METODOLOGÍA PARA LA IMPLEMENTACIÓN DEL PROTOCOLO SECURE SOCKETS LAYER / TRANSPORT LAYER SECURITY EN UN SERVIDOR WEB BASADO EN EL MODELO CLIENTE-SERVIDOR

LEIDY JOHANA PARDO CASTILLO
DIEGO RENATO USGAME LOPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERIAS ELECTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2012

METODOLOGÍA PARA LA IMPLEMENTACIÓN DEL PROTOCOLO SECURE SOCKETS LAYER / TRANSPORT LAYER SECURITY EN UN SERVIDOR WEB BASADO EN EL MODELO CLIENTE-SERVIDOR

LEIDY JOHANA PARDO CASTILLO
DIEGO RENATO USGAME LOPEZ

Trabajo de grado para optar al título de
Especialista en Telecomunicaciones

Director
RAÚL BAREÑO GUTIERREZ
Ingeniero de Sistemas, Especialista

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERIAS ELECTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2012

DEDICATORIA

Queremos dedicar el desarrollo de la monografía primero que todo a Dios por brindarnos sabiduría, paciencia e inteligencia.

También queremos agradecer a nuestras familias. Por su apoyo y acompañamiento en nuestro crecimiento profesional e intelectual que con la experiencia y buen desempeño laboral nos permitirá lograr el éxito en nuestra vida laboral.

AGRADECIMIENTOS

Queremos agradecer a Dios por permitirnos realizar y hacer posible esta meta propuesta de ser especialistas en telecomunicaciones, también queremos dar gracias a la institución y a cada uno de los profesores que nos acompañaron en el desarrollo de las materias por compartir y ofrecernos sus conocimientos y experiencias durante el desarrollo de la especialización lo cual nos permitió obtener las bases y conocimiento para el desarrollo de dicha monografía.

También queremos dar gracias a nuestras familias que de una u otra forma siempre están con nosotros y podemos contar con ellos incondicionalmente de igual forma queremos agradecernos mutuamente por el acompañamiento, trabajo en equipo y compañerismo durante el desarrollo de la monografía como compañeros de trabajo en dicha investigación.

TABLA DE CONTENIDO

INDICE DE FIGURAS

RESUMEN

TÍTULO
METODOLOGÍA PARA LA IMPLEMENTACIÓN DEL PROTOCOLO SECURE SOCKETS LAYER /TRANSPORT LAYER SECURITY EN UN SERVIDOR WEB BASADO EN EL MODELO CLIENTE-SERVIDOR

AUTORES
LEIDY JOHANA PARDO CASTILLO Y DIEGO RENATO USGAME LOPEZ

PALABRAS CLAVES
SSL, TLS, CERTIFICADO DIGITAL, APACHE2.

La incorporación de las Tecnologías de Información y Comunicación al trabajo diario en cualquier organización y especialmente en los procesos de movimiento de datos entre servidores, ha creado la necesidad de generar confianza entre los usuarios de la información, mediante la construcción de transacciones seguras a través de las diferentes redes de acceso.

Este documento pretende ofrecer una guía metodológica para la implementación del protocolo SSL/TLS (conexión y transporte de datos seguros), en servidores Web configurados en sistemas operativos Linux. SSL ofrece autentificación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

El éxito de la seguridad en la información depende de la configuración realizada en el respectivo servidor Web para poner en marcha este protocolo. Se deben conocer detalles de la criptografía utilizada, del manejo de claves y de los certificados de autenticación y de auditoria disponibles en cada caso, para evitar la vulnerabilidad de los datos.

Este protocolo es de dominio público, es decir, los usuarios lo pueden descargar de Internet, sin embargo, para asegurar la privacidad de la información y el transporte seguro, toda organización que decida instalarlo y configurarlo debe adquirir un certificado digital qué será único e intransferible y que contendrá no solo datos privados de la empresa sino también, características particulares de la máquina que ofrecerá la información a través de la red.

---

*Proyecto de Grado
** **Facultad** Ingeniería Físico Mecánica **Escuela** Ingenierías Eléctrica, Electrónica y de Telecomunicaciones **Director** Raúl Bareño Gutierrez

ABSTRACT

TITLE
METHODOLOGY FOR THE IMPLEMENTATION OF THE SECURE SOCKETS LAYER / TRANSPORT LAYER SECURITY PROTOCOL IN A WEB SERVER

AUTHORS
LEIDY JOHANA PARDO CASTILLO Y DIEGO RENATO USGAME LOPEZ

KEY WORDS
SSL, TLS, DIGITAL CERTIFICATE, APACHE2.

Whit the use of Information and Communication Technologies in daily work of the organizations and especially in data movement among servers, network and system managers have created the necessity to generate trust between users of the information allowing making sure transaction through different access nets.

This work presents a methodological guide for the implementation of the SSL/TLS protocol (connection and transport of secure data), in Web servers configured in Linux operating systems. The SSL protocol offers data protection and user authentication by using cryptography while the information is traveling by Internet.

The information security depends on the Web server's configuration used to start this protocol. It is important to know details of the cryptography that should be applied, the way of building the security keys, the authentication certificates and the audit processes used to avoid the vulnerability of the data.

This protocol is of the public domain, which means, users may download it from Internet without any change, however, to assure the privacy of the information and its movement through the net, the organizations that should decide to install and configure it, may acquire a digital certificate that will work as a personal and untransferable authentication document which not only contains private data of the company but also, particular characteristics of the server hardware that will offer the information.

*Degree Project
** **Faculty** Engineering Fhysics Mechanics **School** Engineerings Electric, Electronics and of Telecommunications **Direc** Raúl Bareño Gutierrez

GLOSARIO DE TÉRMINOS

AC: Autoridad Certificadora.

AES (*Advanced Encryption Standard*): Estándar de encriptación avanzada o cifrado simétrico por bloques.

***BlowFish*:** Algoritmo de cifrado simétrico.

CBC (*Cipher-Block Chaining*): Algoritmo de cifrado de clave simétrica que opera en grupos de bits de longitud fija llamados bloques, aplicándoles una transformación variante.

Debian: Sistema operativo Linux de libre distribución, desarrollado por más de mil voluntarios alrededor del mundo que colaboran a través de Internet.

DH (*Diffie Hellman*): Protocolo criptográfico de establecimiento de claves entre partes, se utiliza para acordar claves simétricas que serán empleadas para el cifrado de una sesión.

FORTEZZA o DSA (*Digital Signature Algorithm*): Algoritmo de cifrado simétrico para utilizar con el Estándar de Firma Digital (DSS).

FUNCIÓN HASH CRIPTOGRÁFICA O FUNCIONES HASH: Este tipo de funciones se caracteriza por cumplir propiedades que las hacen resistentes frente a ataques maliciosos, que intentan romper la seguridad en los sistemas que confían en la criptografía.

GSE (Gestión de Seguridad Electrónica): Una de las entidades encargadas de emitir certificados digitales que validan la identidad de los servidores.

IANA (*Internet Assigned Numbers Authority*): Entidad que supervisa la asignación mundial de direcciones IP.

IETF (*Internet Engineering Task Force*): Organización internacional de normalización, que contribuye a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento y seguridad. Es la entidad que regula las propuestas y los estándares de Internet conocidos como RFC.

MAC (*Message Authentication Code*): Código de Autenticación de Mensaje. Se utiliza para garantizar que los mensajes no sean alterados o dañados durante el viaje por la red.

MD5 *(Message Diggest 5*) o SHA-1: Algoritmo utilizado para realizar la comprobación de la integridad de archivos binarios. Permite saber si alguien modificó un archivo, y comprobar si un determinado archivo se descargó correctamente.

NSS (*Name Service Switch*): Provee una interfaz para configurar y acceder a diferentes bases de datos de cuentas de usuarios y claves.

OpenSSL: Proyecto de software libre que consiste en un robusto paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes y navegadores Web para acceso seguro a sitios HTTPS.

RC4 o RC5: Algoritmo de cifrado simétrico.

RFC (*Request for Comments*): Documentos que especifican estándares en Internet.

RSA (*Rivest, Shamir y Adleman*): Sistema criptográfico de clave pública, utilizado tanto para cifrar como para firmar digitalmente.

SSL (*Secure Sockets Layer*): Capa de conexiones seguras.

TCP (*Transmission Control Protocol*): Conjunto de reglas utilizado junto con el Protocolo de Internet (IP) para enviar datos o mensajes entre computadores a través de Internet.

TLS (*Transport Layer Security*): Seguridad en la capa de transporte de datos.

Triple-DES: Algoritmo de cifrado simétrico.

Ubuntu: Sistema operativo Linux de código libre cuyo núcleo está basado en Linux Debian.

INTRODUCCIÓN

Debido al crecimiento acelerado y a la globalización del Internet, así como a la potencialidad de las nuevas herramientas y aplicaciones, las empresas y la población en general que han decidido adoptar estas tecnologías, se han dado a la búsqueda de procesos y aplicaciones especificas que garanticen la autenticación, privacidad e integridad de los datos cuando estos viajan a través de la red, ya que son susceptibles a ataques de personas maliciosas como "*Hackers*" y "*Crackers*" quienes buscan alterar la información mediante procesos de suplantación de identidad, accesos no autorizados, detección de contraseñas y sabotaje, entre otros.

Para controlar los problemas mencionados, existe la necesidad de conocer e implementar protocolos de seguridad efectivos y que creen confianza en el usuario final de la información, por eso, mediante el presente trabajo, se lleva a cabo un estudio detallado del protocolo SSL/TLS (que ofrece seguridad en la conexión y el transporte de los datos), para generar una metodología que facilite su implementación en servidores Web configurados bajo sistemas operativos Linux. Este protocolo se ha seleccionado por ser el más utilizado actualmente por las organizaciones y porque es una aplicación de libre distribución.

Este documento está organizado en cinco secciones cuyos contenidos se introducen a continuación: En la sección 1, se hace un análisis de los posibles ataques que puede sufrir la información cuando viaja a través de las redes. La definición, historia, evolución, funcionamiento y componentes del protocolo SSL/TLS son temas específicos tratados en la sección 2. La sección 3, presenta los requerimientos para la configuración del protocolo SSL/TLS en servidores

Web bajo sistemas operativos Linux. En la cuarta sección, se desarrolla una metodología para guiar el proceso de instalación y configuración del protocolo y, finalmente en la sección 5, se presentan algunas conclusiones que cierran el documento.

1. PROBLEMAS DE SEGURIDAD PRESENTADOS EN UNA RED

Los problemas de seguridad que más comúnmente se presentan al trabajar la información a través de las redes, tienen que ver con la pérdida, robo, extracción y eliminación de la información. A continuación se mencionan algunos de los tipos de ataques más conocidos que afectan la seguridad de los datos:

Ataques de cifrado

Consiste en obtener la clave para descifrar cualquier mensaje cifrado, aprovechando la vulnerabilidad de los algoritmos criptográficos.

Ataques de texto plano

Consiste en que el agresor conoce el contenido del mensaje cifrado y utiliza esta información para descubrir la clave criptográfica empleada para codificar el texto. Una vez obtenida la clave, el mensaje se puede desencriptar y mostrar su contenido real.

Ataques de repetición

El atacante graba o registra la comunicación entre el cliente y el servidor, después se conecta al servidor con la información extraída.

Ataque “hombre en el medio”

El intruso se ubica en medio del cliente y el servidor para suplantar al servidor real. Al engañar al cliente conectado al servidor, puede descifrar los mensajes enviados por el cliente, recibir los datos y después retransmitir al servidor real, lo que le permitirá conocer la información transmitida entre el cliente y el servidor.

Snooping **(Escucha pasiva)**

El agresor vigila el tráfico de red a medida que viaja, con el fin de obtener información confidencial del usuario. Este ataque tiene como objetivo obtener información sin modificarla. Además de interceptar el tráfico de red, captura los documentos, mensajes de correo electrónico e información guardada, descargando esa información a su propio computador.

El *Snooping* se utiliza para el robo de información. Por ejemplo, en alguna época se robaron un archivo con más de 1700 números de tarjetas de crédito a una entidad bancaria y se difundieron ilegalmente informes oficiales reservados de las Naciones Unidas acerca de la violación de derechos humanos en algunos países Europeos.

Tempering **(Manipulación)**

El atacante monitorea el tráfico de red y maliciosamente cambia los datos durante su transmisión. (Por ejemplo, un intruso podría modificar el contenido de un mensaje de correo electrónico). Múltiples sitios Web han sido víctimas del cambio de sus páginas principales por imágenes terroristas, humorísticas, reemplazo de versiones de software que incorporan código malicioso, etc. Otro ejemplo de este ataque son los troyanos, los cuales son falsas versiones de un software con el fin de averiguar información, borrar archivos e incluso tomar control remoto del computador a través de Internet como es el caso del *Back Orifice* y el *NetBus* (ver mayor información en [15]).

Spoofing **(Suplantación de identidad)**

El objetivo de este ataque es actuar en nombre de otros usuarios. Es una forma de engañar para que revelen información personal o financiera mediante un mensaje de correo electrónico o sitio Web fraudulento. Por ejemplo, el envío de correos electrónicos falsos haciendo referencia a entidades bancarias reales con el fin de obtener información confidencial.

***Hijacking* (Secuestro)**

Su objetivo es el robo de sesiones de usuarios para lograr acceso privilegiado a una aplicación Web comprometiendo la integridad y confidencialidad de la información.

Es un ataque donde el intruso toma control de una sesión de comunicación existente entre un servidor y un usuario legítimo que se ha conectado y autenticado con el servidor. El intruso puede supervisar la sesión de manera pasiva al registrar la transferencia de información confidencial como contraseñas y códigos. Otro tipo de *hijacking* se presenta al forzar la desconexión del usuario y tomar control de la sesión. El intruso comienza a actuar como el usuario, ejecutando comandos y enviando información al servidor.

Capturas de tramas de red

Existen programas conocidos como *Sniffers* con los que se puede ver el tráfico de datos que recorren la red, capturando tramas de información. Los *Sniffers* permiten analizar la información real que se trasmite por la red, capturar contraseñas y nombres de usuarios. También son utilizados para capturar números de tarjetas de crédito, direcciones de correo electrónico entrante y saliente entre otras funciones.

Existen herramientas como el *Sniffer Wireshark* que son utilizadas para verificar el funcionamiento del protocolo SSL.

Jamming* o *flooding

Este tipo de ataques desactivan o saturan los recursos del sistema. Por ejemplo, un intruso puede consumir toda la memoria o espacio en disco disponible, así como enviar gran cantidad de tráfico a la red para que nadie más pueda utilizarla. Muchos proveedores de Internet han sufrido bajas del servicio por ataques que congestionan el protocolo TCP. En este caso, el atacante satura el sistema con

mensajes que requieren establecer conexión, sin embargo, en vez de proveer la dirección IP del emisor, el mensaje contiene falsas direcciones IP (este ataque involucra también *spoofing*). El sistema responde el mensaje, pero como no recibe respuesta, acumula *buffers* con información de las conexiones abiertas, sin dejar lugar a las conexiones legítimas.

Muchos nodos de Internet han sido dados de baja por el "*ping de la muerte"*, una versión trampa del comando *ping*, mientras que el *ping* normal simplemente verifica si un sistema está activo en la red, el "*ping de la muerte"* causa un fallo en el sistema haciendo que los equipos dejen de funcionar de manera correcta.

2. PROTOCOLO SECURE SOCKETS LAYER/TRANSPORT LAYER SECURITY (SSL/TLS)

Esta sección presenta información detallada acerca del protocolo SSL/TLS: definición, historia y funcionamiento, características y ventajas de su implementación en un servidor Web.

2.1 Definición

SSL y su sucesor TLS, es un protocolo de seguridad desarrollado para lograr que la transmisión y recepción de datos que viajan a través de Internet entre un servidor y un cliente sean seguras. Para esto, utiliza un sistema de encriptación de datos que previene la falsificación, modificación o interceptación de la información hecha por terceras personas [8].

SSL/TLS resuelve tres problemas principales que están relacionados con: la seguridad de la información, la integridad y la autenticidad de los datos. Se basa

en la utilización de un sistema de cifrado que emplea algoritmos matemáticos como el *Avanced Encryption Standard* (AES) o el *Rivest*, *Shamir* y *Adleman* (RSA), entre otros y un sistema de claves públicas y privadas.

Generalmente la llave pública encripta la información que sólo puede ser desencriptada por la llave privada. También se puede emplear la misma clave para encriptar y desencriptar la información, de esta manera, alguien que no tenga acceso a la llave privada no podrá leer su contenido. Además, el empleo de funciones *Hash* garantiza que los datos no hayan sido alterados. En otras palabras, el protocolo SSL/TLS está basado en la aplicación conjunta de criptografía asimétrica o de llave pública, criptografía simétrica, certificados y firmas digitales, para conseguir un canal seguro de comunicación a través de la red.

2.2 Historia

SSL fue desarrollado e implementado por *Netscape Communications* en 1994, con el propósito de garantizar a sus productos (navegador y servidores) conexiones seguras entre ellos, convirtiéndolo en el estándar internacional de comunicaciones seguras y siendo adoptado por bancos y muchas empresas. El objetivo de *Netscape* era el de crear un canal de comunicaciones seguro entre un cliente y un servidor, que fuera independiente del sistema operativo utilizado y que se beneficiara de los nuevos adelantos en materia de cifrado (ver mayor información en [1]).

La primera versión del SSL no fue implementada, y la versión 2.0 que apareció unos meses después presentó diversos errores de diseño. En noviembre de 1995 y tras la liberación del SSL 3.0 se mejoró la versión que evolucionó en la creación del protocolo TLS (ver mayor información [2]).

SSL 2.0 ofrecía cifrado de datos, autenticación de servidores, integridad del mensaje y autenticación del cliente en conexiones TCP/IP, pero tuvo falencias en:

- Una MAC débil y basada en MD5.
- Utilizaba las mismas claves para la autenticación y el cifrado.
- Utilizaba el protocolo TCP, lo que lo hacía susceptible a ataques de truncamiento y falsificación.

El TLS 1.0 que evolucionó del SSL 3.0 tuvo las siguientes características:

- Mejora en la expansión de las claves.
- Se basó en HMAC.
- Soportó el protocolo DH, el intercambio de claves y el algoritmo de encriptamiento Triple-DES.

El TLS nació de la mano del IETF en forma de RFC en enero de 1999. Se construyó a partir de las especificaciones del SSL 3.0. Las diferencias entre este protocolo y el SSL 3.0 no son muchas, pero sí suficientes para que los dos no puedan interoperar entre sí.

Las principales diferencias entre SSL 3.0 y TLS 1.0 son:

- En los mensajes *Certificate Request y Certificate Verify* del protocolo de *Handshake* en SSL 3.0, si el servidor solicita un certificado al cliente para que se autentique, este debe responder con un mensaje de alerta advirtiendo que no lo tiene, en cambio en TLS 1.0 si el cliente no posee certificado no responde al servidor de ninguna manera a este requerimiento.
- El mecanismo utilizado para construir las claves de sesión es diferente en TLS 1.0.

- TLS 1.0 no soporta el algoritmo de cifrado simétrico *FORTEZZA* o DSA que sí es soportado por SSL 3.0. Esto es debido a que TLS busca ser íntegramente público mientras que *FORTEZZA* es un algoritmo propietario.
- TLS utiliza un mecanismo diferente y más seguro en el cálculo del MAC.
- TLS 1.0 introduce nuevos códigos de alerta, no contemplados por SSL 3.0
- TLS 1.0 introduce un nuevo mecanismo en el relleno de los bloques para prevenir ataques basados en el análisis de la longitud de los mensajes (para mayor información ver [1]).

