修正為

$$Y_B' = (Y_i')^n \bmod p$$
$$= ((ID_i^n)^{XS} \bmod p)$$，n為與p互質的正整數

如此，攻擊者便可以產生多組假造的驗證資訊$(ID_B, Y_B')$去入侵遠端的系統。

## 攻擊方法三：

假使一個攻擊者擁有兩張智慧卡並且知道這兩張智慧卡的$(PW_1, PW_2)$和$(Si_1, Si_2)$，這個攻擊者可以以組合的方式來計算出其他合法的使用者驗證資料，進而偽裝成其他的使用者入侵系統，整個入侵方法如下所示：

Step1: 攻擊者使用$PW_1$, $PW_2$, $Si_1$, $Si_2$去計算出$Y_1' = (ID_1^{XS} \bmod p)$ and $Y_2' = (ID_2^{XS} \bmod p)$。

Step2: 攻擊者計算

$$ID_A = ID_1 * ID_2 \bmod p$$
$$Y_A' = (Y_1' * Y_2') \bmod p$$
$$= (ID_1^{XS} \bmod p) * (ID_2^{XS} \bmod p) \bmod p$$
$$= (ID_1^{XS} * ID_2^{XS}) \bmod p$$
$$= (ID_1 * ID_2)^{XS} \bmod p$$

Step3: 攻擊者可以使用$ID_A$和$Y_A'$去假冒另外一個使用者$ID_A$登入系統。

同樣的，因為該驗證機制在驗證期時伺服器會去檢查$ID_i$的格式是否正確與否，因此，攻擊者利用攻擊方法三所產生的使用者驗證資訊並不一定能夠通過伺服器的檢驗，但是如果擴充了攻擊方法三，來產生出多組的使用者驗證資訊，便可以增加入侵成功的機率。

**攻擊方法四：**

延伸攻擊方法三的概念，假如一個攻擊者獲得了n個智慧卡，並且得知這n個使用者的密碼以及指紋，這個攻擊者便可以以組合的方式，去產生多組使用者$ID_B$，以入侵到遠端的系統，這時攻擊者要將攻擊方法三的步驟二之

$$ID_A = ID_1 * ID_2 \bmod p$$

修正為

$$ID_B = \prod ID_i \bmod p$$，n為與p互質的正整數

將

$$Y_A' = (Y_1' * Y_2') \bmod p$$

修正為

$$Y_B' = (\prod Y_j^{XS}) \bmod p$$

$$= (\prod (ID_{Aj}^{xs} \bmod p)) \bmod p$$，n 為與 p 互質的正整數

這樣攻擊者便可以產生多組使用者$ID_B$入侵到遠端的系統之中。

## 3.3 本研究所提出的改善驗證機制

這個部份將開始介紹本研究所提出的改進驗證機制，這個改善驗證機制主要是以西元 2007 年學者 Khan 等人[15]所提出的驗證機制為基礎，整個改進驗證機制流程一樣分成了註冊期(Registration Phase)、登入期(Login Phase)、驗證期(Authentication Phase)和密碼改變期(Password Change Phase)等四個部分，介紹如下：

### 3.3.1 註冊期(Registration Phase)

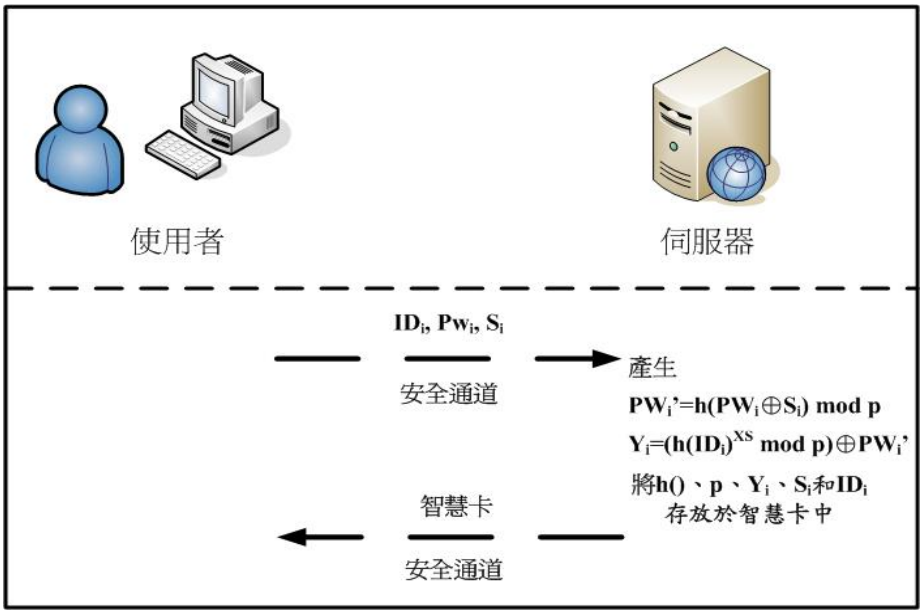


**圖 3.9：本研究所提出改進驗證機制的註冊期**

整個註冊期主要是在伺服器端執行，令XS是由伺服器所保管的祕密鑰匙，當使用者想要註冊及成為一個合法的使用者時，這個使用者必須先將自己的指紋按在輸入的機器上，並且將他/她的指紋($S_i$)、帳號($ID_i$)以及密碼($PW_i$)以安全通道的方式送到伺服器，讓伺服器去進行計算工作，最後將計算的結果存放於智慧卡中，再以安全通道的方式將這張智慧卡轉交到使用者手中，整個註冊期執行步驟如下：

Step1: U -> S : $S_i$, $ID_i$, $PW_i$

使用者將$S_i$、$ID_i$、$PW_i$以安全通道的方式傳送給伺服器。

Step2: 伺服器使用$S_i$、$ID_i$、$PW_i$、p和XS去計算$PW_i' = h(PW_i \oplus S_i) \bmod p$和$Y_i = (h(ID_i)^{XS} \bmod p) \oplus PW_i'$

Step3: 伺服器將存有h()、p、$Y_i$、$S_i$、$ID_i$的智慧卡交給使用者。

## 3.3.2 登入期(Login phase)

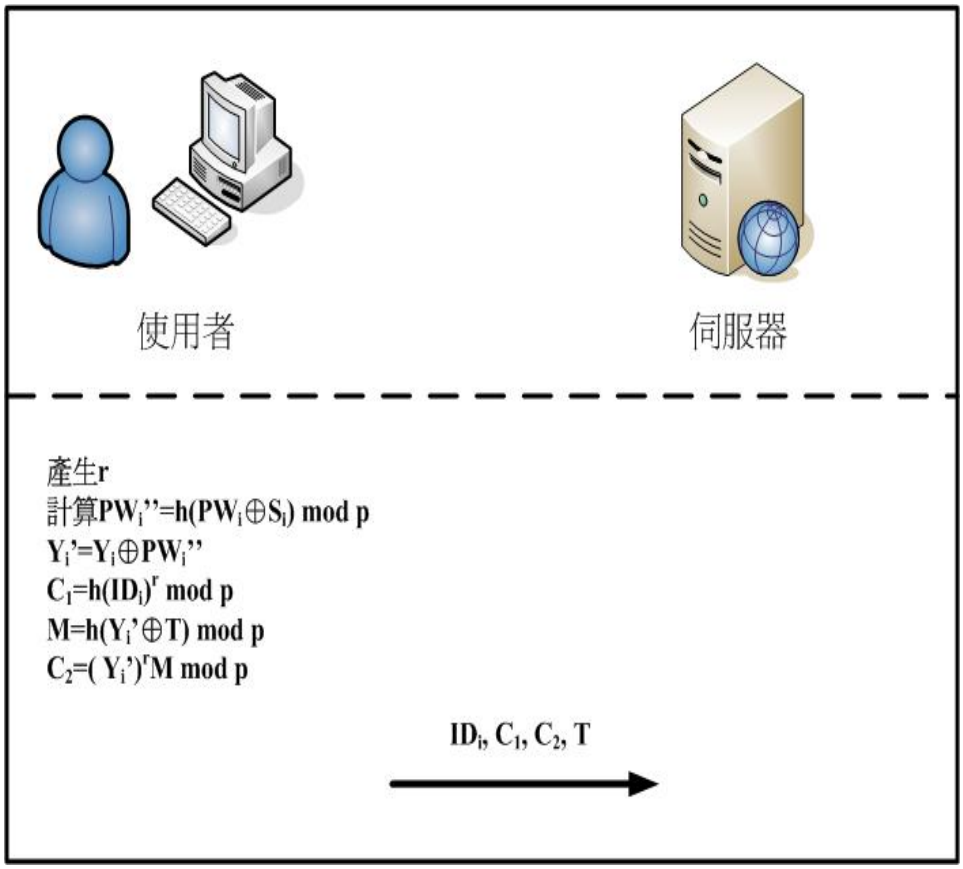


**圖 3.10：本研究所提出驗證機制的登入期**

當一個合法的使用者想要進入系統時，他必須先將他的智慧卡插入讀卡機中，在指紋機器上按下他的指紋，並且輸入他個人的密碼，然後進行登入的程序，整個登入期的執行步驟如下所示：

Step1: 使用者從指模樣板(fingerprint template)中的使用指紋特徵點(minutiae)產生一個隨機亂數 r 。

Step2: 使用者使用 $PW_i$、$S_i$、p 和 $Y_i$ 去計算 $PW_i''=h(PW_i \oplus S_i) \bmod p$ 和 $Y_i'=Y_i \oplus PW_i''=h(ID_i)^{XS} \bmod p$。

Step3: 使用者使用 $h(ID_i)$、r、p、Y' 和 $T_1$ 去計算 $C_1=h(ID_i)^r \bmod p$ 和 $M=h(Y_i' \oplus T_1) \bmod p$，$T_1$ 是目前的時間。

Step4: 使用者使用 $Y_i'$、M、r、p 去計算 $C_2=(Y_i')^r M \bmod p$.

Step5: U -> S : $ID_i$, $C_1$, $C_2$, $T_1$

使用者將 $ID_i$、$C_1$、$C_2$ 和 $T_1$ 傳送到伺服器。

## 3.3.3 驗證期(Authentication Phase)

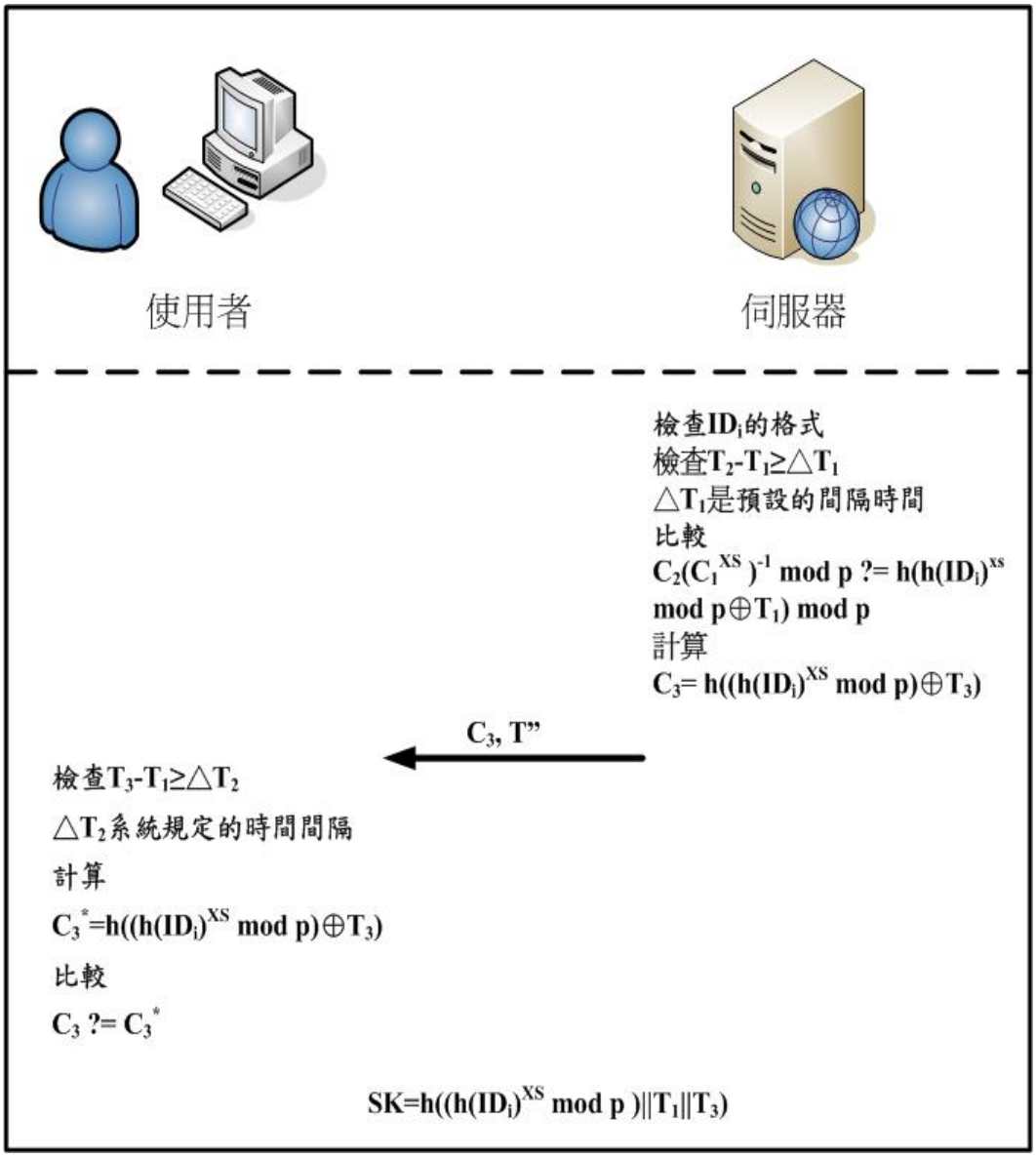


**圖 3.11：本研究所提出驗證機制的驗證期**

當伺服器收到使用者傳來的連接訊息之後，伺服器會先計算出目前的時間$T_2$。此時，會進行下面的步驟：

Step1: 伺服器檢查$ID_i$的格式是否正確與否，如果格式不正確的話，系統便會拒絕使用者的登入要求。如果正確的話，便進行下面的步驟。

Step2: 伺服器計算$T_2-T_1$，假如$T_2-T_1 \geq \triangle T_1$，伺服器拒絕了使用者的登入要求，否則，伺服器接受了使用者的登入請求，$\triangle T_1$是系統預設的時間間隔。

Step3: 伺服器使用$ID_i$、XS、p去計算$h(ID_i)^{xs} \bmod p$，然後比對$C_2(C_1^{XS})^{-1} \bmod p \stackrel{?}{=} h(h(ID_i)^{xs} \bmod p \oplus T_1) \bmod p$ 是否相同，如果他們相同，伺服器接受使用者的登入要求，否則，伺服器拒絕了使用者的登入要求。

Step4: S->U: $C_3$, $T_3$

伺服器首先取出目前的時間$T_3$並且使用$ID_i$、XS、p和$T_3$去計算$C_3 = h((h(ID_i)^{XS} \bmod p) \oplus T_3)$。之後，伺服器將$C_3$、$T_3$傳送給使用者。

Step5: 當使用者收到伺服器傳來的訊息時，使用者計算$T_3-T_1$。假如$T_3-T_1 \geq \triangle T_2$，使用者會拒絕這個連線，否則，使用者會相信伺服器正確無誤並且接受這個連線，$\triangle T_2$是使用者預設的時間間隔。

Step6: 使用者使用 $h(ID_i)^{XS} \bmod p$、$T_3$去計算$C_3^* = h((h(ID_i)^{XS} \bmod p) \oplus T_3)$，並且比較$C_3 \stackrel{?}{=} C_3^*$。假如他們是一樣的，使用者會相信伺服器正確無誤並且接受這個連線，否則，使用者中斷這個連線。

Step7: 在驗證完伺服器與使用者的身分之後，伺服器和使用者都會產生一把階段鑰匙(session key) $SK = h((h(ID_i)^{XS} \bmod p) \| T_1 \| T_3)$。

## 3.3.4 密碼變更期(Password Change Phase)

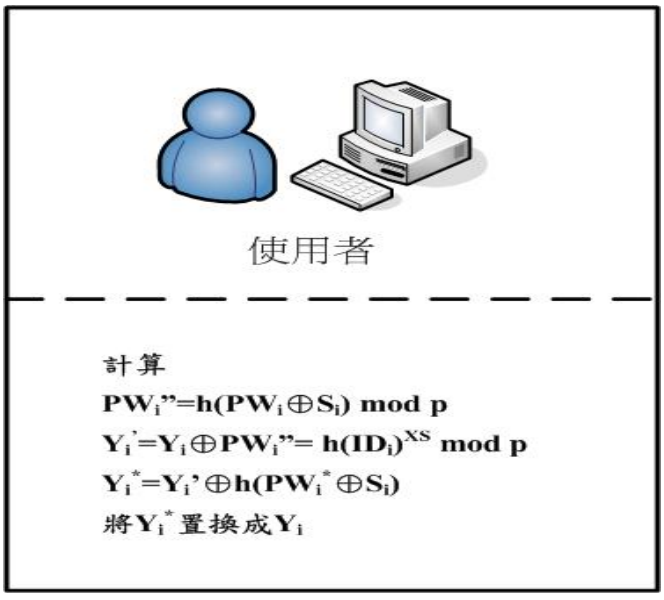


**圖 3.12：本研究所提出改進驗證機制的密碼改變期**

當使用者想要將他的舊密碼$PW_i$改成新密碼$PW_i^*$時，這個使用者必須要將個人的智慧卡插入到讀卡機中。然後在指紋機器上按下他的指紋，這時，讀卡機會比對使用者的指紋，如果使用者通過了指紋的比對後，使用者便需要輸入舊密碼$PW_i$以及新密碼$PW_i^*$，此時，讀卡機會進行下面的步驟：

Step1:讀卡機使用$PW_i$、$S_i$和p去計算$PW_i$"=$h(PW_i \oplus S_i)$ mod p。

Step2:讀卡機使用PW"、$Y_i$去計算出$Y_i$'=$Y_i \oplus PW_i$"= $h(ID_i)^{XS}$ mod p。

Step3:讀卡機使用$Y_i$'、$h(PW_i^* \oplus S_i)$去計算$Y_i^*$=$Y_i$'$\oplus h(PW_i^* \oplus S_i)$。

Step4: 讀卡機將$Y_i^*$置換成$Y_i$。

## 3.4 安全性分析

在介紹完本研究提出的改善驗證機制之後，下面將會透過各種已知的攻擊方式來證明本研究提出的改善驗證機制是安全的。

### 3.4.1 重送攻擊(Reply Attack)

在本研究所提出驗證機制中，因為使用了時間戳記(timestamp)去防止攻擊者使利用重送攻擊的方式去攻擊系統，因此攻擊者無法利用重送攻擊的方式來將之前驗證的資訊重新送到伺服器，來偽裝成其他使用者登入系統，所以我們的驗證機制可以有效地避免重送攻擊(Repky Attack)。

### 3.4.2 密碼表被竊攻擊(Stolen-verifier Attack)

因為本研究所提出的改善驗證機制並不需要在伺服器上存放任何的使用者驗證表，因此攻擊者無法利用安全漏洞或是透過內部人員竊取等方式從伺服器中偷取出使用者驗證表，進而竊取出有用的使用者驗證資訊，所以本研究所提出的改善驗證機制可以有效地避免驗證表被竊攻擊(Stolen-verifier Attack)。

### 3.4.3 伺服器假冒攻擊(Server Spoofing Attack)

在本研究所提出的驗證機制中，很明顯的提供了相互驗證(Mutual Authentication)的驗證機制，因此，假設一個攻擊者想要偽裝成一台合法的伺服器去欺騙使用者，這個攻擊者就必須要擁有秘密鑰匙 XS 才能夠跟使用者進行雙向驗證(Mututal Authentication)，進而騙取使用者登入系統之中。假設使用者發現到伺服器是攻擊者所假冒的，使用者將會中斷與該伺服器之間的連線，因此，本研究所提出的驗證機制可以有效的避免伺服器偽裝攻擊(Server Spoofing Attack)。

