UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CANAL CIFRADO PARA COMUNICACIÓN CLIENTE SERVIDOR

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

GARCÍA RUBÍ DANIELA IVETTE

RÍOS CLEMENTE EDGAR

DIRECTOR DE TESIS:

M. I. JUAN CARLOS ROA BEIZA

CIUDAD UNIVERSITARIA, MÉXICO D.F. ABRIL 2011.

A nuestras familias, profesores y amigos que nos han acompañado en el camino de nuestra formación como ingenieros, y muy especialmente a nuestra alma mater.

MÉXICO… PUMAS… UNIVERSIDAD!

A Dios, por caminar junto a mí en cada paso.

Mamá y Papá, por darme el regalo más grande: la vida, los amo.

Pam y Pepe, por ser mis cómplices de vida, los amo hermanitos.

A mi abue Bertha, por tu amor y apoyo incondicional, eres mi ejemplo de fortaleza.

A mi madrina Alicia, por ser un ángel que me cuida incondicionalmente.

A mi madrina Liz, gracias infinitas por tu amor incansable, eres una luz que guía mi camino.

A mi familia, por la confianza y cariño, por enseñarme el valor de los lazos más fuertes.

A mis amigos, por compartir tantas experiencias y convertirse en parte de mi familia, los quiero!

A mi Universidad, por brindarme el mejor de los conocimientos.

Con todo mi corazón, Daniela.

A mis padres Manuel y Hortencia, por su enorme paciencia y apoyo, y sobre todo por brindarme la oportunidad de desarrollarme en lo que me agrada.

A mi hermano Eduardo por estar a mi lado en las buenas y en las malas, te quiero.

A todas mis tías Clemente Gómez y Ríos Morales, mil gracias por tanto amor y cariño y brindarme su apoyo desde siempre, las quiero y siempre las llevo en mente.

A mis amigas y amigos, ha sido un verdadero placer haber compartido todo este tiempo con ustedes. Gracias por ser parte de mi vida y estar siempre cuando se les necesita, cuenten conmigo.

A mis profesores que han compartido sus experiencias con nosotros.

A los instructores y quienes hicieron posibles los programas de becas que nos permiten tener más conocimientos y especializarnos, gracias.

Edgar.

TABLA DE CONTENIDO

CAPÍTULO 1

ENTORNO DEL PROBLEMA

CAPÍTULO 1

ENTORNO DEL PROBLEMA

1.1 INTRODUCCIÓN

Cualquier tipo de relación necesita que exista comunicación; como es bien sabido, la comunicación es una actividad milenaria y el hombre siempre ha buscado mejorar el intercambio de información a través de diversas redes como medios de trasporte terrestre, aéreo y marítimo, telégrafo, fax, correo postal, hasta llegar a las redes de computadoras y telecomunicaciones.

La historia nos ha mostrado la importancia que tiene la información en el desarrollo de la sociedad humana. Sin embargo, actualmente el tratamiento de la información con nuevas tecnologías permite conocer y manipular datos a través de sistemas que pueden estar conectados en una red local o mundial.

La seguridad de la información contiene como conceptos fundamentales la confidencialidad, integridad y disponibilidad, conocida como “tríada CIA” por sus siglas en inglés **Confidentiality** (**confidencialidad**), **Integrity** (**integridad**), **Availability** (**disponibilidad**), vea figura 1.1.1.

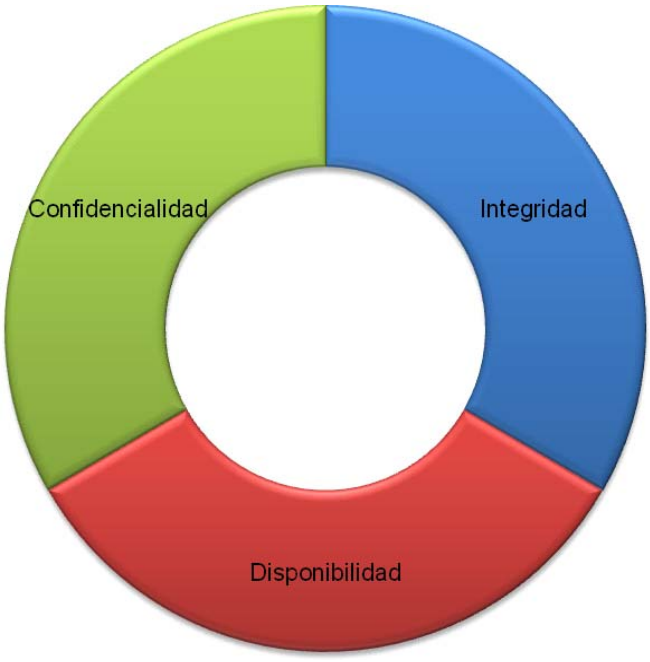


Figura 1.1.1 Tríada CIA.

La confidencialidad es la propiedad de prevenir la divulgación de información a personas o sistemas no autorizados, la integridad es la propiedad de mantener los datos libres de modificaciones no autorizadas, en tanto que la disponibilidad es la propiedad de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones.

Para que pueda existir una comunicación entre dos interlocutores, ambas partes deben tener la certeza de que los mensajes que envían son recibidos íntegramente por el destinatario, así como los mensajes que reciben fueron emitidos íntegramente por el emisor. En cualquier otro caso la comunicación no podrá llevarse a cabo por completo. La seguridad en las comunicaciones, en su forma más sencilla, garantiza que los mensajes dirigidos a ciertos destinatarios no se puedan leer y/o modificar por terceros no autorizados. La comunicación segura entre computadoras se basa en el mismo principio de la comunicación humana,

donde los elementos básicos son: un emisor, un receptor, un mensaje, un medio de transmisión, una interface que genere las señales adecuadas al medio y un lenguaje o protocolo seguro que el emisor y el receptor únicamente comprendan.

El principal objetivo de una red de datos es el intercambio de información y lo que le da valor a las redes de datos es el conjunto de servicios que comparten y que nos permiten realizar operaciones de manera remota como lo es el compartir aplicaciones, bases de datos, etc.

Contar con confidencialidad de la información es un requerimiento importante para proteger la información que se transmite en una red. El constante incremento en la fuga de información electrónica hace necesario crear medios de comunicación seguros entre dos equipos interconectados por una red, en el cual la información sea legible únicamente por los interlocutores involucrados y así garantizar la transferencia segura de información confidencial para evitar los ataques más comunes de intercepción de datos, ya sean internos o externos.

Se sabe que las redes públicas de comunicaciones son inseguras por naturaleza; y están expuestas a diversos riesgos como analizadores de tráfico, los cuales pueden exponer la información a transmitir.

El propósito de este trabajo es presentar una solución en la cual la implementación de los controles de seguridad tenga una relación balanceada de costo-beneficio para proteger uno de los activos más importantes para cualquier institución y/o empresa, la información.

La necesidad de contar con sistemas seguros obliga en la actualidad a los usuarios de todo tipo de servicios a desconfiar de las redes públicas que en su mayoría no están cifradas, por lo cual este proyecto de tesis propone desarrollar un método de seguridad para comunicación de datos que permita cifrar

información con algoritmos de cifrado que garanticen que la información no sea comprometida en caso de ser interceptada por un usuario no autorizado y de esta forma protegerse de amenazas internas y externas.

Recientemente el Gobierno Federal publicó a través del Diario Oficial de la Federación (DOF) el 5 de julio del 2010, la Ley Federal de Protección de Datos Personales en Posesión de los Particulares; dicha ley tiene la finalidad de regular su tratamiento legítimo, controlado e informado, a efecto de garantizar la privacidad y el derecho a la autodeterminación informativa de las personas. Por lo que este proyecto apoya el cumplimiento de esta ley al tratar la confidencialidad de la información mediante un canal seguro.

La criptografía es un método para almacenar y transmitir datos en una forma que únicamente los destinatarios pueden leer y procesar. Es considerada como la ciencia de proteger la información mediante codificación en un formato no entendible. Es una forma efectiva de proteger información sensitiva mediante el almacenamiento en medios o transmitida a través de rutas de comunicación no confiables. Uno de los objetivos de la criptografía y de sus mecanismos es esconder la información de individuos con acceso no autorizado.

1.2 ESTÁNDARES, NORMAS Y POLÍTICAS DE SEGURIDAD DE LA INFORMACIÓN QUE SE DEBEN CUMPLIR PARA LA COMUNICACIÓN ENTRE REDES

La normalización, que generalmente se traduce al inglés como **standarization** (**estandarización**), es aquella actividad que tiene por objeto establecer estándares de calidad y de medidas para bienes, productos y servicios así como para los procesos productivos, con la finalidad de mejorar las condiciones de seguridad y protección. Se expresa materialmente en estándares de calidad.

Un estándar de calidad contiene especificaciones técnicas para ser usado como normas, guías o definiciones de características que aseguren que los materiales, productos, procesos y servicios se ajustan a su propósito.

La normalización persigue fundamentalmente tres objetivos:

- Simplificación: Reducción de modelos quedándose únicamente con los más necesarios.

- Unificación: Permite el intercambio a nivel internacional.

- Especificación: Se evitan errores de identificación creando un lenguaje claro y preciso.

La necesidad de contar con estándares ha estado presente durante la historia del hombre, los primeros que fueron utilizados establecieron medidas de longitud y peso.

Los estándares, dependiendo del organismo que las elabora, pueden ser de uso obligatorio o de uso optativo; sin embargo, aunque la mayor parte de la normalización a nivel internacional es de carácter optativo, se ha vuelto indispensable adoptar sus criterios, ya que actualmente los compradores y consumidores exigen que los productos y servicios que adquieren cumplan con los requerimientos marcados por la normalización.

La normalización internacional es necesaria porque la existencia de estándares no alineados en tecnologías similares en diferentes países o regiones, retrasa el desarrollo y hace más difícil el intercambio de información, lo cual contribuye a formar barreras técnicas. Un ejemplo común es la incompatibilidad entre

programas para computadora. Normalizar es una manera de facilitar los procedimientos.

Usualmente la normalización es promovida por algunas de las industrias pertenecientes a un sector que se enfrentan con problemas a causa del número indeterminado de modelos que no se ajustan a sus necesidades. Es entonces cuando se convoca a los afectados y a especialistas en la materia para que propongan un estándar único que solucione dichos problemas.

Las especificaciones técnicas no pueden garantizar por sí mismas el éxito de un producto o servicio. Sin embargo, cuando un proceso es normalizado, además de facilitarse, se encuentra en posibilidad de disponer de la documentación que lo compruebe y con ella demostrar que se administra con calidad, y por lo tanto asegurar la calidad de los productos y servicios a los usuarios.

En materia de redes existen diversos organismos que regulan las tecnologías del intercambio de información entre las redes y los diferentes dispositivos de computadoras, algunas de ellas se mencionan en la tabla 1.

ANSI American National Standards Institute	**American National Standards Institute** El Instituto Nacional de Normalización Estadounidense (ANSI por sus siglas en inglés) es una organización privada sin fines de lucro que administra y coordina la normalización voluntaria y las actividades relacionadas a la evaluación de conformidad en los Estados Unidos.

ETSI World Class Standards	**European Telecomunications Standards Institute.** El Instituto Europeo de Normas de Telecomunicaciones (ETSI por sus siglas en inglés) es una organización de estandarización de la industria de las telecomunicaciones y fabricantes de equipos y operadores de redes) de Europa, con proyección mundial. El ETSI ha tenido gran éxito al estandarizar el sistema de telefonía móvil GSM.
IEEE	**Institute of Electrical Electronics Engineers.** El Instituto de Ingenieros Electricistas y Electrónicos (IEEE por sus siglas en inglés), es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros electricistas, ingenieros en electrónica, ingenieros en computación, ingenieros en biomédica, ingenieros en telecomunicación e ingenieros en mecatrónica.
I E T F	**Internet Engineering Task Force.** Grupo de Trabajo en Ingeniería de Internet (IETF

	por sus siglas en inglés) es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Fue creada en EE. UU. en 1986. La IETF es mundialmente conocida por ser la entidad que regula las propuestas y los estándares de Internet, conocidos como RFCs.
	**International Organization for Standarization.** La Organización de Estándares Internacionales es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional. Es decir, son acuerdos para acatar la mejor propuesta para solucionar un problema.
	**International Telecomunications Union.** La Unión Internacional de Telecomunicaciones (UIT por sus siglas en inglés) es el organismo especializado de la Organización de las Naciones

	Unidas encargado de regular las telecomunicaciones a nivel internacional entre las distintas administraciones y empresas operadoras.
	**System Administration Network Security.** El Instituto SANS (SysAdmin Audit, Networking and Security Institute) es una institución con ánimo de lucro, agrupa a 165,000 profesionales de la seguridad informática (consultores, administradores de sistemas, universitarios, agencias gubernamentales, etc.). Igualmente, el SANS Institute es una universidad formativa en el ámbito de las tecnologías de seguridad. Es una referencia habitual en la prensa sobre temas de auditoría informática. Sus principales objetivos son: Reunir información sobre todo lo referente a seguridad informática (sistemas operativos, ruteadores, firewalls, aplicaciones, IDS, etc.). Ofrecer capacitación y certificación en el ámbito de la seguridad informática.
	**National Institute of Standards and Technology.** El Instituto Nacional de Normas y Tecnología (NIST por sus siglas en inglés) es una agencia de la Administración de Tecnología del Departamento de

	Comercio de los Estados Unidos. La misión de este instituto es promover la innovación y la competencia industrial en Estados Unidos mediante avances en metrología, normas y tecnología de forma que mejoren la estabilidad económica y la calidad de vida.
NOM	**Norma Oficial Mexicana.** La Normatividad Mexicana es una serie de normas cuyo objetivo es asegurar valores, cantidades y características mínimas o máximas en el diseño, producción o servicio de los bienes de consumo entre personas morales y/o físicas, sobre todo los de uso extenso y fácil adquisición por el público en general, poniendo atención en especial en el público no especializado en la materia, de estas normas existen dos tipos básicos en la legislación mexicana, las Normas Oficiales Mexicanas llamadas Normas NOM y las Normas Mexicanas llamadas Normas NMX, de las cuales solo las NOM son de uso obligatorio en su alcance y las segundas solo expresan una recomendación de parámetros o procedimientos, aunque si son mencionadas como parte de una NOM como de uso obligatorio su observancia es a su vez obligatoria.

	**Federal Information Processing Standard.** Los Estándares Federales de Procesamiento de la Información (FIPS por sus siglas en inglés) son estándares anunciados públicamente y son desarrollados por el gobierno de los Estados Unidos para ser utilizados por parte de todas las agencias del gobierno no militares y por los contratistas de su gobierno. Muchos estándares FIPS son versiones modificadas de los estándares usados en las comunidades más amplias (ANSI, IEEE, ISO, etc.).
	**Public-Key Cryptography Standards.** PKCS se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA. A RSA Security se le asignaron los derechos de licenciamiento para la patente de algoritmo de clave asimétrica RSA y adquirió los derechos de licenciamiento para muchas otras patentes de claves. Como resultado de reuniones con un grupo reducido de adoptadores tempranos de la tecnología de claves públicas, los documentos PKCS han sido ampliamente referenciados e implementados a nivel mundial.

Tabla 1. Organismos reguladores.

Una política de seguridad es una declaración que describe cómo los recursos acceden entre sí, qué operaciones pueden llevar a cabo, qué nivel de protección es necesario para un sistema y qué medidas deben tomarse cuando los requisitos no se cumplen. La política describe las expectativas de que el hardware y el software deben cumplir para ser considerados en cumplimiento.

Un modelo de seguridad describe los requisitos necesarios para apoyar e implementar adecuadamente una determinada política de seguridad.

Un programa completo de seguridad debe contener todas las piezas necesarias para proveer protección en su totalidad y estar dentro de la estrategia de seguridad a largo plazo, así mismo debe contener políticas de seguridad, procedimientos, estándares, directrices, líneas base, entrenamiento de concientización de la seguridad, plan de atención a incidentes y un programa de cumplimiento de leyes.

Mientras más detalladas son las reglas, es más fácil saber cuándo una ha sido violada y muchas veces mientras más formales son las reglas, son más difíciles de hacer cumplir.

1.3 IMPORTANCIA DE LA SEGURIDAD EN LAS COMUNICACIONES

Tomando en cuenta el hecho de que los interlocutores podrán transmitir cualquier tipo de información por el canal, incluyendo información sensitiva; el canal de comunicación debe de asegurar que todo lo que viaje a través de él no podrá ser recuperado por cualquier otro interlocutor que no sea el emisor y el receptor originales. Esto asegurará la confianza de los interlocutores para transmitir todo tipo de información.

En la actualidad no es posible concebir una comunicación entre dos puntos interconectados por una red sin que exista algún tipo de seguridad en el medio. Esto debido a los múltiples y sofisticados tipos de intervenciones en las comunicaciones, fuga y/o robo de información, que además se encuentran en constante evolución debido a que son una fuente muy rentable para muchas personas.

Desde hace mucho tiempo y hasta ahora, los sistemas de computadoras están configurados predeterminadamente, con frecuencia por comodidad de los administradores de sistemas.

La intervención de un canal de comunicación puede derivar en severas consecuencias como la suplantación de la identidad de un usuario en una cuenta bancaria en línea, la divulgación de información confidencial de alguna compañía, el robo de credenciales de acceso a un sistema restringido, entre muchas otras.

Una vulnerabilidad es cualquier debilidad que puede explotarse para causar pérdida o daño al sistema.

Una amenaza es cualquier circunstancia con el potencial suficiente para causar pérdida o daño al sistema.

Un ataque son todas aquellas acciones que suponen una violación de la seguridad de los sistemas, esto es su confidencialidad, integridad y/o disponibilidad.

Los ataques se pueden clasificar de acuerdo con los efectos que tienen:

- Interrupción: Cuando un activo del sistema se pierde, se hace indisponible o inutilizable.

- Intercepción: Cuando una persona, proceso u otro sistema de cómputo no autorizado logra acceso a un activo del sistema.

- Modificación: Cuando se alteran datos, comprometiendo la integridad de la información.

- Fabricación: Cuando se inserta código malicioso en los proceso de manufactura o distribución.

A continuación se describen brevemente algunos de los ataques más comunes dirigidos a un medio de comunicación:

- Sniffering

- Hijacking

- Spoofing

- Man in the middle

Sniffering

Un sniffer es un programa y/o dispositivo de red que monitorea toda la información que es transmitida a través de una red de computadoras. Tiene la capacidad de observar los datos que pasan a través de la red y determinar hacia dónde se dirigen los datos, desde dónde vienen y la información que contienen. Además de estas funciones básicas, los sniffers pueden tener algunas características extra que le permiten filtrar cierto tipo de datos, capturar contraseñas y demás (Figura 1.3.1).

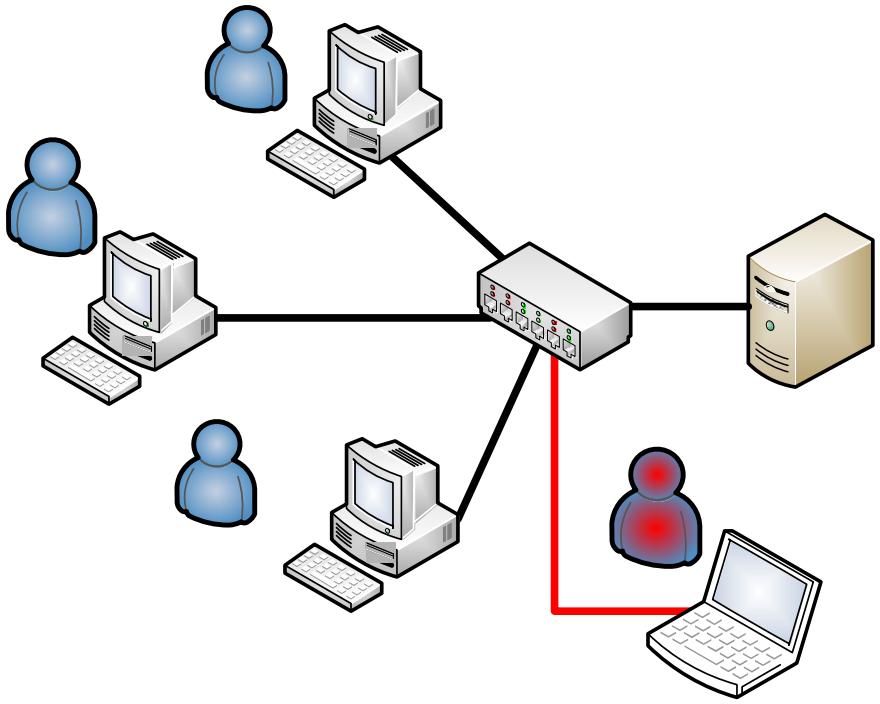


Figura 1.3.1. Estructura de un sniffer.