Phillip Rogaway descubrió una vulnerabilidad en TLS 1.0 en el cifrado del bloque (CBC), en 2002. Mozilla actualiza las versiones de sus bibliotecas NSS para mitigar los ataques. NSS es utilizado por Mozilla Firefox y Google Chrome para implementar SSL. Este ataque fue llamado “BESTIA” el cual se previno mediante la eliminación de todos los sistemas de cifrado CBC, dejando sólo el sistema de cifrado RC4, que sigue siendo ampliamente compatible con la mayoría de los sitios Web. Los usuarios de Windows 7 y Windows Server 2008 pueden habilitar el uso de TLS 1.1 y 1.2 [2].

TLS 1.1 fue definido en el RFC 4346 en abril de 2006. Siendo ésta una actualización de la versión 1.0 de TLS. Las diferencias más significativas fueron:

- Alta protección contra ataques al CBC.
- Cambio en el manejo de los errores de relleno.
- Apoyo para el registro de los parámetros de la IANA.

La primera definición de TLS apareció en el RFC 2246 titulada "*The TLS Protocol Version* 1.0". Otros RFC posteriores extendieron TLS así:

- RFC 2712: "*Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)*". Este documento propone la inclusión de nuevas “ciphersuites” al

protocolo TLS que soporte la autenticación basada en Kerberos. Las credenciales Kerberos se utilizan para lograr autenticación mutua y para establecer una clave maestra secreta que es subsecuentemente utilizada para proteger la comunicación cliente-servidor.

- RFC 2817: "*Upgrading to TLS Within HTTP/1.1*". Este explica cómo usar el mecanismo de actualización en HTTP/1.1 para iniciar TLS sobre una conexión TCP existente. El tráfico HTTP inseguro y seguro comparten el mismo puerto conocido (en este caso, 80 para el http en lugar del 443 para el https).
- RFC 2818: "*HTTP Over TLS*". Diferencia tráfico seguro de tráfico inseguro mediante el uso de un puerto de servidor diferente.
- RFC 3268: "*AES Cipher suites for TLS*". Añade la familia de cifrado AES a los cifrados simétricos previamente existentes.
- RFC 3546: "*Transport Layer Security (TLS) Extensions*", añade un mecanismo para negociar extensiones de protocolos durante la inicialización de sesión y define algunas extensiones.
- RFC 4279: "*Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*", añade tres conjuntos de nuevas familias de cifrados para que el protocolo TLS permita la autenticación basada en claves previamente compartidas.

2.3 Conceptos asociados a SSL/TLS

Para lograr entender el funcionamiento de esta tecnología, es necesario conocer y tener claros algunos conceptos que forman parte de su estructura:

Criptografía o cifrado

La criptografía permite asegurar los datos que viajan a través de una red. Existen muchos algoritmos criptográficos los cuales ofrecen los siguientes servicios:

- **Confidencialidad**: Garantiza que sólo autorizados acceden a la información.
- **Integridad**: Garantiza la no alteración de los datos o información que viaja a través de la red.
- **Autenticación**: Permite identificar el creador de la información. Ejemplo, cuando se recibe un mensaje de alguien se asegura de que fue enviado por ese alguien y que no sea una suplantación de identidad.
- **No repudio**: Este servicio permite identificar y probar que cada una de las entidades comunicantes han participado en una comunicación. La prueba puede ser de dos tipos:

  - Con chequeo de origen: Cuando el destinatario tiene prueba del origen de los datos.
  - Con chequeo de entrega: Cuando el origen tiene prueba de la entrega de los datos al destinatario deseado.

  Es decir, que el no repudio consiste en que no se pueda negar la autoría del envío de mensajes entre el cliente y el servidor.

En conclusión, el cifrado o encriptación es el proceso que transforma la información de manera que no cualquier usuario pueda entenderla; se realiza con base en un elemento único conocido como llave, así nadie, excepto el poseedor de la clave puede leerla.

Algoritmos Criptográficos

A continuación se describen algunos algoritmos criptográficos que son implementados por el protocolo SSL/TLS con el fin de asegurar los datos que viajan a través de una red.

- **Cifrado de Clave Simétrica**

  Consiste en cifrar y descifrar datos utilizando una única clave o llave como se muestra en la figura 1. La clave y el mensaje del emisor se convierten al algoritmo de cifrado, obteniendo el mensaje cifrado. Este resultado puede ser enviado a través de un medio no seguro, permitiendo que sólo el receptor que tiene la clave descifre el mensaje. Esta clave debe permanecer en secreto para que el esquema sea eficaz.

  Si se presenta un intercambio de claves se corre el riesgo que estas sean descifradas. Una solución al envió y recepción de claves, es utilizar un protocolo de intercambio de claves criptográficas o de cifrado llamado *Diffie-Hellman*, proporcionado por OpenSSL, que garantiza la no divulgación de la clave en la red.

  Existen varios tipos de cifrados simétricos como 3DES, AES, RC4, RC5, *BlowFish*, entre otros. Unos de los cifrados simétricos más utilizados son el Triple DES (3DES o DES3) y el AES, que es el nuevo estándar avanzado de encriptación, el cual sustituirá a 3DES porque es más rápido.

  Es importante tener en cuenta que otro factor para garantizar la seguridad es el tamaño o longitud de la clave, ya que cuanto más grande sea, mayor seguridad existe, por eso se recomienda que ésta sea mayor o igual a 80 bits.

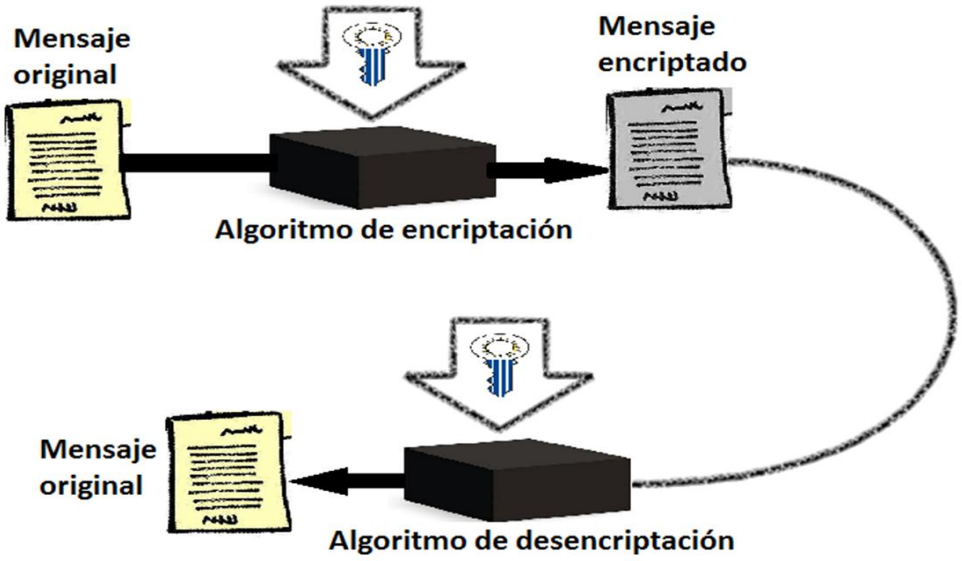


Figura 1.Cifrado de clave simétrica

3DES tiene una longitud fija de 112 bits y no se recomienda el uso de claves pequeñas como 56 y 40 bits ya que son demasiado débiles y las pueden descifrar fácilmente.

AES Es el estándar de encriptación avanzado que permite la creación de claves con tamaños de 128, 192 y 256 bits. Sus características más relevantes son:

- Se trabaja a nivel de byte.
- Tiene sus propias operaciones aritméticas como suma exclusiva bit a bit y multiplicación.
- Está implementado para trabajar en los procesadores de 8 bits utilizados en tarjetas inteligentes y en CPUs de 32 bits.
- No es de tipo *Feistel* [16] que es un método de cifrado en bloque con una estructura particular.

RC5 realiza operaciones OR exclusivo, suma modular y desplazamiento de bits; cifra la información en bloques de 32, 64 o 128 bits. El tamaño de la clave se sugiere de 128 bits. Entre sus características más destacadas están:

- Es muy rápido.
- La arquitectura es simple.
- Bajos requisitos de memoria.
- Alta seguridad.

BlowFish es un algoritmo tipo *Feistel.* Su clave variable, cifra bloques de texto de 64 bits. El tamaño de la clave va desde 32 hasta 448 bits. Sus principales características son:

- Utiliza una función con las cuatro *cajas-S* [17] y operaciones básicas de suma y OR exclusivo.
- Solo necesita 5K de memoria.
- Es rápido.
- Es sencilla su implementación.
- Su fortaleza es la longitud de la clave**.**

❖ **Cifrado de Clave Pública o Cifrado Asimétrico**

También es conocido como cifrado de clave pública y permite la comunicación cifrada entre dos partes. Utiliza claves distintas, es decir que cada una de las partes que intervienen en la transmisión y recepción de información cuentan con un par de claves una privada y una pública; la pública puede distribuirse y es de libre disposición de todos los interesados, mientras que la clave privada siempre se mantiene segura, los datos que están encriptados o cifrados con la llave pública pueden ser desencriptados o descifrados únicamente con la clave privada como lo muestra la figura 2. El algoritmo más conocido de cifrado asimétrico es el RSA [18].

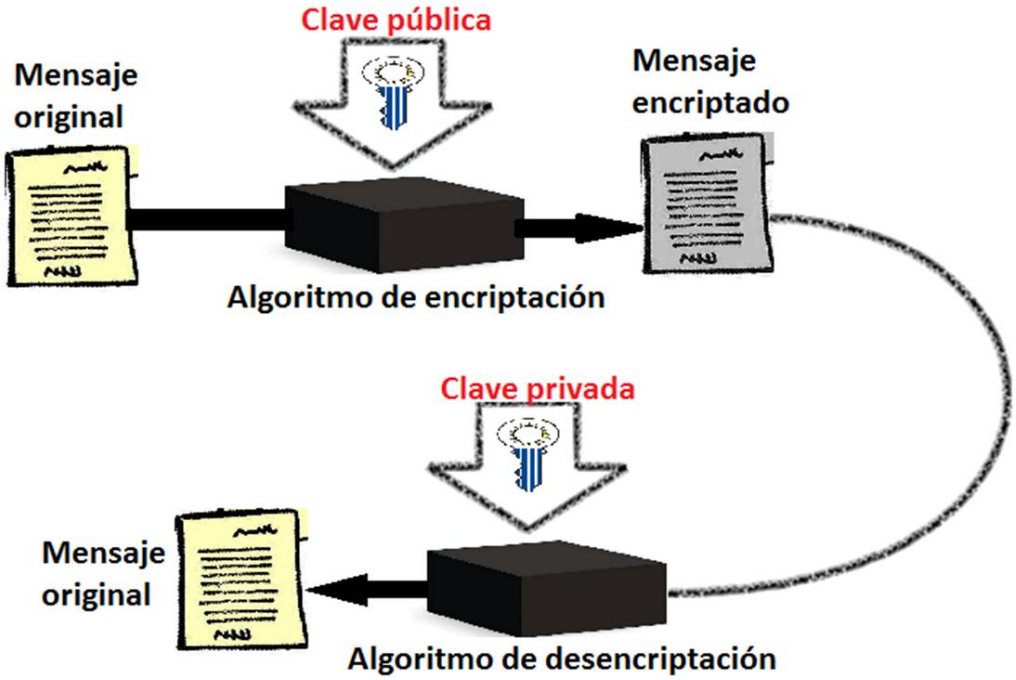


Figura 2. Cifrado de clave pública o cifrado asimétrico

La longitud en bits de la clave pública o cifrado asimétrico es extremadamente grande. Utiliza claves de 1024 bits o mayores que lo hacen demasiado lento y de 512 bits que son débiles. El sistema de cifrado ECC (Criptografía De Curva Elíptica) modifica el cifrado asimétrico y ofrece las mismas garantías de seguridad y tamaños de las claves pero con mayor velocidad. En la actualidad OpenSSL no es compatible con ECC (para mayor información ver [4]).

Los protocolos que utilizan estos algoritmos son: DSS (*digital Signature Standard*), DSA (*Digital Signature Algorithm*), PGP, GPG, SSH, SSL y TLS. Los algoritmos que utilizan esta técnica son: *Diffi-Hellman*, RSA, DSA, El Gamal, Criptografía de curva elíptica, entre otros.

RSA es el algoritmo utilizado para generar claves numéricas. Su principio de funcionamiento es la factorización de los números enteros y la elección de

dos números primos de aproximadamente la mitad del tamaño de la clave. Sus principales características son:

- Permite firmar digitalmente los mensajes que se envían.
- Cualquiera de las dos claves se pueden utilizar para el encriptado y desencriptado.
- Más complejidad en los procesos de gestión de claves (generación, almacenamiento, distribución y mantenimiento).
- La resolución de la desencriptación es un problema completo: no existe un algoritmo conocido capaz de realizar la desencriptación en un tiempo polinomial.

Código de Autenticación de Mensajes

Los códigos de autenticación de mensajes (MAC) se utilizan para garantizar que los mensajes no sean alterados o dañados durante el viaje por la red. Consisten en un resumen del mensaje que incluye una clave generada al enviar los datos o el mensaje y que verifica que los datos se reciban bien. No es posible ver el resumen sin conocer tanto el mensaje introducido y la clave, por lo tanto un posible atacante necesita saber la clave para construir una MAC [4].

Actualmente, existen 3 funciones MAC:

- **CBC-MAC**: que consiste en convertir un algoritmo de cifrado simétrico en una función MAC. El funcionamiento básico es cifrar el mensaje utilizando un algoritmo CBC y eliminar todo el resultado de texto cifrado a excepción del último bloque.
- **HMAC**: Utiliza una función *hash* para implementar una función MAC. La opción más popular hoy en día es la de utilizar HMAC-SHA-256.
- **UMAC**: Es un tipo de código MAC que se calcula eligiendo una función *hash*

Bloqueo

Consiste en tomar una cantidad fija de datos para cifrarlos y descifrarlos. Como SSL / TLS permite la transferencia de gran cantidad de datos, el bloqueo le facilita la división de estos datos en bloques. Este enfoque se denomina Libro de Código Electrónico (BCE). Sin embargo, su principal inconveniente consiste en que si el texto plano para dos bloques es el mismo, el texto cifrado será el mismo. Esta información puede ser útil para que un atacante pueda descifrar la secuencia [2].

Los siguientes son los modos de operación para algoritmos de cifrado por bloques:

Modo ECB (*Electronic Code Book*)

Es el más simple, y consiste en dividir el texto en bloques y cifrar cada uno de ellos independientemente.

- Ventajas:
  - Permite codificar bloques independientemente de su orden.
  - Es resistente a errores.
- Desventajas:
  - Si el bloque presenta mensajes que se repiten, el texto cifrado también los presentará.

Modo CBC (*Cipher Book Chaining*)

Se suma a cada bloque de texto, antes de cifrarlo, (bit a bit, con XOR) el bloque anteriormente cifrado. Al primer bloque se le suma un vector de inicialización, que es un conjunto de bits aleatorios de la misma longitud que un bloque. Escogiendo vectores distintos cada vez, aunque el texto en claro sea el mismo, los datos cifrados serán distintos. El receptor debe conocer el valor del vector antes de empezar a descifrar, pero es necesario

guardar este valor en secreto, que normalmente se transmite como cabecera del texto cifrado.

- Ventajas:
  - Protege respecto a la sustitución de bloques.
  - Es resistente a errores.

Modo CFB (*Cipher Feedback Mode*)

El algoritmo de cifrado no se aplica directamente al texto sino a un vector auxiliar. Del resultado del cifrado se toman *n* bits que se suman a *n* bits del texto original o sin cifrar para obtener *n* bits de texto cifrado. Estos bits cifrados se utilizan también para actualizar el vector auxiliar. El número *n* de bits generados en cada repetición puede ser menor o igual que la longitud de bloque *b*. Tomando como ejemplo *n* = 8, se obtiene un cifrado que genera un byte cada vez, sin que sea necesario esperar a tener un bloque entero para poderlo descifrar.

Modo OFB (*Output FeedBack Mode*)

Opera como el CFB pero en lugar de actualizar el vector auxiliar con el texto cifrado, se actualiza con el resultado obtenido del algoritmo de cifrado. La propiedad que distingue este modo de los demás consiste en que un error en la recuperación de un bit cifrado, afecta solamente al descifrado de este bit.

Llave Pública y Privada

Son un par de llaves digitales asociadas a una persona o entidad y generadas mediante métodos criptográficos. La llave pública se utiliza para cifrar la información, mientras que la llave privada se utiliza para descifrarla, por lo tanto ésta debe mantenerse en secreto [3].

Una clave criptográfica es un dato que controla la operación de un algoritmo de criptografía, generalmente esta información es una secuencia de números o letras mediante la cual en criptografía se especifica como la transformación del texto plano en texto cifrado o viceversa.

HTTPS

Es la combinación del protocolo HTTP utilizado en una transacción Web protegida con el protocolo SSL/TLS para establecer comunicaciones cifradas [3].

Funciones *Hash* o de resumen

La función *hash* toma como entrada un mensaje al cual se le aplica una serie de operaciones aritméticas y lógicas y genera otro mensaje de igual tamaño o reducido entre 128 o 160 bits.

Esta función *hash* debe producir un notable cambio en el mensaje para que no sea entendido y en caso de ser interceptado sea imposible conocer el mensaje original a partir del mensaje resumen, esto con el fin de que cuando se envié un mensaje por la red se pueda estar seguro que nadie haya modificado su contenido ya que si al escribir el mensaje se le aplica la función *hash*, se produce un resumen el cual se adjunta y cuando éste sea recibido se le vuelve aplicar esta función *hash* para comprobar si coincide con el resumen. Si esto sucede, significa que el mensaje no ha sido modificado, si no, es porque éste no es igual y por lo tanto ha sido interceptado y modificado por una tercera persona.

En conclusión, esta función permite verificar si el mensaje ha sido interceptado y modificado durante su viaje por la red.

Algunas de las funciones *hash* más conocidas son la MD5 o *Message Diggest* nº5, que hace un resumen de 128 bits, y el SHA1 y SHA2 *Standard Hash Algorithm* nº1 y 2, que generan un resumen de 160 bits.

Existe un gran parecido entre la función que cumple la encriptación o cifrado con la función *hash,* por lo que se debe tener claro que los mensajes intercambiados entre un cliente y un servidor o las dos partes pueden tener un gran tamaño, lo que dificulta el proceso de cifrado, entonces no se cifra el mensaje entero sino un resumen del mismo que se obtiene al aplicar al mensaje una función *hash*.

Lo anterior se realiza con el fin de que el mensaje original varíe su tamaño de acuerdo a su volumen y al convertirlo con la función *hash* éste tamaño se vuelve fijo, generalmente de 160 bits. Para lograr esto, el mensaje se divide en varias partes cada uno de las cuales tendrá el tamaño de 160 bits; una vez dividido, se combinan elementos tomados de cada una de las partes resultantes, para formar el mensaje resumen o *hash* que también tendrá un tamaño fijo de 160 bits. Este resumen de tamaño fijo es el que se cifrará utilizando la clave privada del emisor del mensaje.