### 3.4.4 使用者假冒攻擊(Impersonation Attack)

在本研究所提出的改善驗證機制中，一個攻擊者無法假冒成一個合法的使用者入侵系統。如果攻擊者想要成功的假冒成一個合法的使用者，這個攻擊者就必須先要知道伺服器所擁有的祕密鑰匙XS，才能計算出使用者所擁有的$h(ID_i)^{XS} \bmod p$，況且，本研究利用了單向雜湊函數的特性，讓攻擊者無法在已知$h(ID_i)^{XS} \bmod p$的情況下去計算出其他的合法使用者所擁有的$h(ID_j)^{XS} \bmod p$，這樣可以有效避免攻擊者利用使用者假冒攻擊去攻擊系統，所以我們提出的改善驗證機制可以有效的避免使用者偽裝攻擊(Impersonation Attack)。

### 3.4.5 階段鑰匙的安全性(Security of Session Key)

本研究所提出的改善驗證機制之階段鑰匙(session key) $SK=h((h(ID_i)^{XS} \bmod p)||T1||T3)$主要是由$h(ID_i)^{XS} \bmod p$、T1、T3 所產生的，T1、T3 是由伺服器與使用者各自產生的，不但每次通訊時它們的值都不一樣，而且每當使用者與伺服器通訊結束時，這把階段鑰匙(Session key)將會被丟棄，並不會使用在下次的通訊上，所以每當使用者重新進入系統時，一把新的階段鑰匙(Session key)將會自動產生去加解密雙方通訊的資料。因此，假設一個攻擊者得到了一把階段鑰匙(session key)，他將無法利用這一把階段鑰匙(session key)去計算出其他次通訊的資料，而且因為單向雜湊函數的特性，假設一個攻擊者知道了一把階段鑰匙(Session Key)的值，攻擊者也無法從中去推測出其他連結用的階段鑰匙(session key)的值，所以本研究的階段性鑰匙(Session Key)是十分的安全。

## 3.5 與其他兩個驗證機制比較

在這小節，將會將本研究所提出的改善驗證機制與學者 Lin 等人和學者 Khan 等人所提出的驗證機制做一個比較。本研究的改善驗證機制和其他兩個驗證機制都不需要在伺服器上存放任何的驗證表，因此並沒有驗證表被竊攻擊(Stolen Verifier Attack)的可能性。學者 Khan 等人曾經指出學者 Lin 等人有著伺服器偽造攻擊(Server Spoofing Attack)的安全性問題，因為它只提供伺服器去驗證使用者的身份是否合法，而未提供使用者去驗證伺服器的身份是否正確無誤。而本研究的驗證機制以及學者 Khan 等人所提出的驗證機制因為提供使用者去驗證伺服器的機制，因此，不會遭到攻擊者以伺服器偽裝攻擊(Server Spoofing Attack)的方式來攻擊系統。在本研究中，我們指出學者 Lin 等人和學者 Khan 等人所提出的驗證機制都會遭受到使用者偽裝攻擊(Impersonation attack)，而本研究的驗證機制因為改善了這個安全性問題，所以並不會遭到攻擊者以使用者偽裝攻擊(Impersonation Attack)來攻擊系統。此外，為了保護後續資料的安全性，本研究在驗證結束時，使用者與伺服器雙方都會產生一把階段鑰匙(Session key)，用以保護後續資料的傳送，但是學者 Lin 等人的驗證機制和學者 Khan 等人的驗證機制並沒有提供階段鑰匙(session key)，無法保護後續資料的傳輸。整個比較彙整如下表所示：

表 3.2：與其他兩個機制比較

| | 是否需要提供驗證表 | 是否會遭受使用者假冒攻擊 | 是否提供相互驗證 | 是否會遭受離線密碼猜測攻擊 | 是否有提供 Session key | 是否會遭送伺服器偽裝攻擊 |
|---|---|---|---|---|---|---|
| 本研究提出 | No | No | Yes | No | Yes | No |
| 學者 Lin 等人[4] | No | Yes | No | No | No | Yes |
| 學者 Khan 等人[15] | No | Yes | Yes | No | No | No |

## 3.6 結論與後續分析

在這個章節中，本研究指出學者 Lin 等人所提出的驗證機制以及學者 Khan 等人所提出的驗證機制都有使用者假冒攻擊(Impersonation Attack)的安全性問題，為了有效避免這個攻擊的發生，本研究提出了一個新的改善驗證機制去改善這樣的缺點，這個改善驗證機制不但安全，而且非常適合使用於需要高度安全的應用程式之中，不過可惜的是這個改善驗證機制還是以時間戳記為基礎，所有有著時間同步的問題的存在，未來工作可以朝向以隨機亂數(nonce)來設計出以隨機亂數(nonce)為基礎的改善驗證機制。

# 第四章 改善學者Chang和學者Chang所提出的驗證機制

## 4.1 前言

使用者身份驗證一直是網路系統中最基本的安全機制，對於一個安全無虞的驗證機制來說，不僅僅是伺服器要去驗證使用者身份，更重要的是必須提供使用者能夠去驗證伺服器的身分，這樣才不會被假冒的使用者或是偽裝的伺服器所欺騙，導致重要的資訊遭到攻擊者所偷竊。在所有的驗證機制中，以密碼為主的身分驗證機制是目前使用地最廣泛的一種身分驗證機制，西元 2000 年，學者 Peyavian 和學者 Zunic[16]提出一種以單向雜湊函數為基礎的使用者驗證機制，它可以在不安全的網路環境之下讓驗證資訊安全無虞地送到對方手上，並且不需要使用對稱性加密演算法或是非對稱性加密演算法去加密通訊的資料，以減少使用者或是伺服器的運算量。

西元 2002 年學者 Hwang 和學者 Yeh[10]指出學者 Peyravian 等人所提出的驗證機制有許多安全性的問題，因為學者 Peyravian 等人所提出的驗證機制是以密碼來驗證遠端的使用者，所以學者 Peyravian 等人所提出的驗證機制容易遭受攻擊者利用密碼猜測攻擊(password guessing attack)、伺服器偽裝攻擊(server spoofing attack)以及伺服器資料竊聽攻擊(server data eavesdropping attack)等攻擊方式入侵系統。為了要改善這些安全性問題，學者 Hwang 和學者 Yeh 提出了一種改善的驗證機制來避免這些安全性問題的發生，這個改善的驗證機制除了可以有效的避免以上的安全性問題外，更在驗證結束之後讓使用者與伺服器去定義出一把階段鑰匙(session key)以保障後續資料傳輸的安全性，讓系統的安全性更加完善。不幸的是，西元 2003 年學者 Lin 和學者 Hwang[6]指出學者 Hwang 和學者 Yeh 所提出的驗證機制容易遭受阻絕服務攻擊(Denial of Service Attack)以及欠缺前推安全(forward secrecy)，所以還不是十分的安全，為了改善這些安全性問題，學者 Lin 和學者 Hwang 也提出了一種改善的驗證機制去改善學者 Hwang 等人所提出驗證機制的安全性問題。可惜的是，西元 2005 年學者 Chang 和學者 Chang[24]指出這個驗證機制有著驗證表被竊攻擊(Stolen-verifier attack)的安全性問題，因為這些驗證機制還是需要有驗證表，為了有效避免驗證表被竊攻擊的發生，他們提出了兩個新的無驗證表驗證機制去改善這樣的安全性問題。

本研究發現西元 2005 年學者 Chang 和學者 Chang 所提出的兩個驗證機制中，有一個驗證機制有著使用者假冒攻擊(Impersonation Attack)的安全性問題，靠著這個使用者偽造攻擊，合法的使用者可以很輕鬆的產生多組帳號及密碼而不需要知道伺服器所隱藏的資訊，讓系統的安全性大受影響，為了有效的改善這個安全性問題，本研究也提出了一個新的改善驗證機制去有效防範攻擊者使用使用者偽裝攻擊(Impersonation Attack)去入侵系統。

## 4.2 學者 Chang 和學者 Chang 所提出的驗證機制

在這個地方，我們將會簡單介紹西元 2005 年學者 Chang 和學者 Chang[24]所提出的驗證機制，這個驗證機制主要包括了兩個部份：註冊期(Registration phase)以及登入期(Login phase)，在我們介紹這個驗證機制之前，我們首先介紹在驗證機制中所使用到的符號，如下表所示：

**表 4.1：符號說明表**

| 符號 | 說　　明 |
|---|---|
| h() | 單向雜湊函數 |
| ⊕ | 互斥或 |
| ID | 使用者帳號 |
| PW | 使用者密碼 |
| ‖ | 參數連結 |
| N | 使用者及伺服器產生的隨機亂數 |
| X->Y:M | 表示從 X 傳送一個 M 訊息到 Y |
| U，S | 分別代表使用者及伺服器 |

### 4.2.1 註冊期(Registration Phase)

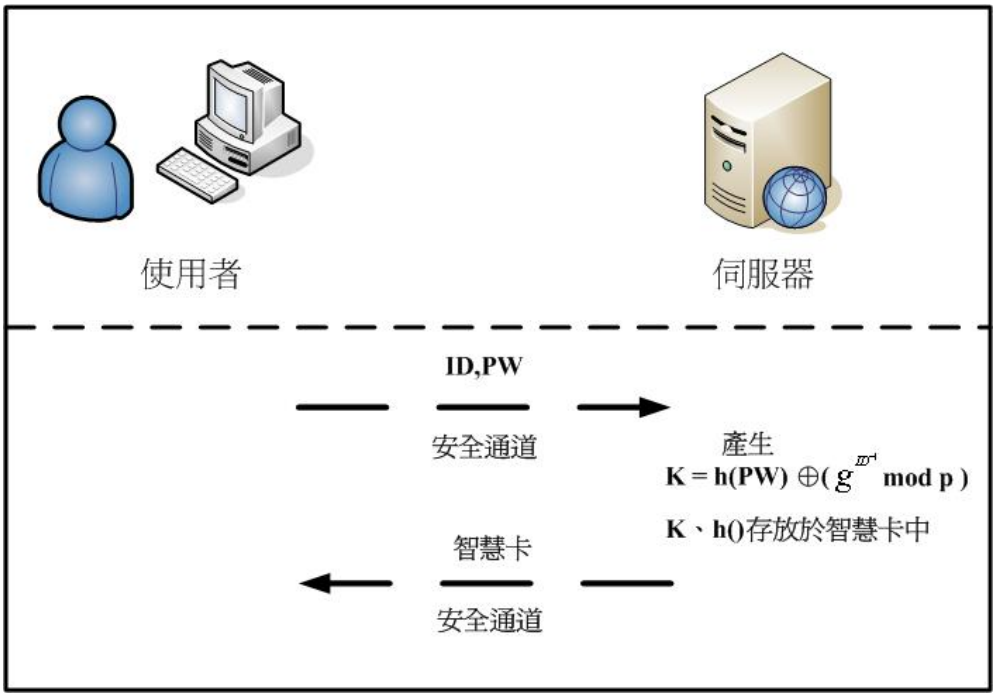


**圖 4.1：學者Chang等人所提出驗證機制的註冊期**

整個註冊期主要執行在伺服器上，令 g 是一個在 GF(p)的質數、p 是一個非常大的質數，g 和 p 這兩個系統變數都保存在伺服器之中，當一個使用者想要註冊成為一個新的使用者時，這個使用者必須將使用者帳號(ID)和使用者密碼(PW)以安全通道的方式送到伺服器，所有的註冊期執行如下所示：

Step1: 使用者自由的選擇一個使用者密碼 PW，隨後將使用者帳號(ID)和使用者密碼(PW)以安全通道的方式傳送到遠端的伺服器。

Step2: 當伺服器收到了使用者傳來的帳號(ID)和密碼(PW)時，伺服器計算 K = h(PW) $\oplus$ ( $g^{ID^{-1}} \bmod p$ )，隨後，伺服器將 K 和 h()存放在智慧卡中，並且將這個智慧卡以安全通道的方式交給使用者使用。

## 4.2.2 登入期(Login Phase)

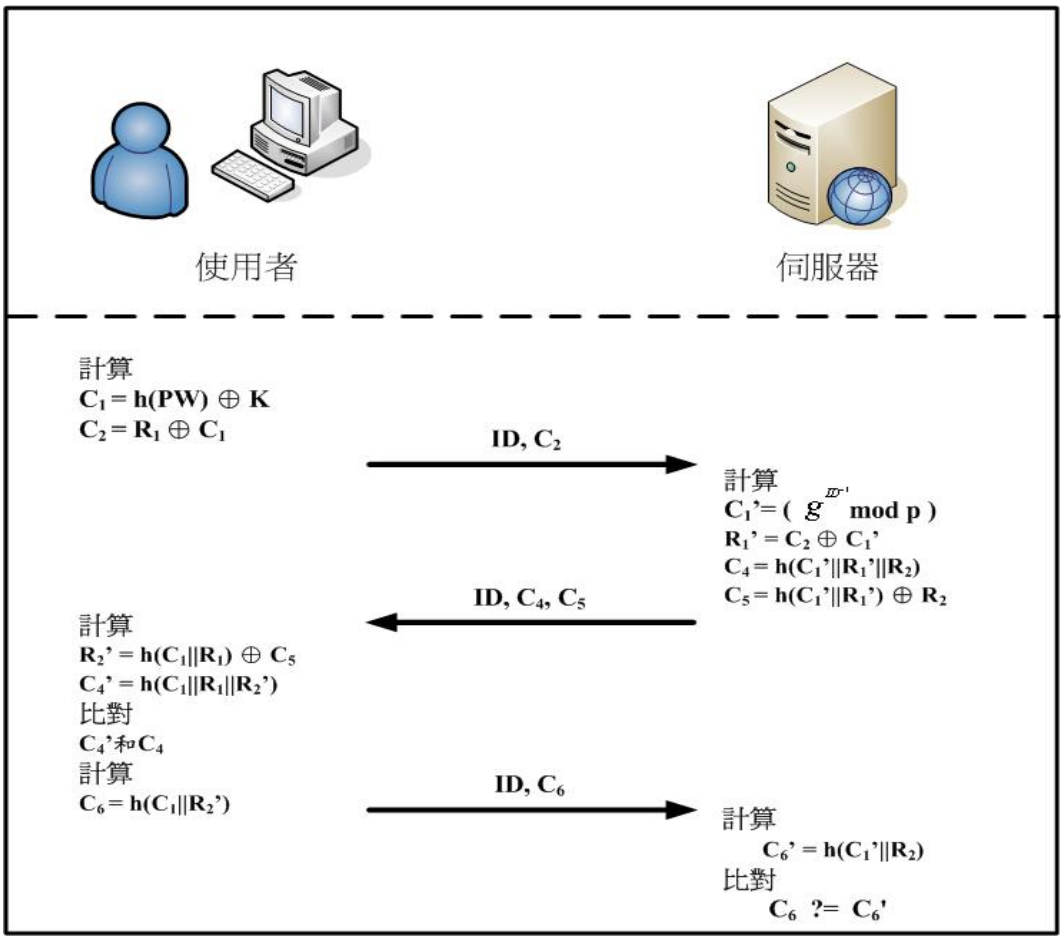


**圖 4.2：學者Chang等人所提出驗證機制的登入期**

當一個合法的使用者想要登入到遠端的系統之中，使用者必須要先插入他的或她的智慧卡於讀卡機中，並且輸入他的或她的帳號(ID)和密碼(PW)，這時便會執行下面的登入步驟：

Step1: U->S: ID, $C_2$

使用者隨機產生一個亂數$R_1$，並且使用PW、K、$R_1$去計算出$C_1 = h(PW) \oplus K = (g^{ID^{-1}} \bmod p)$以及$C_2 = R_1 \oplus C_1$，隨後，使用者將ID, $C_2$送到伺服器進行驗證。

Step2: 當伺服器從使用者端收到訊息時，伺服器首先利用ID、g、p去計算$C_1' = (g^{ID^{-1}} \bmod p)$並且使用$C_1'$、$C_2$去計算$R_1' = C_2 \oplus C_1'$。

Step3: 伺服器使用$C_1'$、$R_1'$、$R_2$去計算$C_4 = h(C_1'||R_1'||R_2)$並且使用$C_1'$, $R_1'$, $R_2$去計算$C_5 = h(C_1'||R_1') \oplus R_2$。

Step4: S->U: ID, $C_4$, $C_5$

伺服器將ID、$C_4$、$C_5$送往使用者。

Step5: U->S: ID, $C_6$

當使用者收到伺服器傳來的訊息之後，使用者使用$C_1$、$R_1$、$C_5$去計算出$R_2' = h(C_1||R_1) \oplus C_5$，並且使用$C_1$、$R_1$、$R_2'$去計算$C_4' = h(C_1||R_1||R_2')$，隨後使用者比對$C_4'$和$C_4$，假如他們一樣，使用者使用$C_1$、$R_2'$去計算出$C_6 = h(C_1||R_2')$並且將ID、$C_6$送往伺服器，否則，使用者中斷伺服器的連結請求。

Step6: 當伺服器收到使用者傳來的登入訊息之後，伺服器使用$C_1'$、$R_2$去計算$C_6' = h(C_1'||R_2)$，並且比對$C_6$和$C_6'$。如果他們相同的話，伺服器接受使用者登入的請求，否則，伺服器拒絕使用者的登入要求。

### 4.2.3 本研究所提出的攻擊方法

使用者偽裝攻擊(Impersonation Attack)是一種十分危險的網路攻擊方法，一個攻擊者可以使用這樣的攻擊方法去偽裝成其他的合法使用者入侵系統，根據西元2000年學者Chan與學者Cheng[5]和西元2003年學者Chang與學者Hwang[3]所發表的期刊論文中，曾提出一種使用者偽造攻擊，這種攻擊方法可以讓一個合法的使用者去產生另一組合法的帳號密碼，而不需要知道伺服器所保存的私密參數。利用這種攻擊方法，我們發現到學者Chang和學者Chang[24]所提出的驗證機制也可以利用這種使用者偽裝攻擊(Impersionation Attack)來產生出多組偽造的驗證資訊，進而假冒使用者登入系統之中。不過，有些許不一樣的是學者Chang與學者Chang所提出的驗證方法隱藏了變數 p，因此，如果我們也要利用同樣的方式攻擊驗證系統的話，必須要滿足一個先備條件，才能夠在不知這個變數p之下攻擊系統。在介紹本研究所提出的攻擊法之前，下面將會先介紹一下這個先備條件，這個先備條件如下所示：

$$(a \bmod p) * (b \bmod p)$$
$$if\ (a \bmod p) * (b \bmod p) < p$$
$$= (a * b \bmod p)$$

證明

$$\text{Let } a = S_1 p + t_1, b = S_2 p + t_2 \quad (S_1, S_2 \in n, 0 \le t_1 < p, 0 \le t_2 < p)$$

$$(a \bmod p) * (b \bmod p)$$
$$= [(S_1 p + t_1) \bmod p] * [(S_2 p + t_2) \bmod p]$$
$$= t_1 * t_2$$
$$if\ (a \bmod p) * (b \bmod p) = (t_1 * t_2) < p$$
$$= (t_1 * t_2) \bmod p$$
$$= (S_1 p * S_2 p + S_1 p t_2 + S_2 p t_1 + t_1 * t_2) \bmod p$$
$$= (S_1 p + t_1) * (S_2 p + t_2) \bmod p$$
$$= (a * b) \bmod p$$

介紹完這個先備條件之後，下面開始介紹本研究之攻擊方法，如下所示：

## 攻擊方法一：

如果要產生出其他的使用者驗證資訊的話，攻擊者在攻擊前必須要滿足兩個條件，第一、攻擊者必須先系統上註冊，其帳號必須要能被 2 所整除，如果註冊成功，攻擊者便可以獲得一個含有 K = h(PW) $\oplus$ ( $g^{ID^{-1}} \bmod p$ )和 h()的智慧卡，這樣攻擊者才可以使用個人的使用者密碼去獲得 $g^{ID^{-1}} \bmod p$，第二、攻擊者所擁有的 $g^{ID^{-1}} \bmod p$ 必須滿足( $g^{ID^{-1}} \bmod p$ )*( $g^{ID^{-1}} \bmod p$ )< p，這樣攻擊者才可以在不需要知道 p 值的情況下，利用 $g^{ID^{-1}} \bmod p$ 進行使用者假冒攻擊(Impersonation Attack)，靠著計算：

