ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

SEGURIDADES PARA SERVIDORES MAIL Y WEB BASADOS EN EL PROTOCOLO SSL SOBRE EL SISTEMA OPERATIVO LINUX

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

EDUARDO FRANCISCO ANGUETA RAMÍREZ

DIRECTOR: ING. PABLO HIDALGO LASCANO

QUITO, OCTUBRE 2004

DECLARACIÓN

Yo, Eduardo Francisco Angueta Ramírez, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley, Reglamento Intelectual y por la normatividad institucional vigente.

EDUARDO FRANCISCO ANGUETA RAMÍREZ

CERTIFICACIÓN

Certifico que el presente trabajo ha sido elaborado en su totalidad por el Sr. Eduardo Francisco Angueta Ramírez, bajo mi supervisión.

Ing. Pablo Hidalgo L.
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

A la Escuela Politécnica Nacional, por haberme acogido en sus aulas y formarme como un profesional de servicio a la patria.

Al Sr. Ing. Pablo Hidalgo y a todos mis maestros de la carrera, por haberme impartido sus valiosos conocimientos gracias a los cuales he podido desarrollarme satisfactoriamente en mi vida profesional.

DEDICATORIA

A Dios, por recibir de sus manos la fuerza y sabiduría necesarias para vencer los obstáculos que día a día se ha presentado durante esta importante fase de mi vida.

A mi madre Marina, por haberme brindado su apoyo y amor todos los días de mi vida, pues sin su valerosa ayuda no hubiese podido llegar tan lejos.

A mi padre Juan Francisco, por su rectitud y amor; especialmente por darme el mejor regalo que un padre puede dar a sus hijos, la educación.

A mis hermanos Juan Carlos, Cristhian Hernán y Andrea Estefanía. Por haberme escuchado y brindado sus consejos en los momentos que más los he necesitado.

A mi esposa Andrea, por su amor, compresión y apoyo, por nuestra hija María Emilia, que con sus sonrisas llena de alegría nuestras vidas.

A todos mis amigos de la universidad, por compartir conmigo infinitos momentos tristes y alegres dentro y fuera de las aulas y haberme enseñado que el estudio no es lo único importante.

ÍNDICE

CAPÍTULO UNO

CAPÍTULO CUATRO

ÍNDICE DE FIGURAS

ANEXOS

RESUMEN

En el primer capítulo, se aborda de manera concreta el problema de la seguridad del tránsito de los datos en medios hostiles tales como el Internet, por ello se exponen herramientas tecnológicas desarrolladas tales como la encripción simétrica y asimétrica, técnicas de *hashing* para verificación de la integridad de los datos, y la autenticación para proveer control de acceso a los recursos. Posteriormente en el capítulo se introducen los conceptos de autoridades de certificación y certificados digitales, lo cual ayuda a proveer servicios de autenticación.

En este capítulo segundo se estudia el protocolo SSL *(Secure Sockets Layer)*; se analizan las fortalezas del protocolo desde el punto de vista de algoritmos de encripción, algoritmos de *hashing* y algoritmos de intercambio de llaves soportados. Se describen las etapas y agentes involucrados en la transmisión de los datos, así como los mensajes intercambiados entre el cliente y servidor que intervienen en el establecimiento de las sesiones seguras. Se muestran de manera concreta cada uno de los protocolos que conforman el protocolo SSL, y finalmente se hace una análisis a la seguridad del antes mencionado protocolo.

En el capítulo tercero, se muestran los pasos que deben seguirse para la configuración de un servidor *web* seguro que cuente con la autenticación tanto del servidor como del cliente. También se describe el proceso de instalación y configuración del servidor de correo electrónico con sus respectivas pruebas de funcionamiento. Para ello se inicia con una introducción del funcionamiento y configuración del servidor *web* Apache y otros protocolos relacionados tales como DNS, POP3 e IMAP, los cuales guardan también relación con la implementación del servidor de correo electrónico seguro.

Luego de la implementación de los servidores *web* y *mail*, en el capítulo cuarto se hace un análisis del establecimiento de las sesiones. Dicho análisis se basa en el uso de un analizador de protocolos para mostrar los paquetes intercambiados entre cliente y servidor cuando se envían los mensajes del establecimiento de las sesiones, sean éstas seguras o no seguras.

En el capítulo quinto se muestran conclusiones relacionadas con los servicios de seguridad los datos así como con aspectos propios de la implementación de los servidores *web* y *mail*. Además en el capítulo se hacen recomendaciones para posibles estudios posteriores, los cuales pueden tomar como base el estudio actual.

Finalmente, en los anexos se muestra de manera detallada todos y cada uno de los pasos que se realizan durante las diferentes fases del proyecto de titulación. Se incluyen ilustraciones gráficas y textuales de los pasos que se deben seguir a manera de guía para llevar a cabo las configuraciones.

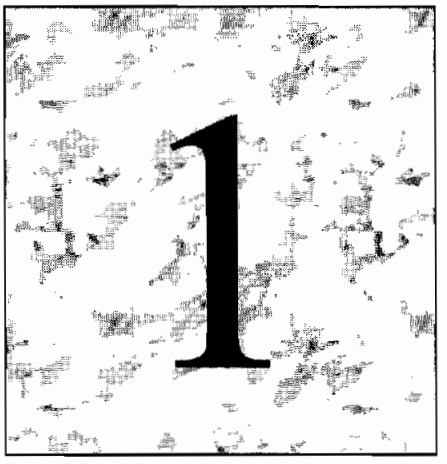
PRESENTACIÓN

El objetivo principal del presente proyecto de titulación es constituirse en una guía para la implementación de un servidor *web* y un servidor de correo electrónico seguros. Ambos basados en el uso de un sistema operativo de distribución gratuita tal como lo es *Red Hat Linux* juntamente con el protocolo SSL *(Secure Sockets Layer)* para proveer servicios de seguridad.

Se pretende mostrar una alternativa económica válida en la implementación de una infraestructura básica de una Intranet, la misma que posea los servicios de un servidor de correo electrónico y de un servidor *web* seguros. Las alternativas económicas y de optimización de recursos en nuestro país son implementadas día a día; por lo tanto se augura que el presente proyecto tenga repercusión a nivel de empresas que desean implementar una infraestructura segura sin tener que invertir altas cifras de dinero.

Este trabajo está enfocado principalmente a personas que desean experimentar y contrastar las alternativas que hoy en día se ofrecen a nivel de implementaciones gratuitas y económicamente alcanzables, frente a alternativas no gratuitas y de altos costos de implementación.

CAPÍTULO



GENERALIDADES DE LA SEGURIDAD DE LA INFORMACIÓN

CAPÍTULO UNO

GENERALIDADES DE LA SEGURIDAD DE LA INFORMACIÓN

1.1 EL PROBLEMA DE LA SEGURIDAD DE LA INFORMACIÓN

1.1.1 INTRODUCCIÓN

Desde el inicio de los tiempos el hombre ha necesitado comunicarse con los demás de su especie. Para ello ha inventado mecanismos cada vez más innovadores con el único objetivo de suplir una de sus necesidades más básicas, la comunicación.

Luego que este objetivo se ha cumplido exitosamente se busca la manera de mejorar el proceso, y es el punto donde se toman en cuenta e integran factores que facilitan y otros que dificultan la consecución del mismo.

Cuando una persona se encuentra en alguna ciudad y desea enviar correspondencia a otra persona que está en otra ciudad, dicha correspondencia debe tener información necesaria tal como los datos del remitente y del destinatario. El remitente por su parte escogerá una compañía en la que confía para poder enviar su correspondencia, pues sabe que de esta manera el paquete llegará a su destino.

Luego nacen otras necesidades por parte del destinatario, principalmente en torno a la correspondencia, tales como: asegurar que llegue en perfecto estado (pues si envía un artefacto electrónico no serviría de nada que llegara destruido), asegurar que la persona quien recibe su correspondencia es el verdadero destinatario y no una persona quien ha decidido suplantar la identidad del destinatario (lo cual implicaría un robo), y por si fuera poco, el

remitente desea un documento que indique que el destinatario ha recibido la correspondencia; este documento sirve al remitente para que posteriormente el destinatario no pueda alegar no haber recibido la correspondencia.

Un caso parecido es el de otra persona que desea adquirir un bien de cualquier naturaleza; entonces acude a un lugar donde considera que sus compras serán respaldadas por el proveedor. Pues solamente de esta manera el cliente siente la seguridad de haber hecho una buena compra; pues si el bien adquirido tiene algún desperfecto tendrá la certeza que el proveedor asumirá la responsabilidad del mismo, y que su problema se solucionará según los parámetros de compra acordados inicialmente.

Los niveles de confianza que se lleguen a tener entre los participantes de una transacción dependen en mayor parte del grado de seriedad con lo cual se lleven a cabo los procesos involucrados para llegar a la consecución del objetivo. Es entonces donde entra en juego la seguridad de los procesos involucrados.

Los siguientes párrafos enfocan de manera concreta los procesos más importantes involucrados en el viaje de la información en general, dando de esta manera al lector una visión objetiva sobre los problemas que pueden originarse mientras la información viaja desde su origen hasta su destino.

1.1.2 PRIVACIDAD DE LOS DATOS

Todas las organizaciones poseen activos de toda índole a los cuales desean proteger por razones diferentes. La privacidad del acceso a dichos activos permite asegurar que éstos serán accedidos solamente por personal autorizado y que éstos no se encuentren a disposición del personal no autorizado.

Concretamente, cuando se habla del transporte de la información este aspecto es clave. La privacidad de la comunicación permite a los participantes asegurar que las personas que ellos saben están participando son efectivamente los únicos involucrados en la transferencia de la información.

Si se considera un posible ataque a los datos o al software que maneja una empresa, indudablemente este último es mucho más leve que el ataque a los datos, pues se puede restaurar el sistema operativo o las aplicaciones con los medios originales de instalación; mientras que, los datos del usuario a no ser que se tenga una política de seguridad muy estricta pueden verse definitivamente comprometidos.

Pero no solamente la pérdida parcial o total de los datos debe ser considerada como un ataque a la privacidad de los mismos. Existen algunas otras amenazas (esquematizadas en la figura 1.1) que deben ser tomadas en cuenta:

- *Interrupción*

  Se entiende por interrupción al hecho de impedir que los datos lleguen a su destino final, habiendo sido de esta manera victimas de un ataque.

- *Intercepción*

  Cuando la intercepción ha sido llevada a cabo se rompe todo concepto de privacidad, ya que el contenido de los datos ha sido accedido por personas no autorizadas.

1.1.3 AUTENTICACIÓN DE LAS PARTES

La autenticación es la parte central de la seguridad, ya que basándose en la identificación de las personas, tomando en cuenta en algo que ellos saben (en el caso de *passwords*), o algo que ellos tienen (en la caso de tarjetas electrónicas y otros dispositivos), o algo que ellos poseen (en el caso de sistemas de

autenticación biométricos), los sistemas pueden tomar decisiones para permitir el acceso a los recursos y ejecutar acciones tan variadas como abrir una puerta o permitir el descargo de información crucial para los negocios de la empresa.

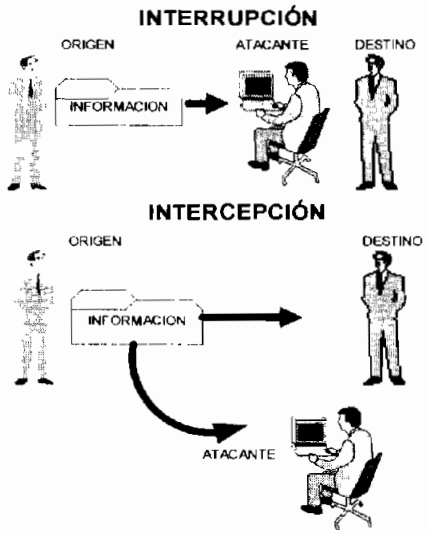


FIGURA 1.1 Amenazas a la Privacidad de los Datos [1]

De esta manera la Autenticación se convierte en el proceso mediante el cual una administración ya sea modular o centralizada, verifica que los usuarios que tratan de acceder a un determinado recurso sean quienes dicen ser.

Hay ocasiones que la Autenticación se torna bidireccional; es el caso cuando tanto el emisor como el receptor deben estar seguros de la identidad de su par. Este caso se presenta a menudo cuando una persona que busca cierta información la encuentra de un proveedor el cual no conoce ni tiene ninguna referencia, entonces dicha persona deberá estar seguro que la información que se le entregue sea totalmente confiable, para ello se Autentica también al proveedor de la información; adicionalmente el proveedor deberá estar seguro de la Identidad del investigador para poder entregarle lo que busca.

Uno de los mayores logros de la Autenticación es proveer cierto control

sobre el acceso a la información a personas no autorizadas, basando dicha seguridad en algo que ellos saben, las claves de acceso.

La Autenticación en general puede ser de dos tipos: la Autenticación Débil y la Autenticación Fuerte. La primera, se usa en la mayor parte de los sistemas, de esta manera el usuario que desea acceder a un determinado recurso revela algo que él sabe, como por ejemplo un *password*. En cambio, cuando se hace uso de la Autenticación Fuerte, el usuario no revela ningún secreto; este ambiente se presenta normalmente cuando se hace uso de encripción asimétrica, en el cual el usuario que desea acceder encripta un mensaje de una manera que solo él sabe como hacerlo y todos los demás saben como desencriptarlo, el secreto no es compartido en ningún momento.

a. AUTENTICACIÓN DE DOS PARTICIPANTES

Hay varias maneras con las cuales un cliente quien desea acceder a los recursos de un sistema puede ser autenticado por otra entidad, generalmente llamado servidor:

- *Procedimiento de Autenticación Simple No Protegido*

  Usado cuando un cliente revela algo que él sabe, por ejemplo un *password*. El inconveniente es que el cliente no sabe si está hablando con el servidor correcto o peor aún un tercero puede interceptar el *password* cuando es enviado al servidor.

- *Procedimiento de Autenticación Simple*

  En este método se usan funciones de una sola vía. Las salidas de estas funciones son únicas y no puede generarse la entrada en base al resultado generado por la función.

  El cliente envía este resultado el cual está en función del *password* del cliente, entre otros parámetros; entonces el servidor realiza el

mismo proceso haciendo uso de la clave del cliente que guarda en sus registros, con lo cual verifica que el resultado generado por el servidor, sea el mismo que fue enviado por el cliente.

- *Procedimiento de Autenticación Fuerte*

Los métodos mostrados anteriormente tienen el inconveniente que toda la seguridad reside en el servidor y en el hecho que éste no revele los *passwords* de sus clientes, por lo cual podría ser el servidor un sitio estratégico y centralizado para ejecutar un ataque.
La solución a estos problemas inherentes a los métodos anteriores es el uso de algún procedimiento de autenticación fuerte, por ejemplo encripción asimétrica. Mediante este método el usuario encripta su información con una llave privada conocida solamente por él; mientras que los demás desencriptan esta información haciendo uso de la llave pública correspondiente a este usuario y conocida por todos los demás.

b. AUTENTICACIÓN DE TRES PARTICIPANTES

Hasta ahora los mecanismos mencionados tienen solamente dos participantes, el problema se genera cuando se tiene ambientes con muchos usuarios, entonces es necesario compartir los *passwords* o las llaves públicas de quienes se desean autenticar, lo cual los hace muy poco prácticos.

La alternativa a este problema es tener una entidad centralizada en la cual los clientes de una sesión confían; mediante este método los clientes no necesitan autenticar a sus pares ya que esta tarea es llevada cabo por la entidad centralizada antes mencionada.

Kerberos, es una implementación de este tipo, fue creada por el MIT (*Massachusetts Institute of Technology*) durante el desarrollo del proyecto

Athena. El servidor de Autenticación Kerberos genera llaves para el uso de los clientes en un sistema, el problema de esta implementación es que toda la seguridad reside en el proceso de autenticación llevado a cabo por el servidor. La llave privada del cliente es compartida con el servidor y es presentada al momento de la autenticación. El procedimiento de establecimiento de la sesión con otro usuario se describe a continuación y se indica en la figura 1.2.

El usuario 1 presenta su identificación ante el servidor Kerberos solicitando su autenticación. (Paso 1)

Si es afirmativamente autenticado el servidor remite al usuario 1 un *ticket* (Paso 2), el mismo que es presentado ante el TGS (*Ticket Granting Server*), el cual es un servidor especial de generación de *tickets*.

Cuando el usuario 1 contacta al TGS (Paso 3), éste le remite un *ticket* 2 para ser usado con el usuario 2 como llave de encripción (Paso 4).

De esta manera la comunicación puede ser lleva haciendo uso del *ticket* 2 para efectos de encripción de los datos (Paso 5).

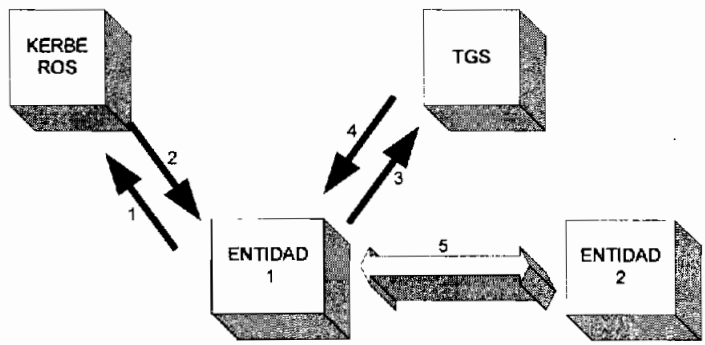


FIGURA 1.2 Proceso de autenticación en Kerberos [12]

1.1.4 INTEGRIDAD DE LOS DATOS

Se considera a la Integridad de los datos como un servicio a la seguridad de los mismos. Se puede decir que la Integridad permite verificar que los datos, los cuales llegan de un determinado emisor, no han sido modificados

en su transporte.

Existen varias formas mediante las cuales los datos pueden ser víctimas de ataques a su integridad, entre las que se pueden citar:

- *Modificación*

  Si luego de haber obtenido acceso al contenido de los datos, el atacante escribe, los cambia o borra parcialmente o totalmente, se entiende que los datos han sido víctimas de Modificación u Destrucción (figura 1.3).

- *Fabricación*

  Consiste en construir un contenido igual o parecido al contenido original de los datos de tal manera que no pueda ser detectado por el destinatario (figura 1.3).

FIGURA 1.3 Amenazas a la Integridad de los Datos [11]

1.1.5 CONSECUENCIAS DE LA FALTA DE SEGURIDAD

Los servicios enunciados anteriormente proveen pautas que pueden ser seguidas si se desea dar seguridad a los datos que viajan en medios hostiles, en

donde sus datos puede verse comprometidos. Los usuarios por su parte tienen libertad de usarlos o no, pero deben correr los riesgos de exponer información confidencial para las empresas en las que trabajan e inclusive información confidencial de índole personal.

Se han dado casos en los cuales el no uso de estas técnicas para asegurar los datos de los usuarios se han visto reflejados en robos basados en transacciones millonarias en cuentas bancarias, o peor aún casos en los cuales la información confidencial de las compañías se ha visto comprometida y por consiguiente sus negocios.

Es evidente que la falta de seguridad de los datos confidenciales o privados sea cual sea la denominación que los usuarios quieran darles, no beneficia a nadie, ni al usuario ni a la empresa en la cual presta sus servicios.

Especialmente en los últimos años, las compañías se han visto hasta cierto punto obligadas a enfatizar el uso de técnicas que les permitan asegurar sus datos, pues las pérdidas económicas en las cuales se han visto reflejadas han sido muy significativas por el hecho de no tomar medidas oportunas. Aunque si bien estas cifras tienden a bajar (a manera general), es muy importante destacar que esto se ha logrado gracias al esfuerzo de las empresas y su empeño en asegurar unos de sus más preciados activos, la red y sus datos.

Las técnicas que han tenido acogida para asegurar de una u otra manera los datos y las redes son:

1. Detección de Intrusos;
2. Seguridad Física;
3. *Login* Encriptado;
4. *Firewalls*;
5. *Passwords*;
6. *Software* Antivirus;

7. Encripción de Archivos;
8. Seguridad Biométrica; y
9. Control de Acceso, entre otras.

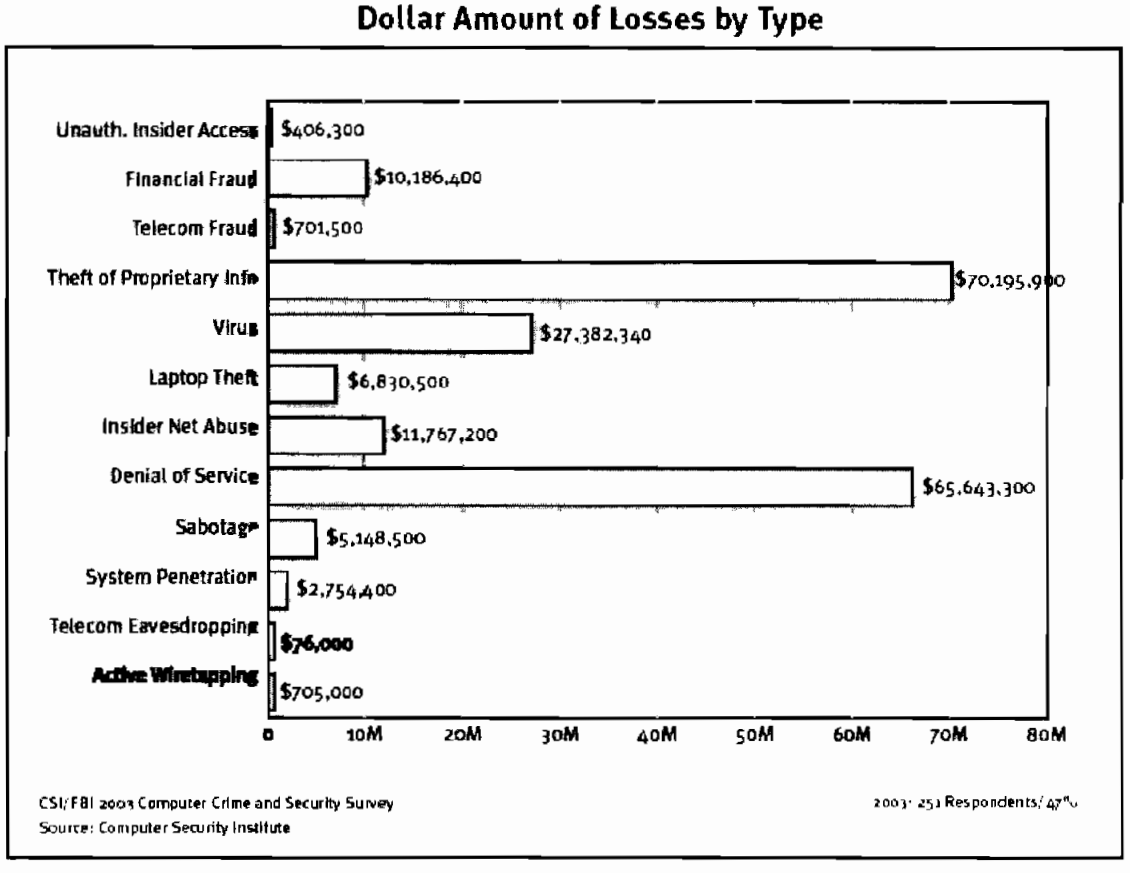


FIGURA 1.4 Cifras Importantes de los Ataques [3]

A pesar de usar una variedad de técnicas de seguridad las empresas reconocen que la mayoría de ataques provienen desde el interior de las mismas, razón por la cual se han visto obligadas no solo dar seguridad de acceso a los recursos en contra de los intrusos exteriores de la red, sino también en contra de los intrusos que se encuentran en el interior de la misma.

El robo de la información propietaria ha sido siempre la mayor cifra presentada por los afectados. Muchas maneras de robarla existen actualmente; por ejemplo, los empleados que tienen acceso a la información entregan a otros

que no la tienen, o también, violando las leyes de derechos de autor se fotocopian libros y revistas.

Hay varios grupos plenamente identificados los cuales suelen ser las fuentes de ataques en contra de las entidades; los más grandes están constituidos por los empleados o ex-empleados inconformes, los *Hackers* y la competencia. Mientras que, en un grupo minoritario se tiene a funcionarios del gobierno, los cuales básicamente juegan un papel de espías.

Por otra parte, la integración de *Hackers* reformados (según ellos) a trabajos de seguridad en redes, es un tema que se discute todos los años con gran ímpetu. Sin embargo, la mayoría de las empresas manifiestan que no es la mejor opción el integrarlos a la sociedad legal dado el peligro que constituyen sus conocimientos; pero se debe tomar en cuenta que estas personas son las que mejor conocen las fortalezas y vulnerabilidades de los sistemas, lo cual se convierte en un punto a su favor al momento de tomar una decisión tan importante como la de incorporarlas a una compañía.

A pesar que las compañías de pequeña, mediana o grande escala sufren varios tipos de ataques a sus redes o a sus datos, éstas deciden no reportar los hechos, principalmente por evadir la publicidad negativa que se generaría hacia sus clientes; o también, por el temor que dichas publicaciones sean aprovechadas por la competencia para explotarlas ya sea con publicidad negativa adicional o ataques a la infraestructura declarada como vulnerable.

Finalmente, lo números hablan por sí solos, se muestra en la tabla 1.1 los montos a los que han ascendido y la tendencia que los mismos han tomado respecto a los ataques más típicos que una organización se expone.

The Cost of Computer Crime

The following table shows the aggregate cost of computer crimes and security breaches over a 48-month period

In 2003, 75% of our survey respondents acknowledged financial losses, but only 47% could quantify the losses.

How Money Was Lost

	Lowest Reported				Highest Reported				Average Losses				Total Annual Losses			
	00	01	02	03	00	01	02	03	00	01	02	03	00	01	02	03
Theft of proprietary info.	$1K	$100	$1K	$2K	$25M	$50M	$50M	$35M	$1,032,808	$1,447,000	$6,571,000	$2,099,842	$66,706,000	$151,230,000	$170,827,000	70,395,000
Sabotage of data of networks	1K	100	1K	500	15M	3M	10M	2M	969,577	199,350	541,000	214,521	27,148,080	5,183,100	15,134,000	5,148,500
Telecom eavesdropping	200	1K	5K	1K	100K	500K	5M	50K	66,080	55,375	1,205,000	15,200	991,200	886,000	146,000	76,000
System penetration by outsider	1K	100	1K	100	5M	10M	5M	1M	244,965	463,067	226,000	56,212	7,104,080	19,066,600	13,055,008	2,754,400
Insider abuse of Net access	240	100	1K	100	15M	10M	10M	6M	317,524	357,160	536,000	176,255	27,984,740	15,001,650	50,040,000	11,767,200
Financial fraud	500	500	1K	1K	25M	40M	50M	4M	1,646,941	4,420,738	4,632,000	328,594	55,996,000	92,935,500	115,753,000	10,186,400
Denial of service	1K	100	1K	500	5M	2M	50M	60M	108,717	122,189	297,000	1,427,028	8,247,500	4,283,000	18,370,500	16,643,300
Virus	100	100	1K	40	10M	20M	9M	6M	188,092	243,835	283,000	199,871	29,171,700	45,288,350	49,979,000	27,382,340
Unauthorized insider access	1K	1K	2K	100	2M	5M	1.5M	100K	1,124,725	276,636	300,00	71,254	22,554,500	5,004,000	4,503,000	400,300
Telecom fraud	1K	500	1K	100	3M	8M	100K	250K	212,000	502,278	22,000	50,107	4,028,000	9,041,000	6,015,00	701,500
Active wiretapping	5M	0	0	5K	5M	0	0	700K	5M	0	0	352,500	5,000,000	0	0	705,000
Laptop theft	500	1K	1K	2400	1.2M	2M	5M	2M	58,794	61,881	89,000	47,107	10,404,380	8,849,000	11,766,500	6,830,500
												Total Annual Losses	265,337,090	377,826,700	456,698,000	201,707,340

CSI/FBI 2003 Computer Crime and Security Survey
Source: Computer Security Institute

TABLA 1.1 El Costo del Crimen [3]

1.2 MECANISMOS PARA ASEGURAR LOS DATOS

1.2.1 SEGURIDAD LÓGICA Y SEGURIDAD FÍSICA

Seguridad es un término el cual indica que un sistema no es vulnerable a ataques de cualquier índole. En el caso especial cuando se trata de redes de información el término seguridad carece de significado explícito, ya que no existen sistemas absolutamente seguros. Independientemente de las herramientas que se usen para asegurar sus elementos, siempre existirán brechas en la seguridad, la evaluación de dichas herramientas tienden a ser calificadas solamente con el grado de mayor o menor seguridad.

Particularizando el término *seguridad* para las redes de computadores, se tiende a nombrar a la *fiabilidad* como un término que realmente es alcanzable. La *seguridad*, a pesar de ser un objetivo claro en la administración de redes no deja de ser un problema inherente a los sistemas que las sustentan.

Por esa razón principalmente en la actualidad los administradores de los sistemas desean obtener *fiabilidad* en sus redes de datos de tal manera de

asegurar en lo mínimo la confidencialidad, la integridad, y disponibilidad de los datos.

El asegurar el patrimonio de una empresa involucra también al *hardware* y al *software*. Se entiende como *hardware* a los equipos que permiten que la comunicación sea una realidad e inclusive medios físicos de respaldo de la información. Por otra parte, se entiende como *software* al conjunto de plataformas tales como sistemas operativos y aplicaciones que permiten manipular al *hardware*.

De esta manera, se entiende por seguridad lógica los métodos que permitan asegurar el *software*, y por seguridad física los métodos que permitan asegurar el *hardware*.

Es muy importante identificar los grupos de los cuales se debe proteger la información, entre los más importantes se pueden mencionar a los siguientes:

- *Personal de la empresa*

  A veces se tiende a pensar que internamente en una empresa hay un ambiente de confianza, lo cual en la mayor parte de los casos no es aplicable. Se debe tomar en cuenta que el dar acceso con privilegios a personal que no corresponda es comprometer la seguridad de los sistemas. Los daños al *software* o al *hardware* pueden ser ocasionados premeditadamente o por accidente, razón por la cual se debe mantener un estricto acceso por parte de los usuarios a los recursos y a los emplazamientos de los sistemas.

- *Ex – Empleados*

  Este grupo de personas es muy importante, ya que aquí se hallan antiguos empleados de las empresas, y en ese grupo se encuentran algunas que abandonaron la empresa en contra de su voluntad o

simplemente porque pasaron al lado de la competencia. De esta manera, al conocer perfectamente las vulnerabilidades y fortalezas del sistema pueden ocasionar graves daños. Por ello es una muy buena política el deshabilitar todos los accesos al sistema, tales como claves y tarjetas de acceso tan pronto como el empleado deje de pertenecer a la organización.

- *Curiosos*

Algunos empleados de la organización se sienten inconformes con la restricción de accesos que el sistema controla, a la vez que desean poner a prueba la fortaleza del esquema de seguridad y juntamente con el afán de curiosidad propia del ser humano, buscan herramientas que permitan romper dicho esquema y de esta manera conseguir acceso a información no autorizada.

- *Hackers*

El objetivo de este grupo es hacer uso de los elementos del sistema que están atacando. Los usos más comunes son obtener la información y vendérsela a la competencia u obtener acceso a la red y suplantar la identidad de la organización para llevar a cabo un ataque a otra organización, con el consiguiente deterioro de la imagen que se consigue al saber que se ha sido objeto de apoyo para un ataque informático.

a. POLÍTICAS DE SEGURIDAD

Se entiende por Políticas de Seguridad al grupo de reglas que especifican lo que está y lo que no está permitido en una organización. Generalmente estas reglas son elaboradas por el conjunto de directivos administrativos del sistema.

Las Políticas de Seguridad describen el cómo las entidades pertenecientes a un sistema obtienen acceso a los recursos. También se debe justificar la inversión en su implementación mediante un análisis costo – beneficio.

Las empresas invierten en seguridad de acuerdo a la naturaleza de su negocio, hay empresas de toda índole, ventas, mercadeo, transacciones, representaciones, entre otras; cada una con elementos diferentes en sus sistemas por defender. Se debe tomar en cuenta que hay un punto de equilibrio en el cual el costo de la inversión en seguridad iguala al costo que tendría el perder cierto elemento del sistema siendo víctima de un ataque. Éste es el punto en el cual se debería lograr mantener dicha inversión.

Un exhaustivo análisis de las amenazas del sistema ayuda enormemente en la elaboración de las Políticas de Seguridad; con este análisis se puede construir una lista de las posibles amenazas y colocar una ponderación a cada una de ellas, y en función de las amenazas más graves construir las Políticas de Seguridad.

Luego de definir las Políticas de Seguridad el siguiente paso es la definición del mecanismo de seguridad. Dicho mecanismo determina los pasos que deberán seguirse para cumplir con las Políticas diseñadas. El mecanismo puede ser tan sencillo como un control de acceso en el que se definan cuáles elementos del sistema puedan tener acceso hacia algún recurso.

Cualquier acción sea intencionada o no, la cual comprometa a las Políticas de Seguridad del Sistema, es considerada como una violación de seguridad.

1.2.2 ENCRIPCIÓN DE LOS DATOS

Se entiende por encripción al proceso de transformación del formato de

los datos; con esta transformación se logra que el viaje de los datos sea seguro. La seguridad de los datos se logra basándose en varias técnicas tales como encripción, desencripción y autenticación.

El proceso de transformar el formato de los datos logra que un atacante a pesar de ver los datos no pueda interpretarlos, bajo ningún punto de vista la encripción evita que un atacante pueda ver los datos, lo único que logra es que no entienda su contenido.

Se entiende por encripción de datos cualquier método que altere el formato original de los datos y que su proceso reverso, el de desencripción, sea complejo. Adicionalmente, se puede considerar como *Encripción Fuerte* al método que amerite poner a trabajar unos cuanto cientos de computadores en un mismo proceso para que luego de algunas decenas de centenas de años puedan hallar la información originalmente encriptada.

Un *Cipher* o Algoritmo Criptográfico es una función matemática que permite el mencionado cambio del formato de los datos sea para su encripción o desencripción.

Existen principalmente dos grupos en los cuales los *Ciphers* se encuentran agrupados. En el primero se tiene a los *Restricted Ciphers*, los cuales se caracterizan porque su seguridad reside en el secreto de la implementación del algoritmo criptográfico como tal; este secreto es relativamente difícil de mantener cuando se tiene algunos cientos de usuarios, pues es solo cuestión de tiempo para que alguien descubra dicho algoritmo y toda la seguridad se vea comprometida. Por otra parte se tiene a los *Keyed Ciphers*, los cuales son más utilizados en la actualidad; su algoritmo criptográfico al contrario de los anteriores es público, la seguridad de éstos reside en una llave con la cual se hace uso del algoritmo para cambiar el formato de los datos. Por lo tanto, en este caso la seguridad de los datos reside en la llave del

algoritmo criptográfico, mas no en el algoritmo mismo.

Si bien es cierto que la encripción de los datos permite que éstos puedan viajar seguros en medios hostiles tales como Internet, se necesita hacer uso de ciertos criterios que afiancen la confidencialidad tales como:

- *Autenticación*

  Es un método que permite verificar que el remitente de los datos es quien dice ser. Los atacantes pueden de alguna manera ser detectados por medio de la autenticación.

- *Chequeo de Integridad*

  Método utilizado para verificar que los datos que fueron enviados no hayan sido cambiados en el trayecto de su viaje. Los ataques que cambian los datos pueden ser detectados haciendo uso del chequeo de integridad.

- *No repudio*

  Haciendo uso de algoritmos que garanticen No Repudio, el receptor puede probar que el remitente realmente envió los datos en cuestión.

a. ENCRIPCIÓN SIMÉTRICA

La encripción simétrica se fundamenta en *Keyed Ciphers*, los mismos que basan su seguridad en una sola llave para cada sesión; ambos procesos encripción y desencripción hacen uso de esta única llave. Tanto el emisor como el receptor del mensaje encriptado deben conocerla.

Los pasos más relevantes para la comunicación se detallan a continuación:

1. El usuario A, encripta el mensaje usando la llave conocida.

2. El usuario B, desencripta el mensaje usando la misma llave usada para la encripción.

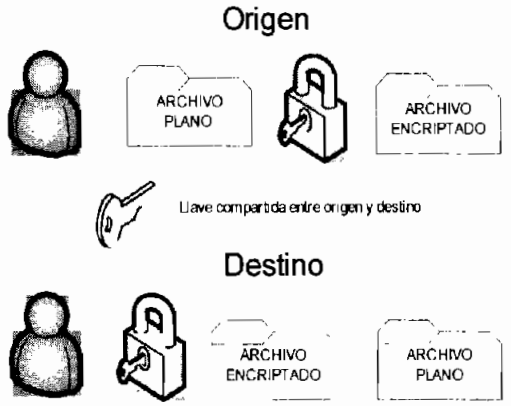


FIGURA 1.5 Proceso de encripción / desencripción con llaves simétricas.

La ventaja tiene que ver con el tiempo de procesamiento que necesitan los procesadores de los computadores para poder encriptar y desencriptar los mensajes. Otra ventaja constituye el hecho que pueden ser fácilmente implementados en *hardware*.

El problema cuando se hace uso de algoritmos basados en clave simétrica reside en el método usado por el transmisor para indicar al receptor cuál es la clave usada en la encripción y de esta manera poder desencriptar el mensaje.

Principalmente se puede clasificar a los algoritmos de encripción simétrica en dos grupos: Algoritmos de Bloque y Algoritmos de Flujo.

a.1 ALGORITMOS DE BLOQUE

Los algoritmos basados en codificación por bloque actúan sobre el mensaje tomando como referencia bloques de bits para llegar a su objetivo.

La encripción mediante algoritmos de bloque nace del uso de técnicas

criptográficas sencillas tales como: transposición, sustitución, adición, entre otras; se obtiene un algoritmo más fuerte que los sencillos originales.

Los algoritmos de Bloque tienen dos modos principales de operación: EBC *Electronic BookCode Mode* y CBC *Cipher Block Chaining.*

EBC (*ELECTRONIC BOOKCODE MODE)*

Haciendo uso de este algoritmo se tiene bloques fijos de 64 bits, es decir, se tienen 2^{64} posibles bloques de bits de entradas, los cuales tienen 2^{64} correspondientes salidas.

De esta manera un ataque sencillo puede construirse en base a un *BookCode* en donde se tenga una correspondencia entre el bloque plano y el bloque cifrado resultante.

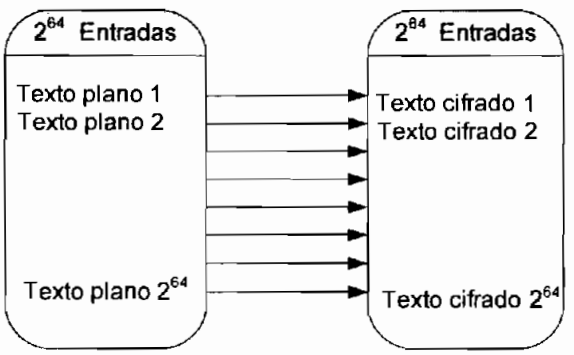


FIGURA 1.6 Electronic BookCode Mode

CBC (*CIPHER BLOCK CHAINING)*

Debido a la vulnerabilidad que presenta el uso de EBC, se toma como alternativa al modo CBC. En esta técnica el texto encriptado resultante sirve como entrada para la encripción del siguiente texto plano, siendo, de esta manera, el texto encriptado resultante dependiente tanto del texto plano de entrada como del texto encriptado que se generó anteriormente.

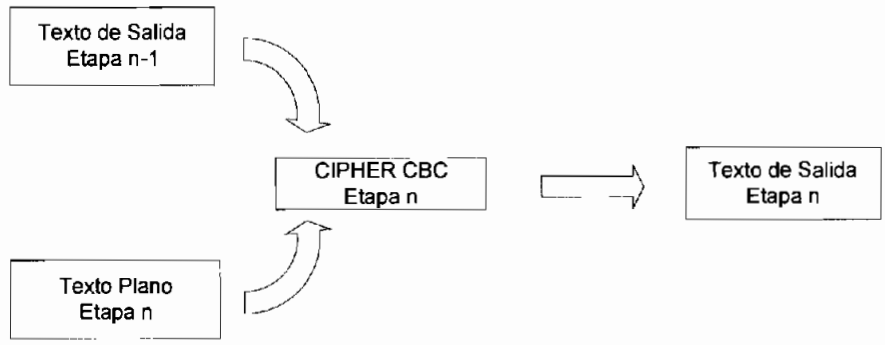


FIGURA 1.7 Cipher Block Chaining

a.2 ALGORITMOS DE FLUJO

Los algoritmos de flujo se caracterizan por ser más rápidos que los algoritmos de bloque ya que actúan sobre el texto plano pero solo sobre un *bit* o un *byte* a la vez.

Tampoco se puede atacar un algoritmo de flujo basándose en un *codebook* ya que el resultado de encriptar un texto plano es siempre diferente, puesto que para la encripción del texto plano actual se toma como entrada el texto plano y el texto plano precedente.

a.3 BREVE DESCRIPCIÓN DE LOS ALGORITMOS DE ENCRIPCIÓN SIMÉTRICA MÁS USADOS

DES - DATA ENCRIPTION STANDARD

"Lucifer" fue el nombre original de este popular estándar de encripción de datos desarrollado por IBM *International Business Machines*.

DES como fue llamado luego por el gobierno de los Estados Unidos de Norteamérica en su forma original utiliza una llave de 56 bits y un bit de paridad en cada octavo bit, lo que origina una llave de 64 bits. De esta llave de 64 bits se originan otras 16 sub-llaves usadas en el proceso de encripción.

DES es uno de los algoritmos de *encripción por bloque* más conocidos, pues trabaja sobre bloques de 64 bits de texto plano para generar bloques de 64 bits de texto encriptado.

El ataque más práctico en contra de DES es el llamado ataque de fuerza bruta, el cual consiste en probar todas las posibles combinaciones hasta obtener resultados favorables al ataque.

Tal como se muestra en la figura 1.9, de manera general, la encripción DES consta de 19 etapas.

- La primera etapa es una transposición de los datos independientemente de la llave de 56 bits.
- La última etapa es exactamente inversa a la primera haciendo uso de la transposición inicial.
- La etapa previa a la última intercambia los 32 bits de la derecha hacia la izquierda y los 32 bits de la izquierda hacia la derecha.
- Las otras dieciséis etapas restantes intermedias están en esencia basadas en la misma lógica. En cada etapa se toman dos entradas de 32 bits y se originan dos salidas igualmente de 32 bits. En la parte izquierda se tiene una copia idéntica de la entrada de 32 bits de la derecha, mientras que, la salida de 32 bits de la derecha es el resultado de una función que toma como entrada los 32 bits de la izquierda y una de las 16 llaves generadas previamente y correspondiente a la etapa. La función que se ejecuta es un OR EXCLUSIVO a nivel de bit entre los 32 bits de la derecha y la mencionada sub-llave de la etapa.

Para el proceso de desencripción del algoritmo DES, solamente se hace el proceso contrario a la encripción.