El sniffer es una de las herramientas más importantes de los usuarios mal intencionados. El sniffer proporciona al atacante un escenario completo (topología de red y direcciones IP) de los datos enviados y recibidos de la computadora o red que se esté monitoreando. Estos datos incluyen correos electrónicos, contraseñas, nombres de usuario y documentos, entre otros. Con todos estos datos, un usuario mal intencionado puede formarse un panorama completo de los datos que viajan en la red.

Para que una computadora y/o dispositivo tenga la capacidad de husmear en una red, debe de tener la tarjeta de red en un modo especial, llamado modo promiscuo, lo cual significa que puede recibir todo el tráfico enviado a través de la red. Una tarjeta de red normalmente solo acepta el tráfico que ha sido enviado específicamente a su dirección de red o Media Access Control (MAC).

Los sniffers son una forma de detectar problemas en la red y hacer diferentes tipos de pruebas para corregir problemas, sin embargo, en la actualidad también son utilizados para obtener ilegalmente la información que viaja en una red.

Los principales usos que les dan los administradores de redes son:

- Análisis de fallos para descubrir problemas en la red.
- Medición del tráfico, mediante el cual es posible descubrir cuellos de botella en algún lugar de la red.
- Para los desarrolladores, en aplicaciones cliente-servidor. Les permite analizar la información real que se transmite por la red.

Algunos sniffers trabajan sólo con paquetes de TCP/IP, pero hay otros más sofisticados que son capaces de trabajar con un número más amplio de protocolos e incluso en niveles más bajos tal como el de las tramas del Ethernet. Algunos de los más utilizados son los siguientes:

- Wireshark, antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones para desarrollo de software y protocolos.
- Ettercap, es un interceptor/sniffer/registrador para redes de área local (LANs por sus siglas en inglés) con switch. Soporta direcciones activas y pasivas de varios protocolos (incluso aquellos cifrados, como SSH – Secure Shell y HTTPS – Hyper Text Transfer Protocol Secure). También hace posible la inyección de datos en una conexión establecida y filtrado al vuelo aun manteniendo la conexión sincronizada.

- Kismet, es un sniffer, y un sistema de detección de intrusiones para redes inalámbricas 802.11. Kismet funciona con cualquier tarjeta inalámbrica que soporte el modo de monitoreo a bajo nivel y puede rastrear tráfico 802.11b, 802.11a y 802.11g. El programa se ejecuta bajo Linux, FreeBSD, NetBSD, OpenBSD, y Mac OS X. El cliente puede también funcionar en Windows.

- TCPDUMP, es una herramienta en línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red. Permite al usuario capturar y mostrar en tiempo real los paquetes transmitidos y recibidos en la red a la cual la computadora está conectada.

La única forma completamente efectiva de evitar el sniffing de una red es cifrar el tráfico.

Hijacking

Se refiere a la explotación de una sesión válida para obtener acceso no autorizado a la información transmitida durante la sesión. Generalmente es utilizado para secuestrar la cookie utilizada para autenticar a un usuario en un servidor remoto.

Existen cuatro métodos utilizados para llevar a cabo un secuestro de sesión:

Session fixation, cuando un atacante fija un identificador de sesión a un usuario conocido, enviándole una liga que contiene ese identificador de sesión particular y esperando a que el usuario se autentique en la sesión.

Session sidejacking, cuando el atacante utiliza el sniffing de los paquetes de la red para robar la cookie de sesión, aprovechando que muchos sitios web solamente cifran la autenticación de la sesión pero no cifran el resto de la sesión, lo que

permite a un atacante obtener mediante el sniffer los paquetes de red que contienen la cookie de sesión.

Cross-site scripting, cuando el atacante engaña al usuario de la computadora ejecutando un código que parece pertenecer al servidor, permitiendo al atacante obtener una copia de la cookie de sesión o ejecutar otras operaciones.

Acceso al equipo, puede permitir al atacante intentar robar la cookie de sesión obteniendo los archivos apropiados en el equipo cliente o en el servidor.

Spoofing

Se refiere a que una persona y/o programa se hace pasar por otra, falsificando sus datos y mediante estas técnicas obteniendo un acceso para suplantar a uno de los interlocutores.

Existen varios tipos de spoofing, dependiendo del tipo de tecnología que se desee suplantar, entre los cuales se encuentran los siguientes:

- ARP Spoofing, suplantación de identidad por falsificación de tabla de resolución de direcciones de protocolos (ARP por sus siglas en inglés), cuando se realiza la traducción de una dirección MAC a una dirección IP.

- IP Spoofing, suplantación de la dirección IP origen, directamente en los paquetes de la red.

- DNS Spoofing, suplantación de identidad por nombre de dominio, al realizar una consulta al DNS para la resolución de una IP por un nombre de dominio y viceversa.

- Web Spoofing, suplantación de una página web real en donde se encamina la conexión de una víctima a través de una página falsa hacia otras páginas maliciosas.

- Mail Spoofing, suplantación en correo electrónico de la dirección de correo electrónico de otras personas o entidades.

Man in the middle

Este tipo de ataque se basa en la premisa de que el atacante tiene acceso al medio por el cual se encuentran comunicándose dos interlocutores, teniendo la capacidad de interceptar y además modificar todos los datos de la comunicación entre ellos sin que ninguno de ellos se percate de que la comunicación está siendo intervenida.

Basándose en este tipo de ataque también podría llevarse a cabo otro tipo de intercepción de la información llamada **eavesdropping** (**escuchar secretamente**). En este tipo de ataques solamente se basa en la captura o escucha de la información que se transmite por el canal y se aplica principalmente a los canales de voz, en donde tiene la capacidad de capturar conversaciones telefónicas.

Este tipo de ataques son considerados ataques pasivos, debido a que el atacante no está afectando el protocolo, el algoritmo, la clave, el mensaje o alguna parte del cifrado del mensaje. Los ataques pasivos son muy difíciles de detectar por lo que en la mayoría de los casos los controles de seguridad son implementados para prevenirlos.

Alterar mensajes, modificar los archivos del sistema o suplantar otra persona son actos considerados como ataques activos debido a que el atacante se encuentra

haciendo algo en vez de solo recolectar datos. Los ataques pasivos son utilizados para obtener la primera información antes de llevar a cabo un ataque activo.

1.4 Requerimientos de la aplicación en cuanto a software y hardware

HARDWARE

Cada distribución de Linux tiene su propio propósito y existen algunos factores que se deben tomar en cuenta para decidir que distribución es la mejor para cada usuario de acuerdo a sus necesidades. Algunas distribuciones tienen mejor desempeño para usuarios caseros, otras son excelentes para configuraciones comerciales y algunas otras funcionan mejor con una arquitectura Intel o Macintosh, además hay otras que son excelentes para ser utilizadas en equipos de alto rendimiento. El hardware mínimo requerido será el indicado por el sistema operativo que se instale.

Los requerimientos mínimos de hardware para una distribución Ubuntu 10.04 son los siguientes (sugerido para la instalación de esta aplicación):

- Procesador Intel o AMD a 800Mhz.
- Memoria RAM de 256M.
- Memoria de video de 16MB.
- Disco Duro de 20GB.

SOFTWARE

- Dos equipos interconectados en red LAN, con comunicación entre ellos.
- Cada uno de los equipos deberá contar con sistema operativo de alguna distribución UNIX, se sugiere Ubuntu 10.04.
- Librerías utilizadas en los equipos UNIX:

- cpp, preprocesador C de GNU (se utilizó la v4.4.3-1 para la aplicación).
- g++, compilador de C++ de GNU (se utilizó la v4.4.3-1 para la aplicación).
- gcc, compilador C de GNU (se utilizó la v4.4.3-1 para la aplicación).
- Perl, Practical Extraction and Report Languaje (se utilizó la v5.10.1-8 para la aplicación).
- libcrypt-gcrypt, interface de Perl para la librería de criptografía de GNU (se utilizó la v1.23-1 para la aplicación).
- libperl, librería de Perl (se utilizó la v5.10.1-8 para la aplicación).
- openssl, Secure Socket Layer (SSL) archivos binarios y herramientas criptográficas relacionadas (se utilizó la v0.9.8k-7 para la aplicación).
- libssl, librería de SSL (se utilizó la v0.9.8k-7 para la aplicación).

CAPÍTULO 2

MARCO TEÓRICO

CAPÍTULO 2

MARCO TEÓRICO

2.1 METODOLOGÍA BÁSICA DE SEGURIDAD

La seguridad en las comunicaciones, en principio, debe estar respaldada por la confianza de que los equipos desde donde se realiza el envío de información se encuentran correctamente asegurados.

A continuación se describen brevemente los principios básicos de la seguridad y la metodología para detectar las fuentes más expuestas para ser comprometidas.

Algunos de los puntos básicos que deben de tenerse en cuenta para asegurar un sistema de información:

- Analizar los riesgos y las vulnerabilidades.
- Diseñar controles y alinearse a las normas y estándares de cada tecnología.
- Priorizar las contramedidas e implementarlas.
- Implementar la detección de intrusos y respuesta a incidentes.
- Crear políticas de seguridad.
- Dirigir revisiones periódicas y pruebas.

Una de las principales indicaciones que se mencionan anteriormente es el detectar correctamente los puntos vulnerables para poder enfocar los esfuerzos principalmente en ellos.

Se denominan activos a los recursos del sistema de información o relacionados con éste, necesarios para que la organización funcione correctamente y alcance

los objetivos propuestos por su dirección. El activo esencial es la información que maneja el sistema, es decir, los datos.

La información es un recurso del cual la organización espera obtener un beneficio. Existen activos tangibles (el capital, la maquinaria, las materias primas, etc.) y activos intangibles vinculados con la propiedad intelectual, el saber hacer y al conocimiento en base a la experiencia, las bases de datos de clientes y las habilidades del personal.

Tipos de Activos:

- Aplicaciones (software) que permiten manejar los datos.
- Equipo (hardware) que permite almacenar datos, aplicaciones y servicios.
- Servicios que se pueden prestar debido a los datos, y los servicios que se necesitan para gestionar dichos datos.
- Soportes de información que son dispositivos de almacenamiento de datos.
- Redes de comunicaciones que permiten intercambiar datos.
- Instalaciones en donde se resguardan los equipos informáticos y los equipos de comunicaciones.
- Personas que explotan u operan todos los elementos anteriormente citados.

La identificación correcta y completa de los recursos permite obtener un listado final en el que se incluirá todo lo que se requiere proteger en la organización.

A continuación se menciona la metodología para identificación de los recursos de información corporativos, diseñada por Burk y Horton:

- Elaboración de un inventario de los recursos potenciales de información de cada área.

- Medición y establecimiento de relaciones entre los costos y los valores de estos recursos.
- Elaboración y análisis de un mapa y/o matrices para mostrar la distribución, localización, significado e interrelaciones de dichos recursos.
- Identificación de los recursos de información corporativos y la determinación de las ventajas y desventajas en cuanto a los acervos y funciones de información, gerencia de recursos de información y de contabilidad y presupuesto como se muestra en la figura 2.1.1.

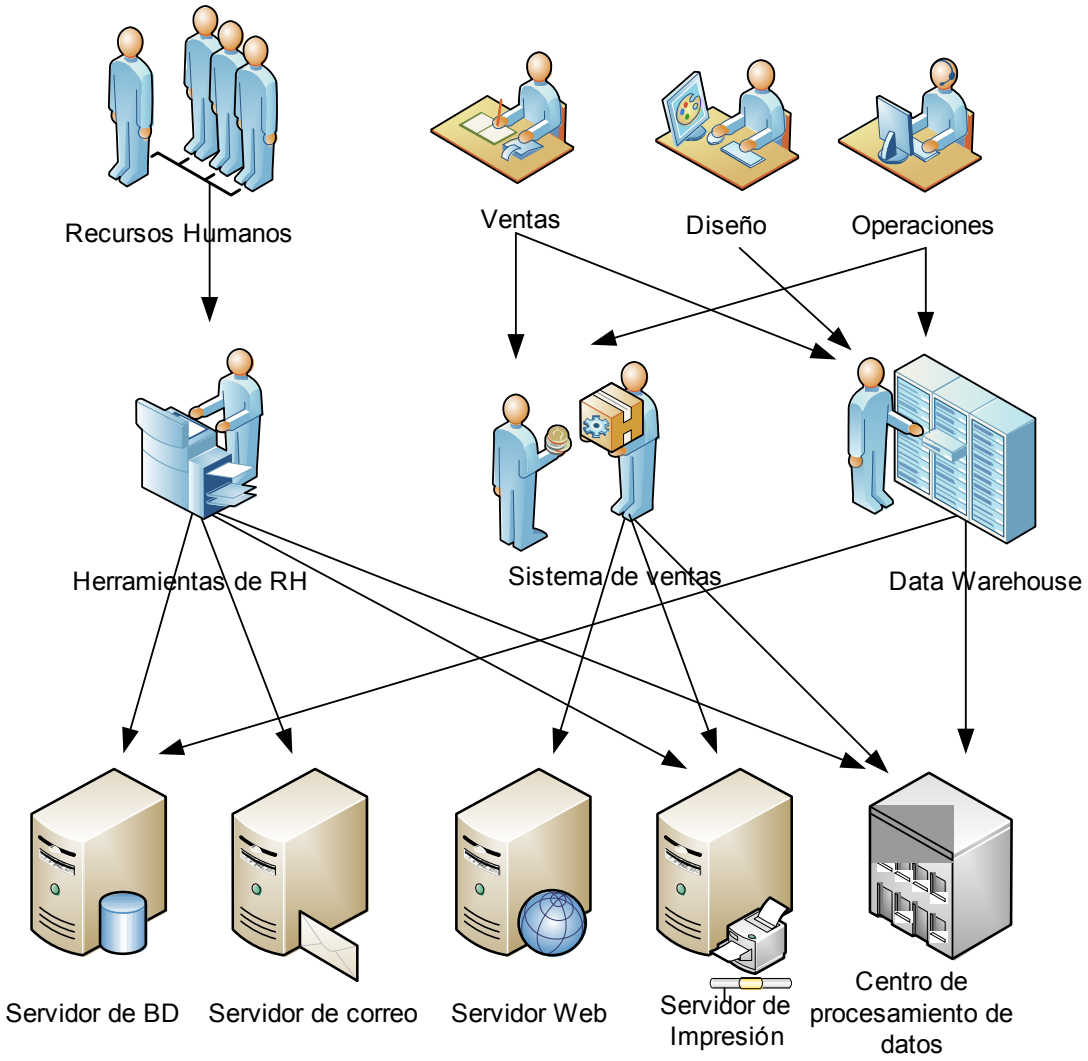


Figura 2.1.1. Identificación de recursos.

Se denomina riesgo a la medida del daño probable sobre un sistema. Conociendo el impacto de las amenazas sobre los recursos, podemos obtener el riesgo inherente, teniendo en cuenta la frecuencia de ocurrencia. También se denomina riesgo a la posibilidad que una amenaza pueda llegar a materializarse. El riesgo crece con el impacto y con la frecuencia como se muestra en la tabla 2.

		FRECUENCIA				
		Infrecuente	Poco frecuente cada varios años	Una vez al año	Mensual	Diario
IMPACTO	Muy Alto	Medio	Alto	Muy Alto	Muy Alto	Muy Alto
	Alto	Bajo	Medio	Alto	Muy Alto	Muy Alto
	Medio	Muy bajo	Bajo	Medio	Alto	Muy Alto
	Bajo	Inexistente	Muy bajo	Bajo	Medio	Alto
	Muy bajo	Inexistente	Inexistente	Muy bajo	Bajo	Medio

Tabla 2. Estimación del riesgo, en función del impacto y la frecuencia.

Existen diversas metodologías para el proceso de identificación de los riesgos basados en las diferentes normas o estándares de control interno.

Mediante la identificación de los riesgos se propone identificar la exposición de una empresa a la incertidumbre.

A continuación se describe brevemente una metodología para realizar la identificación de los riegos:

MAGERIT

Son las siglas de la metodología de análisis y gestión de riesgos de los sistemas de tecnologías de información para realizar un análisis de riesgos y recomendar los controles necesarios para su minimización.

Es una metodología de implementación de sistemas de tecnologías para la seguridad de la información. Se basa en una aproximación cualitativa que intenta cubrir un amplio espectro de usuarios genéricos gracias a un enfoque orientado a la adaptación del mecanismo dentro de diferentes entornos, generalmente con necesidades de seguridad y nivel de sensibilidad también diferentes.

El modelo MAGERIT se apoya en tres submodelos:

- Submodelo de Elementos.
- Submodelo de Eventos.
- Submodelo de Procesos.

Submodelo de Elementos

Se compone de seis entidades clásicas, las cuales se enlistan junto con su definición:

Activos

Los activos son los recursos del sistema de Información o relacionados con éste, necesarios para que la organización funcione correctamente y alcance los objetivos propuestos por su dirección. Se pueden estructurar en cinco categorías:

- Entorno del sistema de información necesario para su funcionamiento: instalación física, infraestructura de comunicaciones, suministros, personal operacional o desarrollador de aplicaciones.
- El sistema de información: hardware, redes propias, software básico, aplicaciones.
- La propia información.

- Las funcionalidades de la organización, que justifican y dan funcionalidad a la existencia de los sistemas de información.
- Otros activos: la credibilidad de una persona jurídica o física, su integridad, su imagen, etc.

Amenazas

Se definen como los eventos que pueden desencadenar un incidente en la organización, produciendo daños materiales o pérdidas no materiales en sus activos. Las amenazas se pueden materializar y transformarse en agresiones.

Vulnerabilidad

Es la ocurrencia real de materialización de una amenaza sobre un activo. Es el mecanismo de paso desde la amenaza hasta la agresión materializada.

Impacto

Es el daño producido a la organización por un posible incidente y es el resultado de la agresión sobre el activo. MAGERIT clasifica el impacto atendiendo a sus consecuencias cualitativas o cuantitativas:

- Impacto cuantitativo, si representa pérdidas cuantitativas rastreables.
- Impacto cualitativo con pérdidas humanas: por ejemplo, daño de personas.
- Impacto cualitativo con pérdidas funcionales.

Riesgos

Se define como la posibilidad de que se produzca un impacto dado en la organización. Es un indicador resultante de la combinación de la vulnerabilidad y el impacto procedente de la amenaza actuante sobre el activo.

El riesgo calculado permite tomar decisiones para cumplir con el objetivo de seguridad de la organización. Para ello, se compara con el umbral de riesgo (nivel determinado con ayuda de la política de seguridad): si es superior implica una decisión de reducción de riesgo y si es inferior queda como un riesgo residual asumible.

Controles

Para reducir el riesgo se necesitan mejorar los controles existentes o incorporar nuevos. MAGERIT distingue entre controles o servicios y mecanismos de control:

- Controles o servicios: es la acción para reducir un riesgo.
- Mecanismo de control: dispositivo, físico o lógico, capaz de reducir el riesgo.

Existen dos tipos de controles:

- Controles preventivos: actúa sobre la vulnerabilidad, neutralizando la materialización de la amenaza antes de que ésta actúe. Es válida en general para amenazas de origen humano, ya sean por error o intencionales.
- Controles correctivos: actúa sobre el impacto, reduciendo el resultado de la agresión, es decir, después de ésta. Es válida en general para amenazas de tipo incidente.

Submodelo de Eventos

Representando intuitivamente este modelo, se define como una “ciudad amurallada”, en la que los activos están dentro y las amenazas son el enemigo exterior. Los controles existentes son las murallas y sus brechas son las vulnerabilidades.

El submodelo de eventos trabaja con dos escenarios:

- Escenario de ataque: coincide con el análisis de los riesgos y parte de la materialización de la amenaza a uno o varios tipos de activos de la organización.

- Escenario de defensa: coincide con la administración de los riesgos y muestra cómo se pueden articular, frente a cada secuencia de ataque, los controles apropiados que funcionarán de una forma específica.

Así, y para cada activo, el escenario de ataque empieza con una amenaza como evento potencial. La vulnerabilidad específica asociada al activo para dicha amenaza permite concretarla o no en una agresión. Ésta desencadena distintos tipos de impacto: un deterioro del activo que puede concretarse en un daño y finalmente en una pérdida, con valor económico o no, según sea el activo.

Submodelo de Procesos

El submodelo de procesos está dividido en cuatro etapas, compuestas por actividades y éstas se desglosan en tareas.