$ID_A = ID/2$

$$g^{ID_A^{-1}} \bmod p = (g^{ID_A^{-1}} \bmod p)$$
$$= (g^{ID^{-1}} \bmod p) * (g^{ID^{-1}} \bmod p)$$
$$\because (g^{ID^{-1}} \bmod p) * (g^{ID^{-1}} \bmod p) < p$$
$$= g^{(ID/2)^{-1}} \bmod p$$

如此一來，攻擊者便可以使用($ID_A$, $g^{ID_A^{-1}} \bmod p$ )=(ID/2, $g^{(ID/2)^{-1}} \bmod p$ )假冒成一個合法的使用者$ID_A$去登入遠端的系統。

## 攻擊方法二：

擴充攻擊方法一，假設攻擊者已經成功在系統上註冊，註冊時所使用的帳號 ID 並不為質數，所以攻擊者獲得一個含有 K = h(PW) $\oplus$ ( $g^{ID^{-1}} \bmod p$ )和 h()的智慧卡，

此時如果攻擊者所擁有的 $g^{ID^{-1}} \bmod p$ 滿足 $(g^{ID^{-1}})^n \bmod p$<p 這樣的條件時，攻擊者便可以使用下面的方法去計算出多組偽裝的使用者驗證資訊：

將攻擊方法一的

$ID_A=ID/2$

改成

$ID_A=ID/n$，(n為一個可以整除ID的正整數)

並且將

$$g^{ID_A^{-1}} \bmod p = (g^{ID_A^{-1}} \bmod p)$$
$$= (g^{ID^{-1}} \bmod p)*(g^{ID^{-1}} \bmod p)$$
$$\because (g^{ID^{-1}} \bmod p)*(g^{ID^{-1}} \bmod p) < p$$
$$= (g^{ID^{-1}} * g^{ID^{-1}}) \bmod p < p$$
$$= g^{(ID/2)^{-1}} \bmod p$$

改成

$$g^{ID_A^{-1}} \bmod p = g^{ID_A^{-1}} \bmod p$$
$$= (g^{ID^{-1}})^n \bmod p$$
$$\because (g^{ID^{-1}})^n \bmod p < p$$
$$= (g^{n*ID^{-1}}) \bmod p < p$$
$$= g^{(ID/n)^{-1}} \bmod p$$，(n 為一個可以整除 ID 的正整數)

此時，攻擊者便可以利用產生出合法使用者 ID/n 去登入遠端的系統，進而入侵系統。這個攻擊方法擴充了攻擊方法一，只要攻擊者所選用 n 可以整除攻擊者所選用的帳號 ID 時，攻擊者便可以利用這個帳號 ID 來產生其他的合法使用者，因此，對於攻擊方法二來說，攻擊者最好選用可以被多種不同正整數整除的帳號 ID 註冊。

## 攻擊方法三：

假設攻擊者得到了兩個含有 $K_1=h(PW_1)\oplus(g^{ID_1^{-1}} \bmod p)$ 和 $K_2=h(PW_2)\oplus(g^{ID_2^{-1}} \bmod p)$ 的智慧卡以及這兩個智慧卡的密碼 $PW_1$ 和 $PW_2$，並且同時滿足 $(g^{ID_1^{-1}} \bmod p) * (g^{ID_2^{-1}} \bmod p) < p$ 與 $ID_1*ID_2$ 可以被 $ID_1+ID_2$ 所整除這兩個條件時，攻擊者便可以不需要知道p值為何，利用 $g^{ID_1^{-1}} \bmod p$ 和 $g^{ID_2^{-1}} \bmod p$ 去計算：

$ID_A=ID_1*ID_2$

$$g^{ID_A^{-1}} \bmod p = g^{ID_A^{-1}} \bmod p$$
$$= (g^{ID_1^{-1}} \bmod p)*(g^{ID_2^{-1}} \bmod p)$$

$$\because \text{當}(g^{ID_1^{-1}} \bmod p) * (g^{ID_2^{-1}} \bmod p) < p$$

$$= g^{ID_1^{-1}} * g^{ID_2^{-1}} \bmod p$$

$$= g^{ID_1^{-1}+ID_2^{-1}} \bmod p$$

$$= g^{(ID_1+ID_2)/ID_1*ID_2} \bmod p$$

$$= g^{((ID_1*ID_2)/(ID_1+ID_2))^{-1}} \bmod p$$

此時，這個攻擊者便可以假冒成一個合法的使用者$(ID_1*ID_2)/(ID_1+ID_2)$。

## 攻擊方法四：

擴充攻擊方法三，我們還可以計算出其他的合法使用者，如果攻擊者擁有 n 塊合法的智慧卡，而且同時滿足$\prod_{i=1}^{n}(g^{ID_i^{-1}} \bmod p) < p$和$(\sum_{i=1}^{n}\frac{1}{ID_i})^{-1}$必須為一正整數這兩個條件時，攻擊者便可以在不需要知道 p 值為何，去計算出多組偽裝的使用者驗證資訊，靠著計算：

將攻擊方法三的

$$ID_A=ID_1*ID_2$$

改成

$$ID_B=(\sum_{i=1}^{n}\frac{1}{ID_i})^{-1}$$

並且將

$$g^{ID_A^{-1}} \bmod p = (g^{ID_1^{-1}} \bmod p)*(g^{ID_2^{-1}} \bmod p\ \ )$$

$$\because \text{當}(g^{ID_1^{-1}} \bmod p) * (g^{ID_2^{-1}} \bmod p) < p$$

$$= g^{ID_1^{-1}+ID_2^{-1}} \bmod p$$

$$= g^{(ID_1+ID_2)/ID_1*ID_2} \bmod p$$

$$= g^{(ID_1*ID_2/ID_1+ID_2)^{-1}} \bmod p$$

改成

$$g^{ID_B^{-1}} \bmod p = \prod_{i=1}^{n}(g^{ID_i^{-1}} \bmod p)$$

$$\because \prod_{i=1}^{n}(g^{ID_i^{-1}} \bmod p) < p$$

$$= g^{\sum_{i=1}^{n}\frac{1}{ID_i}} \bmod p$$

此時，這個攻擊者便可以以其他使用者$ID_B=(\sum_{i=1}^{n}\frac{1}{ID_i})^{-1}$，入侵系統。

## 4.3 本研究所提出的改善驗證機制

下面，本研究將會提出一個改善驗證機制，這個改善驗證機制使用智慧卡當作工具，並且不需要在伺服器上存放任何的使用者驗證表，它改善西元 2005 年學者 Chang 與學者 Chang 所提出的驗證機制有著使用者假冒攻擊(Impersonation Attack)的安全性問題。這個改善驗證機制主要可以包括三個部份：註冊期(Registration phase)、登入期(login phase)和驗證期(verification phase)，整個驗證流程如下所示。

### 4.3.1 註冊期(Registration Phase)

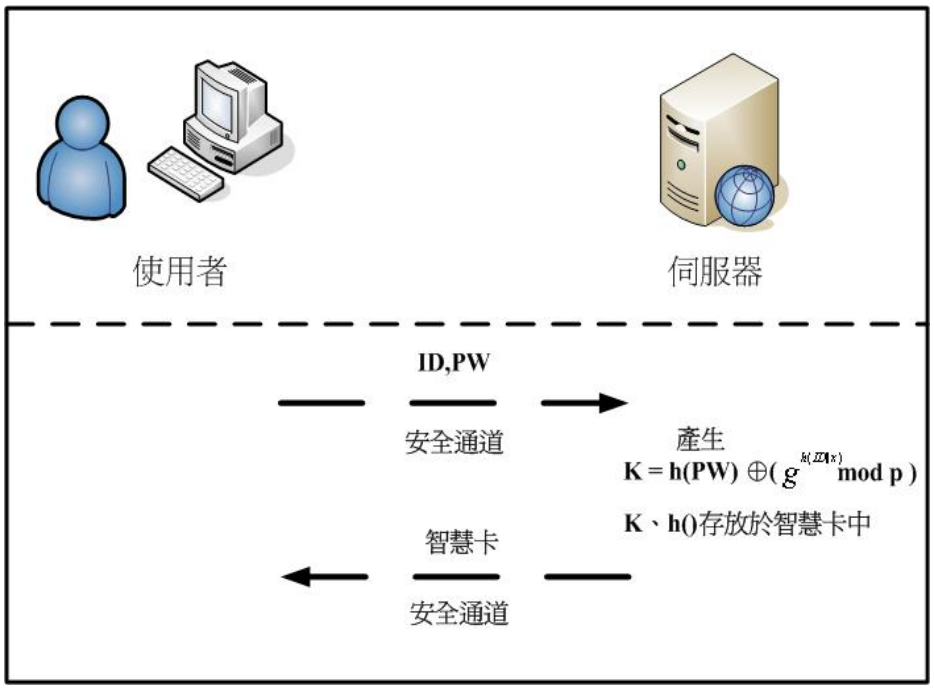


**圖 4.3：本研究所提出改進驗證機制的註冊期**

整個登入期主要執行在伺服器上，當一個使用者想要註冊成為一個新的合法使用者時，這個使用者必須要將使用者帳號(ID)和使用者密碼(PW)以安全通道的方式送到伺服器，以進行註冊。令 x 為伺服器的私密鑰匙，g 是一個在 GF(p)的質因數，p 則是一個非常大的質數，x、g 和 p 是三個被保存在伺服器的系統參數，並不對外公開，整個註冊期驗證流程如下所示：

Step1: 使用者自由地選取密碼(PW)之後，使用者將他的使用者帳號(ID)和使用者密碼(PW)以安全通道的方式送往伺服器。

Step2: 當伺服器收到使用者送過來的使用者帳號(ID)和使用者密碼 (PW)，伺服器計算 K=h(PW) $\oplus$( $g^{h(ID\|x)}$ mod p)，之後，伺服器將 K 和 h()存放在智慧卡中，並且將智慧卡以安全通道的方式交給遠端的使用者使用。

## 4.3.2 登入期(Login Phase)

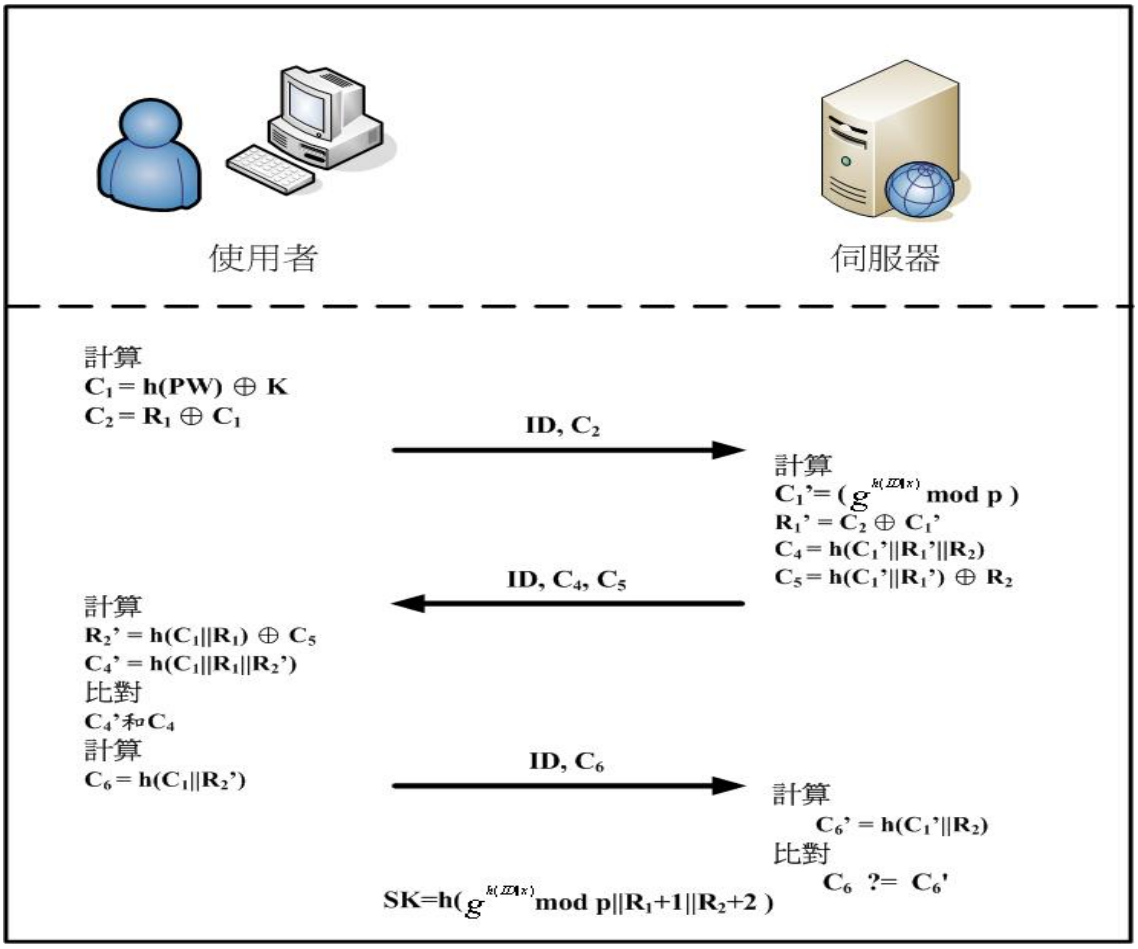


**圖 4.4：本研究所提出改進驗證機制的登入期**

當一個合法的使用者想要登入系統，這個使用者需要先將所擁有的智慧卡插入讀卡機中，並且輸入使用者的帳號(ID)和使用者的密碼(PW)，所有的登入期步驟如下所示：

Step1: U->S: ID、$C_2$

使用者使用PW、K、$R_1$去計算$C_1 = h(PW) \oplus K = (g^{h(ID||x)} \bmod p)$ 和$C_2 = R_1 \oplus C_1$，$R_1$是一個指使用一次的隨機亂數，隨後，使用者將ID、$C_2$送往伺服器。

Step2: 當伺服器收到使用者傳來的訊息之後，伺服器會使用使用者所傳來的ID、x、g、p去計算出$C_1' = (g^{h(ID||x)} \bmod p)$，並且利用$C_1'$、$C_2$去計算出$R_1' = C_2 \oplus C_1'$。

Step3: 伺服器使用$C_1'$、$R_1'$、$R_2$去計算$C_4 = h(C_1'||R_1'||R_2)$並且使用$C_1'$、$R_1'$、$R_2$去計算$C_5 = h(C_1'||R_1') \oplus R_2$。

Step4: S->U: ID、$C_4$、$C_5$

伺服器將ID、$C_4$、$C_5$送往伺服器。

Step5: U->S: ID、$C_6$

當使用者收到伺服器傳送過來的訊息之後，使用者使用$C_1$、$R_1$、$C_5$ 去計算$R_2$' = $h(C_1||R_1) \oplus C_5$，並且使用$C_1$、$R_1$、$R_2$'去計算$C_4$' = $h(C_1||R_1||R_2')$。比對$C_4$'和$C_4$，如果他們相等，使用者便使用$C_1$、$R_2$'去計算$C_6 = h(C_1||R_2')$，隨後，便將ID、$C_6$送往伺服器，否則，便中斷與伺服器之間的連線。

Step6:當伺服器收到使用者傳送過來的訊息之後，伺服器使用$C_1$'、$R_2$去計算$C_6' = h(C_1'||R_2)$，並且比對$C_6$和$C_6$'。如果他們相等的話，伺服器接受使用者的登入要求，並且產生一把階段鑰匙(session key) SK = $h(g^{h(ID||x)} \bmod p \,||\, R_1+1 \,||\, R_2+2)$；否則，伺服器拒絕這個連線要求。

## 4.4 安全性分析

下面將利用幾個著名的攻擊方法來證明本研究的驗證機制是安全的。

### 4.4.1 重送攻擊(Reply Attack)

本研究所提出的驗證機制使用隨機亂數(Nonce)去避免攻擊者利用重送攻擊(Reply Attack)的方式來入侵系統。因為隨機亂數$R_1$和隨機亂數$R_2$在每次連線時都會自動產生而且僅使用一次，所以假設一個攻擊者想要入侵系統，這個攻擊者不能以之前竊聽來的合法使用者登入訊息重新送往伺服器，進而入侵系統，因此，我們的方法可以有效避免重送攻擊(Reply Attack)。

### 4.4.2 驗證表被竊攻擊(Stolen-verifier Attack)

因為伺服器不需要存放任何的驗證表，一個攻擊者不可能藉由使用驗證表被竊攻擊(Stolen-verifier Attack)去竊取入侵系統，因此，本研究所提出的驗證機制可以有效的避免驗證表被竊攻擊(Stolen-verifier Attack)。

### 4.4.3 伺服器假冒攻擊(Server Spoofing Attack)

在本研究的驗證機制中，很明顯的要求使用者在讓伺服器驗證之前必須要知道伺服器是否正確無誤，因此，假如有一個攻擊者想要假冒依台伺服器去欺騙使用者，這個攻擊者必須要擁有伺服器的私密鑰匙 x、g、p，在驗證的過程中，如果使用者發現伺服器是假冒的，這個遠端使用者便會結束所有的連線，所以本研究的驗證機制可以避免伺服器假冒攻擊(Server Spoofing Attack)。

### 4.4.4 使用者偽裝攻擊(Impersonation Attack)

在本研究的驗證機制中，一個攻擊者不可能假冒成一個合法的使用者入侵系統，因為要成功的假冒成使用者，這個攻擊者必須要先知道$g^{h(ID||x)} \bmod p$去產生驗證過程中所需的驗證資訊，這是因為所有的驗證過程中的伺服器與使用者的訊息被$g^{h(ID||x)} \bmod p$所保護，而且$g^{h(ID||x)} \bmod p$只有存放在使用者的智慧卡並且被使用者的密碼所保護，因此，如果有攻擊者試著用任何使用者偽裝攻擊入侵系統是不可能成功的，所以本研究的驗證機制可以有效的避免使用者偽裝攻擊(Impersonation Attack)。

### 4.4.5 階段鑰匙的安全性(Security of Session Key)

一把階段鑰匙(session key)是由$g^{h(ID||x)} \bmod p$、隨機亂數$R_1$、隨機亂數$R_2$所產生，這些參數值只有被伺服器與使用者所知悉，當伺服器與使用者之間的通訊中斷時，這把鑰匙將不會使用在下次的通訊，當使用者重新進入系統時，一把新的階段鑰匙(session key)將會重新產生。假設一個攻擊者得到了一把階段鑰匙(session key)，這個攻擊者也無法利用這一把階段鑰匙(session key)去解密其他連線的通訊資料，因為隨機亂數$R_1$和隨機亂數$R_2$是由伺服器及使用者所隨機產生，因此攻擊者不可能利用一把已知的鑰匙去推算出下一把或是其他把階段鑰匙(session key)，而且這些隨機亂數的值很大，攻擊者不可能藉由猜測的方式去猜測出這把鑰匙的值，因此，階段鑰匙(Session Key)是十分的安全的。

## 4.5 與其他的驗證機制比較

在這個地方，將會將本研究的改善驗證機制與學者 Chang 等人所提出的驗證機制做一個比較。第一、因為本研究的驗證機制與學者 Chang 等人所提出的驗證機制都不需要在伺服器上存放任何的驗證表，所以都不會被驗證表被竊攻擊(Stolen-verifer Attack)所攻擊。第二、本研究的驗證機制與學者 Chang 等人所提出的驗證機制都沒有使用時間戳記(timestamp)，因此本研究的驗證機制與學者 Chang 等人所提出的驗證機制都沒有時間同步的問題。第三、本研究的驗證機制與學者 Chang 等人所提出的驗證機制在驗證的過程中都提供了互相驗證(Mutual Authentication)，讓使用者與伺服器都可以互相驗證使用者與伺服器的身分，因此，使用者與伺服器都可以得知通訊的對方是否合法。第四、在本研究的驗證機制中，當伺服器與使用者相互驗證成功之後，將會產生出一把階段鑰匙(session key)來加密伺服器與使用者之間之後的通訊訊息，以保障這些通訊訊息傳輸的安全性，但是學者 Chang 等人所提出的驗證機制並沒有提供階段鑰匙(Session Key)的功能。第五、本研究指出學者 Chang 等人的驗證機制容易遭受使用者偽造攻擊(Impersonation ATtack)，而本研究的驗證機制因為改善了這個安全性問題，因此不會遭受攻擊者使用使用者偽造攻擊