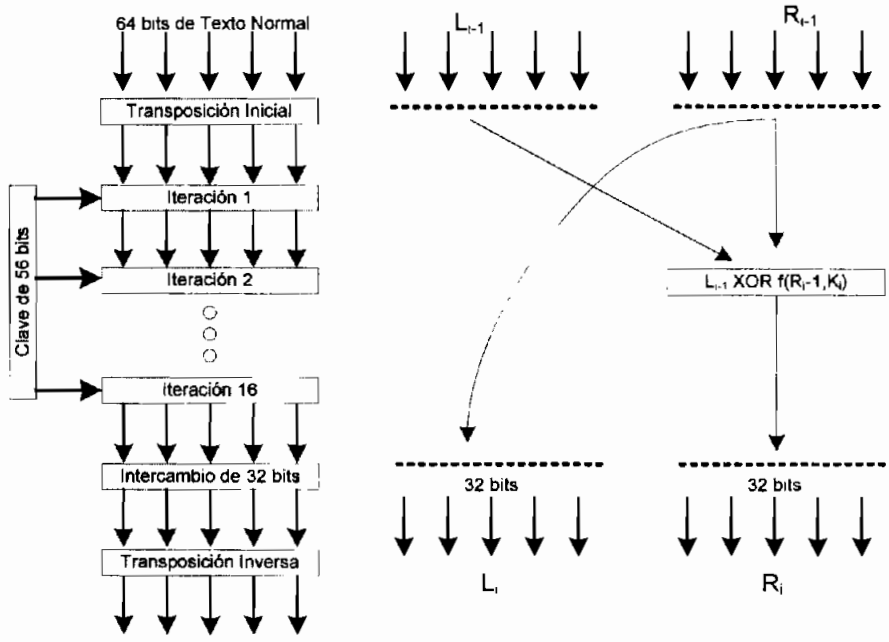


FIGURA 1.9 DES, Esbozo General y Detalle de una Iteración [14]

3DES - TRIPLE DATA ENCRIPTION STANDARD

Triple DES fue creado luego de las evidentes vulnerabilidades de su algoritmo predecesor DES; se hicieron cambios con el objetivo de mejorar su seguridad, viéndose afectada la longitud de la llave total pero sin perder su esencia de algoritmo simétrico de *encripción por bloque*.

Existen varias implementaciones de Triple DES, una de las más conocidas es EDE acrónimo de Encripción, Desencripción y Encripción (figura 1.10), en la cual se usan dos llaves diferentes de 56 bits y tres etapas.

En la primera etapa EDE ejecuta el proceso de encripción haciendo uso de la primera llave, en la segunda etapa se desencripta pero con la segunda llave, y en la etapa final, se ejecuta nuevamente el proceso de encripción usando nuevamente la primera llave. El resultado final, EDE tiene una llave de encripción de 112 bits los cuales son más que suficientes para las aplicaciones

comerciales actuales.

También existe una implementación de Triple DES llamada EEE acrónimo de Encripción, Encripción y Encripción (figura 1.10), la cual usa tres llaves y tres etapas; el resultado, una llave de 168 bits la cual a pesar de ser extremadamente segura, solamente agrega carga extra innecesaria al transporte y administración de las llaves.

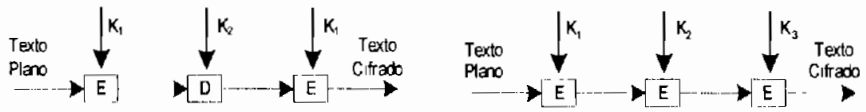


FIGURA 1.10 Triple DES versiones, EDE y EEE [4]

IDEA - INTERNATIONAL DATA ENCRIPTION STANDARD

El algoritmo Internacional de Cifrado de Datos fue creado para ser más rápido que DES y para que pueda ser implementado tanto en software como en hardware.

IDEA usa una llave de 128 bits para generar 52 subclaves de 16 bits cada una.

Para el proceso de encripción, a pesar de la extensa alteración de bits, IDEA necesita solamente ocho iteraciones para generar su texto encriptado.

IDEA es también un algoritmo de *encripción por bloque* y toma al igual que DES bloques de 64 bits de texto plano. En cada una de las ocho iteraciones IDEA usa seis de las cincuenta y dos sub-llaves generadas y las cuatro restantes las usa en la transformación final. El proceso de desencripción de IDEA es exactamente el inverso que el proceso de encripción.

Dada la complejidad con la cual IDEA fue creado, no se han reportado

ataques exitosos, por lo cual se puede considerar a IDEA como un algoritmo de encripción seguro.

RC4 RIVEST CHIPER # 4

Creado por Ron Rivest en RSA *Data Security Inc.* RC4 al contrario de los anteriormente mencionados es un algoritmo de *encripción por flujo* el cual posee una llave variable de hasta 256 bits y se implementa con operaciones orientadas a byte.

RC4 se encuentra implementado en software con muy buenos resultados lo cual ha sido corroborado con el estudio de matemáticos y criptoanalistas.

b. ENCRIPCIÓN ASIMÉTRICA

Estos algoritmos son también conocidos como algoritmos de llave pública. Todos los usuarios tienen un par de llaves, una privada y una pública; para el proceso de comunicación todos los usuarios deben conocer la llave pública del usuario con el cual desean intercambiar información.

b.1 CLAVES ASIMÉTRICAS, FUNCIONAMIENTO

Los pasos más relevantes para la comunicación se detallan a continuación y se esquematizan en la figura 1.11:

1. Se asume que los usuarios han generado su par de llaves previamente para su uso y que su llave pública ha sido distribuida.
2. El usuario A, encripta el mensaje usando la llave pública del usuario B
3. El usuario B, desencripta el mensaje usando su propia llave privada.
4. El usuario B, encripta otro mensaje usando la llave pública del usuario A
5. El usuario A, desencripta el mensaje usando su propia llave privada.

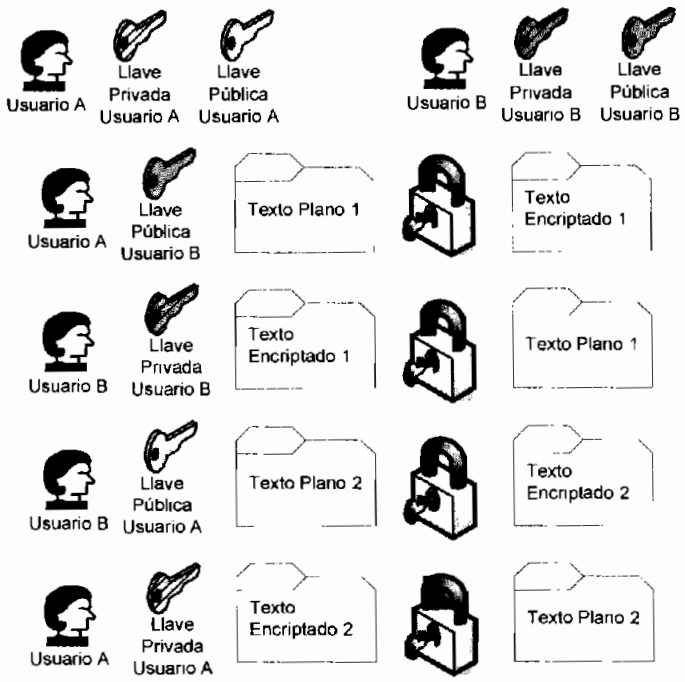


FIGURA 1.11 Encripción Asimétrica

Haciendo uso del esquema de encripción antes mencionado los usuarios pueden tener *Privacidad* en la comunicación. Si se desea tener *Autenticación* el usuario A debe encriptar sus datos con su llave privada y el usuario B desencriptará los datos recibidos con la llave pública del usuario A.

La desventaja del uso de los algoritmos que hacen uso de llaves asimétricas tiene que ver principalmente con el rendimiento de los procesadores de los computadores. Pues son necesarios muchos más recursos computacionales (en comparación con los recursos usados en los algoritmos simétricos), para poder procesar las etapas de encripción / desencripción de los mensajes, lo cual conlleva a que los servidores puedan manejar muchas más conexiones de tipo simétrico que de tipo asimétrico.

La ventaja reside en el manejo de las llaves de la comunicación, en este entorno los usuarios no revelan secretos, y las llaves privadas son resguardadas

por cada uno de los usuarios lo cual lo convierte en un sistema de administración descentralizada difícil de atacar; en tanto que las llaves públicas son distribuidas por los usuarios como un recurso de dominio público. Pero este sistema ve deteriorado su rendimiento cuando el número de usuarios es extenso, ya que es necesario saber la llave pública de todos los demás lo cual se refleja en el consumo de recursos de almacenamiento.

Existen varios malentendidos cuando se aborda el tema de la encripción asimétrica. El primero de ellos, es pensar que los sistemas de encripción asimétrica son más resistentes al criptoanálisis que los sistemas de encripción simétrica; en realidad la seguridad de todo sistema de cifrado radica en la longitud de la llave y el esfuerzo computacional requerido para hallarla, razón por la cual, en principio no hay ninguna diferencia en cuanto a resistencia a criptoanálisis.

De igual manera, se cree que el aparecimiento de técnicas de encripción asimétrica han dejado obsoletas a técnicas de encripción simétrica, lo cual es totalmente falso, pues más bien son usadas en conjunto debido a los grandes recursos computacionales necesarios para el uso de encripción asimétrica. Usualmente, en el proceso de establecimiento de la sesión e intercambio de las llaves se usan algoritmos de encripción asimétrica, mientras que, para el intercambio de la información durante la sesión se usa algoritmos de encripción simétrica.

b.2 BREVE DESCRIPCIÓN DE LOS ALGORITMOS DE ENCRIPCIÓN ASIMÉTRICA MÁS USADOS

RSA - *RIVEST SHAMIR ADELMAN*

Este algoritmo de llave pública fue creado por un grupo de investigadores del MIT (*Massachusetts Institute of Technology*) en el año 1978, sus creadores fueron Ron Rivest, Adi Shamir, y Len Adelman.

La fortaleza de este algoritmo de encripción radica en la dificultad de factorar números primos realmente grandes, los mismos que se encuentran en el rango de 200 dígitos o más. RSA es un algoritmo de *encripción por bloque* y se caracteriza porque tanto el texto plano como el texto encriptado son números enteros.

Potenciales maneras de llevar a cabo ataques en contra de RSA se han desarrollado. La primera es haciendo uso de fuerza bruta con lo que se debe probar todas las claves privadas posibles para generar el texto cifrado interceptado. La solución para este caso es sencilla, solamente se debe aumentar el número de bits de la llave privada y los atacantes deberían dedicar algunos cientos de años extras para lograr su objetivo.

Otra forma de atacar a RSA se ha desarrollado en función del objetivo de factorar el producto de los números primos y obtener los números primos originales, lo cual es computacionalmente muy complejo; aunque se han registrado intentos exitosos haciendo uso de este método, lo único que se ha hecho es aumentar el tamaño de las llaves (actualmente se usan llaves de hasta 1024 bits), con lo cual se han visto inoperantes todo tipo de nuevos ataques a este algoritmo.

AES - *ADVANCED ENCRIPTION STANDARD*

Los autores, Joan Daemen y Vincent Rijmen, con su tesis de doctorado vencieron a criptólogos de considerable fama mundial y a empresas de renombre como IBM y RSA entre otras.

En Agosto del 2001 "RIJNDAEL", como lo llamaron sus creadores era seleccionado por el Instituto Nacional de Estándares y Tecnología (NIST) norteamericano como AES (Estándar de Cifrado Avanzado). Hasta ese entonces era DES el algoritmo oficial de los Estados Unidos de Norteamérica.

Principalmente los motivos que llevaron al NIST a escoger a "Rijndael"

fueron su buena combinación de aspectos vitales tales como: Seguridad, Velocidad, Eficiencia, Sencillez y Flexibilidad.

El algoritmo RIJNDAEL o más conocido como AES, es un sistema de cifrado por bloques, los mensajes se toman en bloques de 128 bits. Hay varias versiones del sistema utilizando claves de 128, 192 o 256 bits.

c. ENCRIPCIÓN HÍBRIDA

Éste es uno de los métodos más utilizados para garantizar, confidencialidad tanto de los mensajes del emisor como del receptor.

Los pasos más relevantes para la comunicación se detallan a continuación y se muestran en la figura 1.12:

El usuario A desea transmitir el mensaje X al usuario B, el usuario A conoce la llave pública del usuario B.

1. El usuario A hace uso de algún algoritmo simétrico para generar la llave X1.
2. El usuario A encripta el mensaje X con una llave X1.
3. Haciendo uso de algún algoritmo asimétrico, la clave X1 es encriptada usando la clave pública del usuario B.
4. Todo viaja en un solo paquete.
5. El usuario B conoce su propia llave privada.
6. El usuario B toma el paquete, haciendo uso del mismo algoritmo asimétrico usando por el usuario A desencripta la clave X1 con su clave privada.
7. Luego de obtener la clave simétrica X1 procede a descifrar el mensaje original X.

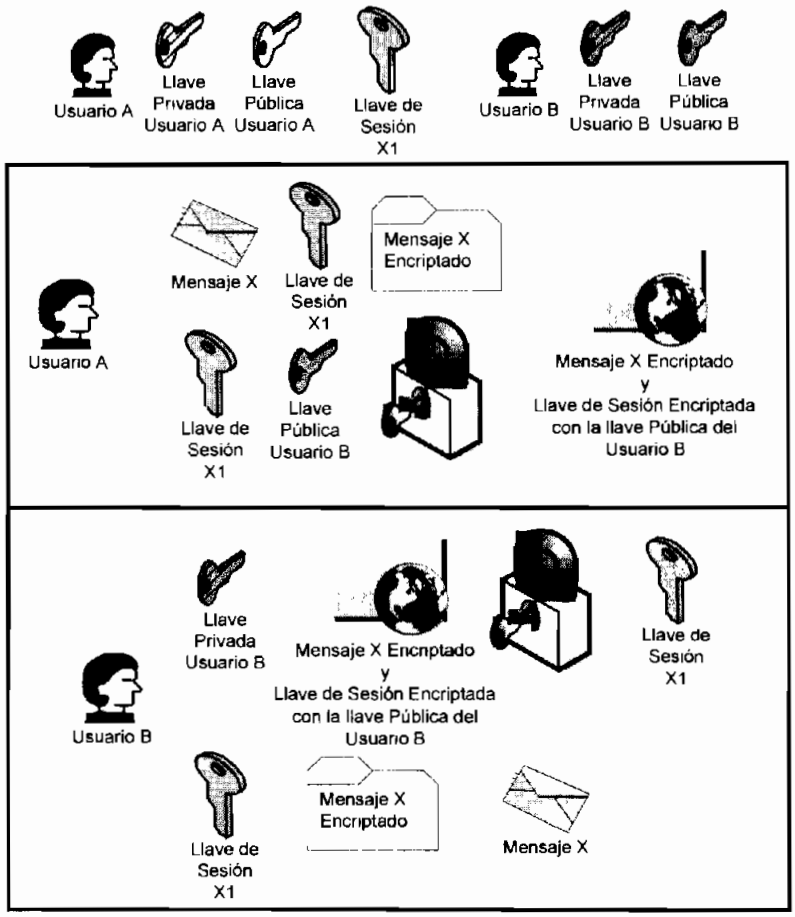


FIGURA 1.12 Encripción Híbrida

Con este método se garantiza, confidencialidad, pues solo el usuario B puede descifrar la clave simétrica X1 ya que nadie más tiene la clave privada.

Se podría también con el cifrado híbrido garantizar autenticación. Si en el paso 2 en lugar de encriptar la clave X1 con la clave pública del usuario B, se lo hace con la clave privada del usuario A. Entonces el usuario B puede hacer uso de la clave pública del usuario A conocida por todos, y hallar la clave simétrica X1 y garantizar que ha sido el usuario A quien ha encriptado ese mensaje.

1.2.3 INTEGRIDAD DE LOS MENSAJES

El problema de cómo mantener los datos inalterados y fidedignos a su origen es tratado por la integridad de los mensajes. Habrá ocasiones en las cuales los atacantes puedan visualizar los datos, pero haciendo uso de la integridad se logra que éstos no puedan alterarlos.

a. FUNCIÓN *HASH*

La función *hash* en general es usada para obtener una serie de caracteres hexadecimales, los cuales son una especie de resumen del texto que la función tiene como entrada.

La función *hash* tiene una caracteristica importante. La salida siempre es distinta, si se altera una sola letra en el mensaje de entrada, la salida será totalmente diferente.

Este tipo de función además no es reversible, es decir, no es posible obtener el texto de entrada basándose en el flujo de caracteres hexadecimales que se han obtenido como salida. Además son fáciles de computar pero definitivamente complejos de reversar.

Adicionalmente, una función *hash* debe ser resistente a colisiones, es decir, que no debe producir dos valores iguales a la salida cuando las entradas son diferentes. La idea tras de esta característica es proveer una *huella* única del mensaje, la cual lo identifique inequívocamente tal como la huella dactilar lo hace con las personas.

El MAC *Message Authentication Code* puede ser generado a partir de funciones *hash.* Para generar un MAC, la entrada de una función *hash* es función tanto del texto que se desea computar como de una llave (por lo general la llave de la sesión). El uso del MAC se describe a continuación (figura 1.13):

1. El usuario A concatena el texto y la llave como entradas para la función *hash*.
2. El resultado es enviado al usuario B junto con el texto plano original.
3. El usuario B conociendo la llave de la sesión efectúa localmente el cómputo del MAC con el texto plano original que ha recibido.
4. Si el MAC generado como resultado coincide con el MAC recibido, entonces se puede asumir que el texto no ha sido modificado en su trayectoria.

* El MAC puede ser usado tanto con encripción simétrica como asimétrica (haciendo uso de la llave pública).

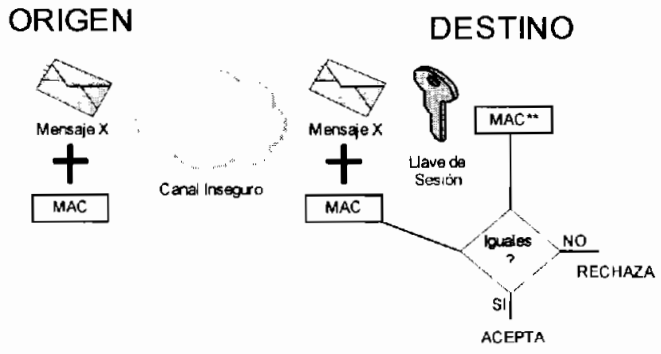


FIGURA 1.13 Message Authentication Code

También existe otro uso de las funciones *hash*; las firmas digitales, las mismas que consisten en encriptar el resultado de una función *hash* mediante el uso de una llave privada. Las firmas digitales son un método utilizado para la distribución de llaves simétricas.

La firma digital de los documentos es usada comúnmente para verificar, tanto la integridad de los datos enviados como la autenticación del emisor de los mismos. El funcionamiento básico se indica en los siguientes pasos (figura 1.14):

1. El usuario A tiene una llave pública, la cual es conocida por el usuario B.

2. El usuario A, aplica al mensaje X un algoritmo que usa la función *hash* y obtiene un valor *hash*.
3. El usuario A, encripta el valor *hash* con su llave privada.
4. Viajan, el mensaje sin encriptar y el valor *hash* encriptado
5. El usuario B, hace uso de la llave pública del usuario A para desencriptar el valor hash originalmente generado.
6. El usuario B, al conocer el texto, aplica el algoritmo usado por el usuario A para hallar su propio valor *hash*.
7. Finalmente el usuario B compara, el valor *hash* que le llegó desde el usuario A, y el valor *hash* hallar por él mismo.

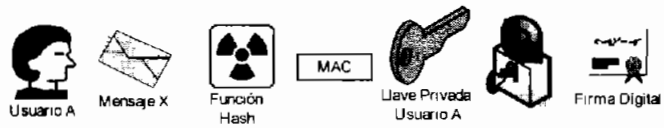


FIGURA 1.14 Firma Digital

Si los valores *hash* son los mismos, entonces se verifica la integridad del mensaje ya que no ha sido cambiado en el trayecto (es decir, está íntegro), y también se autentica la identidad del usuario que envió el mensaje gracias al uso de la encripción asimétrica.

a.1 BREVE DESCRIPCIÓN DE LOS ALGORITMOS DE INTEGRIDAD DE MENSAJES MÁS USADOS

Los algoritmos más conocidos que usan funciones *hash* en sus implementaciones son:

- MD4 *Message Digest* con 128 bits de salida.
- MD5 *Message Digest* con 128 bits de salida.
- SHA *Secure Hash Algorithm* con 160 bits de salida.

En la tabla 1.2 se muestra el resultado de aplicar algunos algoritmos de

hashing al texto "*amigo*" y "*amiga*"

Función Hash	Texto de Entrada	
	Amigo	amiga
SHA-1 160 bits	01F07BEB8567E3A651DCF08CE9A403DC03AAF2DC	11C67D630121EE165471BDC37ABAC50F13B9393E
MD4 128 bits	0B3745425CAFFDA337DBA33D1534D9C7	343A74F8D6ACC33D0D50116AEE55E722
MD5 128 bits	D94729CE13F4EE6395BFC6F1080CC986	836821929CB02AADE5072AEC0C1AA93C

TABLA 1.2 Algoritmos de Hash

Como se puede observar claramente en la tabla 1.2, en todos los casos la alteración de una sola letra del mensaje de entrada produce valores de salida totalmente distintos.

Los algoritmos más usados actualmente son MD5 y SHA-1. El primero de ellos fue diseñado por Ron Rivest quien fue co – autor de RSA. Por su parte SHA-1 fue diseñado por el NIST (*National Institute of Standards and Technology*) en colaboración con NSA (*National Security Agency*).

MD5 produce resultados *hash* de 128 bits; mientras que SHA-1 produce resultados de 160 bits razón por la cual se le acredita mayor seguridad. Ninguno de estos algoritmos toma una llave como entrada, por lo que en principio ninguno sirve para generar un MAC, pero en la práctica es sencillo concatenar el texto con una llave e ingresarlo a la función *hash*, con lo cual se calcularía el mencionado MAC.

1.2.4 AUTENTICACIÓN DE LOS DATOS Y NO REPUDIO

a. CERTIFICADO DIGITAL

El Certificado Digital tiene por objetivo garantizar que la información

contenida en él sea veraz. Con el uso del mismo se consigue que los usuarios quienes reciben información, puedan autenticar a los usuarios emisores y basándose en la identificación confiar o no en el contenido de los datos.

Es una variante del uso de claves asimétricas para la comunicación, pues ahora entra la figura de la Autoridad Certificadora (AC), concepto que se explicará más adelante.

La norma X.509v3 emitida por la Unión Internacional de Telecomunicaciones (ITU) para certificados digitales es la más usada hoy en día, así se especifica que los campos contenidos sean los siguientes:

a.1 ELEMENTOS DEL CERTIFICADO DIGITAL

1. Identificador del propietario
   - ✓ Nombres y Apellidos;
   - ✓ Correo electrónico;
   - ✓ Datos de la empresa tales como:
     - o Organización;
     - o Departamento;
     - o Localidad;
     - o Provincia;
     - o País, entre otras.
2. Identificador de la validez del CD, el cual está a cargo de la AC que lo emite.
3. Fechas de validez del CD
   - ✓ Fecha de Inicio; y
   - ✓ Fecha de Fin. (Luego de este período la llave pública contenida no debe usarse para cifrar o firmar documentos).
4. Identificador del CD. Número de serie, para que la AC identifique unívocamente al certificado emitido.

5. Datos varios tales como:
   - ✓ Versión: 3 en caso de usar X509v3
   - ✓ Algoritmo utilizado por la AC
   - ✓ Nombre de la AC
   - ✓ Extensiones

Finalmente, la firma de la AC la cual garantiza que todos los datos en todos lo campos del certificado son auténticos.

Las extensiones de los certificados pueden ser de dos tipos, las informativas y las restrictivas.

En las *extensiones informativas*, se tiene información adicional sobre el certificado tal como la del dueño del mismo.

Las *extensiones restrictivas*, son un conjunto de condiciones que deben cumplirse para que el CD sea válido. Sirve para que la AC restrinja el uso del mismo dentro de una empresa o de un país.

a.2 TIPOS DE CERTIFICADOS DIGITALES

Existen varias clasificaciones de los certificados digitales, una de las más importantes es la siguiente ya que engloba a la mayoría de usuarios de Internet:

1. Certificados Digitales Personales
2. Certificados Digitales de Servidores
3. Certificados Digitales de Fabricantes de Software
4. Certificados Digitales de Autoridades Certificadoras.

CERTIFICADOS DIGITALES PERSONALES

Los certificados digitales personales sirven para autenticar a un cliente ya sea frente a un servidor o frente a otro cliente.

Dentro de las aplicaciones más comunes está el transporte de correo de manera segura, de esta manera es posible la autenticación de los usuarios que emiten el correo.

CERTIFICADOS DIGITALES DE SERVIDORES

Estos certificados digitales sirven para autenticar a los servidores ante los clientes y garantizar que la información que muestra está verificada por una autoridad certificadora y que el usuario puede confiar en sus contenidos.

También estos certificados, son utilizados para transacciones entre servidores haciendo uso de conexiones seguras usando protocolos tales como SSL (*Secure Sockets Layer*).

La aplicación más común de este tipo de certificados se tiene en empresas que poseen una LAN de servidores los cuales prestan sus servicios tanto en la Intranet como hacia el Internet, para lo cual es necesario que presenten sus identificaciones a sus clientes.

CERTIFICADOS DIGITALES DE FABRICANTES DE SOFTWARE

Este tipo de Certificados Digitales son otorgados a las compañías fabricantes de software, sirven para que ellas puedan firmar el código que distribuyen.

CERTIFICADOS DIGITALES DE AUTORIDADES CERTIFICADORAS

Al analizar un certificado y el nombre de autoridad que lo ha remitido, se puede observar que ésta tiene una Autoridad Certificadora sobre ella, y ésta a su vez otra sobre si misma. Se forma de esta manera una jerarquía de Autoridades Certificadoras.

Lógicamente, el certificado de una Autoridad Certificadora estará firmado por un Autoridad Certificadora superior en la jerarquía.

Otra clasificación interesante de certificados digitales puede ser la siguiente:

HIGH TRUST SIGNING CERTIFICATES

Proveen un alto nivel de protección de la llave privada y del certificado mismo. Las llaves privadas están almacenadas generalmente en dispositivos de hardware tales como *SmartCard* o USB los mismos que son provistos por la autoridad certificadora y viene incluido en el precio del certificado el cual es muy costoso.

MEDIUM TRUST SIGNING CERTIFICATES

Proveen un grado medio de protección de la llave privada y del certificado mismo ya que las llaves están protegidas por software; por otra parte sus precios son menores que los anteriores.

BASIC TRUST SIGNING CERTIFICATES

Son los más conocidos en el mercado. Su protección es mucho más básica que los anteriores ya que los certificados residen en el *Web Browser* de su propietario.

b. AUTORIDAD CERTIFICADORA

La Autoridad Certificadora es quien firma un Certificado Digital y garantiza que los datos contenidos en él son auténticos.

La confianza de los usuarios en la Autoridad Certificadora es vital para el

buen funcionamiento de la infraestructura. Ésta es quien debe cuidar mucho más que los usuarios su clave privada, ya que si alguien llegara a obtenerla, podría hacer cambios en los certificados digitales emitidos y firmarlos nuevamente, como si fuera la Autoridad Certificadoras mismo.

Las tareas más comunes de las Autoridades Certificadoras son:

b.1 REVOCACIÓN DE CERTIFICADOS DIGITALES

La revocación de los certificados puede darse por varias razones:

1. La clave privada del certificado se ha visto comprometida.
2. El uso al cual fue otorgado no es el mismo para el cual está siendo usado.

b.2 RENOVACIÓN DE CERTIFICADOS

Generalmente los CD tienen una duración de un año, luego del cual el CD expira, entonces el usuario titular del CD deberá pedir a la AC una renovación del CD.

b.3 EMISIÓN DE CERTIFICADOS DIGITALES

Es la actividad principal de la autoridad certificadora, aquí se involucran procesos y procedimientos en los cuales participan tanto la autoridad certificadora como el cliente quien desea un certificado digital.

c. PROCESO DE GENERACIÓN DE UN CERTIFICADO DIGITAL

Se muestra a continuación los pasos que un usuario debe realizar en colaboración de una AC para obtener un CD.

c.1 GENERAR CLAVE

Este paso lo lleva a cabo el usuario, genera una clave privada. La clave privada estará siempre en custodia del usuario, y la AC nunca la conocerá. Por

ello es muy importante resguardar en un lugar seguro la llave privada.

** Para la explicación siguiente se asume que el Servidor Apache ha sido instalado haciendo uso de la distribución de Linux Red Hat 9.0

En el sistema operativo Linux Red Hat 9.0 los comandos para generarla son los siguientes:

1. Colocarse en el directorio "conf"

```
cd /etc/httpd/conf
```

2. Borrar la llave y el certificado que fueron generados para el servidor en el momento de la instalación del servidor Web Apache.

```
rm ssl.key/Server.key
rm ssl.crt/Server.crt
```

3. Localizarse en el directorio "certs"

```
cd /usr/share/ssl/certs
```

4. Generar la nueva llave

```
make genkey
```

El sistema pedirá un PEM (*pass phrase*) la cual sirve para generar la llave privada, es recomendable que contenga una combinación alfanumérica y no posea palabras basadas en diccionario por motivos de seguridad. Luego de verificar por segunda vez el *pass phrase,* Linux Red Hat 9.0 por defecto genera una llave privada haciendo uso del algoritmo de encripción asimétrica RSA con una longitud de 1024 bits.

5. Ahora se puede verificar que la nueva llave del servidor (*server.key*) se

localice en el directorio "ssl.key" el cual contiene el archivo.

```
cd /etc/httpd/conf/ssl.key/
```

c.2 GENERAR Y FIRMAR LA PETICIÓN DEL CERTIFICADO (CSR)

El usuario llena un formulario, el cual contiene información necesaria para verificar su identidad contra la AC, posteriormente este formulario es firmado digitalmente con la llave privada del usuario y se envía a la AC juntamente con su llave pública. Este proceso es el que se conoce como generación de una solicitud de CD o CSR (*Certificate Signing Request*)

1. Situarse en el directorio "certs"

   ```
   cd /usr/share/ssl/certs
   ```

2. Iniciar la generación de la petición del certificado

   ```
   make certreq
   ```

   El sistema pedirá nuevamente una *pass phrase* y se debe ingresar la misma que se ingresó al generar la llave privada.
   Luego se deben llenar datos necesarios para el requerimiento tales como:
   País, Estado, Ciudad, Nombre de la Organización, Nombre de la Unidad Organizacional, nombre del dominio, dirección de correo electrónico del administrador, entre otros.

3. El archivo "server.csr" ubicado en "/etc/httpd/conf/ssl.csr" constituye el CSR el mismo que debe ser enviado a la autoridad certificadora que ha sido seleccionada por el usuario.
   Para este caso se ha seleccionado a Verisign como autoridad certificadora; el texto del archivo "Server.csr" debe ser copiado y pegado

en la siguiente dirección de Internet:

http://www.verisign.com/products/srv/trial/intro.html

Luego que Verisign ha verificado los datos que han sido enviados, se proveerá un certificado digital de prueba el cual tiene una duración de 15 días; el certificado digital se remite vía correo electrónico a la dirección suministrada en el CSR y tiene un formato parecido a lo siguiente:


```
-----BEGIN CERTIFICATE-----
MIIDSDCCAvKgAwIBAgIQbW5bARXeubTstgsAw6avqjANBgkqhkiG9w0BAQUFADCB
qTEWMBQGA1UEChMNVmVyaVNpZ24sIEluYzFHMEUGA1UECxM+d3d3LnZlcmlzaWdu
LmNvbS9yZXBvc2l0b3J5L1Rlc3RDUFMgSW5jb3JwLiBCeSBSZWYuIExpYWIuIExU
RC4xRjBEBgNVBAsTPUZvciBWZXJpU2lnbiBhdXRob3JpemVkIHRlc3Rpbmcgb25s
eS4gTm8gYXNzdXJhbmNlcyAoQylWUzE5OTcwHhcNMDQwMzE2MDAwMDAwWhcNMDQw
MzMwMjM1OTU5WjCBhzELMAkGA1UEBhMCRUMxEjAQBgNVBAgTCVBpY2hpbmNoYTEO
MAwGA1UEBxQFUXVpdG8xFDASBgNVBAoUC3Rlc3QgZGUgc3NsMR0wGwYDVQQLFBRl
bmlkYWQgZGUgZGVzYXJyb2xsbzEfMB0GA1UEAxQWd3d3LmVhbmd1ZXRhc3NsZXBu
LmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAyqibYsq4qlRu4pE2zUNS
RmfAbPQNNoL6zhCW4JYBOYhnhkmDbTH59J1NPsWMAU161W5EJ+pu6v+i+8nhLZ81
15Y1rx8kjAGX7UnX0bh+Yxl8j1qKIPDvXz1tRmyDA2YDSDeRj9NYqOnk21RY6Rv1
ce9gVVDi51gIK3NU9O1aNeMCAwEAAaOB0TCBzjAJBgNVHRMEAjAAMAsGA1UdDwQE
AwIFoDBCBgNVHR8EOzA5MDegNaAzhjFodHRwOi8vY3JsLnZlcmlzaWduLmNvbS9T
ZWN1cmVTZXJ2ZXJUZXN0aW5nQ0EuY3JsMFEGA1UdIARKMEgwRgYKYIZIAYb4RQEH
FTA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LnZlcmlzaWduLmNvbS9yZXBvc2l0
b3J5L1Rlc3RDUFMwHQYDVR01BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMA0GCSqG
SIb3DQEBBQUAA0EASLq8/+/W/+STuiY6uh0fh1grCHIaP+RE97A6NOO+zlagiinh
g03Kro0mYgkHtURJk88Zbs/W5rzpQTbvQo53Bw==
-----END CERTIFICATE-----
```


c.3 VERIFICACIÓN DE LA IDENTIFICACIÓN

Haciendo uso de la llave pública enviada por el usuario, la AC, verifica primero la existencia del par de llaves, ya que si no existiese una llave privada, los datos no hubiesen sido posibles descifrar con la llave pública enviada.

Además, depende de la AC que exija algún tipo de información adicional para poder verificar la autenticidad del solicitante.

Otro de los objetivos de este paso es, transportar los datos del CSR de manera segura ya que viajan encriptados con la llave privada del solicitante.

c.4 GENERACIÓN DEL CERTIFICADO DIGITAL

En este estado, la AC empaqueta todos los datos del solicitante, en la estructura del CD y lo firma digitalmente con su llave privada.

c.5 VERICACIÓN DEL CERTIFICADO DIGITAL

El solicitante, verifica que los datos en el CD estén correctos, para lo cual necesita descifrarlos del CD enviado por la AC haciendo uso de la llave pública de la AC.

c.6 PUBLICACIÓN DEL CERTIFICADO DIGITAL

El solicitante puede comenzar a hacer uso de su CD. Desde este instante el usuario está autorizado a distribuir su certificado digital junto con sus datos.

Adicionalmente, para el caso de Verisign como Autoridad Certificadora es necesario descargar e instalar un archivo cuyo nombre es "*getcacert.cer*" el cual contiene el CD de una AC de prueba de Verisign desde la dirección de Internet:

http://www.verisign.com/server/trial/faq/index.html

Para la instalación de la Autoridad Certificadora de Prueba en el *Web Browser* Microsoft Internet Explorer, se selecciona *Certificates* desde *Internet Options*, *Content* (figura 1.15). Luego de lo cual se direcciona el archivo "*getcacert.cer*" y se completa la importación; en la figura 1.16 se puede verificar el certificado digital de prueba de la Autoridad Certificadora llamada "*Verisign Autorized Testing Only*".

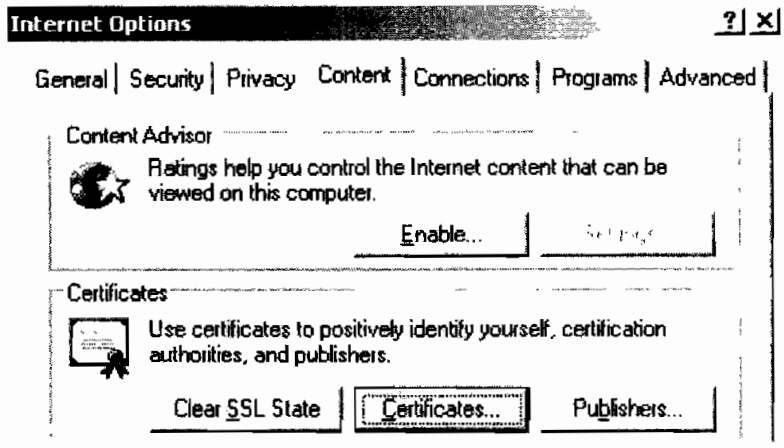


FIGURA 1.15 Visualización de Certificados

FIGURA 1.16 Certificado Digital de la Autoridad Certificadora de Prueba

1.2.5 MÉTODOS MÁS USADOS PARA ROMPER LA SEGURIDAD

Hay algunos puntos a considerar cuando de la vulnerabilidad de un sistema se habla, pero sin lugar a duda las personas relacionadas al mismo constituyen la mayor amenaza; se puede citar el caso de un administrador que no tiene la suficiente experiencia o el de un guardia que sin necesariamente

tener acceso al sistema deja ingresar a cualquier persona al cuarto de comunicaciones.

Pero existen otros tipos de amenazas del personal que está relacionado con el sistema y no necesariamente son ataques, sino que podrían ser catalogados como accidentes, sin embargo se debe evitar llegar al caso de decir *"no lo hice a propósito"* pues no ayudará a recuperar los datos perdidos.

a. INGENIERÍA SOCIAL

Este tipo de ataque se basa en conseguir que los usuarios revelen al atacante información confidencial como por ejemplo claves de acceso a los recursos. Desafortunadamente, a pesar de ser uno de los métodos más sencillos de ataque, es uno de los más efectivos ya que el atacante puede aprovecharse del desconocimiento de algunas medidas mínimas de seguridad por parte de los usuarios y engañarlas para su propio beneficio.

Un ejemplo de ingeniería social puede ser el uso de un correo electrónico enviado por un atacante a los usuarios solicitando el cambio de sus contraseñas por un texto determinado aduciendo cambios en el sistema; de esta manera se puede obtener acceso al correo del usuario y hacer uso de la información almacenada en él.

b. *SHOULDER SURFING*

Basado en la ingenuidad de los usuarios, el ataque conocido como *Shoulder Surfing* consiste en observar físicamente a los usuarios del sistema con el objetivo de obtener claves de acceso a los recursos. Hay muchos casos en los cuales los usuarios tienen que memorizar algunas claves en sus trabajos y la forma más fácil de hacerlo es anotando en papeles adhesivos dichas claves; pero el error fatal de ellos es pegarlos en lugares visibles en sus puestos de trabajo como por ejemplo en el monitor.

Inclusive el *Shoulder Surfing* puede que no se beneficie solamente de la ingenuidad de los usuarios, sino también de las fallas de las aplicaciones; por ejemplo, existen todavía en el mercado aplicaciones que permiten ver el nombre de usuario y su respectiva contraseña en texto plano sin codificar, de tal manera que si el atacante puede observar físicamente el monitor mientras que el usuario ingresa estos datos, podrá memorizarlos fácilmente y hacer uso de éstos en beneficio propio.

c. *MASQUERADING*

El *Masquerading* está basado en la suplantación de la identificación de los usuarios. El atacante consigue acceso a los recursos digitando por ejemplo el nombre de usuario y contraseña del usuario real; inclusive puede ser presentada una tarjeta de acceso falsificada o robada a un guardia y conseguir con ello el ingreso sitios restringidos.

Una variante de *Masquerading* es el llamado *piggybacking*, el cual consiste en seguir físicamente al usuario hasta el área restringida y esperar que acceda para poder ingresar gracias al acceso legítimo; típicamente el atacante se disfraza de empleado de mantenimiento y carga un equipo extremadamente pesado, espera a que llegue el usuario autorizado y le permita el ingreso al área restringida por delante del guardia de seguridad.

CAPÍTULO

EL PROTOCOLO SSL V3.0

CAPÍTULO DOS

EL PROTOCOLO SSL V3.0

2.1 PROPÓSITOS Y VENTAJAS

El protocolo SSL (*Secure Sockets Layer*) creado por Netscape se ha convertido en el estándar para transmisión de datos de manera segura en el Internet.

Como se esquematiza en la figura 2.1, desde el punto de vista del modelo de referencia OSI, SSL corre en la capa de transporte y brinda sus servicios a la capa superior; si se compara con la arquitectura TCP/IP, SSL corre sobre la capa Internet en la capa de Transporte pero también brinda sus servicios a los protocolos tales como http, ftp, telnet, entre otros.

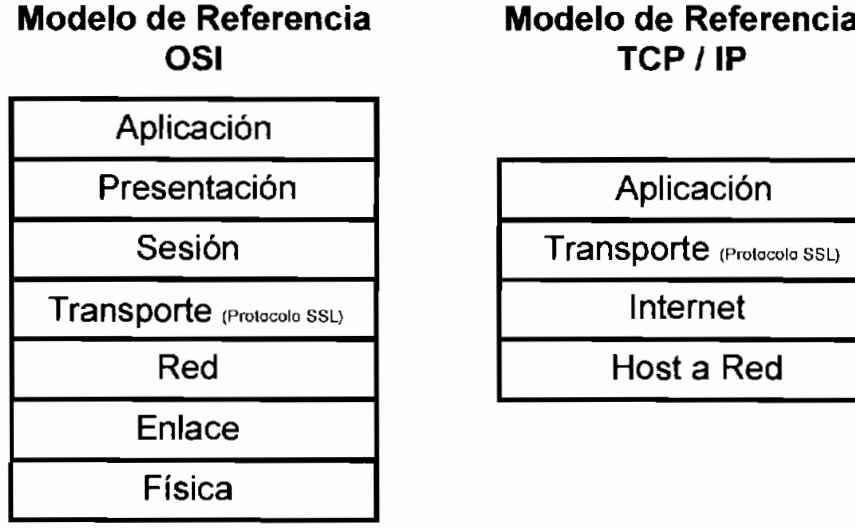


FIGURA 2.1 SSL Comparación OSI vs. TCP/IP

SSL permite autenticar la identidad de un servidor. Los clientes pueden usar técnicas de encripción de llaves públicas para la verificación de la validez del certificado digital que se presenta. Esta autenticación es de gran utilidad

cuando los clientes envían información privada tal como el número de su tarjeta de crédito para autorizar una transacción financiera.

SSL también permite a un servidor confirmar la identidad de un usuario. Haciendo uso de las mismas técnicas utilizadas en la autentificación de un servidor, un servidor SSL puede verificar que el certificado de un cliente es válido y que fue expedido por una autoridad confiable. Este tipo de autentificación puede ser importante para un servidor bancario si va a transmitir información financiera confidencial a un cliente.

SSL ha aportado de gran manera a consolidar las transacciones seguras en Internet; provee importantes características tales como la Autenticación, la Privacidad, la Integridad de los datos y el No Repudio.

En instituciones financieras, la seguridad y privacidad son aspectos críticos con el propósito de proteger la información de los clientes. De esta manera, las soluciones seguras tales como SSL permiten a las instituciones financieras aumentar la efectividad del negocio y la seguridad de la infraestructura de su red.

En instituciones medicas los registros electrónicos de los pacientes deben ser celosamente asegurados. Haciendo uso del protocolo SSL se implementa una infraestructura segura de tal manera de cumplir con las demandas de privacidad en los historiales médicos y que esta información no pueda ser obtenida por personas no autorizadas.

Otro de los usos comunes de SSL esta enfocado en la protección de números de tarjetas de crédito o débito en compras por Internet. Pero como regularmente no se exige el uso del Certificado de Cliente, cualquier persona que obtenga el número de la tarjeta y unos pocos datos personales de los clientes puede realizar compras en su nombre. Esto conlleva el tener que prestar

mucha atención a los resguardos de las operaciones en cajeros automáticos, a desconfiar cuando un empleado de un establecimiento toma la tarjeta momentáneamente para cobrar el importe de la compra, entre otros.

2.1.1 AUTENTICACIÓN EN SSL

En el proceso de autenticación, la mayor parte de los servidores presentan un certificado digital lo cual brinda seguridad al cliente de acceder al servidor adecuado. Por el contrario, muy pocas implementaciones requieren una autenticación en el otro sentido, es decir, no requieren que el cliente presente su certificado digital.