- Planificación del proyecto de riesgos: como consideraciones iniciales para arrancar el proyecto de análisis y administración de riesgos, se estudia la posibilidad de realizarlo, se definen los objetivos que ha de cumplir y el

ámbito que abarcará, planificando los medios materiales y humanos para su realización e inicializando el propio lanzamiento del proyecto.

- Análisis de riesgos: se identifican y valoran los diversos recursos, obteniendo una evaluación del riesgo, así como una estimación del umbral de riesgo deseable.

- Administración de riesgos: se identifican los controles y servicios que minimizarán el riesgo, seleccionando los que son aceptables en función de los controles existentes y las restricciones tras simular diversas combinaciones.

- Selección de controles: se prepara el plan de implantación de los mecanismos de controles elegidos y los procedimientos de seguimiento para la implementación.

Guías metodológicas que componen MAGERIT:

- Guía de aproximación a la seguridad de los sistemas de información.
  Esta guía presenta los conceptos básicos de seguridad de los sistemas de información.

- Guía de procedimientos.
  Esta guía describe metódicamente la fase de análisis y administración de riesgos.

- Guía de técnicas.
  Describe las técnicas y procedimientos de MAGERIT y de los tipos de proyectos en los que el método es útil para mejorar la seguridad de los sistemas de información de las organizaciones, sus departamentos o unidades.

- Guía para desarrolladores de aplicaciones.
  Está diseñada para ser utilizada por los desarrolladores de aplicaciones y está íntimamente ligada con la metodología de planeación y desarrollo de sistemas de información.

- Guía para responsables del dominio.
  Explica la participación de los directivos responsables de un dominio en la realización del análisis y administración de riesgos de aquellos sistemas de información relacionados con los activos cuya administración y seguridad se encuentran bajo su responsabilidad.

2.1.1 Modelos de seguridad

El modelo de seguridad incorpora la política de seguridad que debe ser implementada en el sistema. El modelo es la representación simbólica de la política, en él se interpretan las ideas de los creadores de las políticas en una colección de reglas que debe seguir un sistema de computadoras.

La política de seguridad es un término abstracto que representa las metas y objetivos que un sistema debe reunir y llevar a cabo para ser considerado seguro y aceptable.

El modelo de seguridad interpreta las metas abstractas de la política a términos de sistemas de información para especificar explícitamente las estructuras de datos y las técnicas necesarias para cumplir con la política de seguridad. El modelo de seguridad es representado en ideas matemáticas y analíticas, las cuales son interpretadas en especificaciones del sistema y desarrolladas por los programadores.

Una política de seguridad bosqueja las metas sin considerar cómo serán cumplidas. Un modelo es un marco que da forma a la política y resuelve problemas de seguridad para situaciones particulares. Algunos modelos de seguridad han sido desarrollados para cumplir con las políticas. A continuación se describe brevemente cada uno de los modelos.

Modelo de Máquina de Estados

En el modelo de máquina de estados, para verificar la seguridad de un sistema se utiliza el estado, lo cual significa que todos los permisos y todas las instancias actuales de los sujetos accediendo a objetos deben ser capturados. Manteniendo el estado de un sistema, es posible obtener la asociación de cada sujeto con sus objetos. Si los sujetos pueden acceder solo a los objetos que son concurrentes con la política de seguridad, el sistema es seguro. Los estados de máquina han proporcionado una importancia básica a los modelos de seguridad. El estado de un sistema es una foto instantánea en un momento del tiempo. Muchas actividades pueden alterar su estado, a las cuales nos referiremos como estados de transición. Si todas las actividades que podrían pasar en el sistema, no comprometen el sistema o lo ponen en un estado inseguro, entonces el sistema ejecuta el modelo de máquina de estados seguro.

El modelo de la máquina de estados es utilizado para describir el comportamiento de un sistema bajo diferentes entradas. Proporciona construcciones matemáticas que representan colecciones (sujetos y objetos) y secuencias. Cuando un objeto acepta una entrada, se modifica a un estado variable. Algunos estados de transición son simples, pero la complejidad viene cuando el sistema decide si una transacción debe ser aceptada. Para permitir esta transición, los atributos de seguridad del objeto y los permisos de acceso del sujeto deben ser revisados y permitidos por el sistema operativo.

Se deben identificar los estados iniciales (valores predeterminados de las variables) y bosquejar cómo pueden cambiar estos valores (entradas que pueden ser aceptadas) para que los diversos estados finales (valores de resultado) sigan asegurando que el sistema es seguro.

El sistema que está empleando un modelo de máquina de estados deberá estar en un estado seguro siempre y en cada momento de su existencia. Es muy importante que si algo que no es seguro toma lugar, el sistema sea capaz de salvarse a sí mismo y no volverse vulnerable.
Después de que las funciones de los estados de transición fueron definidas, deben de ser probados para verificar que el total de los estados de la máquina no son comprometidos y que estas funciones de transición conservarán intacta la integridad del sistema en todo momento.

Modelo de Bell-LaPadua

Este modelo de confidencialidad describe el flujo de la información permitida y formaliza la política de seguridad militarizada. Es el primer modelo matemático de una política de seguridad multinivel que define el concepto de un estado seguro y los modos de acceso necesarios.

- La regla de seguridad simple. Un sujeto no puede leer un dato en un nivel de seguridad más elevado.

- La regla de seguridad simple propietaria. Un sujeto no puede escribir datos en un objeto en un nivel de seguridad más bajo.

- La regla propietaria de estrella fuerte. Un sujeto puede ejecutar funciones de lectura y escritura en objetos de su mismo nivel de seguridad.

El modelo Biba

Este modelo protege la integridad de la información dentro de un sistema y las actividades en las que toma lugar.

- El axioma simple de integridad. Un sujeto no puede leer datos de un nivel de integridad más bajo.
- El axioma simple de integridad propietario. Un sujeto no puede modificar un objeto en un nivel de integridad más alto.

El modelo Clark-Wilson

Este modelo de integridad es implementado para proteger la integridad de los datos y para asegurar que se ejecutan transacciones propiamente formadas.

- Los sujetos pueden acceder a objetos solamente a través de programas autorizados.
- La separación de las obligaciones es reforzada.
- La auditoría es requerida.

Modelo de matriz de control de accesos

Éste es un modelo en el cual las decisiones de acceso son basadas en las listas de control de acceso de los objetos y en las tablas de capacidades de los sujetos.

Modelo de flujo de información

Éste es un modelo en el cual la información es restringida en su flujo para ir solamente hacia y desde entidades de una manera en la que no se pueda invalidar la política de seguridad.

El modelo de no interferencia

Este modelo expone aquellos comandos y actividades ejecutándose en un nivel de seguridad en el que no deberían verse o afectar a objetos o sujetos en un nivel de seguridad diferente.

El modelo Brewer y Nash

Este modelo permite el cambio dinámico de control de accesos para protección contra conflictos de intereses. También conocido como el modelo de la muralla china.

El modelo Graham-Denning

Este modelo muestra cómo los sujetos y los objetos deberían ser creados y eliminados. También direcciona como se deben asignar los accesos correctos.

2.2 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DEL LENGUAJE DE PROGRAMACIÓN PERL

Es un lenguaje de programación interpretado, basado en un estilo de bloques, fue creado por Larry Wall en 1987. Su nombre PERL, lenguaje práctico para la extracción de informe por su significado en inglés Practical Extraction and Report Language.

Originalmente se diseñó para la manipulación de texto, pero en la actualidad también es utilizado para muchos más propósitos como para desarrollar scripts

que ayudan a la administración de sistemas UNIX, desarrollo web, programación de sockets, entre muchos otros.

Este lenguaje se deriva principalmente de C, además de los lenguajes de programación de UNIX: shell, sed y AWK, entre otros.

Algunas de las características más importantes del lenguaje es que cuenta con manejo de la memoria automáticamente, tipos de datos dinámicos, cadenas, listas, además de expresiones regulares.

Además de que soporta los siguientes paradigmas de la programación, la imperativa, funcional y orientada a objetos.

Existe una colección muy importante de sitios web que se dedica a almacenar y distribuir fuentes, binarios, documentación, scripts y módulos de perl, llamada CPAN (Comprehensive Perl Archive Network por sus siglas en inglés). Este sitio puede ser accedido desde la siguiente liga http://www.cpan.org.

La ejecución de un programa escrito en perl, se divide en dos fases: tiempo de compilación, en donde se manipula el texto para convertirlo en un árbol sintáctico y el tiempo de ejecución, en donde se ejecuta el programa siguiendo dicho árbol. Una vez realizado este proceso, conocido como parseo, el árbol sintáctico es optimizado antes de ser ejecutado, por lo que la fase de ejecución es más eficiente, lo cual incluye la simplificación de constantes, propagación del contexto y optimización de trozos sueltos de código. Es un lenguaje dinámico y tiene una gramática sensitiva al contexto, utiliza su propio analizador léxico debido a la versatilidad del lenguaje.

Cuenta con una interfaz para bases de datos y debido a su facilidad del manejo de textos permite generar consultas a bases de datos.

Es un lenguaje de código abierto de GNU (General Public License, licencia pública general por sus siglas en inglés) el cual cuenta con distribuciones para muchos sistemas operativos.

Ventajas:

- Poderoso, estable, maduro y portable.
- Utilizado para misiones críticas en los sectores público y privado.
- Código de alta calidad.
- Cuenta con un sistema que permite el fácil procesamiento de textos.
- Cuenta con una colección de módulos muy extensa.
- Permite la fácil manipulación del código.
- Soporta el estándar de codificación de caracteres (unicode).
- Maneja integración a bases de datos.
- Cuenta con una interface con librerías C/C++.
- Puede ser embebido en otros sistemas.
- Fácil de utilizar.
- Soporta:
  - Programación estructurada,
  - Programación orientada a objetos y
  - Programación funcional.
- Está diseñado para el uso eficiente del procesador.
- Software libre, contribuye a la reducción de costos.

Desventajas:

- Lenguaje interpretado, por lo que puede llegar a ser lento en la ejecución.
- Debido a la versatilidad del compilador, los errores pueden ser difíciles de encontrar.

- La portabilidad se puede llegar a dificultar dependiendo de los módulos que se estén utilizando.[1]

2.3 ARQUITECTURAS DE APLICACIONES CLIENTE-SERVIDOR

Se entiende como arquitectura la interacción de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados de forma eficaz dentro de la organización.

El modelo cliente-servidor es utilizado ampliamente y forma la base en gran medida del uso de las redes. Básicamente, la arquitectura cliente-servidor proporciona la capacidad a un sistema de aplicaciones para ser dividido a través de múltiples plataformas que pueden variar en el sistema operativo y el hardware. El cliente solicita servicios y el servidor cubre los requerimientos, por lo que el cómputo cliente-servidor define una arquitectura en la cual la lógica del programa está distribuida entre sistemas cliente y sistemas servidor.

2.3.1 Cliente

El cliente realiza las peticiones hacia el servidor. Es un consumidor de servicios y normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de información; se encuentra desarrollado sobre plataformas que permiten construir interfaces con los usuarios, tienen la capacidad de acceder a los servicios de los servidores.

2.3.2 Servidor

[1] http://www.perl.org, http://perlenespanol.com

Los servidores proveen servicios a uno o varios clientes y están a la espera de las peticiones de los clientes de algún recurso administrado por él, este proceso es conocido con el término back-end. El servidor generalmente maneja todas las funciones relacionadas con las reglas del negocio y los recursos de los datos.
Por cuestiones de desempeño, los servidores deben ser sistemas poderosos.

2.3.3 Características de aplicaciones cliente-servidor

En los ambientes cliente-servidor, el servidor realiza la mayor parte del trabajo mientras que el cliente básicamente muestra la información y recibe las solicitudes de los usuarios. En modelos más balanceados, el cliente y servidor (o múltiples servidores) comparten el trabajo. Las características básicas de las arquitecturas cliente-servidor son:

- Combinación de un cliente o parte de la presentación de una aplicación que interactúa con el usuario y el servidor o parte del procesamiento de datos que interactúan con los recursos compartidos. El proceso del cliente contiene la solución lógica específica entre el usuario y el resto de las aplicaciones del sistema. El proceso del servidor actúa como el motor del software que maneja los recursos compartidos como base de datos, impresoras, módems o procesadores de alto poder.
- Las tareas de cliente y del servidor fundamentalmente tienen diferentes requerimientos de los recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidad del disco, además de dispositivos de entrada-salida.
- El entorno es normalmente heterogéneo y de múltiples proveedores. La plataforma del hardware y sistema operativo del cliente y del servidor no son usualmente la misma. Los procesos del cliente y del servidor se comunican a través de un bien definido conjunto de API's (Application

Program Interface, Interface del programa de aplicaciones) y RPC's (Remote Procedure Call, llamada a procedimiento remoto).

- Una característica muy importante de los sistemas cliente/servidor es la escalabilidad. Pueden ser escalados horizontalmente y verticalmente.
- Escalamiento horizontal significa añadir o remover estaciones de trabajo clientes con solo un leve impacto en el rendimiento.
- Escalamiento vertical significa una migración a un equipo servidor más grande y más rápido o multiservidores.

2.3.4 Ventajas del modelo cliente-servidor

A continuación se describen las ventajas más sobresalientes del modelo cliente-servidor:

- El modelo cliente-servidor ayuda a las organizaciones a reducir los mainframes y minicomputadoras a redes que provean plataformas de comunicación de datos a lo largo de la empresa.
- La división entre cliente y servidor permite a los programadores tomar ventaja de sistemas de clientes poderosos que ejecuten interfaces gráficas como Windows o servidores web.
- Los datos son almacenados cerca de los servidores que trabajan con ellos, minimizando la cantidad de información a enviar por la red; los respaldos son fáciles de realizar.
- Un alto porcentaje de información es almacenada en la memoria del servidor en lugar de cada memoria de cada estación de trabajo que la requiera.
- El tráfico de la red es reducido debido a que el servidor únicamente proporciona al cliente la información requerida, no así grandes cantidades de información que la estación de trabajo debe procesar.

- Los grandes sistemas de servidor pueden liberar aplicaciones que son manejadas por las estaciones de trabajo personales.
- Los datos están a salvo y seguros en una ubicación. El almacén de datos orientado a un ámbito (data warehouse) provee una forma fácil de hacer disponible datos específicos en un grupo de trabajo intermedio mientras mantiene control de los datos.
- Con datos centralizados, los administradores pueden aplicar controles de seguridad para restringir los accesos a los datos y pueden utilizar mecanismos de rastreo para monitorear los accesos a los datos.
- Múltiples sistemas pueden involucrarse en un proceso paralelo, en los cuales ellos cooperan en completar una tarea de procesamiento.

El servidor maneja los servicios de procesamiento de datos y proporciona los resultados procesados al cliente.

El cliente representa la presentación de la aplicación, y el servidor representa las funciones o servicios, lo cual es usualmente una labor más intensa.

La presentación usualmente incluye la interface de usuario y la capacidad de manejo de datos locales, además provee los mecanismos de comunicación que pueden requerir servicios de la parte del servidor de la aplicación

La comunicación en el modelo cliente-servidor, es de la siguiente forma y estos mensajes se muestran en la figura 2.3.4.1.

- El proceso cliente envía una solicitud a través de la red al proceso servidor.
- El cliente espera una respuesta.
- El proceso servidor recibe la solicitud y ejecuta el trabajo que se le pide o busca los datos solicitados.
- El servidor devuelve una respuesta.

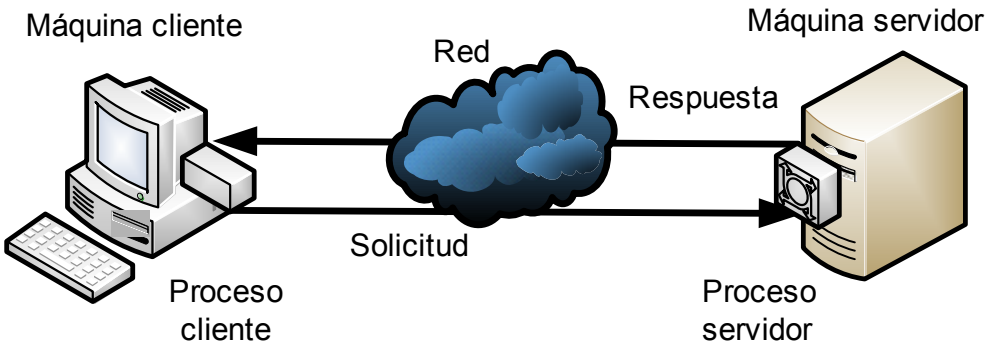


Figura 2.3.4.1 Modelo cliente-servidor el cual implica solicitudes y respuestas.

Un protocolo cliente-servidor dicta la forma en que los clientes solicitan información y servicios de un servidor y también cómo el servidor responde a esa petición. Algunos ejemplos de los protocolos de cliente-servidor son NetBIOS, RPC, comunicación avanzada programa a programa (APPC por sus siglas en inglés), canalizaciones con nombre, sockets, Interface nivel de transporte (TLI), e intercambio de paquetes secuenciados (SPX).

2.3.5. Arquitectura de una capa

Las aplicaciones cliente-servidor de una capa concentran la lógica de presentación, lógica de aplicación y fuente de datos en una sola entidad, por lo que la aplicación se comporta como cliente y servidor simultáneamente.

2.3.6. Arquitectura de dos capas

El modelo de dos capas está compuesto por un cliente que mantiene la lógica de la aplicación, y un servidor que administra las peticiones a la base de datos. Éste tipo de aplicaciones son generalmente utilizadas cuando se requiere poco volumen de procesamiento de datos, además de que la organización tiene una

base de datos centralizada en el mismo servidor donde reside la aplicación. Una arquitectura de dos capas es cuando el cliente se comunica directamente con el servidor, sin un servidor intermedio, esto es usado generalmente en entornos pequeños con menos de 50 usuarios.

2.3.7. Arquitectura de tres capas

En la arquitectura de tres capas se introduce un servidor entre el cliente y el servidor. El rol de este servidor intermedio es variable, puede proveer servicios de traducción, servicios de medición o servicios de agente inteligente.

Con el avance de Internet, el modelo de cómputo cliente-servidor ha evolucionado del modelo de dos capas, es decir de una relación de dos vías, al modelo de tres capas o multicapas, en la cual sus clientes se comunican con aplicaciones servidor intermedias o servidores web, los cuales se comunican también con servidores de datos back-end y/o sistemas que ya existen. Los servidores intermedios regresan las consultas de la base de datos a los clientes.

El modelo de tres capas está constituido en un lado por clientes y los servidores en el otro, la capa intermedia es la que provee los servicios, manejo de transacciones y otras reglas de negocios. El lado del servidor puede consistir en servidores de bases de datos, datawarehouses (colección de datos orientada a un determinado ámbito) y datamarts (pequeños datawarehouse centrados en un tema o un área de negocio específico dentro de una organización), etc. El componente middleware es comúnmente referenciado como el servidor de aplicaciones.

En el modelo de tres capas, la mayoría de los accesos y manipulación de datos es removida del cliente y se coloca en un sistema donde residan los datos en la capa del back-end. En el ambiente corporativo los sistemas de nivel medio pueden contener la lógica del negocio para la organización.

La lógica del negocio incluye reglas, procedimientos y secuencias operacionales que proveen servicios para sistemas de procesamientos de datos en varias capas. Ver figura 2.3.7.1.

