

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Proyecto Fin de Carrera

Implementación y evaluación de las distintas arquitecturas IPSec en sistemas Linux con FreeS/WAN

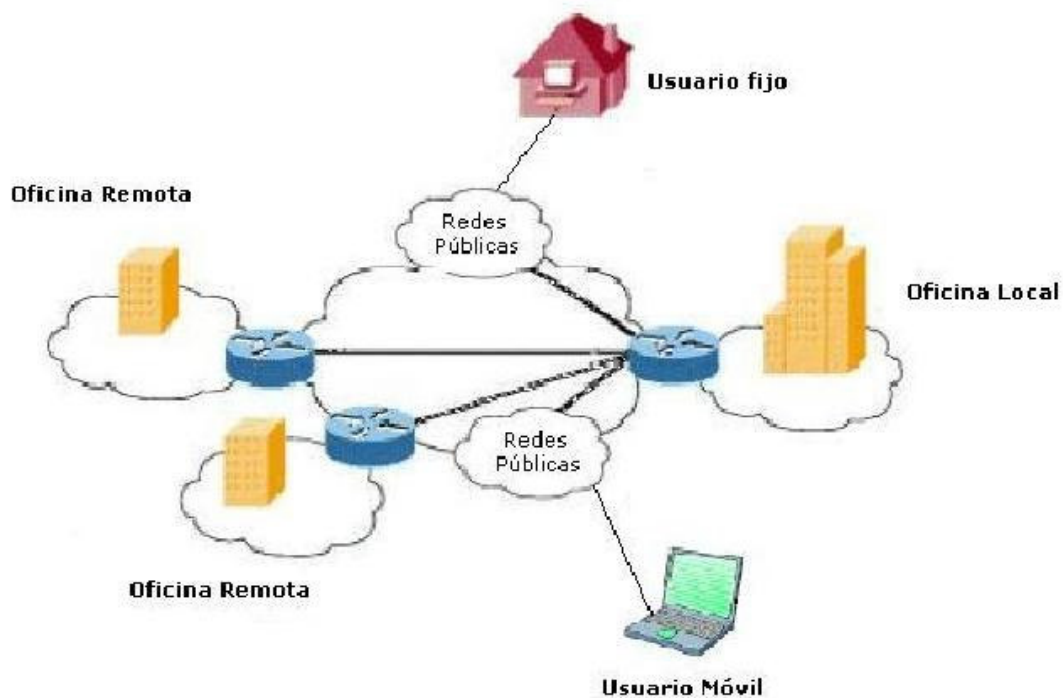


AUTOR: Francisco Javier Picó Odermatt
DIRECTOR: Cristina López Bravo

Diciembre / 2004

Autor	Francisco Javier Picó Odermatt
E-mail del Autor	viejo@ono.com
Director(es)	Cristina López Bravo
E-mail del Director	cristina.lopez@upct.es
Codirector(es)	-
Título del PFC	Implementación y evaluación de las distintas arquitecturas IPSec en sistemas Linux con FreeS/WAN
Descriptores	Red Privada Virtual, IPSec, FreeS/WAN
<p>Resúmen</p> <p>Este proyecto evalúa las posibilidades que ofrece la aplicación FreeS/WAN para construir conexiones seguras a través de la infraestructura de redes públicas, por medio de la arquitectura de seguridad IPSec. En una primera parte, se presenta el concepto de Red Privada Virtual, y se exponen, de manera detallada, los protocolos de comunicación que componen la arquitectura IPSec. La segunda parte se centra en mostrar tres escenarios concretos, como modelos de las configuraciones más representativas de las conexiones IPSec, implementadas con FreeS/WAN. Por último, se muestran las conclusiones que se obtienen de este estudio.</p>	
Titulación	Ingeniería Técnica de Telecomunicación
Intensificación	Especialidad Telemática
Departamento	Tecnología de la Información y las Comunicaciones
Fecha de Presentación	Diciembre - 2004

Implementación y evaluación de las distintas arquitecturas IPSec en sistemas Linux con FreeS/WAN



Índice de contenidos:

Parte I: Conceptos teóricos acerca de redes privadas virtuales e IPSec

Capítulo 1: Resumen de contenidos y objetivos.....	1
Capítulo 2: Introducción	5
2.1. Panorámica sobre las Redes Privadas Virtuales.....	5
2.1.1. Definición	5
2.1.2. Configuraciones.....	7
2.2. IPSec y FreeS/WAN	9
2.2.1. ¿Qué es IPSec?	9
2.2.2. ¿Y FreeS/WAN?.....	10
2.3. Descripción de las herramientas empleadas.....	12
2.3.1. El Servidor de Nombres	12
2.3.2. DHCP	12
2.3.3. <i>Router</i> CISCO	13
2.3.4. OpenSSL.....	13
Capítulo 3: Descripción detallada de la arquitectura IPSec en FreeS/WAN	15
3.1. Protocolos de seguridad empleados.....	15
3.1.1. Asociaciones de seguridad, ISAKMP e IKE.....	15
3.1.1.1. ISAKMP	16
3.1.1.2. IKE.....	17
3.1.2. Protocolo AH.....	20
3.1.3. Protocolo ESP	22
3.2. Cómo se crea una conexión IPSec	24
3.3. Componentes de FreeS/WAN.....	25
3.3.1. Procesos integrados en FreeS/WAN.....	25
3.3.1.1. Pluto.....	25
3.3.1.2. KLIPS	26
3.3.2. Archivos necesarios para FreeS/WAN	27
3.3.2.1. El archivo de configuración <i>ipsec.conf</i>	27
3.3.2.1.1. Sección de configuración	27
3.3.2.1.2. Sección de conexiones.....	28
3.3.2.2. El archivo de claves <i>ipsec.secrets</i>	31
3.3. Funciones complementarias en FreeS/WAN.....	33
3.3.1. La función <i>ipsec ranbits</i>	33
3.3.2. La función <i>ipsec rsasigkey</i>	33
3.3.3. La función <i>ipsec barf</i>	34

3.3.4. La función <i>ipsec showhostkey</i>	34
3.4. Métodos de autenticación del usuario en FreeS/WAN	34
3.4.1. Secretos Compartidos	34
3.4.2. Firmas digitales RSA.....	35
3.4.3. Certificados X.509	36
3.5. Instalación y ampliaciones para FreeS/WAN	38
3.5.1. Instalación.....	38
3.5.2. Ampliaciones	38

Parte II: Configuración de los escenarios

Capítulo 4: Escenario 1: Conexión Directa.....	43
4.1. Introducción	43
4.2. Configuración del escenario	44
4.2.1. Diagramas de situación.....	44
4.2.2. Configuración de equipos paso a paso	45
4.2.2.1. Configuración <i>Routers</i> CISCO 1751.....	45
4.2.2.2. Configuración de las pasarelas de seguridad	49
4.2.2.2.1 Configuración de los parámetros de interconexión.....	50
4.2.2.2.2. Configuración de los archivos de FreeS/WAN	50
4.2.2.2.2.1. Autenticación por secretos compartidos	50
4.2.2.2.2.2. Autenticación por firmas RSA.....	51
4.2.2.2.2.3. Autenticación por certificados X.509	52
4.3. Puesta en marcha de IPSec	55
Capítulo 5: Escenario 2: Road Warrior.....	57
5.1. Introducción	57
5.2. Configuración del escenario	58
5.2.1. Diagrama de situación	58
5.2.2. Configuración de equipos paso a paso	59
5.2.2.1. Configuración <i>Routers</i> CISCO 1751.....	59
5.2.2.2. Configuración de los parámetros de interconexión en <i>PC2</i> y <i>PC4</i>	64
5.2.2.2.1. La pasarela de seguridad: <i>PC4</i>	64
5.2.2.2.2. El cliente móvil: <i>PC2</i>	65
5.2.2.3. Configuración de los archivos de FreeS/WAN	66
5.2.2.3.1. Autenticación por secretos compartidos	67
5.2.2.3.1.1. La pasarela de seguridad: <i>PC4</i>	67
5.2.2.3.1.2. El cliente móvil: <i>PC2</i>	67
5.2.2.3.2. Autenticación por certificados X.509	68
5.2.2.3.3. Autenticación por firmas RSA.....	71
5.2.2.3.3.1. La necesidad de tener un servidor DNS	71

5.2.2.3.3.2. Cambios en la configuración del escenario.....	71
5.2.2.3.3.3. Configuración paso a paso del servidor DNS	73
5.2.2.3.3.4. Configuración complementaria para <i>PC2</i> y <i>PC4</i>	74
5.2.2.3.3.5. Configuración de los archivos de FreeS/WAN	75
5.2.2.3.3.5.1. La pasarela de seguridad: <i>PC4</i>	75
5.2.2.3.3.5.2. El cliente móvil: <i>PC2</i>	76
5.3. Puesta en marcha de IPSec	77
Capítulo 6: Escenario 3: Conexión “Oportunista”	79
6.1. Introducción	79
6.2. Configuración del escenario	81
6.2.1. Diagramas de situación.....	81
6.2.2. Configuración de equipos paso a paso	82
6.2.2.1. Configuración Routers CISCO 1751.....	82
6.2.2.2. Configuración del servidor DNS.....	87
6.2.2.2.1. Parámetros de interconexión.....	87
6.2.2.2.2. Configuración y puesta en marcha del servidor de nombres.....	87
6.2.2.2.3. Configuración de los archivos de FreeS/WAN.	90
6.2.2.3. Configuración de los usuarios: <i>PC2</i> y <i>PC4</i>	91
6.2.2.3.1. Parámetros de interconexión y DNS.....	91
6.2.2.3.2. Archivos de FreeS/WAN.....	92
6.2.2.3.3. <i>Scripts</i> de ayuda a la configuración	94
6.3. Puesta en marcha de IPSec	95
Parte III: Conclusiones	
Capítulo 7: Conclusiones	99
Anexos	
Apéndices.....	105
Apéndice A: Configuración de los <i>Routers</i> CISCO	105
Apéndice B: Capturas de los mensajes del comando <i>ipsec barf</i>	109
Apéndice C: <i>Scripts</i>	111
Referencias.....	115
Bibliografía.....	117
Glosario	119

Parte I:

CONCEPTOS TEÓRICOS ACERCA DE REDES PRIVADAS VIRTUALES E IPSEC

1. Resumen de Contenidos y objetivos
2. Introducción
3. Descripción detallada de la arquitectura IPsec en FreeS/WAN

1. Resumen de contenidos y objetivos

A día de hoy, las comunicaciones a través de las redes de información resultan de vital importancia para un gran número de empresas y organizaciones. Para llegar a su destino, ese tráfico debe atravesar, muy a menudo, una infraestructura de redes públicas (como Internet), lo que lo hace vulnerable a los ataques de usuarios mal intencionados. Ante ese peligro potencial, resulta imprescindible poseer herramientas que permitan proteger el contenido de dicho tráfico, para asegurar tanto su privacidad como su integridad, en las comunicaciones extremo a extremo.

La solución más evidente consiste en montar redes privadas, es decir, para el uso exclusivo de los propietarios de la red, garantizándose la seguridad en las comunicaciones. Sin embargo, esta política de seguridad se está abandonando por dos motivos de peso: el coste y la baja escalabilidad. En efecto, las grandes distancias que separan, en ocasiones, las filiales de una organización, hacen que resulte demasiado cara la construcción de enlaces privados. Por otra parte, el uso de redes privadas supone la implantación de un nuevo enlace cada vez que se quiera unir un nuevo miembro a la red de una organización, con los consiguientes problemas de tiempo y coste.

Como solución más eficiente, aparecen las Redes Privadas Virtuales (VPN), un sistema para construir conexiones seguras a través de la infraestructura de redes públicas, tanto para enlaces punto a punto, como para conectar distintas redes locales entre sí. Este sistema permite aprovechar la infraestructura de comunicaciones existente, aportando los elementos de seguridad necesarios para evitar cualquier intrusión en el contenido del tráfico que protegen. Se consigue así un método de comunicación segura que combina un bajo coste con unos altos niveles de privacidad.

La implementación de VPN suele estar basada en “túneles”: la información que atraviesa las redes públicas se encapsula en paquetes de un protocolo (normalmente IP), de forma que el contenido resulta invisible hasta llegar a su destino, donde se desencapsula el paquete. Las principales arquitecturas de seguridad que emplean túneles son PPTP (*Point to Point Tunneling Protocol*), L2TP (*Layer 2 Tunneling Protocol*), e IPSec (*Internet Protocol Security*). PPTP encapsula el tráfico punto a punto en redes basadas en el protocolo IP. L2TP también protege las conexiones punto a punto, pero puede ser utilizado tanto en redes IP como en redes X.25, *Frame Relay*, o con tecnología ATM. Por su parte, IPSec asegura tanto las conexiones punto a punto como los enlaces entre redes locales, además de ser transparente a la aplicación que lo utilice. Estas dos características son la causa de que nos centremos en la arquitectura IPSec para nuestro estudio.

IPSec ofrece seguridad por medio de una serie de protocolos de comunicación que emplean claves, tanto para codificar la información como para garantizar la identidad de sus usuarios. Cada usuario posee una clave, y deberá conocer la clave del usuario con el que desee establecer una conexión segura. Una vez en comunicación, ambos tendrán que acordar otras claves, de codificación, para proteger los datos.

Para realizar nuestro proyecto hemos elegido la herramienta FreeS/WAN, una implementación libre de IPSec para Linux.

Este documento consta de tres partes; la primera parte presenta los conceptos teóricos relacionados con la arquitectura de seguridad IPSec, implementada con FreeS/WAN. En concreto, se describe el concepto de Red Privada Virtual, así como todos los protocolos y algoritmos de seguridad que utiliza IPSec. También en esta parte, se detallan los elementos de configuración de la aplicación FreeS/WAN.

En la segunda parte, se muestran unos modelos, o escenarios, con la configuración de los tipos de conexión más empleados en FreeS/WAN. Estos modelos resuelven, de manera concreta y detallada, el problema de seguridad que conllevan las conexiones a través de la infraestructura de redes públicas. Mediante la herramienta FreeS/WAN, implementamos la arquitectura de seguridad IPSec, personalizando la configuración en función de las características que deseemos para la conexión segura.

Así, el primer escenario corresponde a la conexión directa, que permite asegurar las comunicaciones entre dos usuarios finales, un usuario y una red local, o bien dos redes de área local. Este tipo de conexión está pensado para dos usuarios (o pasarelas que dan acceso a una red local), alejados entre sí, que conocen sus respectivas claves y direcciones, y que desean establecer un enlace seguro para sus comunicaciones.

El segundo escenario proporciona un modelo de conexión *Road Warrior*, una solución flexible para aquéllos usuarios que necesitan conectar, desde cualquier dirección origen, a un destino fijo. Éste es el caso de los usuarios remotos, que varían constantemente de dirección, y que deben acceder, independientemente de su ubicación, a la red local de su empresa (cuya clave y dirección conocen).

El tercer escenario presenta la conexión “Oportunista”, una alternativa exclusiva de FreeS/WAN, basada en el almacenamiento de las claves de los usuarios IPSec en una base de datos accesible para todos ellos. De este modo, si la clave de un usuario, con el que queremos conectar, está en esa base de datos, podremos establecer una comunicación segura con él, sin necesidad de tener ningún conocimiento previo acerca de dicho usuario.

Por último, en la tercera parte se exponen las conclusiones que se extraen de este trabajo, resumiendo los conceptos más importantes estudiados, y destacando las principales características de los escenarios evaluados.

2. Introducción

En este capítulo introductorio describiremos en primer lugar el concepto de Red Privada Virtual. A continuación, haremos una breve exposición de la arquitectura de seguridad IPSec; nos centraremos, en concreto, en una de sus implementaciones: FreeS/WAN. Para terminar, presentaremos las herramientas externas a IPSec que emplearemos en nuestro estudio.

2.1. Panorámica Sobre las Redes Privadas Virtuales

2.1.1. Definición

Una Red Privada Virtual (en adelante VPN – *Virtual Private Network*) es una red, construida sobre la infraestructura de una red pública, que permite a usuarios remotos comunicarse de forma transparente y segura. Tal y como indica su nombre, es privada dado que garantiza la privacidad en sus comunicaciones, y virtual porque se implementa sobre la infraestructura de redes públicas (por ejemplo, Internet).

Las conexiones VPN permiten a los usuarios acceder, desde casa o desde un punto remoto, al servidor de su organización a través de la infraestructura de encaminamiento que proveen las redes públicas. Desde el punto de vista del usuario, la conexión VPN es una conexión punto a punto entre su ordenador y el servidor de la organización. La naturaleza de las redes intermedias es irrelevante para el usuario, ya que, para él, los datos son enviados como si hubiera un enlace dedicado hasta el servidor. La tecnología VPN también permite a las empresas conectar con sus filiales, o con otras empresas, a través de la infraestructura de redes públicas, ofreciendo comunicaciones seguras. También en este caso, la conexión aparece al usuario como si se estableciera en una red privada.

Para aportar a sus empleados la posibilidad de conectar con sus servidores, una organización debe desplegar una solución escalable de acceso remoto. Normalmente, tienen dos opciones: una solución es crear un departamento de sistemas de información, que se encargue de comprar, instalar y mantener la infraestructura de redes privadas. La otra opción es contratar los servicios de una empresa para que resuelva estos problemas. Ninguna de estas soluciones aporta la escalabilidad suficiente, en términos de costes de despliegue y flexibilidad de administración. Resulta pues interesante reemplazar la infraestructura de red privada por una solución de menor coste, basada en la tecnología Internet. Mediante esta solución, unas pocas conexiones

a través de proveedores de servicios de Internet y servidores VPN pueden satisfacer las necesidades de enlaces remotos de cientos de clientes remotos o empresas alejadas entre sí.

La privacidad se consigue mediante dos mecanismos: la autenticación, que asegura que sólo los usuarios autorizados accedan a un determinado servicio, y el cifrado de datos, que codifica la información enviada de modo que sólo el/los usuario(s) destino puedan descifrarla y por tanto comprenderla. A éstos dos hay que sumar otro aspecto fundamental en la construcción de una VPN: la integridad de los datos, garantía de que la información no pueda ser modificada en su camino hasta el receptor.

Para emular un enlace punto a punto, los datos son encapsulados, y se les añade una cabecera que provee información sobre cómo atravesar las distintas redes para alcanzar su destino.

Comúnmente estos enlaces privados virtuales son denominados “túneles”, ya que el transporte de datos se realiza sin que nadie pueda ver qué es lo que se transmite. Para construir estos túneles entre los usuarios, éstos deben ponerse de acuerdo sobre el esquema de cifrado y autenticación que implementará la VPN.

Para conseguir conexiones seguras, existen unos requisitos que permitan garantizar privacidad, integridad de la información y gestión de las conexiones. Poner en marcha todos estos mecanismos es un proceso complejo, que exige el uso simultáneo de varias tecnologías. Debe permitirse el acceso de clientes remotos a recursos de redes locales, y las oficinas remotas deben de poder compartir información y recursos. Para asegurar la privacidad y la integridad de los datos, son también necesarias tecnologías de autenticación, cifrado y autorización. La solución VPN deberá aportar:

- **Autenticación del usuario:** debe verificarse la identidad de los clientes de la VPN, permitiendo el acceso únicamente a los usuarios autorizados. Se crean unos registros que muestren quién se conectó y por cuánto tiempo.
- **Gestión de direcciones:** deben asignarse direcciones en la red local a los clientes VPN, asegurándose de que no sean visibles fuera de dicha red. También debe darse cierta información a los clientes para que puedan acceder a los recursos protegidos de la red.
- **Cifrado de datos:** los datos que atraviesen las redes públicas deben ser ilegibles para todos, excepto los clientes VPN y su servidor. Esto se consigue mediante tecnologías de cifrado de la información.
- **Gestión de claves:** para cifrar los datos, la solución VPN debe utilizar algún mecanismo de cifrado (basado en claves) que permita crear el túnel. Deben por

tanto generarse, y renovarse cada cierto tiempo, las claves de cifrado, de modo que se mantengan la seguridad y la privacidad en las conexiones VPN.

Así, las VPN constituyen una estupenda combinación entre la seguridad y garantía que ofrecen las costosas redes privadas y el gran alcance, lo asequible y lo escalable del acceso a través de Internet. Esta combinación hace de las Redes Privadas Virtuales una infraestructura confiable y de bajo costo que satisface las necesidades de comunicación de cualquier organización.

2.1.2. Configuraciones

La tecnología VPN permite distintas configuraciones, especializadas para unos casos bien definidos. Estos son:

- **conexión “de igual a igual”** o *peer-to-peer*: se trata de un único enlace que asegura las transmisiones extremo a extremo entre dos usuarios predeterminados.

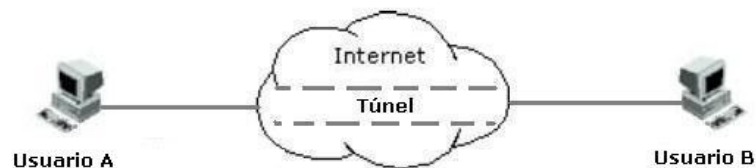


Figura 2.1 – Conexión VPN directa.

- **redes corporativas** o *intranet*: se usa para localizaciones de una misma empresa separadas geográficamente y que necesitan intercambiar datos entre sus redes, acceder a una misma base de datos, aplicación... La VPN permite conexiones confidenciales de una red a otra, creando enlaces confiables a través de las redes públicas, de modo que a la vista de los usuarios de la empresa todo pertenece a una misma red privada.

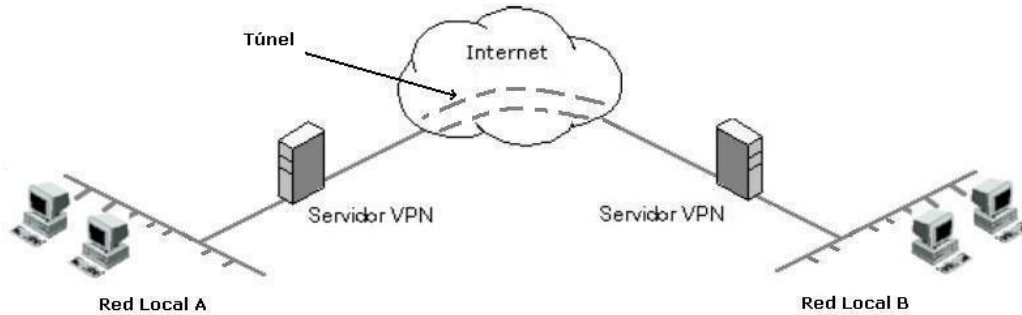


Figura 2.2 – Conexión VPN *intranet*.

- **redes inter-corporativas** o *extranet*: empresas diferentes, pero que trabajan conjuntamente (por ejemplo, proveedor y fabricante), pueden compartir información de manera eficiente y segura entre sus respectivos negocios sin que terceras personas tengan acceso a los datos compartidos. En este caso se interconectan las redes corporativas, aunque sólo se da acceso a un segmento de cada red.

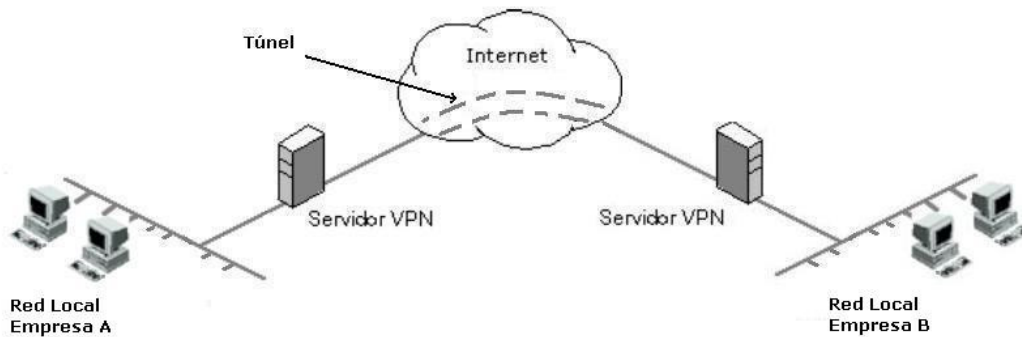


Figura 2.3 – Conexión VPN *extranet*.

- **usuarios móviles** o *Road Warriors* : este concepto se refiere a los usuarios que necesitan acceder a la red de la empresa pero que están constantemente cambiando de ubicación. Con esto se permite al usuario, independientemente de su lugar de conexión, el acceso remoto a la red corporativa, manteniendo las posibilidades de uso como cualquier otro usuario local de la red, y asegurando la privacidad del enlace.

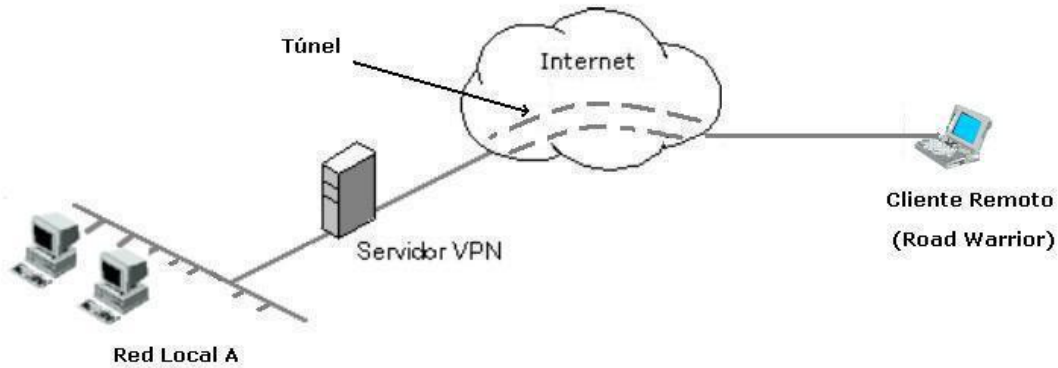


Figura 2.4 – Conexión VPN *Road Warrior*.