Objetivos del protocolo SSL/TLS

- **Realizar la autenticación del cliente y el servidor**: Este proceso se realiza mediante el uso de criptografía de claves (cifrado de clave pública) para autenticar tanto el cliente como el servidor utilizando un certificado digital, que se puede obtener a través de una entidad pública o privada dedicada a la emisión de estos.
- **Garantizar la integridad de los datos:** para que la información enviada a través de la red no sea modificada durante su trayectoria.
- **Asegurar la privacidad de los datos:** para que terceros no tengan acceso a la información durante su transmisión a través de la red y hasta llegar a su destino [11][5].

2.4 Arquitectura SSL/TLS

SSL está conformado por un conjunto de protocolos que se dividen en 2 capas (ver figura 3):

Capa1: compuesta por el Protocolo SSL *Record* que se encarga de la encriptación de los datos, y de la integridad y encapsulación de los mismos cuando son enviados por otros protocolos SSL.

Capa2: en la que se utilizan tres protocolos de enlace SSL: el *SpecProtocol*, el SSL *ChangeCipher* y el de alerta SSL. Estos protocolos trabajan en la administración de sesión, en la administración de parámetros de cifrado y en la transmisión de mensajes SSL entre el cliente y el servidor. Cuando se habla de sesión se hace referencia a una conexión entre el cliente y el servidor con configuraciones de los algoritmos a utilizar, numero de sesión, entre otros (ésta sesión es creada por el protocolo de enlace). La conexión hace referencia a un enlace lógico entre el cliente y el servidor [5].

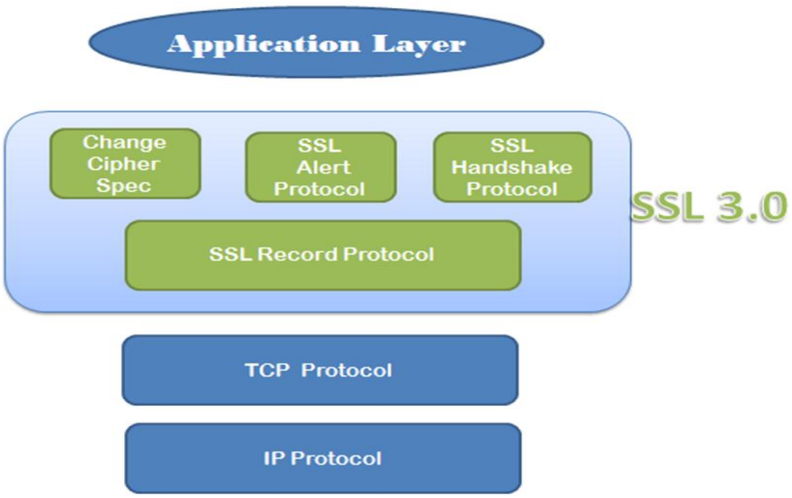


Figura 3. Arquitectura del protocolo SSL

El protocolo SSL/TLS permite asegurar las comunicaciones cliente-servidor a través del protocolo HTTP y también a través de los protocolos FTP, POP e IMAP. SSL/TLS actúa como una capa adicional que garantiza la seguridad de los datos y se ubica entre la capa de aplicación y la capa de transporte del modelo OSI [1].
Es decir, que el protocolo SSL trabaja sobre el protocolo TCP pero también puede ser utilizado por otros protocolos. También se dice que SSL opera entre la capa de transporte y de sesión del modelo OSI, o entre la capa de transporte y de aplicación del modelo TCP y está formado por dos capas y cuatro componentes que se describen a continuación:

El SSL *Record Protocol*

Este protocolo garantiza seguridad e integridad de los datos, recibiendo los datos de las capas superiores (*Handshake, Alert,* CCS, HTTP, FTP, etc.) y transmitiéndolos a la capa TCP después de realizar el procedimiento esquematizado en la figura 4:

a. Fragmentación o división del mensaje en bloques de mínimo 16Kb y máximo de 214Kb.
b. Compresión de los fragmentos por medio de SSL.
c. Generación del resumen de transmisión para garantizar el servicio de integridad.
d. Cifrado de datos para garantizar el servicio de confidencialidad.

Este protocolo agrega un encabezado al mensaje listo para enviarse, indicando el tipo de mensaje según su origen: *Handshake, Alert*, CCS, o datos de aplicaciones como HTTP y FTP; la versión del protocolo SSL utilizado, la longitud de los bloques de datos encapsulados y los cálculos para la MAC. Es responsable de la verificación de la integridad del mensaje incluido en el registro de transmisión. Es decir, que es el resultado de una función *hash* que sigue un algoritmo específico de *hash*, por ejemplo, MD5 o SHA-1. MAC se determina como resultado de una

función de dispersión que recibe los datos siguientes: MAC = función *hash* [clave secreta, datos primarios, el relleno y número de secuencia]. La clave secreta es creada por el cliente o el servidor, dependiendo de quién genere el mensaje o paquete. Después de recibir la información, el receptor calcula su propio valor de la MAC y lo compara con la recibida. Si los valores son iguales, significa que los datos no se han modificado durante la transmisión a través de la red. Después los datos y la MAC son encriptados utilizando un algoritmo de cifrado simétrico predeterminado, por ejemplo, el DES o el triple DES [6] [7].

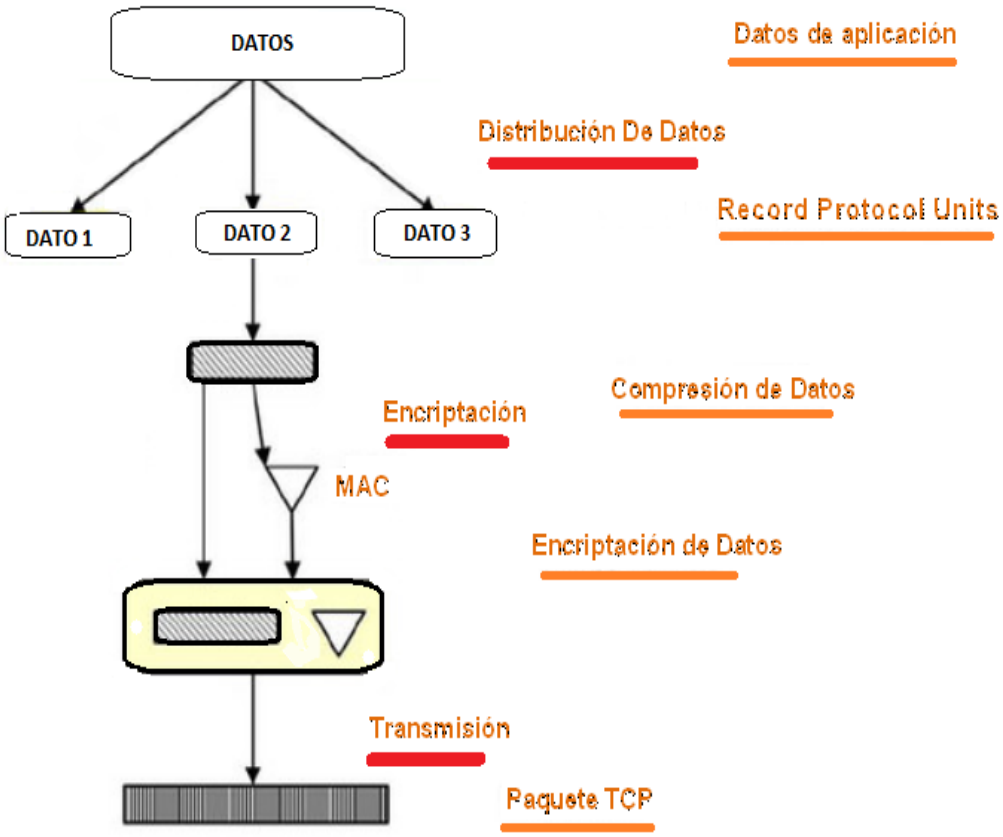


Figura 4. Creación de un paquete de información mediante el "*SSL record protocol*"

El "*Alert Protocol*"

Es utilizado por las partes para transmitir mensajes de la sesión asociados con el intercambio de datos y el funcionamiento del protocolo. Cada mensaje se compone de dos bytes; el primero siempre tiene un valor que determina la gravedad del mensaje enviado y el segundo contiene uno de los códigos de error definidos, que pueden ocurrir durante la comunicación SSL. El envío de un mensaje en el que su "gravedad" es alta, genera la terminación de la sesión SSL [8].

El "ChangeCipher Spec Protocol"

Este protocolo es el más sencillo. Se compone de un solo mensaje que contiene el valor 1. El único propósito de este mensaje es para que el estado de la sesión quede en espera. Por ejemplo, en la definición del conjunto de protocolos utilizados, éste tipo de mensaje debe ser enviado por el cliente al servidor y viceversa. Tras el intercambio de mensajes, el estado de sesión se considera de acuerdo. Este mensaje y los mensajes SSL son transferidos utilizando el SSL *Record Protocol.*

El "*Handshake Protocol*"

En este protocolo hay dos fases principales. La primera establece la conexión y autenticación del servidor y la segunda autentica el cliente. Durante la primera fase, el cliente inicia la conexión con el servidor SSL mediante el envío de un mensaje *cliente-hello* [8]. El servidor responde entonces con un mensaje similar tipo *Server-Hello.* Estos mensajes son utilizados para dar a conocer ciertas características del emisor y receptor como lo utilizaron los algoritmos de cifrado conocidos y preferidos, las longitudes máximas de clave que admite para cada uno de ellos, las funciones *hash* y los métodos de compresión a utilizar.

Después de haberse enviado el mensaje *Server Hello*, el servidor puede enviar su Certificado (X.509) para que sea autenticado por el cliente y que, además, este

reciba su clave pública. Si no es así, le envía al cliente su clave pública mediante un mensaje *Server Key Exchange* [8].

2.5 Funcionamiento del protocolo SSL/TLS

Para establecer una comunicación segura con SSL, se debe abrir un navegador e invocar una URL utilizando el protocolo *https*. Si la solicitud de conexión es aceptada por el servidor, comienzan los procesos de autenticación de usuarios, negociación de claves y certificados y encriptamiento o cifrado de la información, con el fin de constatar la seguridad del canal y que los datos arriben de manera íntegra a su destino [1].

Según el modelo de referencia OSI el protocolo SSL funciona en el nivel de aplicación y se apoya en el protocolo de trasporte TCP utilizando el puerto 443 del mismo. A continuación y mediante el ejemplo de la figura 5, se explica el funcionamiento del SSL utilizando una conexión segura al sitio Web de *Hotmail* o del banco *Davivienda*. Si los servidores constatan el correcto funcionamiento del protocolo *https*, responden con su información a través de una ventana en donde incluyen en la línea de la dirección Web, un candado. Si no aparecen mensajes de error, se garantiza entonces que la conexión es segura. La figura 5, muestra la apariencia de los encabezados presentados en estas ventanas.

El procedimiento general de conexión, autenticación y entrega exitosa de la información mediante un canal seguro protegido con SSL, se puede observar en la figura 6.

Figura 5. Verificación de conexión segura mediante https

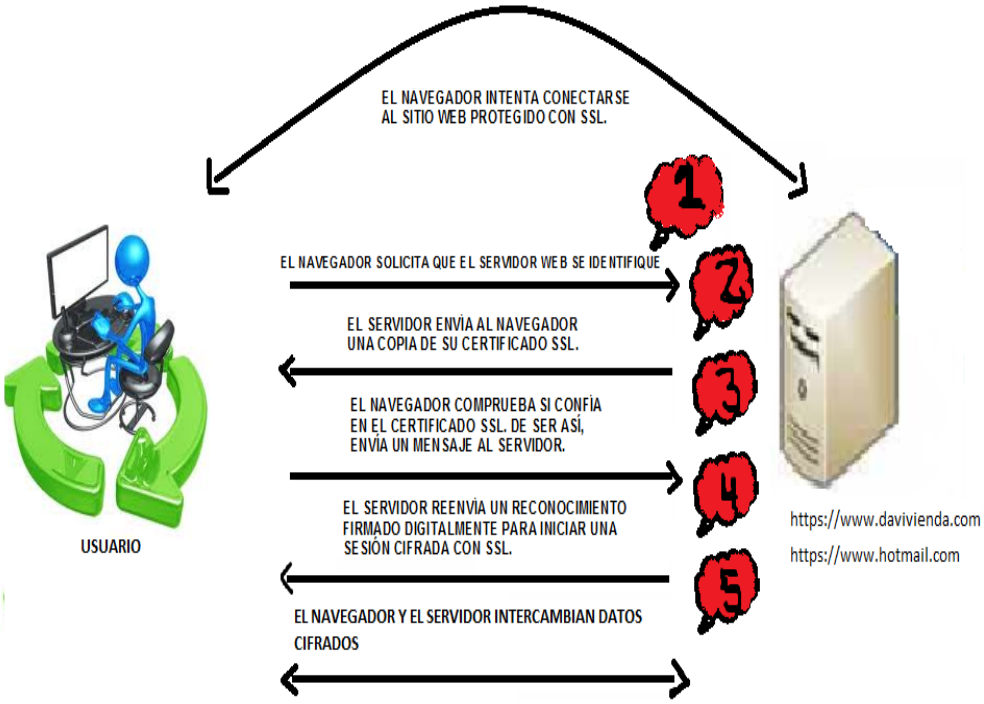
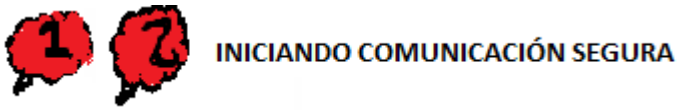


Figura 6. Funcionamiento de SSL/TLS

A continuación se describen los pasos 1 al 5 estipulados en la figura 6:



El navegador realiza la petición del usuario al sitio Web correspondiente (Hotmail o Davivienda), y con base en esta información, el servidor Web responde con un

mensaje informando que está de acuerdo en establecer la conexión segura con los datos de SSL/TLS proporcionados (*https*).

Una vez que ambos conocen los parámetros de conexión, los servidores transmiten al navegador sus certificados digitales para identificarse como sitios confiables [3].

VERIFICACIÓN DE VALIDEZ DEL CERTIFICADO

Una vez que el certificado es recibido por el navegador, se realizan las siguientes verificaciones:

Integridad: descifrando la firma digital incluida en él mediante la llave pública de la entidad certificadora (AC) y comparándola con una firma del certificado generado en ese momento. Si ambas son iguales se concluye que el certificado es válido.

Vigencia: revisando el periodo de validez del certificado, es decir, la fecha de emisión y la fecha de expiración incluidas en él. Ver figura 7.

Emisor: haciendo uso de una lista de Certificados Raíz, almacenados en el computador y que contienen las llaves públicas de la AC conocidas y de confianza. Con base en esta lista, el navegador revisa que la AC del certificado sea de confianza; si no lo es, el navegador mostrará una advertencia indicando que el certificado fue emitido por una entidad no fiable [3].

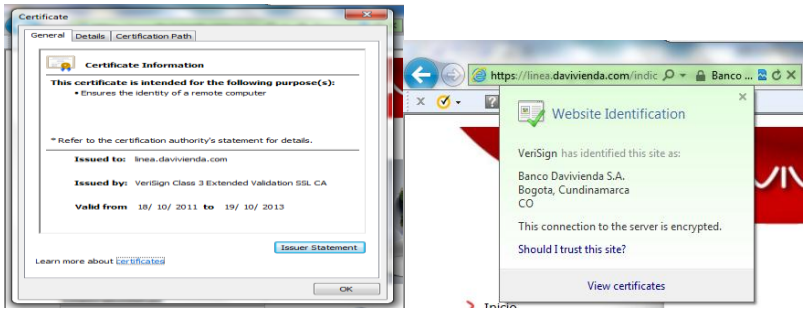


Figura 7. Verificación de vigencia del certificado digital

ESTABLECIENDO LA CONEXIÓN SEGURA

Si el certificado cumple con todas las pruebas del navegador, se establece la conexión, lo cual se traduce en protección para los valiosos datos personales [3].

2.6 Ataques contra el protocolo SSL/TLS

El protocolo SSL/TLS resiste los siguientes ataques:

Lectura de los paquetes enviados por el Cliente y el Servidor

Debido a que los datos viajan cifrados, un atacante que quiera leer los paquetes enviados utilizando técnicas de *Sniffer*, se enfrenta al problema de romper el cifrado de los datos, si quiere interpretar su contenido. Las claves utilizadas para el cifrado se intercambian con métodos de clave pública que el atacante tendría que romper si quiere conocer los valores asignados.

También es importante que la comunicación sea autenticada tanto en el cliente como en el servidor, porque existe la posibilidad de capturar las claves secretas. En esta captura, el espía genera sus propias claves públicas y privadas, y dado que el intercambio es anónimo, el receptor no tiene manera de saber si la clave pública que recibe, es la del emisor auténtico o no. En cambio, si se realiza la

autenticación de servidor y/o cliente, es necesario enviar un certificado donde tiene que estar la clave pública del emisor firmada por una autoridad de certificación que el receptor reconozca, y por tanto, no puede ser sustituida por otra [12].

Suplantación de Servidor o Cliente

Cuando se realiza la autenticación del servidor o cliente, el certificado digital debidamente firmado por la AC sirve para verificar la identidad de su propietario. Un atacante que quiera hacerse pasar por el servidor (o cliente) debería obtener su clave privada, o bien la de la AC que ha emitido el certificado, para poder generar otro con una clave pública diferente y que parezca auténtico.

Alteración de los paquetes

Si un atacante intenta modificar los datos enviados que están cifrados, no podrá conocer el contenido final descifrado. Si pasa esto, el receptor detecta que el paquete ha sido alterado porque el código de autenticación (MAC) es incorrecto. Si la alteración se realiza en los mensajes de negociación, cuando aun no se aplica ningún código MAC con la finalidad por ejemplo de forzar la adopción de algoritmos criptográficos más débiles y vulnerables, ésta manipulación, será detectada en la verificación de los mensajes *Finished*.

2.7 Autoridad Certificadora (AC)

Con el objetivo de aumentar la seguridad en el intercambio de información en una red, se recurre al servicio de la AC, que es quien certifica la identidad de las partes comunicantes a través de un certificado.

Una Autoridad de Certificación, es una organización o empresa que emite certificados. Existen entidades de certificación públicas y privadas. Ésta vincula el certificado con una clave pública y un nombre específico. El certificado debe estar

firmado por la clave privada del emisor el cual contiene la información necesaria para verificar su validez, además el emisor debe contar con su propio certificado de clave pública para completar la validez del mismo.

La AC tiene la responsabilidad de asegurar que los certificados que emite son legítimos. Es decir, debe garantizar que todos los certificados que expidan, contengan una clave pública emitida por la parte que afirma haberlo emitido.
Los encargados de autorizar la creación de una autoridad de certificación o prestador de servicios de certificación en Colombia son:

1. La sociedad comercial de certificación digital (Certicamara)
2. La Gestión de Seguridad Electrónica (GSE)
3. El *Andesscd*
4. El *Verising*

VeriSign es la Autoridad de Certificación Digital líder en el mundo en la emisión de certificados SSL para el aseguramiento de transacciones de comercio electrónico, comunicaciones, e interacciones para sitios Web, Intranet y Extranet. Ésta se encuentra unida a Certicámara S.A., aliado estratégico y representante en el país para la comercialización de sus productos y soluciones de seguridad para sitios, aplicaciones y servicios Web. A través de Certicámara se pueden adquirir los certificados digitales [10].

Una AC privada es implementada en organizaciones o entornos corporativos. Tiene la responsabilidad de emisión de certificados para los miembros de su propia organización. Programas como OpenSSL permiten crear una AC propia configurando todos sus parámetros y generando certificados que son implementados en este tipo de organizaciones sin que esto genere ningún costo.

