網路金鑰交換協定之分析與設計
－針對阻絕服務式攻擊

蘇民揚　張家峯
銘傳大學資訊工程研究所
minysu@mcu.edu.tw

摘要

IPSec(IP 安全協定)對封包進行加密與認證的處理，保障了資料在公眾網路傳送時免於被窺視、修改。然而，在套用 IPSec 之前，通訊的雙方必須先擁有共享的鑰匙，除了最基本的預先共享金鑰外，透過網路金鑰交換(Internet Key Exchange, IKE)協定來進行金鑰與安全關聯協商，可以讓雙方共享的金鑰更具變化，藉以增強資料在網路傳送時的安全度。IKE 是由 IETF 所制定，但是由於協定太過複雜並且容易遭受阻絕服務式(Denial-of-Service, DoS)攻擊，因此有許多學者紛紛提出不同的解決方案要來取代原本的 IKE。在本論文裡，我們分析了現有的 IKE 協定，針對其所不足的地方加以改進，提出了一個更安全且更具效率的網路金鑰交換協定，並透過實驗來驗證，我們的協定在抵擋阻絕服務式攻擊，以及在協商金鑰與安全關聯的效率上有更好的表現。

關鍵詞：網路安全，IP 安全協定，網路金鑰交換協定，阻絕服務式攻擊，身份認證

1. 前言

隨著網路科技的崛起，越來越多的企業與個人利用網路從事著許多的活動，網路上所傳遞的訊息其安全性日益重要，一些機密的資料如不想隨便的暴露在公眾網路裡，IPSec 提供了我們一個不錯的解決方案，它將資料封包以加密的形式在網路中傳送，除溝通的雙方外，其他人是無法得知該訊息的內容。IPSec 可對封包進行認證的處理，如資料在傳送過程中被修改，也可以輕易的檢查出來，確保了資料的機密性及完整性。然而通訊雙方想要使用 IPSec 來增加傳送資料的安全性之前，雙方必須有共通的金鑰與安全關聯，那麼彼此的溝通才具有意義，除最基本的預先設定好金鑰與安全關聯外，IKE[4]提供我們一個更彈性且安全的方案，它可以依據我們安全的需求來選擇我們所要使用的加解密與認證演算法，以及多久的時間間隔可以去協商更換一個新的金鑰與安全關聯，讓我們加密的資料被破解的難度增加。

1976 年，Diffie 與 Hellman[3]提出了一個安全的金鑰協商協定，但是並不具備互相認證對方的性質，後續才有許多研究將認證對方身份的性質加入金鑰交換的協定中，但是時至今日卻面臨 DoS 攻擊的威脅。Hirose[1]的金鑰交換協定是最早被提出來可同時抵擋記憶體與 CPU DoS 攻擊的一個金鑰交換協定，而後續的研究者 Haddad[2]才將 Hirose[1]的金鑰交換協定修改成為可適用於 IPSec 使用的 IKE 協定，並改進原協定的不足，增強了抵擋 DoS 攻擊的能力。

本論文架構如下，第二節介紹現有的 IKE 抵擋 DoS 攻擊的作法，第三節是本論文所提出的網路金鑰交換協定之介紹與分析，第四節是針對本論文的協定所作的實驗，第五節則對本論文作一個總結。

2. IKEs

IKEv1[4]由 IETF 所制定來協商 IPSec 所需之金鑰與安全關聯，但是協定 IKE 協定太過複雜，有四種不同認證對方的方法[4]與三種不同的模式來交換產生金鑰的資訊[4]，並且容易遭受 DoS 攻擊[10][11][12]，包含記憶體與 CPU 的 DoS 攻擊，因此有許多學者針對 IKE 提出了許多的改進[2][7][9][10][11]，修改了原本的弱點，增強了 IKE 的安全性並簡化了原本的協定的複雜度。

IKE-SIGMA[9]在它認為有攻擊之虞時，會發送出一個 routability cookie 給發起者，發起者必須將該 cookie 回傳給回應者，來證明發起者並非使用偽造 IP 的封包，讓回應者免於遭受記憶體的 DoS 攻擊。IKEv2[7]跟 IKE-SIGMA 有類似的機制，在回應者發現有大量半開的連線時，就會由通知承載來挾帶 cookie 給發起者，發起者一樣要將該訊息回傳給回應者來證明並非使用偽造的 IP。