Por ejemplo, una de las aplicaciones más difundidas en Internet con SSL se da en entornos bancarios; cuando un cliente desea consultar su saldo de la cuenta, hacer una transferencia bancaria o hacer uso de algún servicio disponible, el servidor solamente pide el nombre de usuario y la contraseña, proceso que trae consigo problemas inherentes al momento de gestionar las claves de acceso tales como: cambiarlas cada cierto tiempo, mantenerlas protegidas, escoger el tamaño y la combinación adecuada, etc.

Hacer compras por Internet es otra de las aplicaciones más comunes que usa SSL, de esta manera cualquier persona que consiga acceso a un número de tarjeta de crédito y unos pocos datos personales podría hacer compras en nombre del titular verdadero de la tarjeta.

2.1.2 PRIVACIDAD EN SSL

Mientras SSL proporciona la seguridad que el contenido de los datos no podrá ser visualizado aún cuando los datos sean interceptados por entes ajenos al servidor destino, no puede asegurar que al finalizar la transacción el contenido de los datos del usuario sean liberados ni tampoco asegura que no serán usados nuevamente.

De esta manera también depende que la entidad que almacene los

datos que fueron necesarios para la transacción lo haga de una manera segura, y que los atacantes no puedan tener acceso a los mismos.

Gracias a las técnicas de encripción asimétrica que usa SSL, se puede asegurar que la privacidad de los datos sea una de las características más fuertes de este protocolo. Con el uso de estas técnicas se logra que una comunicación sea verdaderamente privada, ya que tanto emisor como receptor haciendo uso de la llave pública de su par, pueden encriptar los datos a transmitir, de tal manera que solo el destinatario pueda desencriptarlos.

2.1.3 INTEGRIDAD DE DATOS EN SSL

La integridad de los datos es otro de los aspectos importante que posee el protocolo SSL, ya que haciendo uso de técnicas tales como los MAC's *Message Authentication Codes*, se puede verificar si los datos han sido o no atacados.

En este caso algún tipo de alteración a los datos se entendería como un ataque; como se explica en el literal "a" de la sección 1.2.3 del capítulo 1, los MAC's son secuencias generadas por funciones *hash* que toman como entrada los datos a asegurar y una llave la cual generalmente es la llave de sesión.

Los datos en texto plano viajan juntamente con el MAC generado en el emisor; de esta manera como la llave de sesión es conocida únicamente por el emisor y el receptor, este último puede realizar un cálculo local del MAC, entonces si el MAC generado localmente es igual al recibido, se puede concluir que los datos están íntegros.

2.1.4 EL NO REPUDIO DE LOS DATOS

Una de las fallas de SSL es que no hay por defecto establecido ningún método para dejar constancia de cuándo se ha realizado una operación, cuál ha sido y quiénes han intervenido en ella. SSL no proporciona formas de emitir recibos válidos que identifiquen una transacción.

Por ejemplo, si una persona hace una compra por Internet, cuando reciba lo que ha solicitado y si no es de su agrado, se lo puede devolver aduciendo que no ha sido él quien ha hecho la compra y que su número de tarjeta de crédito ha sido robado. Por su parte el servidor solo tiene el registro que el número de la tarjeta de crédito ingresado ha sido correcto, no puede verificar que la persona quien suministró dicho número haya sido el titular de la tarjeta.

2.2 CARACTERÍSTICAS PRINCIPALES

El protocolo SSL es el encargado de particionar los flujos de datos que llegan de las capas superiores para luego comprimirlos, añadir un código MAC al flujo comprimido y finalmente encriptarlo previo a la transmisión del mismo (figura 2.2).

Por su parte en la recepción primero se desencripta y descomprime el flujo para luego verificar su integridad mediante la comparación del MAC llegado con el MAC generado en el receptor (figura 2.2).

Dos sub-protocolos forman parte de SSL: El Protocolo de Registro (*Record Protocol*) y el Protocolo de *Handshake*. El Protocolo de Registro define el formato a utilizar para transmitir información; mientras que, el protocolo de *Handshake* involucra al Protocolo de Registro para establecer la conexión inicial y efectuar acciones tales como:

1. Autentificar el servidor al cliente.
2. Permitir que el cliente y el servidor seleccionen un algoritmo de codificación.
3. Opcionalmente se puede identificar el cliente al servidor.
4. Utilizar codificación de llave pública para generar las claves secretas.
5. Establecer una conexión SSL.

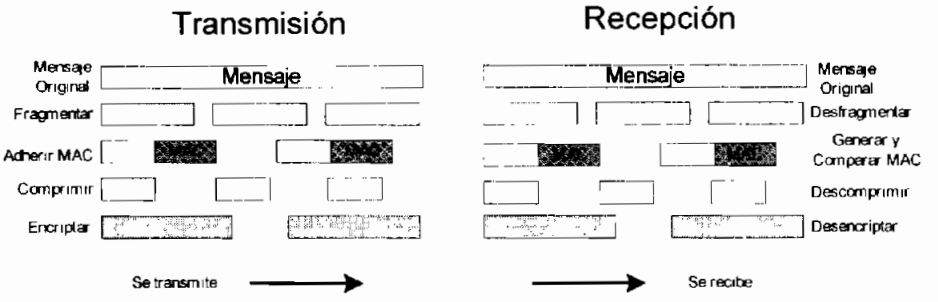


FIGURA 2.2 SSL Transmisión / Recepción

Adicionalmente existe el protocolo *Change Cipher Spec* que sirve para coordinar la transición de estados tal como se verá posteriormente, y el protocolo de Alerta (*Alert Protocol*) el mismo que sirve para notificar las posibles irregularidades que se generen durante los procesos de negociación y transmisión segura de los datos (figura 2.3).

En esta sección se dará una apreciación global de cómo se negocian los parámetros de una sesión entre cliente y servidor que usan el protocolo SSL para su comunicación.

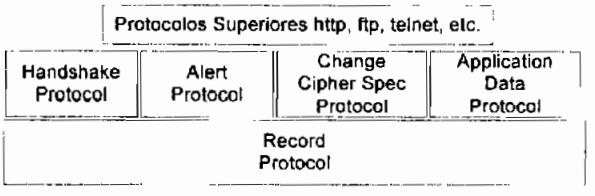


FIGURA 2.3 Estructura de SSL

La compatibilidad hacia atrás en SSL v3.0 está garantizada, los clientes que soportan SSL v3.0 deben enviar los mensajes client hello en formato de SSL v2.0. Por otra parte los servidores implementados con SSL v3.0 aceptan los mensajes client hello enviados en formato SSLv2.0. La única diferencia con la especificación de SSL v2.0 es la capacidad de especificar la versión con un valor de tres (3) y el soporte para una variedad extra de conjunto de algoritmos especificados en el CipherSpec.

Las siguientes especificaciones son importadas en SSLv3.0 desde SSLv2.0:

- V2CipherSpec SSL_RC4_128_WITH_MD5
- V2CipherSpec SSL_RC4_128_EXPORT40_WITH_MD5
- V2CipherSpec SSL_RC2_CBC_128_CBC_WITH_MD5
- V2CipherSpec SSL_RC2_CBC_128_CBC_EXPORT40_WITH_MD5
- V2CipherSpec SSL_IDEA_128_CBC_WITH_MD5
- V2CipherSpec SSL_DES_64_CBC_WITH_MD5
- V2CipherSpec SSL_DES_192_EDE3_CBC_WITH_MD5

2.2.1 ESTADOS DE SESIÓN Y ESTADOS DE CONEXIÓN

El protocolo SSL es el encargado de gestionar la interacción del cliente con el servidor. Bajo este enfoque nacen dos conceptos que son las sesiones y conexiones.

Tal como se esquematiza en la figura 2.4, una sesión de SSL puede tener múltiples conexiones. Entiéndase como sesión al interfaz de comunicación entre el cliente y el servidor. Por ello el cliente puede tener varias interfaces de comunicación con el servidor aún bajo un único identificador de la sesión.

Tanto en el lado del cliente como del servidor se tienen dos estados principales que son: Operativo y Pendiente, y a su vez cada uno de estos estados tiene otros dos estados que son: Lectura y Escritura.

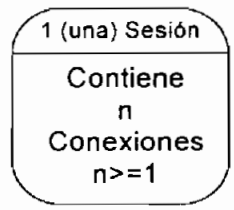


FIGURA 2.4 Sesiones y Conexiones SSL

Cuando el cliente o el servidor recibe un mensaje *Change Cipher Spec* se copia lo existente del *Estado de Lectura Pendiente* en el *Estado de Lectura Actual*. Mientras que, cuando cliente o servidor envian un mensaje *Change Cipher Spec* se copia lo existente del *Estado de Escritura Pendiente* en el *Estado de Escritura Actual* (figura 2.5).

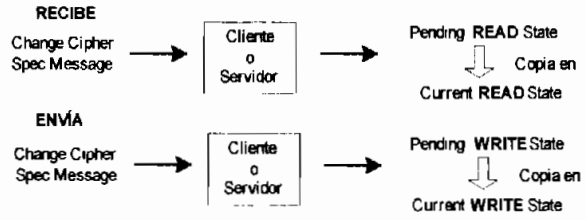


FIGURA 2.5 Estados de Sesión y Estados de Conexión

a. ESTADO DE LA SESIÓN

En SSL V3.0 el estado de la sesión tiene varios elementos detallados a continuación.

1. **Identificador de la sesión**. Secuencia arbitraria de bytes escogida por el servidor para identificar la sesión.
2. **Certificado digital X509 v3 del otro participante en la sesión (opcional).** En el lado del servidor según las especificaciones puede que no sea necesario almacenar el certificado digital del cliente.
3. **Algoritmo de compresión**. Especifica el algoritmo de compresión que usará la sesión previa a la encripción de los datos.
4. **Especificaciones *Cipher Spec***. Especifica los Algoritmos de Encripción, *Hash* e intercarmbio de llaves que serán usados en la sesión.
5. ***Master Secret Key***. Una secuencia secreta de 48 bytes compartidas entre el cliente y el servidor.
6. **Bandera de inicio de conexiones**. Una bandera que indica si la sesión puede soportar múltiples conexiones.

b. ESTADO DE LA CONEXIÓN

La sesión de SSL tiene como mínimo una conexión pero puede soportar varias conexiones dependiendo del parámetro de bandera de inicio de conexiones especificado en el estado de la sesión.

Todas la conexiones existentes tienen el siguiente formato:

1. **Secuencia randómica del cliente y el servidor**. Son secuencias de bytes escogidas por el servidor y por el cliente para identificar la conexión.
2. **MAC de escritura del servidor**. El secreto usado por el servidor para las operaciones de escritura de datos que requieran el uso de MAC's.
3. **MAC de escritura del cliente**. El secreto usado por el cliente para las operaciones de escritura de datos que requieran el uso de MAC's.
4. **Clave de escritura del servidor**. La clave usada por el servidor para la encripción de los datos que serán leídos por el cliente.
5. **Clave de escritura del cliente**. La clave usada por el cliente para la encripción de los datos que serán leídos por el servidor.
6. **Vectores de Inicialización**. Para cifrar un bloque de información se utilizan tanto la clave como los resultados del último bloque. Sin embargo, el primer bloque no tiene bloque precedente para utilizarlo como entrada del cifrado. Si ese primer bloque contiene información conocida es más fácil hacer ingeniería inversa con el primer bloque para descubrir la clave.
   Para evitarlo se utiliza lo que se denomina vector de inicialización. El mismo que es otra matriz de bytes del mismo tamaño que la clave. Se utiliza junto con la clave para obtener un cifrado más seguro del primer bloque.
7. **Números de Secuencia**. Son identificadores de los paquetes enviados a su par, son manejados por el cliente y el servidor por separado. Si se envía un mensaje *Change Cipher Spec* la secuencia empieza en cero y no puede exceder de un máximo de $2^{64}-1$.

2.3 PROTOCOLO DE REGISTRO

El protocolo de Registro corre en la llamada capa de Registro del protocolo SSL, en esta capa se procesan los flujos de los datos provenientes desde las capas superiores.

2.3.1 FRAGMENTACIÓN

Cuando la Capa de Registro fragmenta los flujos de datos que llegan con longitudes aleatorias, los bloques resultantes tienen un tamaño máximo de 2^{14} bytes. Estos bloques son conocidos como *SSLPlainText* (figura 2.6).

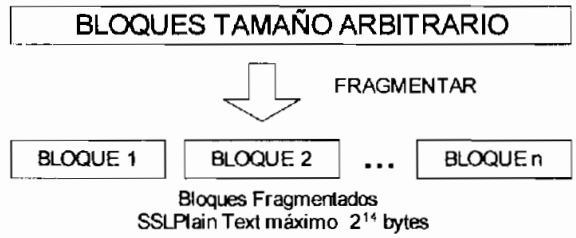


FIGURA 2.6 Fragmentación en la Capa de Registro

El paquete *SSLPlainText* tiene el formato mostrado en la figura 2.7:

TYPE	VERSION	LENGHT	FRAGMENT

FIGURA 2.7 Formato del paquete SSLPlainText

Donde:

Tipo: Es el protocolo de capa superior cuyos datos son procesados.

Versión: Es la versión del protocolo SSL. Para este caso la versión 3.0

Longitud: Especifica la longitud del fragmento. No debe ser mayor a 2^{14} bytes.

Fragmento: Contiene los datos fragmentados del protocolo de capa superior.

2.3.2 REGISTRO DE COMPRESIÓN Y DESCOMPRESIÓN

En este paso se utiliza el algoritmo de compresión especificado como parámetro en el estado de la sesión para cambiar los bloques *SSLPlainText* y

convertirlos en *SSLCompressed*. La característica principal de estos algoritmos es que deben ser tipo *lossless*, es decir que no se pierda información cuando la descompresión se lleve a cabo. Además, si al momento de la descompresión se genera un bloque mayor a 2^{14} bytes se debe generar una alerta para indicar este evento.

El formato del paquete *SSLCompressed* se muestra en la figura 2.8.

FIGURA 2.8 Formato del paquete SSLCompressed

Donde:

Tipo: Es el protocolo de capa superior del cual los datos son procesados.

Versión: Es la versión del protocolo SSL. Para este caso es versión 3.0

Longitud: La longitud máximo del fragmento 2^{14} + 1024

Fragmento: Es la compresión del fragmento del paquete *SSLPlainText*.

2.3.3 REGISTRO DE PROTECCIÓN DE CARGA (*PAYLOAD*) Y *CIPHER SPEC*

En este procedimiento el paquete *SSLCompressed* es transformado al paquete *SSLCipherText*. Se hace uso de las especificaciones *Cipher Spec* del estado de la sesión.

El formato del paquete *SSLCipherText* se muestra en la figura 2.9.

FIGURA 2.9 Formato del paquete SSLCipherText

Donde:

Tipo: Es el protocolo de capa superior del cual los datos son procesados.

Versión: Es la versión del protocolo SSL. Para este caso es versión 3.0

Longitud: La longitud máximo del fragmento 2^{14} + 2048

Fragmento: Es la encripción del fragmento del paquete *SSLCompressed* incluyendo el MAC.

2.4 PROTOCOLO *CHANGE CIPHER SPEC*

El protocolo *Change Cipher Spec* se usa para señalar las transiciones en los parámetros de cifrado. Los mensajes *Change Cipher Spec* son enviados haciendo uso de los parámetros de cifrado especificados en el estado de la sesión.

Cuando un cliente o un servidor desean cambiar los parámetros bajo los cuales los datos están siendo cifrados, se debe iniciar una nueva negociación de los mismos y finalmente tanto cliente como servidor envían un mensaje *Change Cipher Spec* el cual indica que están listos para hacer uso de los nuevos parámetros.

En este caso si el cliente o el servidor envían un mensaje de este tipo el transmisor copia lo existente en el *Estado de Escritura Pendiente* en el *Estado de Escritura Actual*, mientras que el receptor del mensaje copia lo existente en el *Estado de Lectura Pendiente* al *Estado de Lectura Actual*.

Este tipo de mensajes típicamente son enviados al inicio del establecimiento de la sesión luego de la fase de los mensajes *hello* durante la fase de *handshake*.

2.5 PROTOCOLO ALERTA

Este protocolo sirve para gestionar el manejo de errores en las negociaciones de las especificaciones o los errores en la transmisión de los datos.

El protocolo Alerta es independiente del protocolo SSL y es considerado como un protocolo de capa superior. Cuando una conexión termina, su

identificador es invalidado, para prevenir que el establecimiento de una nueva conexión haciendo uso de este identificador sea exitosa.

2.5.1 ALERTAS DE FINALIZACIÓN

Las alertas de finalización son utilizadas tanto por el cliente como por el servidor cuando se desea finalizar una transmisión; el conocimiento de estos mensajes por ambas partes es vital pues necesitan saber que se está finalizando la conexión.

La finalización se hace con el envío de mensajes tipo *close notify*. Luego de enviar este mensaje el transmisor cierra la conexión sin esperar una respuesta del receptor; por su parte el receptor debe enviar un mensaje del mismo tipo pero sin esperar respuesta de la otra parte.

El conocimiento de la finalización de la conexión permite evitar los ataques de truncamiento. En SSL v2 un ataque de este tipo permite parar la sesión sin que el servidor o el cliente lo sepan. Si el atacante sabe algo acerca de la estructura del mensaje y cómo viaja en los paquetes SSL v2, entonces puede usar el ataque de truncamiento para cambiar el significado del mensaje.

2.5.2 ALERTAS DE ERRORES

Tanto cliente como servidor están en la posibilidad de enviar alertas de errores en el momento que consideren pertinente. Como un caso particular cuando se envía un mensaje tipo *fatal alert* ambas partes deben cerrar la conexión inmediatamente.

Existen 11 tipos de alertas definidas las mismas que se detallan a continuación:

unexpected_message: Mensaje inapropiado recibido.
bad_record_mac: Se genera cuando se verifica el MAC de un registro y no es correcto.

decompresion_failure: Los datos descomprimidos tienen una longitud mayor de lo máximo especificado (2^{14} bytes).
handshake_failure: Indica que el transmisor de este mensaje no puede aceptar los parámetros de seguridad que se negocian.
no_certificate: El cliente envía este mensaje si no tiene un certificado digital en respuesta al mensaje *certification request.*
bad_certificate: El certificado enviado no contiene firmas confiables.
unsupported_certificate: Corresponde a un tipo de certificado no soportado.
certificate_revoked: El certificado ha sido revocado por la autoridad certificadora.
certificate_expired: El certificado ha expirado su tiempo útil.
certificate_unknown: Algún otro problema concerniente a certificados digitales, concluye en certificado no aceptable.
ilegal_parameters: Un campo en el *handshake* es inconsistente con otros campos.

2.6 PROTOCOLO *HANDSHAKE*

El Protocolo SSL *Handshake* actúa sobre el Protocolo *Record Layer* pero no recibe los datos directamente desde las capas superiores, sino que está encargado de realizar la primera fase de negociación entre cliente y servidor; es decir, cuando el Protocolo SSL *Handshake* entra en funcionamiento se negocian los parámetros necesarios para establecer una sesión.

2.6.1 FUNCIONAMIENTO GENERAL DE MENSAJES *HELLO* EN EL PROTOCOLO *HANDSHAKE*

Tal como indica la figura 2.10, el cliente empieza enviando un mensaje llamado *client hello.*

Como contestación el servidor envía un mensaje *server hello* o un mensaje *fatal error.*

Con los dos primeros mensajes tanto cliente como servidor básicamente establecen los siguientes parámetros.

- Versión del protocolo SSL que se usará.
- Identificador de la sesión.
- *Cipher Suite*. El *cipher suite* es el conjunto de algoritmos de intercambio de llaves, *hash* y encripción que serán usados en la sesión.
- Método de Compresión.

Los mensajes siguientes del servidor son:

- Envía su certificado digital.
- Envía la petición del certificado del cliente (opcional)
- Envía el mensaje *ServerKeyExchange* (opcional)
- Envía un mensaje *server hello done*, que indica que el servidor finaliza la fase de mensajes *hello*.

Los mensajes siguientes del cliente son:

- Si posee enviará su certificado digital, de otra manera enviará una alerta *no certificate*
- Envía el mensaje *ClientKeyExchange* haciendo uso de los algoritmos negociados con los mensajes *hello*.
- Envía el mensaje *CertificateVerify* con el cual el servidor verifica que el cliente conoce la llave privada correspondiente a la llave pública enviada en el certificado digital.
- Envía el mensaje *Change Cipher Spec* indicando que puede iniciar la transferencia de datos con los parámetros negociados.
- Envía un mensaje *finished* haciendo uso de los nuevos algoritmos, llaves y secretos.

Por su parte el servidor envía su propio mensaje *Change Cipher Spec* y transfiere el contenido del Estado Pendiente *Cipher Spec* en el Estado Actual *Cipher Spec*, finalmente envía un mensaje *finished* haciendo uso de los nuevos

algoritmos, llaves y secretos.

Ahora la fase de *Handshake* ha finalizado y tanto cliente como servidor pueden empezar a intercambiar los datos de la capa aplicación.

Por otra parte, cuando el cliente desea realizar otra conexión con el servidor haciendo uso del identificador de la sesión actual se intercambian los mensajes de la siguiente manera (figura 2.11):

- El cliente envía un mensaje *hello* con el Identificador de la sesión.
- El servidor revisa si el Identificador enviado por el cliente está dentro de su base de datos de sesión, si lo encuentra envía un mensaje *hello* con el mismo Identificador. Si no lo encuentra se realiza todo el proceso de *Handshake* nuevamente.
- El cliente al recibir el mensaje *hello* del servidor, envía un mensaje *Change Cipher Spec.*
- El servidor envía al cliente su propio mensaje *Change Cipher Spec.*

Ahora tanto cliente como servidor pueden comenzar a intercambiar los datos de la capa aplicación.

2.6.2 EL PROTOCOLO *HANDSHAKE* Y EL ESTABLECIMIENTO DE LA SESIÓN

Como se menciona anteriormente, cuando los procedimientos concernientes al protocolo *Handshake* se llevan a cabo se negocian los atributos de seguridad de la sesión que se está estableciendo.

El Protocolo *Handshake* provee mensajes a la Capa de Registro para que sean encapsulados en bloques tipo *SSLPlainText* los cuales serán procesados con las directivas especificadas por la sesión.

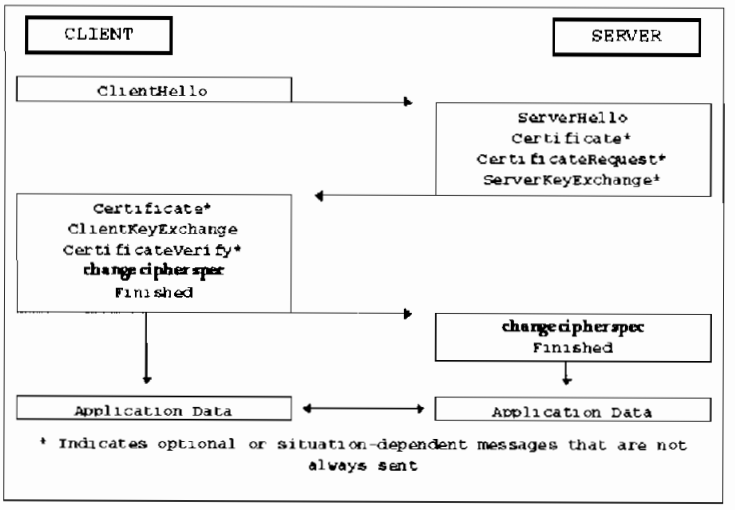


FIGURA 2.10 Intercambio de Mensajes Hello en la Fase Handshake [10]

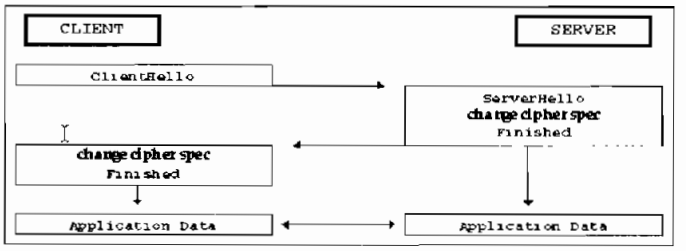


FIGURA 2.11 Establecimiento de una conexión adicional SSL [10]

a. MENSAJES *HELLO*

La fase de mensajes *hello* es usada para la negociación de los parámetros de seguridad tales como encripción, algoritmo *hash*, y algoritmo de compresión los cuales serán utilizados en la sesión. Inicialmente mientras el proceso de negociación se lleva a cabo estos parámetros son nulos.

a.1 *HELLO REQUEST*

El servidor puede enviar este tipo de mensajes a cualquier momento, sin embargo, el cliente puede ignorarlos si el procedimiento de *Handshake* está siendo llevado a cabo. Ésta es una notificación para que el cliente inicie un

proceso de negociación nuevamente enviando un mensaje *client hello* cuando estime pertinente. El servidor no debe enviar nuevamente mensajes *hello request* hasta que el proceso de *Handshake* haya finalizado.

a.2 *CLIENT HELLO*

Es el primer mensaje que se envía cuando se desea establecer una sesión. Enviado por el cliente también cuando se desea iniciar un nuevo proceso de negociación.

En el mensaje *hello* enviado por el cliente se incluyen las combinaciones de algoritmos soportados por el cliente, en orden de preferencia. De esta manera el servidor escoge una de las combinaciones enviadas, pero si el servidor no acepta ninguna de las combinaciones enviará un mensaje *handshake failure* y cerrará la conexión.

Luego que el cliente envía un mensaje *client hello* espera por un mensaje *server hello*, si recibe cualquier otro mensaje es considerado un *error fatal*.

La estructura de un mensaje *client hello* se explica en la figura 2.12:

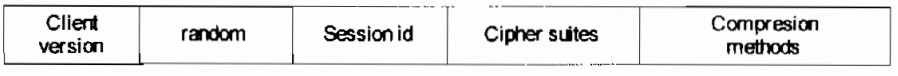


FIGURA 2.12 Paquete Client Hello

Donde:

client_version: Es la versión del protocolo SSL que desea usar el cliente, para este caso deberá ser SSL V3.0

random: Es una estructura cuyo contenido se especificará más adelante.

session_id: El identificador que el cliente desea se use para la sesión, este identificador puede ser uno nuevo generado por el cliente o el identificador de la sesión que se desea modificar. Si este campo está vacío indica que el cliente desea renegociar los parámetros de la sesión.

cipher_suites: Una lista en orden de preferencia de los algortimos soportados por el cliente. Si el campo *session_id* no está vacío, en este campo se debe incluir las especificaciones cipher_suite que tiene actualmente la sesión.

compresion_methods: Una lista en orden de preferencia de las métodos de compresión soportados por el cliente. Si el campo *session_id* no está vacío, en este campo se debe incluir las especificaciones *compresion_methods* que tiene actualmente la sesión.

En la figura 2.13, se describe la estructura *random* de un mensaje *client hello*:

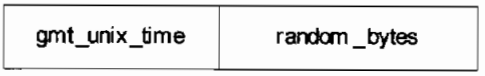


FIGURA 2.13 Estructura Random

Donde:

gmt_unix_time: Es un formato de tiempo y fecha.

random_bytes: Secuencia de 28 bytes provenientes de un generador de números aleatorios.

a.3 *SERVER HELLO*

Como se menciona anteriormente, los mensajes *server hello* son enviados en respuesta de los mensajes *client hello*.

La estructura de un mensaje *server hello* se explica a continuación:

FIGURA 2.14 Estructura del mensaje Server Hello

Donde:

server_version: Contiene la menor versión soportada por el cliente y la mayor versión soportada por el servidor, para este caso será SSL V3.0.

random: contiene la misma estructura *random* explicada anteriormente pero es

independiente de la estructura *random* del cliente.

session_id: Cuando el servidor recibe el mensaje *client hello* analiza el *session_id* que recibe y lo compara en su cache de *session_id*'s, si lo encuentra envía un mensaje *server hello* con el mismo *session_id;* luego las partes deben enviar mensajes *finished* para empezar a transmitir los datos de la aplicación. De otra manera el servidor envía un *session_id* diferente.

cipher_suite: Contiene la combinación de algoritmos seleccionado por el servidor de la lista de combinaciones posibles entregadas por el cliente.

compresion_method: Contiene el algoritmo seleccionado por el servidor de la lista de algoritmos posibles para la compresión entregadas por el cliente.

b. *SERVER CERTIFICATE*

Luego del mensaje *server hello,* el servidor envía su certificado digital si es necesaria su autenticación. Generalmente se envía el certificado digital X.509.V3.

También el certificado digital del servidor puede ser enviado en respuesta a un mensaje *server certificate request* generado por el cliente.

c. *SERVER KEY EXCHANGE MESSAGE*

Este mensaje es enviado por el servidor, si éste no posee certificado digital o posee uno que sirve únicamente para firmar tales como Certificados DSS (los certificado DSS sirven únicamente para firmar, mientras que los certificados RSA sirven para firmar y encriptar).

d. *CERTIFICATE REQUEST*

Un servidor no-anónimo puede solicitar el certificado digital de un cliente según las especificaciones dadas en el *Cipher Spec* de la sesión.

Si el servidor quien solicita el certificado digital del cliente es anónimo se generará una alerta *failure handshake*.

e. SERVER HELLO DONE

Este mensaje es enviado por el servidor para indicar la finalización de intercambio de mensajes *hello*. Luego de esto el servidor espera respuesta del cliente.

f. ***CLIENT CERTIFICATE***

Es el primer mensaje que el cliente puede enviar luego de recibir el mensaje *server hello done*. El cliente envía este mensaje solo si tiene un certificado digital, de lo contrario envía la alerta *no certificate*. Por su parte el servidor puede enviar la alerta *fatal handshake failure,* si es necesario el certificado digital, lo que ocasionaría la finalización del proceso de *handshake*.

Los certificados digitales que envia el cliente siguen las especificaciones X.509.V3.0 como se indicó anteriormente o una versión modificada de los X.509 para el caso de usar Fortezza.

g. ***CLIENT KEY EXCHANGE MESSAGE***

Esta opción depende del algoritmo de llave pública que ha sido seleccionado.

Los algoritmos más comunes usados para la implementación del protocolo son:

- RSA Rivest Shamir Adelman
- Algoritmo Diffie Hellman
- Algoritmo Fortezza

Usando todos los datos generados en el *handshake* hasta ahora, el cliente (dependiendo de las opciones de *cipher suite* que estén siendo usados) crea el *premaster secret key* para esta sesión, lo encripta con la clave pública del *server* (la cual se obtuvo de su certificado), y envia el *premaster secret key* encriptado hacia el servidor.

h. ***CERTIFICATE VERIFY***

Este mensaje es usado para pedir explícitamente que se verifique el certificado enviado por el cliente.

Si el servidor requirió la autentificación del cliente, el cliente entonces deberá autentificarse con el servidor verificando que conoce la clave privada correspondiente al certificado digital que se envió. Para ello, el cliente debe firmar (usando su llave privada) el *premaster secret key* encriptado con la llave pública del servidor.

i. ***FINISHED***

Como se indica en la figura 2.10; este mensaje es enviado luego del mensaje *change cipher spec* para verificar que tanto el intercambio de llaves como la autenticación de las partes se ha llevado a cabo satisfactoriamente. Los mensajes *finished* son los primeros mensajes que están protegidos con los parámetros negociados durante la fase de *handshake*.

2.7 PROTOCOLO *APPLICATION DATA*

Los mensajes generados por el protocolo *Application Data* son transportados por la capa de registro los cuales son fragmentados, comprimidos y encriptados según dictaminen las directivas actuales de la sesión.

2.8 ALGORITMOS CRIPTOGRÁFICOS

Los algoritmos criptográficos son negociados en la fase de *handshake* cuando el cliente y servidor intercambian sus mensajes *hello*.

Las especificaciones de cada cliente en cuanto a los algoritmos que

CAPÌTULO TRES

DISEÑO E IMPLEMENTACIÓN DE LOS SERVIDORES *WEB Y MAIL*

3.1 DISEÑO E IMPLEMENTACIÓN DEL SERVIDOR *WEB* APACHE 2.0 EN LINUX

3.1.1 ESTRUCTURA DEL SERVIDOR APACHE 2.0 EN LINUX

A continuación se describirá la estructura organizacional de *Apache Web Server 2.0*. El servidor Apache desde sus primeras versiones, está organizado por módulos, lo cual provee la ventaja de poder añadir o quitar funcionalidades extras, según sea el caso de la aplicación.

De esta manera los módulos del servidor Apache se pueden enunciar de la siguiente manera:

- **Módulos Base:** En el cual se encuentran las funcionalidades básicas del servidor Apache.
- **Módulos Multiproceso:** Gracias a estos módulos se puede conectar con los puertos de la máquina, aceptar las peticiones, y generar los procesos hijo que se encargan de servirlas.
- **Módulos Adicionales:** En esta categoría están involucrados todos los demás módulos que provean alguna funcionalidad al servidor Apache.

Lo más básico en relación a funcionalidades, se encuentra en los módulos bases, siendo necesario luego completar su potencialidad con los módulos multiproceso los cuales manejan las peticiones de los clientes. En los módulos multiproceso se encuentran variantes para cada sistema operativo sobre el cual se ejecuta *Apache Web Server* con el objetivo de optimizar el

rendimiento y la rapidez del código. Finalmente, los módulos adicionales proveen las funcionalidades adicionales para Apache las mismas que se pueden cargar de manera individual.

Ya que en *Linux* todo es manejado mediante archivos, el servidor *Apache* no es una excepción, de esta manera los archivos más importantes para la configuración del mismo son:

- El archivo *httpd.conf*, en el que se configuran las funcionalidades del servidor.
- El archivo *srm.conf,* donde se configura la raíz del árbol de documentos.
- El archivo *access.conf,* en el que se configuran las políticas de acceso a las carpetas del sitio web.

Los archivos mencionados se ejecutan en el orden descrito. En las versiones anteriores de Apache era necesario modificarlos individualmente para configurar el servidor Apache. En la actualidad con la versión 2.0 es recomendable únicamente la modificación del archivo *httpd.conf* para configurar el servidor, mientras que se deshabilita la lectura de los otros dos archivos hacia los clientes.

Otro archivo importante es *.htaccess* el cual impone las condiciones del acceso a los directorios del servidor a los clientes, también este archivo puede ser modificado según sea necesario.

3.1.2 EL ARCHIVO HTTPD.CONF

Al trabajar sobre el sistema operativo *Linux Red Hat 9.0* con la distribución de *Apache Web Server 2.0,* se encontrará entonces el archivo *httpd.conf* en la ruta */etc/httpd/conf/.*

La estructura del archivo httpd.conf se encuentra dividida en tres

secciones tal como se explica a continuación:

- **Sección 1:** En esta sección se configuran las directivas de las opciones globales del servidor Apache.
- **Sección 2:** En Apache Web Server se tienen dos tipos de servidores, el principal y los *hosts* virtuales. En esta sección se define el cómo se responderá a las peticiones del servidor principal. Además, se configuran las directivas globales de los *hosts* virtuales que se definan posteriormente.
- **Sección 3:** En esta sección se aplican las directivas personalizadas a los *hosts* virtuales configurados.

3.1.3 CARACTERÍSTICAS ESPECIALES DEL SERVIDOR APACHE EN LINUX

El concepto de servidores virtuales debe ser concebido de tal manera de imaginar una serie de servidores *Web* que trabajan independientemente, sin embargo, en la realidad es un solo computador quien alberga a todos los sitios Web.

Con *Apache Web Server* en Linux es posible tener dos tipos de servidores virtuales. Los servidores virtuales basados en nombre y los servidores virtuales basados en dirección IP.

La diferencia básica entre los dos tipos de *hosts* virtuales es que los basados en nombre comparten una misma dirección IP; mientras que, en los *hosts* virtuales basados en IP es necesaria una dirección IP diferente para cada uno de los *hosts*.

Por lo general es aconsejable desde el punto de vista de recursos el uso de *hosts* virtuales basados en nombre, ya que las direcciones IP son un recurso que debe aprovecharse al máximo. Sin embargo hay algunas ocasiones en las cuales será justificable el uso de *hosts* virtuales basados en IP, como por

ejemplo:

- La naturaleza del protocolo SSL no recomienda el uso de *hosts* virtuales basados en nombre, ya que en dicha configuración se comparte una sola dirección IP para varios dominios, y por lo tanto no se podría gestionar todas las peticiones a estos dominios en el puerto estándar de SSL.
- Algunas redes tienen implementada una gestión de ancho de banda las cuales no pueden ser usadas si no existen direcciones IP a diferenciar.

3.1.4 CONFIGURACIÓN DEL SERVIDOR APACHE

a. CONFIGURACIÓN BÁSICA

En las líneas siguientes se explicará las directivas que deben ser configuradas o cambiadas tomando como base al archivo *httpd.conf* que se genera al momento de instalar el servidor Web Apache 2.0 desde la distribución de *Linux Red Hat 9.0.*

a.1 ServerAdmin

```
Sintaxis: ServerAdmin <dirección de correo>
```

Por defecto es *root@localhost*, indica la dirección de correo del administrador del servidor, a esta dirección deberán enviarse los errores que se generen a los visitantes del sitio.

Ejemplo del archivo de configuración:

```
ServerAdmin edu_cito@hotmail.com
```

a.2 *DocumentRoot*

```
Sintaxis: DocumentRoot  <directorio>
```

Señala el directorio raíz a partir del cual se buscarán las páginas que puede mostrar el servidor Web. Por defecto es */var/www/html*. Con la sintaxis del siguiente ejemplo se especifica que el directorio *site* tendrá la página de inicio cuyo nombre se especifica en la directiva *DirectoryIndex.*

Ejemplo del archivo de configuración:

```
DocumentRoot "/var/www/html/site"
```

a.3 *DirectoryIndex*

```
Sintaxis: DirectoryIndex <archivos>
```

Con el uso de esta directiva se puede especificar los nombres de las páginas Web las cuales deben estar dentro del directorio raíz especificado en la directiva *DocumentRoot* las mismas que serán buscadas en el orden de preferencia especificado.

Ejemplo del archivo de configuración:

```
DirectoryIndex index.html
```

Luego de configurar estas directivas, el servidor Web Apache puede ser levantado mediante los comandos especificados en la sección 3.1.5. En el directorio *site* debe existir una página Web cuyo nombre debe ser *index.html.* Para acceder a la página inicial del servidor Web es necesario digitar el siguiente URL: http://localhost. Se obtienen el resultado de la figura 3.1:

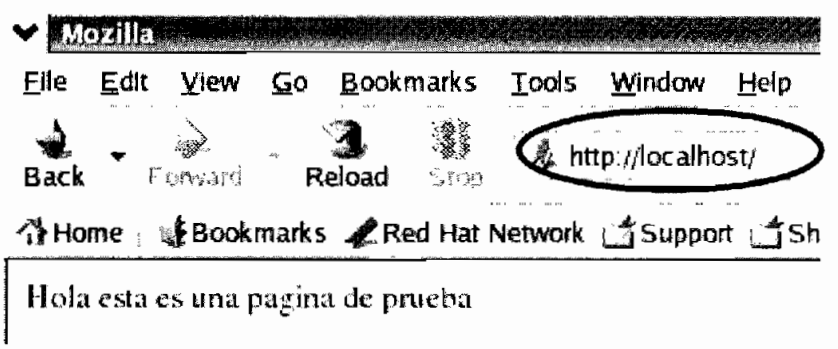


Figura 3.1 Conexión Inicial Apache Web Server

Además se puede verificar que inicialmente el Servidor Apache no presta ninguna seguridad por defecto; las características de Servidor seguro con SSL se configuran posteriormente. Figura 3.2.

b. CONFIGURACIÓN BÁSICA DEL SERVIDOR SEGURO CON AUTENTICACIÒN DEL SERVIDOR

A continuación se explicarán las directivas mínimas de configuración del Servidor Web Apache haciendo uso del *hosting* virtual basado en IP juntamente con las directivas necesarias para proveer autenticación del servidor.

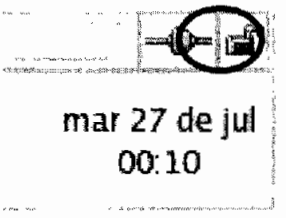


Figura 3.2 Sitio sin Seguridad SSL

b.1 *<VirtualHost> </VirtualHost>*

```
Sintaxis:<VirtualHost dirección:puerto> .. </VirtualHost>
```

Cuando se especifican las directivas que regirán en un determinado *host* virtual éstas deben estar encerradas en este par de directivas. El *host* virtual se puede especificar en términos de su dirección IP, el nombre de dominio, o la cadena *_default_*, de esta manera las características que no se especifiquen serán tomadas desde la configuración del servidor principal.

Ejemplo:

```
<VirtualHost 200.107.6.169:443>
........ directivas
........ directivas
</VirtualHost>
```

b.2 *SSLEngine*

```
Sintaxis: SSLEngine on|off
```

Esta directiva habilita el uso del Protocolo SSL. Usualmente está desactivado para el servidor principal y todos los *hosts* virtuales configurados.

Por lo general esta directiva se coloca dentro de las directivas

<*VirtualHost*> para habilitar el uso de SSL en un servidor específico.

Ejemplo del archivo de configuración:

```
<VirtualHost localhost>
          SSLEngine on
........ directivas
</VirtualHost>
```

b.3 SSLCertificateFile

Sintaxis: SSLCertificateFile *path_certificate*

Mediante el uso de esta directiva se especifica el archivo que contiene el Certificado Digital del Servidor.

Ejemplo del archivo de configuración:

```
SSLCertificateFile /root/opensslLASTEST/certificados/cert/webserver.cer
```

b.4 *SSLCertificateKeyFile*

Sintaxis: SSLCertificateKeyFile *path_key*

La directiva *SSLCertificateKeyFile* apunta al archivo donde se encuentra la clave privada del Servidor.

Ejemplo del archivo de configuración:

```
SSLCertificateKeyFile /root/opensslLASTEST/certificados/webserver.key
```

Las directivas *ServerAdmin*, *DocumentRoot*, y *DirectoryIndex* se configuran igual a lo especificado en la sección anterior.

Para la realización de las pruebas con el servidor, al igual que en la sección anterior se digita la dirección *localhost*, pero ahora con la variante que en lugar de digitar *http://* se deberá escribir *https://*. Figura 3.3.

Igualmente se puede verificar que la conexión es segura mediante la

inspección en la esquina inferior derecha del *Web browser*, región en la cual deberá aparecer un icono de un candado cerrado, lo cual simboliza una conexión segura (ver figura 3.4).

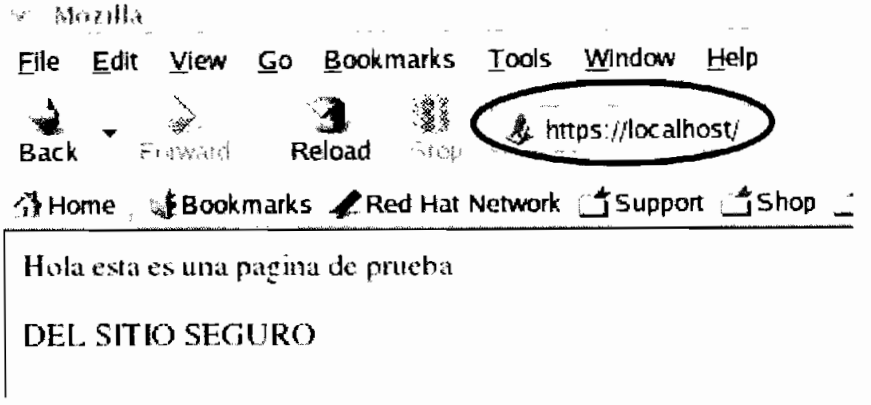


Figura 3.3 Conexión Apache Web Server Seguro

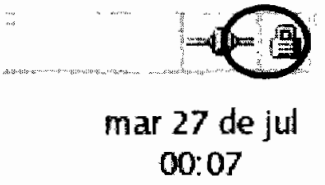


Figura 3.4 Sitio con Seguridad SSL

c. CONFIGURACIÓN BÁSICA DEL SERVIDOR SEGURO CON AUTENTICACIÒN DEL SERVIDOR Y CLIENTE

Para hacer uso de esta configuración es necesario añadir básicamente tres directivas, las mismas que se describen a continuación.

c.1 *SSLVerifyClient*

`Sintaxis: SSLVerifyClient level`

Esta directiva sirve para configurar el nivel de autenticación de los usuarios y tiene cuatro variantes:

- none: No es necesario un certificado digital del cliente.
- optional: El cliente puede presentar un Certificado Digital válido.