(Impersonation Attack)所來入侵系統，所以沒有使用者偽裝攻擊的安全性問題。整個彙整如下表所示：

**表 4.2：與學者 Chang 等人所提出的驗證機制比較**

| | 是否需要驗證表 | 是否有時間同步的問題 | 是否提供互相驗證 | 是否提供(Session key) | 是否遭受使用者偽裝攻擊 |
|---|---|---|---|---|---|
| 本研究 | No | No | Yes | Yes | No |
| 學者 Chang[24] | No | No | Yes | No | Yes |

## 4.6 結論與後續分析

在這個章節中，本研究指出西元2005年學者Chang等人[24]所提出的無認證表驗證機制有著使用者偽裝攻擊的安全性問題，為了避免這個安全性問題的發生，本研究提出了一個改善的無驗證表驗證系統去改善學者Chang 等人所提出的無認證表驗證機制。在本研究的驗證機制中，因為僅僅利用單向雜湊函數和互斥運算來進行運算，因此本研究驗證機制非常適合低運算的網路設備系統，但是相對的，如何選用一個安全的單向雜湊函數對於本研究的驗證機制來說，是非常重要的，西元2005年，山東大學的王小雲教授等人[22]就曾經破解了MD5單向雜湊函數，因此，如果要選用本驗證方法來驗證遠端的使用者，就必須要使用更加安全的單向雜湊函數去保障驗證機制的安全性。

# 第五章 以隨機亂數為基礎(Nonce-based)之無驗證表驗證機制

## 5.1 前言

隨著網路科技的日益發展，越來越多的系統利用網際網路的便利性來提供使用者進行資訊的交流以及互通，但是在這樣的開放環境下，連線的伺服器及使用者很難去知道連線的對方是否為正確無誤，因此如何驗證通訊雙方的身分變得十分的重要。目前的網路系統大多是利用所謂傳統的帳號密碼認證方式，在系統中先建立一個所謂的認證表(verification table)，並將使用者的帳號密碼存放在這個使用者認證表中，當使用者要登入時，系統會查詢一下這個認證表(veritication table)，是否有這一個使用者的存在，如果存在便讓這個使用者使用這個系統內部的資訊。這樣的實作方式雖然很簡單，但是卻也無形中產生了許多安全性的問題，例如：驗證表的安全性、傳輸帳號密碼的安全性等等問題。為了避免因為驗證表遭到攻擊者盜竊，而導致使用者的帳號密碼被攻擊者所知悉，西元1981年學者Lamport[14]提出一種改善方法，這個方法是先將密碼以雜湊函數將其加密後，再存放到伺服器上，這樣的改善方式雖然可以讓攻擊者不能夠在第一時間就得之密碼值為何，但是攻擊者還是可以透過一些解密工具來破解出該密碼值，只是所需花費的時間很大。

不過，隨著電腦運算速度越來越快的影響，攻擊者破解密碼的速度也越來越快，單純的利用單向雜湊函數加密密碼已經不敷使用，為了要有效解決驗證表被竊的安全性問題，開始有學者提出以無認證表的方式來驗證遠端的使用者身分。西元1990年，Hwang、Chen和Lai[19]提出了一個需要智慧卡的無認證表機制，這個認證機制的特色是伺服器不需要存放任何的驗證表，讓認證表被竊的安全性問題得到了一個解決方案。從此以後，許多以智慧卡為基礎且不需要驗證表的驗證機制紛紛出爐，這些出爐的驗證機制有許多都是以離散對數的困難度或是以非對稱性加密的方式來保護傳輸資訊的安全性，因此這些認證機制在執行的效能上十分的大而且困難，為了避免效能上的問題，學者Sun於西元2000年[8]提出了一種以單向雜湊函數及時間戳記(timestamp)為基礎之無認證表驗證機制，有別於其他無認證表驗證機制，這個驗證機制的特點在於僅僅利用單向雜湊函數與Xor運算，大大降低伺服器與使用者的運算量，不過它的驗證方法還是有著安全性的問題，因此，後續陸陸續續有學者以此為基礎，發表了許許多多相關的研究。

可惜的是，這些研究因為都使用了時間戳記(Timestamp)來保障資訊的傳輸，因此，都有著時間同步的問題，所以開始有學者朝向以隨機亂數(Nonce)的方式來建立驗證機制，因為唯有這樣，才能夠適用於目前分散式的網路環境之中。西元2005年學者Lee、Kim、Yoo[18]提出了一個以隨機亂數(Nonce)為基礎的無驗證表驗證機制，在驗證的過程中不需要有驗證表的存在，而且僅僅利用了單向雜湊函數來保護驗證資料的安全性。湊巧的是西元2005年學者Chen、Yeh[23]也提出了另一個以隨機亂數(nonce)為基礎的無認證表驗證機制，在這一個認證機制中不但僅僅只使用單向雜湊函數來驗證使用者外，更在驗證後加入了階段鑰匙(Session Key)的訂定，以加解密後續資料、保護後續資料傳輸的安全性。

在本研究中檢視了兩個以隨機亂數為基礎(nonce-based)的無驗證表驗證機制，分析其缺失及安全性問題，並提出了一個以隨機亂數(nonce)為基礎的驗證機制，這個驗證機制在驗證的過程僅僅利用單向雜湊函數、Xor運算，所以在執行上所需花費運算量十分的小，十分適合目前的網路環境使用，而且在確認雙方的身分合法之後，還會定義出一把階段鑰匙(session key)，用來加密雙方後續傳輸的資料。

## 5.2 以隨機亂數(nonce)為基礎的驗證機制

這一節本研究將會檢視學者Lee、Kim與Yoo所提出的以隨機亂數(nonce)為基礎的無驗證表驗證機制以及學者Chen與Yeh所提出以隨機亂數(nonce)為基礎的無認證表驗證機制，這兩個驗證機制的通訊過程可以簡單的分為註冊階段(Registration Phase)、登入階段(Login Phase)、驗證階段(Authentication Phase)等三個階段，在介紹前首先要先定義一些本章節所需要使用的運算符號，如下表所示：

**表 5.1：符號說明表**

| 符號 | 說　　明 |
|---|---|
| h() | 單向雜湊函數 |
| X | 遠端伺服器的秘密參數，為伺服器所有 |
| ⊕ | 互斥或 |
| ID | 使用者帳號 |
| PW | 使用者密碼 |
| \|\| | 參數連結 |
| N | 使用者及伺服器產生的隨機亂數 |
| X->Y:M | 表示從X傳送一個M訊息到Y |
| U，S | 分別代表使用者及伺服器 |

### 5.2.1 學者 Lee、Kim 與 Yoo 所提出的驗證機制

西元2005年學者Lee、Kim、Yoo[18]提出了一個以隨機亂數(nonce)為基礎的無認證表驗證機制，這個驗證機制利用智慧卡作為儲存工具，在運算上僅僅使用了單向雜湊函數以及Xor運算，來達到互相驗證的效果，因此十分合適分散式的網路系統，下面將開始介紹這個驗證機制。

### 5.2.1.1 註冊期(Registration Phase)

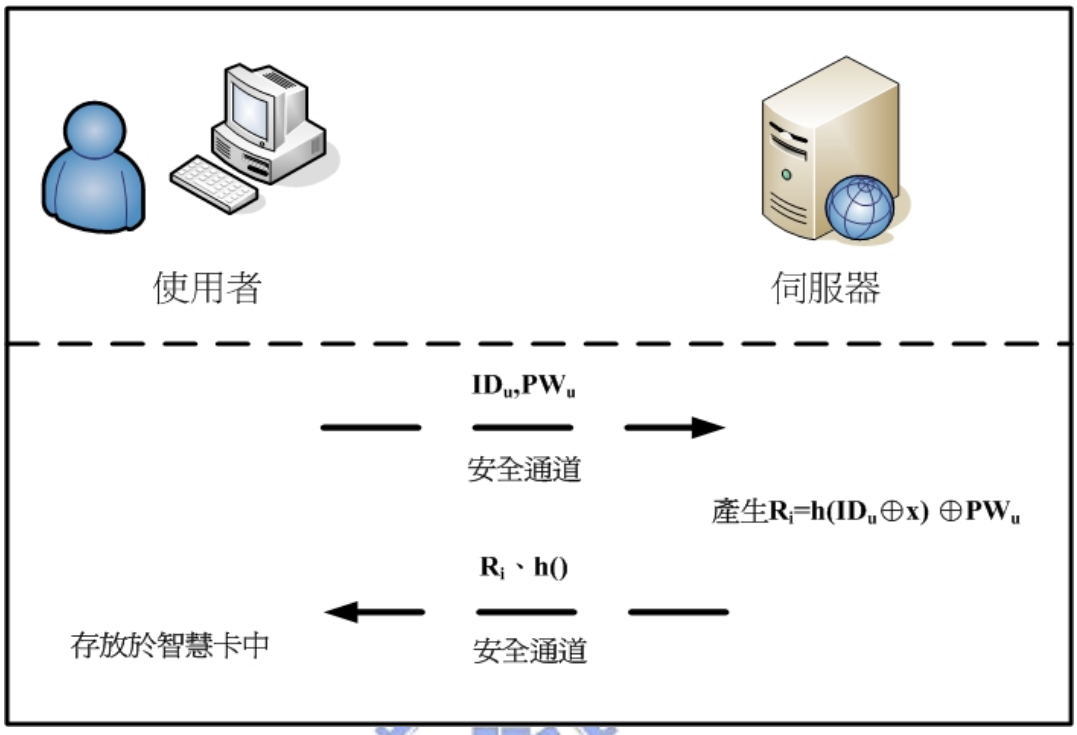


**圖 5.1：學者Lee等人所提出驗證機制的註冊期**

整個註冊期主要是在伺服器上執行，當一個使用者想要註冊成一個新的合法使用者時，它首先必須要將自身的帳號($ID_u$)以及密碼($PW_u$)以安全的通道傳送到伺服器，x為伺服器的私密變數，為伺服器所擁有，整個註冊期步驟如下所示：

Step1：$U_u$->S:$ID_u$、$PW_u$

使用者傳送自己的帳號($ID_u$)以及密碼($PW_u$)以安全通道的方式傳送到伺服器。

Step2：$R_i=h(ID_u \oplus x) \oplus PW_u$

當伺服器收到使用者傳送過來的$ID_u$、$PW_u$之後，利用$ID_u$、$PW_u$以及自身的私密參數x計算出$R_i$。

Step3：S->$U_u$:$R_i$、h()

伺服器將$R_i$以及h()以安全通道的方式傳送到使用者，使用者在將他們存放到自己的智慧卡之中。

### 5.2.1.2 登入期(Login Phase)

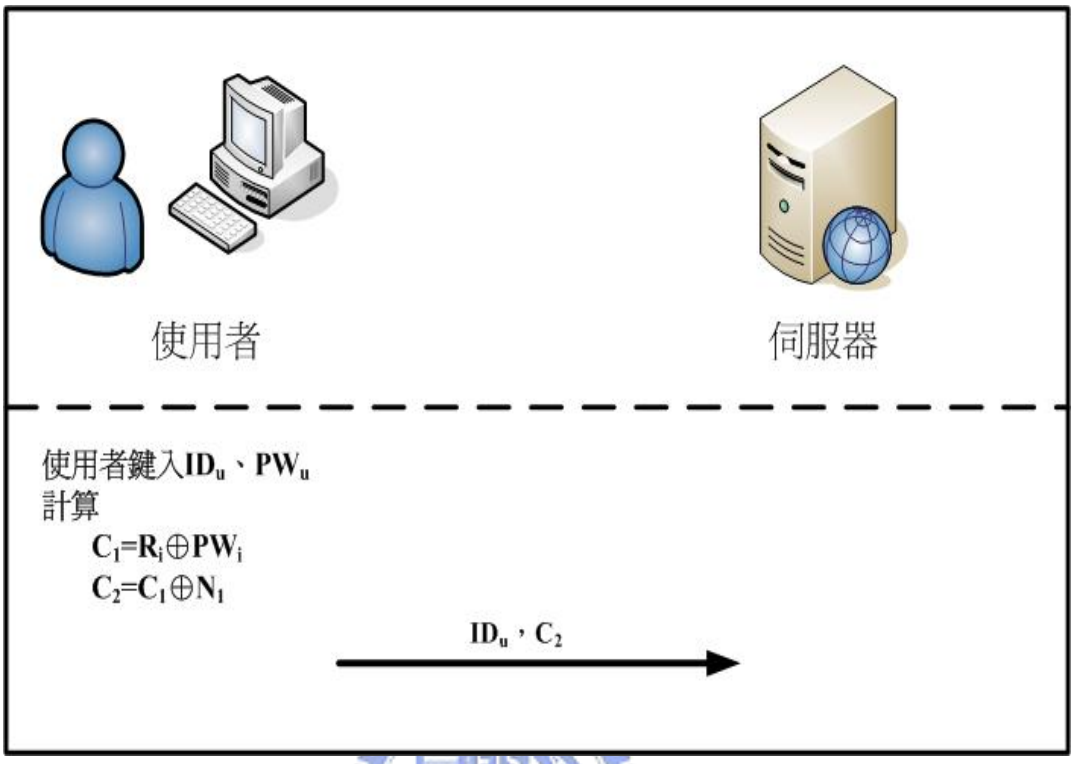


**圖 5.2：學者Lee等人所提出驗證機制的登入期**

當一個合法的使用者想要登入系統的時候，使用者必須先將自己的智慧卡插入讀卡機中，並且輸入個人的使用者帳號($ID_u$)以及密碼($PW_u$)，這時，讀卡機會解開$C_1$並產生一個隨機亂數$N_1$，整個登入期的步驟如下所示：

Step1：$C_1=R_i \oplus PW_u$

使用者利用使用者密碼($PW_u$)去計算出$C_1=h(ID_u \oplus x)$。

Step2：$C_2=C_1 \oplus N_1$

使用者產生一個隨機亂數$N_1$，並且利用$N_1$與$C_1$計算出$C_2$。

Step3：$U_u$->S:ID、$C_2$

使用者將ID、$C_2$傳送到伺服器。

### 5.2.1.3 驗證期(Authentication Phase)

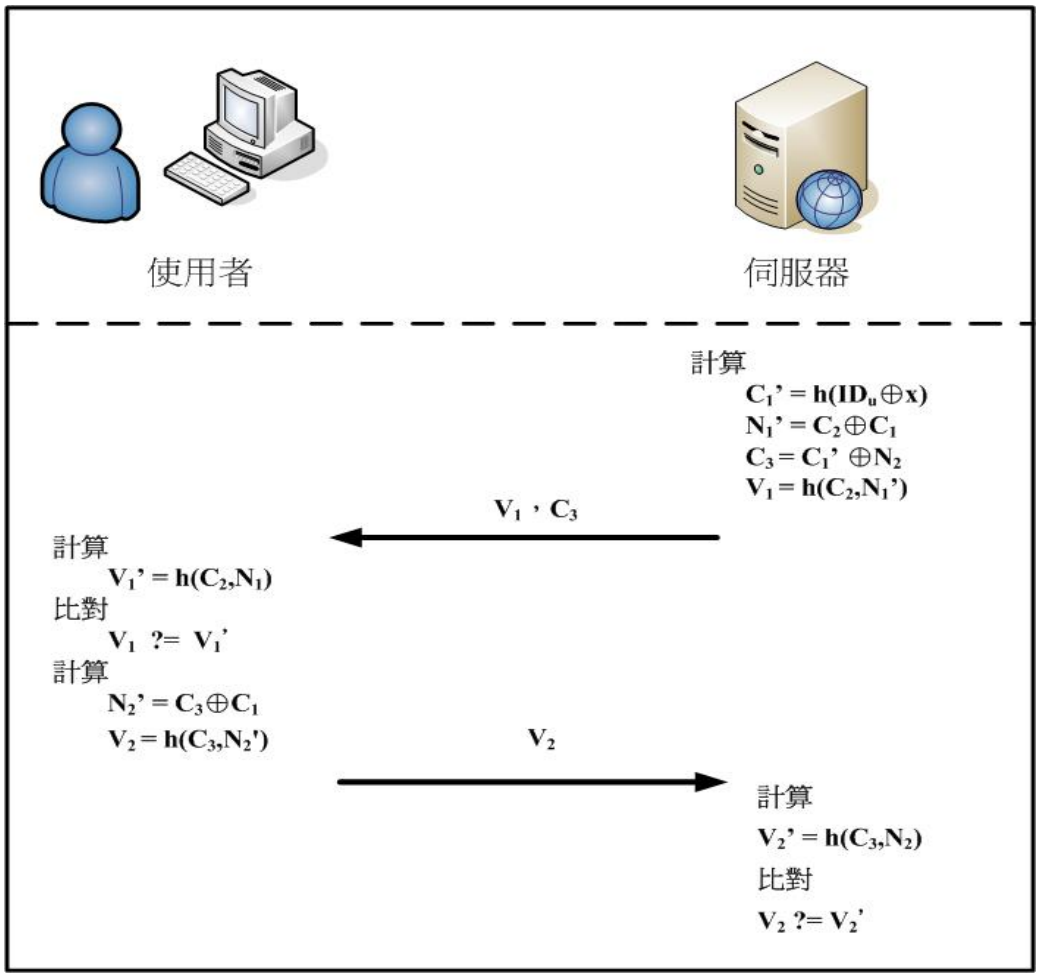


**圖 5.3：學者Lee等人所提出驗證機制的驗證期**

當伺服器收到使用者傳來的登入請求之後，它會先利用自己的私密變數x來計算出$N_1$，而伺服器自己本身也產生一個隨機亂數$N_2$，整個驗證流程將會以這兩個隨機變數來驗證使用者雙方的身分，整個驗證時期通訊流程如下：

Step1：$C_1' = h(ID_u \oplus x)$

當伺服器收到使用者傳來的$ID_u$以及$C_2$之後，首先利用$ID_u$以及伺服器的私密參數計算出$C_1'$。

Step2：$N_1' = C_2 \oplus C_1'$

伺服器利用$C_2$以及$C_1'$計算出$N_1'$。

Step3：$C_3 = C_1' \oplus N_2$

伺服器利用$C_1'$以及伺服器所產生的$N_2$計算出$C_3$。

Step4：$V_1 = h(C_2, N_1')$

伺服器利用$C_2$以及$N_1'$計算出$V_1$。

Step5：S -> $U_u$：$V_1$、$C_3$

伺服器將$V_1$、$C_3$傳送給使用者。

Step6：$V_1$' = h($C_2$,$N_1$)

當使用者收到$V_1$、$C_3$時，使用者利用原本的$C_2$以及$N_1$計算出$V_1$'。

Step7：$V_1$ ?= $V_1$'

使用者比對$V_1$與$V_1$'，如果它們相同，代表伺服器正確無誤，如果不相等代表伺服器的身分有問題，不進行下面的過程，中斷與伺服器的連線。

Step8：$N_2$' = $C_3$⊕$C_1$

使用者利用$C_3$以及$C_1$計算出$N_2$'。

Step9：$V_2$ = h($C_3$, $N_2$')

使用者利用$C_3$以及$N_2$'計算出$V_2$

Step10：$U_u$->S：$V_2$

使用者將$V_2$傳送到伺服器。

Step11：$V_2$' = h($C_3$,$N_2$)

當伺服器收到使用者傳送過來的$V_2$之後，利用原本的$C_3$以及$N_2$計算出$V_2$'= = h($C_3$,$N_2$)。

Step12：$V_2$ ?= $V_2$'