System Parameters
p: a large prime
q: a large prime that can divide p-1, i.e., $q \mid p$-1
g: a generator in GF(p) of order q, i.e, $g^{q} \bmod p = 1$

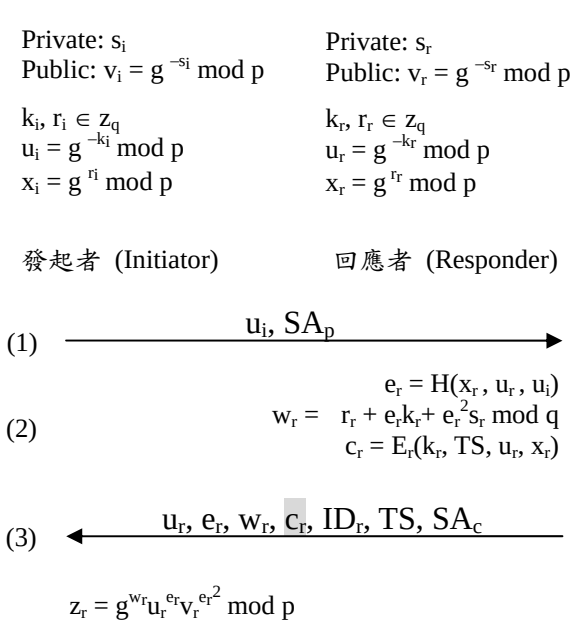

圖 1 Haddad's 網路金鑰交換協定(第一階段協商)

JFK[10]使用 keyed hash 來產生訊息認證碼，發起者須將包含該訊息認證碼的訊息回傳給回應者來證明並非使用偽造的 IP，另外會將所收到的第三個訊息及所回應的第四個訊息，存放到快取裡，目的是如果將來收到重複的第三個訊息封包時，直接回應相對的第四個訊息，無須再付出計算會議金鑰的代價，以避免 CPU 的 DoS 攻擊。Arcanum[11]利用一個只有回應者知道的數值來與其它的資訊產生一個 cookie 值，同樣可避免偽造 IP 的 DoS 攻擊；另外使用 stack 存放預先計算的 Diffie-Hellman 公開參數與私密鑰匙對，可省去在收到發起者的連線請求時所需的線上指數運算，避免 CPU 的 DoS 攻擊。

Haddad[2]修改 Hirose[1]的金鑰交換協定，使之可適用於協商 IPSec 的金鑰與安全關聯，協商過程如圖 1所示(圖中有網底部分為加密的訊息)，為了修正原有的弱點，加入了 TimeStamp(TS)來避免重送(replay)攻擊，使用會議金鑰來對發起者的身份作加密的保護以及確保第三個訊息的完整性，並利用 lookup table 來過濾重複收到的封包，以避免遭受 CPU 的 DoS 攻擊。Haddad[2]的金鑰交換協定僅有第一階段 IKE 的協商，第二階段的協商是利用快速模式來協商 IPSec 的金鑰與安全關聯，在 IKEv1[4]尚需要三個訊息的交換才能完成，協商過程如圖 2所示(圖中有網底部分為加密的訊息)：

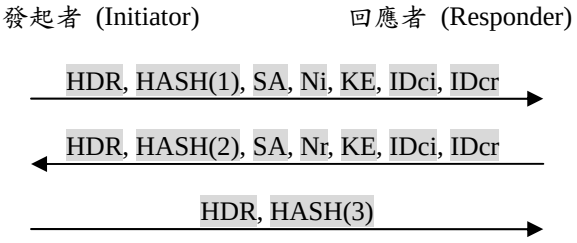

圖 2 第二階段協商(快速模式)