Una AC pública es una entidad de certificación que generalmente emite certificados de sitios Web públicos que requieren de cifrado y autenticación. Para una entidad de certificación pública, verificar la identidad de un sujeto es más difícil de lo que es para una empresa AC privada. La información requerida para demostrar su identidad a la entidad emisora del certificado varía dependiendo de si se trata de un individuo o un negocio, ya que para éste último es necesario la documentación del gobierno que acredite y certifique su nombre y funcionamiento, en cambio, para un individuo las pruebas son más simples [4].

Manejo de certificados

Los certificados poseen jerarquías, de esta manera un certificado emitido por una AC con los permisos necesarios, puede ser utilizado para firmar otro certificado. La AC contiene el certificado raíz el cual no tiene ninguna autoridad que firme, de ahí su nombre. Para verificar la autenticidad y la validez de un certificado, cada certificado de la cadena también debe ser verificado. Si algún certificado de la cadena no es válido, cada certificado debajo en la cadena, también debe ser considerado como no válido. Los certificados no válidos por lo general han caducado o han revocado.

El estándar X.509 es uno de los estándares más utilizados para la generación de certificados por parte de las CA, en infraestructuras de claves públicas (PKI) utilizando formatos como DER (*Distiguish Encoding Rules*) o PEM (*Privacy Enhanced Mail*) entre otros. El formato PEM es el que se utilizará en la implementación del protocolo SSL.

La norma define 14 extensiones para X.509v3 en un esfuerzo de estandarizar las extensiones más comunes llevadas a cabo por terceras partes. Un ejemplo son los usos permitidos para un certificado (si un certificado está autorizado a firmar otro certificado, o se puede utilizar en un servidor SSL). Si cada aplicación tuviera que crear sus propias extensiones diferentes, la información en las extensiones

debe ser utilizable por otras aplicaciones o complicarían considerablemente el proceso de validar el certificado, pues se tendrían un número prácticamente ilimitado de extensiones diferentes [4].

El estándar X.509 permite a SSL implementar algoritmos que proveen verificación de identidad de certificados digitales X.509 haciendo posible:

- La generación de parejas de claves RSA de 512, 1024 y 2048 bits.
- La generación de certificados X.509 relacionados con los usuarios y servidores.
- La renovación de un certificado.
- La exportación de un certificado.
- La revocación de un certificado.
- Gestionar la lista de certificados revocados.
- Generar la clave privada y el certificado de la entidad emisora de certificados.

2.8 Certificado de Firma Digital

Son documentos electrónicos que emite CERTICAMARA en Colombia y que permiten identificar a una persona en medios digitales. Adicionalmente, califica tanto su actividad profesional, como el rol que desempeña en el momento.
La certificación de Firma Digital garantiza:

- Identidad de las partes que tratan entre sí, sin conocerse (emisor y receptor del mensaje).
- Integridad de la transacción (verificar que la información no fue manipulada).

- No puede negar el conocimiento de un mensaje y los compromisos adquiridos (no repudiación).
- Confidencialidad de los contenidos de los mensajes (solamente conocidos por quienes estén autorizados).

Es decir, que mediante la utilización de certificados digitales es posible firmar digitalmente información electrónica obteniendo los siguientes atributos jurídicos:

Autenticidad que permite garantizar la identidad del emisor de un mensaje y/o el origen del mismo, y tener la plena seguridad que quien remite el mensaje es realmente quien dice ser. **Integridad** que garantiza que la información electrónica no ha sido alterada.

De igual manera, la tecnología de certificación digital permite el cifrado de mensajes de datos incorporando el atributo adicional de la **confidencialidad** que garantiza que un mensaje de datos no pueda ser conocido sino por su emisor y los receptores deseados. El contenido de la información no podrá ser conocido por ningún tercero no autorizado. Las firmas digitales generadas mediante el uso de certificados digitales emitidos por Certicámara cuentan con el mismo valor probatorio y fuerza obligatoria de una firma manuscrita [10].

2.9 Certificado Digital SSL/TLS

Es un documento digital único que garantiza el vínculo entre una persona o entidad con su llave pública.

Este certificado contiene información de su propietario, como su nombre, su organización e incluso su llave pública; también contiene información propia del certificado digital como periodo de validez, número de serie, nombre de la AC,

firma digital de la AC cifrada con su llave privada y otros datos que describen cómo usar ese certificado [3].

Los certificados digitales son archivos electrónicos que se utilizan para identificar a las personas y organizaciones a través de la red. También permiten una comunicación segura y confidencial entre el cliente y el servidor. Los certificados son emitidos por entidades llamadas Autoridades Certificadoras (AC) cuyo objetivo es validar la información para verificar su verdadera identidad.
La firma de la AC en el certificado, es como un sello de garantía [7].

Los certificados digitales son un mecanismo que les permite a los clientes y usuarios obtener confianza a la hora de realizar tareas o actividades por medios electrónicos. Un certificado digital garantiza:

- **Autenticidad** de la identidad del aplicativo, del sitio o del servicio Web, ya que el certificado digital SSL contiene información protegida que identifica a sus propietarios, reduciendo así las posibilidades de suplantación de identidad o *phishing*.
- **Confidencialidad** de la información que viaja por la red entre el cliente y el aplicativo, el sitio o el servicio Web, ya que la información viaja cifrada entre las dos partes garantizando que la información solamente pueda ser conocida por los autorizados.
- **Integridad** de la información intercambiada entre los clientes y el aplicativo, el sitio o el servicio Web, ya que al viajar de forma segura garantiza que no va ser modificada por terceros [10].

2.9.1 Clasificación de los certificados SSL/TLS emitidos por la AC

Los siguientes son los tipos de certificados digitales SSL/TLS que puede emitir una entidad certificadora:

Certificado con validación ampliada: Antes de emitir este certificado, la AC, realiza las siguientes actividades:

- Comprobación de la existencia jurídica, física y operativa de la entidad.
- Comprobación de la identidad de la entidad de acuerdo con los registros oficiales.
- Verificación del derecho de la entidad a utilizar el dominio que se especifica en el certificado con validación ampliada.
- Verificación de autorización para la emisión del certificado con validación ampliada.

Esta certificación está disponible para cualquier tipo de entidad. Para mayor información revisar por favor, las referencias [10] y [13].

Una vez instalado el certificado, se puede verificar su funcionamiento observando las siguientes características en la barra de navegación (ver figura 8).

a) La barra de navegación pasa de blanco a verde indicando que el sitio Web está utilizando SSL con validación ampliada.
b) El nombre del propietario de la página Web aparece en la barra de navegación.
c) El protocolo HTTP estándar se convierte en HTTPS.
d) Se activa el candado para indicar que la conexión es segura.

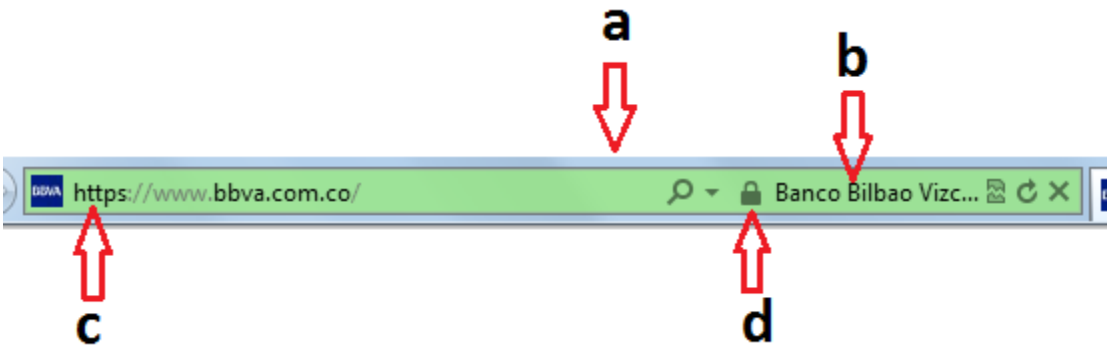


Figura 8. Verificación de funcionamiento del certificado digital con validación ampliada

Certificado con validación de organización: La información corporativa es verificada generalmente por un tercero, y para esto, la AC no deberá tomarse más de dos días laborales. Además la información se muestra al cliente garantizando una mayor credibilidad y seguridad sobre su propietario. Este tipo de certificado digital se identifica en la barra del navegador con las siguientes características (ver figura 9):

a) El protocolo HTTP estándar se convierte en HTTPS.
b) Se activa el candado.

Certificado con validación de dominio: Es un certificado rápido y económico el cual es utilizado por organizaciones que no necesitan o no desean pasar por el proceso de inspección corporativa y requieren del certificado de manera inmediata, ya que solo verifican la propiedad del nombre de dominio. Este tiene la misma apariencia en la barra de navegador correspondiente a la validación de la organización de la figura 9.

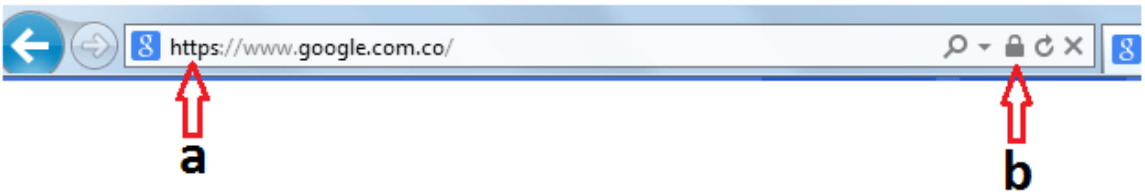


Figura 9. Certificado con validación de organización

Certificaciones ofrecidas por VERISING

- **Certificado digital de sitio seguro:** Este certificado protege la transferencia de datos de las páginas Web, intranet y extranet y posee las siguientes características:

  - Autenticación empresarial completa.
  - Cifrado de 40 a 256 bits.
  - Garantía de USD $100.000.
  - *VeriSign Secured® Seal.*
  - Verificador de instalación.
  - *Seal-in-Search™.*

- **Certificados SSL de 128 bits reales**: Este certificado permite a todos los usuarios de su página Web contar con el cifrado más potente disponible para SSL. Posee las siguientes características:

  - Cifrado mínimo entre de 128 a 256 bits.
  - Autenticación empresarial completa.
  - Garantía de USD $1.250.000.
  - Sello *VeriSign Secured® Seal.*
  - Verificador de instalación.
  - Escaneo de *Malware [19].*

- Escaneo de vulnerabilidades.
- *Seal-in-Search™.*

- **Certificados SSL con *Validación Extendida***: ofrece mayor confianza a sus clientes a la hora de hacer compras en línea. Tiene las siguientes características:

  - Autenticación empresarial completa.
  - Validación Ampliada. Barra de direcciones verde.
  - Garantia de USD $1.500.000.
  - Sello *VeriSign Secured® Seal.*
  - Verificador de instalación.
  - Escaneo de *Malware.*
  - Escaneo de vulnerabilidades.
  - *Seal-in-Search™.*

- **Certificado SSL con *Validación Extendida Real de* 128-bits**: Este certificado permite a todos los usuarios de su página Web contar con el cifrado más potente disponible para SSL. Tiene las siguientes características:

  - Autenticación empresarial completa.
  - Validación Ampliada. Barra de direcciones verde.
  - Cifrado de mínimo 128 a 256 bits.
  - Garantia de USD $1.500.000.
  - Sello *VeriSign Secured® Seal.*
  - Verificador de instalación.
  - Escaneo de *Malware.*
  - Escaneo de vulnerabilidades.
  - *Seal-in-Search™.*

- **Certificados SSL SAN "Subjet Alternative Names" Geo Trust**: Certificado SSL *True BusinessID de GeoTrust*® para navegadores Web y teléfonos móviles, permite asegurar múltiples dominios con una sola IP. Sus características son:

  - Autenticación empresarial completa.
  - Cifrado de 40 bits a 256 bits.
  - Garantía USD $ 250.000.
  - SAN.
  - Incluye 5 SAN. Cada dominio adicional tendrá un costo de USD $ 55.00 más IVA.
  - Solo aplica para UC, Exchange Server 2007/2010, Office Communications Server 2007/2010 entre otros.

- **Certificados SSL SAN *Geo Trust* con validación ampliada**: Certificado SSL *True BusinessID de GeoTrust*® para navegadores Web y teléfonos móviles, ofrece una señal visible y apropiada de que el usuario está en un sitio autenticado de alta fiabilidad y que la información de sus clientes está segura, permite también, asegurar múltiples dominios con una sola IP. Sus características son:

  - Autenticación empresarial completa.
  - Cifrado de 40 a 256 bits.
  - Validación ampliada, barra de direcciones verde.
  - Garantía de USD $ 250.000.
  - SAN.
  - Incluye 5 SAN. Cada dominio adicional tiene un costo de USD $ 55.00 más IVA.

- Solo aplica para UC, Exchange Server 2007/2010, Office Communications Server 2007/2010 entre otros. Para más información revisar las referencias [10] y [13].

2.9.2 Procedimiento para la obtención de un Certificado Digital SSL

Para solicitar un certificado SSL, primero se debe escoger la organización o AC autorizada y luego se debe diligenciar el formato de petición. En la figura 10, se puede ver un ejemplo para la solicitud de este certificado, a través de la página Web http://www.thawte.com/.

Como se probarán los conceptos y se llevará a cabo la configuración del Protocolo SSL con programas de licencia libre, este sitio ofrece un certificado para su funcionamiento y prueba durante 21 días. Seleccionar la opción "*trial"* y proceder a ingresar los datos solicitados en el formulario de la figura 10.

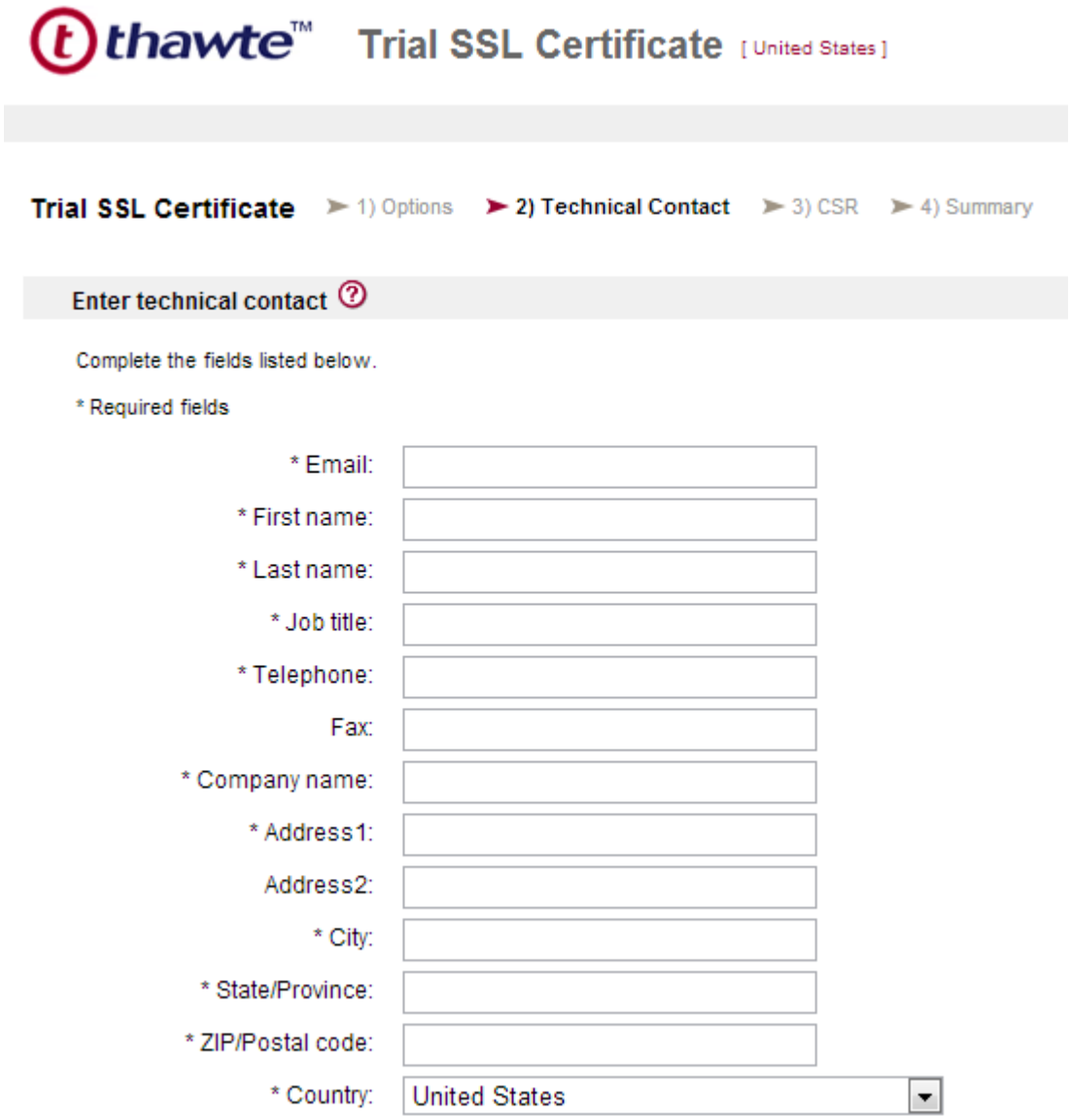


Figura 10. Formato de solicitud del certificado digital

Después de haber enviado el formulario de petición, se ingresa la solicitud de firma de certificado (*Certificate Signing Request* (CSR)) en la zona titulada "espacio para copiar el CSR" según se muestra en la figura 11. Una vez enviado el formulario, la AC emite el respectivo certificado digital [14].

Thawte envía los certificados digitales al correo electrónico ingresado en la solicitud. Los certificados son los siguientes:

- *Thawte trial SSL certificate.*
- *Thawte Trial Secure Server Intermediate CA.*
- *Thawte Test CA Root certificate.*

Estos corresponden al certificado digital, intermedio y raíz, que son necesarios a la hora de configurar el protocolo SSL en el servidor.

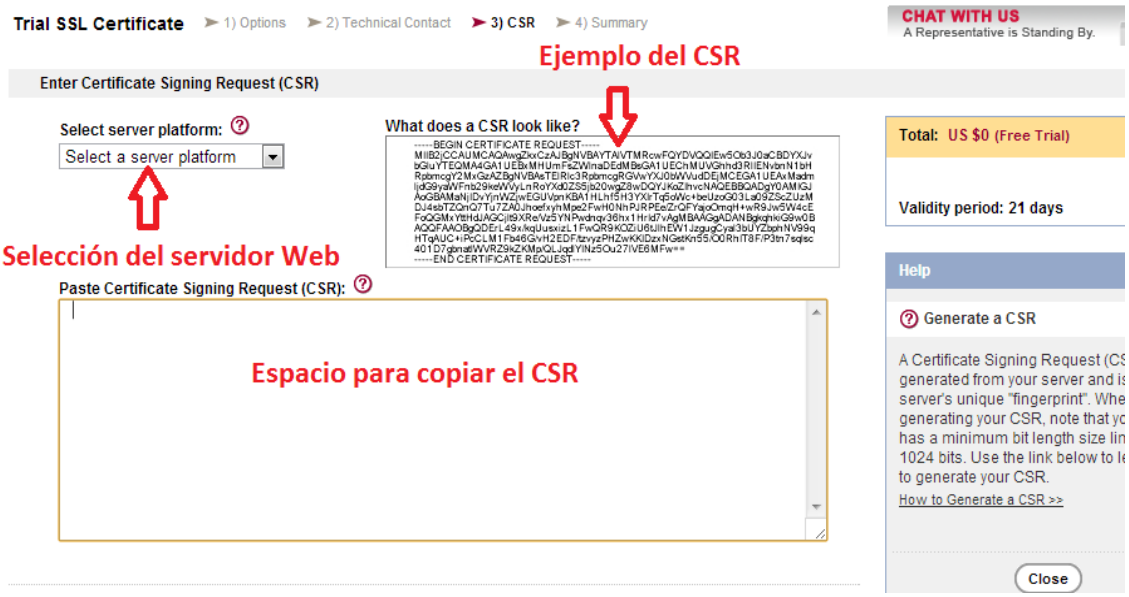


Figura 11. Formulario de petición de firma del certificado

3. REQUERIMIENTOS PARA LA CONFIGURACIÓN DEL PROTOCOLO SSL/TLS

3.1 Generalidades

Para la configuración del protocolo SSL/TLS se requieren conocimientos de administración del sistema operativo que contiene el protocolo y conocimientos técnicos del software **OpenSSL** que implementa dicho protocolo; así como del **servidor Web** que es quien ofrece las conexiones.

