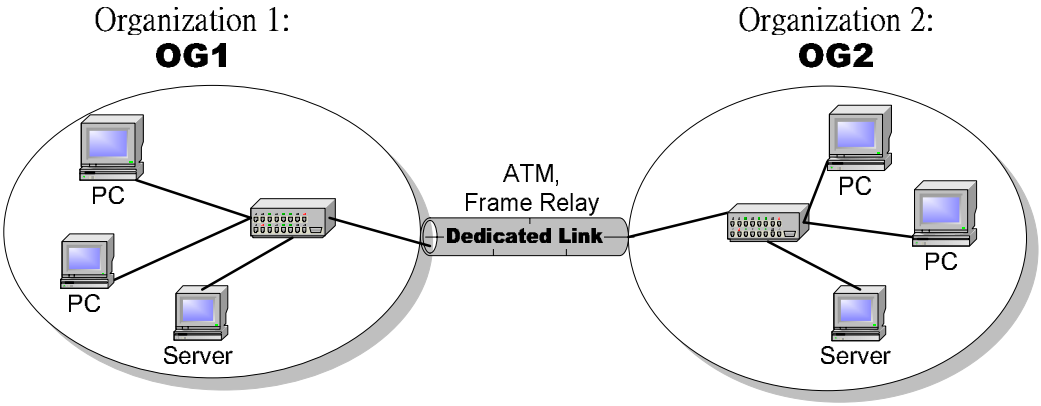
虛擬私有網路(Virtual Private Network - VPN) :

IPSec 功能性、互通性及效能性測試

Ⅰ. 實驗目的

以往企業間要建立 Private Network (即資料的傳輸是具有私密性及可信賴性)的方法，是透過實體線路的建置，使企業與企業之間有專用的線路來傳輸資料:



這樣的方式具有下列缺點:

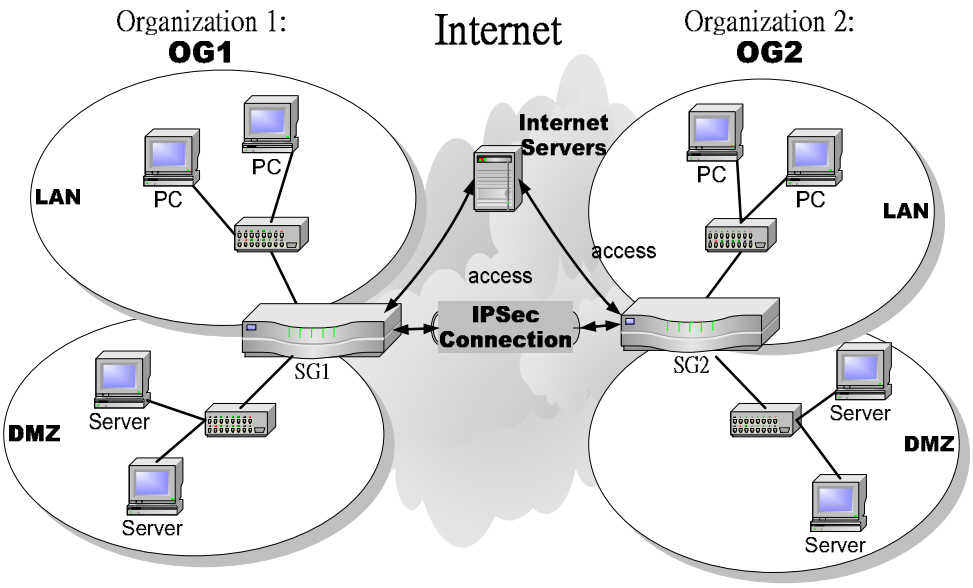
(1) 費用昂貴

(2) 網路設定複雜，不能由使用者自行設定、建置。

(3) 私密性由 ISP 掌控

因此才有 Virtual Private Network – VPN 的出現，VPN 可以說是:“用 Public 網路環境(i.e. Internet)來傳送 Private 資料”,之所以可以在 Public 的網路下傳送需要私密性的資料，這便是靠”密碼學”的方式來達成，Sender 將要傳送的資料加密後才送出，Receiver 將資料解密後才知道真正的資料是什麼，因此密碼學方法解決了專線方法的(1)和(2)這兩種缺點，因為資料是透過 Internet 傳送，使用者只需付上網的費用，不需付專線的費用，同樣地也不用有專線設定上的困擾。

而目前最被廣泛使用的 VPN 技術就是 IP Security – IPSec，其通常的網路建置架構如下:



OG1 可以想成是一間企業的總公司，而 OG2 是其分公司，由於目前 IPv4 的 address 不夠用，所以內部的機器會分配使用 Private IP，在公司的網路出口處再放一台具有轉址(NAT)功能的機器負責 Private IP 和 Public IP 之間的轉換，再者就是公司內部會放置一些 Server (mail, web, ftp)讓外部可以存取，所以這樣的 Server 通常是是和其它機器分隔開成為不同的網段，一般員工使用的機器放在"LAN"網段，而需要讓外部存取的機器放在"DMZ"網段(DeMilitarized Zone; DMZ 非軍事區)，這兩種網段不僅會用 physical interface (port)區別開，避免在 LAN 的機器有被攻擊的危險，更會有 Firewall rule 對 LAN 的機器加以保護，所以說在出口處這台機器會具備 L3/L4 Packet Filter 的功能，當然它也會具有 IPSec 的功能讓 OG1 和 OG2 內部的機器可以透過 IPSec connection 來進行具有 privacy 和 authenticity 的通訊，因為公司之間所傳輸的資料通常是不能對外公開的機密文件，像這樣具有多功能安全性的機器我們稱之為 Security Gateway(SG)，而由上述可以了解通訊時的私密性是由使用者本身所控制，不是讓 ISP 來控制，如此也克服了 dedicated link 式 VPN 的第(3)點缺點。

在 IPSec VPN 這項技術目前運作起來所遇到的問題主要有兩個: 互通性與效能性。互通性的問題是來自於每家廠商在 IPSec 的設定方式上沒有一定的介面，而 IPSec 所需要的參數可多可少，變化很多，所以使用者必須要對產品裡的參數深入地了解才能順利建立 IPSec connection; 效能性方面問題是因為 SG 必須對資料進行加、解密的動作，對 CPU 的負荷相當的大，因此有的廠商便會使用提供加、解密功能的 ASIC (Application-Specific IC)來加速解、解密動作以並且 Offload CPU 的負擔，這樣對於多合一功能的 SG 來說才有辦法可以同時讓多項功能運作順利，綜合以上的原因，我們才會對 IPSec 的功能性、互通性及效能性進行測試。

Ⅱ. 實驗設備

本實驗所用到的硬體及軟體設備如下兩個表格所示。

一、硬體

項目	數量	備註
SmartBits 6000B	1	2 * SmartMetrics card
Security Gateway	8	
PC	2	
Ethernet Switch	3	

二、軟體

項目	數量	備註
SmartFlow	1	

Ⅲ. 背景資料

一、IP Security (IPSec)

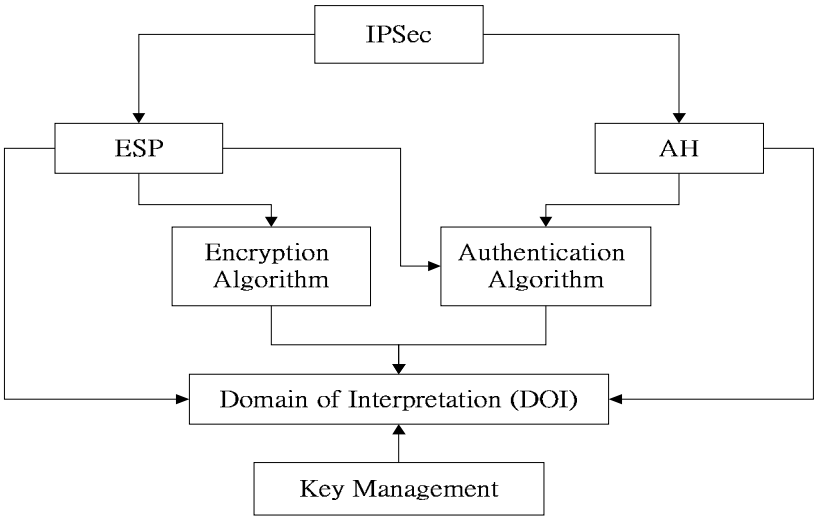
近年來由於網際網路技術的成熟,越來越多的使用者享受著網際網路這樣方便的公眾網路。隨著許多商業服務建構於網際網路上,私密通訊已經成為網路使用者非常關心的議題,因為這些商業服務往往需要機密性資料的傳輸,如果沒有一個值得信賴的安全性網路,便會使得網路使用者對網路商業服務的信心大打折扣。

為解決這樣的問題,許多設計於會議層(Session Layer)與應用層(Application Layer)的網路安全標準相繼被提出。如 SET 與 SSL 可達到 HTTP 的安全性,由 IETF 的 PSRG 小組所提出的 PEM 標準可達到 E-mail 的安全性,RFC1508 與 RFC1509 所訂定的 GSSAPI(General Security Service Application Program Interface) 可以使得 Telnet,FTP,以及 HTTP 達到安全傳輸等等。從上述的各種標準中可以發現,要在一種網路應用上達到安全傳輸,就需要一套專屬的安全標準。事實上,這些網路應用主要都使用同一種通訊協定—網際網路協定(Internet Protocol,IP)。因此,如果能夠訂出一套使用於 IP 網路的安全性機制,就可以整合各種網路安全標準,免除一種網路應用使用一套網路安全標準的麻煩。

在這樣的情況下,為達到網路層中點對點的安全通訊,IETF 開始訂定一套

開放標準網路安全協定 IPSec(IP Security)，來提供現在的 IPv4 網路，以及未來的 IPv6 網路一個標準化，應用密碼技術，以及高品質的通訊環境，希望能夠提供資料送端與收端執行認證(Authentication)，完整性(Integrity)，秘密性(Confidentiality)，以及存取控制(Access Control)等安全服務。

IPSec 第一代的版本(RFC1825 至 RFC1829)於 1995 年提出，其主要的架構包括 IP 認證標頭(Authentication Header，AH)以及 IP 封裝安全裝載(Encapsulation Security Payload，ESP)兩項。前者主要提供資料的完整性以及認證，後者則提供資料的秘密性。為了使得 IPSec 可以使用於 IPv6 的環境中，分別將兩者設計成為延伸標頭(extension header)，也就是認證標頭(Authentication Header)與封裝安全裝載標頭(Encapsulation Security Payload Header)。在第一版的 IPSec 中，並未提到金鑰的交換與管理，其主要在於定義封包格式的轉換。直到 1998 年 IPSec 第二代版本(RFC2401，RFC2402，RFC2406)提出，其中加入安全群組(Security Association，SA)與金鑰管理 IKE(Internet Key Management)後，整個 IPSec 架構才算較為完整。



IPSec protocols	Authentication Header (AH) Encapsulation Security Payload (ESP)
IPSec 主要運作核心	Security Association (SA)
參數定義與描述語言	Domain of Interpretation (DOI) Security Policy Specification Language (SPSL)
Key Management	ISAKMP/OAKLEY IKE

二、Authentication Header (AH) and Encapsulation Security Payload (ESP)

IPSec 主要有兩種模式：AH（Authentication Header）與 ESP（Encapsulating Security Payload）[Elizabeth99]。Authentication Header（AH）定義於 RFC 2402 中，目的是在 IP 封包中加入認證的功能，以確認封包的完整性及不可否認性。而 ESP 則定義了對一個 IP 封包所用的加密與進一步的資料完整性認證的標準。AH 所用的演算法預設為 MD5（定義於 RFC-2403 中），而 ESP 的演算法則預設為 DES（定義於 RFC-2405 中）

AH 標頭[RFC 2402]主要是要用來提供欄位的完整、認證及不可否認性，欄位的能以及各種標頭的格式。完整是指能夠判斷封包的資料是否有被更改，而認證及不可否認性則是能讓使用者可以認證對方的身份，同時也不能否認使用者曾經做過的行為。此外，AH 的機制可以抵擋匿名的攻擊（spoofing attack）以及重播的攻擊（replay attack）。下圖 15-5 為 AH 之標頭格式。

Next Header: 8 bits	Payload Len: 8 bits	RESERVED: 16 bits
Security Parameters Index (SPI): 32 bits		
Sequence Number Field: 32 bits		
Authentication Data (variable)		

ESP 標頭[RFC 2406]則主要用於資料之保密，所有要加密的資料完全放於 ESP 標頭下，除了加密之外，對於加密之資料還可跟 AH 一樣，加上一認證值以確保資料之完整性。