3. 本論文之網路金鑰交換協定

Haddad[2]的金鑰交換協定主要針對第一階段 IKE 金鑰與安全關聯的協商部分進行探討，並不包含第二階段 IPSec 安全關聯的協商，第二階段的協商是利用第一階段協商所得到的金鑰來加密交換的訊息，在進行一次第一階段協商後，可以再進行多次的第二階段協商，藉以增加 IPSec 的安全性。最早在 IKEv2[7]允許通訊雙方在第一次進行協商時，將第二階段的協商於第一階段來進行，之後如 IPSec 欲更新彼此的金鑰與安全關聯，可以直接進行第二階段協商，這樣可以減少雙方在第一次進行協商時所需交換的訊息數目，增加金鑰交換協定協商的效率，因此本論文的金鑰交換協定將會採用這樣的設計方式。圖 3為本論文之網路金鑰交換協定，圖中有網底部分為加密的訊息，框線部分為本論文的網路金鑰交換協定與 Haddad[2]協定的差異之處，除此之外於第三個訊息無需再將 u_r 回傳給回應者。

將一些與使用者身份無關的數值(k_r, r_r, u_r, x_r)透過離線(off-line)的預先運算(pre-computation)，可以省去回應者在收到連線請求後所需的線上(on-line)指數運算，然而 Hirose[1]與 Haddad[2]都未交代如何來處理並使用預先運算的數值，假如回應者在收到一個連線請求後，就使用掉一組預先運算的數值，攻擊者將可利用發送大量的連線請求給回應者，使得回應者用完所有預先運算的數值，此時回應者就必須再經計算以產生一些數值來使用，而產

生這些數值(k_r, r_r, u_r, x_r)所需的計算成本比產生會議金鑰所需的計算成本要高，因此將會導致回應者遭受 CPU 的 DoS 攻擊。

Haddad[2]為了增加原本協定的安全性加入了一些的數值(如 c_i、c_r、TS 與 h_i)，讓惡意的攻擊者無法輕易對協定造成威脅，當模數 p 越大時(bits 數越多)所增加的資料量幾乎以倍數成長，以乙太網路的最大傳輸單元(MTU)1500bytes 而言，超過該大小的封包都需要進行 IP 分段(IP fragment)傳送，回應者在收到分段傳送的封包時，會將它先存放到等待重組的佇列中，等收到剩下的封包後便進行 IP 分段重組(IP fragment reassembly)，而在佇列裡的分段封包，如未經重組將會保留 30 或 60 秒[6]。由於金鑰交換協定大都利用 UDP 協定來進行傳送，封包是否成功的傳送給對方將無法得知，因此惡意的攻擊者仍可利用大量分段但不完整的封包，佔據回應者等待重組的佇列，讓正常使用者的分段封包無法進行重組，也因此無法進行金鑰的協商，導致 DoS 攻擊[6]。

本論文所提之協定將這些可預先計算的數值先行計算後存放到一個檔案中，當需要用到這些數值時，便從檔案之中將數值取出，透過這樣的方式，原本 Haddad[2]金鑰交換協定裡的 $c_r = E_r(k_r, TS, u_r, x_r)$將改成 $c_r = E_r(offset, TS)$，使用 offset 到檔案之中取出相關的數值，另外發起者回傳的第三個訊息無須再夾帶 u_r，如此一來便能大幅的減少 c_r 的資料量，相對的第一與第三個訊息的封包大小將大幅縮減，而 offset 的資料型態為長整數(long)，可使得預先計算的資料量達到 2GB 左右的大小，可以讓伺服器端用上好長一段時間。所用掉的數值，可利用 CPU 空閒時計算並存放到另一個暫存檔中，當原本檔案裡頭的數值全部用完時，再使用暫存檔取代掉原本的檔案即可。

為了避免預先計算的數值被惡意的攻擊者耗費掉，如惡意的攻擊者僅是一直不斷地在發送第一個訊息，而沒有真正的完成整個協商，回應者的 offset 並不會指到下一組的數值，除非一個完整的協商過程有完成，回應者才會將 offset 指到下一組的數值，不會將預先計算的數值浪費掉。