Para el caso de estudio, se han seleccionado el sistema operativo **Linux Ubuntu** y el **servidor Web Apache** por ser aplicaciones de libre distribución.

Los principios básicos del sistema operativo Linux Ubuntu son:

- Libertad para el usuario de descargar, ejecutar, distribuir, cambiar, estudiar, compartir y mejorar su software para cualquier propósito sin tener que pagar derechos de licencia.
- Está escrito en múltiples lenguajes lo que le permite al usuario llevar a cabo su instalación y configuración de una manera personalizada de acuerdo con sus preferencias linguísticas.
- Facultar al usuario independientemente de sus condiciones físicas y mentales, la utilización completa de todas sus herramientas y aplicaciones.

Ubuntu publica una nueva versión en el mes de abril y otra en el mes de octubre. Este número de versión se incluye en el nombre de su paquete-software lo cual se refleja en la numeración de sus versiones. La última versión estable de Ubuntu es la 12.10, la cual se liberó en el mes de octubre del año 2012. Cada versión de Ubuntu recibe soporte al menos durante 18 meses con actualizaciones genéricas y de seguridad y se puede descargar gratis de su página oficial [9].

Igualmente, se requiere de la configuración de un servidor **NTP** encargado de sincronizar los relojes de los computadores a través de Internet y de un **Servidor DNS BIND9** para la conversión de nombres de máquina y números IP. Este servidor el más comúnmente utilizado en Internet e implementado específicamente en sistemas Unix.

El **servidor Web Apache** es un servidor HTTP (Web). Su *software* emplea código abierto y fue desarrollado por Apache *Software Foundation* el cual se basa en una arquitectura modular. El objetivo de este, es ofrecer un servidor seguro, eficiente y extensible con servicios HTTP en sincronía con los estándares HTTP actuales, generando así sistemas de calidad comercial.

Uno de los módulos principales con los que cuenta Apache es el *mod_ssl* necesario para la implementación del protocolo SSL/TLS, que permite trabajar con las versiones SSLv1, SSLv2, SSLv3 y TLSv1.

El **OpenSSL** es un conjunto de herramientas de administración desarrolladas por *Eric A. Young y Tim J. Hudson*, a nivel comercial que contiene todas las funciones y aplicaciones del *Open Secure Sockets Layer* (SSL v2/v3) y del *Transport Layer Security* (TLS v1). Este conjunto de herramientas cuenta con un servicio completo y un uso general de la biblioteca de criptografía.

OpenSSL permite configurar una estructura de PKI (*Public Key Infrastructure*) implementada con una serie de certificados y claves tanto públicas como privadas. Esencialmente utiliza dos herramientas en una, pues cuenta con una biblioteca de criptografía que proporciona los algoritmos conocidos de cifrado simétrico como RC4, 3DES y AES, criptografía de clave pública como DSA y RSA, algoritmos de hash MD5 de hasta 128 bits y SHA1 de 160 bits, y cuenta

además con un conjunto de herramientas de SSL que implementa todas las versiones del protocolo SSL, incluyendo la TLSv1.

Igualmente esta aplicación ofrece apoyo a la manipulación de formatos de certificación como X.509v3 y gestión de claves. También paquetes de precisión de matemática arbitraria y generación de números aleatorios seguros para una programación más robusta en criptografía.

Este software es libre de obtener y utilizar tanto para fines comerciales y no comerciales. Todas las implementaciones del SSL están actualmente disponibles con OpenSSL y funcionan sobre las principales plataformas, incluyendo los sistemas operativos Unix y las versiones comunes de Microsoft Windows [5].

3.2 Instalación y configuración del OpenSSL

La versión 1.0.1c de OpenSSL se instalará y configurará en una máquina con sistema operativo Linux Ubuntu11.04 versión de kernel 2.6.38-8-generic i686.

Para esto se deben tener en cuenta los siguientes pasos:

1. Abrir una terminal de comandos e ingresar:
   $sudo -i

2. Para acceder como usuario privilegiado (root o admin) se digita la clave de usuario a continuación, cuando este se solicite.

3. Si la autenticación fué exitosa, moverse a la carpeta raíz del usuario root, en este caso ***/home/host/*** utilizando el comando:
   #cd /home/host/

4. Crear una carpeta con el nombre *descargas*, para guardar el archivo de instalación de la aplicación. Para esto utilizar el comando:

   #mkdir descargas

5. Asignar permisos de lectura, escritura y ejecución en la carpeta creada en 4.

   #chmod 777 descargas

6. Abrir un navegador e ir a la página Web http://www.openssl.org/ que contiene el paquete o*penssl-1.0.1c.tar.gz* y descargarlo en la carpeta creada en 4, o sea ***/home/host/descargas.***

7. Ir a la carpeta ***descargas*** desde un comando de la terminal y descomprimir el archivo descargado así:

   #cd /home/host/descargas

   #tar xzvf openssl-1.0.1c.tar.gz

8. Moverse a la carpeta raíz del OpenSSL creada después de la descompresión del archivo realizado en 7:

   #cd /etc/descargas/openssl-1.0.1.c/

9. Compilar el programa utilizando los siguientes comandos:

   #./config

   #make

   #make test

   #make install

10. Crear la carpeta donde residirán los certificados provenientes de la instalación del OpenSSL y que permitirán instalar el SSL en el servidor Web. Para esto ejecutar los siguientes comandos:

#cd /etc

#mkdir certificados

- **Creación de la AC**

1. Crear las carpetas donde residirán los certificados provenientes de la AC. En este caso y para el ejemplo, las carpetas ***uisac***, ***private y certs.*** Para esto ejecutar los siguientes comandos:

   # cd /etc/ssl

   #mkdir uisac

   #cd /etc/ssl/uisac

   #mkdir private certs

2. Crear los archivos de texto que servirán para controlar los certificados:

   #echo '01' > serial

   #touch index.txt

3. Copiar el archivo de configuración a la carpeta **uisac** creada en 1, ejecutando los siguientes comandos:

   #cp /etc/ssl/openssl.cnf /etc/ssl/uisac/openssl.cnf

   #cd /etc/ssl/uisac

4. Abrir el archivo de configuración utilizando el siguiente comando:

   #gedit openssl.cnf

5. Reescribir la línea 8 indicando la ubicación de la carpeta que va a contener los certificados de la AC así:

   dir = /etc/ssl/uisac

Para que los certificados que se generen sean de 2048 bits también es necesario editar la línea 101, del mismo archivo así:

default_bits = 2048

6. Para crear los certificados de la AC utilizando OpenSSL se deben ejecutar los siguientes comandos:

   #cd /etc/ssl/uisac

   #openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem \-out cacert.pem -days 3650 -config ./openssl.cnf

7. Suministrar a continuación los datos solicitados. La siguiente secuencia introduce este proceso.

   Enter PEM pass phrase: *[ingrese una clave o frase de seguridad para el certificado]*

   Verifying - Enter PEM pass phrase: *[ingrese nuevamente la clave o frase]*

   Country Name (2 letter code)[AU]: *[Ingrese el país, para Colombia CO]*

   State or Province Name (full name)[some-state]: *[ingrese el departamento]*

   Locality Name (eg, city)[]: *[Ingrese la ciudad]*

   Organization Name (eg, company)[Internet Widgets Pty Ltd]: *[ingrese el nombre de la organización o compañía]*

   Organizational Unit Name (eg, sector)[]:

   Comun Name (eg, Your name)[]: [*ingrese el nombre de dominio ejemplo: host.server.com]*

Lo anterior permite la generación de los archivos *cakey.pem* y *cacert.pem* que son la clave privada y el certificado raíz de la AC respectivamente, los cuales se emplean para firmar los certificados emitidos por la AC. El nombre *host.especssl.com* es el nombre del servidor Web que se ha elegido para este ejemplo.

- **Creación de la clave privada del servidor y el archivo de solicitud del certificado digital**

  El siguiente procedimiento se realiza para probar que los certificados de la AC creada sirven para firmar otros certificados. Más adelante se mostrará que los certificados emitidos por la AC que se crearán, tienen similares características que un certificado de *Thawte* [14].

  Se crea la clave privada y el archivo de petición del certificado digital para el servidor *host.especssl.com* con el siguiente comando:

  ```
  #openssl req -new -nodes -out
  solicitud.host.especssl.com.pem -config ./openssl.cnf
  ```

  Los archivos resultantes serian el *privkey.pem* que contiene la clave privada y el *solicitud.host.especssl.com.pem*, que se utilizará para que la AC emita el certificado digital.

- **Firma por parte de la AC**

  La solicitud del certificado se firma utilizando el siguiente comando:

  ```
  #openssl ca -out certificado.host.especssl.com.pem -config
  ./openssl.cnf -days 3650 \-infiles
  solicitud.host.especssl.com.pem
  ```

  Ante la solicitud de la clave, se debe ingresar la que fue asignada al *cacert.pem*.

En resumen, los certificados resultantes generados que residen en la carpeta `/etc/ssl/uisac` son:

- Certificado raíz: *cacert.pem*
- Clave privada del servidor: *privkey.pem*
- Certificado emitido por la CA: *certificado.host.especssl.com.pem*

Se pueden utilizar certificados digitales emitidos por diferentes entidades de certificación sin ningún inconveniente, siempre que utilicen cifrado de clave pública.

4. INSTALACIÓN Y CONFIGURACIÓN DEL PROTOCOLO SSL/TLS

Este capítulo muestra los pasos a seguir para la configuración del protocolo SSL/TLS en un servidor Web Apache, donde se utiliza OpenSSL para generar la clave privada y la solicitud de firma de certificado (CSR), procediendo a la instalación de la versión de prueba gratuita del certificado emitido por la AC *Thawte* como se presentó en la sección 2.9.2.

Con el protocolo SSL/TLS funcionando y el certificado correctamente instalado se realizan capturas de tramas de una petición al servidor Web por medio de un *Sniffer*. Las tramas son analizadas para comprobar que el protocolo SSL/TLS está cifrando y que la transferencia de información se realiza de forma segura.

4.1 Guía de configuración del protocolo SSL/TLS en un servidor Web Apache

Un servidor Web Apache se instaló en un computador con sistema operativo Linux Ubuntu. Al dominio IP para dicha máquina se le denominó *especssl.com* y el nombre utilizado para identificar la máquina fue *diego.*

Los siguientes pasos introducen este proceso de configuración:

1. Abrir una terminal de comandos e ingresar:
   $sudo -i

2. Para acceder como usuario privilegiado (root o admin) se digita la clave de usuario a continuación, cuando esté se solicite.

3. Moverse a la carpeta *certificados* (creada en el paso 10 de la sección 3.2) donde se guardarán los certificados reales emitidos por la AC *Thawte*, así cómo los generados durante la configuración del protocolo SSL. Ver esta acción en la figura 12.

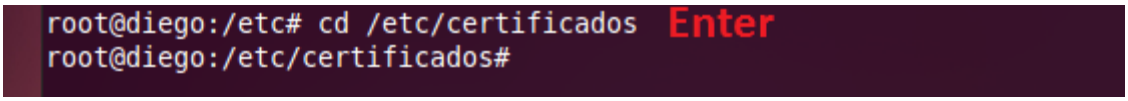


Figura 12. Acceso a la carpeta certificados

4. Generar la clave privada ejecutando el comando `openssl genrsa -des3 -out diego.especssl.com.key 2048` y a continuación ingresar la contraseña para la clave privada, cuando ésta se solicite, según se muestra en la figura 13. El sistema pedirá confirmación de esta

contraseña (ver figura 14). Ingresar entonces de nuevo la misma contraseña.

Mediante éste proceso se crea la clave privada RSA, la cual se cifra con el sistema de encriptación Triple DES de 2048 bits (configurado en la sección 3.2), y que se guarda con el nombre de *diego.especssl.com.key* en la carpeta *certificados*. El aspecto de esta clave se puede observar en la figura 15, resultado de la utilización del comando:

```
root@diego:/etc/certificados# gedit diego.especssl.com.key
```

Figura 13. Generación de clave privada

Figura 14. Confirmación de contraseña

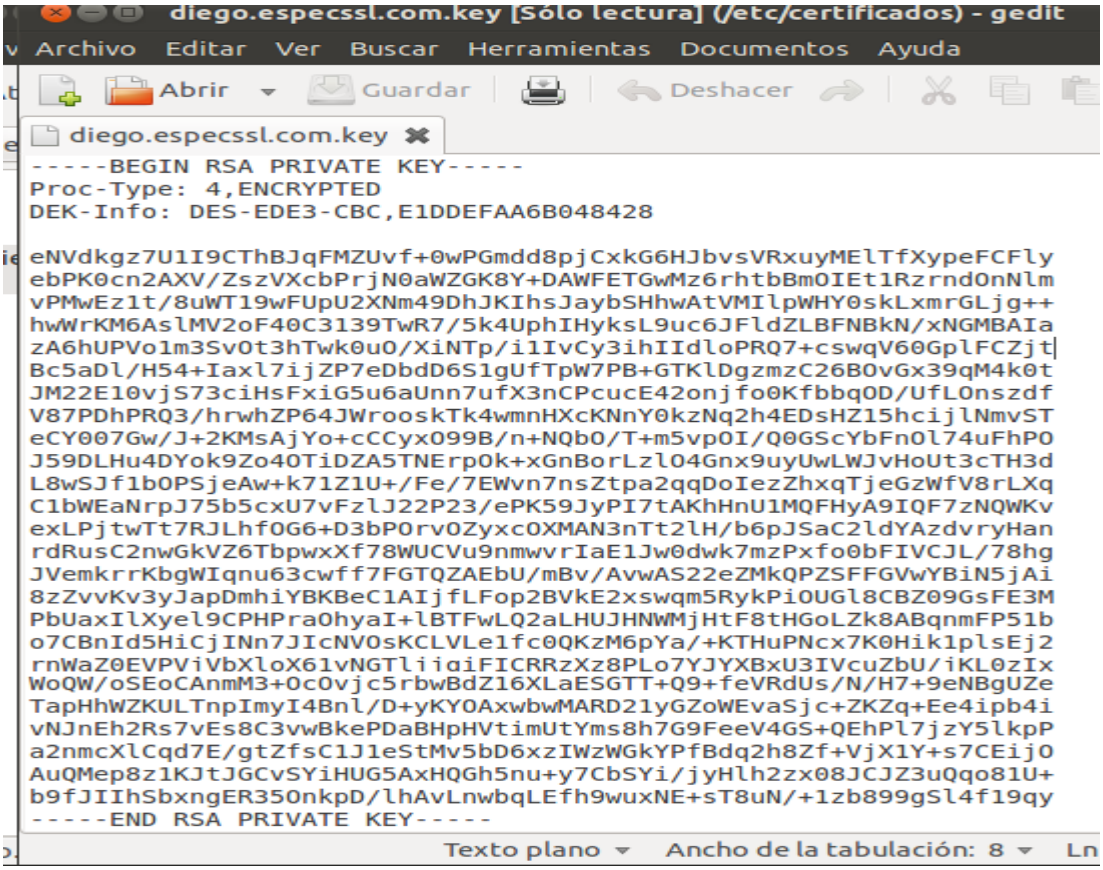


Figura 15. Clave privada

Nota: Los 48 bits son requisito del proveedor del certificado, en este caso *Thawte*.

5. Generar el CSR (archivo solicitado por la entidad encargada de emitir el certificado SSL como se explicó en la sección 2.9.2), ejecutando el comando `openssl req -new -key diego.especssl.com.key -out diego.especssl.com.csr` y a continuación ingresando la contraseña de la clave privada cuando ésta se solicite. Ver este proceso en la figura 16.

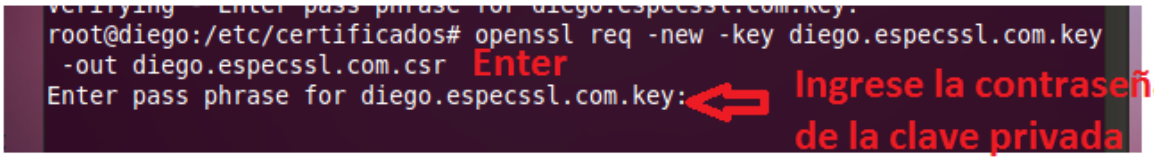


Figura 16. Generación del CSR

6. Suministrar los datos solicitados a continuación según se muestra en la figura 17.

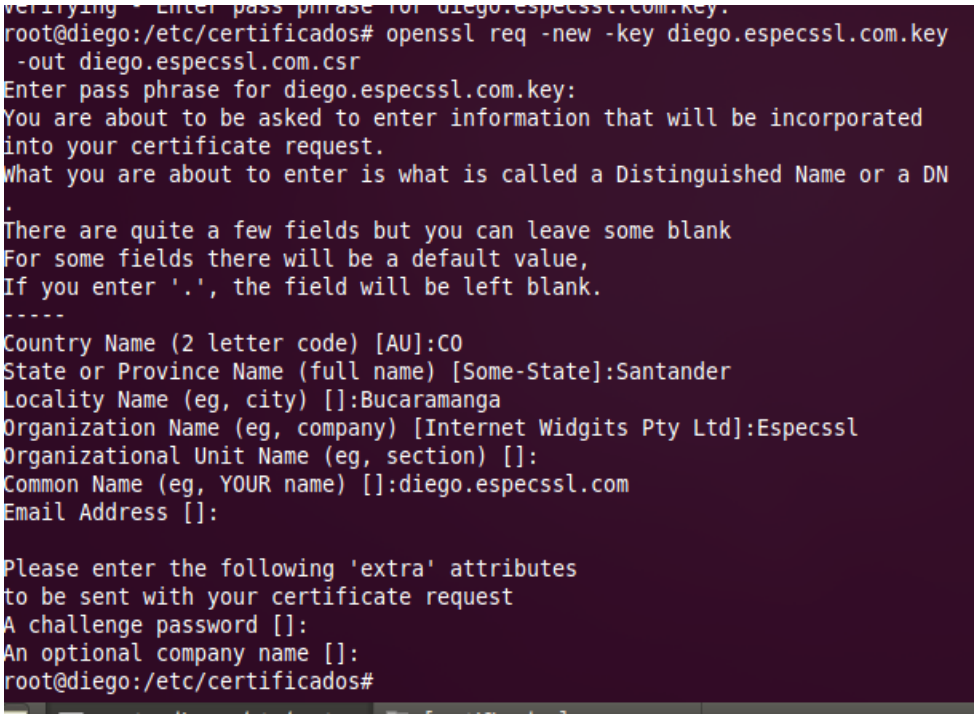


Figura 17. Ingreso de datos del usuario del certificado solicitado

El archivo CSR se genera con el nombre *diego.especssl.com.csr* y se guarda en la carpeta *certificados.*

7. Descargar la versión de prueba gratuita del certificado SSL emitida por la AC *Thawte*, como se explica en la sección 2.9.2.