2.2. IPSec y FreeS/WAN

2.2.1. ¿Qué es IPSec?

Entre los distintos protocolos que permiten construir una VPN, centraremos nuestra atención en IPSec (*Internet Protocol Security*), uno de los más extendidos; prueba de ello es que la mayoría de los fabricantes de equipos de redes de comunicaciones incluyen soporte para éste. IPSec es en realidad un conjunto de protocolos que construyen un marco de seguridad para IP; es decir una implementación (es un *software* al fin y al cabo) que protege los paquetes IP mediante enlaces seguros extremo a extremo en las comunicaciones. IPSec se sitúa por tanto en el nivel de red de la arquitectura OSI (un nivel intermedio encargado de asegurar la comunicación extremo a extremo).

Como toda tecnología de construcción de VPN, IPSec incluye servicios de autenticación, confidencialidad e integridad, con la particularidad de que, al trabajar en el nivel de red, estos mecanismos son transparentes al usuario y a las aplicaciones que éste utilice (por tanto las aplicaciones no tienen que ser modificadas para hacer compatible el uso de IPSec). Como veremos más adelante en detalle, estos servicios de seguridad añaden unas cabeceras al paquete IP original, cifrando además su contenido en el caso de la confidencialidad.

IPSec ofrece dos modos de funcionamiento:

- **modo transporte**, en el que sólo se cifran los datos y se dejan visibles las cabeceras (incluidas las direcciones origen y destino). El enlace seguro se establece entre los dos extremos de la comunicación, sin intermediarios.

- **modo túnel**, en el que se cifra todo el paquete IP. En este caso, es necesario que al menos uno de los extremos sea una pasarela de seguridad (*Security gateway*). Llegados a este punto, debemos dejar claro el concepto de pasarela de seguridad: se trata de un equipo (un PC o un equipo de red diseñado únicamente para este cometido) que enlaza las redes entre sí, actuando como intermediario. De este modo, para pasar de una red a otra todo el tráfico deberá atravesar esta pasarela. El caso más frecuente lo tenemos en la conexión de una red local a Internet: todos los PCs de la red local acceden a Internet a través de un solo equipo, que puede contener los servicios de seguridad necesarios para evitar ataques al tráfico de datos: la pasarela de seguridad.

Volviendo al modo túnel, la pasarela de seguridad, que implementa IPSec, codifica todo el paquete (cabeceras incluidas), de modo que las direcciones originales no sean visibles (esto evita, por ejemplo, que se envíen paquetes que incluyan en su cabecera una dirección de red local como dirección origen). En su lugar, coloca una cabecera que llevará como dirección origen la suya propia. Como destino, mantendrá el destino original (si el otro extremo implementa IPSec por sí mismo, es decir que no está asociado a ninguna pasarela) o bien lo cambiará por la dirección de la pasarela que da acceso al otro extremo. El túnel se establece así entre las pasarelas de seguridad de ambos extremos, de forma transparente a éstos.

2.2.2. ¿Y FreeS/WAN?

FreeS/WAN (*Free Secure Wide Area Network*) es una de las implementaciones de IPSec para Linux, pero su rápida extensión y flexibilidad han hecho que sea prácticamente la única utilizada en Linux, hasta el punto de que algunas distribuciones como SUSE la incluyen en sus paquetes de *software*. Una de las grandes ventajas de FreeS/WAN es su compatibilidad con otros sistemas operativos, lo que nos permite crear conexiones IPSec a través de redes heterogéneas, y con usuarios remotos que no usan Linux.

Las conexiones realizadas mediante FreeS/WAN se basan en dos parámetros fundamentales: las direcciones de los extremos del túnel, y las claves de cada uno de ellos. Las direcciones son imprescindibles (salvo para el caso “oportunist” que describiremos más adelante en este apartado) para saber dónde se quiere construir el túnel, es decir, cuáles son el origen y el destino de las transmisiones. Mediante las claves que comparten/intercambian los usuarios se consiguen cifrado y autenticado fiables (dado que sólo ellos conocen esas claves, serán los únicos que “comprendan” los mensajes codificados que intercambien).

Podemos clasificar las posibilidades que ofrece FreeS/WAN en tres categorías, según las características de la conexión a realizar:

- **conexión directa**, para construir una conexión IPSec entre dos usuarios conocidos entre sí, con direcciones IP fijas (es decir que cada extremo sabe cómo localizar al otro). FreeS/WAN incorpora además la posibilidad de incluir una subred al lado de cada extremo de la conexión; se construyen túneles para conectar estas dos subredes remotas. Se crea así una VPN, en la que los extremos de la conexión se convierten en pasarelas de seguridad, encargadas de todos los procesos que conlleva IPSec (encapsular paquetes, cifrar, autenticar...), de manera transparente a los usuarios de estas subredes. En este caso, como en todos salvo la conexión “Oportunista”, es necesario que ambos extremos conozcan previamente las claves de autenticación del otro, para que pueda establecerse el túnel IPSec.
- **Road Warrior** (similar al caso descrito en el punto 2.1.2), para dar servicio a los usuarios móviles que quieren un enlace seguro a la red de su empresa, sin importar desde dónde realicen la conexión. En este caso, uno de los extremos será un cliente móvil, y el otro una pasarela de seguridad que hará de intermediario en las conexiones con la subred (situada en el lado de la pasarela). Para esto, es necesario que el cliente móvil conozca dirección y clave de autenticación de la pasarela, mientras que la pasarela sólo necesita la clave de autenticación del cliente móvil (aceptando cualquier dirección origen siempre que venga autenticada). Con estas claves podrá crearse el túnel IPSec.
- **Conexión “Oportunista”** (en inglés *Opportunistic*). Este tipo de conexión es exclusivo de FreeS/WAN, su objetivo es que cualquier usuario pueda realizar conexiones IPSec. Para ello, de entre los posibles métodos de autenticación (ver Apartado 3.4) se deben utilizar las claves RSA. En este caso son los Servidores de Nombres (en adelante DNS) los que administran una base de datos con las claves RSA de todos los usuarios con IPSec. A la hora de conectar con otro usuario (llamémosle “u2”), el usuario “u1” comprueba si la base de datos del DNS contiene alguna clave de “u2”. Concretamente, “u1” consulta a su DNS por defecto, quien podrá consultar con otros DNS hasta obtener una respuesta. Si se localiza una clave para “u2”, el DNS la envía a “u1”, quien la usará para construir el túnel con “u2” (previamente “u2” consultará la clave de “u1” en el DNS para que ambos se “comprendan”).

Este modo es el más ambicioso, ya que el objetivo final es que todos los usuarios de IP puedan conectarse mediante túneles IPSec, sin ningún conocimiento previo (basándose en consultas al DNS). Pese a que los promotores de FreeS/WAN desistieron recientemente en convertir la conexión “Oportunista”

en un estándar de Internet, otros proyectos como *Openswan* o *Strongswan* continúan trabajando en este sentido [1].

2.3. Descripción de las herramientas empleadas

En este apartado realizaremos una breve descripción de los protocolos y herramientas que no pertenecen a la arquitectura IPSec, pero que, por sus características, nos serán de utilidad para su estudio.

2.3.1. El Servidor de Nombres

El Servidor de Nombres (del inglés DNS -*Domain Name System*) es un sistema, organizado en una jerarquía de dominios, para asignar nombres a equipos y servicios de red. Cuando un usuario escribe un nombre DNS en una aplicación, los servicios DNS consultan su base de datos y traducen este nombre a otra información asociada con el mismo, como una dirección numérica. Así, cada vez que usamos un navegador e introducimos una determinada página web, se trata de un nombre DNS, que es “traducido” a su correspondiente dirección IP por nuestro servidor DNS (ya que los equipos de red se reconocen únicamente mediante direcciones IP).

Dentro de las bases de datos de los servidores DNS se puede incluir también otro tipo de información útil; en concreto nos interesa la posibilidad de almacenar las claves públicas RSA de los equipos de red (ver apartado 3.4.2). Gracias a esto, al igual que el navegador consulta la dirección IP de las páginas web, FreeS/WAN puede consultar si existe clave pública RSA para una determinada dirección IP, llevando a cabo conexiones *Opportunistic* siempre que sea posible (es decir, siempre que reciba una clave pública RSA asociada a la IP por la que consultó al servidor DNS).

2.3.2. DHCP

El Protocolo de Configuración Dinámica de Host (DHCP - *Dynamic Host Configuration Protocol*) se usa para asignar direcciones IP a los equipos de una red privada, de manera automática y centralizada. El administrador de red configura el servidor DHCP, encargado de distribuir dinámicamente las direcciones IP a los clientes de la red conforme éstos se conectan. Del mismo modo, cuando un cliente se desconecta la dirección que le fue otorgada queda libre y puede ser reutilizada. Al

servidor DHCP se le asigna un rango de direcciones, que servirá para limitar las direcciones que otorga a sus clientes. Dentro de este rango, DHCP permite reservar algunas direcciones con carácter fijo, para el caso en que algunos equipos importantes deban mantener la misma dirección siempre; las direcciones restantes quedan entonces para uso “dinámico”.

En nuestro estudio esta utilidad es interesante, ya que mediante DHCP podemos tener una dirección IP variable (simulando así el cliente móvil o *Road Warrior*) que nos permita comprobar el correcto funcionamiento de IPSec en estos casos.

2.3.3. Router CISCO

Un *Router* (“encaminador” en castellano) es un equipo de red que conecta redes entre sí, con capacidad para almacenar información que le ayude en el control y el encaminamiento del tráfico de datos. Entre otras muchas opciones, un *Router* puede almacenar unas tablas con las rutas a seguir para llegar a distintos destinos (siempre tendrá una dirección IP como destino final, acompañada de la dirección de la siguiente escala en el camino hacia ese destino final): son las llamadas tablas de encaminamiento.

Para nuestro trabajo hemos utilizado el *Router* CISCO 1751, un equipo completo con un *software* flexible y que permite configurar hasta el más mínimo detalle. De entre su gama de posibilidades, tan sólo hemos utilizado el servidor DHCP y la tabla de rutas para el encaminamiento.

2.3.4. OpenSSL

OpenSSL (*Open Secure Socket Layer*) es el resultado del proyecto de un grupo de expertos con el objetivo de ofrecer una implementación en código abierto (*Open Source*) del protocolo SSL, así como otras librerías de criptografía. SSL proporciona mecanismos para establecer una comunicación segura entre dos extremos de una comunicación, por medio de autenticación, cifrado, y el uso de firmas digitales. Un uso típico de SSL lo encontramos en las compras electrónicas a través de Internet: cliente y servidor web intercambian los datos bancarios protegiéndolos con esta herramienta.

La diferencia principal con IPSec radica en que SSL es un protocolo situado en la capa de Transporte dentro de la arquitectura OSI, por lo que su uso NO es transparente a las aplicaciones, y éstas deberán ser modificadas para darle soporte (no hace falta recordar

que IPSec está por debajo de las aplicaciones y, por tanto, su uso SI es transparente).

SSL aporta servicios complementarios a IPSec, pero su uso está más allá de los objetivos de este proyecto. Sin embargo, existe una herramienta en OpenSSL que será de nuestra utilidad: la creación de certificados digitales X.509 (ver apartado 4.2.2.2.3). Como adelanto, diremos que se trata de un sistema de autenticación que asegura también la integridad en la comunicación, y que nos servirá para probar un modo alternativo de hacer seguras las conexiones IPSec (ver apartado 3.4.3).

3. Descripción detallada de la arquitectura IPsec en FreeS/WAN

A lo largo de este capítulo detallaremos de qué se compone, y cómo funciona IPsec. Comenzaremos por la presentación de los protocolos fundamentales en su arquitectura, seguiremos con la enumeración de elementos necesarios para su puesta en marcha, y finalizaremos analizando las posibilidades que ofrece esta arquitectura.

3.1. Protocolos de seguridad empleados

Entre los múltiples protocolos que emplea IPsec, son fundamentales aquéllos que proveen de mecanismos de seguridad a la arquitectura: estos son IKE, para la negociación de conexiones, y AH y ESP, protocolos encargados de autenticación, confidencialidad e integridad de la información. A continuación, describiremos en profundidad el funcionamiento de estos protocolos.

3.1.1. Asociaciones de Seguridad, ISAKMP e IKE

Una Asociación de Seguridad (SA - *Security Association*) es un acuerdo entre dos o más entidades sobre los mecanismos de seguridad (desde la elección del protocolo de autenticación hasta el intercambio de claves) que emplearán para mantener una comunicación segura. Estas entidades intercambian mensajes para determinar los parámetros que configurarán su enlace seguro; estos parámetros son necesarios para la puesta en marcha de los mecanismos de seguridad que construyen la Asociación de Seguridad.

Esta definición es necesaria para comprender en qué consisten los protocolos IKE (protocolo de Intercambio de Claves, del inglés *Internet Key Exchange*) e ISAKMP (Protocolo de Asociación de Seguridad y Gestión de Claves, del inglés *Internet Security Association and Key Management Protocol*). Ambos han sido diseñados para la puesta en marcha de Asociaciones de Seguridad, y podríamos decir que son complementarios. En efecto, ISAKMP provee un marco para negociar la autenticación y el intercambio de claves, mientras que IKE es un método concreto de intercambio de claves.

3.1.1.1. ISAKMP

ISAKMP define el proceso y el formato de paquetes para establecer, negociar, modificar y eliminar Asociaciones de Seguridad, independientemente del protocolo que se utilice. Esto incluye la creación de mensajes que servirán para negociar la generación de claves y el tipo de autenticación. Su función se limita a crear un marco robusto para negociar conexiones, sin implementar nada respecto al contenido de esa negociación. Centralizando la gestión de Asociaciones de Seguridad, ISAKMP unifica los procesos relativos a los protocolos de seguridad, y permite reducir el tiempo necesario para la configuración de las conexiones.

Las Fases de la negociación

La negociación de ISAKMP tiene lugar en dos fases. En primer lugar las entidades acuerdan el modo de proteger los datos que intercambiarán para negociar, estableciendo una Asociación de Seguridad ISAKMP (en adelante ISKAMP SA). La ISAKMP SA es utilizada para proteger el tráfico de datos de la segunda fase. En la segunda fase se establece una Asociación de Seguridad específica para el protocolo de seguridad elegido por las entidades; esta asociación será la que cree un enlace seguro entre ambos extremos.

Esta división en dos fases aporta una serie de ventajas: primero, que una vez establecida la primera fase pueden negociarse varias SA, evitando tener que realizar el proceso completo para cada nueva SA entre dos usuarios. Y segundo, la negociación de la primera fase aporta algunos servicios de seguridad a la segunda fase (por ejemplo, el cifrado utilizado por la ISAKMP SA sirve como garantía de integridad para los intercambios de la segunda fase).

Formato de la cabecera ISAKMP

8	12	16	24	32 bits
Initiator Cookie				
Responder Cookie				
Next Payload	MjVer	MnVer	Exchange Type	Flags
Message ID				
Length				

Figura 3.1 - Formato de la cabecera ISAKMP.

Descripción de los campos:

Initiator Cookie - Cookie de la entidad que inició el mensaje asociado a una SA (creación, notificación o cancelación).

Responder Cookie - Cookie de la entidad que responde a una petición concreta.

Next Payload - Tipo del campo de datos en el mensaje.

Major Version - La mayor versión de ISAKMP en uso.

Minor Version - La menor versión de ISAKMP en uso.

Exchange Type - Tipo de intercambio de claves utilizado.

Flags - Opciones diversas para el intercambio ISAKMP.

Message ID - Identificador único para el mensaje, utilizado para conocer el estado de la negociación.

Length - Tamaño total del mensaje (cabecera + *payloads*) en octetos.

3.1.1.2. IKE

IKE es en realidad un protocolo híbrido, que mezcla los protocolos Oakley y SKEME (*Secure Key Exchange Mechanism for Internet*), usando el marco que proporciona ISAKMP. Oakley describe un protocolo de generación de claves de sesión basado en el algoritmo Diffie-Hellman, mientras que SKEME aporta una técnica versátil de intercambio de claves.

La estructura y el formato de paquetes son los de ISAKMP, el intercambio de claves se realiza utilizando SKEME, y la clave utilizada en la Fase 1 es generada según Oakley.

Mediante esta combinación de protocolos, IKE proporciona a IPSec el proceso de creación de Asociaciones de Seguridad, elementos vitales para establecer y mantener las conexiones seguras.

Siguiendo la estructura que proporciona ISAKMP, la negociación para establecer la SA se compone de dos fases, que pasamos a detallar a continuación:

Fase 1: Modo Principal (*Main Mode*) – Este modo es la instanciación del proceso de Intercambio con Protección de Identidades (*Identity Protect Exchange*). Se determinan los siguientes procesos para la ISAKMP SA:

- el algoritmo de cifrado: DES, 3DES, 40BITDES o ninguno.
- el algoritmo de integridad: MD5 o SHA.
- el método de autenticación: clave compartida, certificado con clave pública (como RSA), o Kerberos V5.
- intercambio de información para generar la clave compartida utilizada para la ISAKMP SA. Esta clave es generada mediante el algoritmo de claves asimétricas Diffie-Hellman.

El modo está compuesto de un bloque de seis mensajes, divididos en tres envíos y sus respuestas: los dos primeros mensajes son para negociar la política de seguridad entre las entidades, los dos siguientes son para el intercambio de claves públicas del algoritmo Diffie-Hellman (que permitirán a la entidad codificar sus envíos al otro extremo), y los dos últimos sirven para autenticar este intercambio de claves.

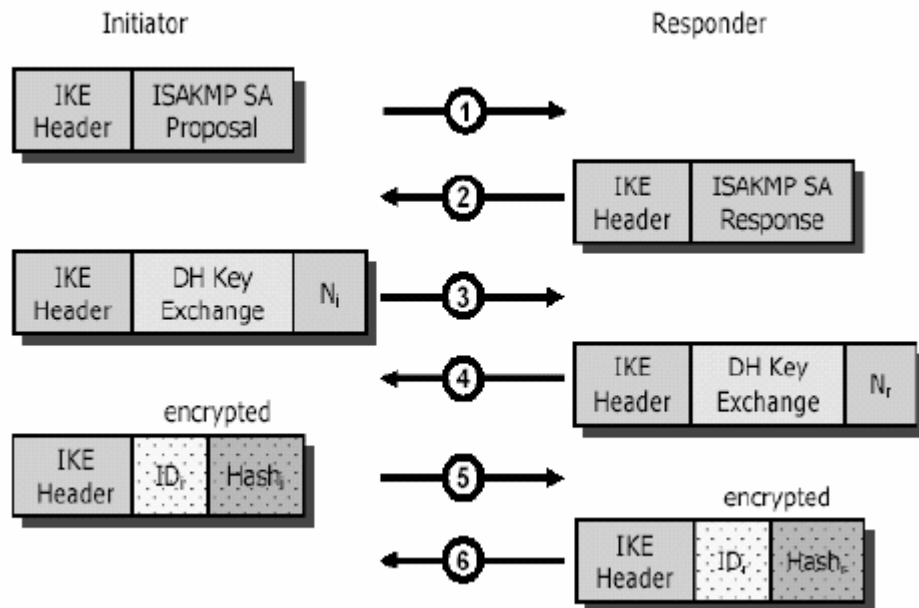


Figura 3.2 - Esquema de mensajes de la Fase 1.

Fase 2: Modo Rápido (*Quick Mode*) – Durante esta fase se negocia y establece la Asociación de Seguridad para el protocolo de seguridad elegido (en nuestro caso será IPsec, por lo que la llamaremos IPsec SA). Esta negociación está protegida por la ISAKMP SA, creada en el Main Mode (fase 1).

Se determina los siguientes parámetros para la IPsec SA:

- el protocolo de autenticación/cifrado: AH o ESP.
- el algoritmo de integridad: MD5 o SHA.
- el algoritmo de cifrado: DES, 3DES, 40BITDES o ninguno.

Con estos parámetros se crean dos IPsec SA, una para cada sentido de comunicación.

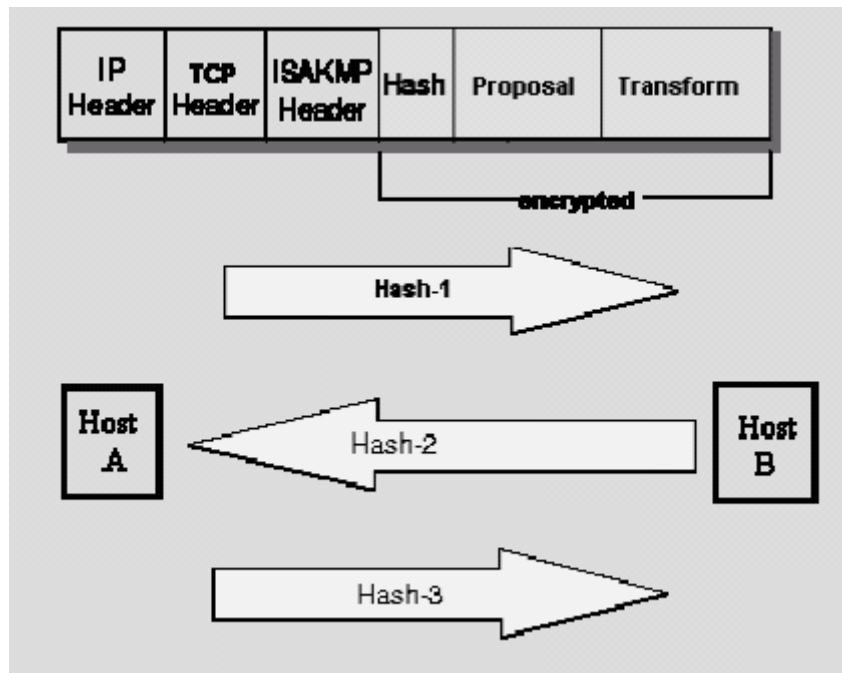


Figura 3.3 - Esquema de mensajes de la Fase 2.

3.1.2. Protocolo AH

La “Cabecera de Autenticación” (AH – *Authentication Header*) es un protocolo diseñado para IP con el objetivo de proveer autenticación, integridad, y protección frente a réplicas a los datagramas IP. Este servicio opcional puede utilizarse cuando se establece una SA.

AH ofrece integridad al campo de datos y a todos los campos fijos de la cabecera IP, sin embargo, esto no es posible para aquellos campos variables en la cabecera (como el *Fragment Offset*, los *Flags* o el *TOS*). Autenticación e integridad se basan en el uso de un *Integrity Check Value* (ICV - Valor de Comprobación de Integridad). Este valor se obtiene empleando el algoritmo elegido por la SA (MD5 o SHA) y utilizando como argumento todos los bytes del datagrama IP que serán autenticados (esto es, todo el datagrama salvo los campos variables de la cabecera). El ICV es almacenado en el campo *Authentication Data*, uno de los campos incluidos en la cabecera AH (cabecera que se añade al datagrama IP original).

En cuanto a los algoritmos posibles, tanto MD5 como SHA están basados en funciones *Hash* (funciones de dispersión): estas funciones toman como entrada un bloque de texto de longitud variable y devuelven un número de X bits (llamado *message digest*), de modo que la probabilidad de que dos bloques de texto distintos tengan el mismo es de 2^{128} (es decir, muy improbable). El receptor hará el mismo cálculo para comprobar que el bloque de texto le da el mismo *message digest*. La diferencia entre estos dos algoritmos reside en el tamaño del *digest*: para MD5 es un número de 128 bits, mientras que para SHA es de 160 bits.

AH provee estos servicios añadiendo una cabecera denominada Cabecera AH (*AH Header*) al paquete IP, situándola justo detrás de la cabecera IP.

8	16	32bits
Next Header	Payload Length	Reserved
Security parameters index (SPI)		
Sequence Number Field		
Authentication data (variable)		

Figura 3.4 - Formato de la cabecera AH.

Descripción de los campos:

Next Header – Identifica el protocolo de la cabecera que sigue a esta cabecera AH.

Payload Length – Longitud de la cabecera AH en octetos.

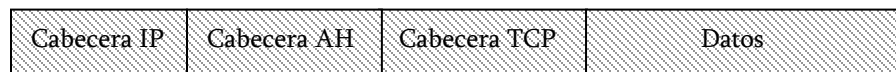
Security Parameter Index – Parámetro de seguridad para identificar a la SA del datagrama.

Sequence Number Field – Contador que se incrementa con cada paquete. Este valor protege frente a réplicas no válidas.

Authentication Data – Contiene el ICV, resultado del algoritmo de autenticación.

Según el modo de conexión IPSec que se utilice, el paquete IP autenticado con AH variará su formato. Si es en modo transporte, la conexión segura se establece directamente entre los extremos, por lo que no se modifica la cabecera IP. En este caso la cabecera AH se sitúa detrás de dicha cabecera. Si es en modo túnel, entran en juego una o dos pasarelas (*gateway*) para crear el enlace seguro, por lo que IPSec añade, al comienzo del paquete, una cabecera con las direcciones de la(s) pasarela(s). En este caso la cabecera AH se sitúa detrás de esta nueva cabecera.

En ambos modos, AH autenticará todo el paquete, salvo los campos variables de la cabecera IP.



Modo Transporte



Modo Túnel



Autenticado (salvo campos variables IP)

Figura 3.5 – Formato de paquetes con cabecera AH, según el modo.

3.1.3. Protocolo ESP

ESP (*Encapsulating Security Payload*) es un protocolo fundamental en la arquitectura IPsec. Al igual que AH, requiere que se haya establecido una SA para ser utilizado. Su función es la de aportar mecanismos de confidencialidad en las comunicaciones, y opcionalmente integridad y autenticación, todo esto mediante el cifrado de datos. Más exactamente, ESP puede emplearse para conseguir confidencialidad de contenidos, integridad, servicio contra réplicas no deseadas, y opcionalmente autenticación.