另外，Haddad[2]仍舊保留原有的複雜計算，重複的使用 e_r、e_i 來進行 z_r 與 z_i 的運算以認證對方，而計算式 $z = g^w u^e v^{e^2} \bmod p$ 的用途除了用來認證對方外。第二個用途是為了讓發起者先行付出計算的成本，來避免惡意的攻擊者可以輕易的發動 DoS 攻擊。第三個用途則是用來驗證對方所使用的公開參數 u 是否是使用相同的 p、q、g 所合法產生，檢驗為合法的數值之後，方能用來產生彼此的會議金鑰。但刻意地重複使用 e 來增加發起者的計算成本，將同時影響到金鑰協商的效率。

原本在 Hirose[1]的金鑰交換協定中，回應者是先計算 z_i 來認證對方後，再計算彼此的會議金鑰，但是在 Haddad[2]的金鑰交換協定中，則是先計算彼此的會議金鑰，再計算 z_i 來認證對方。因此我們將不再重複的使用 e，且回應者在計算 z_i 時不再將 u_i 代入進行驗證。因此經過修改後，發起者所要作的計算為 $z_r = g^{w_r} u_r v_r^{e_r} \bmod p = g^{(r_r + k_r + e_r s_r)} g^{-k_r} g^{-s_r e_r} \bmod p = g^{r_r} \bmod p = x_r$；回應者所要作的計算為 $z_i = g^{w_i} v_i^{e_i} \bmod p = g^{(r_i + e_i s_i)} g^{-s_i e_i} \bmod p = g^{r_i} \bmod p = x_i$，能提高金鑰與安全關聯協商的效率。

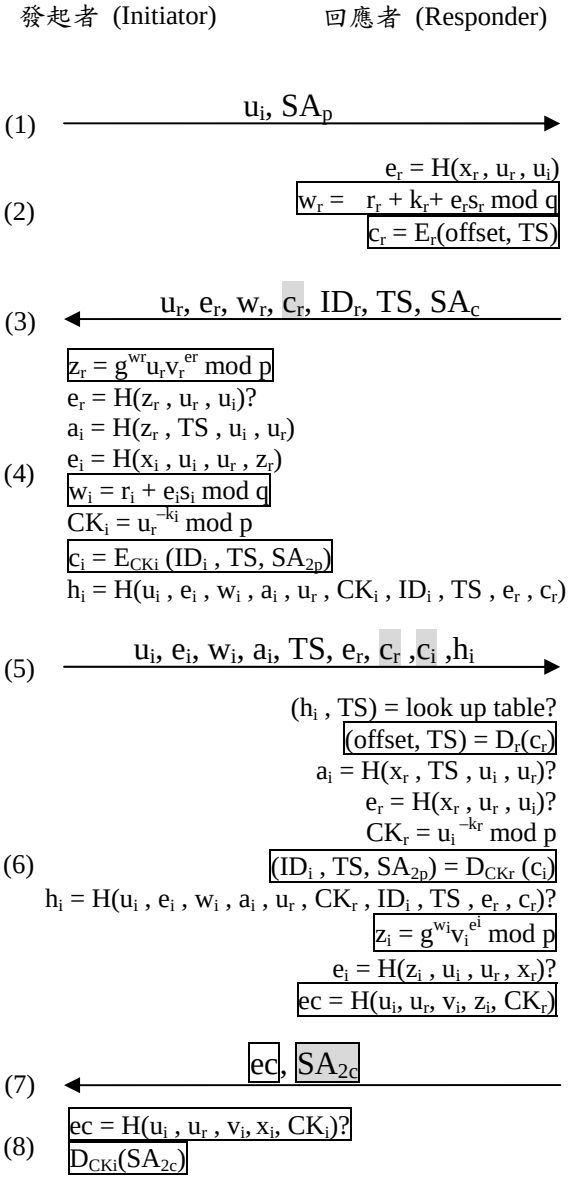

圖 3 本論文之網路金鑰交換協定

另外，Haddad[2]並沒有讓發起者得知回應者是否真正的計算了彼此的會議金鑰，因此發起者無法滿足明確的金鑰確認(Explicit Key Confirmation)[8]的特性，為了讓發起者也能擁有明確的金鑰確認，我們在第四個訊息裡加入了 ec 這個雜湊值，是回應者利用所計算出來的會議金鑰 CK_r 解開 c_i，並取出發