比對$V_2$是否等於$V_2$'，如果相等，代表使用者身份正確，如果不相等，代表使用者身份有問題，伺服器會拒絕使用者的登入要求。

## 5.2.2 驗證機制分析

在安全性上，學者Lee、Kim與Yoo的驗證機制有安全性的問題存在，其原因在於Xor的運算(a⊕0=0⊕a=a)。如果一個攻擊者在登入時傳送$ID_u$=0,$C_2$=0給伺服器以進行驗證程序時，此時伺服器會利用$ID_u$與伺服器的私密參數x計算出$C_1$'=h($ID_u$⊕x)，這時攻擊者所提供的值導致$C_1$'=h(x)，讓原本應該要計算出$N_1$'=$C_2$⊕$C_1$'，卻因為攻擊者的關係，反而計算出$N_1$'=0⊕h(x)=h(x)。讓原本應該要計算$C_2$=$C_1$'⊕$N_2$，$V_1$=h($C_2$,$N_1$')，卻因為攻擊者的關係，整個$V_1$就變成了h(0,h(x))，當伺服器將這個$V_1$傳送給攻擊者時，攻擊者便可以利用離線密碼猜測的方式來進行攻擊，首先先攻擊者可以猜測出h(x)的值為何，如果當攻擊者猜測出來的話，攻擊者便可以假冒一個使用者($ID_u$=0)進入系統，此外，攻擊者還可以再以離線密碼猜測來猜測伺服器的私密鑰匙x為何。如果當攻擊者很順利的得知伺服器的私密變數x時，攻擊者可以利用來計算出其他合法使用者帳號密碼或是假冒伺服器。

就使用上來說，整個h($ID_u$⊕x)是利用密碼以Xor運算的方式來保密，用這樣的方式來保密的話，通常使用者密碼($PW_u$)的長度必須要跟h($ID_u$⊕x)一樣長，如果使用者密碼($PW_u$)的長度比h($ID_u$⊕x)還要短的話，就必須將密碼以補充其他字元的方式或是以區塊性加密的方式來保護資訊，安全上大受影響，如果是以補充字元的方式的話，攻擊者可以藉由分析補充字元的規則為何，推導出部分的h($ID_u$⊕x)值，如

果是以區塊性的方式的話，攻擊者有可能藉由觀察密文的方式來推導出$h(ID_u \oplus x)$為何，但是如果密碼的長度與$h(ID_u \oplus x)$一樣的話，通常$h(ID_u \oplus x)$的長度是非常的長的，因此，要求使用者要記憶這樣長串的密碼是不可能的事情，因此使用上不夠人性化，除此之外，在認證完使用者的身分之後，這個驗證機制未加入如何訂定出階段鑰匙(session key)，用以保護後續資料的傳送。

## 5.3 學者 Chen 與學者 Yeh 所提出的驗證機制

西元2005年學者Chen與學者Yeh[23]也提出了一個以隨機亂數(Nonce)為基礎的無認證表驗證機制，在他們提出的驗證機制中除了提供互相驗證外，同時也加入了階段鑰匙(session key)的訂定，讓使用者與伺服器之間的後續資料能夠利用這把階段鑰匙(session key)進行加解密，以保障後續傳輸資訊的安全性。

### 5.3.1 註冊期(Registration Phase)

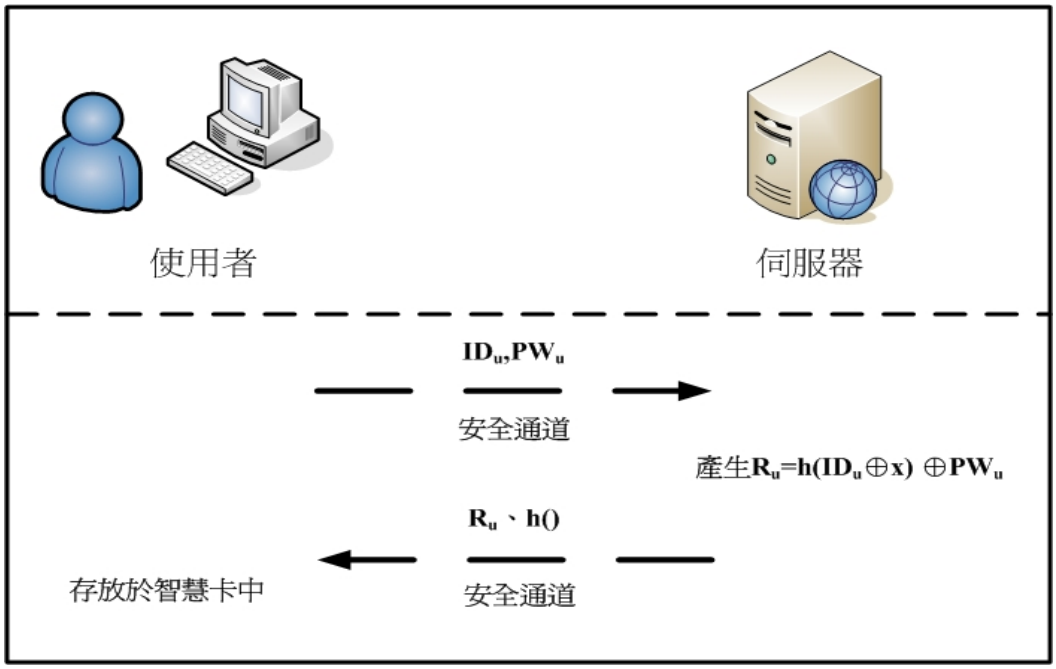


**圖 5.4：學者Chen等人所提出驗證機制的註冊期**

整個註冊期主要是在伺服器上執行，當一個使用者想要使用註冊成為一個新的合法使用者時，使用者首先必須要將自身的帳號($ID_u$)以及密碼($PW_u$)以安全的通道的方式傳送到伺服器，x為伺服器的私密變數，為伺服器所擁有，整個註冊期的步驟如下：

Step1：$U_u$ -> S：$ID_u$、$PW_u$

使用者將卡片插入讀卡機後，輸入自己的帳號($IDB_u$)、密碼($PWB_u$)，並

以安全通道傳送到伺服器中。

Step2：$R_u=h(ID_u \oplus x) \oplus PW_u$

當伺服器收到使用者的帳號($ID_u$)、密碼($PW_u$)之後，便與自己的秘密參數x結合，計算出$R_u=h(ID_u \oplus x) \oplus PW_u$。

Step3：S -> $U_u$：$R_u$、h()

伺服器將$R_u$、h()傳送到智慧卡存放起來，並且以安全通道的方式將智慧卡轉交給使用者使用。

## 5.3.2 登入期(Login Phase)

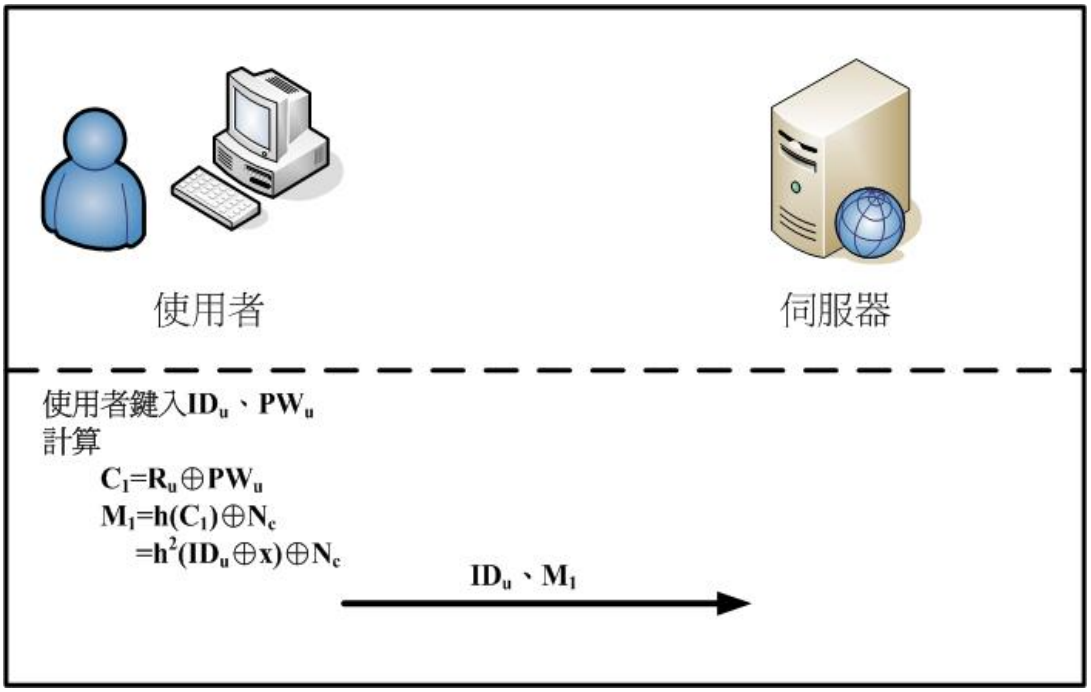


**圖 5.5：學者Chen等人所提出驗證機制的登入期**

當一個合法的使用者想要登入系統的時候，他必須先將智慧卡插入讀卡機之中，並輸入他的帳號($ID_u$)以及密碼($PW_u$)，此時讀卡機會解開$C_1$並產生一個隨機亂數$N_1$，整個登入期間進行下面的步驟：

Step1：$C_1=R_u \oplus PW_u$

當使用者輸入完密碼($PW_u$)之後，讀卡機利用$R_u$及$PW_u$計算出$C_1$。

Step2：$M_1=hP^2(ID_u \oplus x) \oplus N_c$

讀卡機產生一個隨機亂數$NB_c$，並利用$h(ID_u \oplus x)$和$N_c$去計算出$M_1$。

Step3：$U_u$ -> S：$ID_u$、$M_1$

使用者將$ID_u$、$M_1$傳送到伺服器。

## 5.3.3 驗證期(Authentication Phase)

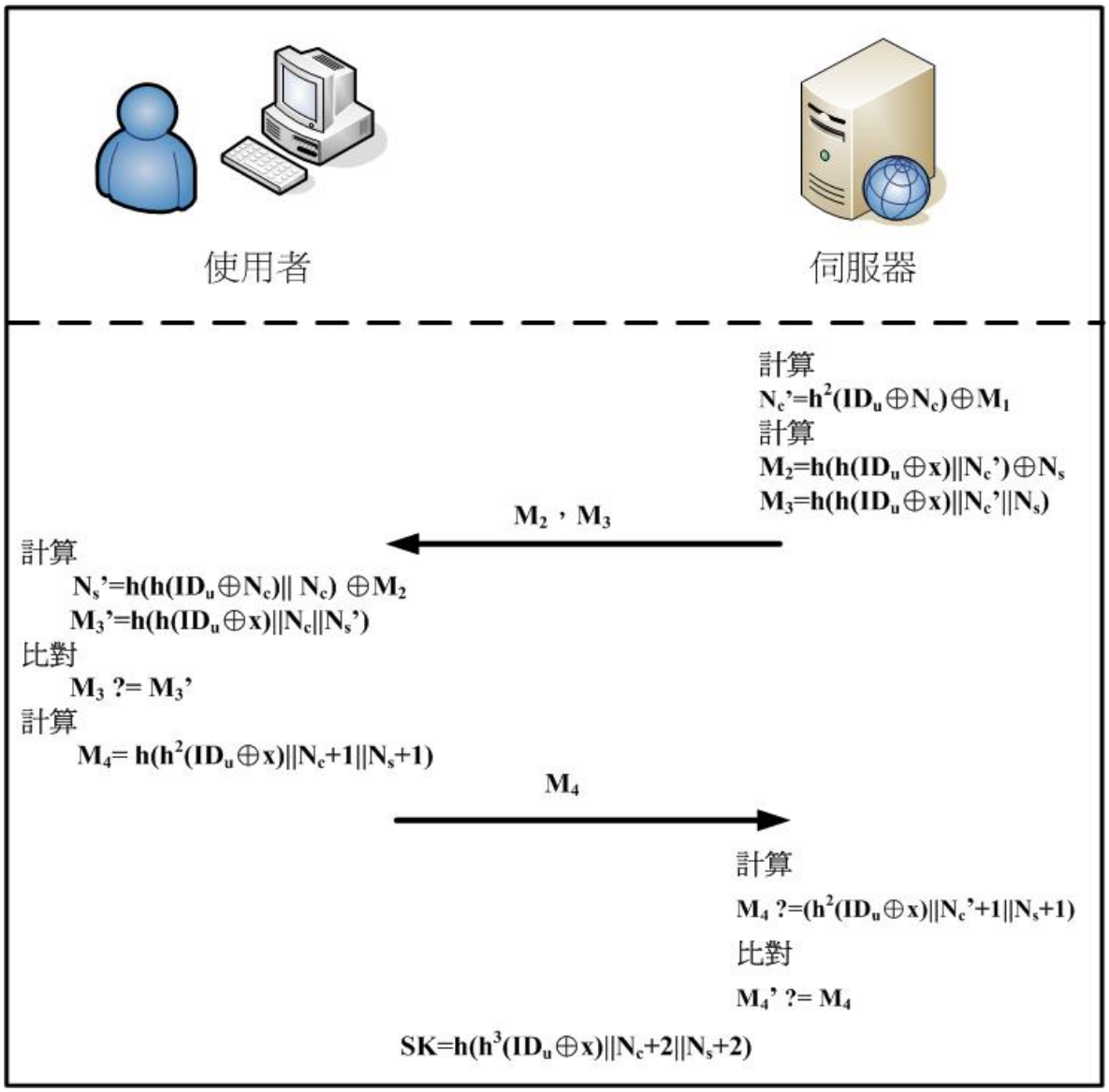


**圖 5.6：學者Chen等人所提出驗證機制的驗證期**

在驗證期中，使用者會使用在登入期間時產生的隨機亂數$N_c$以及伺服器在驗證時期時產生隨機亂數$N_s$來驗證使用者與伺服器的身分，而且每次連線都不一樣，因此，每次登入時驗證資訊皆不同，並在整個驗證的最後會產生一個階段鑰匙(session key)，用以保護後續資料傳輸的安全性，整個驗證時期的步驟如下：

Step1：S-> $U_u$：$M_2$，$M_3$， $M_2=h(h(ID_u \oplus x)||N_c') \oplus N_s$、$M_3=h(h(ID_u \oplus x)||N_c'||N_s)$

當伺服器收到使用者傳來的$ID_u$以及$M_1$之後，利用自己本身的私密參數x以及$ID_u$先計算出$h^2(ID_u \oplus x)$，再利用$h^2(ID_u \oplus x)$與$M_1$運算解出$N_c'$，$N_c'=h^2(ID_u \oplus x) \oplus M_1$，並且自己產生了一個隨機亂數$N_s$，用以計算出$M_2$、$M_3$，並將它送往使用者。

Step2：$N_s'=h(h(ID_u \oplus x)||\ N_c) \oplus M_2$

當使用者收到了伺服器傳來的$M_2$、$M_3$以後，利用$M_2$及$N_c$來計算出$N_s'$。

Step3：$M_3$ ?= $M_3'$

利用$N_s'$計算出$M_3'$ $=h(h(ID_u \oplus x)||N_c||N_s')$，並比對$M_3$與$M_3'$是否相等，如果相等，代表伺服器合法無誤，進行下面的驗證工作，如果不相等代表伺服器有問題，不繼續下面的步驟。

Step4：$U_u$->S：$M_4$，$M_4= h(h^2(ID_u \oplus x)||N_c+1||N_s+1)$

使用者計算$M_4$，$M_4=h(h^2(ID_u \oplus x))||N_c+1||N_s'+1)$，並將$M_4$傳送到伺服器進行驗證。

Step5：$M_4$ ?= $h(h^2(ID_u \oplus x)||N_c'+1||N_s+1)$

當伺服器收到使用者所傳來的$M_4$時，伺服器利用$N_c'$、$N_s$、$h^2(ID_u \oplus x)$來驗證$M_4$ ?= $h(h^2(ID_u \oplus x)||N_c'+1||N_s+1)$，如果驗證成功，便是使用者無誤，允許使用者與伺服器溝通。

Step6：在伺服器與使用者互相認證之後，利用之前的$N_c$以及$N_s$來建立一把階段鑰匙(session key)，$SK=h(h^3(ID_u \oplus x)||N_c+2||N_s+2)$

### 5.3.4 驗證機制分析

就使用上來說，整個$h(ID_u \oplus x)$ 是利用密碼以Xor運算的方式來保密，通常PW的長度跟$h(ID_u \oplus x)$一樣長，如果PW的長度比$h(ID_u \oplus x)$還要短的話，就必須將密碼以補充其他字元的方式或是以區塊性加密的方式來保護資訊，安全上大受影響，如果是以補充字元的方式的話，攻擊者可以藉由分析補充字元的規則為何，推導出部分的$h(ID_u \oplus x)$值，如果是以區塊性的方式的話，攻擊者有可能藉由觀察密文的方式來推導出$h(ID_u \oplus x)$為何，如果長度一樣的話，要求使用者要記憶這樣長串的密碼是不可能的事情，因此使用上不夠人性化。

這個機制與上面學者Lee、Kim、Yoo提出的驗證機制都是利用單向雜湊函數與Xor運算來達成身分的驗證，不過就其效能上來說，這個機制所使用的單向雜湊次數比前面的機制還要來的多，其運算量上比前面的還要來的大，但是這個驗證機制在驗證完使用者的身分之後，驗證的雙方也定義出一把階段鑰匙(session key)，用以保護後續資料傳輸的安全性。

## 5.4 本研究提出的驗證機制

有別於上面的驗證機制，本研究提出了另外一種以隨機亂數(Nonce)為基礎的無認證表驗證機制，這個驗證機制利用智慧卡當作存放工具，並且僅僅使用單向雜湊函數來進行雙方的驗證工作，整個身分認證機制可以分為註冊期(Registration Phase)、登入期(Login Phase)、驗證期(Authentication Phase)等三個階段，下面將會開始介紹本研究提出的驗證機制。

### 5.4.1 註冊期(Registration Phase)

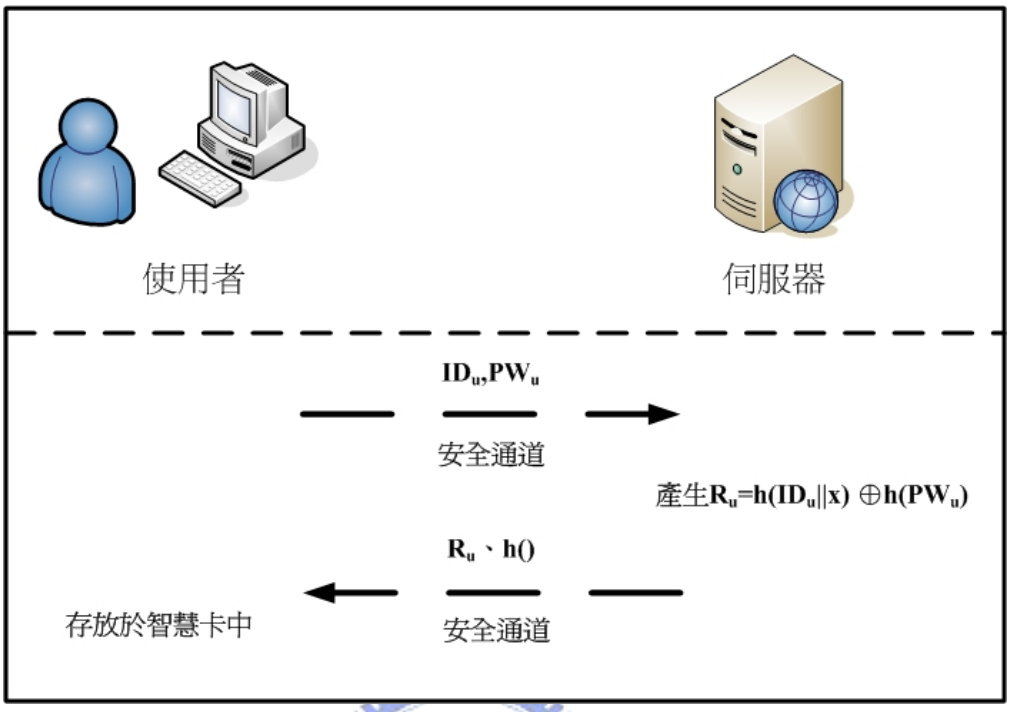


**圖 5.7：本研究所提出驗證機制的註冊期**

整個註冊期主要是在伺服器上執行，當一個使用者想要註冊成一個新的合法使用者時，使用者首先必須要先將自身的$ID_u$以及$PW_u$以安全的通道傳送到伺服器，x為伺服器的私密變數，為伺服器所擁有，在這邊要特別說明的是在很多無認證表系統的$R_u$都是$R_u=h(ID_u \oplus x) \oplus PW_u$，而本研究為了考慮到實作的方便性，因此，與其它的無認證表系統的設計不一樣的地方是，本研究採用的方式是$R_u=h(ID_u||x) \oplus h(PW_u)$，這樣使用者使用起來才不會要記十分長串的帳號$ID_u$或是需要透過其他的方法來讓私密鑰匙x與帳號ID的長度一樣，整個註冊期流程如下面所示：

