一個以隨機亂數為基礎之多伺服器無驗證表驗證機制

蔡佳倫
交通大學
crousekimo@yahoo.com.tw

李榮耀
交通大學
jlee@math.nctu.edu.tw

摘要

西元 2000 年學者 Sun[2]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，從此以後，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，他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，本研究提出了一個多伺服器的無驗證表驗證方式，在這個驗證方式中，伺服器與註冊中心都不需要存放任何的使用者驗證表，而且僅僅使用單向雜湊函數(hash function)來保障資訊傳輸的安全性。

關鍵詞：驗證、多伺服器、智慧卡。

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[5]提出了一個如何在不安全的網路環境中驗證使用者的身分之後，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之後，容易發生驗證表外洩的問題，直到西元 1990 年，學者 Hwang 、Chen 和 Laih[6]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[2]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，他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[4]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[3]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[7]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[8]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[1]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，本研究提出的多伺服器驗證機制是以雜湊函數為基礎，而且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，而不需要進行多次的註冊。而且本研究所提出的多伺服器驗證機制在佳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，此外，本研究所提出的多伺服器驗證機制使用了隨機亂數(nonce)，所以並沒有時間同步的問題存在，非常適合分散式的網路環境中使用。

2. 本研究所提出的多伺服器驗證方法

這一節我們開始介紹我們的驗證流程，整個驗

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 SIDj 來計算出伺服器所私有的 Rs= h(SIDj||y)，Rs 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 Rs 轉交給該伺服器，讓該伺服器可以利用 RS 來進行我們的驗證機制，在開始介紹之前，我們先定義一下本研究所使用的相關符號。

表 1 本研究所需之相關符號

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, PW	使用者帳號及密碼
||	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	伺服器的身分帳號

2.1 使用者註冊期

在我們的驗證機制中，主要是透過一台驗證中心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，這個使用者必須要先進行註冊，因此，一開始使用者必須要以安全通道將自己本身的帳號(ID_u)和密碼(PW_u)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，它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與註冊中心的私秘參數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。