起者的公開鑰匙 v_i，連同計算出來的 z_i 以及本次產生會議金鑰用的公開數值 u_i 與 u_r 所產生，除了用來讓發起者作金鑰確認外，也能讓發起者得知回應者是否有對它進行認證。

安全性與效能分析

本論文的網路金鑰交換協定比起 Haddad[2]的協定少了兩個指數運算，以及僅使用四個訊息交換來協商 IKE 與 IPSec 所需之金鑰與安全關聯，減少了協商的回合數，進而提高協商的效率。而 Haddad[2]第一階段與第二階段共需要六個訊息交換才能協商出 IPSec 所需金鑰與安全關聯。

在我們的協定中，當回應者收到第一個連線請求後，將相關的資訊透過對稱式加密傳送給發起者後，並不需要紀錄任何的資訊，因此不會遭受記憶體的 DoS 攻擊。透過預先運算能避免在收到連線請求後所需的線上運算，可以避免遭受 CPU 的 DoS 攻擊，並且只有在一次成功的金鑰協商後，回應者才會使用下一組預先計算的資料，不會因為攻擊者不斷重複的發送第一個訊息，而把預先運算的資料給耗盡，也能有效的避免遭受 CPU 的 DoS 攻擊。

如攻擊者想利用不斷重複的發送第三個訊息給回應者，迫使回應者去作產生金鑰與認證對方的運算，也將會被回應者所建立的表格檢查出封包是否重送，因此也無法造成 CPU 的 DoS 攻擊。所建立的表格並不會無限的擴大，所記錄的資料筆數將會是在有效的 TimeStamp(TS)裡，所能進行金鑰協商的最大次數，以防止記憶體的 DoS 攻擊並且增進查表的效率。另外，節省傳送所需的資料量(c_r)可以減少第二與第三個訊息封包的大小，盡可能避免封包需要 IP 分段傳送，可以降低發生 IP 分段 DoS 攻擊的機會，這樣的作法使得 c_r 的資料長度變短，有可能讓攻擊者較容易猜到原本加密的內容，但是即使攻擊者猜到內容為何，也無法取得用來計算會議金鑰用的 k_r 值，及用來確認對方有對自己認證用的 x_r 值。

Haddad[2]僅有讓回應者去檢驗發起者是否有真正的計算會議金鑰，並沒有讓發起者去檢驗回應者是否有真正的計算會議金鑰，在本論文的協定中我們使用了 ec 來讓發起者滿足明確的金鑰確認(Explicit key confirmation)的特性，因此可避免未知金鑰攻擊[8]。

4. 實驗與結果

我們以 Linux 作業系統作為程式的開發環境，分別實作出了 Haddad[2]以及本論文的網路金鑰交換協定，並進行了 DoS 攻擊的測試，協定中所使用之雜湊函數、加解密演算法以及相關亂數、質數的產生與運算皆使用 OpenSSL[5]所提供之 Library。

4.1 實驗環境與設定

實驗環境架構如圖 4所示，PC-A 與 PC-B 皆使用 Linux 作業系統，並且都不運行 X-Windows，以 PC-A 作為發起者，PC-B 作為回應者，這兩部電腦是透過 Switch (D-Link DES-1016D 16-Ports 10/100 Fast Ethernet Switch) 連接在同一個區域網路裡，而當作攻擊者的電腦則位於另外一個子網路中。

實驗的目的是為了測試當回應者遭受到 DoS 攻擊時，正常的發起者在跟回應者進行金鑰協商時，是否會受到攻擊者影響，因此我們分別記錄了發起者在正常情況下及回應者在遭受 DoS 攻擊情況下，發起者與回應者在進行金鑰協商所需的時間，另外記錄了回應者在遭受 DoS 攻擊時 CPU 的使用狀態。