8. Identificar los certificados entregados por la AC y guardarlos con los nombres *certificado.diego.especssl.com.crt* para el certificado *"Thawte trial SSL certificate"* e *intermedio.diego.especssl.com.crt* para el certificado *"Thawte Trial Secure Server Intermediate CA"*.

   Hasta el momento, residirán entonces en la carpeta *certificados* los siguientes archivos:

   - *diego.especssl.com.key*
   - *diego.especssl.com.csr*
   - *certificado.diego.especssl.com.crt*
   - *intermedio.especssl.com.crt*

9. Activar el módulo SSL ejecutando el comando `a2enmod ssl` según se muestra en la figura 18.

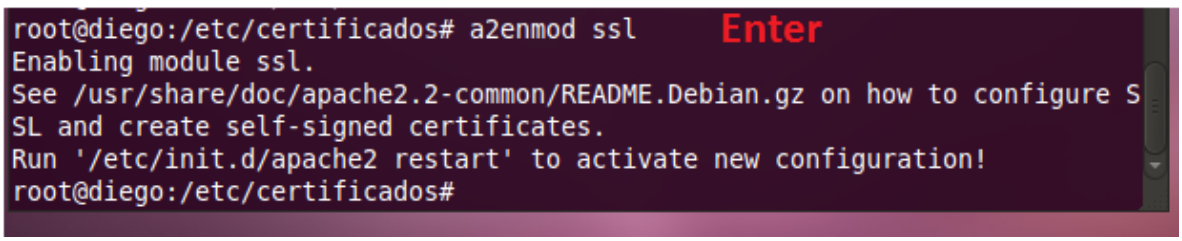


Figura 18. Activación del módulo SSL

10. Habilitar el servidor para que escuche peticiones por el puerto 443 utilizando el comando `a2ensite default-ssl` (ver figura 19).

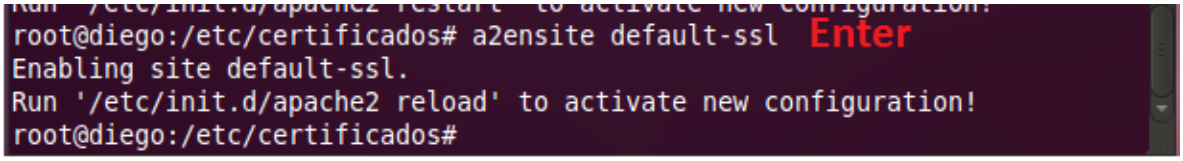


Figura 19. Activación del puerto 443 para peticiones

11. Reiniciar el servidor Web Apache ejecutando el comando `/etc/init.d/apache2 restart` con el fin de hacer efectivos los cambios introducidos en los pasos 9 y 10 (ver figura 20) y luego ejecutar el comando `/etc/init.d/apache2 reload` según se muestra el la figura 21.

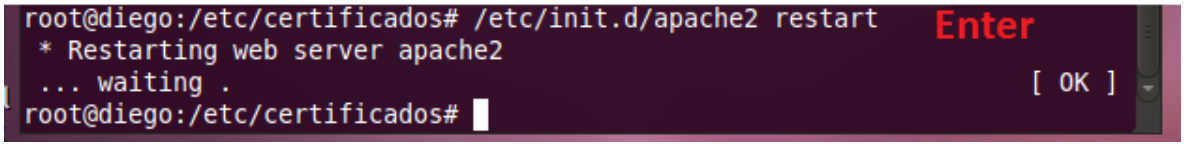


Figura 20. Reiniciación del servidor Web para hacer efectivo los cambios

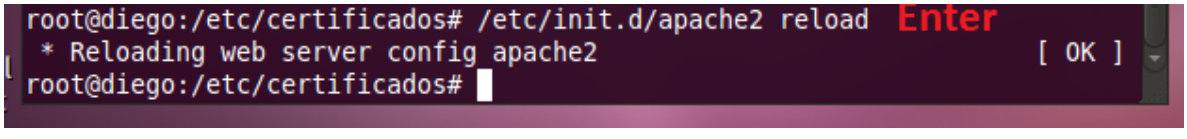


Figura 21. Reconfiguración del servidor Web

12. Abrir el archivo de configuración utilizando el comando `gedit /etc/apace2/sites-avaavailable/default-ssl` según se muestra en la figura 22.

13. Editar las líneas 51, 52 y 61 según se muestra en la figura 23, con el fin de asignar la ruta de los certificados. Guardar los cambios realizados en el archivo y cerrarlo.

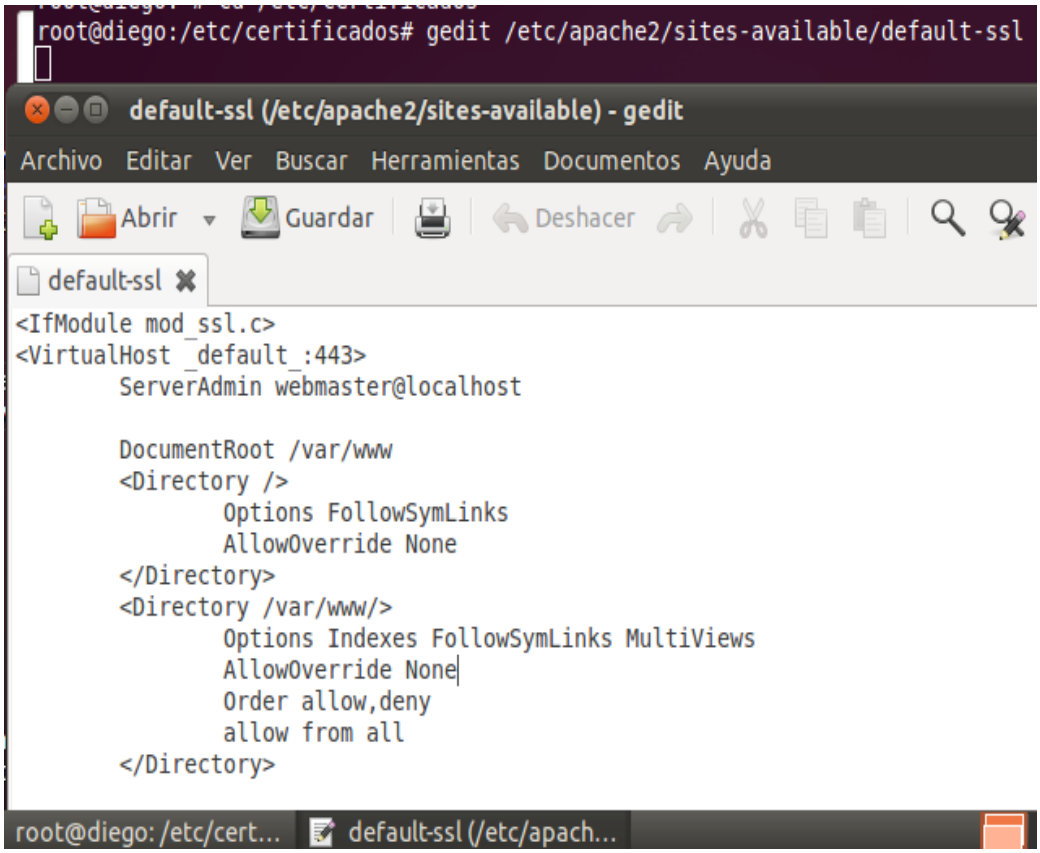


Figura 22. Archivo de configuración del modulo SSL

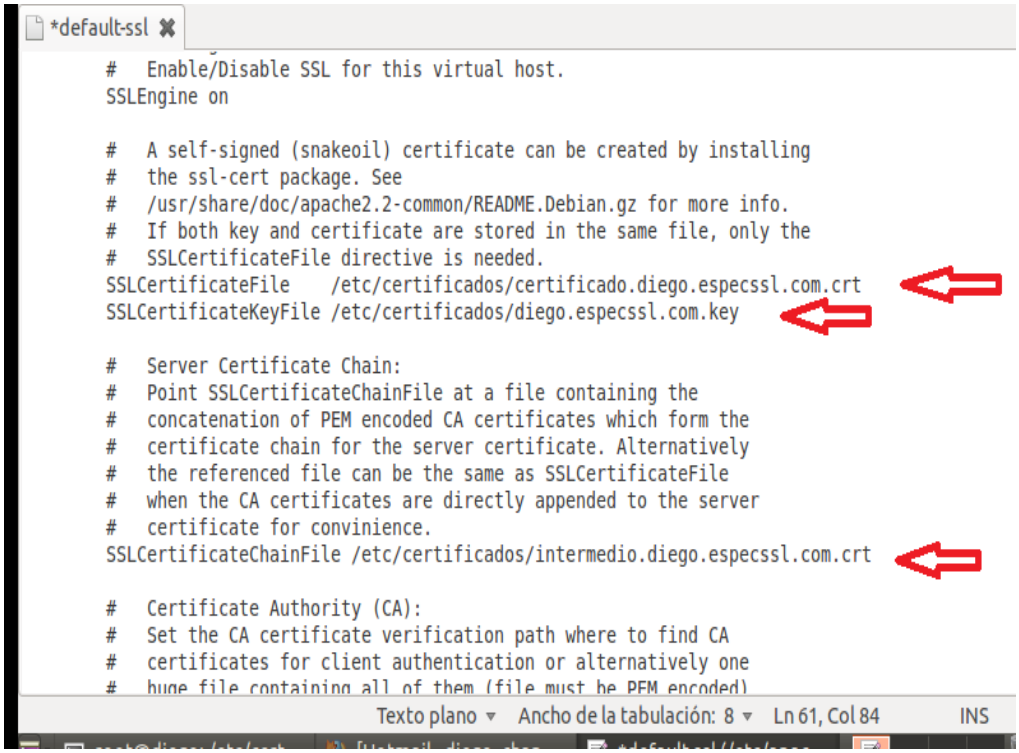


Figura 23. Edición del archivo de configuración del modulo SSL

14. Reiniciar el servidor Web utilizando el comando `/etc/init.d/apache2 restart`, con el fin de aplicar los cambios mencionados en 13. El servidor pedirá a continuación el ingreso de la contraseña de la clave privada (asignada en el paso 4). Ésta es requerida por el servidor Web cuando se inicia. Ver proceso en la figura 24.

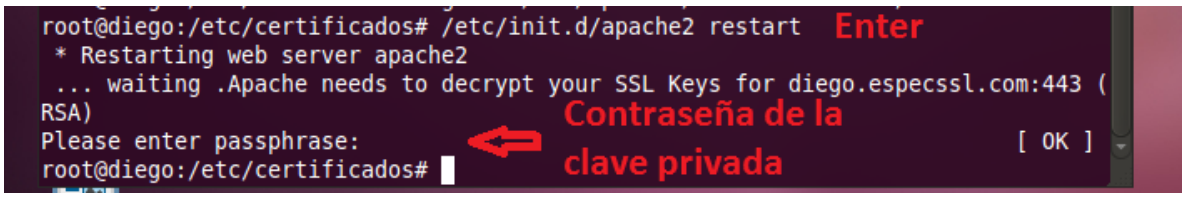


Figura 24. Reinicio del servidor Web

15. Abrir un navegador de Internet y acceder al sitio *https://diego.especssl.com* para comprobar el buen funcionamiento del módulo SSL.

4.2 Validación de la guía de configuración

La validación de la guía fué realizada en un computador con Windows 7, el cual tiene instalado la aplicación *VMware* que permite crear máquinas virtuales y simular una red de computadores. En este caso, la red consta de un usuario Windows 7 y un servidor Web como se muestra en la figura 25. El servidor Web es simulado por una máquina virtual con sistema operativo Ubuntu 11.04 (ver figura 26).

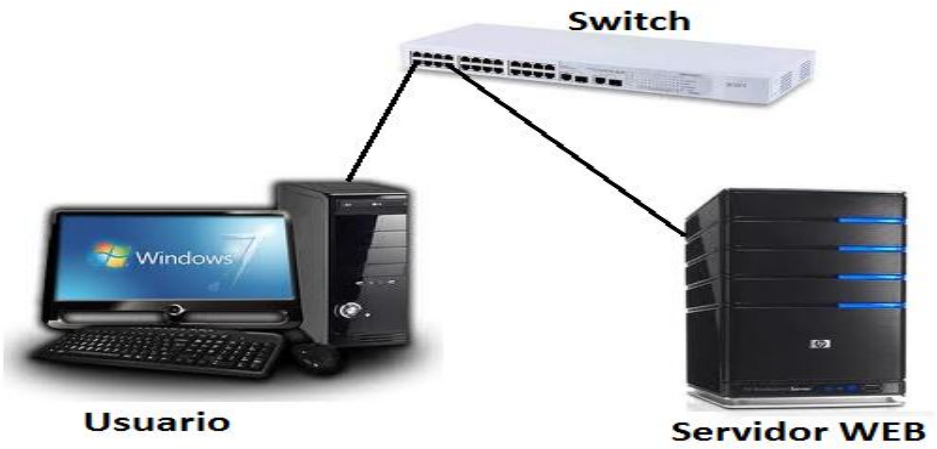


Figura 25. Arquitectura de prueba soportada por SSL

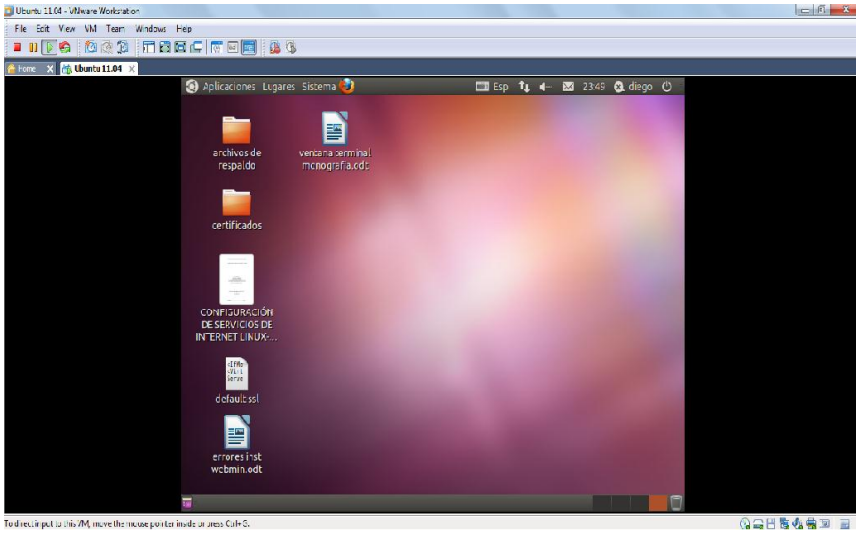


Figura 26. Sistema operativo Ubuntu instalado en *VMware*

Para observar el funcionamiento del protocolo SSL/TLS se instala en el computador del usuario el programa *Wireshark-win64-1.6,* que permite ver tramas de información que pasan por la red.

En este caso, el usuario tiene asignada la dirección IP 192.168.0.29 y el servidor la dirección IP 192.168.0.30 con nombre de dominio *diego.especssl.com.*

Desde el computador del usuario, se abre el explorador *Google Chrome* y se accede a la página https://diego.especssl.com (figura 27). A continuación aparece un mensaje advirtiendo sobre la seguridad del certificado del sitio. Escoger la opción "*Continuar de todos modos"* para permitir mostrar la página alojada en el servidor Web (ver figura 28).

Figura 27. Mensaje de confirmación del certificado de seguridad

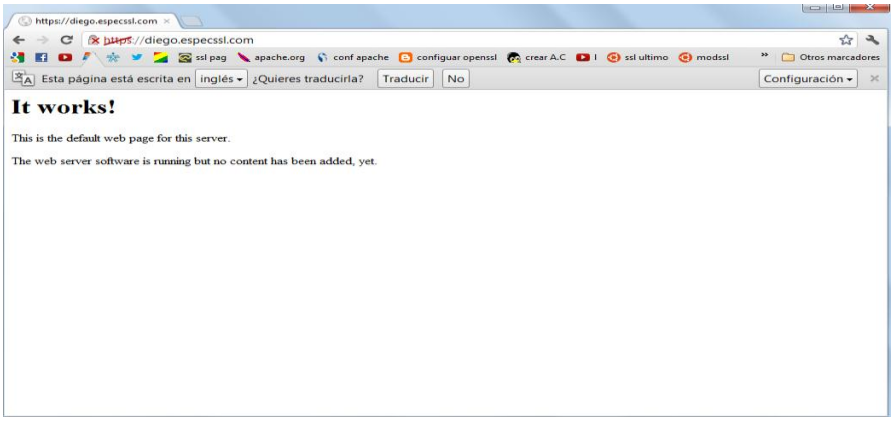


Figura 28. Página inicial cuando el usuario visita la página con SSL

En la figura 29, se muestran las características del certificado instalado. Además se observa que el certificado raíz (*Thawte Test CA Root certificate*) no aparece instalado en el explorador utilizado, por lo que se hace necesario llevar a cabo este proceso (ver [20]). La validez del certificado se puede observar en la figura 30.