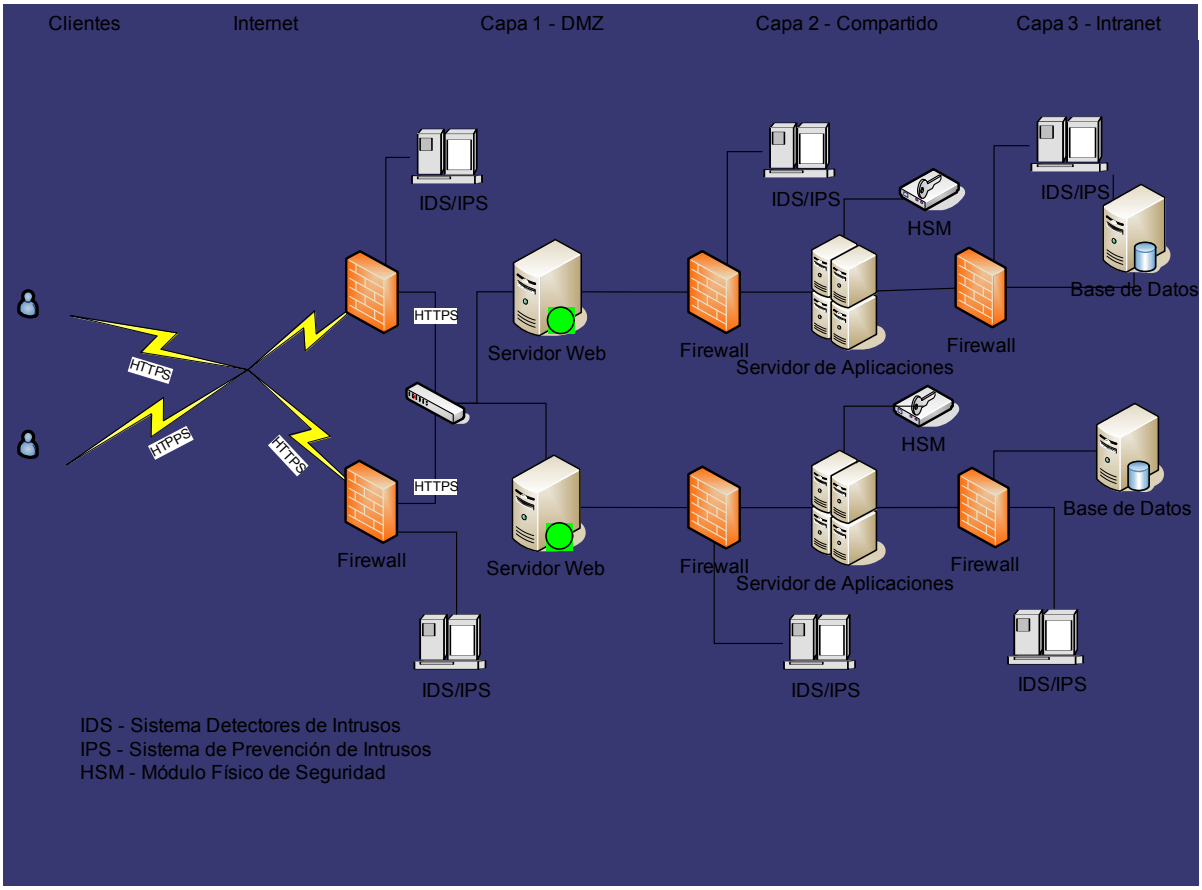


Figura 2.3.7.1 Arquitectura de tres capas en alta disponibilidad.

2.3.8. Aplicaciones distribuidas o multicapas

El modelo distribuido de aplicaciones quiere decir que los clientes pueden interactuar con varios servidores diferentes que se encuentren en la red; los navegadores de red son los clientes universales para acceder a las aplicaciones y recursos de sistemas locales y remotos dentro o fuera de la red. El enfoque

objeto-componente se refiere a descomponer programas complejos en componentes más pequeños que hacen más fácil de distribuir y actualizar la aplicación. El modelo distribuido de aplicaciones es un modelo que balancea las cargas de procesamiento entre el cliente y el servidor. Vea la figura 2.3.8.1.

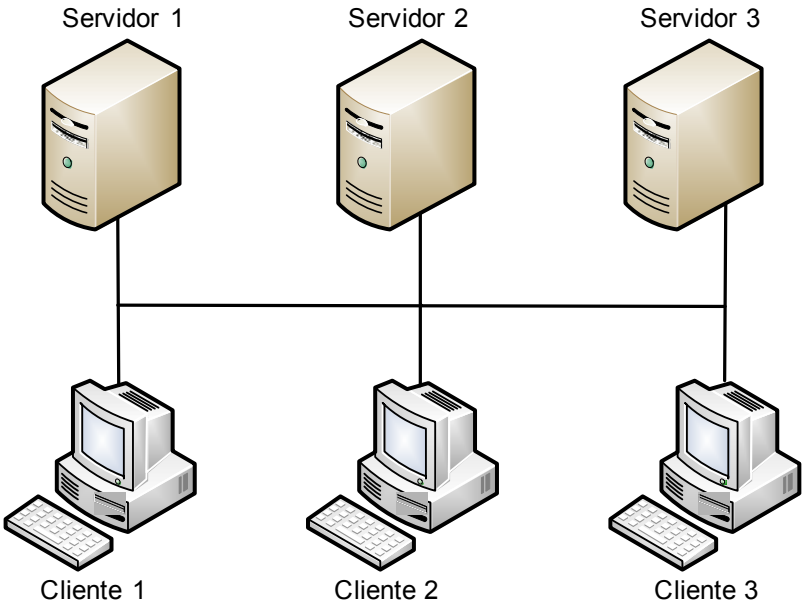


Figura 2.3.8.1 Configuración cliente-servidor distribuido.

2.3.9 Sockets

Se conoce como socket al punto de comunicación a través del cual un proceso puede interactuar con otro. Los procesos que intercambian información a través de sockets pueden pertenecer a la misma capa o bien pueden encontrarse en diferentes capas de arquitectura.

Los sockets de Berkeley junto con las TLI, por sus siglas en inglés de Transport Layer Interfase del System V, son las interfaces API (Application Programming Interface por su nombre en inglés) de comunicaciones más comunes en sistemas UNIX.

Dominios de un socket

Los dominios definen básicamente dos características:

- La familia de protocolos que estarán disponibles para mediar el intercambio de datos entre los dos sockets.
  - El formato de las direcciones de red que se usarán para identificar ambos extremos de la comunicación.

Los dominios más comunes son:

- Dominio UNIX: los sockets de este dominio son locales al sistema donde son creados y permiten la comunicación interna entre dos procesos.
- Dominio Internet: los sockets de este dominio hacen posible la comunicación de dos procesos a través de una red TCP/IP. Para ello las direcciones de los sockets de este dominio permiten especificar direcciones IP y puertos TCP/UDP.

2.4 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DE CIFRADO SIMÉTRICO Y ASIMÉTRICO

Las dos piezas principales del cifrado son los algoritmos y las claves. Los algoritmos utilizados en los sistemas son fórmulas matemáticas complejas que dictan las reglas de cómo el texto plano será convertido a texto cifrado. Una clave es una cadena de bits aleatorios que serán utilizados por el algoritmo para agregarse a la aleatoriedad del proceso de cifrado. Para que dos dispositivos sean capaces de comunicarse de forma cifrada, deben utilizar el mismo algoritmo y, en algunas ocasiones, la misma llave. En algunas tecnologías de cifrado, deberán usarse claves diferentes pero relacionadas para propósitos de cifrado y descifrado.

Existen algoritmos criptográficos que pueden ser algoritmos simétricos, los cuales utilizan claves simétricas, y algoritmos asimétricos, los cuales emplean claves asimétricas, también conocidas como claves públicas y privadas.

Existen dos grandes grupos de cifrados: los algoritmos que usan una única clave tanto en el proceso de cifrado como en el de descifrado, y los que emplean una clave para cifrar mensajes y una clave distinta para descifrarlos. Los primeros se denominan cifras simétricas, de clave simétrica o de clave privada, y son la base de los algoritmos de cifrado clásico. Los segundos se denominan cifrados asimétricos, de clave asimétrica o de clave pública y forman el núcleo de las técnicas de cifrado modernas.

Algoritmos simétricos

En un criptosistema que emplee criptografía simétrica, el emisor y receptor utilizan dos instancias de la misma clave para cifrado y descifrado como se muestra en la figura 2.4.1 entonces la clave tiene una función dual, en la que lleva a cabo el proceso de cifrado y descifrado. Las claves simétricas también son conocidas como claves secretas, debido a que este tipo de cifrado requiere que cada usuario mantenga la clave secreta protegida adecuadamente. Si un usuario mal intencionado obtiene esta llave, puede descifrar cualquier mensaje que haya interceptado y que esté cifrado con esta llave.

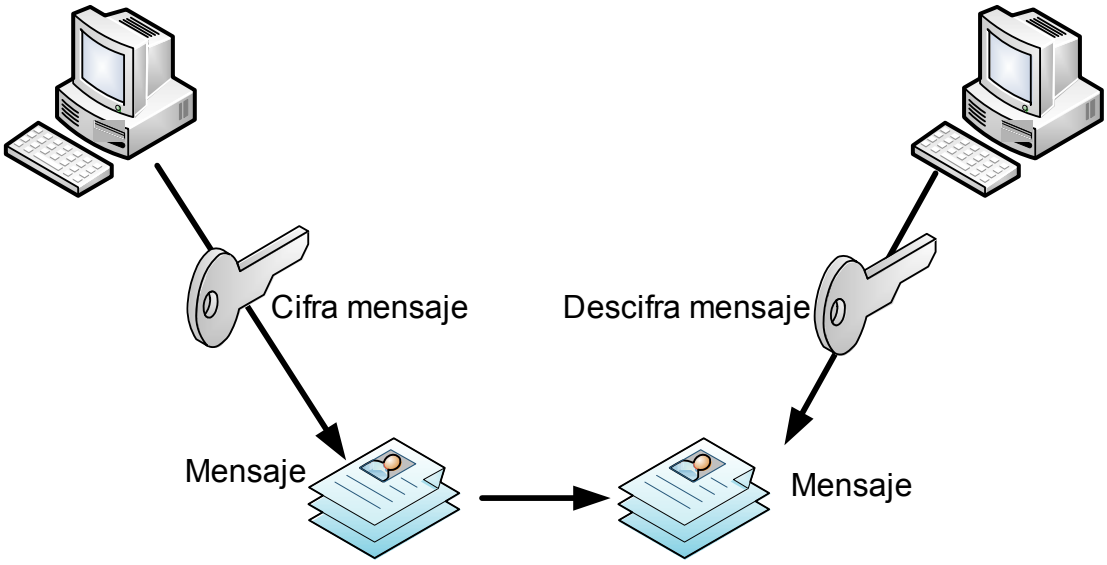


Figura 2.4.1 El cifrado simétrico utiliza la misma llave.

Cada par de usuarios que requieran intercambiar datos utilizando cifrado de claves simétricas deberán tener dos instancias de la misma llave. Eso significa que si Daniela y Edgar se quieren comunicar, ambos requieren obtener una copia de la misma llave. Si Daniela también se quiere comunicar utilizando cifrado simétrico con Eduardo y Pamela, ella requiere tener tres claves separadas para cada comunicación. Esto no es mucho problema hasta el momento en que se requiera comunicar con cientos de personas, por lo que darle seguimiento y usar la clave correcta que corresponda a cada receptor específico puede ser una tarea pesada. Si un grupo de diez personas requieren comunicarse en forma segura entre ellos, entonces se requieren 45 claves. Si 100 personas se quieren comunicar, entonces 4950 claves son involucradas. La ecuación que se utiliza para calcular el número requerido de claves simétricas es:

$$N(N-1)/2 = \text{número de claves requeridas.}$$

Donde N es igual al número de integrantes que se quieren comunicar.

Algoritmos asimétricos

La criptografía de clave pública a diferencia de los algoritmos de clave simétrica, requiere que cada usuario tenga dos claves, una clave pública y una clave privada. La clave pública es usada por cualquiera para cifrar los mensajes a enviar a éste usuario. La clave privada es requerida por el usuario para descifrar los mensajes. De esta forma la robustez del algoritmo depende de la fórmula matemática y de la longitud de la clave empleada.

La principal característica de la criptografía de clave pública, es el uso de algoritmos de clave asimétrica, donde la clave de cifrado es diferente que la clave de descifrado. Las claves se encuentran relacionadas matemáticamente, pero no es factible que la clave privada sea derivada de la clave pública.

Las técnicas de claves asimétricas son frecuentemente utilizadas para proteger las comunicaciones o para autenticar mensajes.

Aunque RSA es utilizado ampliamente, existen otros esquemas de clave pública los cuales se basan en la dificultad para calcular logaritmos discretos como “El Gamal”, “Schnor” o los basados en curvas elípticas, pero las categorías principales son las basadas en la dificultad para factorizar números grandes y calcular logaritmos discretos, módulo un número primo grande.

Los cifrados asimétricos más conocidos y utilizados como algoritmos de firmas digitales son RSA y DSA (por sus siglas en inglés Digital Signature Algorithm).

El algoritmo DSA fue desarrollado por la NSA, por sus siglas en inglés de agencia de seguridad nacional de EUA. DSA, a diferencia de RSA, solo puede ser utilizado para firmas digitales, cifrado y la distribución segura de claves simétricas.

La longitud de la clave asimétrica es directamente proporcional a la robustez del cifrado resultante, esto es, mientras mayor sea la longitud de la llave, será más difícil de romper el cifrado. Sin embargo, la longitud de la clave es directamente proporcional al esfuerzo o poder computacional requerido para cifrar la cadena de texto original. Por lo tanto, se recomienda emplear una longitud de clave que sea suficientemente segura, y que sea capaz de incrementar la longitud con el tiempo a la par en que se va incrementando el poder del cómputo.

La mayor desventaja a diferencia de los algoritmos de clave simétrica, es que computacionalmente son más lentos.

CAPÍTULO 3

ANÁLISIS Y PLANTEAMIENTO DEL PROBLEMA

CAPÍTULO 3

ANÁLISIS Y PLANTEAMIENTO DEL PROBLEMA

3.1 CONTEXTO SITUACIONAL DEL PROBLEMA

El constante incremento en la fuga de información electrónica hace necesario crear medios de comunicación seguros entre dos equipos interconectados por una red, en el cual, la comunicación sea legible únicamente por los interlocutores involucrados.

Para los usuarios es muy importante contar con medios de comunicación seguros para poder transmitir información confidencial a través de una red de computadoras.

En la figura 3.1.1 se muestra el modelo simple de comunicación con un emisor que transmite un mensaje a un receptor.

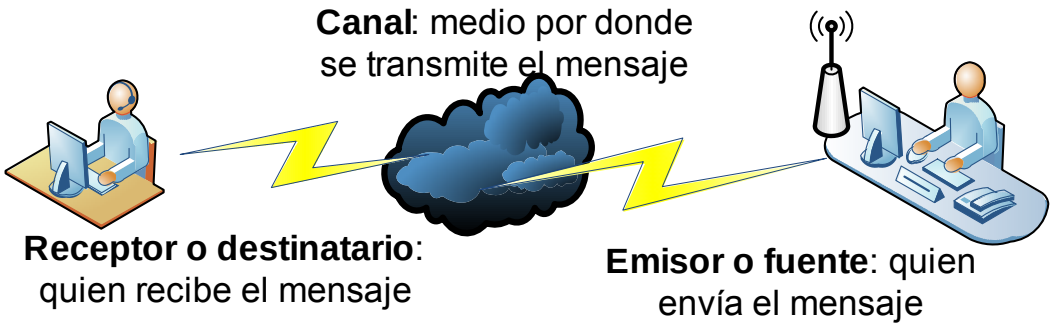


Figura 3.1.1 Modelo simple de comunicación.

Actualmente no se puede garantizar que la información no pueda ser interceptada debido a que el medio es vulnerable por defecto, como se muestra en la figura 3.1.2, la cual representa la fuente de la información donde el codificador que envía

el mensaje a través del medio de comunicación o canal de comunicación puede ser interceptado (representado en el rayo que cae), llega a un decodificador y luego al receptor, el cual podría emitir a su vez una respuesta.

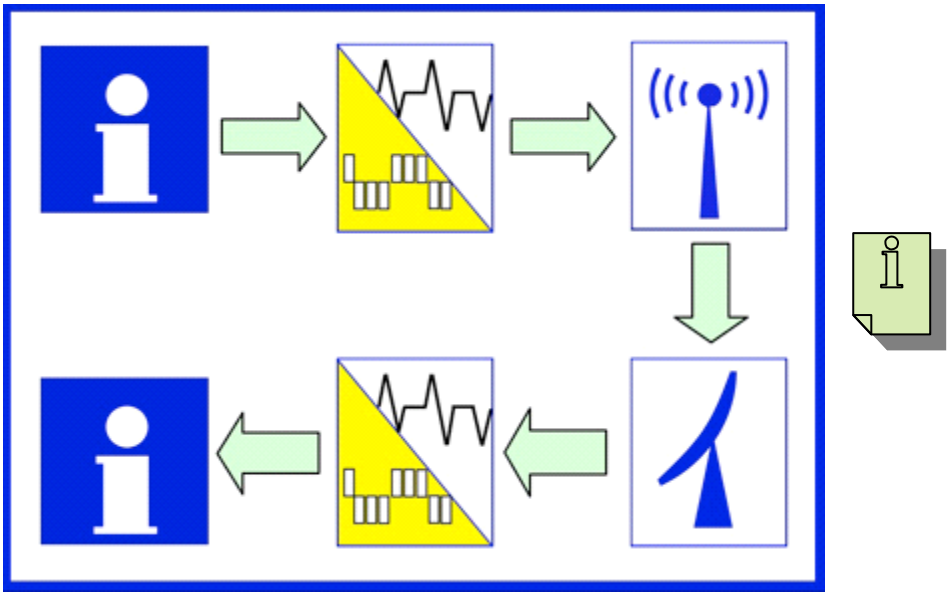


Figura 3.1.2 Esquema del proceso de comunicación o transmisión de la información.

Dentro de los ataques más comunes para intercepción de comunicaciones entre dos sistemas, se encuentran los siguientes ataques:

- Sniffering
- Telnet Hijacking
- Session Hijacking
- Spoofing
- Man In The Middle

Actualmente se cuenta con herramientas de código abierto, las cuales están disponibles para usuarios que no cuenten con presupuesto suficiente para asegurar sus comunicaciones.

De acuerdo al Top 10 del 2010 de riesgos de seguridad en aplicaciones Web del OWASP, (proyecto abierto de seguridad en aplicaciones web, por sus siglas en inglés) se encuentran los siguientes riesgos:

- Inyección de código
- Cross-Site Scripting (XSS por sus siglas en inglés)
- Autenticación rota y manejo de sesión
- Referencia insegura directa a objetos
- Cross-Site Request Forgery (CSRF por sus siglas en inglés)
- Mala configuración de seguridad o predeterminada
- Almacenamiento criptográfico inseguro.
- Fallas de acceso restringido a URL
- Protección insuficiente en la capa de transporte
- Redireccionamientos y reenvíos no validados

En la tabla 3 se describe el riesgo de protección insuficiente en la capa de transporte.

Amenaza	Vectores de Ataque	Debilidades de seguridad		Impactos técnicos	Impacto al negocio
———	**Dificultad de explotación**	**Predominio COMÚN**	**Detección FÁCIL**	**Impacto MODERADO**	———
Considerar a cualquiera que pueda monitorear el tráfico en la red. Si la aplicación está en internet, saber quién sabe cómo los usuarios acceden. No olvidar conexiones back-end.	Monitorear el tráfico de los usuarios de la red puede resultar difícil, pero muchas veces es fácil. La dificultad primaria recae en monitorear adecuadamente el tráfico de la red mientras los usuarios accedan un sitio vulnerable.	Las aplicaciones frecuentemente no protegen el tráfico de la red. Se puede utilizar SSL/TLS durante la autenticación, así como el no exponer los datos y los identificadores de la sesión para ser interceptados. Los certificados expirados o mal configurados también son errores comunes. Detectar defectos de seguridad básicos es fácil, solo hay que observar el tráfico de la red, los defectos de seguridad más complejos requieren inspeccionar el diseño de la aplicación y configuración de los sistemas.		Dichos defectos exponen los datos de los usuarios y puede derivar en el robo de cuentas. Si la cuenta comprometida es de un administrador, el sitio completo pudo ser comprometido. Una mala configuración del SSL puede facilitar ataques de phishing y de Man In The Middle.	Considerar el valor de negocio de los datos expuestos en los canales de comunicaciones en términos de necesidades de confidencialidad e integridad y la necesidad de autenticar ambos participantes.

Tabla 3. Insuficiente protección en la capa de transporte.

3.2 REQUERIMIENTOS GENERALES Y PARTICULARES DE INFORMACIÓN

La seguridad en los sistemas de tecnologías de la información en la actualidad es una necesidad para proteger uno de los activos más importantes de cualquier organización, la información.

Cualquier sistema de información y/o comunicación debe contar con al menos algún método para evitar que los datos puedan ser comprometidos por algún intruso y preferiblemente deben de tratar de alcanzar las metas de la seguridad de

la información, es decir, que la información debe estar disponible, íntegra y confidencial.

3.2.1 Requerimientos generales

- Establecer una comunicación en tiempo real entre los usuarios de la red ya que es imprescindible para la comunicación eficiente entre las áreas.
- Debido a la naturaleza de la información que podrían manipular los usuarios, la comunicación debe ser confidencial, es decir, se debe garantizar que únicamente los involucrados tengan acceso a la información destinada para ellos.
- Se requiere implementar un canal de comunicación que proporcione a los usuarios de la red la plena confianza de que la información que transmitan a través de este canal no podrá ser recuperada por algún otro usuario y/o intruso.
- El canal de comunicación se establecerá de punto a punto desde las estaciones de trabajo hacia el servidor, en los cuales se autenticarán con sus respectivas credenciales durante su inicio de sesión.
- El certificado con el cual será cifrada la información deberá poder ser verificado con una entidad certificadora.
- La comunicación entre los usuarios deberá ser en tiempo real o al menos, el retraso en la transmisión de los mensajes debe ser relativamente bajo como para que los usuarios no se percaten.
- La comunicación establecida entre el cliente y el servidor debe ser bidireccional, es decir, ambos tendrán la capacidad de enviar y recibir información.
- El ejecutable del programa que hace las funciones del servidor deberá estar alojado en un equipo que se encuentre disponible permanentemente para todos los usuarios que deseen comunicarse con él.