Step1：$U_u$->S：$ID_u$

使用者將自己的$ID_u$、$PW_u$以安全通道的方式傳送到註冊中心進行註冊。

Step2：$R_u= h(ID_u||x)$

伺服器利用使用者的$ID_u$與自己的私密變數x，計算出$R_u= h(ID_u||x)$。

Step3：$C_0=R_u \oplus h(PW_u)$

伺服器利用$R_u$與$PW_u$計算出$C_0=R_u \oplus h(PW_u)$。

Step4：S->$U_u$：$C_0$

伺服器將$C_0$以安全管道的方式傳送給使用者使用，使用者將它存放到智慧卡中。

## 5.4.2 登入期(Login Phase)

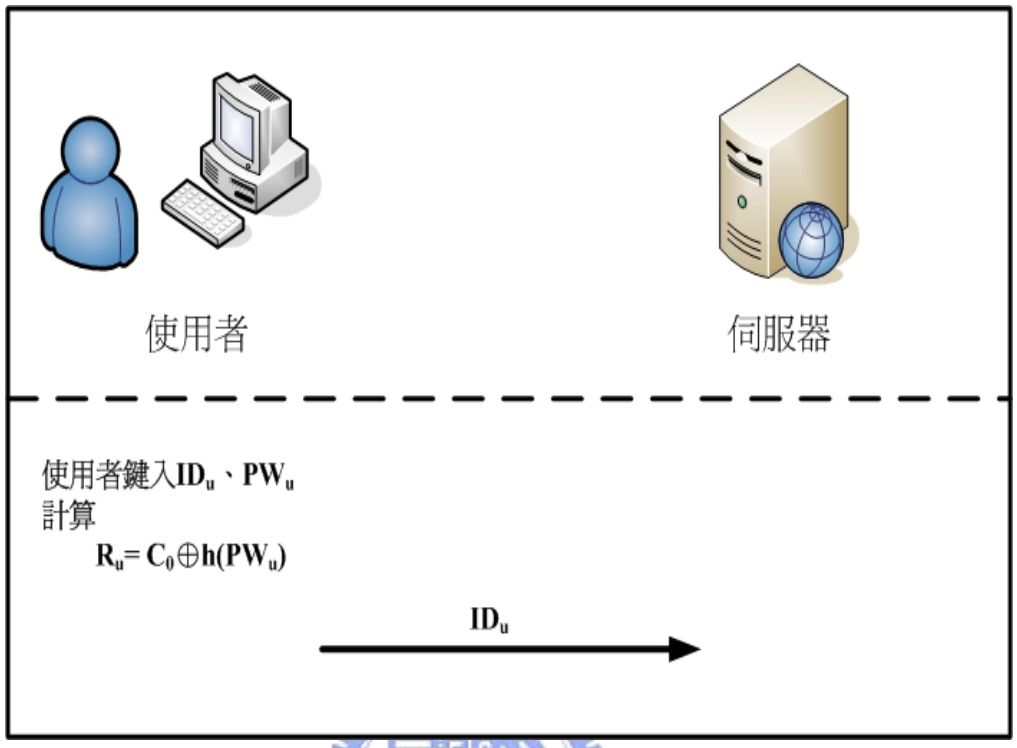


**圖 5.8：本論文所提出驗證機制的登入期**

當一個合法的使用者想要登入系統時，使用者必須先將個人的智慧卡卡片插入讀卡機中，此時，讀卡機會讀取智慧卡中所存放的$C_0$，並且要求使用者輸入他的帳號($ID_u$)以及密碼($PW_u$)，整個登入期間進行的步驟如下所示：

Step1：$R_u = C_0 \oplus h(PW_u)$

使用者利用$PW_u$以及智慧卡中的$C_0$計算出$R_u=h(ID_u||x)$。

Step2：$U_u$ -> S：$ID_u$

使用者將自身的$ID_u$傳送給伺服器

## 5.4.3 驗證期(Authentication phase)

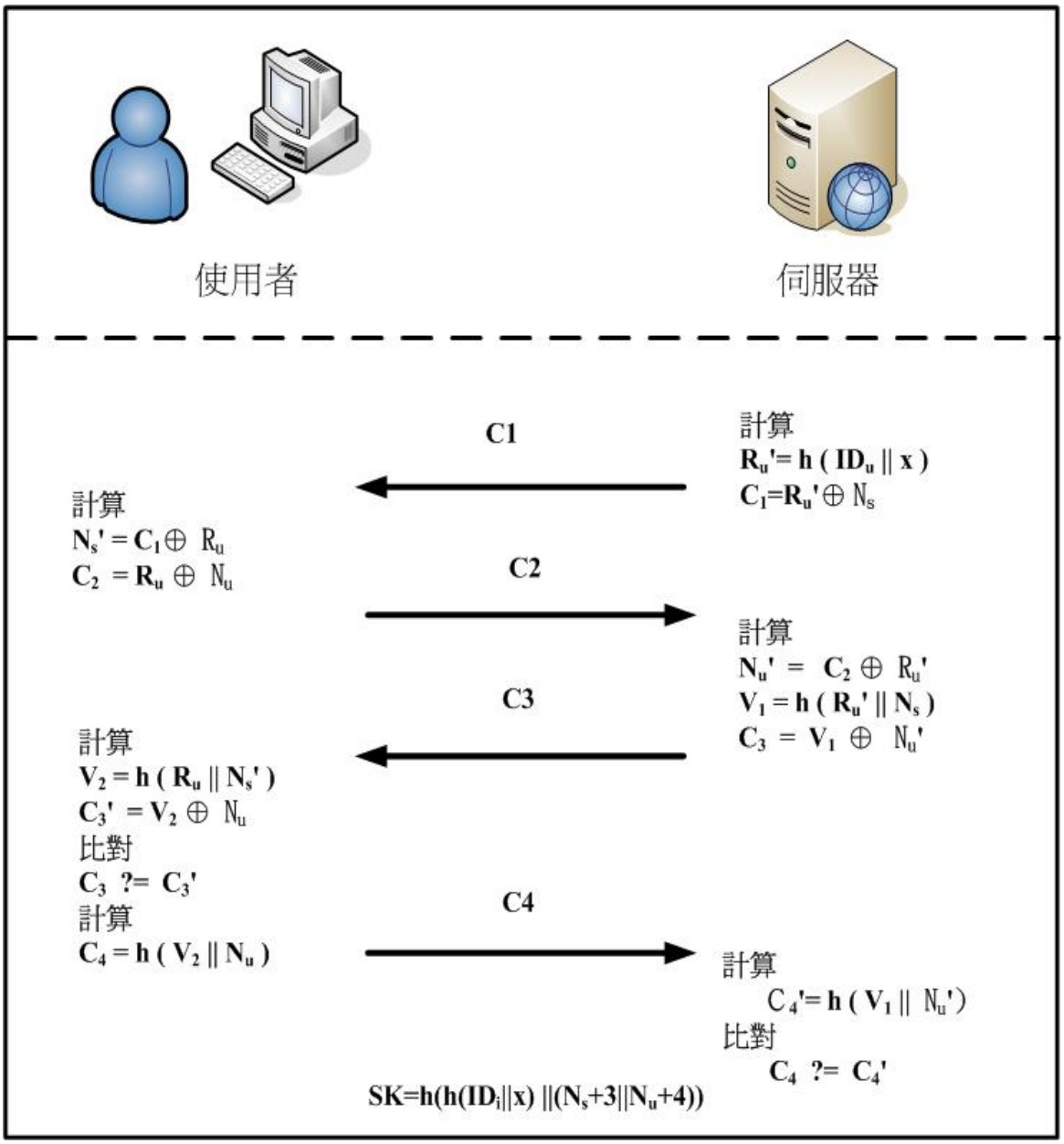


**圖 5.9：本論文所提出驗證機制的驗證期**

當使用者傳送了帳號($ID_u$)給伺服器之後，伺服器與使用者會各自產生隨機亂數$N_s$與隨機亂數$N_c$，然後互相傳遞交換給對方，當然隨機亂數$N_s$和隨機亂數$N_u$都會用$h(ID_u||x)$進行保護後再傳送給驗證的對方，以免被攻擊者利用竊聽的方式去得知這兩個隨機亂數。交換完這兩個隨機亂數之後，將會進行驗證對方在分解這兩個隨機亂數上是否正確，用以驗證使用者以及伺服器的身分是否合法。在驗證確認使用者以及伺服器的身分之後，通訊的雙方會產生一把階段鑰匙(session key)，用以加解密驗證時期以後資訊，保護資訊傳輸的安全性，以下是整個驗證時期的步驟：

Step1：$R_u'=h(ID_u||x)$

當伺服器收到使用者傳送過來的$ID_u$之後，用$ID_u$及伺服器的秘密參數x計算出$R_u'$。

Step2：$C_1 = R_u' \oplus N_s$

伺服器產生一個伺服器的隨機亂數Ns，利用$R_u'$及$N_s$計算出$C_1$。

Step3：S->$U_u$：$C_1$

伺服器將$C_1$傳送給使用者。

Step4：$N_s'=C_1 \oplus R_u$

當使用者收到$C_1$之後，先利用自己本身的$R_u$與$C_1$來計算出$N_s'$。

Step5：$C_2=R_u \oplus N_u$

使用者產生一個隨機亂數$N_u$，並將$N_u$與$R_u$計算出$C_2$。

Step6：$U_u$->S：$C_2$

使用者將$C_2$傳送給伺服器。

Step7：$N_u'=C_2 \oplus R_u'$

當伺服器收到$C_2$時，他會利用$R_u'$與$C_2$先算出$N_u'$。

Step8：$V_1=h(R_u'||N_s)$

伺服器利用$R_u'$與$N_s$計算出$V_1$。

Step9：$C_3 = V_1 \oplus N_u'$

伺服器利用$V_1$與$N_u'$計算出$C_3$。

Step10：S -> $U_u$：$C_3$

伺服器將$C_3$傳送給使用者。

Step11：$V_2=h(R_u||N_s')$

當使用者收到$C_3$之後，他首先利用$R_u$與$N_s'$計算出$V_2$。

Step12：$C_3'=V_2 \oplus N_u$

使用者利用$V_2$與$N_u$計算出$C_3'$。

Step13：$C_3$ ?= $C_3'$

使用者比對$C_3$與$C_3'$，如果成功，伺服器正確無誤，進行下面的步驟，如果錯誤，便可以知道該伺服器是假冒的，因此停止下面步驟。

Step14：$C_4=h(V_2||N_u)$

使用者利用$C_3'$與$N_u$計算出$C_4$。

Step15：$U_u$->S：$C_4$

使用者將$C_4$傳送給伺服器。

Step16：$C_4'=h(V_1||N_u')$

當伺服器收到使用者傳來的$C_4$之後，利用之前的$V_1$以及$N_u'$計算出$C_4'$。

Step17：$C_4$ ?= $C_4$'

伺服器比對$C_4$與$C_4$'，如果成功，允許使用者進入，如果失敗，不允許使用者進入。

Step18：$SK=h(h(ID_i||x)||(N_s+3||N_u+4))$

當雙方驗證完畢之後，雙方會訂定出一把階段鑰匙(session key)，這一把階段鑰匙是用來加解密後續傳輸資料的鑰匙。

## 5.5 安全性分析

下面將會利用幾個已知的網路攻擊方法來檢視一下本研究所提出的驗證機制，以證明本研究所提出的驗證機制是安全的。

### 5.5.1 重送攻擊(Reply Attack)

在每次的驗證過程中，通訊的雙方都會自行隨機產生使用者的隨機亂數$N_c$以及伺服器的隨機亂數$N_s$來改變通訊的內容，因此每次通訊的資訊都是不一樣的，所以攻擊者無法在得知幾次的通訊資料之後，將這些訊息重新送往系統，進而假冒使用者入侵系統，而且在驗證的過程中只要伺服器發現到使用者無法傳送正確的認證訊息時，便會判斷該使用者是有問題的，這時，系統便會終止使用者的登入要求，所以本研究所提出的驗證機制可以有效的避免重送攻擊(Reply Attack)。

### 5.5.2 驗證表被竊攻擊(Stolen-Vertifier Attack)

在本研究所提出的驗證機制中，伺服器並沒有存放任何的使用者驗證表，所以攻擊者無法利用各種安全漏洞或是透過內部人員等攻擊方式，從伺服器上竊取到任何的使用者驗證表，因此，本研究所提出的驗證方法可以有效的避免驗證表被竊攻擊(Stolen-Verifier Attack)的攻擊。

### 5.5.3 伺服器偽裝攻擊(Server Spoofing Attack)

在本研究所提出的驗證機制中，很明顯要求在伺服器驗證使用者的身分合法之前，使用者要先知道該伺服器是否正確無誤，因此，如果攻擊者想要假冒伺服器以騙取使用者，這個攻擊者就必須先在通訊的過程中事先知道伺服器私密參數x為何，才能夠假冒伺服器來欺騙使用者進入系統或騙取使用者驗證用的資訊。如果使用者發現到有人正在假冒伺服器來進行欺騙時，使用者端的程式便會中斷整個驗證流程，所以本研究所提出的驗證機制可以有效的避免伺服器偽裝攻擊(Server Spoofing Attack)。

### 5.5.4 使用者假冒攻擊(Impersonation Attack)

在本研究所提出的驗證機制中，一個攻擊者無法利用使用者假冒攻擊(Impersonation Attack)來假冒成其他的合法使用者入侵系統，因為為了要成功利用使用者假冒攻擊去攻擊系統，這個攻擊者必須要知道使用者智慧卡中所存放的$h(ID_u||x)$，這樣攻擊者才能夠順利的製造出合法的驗證訊息，進而假冒使用者入侵系統，不過這是不可能的，因為在登入的訊息中， $h(ID_u||x)$是被伺服器所產生的隨機亂數$N_s$和使用者所產生的隨機亂數$N_u$所保護，攻擊者無法從中解出正確的$h(ID_u||x)$，而智慧卡中的$h(ID_u||x)$則是被使用者記憶的密碼所保護，因此，本研究的驗證機制可以有效的避免使用者假冒攻擊(Impersonation Attack)。

### 5.5.5 階段鑰匙的安全性(Security of session key)

一把階段鑰匙(session key)的產生來自於$h(ID_u||x)$、$N_s$以及$N_u$，這些參數值只有伺服器以及使用者知道，每當使用者與伺服器在通訊結束之後，這把鑰匙便會立即銷毀，並不會留到下次通訊時繼續使用。而當使用者重新進入系統時，整個驗證機制再驗證成功之後會另外產生一把新的階段鑰匙(session key)來加密通訊過程中的資訊，所以假設一個攻擊者在利用某種安全性問題而得知一把階段鑰匙(Session Key)時，該攻擊者也無法再利用這一把階段鑰匙(Session Key)來解密其他通訊過程中的資訊。更因為$N_s$以及$N_u$都是隨機產生的，而且都是由$h(ID||x)$所隱藏，因此攻擊者無法利用一把已知的階段鑰匙(Session key)來推算出下一把階段鑰匙(Session key)的值為何。而且$N_s$以及$N_u$是分別由伺服器以及使用者所產生，這個兩個隨機亂數的數字都非常的大，所以攻擊者很難以猜測的方式猜測出這把鑰匙的值為何，所以階段鑰匙(Session Key)是非常安全的。

## 5.6 與其他驗證機制比較

在這邊將會把本研究所提出的驗證機制與前面介紹的兩個機制做一個總比較，因為本機制與其他兩個機制都是採用隨機亂數(nonce)為主的無認證表系統，因此，在時間的同步上以及是否需要認證表上都是不需要。而就其使用者使用上來看，因為本機制採用人性化的使用方式，不希望讓使用者記憶太過長串的密碼，因此所提出的驗證機制在註冊時期中伺服器需要的單向雜湊函數加密次數都比其他的驗證機制所需要的單向雜湊函數加密次數要多一次，也因為人性化使用的原因，在使用者登入系統時，本驗證機制在使用者登入期(Login Phase)的執行效能比 Lee 等人所提出的驗證機制在使用者登入期(Login Phase)機制要多一次單向雜湊函數加密，但是還是比 Chen 等人所提出的驗證機制少一次雜湊函數加密的次數，在登入系統時伺服器所花費的加密量來說，本驗證機制所花費的加密量與 Lee 等人所提出的驗證機制是一樣的，但是比 Chen 等人的驗證機制要來的少，而且 Lee 等人的驗證機制在最後並不會產生一把階段鑰匙(session key)，而本驗證機制與 Chen 等人所

提供的驗證機制一樣,都會在通訊之後產生一把階段鑰匙(session key)來加密後續資料，除此之外，學者 Chen 等人所提供的驗證機制在安全上些許的問題，有洩漏伺服器私密鑰匙的疑慮，讓攻擊者可以利用離線猜測密碼的方式來解出伺服器的私密鑰匙 x，因此，並不是十分的安全，整個比較彙整如下表所示：

**表 5.2：與其他兩個驗證機制比較**

| | 加密量 | | | 是否需要驗證表 | 是否需要 Timestamp | 是否提供互相驗證 | 是否會遭受離線密碼攻擊 | 是否提供 Session key | 使用者使用上是否友善 |
|---|---|---|---|---|---|---|---|---|---|
| | 註冊期 | 登入系統 | | | | | | | |
| | | 使用者 | 伺服器 | | | | | | |
| 本研究機制 | 2H | 3H | 3H | No | No | Yes | No | Yes | Yes |
| Lee 等人[18] | 1H | 2H | 3H | No | No | Yes | Yes | No | No |
| Chen 等人[23] | 1H | 4H | 5H | No | No | Yes | No | Yes | No |
| H: 單向雜湊函數運算量 | | | | | | | | | |

## 5.7 結論與後續分析

本研究主要是以隨機亂數(nonce)為基礎，並利用單向雜湊函數以及 Xor 運算來建立一個以智慧卡為儲存工具的無認證表驗證機制。為了驗證這個驗證機制安全而且可行，本研究也實際將這一個機制實作於網路環境之中，以了解它是否安全，並去除掉部份不合理的地方。不過，以單向雜湊函數為基礎的驗證機制其安全性皆仰賴於單向雜湊函數的特性-抗碰撞性、不論明文大小皆能輸出固定長度、執行的速度快等等，不過，西元 2004 年 MD5 單向雜湊函數就曾經被中國山東大學的王小雲教授等學者[22]所破解，因此對於雜湊函數的選取上必須盡量選用安全度較高的單向雜湊函數，以避免驗證機制被攻擊者所破解。

# 第六章 以隨機亂數為基礎(Nonce-based)的多伺服器驗證機制

## 6.1 前言

隨著網際網路的蓬勃發展，越來越多的網路系統漸漸搬移到網路環境上，希望透過網際網路的便利性，讓遠端的使用者可以互相分享資訊，但是有些資料十分寶貴，必須只能讓特定的人才能夠存取，因此，如何驗證使用者的身分變得十分的重要。西元1981年學者Lamport[14]提出了一個如何在不安全的網路環境中驗證使用者身分的驗證機制之後，很多的使用者身份驗證機制漸漸的被提出，但是這些驗證機制在伺服器上都需要存放一個驗證表來驗證使用者，容易造成當伺服器被攻擊時，發生驗證表外洩的問題，直到西元1990年，學者Hwang、學者Chen和學者Laih[19]提出了一個需要智慧卡的單伺服器無認證表機制後，便開始有越來越多的學者提出各種不同的單伺服器無認證表驗證機制，不過這些無驗證表驗證機制大多是利用非對稱性加密的方式或是利用解離散對數的困難來保護資料傳輸的安全性，所以在計算的效能上十分的龐大。西元2000年學者Sun[8]提出了一種以單向雜湊函數為基礎的單伺服器無認證表驗證機制，這個機制僅僅使用了時間戳記(timestamp)以及雜湊函數來驗證遠端使用者的身分，大大降低了龐大的執行運算，不過這個驗證機制有許多的安全性問題以及使用上的缺點存在，因此，最近的幾年中，不斷地有學者提出利用雜湊函數的單伺服器無驗證表驗證機制。