CD de una AC.

OpenSSL es un paquete de software gratuito el cual permite entre otras cosas gestionar los CD; se describe a continuación el procedimiento de instalación de este paquete, así como los pasos a seguir para obtener un CD para un cliente.

d.1 DESCARGA E INSTALACIÓN DE *OPENSSL*

Descargar el paquete *OpenSSL* desde www.openssl.org

Al momento de la realización de este proyecto, la última versión disponible de este paquete es *openssl-0.9.7d*

Nota: Todos los comandos enmarcados a continuación deben ejecutarse desde una ventana de *Terminal* del sistema operativo.

1. Acceder a la carpeta de *openssl*

```
cd /root/tesis/openssl-0.9.7d
```

2. Ejecutar los siguientes comandos como parte de la instalación:
Este proceso lleva aproximadamente 20 minutos.

```
./config
make
make test
make install
```

3. En el proceso de instalación se crean los directorios:

```
/usr/local/ssl
/var/ssl
```

4. Copiar los archivos *index.txt* y *serial* presentes en el directorio openssl-0.9.7d/apps/demoCA al directorio /var/ssl

```
openssl-0.9.7d/apps/demoCA en el directorio /var/ssl
cd /root/tesis/openssl-0.9.7d/apps/demoCA
cp index.txt /var/ssl/index.txt
cp serial /var/ssl/serial
```

d.2 OPERACIÓN BÁSICA DE *OPENSSL*

Una vez instalado y configurado correctamente *OpenSSL*, a continuación se describen los pasos que se deben realizar para poder generar un Certificado Digital de una Autoridad Certificadora y con éste posteriormente firmar un Certificado Digital de Cliente.

d.3 CREACIÓN DE UN CERTIFICADO DIGITAL DE AUTORIDAD CERTIFICADORA

1. Haciendo uso de una ventana de terminal de consola de Linux, se debe escribir el siguiente comando para localizarse en el directorio donde se ha instalado *openssl.*

```
cd /usr/local/ssl
```

2. Crear una llave privada para la CA.

```
openssl genrsa –des3 –out ca.key 1024
```

Con este comando se crea una llave privada encriptada con 3DES y una longitud de 1024 bits.

Al momento de la creación de la misma el sistema solicita un *passphrase* (frase clave) el cual es vital para la generación de la llave privada y para eventos posteriores tales como la firma del CD de clientes.

3. Creación del Certificado Digital de Autoridad Certificadora

```
openssl req –config openssl.cnf –new –x509 –days 1001 –key
ca.key –out certs/ca.cer
```

Ejecutando este comando, se inicia el proceso de creación del CD para la AC, luego de esto el sistema pregunta el *passphrase* con el cual la llave privada fue creada y luego se completa la información necesaria para la generación del certificado digital.

4. Conversión del archivo ca.cer al formato ca.p12

```
openssl pkcs12 –export –in certs/ca.cer –inkey ca.key –out certs/ca.p12
```

Con este comando se convierte el CD de la AC del formato *.cer* al formato *.p12*. Los CD que tienen formato *.p12* son generalmente usados por *Netscape*, mientras que los CD con formato *.pfx* se usan en *Internet Explorer*.

Por defecto, Internet Explorer maneja sus CD con formato *.pfx*; por su lado el navegador Netscape los maneja con formato *.p12*. Sin embargo, guardan compatibilidad, es decir los archivo *.pfx* pueden ser importados en Netscape como los *.p12* pueden ser importados por Internet Explorer.

El sistema solicitará el *passphrase* para la transformación de los formatos; además, se pedirá un *password* de exportación el cual servirá posteriormente para la importación del CD en un *web browser*

d.4 FIRMAR UN CERTIFICADO DIGITAL DE CLIENTE

De manera general, para firmar un CD para un cliente, se necesita la llave privada de la AC así como su CD y además, un requerimiento de certificado digital firmado por el cliente.

*Todos los comandos enmarcados a continuación deben ejecutarse desde una ventana de *Terminal* del sistema operativo.

1. Crear una llave privada para "usuario1"

```
cd /etc/httpd/conf
make usuario1.key
```

El primer paso es ubicarse en el directorio /etc/httpd/conf y ejecutar el comando *make* especificando que se desea generar una llave privada (por la extensión *.key* del archivo). Entonces el sistema preguntará un *passphrase* para la generación de la llave privada del "usuario1"

2. Crear un requerimiento de certificado digital firmado

```
make usuario1.csr
```

Con este comando el sistema solicita el *passphrase* y seguidamente se procede a ingresar la información del usuario para generar el requerimiento. Dicho requerimiento también es conocido como CSR por sus siglas en inglés de *Certificate Signing Request.*

3. Copiar los archivos "usuario1.key" y "usuario1.csr"

Por facilidad de la gestión de los archivos se copian al directorio de *Openssl.*

```
cp usuario1.csr /usr/local/ssl/usuario1.csr
cp usuario1.key /usr/local/ssl/usuario1.key
```

4. Localizarse nuevamente en el directorio donde se ha instalado SSL y Firmar el CSR del "usuario1" con la llave privada de la AC

```
cd /usr/local/ssl
openssl ca –policy policy_anything –config /usr/local/openssl.cnf –
cert certs/ca.cer –in usuario1.csr –keyfile ca.key –days 360 –out
certs/usuario1.cer
```

El sistema solicitará el *passphrase* de la llave privada de la AC para firmar el CD del Cliente.

5. Conversión del archivo usuario1.cer al formato usuario1.p12

```
openssl pkcs12 –export –in certs/ usuario1.cer –inkey usuario1.key
–out certs/ usuario1.p12
```

3.1.5 COMANDOS PARA PROCEDIMIENTOS DE RUTINA EN APACHE WEB SERVER 2.0

En el sistema operativo Linux, el programa *httpd* se ejecuta como demonio (*daemon*) y es quien atiende las peticiones de los clientes conforme llegan.

Es necesario tener los privilegios del usuario *root* para poder iniciar el servidor Apache, luego de lo cual el servidor abre los archivos *log* para documentar sus actividades e inicia algunos procesos hijos los mismos que atenderán las peticiones de los clientes.

La forma correcta de llamar al programa *httpd* es mediante el *script apachectl* (figura 3.6), de todas maneras cualquier argumento que tome *apachectl* lo pasará a *httpd*, de tal manera que todas las opciones usadas con *apachectl* pueden ser usadas con *httpd* (figura 3.5).

Según lo explicado anteriormente, los siguientes comandos son equivalentes:

ACCIÓN	COMANDO 1	COMANDO 2
INICIO	service httpd start	Apachectl -k start
PARADA	service httpd stop	Apachectl -k stop
REINICIO	service httpd restart	Apachectl -k restart

Tabla 3.1 Comandos Básicos en Apache Web Server.

Pocas son las veces en las cuales un computador que trabaje con Linux Red Hat debe ser reiniciado; para estos casos si se desea el servidor *Web* Apache arranque juntamente con el computador es necesario que sea configurado en los servicios de arranque tal como se muestra en la figura 3.7.

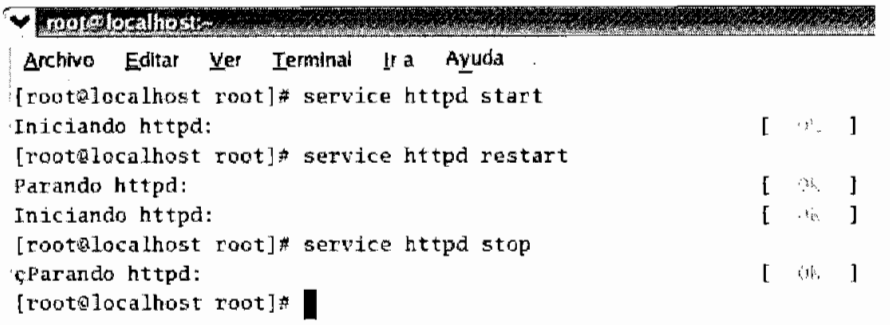


Figura 3.5 Gestión del Servidor, comando service httpd xxx.

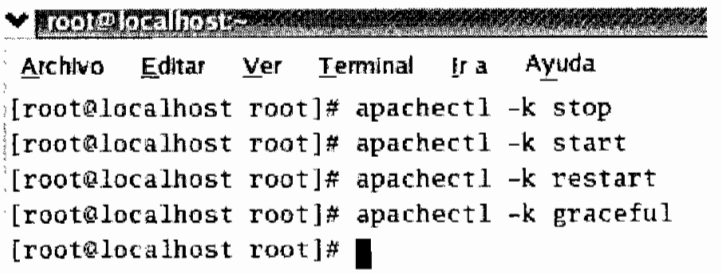


Figura 3.6 Gestión del Servidor, comando apachectl xxx.

3.1.6 PRUEBAS REALIZADAS DE ACCESO AL SITIO WEB SEGURO

En el **Anexo 2** se muestran las pruebas realizadas de acceso al sitio *web*

seguro; como se puede observar, el cliente hace una petición de conexión al servidor seguro, éste por su parte pide la autenticación del cliente, entonces el *Web Browser* muestra al cliente el o los certificado digitales que tenga instalados para presentar al servidor.

3.2 INTRODUCCIÓN AL FUNCIONAMIENTO DEL PROTOCOLO *DOMAIN NAME SERVICE* (DNS)

3.2.1 UTILIDADES DEL PROTOCOLO

Los computadores poseen direcciones IP, las mismas que constan de 32 bits según el estándar IPv4, normalmente se las representa en una notación decimal separada por puntos que es más fácil de recordar antes que una serie de 32 bits.

A pesar de ello, la notación decimal no es muy amigable para los humanos, pues es más fácil tratar de recordar direcciones de computadores en Internet expresadas en nombres antes que expresadas en notación decimal.

El objetivo principal del protocolo DNS es tener una Base de Datos en las cuales se almacene la correspondencia entre direcciones IP y nombres de texto los cuales son fáciles de recordar para los humanos. Para los usuarios es mucho más fácil recordar www.networkingssl.fadlan.com antes que 200.107.6.169.

El protocolo DNS tiene además otra utilidad. Hay veces que al analizar los *logs* de los sistemas ya sea por motivos administrativos, auditoría, o seguridad, es necesario saber a qué organización pertenecen los paquetes que han llegado, de esta manera una consulta oportuna al servidor DNS puede ayudar en las averiguaciones.

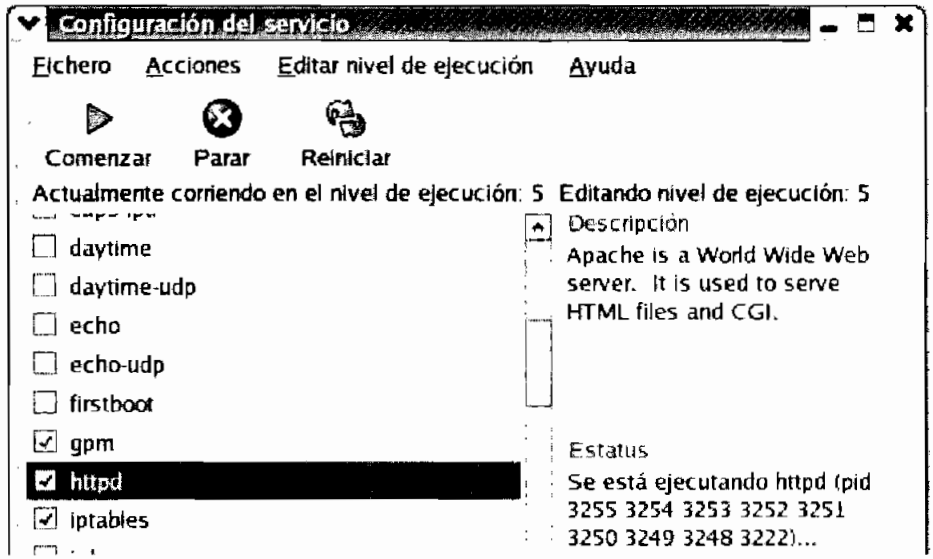


Figura 3.7 Servidor Web como servicio.

3.2.2 ESTRUCTURA JERÁRQUICA

Actualmente en Internet se calcula que existen unos cuantos cientos de millones de máquinas y todas hacen uso de DNS; por lo tanto por razones de escalabilidad y para evitar un único punto de falla la base de datos de DNS está distribuida en todo el Internet.

La Base de Datos de DNS es distribuida y jerárquica; cada uno de los servidores almacena una pequeña porción de toda la Base de Datos. De esta manera un servidor DNS sólo necesita saber las direcciones IP de los servidores de nivel inferior y no necesita ninguna autorización de los servidores de nivel superior para hacer algún cambio.

Como toda estructura jerárquica hay una raíz, en DNS el servidor raíz es representado como un punto (.) y este servidor solo necesita saber las direcciones IP de los servidores de nombres de nivel inferior, los llamados Dominios de Primer Nivel (*Top Level Domain TLD*).

Se tienen definidos alrededor de 200 TLD's los cuales están repartidos entre Dominios Nacionales y Dominios Genéricos. Los primeros identifican a los países de mundo mediante dos iniciales, para el caso de Ecuador el identificador es "ec"; mientras que los dominios genéricos identifican al tipo de organización y pueden ser: .net, .com, .gov, .edu, .mil, .int, y .org.

De acuerdo a esta organización jerarquizada, los servidores DNS pueden delegar la gestión de un espacio de nombres a otros servidores DNS. Por ejemplo, el servidor del dominio ".com" ha delegado a otro servidor la gestión del dominio "google.com".

3.2.3 PROCEDIMIENTO DE CONSULTAS A SERVIDORES DNS

Basándose en la figura 3.8 y suponiendo que un computador dentro del dominio *flits.cs.vu.nl* quiera mantener una comunicación con algún computador dentro del dominio *cs.yale.edu*, el proceso de averiguación de la respectiva dirección IP se describe a continuación.

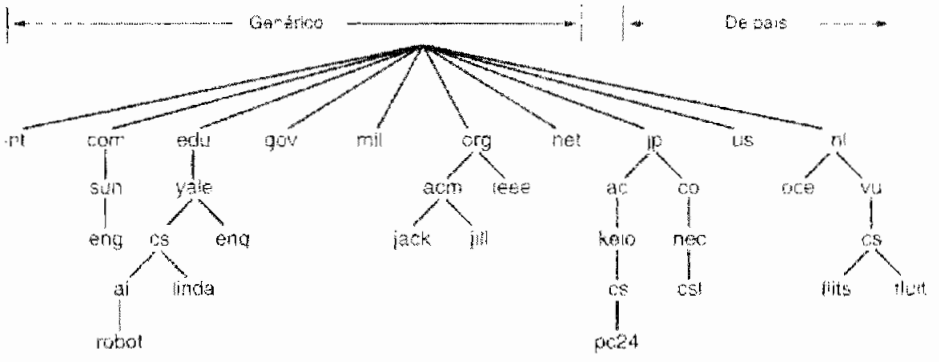


Figura 3.8 Jerarquía de DNS [41]

El computador A dentro del dominio *flits.cs.vu.nl* pregunta a su servidor DNS *cs.vu.nl* la dirección IP del computador que desea ubicar. Si éste lo tiene en caché lo remite, caso contrario puede hacer una consulta al servidor de nivel superior *.edu*.

Es improbable que el servidor del dominio *.edu* sepa la ubicación del servidor del dominio *cs.yale.edu* pero sabe la ubicación del servidor DNS del dominio *.yale.edu*, quien finalmente conoce la dirección IP del servidor *cs.yale.edu*.

La información se remite ahora en orden ascendente hasta llegar al computador A.*flits.cs.vu.nl*, el mismo que ahora ya sabe la dirección IP del computador B.*cs.yale.edu*. Luego de este proceso se envían los paquetes con la dirección IP del destinatario para poder iniciar la negociación del establecimiento de una sesión de comunicaciones.

3.2.4 CÓMO REGISTRAR UNA IP PÚBLICA EN UN SERVIDOR DNS

Existen varias soluciones para obtener que un dominio esté direccionado a una dirección IP deseada; uno de los métodos es la compra del dominio. Sin embargo, para este caso se ha escogido que la redirección del dominio sea gratuita la misma que ha sido registrada en la dirección http://www.fadlan.com.

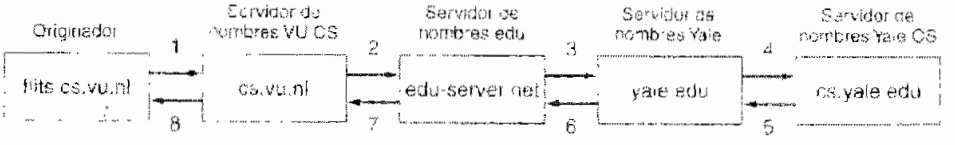


Figura 3.9 Consulta de DNS [41]

Se deben ingresar varios datos informativos tales como:

- Nombre del dominio: networkingsslepn.fadlan.com
- La dirección ip a apuntar: 200.107.6.169
- *email* de contacto: edu_cito@hotmail.com
- contraseña: xxxxxxxx

Luego de registrar el dominio es necesario esperar durante un lapso aproximado de tres días para que la base de datos de DNS se refresque y todos los dominios sepan como alcanzar el dominio nuevo registrado. El resultado es mostrado en la figura 3.10; se hace ping al nombre de dominio registrado y muestra la dirección IP correspondiente.

```
[root@networkingsslepn mailserver]# ping networkingsslepn.fadlan.com
PING networkingsslepn.fadlan.com (200.107.6.169) 56(84) bytes of data.
64 bytes from 200.107.6.169: icmp_seq=1 ttl=64 time=0.087 ms
64 bytes from 200.107.6.169: icmp_seq=2 ttl=64 time=0.082 ms
```

Figura 3.10 Registro de Dominio

3.2.5 CÓMO CONFIGURAR EL ARCHIVO HTTPD.CONF PARA HACER USO DEL SERVIDOR DNS

La única directiva que se debe configurar en el archivo *httpd.conf* para hacer uso de un dominio DNS registrado es la directiva *ServerName*.
Ejemplo del archivo de configuración:

```
ServerName networkingsslepn.fadlan.com
```

Con esta configuración cuando los clientes de Internet que deseen acceder al servidor, harán sus peticiones a nombre del servidor "networkingsslepn.fadlan.com", y se harán las consultas necesarias para poder alcanzar el dominio "fadlan.com" el cual finalmente guiará las peticiones hacia el servidor buscado.

3.3 DISEÑO E IMPLEMENTACIÓN DEL SERVIDOR *KERIO MAIL SERVER* V5.7.6 EN LINUX

3.3.1 INTRODUCCIÓN AL PROTOCOLO POP3

POP3, son las iniciales de Protocolo de Oficina Postal (Post *Office Protocol*). Este protocolo sirve para la entrega de correo electrónico al usuario, la ventaja de su uso es que los correos pueden ser almacenados localmente en los

computadores de los usuarios, ayudando de esta manera a liberar espacio de almacenamiento en el *host* principal donde residen los correos, al mismo tiempo que se evita la llegada de mensajes indicando que su buzón está lleno.

A nivel de protocolos que gestionan la entrega de correo a los usuarios, POP3 es el más extendido a nivel mundial por la facilidad que presta a usuarios que siempre se conectan a revisar su correo desde un mismo computador.

3.3.2 INTRODUCCIÓN AL PROTOCOLO IMAP

IMAP, son las iniciales de Protocolo Interactivo de Acceso a Correo (*Interactive Mail Access Protocol*). Es otro protocolo que sirve para gestionar la entrega de correo a usuarios.

La principal diferencia entre IMAP y POP3, es que IMAP no almacena localmente en el computador el correo de los usuarios, lo cual aparentemente es una desventaja, pero en realidad es muy útil cuando un usuario no tiene un lugar de trabajo fijo, de esta manera el usuario puede usar computadores compartidos.

Originalmente IMAP se diseñó como un protocolo de ayuda para usuarios que poseen varios computadores, por ejemplo, una estación en la oficina, una estación en la casa y un portátil.

Es un protocolo que gestiona con mayor seguridad el correo, pues al no almacenarse localmente en el computador no se corre el riesgo que pueda ser leído por terceras personas.

Sin embargo este esquema no libera al usuario de la responsabilidad de liberar espacio en el buzón del *host* pues se corre el riesgo de llenarlo y dejar de recibir correspondencia.

3.3.3 EL SERVIDOR DE CORREO *KERIO MAIL SERVER* EN LINUX RED HAT 9.0

a. CARACTERÍSTICAS GENERALES

Kerio Mail Server es un servidor de correo seguro que puede servir a múltiples dominios, además trabaja con una gran variedad de clientes de correo electrónico para sistemas operativos tales como: MS Windows, Linux e inclusive en plataformas Mac.

Kerio Mail Server tiene la potencialidad de manejar un antivirus propio en su implementación, integra el soporte del antivirus McAfee; chequea los correos entrantes y salientes, reduciendo de esta manera las posibilidades de propagación de virus.

Tiene incorporado un módulo que elimina correo *spam*, proveyendo de esta manera una solución oportuna para las empresas y evitando el involucramiento en problemas legales y riegos de seguridad asociados con el *spam*.

Esta implementación de servidor *mail* es totalmente escalable. Puede soportar desde 20 usuarios en redes pequeñas hasta 500 clientes concurrentes protegidos con antivirus y con protección de *spam*.

La versión actual de *Kerio Mail Server* con su respectiva documentación puede ser descargada desde el sitio de su distribución: http://www.kerio.com/kms_download.html

b. INSTALACIÓN DEL SERVIDOR MAIL EN LINUX RED HAT 9.0

Luego de descargar *Kerio Mail Server* se deben seguir los siguientes pasos para la instalación:

1. Ejecutar el paquete kerio-mailserver-mcafee-5.7.6-rh7.i386.rpm
2. Ubicarse en el directorio */opt/kerio/mailserver* y ejecutar "*cfgwizard*" (figura

3.11.)

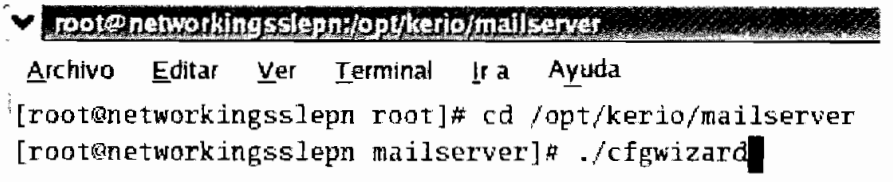


Figura 3.11 Asistente de Configuración

3. Se ejecutará un asistente de configuración

En el **Anexo 3** se muestra los pasos a seguir con el asistente de configuración *Kerio Mail Server*. A manera general, se ingresa el nombre del dominio inicial que se desea administre el servidor mail, luego se ingresa *password* de administración de la consola de configuración de correo; y finalmente se indica el directorio en el cual las cuentas de correo almacenarán su información.

4. Comandos para parar y reiniciar el servidor de correo electrónico.

El servidor de correo electrónico se instala al igual que el servidor web, como un servicio del sistema operativo. Por lo tanto, los comandos *service keriomailserver* con sus variantes *start*, *restart*, o *stop* sirven para gestionar de manera básica al mismo (figura 3.12).

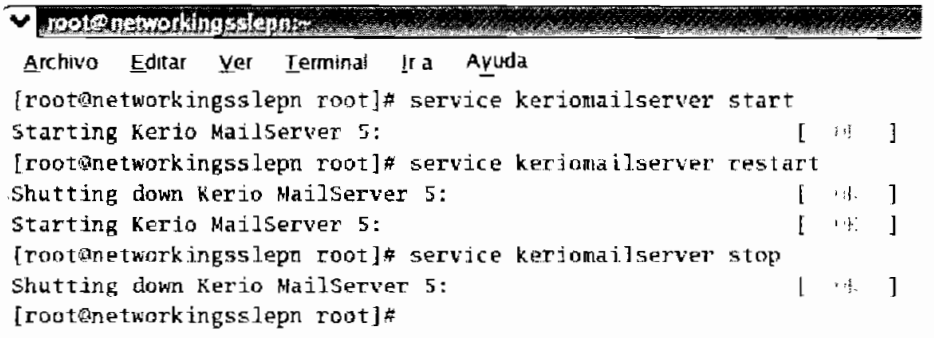


Figura 3.12 Gestión de Kerio Mail Server

5. Luego es necesario instalar la consola de administración ejecutando el paquete kerio-mailserver-admin-5.7.6-rh7.i386.rpm

Para acceder a la consola de administración se debe ejecutar el comando:

kerioadmin desde una ventana de *terminal* bajo cualquier directorio. Luego de lo cual el sistema pedirá la clave de administración configurada anteriormente. Se obtiene como resultado la visualización de la consola de administración. (Figura 3.13)

c. CONFIGURACIONES BÁSICAS

c.1 SERVICIOS

Se configuran los servicios que tendrá el servidor *mail* y en los puertos que éstos han de escuchar (Figura 3.14). Es importante observar que el servidor seguro de correo electrónico por defecto escucha en el puerto 443, lo cual puede ocasionar problemas cuando el servidor *web* escucha peticiones en el mismo puerto; por esta razón se debe cambiar el puerto de escucha ya sea del servidor de correo o del servidor *web.*

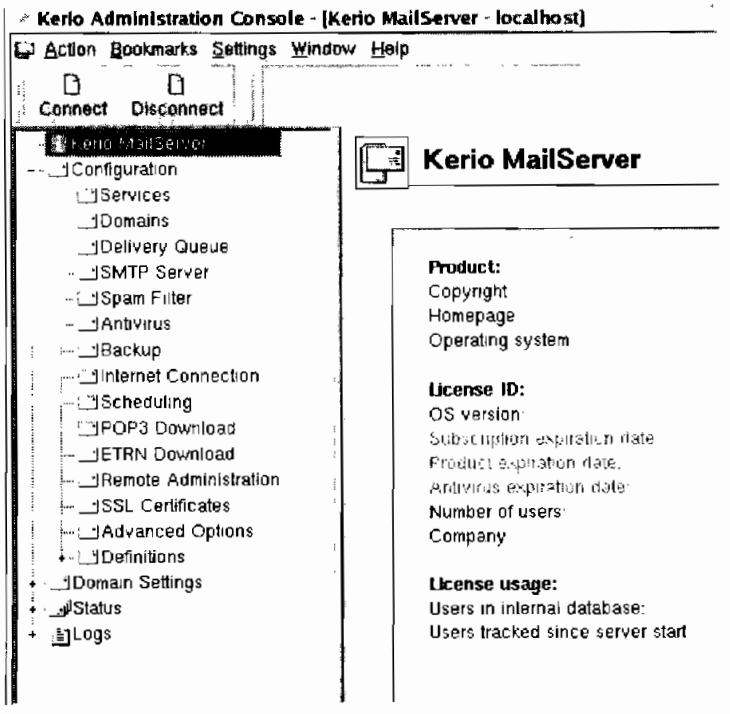


Figura 3.13 Consola de Administración

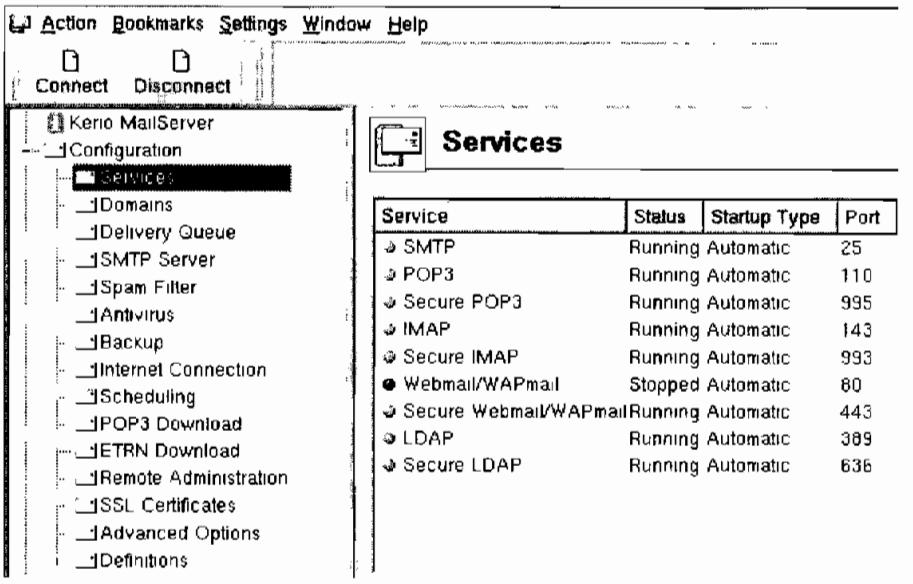


Figura 3.14 Configuración de Servicios

c.2 DOMINIOS

Kerio Mail Server puede ser configurado para servir a varios dominios. El dominio *networkingsslepn.fadlan.com* es el que fue originalmente configurado con el asistente (Figura 3.15).

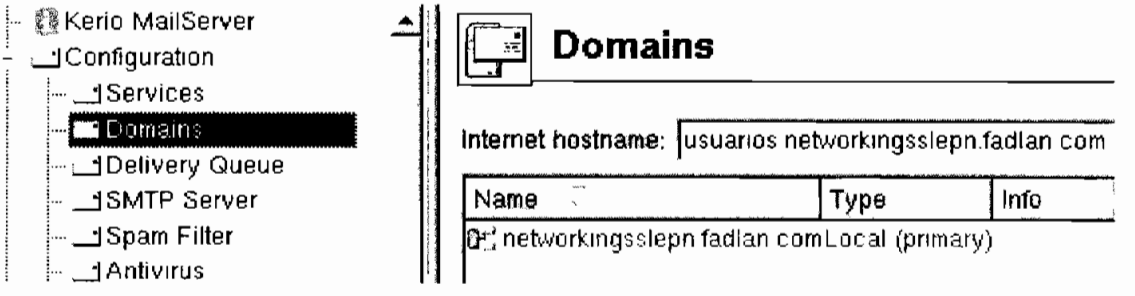


Figura 3.15 Configuración de Dominios

c.3 CERTIFICADOS DIGITALES

Ahora es necesario seleccionar el certificado digital que será usado por el servidor de correo electrónico para proveer una conexión segura (Figura 3.16)

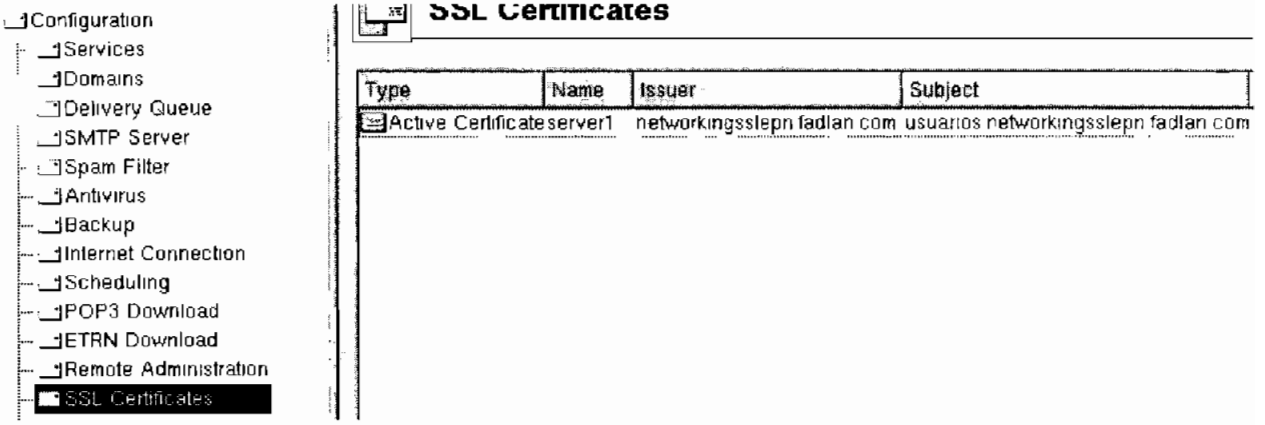


Figura 3.16 Certificados Digitales

c.4 CUENTAS DE USUARIOS

En el Anexo 4 se muestra la configuración de cuentas de usuarios en la consola de *Kerio Mail Server*. En esencia lo que se configura es: nombre de usuario, su *password* inicial, y la cuota de disco asignado en el servidor.

La figura 3.17 muestra el resultado de configurar algunas cuentas de usuarios.

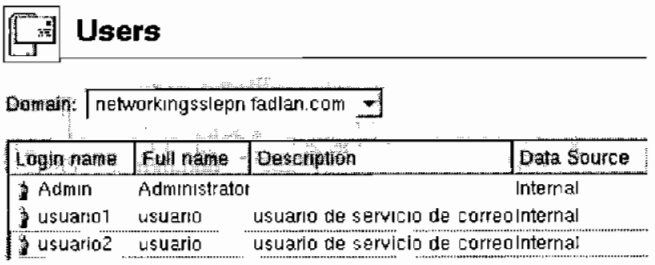


Figura 3.17 Cuentas de Usuarios

d. CONFIGURACIONES AVANZADAS

d.1 ANTIVIRUS

En la figura 3.18 se muestra el antivirus *McAfee* que viene integrado con *Kerio Mail Server.* El antivirus puede ser configurado para actualizarse cada cierto tiempo en este caso cada 6 horas.

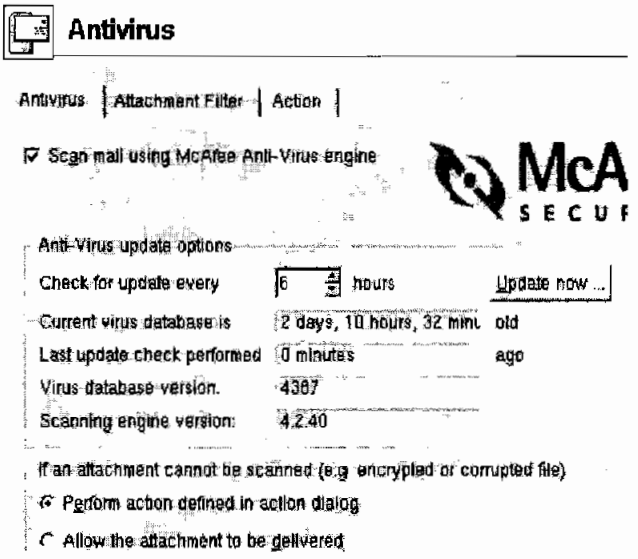


Figura 3.18 Antivirus (1)

Además se configura las extensiones de archivos que el antivirus ha de chequear en busca de virus (Figura 3.19).

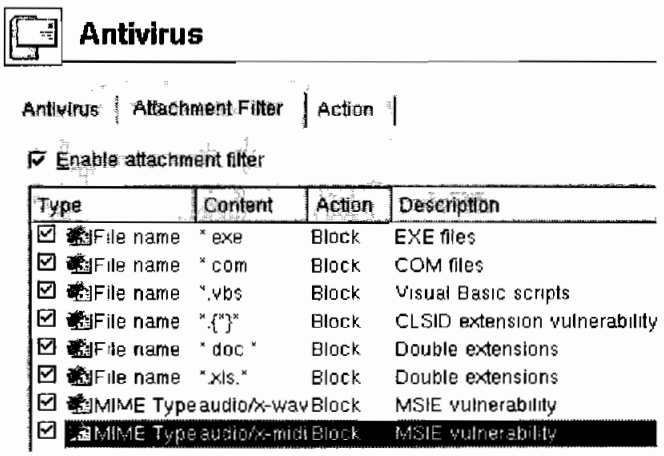


Figura 3.19 Antivirus (2)

d.2 *ANTI SPAM*

El filtro de correo *Spam* ejecuta varias pruebas a los mensajes de correo electrónico para asignar un número de 0 a 10 para indicar la probabilidad de *spam*. La figura 3.20 habilita el servicio *Anti Spam*

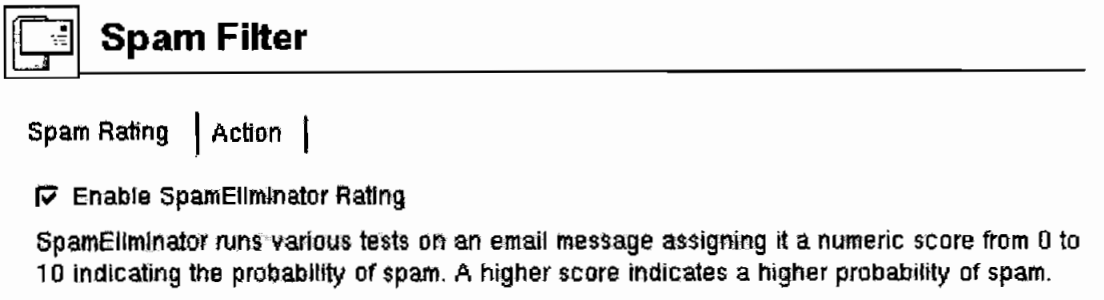


Figura 3.20 Anti Spam

d.3 *BACKUP*

Esta opción (figura 3.21) se configura para efectos de respaldo de información de los usuarios. Se pueden respaldar los mensajes enviados en el mismo dominio (locales), los enviados desde otros dominios (entrantes), los enviados a otros dominios (salientes), y los enviados a otro servidor de correo de *backup*. Los respaldos se almacenan en el directorio *mail_backup* el mismo que se encuentra en la ruta de instalación.

Además se configura un usuario que tendrá acceso a la información respaldada.

e. MONITOREO DEL SERVIDOR *MAIL*

e.1 VISUALIZACIÓN DE TRÁFICO DE CORREO ENTRANTE Y SALIENTE

Se puede visualizar el tráfico (correo entrante y saliente) que maneja el servidor de correo electrónico (Figura 3.22).

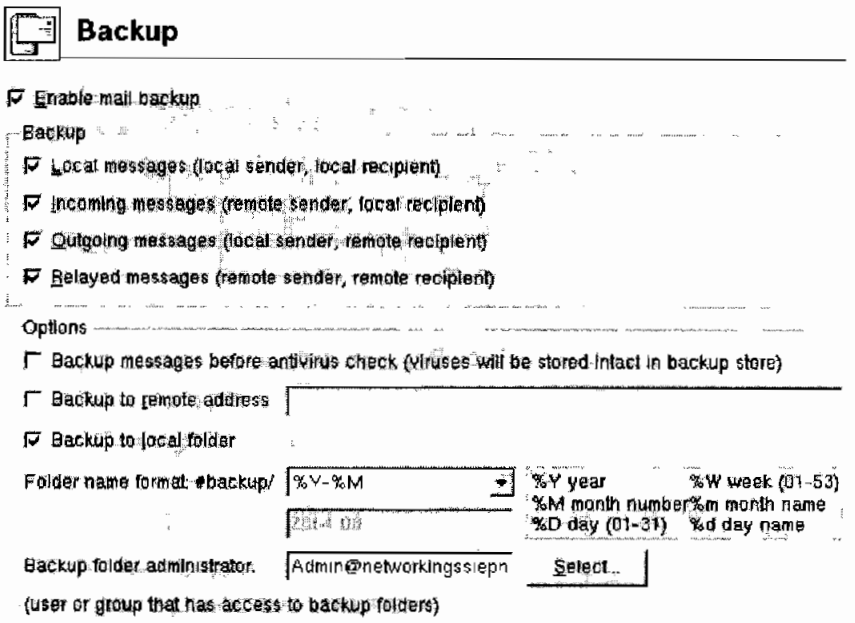


Figura 3.21 Backup

e.2 VISUALIZACIÓN DETALLADA DE TRÁFICO

Para efectos de auditoria a los usuarios, se puede observar el origen y destino de los mensajes de correo electrónico que llegan y salen del servidor (Figura 3.23)

3.3.4 CONFIGURACIÓN DEL CLIENTE DE CORREO

En los siguientes anexos (5, 6 y 7) se muestra el procedimiento de configuración de los *web browsers* para que usen únicamente el protocolo SSL v3.0 para comunicarse con el servidor; además se explica la importación de los certificados digitales tanto de la autoridad de certificación como el del cliente.

Anexo 5: Configuración de *Mozilla Mail & News Groups*

Anexo 6: Configuración de *Netscape Mail & News Groups*

Anexo 7: Configuración de *Microsoft Internet Explorer*

3.3.5 PRUEBAS REALIZADAS CON CADA UNO DE LOS CLIENTES DE CORREO

La configuración de los diferentes clientes de correo se muestran en los anexos

8, 9 y 10. En todos los casos se muestran las pruebas de envío y recepción de mensajes encriptados; finalmente, el anexo 11 muestra el acceso a la cuenta "usuario1" vía Internet Explorer.

Anexo 8: Configuración y prueba de Correo Electrónico con *Mozilla Mail & News Groups*

Anexo 9: Configuración y prueba de Correo Electrónico con *Netscape Mail & News Groups*

Anexo 10: Configuración y prueba de Correo Electrónico con *Microsoft Outlook*

Anexo 11: Prueba de Acceso *Web* con *Microsoft Internet Explorer*

3.4 PRESUPUESTO REFERENCIAL

Microsoft Exchange Server es una de las implementaciones más comunes de servidores de correo electrónico, por ello se realizará una comparación de costos entre este producto y la solución presentada en este capítulo, *Kerio Mail Server.*

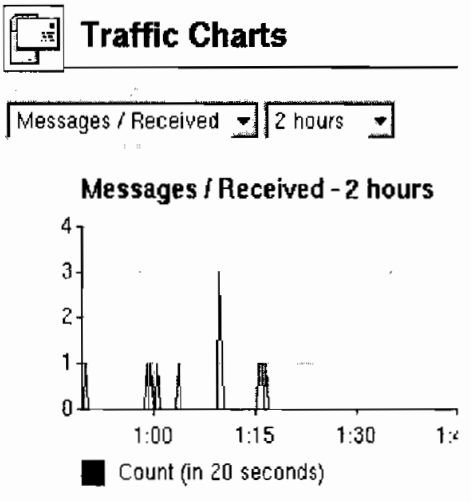


Figura 3.22 Monitoreo

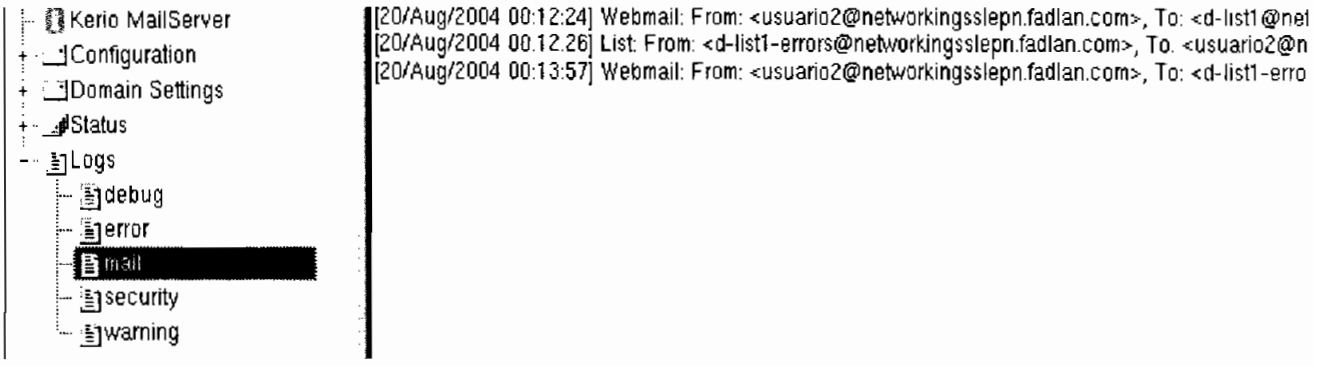


Figura 3.23 Log de Tráfico de Correo

Según una cotización realizada, el costo del paquete *Microsoft Exchange Server* y las licencias para un número dado de usuarios son:

Microsoft Exchange Server (discos de instalación y Licencia)	USD 784
Costo por licencia por usuario (paquete de 100)	USD 89

Estos costos incluyen el Impuesto al Valor Agregado (IVA).