Security Parameters Index (SPI): 32 bits		
Sequence Number: 32 bits		
Payload Data (variable)		
Padding (0-255 bytes)		
Pad Length: 8 bits	Next Header: 8 bits	
Authentication Data (variable)		

ESP 的主要欄位包括 SPI、Sequence Number、加密過的資料（Payload）、認證資料(Authentication Data)，如同 AH 的封裝機制，SPI 用來辨別不同的連線，而 Sequence Number 用來辨別同一個連線的不同封包，Payload 是放置加密後的結果，Authentication Data 則用來放 ESP header 的 hash value。

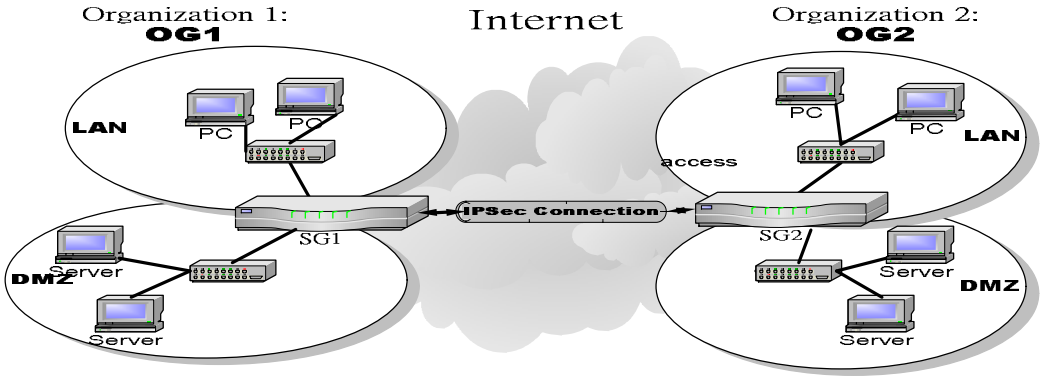
三、Security Association (SA)

IPSec 要達到私密通信，首先必須提供一個安全的環境提供資料送端與資料收端來傳送資料。而 SA 最主要的目的，就是在定義一個安全的環境。在整個 IPSec 的架構中，SA 可說是最重要的一個概念。

對資料送端與資料收端來說，SA 就是一個「提供安全傳輸的單向連線」。SA 中定義了一些參數，如認證所採用的演算法與所使用的金鑰，加(解)密演算法與其金鑰，以及金鑰的有效期限等等。因此只要通訊雙方使用的同一個 SA，就可以進行秘密通信。一個 SA 可用一個 32 位元的安全參數索引(Security Parameter Index，SPI)來描述，而一台機器的 IP 位址，一個安全協定識別碼(代表 AH 或 ESP)，再加上 SPI，就可以定義一個唯一的 SA。舉例來說，假設 SPI 值為 2500 所代表的安全群組使用 MD5 演算法進行驗證，使用金鑰為 X，安全協定識別碼為代表 AH 之識別碼。資料送端 A 可以通知資料收端 B 安全群組之 SPI 值為 2500，安全協定識別碼為代表 AH 之識別碼。接著資料送端 A 就利用 SPI 為 2500 所對應的安全群組參數加密資料並送至資料收端 B，資料收端 B 在收到加密後的資料後，透過封包內來源 IP 位址可知資料由 A 送來，配合 SPI 與安全協定識別碼就可以進行認證的工作。要注意的是，由於 SA 是屬於單向的關係，因此若要建立點對點的雙向安全傳輸，就需要兩個 SA。此外，一個 SA 只能使用 AH 或 ESP 其中一種安全協定，因此如果要同時使用 AH 與 ESP 兩種安全協定，也需要兩個 SA。

四、Transport Mode and Tunnel Mode

IPSec 提供兩種不同的傳送模式，稱為直接傳輸模式（transport mode）及 通道傳輸模式（tunnel mode）。在 transport mode 中，只有傳輸層（Transport Layer）的資料會被保護，其主要目的是讓要通訊的兩點(ex.圖 x 的 SG1 和 SG1)間本身建立一條安全的通訊連線，而在 tunnel mode 中，可以保護整個 IP 封包，其主要目的是讓兩組要通訊的群組(ex. 圖 x 的 OG1 和 OG2 的 LAN/DMZ)可以透過某兩點(ex.圖 x 的 SG1 和 SG1)圖所建立的安全連線來互相傳遞資料。



五、Key Management – Internet Key Exchange (IKE)

IPSec 中關於金匙的管理，有分成手動式(Manual Key)和自動式(Aotumated)兩種，依技術面來講可以分成金匙的產生(Key determination)和金匙的交換(Key distribution)兩大部分。

手動式金匙的管理 — 系統管理者需要手動設定所需要的每個金匙的內容，並手動的傳送給其他每台使用 IPSec 通訊的機器。這適用於一個小範圍且變動不大的區域網路中。

自動式金匙的管理 — 金匙只在有需要的時候才自動的產生，所有 SA 的交換都在自動的方式下傳送，適用於比較大的分散式系統中，在 IPSec 中內定的自動式金匙管理方式是 IKE 協定。

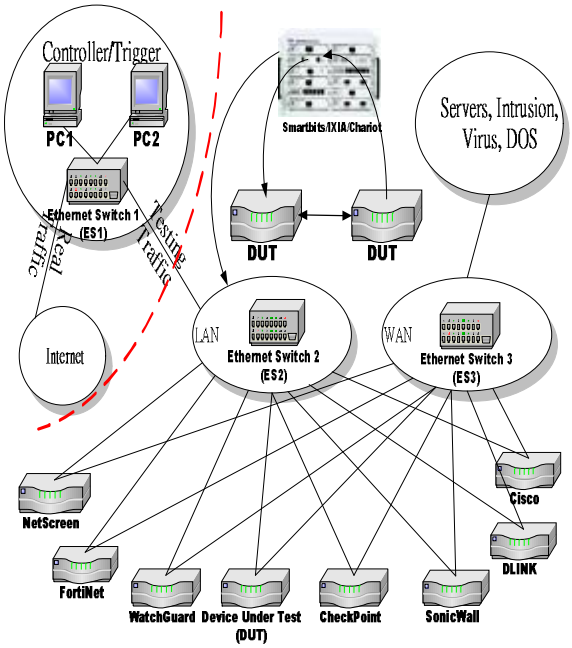
IKE 是一個在網路上的兩台主機之間自動建立、協商、修改和刪除 SA 的協定。SA 包含了資訊用來在網路上資料交流的雙方之間建立一安全的通道，如封包加密或認證時用的演算法、加密和認證用的金鑰、加密金鑰的存活期限、SA 的存活期限和防止封包複製攻擊的序號等。IKE 包含了 Oakley Key Determination Protocol、Internet Security Association and Key Management Protocol (ISAKMP)兩大部分:

Oakley — Oakley 是一個架構在 Diffie-Hellman 演算法上的金匙交換協定，但其改良了一些 Diffie-Hellman 所沒有的安全機制。

ISAKMP — ISAKMP 提供了一個在 Internet 上金匙交換的平台架構，其中包括了各種金匙交換協定的支援，其中包括了所要傳送的各種封包格式，以及所用來協調連線雙方所要用的 SA 與其他與安全性有關的屬性等。ISAKMP 本身並不包括任何指定的金匙交換演算法，相反的，ISAKMP 提供了一系列的封包格式，以用來支援許許多多的金匙交換演算法，其中，OAKLEY 就是 ISAKMP 剛開始提出時所採用的一種金匙交換演算法。對於更深入的內容請參考附錄，裡有特別針對 OAKLEY、ISAKMP 以及 IKE 作詳細的介紹。

Ⅳ. 實驗方法

實驗環境之網路架構圖:



在此網路架構中有幾點要注意到的是:

(1) 虛線的左邊代表工作桌上的測試環境，右邊代表機櫃裡的測試環境。

(2) PC1 和 PC2 連接到 ES1，所有 SG 的 LAN interface 連接在 ES2，所有 SG 的 WAN interface 連接在 ES3，所以 PC1 和 PC2 可以設定成任何一台 SG 的 LAN 下的一台機器。

(3) 除了可以用 PC1 和 PC2 來設定 SG 之外，Smartflow 也是灌在 PC1 及 PC2 上，所以待測試"效能性"時，也是用 PC1 和 PC2 來 trigger SmartBits 產生流量。

(4) DUT 並不會固定是哪一家廠商、哪一個型號的機器，待作業公告後再決定之。

一、功能性測試方法

拿一台"golden sample"來和 DUT(Device Under Test)建立 IPSec connection，接著在此 connection 中傳輸各種不同的 application data，若能順利完成測試，代表此次所設定的參數其功能性沒有問題，至於"golden sample" 則待作業公告後再決定之。

測試項目

(1) Tunnel mode, ESP

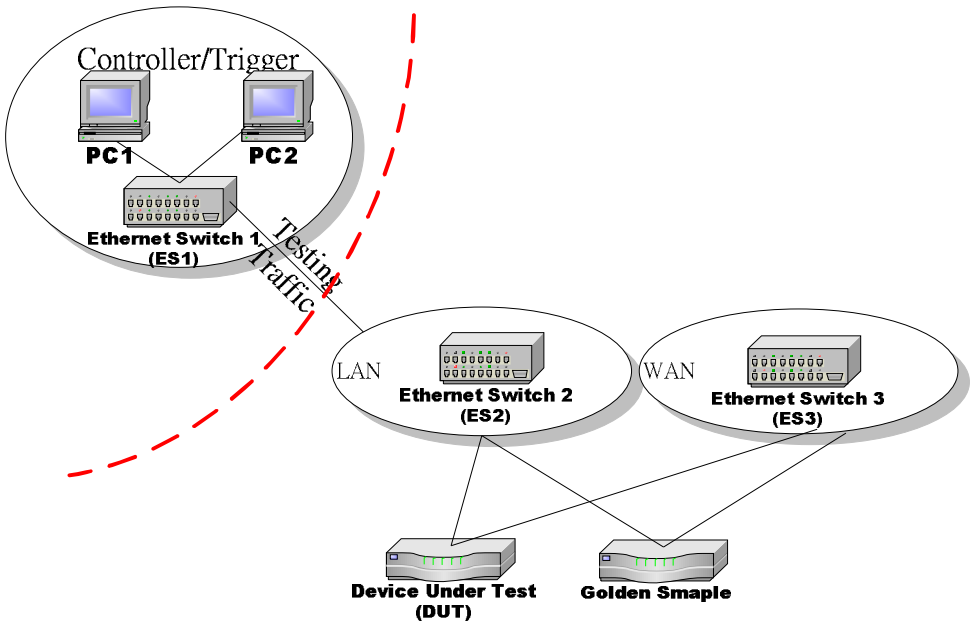
(2) Encryption algorithms

(3) Hash algorithms

(4) Rekey (SA lifetime)

(5) Initiator/Responder
(6) IKE Phase 1 Key group
(7) IKE Phase 2 DH group
(8) Application layer data: ping, ftp

測試組態



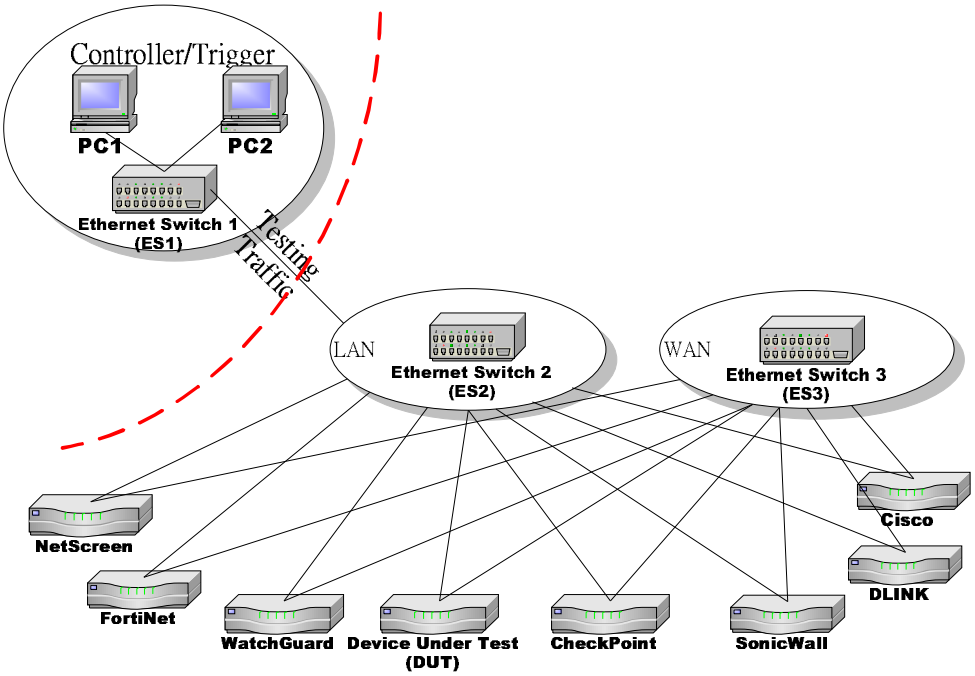
二、互通性測試方法

用 DUT 與其它家廠商的 SG 建立 IPSec connection，能與愈多家廠商的 SG 在各種設定下皆能建立 IPSec connection 的話，代表其互通性能力愈佳。請注意，在這裡選擇的互通之對象不能為 DUT 本身以及”golden sample”。