También es posible limitarse a utilizar ESP para el cifrado de datos (es su uso más común), dejando las conexiones sin autenticación, o bien empleando el protocolo AH para tal cometido. En cualquier caso, siempre es recomendable acompañar el cifrado con algún tipo de autenticación, para evitar posibles ataques al contenido de los paquetes.

ESP está diseñado para utilizar algoritmos de cifrado simétricos, es decir aquellos que funcionan con la misma clave (o claves) para cifrar y descifrar. Dentro de esta categoría, IPsec implementa para ESP el algoritmo 3DES (*Triple Data Encryption Standard*), un algoritmo de cifrado por bloques de 64 bits, que emplea tres claves de 56 bits para cifrar sucesivamente, lo que aumenta su seguridad. En cuanto a la autenticación, ESP ofrece los mismos algoritmos que AH: MD5 y SHA.

El protocolo añade una cabecera, además de una cola (*trailer*) al paquete IP, colocando la cabecera ESP inmediatamente tras la cabecera IP, y la cola ESP justo después del campo de datos del paquete. El datagrama IP con la información de ESP es denominado paquete ESP; éste es su formato:

16	24	32bits
Security association identifier (SPI)		
Sequence Number		
Payload data (variable length)		
Padding (0-255 bytes)		
	Pad Length	Next Header
Authentication Data (variable)		

Figura 3.7 - Formato del paquete ESP.

Descripción de los campos:

Security Association identifier - Parámetro de seguridad para identificar a la SA del datagrama.

Sequence Number - Contador que se incrementa con cada paquete. Este valor protege frente a réplicas no válidas.

Payload Data - Campo de longitud variable que contiene los datos cifrados.

Padding - Bytes de relleno que pueden necesitarse cuando el algoritmo de cifrado requiere bloques completos (es el caso de 3DES).

Pad Length - indica el número de bytes de relleno del campo anterior.

Next Header - identifica el tipo de datos que contiene el campo *Payload Data*.

Authentication Data - contiene el valor del ICV (ver en AH), parámetro resultante del algoritmo de autenticación.

Al igual que para AH, el formato del paquete que contiene ESP variará en función del modo de conexión IPSec que se utilice. Si es en modo transporte, la conexión segura se establece directamente entre los extremos, por lo que no se modifica la cabecera IP. En este caso la cabecera ESP se sitúa detrás de dicha cabecera. Si es en modo túnel, entran en juego una o dos pasarelas (*gateway*) para crear el enlace seguro, por lo que IPSec añade, al comienzo del paquete, una cabecera con las direcciones de la(s) pasarela(s). En este caso la cabecera ESP se sitúa detrás de esta nueva cabecera. Para ambos modos, la cola ESP se coloca justo detrás del campo de datos.

A continuación, mostramos el formato del paquete procesado por ESP, tomando el caso en que se encarga del cifrado y también de la autenticación, para mostrar los campos del paquete a los que atañen estos mecanismos de seguridad.

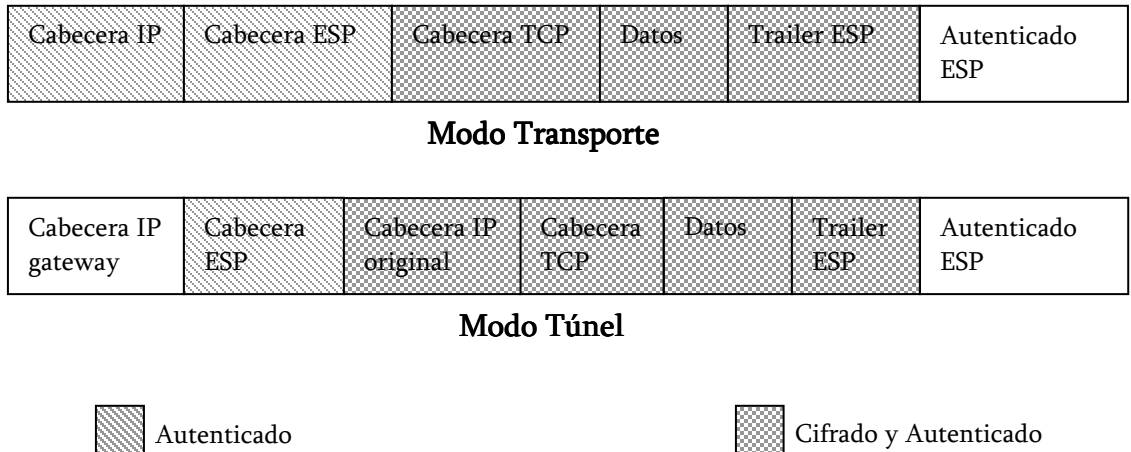


Figura 3.6 – Formato de paquetes cifrados y autenticados con ESP, según el modo.

3.2. Cómo se crea una conexión IPsec

Una vez examinados los protocolos que lo componen, tenemos la base suficiente para entender el proceso que realiza IPsec para que se establezca la conexión segura. Podemos descomponer este proceso en cinco pasos, que detallamos a continuación:

Paso 1: El tráfico adecuado arranca el proceso

Para cada paquete entrante/saliente, los equipos que implementan IPsec consultan sus tablas de encaminamiento. En el caso del tráfico saliente, se comprueba si el destino corresponde con alguno de los destinos para los que debe crearse la conexión IPsec. Si es así, el proceso continúa como se describe en el paso 2; si no, el tráfico es enviado sin ninguna protección.

En el caso del tráfico entrante, se comprueba en primer lugar si se trata de tráfico IPsec. Si es así, se verifica que ese tráfico pertenece a una conexión IPsec abierta, en cuyo caso se descifra. Si se trata de tráfico IPsec que no corresponde a ninguna de las conexiones abiertas, se descarta. Finalmente, si es tráfico externo a IPsec, no se aplica ninguna medida y se pasa directamente a las aplicaciones del usuario.

Paso 2: Establecer la Asociación de Seguridad ISAKMP

Corresponde al “Modo Principal” en la negociación que lleva a cabo IKE, tal y como vimos en el apartado 3.1.1.2. En esta fase se negocian los parámetros de seguridad

necesarios para crear una Asociación de Seguridad ISAKMP, una asociación independiente del protocolo de seguridad que aporta un marco donde poder acordar la conexión segura. Se da por finalizada una vez establecida la *ISAKMP SA*.

Paso 3: Establecer la Asociación de Seguridad IPSec

Este paso corresponde al “Modo Rápido” en la negociación de IKE, como vimos en el apartado 3.1.1.2. Los extremos acuerdan los parámetros para crear dos Asociaciones de Seguridad IPSec, una para cada sentido de la comunicación (lo más importante es acordar el modo de autenticación y de cifrado), bajo la protección de la Asociación de Seguridad ISAKMP. Finaliza con el establecimiento de las *IPSec SA*.

Paso 4: Permitir las comunicaciones seguras

Una vez establecidas las asociaciones, todo tráfico entre los dos extremos de la conexión IPSec es cifrado con ESP, y opcionalmente autenticado con AH o ESP.

Paso 5: Cerrar la conexión IPSec

La conexión puede terminar por dos motivos: porque acaba el tiempo de vida de la Asociación de Seguridad IPSec, o bien porque uno de los dos extremos se desconecta. Si continúa fluyendo tráfico cuando acaba la *IPSec SA*, se vuelve al Paso 3 para renegociar las claves necesarias para una nueva Asociación.

3.3. Componentes de FreeS/WAN

Para llevar a cabo nuestro estudio sobre la arquitectura IPSec, empleamos la versión 1.95 de la herramienta FreeS/WAN, incluida en la distribución 8.0 de SUSE-Linux. A continuación, describimos los procesos y archivos que componen esta aplicación.

3.3.1. Procesos integrados en FreeS/WAN

3.3.1.1. Pluto

Pluto es el “demonio” (del inglés *daemon*) que implementa el protocolo IKE en FreeS/WAN. Su función consiste en:

- Gestionar las ISAKMP SA de la primera fase de negociación IKE.
- Llevar a cabo el control de autenticación de *hosts* y negociar conexiones con otros usuarios de IPSec.

- Crear las IPSec SA y pasar los datos necesarios a KLIPS para su funcionamiento.
- Ajustar las configuraciones de encaminamiento y cortafuegos del usuario, de modo que sean compatibles con IPSec.

Si Pluto es detenido, todas las asociaciones de seguridad creadas por él se cierran inmediatamente.

Para el control de autenticación de *hosts*, el “demonio” consulta el archivo que contiene las claves de autenticación de su equipo, *ipsec.secrets* (describimos su función más adelante, en el apartado 3.3.3.2), para comparar con la clave que recibe del otro extremo de la comunicación. En función del tipo de conexión IPSec, esta comprobación se hará del modo adecuado (ver apartado sobre autenticación, 3.4).

A la hora de iniciar el proceso de creación de una conexión, o de responder a una propuesta de crear una SA, Pluto consulta las conexiones que la configuración establecida contempla proteger; esta información está contenida en el archivo *ipsec.conf* (descrito en el apartado 3.3.3.1). Sólo si la dirección origen/destino está incluida en esta base de datos se da luz verde a la conexión IPSec.

Cuando Pluto debe crear una SA para una conexión, se asegura de que ésta no cause conflicto con otra SA ya establecida. Si esto sucede, se comprueba si la asociación existente tiene una IPSec SA establecida y funcionando, y si es así, la nueva conexión no puede realizarse. Si la IPSec SA no está establecida o está funcionando, la antigua conexión es destruida y se reemplaza por la nueva. Sin embargo, existe una excepción a este comportamiento: si Pluto recibe una propuesta de conexión para enlazar con una subred del equipo que protege, entonces toda SA que cause conflicto con la propuesta es destruida, para dejar sitio a la nueva conexión. Esto es así para dar prioridad a las conexiones de tipo *Road Warrior*.

El uso de Pluto debe combinarse con KLIPS, ya que uno depende del otro para poder llevar a cabo las conexiones IPSec.

3.3.1.2. KLIPS

KLIPS (Soporte al Núcleo para IPSec, en inglés *Kernel IPSec Support*) aporta las modificaciones necesarias para que el núcleo de Linux soporte la arquitectura IPSec. Es el elemento encargado de la gestión de las IPSec SA y de los paquetes IPSec, lo que incluye:

- Realizar los cálculos para obtener y comprobar las claves de autenticación de paquetes.
- Crear las cabeceras ESP (y AH si es empleado) para los paquetes salientes.
- Leer y decodificar las cabeceras ESP (y AH si es empleado) para los paquetes entrantes.

KLIPS también comprueba todos los paquetes externos a IPSec, para asegurarse de que no debe aplicarle las políticas de seguridad (no pueden dejarse paquetes sin proteger por error).

3.3.2. Archivos necesarios para FreeS/WAN

3.3.2.1. El archivo de configuración *ipsec.conf*

El archivo *ipsec.conf* especifica la información de control y configuración que necesita FreeS/WAN. Se trata de un archivo de texto, dividido en dos secciones principales: configuración y conexiones. El formato de cada línea para estas dos secciones es el mismo: **parámetro=valor**. Se puede también hacer referencia a otros archivos (situados en el mismo directorio), para que su contenido se lea como parte de la configuración, nombrando estos archivos, precedidos de la directiva **include**. [2]

3.3.2.1.1. Sección de configuración

La sección de configuración contiene información que FreeS/WAN emplea cuando es arrancado. Mostramos, a modo de ejemplo, el aspecto que tendrá esta sección del archivo:

```

config setup
    interfaces="ipsec0=eth1"
    klipsdebug=none
    plutodebug=all
    pluto load=%search
    pluto start=%search
    
```

La expresión **config setup** indica el comienzo de la sección de configuración. Le siguen una serie de parámetros igualados a un cierto valor, siempre uno por línea y tabulado respecto al alineamiento de **config setup**. A continuación, describimos los parámetros más importantes dentro de esta sección:

interfaces: (Parámetro obligatorio) indica las interfaces de red que usará IPSec. Puede tomar el valor de un identificador de interfaz, o bien el valor **%defaultroute**, en cuyo caso IPSec usará la interfaz apuntada por la ruta por defecto definida en el equipo.

klipsdebug: indica si deben depurarse por pantalla los mensajes que devuelve el proceso KLIPS. Existen dos valores posibles: **none**, en caso de no querer ver la depuración, y **all** si queremos estos detalles en pantalla. Si no se especifica ningún valor se toma por defecto **none**.

plutodebug: indica si deben depurarse por pantalla los mensajes que devuelve el proceso Pluto. Existen dos valores posibles: **none**, en caso de no querer ver la depuración, y **all** si queremos estos detalles en pantalla. Si no se especifica ningún valor se toma por defecto **none**.

plutoload: indica el nombre de las conexiones que deberá cargar Pluto, al arrancar, en su base de datos. También puede tener el valor **%search**, en cuyo caso todas las conexiones definidas con el parámetro **auto=route**, **auto=add** o **auto=start** (ver sección de conexiones) se cargan en dicha base de datos.

plutostart: indica el nombre de las conexiones que Pluto deberá tratar de negociar al arrancar. Si un nombre de conexión aparece aquí pero no está en **plutoload**, entonces es añadido implícitamente a éste último. Si se le asigna el valor **%search**, las conexiones que contengan el parámetro **auto=route** serán encaminadas, y con **auto=start** estas conexiones se crearán de inmediato.

uniqueids: indica si el identificador de cada participante en conexiones IPSec debe ser único. Si le damos el valor **yes**, el identificador para cada participante será único, por lo que toda nueva conexión que utilice el mismo identificador reemplazará la existente. Su valor por defecto es el contrario, **no**.

3.3.2.1.2. Sección de conexiones

Esta sección contiene la especificación de las conexiones (en realidad habrá tantas secciones de conexión como conexiones se describan), define las conexiones de red que podrá proteger IPSec. Cada una de estas conexiones comienza con la directiva **conn**, seguida de un nombre arbitrario que identificará dicha conexión. Le siguen una serie de parámetros con su correspondiente valor, siempre uno por línea y todos tabulados respecto al alineamiento de **conn <nombre>**. A continuación, mostramos un ejemplo de sección de conexión:

```

conn peer-to-peer
  authby=rsasig
  left=192.168.1.1
  leftnexthop=192.168.1.254
  leftsubnet=192.168.7.0/24
  leftrsasigkey= 0sAQPI3gZ9kHqGJjxI...
  right=192.168.2.1
  rightrightnexthop=192.168.2.254
  rightsubnet=192.168.4.0/24
  rightrsasigkey= 0sAQN00tVxASNeTjT/ak...
  auto=start

```

Los parámetros de conexión se dividen en tres categorías: parámetros generales, parámetros para modo manual y parámetros para modo automático. Los parámetros generales son válidos para ambos modos. En modo manual se debe especificar el algoritmo de cifrado a emplear, así como las claves de cifrado de ambos participantes, mientras que en modo automático es Pluto el encargado de generar y renovar las claves de cifrado, utilizando ESP y tomando el algoritmo de cifrado por defecto (algoritmo 3des-md5-96). La generación de claves manual es menos segura, por eso en nuestro estudio emplearemos siempre el modo automático.

Si quisiéramos emplear el modo manual, habría que iniciar cada conexión en ambos participantes, mediante el comando *ipsec manual --up <nombre_conexion>* (en la consola de Linux). En modo automático, FreeS/WAN consulta el parámetro **auto** para cada conexión descrita, y según su valor actuará como se describe en “3-Parámetros Para modo automático”. Con esto se evita tener que iniciar explícitamente cada conexión por consola.

A continuación, detallamos la función de los principales parámetros de la sección de conexiones, divididos por categorías (aquéllos marcados con * deben ser iguales en los archivos de ambos participantes en la conexión):

Parámetros generales

type*: indica el modo de conexión IPSec a realizar, modo túnel (**tunnel**, valor por defecto) o modo transporte (**transport**).

left (right)*: (Parámetro obligatorio) dirección IP del participante de la izquierda (derecha). La distinción entre izquierda y derecha es meramente simbólica, no tiene porqué corresponder con la ubicación geográfica.

leftnexthop (rightnexthop)*: (Parámetro obligatorio) dirección IP de la pasarela que permite al participante de la izquierda (derecha) conectarse a las redes públicas. Su valor por defecto es el que tenga **right (left)**.

leftsubnet (rightsubnet)*: subred privada que se encuentra detrás del participante de la izquierda (derecha). Su formato es *dirección de red/máscara*; su valor por defecto es **left/32 (right/32)**.

Parámetros para modo manual

spi*: número SPI que utilizará la conexión. Debe tener la forma **0x”hex”**, siendo **”hex”** un número hexadecimal. KLIPS sólo admite valores mayores que 0x100, hasta 0xffff.

esp*: algoritmo de cifrado (y autenticado si es utilizado) que empleara el protocolo ESP para proteger los paquetes. Los algoritmos posibles son **3des**, **3des-md5-96**, **3des-sha1-96** y **null-sha1-96**.

espenckey: clave de cifrado del protocolo ESP. Puede especificarse una diferente para cada participante, con **leftespenckey** y **rightespenckey**.

espauthkey: clave de autenticación del protocolo ESP. Puede especificarse una diferente para cada participante, con **leftespauthkey** y **rightespauthkey**.

ah*: algoritmo que empleará el protocolo AH para proteger los paquetes. Los algoritmos posibles son **hmac-md5-96** o **hmac-sha1-96**.

ahkey: clave de autenticación para el protocolo AH. Puede especificarse una diferente para cada participante, con **leftahkey** y **rightahkey**.

Parámetros para modo automático

leftid (rightid)*: nombre que identifica al participante de la izquierda (derecha) para los mecanismos de autenticación de *host*. Puede ser una dirección IP o un nombre cualquiera. Su valor por defecto es el de **left (right)**.

leftsasigkey (rightsasigkey)*: clave pública del participante de la izquierda (derecha) para la autenticación con RSA. También puede contener el valor **%dns**, para las conexiones de tipo “oportunista”, o **%any** para las de tipo *Road Warrior*.

Nota: Si utilizamos FreeS/WAN con el parche para certificados X.509, existe un tercer valor posible: **%cert**, que indica que la clave está contenida en un certificado. En ese caso habrá un nuevo parámetro llamado **leftcert (rightcert)** que especificará el nombre del fichero que contiene el certificado.

auto: indica el tipo de operación a realizar automáticamente cuando se arranque FreeS/WAN, respecto a esta conexión. Sus posibles valores son **add** (añade los parámetros de esta conexión a la base de datos de Pluto, a no ser que ya exista otra conexión con el mismo nombre), **start** (ordena a Pluto establecer esta conexión una vez

arrancado), o **route** (se establece la ruta para esta conexión, pero no la conexión en sí). Para que este parámetro se tenga en cuenta, el nombre de la conexión debe estar incluida en los parámetros de configuración **plutoload** o **plutostart** (o bien que alguno de éstos tenga el valor **%search**).

authby*: indica si el proceso de autenticación de *hosts* se hará mediante secretos compartidos (**secrets**) o mediante firmas RSA (**rsasig**).

auth*: protocolo de autenticación que empleará la conexión, pudiendo elegir entre AH (**ah**) y ESP (**esp**). Este último es su valor por defecto.

keylife: tiempo durante el que serán válidas las claves generadas para autenticación y cifrado. Sus valores válidos son: un número entero seguido de **s** (segundos), un decimal seguido de **m**, **h** o **d** (minutos, horas o días). Normalmente la conexión se renegocia antes de que las claves pierdan su validez.

keyingtries: número de intentos que pueden hacerse para establecer una conexión. Debe introducirse un número entero. El valor **0** equivale a un número infinito de intentos.

3.3.2.2. El archivo de claves *ipsec.secrets*

Este archivo contiene una secuencia de entradas con los secretos o claves que empleará IKE para el control de autenticación llevado a cabo en la primera fase de negociación (ver apartado 3.1.1.2). [2]

Cada entrada del archivo la forman una lista de índices, separada de su correspondiente clave por “:”. Un índice puede ser una dirección IP, un Nombre de Dominio Completamente Cualificado (del inglés *FQDN – Fully Qualified Domain Name*), **%any** o **%any6**. Estos dos últimos valores se emplearán cuando la clave que lo acompañe sea válida para todo *host*, y son equivalentes a utilizar la dirección **0.0.0.0** (la diferencia es que **%any** es para direcciones IPv4, **%any6** para direcciones IPv6, y **0.0.0.0** fue el valor que se propuso como “comodín” en las primeras versiones de FreeS/WAN).

Para autenticar una conexión entre *hosts*, se busca la entrada del archivo *ipsec.secrets* que mejor coincida con los identificadores de ambos *hosts*, y se emplea su clave. Una entrada sin índice será válida para todos los participantes. Si sólo contiene un índice, deberá corresponder al identificador del usuario local. Si contiene dos índices, se buscará correspondencia con los identificadores de ambos participantes. Por último, si es una lista con más de dos índices, los identificadores de los dos participantes deberán estar en la lista.

Ejemplo de archivo *ipsec.secrets*:

```
192.168.2.1 192.168.1.1 : PSK "clave compartida1"
192.168.2.1 192.168.8.1 : PSK "clave compartida2"
192.168.5.1 : RSA {
    #RSA 2048 bits  pc2    Tue Jun 24   12:45:46 2004
    #pubkey=0s<clave pública>
    #IN KEY 0x4200 4 1 <clave pública>
    Modulus: <módulo>
    PublicExponent:0x03
}
```

En función del método de autenticación que se emplee en cada conexión, la entrada correspondiente en el archivo *ipsec.secrets* variará, para mostrar la información relativa a esa clave.

Autenticación por secreto compartido

Ejemplo de entrada en *ipsec.secrets*:

```
<lista de índices> : PSK "<secreto compartido>"
```

La autenticación por secreto compartido (PSK – *PreShared Key*) requiere que ambos participantes tengan el mismo secreto, por lo que se recomienda que se haga una copia para ambos. La clave PSK suele representarse como una cadena de caracteres, delimitada por comillas dobles; es el método de autenticación por defecto. Para información mas detallada sobre PSK, véase el apartado 3.4.1.

Autenticación mediante RSA

Ejemplo de entrada en *ipsec.secrets*:

```
<lista de índices> : RSA {
    #RSA 2048 bits  pc2    Tue Jun 24   12:25:46 2004
    #pubkey=0s<clave pública>
    #IN KEY 0x4200 4 1 <clave pública>
    Modulus: <módulo>
    PublicExponent:0x03
    PrivateExponent:<Exponente privado>
    Prime1:<número primo 1>
    Prime2:<número primo 2>
    Exponent1:<exponente 1>
    Exponent2:<exponente 2>
    Coefficient:<coeficiente>
}
```

La autenticación por firmas RSA requiere que cada usuario posea su propia clave privada. Dado que no es seguro compartir estas claves, las entradas con claves privadas RSA suelen ser de uno o ningún índice (es decir que serán válidas para todo participante remoto). Un *host* puede emplear distintas claves privadas para sus distintas

conexiones. Una clave privada RSA está compuesta de ocho números grandes, añadiendo antes la expresión “0s” cuando estén expresados en base 64. Para información más detallada acerca de RSA, véase el apartado 3.4.2.

Autenticación mediante certificados X.509

Este método de autenticación no está incluido en las versiones oficiales de FreeS/WAN, pero la gran aceptación de estos certificados en Internet ha hecho que varios usuarios se encargaran de crear parches para poder emplearlos en sus conexiones IPSec basadas en FreeS/WAN (en el apartado 3.5.2 se indica dónde encontrar estos parches). Una vez instalado el parche adecuado a nuestra versión, podremos emplear los certificados X.509 indicando lo siguiente en la entrada correspondiente del archivo *ipsec.secrets*:

```
<lista de índices> : RSA <archivo que contenga la clave privada RSA>  
"<secreto compartido opcional>"
```

Para saber cómo funcionan los certificados X.509 consultar el apartado 3.4.3.

3.3.3. Funciones complementarias en FreeS/WAN

3.3.3.1. La función *ipsec ranbits*

Esta función devuelve un número aleatorio de longitud *nbits*, pensado para ser utilizado como clave en los algoritmos que emplean los protocolos AH y ESP. En función del valor de *nbits*, la clave será adecuada para un determinado algoritmo. [2]

La función puede llamarse desde consola introduciendo `ipsec ranbits <nbits>`.

3.3.3.2. La función *ipsec rsasigkey*

Esta función genera un par de claves RSA (clave pública y privada) de longitud *nbits*, teniendo que ser este número múltiplo de 16. El valor del exponente público (*public exponent*) será siempre **3**, ya que este valor agiliza los trámites de comprobación de firmas (para saber más, consultar apartado 4.1.1.2). [2]

La función puede llamarse desde consola introduciendo `ipsec rsasigkey <nbits>`.

3.3.3.3. La función *ipsec barf*

Esta función es de gran utilidad para comprobar el correcto funcionamiento de la creación de las conexiones IPSec. En concreto, lo que hace es mostrar por pantalla la información depurada sobre los sistemas de cifrado y autenticación que emplea IPSec. En definitiva, se trata de un comando que devuelve toda la información relevante para diagnosticar problemas en el funcionamiento de IPSec. [2]

La función puede llamarse desde consola introduciendo `ipsec barf`.

3.3.3.3. La función *ipsec showhostkey*

Esta función devuelve la clave pública del usuario en el que se ejecute, por medio de la lectura del archivo de claves *ipsec.secrets*. En función del parámetro que acompañe (opcionalmente) a la función, devolverá esta clave en un formato concreto. Tenemos las siguientes opciones:

```
ipsec showhostkey
```

Devuelve la clave pública en formato KEY, un formato que será de utilidad para el servidor DNS si realizamos conexiones “Oportunistas” (ver capítulo 6).

```
ipsec showhostkey --txt <dirección IP de la interfaz que usará  
FreeS/WAN>
```

Devuelve la clave pública en formato TXT, un formato que será de utilidad para el servidor DNS si realizamos conexiones “Oportunistas” (ver capítulo 6).

```
ipsec showhostkey --left (o --right)
```

Devuelve la clave pública, precedida de la expresión *leftrsigkey=* (o *rightrsigkey=*), para poder emplearlo directamente en la configuración del archivo *ipsec.conf*. [2]

3.4. Métodos de autenticación del usuario en FreeS/WAN

3.4.1. Secretos Compartidos

Este tipo de autenticación requiere que los participantes en una conexión tengan exactamente el mismo secreto o clave, representado por una serie de caracteres de texto. A la hora de establecer el túnel IPSec, el participante que recibe la propuesta de conexión comprueba que recibe como autenticación del otro extremo la misma clave

que tiene en su archivo *ipsec.secrets*. Sólo si se cumple esto dará luz verde a la creación de la conexión IPsec.