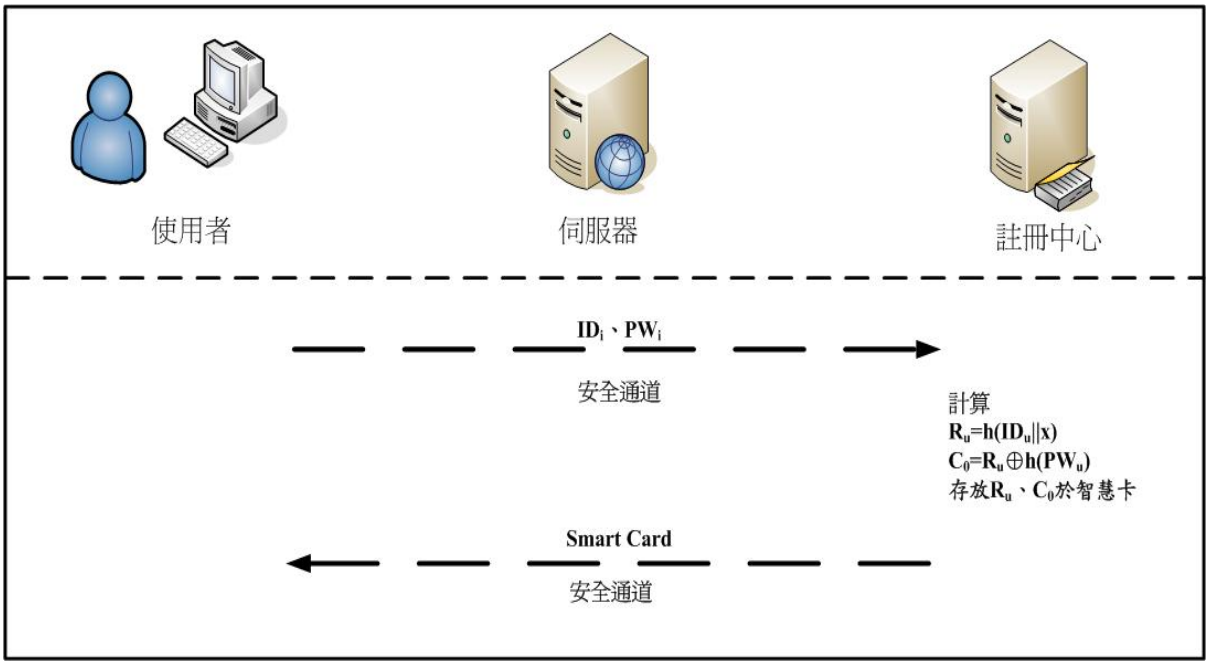

圖 1 註冊期

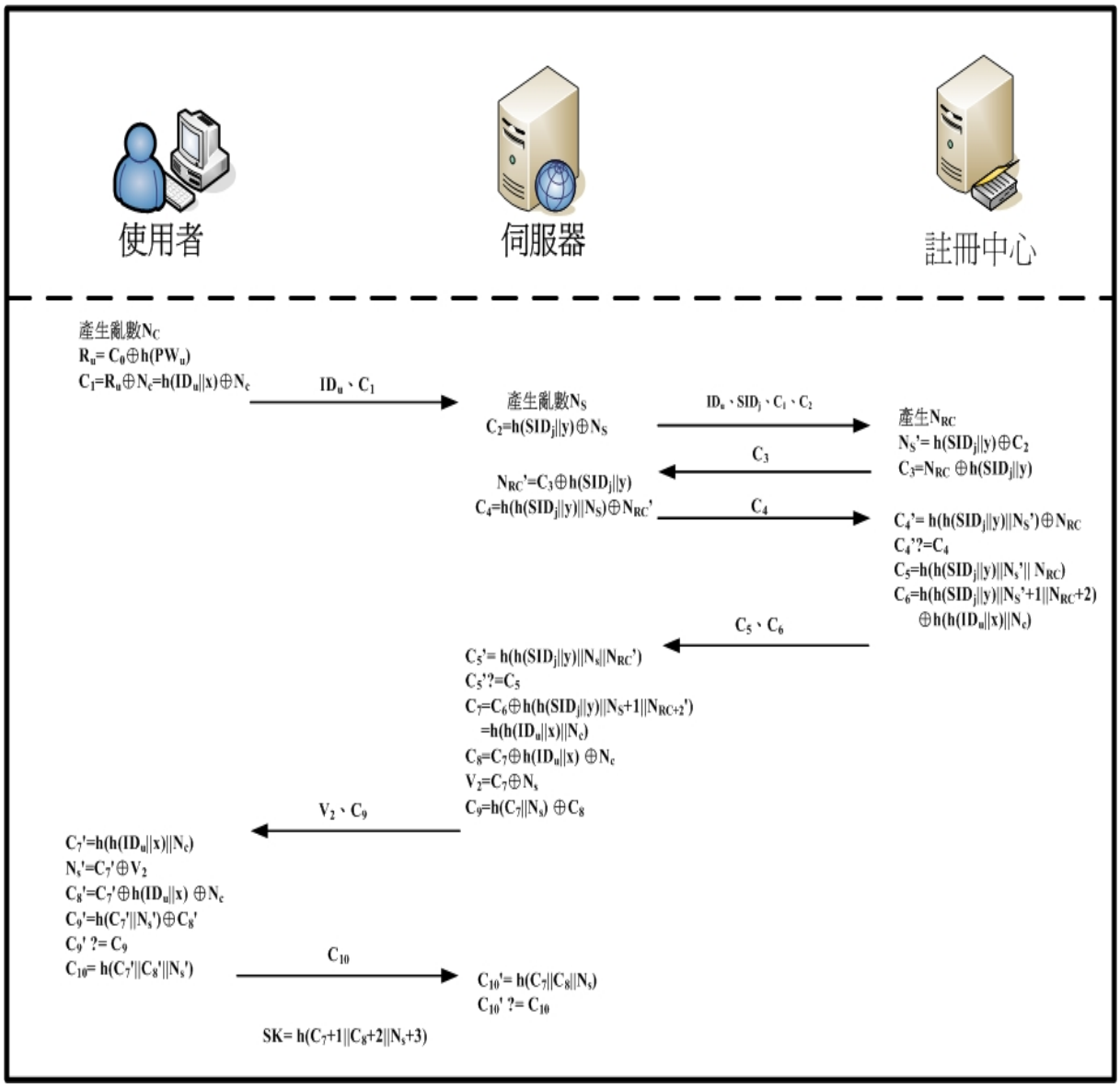

圖 2 本研究所提出的多伺服器無驗證表驗證機制

2.2 登入期

當使用者想要登入系統的時候，他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(nonce)來改變傳送的資訊，以避免攻擊者的借由竊聽的方式偷取卡片中重要的資訊，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進行使用者身份的驗證工作。

2.3 伺服器與註冊中心驗證期

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(nonce)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(nonce)來保護傳送資訊的安全，因此，整個驗證過程都是十分的安全。為了避免不斷的重覆驗證伺服器與註冊中心，我們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、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鑰匙，我們可以設定註冊中心及伺服器在驗證使用者幾次之後，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，whereC_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。

2.4 伺服器與使用者驗證期

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提供另外一個使用者驗證分解參數$h(h(ID_u||x)||N_c)$，這個使用者驗證分解參數是會因為使用者的隨機亂數N_u而不斷的改變。當伺服器拿到這一個使用者驗證分解參數後，便可以拿來與使用者互相驗證對方身分：