測試項目

(1) 各家廠商的 SG
(2) Tunnel mode, ESP
(3) Encryption algorithms
(4) Hash algorithms
(5) Rekey (SA lifetime)
(6) Initiator/Responder
(7) IKE Phase 1 Key group
(8) IKE Phase 2 DH group
(9) Application layer data: ping, ftp

測試組態



三、效能性測試方法

使用兩台 DUT 建立 IPSec connection，成後建立之後使用 Smartbits/Smartflow 產生流量，由一台 DUT 的 LAN interface 進入，從另一台 DUT 的 LAN interface 出來，中間經過 DUT 的 Encryption/Hash 的處理。

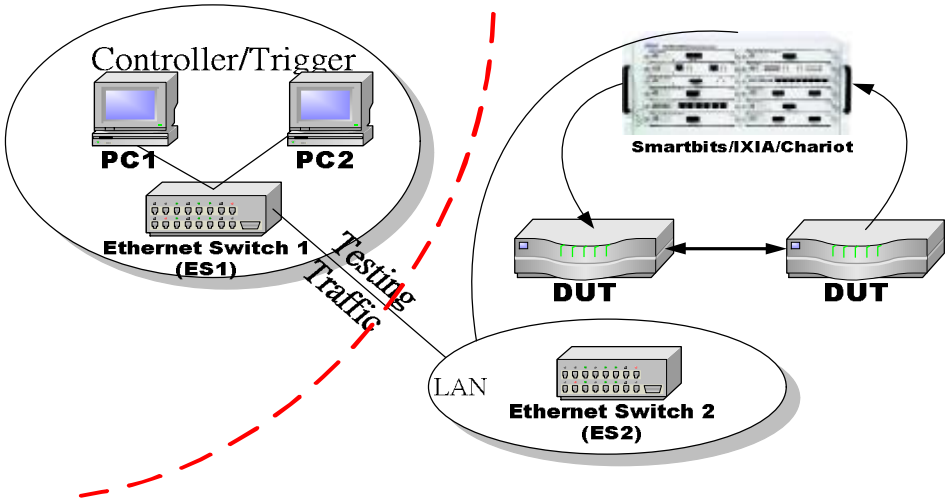
測試項目

(1) Frame size (byte): 64, 512, 1450, 1518

(2) Encryption/hash algorithms: Null/SHA1, DES/SHA1, 3DES/SHA1, AES/SHA1

(3) Transport layer protocol: TCP/UDP

測試組態



V. 實驗步驟

一、功能性實驗步驟

固定測試參數方法:

請注意 IPSec 參數的設定要符合"實驗方法"裡面所列之"實驗項目"，以下所列的例子僅供參考用:

step 1. 找出一組可以成功建立起 IPSec connection 的參數
Ex. ESP, DES/MD5, SA lifetime=86400 seconds, DUT as Initiator, Key group: DH1, PFS: DH1
step 2. 改變要測試的項目其所對應的參數，其它參數則固定不變
Ex. ESP, ***3DES***/MD5, SA lifetime=86400 seconds, DUT as Initiator, Key group: DH1, PFS: DH1，這個例子裡只改變 encryption algorithm，從 DES 改成 3DES。
step 3. 依尋上述法則將所有的測試項目完成。

設定 DUT 與"golden sample" **【記錄 1】**

以 D-Link DFL-900 及 NetScreen NS-5GT 為例:
兩者的網路設定如下:

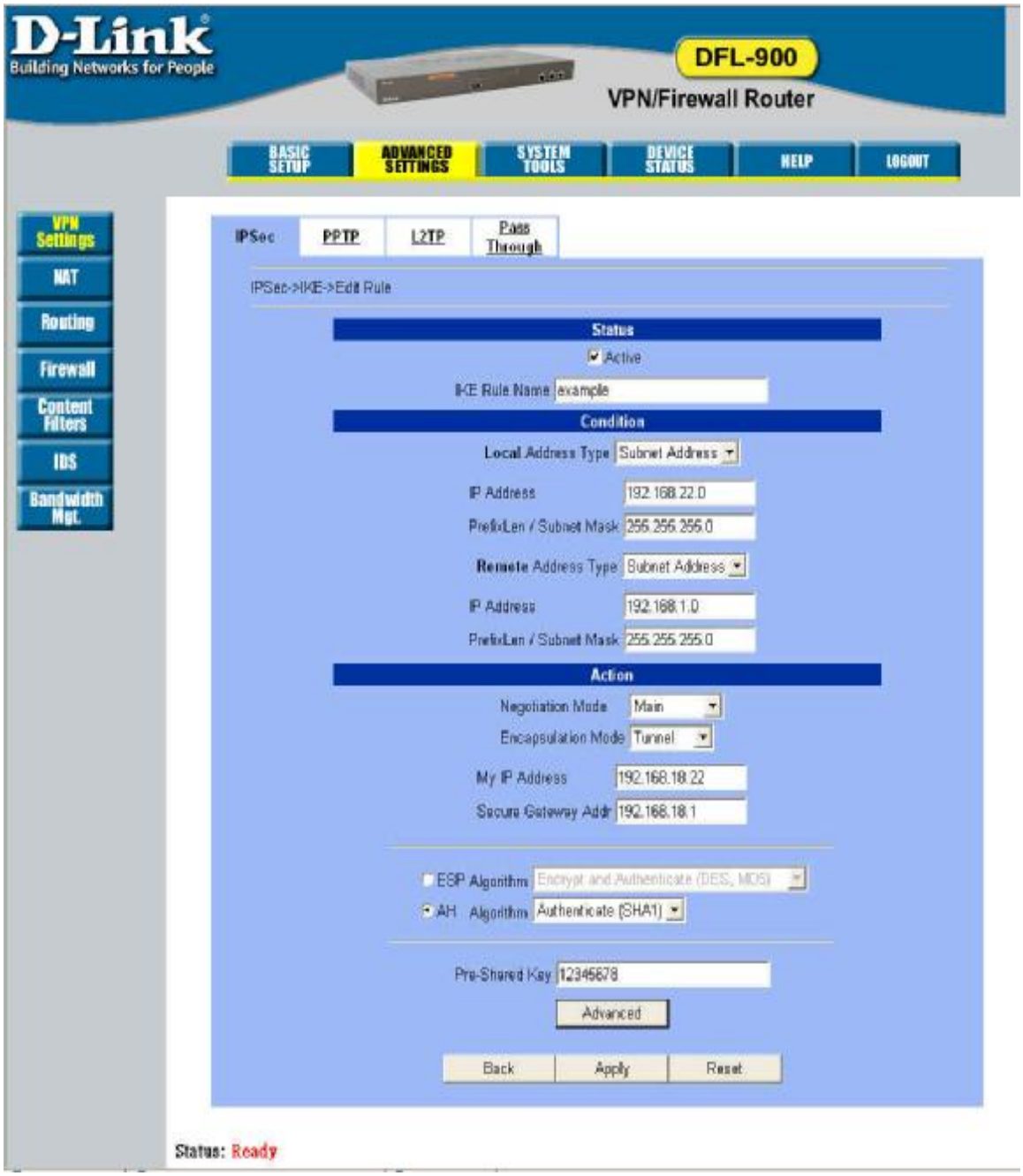
	DFL-900	NS-5GT
WAN IP/Netmask	192.168.18.22/24	192.168.18.1/24

LAN IP/Netmask	192.168.22.254/25	192.168.1.254/24
Default gateway	192.168.18.254	192.168.18.254

D-Link DFL-900 的 IPSec(IKE)設定如下圖: (參考用)

Step 1. 選 ADVANCED SETTINGS->VPN Settings->IKE->Add

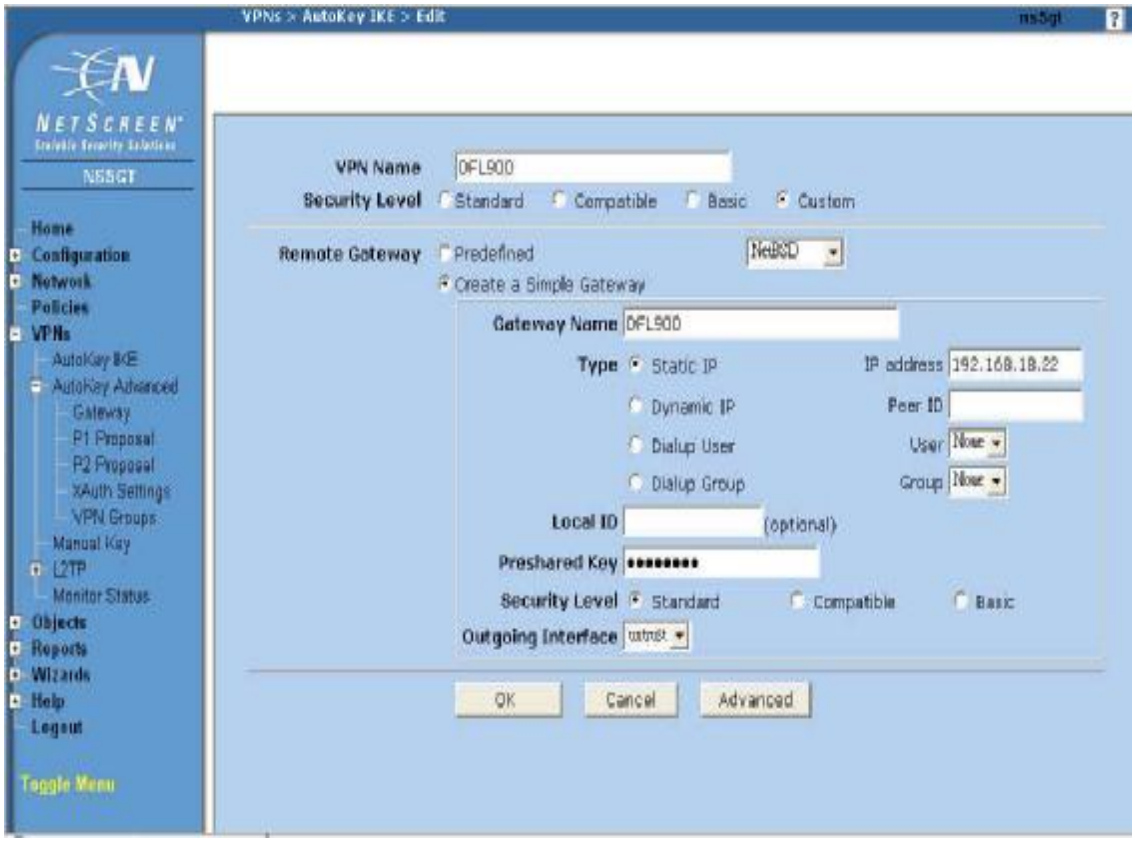
Step 2. 設定細部參數



Step 3. 勾選”Enable IPSec”->Apply

NetScreen NS-5GT 的 IPSec(IKE)設定如下圖: (參考用)