El uso de claves PSK (*PreShared Keys*) tiene dos ventajas importantes. En primer lugar, es bastante sencillo configurar este método de autenticación con FreeS/WAN; y además, es el método más extendido hasta ahora en las distintas implementaciones de IPsec, por lo que tendremos más posibilidades de inter-operar con otras implementaciones si empleamos secretos compartidos.

Los inconvenientes que presenta son evidentes, en especial la poca robustez que presenta este sistema de autenticación: basta con conseguir el bloque de texto que constituye la clave compartida para tener acceso como usuario autorizado. Ligado a este inconveniente, está la necesidad de haber acordado previamente el secreto compartido. Esto puede resultar poco seguro si los usuarios se comunican la clave a través de Internet, y en cualquier otro caso será mucho más lento (correo tradicional, entrevista personal).

3.4.2. Firmas digitales RSA

RSA (*Rivest Shamir Adleman*, los tres creadores de este sistema criptográfico) es en realidad un algoritmo de seguridad que puede emplearse tanto para cifrar datos como para autenticarlos. En el caso de FreeS/WAN, RSA se ofrece como posibilidad para la autenticación del usuario, es decir que se usan las firmas digitales RSA.

RSA es un sistema asimétrico, basado en parejas de claves pública y privada. La clave pública sirve para cifrar los datos, pero sólo con la clave privada correspondiente podrán ser descifrados. En el caso de las firmas digitales, sólo aquél que posea la clave privada podrá firmar los mensajes, mientras que el otro extremo de la comunicación verificará esa firma mediante la clave pública correspondiente. Por eso resulta irrelevante que sea conocida la clave pública de un usuario, ya que con ésta sólo se puede codificar datos o verificar firmas. De hecho, para que dos usuarios puedan usar RSA, es necesario que conozcan previamente la clave pública del otro, mientras que sus claves privadas permanecerán secretas y a buen recaudo.

Veamos cómo funciona el algoritmo: en un principio se toman dos *números primos* muy grandes p y q , cuyo producto es llamado *módulo*: $n=pq$. Se busca un número e , menor que n y tal que sea relativamente primo a $(p-1)$ y $(q-1)$, es decir que e no tenga factores comunes con $(p-1)$ ni $(q-1)$ - salvo el 1 - ; e es llamado *exponente público*. Se busca también un número d , llamado *exponente privado*, tal que $(ed-1)$ sea divisible

por $(p-1)(q-1)$. La *clave pública* la forman los números (n,e) , mientras que la *clave privada* será la pareja de números (n,d) .

Para firmar digitalmente un mensaje, el participante que envía la propuesta de conexión (llamémosle A) realiza una serie de cálculos utilizando como argumentos su clave privada y el mensaje que envía. Junto al mensaje, envía el resultado de estos cálculos, que llamaremos r , al destino (B). Al recibir el mensaje junto con r , B utiliza la clave pública de A para realizar los cálculos inversos, de modo que, si todo es correcto, a partir de r obtendrá el mensaje original. Es decir, sólo si el mensaje lo firmó quien debía hacerlo, obtendrá como resultado del proceso el mismo mensaje que ha recibido junto con r .

Se comprueba así de manera eficiente la identidad de los extremos de la negociación de conexión: sólo si la firma de los mensajes es correcta podrá continuar el proceso de creación de la SA.

Como para los secretos compartidos, el inconveniente de este sistema de autenticación es que los extremos de la comunicación deben conocer la clave pública del otro antes de comenzar la negociación, pero en este caso es menos importante, ya que las claves públicas no permiten descifrar ni firmar, y pueden por tanto intercambiarse sin ningún miedo. Además, la opción de conexión Oportunista (ver capítulo 6) evita este requisito construyendo una base de datos de claves públicas en los servidores DNS. No hace falta decir que este método es mucho más robusto que los secretos compartidos, hasta ahora permanece como método indescifrable, y es muy usado en Internet.

3.4.3. Certificados X.509

Se trata de un estándar definido por la UIT (Unión Internacional de Telecomunicaciones) para firmas digitales. El uso de certificados llega como solución al gran problema de los sistemas de claves asimétricas: ¿cómo sabe el usuario que la clave pública que recibe proviene realmente de quién dice provenir?

Un ejemplo: dos usuarios (A y B) que utilizan claves RSA desean crear una conexión IPsec para asegurar sus comunicaciones. Cada uno desconoce la clave pública del otro, así que las intercambian previamente. Sin embargo, un tercer usuario mal intencionado (C) intercepta el envío de la clave pública de A a B , y en su lugar envía a B su propia clave pública. B sólo sabe que ha recibido una clave pública (creyendo que es la de A), por lo que la utilizará para cifrar datos destinados a A y para comprobar las firmas digitales de los mensajes que reciba. De esta forma, C suplanta la identidad de A y podrá descifrar todo mensaje enviado desde B hacia A .

La solución está en emplear una tercera entidad de confianza, llamada Autoridad de Certificación (en adelante CA – *Certification Authority*), que asegura la identidad de las claves públicas. Así, cuando un usuario crea su clave privada, crea al mismo tiempo una petición de certificación de esta clave. La petición es enviada a una CA de confianza para el usuario, y ésta devolverá un certificado de la clave pública. El certificado contiene el nombre del usuario, la clave pública, y una firma digital de la CA que garantiza la autenticidad de la clave pública. De este modo, a la hora de establecer conexiones seguras, los usuarios intercambian los certificados de claves públicas. Cada uno puede entonces consultar con su CA (las CA tienen una estructura jerárquica para las consultas de autenticidad, de manera similar a la estructura de servidores DNS) para comprobar la firma del certificado recibido, y saber así que puede confiar en la clave pública incluida en éste. En otras palabras: si la firma que lleva el certificado es de confianza para nuestra CA, entonces podemos fiarnos de que la clave pública es de quien dice ser.

Contenido del certificado X.509 versión 3

Versión: versión del protocolo X.509.

Número de serie: identificador único del certificado, asignado por la CA.

Identificador del Algoritmo: algoritmo empleado para firmar el certificado.

Autoridad de certificación: nombre de la CA que firma el certificado.

Fechas de inicio y fin: el certificado tendrá validez entre esas dos fechas.

Usuario: nombre del usuario del certificado.

Clave pública del usuario: clave pública del usuario del certificado.

Firma: firma de la CA.

El uso de certificados X.509 está teniendo una gran expansión en las comunicaciones a través de Internet, gracias a su eficiencia para autenticar datos. Este sistema se perfila como el presente y futuro cercano para firmar digitalmente los mensajes; de hecho los trabajos que continúan los pasos de FreeS/WAN (Strongsec y Strongswan) incluyen soporte para X.509 en sus implementaciones. Para saber cómo construir un sistema de certificados X.509, consultar apartado 4.2.2.2.2.3.

3.5. Instalación y ampliaciones para FreeS/WAN

3.5.1. Instalación

Algunas distribuciones de Linux, como SUSE, incluyen FreeS/WAN en su paquete de *software*. En este caso la instalación es inmediata, tan sólo debe comprobarse que el código del núcleo de Linux (*kernel*) incluye los parches para FreeS/WAN. Sin embargo, debido a las restricciones que imponen las “leyes de exportación” (*export laws*), en Estados Unidos, para elementos de criptografía, otras distribuciones no pueden incluirlo entre su *software*. Para estos casos, habrá que descargar el RPM de FreeS/WAN de la versión que deseemos [3].

Una vez descargado, el RPM puede instalarse mediante la herramienta de instalación de paquetes que tenga la distribución de Linux, o bien por consola introduciendo `rpm -i freeswan-X.XX Y.Y.Y-i586.rpm` en el directorio en que se encuentre dicho paquete (donde *X.XX* corresponde a la versión de FreeS/WAN, e *Y.Y.Y* a la versión del parche para certificados X.509, disponible a partir de la versión 1.99 de FreeS/WAN).

3.5.2. Ampliaciones

Existen varias ampliaciones para FreeS/WAN, implementadas siempre por sus propios usuarios (es decir, que no son oficiales). Sin embargo, una de estas ampliaciones ha alcanzado mucha aceptación entre los usuarios de FreeS/WAN por su utilidad. Se trata de un parche que permite el uso de certificados X.509 para la autenticación de usuarios (ver apartado 3.4.3 para saber cómo funciona X.509). Este parche viene incluido en alguno de los paquetes de instalación de FreeS/WAN, como el que incluye SUSE en sus distribuciones [4].

A día de hoy, FreeS/WAN no va a continuar siendo desarrollado, por lo que no se podrán encontrar nuevas ampliaciones o actualizaciones, pero existen dos grupos de trabajo basados en FreeS/WAN que continúan en activo: Strongsec y Strongswan. Ambos tienen como base la versión 2.04 de FreeS/WAN, a la que cada uno añade una serie de características, que complementan las posibilidades de esta herramienta [1].

Parte II:

CONFIGURACIÓN DE LOS ESCENARIOS

4. Escenario 1: Conexión Directa
5. Escenario 2: Conexión *Road Warrior*
6. Escenario 3: Conexión “Oportunista”

4. Escenario 1: conexión directa

4.1. Introducción

El objetivo de este escenario es construir un túnel IPSec, basado en FreeS/WAN, entre dos usuarios conocidos. La conexión directa IPSec puede emplearse para crear un enlace seguro entre dos usuarios finales (que serán los extremos de la comunicación), o bien para crear una red privada virtual entre las subredes que se hayan detrás de estos usuarios (en cuyo caso actúan como pasarelas de seguridad). En ambos casos, el túnel se establece entre los dos equipos que implementan IPSec. La diferencia reside en que, en el caso con subredes, estos equipos son intermediarios encargados de codificar y autenticar la información que se envía a la subred remota. Se consigue así enlazar las dos subredes como si fueran una sola.

Hemos elegido realizar la configuración con subredes para mostrarla en nuestro trabajo, ya que ésta es empleada con mayor frecuencia que la conexión directa entre dos usuarios finales. Esta configuración puede resultar útil a la hora de conectar de modo seguro dos redes locales de una misma organización alejadas entre sí (como por ejemplo, dos filiales de una empresa que necesitan compartir recursos).

La puesta en marcha de este tipo de escenario tiene tres requisitos:

- Las direcciones IP de los equipos que construyen el túnel (en adelante los llamaremos pasarelas) deberán ser fijas, y en su defecto cualquier cambio de dirección deberá ser comunicado al otro extremo, si no la conexión no podrá realizarse.
- Las pasarelas conocerán la dirección IP de la pasarela remota (aquella con la que establecen el túnel).
- Las pasarelas habrán intercambiado previamente, y de manera segura, sus claves de autenticación de usuario. Esto no será necesario si se emplean certificados X.509 para la autenticación (ver apartado 4.2.2.2.3).

Nota: La configuración que exponemos a continuación fue realizada en un laboratorio de la Universidad Politécnica de Cartagena para comprobar su correcto funcionamiento. Si quisiera emplearse este documento para crear una conexión personalizada, bastaría con cambiar, en el archivo de configuración *ipsec.conf*, las direcciones IP que aparecen aquí por las que vayan a usarse.

4.2. Configuración del escenario

4.2.1. Diagramas de situación

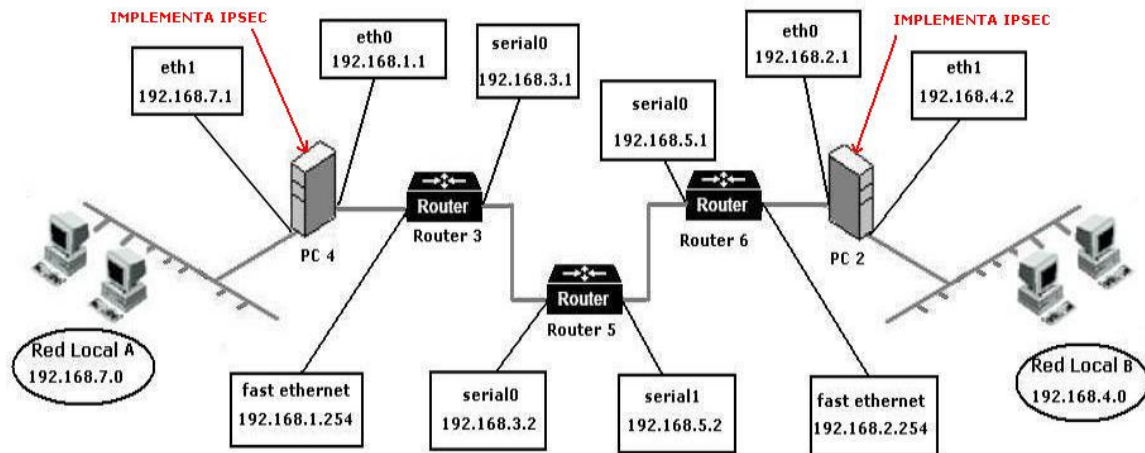


Figura 4.1 – Diagrama del escenario de conexión directa.

La *figura 4.1* muestra la topología diseñada para probar las configuraciones de conexión directa IPsec con subredes. *PC2* y *PC4* son nuestras pasarelas de seguridad: los equipos que implementan FreeS/WAN y establecen el túnel para comunicar sus respectivas subredes (de direcciones 192.168.7.0/24 y 192.168.4.0/24). Los *routers* situados entre las pasarelas tienen como objetivo simular la infraestructura de una red de interconexión pública (por ejemplo, Internet), una “nube”, de modo que la ruta que conecte las dos pasarelas no sea directa. Obtenemos así un escenario comparable (a escala, en el laboratorio) a las situaciones más comunes, en las que la conexión directa IPsec debe hacerse a través de la infraestructura de redes públicas.

Para cada equipo de red utilizado (pasarelas y *routers*), indicamos en la figura las interfaces de red usadas y sus direcciones IP correspondientes.

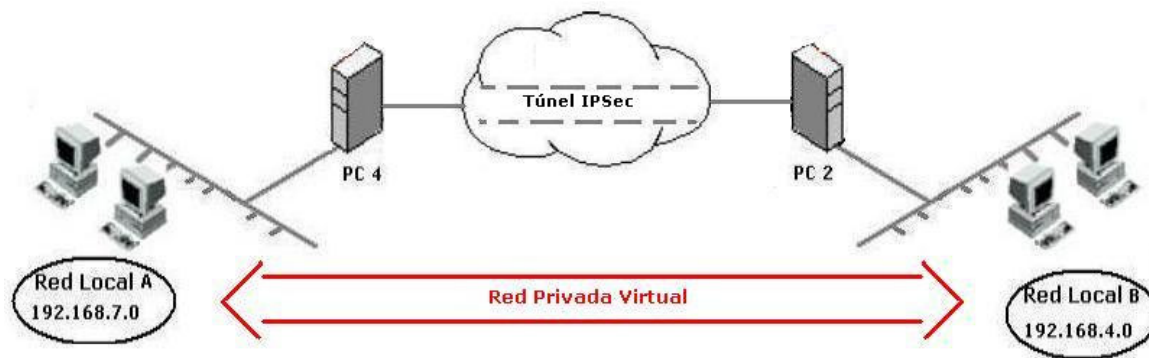


Figura 4.2 – Diagrama equivalente túnel IPSec.

4.2.2. Configuración de equipos paso a paso

Tanto para este escenario como para el escenario *Road Warrior*, existen tres posibilidades para asegurar la autenticación del usuario, tal y como vimos en el capítulo 3 (apartado 3.4): secretos compartidos, firmas RSA y certificados X.509. Dedicaremos, para cada uno de los tres, un apartado indicando la configuración específica que requiere su utilización en FreeS/WAN. Sin embargo, comenzaremos por detallar la configuración de los elementos independientes del método de autenticación empleado. En esta categoría se incluye la configuración de los *routers* CISCO, así como la configuración de los parámetros de interconexión (direcciones, tablas de encaminamiento) de las pasarelas de seguridad (en nuestro caso *PC2* y *PC4*).

4.2.2.1. Configuración *Routers* CISCO 1751

La función de estos equipos de red se limita, en nuestro trabajo, a ofrecer interconexión entre las distintas redes de la infraestructura. Para lograrlo, debemos configurar las interfaces de red y las tablas de encaminamiento de cada *router* [5].

Hemos elegido la aplicación *Microsoft Hyperterminal* para acceder a los menús de configuración de los *routers*, pero cualquier otro *software* de emulación de Terminal puede emplearse. Conectamos, en primer lugar, la entrada del *router* denominada *Console* con el puerto serie del ordenador que ejecuta *Hyperterminal*, para que pueda haber comunicación. Arrancamos el programa y, tras darle un nombre a la conexión, debemos configurar las siguientes características para conectar con el *router*:

- 9600 baudios.
- 8 bits de datos.
- Sin bit de paridad.
- 1 bit de stop.
- Sin control de flujo (en inglés *flow control*).

Hecho esto, el terminal conecta con el *router*, y tras unos instantes debe aparecer un mensaje como este en consola:

Router> (o en su defecto el identificador asignado al router)

A partir de este momento estamos, en comunicación directa con el *router*, y tendremos que emplear los comandos que “comprenda” su sistema operativo para configurarlo. En todo momento podremos introducir “?” para conocer los distintos comandos posibles.

1. Comenzamos por pasar a modo privilegiado, lo que nos dará autorización para modificar todos los parámetros del *router*:

Router> enable

Sabremos que estamos en modo privilegiado porque el mensaje de consola pasa a ser *Router#*

2. Entramos en el menú de configuración:

Router# configure terminal

Sabremos que estamos en el menú de configuración porque el mensaje de consola pasa a ser *Router(config)#*

3. Entramos en el menú de la interfaz *Fast Ethernet*:

Router(config)# configure interface fastethernet 0/0

Sabremos que estamos en el menú de una interfaz de red porque el mensaje de consola pasa a ser *Router(config-if)#*

4. Configuramos la dirección IP de esta interfaz de red:

Router(config-if)# ip address <dirección IP> <máscara de red>

Con este comando indicaremos la dirección IP que queremos asignar a la interfaz de red, seguida de la máscara de red aplicable a dicha dirección.

5. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

6. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

7. Entramos en el menú de la interfaz *Serial0*:

```
Router(config)# configure interface serial 1/0
```

8. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

9. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

10. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

11. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

12. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

13. Entramos en el menú de la interfaz *Serial1*:

```
Router(config)# configure interface serial 1/1
```

14. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

15. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

16. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

17. Activamos esta interfaz de red:

Router(config-if)# no shutdown

18. Salimos del menú de la interfaz de red:

Router(config-if)# exit

19. Rellenamos la tabla de encaminamiento del *router* empleando tantas veces como sea necesario este comando:

Router(config)# ip route <dirección de red destino> <máscara de red> <siguiente pasarela>

Para cada ruta, habrá que indicar la dirección IP de la red destino, seguido de la máscara de red aplicable a esta dirección, y de la dirección IP de la siguiente pasarela en la ruta hacia la red destino.

20. Salimos del menú de configuración:

Router(config)# exit

21. Grabamos los cambios efectuados en la configuración en la memoria de arranque del *router*:

Router(config)# copy run start

Nota: Los pasos de configuración de la interfaz *Serial1* (13 a 18) sólo son aplicables al *Router 5*, el único que utiliza esta interfaz. Del mismo modo, los pasos para configurar la interfaz *Fast Ethernet* (3 al 6) no se aplican al *Router 5*, puesto que no la emplea.

A continuación, indicamos las direcciones IP (por medio de los comandos descritos arriba) a asignar a cada interfaz de red, para cada *router*, así como sus respectivas tablas de encaminamiento:

	ROUTER 3		ROUTER 5		ROUTER 6	
Interfaz	Fast Ethernet	Serial 0	Serial 0	Serial 1	Fast Ethernet	Serial 0
Dirección IP	192.168.1.254	192.168.3.1	192.168.3.2	192.168.5.2	192.168.2.254	192.168.5.1
Máscara de red	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Figura 4.3 – Tabla de asignación de direcciones IP.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.2.0	255.255.255.0	192.168.3.2
192.168.5.0	255.255.255.0	192.168.3.2

Figura 4.4 – Tabla de encaminamiento del *Router 3*.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.3.1
192.168.2.0	255.255.255.0	192.168.5.1

Figura 4.5 – Tabla de encaminamiento del *Router 5*.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.5.2
192.168.3.0	255.255.255.0	192.168.5.2

Figura 4.6 – Tabla de encaminamiento del *Router 6*.

4.2.2.2. Configuración de las pasarelas de seguridad

Son varios los parámetros a configurar en las pasarelas de seguridad, *PC2* y *PC4*. Consideramos que la herramienta FreeS/WAN está correctamente instalada en ellos, por lo que pasamos directamente a cuestiones de configuración. Tendremos que modificar, en primer lugar, la configuración de red de cada equipo, y luego pasaremos a los archivos de FreeS/WAN: *ipsec.conf* e *ipsec.secrets*, que variarán según el método de autenticación elegido.

4.2.2.2.1. Configuración de los parámetros de interconexión

1. Es imprescindible permitir el reenvío de paquetes en ambas pasarelas. Esto se consigue asignando al archivo `/proc/sys/net/ipv4/ip_forward` el valor “1” (por defecto está a 0). Podemos hacer esto mediante el comando del terminal de Linux `echo`:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

2. Configuramos las direcciones IP de las interfaces de red de las pasarelas, mediante el comando del terminal de Linux `ifconfig`:

Uso de `ifconfig`: `ifconfig <interfaz> <dirección a asignar> netmask <máscara de red>`

```
PC2: ifconfig eth0 192.168.2.1 netmask 255.255.255.0
     ifconfig eth1 192.168.4.2 netmask 255.255.255.0
```

```
PC4: ifconfig eth0 192.168.1.1 netmask 255.255.255.0
     ifconfig eth1 192.168.7.1 netmask 255.255.255.0
```

3. Configuramos las tablas de encaminamiento de las pasarelas, mediante el comando del terminal de Linux `route`:

Uso de `route` para añadir una entrada a la tabla de encaminamiento:

```
route add -net <dirección de red destino> netmask <máscara de red> gw
<dirección de la siguiente pasarela en ruta hacia red destino>
```

```
PC2: route add -net 192.168.1.0 netmask 255.255.255.0 gw
     192.168.2.254
```

```
PC4: route add -net 192.168.2.0 netmask 255.255.255.0 gw
     192.168.1.254
```

4.2.2.2.2 Archivos de FreeS/WAN

4.2.2.2.2.1. Autenticación por secreto compartido

1. Editamos el archivo `/etc/ipsec.secrets` y colocamos la siguiente entrada (habrá que hacer lo mismo para las dos pasarelas):

```
: PSK "Esteesmisecreto"
```

Nota: Cualquier cadena de caracteres entre comillas es válida como secreto compartido.

2. Editamos el archivo `/etc/ipsec.conf` y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta

con agregar al final de éste la sección *conn peer-to-peerPSK*). El contenido será el mismo para ambas pasarelas:

```

config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces="ipsec0=eth1"
    uniqueids=yes

conn peer-to-peerPSK
    type=tunnel
    authby=secret
    right=192.168.2.1
    rightright=192.168.2.254
    rightsubnet=192.168.4.0/24
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.7.0/24
    auto=start

```

4.2.2.2.2. Autenticación por firmas RSA

1. Para cada pasarela, creamos un par de claves RSA mediante el comando *ipsec rsasigkey*, y las incluimos en el fichero */etc/ipsec.secrets* (en nuestro escenario serán claves de 2048 bits). Junto a la clave RSA, habrá que incluir unos parámetros para que FreeS/WAN reconozca el formato del archivo (emplearemos el comando de Linux *echo*):

```

echo ": RSA {" > /etc/ipsec.secrets
ipsec rsasigkey 2048 >> /etc/ipsec.secrets
echo "}" >> /etc/ipsec.secrets

```

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn peer-to-peerRSA*). Para rellenar los parámetros de clave pública RSA (*leftrsasigkey* y *rightrsasigkey*), hay que tener en cuenta que tomamos *PC4* como la pasarela de izquierda (*left*) y *PC2* derecha (*right*). Así, en *leftrsasigkey* copiaremos la cadena de caracteres que aparece a continuación de "*#pubkey=*" en el archivo *ipsec.secrets* de *PC24* y en *rightrsasigkey* haremos lo mismo pero con el archivo *ipsec.secrets* de *PC2*.

```

config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search

```

```

interfaces="ipsec0=eth1"
uniqueids=yes

conn peer-to-peerRSA
type=tunnel
authby=rsasig
right=192.168.2.1
rightnexthop=192.168.2.254
rightsubnet=192.168.4.0/24
rightrsasigkey=<clave pública RSA PC2>
left=192.168.1.1
leftnexthop=192.168.1.254
leftsubnet=192.168.7.0/24
leftrsasigkey=<clave pública RSA PC4>
auto=start

```

NOTA: Puesto que este archivo deberá ser exactamente igual en las dos pasarelas, es recomendable rellenarlo en una de ellas (aunque necesitará la clave pública de la otra) y hacer una copia para la otra. Se recomienda intercambiar los ficheros con SSH o alguna otra aplicación para comunicaciones seguras.