- Los ejecutables de los programas clientes deberán estar alojados en los directorios de sesión de cada uno de los usuarios que deseen establecer una comunicación con el servidor.
- El programa del servidor deberá permanecer siempre en ejecución y disponible para que los usuarios envíen los mensajes.
- Los programas clientes podrán ejecutarse siempre que el usuario lo requiera.
- El método de autenticación de los clientes será a través del inicio de sesión de cada uno de los usuarios.
- El método de autenticación del servidor será mediante el inicio de sesión del administrador en el equipo en donde se encuentre alojada la aplicación del servidor.
- La interfaz de acceso será manejada mediante la terminal de comandos de la estación de trabajo.
- La aplicación deberá ser desarrollada con herramientas de software libre.
- Los usuarios que deseen utilizar este medio de comunicación seguro, deberán tener acceso al sistema, mediante una cuenta de acceso previamente configurada.
- La sesión de acceso a los recursos, será otorgada por el administrador del sistema.

3.2.2 Requerimientos particulares

Requerimientos del certificado:

- El certificado del servidor o claves de acceso solamente se generarán una vez.
- El certificado debe ser avalado por alguna entidad certificadora, en donde los usuarios podrán verificar la clave pública.

- Los datos proporcionados para crear el certificado serán un nombre de usuario, un id de la sesión y una contraseña.
- Se utilizará la herramienta Secure Socket Layer (SSL) para la creación del certificado.
- Se crearán archivos que contengan las claves: pública y privada.

Requerimientos del cifrado del canal:

- La clave con la que será cifrada la comunicación (clave simple) deberá ser creada por el cliente con un algoritmo RSA de al menos 256 bits.
- La clave simple deberá ser cifrada en el cliente con el certificado del servidor antes de ser transmitida al servidor.
- El servidor deberá ser capaz de descifrar la clave simple mediante el uso del cifrado asimétrico.
- La comunicación completa será cifrada y descifrada por el cliente y el servidor con la clave simple.
- El cifrado/descifrado de los mensajes deberá realizarse en tiempo real.

Requerimientos de la comunicación:

- El servidor deberá ser capaz de recibir y enviar mensajes al cliente en tiempo real.
- El cliente deberá ser capaz de recibir y enviar mensajes al servidor en tiempo real.
- La comunicación deberá ser de la forma de un mensajero instantáneo.

Requerimientos de la aplicación:

- Los equipos en donde se encuentre ejecutando el cliente y el servidor deberán tener un sistema operativo Linux (ver capítulo 1.4).

- Las sesiones en donde se encuentren ejecutando el servidor y los clientes deberán contar con las librerías requeridas para su correcta ejecución (ver capítulo 1.4).
- Los archivos de los ejecutables del servidor y clientes deberán tener permisos de lectura y ejecución a nivel del sistema operativo.

3.3 RECOPILACIÓN Y ANÁLISIS DE LA INFORMACIÓN ACTUAL

La comunicación actual entre los usuarios y el servidor se realiza mediante las utilerías del sistema operativo, sin embargo, no proporcionan la confianza suficiente para transmitir la información que se requiere.

Los sistemas UNIX en modo comandos, cuentan con algunas herramientas para transmitir mensajes entre los usuarios que se encuentran conectados a la red.

El problema con estas herramientas es que son muy sencillas y además de que la transmisión de los mensajes se realiza en texto plano, un usuario puede transmitir información a cualquier otro dentro de la red sin solicitar su autorización.

Este tipo de herramientas con las que cuentas por defecto los sistemas UNIX no proporcionan a los usuarios la confianza de transmitir cualquier tipo de información y mucho menos información confidencial, debido a que no se puede garantizar que no podrá ser recuperada por cualquier otro usuario y aun peor, por algún intruso.

Y aunque los usuarios pueden acceder a ellas de manera inmediata, el costo de que alguna información confidencial sea recuperada por una persona indebida puede ser muy alto.

3.3.1 Utilería WRITE

El comando write permite a los usuarios enviar mensajes a otros usuarios en un ambiente de comandos en UNIX. Esto se realiza copiando las líneas desde la terminal de un usuario en las del otro.

Cuando un usuario ejecuta el comando write, el usuario al que se le escribe recibe un mensaje en donde se indica el usuario que lo envía, el servidor, la terminal y la hora.

Después de eso, todas las líneas siguientes que se escriban en la terminal serán copiadas en la terminal del usuario destino.

Para detener la comunicación se escribe un fin de archivo o el carácter de interrupción. El otro usuario verá un mensaje que indica el fin de archivo o en este caso el fin de la comunicación.

3.3.2 SSH (Secure Shell)

Funciona como un mecanismo de túnel que proporciona un acceso remoto a una terminal. SSH es un programa y protocolo que puede ser utilizado para ingresar a una computadora en una red. Proporciona autenticación y transmisión segura sobre canales vulnerables como internet.

Es recomendable utilizar SSH en vez de telnet, FTP, rlogin, rexec o rsh. El túnel que provee SSH se realiza entre dos computadoras, las cuales intercambian las claves de autenticación, después de haberse identificado, las cuales serán utilizadas durante la sesión para cifrar y descifrar los datos enviados. Como se muestra en la figura 3.3.1.

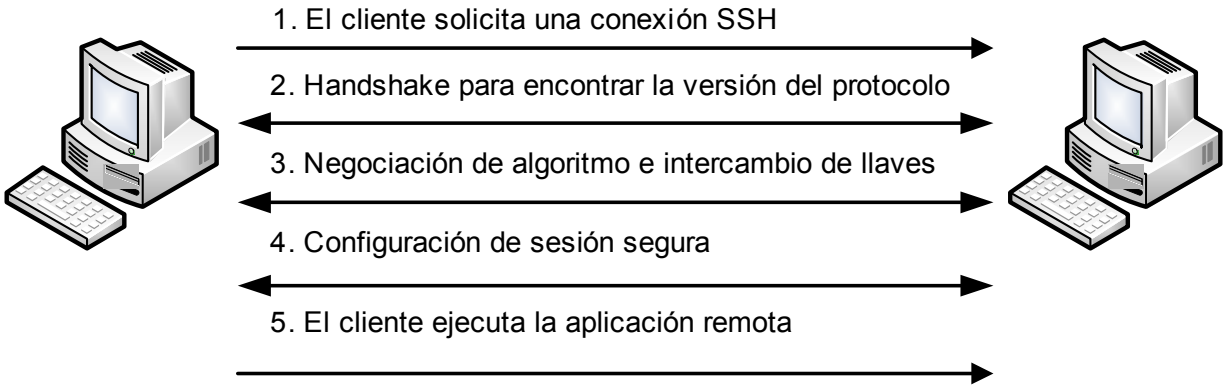


Figura 3.3.1 Pasos para establecer una conexión segura con SSH.

Una vez que las computadoras han establecido la comunicación y han enviado las claves, la comunicación puede enviarse con la confianza de que serán cifradas y que la integridad será protegida, como se muestra en la figura 3.3.2.

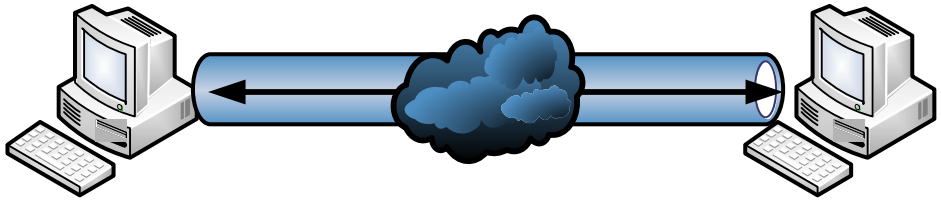


Figura 3.3.2. Canal seguro entre dos computadoras.

Existen también algunos métodos para establecer una conexión segura en una red de área local o entre dos puntos que se encuentra en dos redes diferentes.

3.3.3 IPSec (Seguridad IP)

IPSec es un conjunto de protocolos que fue diseñado para proporcionar a los sistemas de usuario final un método para autenticarse el uno con el otro y para proteger los datos en tránsito de que sean escuchados secretamente (eavesdropping) y otro tipo de ataques.

Es un estándar de seguridad que se ejecuta en la capa de red del modelo de referencia de interconexión de sistemas abiertos (OSI, por sus siglas en inglés open system interconnection) y provee seguridad en el protocolo de internet (IP) cifrando cada uno de los paquetes IP en un flujo de datos.

Debido a que el alto rendimiento es crucial en este nivel de comunicación, se basa en la criptografía simétrica (que es mucho más rápida que la asimétrica).

IPSec protege las comunicaciones de diferentes entornos como redes privadas, sitios corporativos y redes de área local; además de ser utilizado entre socios que se encuentran en diferentes redes y para aplicaciones de comercio electrónico, utilizando la criptografía.

Este protocolo fue diseñado para IPv4 y para IPv6.

Una conexión, en términos de IPSec, es llamada asociación segura (SA, security association por sus siglas en inglés), es de tipo simplex y tiene asociado un identificador de seguridad. Si se requiere una conexión en ambas direcciones, es necesario utilizar dos SA's. Los identificadores de seguridad son transportados en los paquetes de las conexiones seguras y son utilizados para buscar claves y otra información relevante, como se muestra en la figura 3.3.3.

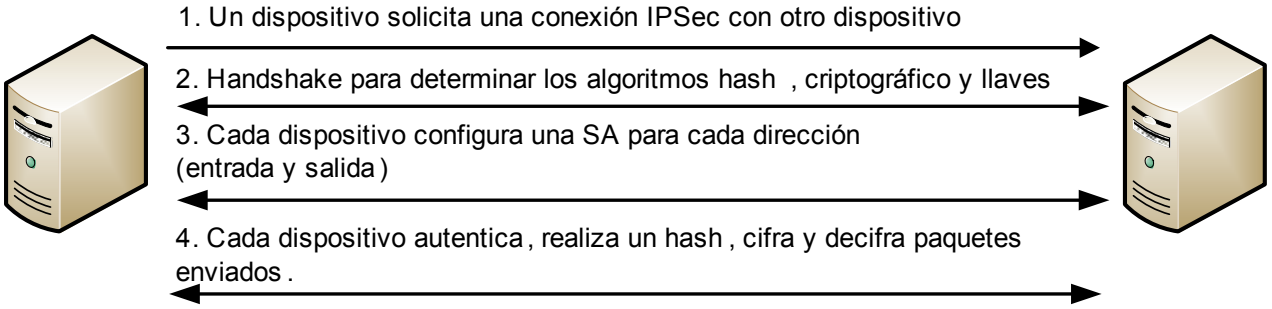


Figura 3.3.3. Pasos a seguir entre dos computadoras que usan IPSec.

IPsec puede utilizarse en dos modos:

- Modo de transporte, la cabecera de IPsec es insertada justo después del encabezado de IP. En el campo “protocolo” de la cabecera IP es cambiado para indicar que una cabecera IPsec sigue el encabezado de IP normal. El encabezado de IPsec contiene información de seguridad, principalmente el identificador SA, un nuevo número de secuencia y en algunas ocasiones la verificación de la integridad del campo de carga.
- Modo de túnel, todo el paquete IP es encapsulado en el cuerpo de un nuevo paquete IP con una cabecera completamente nueva. El modo de túnel es útil cuando el túnel termina en una localización diferente al destino final (por ejemplo un firewall).

3.3.4 Redes Privadas Virtuales (Virtual Private Network, VPN por sus siglas en inglés)

Las redes privadas virtuales son redes recubiertas que trabajan sobre redes públicas pero con muchas de las propiedades de las redes privadas, son llamadas virtuales debido a que son una ilusión.

Normalmente son diseñadas con firewalls, creando túneles a través de internet entre un par de sitios. Se utiliza IPSec para el túnel y con esto es posible enviar todo el tráfico con una sola asociación segura para la autenticación.

Los firewalls son los que se encargan de los parámetros de las SA’s (servicios, modos, algoritmos y claves). La mayoría de los firewalls están habilitados para trabajar sobre VPN’s.

Una vez que las SA's han sido establecidas, el tráfico puede enviarse a través de internet, Un paquete que viaja a través de un túnel de VPN no es más que un paquete ordinario ya que solamente se le ha agregado la cabecera de IPSec.

3.4 IDENTIFICACIÓN DE LOS POSIBLES MÓDULOS DE LA APLICACIÓN

A continuación se detallan los módulos que definirán el canal de comunicación cifrado, ver figura 3.4.1:

- Módulo de certificado.
- Módulo de clave simple.
- Módulo de clave de sesión (cliente).
- Módulo de clave de sesión (servidor).
- Módulo de transmisión y validación de certificado.
- Módulo de transmisión de la clave de sesión.
- Módulo de comunicaciones.
- Módulo de usuario.

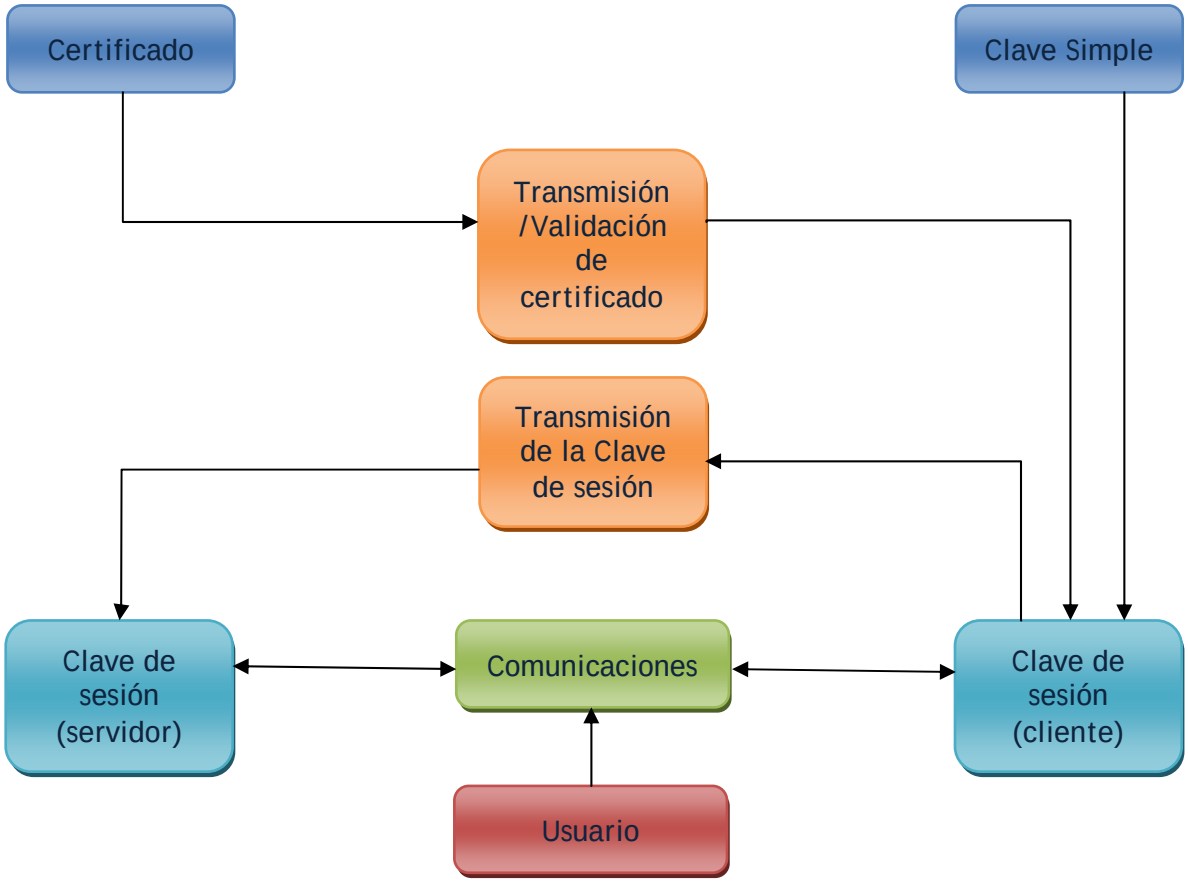


Figura 3.4.1. Módulos de la aplicación.

En la figura 3.4.1 puede observarse la interacción de los módulos de la aplicación.

3.4.1 Módulo de certificado

En el módulo del certificado se realizará la creación del certificado y su publicación en la entidad certificadora, ver figura 3.4.1.1.

Figura 3.4.1.1 Módulo de certificado.

Datos del usuario.

En éste módulo se definen los datos de creación del certificado, se ejecuta una aplicación que solicita los datos al usuario.

Creación del certificado.

Este módulo define la creación del certificado, mediante las herramientas de SSL.

Publicación del certificado.

Este módulo define la publicación de la clave pública del certificado mediante la entidad certificadora, para que pueda ser comparada y verificada.

3.4.2 Módulo de clave simple

En el módulo de la clave se realizará la creación de la clave simple para el cifrado de la comunicación en el canal, ver figura 3.4.2.1.

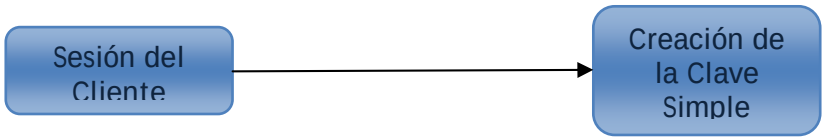


Figura 3.4.2.1 Módulo de la clave sesión del cliente.

Este módulo indica que en cada sesión del cliente, se generará una clave para cifrar la comunicación.

Creación de la clave simple.

Módulo que define la creación de la clave mediante un algoritmo RSA de 256 bits.

3.4.3 Módulo de la transmisión y validación de certificado

Módulo que define la metodología para la transmisión del certificado del servidor al cliente, además de la validación del certificado por parte del cliente en la entidad certificadora, ver figura 3.4.3.1.

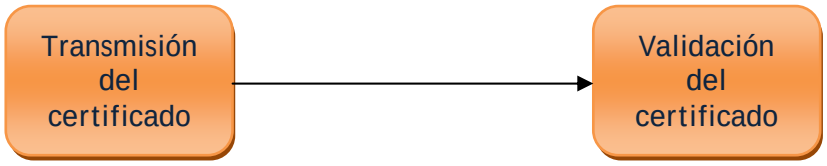


Figura 3.4.3.1 Módulo de la transmisión y validación del certificado.

Transmisión del certificado.

Este módulo define cómo se realizará la transmisión de la clave pública del certificado del servidor al cliente.

Validación del certificado.

Módulo que indica cómo se realizará la comparación entre la clave pública recibida desde el servidor con la clave pública avalada por la entidad certificadora.

3.4.4 Módulo de clave de sesión (cliente)

En este módulo se define la metodología a seguir para el cifrado de la clave con la que se asegura la comunicación, antes de su transmisión por el canal, ver figura 3.4.4.1.

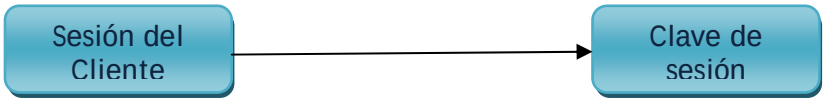


Figura 3.4.4.1 Módulo de clave de sesión (cliente).

Sesión del cliente.

Este módulo indica que en cada sesión del cliente, se cifrará la clave generada para su transmisión del cliente al servidor.

Clave de sesión.

Se define la metodología a seguir para obtener la clave de sesión en base a la clave pública del servidor y la clave simple del cliente, utilizando el algoritmo de clave asimétrica.

3.4.5 Módulo de clave de sesión (servidor)

En este módulo se define la metodología a seguir para el descifrado de la clave con la que se asegura la comunicación, después de su transmisión por el canal, ver figura 3.4.5.1.

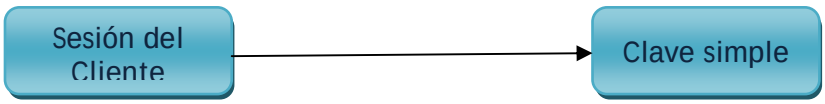


Figura 3.4.5.1 Módulo de clave de sesión (servidor).