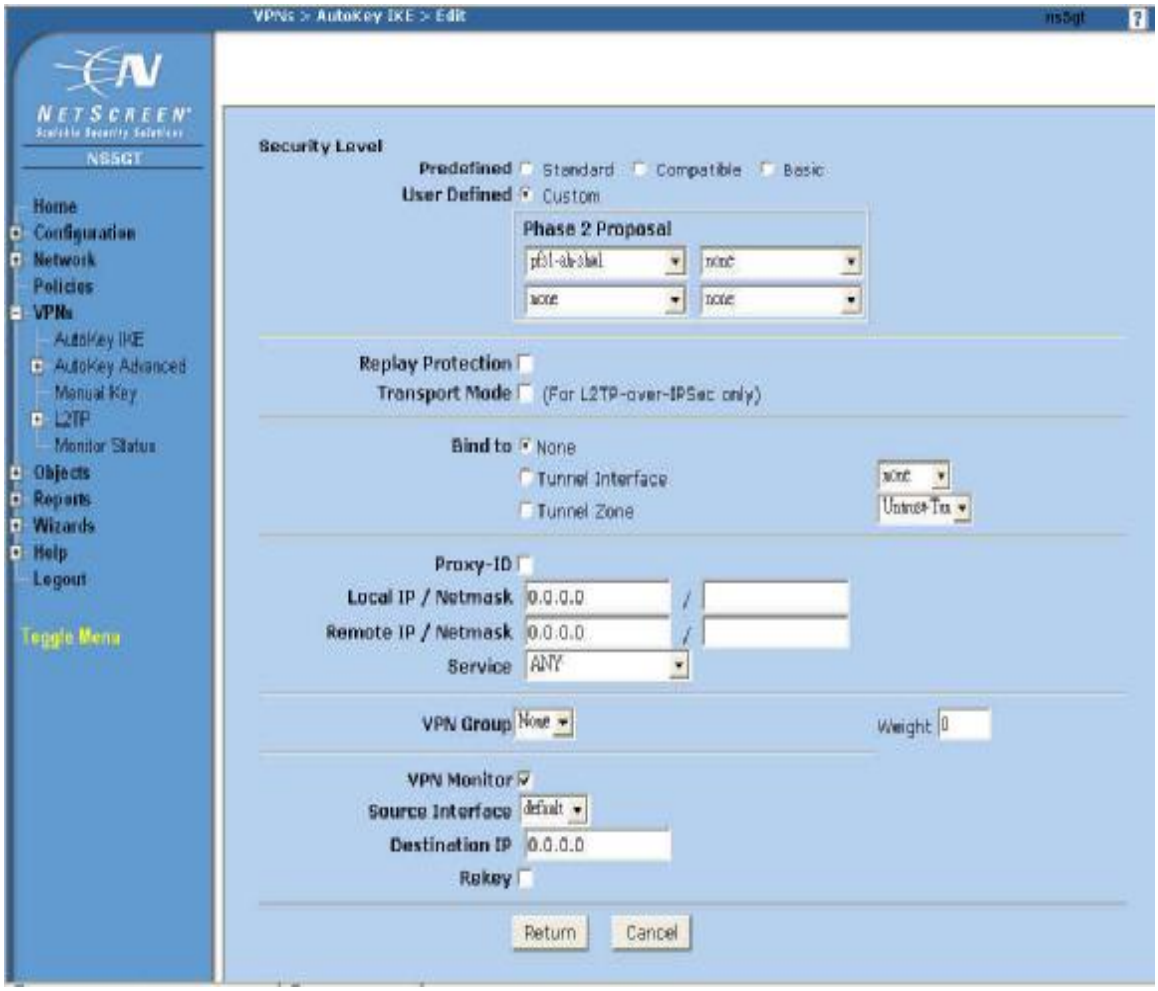
Step 1. 選 VPNs->AutoKey IKE，填入參數，要注意的參數為: Security Level: Custom, Remote Gateway: Create a Simple Gateway，接著選 Advanced。



Step 2. 選 Advanced，填入適當參數:

Phase 2 Proposal: 對應 DFL-900，Bind to: None, VPN Monitor: checked。

設定完畢後，點選 Return 回到 Step 1 的畫面，接著再點選 OK



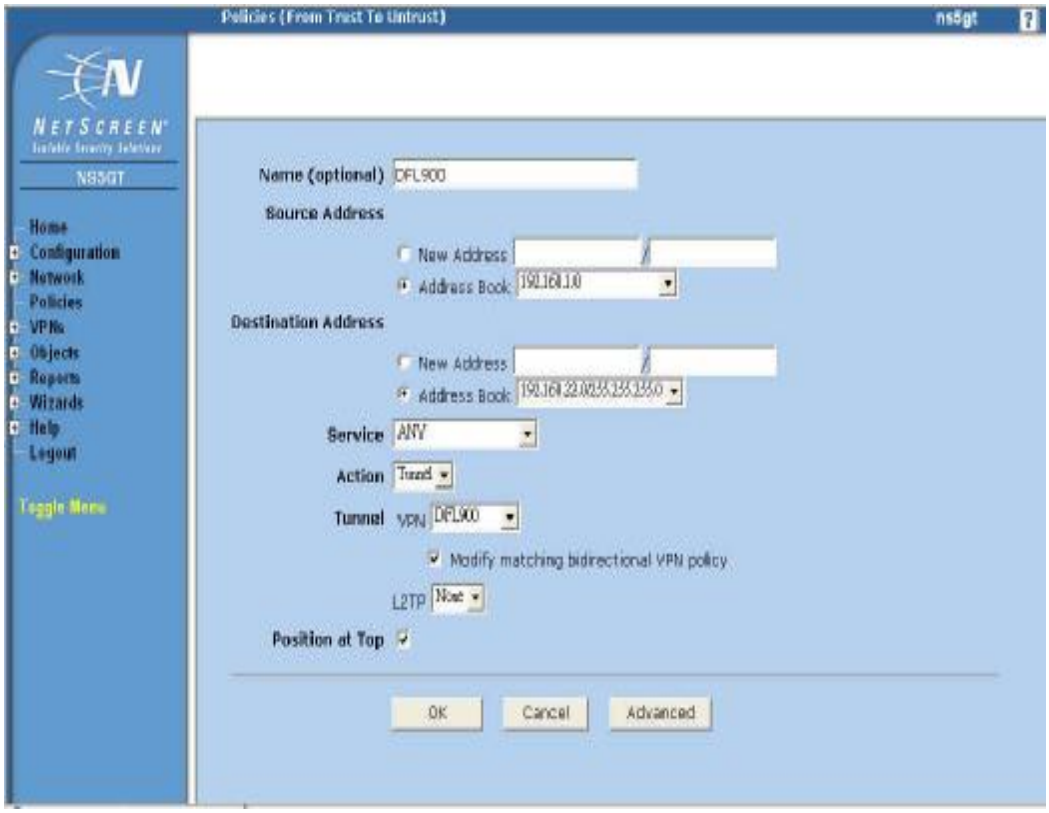
Step 3. 點選 Policies:

From 選擇 Trust, To 選擇 Untrust，接著點選 New 來產生一個新的 policy。

填 Source Address 和 Destination Address 時有兩種方法，一是選 New Address，二是選 Address Book，從現有的 address 裡面選，如果現有的裡面已有了的話就一定要用現有的。

Action 要選 Tunnel，Tunnel VPN 就選之前在 Auto IKE 裡定義過的 DFL900。

設定完畢後選擇 OK



建立 IPSec connection，進行功能性測試

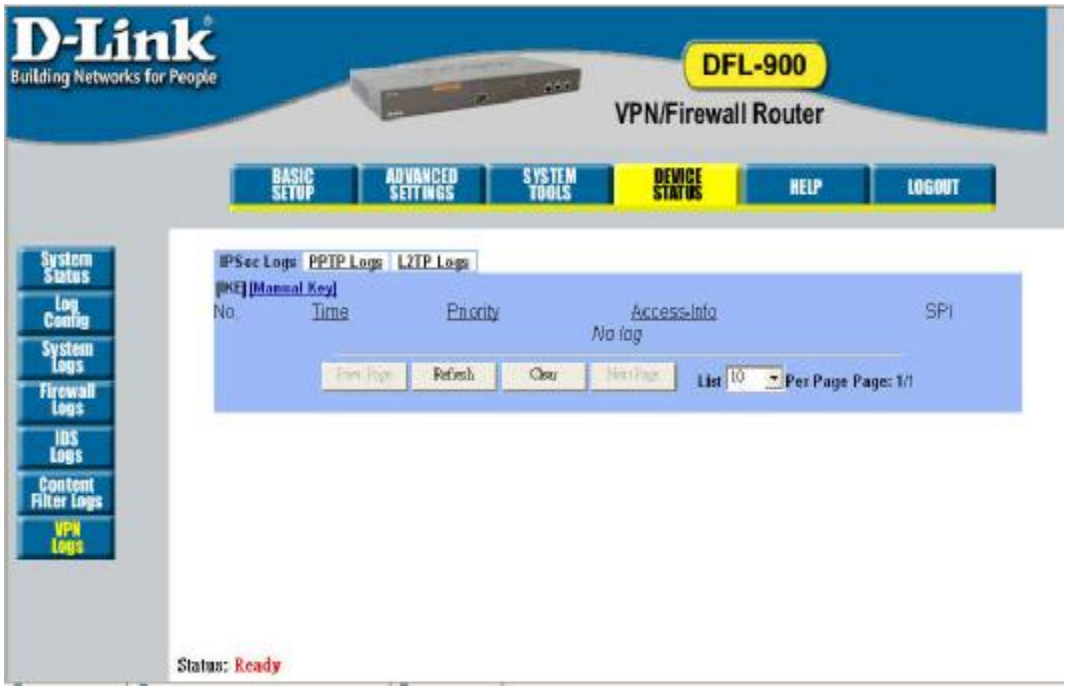
用”ping”來 trigger 並建立起 IPSec connection，用”ftp”在 IPSec connection 中傳送資料，傳送資料時 PC1 和 PC2 都要能夠 upload 和 download。

搜集 log 資訊 【記錄 2】

以 D-Link DFL-900 及 NetScreen NS-5GT 為例:

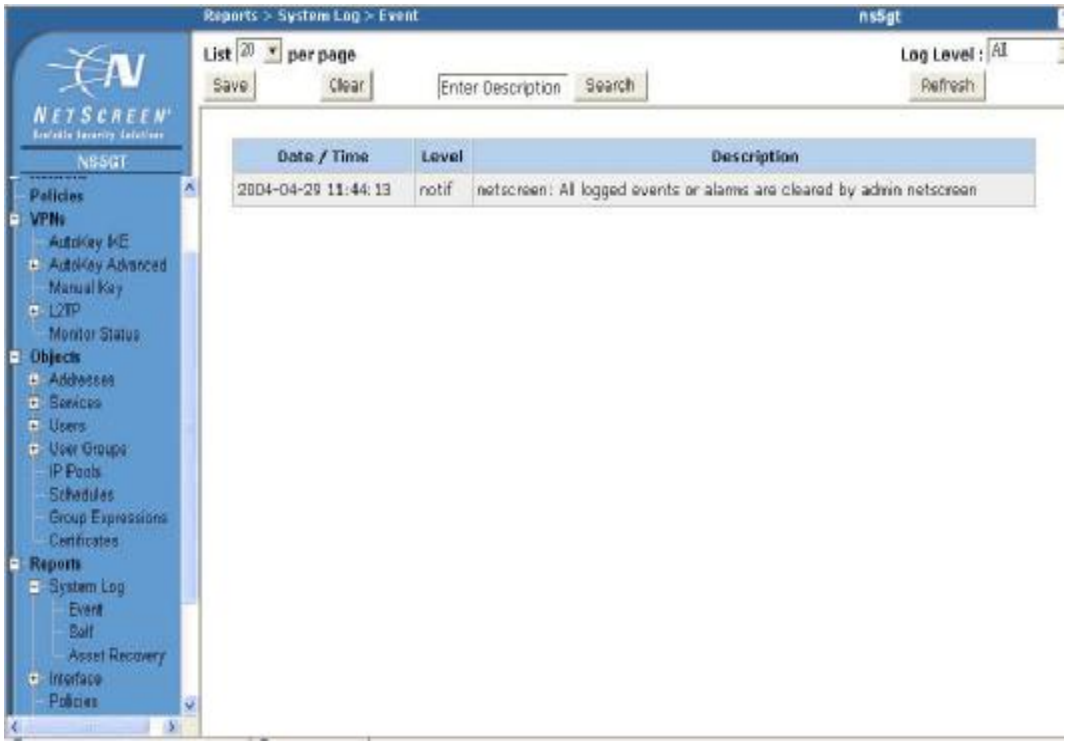
DFL-900 的 IPSec log:

點選 DEVICE STATUS->VPN Logs->IPSec Logs->IKE



NS5GT 的 IPSec log:

點選 Reports->System Log->Event



二、互通性實驗步驟

固定測試參數方法:

請參考”功能性實驗步驟”的”固定測試參數方法”。

設定 DUT 與欲互通之 SG【記錄 4】

請參考”功能性實驗步驟”的”設定 DUT 與 golden sample”，請注意互通之對象不能為 DUT 本身以及 golden sample。

建立 IPSec connection，進行互通性測試

請參考”功能性實驗步驟”的”建立 IPSec connection，進行功能性測試”。

搜集 log 資訊 【記錄 5】

請參考”功能性實驗步驟”的”搜集 log 資訊”。

三、效能性實驗步驟

設定 DUT 【記錄 7】

將兩台 DUT 放在圖 x 的測試環境下，然後設定 DUT，請參考”功能性實驗步驟”的”設定 DUT”。

建立 IPSec connection

用”ping”先建立起兩台 DUT 間的 IPSec connection。

設定 Smartflow 【記錄 8】

承接功能性測試裡，DFL900 和 NS5GT 的例子，當兩者建立起 IPSec connection 後，接著就是進行效能性的測試:

Step 1.