4.2.2.2.3. Autenticación por certificados X.509

Pasos para obtener los certificados X.509

1. En una de las pasarelas, construimos nuestra propia Autoridad de Certificación (CA) mediante el comando *openssl* del terminal de Linux. Esto nos permitirá firmar nuestros propios certificados para las claves RSA:

```
openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem
```

Al ejecutar el comando habrá que rellenar una serie de parámetros sobre la CA que estamos creando. Podemos hacerlo a nuestro gusto, pero teniendo cuidado de poner lo mismo en el campo “*Organization Name*” que cuando se nos pregunte por ese campo al crear los requerimientos de certificado.

Esto genera una clave privada (*cakey.pem*), que moveremos a la carpeta */etc/ipsec.d/private*, y un certificado público firmado (*cacert.pem*), que moveremos a la carpeta */etc/ipsec.d/cacerts*.

2. En ambas pasarelas, creamos una clave privada RSA (en nuestro escenario son claves de 1024 bits) y su correspondiente petición de certificado, utilizando el comando *openssl* desde el directorio */etc/ipsec.d*:

```
PC2: openssl req -newkey rsa:1024 -keyout pc2key.pem -out pc2req.pem
```

```
PC4: openssl req -newkey rsa:1024 -keyout pc4key.pem -out pc4req.pem
```

Para crear la petición de certificado habrá que rellenar unos parámetros, con el requisito de introducir lo mismo que pusimos para la CA en el campo “*Organization Name*”. Como resultado del comando, se genera la clave privada (*pcXkey.pem*), que moveremos a la carpeta */etc/ipsec.d/private*, y una petición de certificación que deberá firmar la CA.

3. Para que nuestra CA pueda firmar las peticiones de certificación, tendremos que hacer una serie de cambios en la pasarela en la que fue creada: habrá que copiar el archivo */usr/share/ssl/openssl.cnf* al directorio */etc/ipsec.d/*, editarlo y modificar algunos valores. En concreto, bajo la sección “[*CA_default*]” asignaremos a la variable “*dir*” la expresión “*/*” (en lugar de *./demoCA*), y a la variable “*certificate*” le daremos el valor “*\$dir/cacerts/cacert.pem*”. Además, tendremos que crear dos archivos en el directorio */etc/ipsec.d:* *index.txt*, que dejaremos vacío, y *serial*, que editaremos para escribir en él “01” [6].

4. La CA deberá firmar las peticiones de certificado, por lo que la pasarela que NO construyó la CA tendrá que enviarle su petición (*pcXreq.pem*) a la otra. Copiaremos esa petición al directorio */etc/ipsec.d* y desde ahí firmaremos ambas peticiones con el comando *openssl*:

```
openssl ca -in pc2req.pem -out pc2cert.pem -notext -config \
./openssl.cnf

openssl ca -in pc4req.pem -out pc4cert.pem -notext -config \
./openssl.cnf
```

Esto generará los certificados de clave pública para ambas pasarelas (*pcXcert.pem*).

5. Habrá que enviar estos certificados (*pcXcert.pem*), así como el certificado de la CA (*ca.cert.pem*) a la pasarela que no construyó la CA. Una vez copiados, ubicaremos los tres archivos en el directorio */etc/ipsec.d*.

Configuración de archivos

1. Para cada pasarela, editamos el archivo */etc/ipsec.secrets* e introducimos lo siguiente (en lugar de indicar directamente una clave RSA indicamos el archivo en el que se encuentra, seguido de una clave opcional para aportar más seguridad):

```
PC2:   : RSA pc2key.pem "hola"
PC4:   : RSA pc4key.pem "hola"
```

Nota: La expresión “*hola*” es una clave opcional que debe ser la misma para ambas pasarelas. Cualquier cadena de caracteres entre comillas es válida para esto.

2. Para cada pasarela, editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna

conexión basta con agregar al final de éste la sección *conn peer-to-peerX509*). Los parámetros *rightid* y *leftid* (teniendo en cuenta que PC2 corresponde a *right*, y PC4 a *left*) se obtienen ejecutando, desde el directorio */etc/ipsec.d/*, el comando de *openssl* que extrae el identificador del certificado:

```
rightid → openssl x509 -in pc2cert.pem -noout -subject
```

```
leftid → openssl x509 -in pc4cert.pem -noout -subject
```

En este caso, el archivo *ipsec.conf* SÍ es distinto para cada pasarela, hay un cambio en el campo *left/right-rsasigkey*. Los ficheros quedan así:

```
PC2:  config setup
      klipsdebug=none
      plutodebug=none
      plutoload=%search
      plutostart=%search
      interfaces="ipsec0=eth1"
      uniqueids=yes

conn peer-to-peerX509
  type=tunnel
  authby=rsasig
  right=192.168.2.1
  rightrightnextthop=192.168.2.254
  rightsubnet=192.168.4.0/24
  rightid=<identificador certificado PC2>
  rightcert=pc2cert.pem
  left=192.168.1.1
  leftnextthop=192.168.1.254
  leftsubnet=192.168.7.0/24
  leftid=<identificador certificado PC4>
  leftrsasigkey=%cert          # ¡OJO! #
  leftcert=pc4cert.pem
  auto=start
```

```
PC4:  config setup
      klipsdebug=none
      plutodebug=none
      plutoload=%search
      plutostart=%search
      interfaces="ipsec0=eth1"
      uniqueids=yes

conn peer-to-peerX509
  type=tunnel
  authby=rsasig
  right=192.168.2.1
  rightrightnextthop=192.168.2.254
  rightsubnet=192.168.4.0/24
  rightid=<identificador certificado PC2>
  rightrsasigkey=%cert          # ¡OJO! #
  rightcert=pc2cert.pem
```

```
left=192.168.1.1
leftnexthop=192.168.1.254
leftsubnet=192.168.7.0/24
leftid=<identificador certificado PC4>
leftcert=pc4cert.pem
auto=start
```

4.3. Puesta en marcha del túnel IPsec

Una vez terminado todo el trabajo de configuración, es hora de arrancar la aplicación y comprobar que el túnel IPsec se establece correctamente. Este es el comando que arranca FreeS/WAN (habrá que ejecutarlo en las dos pasarelas):

```
ipsec setup start
```

Para las configuraciones de este escenario, no importa qué equipo arranque antes la herramienta FreeS/WAN, ya que los dos participantes tienen la misma función (son pasarelas de seguridad).

Podemos verificar que no ha habido ningún error mediante el comando *ipsec barf*, que nos devuelve información detallada acerca de las acciones de Pluto. El Apéndice B muestra los mensajes que devuelve *ipsec barf* si todo funciona correctamente.

5. Escenario 2: *Road Warrior*

5.1. Introducción

El objetivo de este escenario es construir una conexión IPSec, basada en FreeS/WAN, entre un cliente móvil (también llamado *Road Warrior*) y una subred, situada detrás de una pasarela de seguridad. Esta configuración se basa en suponer que el cliente móvil conoce la dirección de la pasarela, mientras que la pasarela admitirá cualquier dirección origen, para el cliente, siempre que se identifique mediante la clave de autenticación adecuada. Así, un cliente remoto podrá conectar con su subred de trabajo (a través de la pasarela), desde cualquier dirección, con la única condición de identificarse como usuario autorizado.

En estos casos, el túnel se establece entre el cliente móvil y la pasarela, protegiendo el tráfico de datos entre el cliente y la subred situada tras la pasarela. Esta configuración es muy empleada en el caso de empleados que viajan por cuestiones de trabajo y deben acceder a la red local de su empresa, sin importar desde dónde lo hagan.

Para distinguir las claves pertenecientes a los distintos clientes, FreeS/WAN debe asociar cada clave de autenticación con un identificador del cliente, que podrá ser un nombre de *host*, o una dirección IP (cuando el cliente tenga dirección fija). Esto permite establecer varias conexiones *Road Warrior* simultáneas con la misma pasarela.

Para que pueda realizarse este escenario, deben cumplirse tres condiciones:

- El cliente móvil (en adelante cliente) deberá conocer la dirección IP de la pasarela que le permite enlazar con la subred. Esta dirección tendrá que ser fija, y en caso de cambiar deberá ser comunicada al cliente.
- Cliente y pasarela habrán intercambiado, previamente y de manera segura, sus claves de autenticación, ya que sin ellas no pueden reconocerse entre ellos. Esto no será necesario si se emplean certificados X.509 (ver apartado 5.2.2.3.2).
- Si se emplean firmas RSA como método de autenticación, es necesario que la pasarela conozca el nombre de *host* del cliente, para poder asociarlo con su clave de autenticación (al no tener el cliente una dirección IP fija).

Nota: La configuración que exponemos a continuación fue realizada en un laboratorio de la Universidad Politécnica de Cartagena para comprobar su correcto funcionamiento. Si quisiera emplearse este documento para crear una conexión personalizada, bastaría con cambiar, en el archivo de configuración *ipsec.conf*, así

como en los archivos del servidor DNS las direcciones IP que aparecen aquí por las que vayan a usarse.

5.2. Configuración del escenario

5.2.1. Diagramas de situación

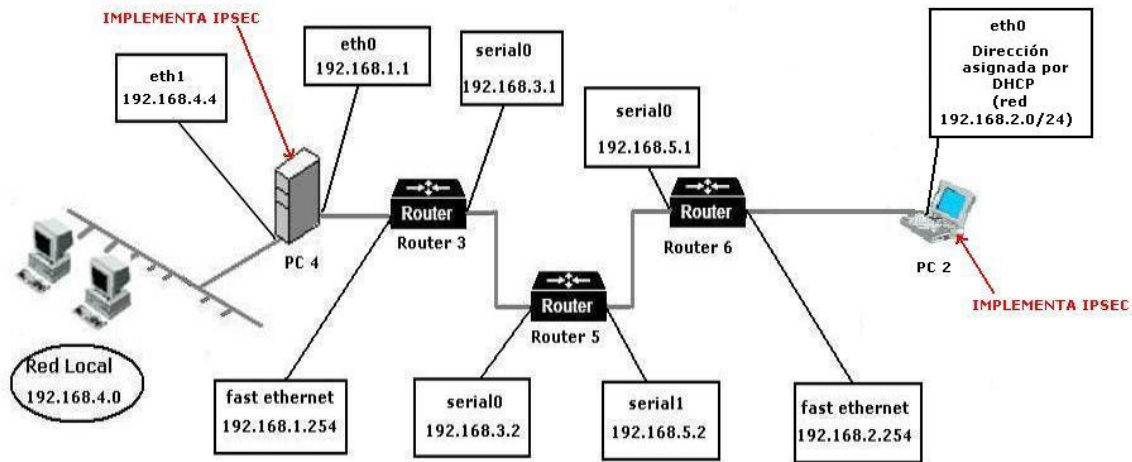


Figura 5.1 – Diagrama del escenario *Road Warrior*.

La *figura 5.1* muestra la topología diseñada para probar las configuraciones de conexión *Road Warrior*. *PC4* es nuestra pasarela de seguridad, que implementa FreeS/WAN y hace de intermediario para comunicar con la subred que tiene detrás (de dirección 192.168.4.0/24). *PC2* es nuestro “cliente móvil”, implementa FreeS/WAN, y será el encargado de dirigirse a la pasarela para establecer el túnel IPsec que le permita comunicar con la subred 192.168.4.0/24. Los *routers* situados entre *PC2* y *PC4* tienen como objetivo simular la infraestructura de una red de interconexión pública (por ejemplo, Internet), una “nube”, de modo que la ruta que conecte los dos equipos no sea directa. Obtenemos así un escenario comparable (a escala, en el laboratorio) a las situaciones más comunes, en las que la conexión IPsec debe hacerse a través de la infraestructura de redes públicas.

Para cada equipo de red utilizado (PCs y *routers*), indicamos en la figura las interfaces de red usadas y sus direcciones IP correspondientes. En el caso de *PC2*, es el *Router 6* el encargado de asignarle una dirección IP por medio del protocolo DHCP (descrito en el apartado 2.3.2). Mediante la asignación dinámica de IP, se simula la movilidad del cliente que caracteriza a los *Road Warrior*.

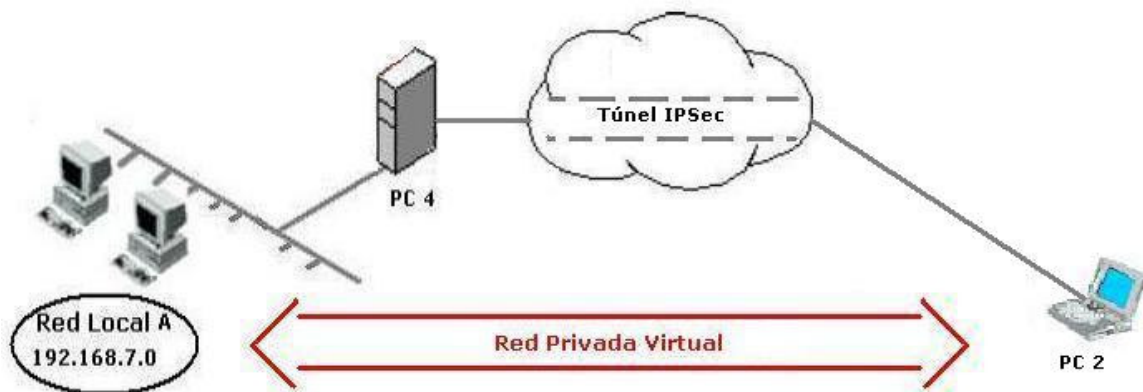


Figura 5.2 – Diagrama equivalente túnel IPsec.

5.2.2. Configuración de equipos paso a paso

Tal y como vimos en el escenario de conexión directa, existen tres posibilidades para asegurar la autenticación del usuario: secretos compartidos, firmas RSA y certificados X.509. Dedicaremos, para cada uno de los tres, un apartado indicando la configuración específica que requiere su utilización en FreeS/WAN. Dejamos la autenticación por firmas RSA en último lugar, ya que su uso requiere un mayor número de cambios (que detallamos en el apartado 5.2.2.3.3).

No obstante, comenzaremos por detallar la configuración de los elementos independientes del método de autenticación empleado. En esta categoría se incluye la configuración de los *routers* CISCO, así como la configuración de los parámetros de interconexión (direcciones, tablas de encaminamiento) de la pasarela de seguridad (*PC4*), y del cliente móvil (*PC2*).

5.2.2.1. Configuración *Routers* CISCO 1751

La función de estos equipos de red es, en nuestro trabajo, ofrecer interconexión entre las distintas redes de la infraestructura. Para lograrlo, debemos configurar las interfaces de red y las tablas de encaminamiento de cada *router*. Además, el *Router 6* tendrá que actuar como servidor DHCP para el cliente *PC2*, de modo que su dirección IP sea variable [5].

Hemos elegido la aplicación *Microsoft Hyperterminal* para acceder a los menús de configuración de los *routers*, pero cualquier otro *software* de emulación de Terminal puede emplearse. Conectamos, en primer lugar, la entrada del *router* denominada *Console* con el puerto serie del ordenador que ejecuta *Hyperterminal*, para que pueda haber comunicación. Arrancamos el programa y, tras darle un nombre a la conexión, debemos configurar las siguientes características para conectar con el *router*:

- 9600 baudios.
- 8 bits de datos.
- Sin bit de paridad.
- 1 bit de stop.
- Sin control de flujo (en inglés *flow control*).

Hecho esto, el terminal conecta con el *router*, y tras unos instantes debe aparecer un mensaje como este en consola:

Router> (o en su defecto el identificador asignado al router)

A partir de este momento estamos, en comunicación directa con el *router*, y tendremos que emplear los comandos que “comprenda” su sistema operativo para configurarlo. En todo momento podremos introducir “?” para conocer los distintos comandos posibles.

1. Comenzamos por pasar a modo privilegiado, lo que nos dará autorización para modificar todos los parámetros del *router*:

Router> enable

Sabremos que estamos en modo privilegiado porque el mensaje de consola pasa a ser *Router#*

2. Entramos en el menú de configuración:

Router# configure terminal

Sabremos que estamos en el menú de configuración porque el mensaje de consola pasa a ser *Router(config)#*

3. Entramos en el menú de la interfaz *Fast Ethernet*:

Router(config)# configure interface fastethernet 0/0

Sabremos que estamos en el menú de una interfaz de red porque el mensaje de consola pasa a ser *Router(config-if)#*

4. Configuramos la dirección IP de esta interfaz de red:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

Con este comando indicaremos la dirección IP que queremos asignar a la interfaz de red, seguida de la máscara de red aplicable a dicha dirección.

5. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

6. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

7. Entramos en el menú de la interfaz *Serial0*:

```
Router(config)# configure interface serial 1/0
```

8. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

9. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

10. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

11. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

12. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

13. Entramos en el menú de la interfaz *Serial1*:

```
Router(config)# configure interface serial 1/1
```

14. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

15. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

16. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

17. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

18. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

19. Rellenamos la tabla de encaminamiento del *router* empleando tantas veces como sea necesario este comando:

```
Router(config)# ip route <dirección de red destino> <máscara de red> <siguiente pasarela>
```

Para cada ruta, habrá que indicar la dirección IP de la red destino, seguido de la máscara de red aplicable a esta dirección, y de la dirección IP de la siguiente pasarela en la ruta hacia la red destino.

20. Salimos del menú de configuración:

```
Router(config)# exit
```

21. Grabamos los cambios efectuados en la configuración en la memoria de arranque del *router*:

```
Router(config)# copy run start
```

Nota: Los pasos de configuración de la interfaz *Serial1* (13 a 18) sólo son aplicables al *Router 5*, el único que utiliza esta interfaz. Del mismo modo, los pasos para configurar la interfaz *Fast Ethernet* (3 al 6) no se aplican al *Router 5*, puesto que no la emplea.

En el *Router 6*, tendremos que realizar unos pasos suplementarios para activar el protocolo DHCP, que dará servicio a *PC2*:

21. Entramos en el menú de configuración:

```
Router# configure terminal
```

22. Creamos una base de datos (un “*pool*”), de nombre *road*, para la configuración DHCP:

```
Router(config)# ip dhcp pool road
```

23. Configuramos la red (192.168.2.0) y la máscara de red que utilizará DHCP para otorgar direcciones IP:

```
Router(dhcp-config)# network 192.168.2.0 255.255.255.0
```

24. Configuramos la dirección del *router* por defecto para los clientes del servicio DHCP (en nuestro caso será la dirección de la interfaz *Fast Ethernet* del *Router 6*):

```
Router(dhcp-config)# default-router 192.168.2.254
```

25. Salimos del menú de configuración de DHCP :

```
Router(dhcp-config)# exit
```

26. Especificamos la única dirección que el servidor DHCP no podrá asignar a ningún cliente (es decir, la dirección de su interfaz *Fast Ethernet*, para que no haya conflicto):

```
Router(config)# ip dhcp excluded-address 192.168.2.254
```

27. Salimos del menú de configuración:

```
Router(config)# exit
```

28. Grabamos los cambios efectuados en la configuración en la memoria de arranque del *router*:

```
Router(config)# copy run start
```

A continuación, indicamos las direcciones IP (por medio de los comandos descritos arriba) a asignar a cada interfaz de red, para cada *router*, así como sus respectivas tablas de encaminamiento:

	ROUTER 3		ROUTER 5		ROUTER 6	
Interfaz	Fast Ethernet	Serial 0	Serial 0	Serial 1	Fast Ethernet	Serial 0
Dirección IP	192.168.1.254	192.168.3.1	192.168.3.2	192.168.5.2	192.168.2.254	192.168.5.1
Máscara de red	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Figura 5.3 – Tabla de asignación de direcciones IP.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.2.0	255.255.255.0	192.168.3.2
192.168.5.0	255.255.255.0	192.168.3.2

Figura 5.4 – Tabla de encaminamiento del *Router 3*.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.3.1
192.168.2.0	255.255.255.0	192.168.5.1

Figura 5.5 – Tabla de encaminamiento del *Router 5*.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.5.2
192.168.3.0	255.255.255.0	192.168.5.2

Figura 5.6 – Tabla de encaminamiento del *Router 6*.

5.2.2.2. Configuración de los parámetros de interconexión en *PC2* y *PC4*

5.2.2.2.1. La pasarela de seguridad: *PC4*

1. Es imprescindible permitir el reenvío de paquetes. Esto se consigue asignando al archivo `/proc/sys/net/ipv4/ip_forward` el valor "1" (por defecto está a 0). Podemos hacer esto mediante el comando del terminal de Linux `echo`:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

2. Configuramos las direcciones IP de las interfaces de red de la pasarela, mediante el comando del terminal de Linux `ifconfig`:

Uso de *ifconfig*: `ifconfig <interfaz> <dirección a asignar> netmask <máscara de red>`

```
ifconfig eth0 192.168.1.1 netmask 255.255.255.0
ifconfig eth1 192.168.4.4 netmask 255.255.255.0
```

3. Configuramos las tablas de encaminamiento de la pasarela, mediante el comando del terminal de Linux *route*:

```
route add default gw 192.168.1.254
```

Dado que no conocemos la dirección del cliente móvil, la pasarela tendrá que ser capaz de dirigirse a cualquier red destino. Mediante este comando, asignamos la dirección 192.168.1.254 (es decir, la interfaz *Fast Ethernet* del *Router 3*) como dirección de salida por defecto, para cualquier destino que no sea un equipo de la subred 192.168.4.0/24. Puesto que la única ruta de salida hacia la “nube” es ésta, aseguramos la salida de paquetes hacia el cliente móvil, independientemente de su dirección.

5.2.2.2.2. El cliente móvil: *PC2*

Como explicamos junto al diagrama de situación del escenario (apartado 5.1), *PC2* tendrá en su interfaz *eth0* una dirección IP variable, asignada por el *Router 6* mediante el protocolo DHCP.

1. Para permitir la asignación dinámica de dirección IP, habrá que entrar en la utilidad de configuración del sistema (en distribuciones de Linux SUSE usamos la herramienta *Yast2*) y localizar la configuración de las interfaces de red. Seleccionamos la interfaz *eth0*, y activamos la casilla que permite el empleo de DHCP para asignarle una dirección IP.

2. Puesto que no conocemos la dirección que le asignará DHCP, no podemos configurar las tablas de encaminamiento. Sin embargo, DHCP envía, junto con la dirección asignada, la dirección de la ruta de salida para la interfaz que configura en el cliente. Así, *PC2* recibirá la dirección 192.168.2.254 (correspondiente a la interfaz *Fast Ethernet* del *Router 6*) como dirección de salida para la interfaz *eth0*. Si desactivamos previamente las interfaces de red restantes en el cliente, ésta será la única ruta de salida que conocerá *PC2*, por lo que se convertirá en la dirección de salida por defecto. Éste es nuestro objetivo, puesto que el *Router 6* es la única vía de conectar con la “nube”.

Para desactivar las interfaces de red (salvo *eth0*), utilizaremos el comando del terminal de Linux *ifconfig*:

```
ifconfig <nombre de la interfaz de red> down
```

Nota: Esto también nos será útil para poder emplear el parámetro *%defaultroute* a la hora de rellenar campos en el archivo *ipsec.conf*. Al existir una sola interfaz de red, FreeS/WAN tomará forzosamente la interfaz *eth0* para construir las conexiones IPSec.

3. Si empleamos firmas RSA como método de autenticación, tendremos que asegurarnos de que el cliente tenga un identificador (un nombre de *host*) único, que coincida con el que incluiremos en el archivo *ipsec.conf*. Para conseguirlo, son necesarios dos pasos:

- Editamos el archivo */etc/hosts*, que contiene los nombres de *host* asignados a cada dirección IP, y eliminamos todos los identificadores presentes. De este modo, el cliente no tendrá un nombre de *host* predefinido para ninguna dirección.
- Asignamos un nombre de *host* al cliente mediante el comando del terminal de Linux *hostname*:

```
hostname pc2.it4.upct.es
```

Donde *pc2.it4.upct.es* es el nombre que damos al cliente.

Nota: Este último paso se debe a que FreeS/WAN debe asociar cada clave RSA con un identificador (una dirección IP o en su defecto un nombre de *host*), para poder distinguir las claves RSA de cada cliente. Ya que, en este escenario, el cliente no dispone de dirección IP fija, tendremos que darle a FreeS/WAN un nombre de *host* del cliente, para que lo asocie con su clave RSA. Este nombre de *host* deberá estar registrado, a su vez, en el propio cliente, para que pueda identificarse con él.

5.2.2.3. Configuración de los archivos de FreeS/WAN

En este escenario, los archivos de configuración serán siempre distintos para *PC2* y *PC4*. En cuanto a la configuración del archivo *ipsec.conf* hay que tener en cuenta que tomamos *PC4* como participante izquierdo (*left*) de la conexión IPSec, y *PC2* como participante derecho (*right*).

5.2.2.3.1. Autenticación por secreto compartido

5.2.2.3.1.1. La pasarela de seguridad: *PC4*

1. Editamos el archivo */etc/ipsec.secrets* y colocamos la siguiente entrada:

```
: PSK "Esteesmisecreto"
```

Nota: Cualquier cadena de caracteres entre comillas es válida como secreto compartido. No colocamos ninguna lista de identificadores para la clave ya que la empleamos como clave por defecto, es decir que será la que se utilice para todas las conexiones IPsec.