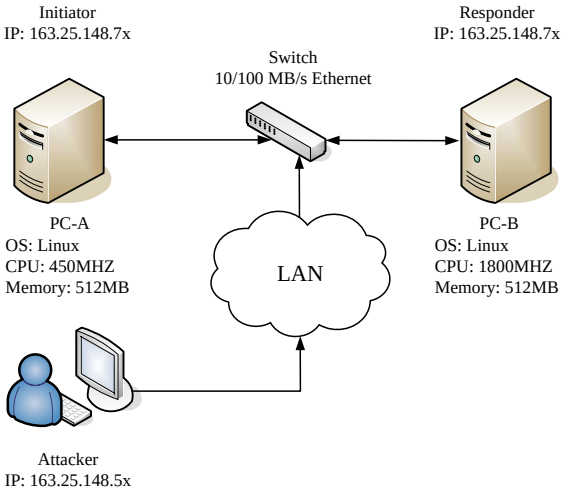

圖 4 DoS 攻擊測試環境圖

我們分別進行了三種 DoS 攻擊的測試，描述如下：

一、 使用第一個訊息，攻擊者僅發送大量的第一個連線請求的訊息給回應者，即攻擊者在發出第一個訊息給回應者後，不理會回應者所回傳的第二個訊息。

二、 使用第三個訊息，攻擊者發出第一個連線請求的訊息給回應者，且收到回應者所回應的第二個訊息後，重複發送有效的第三個訊息給回應者。

三、 使用第一與第三個訊息，攻擊者結合上述兩種攻擊方式，同時發送大量的第一與第三個訊息給回應者。

在第一種攻擊方式中，我們分別讓攻擊者每秒發送 500 次與 1000 次的連線請求給回應者；在第二種攻擊方式中，則分別讓攻擊者每秒回應 500 個與 1000 個有效的第三個訊息給回應者；在第三種攻擊方式中，則分別讓攻擊者每秒發出總數 500 個與 1000 個的第一與第三個訊息給回應者，第一與

第三個訊息所佔比例各為 1/2。

另外也比較了回應者在採用不同的加解密演算法來處理 c_r 值時，是否也會對金鑰協商造成影響，除此之外，在協定中所有的加解密演算法均使用 AES128，雜湊演算法均使用 SHA1。所量測的協商時間為發起者在發送出第一個訊息到收到第四個訊息且確認無誤後所需的時間，為進行 100 次金鑰協商的平均值。

4.2 實驗結果

表 1為回應者在遭受第一種 DoS 攻擊時，發起者進行金鑰協商所需的時間，可以得知在不同規模的 DoS 攻擊下，發起者在進行協商時並不受到攻擊者的影響，且回應者採用不同的加解密演算法來處理 c_r 值時，所需的協商時間也沒有明顯的差別。

在處理第一個連線訊息時，本論文的金鑰交換協定是將一些可以預先運算的值(k、r、u 與 x)存放到一個檔案中，並且在沒有完成一次成功的協商之前，即便收到再多的連線請求也不會取出下一筆資料，可以避免攻擊者將這些預先運算的值耗盡，而迫使回應者要去運算產生這些值(k、r、u 與 x)來回應不當的連線請求。

表 1 第一種攻擊下金鑰協商時間 (時間單位：秒)

	Normal	500Req./s	1000Req./s
DES	0.0223	0.0222	0.0255
2DES	0.0246	0.0237	0.0258
3DES	0.0225	0.0287	0.0238
AES128	0.0225	0.0251	0.0224
AES256	0.0264	0.0248	0.0266

表 2為回應者在遭受第二種 DoS 攻擊時，發起者進行金鑰協商所需的時間，此時協定中所有的加解密演算法均使用 AES128，可以得知攻擊者所發動的第二種 DoS 攻擊並沒有對正常的金鑰協商造成明顯的影響。

由於攻擊者發送有效的第三個訊息給回應者，回應者驗證無誤後，便會將 h_i 與 TS 的值存放到表格中，如果攻擊者重複發送該訊息，欲使回應者去付出計算成本，造成 CPU 的 DoS 攻擊，回應者就會檢查出來並且不再作進一步的處理。

表 2 第二種攻擊下金鑰協商時間 (時間單位：秒)