將兩台機器的 LAN 接線改接到 Smartbits 的任兩個 port，例: 2A1(指的是 Smartbits 上的第二層 module 的 A 卡的第一個 port)及 2A2 。

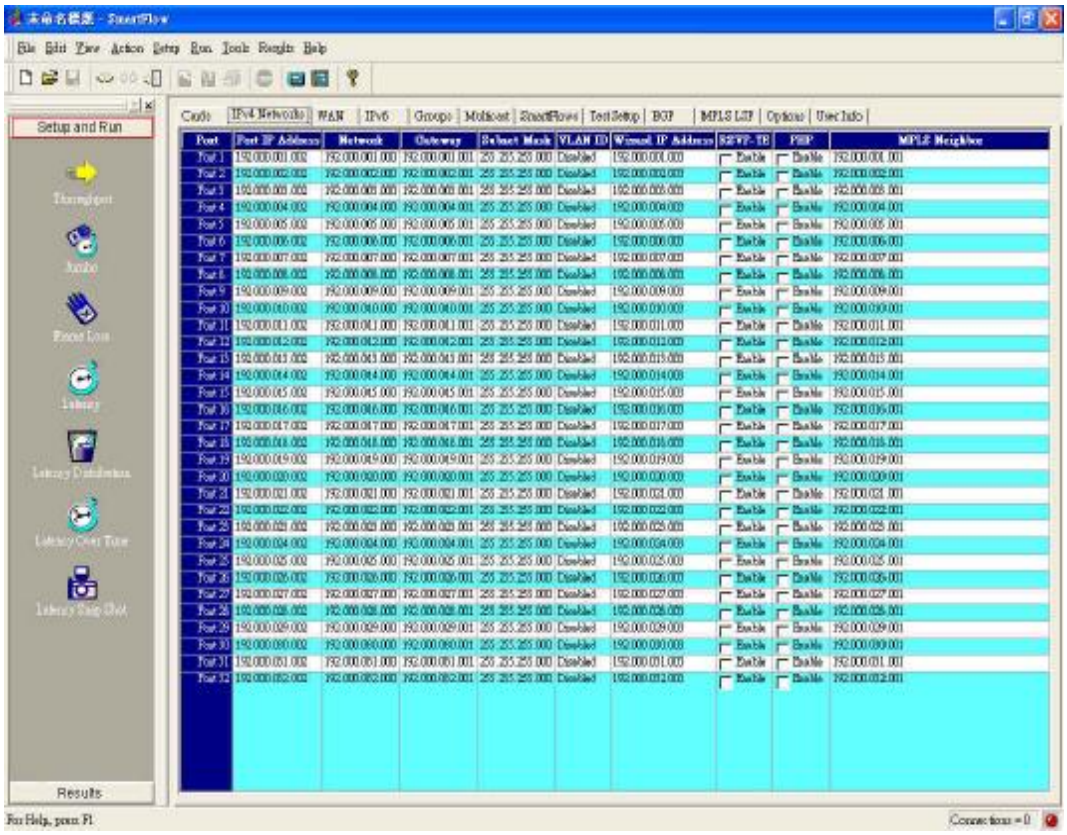
Step 2.

設定 PC 的 IP 和 Smartbits 同一個 IP subnet，讓 PC 可以連到 Smartbits。

Step 3.

在 PC 上執行 Smartflow，將 Smartflow 連上 Smartbits。

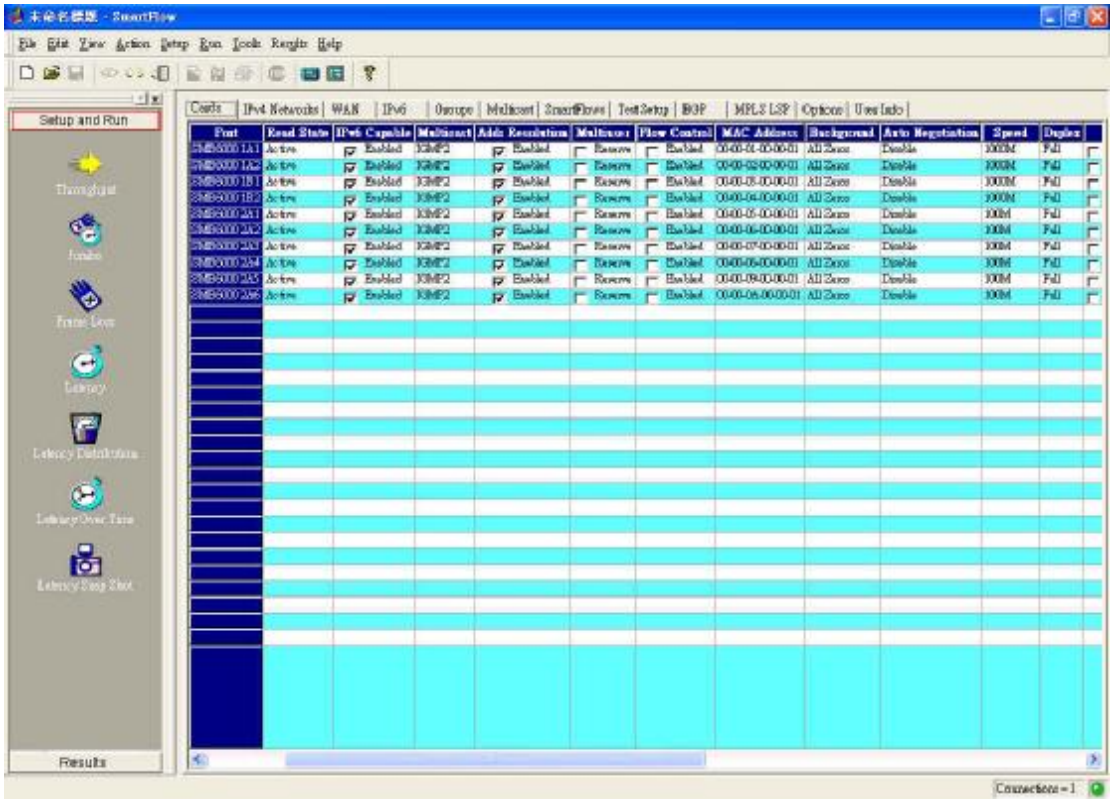
選 Action->Connect



Step 4. 連上後如下圖所示:

Cards 會顯示到所抓到的 port 情況:

這個部分要注意的是，Speed 的部分要配合 DUT，通常都是選 100M。



Step 5. 設定 IPv4 Networks

Port	Port IP Address	Network	Gateway	Subnet Mask	Wizard IP Address
2A1	192.168.1.5	192.168.1.0	192.168.1.254	255.255.255.0	192.168.1.6
2A2	192.168.22.5	192.168.22.0	192.168.22.254	255.255.255.0	192.168.22.6

Step 6. 設定 Groups，定義 smart flow 的流向

選 Groups->Group Wizard，就會跑出一個視窗，視窗裡只需點選 traffic 要從哪一個 port 送到哪一個 port，在此，我們在 A section 點選 2A1，B section 點選 2A2，Direction 就選擇 A->B，接下來的部分都是按下一步，如此就可以產生一個 Flow。

\

Step 7. 設定此 flow 詳細內容要是什麼樣子，選 SmartFlows，在這次的測試項目裡面只需要改到 Traffic->IP's next protocol。

Step 8. 設定這次測試的一些參數，選 Test Setup，在這次的測試項目裡面需要改到 Test Iterations->check “Frame size automation (all flows, w/CRC)”->Custom->裡面填入測試項目所要求的 frame size 種類。

Step 9. 點選 Throughput 測試即開始效能性測試，中途若要停止的話，在 Smartflow 工具列有個紅色 stop 按鈕，按下去即停止測試。

進行效能性測試

將兩台 DUT 放在圖 x 的測試環境下，啟動 Smartbits/Smartflow 產生流量去測試 DUT 的 IPSec 處理效能。

VI. 實驗記錄

【記錄 1】功能性測試時，當某種 IPSec 測試參數可以成功建立 IPsec connection 時，DUT 與”golden sample”的 IPSec 設定。

【記錄 2】功能性測試時，當某種 IPSec 測試參數可以成功建立 IPsec connection 時，DUT 與”golden sample”的 IPSec log 資訊。

【記錄 3】功能性測試時，所有的測試變因其功能是否有正常運作，列一張表格來說明每一項測試的 Pass 或 Fail 的狀況。

【記錄 4】互通性測試時，當某種 IPSec 測試參數可以成功建立 IPsec connection 時，DUT 與其它廠商 SG 的 IPSec 設定。

【記錄 5】互通性測試時，當某一種 IPSec 測試參數可以成功建立 IPsec connection 時，DUT 與其它廠商 SG 的 IPSec log 資訊。

【記錄 6】互通性測試時，DUT 與哪些廠商的 SG 是可以互通的，列一張表格來描述 DUT 的互通性狀況。

【記錄 7】效能性測試時，根據其中一組測試變因下，列出兩台 DUT 的 IPSec 設定。

【記錄 8】效能性測試時，Smartbits/Smartflow 的設定。

【記錄 9】效能性測試時，根據不同的測試變因所得到的測試數據，請列一張表格來描述。

Ⅶ. 問題與討論

一、一個 IPSEC tunnel endpoint 會有幾個 IPSEC SA，一個 IPSEC SA 所代表的意義為何 ？

二、以 IPSEC SA 的角度而言，IKE 和 Manual KEY 差別在哪裡?

三、SA lifetime 的目的為何

四、AH header 和 ESP header 裡都有一個欄位叫作"next header"，為什麼一個在 header (AH) 的開頭，一個在 header (ESP) 的尾端?

五、影響互通性主要的因素有哪些?

六、為何 AES 可以比 3DES 更安全，且在 performance 上又更好?

七、在測 Performance 時有用四種 frame size: 64, 512, 1450, 1518 bytes，請問 Maximum Throughput 是出現在 frame size 1518 bytes 嗎，若不是，請問是出現在那種 frame size 下，而這是什麼原因呢?

Ⅷ. 參考文獻

[1] D. Harkins, D. Carrel, "The Internet Key Exchange", RFC 2409, Nov 1998

[2] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406, November 1998.

[3] S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402, November 1998.

[4] Sample Savolainen, "Internet Key Exchange(IKE)", http://www.niksula.cs.hut.fi/~sjsavola/SoN/essay.html#chap4.3, Nov-22-1999

[5] Cisco System Inc., "Internet Key Exchange Security Protocol", http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113t/113t_3/isakmp.htm#xtocid1053413

[6] Michael S. Borella, "Methods and Protocols for Secure Key Negotiation Using

IKE", 3com Corp., IEEE Network・July/August 2000

IX. 附錄

一、*Oakley*

OAKLEY 是一種改進 Diffie-Hellman 金匙交換法的新演算法。Diffie-Hellman 演算法有以下兩個特性：第一、祕密金匙（Secret Key） 唯有在有需要的情況下才產生。所以祕密金匙並不需要被儲存很久的時間，如此可以增加安全性。第二，在交換的過程中並不需要任何特殊的格式，只需傳送一些共同的參數，但是 Diffie-Hellman 演算法也有一些缺點:

1. 對於交換金匙的雙方並沒有做任何的身份驗證。
2. 不能避免 man-in-the-middle-attack。
3. 計算量非常的大。會遭受一種名為 clogging attack 的攻擊，攻擊者可以對目標系統要求大量的金匙，造成該系統忙於產生不必要的金匙而造成系統負荷大增。

OAKLEY 則保留了 Diffie-Hellman 演算法的所有優點並改進了它的缺點。OAKLEY 有以下的五點特性：

1. OAKLEY 提出了一種名為'cookies'的機制用來避免 clogging attack。
2. OAKLEY 允許交換金匙的雙方協調使用一個'群組'的參數，可以指定不同的 Diffie-Hellman 中的共同參數。
3. 採用了'nonce'的機制來避免重複攻擊(replay-attack)。
4. 允許交換 Diffie-Hellman 中的公開金匙。
5. 對於要交換金匙的雙方可以做身份驗證，避免 man-in-the-middle-attack。

首先討論 clogging attack 的原理。在 clogging attack 中，攻擊者假冒一個合法的使用者，然後將其公開金匙送至所攻擊的系統，被攻擊的系統會做一些指數運算以將其祕密金匙求出。如果攻擊者不斷的送這類型的訊息至被攻擊的系統中，則該被攻擊的系統將浪費許許多多的時間在求解祕密金匙，終將瘋瘓系統，故稱為 clogging attack。OAKLEY 改進了這種缺點，提出了'cockie'的方式。在交換金匙的過程中，欲交換金匙的雙方必須先交換一個稱為'cockie'的亂數，在對方收到這個 cockie 後必須送個回應訊息（acknowledgement）。假設來源位址是假冒的，則攻擊者將沒辦法收到這個回應訊息，如此被攻擊的系統最多只會送些回應訊息出去，而不會從事耗時的指數運算。OAKLEY 中對於 cookie 的產生有以下三點的限制：