Sesión del cliente.

Este módulo indica que en cada sesión del cliente, se descifrará la clave generada una vez que se ha transmitido del cliente al servidor.

Clave simple.

Se define la metodología a seguir para obtener la clave simple con base en la clave privada del servidor y la clave de sesión enviada por el cliente, utilizando el algoritmo de clave asimétrica.

3.4.6 Módulo transmisión de la clave de sesión.

Se define la metodología utilizada para enviar la clave de sesión desde el cliente hacia el servidor, ver figura 3.4.6.1.

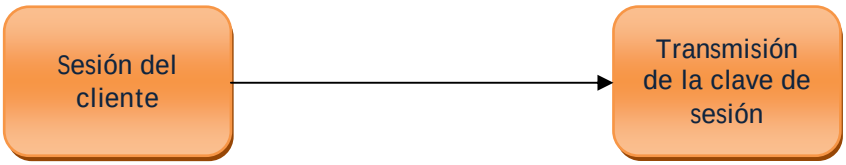


Figura 3.4.6.1 Módulo transmisión de la clave de sesión.

Sesión del cliente.

Este módulo indica que en cada sesión del cliente, se transmitirá la clave de sesión generada.

Transmisión de la clave de sesión.

Este módulo indica la metodología para transmitir la clave de sesión generada.

3.4.7 Módulo comunicaciones

Define cómo se gestiona la comunicación entre el cliente y servidor para asegurar que sea bidireccional y en tiempo real, ver figura 3.4.7.1.

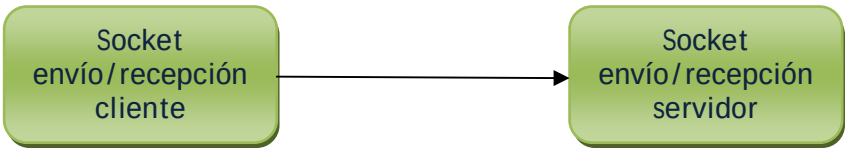


Figura 3.4.7.1 Módulo comunicaciones.

Socket envío/recepción cliente.

Metodología para la creación del socket bidireccional para la transmisión y recepción de datos desde el cliente.

Socket envío/recepción servidor.

Metodología para la creación del socket bidireccional para la transmisión y recepción de datos desde el servidor.

3.4.8 Módulo usuario

Define las características de cómo se realizará la interacción entre la sesión del sistema operativo del usuario y la aplicación, ver figura 3.4.8.1.

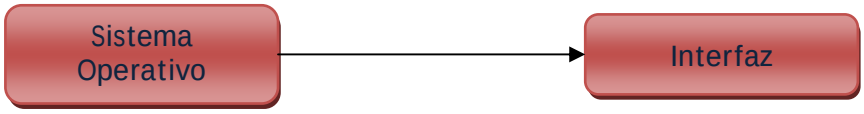


Figura 3.4.8.1 Módulo usuario.

Sistema operativo.

Define los requerimientos y características del sistema operativo para que la aplicación funcione correctamente.

Interfaz.

Define cómo el usuario interactuará con la aplicación dentro de su sesión de usuario del sistema operativo.

3.5 SELECCIÓN DE LA ARQUITECTURA DE REDES CLIENTE-SERVIDOR

Como se comentó en el capítulo 2, la arquitectura cliente-servidor proporciona la capacidad a un sistema de aplicaciones, para ser dividido a través de múltiples plataformas que pueden variar en el sistema operativo y el hardware, esto lo hace ser más seguro, si se cuenta con protecciones en las diversas capas.

En el proyecto se realiza una comunicación a través de una red TCP/IP, en la cual participan dos procesos de sistemas diferentes que intercambian datos a través de la red.

Cada uno de estos procesos (un proceso para el cliente y un proceso para el servidor) se encuentran en un extremo de la comunicación (socket).

La arquitectura de redes cliente-servidor a emplear será de dos capas, ya que como se comentó en el capítulo 2, se cuenta con un cliente el cual mantiene la lógica de la aplicación, y adicional se cuenta con un servidor que administra las peticiones a los componentes de las claves. Se optó por utilizar esta solución debido a que se requiere poco volumen de procesamiento de datos, en caso de requerirse mayor volumen se debe considerar utilizar un módulo de seguridad criptográfico en hardware (i.e. HSM por sus siglas en inglés de Hardware Security Module).

3.6 SELECCIÓN DEL TIPO DE CIFRADO

Ningún algoritmo para cifrar información, es ideal para todos los requerimientos, para elegir el algoritmo que se adapte a las necesidades de la aplicación se debe de tomar en cuenta lo siguiente:

- El cifrado más seguro suele consumir más recursos del CPU que un cifrado menos seguro.
- Las claves largas suelen producir un cifrado más seguro que las claves cortas.
- El cifrado asimétrico es más seguro que el simétrico con la misma longitud de clave, pero es relativamente más lento.
- Las contraseñas largas y complejas son más seguras que las contraseñas cortas.

- Si se requiere cifrar una gran cantidad de datos, debe cifrar los datos con una clave simétrica y cifrar la clave simétrica con una clave asimétrica.
- Si utiliza compresión, se deben de comprimir los datos antes de cifrarlos ya que los datos cifrados no se pueden comprimir, pero los datos comprimidos sí se pueden cifrar.

Tomando en cuenta lo anterior y analizando nuestros requerimientos, la mejor solución para cifrar varios datos que serán intercambiados en la comunicación se realizará mediante el cifrado simétrico para que el proceso sea más rápido. Para realizar el intercambio de la clave simétrica se cifrará esta clave con una clave asimétrica.

La distribución de claves siempre ha sido la parte débil de la mayoría de los criptosistemas. La criptografía de clave pública hace posible que las personas que no comparten una clave común (simétrica) se comuniquen en un canal seguro. Por esta razón se utiliza la combinación de criptografía asimétrica con criptografía simétrica.

El problema radica que si Daniela y Edgar no se conocen entre sí, se requiere un tercero confiable donde se coloquen las claves públicas, es decir una entidad certificadora confiable en un sitio web. Esto no funcionaría correctamente si Eduardo intercepta la solicitud de Daniela y responde con una página de inicio falsa como la de Edgar junto con la clave pública de Eduardo.

El algoritmo simétrico AES, por sus siglas en inglés de Advanced Encryption Standard, es también conocido como cifrado de Rijndael por la derivación de los apellidos de sus autores Rijmen y Daemen. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología de EUA (NIST por sus siglas en inglés)

como FIPS PUB 197 de los Estados Unidos. Desde el 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica.

Los algoritmos criptográficos de clave simétrica más conocidos son el 3DES y AES, pero se han diseñado otros cifrados de clave simétricas que están incluidos en varios productos, vea tabla 4.

CIFRADO	AUTOR	LONGITUD DE CLAVE	COMENTARIOS
Rijndael	Rijmen y Daemen	128-256 bits	La mejor opción.
3DES	IBM	168 bits	Segunda mejor opción.
Twofish	Bruce Schneier	128-256 bits	Muy robusto; ampliamente utilizado.
Serpent	Anderson, Biham, Knudsen	128-256 bits	Muy robusto.
IDEA	Massey y Xuejia	128 bits	Bueno pero patentado.
RC5	Ronald Rivest	128-256 bits	Bueno pero patentado.
RC4	Ronald Rivest	1-2048 bits	Algunas claves son débiles por lo que hay que tener precaución.
Blowfish	Bruce Schneier	1-448 bits	Antiguo y lento.
DES	IBM	56 bits	Débil para utilizarlo actualmente.

Tabla 4. Algoritmos criptográficos de clave simétrica.

AES es rápido tanto en software como en hardware, es relativamente fácil de implementar y requiere poca memoria. Como nuevo estándar de cifrado se está utilizando actualmente a gran escala.

Por lo que para el cifrado simétrico se utilizará AES de 256 en modo ECB (Electronic Code Book por sus siglas en inglés) y para la distribución de la clave simétrica utilizada utilizamos RSA a 1024bits.

El algoritmo asimétrico de RSA (Rivest, Shamir y Adleman) se basa en ciertos principios de la teoría de números, a continuación se muestra de forma resumida. Para los detalles se puede consultar la descripción del estándar, en el documento "A method for obtaining digital signatures and public-key cryptosystems" escrito

por Rivest, Shamir y Adleman, en donde se describen las características matemáticas del algoritmo.

- Se seleccionan dos números primos grandes, *p* y *q* los cuales generalmente son de 1024 bits.
- Se calcula $n = p * q$ y $z = (p-1) * (q-1)$.
- Se selecciona un número primo *d*, con respecto a *z*.
- Encontrar *e* tal que $e * d = 1 \bmod z$.

Si calculamos estos parámetros por adelantado, se puede realizar el cifrado, para esto dividimos el texto claro en bloques, conocido como una cadena de bits, para que cada mensaje de texto claro, *P*, caiga en el intervalo $0 \leq P \leq n$. Esto puede realizarse si se agrupa el texto claro en bloques de *k* bits, donde *k* es el entero más grande para el que $2^k < m$ es verdadero.

Para cifrar un mensaje, *P*, se calcula $C = P^e \pmod n$. Para descifrar *C*, se calcula $P = C^d \pmod n$. Las funciones de cifrado y descifrado son inversas, ya que para cifrar se requiere *e* y *n*, para descifrar se requiere *d* y *n*. Por lo tanto, el cifrado asimétrico consiste en el par (e, n) y la clave privada consiste en (d, n).

En el estándar “Public-key cryptography standards (PKCS) PKCS #1: RSA cryptography standard” se proveen recomendaciones para la implementación de criptografía de claves públicas basadas en el algoritmo RSA, dichas recomendaciones son realizadas para aplicaciones generales de cómputo y de sistemas de comunicación.

CAPÍTULO 4

ELECCIÓN DE LA METODOLOGÍA DE DESARROLLO

CAPÍTULO 4

4.1 ELECCIÓN DE LA METODOLOGÍA DE DESARROLLO

El modelo del proceso o paradigma de ingeniería de software que se utiliza para resolver problemas de ingeniería de software, depende en gran medida de la naturaleza del proyecto y de la aplicación así como los métodos y las herramientas a utilizarse, además de los controles y entregables que se requieren.

Para nuestro caso, dada la naturaleza tan específica de nuestra aplicación se utilizará el modelo lineal secuencial como estrategia de desarrollo.

4.1.1 Modelo Lineal Secuencial

Este modelo es también conocido como ciclo de vida básico o modelo de cascada. Para el desarrollo de un sistema sugiere un enfoque sistemático, secuencial que comienza en un nivel de sistemas y progresa con el análisis, diseño, programación, pruebas y mantenimiento, como se muestra en la figura 4.1.1.1.

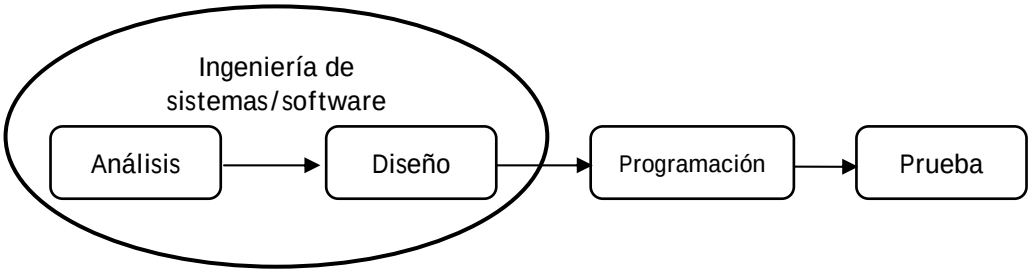


Figura 4.1.1.1 Modelo lineal secuencial.

El modelo lineal secuencial comprende las siguientes actividades:

- Ingeniería y modelado de Sistemas/Información

Se refiere a la recolección de los requisitos de todos los elementos del sistema. Cuando el software interactúa con otros elementos como hardware o personas, se le debe asignar un subgrupo de estos requisitos.
En esta etapa es fundamental la presencia del cliente que documenta y revisa los requisitos.

La ingeniería de sistemas obtiene los requisitos del sistema con una pequeña parte de análisis y diseño. Mientras que la ingeniería de la información obtiene los requisitos estratégicos del área de negocio.

- Análisis de los requisitos del software

  En esta actividad se reúnen los requisitos específicamente del software, se deben conocer el dominio de la información del software así como la función requerida, comportamiento, rendimiento e interconexión.

- Diseño

  Es una etapa dirigida hacia la estructura de datos, la arquitectura del software, las representaciones de la interfaz y algoritmo. En forma general se hace un esbozo de lo solicitado y se documenta haciéndose parte del software. Se traducen los requisitos en una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

- Generación de código

  Es la etapa en la cual se traduce el diseño para que sea comprensible por la máquina. Esta etapa va a depender estrechamente de lo detallado del diseño.

- Pruebas

  Esta etapa se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

- Mantenimiento

  Debido a que el programa sufrirá cambios después de haber sido entregado ya que puede tener errores, puede necesitar eventualmente acoplarse a los cambios en su entorno o puede que el cliente requiera mejoras funcionales o de rendimiento. Esto quiere decir que no se rehace el programa, sino que sobre la base de uno ya existente se realizan algunos cambios.

El modelo lineal secuencial es el paradigma de desarrollo de software más antiguo y más utilizado en la ingeniería del software, sin embargo, esto no ha impedido que se haya creado una desconfianza alrededor de él ya que a menudo es difícil que el cliente exponga exactamente todos los requisitos.

4.2 DIAGRAMACIÓN

Diagrama de flujo y procesos

Una vez que se ha analizado la aplicación y que se han definido claramente los diferentes módulos que interactúan en la aplicación, se procede a determinar los elementos que se reciben del exterior y la información que se genera a través de la aplicación.

En esta sección se realizará una representación gráfica de los elementos que integran la aplicación que permitirá definir entradas, procedimientos y salidas de la información, permitiendo de esta manera comprender plenamente los procedimientos existentes.

Por otra parte el diagrama de proceso de datos representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida de la aplicación. Además de que proporciona un mecanismo para el modelado funcional así como el modelado del flujo de información.

4.2.1 Diagrama de flujo

El diagrama de flujo es un esquema para representar gráficamente un algoritmo. Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación.
A continuación se muestra el diagrama general del algoritmo de la aplicación, indicando la secuencia de los procesos principales y su interacción, ver figuras 4.2.1.1 y 4.2.1.2.

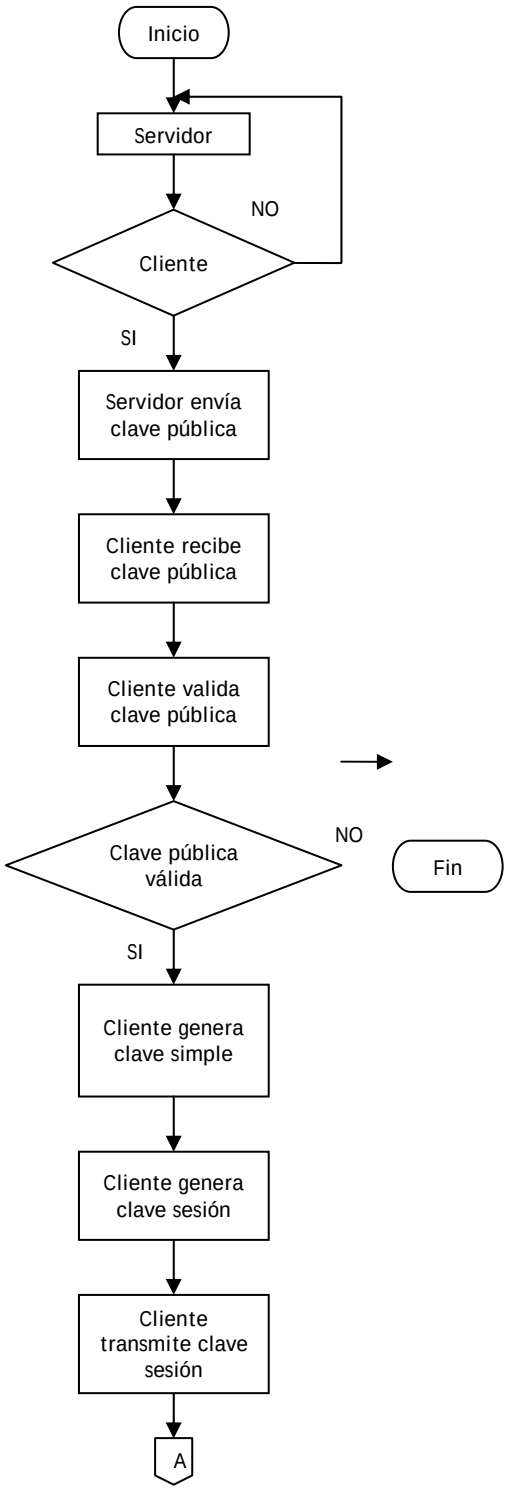


Figura 4.2.1.1 Algoritmo general de la aplicación.

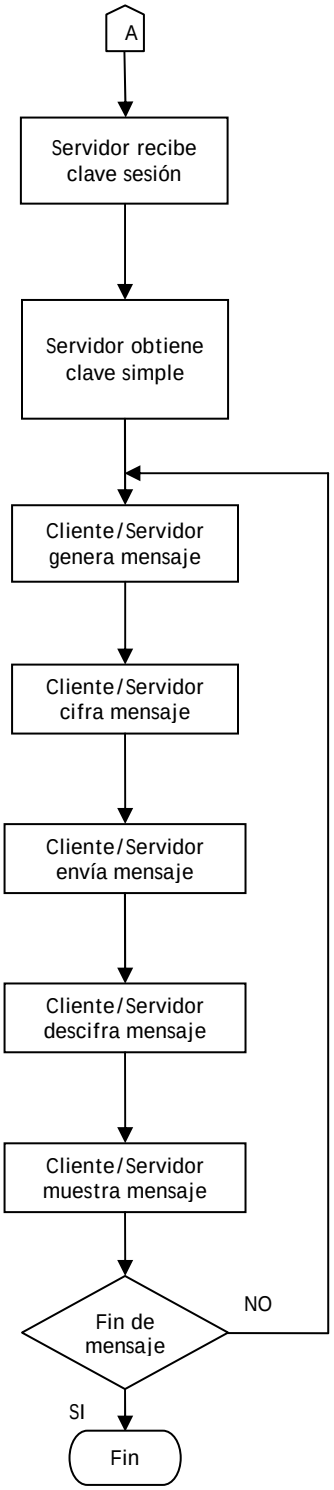


Figura 4.2.1.2 Continuación algoritmo general de la aplicación.

El siguiente diagrama muestra el algoritmo del proceso de creación del certificado que identificará al servidor, ver figura 4.2.1.3.

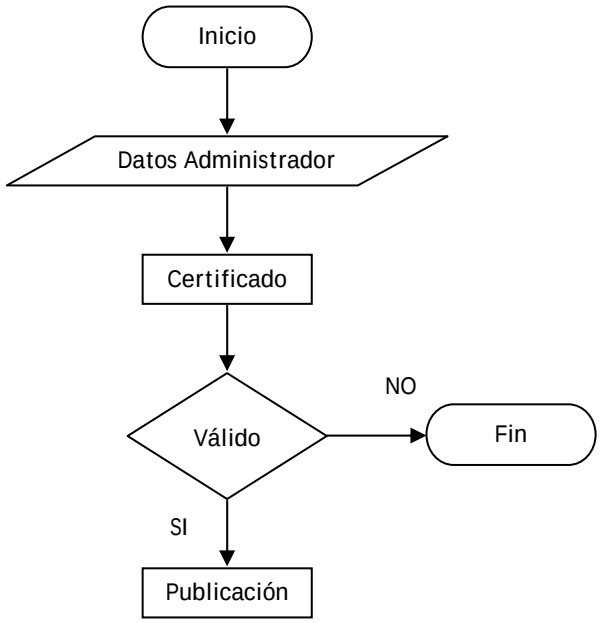
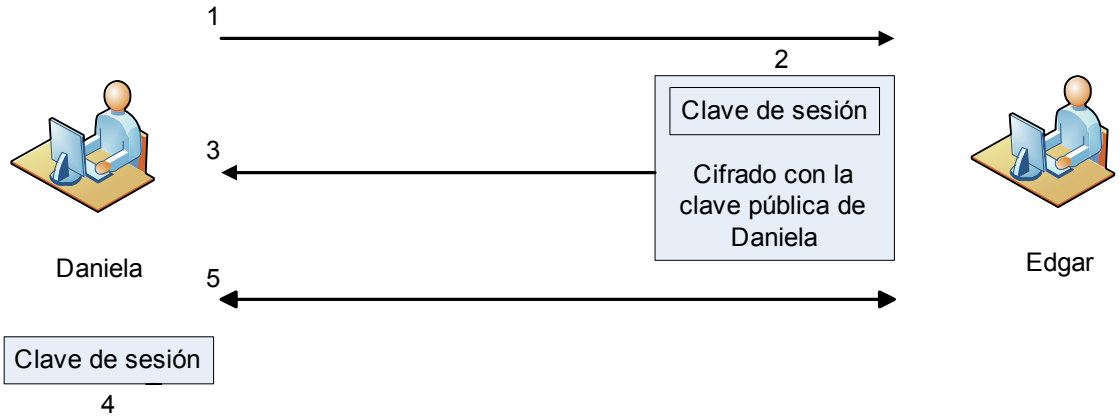


Figura 4.2.1.3 Algoritmo creación de certificado.