Por lo tanto, si se desea implementar una solución con *Microsoft Exchange Server* en una mediana empresa de promedio 100 usuarios se tendrían los siguientes costos por la compra del *software* servidor y las correspondientes licencias.

Costo del *Software* Servidor	USD 784
Costo de licencias (100 usuarios a USD 89 c/u)	USD 8900

Adicionalmente se deben considerar gastos tales como la instalación y configuración y mantenimiento tanto del servidor y de las cuentas de los usuarios. Lo cual tiene un costo aproximado de USD 400 de instalación y configuración y USD 200 mensuales durante un año por mantenimiento.

De esta manera el costo por Instalación, Configuración y Mantenimiento

durante en primer año es: USD 2800

Total USD 12484

** Tomar en cuenta que no se incluyen módulos tales como el Antivirus.

Por su parte *Kerio Mail Server* con módulo de antivirus *McAfee* incluido presenta los siguientes costos.

Kerio Mail Server con *McAfee* (incluye licencia de 20 usuarios)	USD 699
Paquete de 100 usuarios	USD 799
Costo por Suscripción	USD 499
Costo por Instalación y Configuración Inicial	USD 300
Costo Total de *Software* Servidor y licencias de usuarios	USD 2297

La suscripción incluye:

- Actualizaciones gratuitas del servidor de correo por un período de un año.
- Soporte técnico gratuito vía telefónica.
- Actualizaciones de las definiciones de las bases de datos del antivirus.

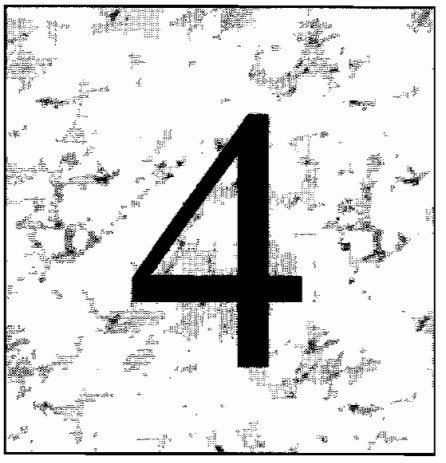
Finalmente se deben considerar gastos comunes para ambas implementaciones tales como:

- El costo del servidor, en el cual en promedio se invierte USD 1000 para compra del hardware del computador.
- La compra de los certificados digitales a las autoridades de Certificación de reconocidas mundialmente tales como Verisign, Thawte o Geotrust. Teniendo éstos los siguientes costos:
    - Verisign USD 895
    - Thawte USD 149
    - Geotrust USD 229

Los costos contemplan Certificados Digitales que soportan encripción de hasta 128 bits y su tiempo de validez es un año.

Uno de los objetivos de este plan de titulación era presentar a *Kerio Mail Server* como una solución económicamente alcanzable por pequeñas y medianas empresas, lo cual se demuestra en base a los costos presentados en este presupuesto referencial.

CAPÍTULO



ANÁLISIS DE SESIONES SSL

CAPÍTULO CUATRO

ANÁLISIS DE SESIONES SSL

4.1 EL *SNIFFER* COMO HERRAMIENTA DE ANÁLISIS DE PROTOCOLOS

El *Sniffer* es una herramienta cuyo objetivo principal es permitir la captura y visualización de los paquetes que circulan en una red de datos.

Se puede pensar en un *Sniffer* como un agente que permite visualizar todo cuanto pase por los cables de la red (siempre y cuando se tengan las condiciones necesarias); así como el voltímetro para un electricista, el *Sniffer* es una herramienta esencial para que un Administrador de Red pueda saber que está pasando en el entorno bajo su administración.

El *Sniffer* es una herramienta que también puede ser utilizada como Analizador de Protocolos, ya que al capturar todos los paquetes que circulan por la red y desplegarlos para su respectiva visualización, permite analizar los diferentes paquetes que se deben intercambiar entre dos o más entidades haciendo uso de un determinado protocolo para que su comunicación se lleve a cabo.

Se tienen diferentes tipos de Analizadores de Protocolos, típicamente se dividen en analizadores por software y analizadores por hardware. En el primer grupo un computador es el encargado de capturar y visualizar los paquetes haciendo uso de su propia memoria. En los analizadores por hardware es otro el dispositivo que captura los paquetes haciendo uso de su memoria y luego transmite estos datos a un computador para su respectiva visualización (figura 4.1).

4.2 EL *SNIFFER* ETHEREAL

En tiempos pasados el *Sniffer* era una herramienta muy costosa además de propietaria, sin embargo, hoy en día Ethereal es una herramienta gratuita distribuida bajo la licencia GNU *General Public Licence* (GPL).

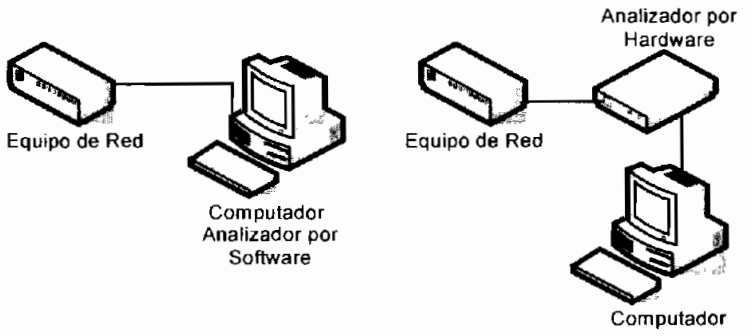


Figura 4.1 Analizador por Software vs. Analizador por Hardware

Ethereal es uno de los analizadores de protocolos más usados por esta característica y además por soportar una gran cantidad de protocolos tal como se puede observar en la siguiente referencia:
http://www.ethereal.com/docs/dfref/

Entre las características principales de Ethereal se pueden mencionar las siguientes:

- Disponible para entornos Unix y Windows.
- Captura en vivo de los paquetes.
- Despliegue de paquetes con información detallada de sus protocolos.
- Abrir y Guardar información capturada.
- Importar y Exportar la información capturada desde una variedad de formatos originados por otros programas de capturas, tales como: *Novell LANalyzer, Sun Snoop, Microsoft Network Monitor*, entre otros.
- Filtros de información con varios criterios.

- Búsqueda de información con varios criterios, entre otras.

Para mayor información tanto la distribución de Ethereal como la documentación se pueden conseguir en las siguientes direcciones: http://www.ethereal.com/download.html , http://www.ethereal.com/docs

4.3 CAPTURA DE PAQUETES DURANTE EL ESTABLECIMIENTO DE LA CONEXIÓN AL SITIO WEB SEGURO

4.3.1 MENSAJE *CLIENT HELLO*

El mensaje *Client hello* (figura 4.2) es el primer mensaje que se envía durante el establecimiento de una sesión SSL. Se puede observar que la capa de registro (*Record Layer*) es la base sobre la cual está la capa de saludo (*Handshake protocol*), además, se puede verificar la existencia de los siguientes parámetros:

- ***Handshake type***: 1. Quiere decir que el mensaje es un *client hello.*
- ***Version:SSL3.0***. Significa que se usa el protocolo SSLv3.0.
- ***Random.gmt_unix_time***. Es una fecha aleatoria que toma el cliente.
- ***Sesion ID***. Es el identificador que el cliente desea usar para la sesión. El servidor verificará este identificador, de estar disponible se lo podrá usar.
- ***Cipher Suites.*** El cliente enumera en orden de prioridad el conjunto de algoritmos que puede usar en la sesión; el servidor por su parte escogerá la combinación más segura que pueda usar.
- ***Compression Methods.*** Son los algoritmos de compresión que el cliente puede usar en la sesión.

4.3.2 MENSAJE *SERVER HELLO*

El mensaje *Server Hello* llega al cliente como respuesta del mensaje

Client Hello enviado. De igual manera se puede observar en la figura 4.3 que la capa de registro (*Record Layer*) es la base sobre la cual interactúa el protocolo de saludo (*Handshake Protocol*); en esta fase los siguientes parámetros son propios del mensaje *Server Hello*.

- ***Handshake Type: 1***. Identifica al mensaje como *Server Hello*.
- ***Version:SSL 3.0***. Indica que el servidor manejará SSL v3.0 para la sesión.
- ***Random.gmt_unix_time***. Es una fecha aleatoria escogida por el servidor.
- ***Sesion ID***. Es el identificador que el servidor otorga a la sesión; puede aceptar el identificador propuesto por el cliente si éste no existe previamente en su base de datos de identificadores de sesiones.
- ***Cipher Suite***. Es la combinación seleccionada por el servidor de las enviadas como candidatas por el cliente. El *Cipher Suite* es la combinación de algoritmos de intercambio de llaves, encripción y *hashing* que serán usados en la sesión.
- ***Compression Method***. Es el algoritmo de compresión de datos que el servidor ha escogido para usar, basándose en las alternativas que ha presentado el cliente en el correspondiente mensaje *client hello*.

A continuación el servidor envía el mensaje *certificate*. En el mismo viaja el certificado digital que se presentará visualmente al cliente para que éste pueda autenticar al servidor. Se puede observar en la figura 4.3 varios parámetros del certificado digital tales como: Identificación del firmante, Período de validez del certificado, el nombre del titular del certificado, entre otros.

4.3.3 MENSAJES *CERTIFICATE, CLIENT KEY EXCHANGE, CERTIFICATE VERIFY, CHANGE CIPHER SPEC, Y FINISHED*

A continuación el cliente presenta una serie de mensajes para completar por su parte la negociación del establecimiento de la sesión SSL.

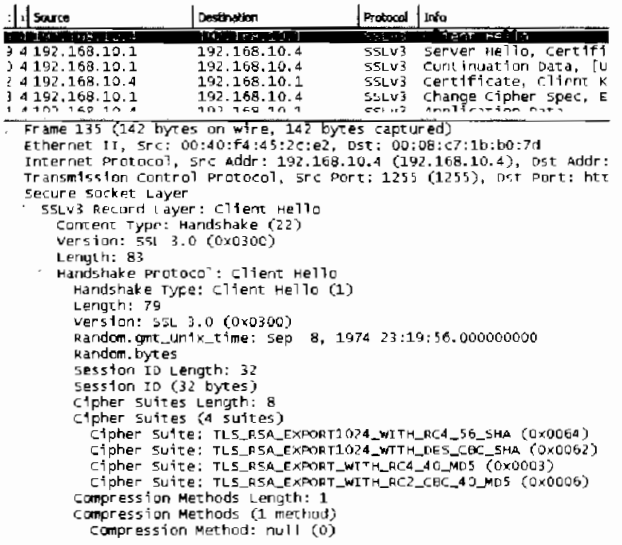


Figura 4.2 Mensaje Client Hello, conexión al sitio web seguro

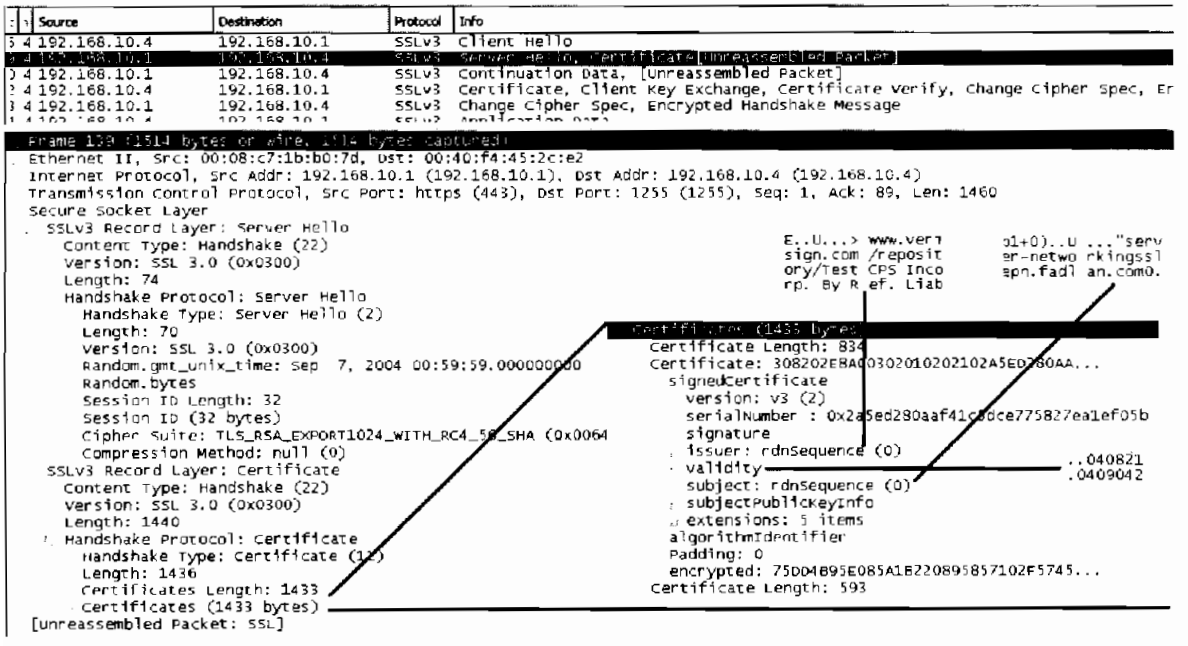


Figura 4.3 Mensaje Server Hello, conexión al sitio Web Seguro

Envía su certificado digital el cual según se puede observar en la figura 4.4 tiene parámetros tales como: la identificación del firmante del certificado, las fechas de validez del mismo, y la identificación del titular del certificado, entre otros.

Luego el cliente también envía los mensajes *Client Key Exchange*, *Certificate Verify* (con el cual explícitamente se solicita la verificación del certificado digital), *Change Cipher Spec* (el cual sirve para que el servidor copie lo existente en el Estado de Lectura Pendiente en el Estado de Lectura Actual) y el mensaje *Finished*, el mismo que está representado por un mensaje encriptado bajo los parámetros anteriormente negociados; el *sniffer* lo identifica como un *Encripted Handshake Message*.

4.3.4 MENSAJES *CHANGE CIPHER SPEC* Y *FINISHED* DEL SERVIDOR

Finalmente como se observa en la figura 4.5, el servidor envía un mensaje *Change Cipher Spec* con lo cual el cliente copia lo existente en el Estado de Lectura Pendiente en el Estado de Lectura Actual, así como su mensaje *Finished* que al igual al caso del mensaje *Finished* del Cliente se representa por un *Encrypted Handshake Message*, el cual es un mensaje encriptado con los parámetros que se han negociado previamente.

Los mensajes *Finished* son los primeros mensajes que están protegidos con los parámetros negociados durante la fase de *handshake*.

4.3.5 *APPLICATION DATA*

Luego de la fase de *handshake*, los datos viajan encriptados entre cliente y servidor. Dichos datos se transmiten bajo los parámetros de seguridad negociados previamente (figura 4.6)

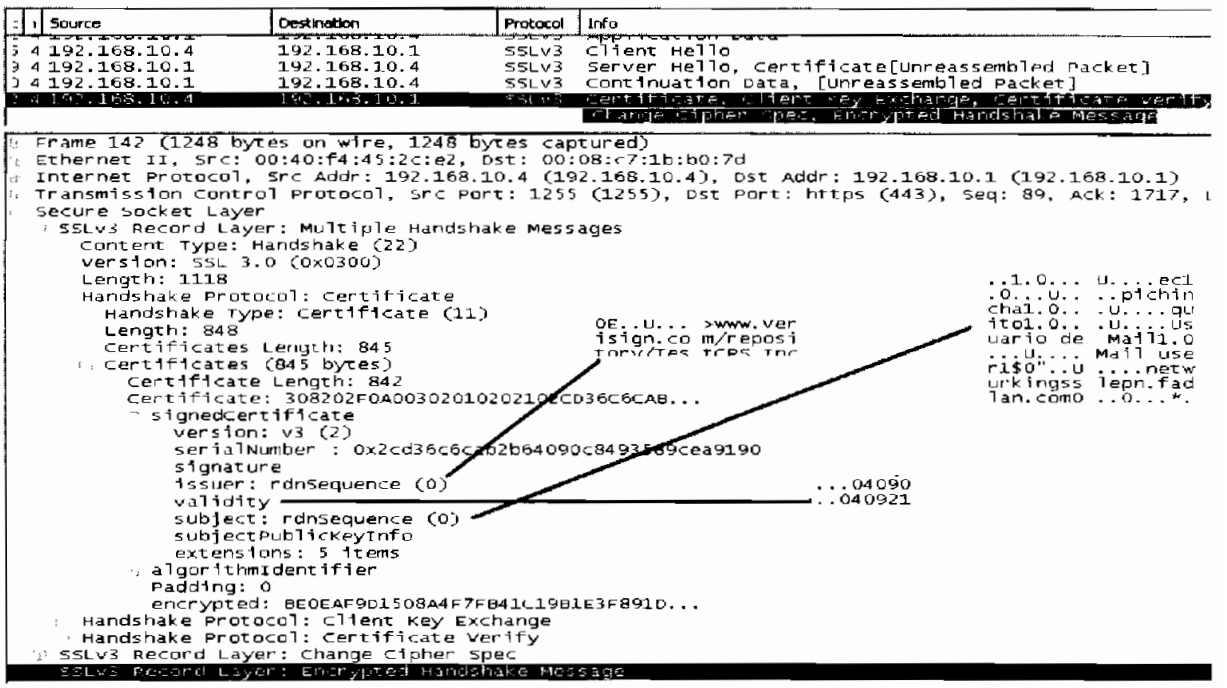


Figura 4.4 Mensajes múltiples del cliente, sesión al sitio web seguro

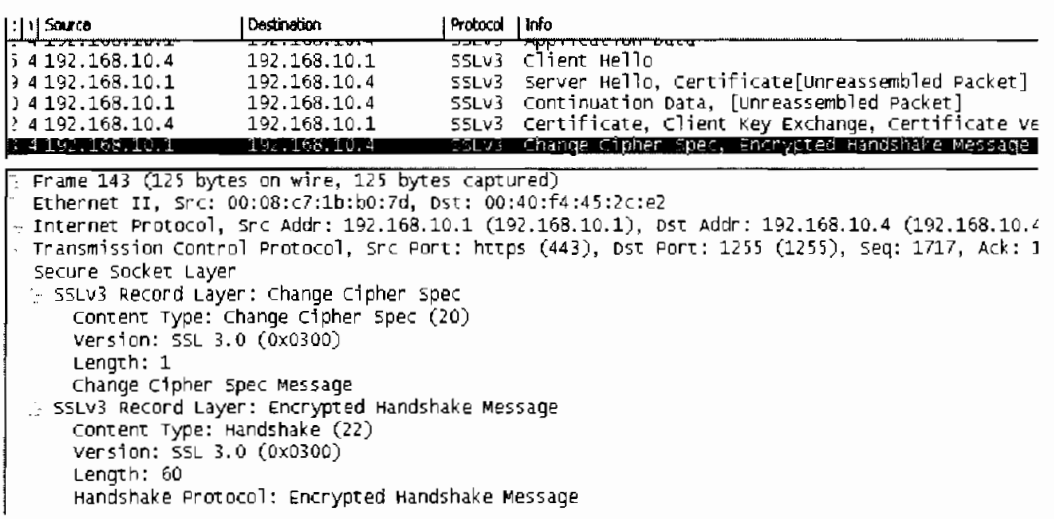


Figura 4.5 Mensajes múltiples del servidor Web

:	)	Source	Destination	Protocol	Info
)	4	192.168.10.1	192.168.10.4	SSLv3	Continuation Data
2	4	192.168.10.4	192.168.10.1	SSLv3	Certificate, Clie
3	4	192.168.10.1	192.168.10.4	SSLv3	Change Cipher Spe
4	4	192.168.10.4	192.168.10.1	SSLv3	Application Data
5	4	192.168.10.1	192.168.10.4	SSLv3	Application Data

```
Frame 144 (425 bytes on wire, 425 bytes captured)
Ethernet II, Src: 00:40:f4:45:2c:e2, Dst: 00:08:c7:1b:b0:7d
Internet Protocol, Src Addr: 192.168.10.4 (192.168.10.4), Dst A
Transmission Control Protocol, Src Port: 1255 (1255), Dst Port:
Secure Socket Layer
    SSLv3 Record Layer: Application Data
        Content Type: Application Data (23)
        Version: SSL 3.0 (0x0300)
        Length: 366
        Application Data
```

Figura 4.6 Datos de aplicación encriptados

4.4 ANÁLISIS DE LAS FASES DEL PROTOCOLO SSL EN LA CONEXIÒN AL SITIO WEB SEGURO

A continuación se explicará el proceso de establecimiento de una sesión SSL basándose en la figura 4.7.

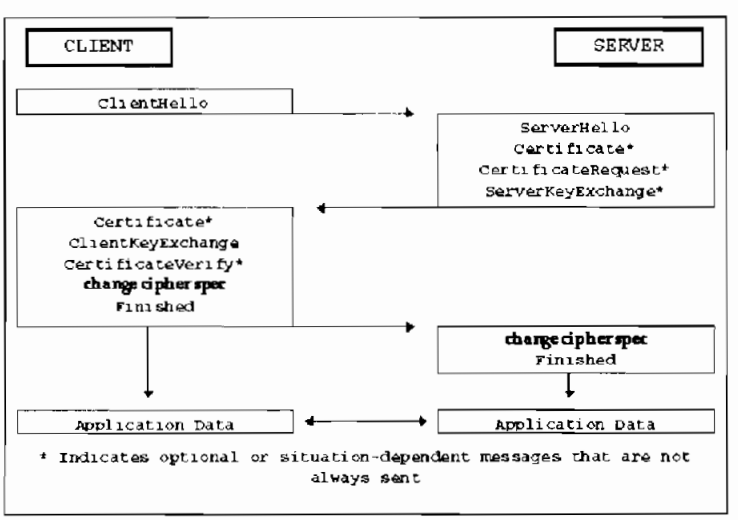


Figura 4.7 Fases del establecimiento de una sesión SSL [10]

La figura 4.2 muestra la captura de un paquete que lleva el mensaje

client hello. Éste, al ser el primer mensaje que se intercambia en el establecimiento de una sesión SSL, lleva consigo parámetros vitales como son: la versión de SSL (en este caso SSL v3.0) que el cliente usará, las combinaciones *Cipher Suite* en las cuales se especifican los algoritmos que el cliente puede soportar en la sesión para proteger los datos en orden de prioridad; así como los algoritmos que el cliente puede usar para comprimir los datos (en este caso no se usa algún algoritmo de compresión).

Para el establecimiento de esta sesión SSL el cliente envía las combinaciones en su mensaje *client hello* (figura 4.8).

Como respuesta al mensaje *Client Hello*, el servidor envía su mensaje *Server Hello* en el cual, según la figura 4.3, se observa que el servidor también usa SSL v3.0 para la negociación de la sesión. Además, según el paquete visualizado, el servidor ha escogido la primera combinación de algoritmos de las posibles enviadas por el cliente ya que ésta es la más segura soportada por el mismo.

```
Cipher Suite: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x0
Cipher Suite: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x
Cipher Suite: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x0003)
Cipher Suite: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x0
```

Figura 4.8 Combinaciones posibles soportadas por el cliente

Existen mensajes que pueden ser enviados opcionalmente por el servidor en esta etapa de la negociación, como son: *Certificate, Certificate Request, y Server Key Exchange*. Dado que el objetivo de este proyecto es autenticar al servidor y al cliente, éstos se envían desde el servidor.

Posteriormente, el servidor envía un mensaje que contiene su certificado digital (CD). La figura 4.3 muestra detalles importantes en el CD del servidor tales como:

- Nombre de la Autoridad Certificadora que firma el CD. En este caso es *Verisign Repository Test.*
- Fechas de validez del CD. No válido antes de 040821 (agosto 21 del 2004) y no válido después de 040904 (septiembre 4 del 2004).
- Nombre del Titular del CD. El titular es:

server-networkingsslepn.fadlan.com.

Por su parte el cliente envía los siguientes mensajes:
El mensaje de Certificado Digital; según se muestra en la figura 4.4 el CD del cliente tiene los siguientes parámetros:

- Nombre de la Autoridad Certificadora que firma el CD. En este caso es *Verisign Repository Test.*
- Fechas de validez del CD. No válido antes de 040906 (septiembre 6 del 2004) y no válido después de 040921 (septiembre 21 del 2004).
- Nombre del Titular del CD. El titular es: Usuario de Mail.

El mensaje *Client Key Exchange*, el mensaje *Certificate Verify* (con el cual se verifica que el cliente posee la llave privada del certificado presentado), el mensaje *Change Cipher Spec* (con el cual el servidor copia lo existente en el estado de Lectura Pendiente en el Estado de Lectura Actual) y el mensaje *Finished* que constituye el primer mensaje encriptado; dando de esta manera por terminada la etapa de *Handshake* por parte del cliente.

Para finalizar la etapa de *Handshake*, el servidor envía un mensaje *Change Cipher Spec* y luego un mensaje *Finished* (figura 4.5). Luego de este mensaje todos los paquetes que se envíen forman parte de los datos de la aplicación, éstos viajarán de acuerdo con los algoritmos que se hayan negociado anteriormente (figura 4.6).

4.5 CAPTURA DE PAQUETES DURANTE EL ESTABLECIMIENTO DE LA CONEXIÓN A UN SITIO WEB NO SEGURO

Al analizar el tráfico que se genera al acceder a un sitio no seguro (figura 4.9), se puede observar el tráfico tipo HTTP (*HiperText Transfer Protocol*).

HTTP permite la comunicación entre cliente y servidor, éstos se comunican mediante los métodos determinados por el protocolo.

Existen dos tipos de solicitudes en HTTP: las sencillas y las completas. La primera de ellas no especifica el protocolo ni la versión del mismo cuando se hace la petición de una página deseada. Mientras que, las solicitudes completas especifican también el protocolo y la versión que se usa.

GET es uno de los métodos de HTTP. Como se puede observar en la figura 4.9 existen solicitudes completas que hacen uso del protocolo HTTP y la versión 1.1.

	Source	Destination	Protocol	Info
19	192.168.10.4	192.168.10.1	TCP	1370 > http [SYN] Seq=0 Ack=0 Win=8192 Len=0 MSS=1460
20	192.168.10.1	192.168.10.4	TCP	http > 1370 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
21	192.168.10.4	192.168.10.1	TCP	1370 > http [ACK] Seq=1 Ack=1 Win=8760 Len=0
22	192.168.10.4	192.168.10.1	HTTP	GET /index_files/pixel.gif HTTP/1.1
23	192.168.10.1	192.168.10.4	TCP	http > 1369 [ACK] Seq=1 Ack=245 Win=6432 Len=0
24	192.168.10.4	192.168.10.1	HTTP	GET /index_files/leftbox_top.gif HTTP/1.1
25	192.168.10.1	192.168.10.4	TCP	http > 1370 [ACK] Seq=1 Ack=251 Win=6432 Len=0
26	192.168.10.1	192.168.10.4	HTTP	HTTP/1.1 200 OK (GIF89a)
27	192.168.10.1	192.168.10.4	TCP	http > 1369 [FIN, ACK] Seq=299 Ack=245 Win=6432 Len=0
28	192.168.10.4	192.168.10.1	TCP	1369 > http [ACK] Seq=245 Ack=300 Win=8462 Len=0

Figura 4.9 Acceso a sitio no seguro

El protocolo HTTP tiene por objetivo la transferencia de datos. En la mayor parte de los casos sus implementaciones hacen uso del protocolo TCP (*Transmission Control Protocol*), con lo cual se asegura únicamente la entrega confiable de los datos al destino; no se puede evitar intercepción, sustitución o algún otro tipo de ataque a los datos.

La figura 4.10 muestra la captura de un paquete que es enviado desde el servidor de páginas *web* hacia el cliente; como se puede observar los datos viajan sin algún tipo de seguridad, por lo cual pueden ser interceptados e interpretados por el atacante, informándose de esta manera del contenido de los datos.

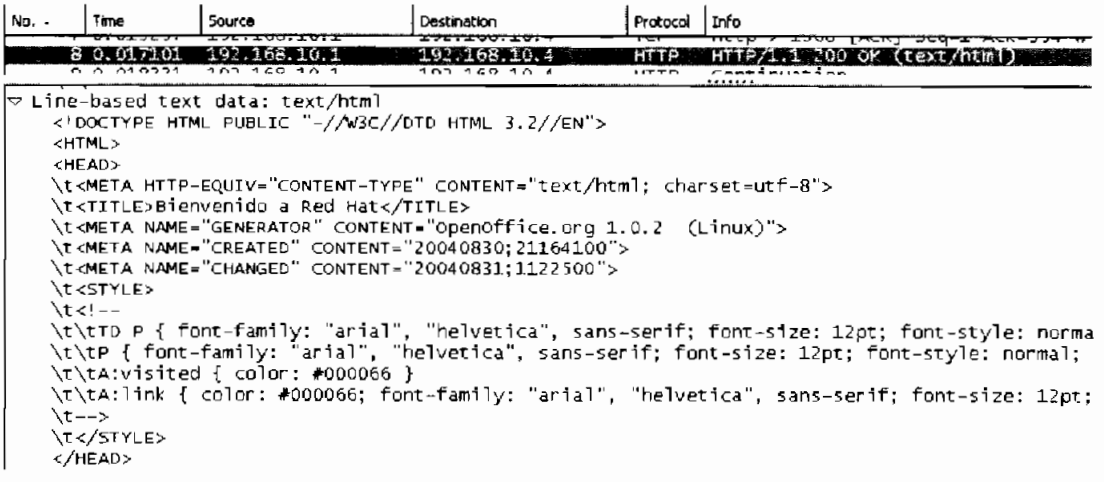


Figura 4.10 El protocolo HTTP en la transferencia de datos

4.6 CAPTURA DE PAQUETES DURANTE EL ESTABLECIMIENTO DE LA CONEXIÓN AL SERVIDOR DE CORREO ELECTRÓNICO SEGURO

Tal como se explica en la sección 4.3.1 el mensaje *Client Hello* consta de la versión de SSL a usar, el identificador de la sesión, las combinaciones de *Cipher Suites* así como los algoritmos de compresión soportados por el cliente para la sesión SSL, entre otras (figura 4.11).

Por su parte el servidor responde con el mensaje *Server Hello* en el cual según la figura 4.12, se pueden citar parámetros tales como: la versión de SSL, el identificador asignado a la sesión, la combinación *Cipher Suite* así como el

método de compresión escogidos. Adicionalmente, el servidor envía otro mensaje; en esta ocasión el certificado digital que será presentado al cliente para la correspondiente autenticación del servidor. En este mensaje se pueden distinguir parámetros tales como: El nombre del Firmante del CD, las fechas de validez del mismo, y el nombre del titular del CD, entre otros.

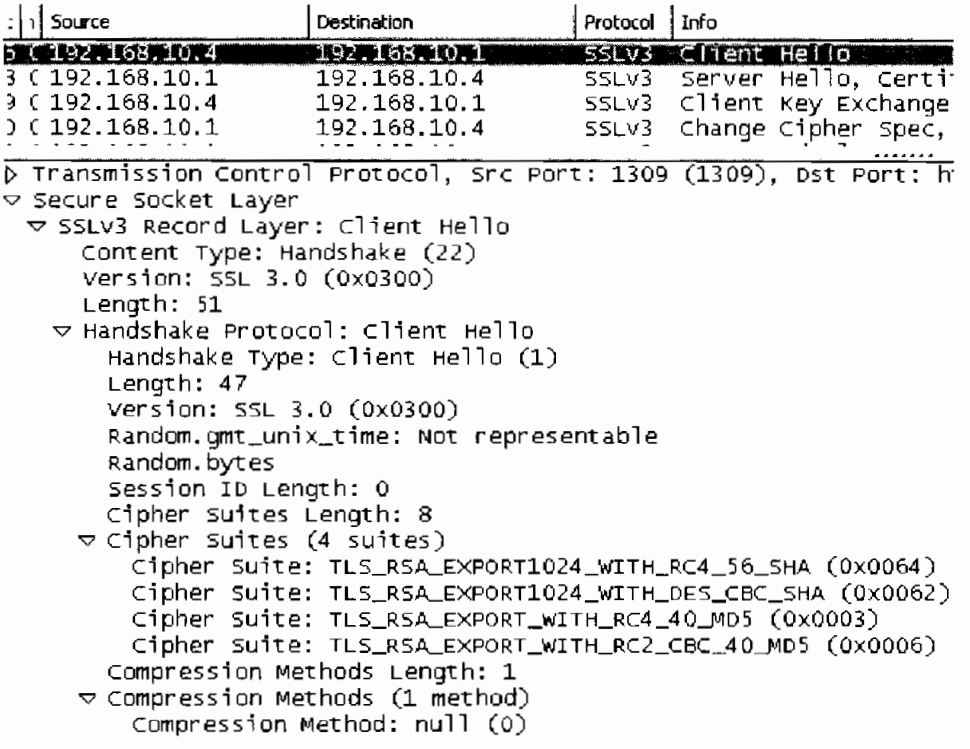


Figura 4.11 Mensaje Client Hello, conexión al servidor mail seguro

Para dar por finalizada la fase de mensajes *hello* el servidor envía un mensaje *Server Hello Done*.

Caso contrario al análisis del establecimiento de la sesión con el servidor *web*, el servidor *mail* no ha hecho la petición del certificado del cliente (con un mensaje *CertificateRequest*), por ello solo se realiza la autenticación del servidor por parte del cliente.

Finalmente en la figura 4.13, el cliente envía un mensaje *Change Cipher Spec* y seguidamente un mensaje *Finished*, caracterizado en Ethereal como un *Encrypted Handshake Message*.

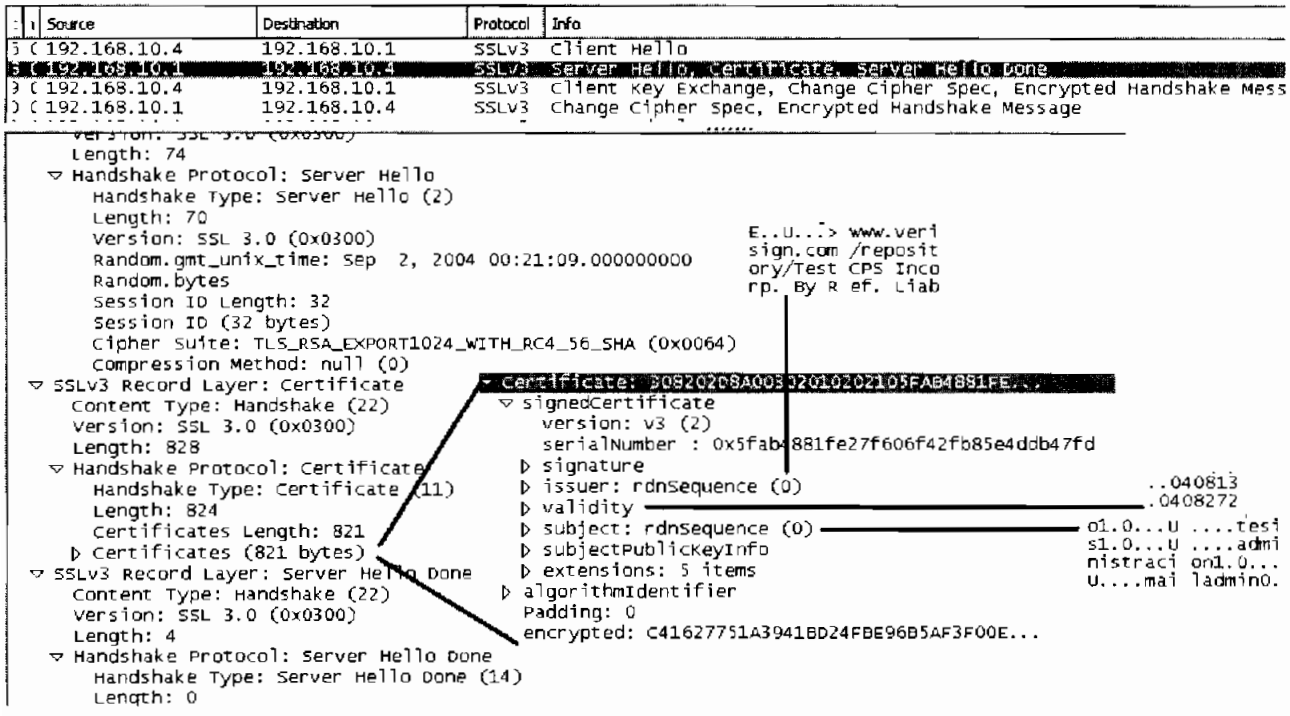


Figura 4.12 Mensaje Server Hello, conexión al servidor mail seguro

Luego de finalizar la fase de *handshake,* al igual que en el caso del servidor *web,* tanto cliente como servidor envian sus datos bajo los parámetros de seguridad negociados previamente, los cuales viajan haciendo uso del protocolo *Application Data.*

4.7 ANÁLISIS DE LAS FASES DEL PROTOCOLO SSL EN LA CONEXIÓN AL SERVIDOR DE CORREO ELECTRÓNICO SEGURO

Cuando se analiza el proceso de establecimiento de la sesión SSL entre el cliente y el servidor en el servidor de correo electrónico, se tienen algunas

variantes.

Como se observa en la figura 4.11, el campo de identificador de sesión del mensaje *Client Hello* carece de contenido, lo que significa que el cliente desea establecer una conexión nueva con el servidor y no propone un identificador de la sesión como lo hace en el caso examinado del servidor web.

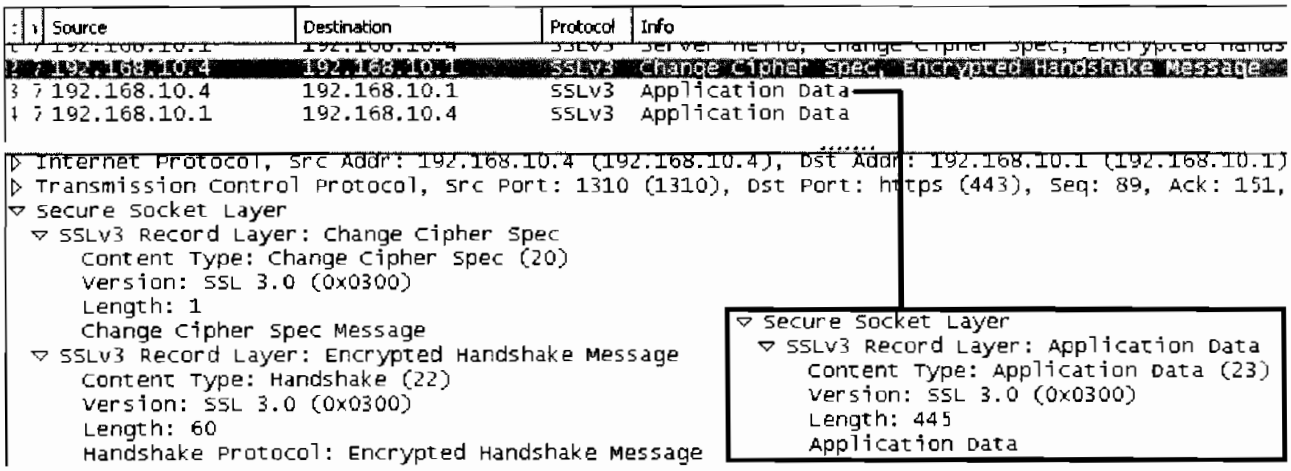


Figura 4.13 Mensajes múltiples de Cliente, sesión al sitio mail seguro

De igual manera el cliente envía la combinación de algoritmos que desearía usar en orden de preferencia, así también no usa algún algoritmo de compresión para sus datos.

Por su parte el servidor envía su correspondiente mensaje *Server Hello* (figura 4.12), en el cual especifica un identificador que ha asignado a la sesión, así como la combinación de algoritmos escogida para usar en la sesión y el algoritmo de compresión correspondiente, el cual en este caso también es nulo.

Adicionalmente el servidor envia su CD en el mensaje *Certificate,* el cual según la figura 4.12 tiene los siguientes parámetros:

- Nombre de la Autoridad Certificadora que firma el CD. En este caso es

Verisign Repository Test.

- Fechas de validez del CD. No válido antes de 040813 (agosto 8 del 2004) y no válido después de 040827 (agosto 27 del 2004).
- Nombre del Titular del CD. El titular es: Mail Admin.

Por último se envía el mensaje *Server Hello Done*, con el cual el servidor finaliza por su parte el intercambio de mensajes *Hello*.

El cliente para finalizar el *handshake* envía un mensaje *Change Cipher Spec* y luego un mensaje *Finished* (figura 4.13). Luego de esto el cliente comienza a interactuar con el servidor enviando y recibiendo datos de la aplicación los mismos que son gestionados bajo los parámetros negociados.

4.8 CAPTURA DE PAQUETES DURANTE EL ESTABLECIMIENTO DE LA CONEXIÓN AL SERVIDOR DE CORREO ELECTRÓNICO NO SEGURO

Para fines de contraposición de una sesión segura contra una sesión no segura se ha configurado previamente el usuario de correo "usuario1" el cual tiene por *password* "usuario1".

El protocolo HTTP tiene también los métodos PUT y POST. El primero de ellos es contrario al método GET ya que en lugar de leer una página, la escribe; es usado para construir cabeceras de validación de usuarios. Por su parte el método POST también lleva un URL (*Uniform Resource Locator*) pero en lugar de reemplazar los datos existentes, se anexa a ellos.

Según la figura 4.14, el método POST es usado para enviar el nombre de usuario y contraseña hacia el servidor.

De esta manera, en la figura 4.15, se puede observar como se envía en texto plano tanto el nombre del usuario como su contraseña, poniendo en evidencia algo muy valioso para el cliente. Si los datos viajan en texto plano son vulnerables a intercepción de los mismos.

Suponiendo que tanto el nombre de usuario cómo su contraseña han sido capturados, el atacante puede ingresar a revisar el correo de su víctima o aún peor, puede enviar correos a otras personas suplantando la identidad, lo cual puede acarrear serios inconvenientes.

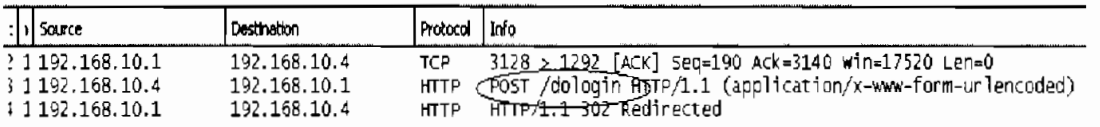


Figura 4.14 Uso del método POST

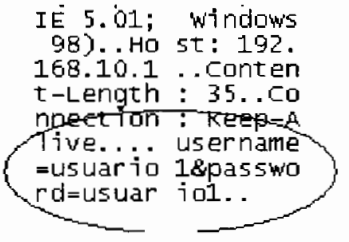


Figura 4.15 Datos en texto plano (1)

Se puede concluir que al hacer uso del protocolo http, tanto cliente como servidor están expuestos a que sus datos puedan ser interceptados y sufrir serios daños.

La figura 4.16 muestra una fracción de la información transmitida desde el servidor hacia el cliente, se puede observar claramente que ésta viaja también como texto plano, constituyendo de esta manera otro punto vulnerable en la comunicación, si un atacante tiene como objetivo enterarse del contenido de una conversación entre el servidor y el cliente, basta con colocar un *sniffer* en medio e interpretar los paquetes capturados para lograr su objetivo.