Normal	500Msg.3/s	1000Msg.3/s
0.0225	0.0232	0.0233

表 3為回應者在遭受第三種 DoS 攻擊時，發起者進行金鑰協商所需的時間，可以得知在第三種 DoS 攻擊的情況下，發起者在進行金鑰協商時所需的時間出現了比較明顯的延遲。

表 3 第三種攻擊下金鑰協商時間 (時間單位：秒)

	Normal	500Msg.1&3/s	1000Msg.1&3/s
DES	0.0223	0.0286	0.0320
2DES	0.0246	0.0299	0.0342
3DES	0.0225	0.0297	0.0370
AES128	0.0225	0.0305	0.0356
AES256	0.0264	0.0297	0.0356

圖 5為使用每秒 1000 次的第一個連線請求訊息進行 DoS 攻擊時回應端的 CPU 使用狀態(僅擷取某分鐘的資料)；圖 6為使用每秒 1000 個有效的第三個訊息進行 DoS 攻擊時回應端的 CPU 使用狀態；圖 7為使用每秒 1000 個的第一與第三個訊息進行 DoS 攻擊時回應端的 CPU 使用狀態(User 代表回應端程式所佔用的 CPU 時間，Sys 代表作業系統所佔用的 CPU 時間，Idle 則代表 CPU 空閒的時間)。可以得知回應端在處理同數量 DoS 攻擊的訊息時，只處理第一個訊息時 CPU 所花的代價是最高的，而只處理第三個訊息時 CPU 所花的代價是最低的，而同時處理第一與第三個訊息時 CPU 所花的代價則是介於上述兩者之間。

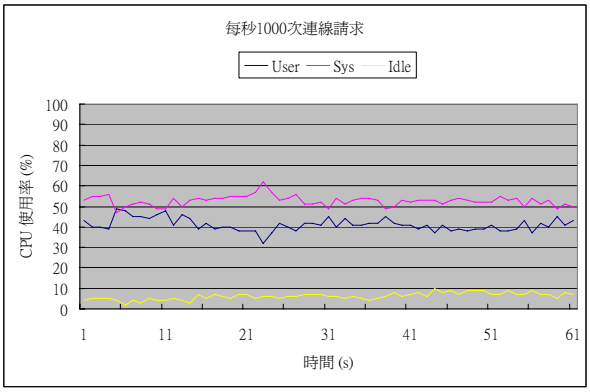

圖 5 回應端於第一種攻擊下 CPU 使用狀態

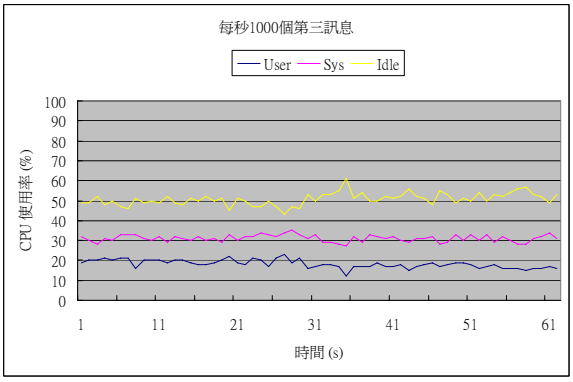

圖 6 回應端於第二種攻擊下 CPU 使用狀態

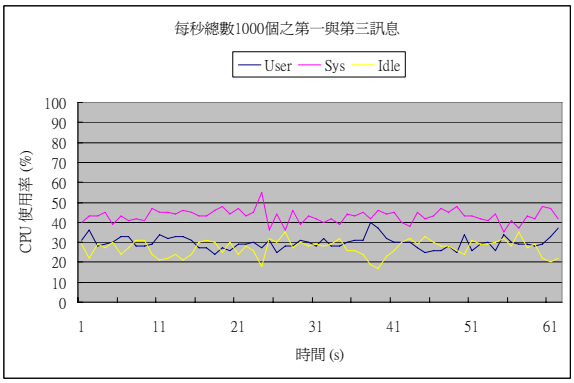

圖 7 回應端於第三種攻擊下 CPU 使用狀態

當攻擊者所發動的 DoS 攻擊，造成回應者 CPU 的空閒時間處於 0%時，即超過了回應者 CPU 處理能力的上限，回應者因無法馬上處理完攻擊者的封包，在回應者接收封包的暫存區將充斥著攻擊者的封包，使得正常發起者所發送的封包無法被回應者接收，即使回應者此時能夠回覆第二個訊息給發起者，但是發起者所回傳的第三個訊息也不一定能夠再進得了暫存區讓回應者來接收。只要攻擊者停止了 DoS 攻擊，回應者 CPU 的空閒時間很迅速的就能恢復到 100%，繼續服務正常的連線請求。