但是，這些單伺服器無驗證表驗證機制有個嚴重的缺失，如果一個使用者要使用很多伺服器的網路服務時，這個使用者就必須要在這些伺服器上面一一註冊，因此對於一個需要很多伺服器服務的使用者來說，使用上就變得十分的麻煩，為了改善這個缺點，西元2001年學者Li等人[13]提出了一個使用類神經網路的多伺服器驗證機制，這個多伺服器驗證機制可以讓使用者僅需要註冊一次，便可以使用多台伺服器所提供的網路服務，其後有許許多多的研究分別利用不同的方式來提供多伺服器環境的使用，例如：西元2003年學者Lin等人[6]提出一種以離散對數為基礎的多伺服器驗證機制，西元2004年學者Tsaur等人[20]提出以RSA非對稱性加密法以及Lagrange interpolating polynomial為基礎的多伺服器驗證機制等等研究，可惜的是這些驗證機制在運算的花費上都十分的高，直到西元2004年學者Juang[21]提出了一個以對稱性加密演算法為基礎的多伺服器驗證機制，在這個多伺服器驗證機制裡同時提供了需要存放驗證表以及不需要存放驗證表兩種不同的驗證方式，其作法是透過一台額外的註冊中心(Register center，RC)來幫助伺服器進行驗證，使用者只需要在註冊中心(Register center，RC)上註冊一次，便可以在多台伺服器上驗證使用者的身分，這樣的驗證方式不但改善了使用者需要多次註冊的問題，同時也解決了計算效能過於龐大的問題，西元2004年學者Chang等人[2]也提出了一個改善學者Juang的驗證機制，這個驗證機制也是以對稱性加密為基礎。

本研究提出了另外一種多伺服器驗證機制，這個多伺服器驗證機制跟其他多伺服器驗證機制不同的地方是，第一、使用者僅需要在註冊中心(Registration Center，RC)上註冊過一次，便可以在多伺服器的環境中驗證使用者的身分。第二、在加密

上僅僅使用單向雜湊函數，因此在運算的效能上比起其他利用對稱性加密演算法或非對稱性加密演算法的多伺服器驗證機制要來的低。第三、這個多伺服器驗證機制是以隨機亂數(nonce)為基礎，所以並沒有時間同步的問題存在。第四、在伺服器以及註冊中心中皆不需要存放任何的使用者驗證表。第五、在每次的使用者登入時，伺服器僅僅能夠知道暫時性的使用者驗證分解參數，就算伺服器被攻擊者所入侵，它也無法得知使用者的驗證資訊。第六、在驗證流程結束之後，伺服器與使用者會建立一把階段鑰匙(session key)用來做後續資料加密之用。

## 6.2 本研究所提出的多伺服器驗證機制

這一節將會開始介紹本研究所提出的多伺服器驗證機制，整個多伺服器驗證可以分為使用者註冊期(User Registration Phase)、登入期(Login Phase)、驗證伺服器與註冊中心期(Authenticate Server and Register Center Phase)、驗證伺服器與使用者期(Authenticate Server and User Phase)等四個部分，當註冊中心同意讓某台伺服器加入時，註冊中心會以該伺服器所提供的$SID_j$來計算出伺服器所私有的$R_s= h(SID_j||y)$，$R_s$是註冊中心用來確認伺服器的身份是否合法的一個私密鑰匙，註冊中心(Registration Center，RC)會以安全通道的方式將$R_s$轉交給該伺服器，讓該伺服器可以利用$R_S$來與註冊中心進行雙向驗證工作，以確認伺服器與註冊中心彼此的身份是否合法，在開始介紹本研究所提出的驗證機制之前，首先要先定義一下本研究所使用的相關符號。

**表 6.1：符號說明表**

| 符號 | 說　　明 |
|---|---|
| h() | 單向雜湊函數 |
| x | 為使用者私秘參數，為註冊中心所擁有 |
| y | 為伺服器私秘參數，為註冊中心所擁有 |
| ⊕ | 互斥或 |
| ID | 使用者帳號 |
| PW | 使用者密碼 |
| \|\| | 參數連結 |
| N | 使用者及伺服器產生的隨機亂數 |
| X->Y:M | 表示從X傳送一個M訊息到Y |
| U，S | 分別代表使用者及伺服器 |
| RC | 註冊中心 |
| SID | 註冊中心的身分帳號 |

## 6.2.1 使用者註冊期(User Registration phase)

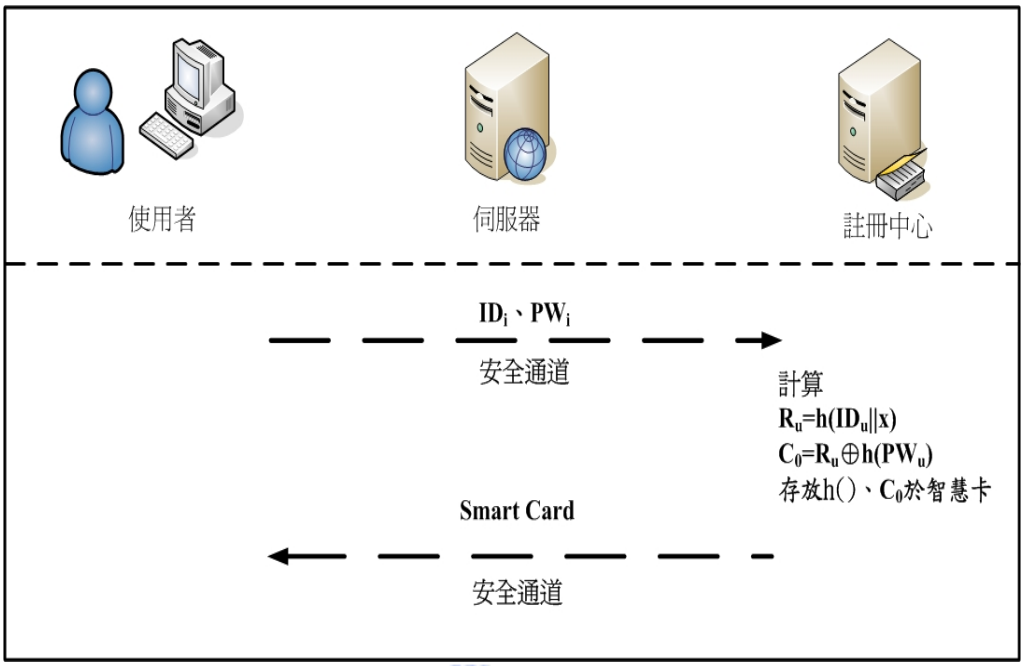


**圖 6.1：本論文所提出驗證機制的使用者註冊期**

在本研究所提的多伺服器驗證機制中，主要是透過一台額外的驗證中心(Registration Center，RC)來驗證遠端的使用者及伺服器，因此，一個使用者如果想要成為一位合法的使用者的話，這個使用者必須先將自己本身的帳號($ID_u$)和密碼($PW_u$)以安全通道的方式傳送到註冊中心進行註冊。當註冊中心願意接受該使用者時，註冊中心便會將使用者登入時所需要的資訊存放於智慧卡中，再以安全通道的方式交給使用者使用，整個註冊期的執行步驟如下所示：

Step1：$U_u$->RC：$ID_u$、$PW_u$

遠端的使用者輸入自身的帳號$ID_u$及密碼$PW_u$，並將自己的帳號和密碼以安全通道的方式傳送給註冊中心。

Step2: 註冊中心在收到使用者的申請之後，決定是否要讓該使用者成為合法使用者，如果願意接受該使用者的申請，註冊中心使用使用者帳號($ID_u$)與註冊中心的私祕參數x，去計算出$R_u= h(ID_u||x)$，並且使用$R_u$和$PW_u$去計算出$C_0=R_u \oplus h(PW_u)$。

Step3：註冊中心將h()和$C_0$存放於智慧卡中，並且將智慧卡以安全通道的方式交給使用者使用。

## 6.2.2 登入期(Login phase)

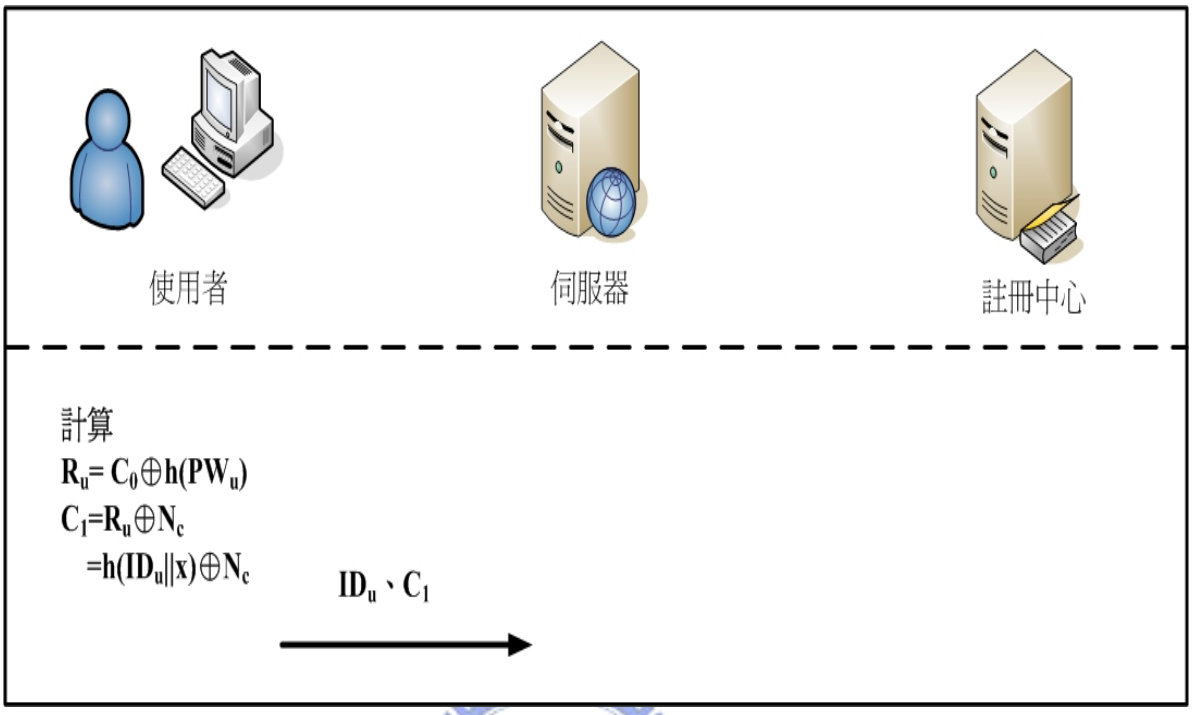


**圖 6.2：本論文所提出驗證機制的登入期**

當使用者想要登入系統的時候，使用者首先必須將自己的智慧卡卡片插入讀卡機中，並輸入自身的帳號($ID_u$)以及密碼($PW_u$)，進而登入系統。在登入時使用者會產生一個隨機的隨機亂數$N_c$來改變傳送的資訊，以避免攻擊者藉由竊聽等方式偷取通訊過程中重要的資訊，整個登入期(Login Phase)可以分為三個步驟：

Step1：$R_u = C_0 \oplus h(PW_u)$

當使用者輸入他的使用者密碼($PW_u$)之後，讀卡機利用這個使用者密碼($PW_u$)從智慧卡中計算出$R_u = C_0 \oplus h(PW_u) = h(ID_u||x)$。

Step2：$C_1 = h(ID_u||x) \oplus N_c$

使用者產生一個隨機亂數$N_c$，並且利用這個隨機亂數$N_c$以及$h(ID_u||x)$去計算$C_1 = h(ID_u||x) \oplus N_c$。

Step3：$U_u$ -> $S_j$：$ID_u$、$C_1$

使用者將帳號($ID_u$)和$C_1$送往伺服器$S_j$，讓伺服器$S_j$進行使用者身份驗證的相關工作。

### 6.2.3 驗證伺服器與註冊中心期(Authenticate Server and Registration Center phase)

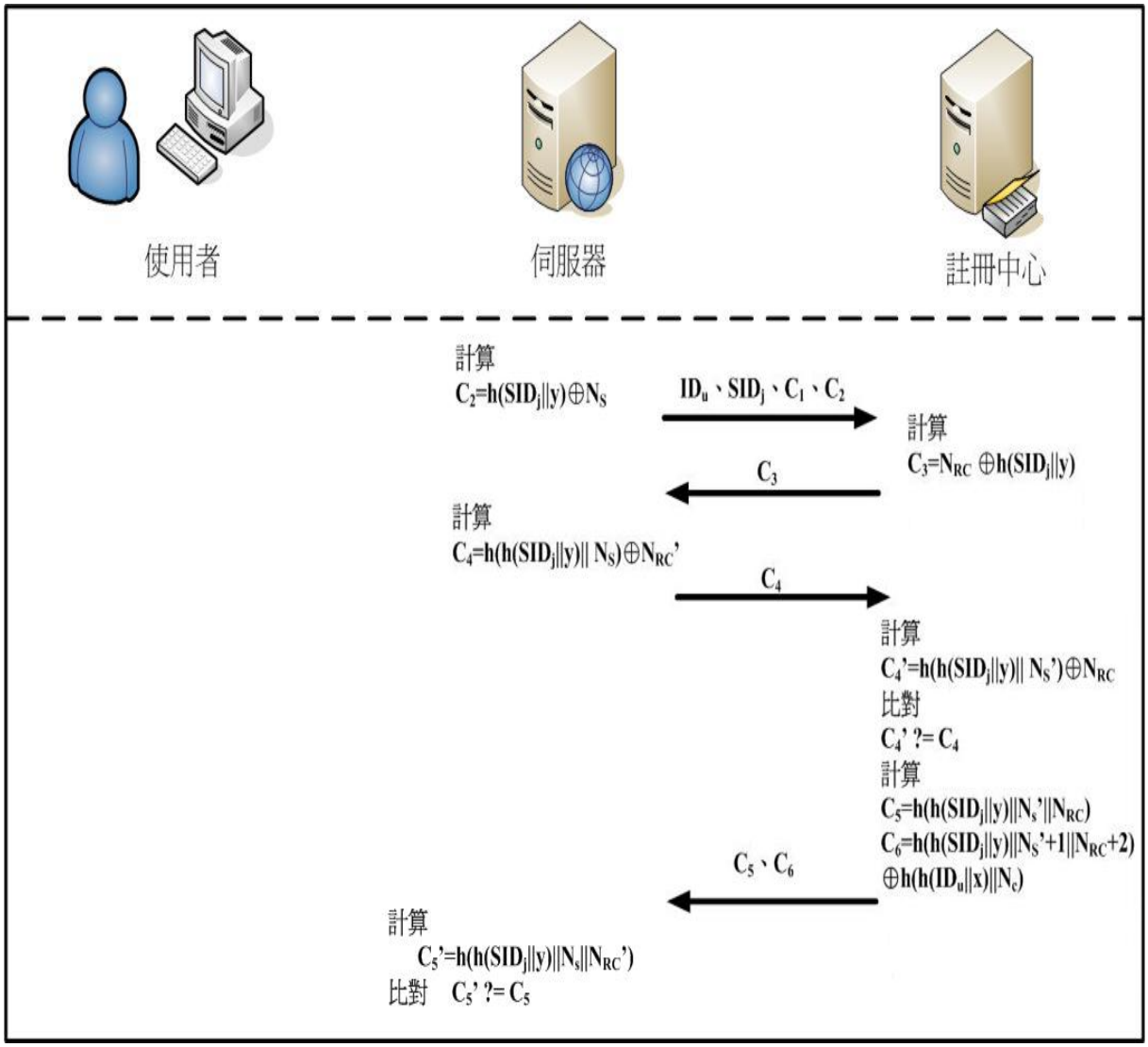


**圖 6.3：本論文所提出驗證機制之驗證伺服器與註冊中心期**

當伺服器收到使用者登入的申請之後，伺服器會產生一個伺服器的隨機亂數$N_s$來保護傳送資訊的安全，而註冊中心也會自動產生一個註冊中心的隨機亂數$N_{RC}$來保護傳送資訊的安全，因此，整個驗證過程是十分安全的。為了避免不斷的重覆驗證伺服器與註冊中心，可以在一次驗證成功之後，註冊中心與伺服器皆存放保護使用者驗證分解參數鑰匙，這時便可以省略通訊的第一步驟$C_2$的計算及傳送、第二步驟、第三步驟、第四步驟以及第五步驟的$C_5$計算及傳送(只需執行步驟一的$ID_u$、$SID_j$、$C_1$之傳送以及步驟五的$C_6$之傳送)。當然為了避免攻擊者因為竊聽過幾次伺服器與註冊中心之間的通訊，進而推敲出保護使用者驗證分解參數之鑰匙，我們可以設定註冊中心及伺服器在驗證使用者幾次身分之後，就必須要重新溝通出新的保護使用者驗證分解參數之鑰匙，以保護使用者分解參數的安全性。整個驗證流程的步驟可以分為下面幾個部份：

Step1：$S_j$ -> RC：$ID_u$、$SID_j$、$C_1$、$C_2$

當伺服器$S_j$收到使用者傳來的$ID_u$以及$C_1$，伺服器產生一個隨機亂數$N_s$，並且利用它和$h(SID_j||y)$去計算出$C_2=h(SID_j||y)\oplus N_S$，之後伺服器將$ID_u$、$SID_j$、$C_1$、$C_2$傳送到註冊中心。

Step2：RC -> $S_j$：$C_3$，where$C_3=N_{RC}\oplus h(SID_j||y)$

註冊中心產生一個隨機亂數$N_{RC}$去計算出$C_3=N_{RC}\oplus h(SID_j||y)$，並且將$C_3$送給伺服器$S_j$.