1. Cockie 必須與所要通訊的雙方有關。如此可以避免攻擊者使用真的 IP 與 UDP port 以取得一個真的 cockie 後，用其他偽造的 IP 或 UDP port 製造假

的封包。

2. 任何人都無法偽造該 cookie。
3. Cockie 的產生與驗證所需的計算量都非常的小，如此可避免耗費太多的系統資源。

一種產生 Cockie 的方式是將來源與目的地兩方的 IP 位址，UDP Port 以及一個祕密值用 MD5 的演算法產生。

OAKLEY 可以採用不同的'群組' 參數，每種的群組包括了兩個主要參數以及該群組的識別碼，以下是目前有用到的群組：

1. 768-bit 的指數運算
2. 1024-bit 的指數運算
3. 1536-bit 的指數運算
4. 2^{155} 的橢圓演算法
5. 2^{185} 的橢圓演算法

OAKLEY 採用了一種稱為'nonce'的方法來避免 replay-attack。每個 nonce 都是一個臨時產生的亂數，在傳送 nonce 至對方時會先加以加密，如此可以避免該 nonce 被猜出或是被重複使用。

至於身份認証，OAKLEY 提供了三種不同的機制:

1. 數位簽章
2. 公開金匙加密
3. 祕密金匙加密

二、*ISAKMP*

ISAKMP 提供的功能為 Security Association 的管理機制，使用者可透過 ISAKMP 協定建立 SA、交換 SA 的屬性、以及更改、刪除已建立之 SA 等。不過 ISAKMP 僅定義 SA 的管理功能，缺乏以認證過並具機密性的 SA 建立機制，所以引用了 omain of Interpretation (DOI)的使用，例如，建立 IPSec 協定的時，SA ISAKMP 會需要 IPDOI（IPSec DOI）的配合。

身份認證也是網路上安全傳遞所必須的做到的部分，雖然資料有密碼的護，但是如果沒有做認證的程序，可能會遭遇到 Man-in-the-middle 的攻擊，所以，ISAKMP 能夠提供有力的認證保護機制，也包含了數位簽章的功能，所以通訊雙方可以利用 ISAKMP 的訊息傳遞，來交換彼此認為安全公正之認證資料。同時，加密也是安全溝通的必備條件之一，要如何安全、有效地傳遞加密所使用的金鑰，是個相當重要之課題，因此，ISAKMP 也提供金鑰管理的功能，不論使用者環境是使用哪種金鑰交換系統，通訊雙方只要採用彼此協調好的金鑰交換方法，再利用 ISAKMP 的協定提供的資料格式及流程，就能夠安全地傳輸金鑰了。

在 ISAKMP 的保護方面，它採用了 Anti-clogging token 的機制，使得主機的計算

資源受到某一程度的保護，可避免 Denial of Service 的攻擊。同時，ISAKMP 將認證資料與 KEY、SA 的交換相連結，可避免攻擊者在 A 認證過 B 後，阻擋在 AB 間，假造 Key 及 SA 的交換，所以，可避免的 Connection Hijacking 攻擊。更由於 ISAKMP 將所有訊息相連起來的機制，使得某一訊息被攔截、刪除、修改，或插入一訊息的攻擊方式不會成功，也就是說，ISAKMP 也可以抵擋 Man-in-the-middle 的攻擊方式了。

接著介紹 ISAKMP 的完整封包格式:

<table>
<tr><td colspan="5">Initiator Cookie (64 bits)</td></tr>
<tr><td colspan="5">Responder Cookie (64 bits)</td></tr>
<tr><td>Next Payload
(8 bits)</td><td>MjVer
(4 bits)</td><td>MnVer
(4 bits)</td><td>ExchangeType
(8 bits)</td><td>Flags
(8 bits)</td></tr>
<tr><td colspan="5">Message ID (32 bits)</td></tr>
<tr><td colspan="5">Length (32 bits0</td></tr>
</table>

其欄位解釋如下:

Initiator Cookie: 建立 SA，刪除 SA 或更動 SA 時，發送端所送出的 Cookie。

Responder Cookie: 回應 initiator 所使用的 SA。

Next Payload: 第一個 ISAKMP 的 Payload 的種類。

MjVer: ISAKMP 的 Major Version number。

MnVer: ISAKMP 的 Minor Version number。

ExchangeType: 所要交換的內容的種類。

Flags: 供系統號誌之用，可以用來設定 Payload 內容是否需要加密等。

Message ID: 此 ISAKMP 封包的唯一的識別號碼。

Length: 包括 Payload 在內的 ISAKMP 封包長度。

ISAKMP 的 payload 部分也有固定的 header 如下圖所示:

Next Payload: 8 bits	Reserved: 8 bits	Payload Length: 16 bits

Next Payload 欄為 0 表示本身是最後一個 Payload，否則就是下一個 Payload 的種類。Payload Length 為目前 Payload 所包含的位元數。下表列出了 ISAKMP 中所定義的 Payload 種類、參數以及該 Payload 的說明。

Payload 種類	參數	說明
SA	DOI, Situation	最主要的 Payload，用來交換 SA 的資訊，DOI 等

Proposal	Proposal #, Protocol-ID, # of Transforms, SPI	在交換 SA 時用來協調雙方的 Proposal
Transform	Transform #, Transform-ID, SA Attributes	協調彼此支援的Transform 種類
Key Exchange	Key Exchange Data	可以用來使用的金匙交換協定，如 OAKLEY
Identification (ID)	ID Type, ID Data	用來交換彼此的身分識別碼
Certificate (Cert)	Cert encoding, Cert Date	用來交換 Certificate
Certificate Request (CR)	# Cert Types, Cert Types, # Cert Auths, Certificate Authorities	提出 Certificate 的交換請求，包括使用哪一種 Certificate，與可以使用的 CA 等資訊
Hash	Hash Data	Hash function 所產生的資料
Signature	Signature Data	數位簽名函式所產生的資料
Nonce	Nonce Data	Nonce
Notification	DOI， Protocol-ID， SPI Size， Notify Message Type ， SPI, Notification Data	傳送一些其他的資訊，例如錯誤資訊等
Delete	DOI， Protocol-ID， SPI Size，# of SPIs, SPI (one or more)	刪除 SA

ISAKMP 提供了一個平台架構以用來交換各種不同的 Payload 訊息。在 ISAKMP 中共定義了五種的預設交換種類，每個 SA 在交換的過程中都包括了所要使用的 Protocol 與 Transform 的 Payload。下表列出了這些交換的種類:

（a） Basic Exchange
(1) I → R : SA; NONCE 開始 ISAKMP-SA 交換 (2) R → I : SA; NONCE 基本的 SA 認同方式 （ 1 至 2） (3) I → R : KE; IDI; AUTH 產生 Key，回應者確認發起者的身分 (4) R → I : KE; IDR; AUTH 產生 Key，發起者亦確認回應者的身分，雙方建立 SA
（b） Identify Protection Exchange
(1) I → R : SA 開始 ISAKMP-SA 交換

(2) R → I : SA 基本的 SA 認同方式 （ 1 至 2）
(3) I → R : KE; NONCE 產生 Key
(4) R → I : KE; NONCE 產生 Key
(5) *I → R : IDI; AUTH 回應者確認發起者的身分
(6) *R → I : IDR; AUTH 發起者亦確認回應者的身分，雙方建立
（c） Authentication Only Exchange
(1) I → R : SA;NONCE 開始 ISAKMP-SA 交換
(2) R → I : SA;NONCE;IDR;AUTH 基本的 SA 認同方式，發起者確認回應者的身分
(3) I → R : IDI;AUTH 回應者確認發起者的身分，建立 SA
（d） Aggressive Exchange
(1) I → R : SA;KE;NONCE;IDI 開始 ISAKMP-SA 交換
(2) R → I : SA;KE;NONCE;IDR;AUTH 回應者確認發起者的身分，產生 Key，基本的 SA 認同
(3) *I → R : AUTH 發起者確認回應者的身分，建立 SA
（e） Information Exchange
（1）*I → R: N/D 錯誤訊息，刪除 SA， 或是其他狀態

在 Basic Exchange 中，Key 的交換與身份驗證是放在同一個訊息中送出的，如此可以減少封包的數量。在前兩個步驟中，發送者與回應者交換彼此的 Cookie，可以使用的 Protocol 和 Transform 的種類，而訊息中也採用了 NONCE 來避免 Replay-Attack 的可能性。最後兩個訊息用來交換所產生的 Key，並用 AUTH 這種 Payload 來確認傳送彼此的身分資料，Key，及之前所產生的 Nonce。

在 Identify Protection Exchange 中延伸第一種的 Basic Exchange 的方法以做更嚴謹的身份驗證。前兩個訊息建立了 SA，接著兩個訊息交換彼此產生的 Key 與 Nonce，當 Session Key 產生出來後，彼此就將身分資料，數位簽章等用 Session Key 加密後傳送。

在 Authentication Only Exchange 中彼此並不交換 Key，只單純的用來做雙方的身份驗證。前兩個訊息建立 SA，但在第二個訊息中，回應者同時傳送了自己的身分資料並用 AUTH Payload 來保護這些資料。接著發送者也用 AUTH Payload 將自己的身分資料傳送給回應者。

在 Aggressive Exchange 中採用了最少的訊息來交換 Key，但不對身分資料做加密保護。在第一個訊息中，發送者將自己的 SA， Protocol， Transform 等資料傳送至回應者，而在同時也將 Key 與自己的身分資料一併送過去。在第二個訊息中，回應者將 SA， 所要採用的 Protocol 與 Transform 等資訊傳送給發送者，完成 Key 的交換並對發送者的資料做確認。而在第三個訊息中，則是發送者對回應者的身分資料做確認。

最後的 Informational Exchange 則是用於 SA 管理上一些單向的訊息傳送。

三、*IKE 詳細運作流程*

IKE 在資料交流的雙方協商好這次的資料要用哪種加密演算法和認證用的演算法後，便開始進行 Session Key 的產生及交換:

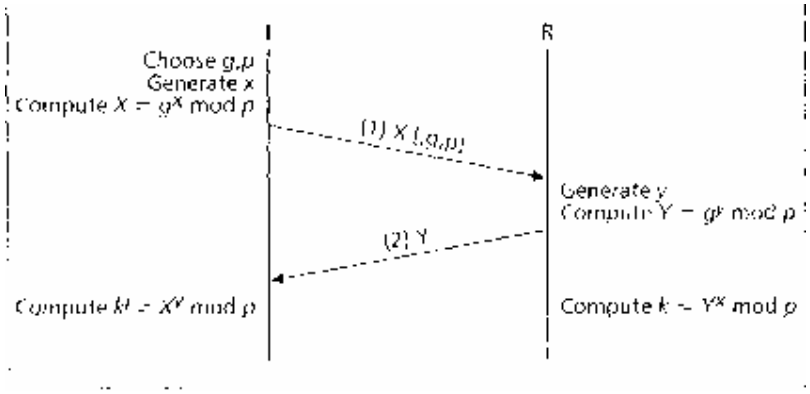


圖 3.1 A basic Diffie-Hellman exchange

如圖 3.1 所示，根據 Diffie-Hellman 的原理，發送方(簡稱 I)先選好兩個數 g、p，p 是個質數，而 g 是小於 p 的 primitive root (也就是 g^1 mod p、g^2 mod p、g^3 mod p 一直到 g^p-1 mod p 都是不同的數)，並在自行選擇一個小於 p 的數 x 後，算出一把 DH half-key X，X=g^x mod p，I 把算好的 X 及 g,p 傳遞給接收方(簡稱 R)後，R 也自行選擇一個小於 p 的數 y 之後，算出另一把 D-H half key Y，Y=g^y mod p，接著把 Y 傳遞給 I 方後，I、R 雙方各自建立了一把 Secret Key K，K=X^y mod p = g^xy mod p = X^y mod p，因此完成了 Diffie Hellman 的金鑰交換。但是，為了防止 denial of service 和 man-in-the-middle 的攻擊，IKE 分別在 Diffle-Hellman exchange 開始前和結束後加入 Cookie 及 ID(identification)的應變措施。首先就 Cookie 方面，對每一次的 SA establishment 而言，Cookie 必須是獨一無二的，因此可避免封包複製攻擊(replay attacks)，於是 I 方為了產生獨一無二的 Cookie，利用 Hash function(如 MD5)把 Source address、Destination address、local secret 和 time stamp 作一番轉換，產生一個 Cookie_i，然後把它傳送給 R 方，R 方在收到後，也利用 Hash function 做了同樣的動作，產生一個 Cookie_r 然後連同 A 方送來的 Cookie_i 一同送回去給 I 方，在完成這兩個動作後，所有接下來的金鑰交換訊息都將包含這個 cookie pair Cooke_i，Cookie_r，