5. 結論

本論文基於 Haddad[2]的金鑰交換協定，提出了一個改良的網路金鑰交換協定，使用了四個訊息來協商 IPSec 所需的金鑰與安全關聯，透過較少的計算量、資料量及協商的回合數來完成協商，提高了金鑰與安全關聯協商的效率；而對預先運算的處理，也讓攻擊者的 DoS 攻擊無法產生作用；另外也增進了原本協定的安全性，讓通訊雙方都能夠去作明確的金鑰確認，可避免未知金鑰攻擊；實驗的結果也驗證了我們的網路金鑰交換協定具有抵擋 DoS 攻擊的能力，讓 DoS 攻擊無法對金鑰與安全關聯的協商造成影響。

經由協定的修改，可讓攻擊者無法輕易的達成 DoS 攻擊的目的，降低 DoS 攻擊所造成的影響，使用不同等級的電腦其所能承受 DoS 攻擊的程度也不同，且每部電腦的處理能力都是有限的，要全面防堵 DoS 攻擊仍需搭配網路上其他的設備，譬如防火牆、網路入侵偵測或防禦系統，方能夠架構出一個安全的網路環境。

誌謝:感謝行政院國科會專題研究計畫之補助，使本論文得以順利完成；計畫編號 NSC94-2213-E-130-022

參考文獻

[1] S. Hirose, K. Matsuura, “Key Agreement Protocols Resistant to a Denial-to-Service” IEICE Trans. On Information and Systems, Vol.E84-D, No.4, pp.477-484, April 2001.

[2] Haddad, H., Berenjkoub, M., Gazor, S., “A PROPOSED PROTOCOL FOR INTERNET KEY EXCHANGE(IKE)”, Electrical and Computer Engineering, Canadian Conference, May 2004.

[3] Diffie W, Hellman M, “New direction in cryptography”, IEEE transactions on Information Theory, Vol. 22, Issue 6, p.644-654 , Nov 1976.

[4] D. Harkins and D. Carrel, “The Internet Key Exchange (IKE)”, RFC 2409, November 1998.

[5] OpenSSL Project, http://www.openssl.org/

[6] Charlie Kaufman, Radia Perlman and Bill Sommerfeld, “DoS Protection for UDP-Based Protocols”, CCS’03, Washington, DC, USA, October 27-31, 2003.

[7] Charlie Kaufman, “Internet Key Exchange (IKEv2) Protocol”, IETF, draft-ietf-ipsec-ikev2-17, September 23, 2004.

[8] Blake-Wilson S, Menezes A, “Authenticated Diffie-Hellman key agreement protocols”, Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC’98), Lecture Notes in Computer Science 1556, p.339-361, 1999.

[9] H. Krawczyk, “The IKE-SIGMA Protocol”, Internet Draft (draft-krawczyk-ipsec-ike-sigma-00.txt), Nov, 2001.

[10] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis and Omer Reingold , “Just Fast Keying: Key Agreement in a Hostile Internet”, ACM Transactions on Information and System Security (TISSEC), Vol. 7, No. 2, p.242-273, May 2004.

[11] Ajmal S. Mian and Ashraf Masood, “Arcanum: A Secure and Efficient Key Exchange Protocol for the Internet”, IEEE Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’04), Vol. 1, p.17-21, 2004.

[12] Kanta Matsuura, Hideki Imai, “Modification of Internet Key Exchange Resistant against Denial-of-Service”, Internet Workshop 2000 (IWS2000), pp.167-174, Feb 2000.