El caso contrario se observa en la figura 4.17, los datos ya trabajando con el protocolo SSL viajan seguros, no viajan en texto plano; de esta manera si un atacante captura los paquetes de la conversación sostenida entre el cliente y servidor, al momento de analizarlos no podría interpretarlos.

De esta manera los métodos para lograr la comunicación cada vez requieren más seguridad por los potenciales peligros que la red de Internet representa. Haciendo un análisis desde el punto de vista de la evolución de los métodos que brindan seguridad a los datos y como nacieron estas necesidades se puede citar lo siguiente.

```
0010  01 6f a8 0a 40 00 80 06  bc 28 c0 a8 0a 04 c0 a8   .o..@... .(......
0020  0a 01 05 1a 00 50 00 42  97 ad 87 37 e5 bc 50 18   .....P.B ...7..P.
0030  22 38 f3 db 00 00 47 45  54 20 2f 67 66 78 2f 74   "8....GE T /gfx/t
0040  72 65 65 5f 6e 2e 67 69  66 20 48 54 54 50 2f 31   ree_n.gi f HTTP/1
0050  2e 31 0d 0a 41 63 63 65  70 74 3a 20 2a 2f 2a 0d   .1..Acce pt: */*.
0060  0a 52 65 66 65 72 65 72  3a 20 68 74 74 70 3a 2f   .Referer : http:/
0070  2f 31 39 32 2e 31 36 38  2e 31 30 2e 31 2f 6c 69   /192.168 .10.1/li
0080  73 74 3f 66 6f 6c 64 65  72 3d 25 37 45 75 73 75   st?folde r=%7Eusu
0090  61 72 69 6f 31 40 6e 65  74 77 6f 72 6b 69 6e 67   ario1@ne tworking
00a0  73 73 6c 65 70 6e 2e 66  61 64 6c 61 6e 2e 63 6f   sslepn.f adlan.co
00b0  6d 2f 49 4e 42 4f 58 0d  0a 41 63 63 65 70 74 2d   m/INBOX. .Accept-
00c0  4c 61 6e 67 75 61 67 65  3a 20 65 73 2d 6d 78 0d   Language : es-mx.
00d0  0a 41 63 63 65 70 74 2d  45 6e 63 6f 64 69 6e 67   .Accept- Encoding
00e0  3a 20 67 7a 69 70 2c 20  64 65 66 6c 61 74 65 0d   : gzip,  deflate.
00f0  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   .User-Ag ent: Moz
0100  69 6c 6c 61 2f 34 2e 30  20 28 63 6f 6d 70 61 74   illa/4.0  (compat
0110  69 62 6c 65 3b 20 4d 53  49 45 20 35 2e 30 31 3b   ible; MS IE 5.01;
0120  20 57 69 6e 64 6f 77 73  20 39 38 29 0d 0a 48 6f    Windows  98)..Ho
0130  73 74 3a 20 31 39 32 2e  31 36 38 2e 31 30 2e 31   st: 192. 168.10.1
0140  0d 0a 43 6f 6e 6e 65 63  74 69 6f 6e 3a 20 4b 65   ..Connec tion: Ke
0150  65 70 2d 41 6c 69 76 65  0d 0a 43 6f 6f 6b 69 65   ep-Alive ..Cookie
0160  3a 20 75 73 65 72 69 64  3d 64 31 39 38 30 37 30   : userid =d198070
0170  36 62 62 39 38 32 64 34  61 0d 0a 0d 0a           6bb982d4 a....
```

Figura 4.16 Datos en texto plano (2)

Tomando en cuenta que desde sus inicios, el Internet fue ideado netamente para el intercambio de información entre científicos e investigadores, los cuales llegaron a tener a su disposición ingentes cantidades de información organizada en la base de datos más potente que jamás pudieron imaginar. Además tanto los centros de investigación como las universidades pudieron acceder a este medio de información lo cual enalteció mucho más esta potencialidad.

El principio de funcionamiento de esta gran base de datos era sencillo, la

distribución de la información de forma gratuita hacía posible que todos los usuarios tengan acceso a toda la información publicada por los demás.

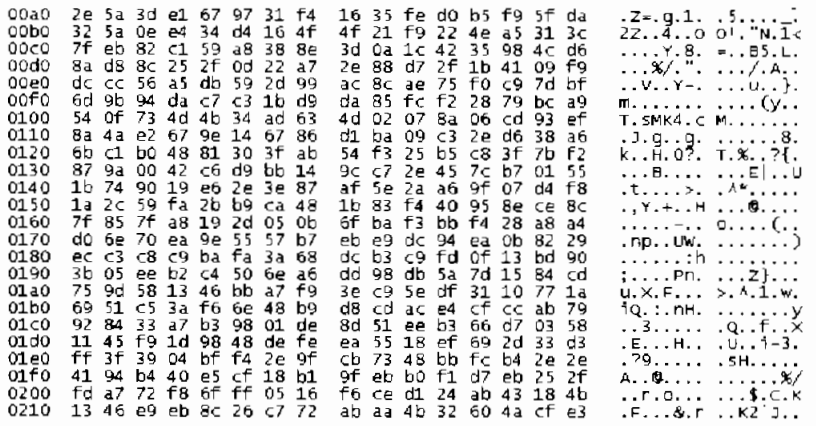


Figura 4.17 Datos encriptados

Sin embargo, la información fue creciendo cada vez más; y a la par los usuarios crearon nuevas aplicaciones para esta autopista de la información, tales como venta de servicios, productos, entre otros, y surgieron de esta manera las primeras operaciones comerciales en el Internet.

Pero no todas las personas que de una u otra manera se conectan al Internet persiguen el mismo objetivo, existen usuarios que buscan hacer uso del desconocimiento de otros para su beneficio propio; los llamados piratas de la red se dedican entre una de sus principales ocupaciones a capturar los paquetes que viajan por la red (haciendo uso de herramientas tales como los *sniffers*), luego de lo cual los analizan e interpretan los resultados.

Entonces los usuarios se ven obligados a buscar métodos que ayuden a proteger la información que consideran como privada, pudiendo ser ésta de cualquier naturaleza: números de tarjetas de crédito, nombres de usuarios y contraseñas, inclusive cualquier texto que puede revelar información valiosa para el atacante.

El protocolo SSL ofrece una variedad de métodos que conjugados

trabajan como un muro de protección para los datos que viajan por el Internet, proveyendo de esta manera encripción, compresión, y métodos que permiten saber que el mensaje que llega al destino es efectivamente el que se envió.

Sin embargo, los usuarios están en la libertad de adoptar o no estos métodos para asegurar sus datos, claro está, bajo cuenta y riesgo propio de exponerse a posibles robos de su información.

Así como una de las necesidades del ser humano es sentirse seguro, *Secure Sockets Layer* constituye una plataforma de seguridad para los datos, proveyendo de esta manera un valor agregado no tangible, como es el permitir a sus usuarios gozar de plena tranquilidad sabiendo que su información está segura de terceros no autorizados.

CAPÍTULO

CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO CINCO

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La autenticación, la integridad de los datos, la encripción y el no repudio, son métodos mediante los cuales se puede proveer ciertas características de seguridad a los datos. La autenticación, permite verificar que alguien es quien dice ser. La integridad de los datos, permite verificar que los datos han llegado a su destino sin haber sufrido alteraciones de ningún tipo. La encripción, por su parte permite a los datos soportar un ataque de intercepción, ya que aplicando la encripción, los datos son ilegibles para el atacante. Finalmente, el no repudio, permite al remitente estar seguro que el destinatario ha recibido la información y que éste no pueda alegar posteriormente no haberla recibido.

- Todas las empresas protegen sus activos, hoy en día inclusive el *software* es tomado como parte de los activos de una empresa pues a pesar de ser un intangible, provee facilidades inherentes a la automatización de los procesos de la misma.

- Tanto *software* como *hardware* en una red permiten el transporte de los datos de los usuarios, los mismos que pueden ser de cualquier índole por ejemplo, texto, imágenes, vídeos, etc. Sin importar su origen, los datos son importantes para cada usuario, es por ello primordial proveer servicios que permitan el tránsito seguro de los mismos como los citados anteriormente.

- Cuando se habla de redes de datos no se puede decir que esta es totalmente segura, más correcto es hablar de la fiabilidad de la misma. Siempre se tendrán zonas de riesgo en los sistemas lo cual permite que la vulnerabilidad pueda ser explotada; es por ello que, ofreciendo los servicios de seguridad mencionados en el proyecto, se puede hablar con certeza de una fiabilidad de la red, ya que ante un ataque ésta podrá reaccionar según los parámetros de su diseño.

- Existen varias alternativas tanto en *software* como en *hardware* para proveer cierto tipo de seguridad a los datos. En particular, el protocolo SSL es uno de los más difundidos por su uso en Internet, ya que conjuga todas las características que necesitan los datos para viajar seguros tales como: Encripción, Funciones *Hash*, y Compresión.

- El protocolo SSL se compone de otros protocolos, tales como: *Handshake Protocol, Alert Protocol, Change Cipher Spec Protocol y Application Data Protocol.* Cada uno de ellos juega un papel importante dentro del la estructura del protocolo, y permite el correcto funcionamiento de SSL.

- Una de las aplicaciones de SSL son las transacciones electrónicas mediante las cuales se asegura que los números de tarjetas de crédito no podrán ser visualizados por el atacante, pero no se puede asegurar lo que el receptor haga luego con esa información.

- El servidor Web Apache consta principalmente de tres módulos: Los Módulos Base, en los cuales se tienen las funcionalidades básicas del servidor. Los Módulos multiproceso, son responsables de conectar con los puertos de la máquina, aceptar las peticiones, y generar los procesos hijo que se encargan de servirlas. Finalmente

los módulos adicionales, los cuales proveen alguna funcionalidad extra al servidor *web*.

- Para configurar el servidor *Web* Apache en Linux Red Hat, es necesario entender la estructura del archivo httpd.conf, el cual es el corazón mismo del servidor, y adicionalmente configurar directivas extras que permiten dar cualidades especiales al servidor tales como la autenticación del servidor o inclusive del cliente.

- El servidor Web Apache permite el funcionamiento simultáneo de uno o varios dominios en un mismo computador, lo cual es conocido como *hosts* virtuales. Gracias a esta característica se pueden ahorrar los recursos computacionales, lo cual tiene gran acogida en empresas que optimizan sus recursos.

- Existen dos clases de *hosts* virtuales; los basados en nombre y los basados en IP. Los primeros comparten en el mismo computador una sola dirección IP para servir a varios dominios. Los basados en IP, solo conviven en el mismo computador pues es necesario que cada uno de ellos tenga una dirección IP en la cual se escuchen las peticiones de sus clientes.

- El servidor *Web* Apache gracias a sus directivas presenta alternativas personalizables las cuales deben ser configuradas por el administrador del servidor; pero al mismo tiempo pueden constituir agujeros de seguridad si se permite más de lo necesario, o por otro lado puede desaprovecharse el potencial del mismo si no se configura lo mínimo para proveer el servicio.

- Gracias a la implementación del servidor Web Apache y el servidor de correo electrónico Kerio, se ha conseguido satisfacer el objetivo

de este proyecto de titulación, que era proveer una solución económica de bajo costo como parte de la infraestructura de una Intranet aplicable a pequeñas y medianas empresas.

5.2 RECOMENDACIONES

- Existen implementaciones de toda índole que hacen uso del protocolo SSL, dos de ellas se han realizado en este proyecto, un servidor Web y un servidor *mail*, los cuales proveen autenticación ya sea al cliente o al servidor; sin embargo, las opciones que el servidor Web presenta en su configuración son muchas. Por ello se recomienda la investigación e implementación de otro protocolo que provea servicios equivalentes a SSL y contrastar las soluciones en función de seguridad.

- Como aporte extra al presente proyecto de titulación, haciendo uso del paquete OpenSSL se ha construido un CD para una AC y con éste se han firmado los CD de los clientes. Se recomienda la implementación de una infraestructura similar en la Escuela Politécnica Nacional, ya que de esta manera se puede autenticar los clientes que deseen acceder a determinados recursos.

- Se recomienda la implementación de un servidor de correo electrónico que goce con beneficios comparables a los que en este documento se presenta, ya que el asegurar los datos de los usuarios como por ejemplo sus contraseñas es vital.

- Si se desea reducir aun más los costos en implementación del servidor de correo electrónico, se recomienda el estudio e

implementación del servidor de correo electrónico que se incluye en las distribuciones de Linux como es *SendMail*. Ésta es una herramienta gratuita tal como el servidor *Web* Apache distribuida bajo licencia GNU (*General Public Licence* GPL).

- Servir a varios dominios, respaldar la información de las cuentas, crear grupos de distribución, entre otros, son tareas pendientes que se recomiendan estudiar para aprovechar de mejor manera al servidor de correo electrónico Kerio.

- En el presente proyecto se han registrado dos dominios para mostrar la operación del protocolo SSL, no se ha configurado un servidor DNS (*Domain Name Server*), por lo tanto, se recomienda configurar el servidor BIND propio de las distribuciones Linux Red Hat, con lo cual se debería registrar un solo dominio y gestionarlo según sea la necesidad de la solución a implementarse.

- Si bien es cierto, en el presente proyecto se han gestionado de una u otra manera los certificados digitales de los clientes, se recomienda profundizar en un estudio que implemente la infraestructura de una autoridad de certificación, la cual gestione de manera más automatizada la petición, generación, distribución, renovación y revocación de los certificados a los clientes. Se recomienda la investigación de OpenCA, el cual es un software gratuito que presta las facilidades antes mencionadas entre otras.

REFERENCIAS BIBLIOGRÁFICAS

[1] HUERTA, Antonio. Seguridad en Unix y Redes, Versión 2.1, Julio2002

[2] OLOVSSON Tomas, A Structured Approach to Computer Security. Department of Computer Engineering Chalmers University of Technology. Technical Report No 122, 1992

[3] CSI/FBI. Computer Crime And Security Survey. 2003

[4] TANENBAUM Andrew, Redes de Computadoras. Tercera Edición.

[5] IBM. A Comprehensive Guide to Virtual Private Networks, Volume I: IBM Firewall, Server and Client Solutions. Primera Edición. 1998

[6] STALLINGS, William. Data & Computer Communications. 6ta Edición. Prentice Hall. 2000.

[7] RED HAT INC. Red Hat Linux Customization Guide. Copyright© 2003

[8] RED HAT INC. Red Hat Linux Reference Guide. Copyright© 2003

ARTÍCULOS

[9] VERISIGN. Building an E-Commerce Trust Infrastructure. SSL Server Certificates and Online Payment Services

DIRECCIONES ELECTRÓNICAS

[10] NETSCAPE. http://wp.netscape.com/eng/ssl3/ssl-toc.html

[11] NETSCAPE.
http://developer.netscape.com/docs/manuals/security/sslin/contents.htm#1045623

[12]RSA SECURITY.
http://www.rsasecurity.com/standards/index.html

[13] NETSCAPE . http://wp.netscape.com/security/techbriefs/index.html

[14] GEOCITITES. http://www.geocities.com/jgarcia596/prot16.html

Anexo 1: Configuración Básica Servidor Seguro, Cliente y Servidor Autenticados

A continuación se muestra el archivo de configuración que permite la autenticación tanto del cliente como del servidor.

```
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs-2.0/> for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# The configuration directives are grouped into three basic sections
#  1. Directives that control the operation of the Apache server process as a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default' server,
#     which responds to requests that aren't handled by a virtual host.
#     These directives also provide default values for the settings
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/foo.log"
# with ServerRoot set to "/etc/httpd" will be interpreted by the
# server as "/etc/httpd/logs/foo.log".
#

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# Don't give away too much information about all the subcomponents
# we are running.  Comment out this line if you don't mind remote sites
# finding out what major optional modules you are running
ServerTokens OS

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at <URL:http://httpd.apache.org/docs-2.0/mod/core.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/etc/httpd"

#
# ScoreBoardFile: File used to store internal server process information.
# If unspecified (the default), the scoreboard will be stored in an
# anonymous shared memory segment, and will be unavailable to third-party
```

```
# applications
# If specified, ensure that no two invocations of Apache share the same
# scoreboard file. The scoreboard file MUST BE STORED ON A LOCAL DISK
#
#ScoreBoardFile run/httpd scorehoard

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile run/httpd pid

#
# Timeout: The number of seconds before receives and sends time out
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive Off

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection
#
KeepAliveTimeout 15

##
## Server-Pool Size Regulation (MPM specific)
##

# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are kept spare
# MaxSpareServers: maximum number of server processes which are kept spare
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
MaxClients       150
MaxRequestsPerChild  1000
</IfModule>

# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>

# perchild MPM
# NumServers: constant number of server processes
# StartThreads: initial number of worker threads in each server process
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
# MaxThreadsPerChild: maximum number of worker threads in each server process
```

```
# MaxRequestsPerChild: maximum number of connections per server proeess
<IfModule perchild.c>
NumServers          5
StartThreads        5
MinSpareThreads     5
MaxSpareThreads     10
MaxThreadsPerChild  20
MaxRequestsPerChild  0
</IfModule>

#
# Listen: Allows you to bind Apache to speeific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 80

#
# Load config files from the config directory "/etc/httpd/conf.d".
#
Include conf.d/*.conf

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule access_module modules/mod_access.so
LoadModule auth_module modules/mod_auth.so
LoadModule auth_anon_module modules/mod_auth_anon.so
LoadModule auth_dbm_module modules/mod_auth_dbm.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule env_module modules/mod_env.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule cern_meta_module modules/mod_cern_meta.so
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule asis_module modules/mod_asis.so
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule imap_module modules/mod_imap.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so

<IfModule prefork.c>
LoadModule cgi_module modules/mod_cgi.so
```

```
</IfModule>

<IfModule worker.c>
LoadModule cgid_module modules/mod_cgid.so
</IfModule>

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared memory as nobody, and the
#    suggested workaround is to create a user www and use that user.
#  NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
#  when the value of (unsigned)Group is above 60000,
#  don't use Group #-1 on these systems!
#
User apache
Group apache

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.  e.g. admin@your-domain.com
#
#ServerAdmin localhost@localhost.com

#
# ServerName gives the name and port that the server uses to identify itself
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If this is not set to valid DNS name for your host, server-generated
# redirections will not work.  See also the UseCanonicalName directive.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#
#ServerName new host name:80

#
# UseCanonicalName: Determines how Apache constructs self-referencing
# URLs and the SERVER_NAME and SERVER_PORT variables.
# When set "Off", Apache will use the Hostname and Port supplied
# by the client.  When set "On", Apache will use the value of the
# ServerName directive.
#
UseCanonicalName Off

#
# DocumentRoot. The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html/site"
```

```
#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories)
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/html">

#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI Multiviews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None

#
# Controls who can get stuff from this server
#
    Order allow,deny
    Allow from all

</Directory>

#
# Disable autoindex for the root directory, and present a
# default Welcome page if no other index page is present.
#
<LocationMatch "^/$>
    Options -Indexes
    ErrorDocument 403 /error/noindex.html
</LocationMatch>

#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
# The path to the end user account 'public_html' directory must be
# accessible to the webserver userid.  This usually means that ~userid
# must have permissions of 711, ~userid/public_html must have permissions
# of 755, and documents contained therein must be world-readable.
# Otherwise, the client will only receive a "403 Forbidden" message.
#
# See also: http://httpd.apache.org/docs/misc/FAQ.html#forbidden
#
```

```
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    UserDir disable

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disable" line above, and uncomment
    # the following line instead:
    #
    #UserDir public_html

</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only
#
#<Directory /home/*/public_html>
#    AllowOverride FileInfo AuthConfig Limit
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS>
#        Order allow,deny
#        Allow from all
#    </Limit>
#    <LimitExcept GET POST OPTIONS>
#        Order deny,allow
#        Deny from all
#    </LimitExcept>
#</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents.  The MultiViews Option can be used for the
# same purpose, but it is much slower
#
DirectoryIndex index.html

#index.html.var

#
# AccessFileName:  The name of the file to look for in each directory
# for access control information.  See also the AllowOverride directive.
#
AccessFileName  htaccess

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ~ "^\ ht">
    Order allow,deny
    Deny from all
</Files>

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found
#
TypesConfig /etc/mime.types

#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.  If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
```

```
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
    MIMEMagicFile conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here.  Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
# CustomLog logs/access_log common
CustomLog logs/access_log combined

#
# If you would like to have agent and referer logfiles, uncomment the
# following directives.
#
#CustomLog logs/referer_log referer
#CustomLog logs/agent_log agent

#
# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog logs/access_log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
```

```
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL.  So "/icons" isn't aliased in this
# example, only "/icons/".  If the fakename is slash-terminated, then the
# realname must also be slash terminated, and if the fakename omits the
# trailing slash, the realname must also omit it.
#
# We include the /icons/ alias for FancyIndexed directory listings.  If you
# do not use FancyIndexing, you may comment this out
#
Alias /icons/ "/var/www/icons/"

<Directory "/var/www/icons">
   Options Indexes MultiViews
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>

#
# This should be changed to the ServerRoot/manual/.  The alias provides
# the manual, even if you choose to move your DocumentRoot.  You may comment
# this out if you do not care for the documentation.
#
Alias /manual "/var/www/manual"

<Directory "/var/www/manual">
   Options Indexes FollowSymLinks MultiViews
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>

<IfModule mod_dav_fs.c>
   # Location of the WebDAV lock database.
   DAVLockDB /var/lib/dav/lockdb
</IfModule>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

<IfModule mod_cgid.c>
#
# Additional to mod_cgid.c settings, mod_cgid has Scriptsock <path>
# for setting UNIX socket for communicating with cgid
#
Scriptsock        run/httpd.cgid
</IfModule>

#
# "/var/www/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/var/www/cgi-bin">
   AllowOverride None
   Options None
   Order allow,deny
   Allow from all
</Directory>

#
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
```

```
# clients where to look for the relocated document
# Example:
# Redirect permanent /foo http://www.example.com/bar

#
# Directives controlling the display of server-generated directory listings.
#

#
# FancyIndexing is whether you want fancy directory indexing or standard.
# VersionSort is whether files containing version numbers should be
# compared in the natural way, so that `apache-1.3.9.tar' is placed before
# `apache-1.3.12.tar'
#
IndexOptions FancyIndexing VersionSort NameWidth=*

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions.  These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes.  These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes
ReadmeName README.html
HeaderName HEADER.html

#
```

```
# IndexIgnore is a set of filenames which directory indexing should ignore
# and not include in the listing.  Shell-style wildcarding is permitted
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

#
# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above
#
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

#
# DefaultLanguage and AddLanguage allows you to specify the language of
# a document.  You can then use content negotiation to give a browser a
# file in a language the user can understand
#
# Specify a default language.  This means that all data
# going out without a specific language tag (see below) will
# be marked with this one. You probably do NOT want to set
# this unless you are sure it is correct for all cases
#
# * It is generally better to not mark a page as
# * being a certain language than marking it with the wrong
# * language!
#
# DefaultLanguage nl
#
# Note 1: The suffix does not have to be the same as the language
# keyword --- those with documents in Polish (whose net-standard
# language code is pl) may wish to use "AddLanguage pl .po" to
# avoid the ambiguity with the common suffix for perl scripts
#
# Note 2: The example entries below illustrate that in some cases
# the two character 'Language' abbreviation is not identical to
# the two character 'Country' code for its country,
# E.g. 'Danmark/dk' versus 'Danish/da'.
#
# Note 3: In the case of 'ltz' we violate the RFC by using a three char
# specifier. There is 'work in progress' to fix this and get
# the reference data for rfc1766 cleaned up.
#
# Danish (da) - Dutch (nl) - English (en) - Estonian (et)
# French (fr) - German (de) - Greek-Modern (el)
# Italian (it) - Norwegian (no) - Norwegian Nynorsk (nn) - Korean (kr)
# Portugese (pt) - Luxembourgeois* (ltz)
# Spanish (es) - Swedish (sv) - Catalan (ca) - Czech(cz)
# Polish (pl) - Brazilian Portuguese (pt-br) - Japanese (ja)
# Russian (ru) - Croatian (hr)
#
AddLanguage da .dk
AddLanguage nl .nl
AddLanguage en .en
AddLanguage et .et
AddLanguage fr .fr
AddLanguage de .de
AddLanguage he .he
AddLanguage el .el
AddLanguage it .it
AddLanguage ja .ja
AddLanguage pl .po
AddLanguage kr .kr
AddLanguage pt .pt
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pt-br .pt-br
AddLanguage ltz .ltz
AddLanguage ca .ca
AddLanguage es .es
AddLanguage sv .sc
AddLanguage cz .cz
AddLanguage ru .ru
AddLanguage tw .tw
AddLanguage zh-tw .tw
```

```
AddLanguage hr .hr

#
# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
#
# Just list the languages in decreasing order of preference. We have
# more or less alphabetized them here. You probably want to change this.
#
LanguagePriority en da nl et fr de el it ja kr no pl pt pt-br ltz ca es sv tw

#
# ForceLanguagePriority allows you to serve a result page rather than
# MULTIPLE CHOICES (Prefer) [in case of a tie] or NOT ACCEPTABLE (Fallback)
# [in case no accepted languages matched the available variants]
#
ForceLanguagePriority Prefer Fallback

#
# Specify a default charset for all pages sent out. This is
# always a good idea and opens the door for future internationalisation
# of your web site, should you ever want it. Specifying it as
# a default does little harm, as the standard dictates that a page
# is in iso-8859-1 (latin1) unless specified otherwise i.e. you
# are merely stating the obvious. There are also some security
# reasons in browsers, related to javascript and URL parsing
# which encourage you to always set a default char set
#
AddDefaultCharset ISO-8859-1

#
# Commonly used filename extensions to character sets. You probably
# want to avoid clashes with the language extensions, unless you
# are good at carefully testing your setup after each change.
# See ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets for
# the official list of charset names and their respective RFCs
#
AddCharset ISO-8859-1  iso8859-1  .latin1
AddCharset ISO-8859-2  .iso8859-2  .latin2 .cen
AddCharset ISO-8859-3  .iso8859-3  .latin3
AddCharset ISO-8859-4  .iso8859-4  latin4
AddCharset ISO-8859-5  .iso8859-5  latin5 .cyr .iso-ru
AddCharset ISO-8859-6  .iso8859-6  .latin6 .arb
AddCharset ISO-8859-7  iso8859-7  .latin7 .grk
AddCharset ISO-8859-8  .iso8859-8  .latin8 .heb
AddCharset ISO-8859-9  .iso8859-9  .latin9 .trk
AddCharset ISO-2022-JP .iso2022-jp .jis
AddCharset ISO-2022-KR .iso2022-kr .kis
AddCharset ISO-2022-CN .iso2022-cn .cis
AddCharset Big5      .Big5      .big5
# For russian, more than one charset is used (depends on client, mostly):
AddCharset WINDOWS-1251 .cp-1251  .win-1251
AddCharset CP866      cp866
AddCharset KOI8-r     .koi8-r .koi8-ru
AddCharset KOI8-ru    koi8-uk .ua
AddCharset ISO-10646-UCS-2 .ucs2
AddCharset ISO-10646-UCS-4 .ucs4
AddCharset UTF-8      .utf8

# The set below does not map to a specific (iso) standard
# but works on a fairly wide range of browsers. Note that
# capitalization actually matters (it should not, but it
# does for some browsers).
#
# See ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets
# for a list of sorts. But browsers support few
#
AddCharset GB2312     .gb2312  gb
AddCharset utf-7      .utf7
AddCharset utf-8      .utf8
AddCharset big5       .big5 .b5
AddCharset EUC-TW     .euc-tw
AddCharset EUC-JP     .euc-jp
AddCharset EUC-KR     .euc-kr
AddCharset shift_jis  .sjis
```

```
#
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
AddType application/x-tar .tgz

#
# AddHandler allows you to map certain file extensions to "handlers":
# actions unrelated to filetype. These can be either built into the server
# or added with the Action direetive (see below)
#
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
#
#AddHandler cgi-script .cgi

#
# For files that include their own HTTP headers:
#
#AddHandler send-as-is asis

#
# For server-parsed imagemap files:
#
AddHandler imap-file map

#
# For type maps (negotiated resources):
# (This is enabled by default to allow the Apache "It Worked" page
#  to be distributed in multiple languages.)
#
AddHandler type-map var

# Filters allow you to process content before it is sent to the client.
#
# To parse .shtml files for server-side includes (SSI)
# (You will also need to add "Includes" to the "Options" directive.)
#
AddOutputFilter INCLUDES .shtml

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples.
#ErrorDocument 500 "The server made a boo boo "
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#

#
# Putting this all together, we can Internationalize error responses.
#
# We use Alias to redirect any /error/HTTP_<error> html var response to
# our collection of by-error message multi-language collections.  We use
# includes to substitute the appropriate text.
#
# You can modify the messages' appearance without changing any of the
# default HTTP_<error>.html.var files by adding the line;
#
#   Alias /error/include/ "/your/include/path/"
#
# which allows you to create your own set of files by starting with the
# /var/www/error/include/ files and
# copying them to /your/include/path/, even on a per-VirtualHost basis.
#
```

```
Alias /error/ "/var/www/error/"

<IfModule mod_negotiation.c>
<IfModule mod_include.c>
    <Directory "/var/www/error">
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en cs de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>

    ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
    ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
    ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
    ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
    ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
    ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
    ErrorDocument 410 /error/HTTP_GONE.html.var
    ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
    ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
    ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
    ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
    ErrorDocument 415 /error/HTTP_SERVICE_UNAVAILABLE.html.var
    ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
    ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
    ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
    ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
    ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var

</IfModule>
</IfModule>

#
# The following directives modify normal HTTP response behavior to
# handle known problems with browser implementations
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

#
# The following directive disables redirects on non-GET requests for
# a directory that does not include the trailing slash.  This fixes a
# problem with Microsoft WebFolders which does not appropriately handle
# redirects for folders with DAV methods
#
BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your-domain.com" to match your domain to enable.
#
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .your-domain.com
#</Location>

#
# Allow remote server configuration reports, with the URL of
#  http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your-domain.com" to match your domain to enable.
#
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .your-domain.com
```

```
#</Location>

#
# Proxy Server directives  Uncomment the following lines to
# enable the proxy server
#
#<IfModule mod_proxy.c>
#ProxyRequests On
#
#<Proxy *>
#    Order deny,allow
#    Deny from all
#    Allow from  your-domain.com
#</Proxy>

#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of  Off | On | Full | Block
#
#ProxyVia On

#
# To enable the cache as well, edit and uncomment the following lines.
# (no cacheing without CacheRoot)
#
#CacheRoot "/etc/httpd/proxy"
#CacheSize 5
#CacheGcInterval 4
#CacheMaxExpire 24
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache a-domain.com another-domain.edu joes.garage-sale.com

#</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them. Most configurations
# use only name-based virtual hosts so the server doesn't need to worry about
# IP addresses  This is indicated by the asterisks in the directives below.
#
# Please see the documentation at
# <URL.http://httpd.apache.org/docs-2.0/vhosts/>
# for further details before you try to setup virtual hosts.
#
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# Use name-based virtual hosting.
#
#NameVirtualHost *

#
# VirtualHost example.
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for requests without a known
# server name.
#
#<VirtualHost *>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example com
#    ErrorLog logs/dummy-host.example com-error_log
#    CustomLog logs/dummy-host.example com-aecess_log common
#</VirtualHost>

# Virtual host server-networkingsslepn fadlan.com

<VirtualHost 200.107 6 169:443>
DocumentRoot /var/www/html/site
ServerAdmin edu_cito@hotmail com
ServerName server-networkingsslepn fadlan com
```

Anexo 2: Pruebas de Acceso al Sitio *Web* Seguro

*Este anexo guía ha sido elaborado con la colaboración del *Microsoft Internet Explorer*.

Al solicitar el URL: https://server-networkingsslepn.fadlan.com el servidor pide al usuario que presente un certificado digital para ser autenticado (Figura AN2.1).

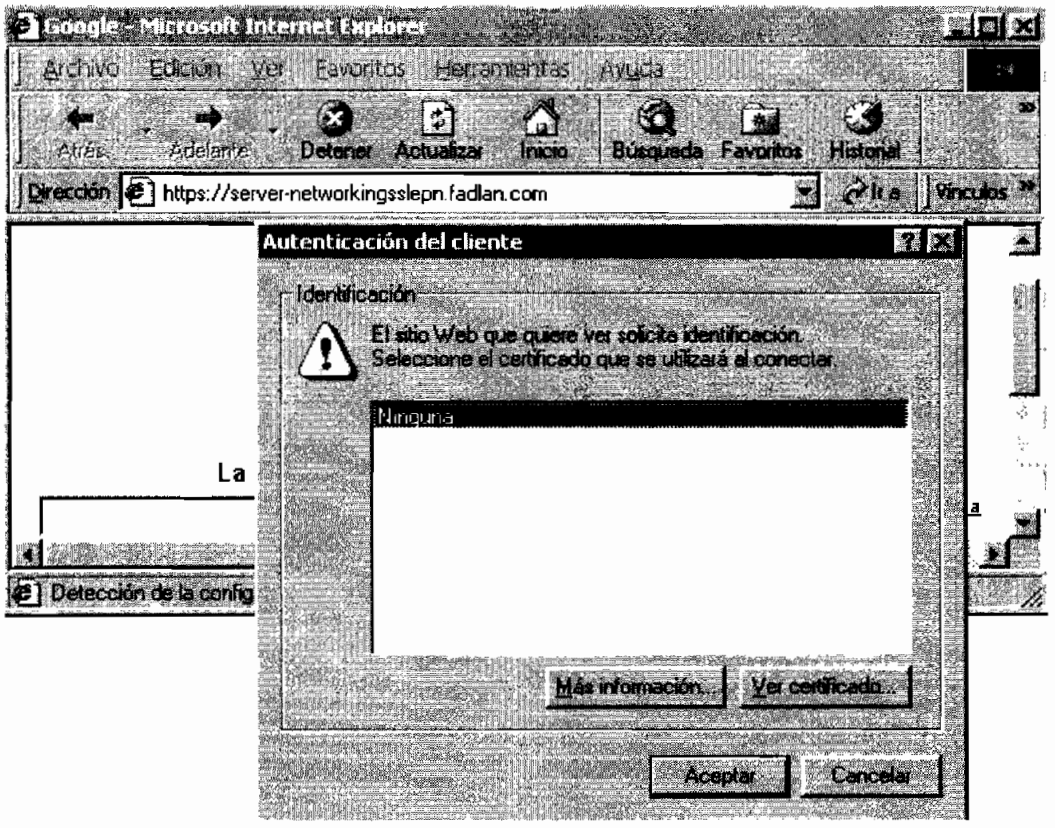


Figura AN2.1 Selección de Certificado

Haciendo *click* en el botón "Ver certificado" se puede visualizar el certificado digital que se va a presentar al servidor para efectos de autenticación obteniéndose los resultados de las figuras AN2.2 y AN2.3.

Luego de la autenticación del usuario, el servidor permite el acceso al sitio, la sesión se establece según los parámetros negociados en la fase de *Handshake* de SSL y finalmente se muestra la página principal del sitio (figura AN2.4).

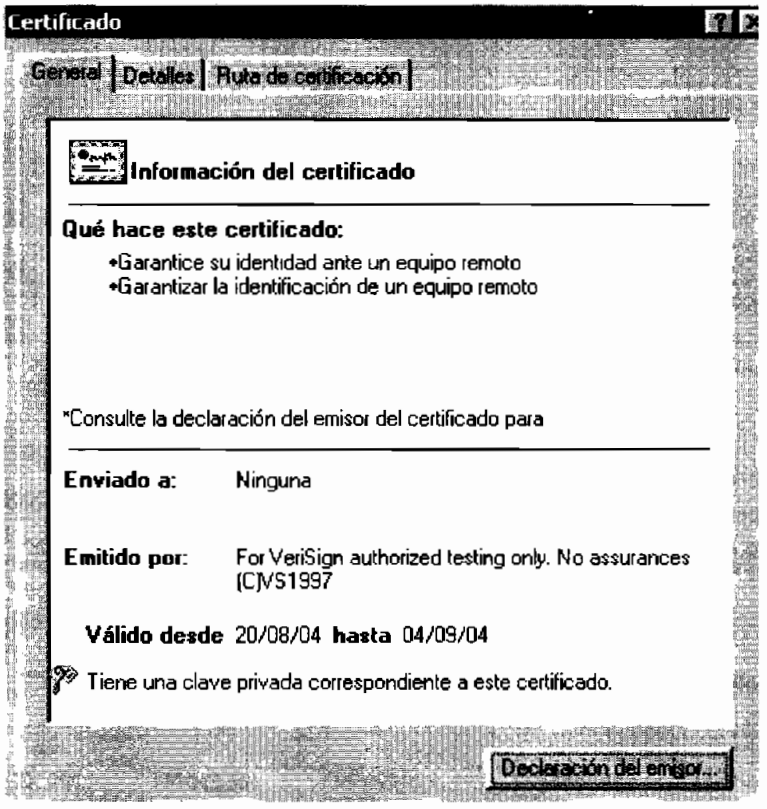


Figura AN2.2 Certificado de Cliente (1)

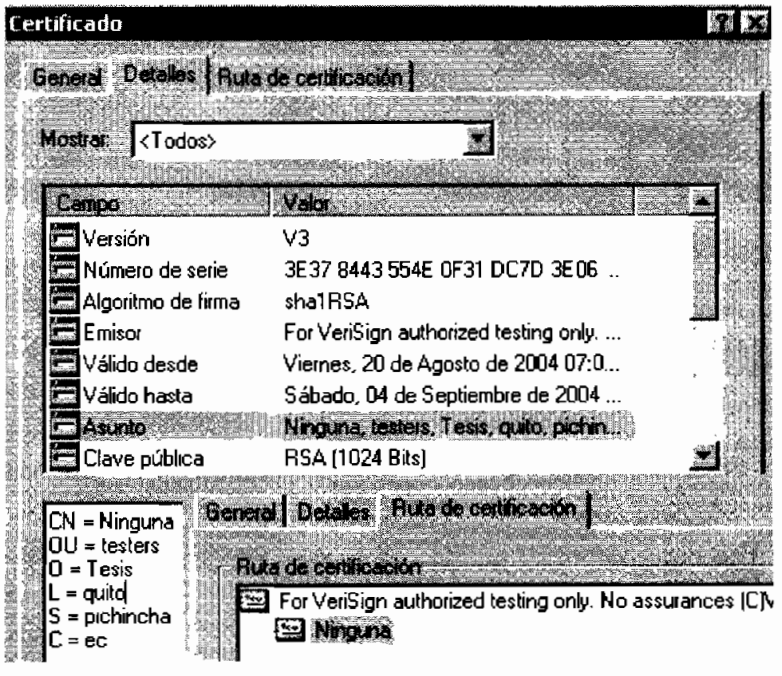


Figura AN2.3 Certificado de Cliente (2)

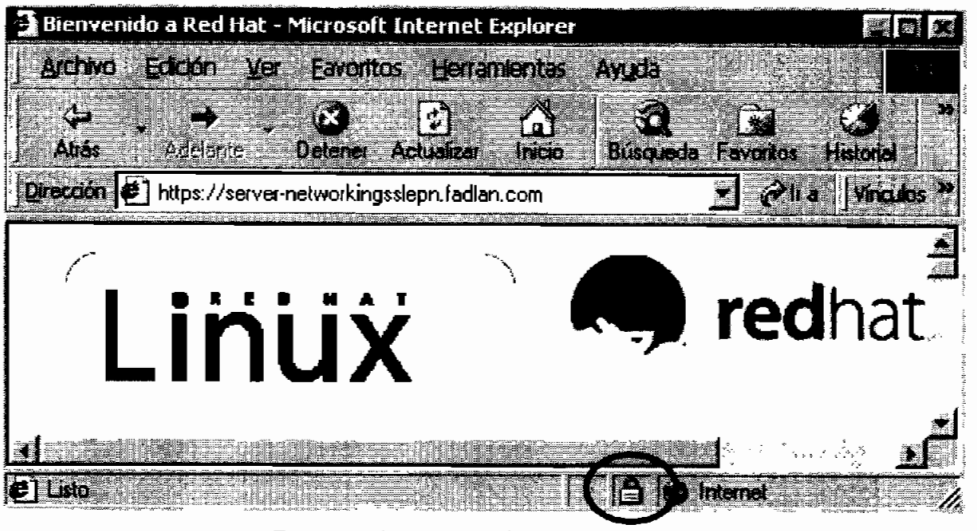


Figura AN2.4 Acceso al Sitio Web Seguro

La figura AN2.5 muestra el certificado digital del servidor al cual puede accederse haciendo *clic* sobre el candado ubicado en la parte inferior derecha de la página web.

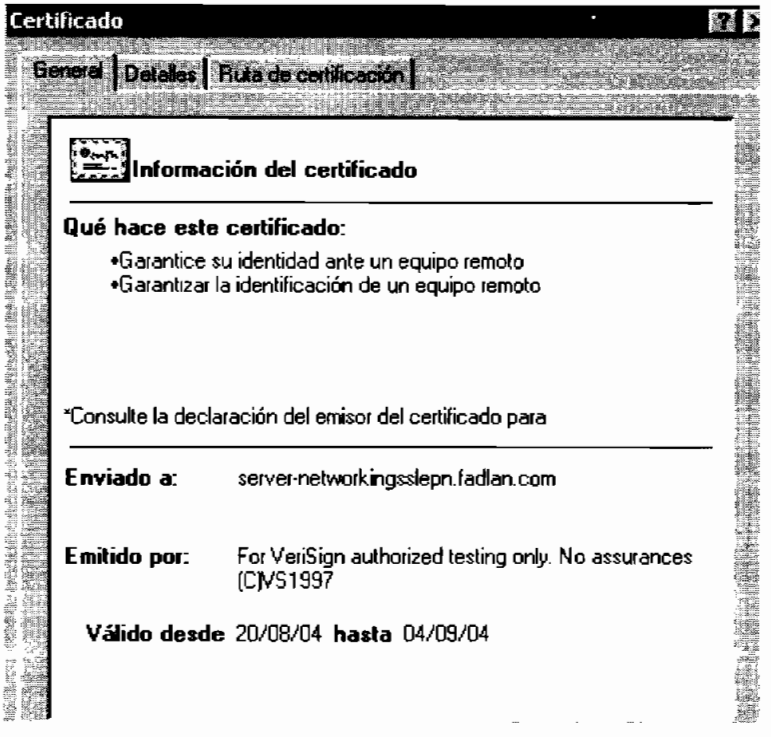


Figura AN2.5 Certificado Digital de Servidor

Anexo 3: Asistente de Configuración *Kerio Mail Server*

El presente anexo muestra la configuración inicial de *Kerio Mail Server* haciendo uso del asistente que se inicia al ejecutar el comando "*./cfgwizard*". desde una ventana de *terminal* del sistema operativo.

Inicialmente el asistente indica que guiará en las configuraciones más escenciales (figura AN3.1).

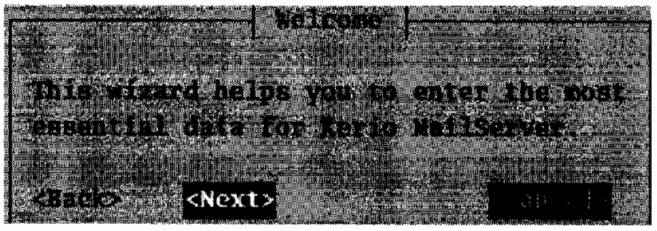


Figura AN3.1 Configuración Kerio Mail Server (paso1)

Acto seguido el asistente solicita el nombre del dominio al cual se desea dar servicio inicialmente; posteriormente desde la consola de administración pueden ser agregados dominios extras. En este caso el servidor de correo electrónico prestará sus servicios al dominio "networkingsslepn.fadlan.com" (figura AN3.2), el cual ha sido previamente registrado siguiendo el procedimiento descrito en la sección 3.2.5 del capítulo tres.