如圖 3.2 所示。

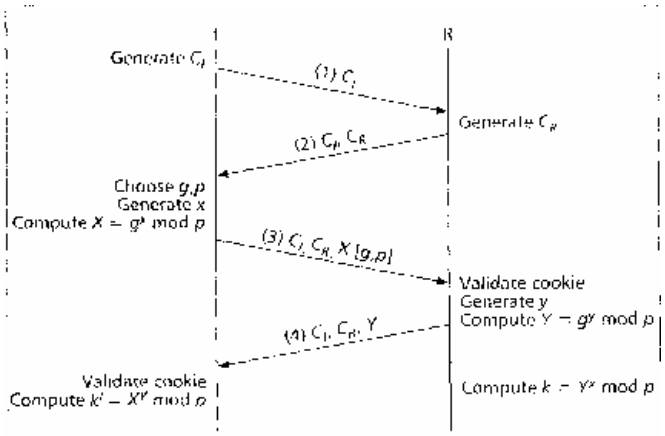


圖 3.2 A basic DH exchange with cookies

另外就 ID 方面，ID 的內容可以是 an IP address，a user name 和 fully qualified domain name，a certificate，or some other token，在 Diffie-Hellman 金鑰交換後，I 方首先數位簽章於自己的 identity，並利用 Diffie-Hellman 交換後得到的 session key k 作加密的動作，形成 ID_i，然後傳送給 R 方，R 方接收到後，進行解密和驗證 ID_i 的動作，然後也數位簽章和加密於自己的 identity，形成 ID_r，送回給 I 方，I 方在解密和驗證完 ID_r 之後，便可達到確認對方是他所宣稱的身分的目的，以防止 man-in-the-middle attack，如圖 3.3 所示。

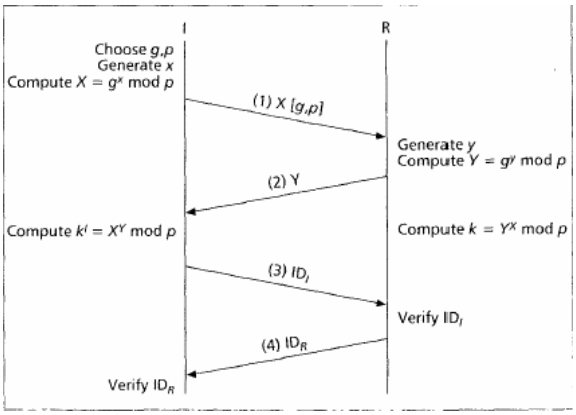


圖 3.3 ID extension to DH

為了達成 ISAKMP SA 與非 ISAKMP SA 協商的目的，IKE 分兩個階段來完成：

IKE Negotiation – Phase 1

第一階段的 IKE 協商使用 Diffie Hellman exchange 和 ID exchange 建立了 ISAKMP SA，進一步的保護第二階段的訊息傳遞，如此一來，之後的資料將可具有隱密性和完整性，而且資料交流的雙方事先經過了認證，確保跟自己交流資料的對方是本人，而不是偽照的。其中，第一階段的 IKE 協商提供兩種模式，Main mode 和 Aggressive mode，每一種模式的目的都是為了 ISAKMP SA 的建立，另外，每一種模式都產生驗證用的金鑰從短暫的 Diffie-Hellman 金鑰交換系統。 ISAKMP

SA 可稱作一種"控制通道(control channel)"，IKE 則可稱為控制通道應用的"控制協定(control protocol)"。

Main mode 包括六個訊息在發送方和回應方之間傳遞，雙方各發出三個訊息，而 Main mode 主要能提供身份的保護，Aggressive mode 只包括三個訊息在發送方與接收方之間傳遞，由於這個模式只用三個訊息便能達到與 Main mode 一樣的目的，因此雖然這模式未能提供身份的保護，但它最主要的好處就是花費比 Main mode 更少的時間。

圖 4.1 所示為第一階段的 IKE 協商的基本模式，下列敘述的四種認證方法都依據這個模式去做不同的認證法. 除了上一段提到的 Cookie、D-H half key 和 ID 之外，又多了三個元件，包括 ISAKMP security associations(簡稱 ISA)、nonces 和 authentication。

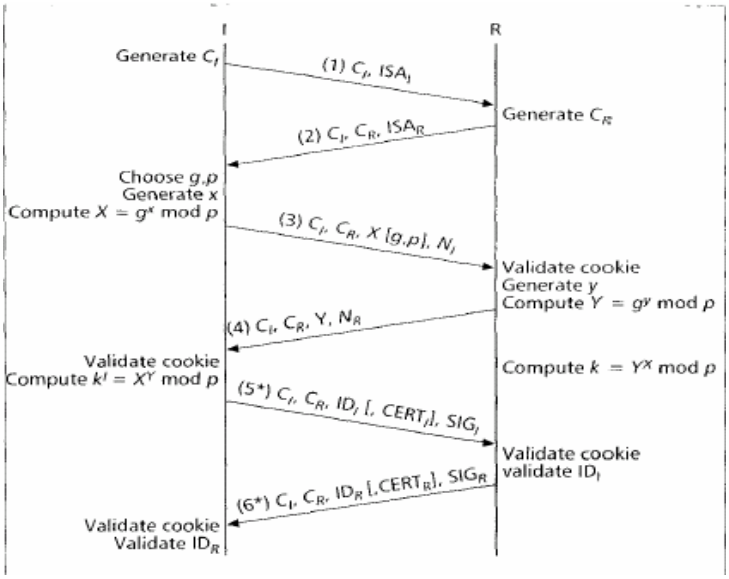


圖 4.1 Phase 1 with ISAKMP security associations, nonces and authentication

在 message(1)中，ISA_i 包括了所有 I 方所提出的 proposal 給 R 方作選擇，proposal 中包含了加密用的演算法(如 DES,3DES)和認證用的演算法(如 MD5,SHA-1)，在 message(2)中，ISA_r 包含了 R 方決定要用的 proposal、加密用演算法和認證用演算法。圖 4.2 舉了個例子說明 ISAKMP SA 的協商。

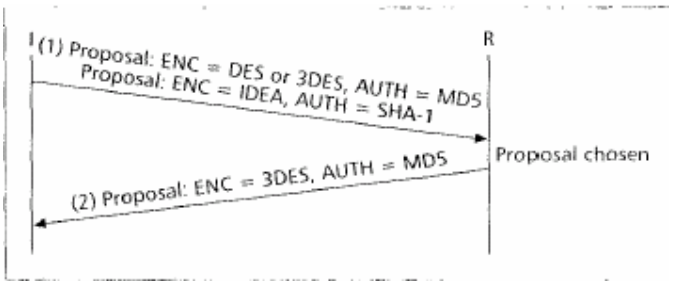


圖 4.2 An example of ISAKMP security association negotiation

在 message(3)和(4)中，發送方和回應方的 nonce 分別為 N_i,N_r，
其中，nonce 是一為數不小的亂數，長度在 64 到 2048 個位元之間，用來擔保 session 的活性和每一個 session 的獨一無二性。message(5)和(6)利用 message(1)和(2)協商好的加密演算法進行加密(圖中星號"*"代表加密的意思)，其中包括雙方的 identity，ID_i 和 ID_r，以及雙方認證用的資料 AUTH_i 和 AUTH_r。

關於金鑰的產生方面，Main mode 和 Aggressive mode 會產生不同的金鑰用來配合加密演算法和雜湊(hashing)演算法作加密和認證用。 數值 SKEYID 和從 D-H exchange 取得的金鑰 k 用來產生三把不同用途的金鑰：

$$SKEYID_d = hashfunc(SKEYID, k|C_I|C_R|0)$$
$$SKEYID_a = hashfunc(SKEYID, SKEYID_d|k|C_I|C_R|1)$$
$$SKEYID_e = hashfunc(SKEYID, SKEYID_a|k|C_I|C_R|2)$$

其中 hashfunc(key,data)是一協商好的雜湊函式用 key 和 data 當作輸入的值，SKEYID 針對每個不同的認證方法有不同的值，SKEYID_d 的目的是用來產生更多的金鑰，SKEYID_a 則是用來當作認證用的金鑰，SKYID_e 是用來加密用的金鑰。另外，認證用的 hash 值用了 SKEYID_a 這把金鑰和協商過程中所有用到的元件當作資料，除了 nonces 之外：

$$HASH_I = hashfunc(SKEYID_a, X|Y|C_I|C_R|ISA_I|ID_I)$$
$$HASH_R = hashfunc(SKEYID_a, Y|X|C_R|C_I|ISA_I|ID_R)$$

關於認證的方法方面，IKE 第一階段中提供了四種不同的認證方法可應用於 Main mode 和 Aggressive mode 中，由雙方共同決定一個認證方法和訊息傳遞的模式，包括 preshared key、digital signature、public key encryption with four encryptions 和 public key encryption with two encryption。

■ Preshared Key Authentication

(1) 利用發送方和接收方都同意的 preshared key 最簡單地做到認證的目的
(2) SKEYID = hashfunc(PSKEY, N_i | N_r)，其中 PSKEY 就是 preshared key
(3) 利用 preshared key 產生 SKEYID 的值後，再用 SKEYID 去產生 SKEYID_a 的金鑰當作上述 HASH_i 和 HASH_r 的輸入金鑰，此時，產生的 HASH_I 和 HASH_r 的值便成了 message(5)和 message(6)中認證用的值。見圖 4.3。

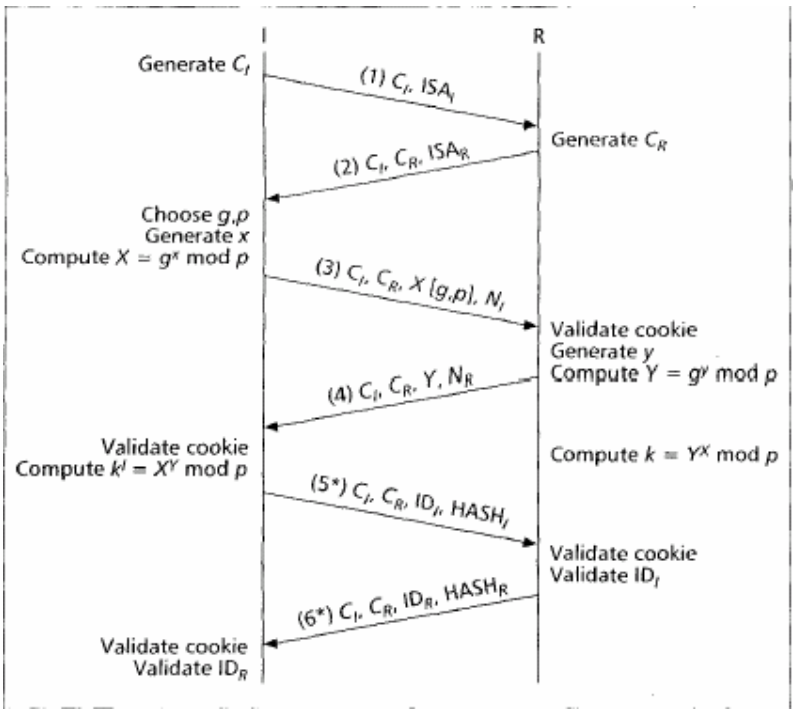


圖 4.3 Secret Key negotiation using a preshared key

- Digital Signature Authentication

(1) 利用 DSS 或 RSA 的技術，公開金鑰和私密金鑰對的方法

(2) SKEYID = hashfunc(N_i | N_r, k)，k (= g^xy)是 D-H exchange 後得到的金鑰

(3) 用雙方各自的私密金鑰對 HASH_i 和 HASH_r 值作簽章變成 SIG_i 和 SIG_r

SIG_i = PRVKEY(HASH_i)

SIG_r = PRVKEY(HASH_r)