4.2.2 Diagrama de procesos

A continuación se muestran los diagramas de los procesos involucrados, en donde se detalla el flujo de la información y las transformaciones de los datos desde la entrada de cada proceso hasta la salida.

En el diagrama 4.2.2.1 puede observarse el proceso general del cifrado del mensaje.



1. Daniela manda a Edgar su clave pública.
2. Edgar genera una clave de sesión aleatoria y la cifra usando la clave pública de Daniela .
3. Edgar envía la clave de sesión cifrada con la clave pública de Daniela, a Daniela.
4. Daniela descifra el mensaje de Edgar con su clave privada y obtiene una copia de la clave de sesión.
5. Daniela y Edgar usan la clave de sesión para cifrar y descifrar los mensajes del otro .

Figura 4.2.2.1. Cifrado del mensaje.

Diagrama de creación del certificado

En este proceso se realiza la creación del certificado que identificará al servidor ante cualquier cliente a partir de los datos de identificación introducidos por el administrador, ver figura 4.2.2.2.

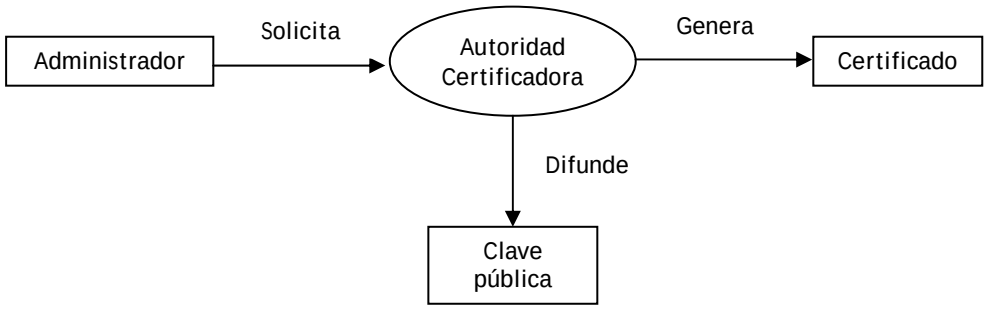


Figura 4.2.2.2 Diagrama creación certificado.

Diagrama de inicio del servidor

El proceso que indica la inicialización del servicio del servidor para estar disponible ante cualquier solicitud de un cliente, ver figura 4.2.2.3.

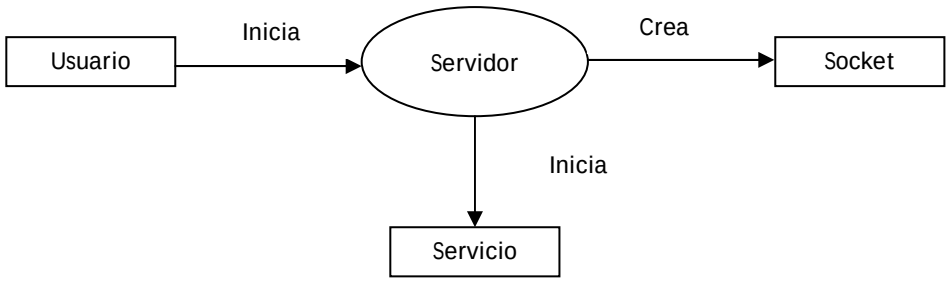


Figura 4.2.2.3 Diagrama inicio aplicación servidor.

Diagrama de inicio del cliente

El proceso que indica la inicialización de la solicitud de un cliente para utilizar los servicios ofrecidos por el servidor, ver figura 4.2.2.4.

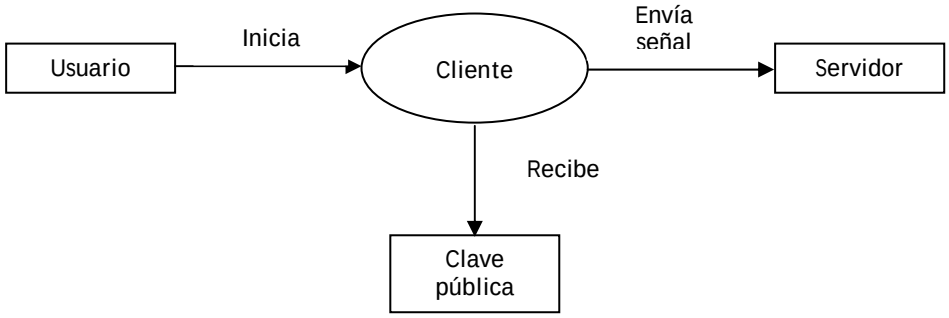


Figura 4.2.2.4 Diagrama inicio aplicación cliente.

Diagrama de verificación de la clave pública

Durante este proceso se realiza la validación de la identidad del servidor a través de la clave pública de su certificado, ver figura 4.2.2.5.

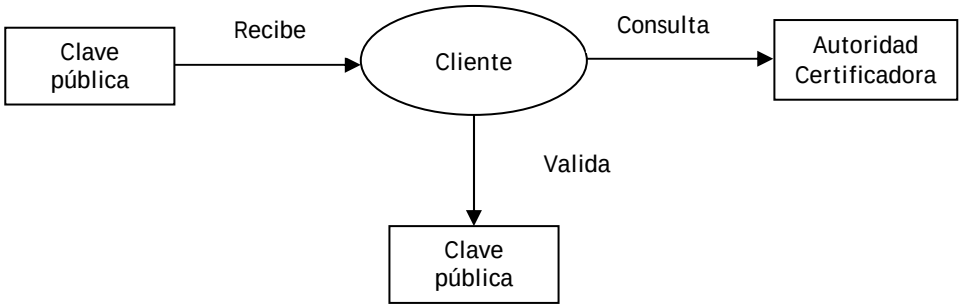


Figura 4.2.2.5 Diagrama validación de clave pública del certificado.

Diagrama de creación de la clave simple

En este proceso se indica cómo el cliente, después de validar la clave pública del servidor, genera la clave simple, ver figura 4.2.2.6.

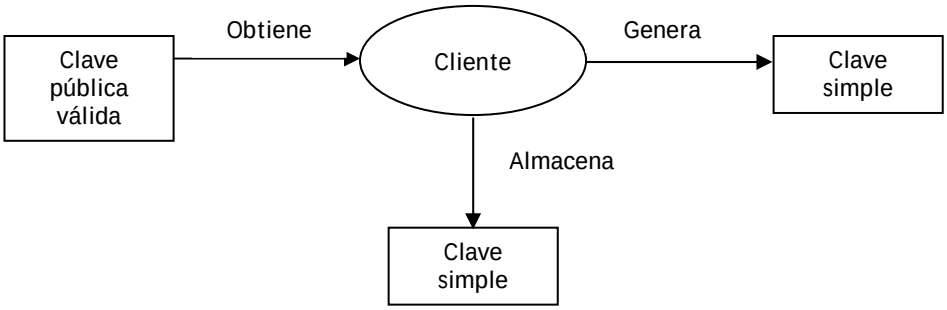


Figura 4.2.2.6 Diagrama creación clave simple.

Diagrama de creación de la clave de sesión

Durante este proceso el cliente obtiene la clave de sesión cifrando la clave simple con la clave pública, ver figura 4.2.2.7.

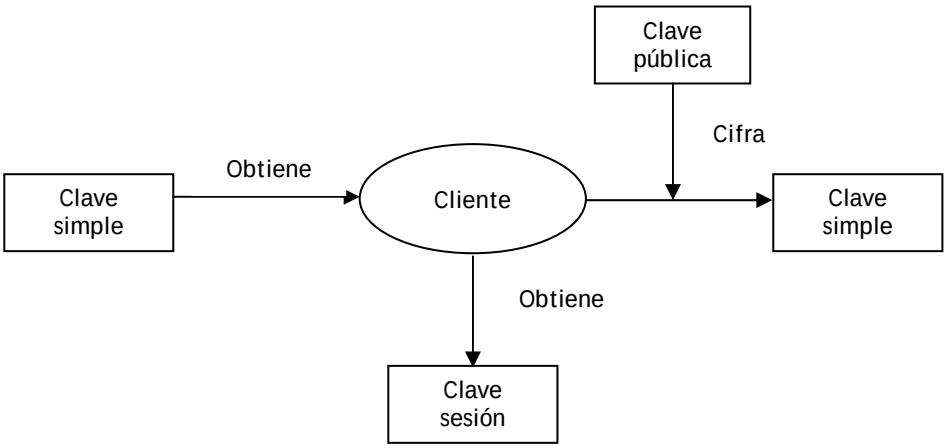


Figura 4.2.2.7 Diagrama clave simple.

Diagrama de transmisión de la clave de sesión

En este proceso el cliente envía la clave de sesión al servidor una vez que ha sido creada, ver figura 4.2.2.8.

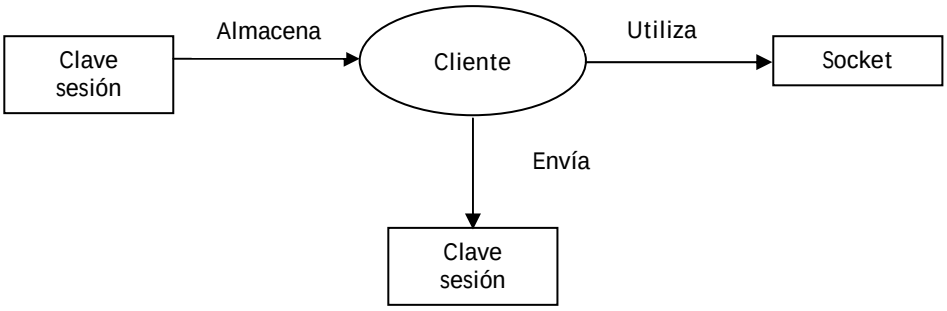


Figura 4.2.2.8 Diagrama envío clave de sesión.

Diagrama de recepción de la clave de sesión

En este proceso el servidor recibe la clave de sesión enviada por el cliente, ver figura 4.2.2.9.

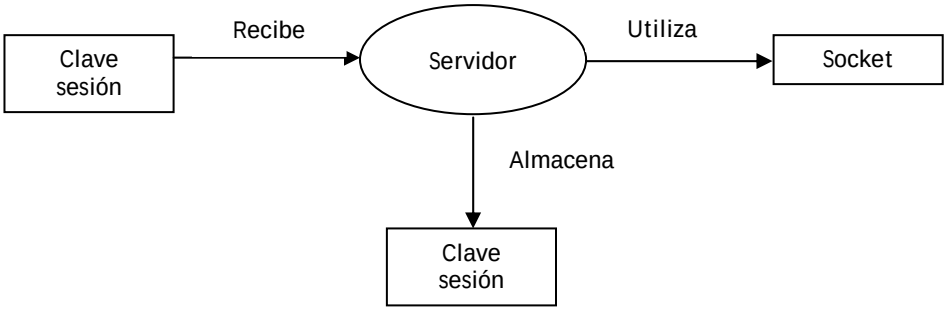


Figura 4.2.2.9 Diagrama recepción clave de sesión.

Diagrama de obtención de la clave de simple

Durante este proceso el servidor descifra la clave de sesión con la clave privada de su certificado para obtener la clave simple, ver figura 4.2.2.10.

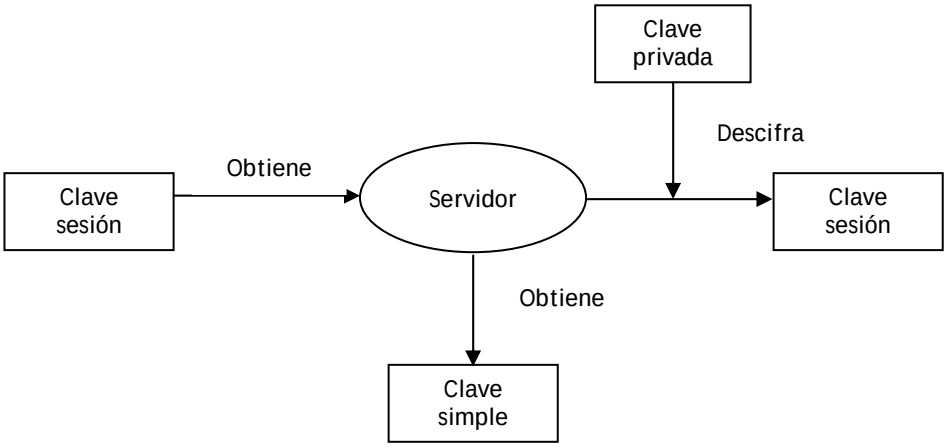


Figura 4.2.2.10 Diagrama descifrado.

Diagrama de envío de mensajes

Este proceso detalla el tratamiento de los mensajes durante su envío, ver figura 4.2.2.11.

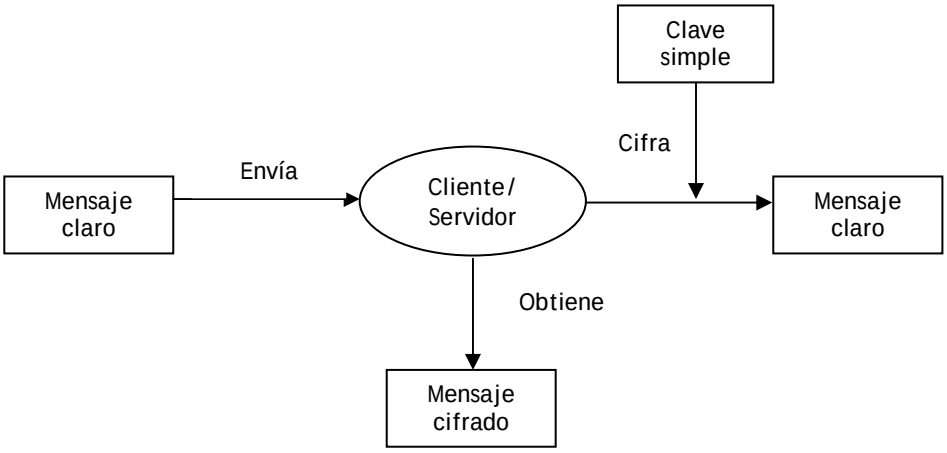


Figura 4.2.2.11 Diagrama envío de mensajes.

Diagrama de recepción de mensajes

Este proceso detalla el tratamiento de los mensajes durante su recepción, ver figura 4.2.2.12.

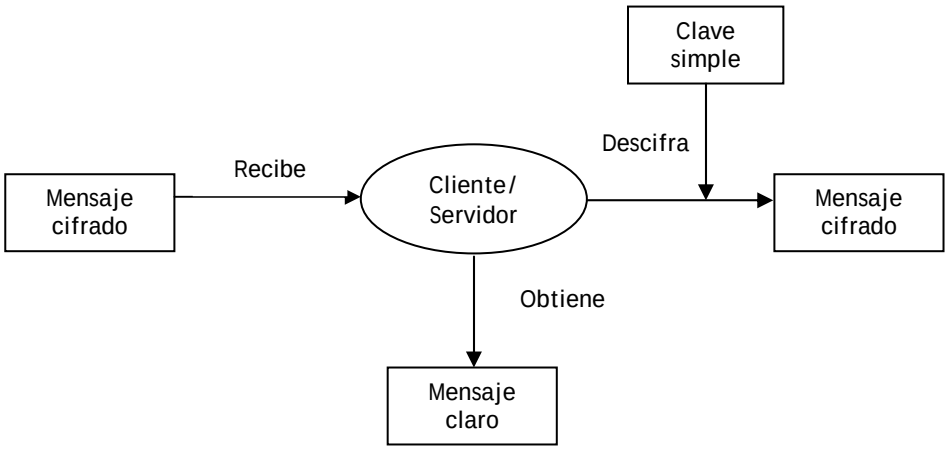


Figura 4.2.2.12. Diagrama de recepción de mensajes.

4.2.3 Diagrama de arquitectura

El diagrama de la arquitectura cliente-servidor empleado en la aplicación se muestra en la figura 4.2.3.1.

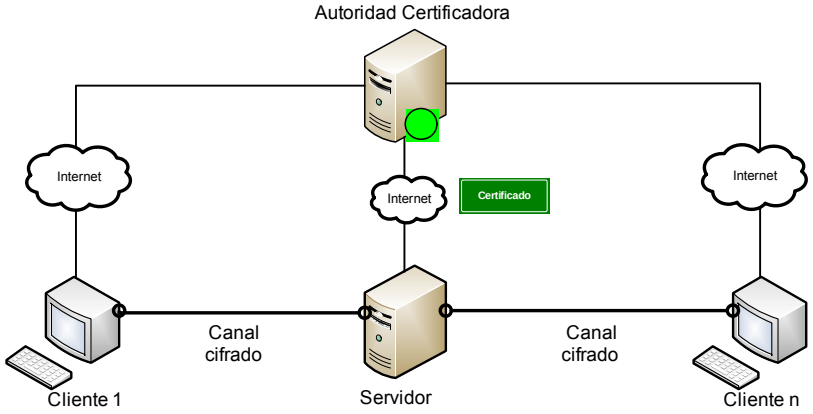


Figura 4.2.3.1 Arquitectura cliente-servidor.

4.3 DISEÑO Y CONSTRUCCIÓN DE LA APLICACIÓN

El canal de comunicación cifrada que permite enviar mensajes entre dos sistemas interconectados en una red, es diseñada con herramientas open source para reducir costos.

En general la aplicación fue construida en lenguaje de programación perl utilizando algunos módulos adicionales para la generación, cifrado y descifrado de las claves, como OpenSSL y RSA.

Se utiliza un cifrado simétrico para la transmisión de los mensajes en tiempo real y un cifrado asimétrico para la transmisión segura de la clave de cifrado de la comunicación, además de utilizar la infraestructura de clave pública para el manejo y validación del certificado del servidor.

La clave simétrica utilizada para cifrar la comunicación es creada por la aplicación y es aleatoria de 256 bits, utilizando el algoritmo RSA. Los mensajes transmitidos durante la comunicación son cifrados con un algoritmo AES de 256 bits, utilizando esta clave.

La clave asimétrica utilizada para cifrar la transmisión de la clave de sesión, es creada por un módulo independiente a la aplicación que genera un certificado, del cual es distribuida la clave pública por medio de la página web de una entidad certificadora.

4.3.1 Generación certificado (clave privada y clave pública)

La creación de la clave privada se realiza principalmente mediante la función “genrsa” del conjunto de herramientas de OpenSSL.

Función:
genrsa – Comando para generar claves privadas de RSA.

Sinopsis:
openssl genrsa [-out filename] [-passout arg] [-des] [-des3] [-idea] [-f4] [-3] [-rand file(s)] [-engine id] [numbits]

En nuestro caso, las opciones utilizadas son:

```
system("openssl genrsa -des3 -out key.prv 1024");
```

-des3: cifrar la clave con el algoritmo triple DES, utilizando la contraseña introducida por el usuario.

-out filename: indica el nombre del archivo en donde se almacenará la salida del comando.

numbits: el tamaño de la clave privada a generar, en nuestro caso 1024 bits.
Ejemplo de la clave privada generada, ver figura 4.3.1.1.