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn roadWarriorPSK*):

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces="ipsec0=eth1"
    uniqueids=yes

conn roadWarriorPSK
    type=tunnel
    authby=secret
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    right=%any                #¡OJO!#
    keyingtries=1
    auto=add
```

5.2.2.3.1.2. El cliente móvil: *PC2*

El proceso es similar al descrito para la pasarela, sólo que habrá que cambiar varios campos en el archivo *ipsec.conf*. Para evitar confusiones, detallamos todos los pasos necesarios a realizar en este equipo:

1. Editamos el archivo */etc/ipsec.secrets* y colocamos la siguiente entrada:

```
: PSK "Esteesmisecreto"
```

Nota: Cualquier cadena de caracteres entre comillas es válida como secreto compartido.

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn roadWarriorPSK*):

```

config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces=%defaultroute      #¡OJO!#
    uniqueids=yes

conn roadWarriorPSK
    type=tunnel
    authby=secret
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    right=%defaultroute          #¡OJO!#
    keyingtries=0
    auto=start                    #¡OJO!#

```

5.2.2.3.2. Autenticación por certificados X.509

Pasos para obtener los certificados X.509

1. En uno de los equipos (elegimos *PC4* por ser el equipo “fijo”), construimos nuestra propia Autoridad de Certificación (CA) mediante el comando *openssl* del terminal de Linux. Esto nos permitirá firmar nuestros propios certificados para las claves RSA:

```
openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem
```

Al ejecutar el comando habrá que rellenar una serie de parámetros sobre la CA que estamos creando. Podemos hacerlo a nuestro gusto, pero teniendo cuidado de poner lo mismo en el campo “*Organization Name*” que cuando se nos pregunte por ese campo al crear los requerimientos de certificado.

Esto genera una clave privada (*cakey.pem*), que moveremos a la carpeta */etc/ipsec.d/private*, y un certificado público firmado (*cacert.pem*), que moveremos a la carpeta */etc/ipsec.d/cacerts*.

2. En ambos equipos, *PC2* y *PC4*, creamos una clave privada RSA (en nuestro escenario son claves de 1024 bits) y su correspondiente petición de certificado, utilizando el comando *openssl* desde el directorio */etc/ipsec.d*:

```
PC2: openssl req -newkey rsa:1024 -keyout pc2key.pem -out pc2req.pem
```

```
PC4: openssl req -newkey rsa:1024 -keyout pc4key.pem -out pc4req.pem
```

Para crear la petición de certificado habrá que rellenar unos parámetros, con el requisito de introducir lo mismo que pusimos para la CA en el campo “*Organization*

Name". Como resultado del comando, se genera la clave privada (*pcXkey.pem*), que moveremos a la carpeta */etc/ipsec.d/private*, y una petición de certificación que deberá firmar la CA.

3. Para que nuestra CA pueda firmar las peticiones de certificación, tendremos que hacer una serie de cambios en la pasarela en la que fue creada: habrá que copiar el archivo */usr/share/ssl/openssl.cnf* al directorio */etc/ipsec.d/*, editarlo y modificar algunos valores. En concreto, bajo la sección "[*CA_default*]" asignaremos a la variable "*dir*" la expresión *./* (en lugar de *./demoCA*), y a la variable "*certificate*" le daremos el valor "*\$dir/cacerts/cacert.pem*". Además, tendremos que crear dos archivos en el directorio */etc/ipsec.d*: *index.txt*, que dejaremos vacío, y *serial*, que editaremos para escribir en él "01" [6].

4. La CA deberá firmar las peticiones de certificado, por lo que *PC2* tendrá que enviarle su petición (*pc2req.pem*) a *PC4*, que contiene la CA. Copiaremos esa petición al directorio */etc/ipsec.d* y desde ahí firmaremos ambas peticiones con el comando *openssl*:

```
openssl ca -in pc2req.pem -out pc2cert.pem -notext -config \
./openssl.cnf
openssl ca -in pc4req.pem -out pc4cert.pem -notext -config \
./openssl.cnf
```

Esto generará los certificados de clave pública para ambos equipos (*pcXcert.pem*).

5. Habrá que enviar estos certificados (*pcXcert.pem*), así como el certificado de la CA (*ca.cert.pem*) a *PC2*. Una vez copiados, ubicaremos los tres archivos en el directorio */etc/ipsec.d*.

Configuración de archivos

1. Para cada equipo, editamos el archivo */etc/ipsec.secrets* e introducimos lo siguiente (en lugar de indicar directamente una clave RSA indicamos el archivo en el que se encuentra, seguido de una clave opcional para aportar más seguridad):

```
PC2:   : RSA pc2key.pem "hola"
PC4:   : RSA pc4key.pem "hola"
```

Nota: La expresión "*hola*" es una clave opcional que debe ser la misma para ambos equipos. Cualquier cadena de caracteres entre comillas es válida para esto.

2. Para cada equipo, editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn roadWarriorX509*). Los parámetros *rightid* y *leftid* (teniendo en cuenta que *PC2* corresponde a *right*, y *PC4* a

left) se obtienen ejecutando, desde el directorio */etc/ipsec.d/*, el comando de *openssl* que extrae el identificador del certificado:

```
rightid → openssl x509 -in pc2cert.pem -noout -subject
```

```
leftid → openssl x509 -in pc4cert.pem -noout -subject
```

El archivo *ipsec.conf* queda así para cada equipo:

Pasarela de seguridad *PC4*:

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces="ipsec0=eth1"
    uniqueids=yes

conn roadWarriorX509
    type=tunnel
    authby=rsasig
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    leftid=<identificador certificado PC4>
    leftcert=pc4cert.pem
    right=%any #¡OJO!#
    rightid=<identificador certificado PC2>
    rightrsasigkey=%cert #¡OJO!#
    rightcert=pc2cert.pem
    keyingtries=1
    auto=add
```

Cliente móvil *PC2*:

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces=%defaultroute
    uniqueids=yes

conn roadWarriorX509
    type=tunnel
    authby=rsasig
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    leftid=<identificador certificado PC4>
    leftrsasigkey=%cert #¡OJO!#
    leftcert=pc4cert.pem
    right=%defaultroute #¡OJO!#
    rightid=<identificador certificado PC2>
```

```
rightcert=pc2cert.pem
keyingtries=0
auto=start                                #¡OJO!#
```

5.2.2.3.3. Autenticación por firmas RSA

5.2.2.3.3.1. La necesidad de tener un servidor DNS

Para poder emplear las firmas RSA como método de autenticación para clientes móviles, necesitamos añadir a la topología una tercera entidad: un servidor DNS. Como comentamos en la introducción, FreeS/WAN está construido de forma que necesita asociar las claves públicas RSA con algún parámetro identificativo, sea una dirección IP o un nombre de *host*. Esto es así para que puedan distinguirse las claves de los distintos clientes de una pasarela de seguridad. En nuestro escenario, el cliente móvil no tiene asignada una dirección IP fija, por lo que la pasarela reconocerá al cliente por su nombre de *host*. Una vez identificado el cliente, la pasarela consultará la clave pública RSA asociada a ese nombre de *host*, para comprobar que recibe la misma clave por parte del cliente. Sólo si se cumplen las dos condiciones la pasarela considerará autorizado al cliente móvil.

Ahora bien, antes de estas comprobaciones, FreeS/WAN se asegura, al ser arrancado en la pasarela, de que el identificador que tiene almacenado para el cliente es válido y está vigente. Con ese objetivo, realiza una consulta a su servidor DNS, preguntando por alguna dirección IP asociada al nombre de *host* que tiene registrado para el cliente. Si el DNS no tiene ningún registro que contenga el nombre de *host* del cliente, devolverá un mensaje de error a la pasarela, y ésta no admitirá ninguna conexión con el cliente (al no considerar válido su nombre de *host*). Para evitar este error, debemos construir nuestro propio servidor DNS, incluir un identificador para *PC2* en su base de datos, y configurar *PC4* para que realice la consulta a este DNS.

5.2.2.3.3.2. Cambios en la configuración del escenario

La inclusión del servidor DNS conlleva algunos cambios en la topología de red de nuestro escenario, como detallamos a continuación:

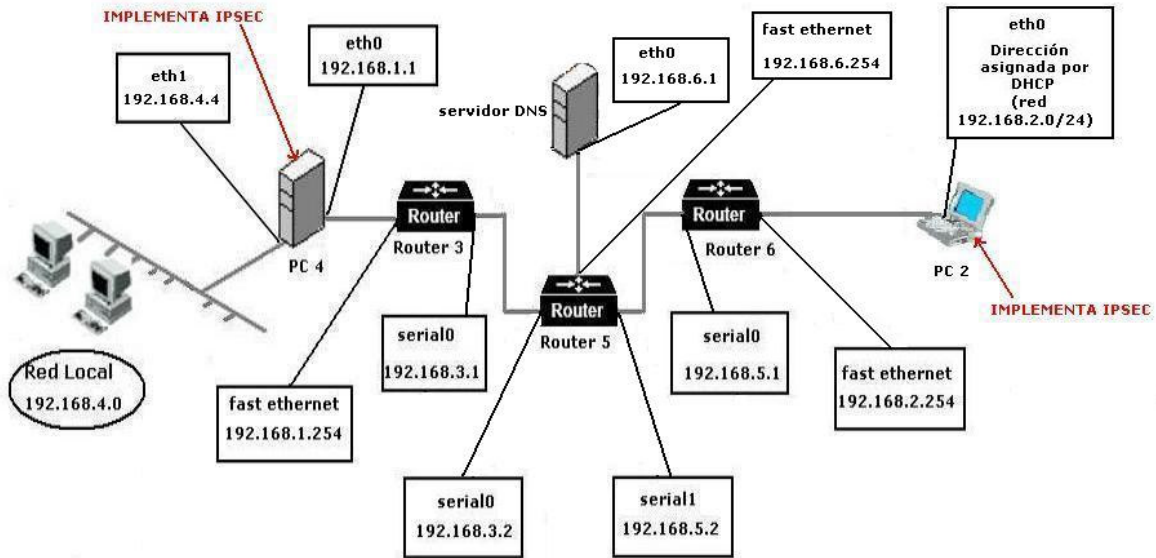


Figura 5.7 – Diagrama del escenario *Road Warrior* con DNS.

Estos cambios de topología implican algunas modificaciones en la configuración del *Router 5*, al que habrá que configurarle la interfaz *Fast Ethernet* con los siguientes parámetros (consultar apartado 5.2.2.1 para saber cómo se configuran las interfaces de red del *router*):

Router 5	Dirección IP	Máscara de red
Interfaz Fast Ethernet	192.168.6.254	255.255.255.0

Figura 5.8 – Parámetros para la interfaz *Fast Ethernet* del *Router 5*.

Además, habrá que incluir en las tablas de encaminamiento de los *Routers 3* y *6* la ruta hacia la red 192.168.6.0/24 (consultar apartado 5.2.2.1 para saber cómo se configuran las tablas de encaminamiento del *router*):

Router	Red destino	Máscara de red	Dirección IP siguiente pasarela
Router 3	192.168.6.0	255.255.255.0	192.168.3.2
Router 6	192.168.6.0	255.255.255.0	192.168.5.2

Figura 5.9 – Parámetros para las tablas de encaminamiento de los *Routers 3* y *6*.

5.2.2.3.3. Configuración del servidor DNS

1. Configuramos la dirección IP de la interfaz de red *eth0* del servidor DNS, mediante el comando del terminal de Linux *ifconfig*:

```
ifconfig eth0 192.168.6.1 netmask 255.255.255.0
```

2. Configuramos las tablas de encaminamiento del servidor DNS, mediante el comando del terminal de Linux *route*:

```
route add default gw 192.168.6.254
```

Puesto que el servidor DNS sólo tiene una interfaz de red activa(*eth0*), su única ruta de salida es la que sale por *eth0*, es decir, la conexión con la interfaz *Fast Ethernet* del *Router 5*. Podemos por tanto configurar como ruta de salida por defecto la dirección IP correspondiente a esta interfaz (192.168.6.254).

3. Asignamos un nombre de *host* al servidor DNS, mediante el comando *hostname* del terminal de Linux:

```
hostname pc5.it4.upct.es
```

4. Editamos el archivo */etc/named.conf*, que permite configurar los parámetros del servidor DNS. En este archivo habrá que realizar los cambios siguientes:

- Incluir la dirección de *PC4* en la lista de direcciones IP a las que se permite realizar consultas: colocamos la dirección 192.168.1.1 dentro de las llaves que aparecen tras la expresión *allow-query*.

```
allow-query {192.168.1.1 ...}
```

- Al final del archivo, incluimos la configuración de la zona correspondiente a nuestro escenario:

```
zone "it4.upct.es" in {
type master;
file "it4.upct.es.zone";
}
```

Con estas líneas, estamos especificando que existe una zona llamada *it4.upct.es*, a la que habrá que dar servicio como servidor DNS maestro (*type master*), y cuya información sobre equipos, direcciones y demás se encuentra en el archivo llamado *it4.upct.es.zone*.

5. Creamos el archivo de texto */var/named/it4.upct.es.zone*, que contendrá la información necesaria para el DNS sobre los equipos de la zona *it4.upct.es* [7].

Escribimos las líneas siguientes en este archivo:

```

it4.upct.es.      IN      SOA      pc5.it4.upct.es.  root.it4.upct.es. (
                  2004092102  ; serial
                  1D          ; Refresh
                  1H          ; Retry
                  1w          ; Expire
                  2D )        ; Default TTL
;servidor de nombres
it4.upct.es.      IN      NS      pc5.it4.upct.es.

;mapeado de nombres-direcciones
pc2.it4.upct.es.  IN      A      0.0.0.0
pc4.it4.upct.es.  IN      A      192.168.1.1
pc5.it4.upct.es.  IN      A      192.168.6.1

```

La única línea que nos es realmente de utilidad es “`pc2.it4.upct.es. IN A 0.0.0.0`”. Con ella, añadimos el nombre de *host* para *PC2* (`pc2.it4.upct.es`) a la base de datos del servidor DNS, asociándolo con la dirección `0.0.0.0`. Colocamos esta dirección ya que desconocemos cuál será la dirección IP del cliente (puesto que no es fija). Lo importante es que en el registro del servidor DNS conste el nombre de *host* de *PC2*, de forma que, cuando la pasarela pregunte por ese nombre, el servidor DNS pueda devolverle alguna respuesta (así la pasarela sabrá que el nombre es válido).

6. Una vez configurados los archivos necesarios al funcionamiento de DNS, arrancamos la aplicación que hace que el equipo se comporte como servidor DNS:

```
ndc start
```

5.2.2.3.3.4. Configuración complementaria para *PC2* y *PC4*

Todos los pasos para configurar la pasarela y el cliente móvil, descritos en el apartado 5.2.2.2, son válidos para este caso. De forma complementaria, habrá que añadir algunos pasos para adaptar los equipos a la aparición del servidor DNS:

1. Hay que asignar nombres de *host* a la pasarela y al cliente móvil, para que puedan ser identificados entre sí y por el servidor DNS. Utilizamos el comando `hostname` del terminal de Linux:

```
PC2: hostname pc2.it4.upct.es
```

```
PC4: hostname pc4.it4.upct.es
```

2. *PC4* debe tener asignado nuestro servidor DNS como servidor DNS primario, de forma que todas sus consultas sean enviadas a éste. Conseguimos esto editando el archivo `/etc/resolv.conf` en *PC4*, y colocando por encima de la primera línea en la que aparezca la expresión `nameserver` la siguiente línea:


```
nameserver 192.168.6.1
```

Asignamos así la dirección *192.168.6.1* como dirección del servidor DNS primario, al que dirigirá sus consultas *PC4*.

5.2.2.3.3.5. Configuración de los archivos de FreeS/WAN

5.2.2.3.3.5.1. La pasarela de seguridad: *PC4*

1. Creamos un par de claves RSA mediante el comando *ipsec rsasigkey*, y las incluimos en el fichero */etc/ipsec.secrets* (en nuestro escenario serán claves de 2048 bits Junto a la clave RSA, habrá que incluir unos parámetros para que FreeS/WAN reconozca el formato del archivo (emplearemos el comando de Linux *echo*):

```
echo ": RSA {" > /etc/ipsec.secrets
ipsec rsasigkey 2048 >> /etc/ipsec.secrets
echo "}" >> /etc/ipsec.secrets
```

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn roadWarriorRSA*). Rellenamos los campos de claves públicas RSA como sigue: en *leftrsasigkey* copiaremos la cadena de caracteres que aparece a continuación de “*#pubkey=*” en el archivo *ipsec.secrets* de *PC2*, y en *rightrsasigkey* haremos lo mismo pero con el archivo *ipsec.secrets* de *PC4*.

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces="ipsec0=eth1"
    uniqueids=yes

conn roadWarriorRSA
    type=tunnel
    authby=rsasig
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    leftrsasigkey=<clave pública RSA PC4>
    right=%any #¡OJO!#
    rightid=pc2.it4.upct.es
    rightrsasigkey=<clave pública RSA PC2>
    keyingtries=1
    auto=add
```

5.2.2.3.3.5.2. El cliente móvil: PC2

El proceso es similar al descrito para la pasarela, sólo que habrá que cambiar varios campos en el archivo *ipsec.conf*. Para evitar confusiones, detallamos todos los pasos necesarios a realizar en este equipo:

1. Creamos un par de claves RSA mediante el comando *ipsec rsasigkey*, y las incluimos en el fichero */etc/ipsec.secrets* (en nuestro escenario serán claves de 2048 bits Junto a la clave RSA, habrá que incluir unos parámetros para que FreeS/WAN reconozca el formato del archivo (emplearemos el comando de Linux *echo*):

```
echo ": RSA {" > /etc/ipsec.secrets
ipsec rsasigkey 2048 >> /etc/ipsec.secrets
echo "}" >> /etc/ipsec.secrets
```

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn roadWarriorRSA*). Rellenamos los campos de claves públicas RSA como sigue: en *leftrsasigkey* copiaremos la cadena de caracteres que aparece a continuación de "*#pubkey=*" en el archivo *ipsec.secrets* de *PC2*, y en *rightrsasigkey* haremos lo mismo pero con el archivo *ipsec.secrets* de *PC4*.

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces=%defaultroute           #¡OJO!#
    uniqueids=yes

conn roadWarriorRSA
    type=tunnel
    authby=rsasig
    left=192.168.1.1
    leftnexthop=192.168.1.254
    leftsubnet=192.168.4.0/24
    leftrsasigkey=<clave pública RSA PC4>
    right=%defaultroute                 #¡OJO!#
    rightid=pc2.it4.upct.es
    rightrsasigkey=<clave pública RSA PC2>
    keyingtries=0
    auto=start                           #¡OJO!#
```

5.3. Puesta en marcha del túnel IPSec

Una vez terminado todo el trabajo de configuración, es hora de arrancar la aplicación y comprobar que el túnel IPSec se establece correctamente. Este es el comando que arranca FreeS/WAN (habrá que ejecutarlo en los dos equipos):

```
ipsec setup start
```

Para este escenario, *PCA* desconoce las direcciones desde las que le llegarán propuestas de conexión, y deberá permanecer a la escucha de peticiones autorizadas. Es por eso que tendremos que arrancar primero FreeS/WAN en la pasarela, de modo que cuando ejecutemos la aplicación en el cliente, éste entre directamente en negociaciones con la pasarela para establecer el túnel IPSec (si lo hacemos al revés y arrancamos primero el proceso en el cliente móvil, éste no recibirá respuesta por parte de la pasarela y no podrá haber conexión segura).

Se puede verificar que no ha habido ningún error mediante el comando `ipsec barf`, que nos devuelve información detallada acerca de las acciones de Pluto. El Apéndice B muestra los mensajes que devuelve `ipsec barf` si todo funciona correctamente.

6. Escenario 3: Conexión “Oportunista”

6.1. Introducción

Este escenario detalla los pasos necesarios para construir una conexión IPSec, basada en FreeS/WAN, entre dos usuarios cualesquiera, que no dispongan de información previa el uno del otro.

El objetivo de los creadores de FreeS/WAN es conseguir que este modo de conexión se convierta en un estándar para Internet, de modo que todos los usuarios de IPSec puedan establecer conexiones seguras sin necesidad de tener que intercambiar claves entre ellos. La inclusión de los certificados digitales X.509, como opción para autenticar a los usuarios de FreeS/WAN, ha hecho que el proyecto “Oportunista” pierda fuerza: los certificados ofrecen un sistema de autenticación fiable y más eficiente que las consultas al servidor DNS. No obstante, este tipo de conexión no deja de ser interesante como opción para permitir un uso universal del protocolo de seguridad IPSec.

Este tipo de conexión se basa en usar un servidor DNS como base de datos de las claves de autenticación de los posibles clientes “oportunistas” IPSec. Así, cada vez que un usuario “oportunista” desea establecer una comunicación IP segura con un destino cualquiera, consulta antes a su servidor DNS, para obtener información acerca de ese destino. Si el servidor DNS tiene alguna clave de autenticación asociada a dicho destino, la enviará al usuario que realizó la consulta, de modo que pueda establecer una conexión IPSec con él. Esto sólo podrá funcionar si el destino también es un usuario de FreeS/WAN “oportunista”; de no ser así, el servidor DNS no dispondrá de una clave asociada a éste. Al recibir una propuesta de conexión IPSec, el usuario destino consultará al servidor DNS, para obtener la clave de autenticación del otro extremo de la comunicación (el origen de la propuesta). Una vez que ambos extremos poseen la clave del otro, puede establecerse una conexión IPSec directa. En caso de que no se reciba ninguna respuesta del servidor DNS (para alguno de los dos extremos), no habrá conexión IPSec, pero por supuesto, si lo desean, los usuarios podrán comunicarse de manera no segura.

Teóricamente, las consultas al servidor DNS siguen el mismo proceso que las consultas de resolución de direcciones IP. Es decir, que si un servidor DNS no tiene información acerca de un usuario, pasará la consulta al siguiente servidor DNS, siguiendo la estructura jerárquica de servidores. No hemos encontrado información que confirme esta afirmación, por lo que, en nuestras pruebas, empleamos el mismo servidor DNS para ambos extremos de la conexión. En cuanto a las claves de autenticación de *host*

almacenadas en la base de datos del servidor DNS, irán siempre asociadas a una dirección IP, y a un nombre de *host* (correspondientes al usuario).

Podemos decir, por tanto, que se trata de una conexión directa IPSec, que puede realizarse sin conocer previamente la clave de autenticación del otro extremo, pero que necesita para ello de la colaboración de un servidor DNS.

El establecimiento de una conexión “Oportunista” conlleva los siguientes requisitos:

- La conexión “Oportunista” sólo puede hacerse utilizando las firmas digitales RSA como método de autenticación de *host*, ya que son las únicas que admite el servidor DNS en su base de datos.
- Los dos extremos de la conexión deberán tener un servidor DNS por defecto que les permita acceder a la información sobre claves de autenticación de otros usuarios “oportunistas”. Esto quiere decir que no podrá tratarse de un servidor DNS aislado (sin posibilidad de consultar a otro servidor DNS) o bien será un servidor DNS especializado en contener claves de autenticación de *host* para este propósito [8]. En nuestro escenario, hemos optado por utilizar el mismo servidor DNS para ambos usuarios, así nos aseguramos de que los dos recibirán una respuesta afirmativa (acompañada de una clave de autenticación) al consultar sobre el otro usuario.
- Antes de cualquier conexión entre usuarios, es muy recomendable establecer una conexión IPSec, entre los usuarios “oportunistas” y el servidor DNS, para que el tráfico entre éstos quede protegido. En nuestro caso, para estos enlaces seguros, emplearemos también una conexión de tipo “Oportunista”.

Nota: La configuración que exponemos a continuación fue realizada en un laboratorio de la Universidad Politécnica de Cartagena para comprobar su correcto funcionamiento. Si quisiera emplearse este documento para crear una conexión personalizada, bastaría con cambiar, en el archivo de configuración *ipsec.conf*, así como en los archivos del servidor DNS, las direcciones IP que aparecen aquí por las que vayan a usarse.

6.2. Configuración del escenario

6.2.1. Diagramas de situación

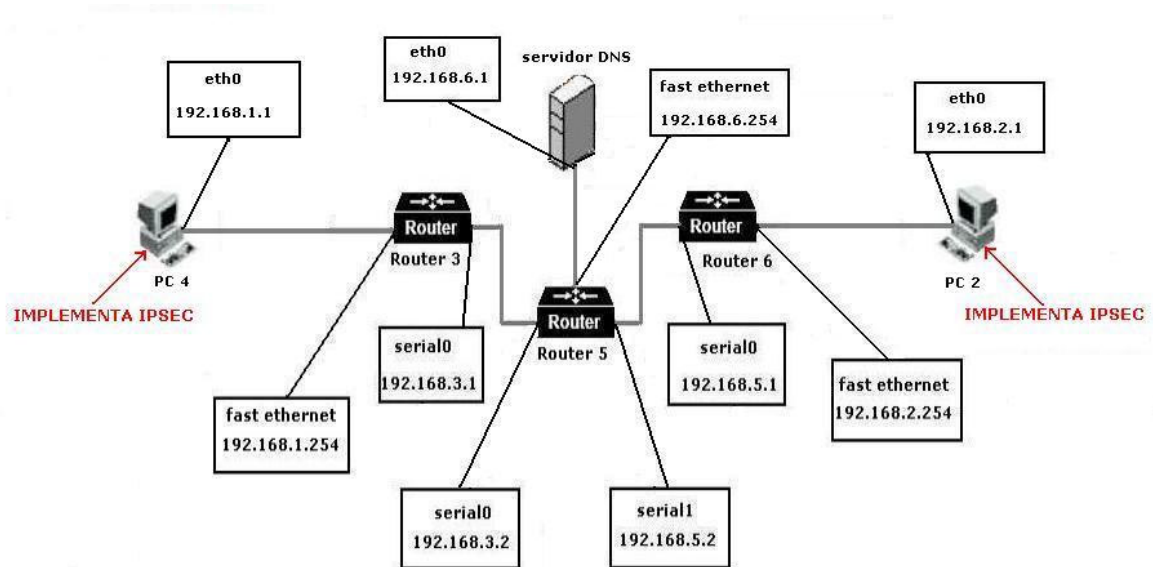


Figura 6.1 – Diagrama del escenario *Opportunistic*.

La *figura 6.1* muestra la topología diseñada para probar la configuración de la conexión “Oportunista”. *PC2* y *PC4* son los usuarios “oportunistas” (en adelante usuarios), que implementan FreeS/WAN. El servidor DNS está configurado como servidor de nombres por defecto para los dos usuarios, y contiene la información sobre las claves de autenticación de ambos. De este modo, los dos podrán obtener la clave del otro cuando consulten al servidor DNS, lo que permitirá establecer la conexión IPsec. Los *routers* situados entre las pasarelas tienen como objetivo simular la infraestructura de una red de interconexión pública (por ejemplo, Internet), una “nube”, de modo que la ruta que conecte las dos pasarelas no sea directa. Obtenemos así un escenario comparable (a escala, en el laboratorio) a las situaciones más comunes, en las que la conexión IPsec debe hacerse a través de la infraestructura de redes públicas.

Para cada equipo de red utilizado (pasarelas y *routers*), indicamos en la figura las interfaces de red usadas y sus direcciones IP correspondientes.

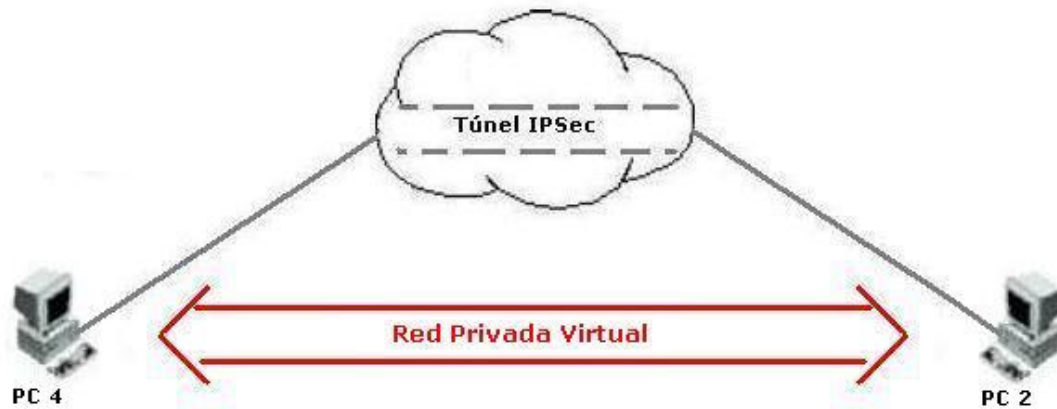


Figura 6.2 – Diagrama equivalente túnel IPsec.

6.2.2. Configuración de equipos paso a paso

A diferencia de los escenarios anteriores, en este caso sólo podremos emplear un tipo de autenticación de usuario: las firmas digitales RSA. En primer lugar, detallaremos la configuración de los *routers* presentes en la topología. Seguidamente, presentaremos la configuración del servidor DNS. Y, por último, la configuración de los usuarios, *PC2* y *PC4*.

6.2.2.1. Configuración *Routers* CISCO 1751

La función de estos equipos de red se limita, en nuestro trabajo, a ofrecer interconexión entre las distintas redes de la infraestructura. Para lograrlo, debemos configurar las interfaces de red y las tablas de encaminamiento de cada *router* [5].

Hemos elegido la aplicación *Microsoft Hyperterminal* para acceder a los menús de configuración de los *routers*, pero cualquier otro *software* de emulación de Terminal puede emplearse. Conectamos, en primer lugar, la entrada del *router* denominada *Console* con el puerto serie del ordenador que ejecuta *Hyperterminal*, para que pueda haber comunicación. Arrancamos el programa y, tras darle un nombre a la conexión, debemos configurar las siguientes características para conectar con el *router*:

- 9600 baudios.
- 8 bits de datos.

- Sin bit de paridad.
- 1 bit de stop.
- Sin control de flujo (en inglés *flow control*).

Hecho esto, el terminal conecta con el *router*, y tras unos instantes debe aparecer un mensaje como este en consola:

Router> (o en su defecto el identificador asignado al *router*)

A partir de este momento estamos, en comunicación directa con el *router*, y tendremos que emplear los comandos que “comprenda” su sistema operativo para configurarlo. En todo momento podremos introducir “?” para conocer los distintos comandos posibles.

1. Comenzamos por pasar a modo privilegiado, lo que nos dará autorización para modificar todos los parámetros del *router*:

Router> enable

Sabremos que estamos en modo privilegiado porque el mensaje de consola pasa a ser *Router#*

2. Entramos en el menú de configuración:

Router# configure terminal

Sabremos que estamos en el menú de configuración porque el mensaje de consola pasa a ser *Router(config)#*

3. Entramos en el menú de la interfaz *Fast Ethernet*:

Router(config)# configure interface fastethernet 0/0

Sabremos que estamos en el menú de una interfaz de red porque el mensaje de consola pasa a ser *Router(config-if)#*

4. Configuramos la dirección IP de esta interfaz de red:

Router(config-if)# ip address <dirección IP> <máscara de red>

Con este comando indicaremos la dirección IP que queremos asignar a la interfaz de red, seguida de la máscara de red aplicable a dicha dirección.

5. Activamos esta interfaz de red:

Router(config-if)# no shutdown

6. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

7. Entramos en el menú de la interfaz *Serial0*:

```
Router(config)# configure interface serial 1/0
```

8. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

9. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

10. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

11. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

12. Salimos del menú de la interfaz de red:

```
Router(config-if)# exit
```

13. Entramos en el menú de la interfaz *Serial1*:

```
Router(config)# configure interface serial 1/1
```

14. Configuramos la velocidad de conexión de la interfaz:

```
Router(config-if)# clock rate 56000
```

Nota: Este paso sólo tendremos que realizarlo en los *routers* que usen el conector DCE.

15. Configuramos el método de encapsulación de paquetes:

```
Router(config-if)# encapsulation hdlc
```

16. Configuramos la dirección IP de la interfaz:

```
Router(config-if)# ip address <dirección IP> <máscara de red>
```

17. Activamos esta interfaz de red:

```
Router(config-if)# no shutdown
```

18. Salimos del menú de la interfaz de red:

Router(config-if)# exit

19. Rellenamos la tabla de encaminamiento del *router* empleando tantas veces como sea necesario este comando:

Router(config)# ip route <dirección de red destino> <máscara de red> <siguiente pasarela>

Para cada ruta, habrá que indicar la dirección IP de la red destino, seguido de la máscara de red aplicable a esta dirección, y de la dirección IP de la siguiente pasarela en la ruta hacia la red destino.

20. Salimos del menú de configuración:

Router(config)# exit

21. Grabamos los cambios efectuados en la configuración en la memoria de arranque del *router*:

Router(config)# copy run start

Nota: Los pasos de configuración de la interfaz *Serial1* (13 a 18) sólo son aplicables al *Router 5*, el único que utiliza esta interfaz.

A continuación, indicamos las direcciones IP (por medio de los comandos descritos arriba) a asignar a cada interfaz de red, para cada *router*, así como sus respectivas tablas de encaminamiento:

	ROUTER 3		ROUTER 5		
Interfaz	Fast Ethernet	Serial 0	Serial 0	Serial 1	Fast Ethernet
Dirección IP	192.168.1.254	192.168.3.1	192.168.3.2	192.168.5.2	192.168.6.254
Máscara de red	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Figura 6.3a – Tabla de asignación de direcciones IP de los *routers* 3 y 5.

ROUTER 6		
Interfaz	Fast Ethernet	Serial0
Dirección IP	192.168.2.254	192.168.5.1
Máscara de red	255.255.255.0	255.255.255.0

Figura 6.3b – Tabla de asignación de direcciones IP del *router* 6.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.2.0	255.255.255.0	192.168.3.2
192.168.5.0	255.255.255.0	192.168.3.2
192.168.6.0	255.255.255.0	192.168.3.2

Figura 6.4 – Tabla de encaminamiento del *Router* 3.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.3.1
192.168.2.0	255.255.255.0	192.168.5.1

Figura 6.5 – Tabla de encaminamiento del *Router* 5.

Red destino	Máscara de red	Dirección IP siguiente pasarela
192.168.1.0	255.255.255.0	192.168.5.2
192.168.3.0	255.255.255.0	192.168.5.2
192.168.6.0	255.255.255.0	192.168.5.2

Figura 6.6 – Tabla de encaminamiento del *Router* 6.

6.2.2.2. Configuración del servidor DNS

6.2.2.2.1. Parámetros de interconexión

1. Configuramos la dirección IP de la interfaz de red *eth0* del servidor DNS, mediante el comando del terminal de Linux *ifconfig*:

```
ifconfig eth0 192.168.6.1 netmask 255.255.255.0
```

2. Configuramos las tablas de encaminamiento del servidor DNS, mediante el comando del terminal de Linux *route*:

```
route add default gw 192.168.6.254
```

Puesto que el servidor DNS sólo tiene una interfaz de red activa (*eth0*), su única ruta de salida es la que sale por *eth0*, es decir, la conexión con la interfaz *Fast Ethernet* del *Router 5*. Podemos por tanto configurar como ruta de salida por defecto la dirección IP correspondiente a esta interfaz (192.168.6.254).

3. Asignamos un nombre de *host* al servidor DNS, mediante el comando *hostname* del terminal de Linux:

```
hostname pc5.it4.upct.es
```

6.2.2.2.2. Configuración y puesta en marcha del servidor de nombres

1. Editamos el archivo */etc/named.conf*, que permite configurar los parámetros del servidor DNS. En este archivo habrá que realizar los cambios siguientes:

- Incluir las direcciones de *PC2* y *PC4* en la lista de direcciones IP a las que se permite realizar consultas: colocamos las direcciones 192.168.1.1 y 192.168.2.1 dentro de las llaves que aparecen tras la expresión *allow-query*.

```
allow-query {192.168.1.1; 192.168.2.1 ...}
```

- Al final del archivo, incluimos la configuración de la zona correspondiente a nuestro escenario:

```
zone "it4.upct.es" in {
    type master;
    file "it4.upct.es.zone";
}
```

Con estas líneas, estamos especificando que existe una zona llamada *it4.upct.es*, a la que habrá que dar servicio como servidor DNS maestro (*type master*), y cuya información sobre equipos, direcciones y demás parámetros se encuentra en el archivo llamado *it4.upct.es.zone*.

De manera análoga, habrá que añadir la configuración de las zonas correspondientes a nuestros dos usuarios, para que puedan obtenerse sus claves de autenticación a partir de sus direcciones IP:

```
zone 1.168.192.in-addr.arpa" in {
type master;
file "192.168.1.zone";
}

zone 2.168.192.in-addr.arpa" in {
type master;
file "192.168.2.zone";
}
```

Estas líneas corresponden a la especificación de dos nuevas zonas, correspondientes a las redes en las que se encuentran nuestros clientes “oportunistas”, y a las que habrá que dar servicio como servidor DNS maestro (*type master*). La información sobre sus equipos, direcciones y demás parámetros se encontrará, respectivamente, en los archivos llamados *192.168.1.zone* y *192.168.2.zone*.

2. Creamos el archivo de texto */var/named/it4.upct.es.zone* (archivo de zona), que contendrá la información necesaria para el servidor DNS sobre los equipos de la zona *it4.upct.es*. Escribimos las líneas siguientes en este archivo:

```
it4.upct.es.      IN      SOA      pc5.it4.upct.es.  root.it4.upct.es. (
                2004092102 ; serial
                1D      ; Refresh
                1H      ; Retry
                1w      ; Expire
                2D )    ; Default TTL
;servidor de nombres
it4.upct.es.     IN      NS       pc5.it4.upct.es.

;mapeado de nombres-direcciones
pc2.it4.upct.es. IN      A       192.168.2.1
pc4.it4.upct.es. IN      A       192.168.1.1
pc5.it4.upct.es. IN      A       192.168.6.1
```

Los archivos de zona tienen siempre una estructura similar:

- La primera línea comienza por el nombre de la zona. A continuación, se indica el inicio de autoridad de la zona (SOA – *Start Of Authority*), en este caso *pc5.it4.upct.es*, y el usuario responsable de la misma (*root.it4.upct.es*).
- Los valores que aparecen entre paréntesis son parámetros de configuración necesarios para el DNS, tales como la versión del archivo (*serial*) o los tiempos límite para distintas acciones (refrescar, reintentar, desechar información sobre los clientes o dirigida a los clientes).
- Seguidamente, aparece el servidor de nombres de la zona (*pc5.it4.upct.es*), siempre precedido de la expresión **NS**.

- Por último, un listado con los nombres de *hosts* de la zona, a los que se asocia con su respectiva dirección IP (uno por línea). Sabremos que se trata de una asignación nombre-dirección por la expresión **A**, colocada antes de la dirección IP.

3. Creamos el archivo de texto `/var/named/192.168.1.zone` (archivo de zona), que contendrá la información necesaria para el servidor DNS sobre el usuario *PC4*. Escribimos las líneas siguientes en este archivo:

```
1.168.192.in-addr.arpa.    IN SOA    pc4.it4.upct.es.
                               (...) root.it4.upct.es.  (
                               2004092102 ; serial
                               1D       ; Refresh
                               1H       ; Retry
                               1w       ; Expire
                               2D )    ; Default TTL
;servidor de nombres
1.168.192.in-addr.arpa.    IN NS     pc5.it4.upct.es.

;mapeado de direcciones-nombres
1.1.168.192.in-addr.arpa. IN PTR     pc4.it4.upct.es.
1.1.168.192.in-addr.arpa. IN KEY     0x4200 4 1 <clave pública RSA
                               (...)PC4>
1.1.168.192.in-addr.arpa. IN TXT     "X-Ipsec-Server(10)=192.168.1.1
                               (...)<clave pública RSA PC4>"
```

Describimos las partes del archivo de zona que no aparecían en el punto anterior:

- La expresión **PTR** indica que se trata de un puntero, es decir que se asocia la dirección IP (aparece en su forma inversa, es decir *1.1.168.192.in-addr.arpa*.)
- Las líneas con **KEY** contienen la clave de autenticación del usuario que devolverá el servidor DNS cuando otros usuarios le consulten acerca de él. Obtendremos el contenido de estas líneas ejecutando, de nuevo, el comando *ipsec showhostkey* en los equipos de los usuarios, pero sin ningún argumento:

```
ipsec showhostkey
```

- Las líneas con **TXT** contienen la clave de autenticación que recibirá el usuario de FreeS/WAN, cuando pida su propia clave al servidor DNS (es la garantía de que se trata de un usuario IPsec autorizado). Para cada archivo de zona, obtendremos la expresión a colocar en estas líneas empleando, en el equipo de cada usuario, el comando del terminal de Linux *ipsec showhostkey*:

```
ipsec showhostkey --txt <dirección IP de la interfaz que usará
FreeS/WAN>
```

Este comando devuelve la línea completa, a excepción de la dirección IP en su forma inversa (*X.X.X.X.in-addr.arpa*), esto es: `"IN TXT X-Ipsec-Server(10)=<dirección IP> <clave pública RSA>"`.

Habrá que enviar las claves en formatos TXT y KEY, desde cada usuario al servidor DNS (en archivos de texto por ejemplo), para que la incluya en los archivos de zona.

4. Creamos el archivo de texto */var/named/192.168.2.zone* (archivo de zona), que contendrá la información necesaria para el servidor DNS sobre el usuario *PC2*. Escribimos las líneas siguientes en este archivo:

```
2.168.192.in-addr.arpa.    IN SOA    pc2.it4.upct.es.
                                (...)root.it4.upct.es.    (
                                2004092102    ; serial
                                1D           ; Refresh
                                1H           ; Retry
                                1w           ; Expire
                                2D )        ; Default TTL
;servidor de nombres
2.168.192.in-addr.arpa.    IN NS     pc5.it4.upct.es.

;mapeado de direcciones-nombres
1.2.168.192.in-addr.arpa. IN PTR     pc2.it4.upct.es.
1.2.168.192.in-addr.arpa. IN KEY     0x4200 4 1 <clave pública RSA
                                (...)PC2>
1.2.168.192.in-addr.arpa. IN TXT     "X-Ipsec-Server(10)=192.168.2.1
                                (...)<clave pública RSA PC2>"
```

5. Una vez configurados los archivos necesarios al funcionamiento de DNS, arrancamos la aplicación que hace que el equipo se comporte como servidor DNS:

```
ndc start
```

6.2.2.2.3. Configuración de los archivos de FreeS/WAN

Tal y como adelantamos en la introducción, es muy recomendable que exista una conexión IPsec entre el servidor DNS y sus clientes, para que se pueda proteger todo el tráfico entre servidor de nombres y usuarios “oportunistas”. Para construir estas conexiones IPsec, el servidor DNS tendrá que enviar su clave pública RSA a todos sus clientes, y obtendrá sus claves de autenticación en la propia base de datos del servicio DNS.

A continuación, mostramos la configuración de los archivos de FreeS/WAN en el servidor DNS:

1. Creamos un par de claves RSA mediante el comando *ipsec rsasigkey*, y las incluimos en el fichero */etc/ipsec.secrets* (en nuestro escenario serán claves de 2048 bits Junto a la clave RSA, habrá que incluir unos parámetros para que FreeS/WAN reconozca el formato del archivo (emplearemos el comando de Linux *echo*):


```
echo ": RSA {" > /etc/ipsec.secrets
ipsec rrsasigkey 2048 >> /etc/ipsec.secrets
echo "}" >> /etc/ipsec.secrets
```

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta con agregar al final de éste la sección *conn opportunistic*). Rellenamos los campos de claves públicas RSA como sigue: copiaremos en *rightrrsasigkey* la cadena de caracteres que aparece a continuación de “*#pubkey=*” en el archivo *ipsec.secrets* del servidor DNS, y en *leftrrsasigkey* el valor *%dns*.

```
config setup
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    interfaces="ipsec0=eth0"           #¡OJO!#
    uniqueids=yes

conn opportunistic
    type=tunnel
    authby=rsasig
    right=192.168.6.1
    rightright=192.168.6.254
    rightrrsasigkey=<clave pública RSA servidor DNS>
    left=%opportunistic               #¡OJO!#
    leftrrsasigkey=%dns                #¡OJO!#
    auto=add
```

6.2.2.3. Configuración de los usuarios: *PC2* y *PC4*

6.2.2.3.1. Parámetros de interconexión y DNS

1. Configuramos las direcciones IP de las interfaces de red de los usuarios, mediante el comando del terminal de Linux *ifconfig*:

Uso de *ifconfig*: *ifconfig* <interfaz> <dirección a asignar> netmask <máscara de red>

PC2: *ifconfig* eth0 192.168.2.1 netmask 255.255.255.0

PC4: *ifconfig* eth0 192.168.1.1 netmask 255.255.255.0

2. Configuramos las tablas de encaminamiento de las pasarelas, mediante el comando del terminal de Linux *route*:

Uso de *route* para añadir una entrada a la tabla de encaminamiento:

route add -net <dirección de red destino> netmask <máscara de red> gw <dirección de la siguiente pasarela en ruta hacia red destino>

```
PC2: route add -net 192.168.1.0 netmask 255.255.255.0 gw
      192.168.2.254
```

```
PC4: route add -net 192.168.2.0 netmask 255.255.255.0 gw
      192.168.1.254
```

3. Tendremos que asegurarnos de que cada usuario tenga un identificador (un nombre de *host*) único, que coincida con el que incluiremos en los archivos del servidor DNS. Para conseguirlo, son necesarios dos pasos en *PC2* y *PC4*:

- Editamos el archivo */etc/hosts*, que contiene los nombres de *host* asignados a cada dirección IP, y eliminamos todos los identificadores presentes. De este modo, el cliente no tendrá un nombre de *host* predefinido para ninguna dirección.
- Asignamos un nombre de *host* al cliente mediante el comando del terminal de Linux *hostname*:

```
PC2: hostname pc2.it4.upct.es
```

```
PC4: hostname pc4.it4.upct.es
```

4. Los dos usuarios deben tener asignado nuestro servidor DNS como servidor DNS primario, de forma que todas sus consultas sean enviadas a éste. Conseguimos esto editando el archivo */etc/resolv.conf* en *PC2* y *PC4*, y colocando por encima de la primera línea en la que aparezca la expresión *nameserver* la siguiente línea:

```
nameserver 192.168.6.1
```

Asignamos así la dirección *192.168.6.1* como dirección del servidor DNS primario, al que dirigirán sus consultas los dos usuarios.

6.2.2.3.2. Archivos de FreeS/WAN

1. En los dos usuarios, creamos un par de claves RSA mediante el comando *ipsec rsasigkey*, y las incluimos en el fichero */etc/ipsec.secrets* (en nuestro escenario serán claves de 2048 bits) Junto a la clave RSA, habrá que incluir unos parámetros para que FreeS/WAN reconozca el formato del archivo (emplearemos el comando de Linux *echo*):

```
echo ": RSA {" > /etc/ipsec.secrets
ipsec rsasigkey 2048 >> /etc/ipsec.secrets
echo "}" >> /etc/ipsec.secrets
```

2. Editamos el archivo */etc/ipsec.conf* y copiamos la configuración que sigue (mostramos el archivo en su totalidad, pero si ya hay definida alguna conexión basta

con agregar al final de éste las secciones *conn opportunistic* y *conn clientToDNS*. Rellenamos los campos de claves públicas RSA como sigue: en la sección *clientToDNS*, copiaremos en *leftrsasigkey* la cadena de caracteres que aparece a continuación de “*#pubkey=*” en el archivo *ipsec.secrets* del usuario. En *rightrsasigkey*, haremos lo mismo con la clave pública del servidor DNS.

```
PC2:  config setup
      klipsdebug=none
      plutodebug=none
      plutoload=%search
      plutostart=%search
      interfaces="ipsec0=eth0"           #¡OJO!#
      uniqueids=yes

      conn opportunistic
      type=tunnel
      authby=rsasig
      left=192.168.2.1
      leftnexthop=192.168.2.254
      leftrsasigkey=%dns                 #¡OJO!#
      right=%opportunistic              #¡OJO!#
      rightrsasigkey=%dns               #¡OJO!#
      keyingtries=1
      auto=route                         #¡OJO!#

      conn clientToDNS
      type=tunnel
      authby=rsasig
      left=192.168.2.1
      leftnexthop=192.168.2.254
      leftrsasigkey=<clave pública RSA PC2>
      right=192.168.6.1
      rightnexthop=192.168.6.254
      rightrsasigkey=<clave pública RSA servidor DNS>
      auto=start                         #¡OJO!#
```

```
PC4:  config setup
      klipsdebug=none
      plutodebug=none
      plutoload=%search
      plutostart=%search
      interfaces="ipsec0=eth0"           #¡OJO!#
      uniqueids=yes

      conn opportunistic
      type=tunnel
      authby=rsasig
      left=192.168.1.1
      leftnexthop=192.168.1.254
      leftrsasigkey=%dns                 #¡OJO!#
      right=%opportunistic              #¡OJO!#
```

```

rightrsasigkey=%dns           #¡OJO!#
keyingtries=1
auto=route                     #¡OJO!#

conn clientToDNS
type=tunnel
authby=rsasig
left=192.168.1.1
leftnexthop=192.168.1.254
leftrsasigkey=<clave pública RSA PC4>
right=192.168.6.1
rightnexthop=192.168.6.254
rightrsasigkey=<clave pública RSA servidor DNS>
auto=start                     #¡OJO!#

```

Nota: En la sección *opportunistic* de ambos usuarios, configuramos el parámetro *leftrsasigkey* con el valor *%dns*, lo que significa que los usuarios obtendrán su propia clave de autenticación del servidor DNS. Nos aseguramos así de que las claves de los dos usuarios están correctamente almacenadas en el servidor DNS (si no es así, FreeS/WAN nos dará un error al arrancar y solicitar su clave al servidor DNS). Para otros casos, en los que el servidor DNS se encuentre alejado del usuario, recomendamos introducir directamente la clave pública RSA en el parámetro *leftrsasigkey*.

6.2.2.3.3. *Scripts* de ayuda a la configuración

Para facilitar la configuración de los usuarios *PC2* y *PC4*, hemos creado dos archivos ejecutables: *extraeClave* y *configConexionOportunista*.

El archivo *extraeClave* podrá ejecutarse en el servidor DNS una vez configurados sus archivos de FreeS/WAN. Su función es copiar la clave pública RSA del servidor DNS a un archivo llamado *claveDNS*, para que se pueda enviar este archivo a los usuarios “oportunistas”. De esta forma, todos los usuarios reciben la clave de autenticación del servidor DNS, lo que les permitirá construir el túnel IPsec con él.

En cuanto al archivo *configConexionOportunista*, podrá ejecutarse en los dos usuarios “oportunistas”, una vez que hayan recibido el fichero *claveDNS* del servidor DNS (lo deberán almacenar en el directorio */etc/*). Los usuarios tendrán que haber generado, también, sus claves RSA. Este ejecutable se encarga de configurar el archivo *ipsec.conf* para que funcione nuestro escenario, haciendo, para ello, algunas preguntas al usuario a través de la consola (por ejemplo, nos preguntará por la interfaz de red que deberá usar FreeS/WAN, o la dirección de la pasarela de salida para dicha interfaz).

La versión actual de este *script* está construida de forma que el contenido anterior del archivo *ipsec.conf* es eliminado, por lo que recomendamos que se haga una copia de seguridad de este archivo antes de ejecutarlo.

Puede consultarse el código de estos archivos en el Apéndice C: *Scripts*.

6.3. Puesta en marcha del túnel IPsec

Una vez terminado todo el trabajo de configuración, es hora de arrancar la aplicación y comprobar que el túnel IPsec se establece correctamente. Este es el comando que arranca FreeS/WAN (habrá que ejecutarlo en los dos equipos):