Step1：S_j -> U_u：V_2、C_9，where$V_2=C_7\oplus N_s$ and $C_9=h(C_7||N_s)\oplus C_8$
當伺服器S_j確認註冊中心合法之後，他首先必須要先計算出使用者驗證參數$C_7=C_6\oplus h(h(SID_j||y)||N_S+1||N_{RC+2}')=h(h(ID_u||x)||N_c)$，$C_7$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$V_2$、$C_9$。

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}，where $C_{10}= h(C_7'||C_8'||N_s')$
計算出$C_{10}= h(C_7'||C_8'||N_s')$，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$C_{10}'= h(C_7||C_8||N_s)$，並比較$C_{10}$跟$C_{10}$'是否相等，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) $SK= h(C_7+1||C_8+2||N_s+3)$，後續資料利用這一把階段鑰匙(Session key)加密，以保護資料的安全性。

3 安全性分析

本驗證機制主要是以雜湊函數為基礎，伺服器以及註冊中心中皆不存放任何的註冊表，當使用者想要使用伺服器所提供的服務時，伺服器會先轉送使用者的登入要求，並且將這個登入要求轉送給註冊中心，並且與註冊中心互相驗證雙方的身分，在確認對方身分無誤之後，註冊中心會送出使用者的驗證分解參數給伺服器去與使用者互相做身份驗證工作，在驗證完之後，會建立一把階段鑰匙(session key)用來做後續資料加密之用，以下我們就幾種常見的攻擊手法去分析這個機制的安全性：

3.1 重送攻擊(Replay attack)

我們的方法可以有效的避免重送攻擊(Replay

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。

3.2 驗證表被竊攻擊(Stolen-verifier attack)

在我們的方法中，伺服器以及認證中心皆沒有存放任何的驗證表，所以攻擊者無法竊取到任何的驗證表，因此，我們的驗證方法可以有效的避免掉驗證表被竊攻擊(stolen-verifier attack)的攻擊。

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以騙取使用者驗證分解參數是不可能的，因為每台伺服器都有註冊中心所提供的$h(SID_j||y)$，註冊中心可以利用這個$h(SID_j||y)$來驗證伺服器是否合法，所以一個攻擊者想要偽裝成一台合法的伺服器是不可能的，所以我們提供的多伺服器驗證機制可以有效的抵抗伺服器偽裝攻擊。

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$h(ID_u||x)$、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。

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，他將會被伺服器所發現，而終止整個連線通訊，因此，本驗證機制可以有效的避免註冊中心假冒攻擊的安全性問題存在。

3.6 使用者的驗證參數被入侵的伺服器知悉

在我們的驗證過程中，為了避免一台伺服器因為被入侵後，透過觀察來得知如何驗證使用者登入的鑰匙，因此我們透過另外一種方式來避免這個情況發生，這個方法是註冊中心會負責將伺服器所傳來的使用者登入時所傳送的ID_u、$h(ID_u||x) \oplus N_c$分解開來，變成ID_u、$h(ID_u||x)$、N_c三個變數，再將這三個變數經由一次的雜湊函數加密，變成$h(h(ID_u||x)||N_c)$，並且以安全的方式交給伺服器，伺服器則利用這一個$h(h(ID_u||x)||N_c)$來與使用者之間做為互相驗證用的參數，因為這一個參數會隨著使用者每次產生的隨機亂數N_c而有所不同，因此，攻擊者就算得知某次的參數，也無法利用來假冒使用者。

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，在我們的研究中，我們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方法不一樣的一點是，它是以隨機亂數(nonce)為基礎，因此，可以廣泛的應用在分散式的網路環境之中，而且在加密的方法上僅僅使用單向雜湊函數，所以可以有效避免伺服器執行效率的問題，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，充分發揮他的功用。

參考文獻

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

[2] H. M. Sun, “An efficient remote use authentication scheme using smart cards,” IEEE Transactions on Consumer Electronics, Vol. 46, No. 4, pp. 958-961, Nov. 2000.

[3] I. C. Lin, M. S. Hwang, and L. H. Li, “A new remote user authentication scheme for multi-server architecture,” Future Generation Computer System, Vol. 19, No. 1, pp. 13-22, Jan. 2003.

[4] 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, Vol. 12, No. 6, pp. 1498-1504, Nov. 2001.

[5] L. Lamport, “Password authentication with insecure communication,” Communications of the ACM, Vol. 24, No. 11, pp. 770-772, Nov. 1981.

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

[7] 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, Vol. 27, No. 1, pp. 39-51, 2004.

[8] W. S. Juang, “Efficient multi-server password authenticated key agreement using smart cards,” IEEE Transactions on Consumer Electronics, Vol. 50, No. 1, pp. 251-255, Nov. 2004.