傳送給對方後，對方用簽章者相對的公開金鑰對這簽章作認證(反簽章)和解密(用 D-H exchange 所得的 key)的動作

(4) 見圖 4.4，message(5)和 message(6)中，AUTH_i = SIG_i，AUTH_r = SIG_r

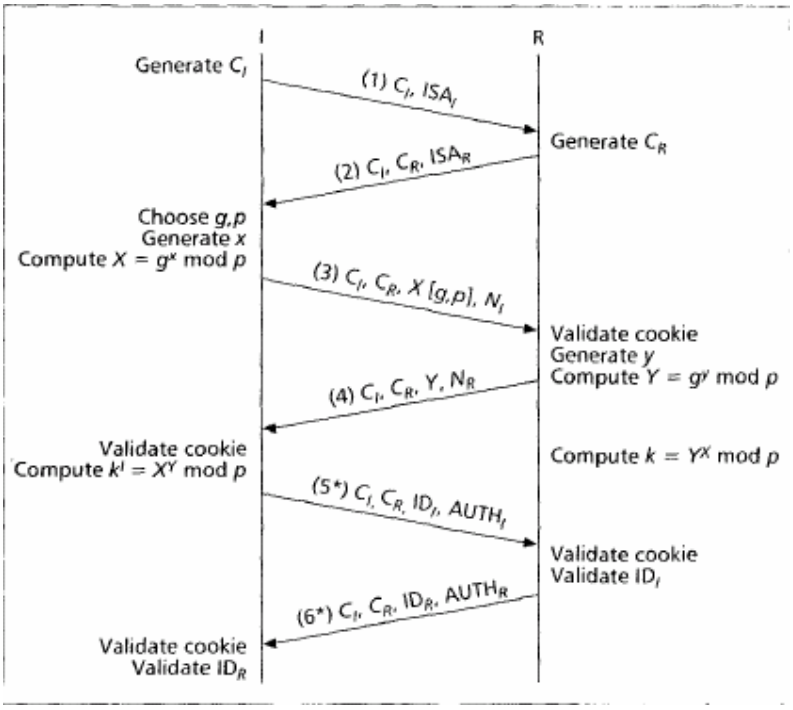


圖 4.4 Secret Key negotiation using a digital signature

- Public Key Encryption Authentication

(1) 見圖 4.5，message(3)中，N_i 和 ID_i 利用 R 方的公開金鑰作加密的動作，PUBKEY_r(N_i), PUBKEY_r(ID_i)，以保護 I 方的身分。

(2) 而假使 R 方有許多的公開金鑰和私密金鑰對的話，當 I 方選定好要用的公開金鑰和私密金鑰對時，在 message(3)中，再多加個 HASH(CERT_r)，HASH 為協商好的雜湊函式，CERT_r 為憑證，包括 R 方的某一把公開金鑰，用來通知 R 方，I 方是用 R 方的哪一把公開金鑰加密以及該用哪一把相對的私密金鑰解密。

(3) message(4)中，R 方同樣的以 I 方的公開金鑰對 N_r 和 ID_r 作加密的動作，以保護自己的身分

(4) SKEYID = hashfunc(hashfunc(N_I | N_r), C_i | C_r)

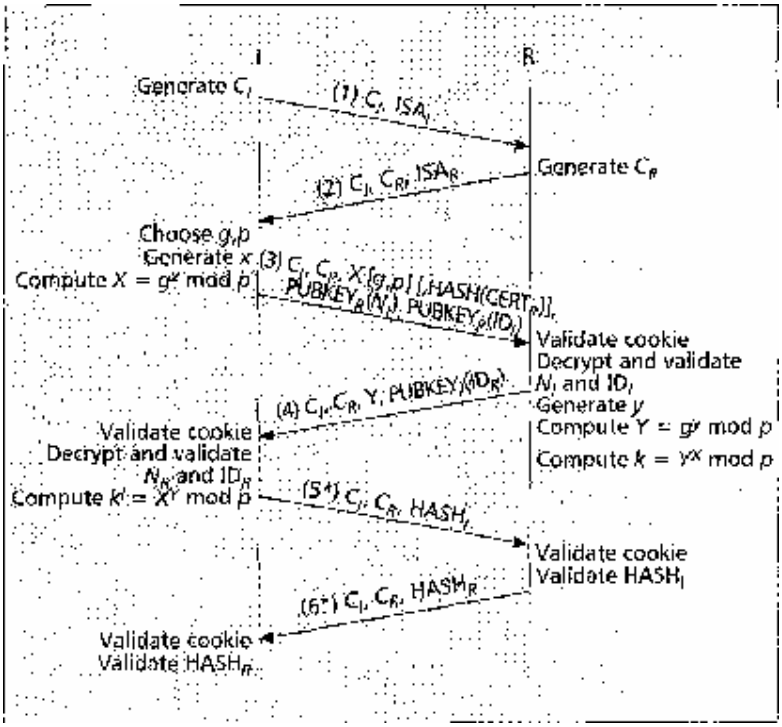


圖 4.5 Secret Key negotiation using public key encryption

■ Aggressive mode negotiation using a preshared key

(1) 為了加快 IKE 的速度，冒著較低安全性的風險，Aggressive mode 只用了三個訊息就完成了金鑰的協商和認證

(2) 見圖 4.6，message(1)包括了 I 方的 cookie, ISAKMP SA 的提議, DH half key, nonce, and identification

(3) message(2)包括了 cookie pair, ISAKMP SA 的回應, DH half key, nonce, identity, and hash

(4) 一直到 R 方接收到 message(3) I 方傳來的 hash 值，認可 I 方的身份後，R 方才求出最後的 D-H exchange key

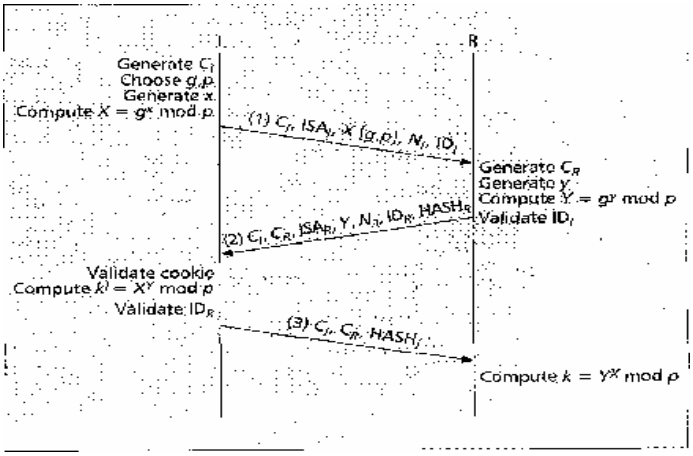


圖 4.6 Aggressive mod negotiation using a preshared key

IKE Negotiation – Phase 2

第二階段的 IKE 協商，在一些安全協定中，為了協商出一個 SA，如 IPsec，和取得新的金鑰。 另外，所有第二階段的封包都會被第一階段協商好的 ISAKMP SA 的加密演算法所加密保護著。其中，Quick mode 和 new group mode 是第二階段 IKE 協商的兩個 mode，這兩個 mode 正是用來達成第二階段的目的，分別用來協商一個非 ISAKMP SA 和取得新的金鑰。

■ Quick mode

(1) 被用來協商 SA 和產生新的金鑰

(2) 在一次的 Quick mode 中，也許會有許多的 SA 被協商好

(3) 不管第一階段是由哪一方開啟的，Quick mode 可以由任一方來開啟

(4) 見圖 4.7，在 message(1)中，I 方傳送了 cookie pair, IPsec SA 的提議, 和 nonce 給 R 方. 假如被要求要 PFS(perfect forward secrecy)的功能，message(1)也會包括 DH half key, g,p 值, 以及 I 方和 R 方的 identification. 另外，HASH(1)是 I 方用來給 R 方認證的雜湊值

HASH(1) = hashfunc(SKEYID_a, M_id | SA_i | N_i) (沒有 PFS)

(其中 M_id 是 ISAKMP 標頭的一部份且被包括在所有的 IKE 封包裡，可證明這個 message 的獨一無二性)

HASH(1) = hashfunc(SKEYID_a, M_id | SA _i | N_i | X | ID_i | ID_r) (有 PFS)

(5) 在 message(2)中，R 方回應了 cookie pair, IPsec SA 的決定, 和自行產生的 nonce. 假如被要求要 PFS 的功能，則 message(2)也包含了 DH half key, g,p 值, 以及 I 方和 R 方的 identification. 另外，HASH(2)是 R 方用來給 I 方認證用的雜湊值.

(6) HASH(2) = hashfunc(SKEYID_a, M_id | SA_r | N_i | N_r) (沒有 PFS) HASH(2) = hashfunc(SKEYID_a, M_id | SA_r | N_i | N_r | Y | ID_i | ID_r) (有 PFS)

(7) 在 message(3)中，I 方在傳送了一次雜湊值給 R 方作認證用 HASH(3) = hashfunc(SKEYID_a, 0 | M_id | N_i | N_r)

(8) 另外，假如 PFS 不用使用的話，新的金鑰 NEWKEY 將被伴隨著新建立的 IPsec SA 來使用且由 SKEYID_d 來產生 NEWKEY = hashfunc(SKEYID_d, protocol | SPI | N_i | N_r)(其中，protocol 和 SPI 是 ISAKMP proposal payload 中的 protocol 和 SPI 欄位)

(9) 假如 PFS 被使用的話，DH exchange 產生的金鑰將被加入，來產生新的金鑰 NEWKE

NEWKEY = hashfunc(SKEYID_d, k | protocol | SPI | N_i | N_r) (k = g^xy)

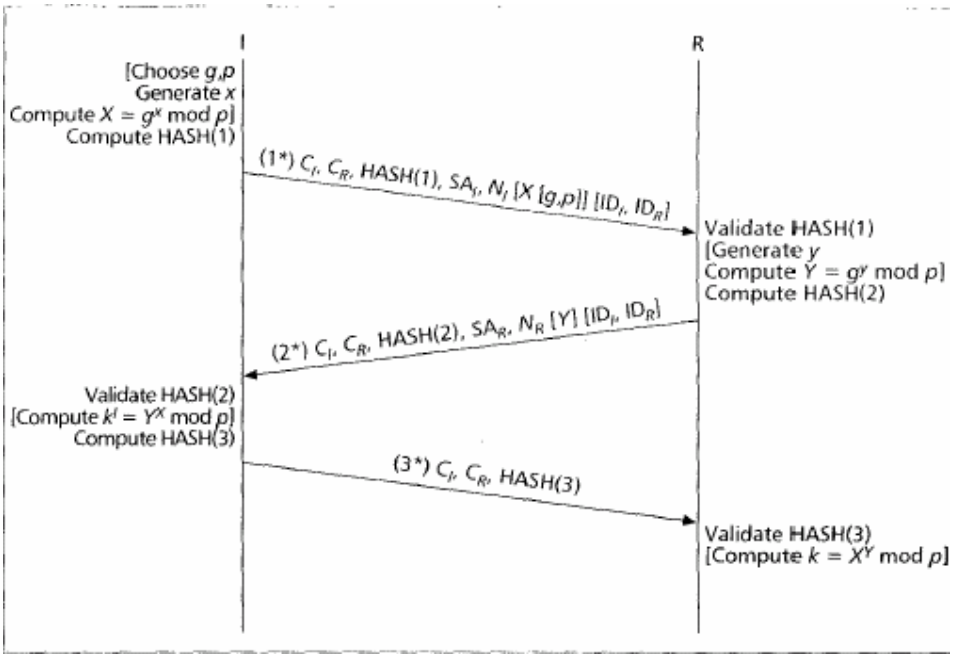


圖 4.7 Phase 2 quick mode

■ New Group Mode

(1) 用來協商一個新的 group(modular exponentiation group MODP or elliptic curve)去做 Diffie-Hellman exchange. MODP 定義了用來計算 Diffie-Hellman 的 group characteristic.

(2) 即使 New Group Mode exchange 不是第二階段的金鑰交換，則一定跟隨在第一階段的金鑰交換之後

(3) 見圖 4.8，在 message(1)中，I 方提議一個新的 group 伴隨著一個 SA payload 給 R 方，HASH(1)是 I 方用來提供給 R 方作認證用 HASH(1) = hashfunc(SKEYID_a, M_id | SA_i)

(4) 在 message(2)中，R 方則回應了 I 方的 new group 的提議伴隨著 R 方自己的 SA payload，HASH(2)是 R 方用來提供給 I 方作認證用 HASH(2) = hashfunc(SKEYID_a, M_id | SA_r)

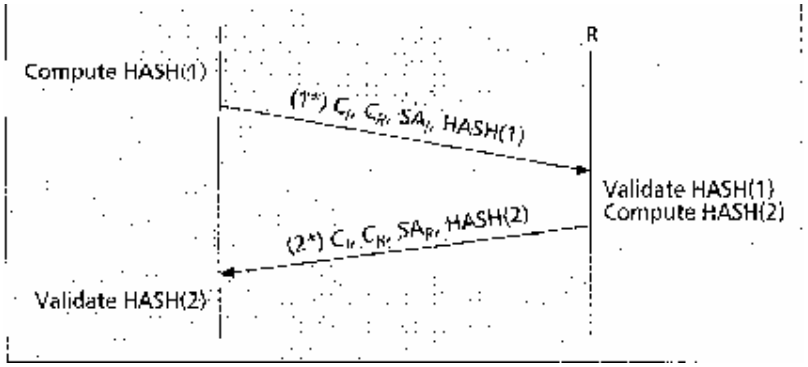


圖 4.8 Phase 2 new group mode