Figura 29. Revisión de datos del certificado

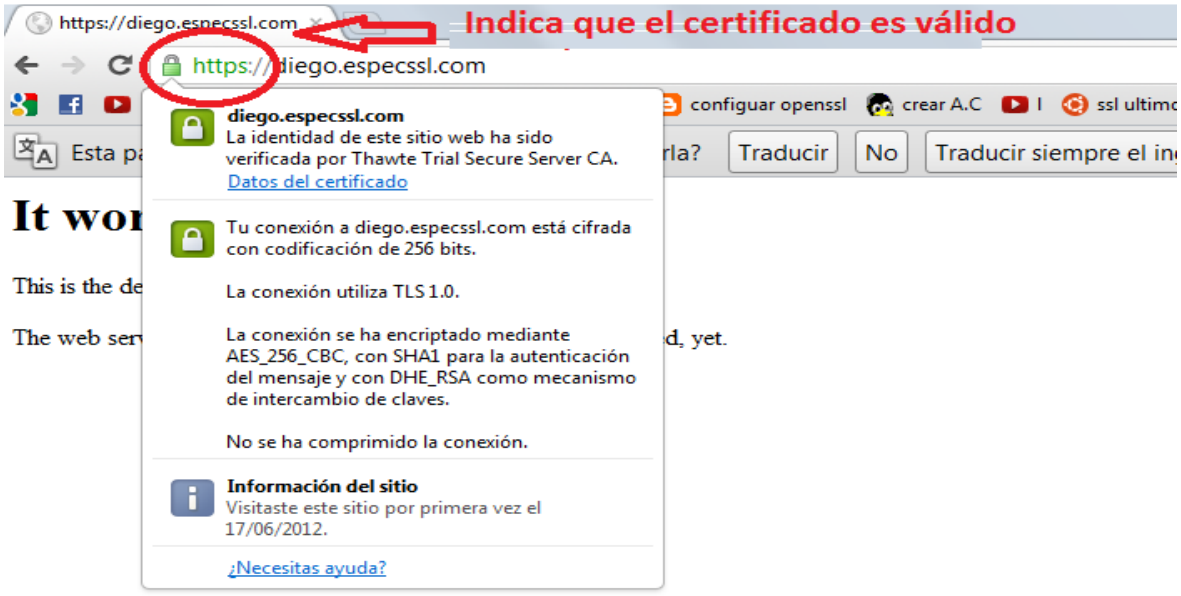
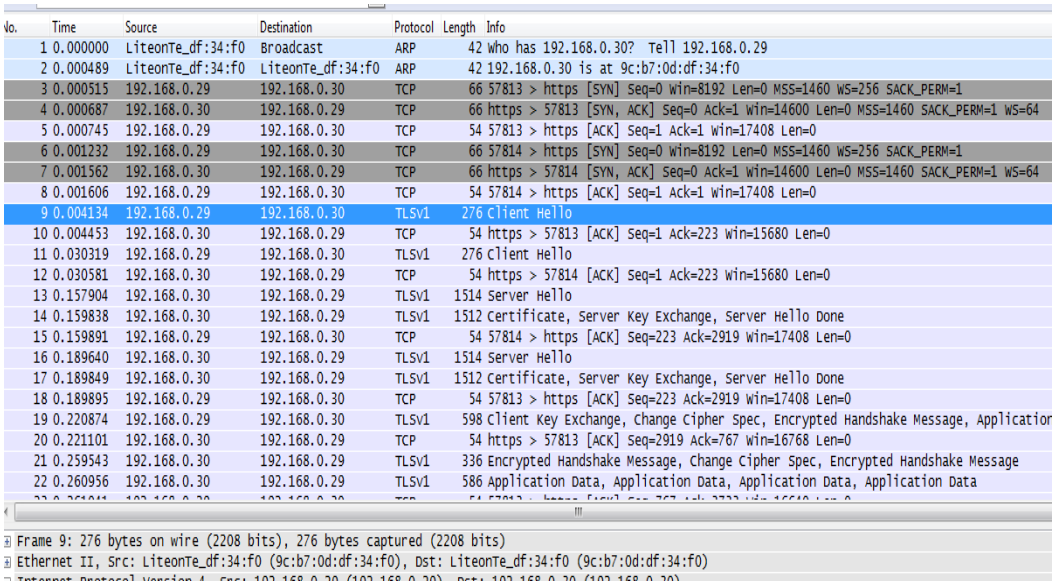


Figura 30. Verificación de características del certificado funcionando correctamente

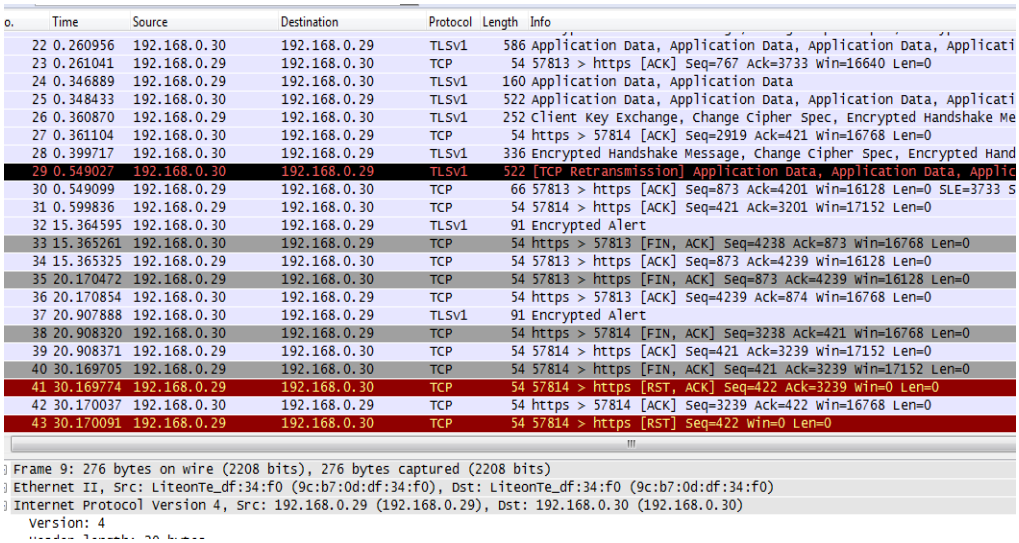
La aplicación *Wireshark* permite la captura del tráfico de la red, en este caso cuando el usuario realiza la solicitud de la página Web al servidor. Este proceso se muestra en las figuras 31 y 32.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	LiteonTe_df:34:f0	Broadcast	ARP	42	Who has 192.168.0.30? Tell 192.168.0.29
2	0.000489	LiteonTe_df:34:f0	LiteonTe_df:34:f0	ARP	42	192.168.0.30 is at 9c:b7:0d:df:34:f0
3	0.000515	192.168.0.29	192.168.0.30	TCP	66	57813 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	0.000687	192.168.0.30	192.168.0.29	TCP	66	https > 57813 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=64
5	0.000745	192.168.0.29	192.168.0.30	TCP	54	57813 > https [ACK] Seq=1 Ack=1 Win=17408 Len=0
6	0.001232	192.168.0.29	192.168.0.30	TCP	66	57814 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	0.001562	192.168.0.30	192.168.0.29	TCP	66	https > 57814 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=64
8	0.001606	192.168.0.29	192.168.0.30	TCP	54	57814 > https [ACK] Seq=1 Ack=1 Win=17408 Len=0
9	0.004134	192.168.0.29	192.168.0.30	TLSv1	276	Client Hello
10	0.004453	192.168.0.30	192.168.0.29	TCP	54	https > 57813 [ACK] Seq=1 Ack=223 Win=15680 Len=0
11	0.030319	192.168.0.29	192.168.0.30	TLSv1	276	Client Hello
12	0.030581	192.168.0.30	192.168.0.29	TCP	54	https > 57814 [ACK] Seq=1 Ack=223 Win=15680 Len=0
13	0.157904	192.168.0.30	192.168.0.29	TLSv1	1514	Server Hello
14	0.159838	192.168.0.30	192.168.0.29	TLSv1	1512	Certificate, Server Key Exchange, Server Hello Done
15	0.159891	192.168.0.29	192.168.0.30	TCP	54	57814 > https [ACK] Seq=223 Ack=2919 Win=17408 Len=0
16	0.189640	192.168.0.30	192.168.0.29	TLSv1	1514	Server Hello
17	0.189849	192.168.0.30	192.168.0.29	TLSv1	1512	Certificate, Server Key Exchange, Server Hello Done
18	0.189895	192.168.0.29	192.168.0.30	TCP	54	57813 > https [ACK] Seq=223 Ack=2919 Win=17408 Len=0
19	0.220874	192.168.0.29	192.168.0.30	TLSv1	598	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message, Applicatior
20	0.221101	192.168.0.30	192.168.0.29	TCP	54	https > 57813 [ACK] Seq=2919 Ack=767 Win=16768 Len=0
21	0.259543	192.168.0.30	192.168.0.29	TLSv1	336	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
22	0.260956	192.168.0.30	192.168.0.29	TLSv1	586	Application Data, Application Data, Application Data, Application Data

Frame 9: 276 bytes on wire (2208 bits), 276 bytes captured (2208 bits)
Ethernet II, Src: LiteonTe_df:34:f0 (9c:b7:0d:df:34:f0), Dst: LiteonTe_df:34:f0 (9c:b7:0d:df:34:f0)

Figura 31. Captura de tramas entre el servidor Web y el usuario con *Wireshark*



No.	Time	Source	Destination	Protocol	Length	Info
22	0.260956	192.168.0.30	192.168.0.29	TLSv1	586	Application Data, Application Data, Application Data, Applicati
23	0.261041	192.168.0.29	192.168.0.30	TCP	54	57813 > https [ACK] Seq=767 Ack=3733 Win=16640 Len=0
24	0.346889	192.168.0.29	192.168.0.30	TLSv1	160	Application Data, Application Data
25	0.348433	192.168.0.30	192.168.0.29	TLSv1	522	Application Data, Application Data, Application Data, Applicati
26	0.360870	192.168.0.29	192.168.0.30	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me
27	0.361104	192.168.0.30	192.168.0.29	TCP	54	https > 57814 [ACK] Seq=2919 Ack=421 Win=16768 Len=0
28	0.399717	192.168.0.30	192.168.0.29	TLSv1	336	Encrypted Handshake Message, Change Cipher Spec, Encrypted Hand
29	0.549027	192.168.0.30	192.168.0.29	TLSv1	522	[TCP Retransmission] Application Data, Application Data, Applic
30	0.549099	192.168.0.29	192.168.0.30	TCP	66	57813 > https [ACK] Seq=873 Ack=4201 Win=16128 Len=0 SLE=3733 S
31	0.599836	192.168.0.29	192.168.0.30	TCP	54	57814 > https [ACK] Seq=421 Ack=3201 Win=17152 Len=0
32	15.364595	192.168.0.30	192.168.0.29	TLSv1	91	Encrypted Alert
33	15.365261	192.168.0.30	192.168.0.29	TCP	54	https > 57813 [FIN, ACK] Seq=4238 Ack=873 Win=16768 Len=0
34	15.365325	192.168.0.29	192.168.0.30	TCP	54	57813 > https [ACK] Seq=873 Ack=4239 Win=16128 Len=0
35	20.170472	192.168.0.29	192.168.0.30	TCP	54	57813 > https [FIN, ACK] Seq=873 Ack=4239 Win=16128 Len=0
36	20.170854	192.168.0.30	192.168.0.29	TCP	54	https > 57813 [ACK] Seq=4239 Ack=874 Win=16768 Len=0
37	20.907888	192.168.0.30	192.168.0.29	TLSv1	91	Encrypted Alert
38	20.908320	192.168.0.30	192.168.0.29	TCP	54	https > 57814 [FIN, ACK] Seq=3238 Ack=421 Win=16768 Len=0
39	20.908371	192.168.0.29	192.168.0.30	TCP	54	57814 > https [ACK] Seq=421 Ack=3239 Win=17152 Len=0
40	30.169705	192.168.0.29	192.168.0.30	TCP	54	57814 > https [FIN, ACK] Seq=421 Ack=3239 Win=17152 Len=0
41	30.169774	192.168.0.29	192.168.0.30	TCP	54	57814 > https [RST, ACK] Seq=422 Ack=3239 Win=0 Len=0
42	30.170037	192.168.0.30	192.168.0.29	TCP	54	https > 57814 [ACK] Seq=3239 Ack=422 Win=16768 Len=0
43	30.170091	192.168.0.29	192.168.0.30	TCP	54	57814 > https [RST] Seq=422 Win=0 Len=0

Frame 9: 276 bytes on wire (2208 bits), 276 bytes captured (2208 bits)
Ethernet II, Src: LiteonTe_df:34:f0 (9c:b7:0d:df:34:f0), Dst: LiteonTe_df:34:f0 (9c:b7:0d:df:34:f0)
Internet Protocol Version 4, Src: 192.168.0.29 (192.168.0.29), Dst: 192.168.0.30 (192.168.0.30)
Version: 4

Figura 32. Captura de tramas entre el servidor Web y el usuario con *Wireshark*

Análisis de resultados

Para verificar que la configuración es correcta, se analizan las tramas de datos de la comunicación entre el usuario y el servidor. Por ejemplo, al acceder a la página Web alojada en el servidor, según se muestra en la figura 31, lo primero que ocurre es una solicitud ARP (*Address Resolution Protocol)* identificada en las tramas 1 y 2. Esta solicitud permite encontrar la dirección física de la IP 192.168.0.30 (servidor). A continuación comienza el proceso de negociación del protocolo TCP identificado en las tramas 3 a 8. En las tramas 9 a 13 se puede observar la manera como se establece la negociación de cifrado entre el servidor y el usuario, la pregunta de autenticación y la versión SSL del usuario. En las tramas 14 y 15 se autentica el usuario y se intercambian claves. En las tramas 19 y 20 se expresa el intercambio de claves del servidor, el cambio de especificaciones de cifrado y "apretón de manos". Las tramas 22 a 31 transportan los datos de la aplicación. Las tramas 32 a 40 corresponden a mensajes de alerta SSL, cifrado SSL y fin de la conexión TCP.

Las tramas más importantes son: el "*cliente hello"* (9), el "*server hello"* (13), el "*certificate and server key Exchange" y* el *"server hello done"* (14), el "*Cliente key Exchange"* como corresponde en la trama 19, los nombres "*change cipher"* y *"encrypted handshake message"* que se representan en la trama 21. Las figuras 33 a 36 muestran detalles del contenido de algunas de estas tramas.