Luego de ingresar el nombre del dominio registrado, el asistente solicitará el nombre de usuario y contraseña del administrador del servidor de correo electrónico (figura AN3.3).

Finalmente, en la figura AN3.4 el asistente solicita un directorio el cual servirá para almacenar la información de las cuentas de los usuarios configurados.

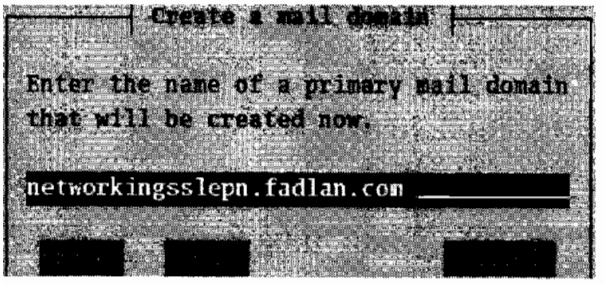


Figura AN3.2 Configuración Kerio Mail Server (paso2)

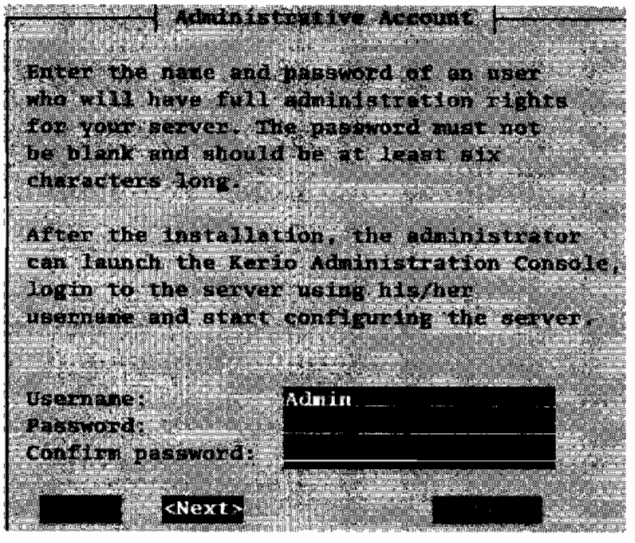


Figura AN3.3 Configuración Kerio Mail Server (paso3)

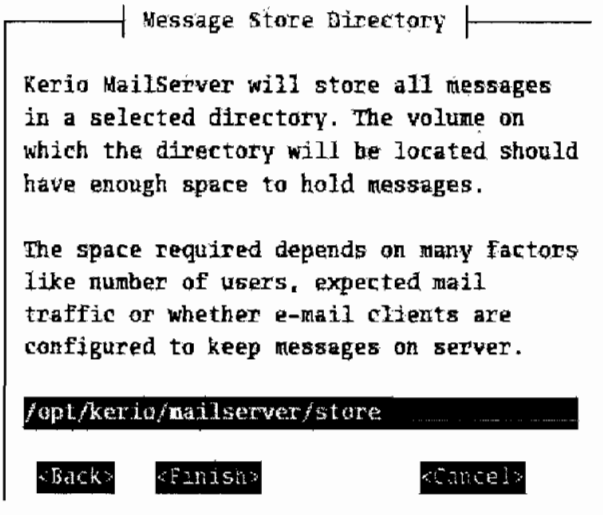


Figura 3.15 Configuración Kerio Mail Server (paso4)

Anexo 4: Configuración de Cuentas de Usuarios

Para configurar cuentas de usuarios se deben seguir los siguientes pasos:

1. En la pestaña *Users* hacer click en *add* para ingresar un nuevo usuario (figura AN4.1) e ingresar los datos del mismo tales como (figura AN4.2):

- Nombre de inicio de Sesión
- Nombre Completo
- Descripción
- *Password* inicial (el usuario puede cambiarlo en su primera sesión)

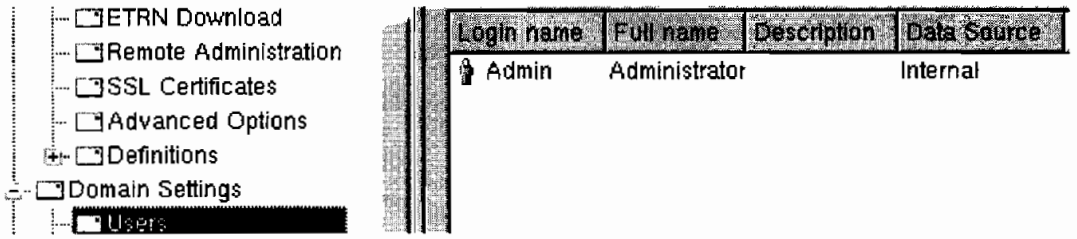


Figura AN4.1 Nuevo Usuario

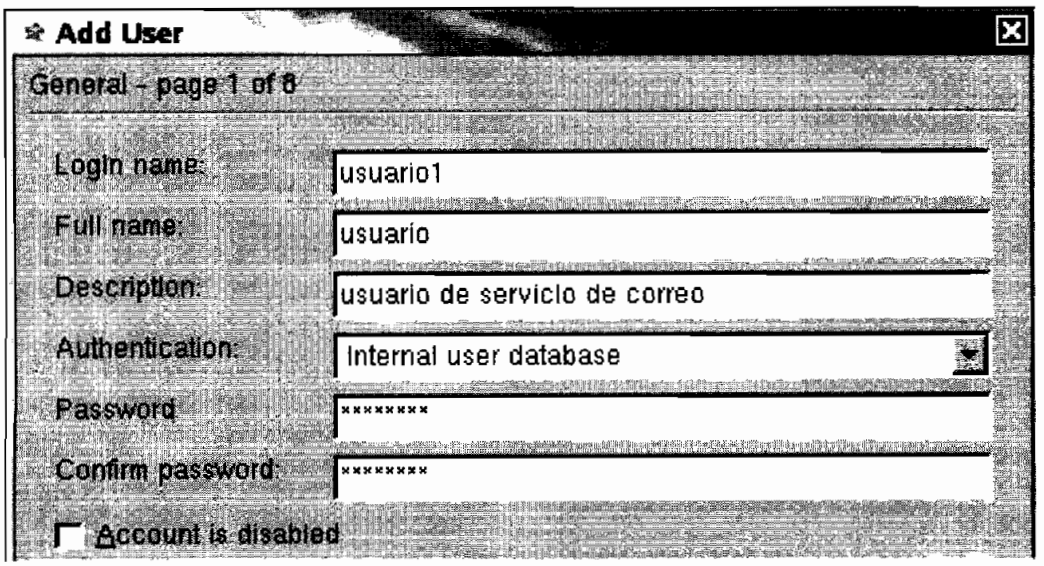


Figura AN4.2 Datos de Usuario

2. En las figura AN4.3 se observa la dirección de correo electrónico que tendrá el usuario. Mientras que en la figura AN4.4 se configura la cuota de

disco en el servidor que tendrá disponible el usuario para almacenamiento de sus mensajes de correo electrónico.

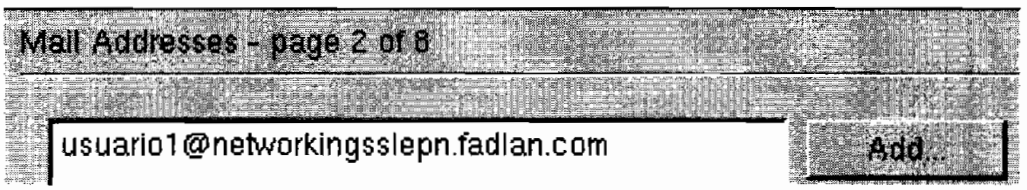


Figura AN4.3 Dirección de Correo

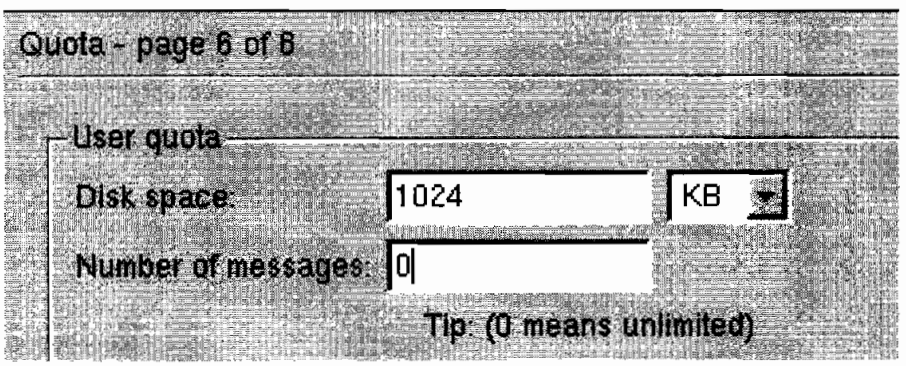


Figura AN4.4 Configuración de Cuota de disco

Anexo 5: Configuración de *Mozilla Mail & News Groups*

Como se puede observar en la figura AN5.1, para que *Mozilla* trabaje únicamente con SSL v3.0 se debe configurar lo siguiente.

1. Hacer *click* en el menú Edición y luego en Preferencias.
2. Ubicarse en las opciones de Seguridad avanzadas, SSL.
3. Seleccionar únicamente SSL v3.0

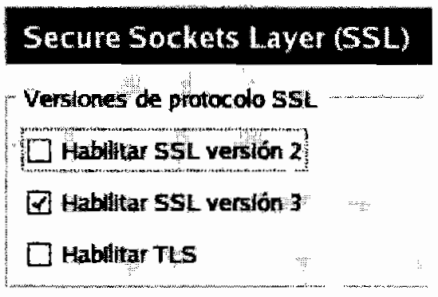


Figura AN5.1 Selección SSL v3.0

Para instalar el CD de la AC se deben seguir los siguientes pasos:

1. En las opciones de Seguridad avanzadas, seleccionar Certificados Digitales
2. Dirigirse hacia la pestaña de Autoridades
3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital de la Autoridad Certificadora en la cual se desea confiar.
4. Seleccionar los propósitos para los cuales se desea confiar en la AC (figura AN5.2).

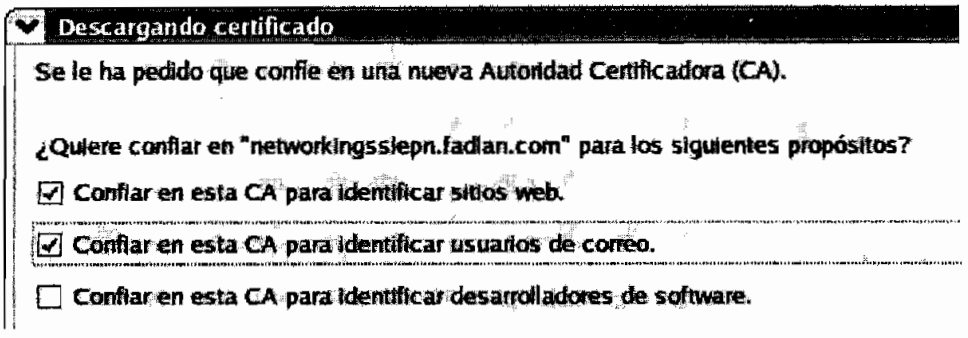


Figura AN5.2 Instalación de CD de AC

5. Luego se puede verificar la instalación correcta del CD (figura AN5.3)

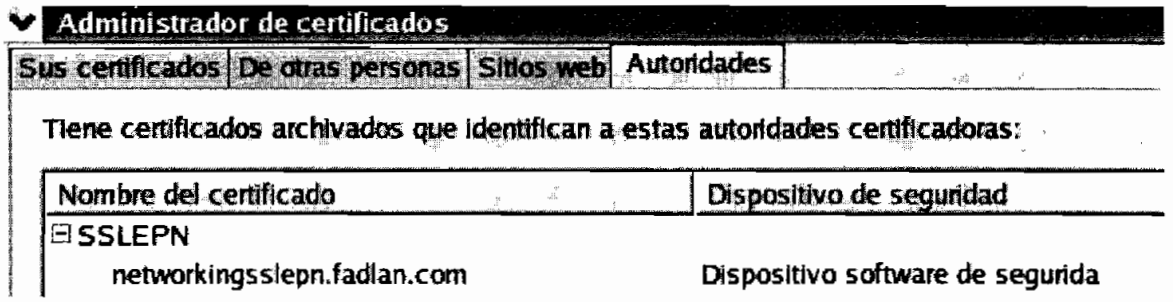


Figura AN5.3 CD de AC instalado

6. Se visualiza finalmente el contenido del CD importado (figura 5.4).

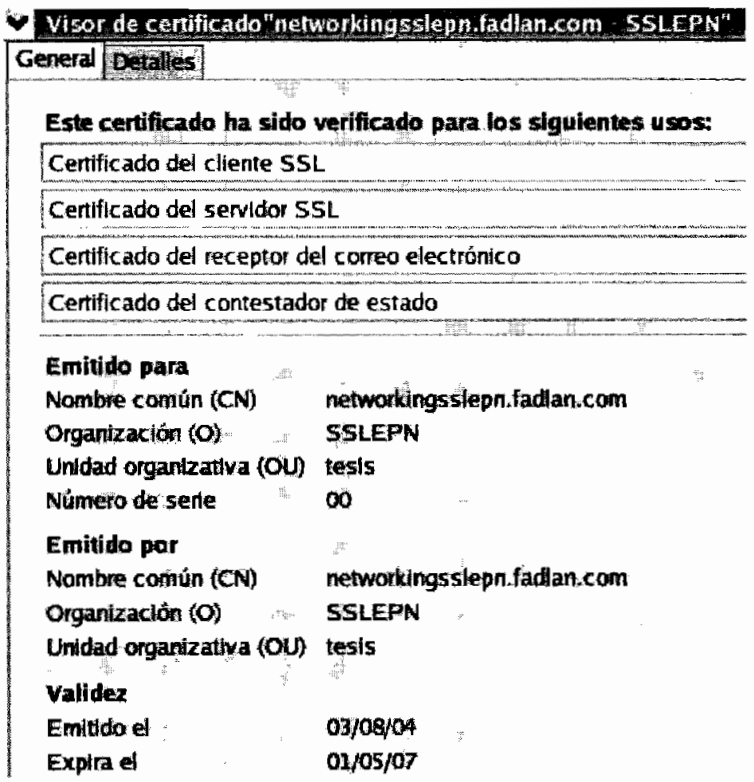


Figura AN5.4 Certificado Digital de Autoridad Certificadora

Para instalar el CD del cliente se deben seguir los siguientes pasos:

1. En las opciones de Seguridad avanzadas, seleccionar Certificados Digitales.
2. Dirigirse hacia la pestaña Sus Certificados.

3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital del cliente y verificar la importación de dicho CD (figura AN5.5)

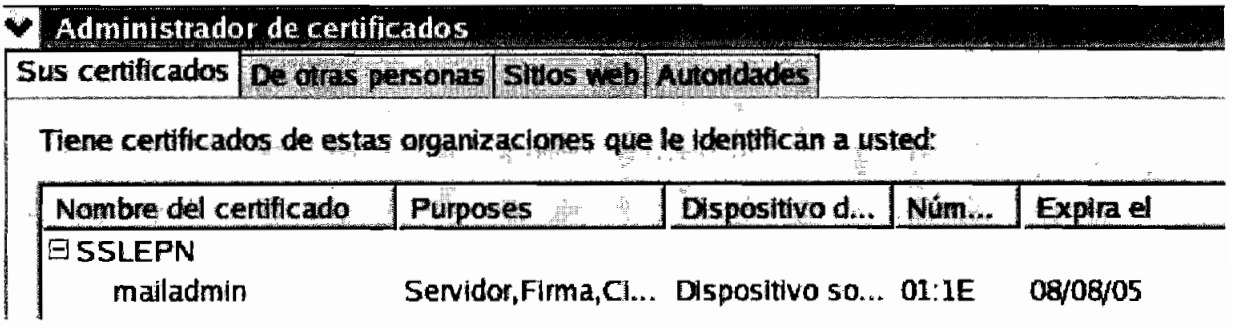


Figura AN5.5 CD de Cliente Instalado

4. Se visualiza finalmente el contenido del CD importado (figura 5.6).

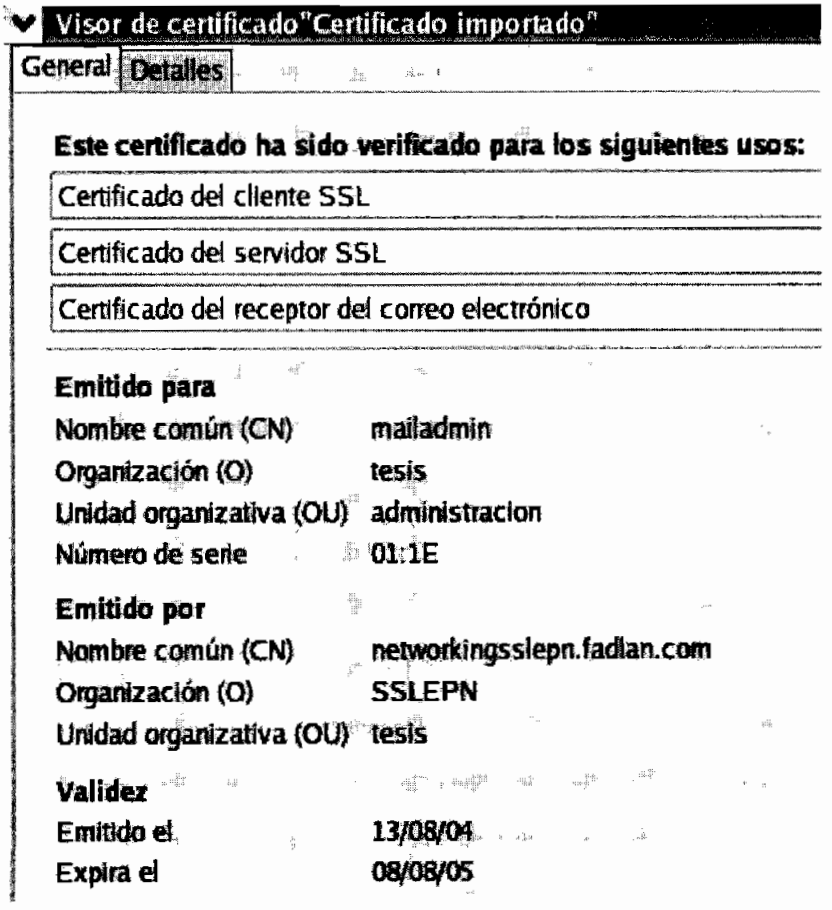


Figura AN5.6 Certificado Digital de Cliente

Anexo 6: Configuración de *Netscape Mail & News Groups*

La figura AN6.1 muestra la configuración necesaria que de sebe seguir en *Netscape* para que el navegador trabaje únicamente con SSL v3.0.

1. Hacer *click* en el menú Edición y luego en Preferencias.
2. Ubicarse en las opciones de Seguridad avanzadas, SSL.
3. Seleccionar únicamente SSL v3.0

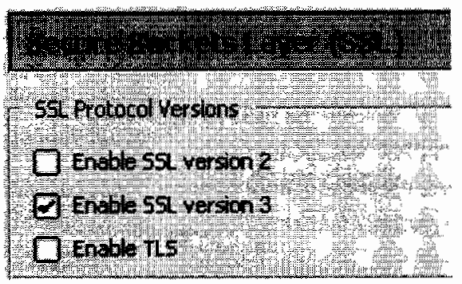


Figura AN6.1 Selección SSL v3.0

Para instalar el CD de la AC se deben seguir los siguientes pasos:

1. En las opciones de Seguridad avanzadas, seleccionar Certificados Digitales
2. Dirigirse hacia la pestaña de *Autorities*.
3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital de la Autoridad Certificadora en la cuál se desea confiar.
4. Seleccionar los propósitos para los cuales se desea confiar en la AC (figura AN6.2).

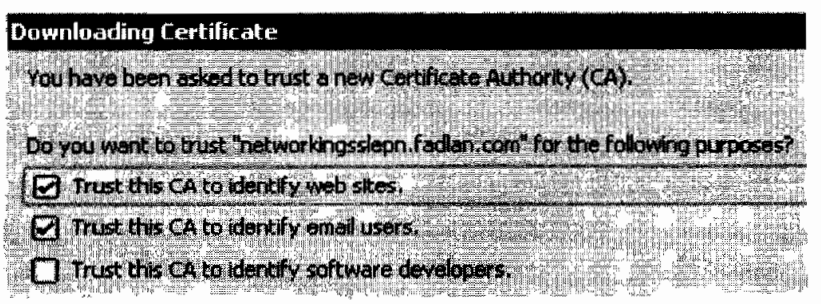


Figura AN6.2 Instalación de CD de AC

5. Luego se puede verificar la instalación correcta del CD (figura AN6.3)

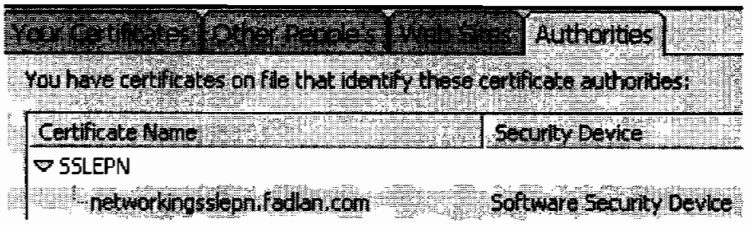


Figura AN6.3 CD de AC instalado

6. Se visualiza finalmente el contenido del CD importado (figura 6.4).

Figura AN6.4 Certificado Digital de Autoridad Certificadora

Para instalar el CD del cliente se deben seguir los siguientes pasos:

1. En las opciones de Seguridad avanzadas, seleccionar Certificados Digitales.
2. Dirigirse hacia la pestaña *Your Certificates*.
3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital del cliente y verificar la importación de dicho CD (figura AN6.5)

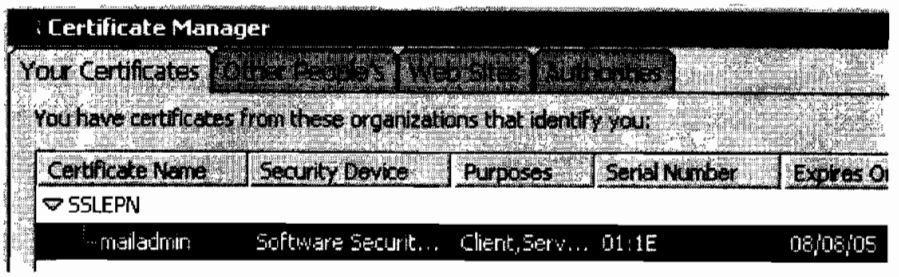


Figura AN6.5 CD de Cliente Instalado

4. Se visualiza finalmente el contenido del CD importado (figura 6.6).

Figura AN6.6 Certificado Digital de Cliente

Anexo 7: Configuración de *Microsoft Internet Explorer*

1. Hacer click en el menú Herramientas y luego en Opciones de Internet.
2. Ubicarse en la pestaña Opciones Avanzadas.
3. Seleccionar únicamente SSL v3.0 (figura AN7.1).

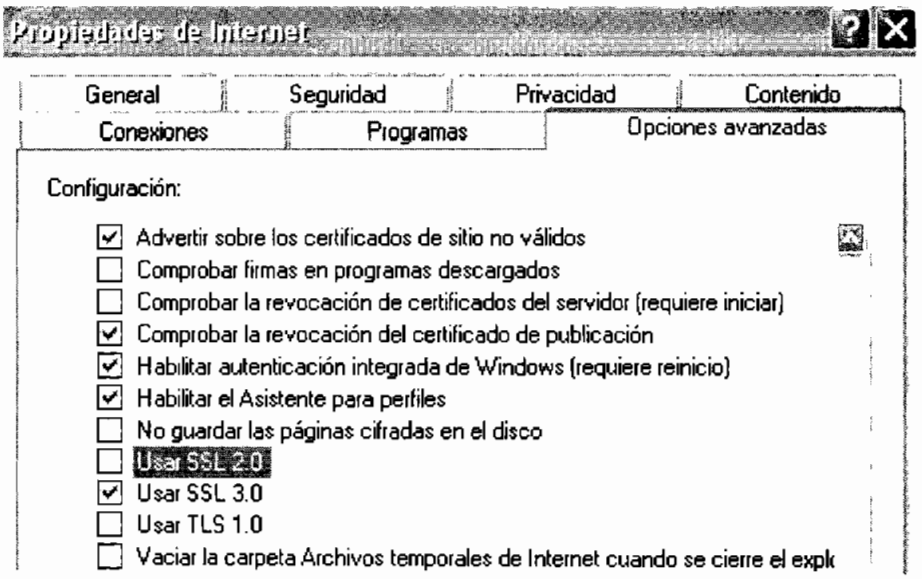


Figura AN7.1 Selección SSL v3.0

Para instalar el CD de la AC se deben seguir los siguientes pasos:

1. En la pestaña Contenido, seleccionar Certificados.
2. Hacer click sobre la pestaña de Entidades Emisoras Raíz de Confianza.
3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital de la Autoridad Certificadora en la cuál se desea confiar.
4. Luego se puede verificar la instalación correcta del CD (figura AN7.2)

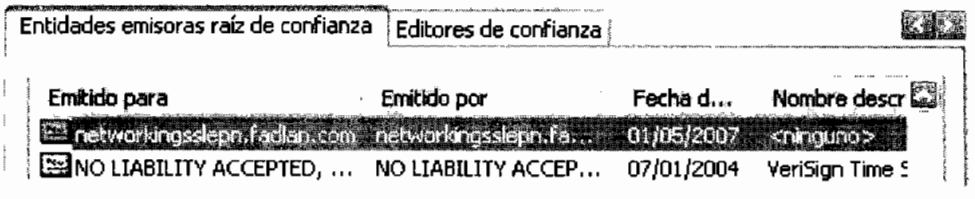


Figura AN7.2 CD de AC instalado

5. Se visualiza finalmente el contenido del CD importado (figura 7.3).

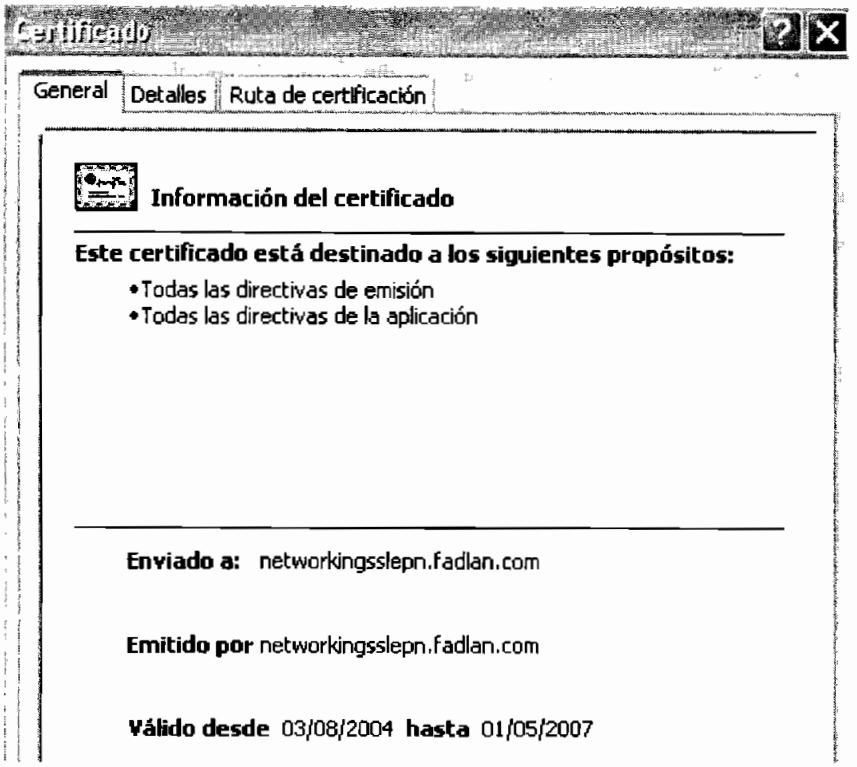


Figura AN7.3 Certificado Digital de Autoridad Certificadora

Para instalar el CD del cliente se deben seguir los siguientes pasos:

1. En las opciones de Seguridad avanzadas, seleccionar Certificados Digitales.
2. Dirigirse hacia la pestaña Personal.
3. Seleccionar, Importar y dirigirse hacia la ubicación donde se encuentre ubicado el Certificado Digital del cliente y verificar la importación de dicho CD (figura AN7.4)

Figura AN7.4 CD de Cliente Instalado

4. Se visualiza finalmente el contenido del CD importado (figura 7.5).

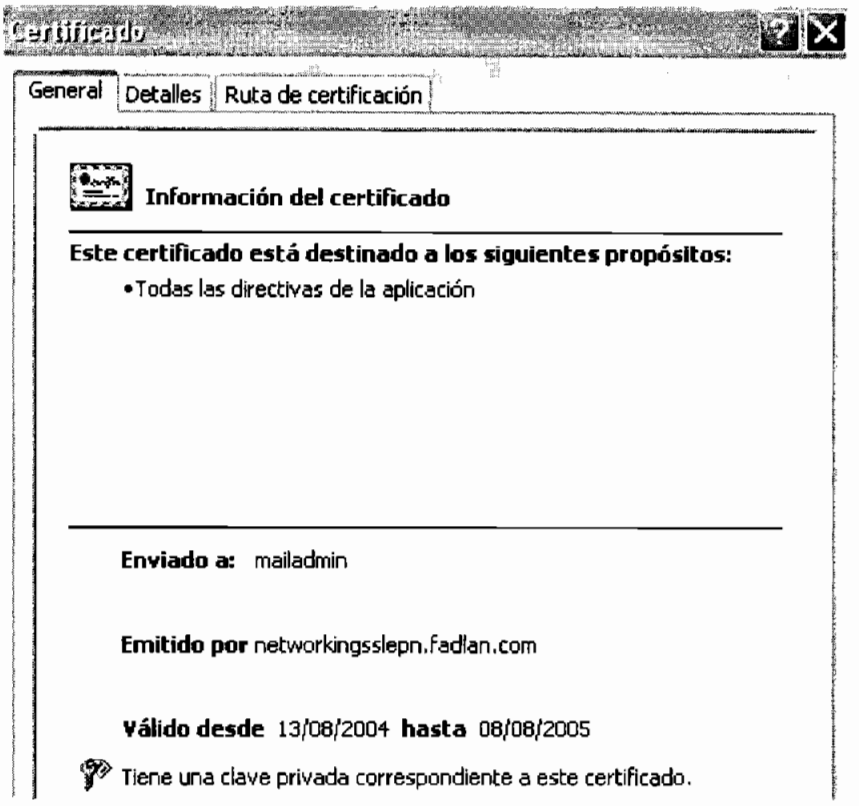


Figura AN7.5 Certificado Digital de Cliente

Anexo 8: Configuración y prueba de Correo Electrónico con *Mozilla Mail & News Groups*

Se deben seguir los siguientes pasos para configurar una cuenta de correo en *Netscape Mail & News Groups*:

1. Seleccionar configurar Nueva Cuenta desde el menú Edición.
2. Seleccionar Cuenta de Correo Electrónico (Figura AN8.1)

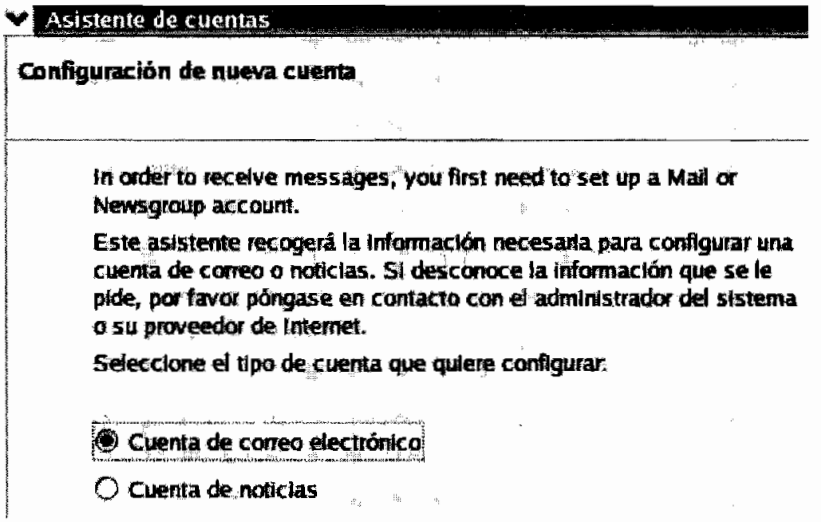


Figura AN8.1 Configuración de Cuenta (1)

3. Ingresar el nombre del dueño de la cuenta así como la dirección de correo electrónico que se está configurando (Figura AN8.2)

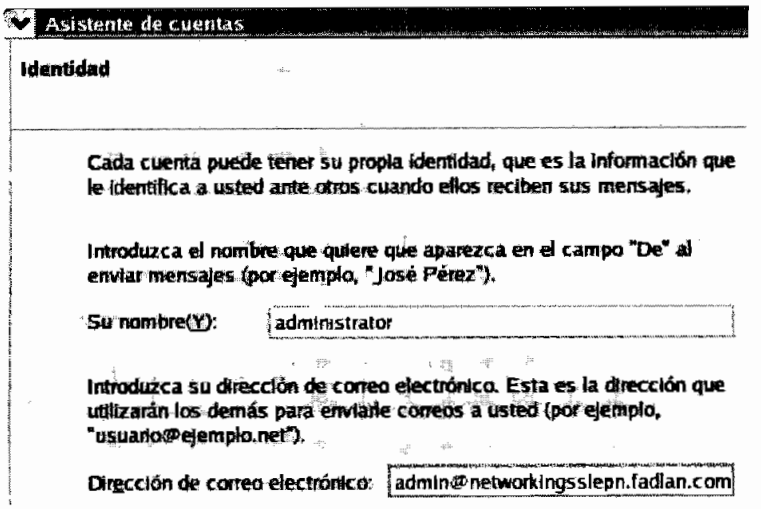


Figura AN8.2 Configuración de Cuenta (2)

4. Ingresar los datos del servidor de correo POP (Figura AN8.3)

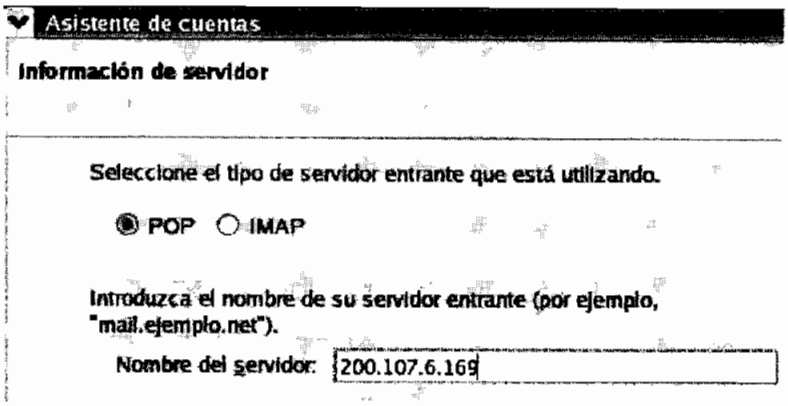


Figura AN8.3 Configuración de Cuenta (3)

5. Ingresar el nombre de la cuenta entregada por el administrador de correo electrónico (Figura AN8.4)

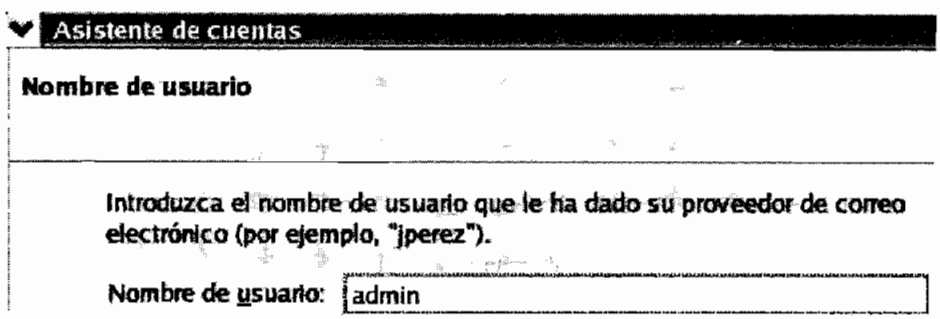


Figura AN8.4 Configuración de Cuenta (4)

6. Ingresar un nombre con el cual se desea referir a esta cuenta de correo electrónico (Figura AN8.5)

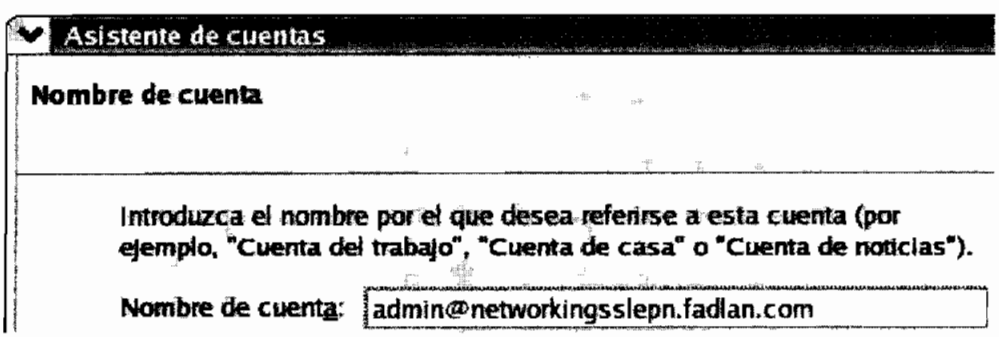


Figura AN8.5 Configuración de Cuenta (5)

7. Verificar los datos ingresados para la Configuración del correo electrónico

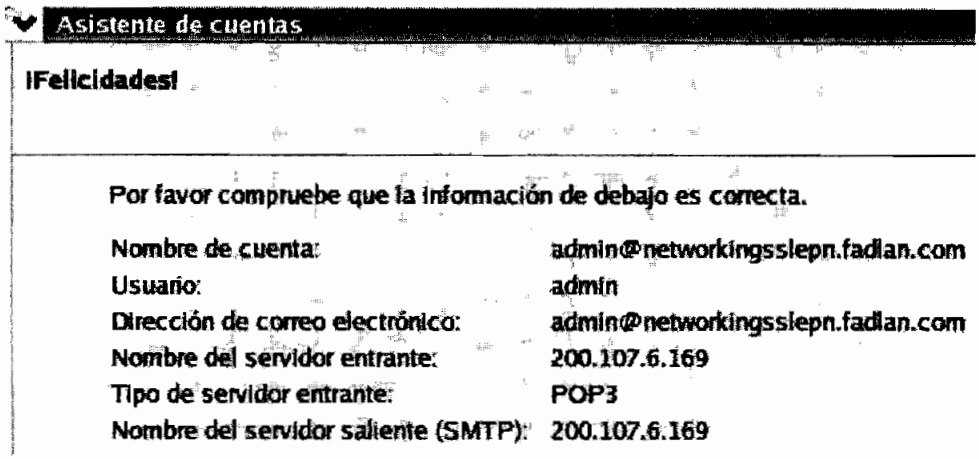


Figura AN8.6 Configuración de Cuenta (6)

Luego es necesario configurar la cuenta de correo para que haga uso del certificado digital de cliente instalado.

1. En el menú Edición, Preferencias, en la opción de Seguridad (Figura AN8.7).

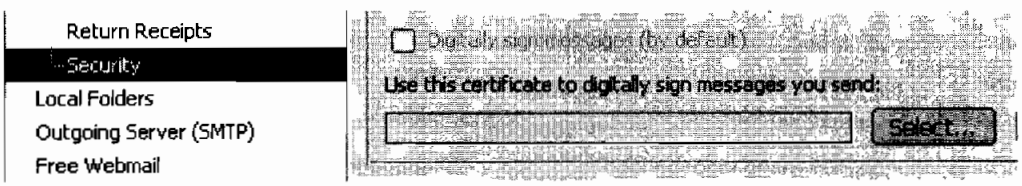


Figura AN8.7 Configuración de Cuenta (7)

2. Seleccionar el certificado digital que se desea usar (Figura AN8.8)

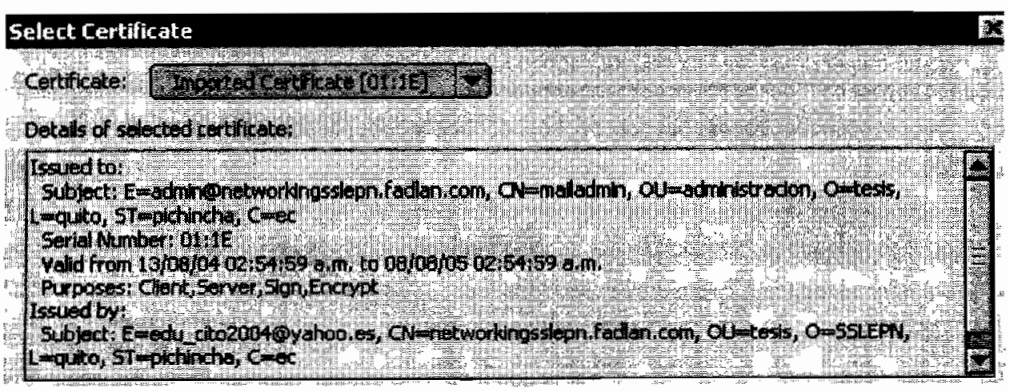


Figura AN8 8 Configuración de Cuenta (8)

Finalmente se muestran las pruebas realizadas de envío de mensajes de correo electrónico.

1. Mensaje sin cifrar ni firmar (Figura AN8.9)

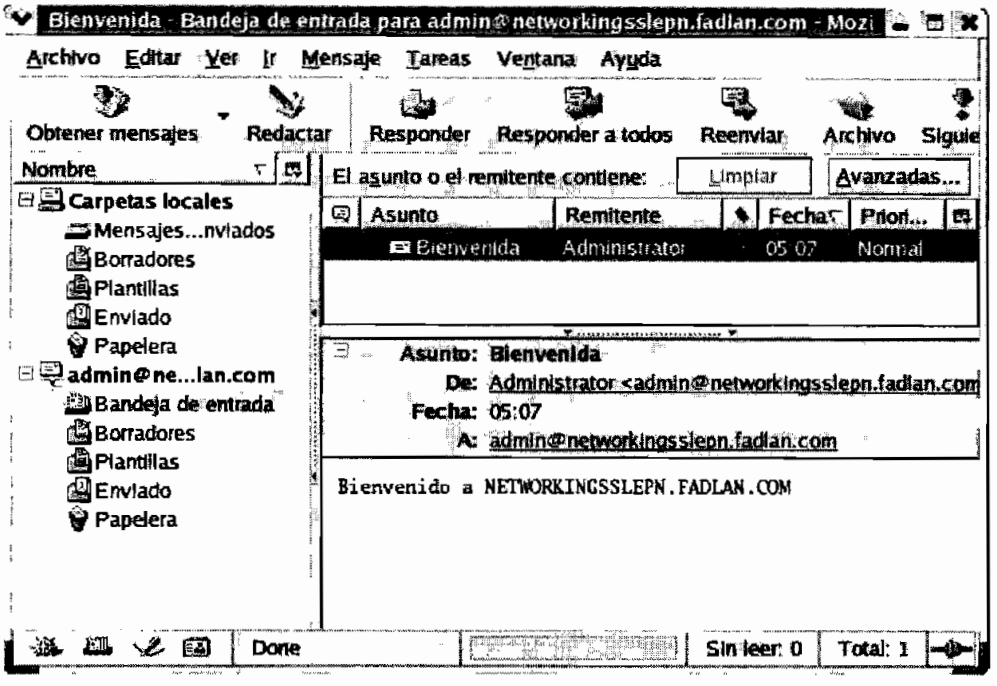


Figura AN8.9 Prueba Texto Plano

2. Envío de un mensaje con encripción y firma digital (Figura AN8.10).
   Se puede observar que es necesario que se seleccionen las opciones de encripción y firma digital.