```
ipsec setup start
```

Para este escenario, tendremos que empezar por arrancar las funciones de servicio de nombres en el servidor DNS (con el comando *ndc start*), así como arrancar FreeS/WAN en éste equipo, para que pueda aceptar las conexiones IPsec procedentes de los clientes. Una vez arrancados ambos procesos en el servidor DNS, los usuarios “oportunistas” podrán arrancar FreeS/WAN en sus equipos.

Se puede verificar que no ha habido ningún error mediante el comando **ipsec barf**, que nos devuelve información detallada acerca de las acciones de Pluto. El Apéndice B muestra los mensajes que devuelve *ipsec barf* si todo funciona correctamente.

Parte III:

CONCLUSIONES

7. Conclusiones

7. Conclusiones

Este trabajo surge ante el problema que plantea la falta de seguridad que presentan las comunicaciones a través de la infraestructura de redes públicas. En respuesta a ese problema, y debido a la gran importancia que ha adquirido el tráfico de información para las grandes organizaciones y empresas, aparecen las redes privadas virtuales. Elegimos la arquitectura de seguridad IPSec, para estudiar una estructura concreta capaz de construir redes privadas virtuales. Entre las distintas implementaciones de esta arquitectura, optamos por FreeS/WAN, por ser una herramienta completa y flexible.

A lo largo de este documento, hemos detallado tanto los conceptos teóricos como distintos casos prácticos de las conexiones IPSec, basadas en FreeS/WAN. En la primera parte, comenzamos por aclarar el concepto de Red Privada Virtual. Seguidamente, realizamos un estudio exhaustivo de los distintos protocolos que conforman la arquitectura IPSec. Y, por último, describimos todos los elementos que ofrece la aplicación FreeS/WAN para configurar conexiones IPSec. La segunda parte muestra un modelo de configuración, detallado paso por paso, para los tres tipos de conexión más representativos de FreeS/WAN: la conexión directa, la conexión *Road Warrior*, y la conexión “Oportunista”.

Como hemos podido comprobar, la conexión directa ofrece un enlace seguro para usuarios, de direcciones fijas y conocidos entre sí, ya sean usuarios finales o pasarelas de seguridad, que controlan el acceso a una red local. Podemos así conectar, de forma segura, dos usuarios, o redes locales, alejados entre sí, creando una red privada virtual que atraviesa la infraestructura de redes públicas.

La conexión *Road Warrior* amplía las posibilidades respecto a la conexión directa, puesto que permite tener un extremo móvil en la comunicación, siempre que se dirija a otro extremo fijo, que reconozca su clave de autenticación. Tenemos, por tanto, la posibilidad de conectar de manera segura con nuestro centro de recursos (una red corporativa o el equipo de nuestra casa) desde cualquier lugar con acceso a Internet, con la única condición de estar autorizados por dicho centro.

La conexión “Oportunista” es sin duda la estrella de FreeS/WAN, pues su objetivo original era convertirse en un estándar para Internet, utilizando los servidores DNS como bases de datos para las claves de autenticación de los usuarios IPSec. Sin embargo, la aparición de los certificados digitales ha permitido implementar un método más sencillo de autenticar las conexiones IPSec entre usuarios desconocidos entre sí, sin necesidad de servicios DNS. De hecho, los proyectos que continúan

actualmente el trabajo que comenzó con FreeS/WAN (que ha dejado de desarrollarse), incluyen el uso de certificados X.509 para establecer sus conexiones seguras.

La arquitectura IPSec constituye sin duda alguna una opción flexible y robusta de asegurar las comunicaciones extremo a extremo sobre IP, a través de la infraestructura de redes públicas. Esperamos que este documento permita comprender cómo funciona, y cómo se configura, sin demasiada dificultad, una conexión IPSec con FreeS/WAN.

Parte IV:

ANEXOS

Apéndice A: Configuración de los *Routers* CISCO

Apéndice B: Capturas de los mensajes del comando
ipsec barf

Apéndice C: *Scripts*

Referencias

Bibliografía

Glosario

Apéndice A: Configuración de los *Routers* CISCO

En este apéndice, se muestra la configuración de los tres *routers* CISCO 1751, empleados en los distintos escenarios de evaluación de FreeS/WAN. En lo que respecta al *Router 6*, la configuración que presentamos incluye los parámetros para el servicio DHCP, que sólo se emplean en el escenario 2 (correspondiente al capítulo 5).

1. Configuración del Router 3

```
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Cisc01
!
logging queue-limit 100
!
memory-size iomem 20
ip subnet-zero
!
!
!
interface FastEthernet0/0
 ip address 192.168.1.254 255.255.255.0
 no ip route-cache
 speed auto
 full-duplex
!
interface BRI0/0
 no ip address
 shutdown
 isdn switch-type basic-5ess
!
interface Serial1/0
 ip address 192.168.3.1 255.255.255.0
 clockrate 56000
!
interface Serial1/1
 no ip address
 shutdown
!
ip classless
ip route 192.168.1.0 255.255.255.0 192.168.1.1
ip route 192.168.2.0 255.255.255.0 192.168.3.2
p route 192.168.5.0 255.255.255.0 192.168.3.2
ip route 192.168.6.0 255.255.255.0 192.168.3.2
no ip http server
!
```

```
!  
!  
line con 0  
line aux 0  
line vty 0 4  
  login  
!  
!  
!  
no scheduler allocate  
end
```

2. Configuración del Router 5

```
version 12.2  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname cisco2  
!  
logging queue-limit 100  
!  
memory-size iomem 20  
ip subnet-zero  
!  
!  
!  
interface FastEthernet0/0  
  ip address 192.168.6.254 255.255.255.0  
  speed 100  
  full-duplex  
!  
interface BRI0/0  
  no ip address  
  shutdown  
!  
interface Serial1/0  
  ip address 192.168.3.2 255.255.255.0  
!  
interface Serial1/1  
  ip address 192.168.5.2 255.255.255.0  
!  
ip classless  
ip route 192.168.1.0 255.255.255.0 192.168.3.1  
ip route 192.168.2.0 255.255.255.0 192.168.5.1  
ip route 192.168.4.0 255.255.255.0 192.168.4.3  
no ip http server  
!  
!  
!  
line con 0  
line aux 0
```



```
line vty 0 4
 login
 !
no scheduler allocate
end
```

3. Configuración del Router 6 con DHCP activo

```
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname cisco3
!
logging queue-limit 100
!
memory-size iomem 20
ip subnet-zero
!
!
ip dhcp excluded-address 192.168.2.254
!
ip dhcp pool road
 network 192.168.2.0 255.255.255.0
 default-router 192.168.2.254
!
!
interface FastEthernet0/0
 ip address 192.168.2.254 255.255.255.0
 speed auto
!
interface BRI0/0
 no ip address
 shutdown
!
interface Serial1/0
 ip address 192.168.5.1 255.255.255.0
 clockrate 56000
!
interface Serial1/1
 no ip address
 shutdown
!
ip classless
ip route 192.168.1.0 255.255.255.0 192.168.5.2
ip route 192.168.3.0 255.255.255.0 192.168.5.2
ip route 192.168.6.0 255.255.255.0 192.168.5.2
!
!
!
line con 0
line aux 0
```

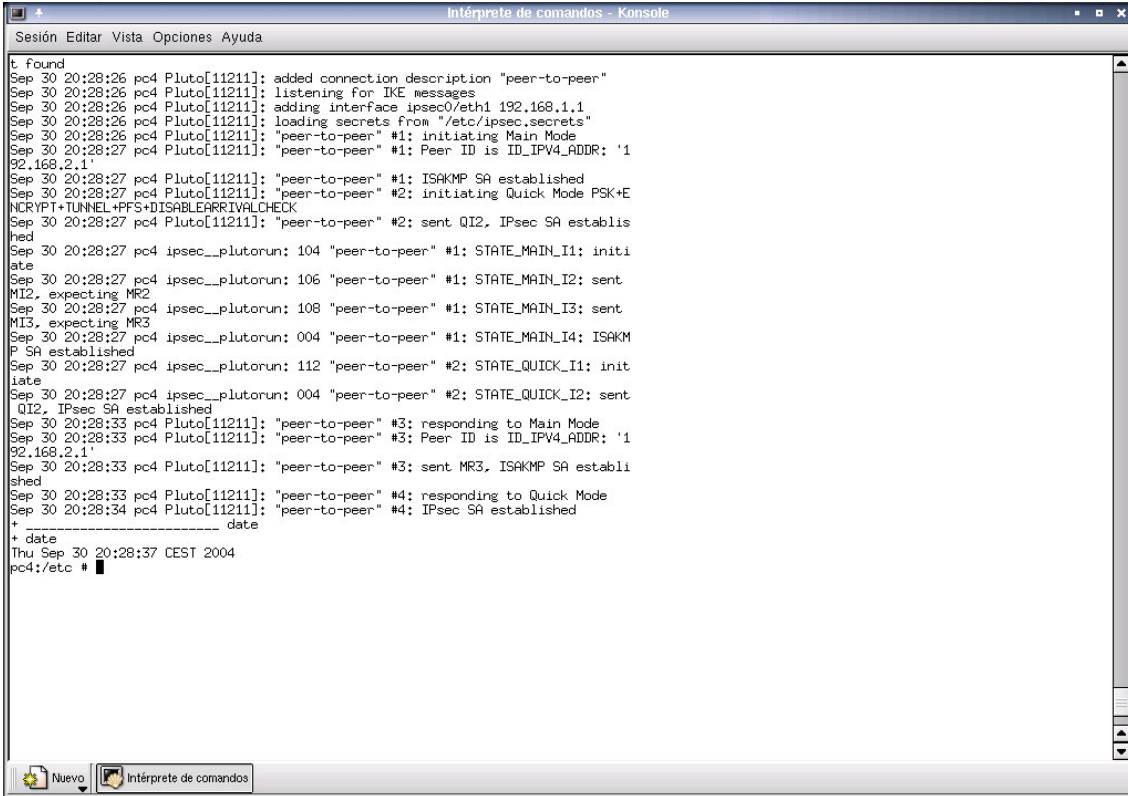
```
line vty 0 4
 login
 !
end
```

Apéndice B: Capturas de los mensajes del comando *ipsec barf*

El comando *ipsec barf* muestra por pantalla la información depurada sobre los sistemas de cifrado y autenticación que emplea IPSec. En definitiva, se trata de un comando que devuelve toda la información relevante para diagnosticar problemas en el funcionamiento de IPSec.

Las capturas que aparecen en este apéndice corresponden a los mensajes que devuelve este comando si las conexiones IPSec funcionan correctamente. La primera captura muestra el resultado si se empleó autenticación por claves compartidas o firmas digitales RSA, y la segunda es similar, pero para el caso de autenticación por certificados X.509.

1. Autenticación por claves compartidas o firmas RSA



```

t Found
Sep 30 20:28:26 pc4 Pluto[11211]: added connection description "peer-to-peer"
Sep 30 20:28:26 pc4 Pluto[11211]: listening for IKE messages
Sep 30 20:28:26 pc4 Pluto[11211]: adding interface ipsec0/eth1 192.168.1.1
Sep 30 20:28:26 pc4 Pluto[11211]: loading secrets from "/etc/ipsec.secrets"
Sep 30 20:28:26 pc4 Pluto[11211]: "peer-to-peer" #1: initiating Main Mode
Sep 30 20:28:27 pc4 Pluto[11211]: "peer-to-peer" #1: Peer ID is ID_IPV4_ADDR: '1
192.168.2.1'
Sep 30 20:28:27 pc4 Pluto[11211]: "peer-to-peer" #1: ISAKMP SA established
Sep 30 20:28:27 pc4 Pluto[11211]: "peer-to-peer" #2: initiating Quick Mode PSK+E
NCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK
Sep 30 20:28:27 pc4 Pluto[11211]: "peer-to-peer" #2: sent QI2, IPsec SA establis
hed
Sep 30 20:28:27 pc4 ipsec__plutorun: 104 "peer-to-peer" #1: STATE_MAIN_I1: initi
ate
Sep 30 20:28:27 pc4 ipsec__plutorun: 106 "peer-to-peer" #1: STATE_MAIN_I2: sent
MI2, expecting MR2
Sep 30 20:28:27 pc4 ipsec__plutorun: 108 "peer-to-peer" #1: STATE_MAIN_I3: sent
MI3, expecting MR3
Sep 30 20:28:27 pc4 ipsec__plutorun: 004 "peer-to-peer" #1: STATE_MAIN_I4: ISAKM
P SA established
Sep 30 20:28:27 pc4 ipsec__plutorun: 112 "peer-to-peer" #2: STATE_QUICK_I1: init
iate
Sep 30 20:28:27 pc4 ipsec__plutorun: 004 "peer-to-peer" #2: STATE_QUICK_I2: sent
QI2, IPsec SA established
Sep 30 20:28:33 pc4 Pluto[11211]: "peer-to-peer" #3: responding to Main Mode
Sep 30 20:28:33 pc4 Pluto[11211]: "peer-to-peer" #3: Peer ID is ID_IPV4_ADDR: '1
192.168.2.1'
Sep 30 20:28:33 pc4 Pluto[11211]: "peer-to-peer" #3: sent MR3, ISAKMP SA establi
shed
Sep 30 20:28:33 pc4 Pluto[11211]: "peer-to-peer" #4: responding to Quick Mode
Sep 30 20:28:34 pc4 Pluto[11211]: "peer-to-peer" #4: IPsec SA established
+-----+
+ date
Thu Sep 30 20:28:37 CEST 2004
pc4:/etc #

```

Figura B.1 – Mensaje devuelto por *ipsec barf* usando autenticación por claves compartidas o firmas RSA

2. Autenticación por certificados X.509

```

Intérprete de comandos - Konsola
Sesión Editar Vista Opciones Ayuda
Sep 30 20:34:16 pc4 ipsec__plutorun: Starting Pluto subsystem...
Sep 30 20:34:16 pc4 Pluto[11733]: Starting Pluto (FreeS/WAN Version 1.95)
Sep 30 20:34:16 pc4 Pluto[11733]: including X.509 patch (Version 0.9.8)
Sep 30 20:34:16 pc4 Pluto[11733]: Changing to directory '/etc/ipsec.d/cacerts'
Sep 30 20:34:16 pc4 Pluto[11733]: loaded cacert file 'cacert.pem' (1399 bytes)
Sep 30 20:34:16 pc4 Pluto[11733]: Changing to directory '/etc/ipsec.d/crls'
Sep 30 20:34:16 pc4 Pluto[11733]: Warning: empty directory
Sep 30 20:34:16 pc4 Pluto[11733]: could not open my X.509 cert file '/etc/x509cert.der'
Sep 30 20:34:16 pc4 Pluto[11733]: OpenPGP certificate file '/etc/pgpcert.pgp' not found
Sep 30 20:34:17 pc4 Pluto[11733]: loaded host cert file '/etc/ipsec.d/certs/pc4cert.pem' (1257 bytes)
Sep 30 20:34:17 pc4 Pluto[11733]: loaded host cert file '/etc/ipsec.d/pc2cert.pem' (1257 bytes)
Sep 30 20:34:17 pc4 Pluto[11733]: added connection description "roadWarriorX509"
Sep 30 20:34:17 pc4 Pluto[11733]: listening for IKE messages
Sep 30 20:34:17 pc4 Pluto[11733]: adding interface ipsec0/eth1 192.168.1.1
Sep 30 20:34:17 pc4 Pluto[11733]: loading secrets from "/etc/ipsec.secrets"
Sep 30 20:34:17 pc4 Pluto[11733]: loaded private key file '/etc/ipsec.d/private/pc4key.pem' (963 bytes)
Sep 30 20:34:17 pc4 Pluto[11733]: "roadWarriorX509" #1: initiating Main Mode
Sep 30 20:34:17 pc4 Pluto[11733]: "roadWarriorX509" #1: Peer ID is ID_DER_ASN1_DN: 'C=sp, ST=murcia, O=it4, OU=che, CN=tes'
Sep 30 20:34:17 pc4 Pluto[11733]: "roadWarriorX509" #1: ISAKMP SA established
Sep 30 20:34:17 pc4 Pluto[11733]: "roadWarriorX509" #2: initiating Quick Mode RSASIG+ENCRYPT+TUNNEL+PFS+DISABLEARRIVALCHECK
Sep 30 20:34:18 pc4 Pluto[11733]: "roadWarriorX509" #2: sent QI2, IPsec SA established
Sep 30 20:34:18 pc4 ipsec__plutorun: 104 "roadWarriorX509" #1: STATE_MAIN_I1: initiate
Sep 30 20:34:18 pc4 ipsec__plutorun: 106 "roadWarriorX509" #1: STATE_MAIN_I2: sent MI2, expecting MR2
Sep 30 20:34:18 pc4 ipsec__plutorun: 108 "roadWarriorX509" #1: STATE_MAIN_I3: sent MI3, expecting MR3
Sep 30 20:34:18 pc4 ipsec__plutorun: 004 "roadWarriorX509" #1: STATE_MAIN_I4: ISAKMP SA established
Sep 30 20:34:18 pc4 ipsec__plutorun: 112 "roadWarriorX509" #2: STATE_QUICK_I1: initiate
Sep 30 20:34:18 pc4 ipsec__plutorun: 004 "roadWarriorX509" #2: STATE_QUICK_I2: sent QI2, IPsec SA established
Sep 30 20:34:23 pc4 Pluto[11733]: "roadWarriorX509" #3: responding to Main Mode
Sep 30 20:34:23 pc4 Pluto[11733]: "roadWarriorX509" #3: Peer ID is ID_DER_ASN1_DN: 'C=sp, ST=murcia, O=it4, OU=che, CN=tes'
Sep 30 20:34:23 pc4 Pluto[11733]: "roadWarriorX509" #3: sent MR3, ISAKMP SA established
Sep 30 20:34:23 pc4 Pluto[11733]: "roadWarriorX509" #4: responding to Quick Mode
Sep 30 20:34:23 pc4 Pluto[11733]: "roadWarriorX509" #4: IPsec SA established
+-----+
+ date
Thu Sep 30 20:34:24 CEST 2004
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
pc4:/etc #
+-----+
Nuevo Intérprete de comandos

```

Figura B.2 – Mensaje devuelto por *ipsec barf*, usando autenticación por certificados X.509

Apéndice C: *Scripts*

A continuación, mostramos el código de los archivos ejecutables implementados para facilitar la configuración de la conexión “Oportunista”.

1. Código del archivo “extraeClave”

```
# "extraeClave" es un archivo ejecutable que deberá ejecutarse en el
# servidor DNS, una vez se
# haya generado su par de claves RSA, y se hayan almacenado en el
# archivo ipsec.secrets.
# Este script extrae la clave pública del servidor DNS, y la guarda en
# un archivo de texto
# llamado "claveDNS", en el formato que necesitarán los clientes
# "oportunistas" para configurar
# su conexión con el DNS desde su archivo ipsec.conf (en nuestro
# escenario lo situamos como
# participante de la derecha)
# Una vez ejecutado, deberá enviarse una copia de "claveDNS" a cada
# cliente "oportunista".
```

```
ipsec showhostkey --right > claveDNS
```

2. Código del archivo “configConexionOportunista”

```
# "configConexionOportunista" es un script que deberá ejecutarse en los
# clientes "oportunistas"
# de FreeS/WAN, una vez que hayan recibido el archivo "claveDNS" desde
# el servidor DNS (tendrán que guardarlo en el directorio /etc).
# Este ejecutable automatiza el proceso de configuración del archivo
# ipsec.conf. El programa
# realiza una serie de preguntas al usuario sobre interfaces de red y
# direcciones IP a emplear,
# y por medio de las respuestas irá completando los campos necesarios en
# el archivo ipsec.conf,
# para configurar la conexión "Oportunista".
```

```
cd /etc
echo "Creando nuevo archivo ipsec.conf.."
echo "# basic configuration
config setup
    klipsdebug=none
    plutodebug=none
```

```
plutoload=%search
plutostart=%search

uniqueids=yes" > ipsec.conf

echo "Su ordenador tiene más de una interfaz de red? (si/no)"
read resp

if test "$resp" = "si"
then
echo "Escriba el nombre de la interfaz de red que usará IPsec"
read int
echo -e "\tinterfaces='''ipsec0=\"$int'''" >> ipsec.conf

echo -e "\n\nconn clientToDNS" >> ipsec.conf

echo "Indique la dirección IP de dicha interfaz"
read dir
echo -e "\tleft=\"$dir" >> ipsec.conf

echo "Indique la dirección IP de su gateway de salida"
read gw
echo -e "\tleftnexthop=\"$gw" >> ipsec.conf

elif test "$resp" = "no"
then
echo -e "\tinterfaces=%defaultroute" >> ipsec.conf
echo -e "\n\nconn clientToDNS" >> ipsec.conf
echo -e "\tleft=%defaultroute" >> ipsec.conf

else
echo "Por favor escribeme si o no"
exit 1
fi

set $(ipsec showhostkey --left)
echo -e "\t$" >> ipsec.conf

echo -e "\tright=192.168.6.1
rightnexthop=192.168.6.254" >> ipsec.conf

set $(more claveDNS)

echo -e "\t{*}
authby=rsasig
auto=start" >> ipsec.conf

echo -e "\n\nconn opportunistic" >> ipsec.conf

if test "$resp" = "si"
then
echo -e "\tleft=\"$dir" >> ipsec.conf
echo -e "\tleftnexthop=\"$gw" >> ipsec.conf
else
echo -e "\tleft=%defaultroute" >> ipsec.conf
```

```
fi
```

```
echo -e "\tlefttrsasigkey=%dns  
authby=rsasig  
right=%opportunistic  
rightrsasigkey=%dns  
auto=route" >> ipsec.conf
```

```
echo "Configuración finalizada. Pruebe arrancar FreeS/WAN con el  
comando: ipsec setup start"
```


Referencias

- [1] Para más información acerca de estos proyectos, consultar sus páginas oficiales:
www.strongsec.com / www.strongswan.org
- [2] Para más información sobre los componentes de FreeS/WAN, consultar los manuales disponibles en la dirección:
http://www.freeswan.org/freeswan_trees/freeswan-1.95/doc/manpages.html
- [3] Los RPM de varias versiones de FreeS/WAN están disponibles en:
<http://rpm.pbone.net> (esta página contiene un buscador de RPM)
- [4] Es posible descargar el parche para certificados X.509 desde la dirección:
<http://www.strongsec.com/freeswan/index.htm>.
- [5] Puede consultarse la configuración de los routers *CISCO* empleada en los escenarios en el Apéndice A: “Configuración de los *Routers* CISCO”.
- [6] Esta información proviene del documento “*Túneles Cifrados con FreeS/WAN*”, escrito por Daniel Esteban Coletti. Puede encontrarse en:
<http://www.xtech.com.ar/articulos/tutorial-freeswan/>
- [7] Para saber más sobre servidores DNS implementados en Linux, consultar el libro “*DNS and Bind*” - Paul Albitz, Cricket Liu - Ed. O’Reilly, 4^o edición (2001). ISBN: 0-596-00158-4.
- [8] Podemos incluir nuestras claves de autenticación de *host* en un servidor DNS construido, entre otros motivos, para permitir su uso a los usuarios de FreeS/WAN, y al que se puede acceder a desde la dirección:
<http://www.fdns.net>

Bibliografía

- “*DNS and Bind*” - Paul Albitz, Cricket Liu - Ed. O’Reilly, 4ª edición (2001). ISBN: 0-596-00158-4.
- “*A Practical Guide to Linux*”(1997) – Mark G. Sobell- Ed. Addison Wesley, 1ª edición. ISBN: 0-201-89549-8
- “*CCSP CISCO Secure VPN – Exam Certification Guide*”(2003) – John F. Roland, Mark J. Newcomb – Ed. Cisco Press, 1ª edición. ISBN:1-58720-070-8.

Documentación Complementaria

- “*Túneles Cifrados con FreeS/WAN*” - Daniel Esteban Coletti.
- “*Estudio sobre las VPN*”- Mª Nieves Gutiérrez González, Ana Rosa Sancho Buzón, Amadeo Casas Cuadrado – Universidad de Valladolid.
- “*IPSEC*” – Víctor Villagra – Universidad Politécnica de Madrid.
- “*Virtual Private Networks, Copying with Complexity*” - Andreas Steffen - *Zürcher Hochschule Winterthur*.
- “*Implementación de redes privadas virtuales utilizando el protocolo IPsec*” - Vicente José Aguilar Roselló, Ricardo Domínguez Jover, Ignacio Sánchez Medina - Curso de Especialista Universitario en Seguridad Informática y Servicios Electrónicos.
- “*IPSec and IKE administration Guide*” – Sun Microsystems, Inc.
- “*Seguridad en las comunicaciones en un mundo todo IP*” – Luis Barriga, Rolf Blom, Christian Gehrman, Mats Näslund – Ericsson Review nº2, 2000.

- RFC 2402: *"IP Authentication Header"*.
- RFC 2406: *"IP Encapsulating Security Payload"*.
- RFC 2408: *"Internet Security Association and Key Management Protocol"*.
- RFC 2409: *"The Internet Key Exchange"*.

Glosario

AH: protocolo de Cabecera de Autenticación - *Authentication Header*.

CA: Autoridad de Certificación - *Certification Authority*.

DHCP: Protocolo de Configuración Dinámica de *Host* - *Dynamic Host Configuration Protocol*.

DNS: Sistema de Nombres de Dominio - *Domain Name System*.

ESP: protocolo de Carga de Encapsulado de Seguridad - *Encapsulating Security Payload*.

FreeS/WAN: Red de Area Amplia Libre y Segura - *Free Secure Wide Area Network*.

IKE: protocolo de Intercambio de Claves en Internet - *Internet Key Exchange*.

IPSec: Seguridad para el Protocolo de Internet - *Internet Protocol Security*.

ISAKMP: Protocolo de Asociación de Seguridad y Gestión de Claves - *Internet Security Association and Key Management Protocol*.

KLIPS: Soporte al Núcleo para IPSec - *KerneL IPSec Support*.

OpenSSL: protocolo Abierto de la Capa de Socket Seguro - *Open Secure Socket Layer*.

PSK: Claves Compartidas - *PreShared Keys*.

RSA: Algoritmo de seguridad que debe su nombre a sus tres creadores: *Rivest Shamir Adleman*.

SA: Asociación de Seguridad - *Security Association*.

SKEME: Mecanismo de Seguridad para el Intercambio de Claves en Internet - *Secure Key Exchange Mechanism for Internet*.

VPN: Red Privada Virtual - *Virtual Private Network*.