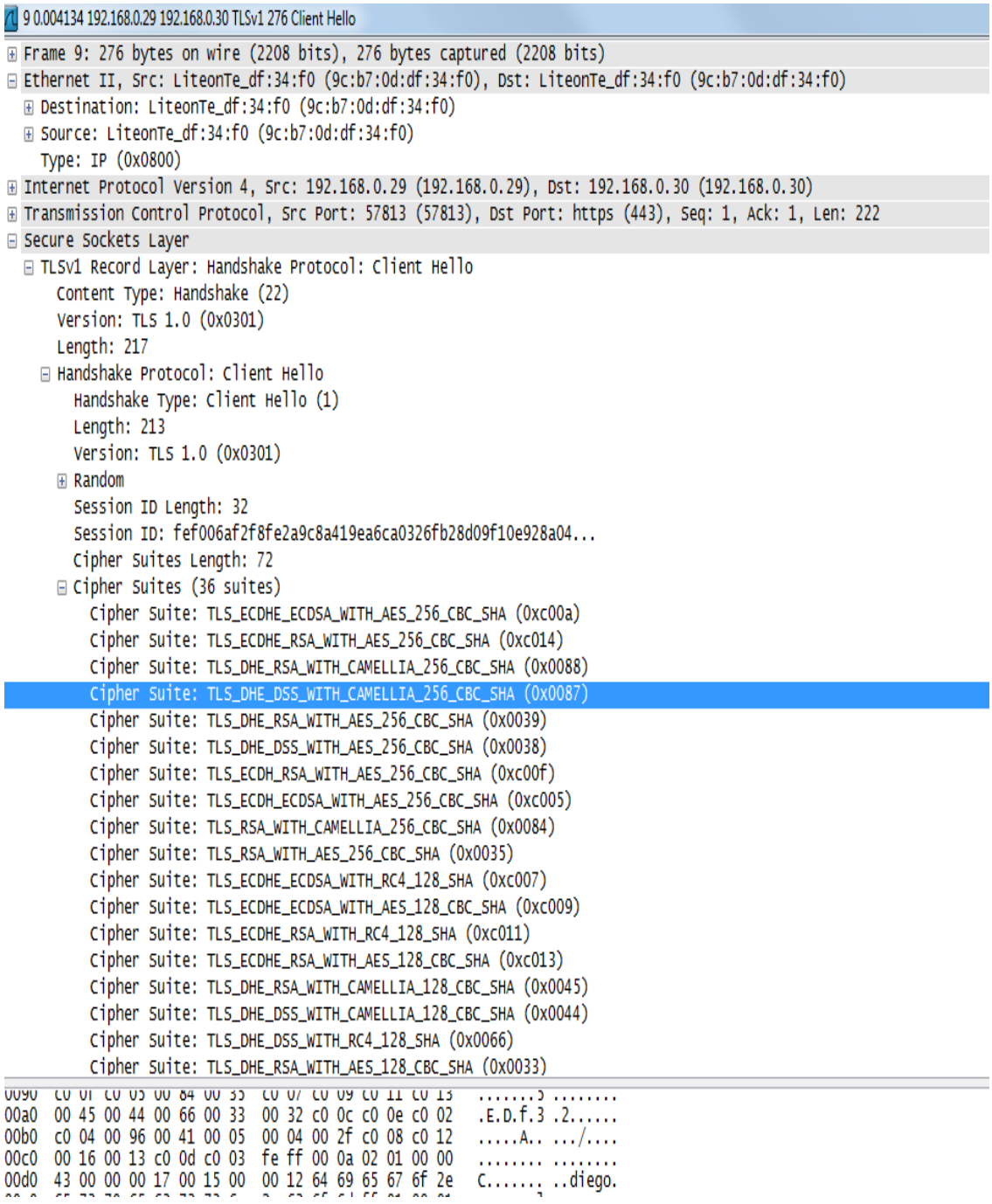


Figura 33. Captura de datos del "*cliente hello"*

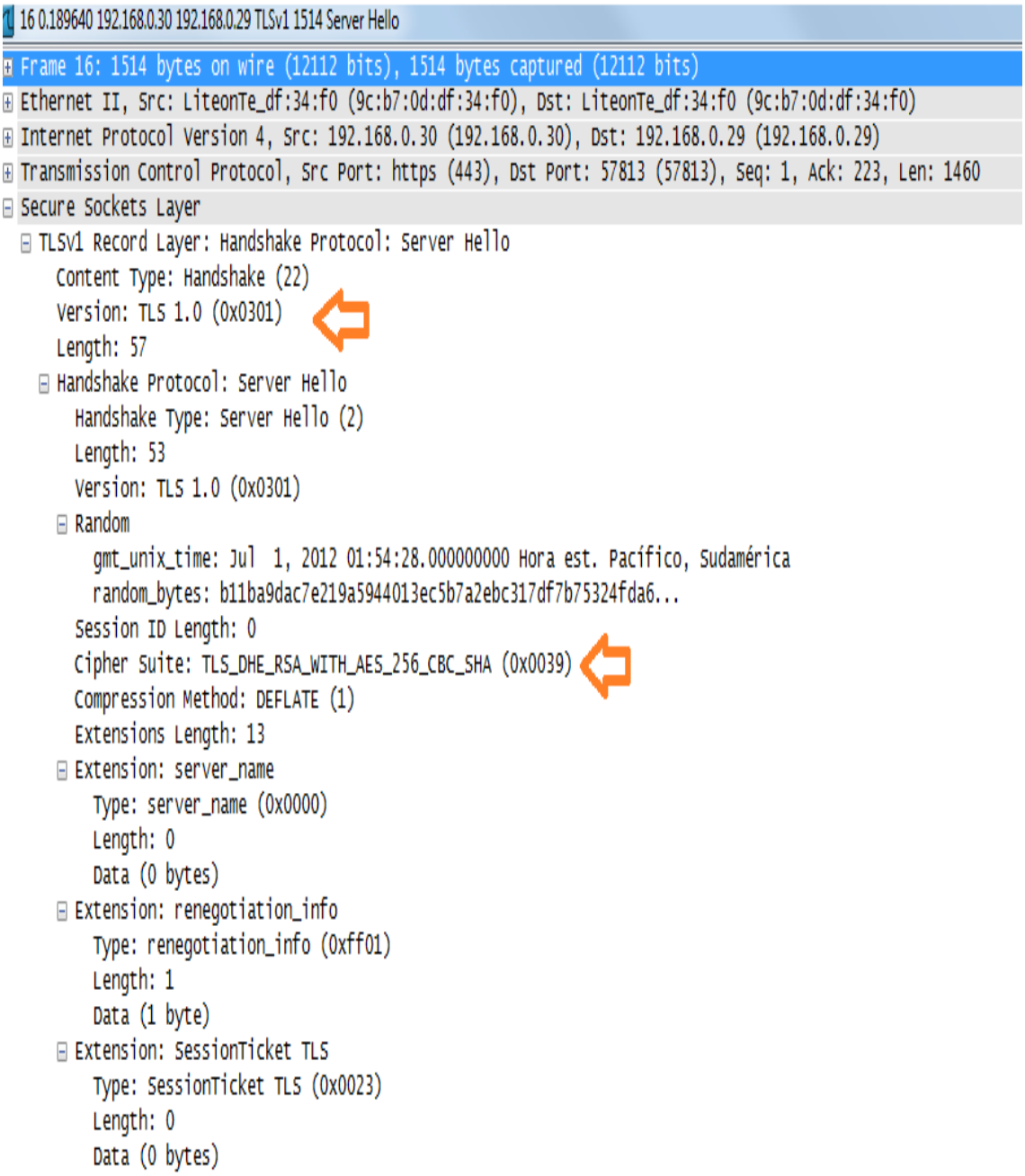


Figura 34. Captura de datos del "*server hello"*

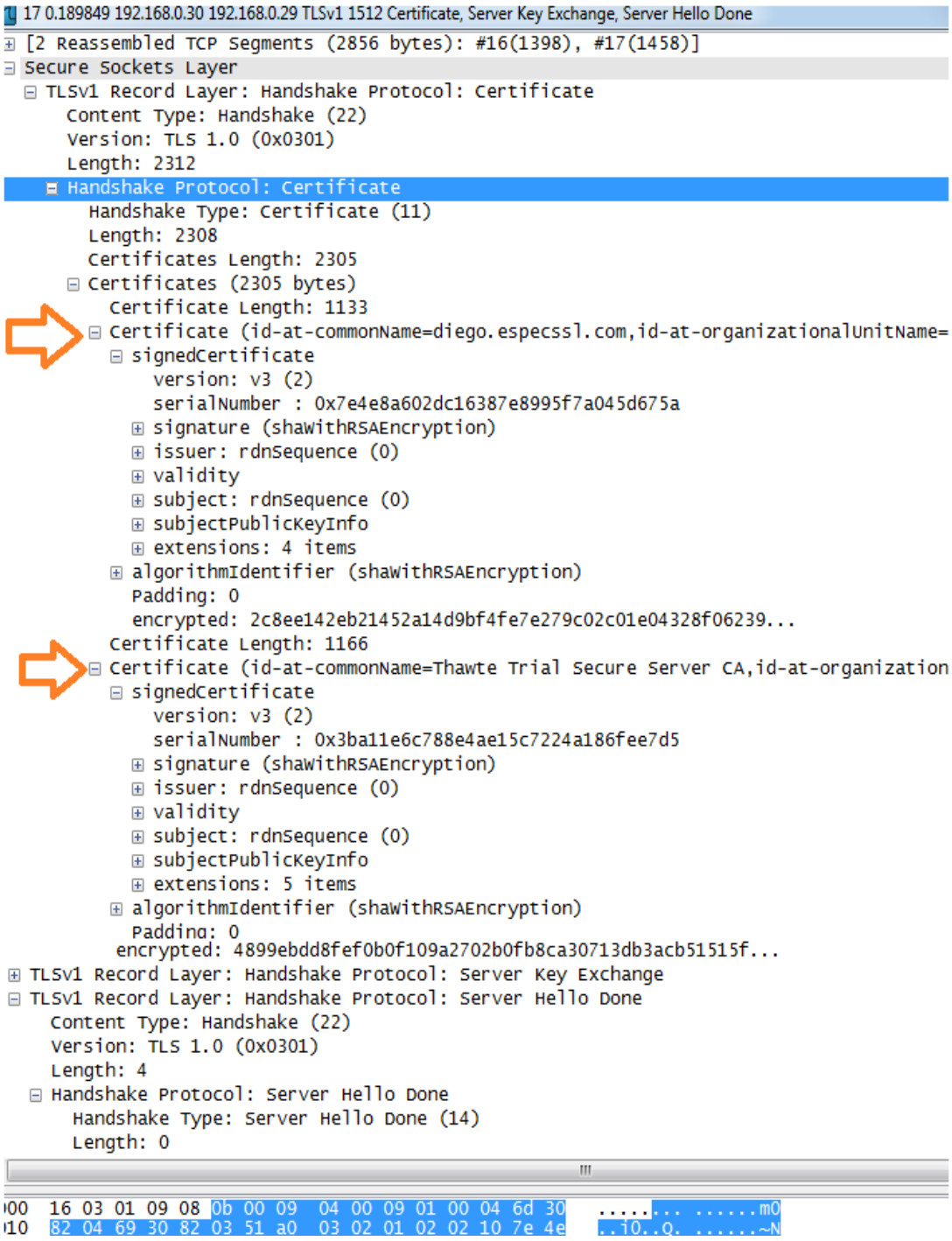


Figura 35. Captura de datos del "*certificate* and *server key Exchange*"

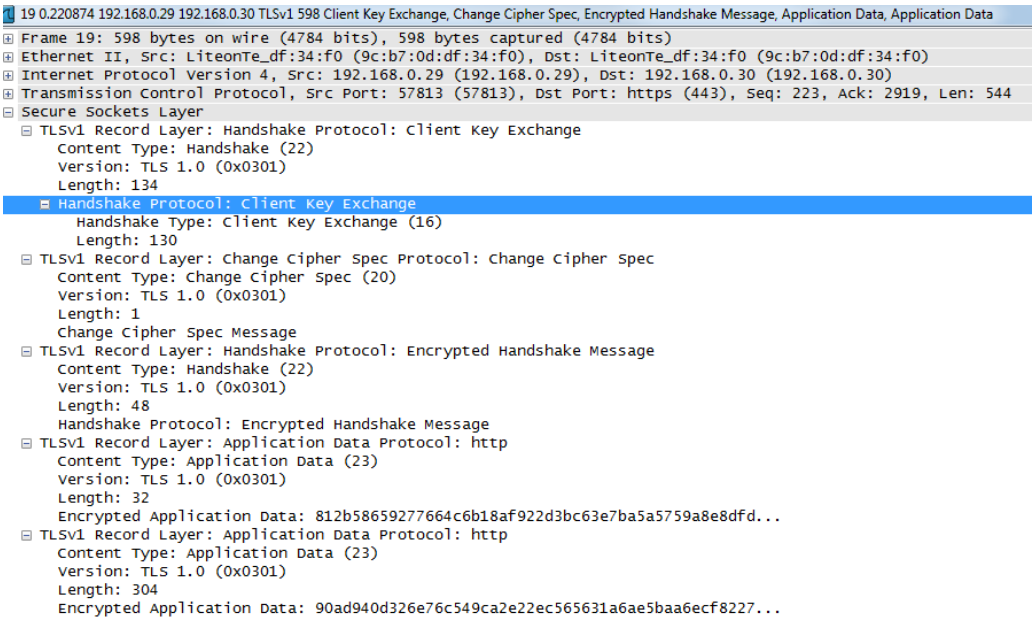


Figura 36. Captura de datos del "*cliente key exchage encrypted handshake"*

El análisis de tramas introducido anteriormente permite verificar el funcionamiento completo del protocolo SSL y sus certificados digitales.

Nota: Las tramas pueden cambiar dependiendo de la configuración dada al servidor y el tipo de certificados instalados.

Captura de datos de usuarios

Para demostrar la transmisión encriptada de la información por parte del protocolo, se configuró una página Web de prueba que captura el nombre de un usuario y su contraseña, según se muestra en la figura 37.

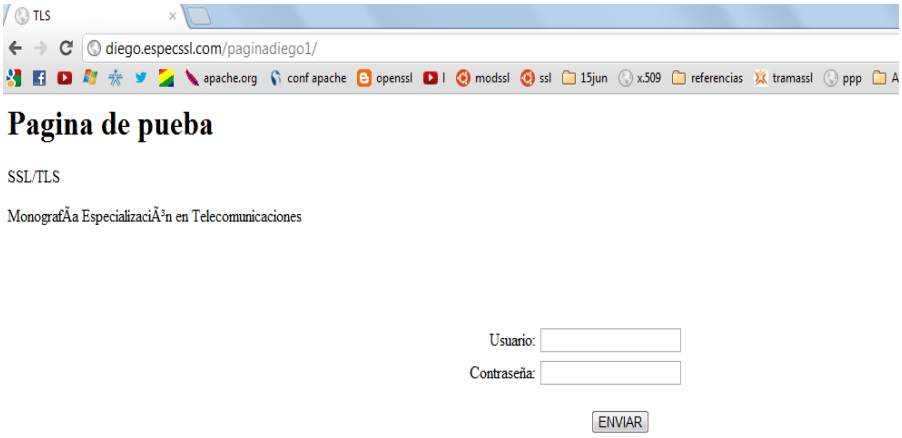


Figura 37. Página Web de prueba que solicita datos privados del usuario

El ejercicio se llevó a cabo en los tiempos siguientes, primero se analizaron las tramas a través de una conexión sin SSL/TLS. Se logró comprobar que el nombre de usuario y contraseña se pudieron capturar y conocer su contenido (ver figura 38). Después, se accedió a la página Web a través de una conexión segura, y se pudo observar el encriptamiento de la información (ver figura 39). La información suministrada para usuario y contraseña fue:

- Usuario: *especialización*
- Contraseña: *teleco*

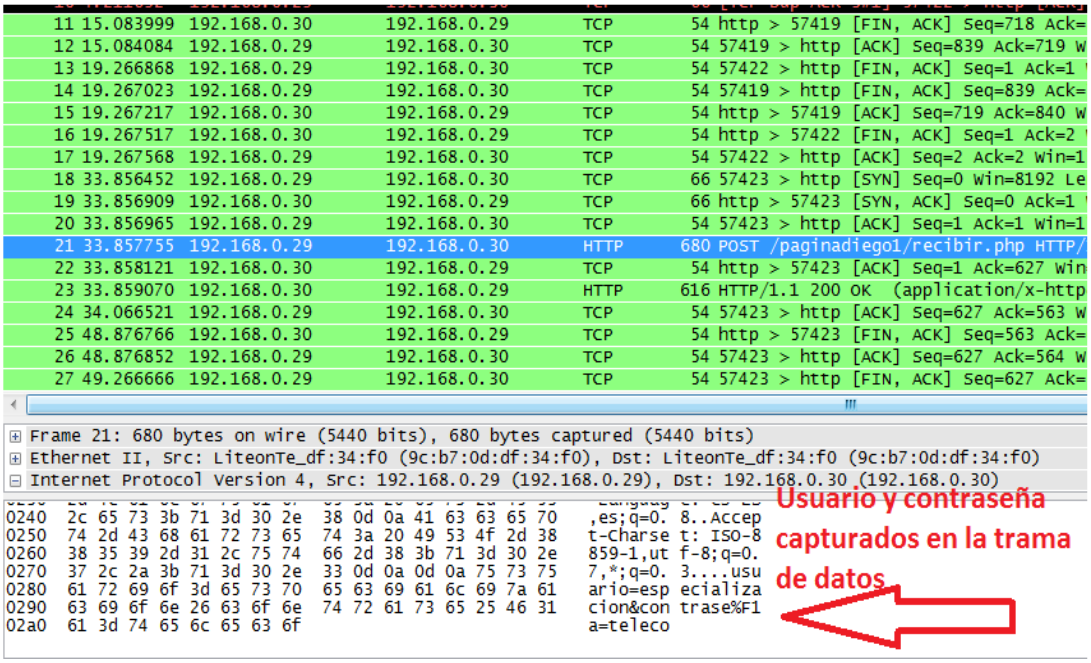


Figura 38. Captura de nombre de usuario y contraseña sin SSL/TLS

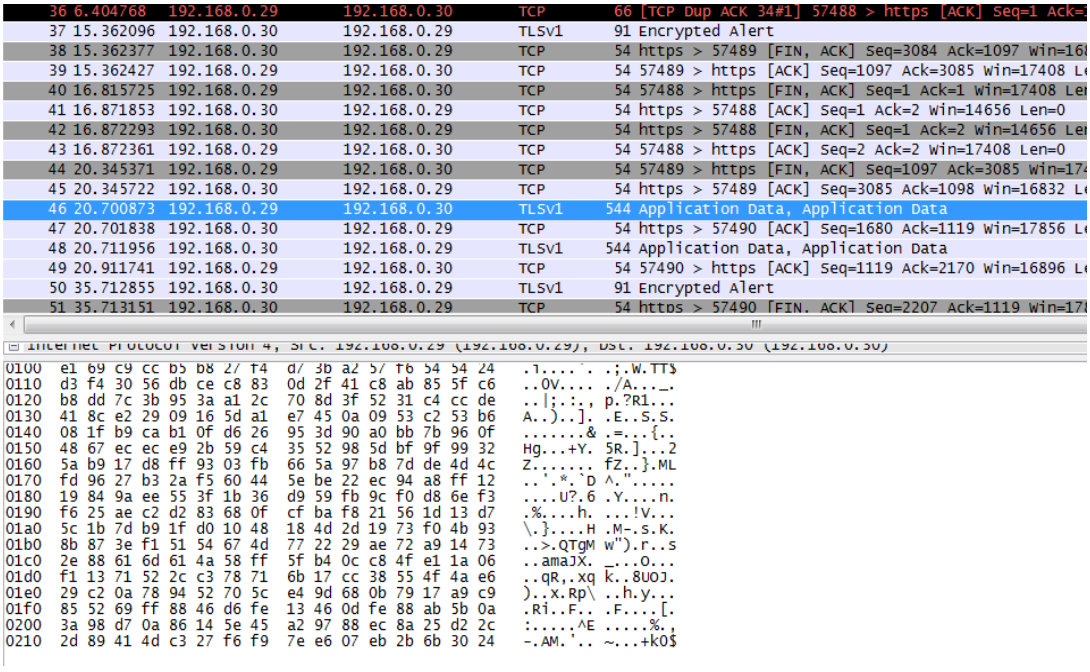


Figura 39. Información encriptada

5. CONCLUSIONES

El protocolo SSL/TLS ofrece mediante sus certificados de clave pública y clave privada y sus sistemas de encriptamiento, alta seguridad, credibilidad y confianza a los usuarios que utilizan procesos de transmisión y recepción de datos a través de servidores Web.

Con el ejercicio realizado para la captura de tráfico se pudo observar claramente que la utilización del protocolo, requiere de un mayor número de tramas, tiempo de proceso y cifrado y descifrado de datos, donde este mayor tiempo resulta en la disminución de la velocidad de comunicación.

SSL/TLS puede ser implementado fácilmente en entornos empresariales gracias a programas como el OpenSSL, que permiten la creación de entidades certificadoras para el aseguramiento y autenticación tanto del cliente como del servidor en el acceso a la información.

Existen programas al alcance de todos como el *WireShark* que capturan tramas de datos, en las que fácilmente se pueden dar a conocer detalles de la información enviada por los clientes. Por esto se hace indispensable, la utilización de un protocolo de seguridad robusto en todas las redes y así proteger la información.

Con el desarrollo de este trabajo se logró identificar la importancia del protocolo SSL/TLS, ya que su implementación permite hoy en día, que muchas actividades realizadas por medio de la red, como transacciones bancarias, se lleven a cabo con éxito, garantizando seguridad y confianza al usuario.

La guía de instalación y configuración de este protocolo, diseñada por medio de este trabajo permite orientar a los usuarios administradores de servidores Web para la puesta en marcha de este importante recurso de seguridad para garantizar la integridad de los datos y autenticar los usuarios de información privilegiada, que viaja a través de las redes.

Utilizar criptografía permite proteger la información. Combinar distintos algoritmos criptográficos y una longitud de clave adecuada, en una comunicación, permite la implementación de sistemas de intercambio de información robustos que garanticen que la información solo este disponible para las personas autorizadas.

6. REFERENCIAS

[1] J.M. Morales, “SSL, Secure Sockets Layer y Otros Protocolos Seguros para el Comercio Electrónico,” Vol. 1,2002.

[2] S. Horman. “SSL and TLS An Overview of A Secure Communications protocol,” Presentado en Security Mini-conf at Linux.Conf.AU Canberra, ACT, Australia, 2005, Abril, pp. 2-5, 10-17.

[3] D.O. Ramírez, C.C. Espinosa,” El Cifrado Web (SSL/TLS). Seguridad Cultura de prevención para TI,” [En línea], (10), Disponible en: http://revista.seguridad.unam.mx/numero-10/el-cifrado-web-ssltls [Visitada: Noviembre 21,2012].

[4] P. Chandra, M. Messier y J. Viega, “Network Security wiht OpenSSL,” O’Reilly, 2002, USA, pp 2-6, 27-42.

[5] T. Onyszko, “Secure Sockets Layer,” WindowSecurity.com, 2002, Julio, [En línea], Disponible en: http://www.windowsecurity.com/articles/SecureSocketLayer.html [Visitada: 22-Nov-2012].

[6] J. Wang**,** “Características principales del protocolo SSL,” 2009, Nov, [En línea], Disponible en: http://www.docstoc.com/docs/26388261/Características-principales-del-protocolo-SSL [Visitada: 22-Nov-2012].

[7] J. R. Vacca, “Understanding Digital Certificates And Secure Sockets Layer (SSL),” en Public Key Infrastructure: Building Trusted Applications and Web Services, 2004, pp. 1, 10-18.

[8] M. H. SHERIF, “SSL (Secure Sockets Layer),” En Protocols for Secure Electronic Commerce, (2da. Ed.), Saba Zamir, 2003, pp. 228-237,247-249.

[9] “Home | Ubuntu,” [En línea], Disponible en: http://www.ubuntu.com/ [Visitada: 22-Nov-2012].

[10] Certicámara S.A, “Autoridad de certificación digital abierta.” [En línea]. Disponible en: http://web.certicamara.com/certificados-digital-seguridad.aspx [Visitada: 22-Nov-2012].

[11] NeoTheK, **“**Certificados de seguridad SSL,” [En Línea], Disponible en: http://www.neothek.com/certificados-ssl/certificado-ssl-gratis [Visitada: 22-Nov-2012].

[12] M.A. Riffo, “Vulnerabilidades de las Redes TCP/IP y Principales Mecanismos de Seguridad”, Director: Nestor Fierro. Tesis de pregrado. Universidad Austral de Chile, Escuela de ingeniería electrónica, 2009.

[13] GlobalSing, “Tipos de certificados SSL,” [En línea]. Disponible en: http://www.globalsign.es/centro-informacion-ssl/tipos-de-certificado-ssl.html [Visitada: 22-Nov-2012].

[14] Thawte, “Buy certificates,” [En línea], Disponible en: http://www.thawte.com/ [Visitada: 22-Nov-2012]

[15] Canalayuda, “Clasificación y Tipos de Ataques Contra Sistemas de Información,” [En línea], Disponible en: http://www.canalayuda.net/ataques_contra_sistemas.htm [Visitada: 22-Nov-2012].

[16] J.I. Argeta, “Criptografía, Red Feistel,” 2010, abril, [En línea], Disponible en: www.openboxer.260mb.com/asignaturas/criptografia/red**Feistel**.pdf [Visitada: Julio 27,2012].

[17] A.S. Tanenbaum, “Redes de Computadoras,” Pearson educación, 4 edición, México, 2003, pp 912.

[18] “Cifrado RSA,” [En línea] Disponible en: julio 15-2012. http://www.dma.fi.upm.es/gsanchez/cifradorsa.PDF [Visitada: Julio 15,2012].

[19] “Preguntas frecuentes sobre malware del sello VeriSign Trust de VeriSign Sarl,” [En línea], Disponible en: http://www.verisign.es/trust-seal/resources/malware-faq/index.html. [Visitada: 22-Nov-2012].

[20] Thawte, “SSL Certificates Support.” [En línea]. Disponible en: https://search.thawte.com/support/ssl-digital-certificates/index?page=content&id=SO13732&actp=search&viewlocale=en_US&searchid=1354202046533 [Visitada: 29-Nov-2012].