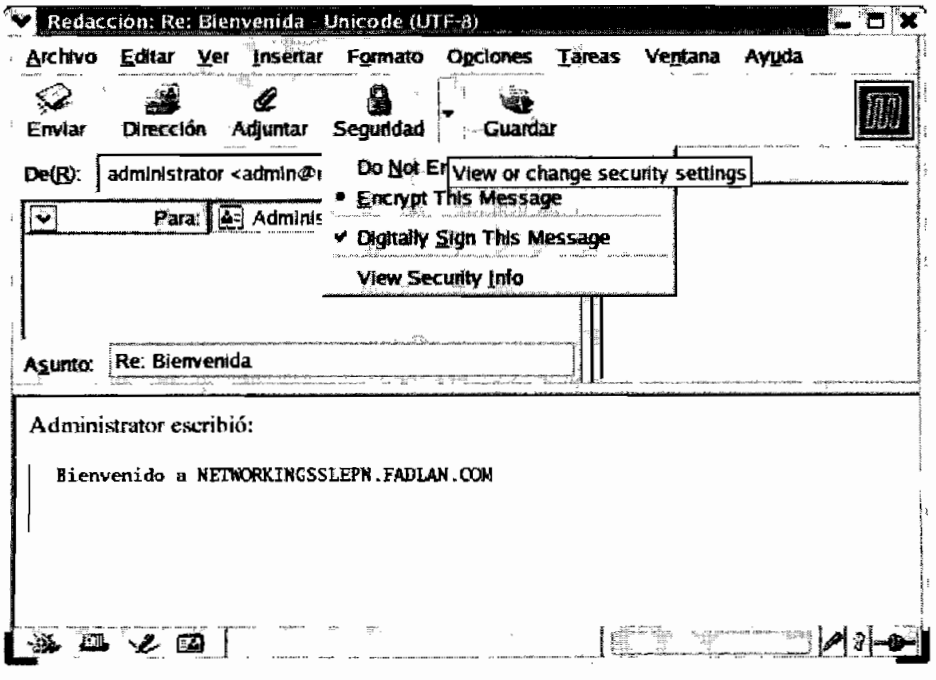


Figura AN8.10 Prueba Texto Encriptado y Firmado (envío)

3. Recepción del mensaje de correo electrónico encriptado y firmado (Figura AN8.11)

   Junto al destinatario y remitente se pueden observar los gráficos de una llave que representa la encripción y el de un bolígrafo que representa la firma digital.

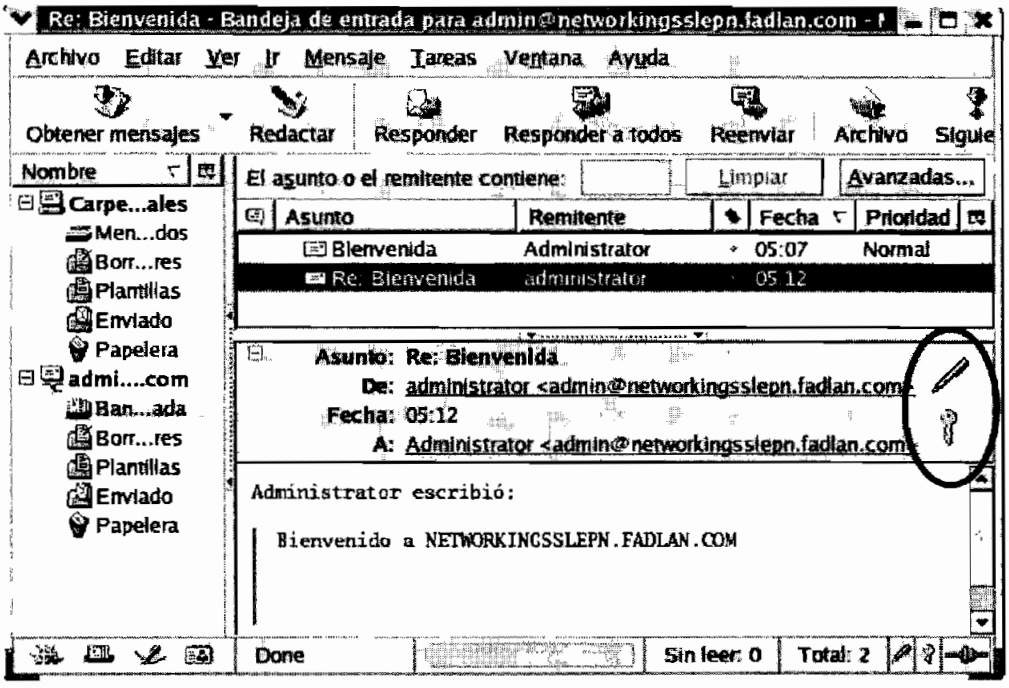


Figura AN8.11 Prueba Texto Encriptado y Firmado (recepción)

Anexo 9: Configuración y prueba de Correo Electrónico con *Netscape Mail & News Groups*

Se deben seguir los siguientes pasos para configurar una cuenta de correo en *Netscape Mail & NewsGroups*:

1. Seleccionar configurar Nueva Cuenta desde el menú Edición.
2. Seleccionar Cuenta de Correo Electrónico (Figura AN9.1)

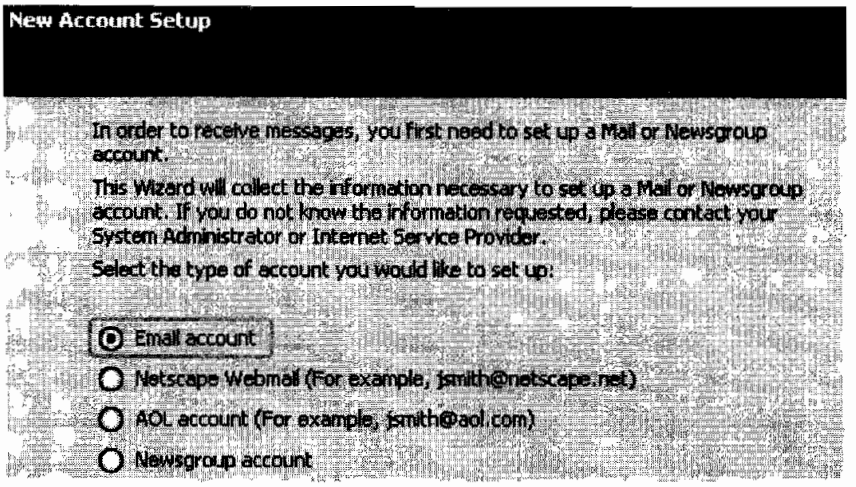


Figura AN9.1 Configuración de Cuenta (1)

3. Ingresar el nombre del dueño de la cuenta así como la dirección de correo electrónico que se está configurando (Figura AN9.2)

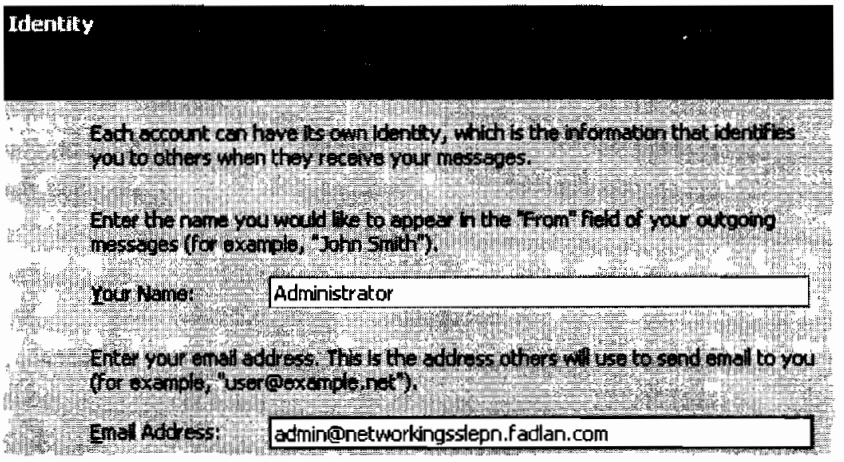


Figura AN9.2 Configuración de Cuenta (2)

4. Ingresar los datos del servidor de correo POP (Figura AN9.3)

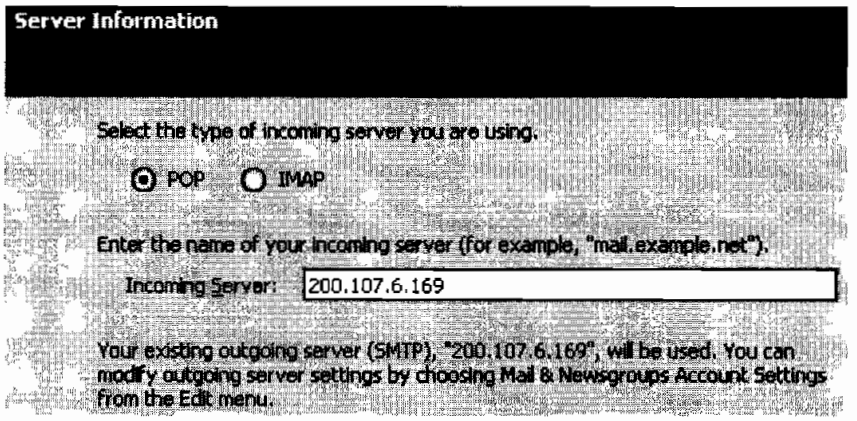


Figura AN9.3 Configuración de Cuenta (3)

5. Ingresar el nombre de la cuenta entregada por el administrador de correo electrónico (Figura AN9.4), puesto que se puede configurar más de una cuenta de correo.

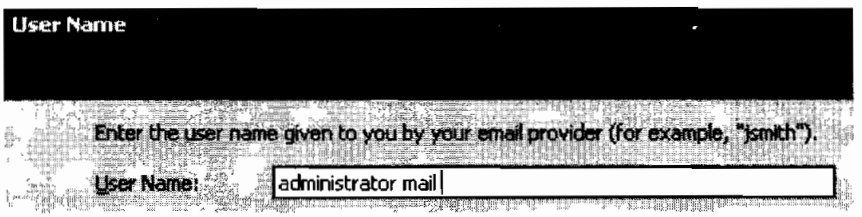


Figura AN9.4 Configuración de Cuenta (4)

6. Ingresar un nombre con el cual se desea referir a esta cuenta de correo electrónico (Figura AN9.5)

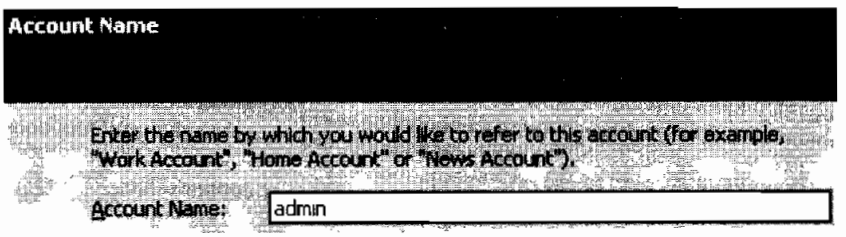


Figura AN9.5 Configuración de Cuenta (5)

7. Verificar los datos ingresados para la configuración del correo electrónico

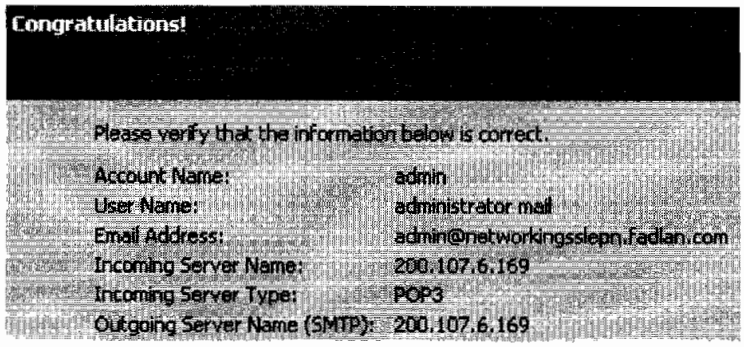


Figura AN9.6 Configuración de Cuenta (6)

Luego es necesario configurar la cuenta de correo para que haga uso del certificado digital de cliente instalado.

1. En el menú Edición, Preferencias, en la opción de Seguridad (Figura AN9.7).

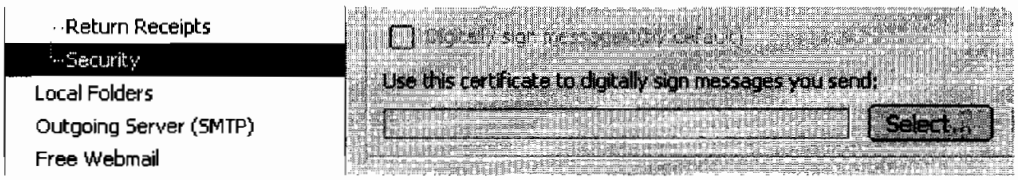


Figura AN9.7 Configuración de Cuenta (7)

2. Seleccionar el certificado digital que se desea usar (Figura AN9.8)

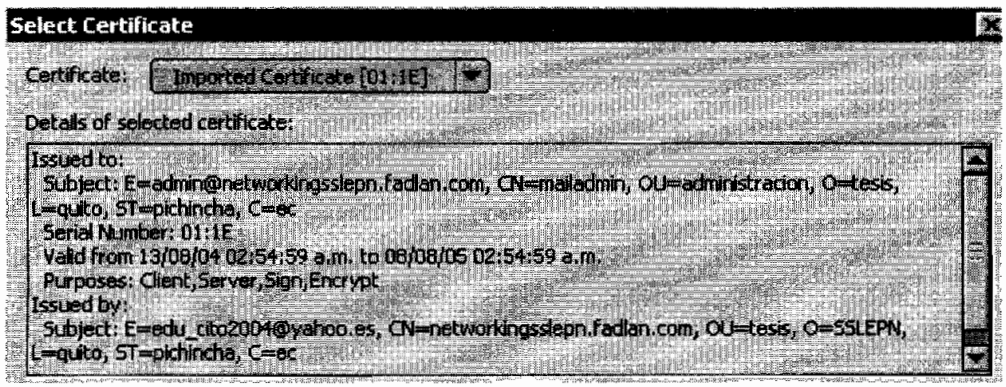


Figura AN9.8 Configuración de Cuenta (8)

Finalmente se muestran las pruebas realizadas de envío de mensajes de correo electrónico.

1. Mensaje sin cifrar ni firmar (Figura AN9.9)

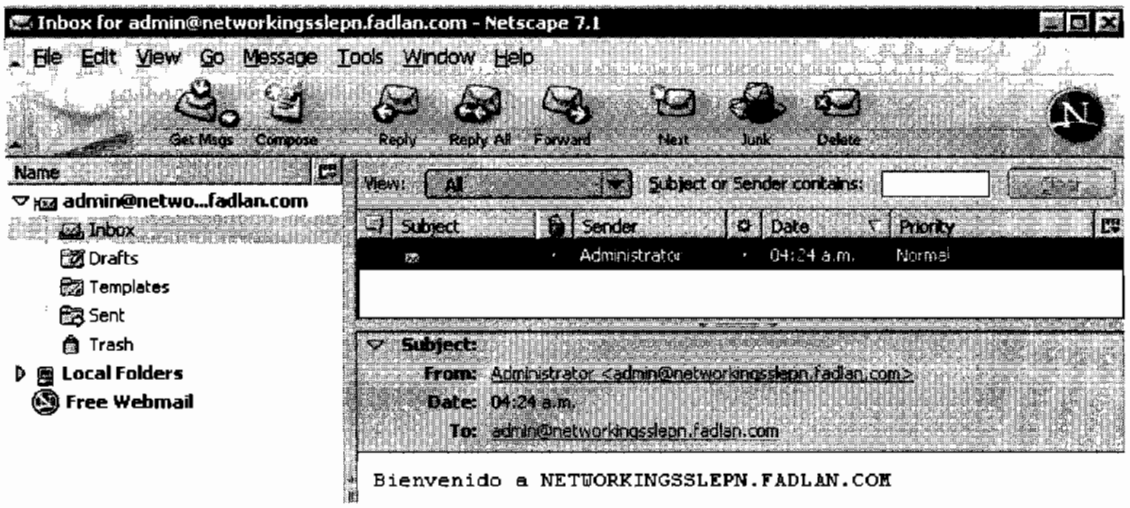


Figura AN9.9 Prueba Texto Plano

2. Envío de un mensaje con encripción y firma digital (Figura AN9.10).
   Se puede observar que es necesario que se seleccionen las opciones de encripción y firma digital.

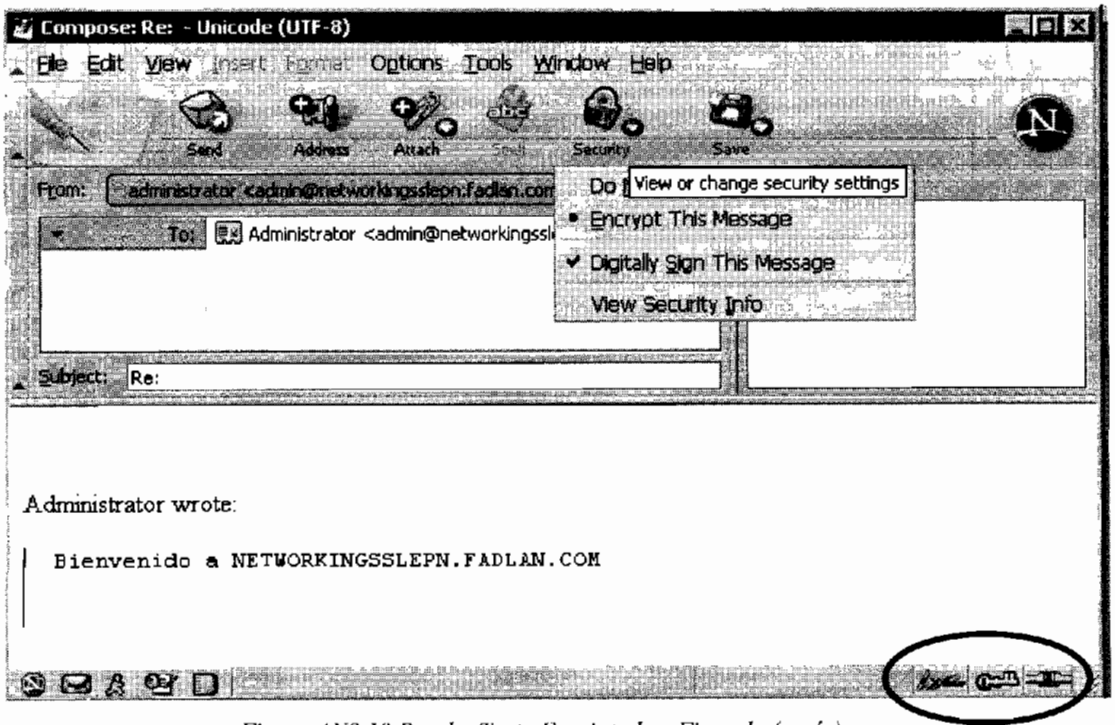


Figura AN9.10 Prueba Texto Encriptado y Firmado (envío)

3. Recepción del mensaje de correo electrónico encriptado y firmado (Figura AN9.11)

Junto al destinatario y remitente se pueden observar los gráficos de una llave que representa la encripción y el de un bolígrafo que representa la firma digital.

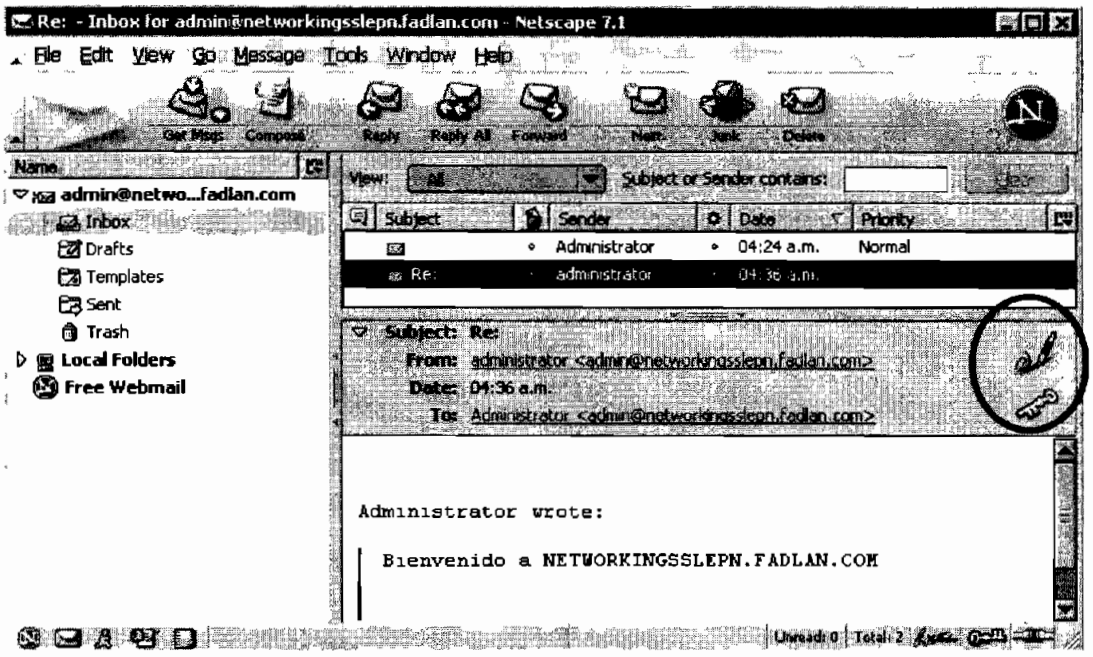


Figura AN9.11 Prueba Texto Encriptado y Firmado (recepción)

Anexo 10: Configuración y prueba de Correo Electrónico con *Microsoft Outlook*

Se deben seguir los siguientes pasos para configurar una cuenta de correo en *Microsoft Outlook*:

1. Sobre el ícono de *Microsoft Outlook* dar click derecho, propiedades.
2. Ingresar el nombre de la cuenta a configurar (figura AN10.1)

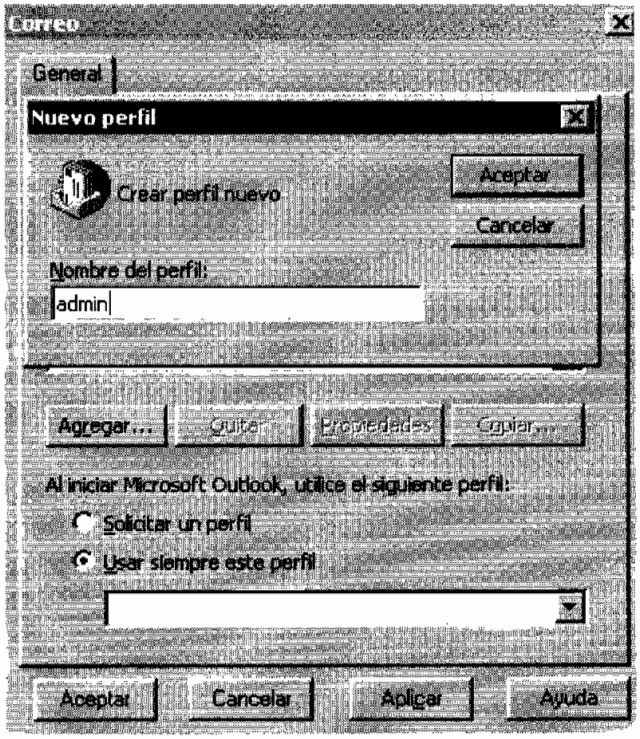


Figura AN10.1 Configuración de Cuenta (1)

3. Seleccionar, Agregar cuenta de Correo Electrónico (Figura AN10.2)

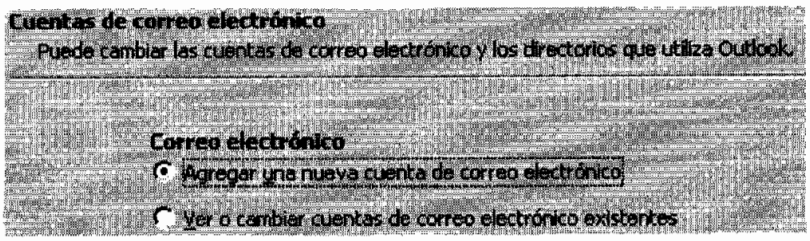


Figura AN10.2 Configuración de Cuenta (2)

4. Seleccionar, Servidor de Correo POP3 (Figura AN10.3)

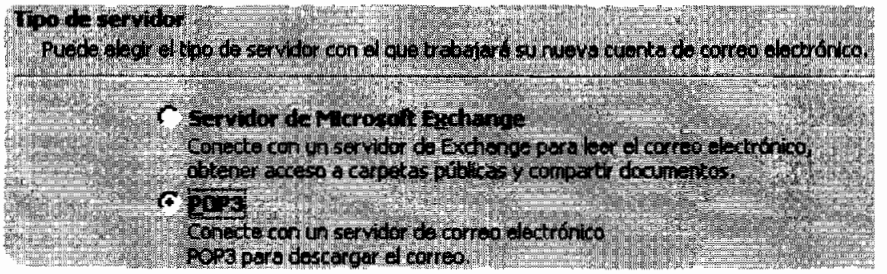


Figura AN10.3 Configuración de Cuenta (3)

5. Finalmente, ingresar los datos de la cuenta (Figura AN10.4)

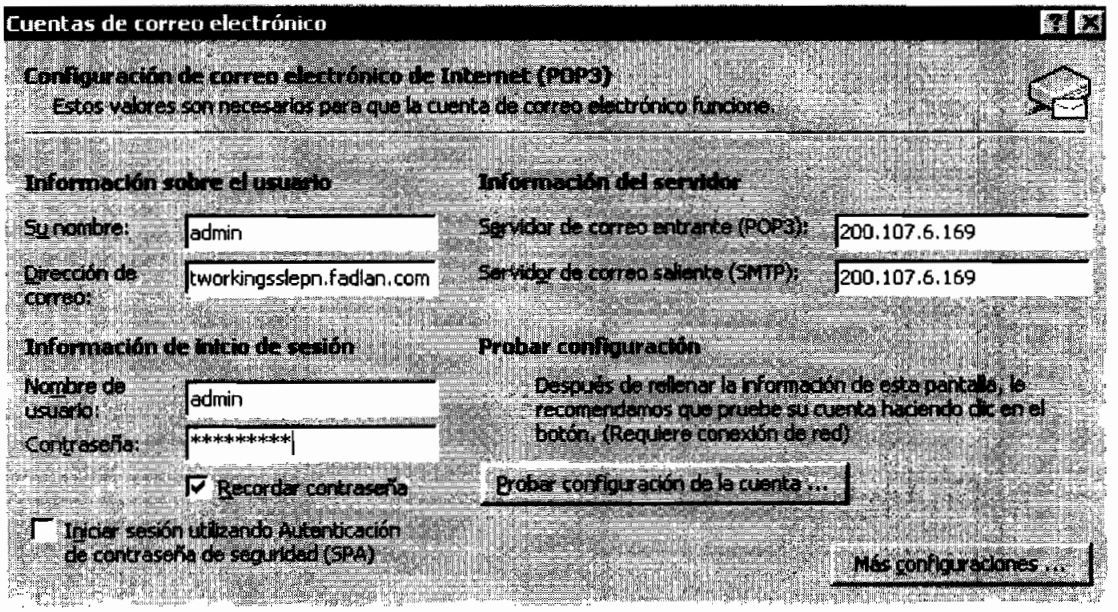


Figura AN10.4 Configuración de Cuenta (4)

Luego es necesario configurar la cuenta de correo para que haga uso del certificado digital de cliente instalado.

1. En el menú Herramientas, Opciones, en la pestaña de Seguridad (Figura AN10.5).
   Hacer click en el botón Configuración para escoger el certificado digital del usuario instalado.

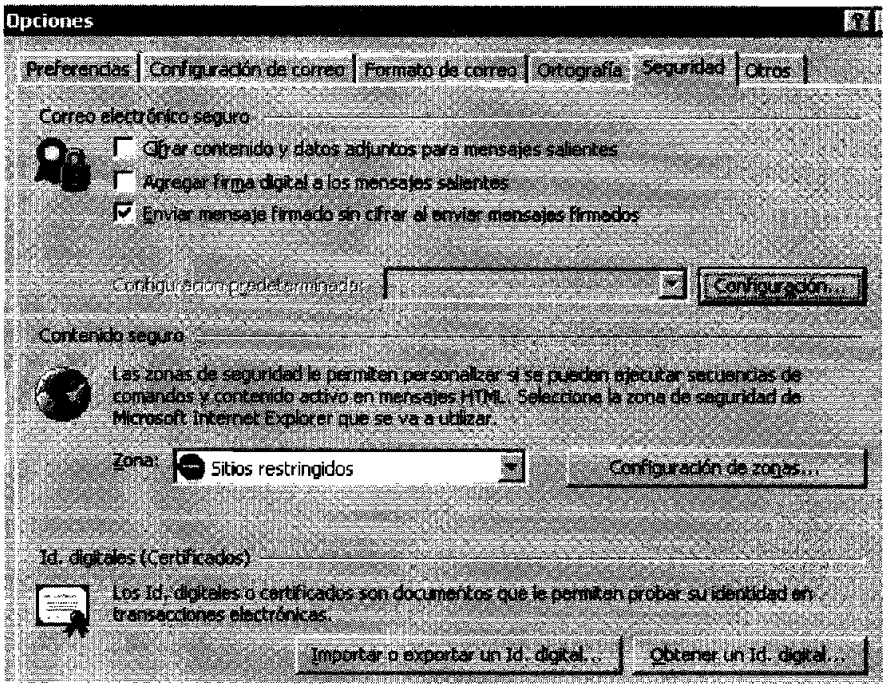


Figura AN10.5 Configuración de Cuenta Segura (1)

2. Haciendo *click* en el botón "Elegir" de los campos Certificado de Firma y Certificado de Cifrado se debe seleccionar el certificado digital que se desea usar para firmar y encriptar (Figura AN10.6)

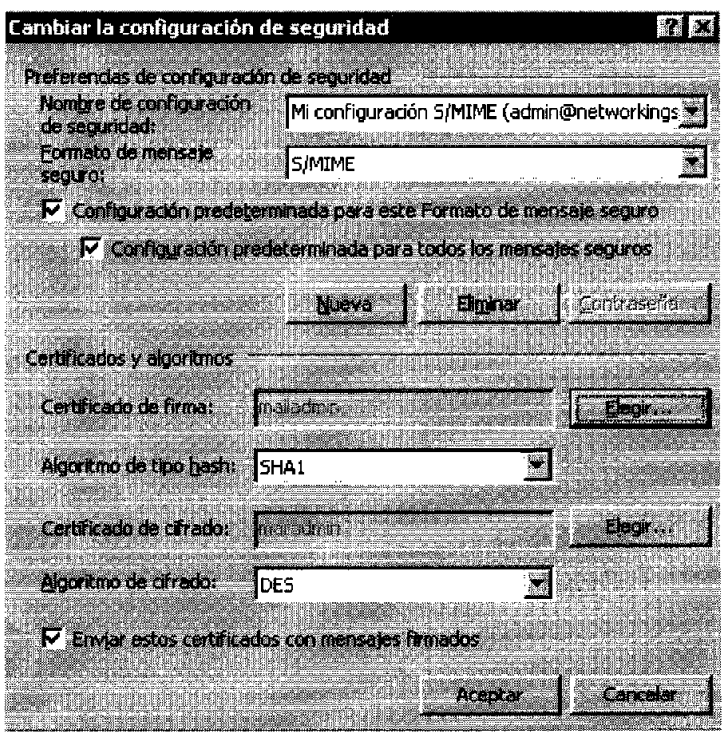


Figura AN10.6 Configuración de Cuenta Segura (2)

3. Finalmente, en la pestaña "Seguridad" se debe seleccionar las opciones de firmado y encriptado haciendo uso del la configuración reciente (figura AN10.7)

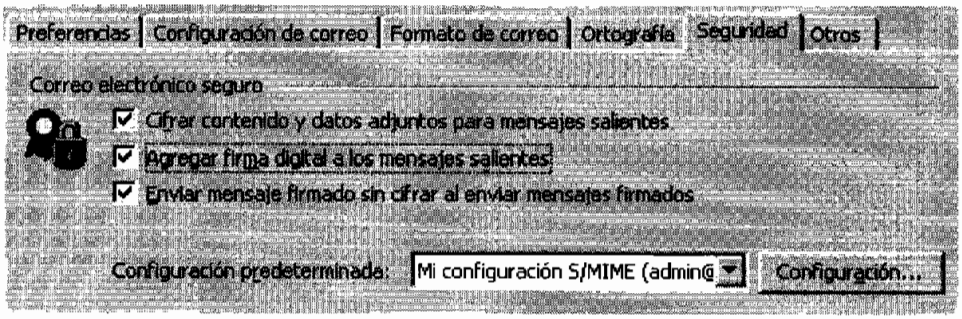


Figura AN10.7 Configuración de Cuenta Segura (3)

Se muestran las pruebas realizadas de envío de mensajes de correo electrónico.

1. Mensaje sin cifrar ni firmar (Figura AN10.8)

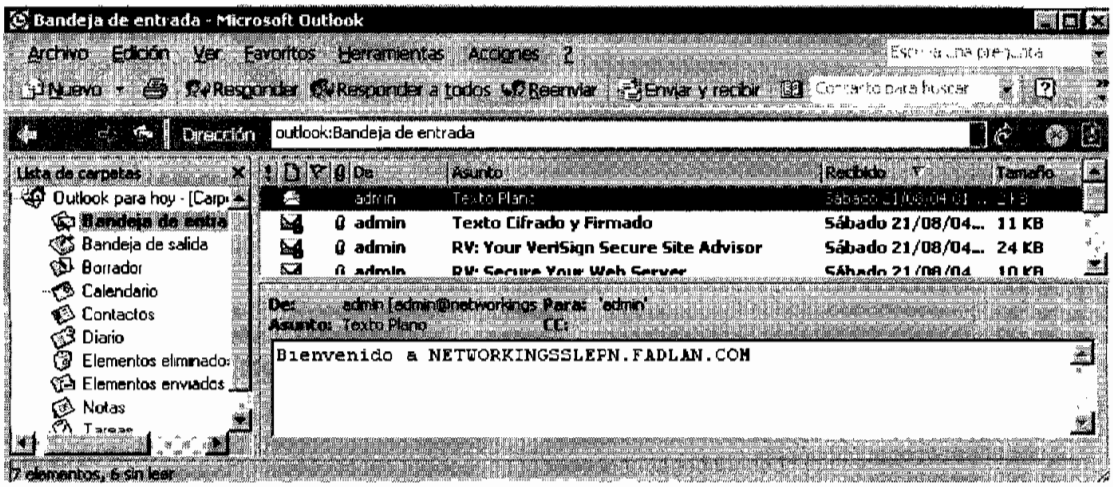


Figura AN10.8 Prueba Texto Plano

2. Envío de un mensaje con encripción y firma digital (Figura AN10.9).
   Se puede observar que ha llegado un mensaje de correo electrónico seguro y es necesario abrirlo en una venta separada.

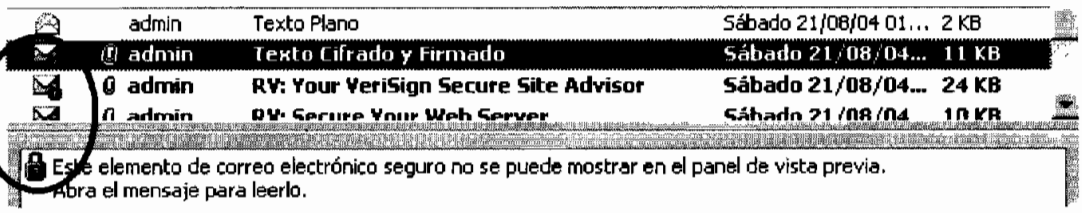


Figura AN10.9 Prueba Texto Encriptado y Firmado (1)

3. Recepción del mensaje de correo electrónico encriptado y firmado (Figura AN10.10)
   Se pueden observar los gráficos que representan la encripción y la firma digital.

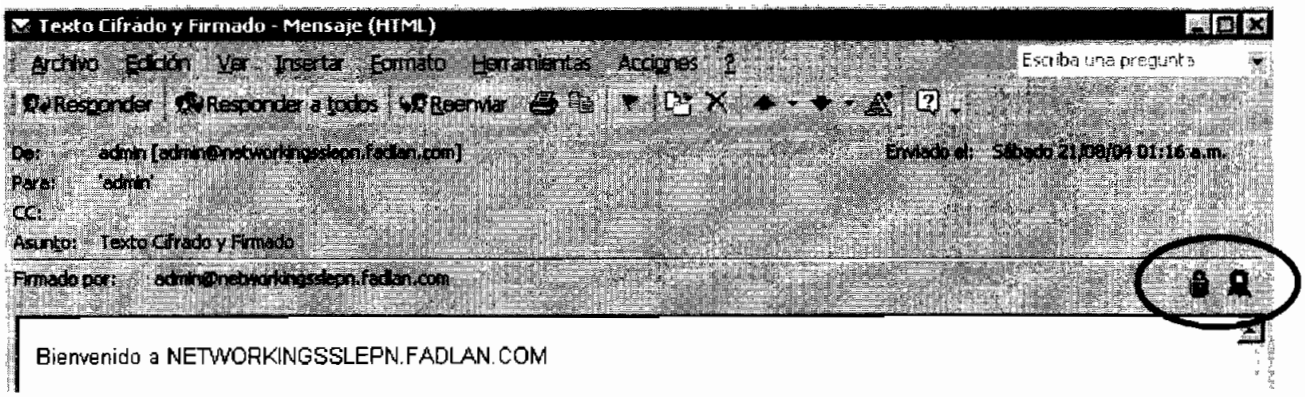


Figura AN10.10 Prueba Texto Encriptado y Firmado (2)

Anexo 11: Prueba de Acceso *Web* con *Microsoft Internet Explorer*

A continuación se muestra una prueba de acceso al correo electrónico haciendo uso de *Microsoft Intenten Explorer*.

1. Se ha solicitado el URL: https://networkingsslepn.fadlan.com (figura AN11.1) La conexión segura se establece entre cliente y servidor al acceder al sitio.

2. El sistema solicita el nombre de usuario y contraseña.

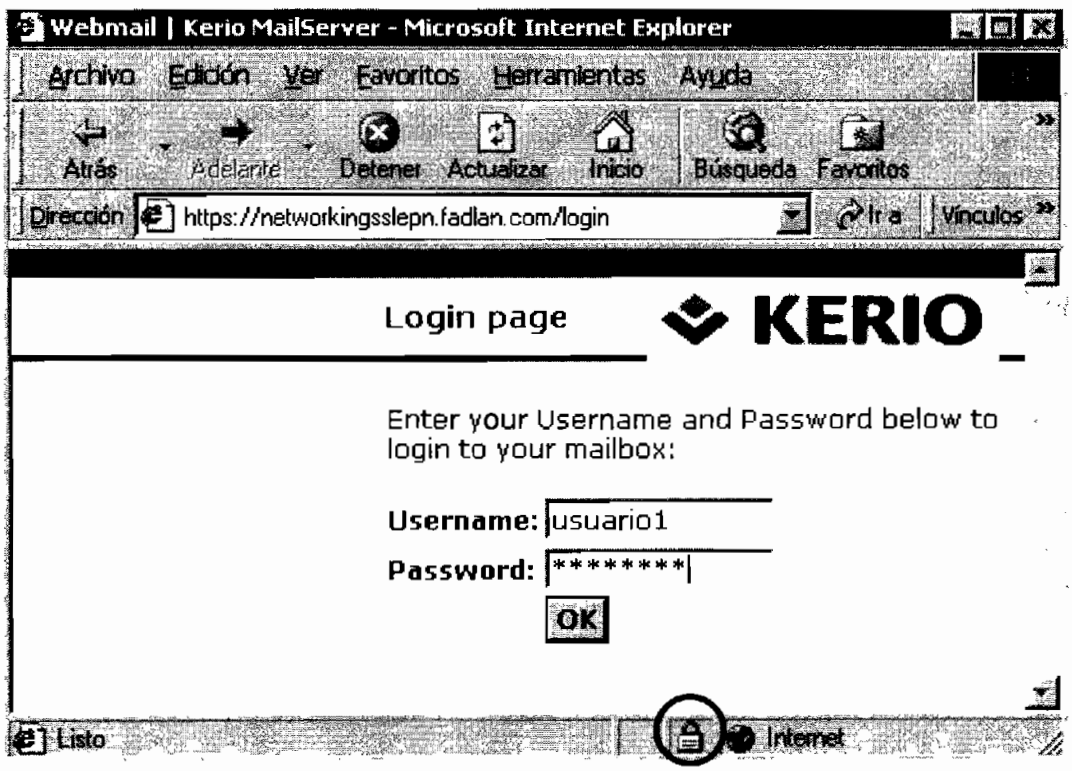


Figura AN11.1 Acceso Web al correo electrónico seguro (1)

3. Se visualiza un mensaje de correo electrónico enviado por el usuario admin@networkingsslepn.fadlan.com con el texto *"Bienvenido a NETWORKINGSSL.FADLAN.COM"* (figura AN11.2)

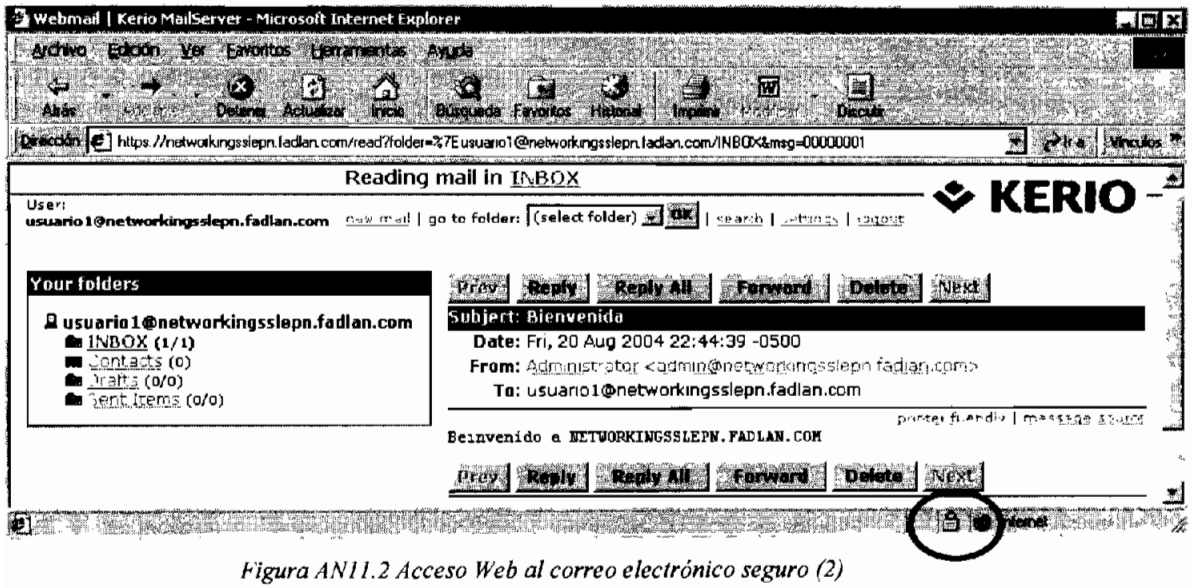


Figura AN11.2 Acceso Web al correo electrónico seguro (2)