Step3：$S_j$ ->RC：$C_4$，where$C_4=h(h(SID_j||y)||N_S)\oplus N_{RC}'$

當註冊中心在收到伺服器$S_j$傳送過來的$C_3$，伺服器$S_j$使用$C_3$和$h(SID_j||y)$去計算出$N_{RC}'$，之後伺服器$S_j$使用$N_{RC}'$、$N_S$、$h(SID_j||y)$去計算出$C_4=h(h(SID_j||y)||N_S)\oplus N_{RC}'$，並且將$C_4$傳送給註冊中心。

Step4：$C_4'$ ?= $C_4$，where$C_4'=h(h(SID_j||y)||N_S')\oplus N_{RC}$

當註冊中心收到伺服器$S_j$傳來的$C_4$，註冊中心計算$C_4'=h(h(SID_j||y)||N_S')\oplus N_{RC}$並且比對$C_4'$和$C_4$是否等於，如果它們相等，代表伺服器$S_j$為合法的伺服器$S_j$無誤，如果不相等，代表伺服器有問題，停止下面的步驟。

Step5：RC -> $S_j$：$C_5$、$C_6$，where $C_5=h(h(SID_j||y)||N_s'||N_{RC})$、 $C_6=h(h(SID_j||y)||N_S'+1||N_{RC}+2)\oplus h(h(ID_u||x)||N_c)$

首先，註冊中心使用$h(SID_j||y)$與$C_2$去計算出$N_s'=C_2\oplus h(SID_j||y)$，再利用$h(SID_j||y)$、$N_S'+1$、$N_{RC}$、$h(ID_u||x)$以及$N_c$去計算$C_5=h(h(SID_j||y)||N_s'||N_{RC})$、$C_6=h(h(SID_j||y)||N_S'+1||N_{RC}+2)\oplus h(h(ID_u||x)||N_c)$，並且將$C_5$、$C_6$傳送給伺服器$S_j$進行驗證工作。這邊要注意的是$h(h(SID_j||y)||N_S'+1||N_{RC}+2)$是雙方用來保護使用者驗證分解參數之鑰匙，讓使用者驗證分解參數能安全的送達伺服器的手中。

Step6：當伺服器$S_j$收到$C_5$、$C_6$時，他先計算出$C_5'$，比對$C_5'$跟$C_5$是否一樣，如果一樣代表正確無誤，進行下面的步驟，如果不一樣，代表註冊中心有問題，不繼續下面的步驟。

### 6.2.4 驗證伺服器與使用者期(Authenticate Server and User phase)

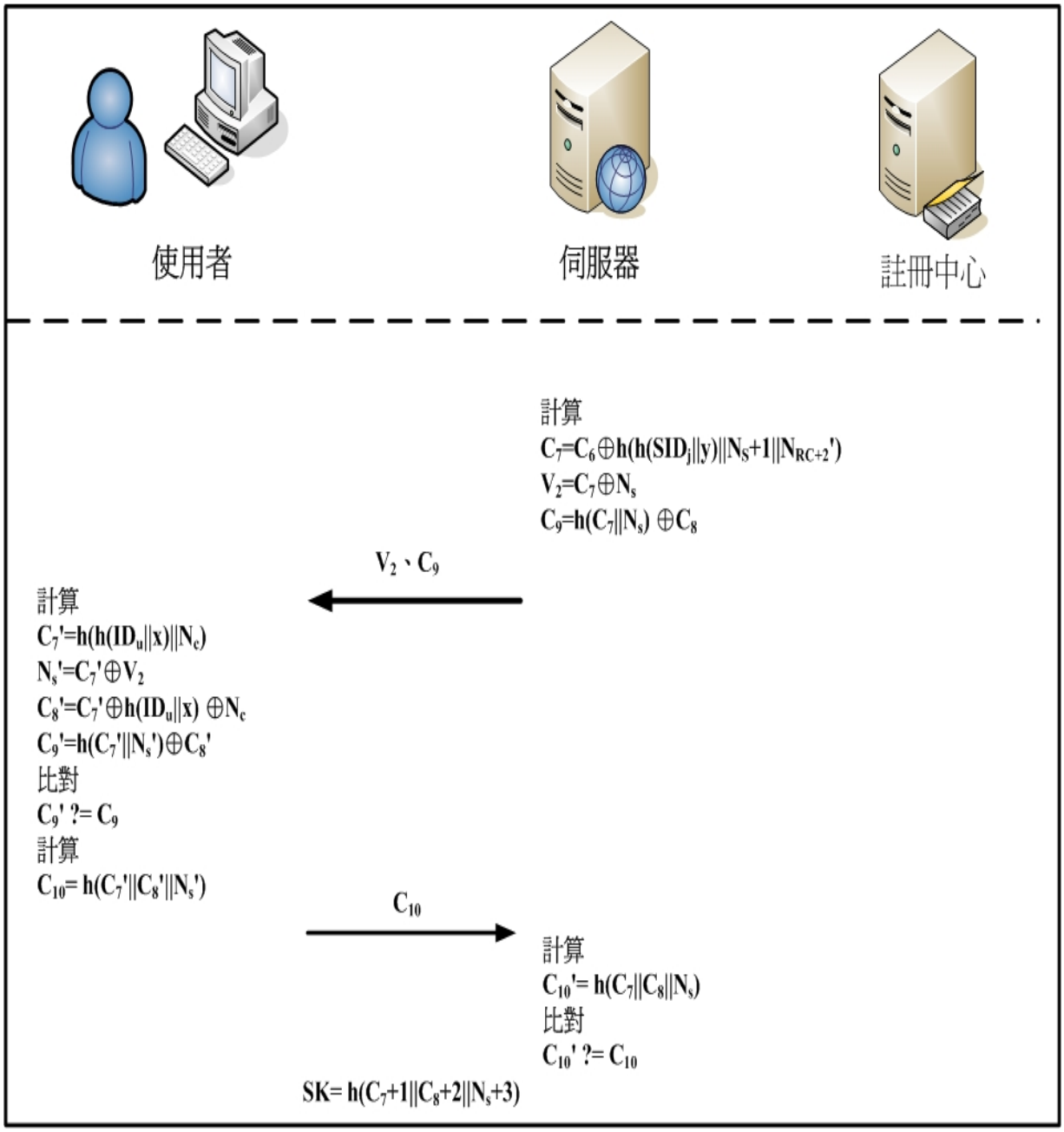


**圖 6.4：本論文所提出驗證機制的驗證伺服器與使用者期**

為了不讓伺服器知道使用者的$h(ID_u||x)$，我們以動態的方式讓註冊中心(Regisrtation Center，RC)提供一個使用者驗證分解參數$h(h(ID_u||x)||N_c)$，當伺服器拿到這個使用者驗證分解參數時，便可以利用它去驗證遠端的使用者身份。這個使用者驗證分解參數會因為使用者的隨機亂數$N_c$而不斷的改變，所以即使攻擊者拿到這個驗證分解參數，攻擊者也僅僅能夠使用在這次的通訊驗證，而無法利用它來做下次驗證通訊之用，整個驗證過程如下所示：

Step1：$S_j$ -> $U_u$：$V_2$、$C_9$，$V_2=C_7\oplus N_s$、$C_9=h(C_7||N_s)\oplus C_8$

當伺服器$S_j$確認註冊中心合法之後，它首先必須要先計算出使用者驗證參數$C_7$，所以它計算$C_7=C_6\oplus h(h(SID_j||y)||N_S+1||N_{RC+2}')=h(h(ID_u||x)||N_c)$，$C_7$便是我們的使用者驗證參數，隨即利用$C_7$去計算$C_8=C_7\oplus h(ID_u||x)\oplus N_c$，然後伺服器$S_j$利用$N_s$、$C_7$以及$C_8$計算出$V_2=C_7\oplus N_s$、$C_9$ $C_9=h(C_7||N_s)\oplus C_8$。

Step2：$C_9'$ ?= $C_9$

當使用者讀卡機收到$V_2$、$C_9$時，他利用$h(ID_u||x)$以及$N_c$計算出使用者驗證分解參數$C_7'=h(h(ID_u||x)||N_c)$，然後計算出伺服器$S_j$的隨機亂$N_s'=C_7'\oplus V_2$以及$C_8'=C_7'\oplus h(ID_u||x)\oplus N_c$，再利用$C_7'$與$C_8'$去計算$C_9'=h(C_7'||N_s')\oplus C_8'$，比對$C_9$與$C_9'$，如果相等，代表伺服器$S_j$無誤，進行下面的步驟，如果不是，停止下面的步驟。

Step3：$U_u$ -> $S_j$：$C_{10}$，$C_{10}= h(C_7'||C_8'||N_s')$

計算出$C_{10}$並將它送到伺服器$S_j$以進行使用者身份驗證工作。

Step4：$C_{10}'$ ?= $C_{10}$

計算出$C_{10}'= h(C_7||C_8||N_s)$，如果不相等，便停止下面步驟，如果相等，代表使用者身份無誤，雙方定義出一把階段鑰匙(session key) SK= $h(C_7+1||C_8+2||N_s+3)$，後續資料便會利用這一把階段鑰匙(Session key)做加解密的工作，以保護後續資料的安全性。

## 6.3 安全性分析

下面將會以幾種常見的攻擊手法去分析這個多伺服器驗證機制是否安全。

### 6.3.1 重送攻擊(Replay attack)

我們的方法可以有效的避免重送攻擊(Reply attack)，因為在每次使用者的連線中，使用者、伺服器、註冊中心皆會自己產生隨機亂數(random nonce)，因為這些隨機亂數(nonce)每次連線皆不一樣，攻擊者無法藉由竊聽一次使用者、伺服器或是註冊中心之間的連線認證資訊之後，將這個連線認證資訊以重送的方式來進入我們的系統之中。

### 6.3.2 驗證表被竊攻擊(Stolen-verifier attack)

在這個驗證機制中，伺服器以及認證中心皆沒有存放任何的使用者驗證表，所以攻擊者無法竊取到任何的使用者驗證表，因此，本研究的驗證機制可以有效的避免驗證表被竊攻擊(stolen-verifier attack)的攻擊。

### 6.3.3 伺服器偽裝攻擊(Server spoofing Attack)

在這個多伺服器驗證機制中，如果攻擊者想要偽裝成一台伺服器去欺騙註冊中心來騙取使用者驗證分解參數是不可能的，因為每台伺服器都必須要有註冊中心所提供的$h(SID_j||y)$，才能通過註冊中心的驗證。如果要直接偽裝成一台伺服器去欺騙使用者也是不可行的，因為既然它無法通過註冊中心的驗證，所以也就無法得到註冊中心提供的使用者驗證分解參數，況且，這個使用者驗證分解參數會隨著使用者每次登入時所產生的隨機亂數$N_u$的不同而有所改變，因此，假設攻擊者偷取到某次的使用者驗證分解參數時，攻擊者也無法使用這把使用者驗證分解參數於下一次的驗證通訊。

### 6.3.4 階段鑰匙的安全性(Security of session key)

當使用者在通過驗證之後，它與伺服器之間所產生的階段鑰匙(session key)是十分的安全的。因為這把階段鑰匙(session key)的決定主要是利用三個參數，分別是$h(ID_u||x)$、隨機亂數$N_c$、隨機亂數$N_s$所組合而成，其中隨機亂數$N_c$、隨機亂數$N_s$在每次連線時都是獨立而且不一樣的，因此，就算攻擊者透過某種攻擊法得知上一把階段鑰匙(session key)的值為何，也無法利用上一把已知的階段鑰匙(session key)來猜測出下一把階段鑰匙(session key)或是其他把的階段鑰匙(Session Key)的值為何。

### 6.3.5 註冊中心假冒攻擊(Register Center spoofing Attack)

如果攻擊者想要偽裝成註冊中心是不可能的，因為每台伺服器上皆有一個$h(SID_j||y)$，這個$h(SID_j||y)$主要是用來驗證註冊中心是否合法無誤，所以如果有一個攻擊者想要假冒註冊中心，這個攻擊者將會被註冊中心所發現，而無法順利得到使用者驗證參數，而無法進行使用者與伺服器之間的驗證工作。此外，每台伺服器的$h(SID_j||y)$皆不一樣，如果一個攻擊者在得知一台伺服器所擁有的$h(SID_j||y)$時，攻擊者將無法利用這把已知的$h(SID_j||y)$來推測出其他伺服器所擁有的$h(SID_j||y)$，進而假冒該伺服器與使用者進行驗證。

### 6.3.6 使用者驗證分解參數的安全性

在本研究的驗證過程中，為了有效避免一台伺服器因為攻擊者所入侵，導致使用者的驗證資訊被竊，因此，在驗證的過程中註冊中心僅提供一把暫時性的使用者驗證分解參數來讓伺服器與使用者互相驗證彼此的身分。當使用者想要登入時，註冊中心會負責將伺服器所轉送過來的使用者登入訊息$\{ID_u、h(ID_u||x)\oplus N_c\}$的$h(ID_u||x)\oplus N_c$分解開來，變成$ID_u$、$h(ID_u||x)$、$N_c$三個變數，再將$h(ID_u||x)$、$N_c$經由一

次的雜湊函數加密，變成$h(h(ID_u||x)||N_c)$，此時，這個$h(h(ID_u||x)||N_c)$便是暫時性的使用者驗證分解參數，因為這一個使用者驗證分解參數會隨著使用者每次隨機產生的隨機亂數$N_c$而有所不同，因此，攻擊者就算得知某次的使用者驗證參數$h(h(ID_u||x)||N_c)$，也無法利用它來假冒遠端的使用者。

## 6.4 結論與後續建議

如何正確的驗證使用者的身分一直是網路安全中一個重要的議題，本章節的研究中，提出了一種新的多伺服器驗證機制，這個驗證機制跟其他的多伺服器驗證機制不一樣的是，它是以隨機亂數(nonce)為基礎，而且運算上僅僅使用單向雜湊函數以及 Xor 運算，因此可以廣泛的應用在分散式的網路環境之中，除了這個優點外，本研究所提出的驗證機制在計算上僅僅使用單向雜湊函數與 Xor 運算，運算量十分的小，所以可以有效避免伺服器與使用者端設備執行效率的問題，透過本機制的研究，將來可以應用在僅具有低運算能力的設備上，充分發揮它的功用。

# 第七章 研究結論與後續建議

## 7.1 研究結論

由於網際網路的蓬勃發展，造成有越來越多的系統轉移到網路上，希望藉由網路的便利性來讓使用者互相交換重要的資訊。雖然網路十分的方便，但是，它卻不是沒有缺點的，因為網路是一個開放的環境，如果沒有安全的保護機制，將會導致重要的資料被攻擊者所竊取或是修改，因此，如何保護重要的資料變成了一個十分重要的研究議題。使用者身份驗證是網路系統的最前線防禦措施，它可以限定系統內部重要的資料只能被合法的使用者所存取，因此，它十分的基本而且重要，不論是在電子商務、網路安全、網路通訊等研究領域上都可以看到它的影子。

在本論文中，我們針對了各種不同的智慧卡驗證機制進行分析以及比較，其用意在於希望能夠藉此提升網路系統的安全性，避免系統因為身分驗證過程中的安全性問題而導致系統安全性瓦解，不過，雖然本論文針對了不同的驗證機制找出其安全性問題，並提出了改善的驗證機制，但是並不代表這些改善的驗證機制就一定是安全無虞的，說不定，未來會有其他的人找到本論文所提出改善的驗證機制之安全性問題。此外，本論文中所提到的驗證機制都是以單向雜湊函數來保障驗證資訊傳輸的安全性，因此，如何選用安全的單向雜湊函數便變得十分重要，因為如果選用了不安全的單向雜湊函數，將容易導致整個驗證機制之安全性的也隨之瓦解。

## 7.2 後續建議

在這邊針對本論文的內容給予一些後續的建議，第一、本論文除了第六章提供了一個多伺服器的無驗證表驗證機制外，大部分都是著墨於單一伺服器的驗證機制，未來可以建議可以朝向擴充不同的驗證機制，使其能夠在多伺服器環境下也能驗證遠端的使用者身份。第二、本論文中的內容大部分侷限在使用者的身分驗證，未來可以結合各種不同存取控制來管理系統使用者的存取權限，讓伺服器可以更有效管理伺服器內部重要的資訊。

# 參 考 文 獻


[1] A. Shimizu, "A Dynamic Password Authentication Method by One-way Function", IEICE Transactions on Consumer Electronics, J73-DI, 7, pp. 630-636, 1990.

[2] C.C. Chang, J.S. Lee, "An efficient and secure multi-server password authentication scheme using smart cards", International Conference on Cyberworlds, pp. 417-422, Nov. 2004.

[3] C.C. Chang, S.J. Hwang, "Some forgery attack on a remote user authentication scheme using smart cards", Informatics, 14, 3, pp. 189-294, 2004.

[4] C.H. Lin, Y.Y. Lai, "A flexible biometrics remote user authentication scheme", Computer Standards & Interfaces, 27, 1, pp. 19-23, 2004.

[5] C.K. Chan, L.M. Cheng, "Cryptanalysis of a remote user authentication scheme using smart cards", IEEE Transaction on Consumer Electronic, 46, pp. 992-993, 2000.

[6] C.L. Lin, T.H. Hwang, "A password authentication scheme with secure password updating", Computers and Security, 22, 1, pp. 68-72, 2003.

[7] E.Z.F. Liu, J.L. Tsai, "An efficient nonce-based remote authentication scheme using smart cards with key agreement", IADIS International Conference of WWW/Internet, Oct. 2006.

[8] H.M. Sun, "An efficient remote use authentication scheme using smart cards", IEEE Transactions on Consumer Electronics, 46, 4, pp. 958-961, Nov. 2000.

[9] I.C. Lin, M.S. Hwang, and L.H. Li, "A new remote user authentication scheme for multi-server architecture", Future Generation Computer System, 19, pp.13-22, Jan. 2003.

[10] J.J. Hwang, T.C. Yeh, "Improvement on Peyravian–Zunic's password authentication schemes", IEICE Transactions on Communications, E85-B, 4, pp. 823–825, April 2002.

[11] J.J. Shen, C.W. Lin, M.S. Hwang, "A modified remote user authentication scheme using smart cards", IEEE Transactions on Consumer Electronic, 49, 2, pp. 414-416, May 2003.

[12] J.K. Lee, S.R. Ryu, and K.Y. Yoo, "Fingerprint-based remote user authentication scheme using smart cards", Electronics Letter, 38, 12, pp. 554-555, 2002.

[13] L.H. Li, I.C. Lin, and M.S. Hwang, "A remote password authentication scheme for multi-server architecture using neural networks", IEEE Transactions on Neural Network, 12, 6, pp. 1498-1504, Nov. 2001.

[14] L. Lamport, "Password authentication with insecure communication", Communications of the ACM, 24, 11, pp. 770-772, Nov. 1981.

[15] M. K. Khan, J. Zhang, "Improving the security of a flexible biometrics remote user authentication scheme", Computer Standards & Interface, 29, 1, pp. 82-85, 2007.

[16] M. Peyravian, N. Zunic, "Methods for protecting password transmission", Computers & Security, 19, 5, pp. 466-469, 2000.

[17] M.S. Hwang, L.H. Li, "A new remote user authentication scheme using smart cards", IEEE Transaction on Consumer Electronics, 46, 1, pp. 28-30, 2000.

[18] S.W. Lee, H.S. Kim and K.Y. Yoo, "Efficient Nonce-based Remote User

Authentication Scheme Using Smart Cards", Applied Mathematics and Computation, 167, 1, pp. 355-361, 2005.

[19] T. Hwang, Y. Chen, and C.S. Laih, "Non-interactive password authentication without password tables", IEEE Region 10 Conference on Computer and Communication System, 1, pp. 429-431, Sept. 1990.

[20] W.J. Tsaur, C.C. Wu, and W.B. Lee, "A smart card-based remote scheme for password authentication in multi-server Internet services", Computer Standard & Interfaces, 27, pp. 39-51, 2004.

[21] W.S. Juang, "Efficient multi-server password authenticated key agreement using smart cards", IEEE Transactions on Consumer Electronics, 50, 1, pp. 251-255, Nov. 2004.

[22] X.Y. Wang, H.G. Yu, "How to Break MD5 and Other Hash Functions", Eurocrypt, pp.19-35, 2005.

[23] Y.C. Chen, L.Y. Yeh, "An Efficient Nonce-based Authentication Scheme with Key Agreement", Applied Mathematics and Computation, 169, 2, pp. 982-993, 2005.

[24] Y.F. Chang, C.C. Chang, "Authentication schemes with no verification table", Applied Mathematics and computation, 167, 2, pp. 820-832, 2005.

[25] 工業技術研究院，我國PKI互通管理及推動計畫—網際網路PKI技術及應用研究報告，存取於2006年5月11日，http://www.pki-pma.org.tw/learn/download/1/2005-02-02-01.rar，2003。

[26] 王旭正、柯宏叡，資訊與網路安全-秘密通訊與數位鑑識新技法，博碩文化股份有限公司，台北，2006。

[27] 北方網(2005)，世界震驚 美國擔心 王小雲破全球兩大密碼算法，存取於2006年五月，http://news.big5.enorth.com.cn/system/2005/03/25/000991494.shtml。

[28] 卡威科技(2005)，IC智慧卡(Smart Card)與電子商務，存取於2006年4月10日，http://www.cardweb.com.tw/304ICS/ICCardInfo/ic_EC.htm。

[29] 李尚宸，「使用通行碼之數位簽章協定」，國立交通大學資訊科學研究所，碩士論文，2003。

[30] 邱玉忠，「應用智慧卡之使用者確認機制之研究」，國立南台科技大學資訊管理研究所，碩士論文，2005。

[31] 周伯錕，「利用智慧卡之遠端身份認證之研究」，國立中興大學資訊科學研究所，碩士論文，2003。

[32] 邱瑩青，RFID實踐-非接觸式智慧卡系統開發，學貫出版社，台北，2005。

[33] 陳彥學，資訊安全理論與實務，文魁資訊股份有限公司，台北，2000。

[34] 曾紹崟，「攻擊智慧卡技術鳥瞰」，電腦與通訊，第九十五期，49-54頁，2001年3月。

[35] 張真誠、林祝興，資訊安全技術與應用，全華科技圖書股份有限公司，台北，2006。

[36] 楊佑寧，「有限信任讀卡機下安全服務機制」，國立暨南大學資訊管理研究所，碩士論文，2006。

[37] 楊中皇，網路安全理論與實務，金禾資訊股份有限公司，台北，2006。

[38] 鄭永祥、譚海林，「RFID在軍事後勤應用之研究(上)」，存取於2006年6月10日，http://www.mnd.gov.tw/publication/subject.aspx?TopicID=872，中華民國國防部海軍學術月刊，第39卷，第07期，2005。

[39] 謝續平，交通大學網路安全授課資料，存取於2006年5月20日，http://dsns.csie.nctu.edu.tw/course/netsec/2004fall/，2004。