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,2ECCCC477DC9AA3F

ULgGyCgiiMspxNd3I3IczGUPLTWzKiIY5nQv0HwBglQKpCn6+UKX/XhzABfnQr+J
79EDRFG8QGlb5OJ4vtFA6dYZrwOvJYHYkdgjaWkzDFPZIexnlfxrip/SaO+LB1Z/
gj/Y2lEwQE+baqg85sE54FOEGTQeLAoqQSRhJ7gMff1GAN8XNHncFQED5F6mjG7W
b+nHEKb+2mQHAnlPtsLk2OnW/B7GxStK0rclI0RwZLs4SBi7igCmUWCv2N31Kngm
xFTImXQMsdkNCxwQMIIX5G8OjPif8xJqwCzjIsto9BUbtQ0NnE4gIDRlLWAuUBih
8+jTYJhId/J9IgC4w44ZEsbqxl06RN1+sukEiKadOzlfqb3VBLqxmg3SgN0Ye6wt
lRgqfZ9R5lghB0Y507/PSJoRy3Xh5ybP9sejDVwglKAnR/uaOgqaGrf9nN37PzCA
MniR+GUqb2wBlFu0NqQuVSmmYH7tSDztZppuxxcpvjGdzkr807hsKMaXyuXcIsoh
89llto49stSr7t3JCvsU6zmiYpqsGmdtJ70bwEVO3m3wLGudDLi2FZyoaor53mU0
KUUMtK+pSPxaL7+T99plmn3nIRWSHsVV5aGuWlAGg6FKalxMDN64gl8Ssr6iOrG6
iqln3dlQyWZ0bNPUxK19wnN7Bw45CZtAk7ZbNEACasrCzeqAo0klwiOMqZew2D5k
3owulPBaWlHI74/lsqore5yqK9l5blZkpckqZulEaXvXiFVYi8Ek2uaWOz0+0nz9
+8zHL3eFmmGP5i3fuVeLv4gd89mbmYceyN57dNRMsNjhFoMdtOuZhg==
-----END RSA PRIVATE KEY-----
```

Figura 4.3.1.1 Clave privada RSA 1024 bits

La generación de la clave pública se realiza mediante el manejo y análisis de los componentes de la clave privada, por lo que es necesario obtener los datos que integran a la clave privada.

Función:
rsa – Herramienta de procesamiento de claves RSA.

Sinopsis:
openssl rsa [-inform PEM|NET|DER] [-outform PEM|NET|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-sgckey] [-des] [-des3] [-idea] [-text] [-noout] [-modulus] [-check] [-pubin] [-pubout] [-engine id]

En nuestro caso, las opciones utilizadas son:

```
system("openssl rsa -in key.prv -text > text.pvr ");
```

-in: Especifica el nombre del archivo de entrada desde donde leerá la clave, si la clave está cifrada, solicita la contraseña.

-text: imprime los componentes de la clave en un archivo de texto plano además de la versión codificada.

Ejemplo de los componentes de la clave privada, ver figura 4.3.1.2 y 4.3.1.3.

```
Enter pass phrase for key.prv:
Private-Key: (1024 bit)
modulus:
    00:d6:00:d6:6c:0c:7c:2c:6d:78:a8:25:79:0c:b5:
    fe:bd:80:61:dc:02:ef:4e:8a:78:df:6f:0e:4c:79:
    e0:70:ab:82:03:f4:ad:75:83:b3:f3:88:3b:b7:80:
    ae:10:47:9b:aa:0f:95:f5:fc:e6:fa:88:5e:e2:9c:
    5e:02:df:f7:bc:10:eb:87:26:0d:19:9a:02:e4:6d:
    fa:07:a1:44:9f:93:a0:ec:ed:a0:9b:ea:4c:81:ab:
    a4:73:33:05:77:3c:16:0d:d2:1b:40:a9:78:ea:55:
    d3:5e:6d:79:3b:5b:ca:0a:8b:e0:71:3b:91:91:42:
    81:29:7d:4e:53:9e:35:72:f9
publicExponent: 65537 (0x10001)
privateExponent:
    0a:09:ac:05:11:68:d9:a4:a0:de:32:33:c5:56:dc:
    06:03:79:93:9a:47:a3:45:77:79:f3:79:96:38:b9:
    40:4f:ab:1f:88:60:82:7c:94:fa:3d:4f:9f:c1:d7:
    68:48:13:93:1c:7c:0a:37:bc:95:cf:c3:b3:99:66:
    07:78:7d:e4:2c:57:85:51:51:18:a0:b3:4e:81:3a:
    03:bd:9c:21:53:21:f1:c3:16:5c:db:a5:e4:fc:17:
    29:66:1d:ca:1b:85:d6:dc:47:26:27:96:31:1d:7e:
    74:be:db:82:b9:af:16:0b:b1:f6:57:21:1e:12:53:
    0e:73:35:f8:08:f6:ba:a1
prime1:
    00:fd:bf:e9:2c:95:9e:ca:1d:3a:e1:43:61:a6:f9:
    8e:6a:83:54:fd:ce:a3:03:26:a6:9e:07:b3:c3:a6:
    16:62:6c:b5:b1:a6:ca:74:6a:8c:84:06:53:d8:0e:
    9e:19:08:d3:68:6d:af:c2:c6:61:70:20:a0:c0:e6:
    9b:f4:48:fe:ed
prime2:
    00:d7:e6:b0:b9:ce:98:24:ae:43:bc:f8:db:a4:06:
    dd:ec:0c:b2:55:69:0b:8e:79:ea:36:12:e1:4e:76:
    2d:9b:8a:2a:7a:e5:2f:86:79:67:ac:31:ca:04:28:
```

Figura 4.3.1.2 Componentes para generar clave pública RSA 1024 bits.

```
    06:97:b7:13:ed:fb:7f:41:df:e8:14:e8:0c:4b:71:
    9b:70:10:76:bd
exponent1:
    3d:1e:01:22:8c:f3:0f:09:55:4a:36:79:89:c7:27:
    e7:3d:b4:b5:e1:14:60:48:e9:ee:bc:3a:2f:10:8c:
    e9:ad:f7:61:4f:a4:09:c0:34:12:04:98:23:3c:fa:
    15:5b:8b:d7:e6:3b:64:35:75:f8:94:d3:43:43:08:
    23:21:85:cd
exponent2:
    19:0e:86:f8:29:87:f2:c9:de:c6:b1:c3:1b:f8:48:
    0d:3b:33:ff:41:9f:bb:bb:5e:79:44:1b:ad:f5:7d:
    39:8e:01:7c:1d:d9:34:2c:26:e2:0f:2d:38:ea:44:
    49:9b:4f:ce:d0:df:66:0f:cc:69:90:bc:cc:8c:b4:
    d3:7e:b5:a9
coefficient:
    00:c2:98:d6:82:fa:83:26:18:4a:6b:14:c9:c2:e6:
    e9:a2:e4:a6:4f:9f:47:24:a2:f1:e5:c8:ef:fe:59:
    76:c1:af:51:10:24:e8:08:1a:cd:98:1e:2f:11:34:
    9b:3b:25:91:5f:1f:33:4a:f4:13:7f:50:5d:3b:df:
    46:b6:0e:3e:1f
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDWANZsDHwsbXioJXkMtf69gGHcAu9Oinjfbw5MeeBwq4ID9K11
g7PziDu3gK4QR5uqD5X1/Ob6iF7inF4C3/e8EOuHJg0ZmgLkbfoHoUSfk6Ds7aCb
6kyBq6RzMwV3PBYN0htAqXjqVdNebXk7W8oKi+BxO5GRQoEpfU5TnjVy+QIDAQAB
AoGACgmsBRFo2aSg3jIzxVbcBgN5k5pHo0V3efN5lji5QE+rH4hggnyU+j1Pn8HX
aEgTkxx8Cje8lc/Ds5lmB3h95CxXhVFRGKCzToE6A72cIVMh8cMWXNul5PwXKWYd
yhuF1txHJieWMR1+dL7bgrmvFgux9lchHhJTDnM1+Aj2uqECQQD9v+kslZ7KHTrh
Q2Gm+Y5qg1T9zqMDJqaeB7PDphZibLWxpsp0aoyEBlPYDp4ZCNNoba/CxmFwIKDA
5pv0SP7tAkEA1+awuc6YJK5DvPjbpAbd7AyyVWkLjnnqNhLhTnYtm4oqeuUvhnln
rDHKBCgGl7cT7ft/Qd/oFOgMS3GbcBB2vQJAPR4BIozzDwlVSjZ5iccn5z20teEU
YEjp7rw6LxCM6a33YU+kCcA0EgSYIzz6FVuL1+Y7ZDV1+JTTQ0MIIyGFzQJAGQ6G
+CmH8snexrHDG/hIDTsz/0Gfu7teeUQbrfV9OY4BfB3ZNCwm4g8tOOpESZtPztDf
```

Figura 4.3.1.3 Componentes y clave pública RSA 1024bits.

4.3.1.1 Clave privada

Para descifrar la clave de sesión se utiliza la fórmula:

a = (m ^ d) mod n

En donde:

a: clave simple.

m: clave de sesión.

d: exponente privado.

n: módulo.

De acuerdo con esto, el archivo de la clave privada está compuesto de los datos requeridos para descifrar la clave de sesión: el nombre del usuario, el exponente privado (d) y el módulo (n), ver figura 4.3.1.1.1.

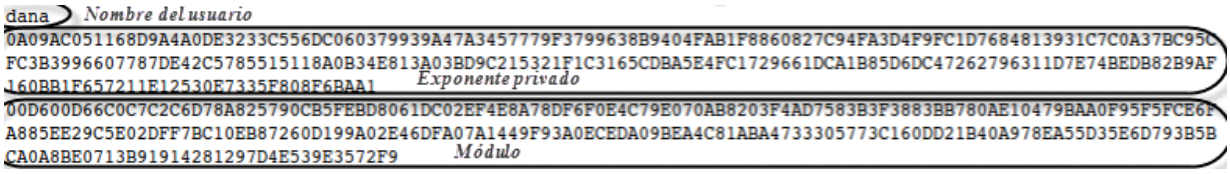


Figura 4.3.1.1.1 Contenido del archivo llave.prv.

4.3.1.2 Clave pública

Para cifrar la clave simple se utiliza la fórmula:

m = (a ^ b) mod n

En donde:

m: clave de sesión.

a: clave simple.

b: exponente público.

n: módulo.

De acuerdo con esto, el archivo de la clave pública está compuesto de los datos requeridos para cifrar la clave simple: el nombre del usuario, el exponente público (b) y el módulo (n), el estándar “10:” y el identificador del usuario, ver figura 4.3.1.2.1.

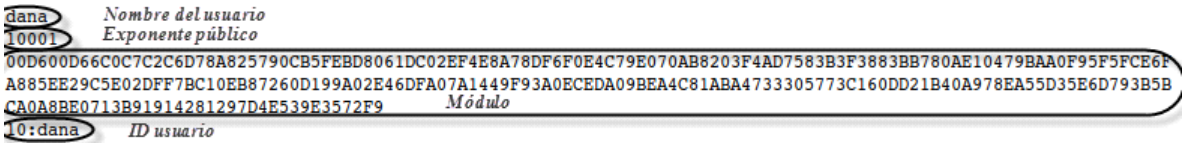


Figura 4.3.1.2.1 Archivo clave pública

4.3.2 Generación de la clave simple de 256 bits

Para generar la clave simple se utilizan números aleatorios, generados con la función srand, contenida en la librería <time.h> de C.

Función:
srand() – Generador de números pseudo-aleatorios.

Sinopsis:
#include <stdlib.h>

int rand(void);
void srand(unsigned int semilla);

Para eliminar el problema de que al ejecutar varias veces el programa la secuencia de números aleatorios se repita, esto es que se obtenga siempre el mismo número cuando se ejecuta de inicio el programa, se debe contar con una semilla la cual se utilizará para inicializar la función. Este problema es común cuando se emplea la función rand() en el lenguaje de programación C.

Para evitar este problema utilizamos la función srand(), a la que se le pasa de parámetro un número que se utilizará como número inicial, conocido como semilla, para las operaciones.

Si se inicializa la función con un número fijo, siempre nos dará el mismo valor, al igual que si se utiliza la función rand() de C. Por lo que para inicializar la semilla de la función que generará números aleatorios utilizaremos la fecha/hora del sistema. En C de linux esta fecha/hora se obtiene con la función time().

```
#include<time.h>
   srand(time(NULL));
```

Adicionalmente, realizamos la operación módulo 16 para obtener un valor hexadecimal aleatorio (que equivale a 4 bits), por lo que al ejecutar la función 64 veces continuas obtenemos una clave aleatoria de 256 bits.

Ejemplos de algunas claves simples generadas continuamente, demostrando que los números aleatorios no son repetidos, ver figura 4.3.2.1.

```
dana@ubuntu:-$ date
mar ago  3 13:36:00 CDT 2010
dana@ubuntu:-$ ./genKey256.exe
a5072e80f2c5b8f3226631bbeb41cbb60bd395387fe37d6afc02dcec72d3d8aedana@ubuntu:-$ date
mar ago  3 13:36:11 CDT 2010
dana@ubuntu:-$ ./genKey256.exe
757b349b7e3d6ad9a3f5fad44dbf2129794add54b8212fac3a124f68c17e397adana@ubuntu:-$ date
mar ago  3 13:36:16 CDT 2010
dana@ubuntu:-$ ./genKey256.exe
23c1085bbcd68ae5ac68fca3448cd1804414c6082dea7df1959814b684255a59dana@ubuntu:-$
```

Figura 4.3.2.1 Generación de números aleatorios.

4.3.3 Sockets de comunicación

El envío/recepción de la comunicación, es decir, mensajes y archivos intercambiados entre el servidor y el cliente se realiza a través de sockets de UNIX.

Los sockets utilizados están basados en el dominio de de AF_INET, el cual permite la comunicación entre máquinas que se encuentren en la red de Internet.
La creación y manejo de los sockets es realizada en lenguaje Perl, utilizando el módulo del repositorio CPAN IO::Socket, el cual proporciona la interface al objeto para las comunicaciones de un socket.

Función:
IO::Socket::INET - Proporciona una interface al objeto para crear y utilizar sockets en el dominio de AF_INET.

Sinopsis:

PeerAddr <hostname>[:<port>]
PeerHost
PeerPort <service>[(<no>)] | <no>
LocalAddr hostname[:port]
LocalHost
LocalPort <service>[(<no>)] | <no>
Proto "tcp" | "udp" | ...

Type SOCK_STREAM | SOCK_DGRAM | ...
Listen

En nuestro caso, las opciones utilizadas son:

SERVIDOR

```
use IO::Socket;
   IO::Socket::INET->new(
      Listen => [Tamaño Cola],
      LocalAddr => [IP Local],
      LocalPort => [Puerto Local],
      Proto     => 'tcp') or die "Can't create server socket: $!";
```

Listen: Indica el tamaño de la cola para escuchar. Si se define este parámetro se crea un socket para escuchar.

LocalAddr: Dirección IP del equipo local.

LocalPort: Puerto del equipo local para la conexión.

Proto: Nombre del protocolo de la conexión, en nuestro caso será una conexión para envío y recepción de datos tcp.

La creación de los sockets en el programa servidor, en espera de la solicitud del cliente puede verse en la figura 4.3.3.1.

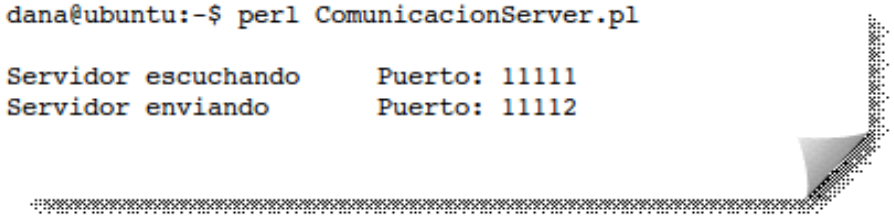


Figura 4.3.3.1 Creación sockets del servidor.

CLIENTE

```
use IO::Socket;

   IO::Socket::INET->new(
      PeerAddr => [IP Servidor],
```

```
PeerPort => [Puerto Servidor],
Proto    => 'tcp') or die "Can't create client socket: $!";
```

LocalAddr: Dirección IP del equipo remoto.

LocalPort: Puerto del equipo remoto para la conexión.

Proto: Nombre del protocolo de la conexión, en nuestro caso será una conexión para envío y recepción de datos TCP.

La creación de los sockets en el programa cliente, para la conexión con el programa servidor puede verse en la figura 4.3.3.2.

```
dana@ubuntu:-$ perl ComunicacionCliente.pl 127.0.0.1

Esperando la llave publica (llave.pub)
```

Figura 4.3.3.2 Creación sockets del cliente.

4.3.4 Cifrado y descifrado

Una vez que los equipos cliente y servidor han intercambiado la información necesaria para el cifrado de los datos, comenzarán a cifrar y descifrar los mensajes.

El cifrado de los mensajes que se envían se realiza en lenguaje Perl, utilizando el módulo del repositorio CPAN Crypt::GCrypt, que proporciona una interface a la librería de criptografía de GNU.

Función:

Crypt::GCrypt - Proporciona una interface a la librería libgcrypt de C. Soporta cifrado/descifrado simétrico y digestión de mensajes.

Sinopsis:

```
use Crypt::GCrypt;

my $cipher = Crypt::GCrypt->new(
  type => 'cipher',
  algorithm => 'aes',
  mode => 'cbc'
);

$cipher->start('encrypting');
$cipher->setkey('my secret key');
$cipher->setiv('my init vector');

my $ciphertext  = $cipher->encrypt('plaintext');
$ciphertext .= $cipher->finish;

my $plaintext  = $cipher->decrypt($ciphertext);
$plaintext .= $cipher->finish;
```

En nuestro caso, son creados dos procesos independientes que se encargarán del envío/recepción de los mensajes respectivamente. Los procesos son creados utilizando la función fork de perl.

PROCESO CIFRADO

```
$cipher = Crypt::GCrypt->new(
   type => 'cipher',
   algorithm => 'aes256',
   mode => 'ecb'
```

```
);
$cipher->start('encrypting');
$cipher->setkey($key);
$text = <STDIN>;
$ciphertext = $cipher->encrypt($text);
$ciphertext .= $cipher->finish;
$client->send($ciphertext);
```

PROCESO DESCIFRADO

```
$cipher = Crypt::GCrypt->new(
   type => 'cipher',
   algorithm => 'aes256',
   mode => 'ecb'
);

$cipher->start('decrypting');
$cipher->setkey($key);
$client->recv($ciphertext,256);
$plaintext  = $cipher->decrypt($ciphertext);
$plaintext .= $cipher->finish;
```

new(): Construye un objeto de Crypt::GCrypt para el cifrado/descifrado de la información, en nuestro caso será instanciado como $cipher.

El argumento “type” debe ser “cipher”.

El argumento “algorithm” indica el tipo de cifrado/descifrado de los mensajes. En nuestro caso es AES y el tamaño de la clave es de 256 bits.

El agumento “mode” indica el modo de cifrado/descifrado de los mensajes. En nuestro caso se realizará mediante un modo “ecb” en el que cada bloque de información se cifrará independientemente.

start(): Este método debe ser llamado antes de cualquier llamada a setkey(). Prepara el método cipher para el cifrado/descifrado, reiniciando el estado interno.

setkey(): El cifrado/descifrado usarán esta clave hasta que sea establecida alguna diferente.

encrypt(): Este método cifra los datos que sean indicados en el argumento, utilizando el objeto $cipher y regresa el texto cifrado. La salida del método es colocada en el buffer, esto significa que solamente se obtendrán múltiplos de texto cifrado del tamaño del bloque de cifrado y que al final es necesario llamar a la función finish().

decrypt(): Este método es la contraparte del método encrypt() y descifra la información que sea indicada en el argumento de la función y obtiene el texto plano original. La salida del método es colocada en el buffer, esto significa que solamente se obtendrán múltiplos de texto plano del tamaño del bloque de cifrado y que al final es necesario llamar a la función finish().

finish(): El algoritmo CBC debe enviar al buffer bloques de datos internamente hasta que sean múltiplos pares de los bloques del algoritmo (típicamente 8 o 16 bits). Después de la última llamada del método encrypt() o decrypt() sed deberá llamar a la función finish() para limpiar el buffer interno y regresar cualquier dato restante. Este método se ocupará también del padding/unpadding de los datos.

4.3.5 Envío y recepción

En este punto se tiene establecidos los sockets para la comunicación y los métodos para el cifrado y descifrado de la información, por lo que se requiere implementar las funciones para el envío y recepción de los mensajes, que como se ha visto en el punto anterior, están inmersos en las funciones de cifrado y

descifrado respectivamente, pero utilizan los métodos establecidos en los sockets de comunicación.
La comunicación entre los sockets del servidor y del cliente es establecida con la función accept, definida en el módulo IO::Socket.

Función:

accept() – Acepta la conexión de un socket. Una vez que es establecida una conexión exitosa regresa un nuevo objeto, este nuevo objeto será utilizado como un socket para la comunicación. Si no hay conexiones pendientes, accept() se bloqueará hasta que se presente una.

Se establece un nuevo socket en la función y regresa un arreglo de dos elementos que contienen el nuevo socket y la dirección del punto.

Sinopsis:

```
while( $client = $sock->accept() ) {
  # do something with the connection
  print "Incoming connection: ", $client->to_string, "\n";
  ...
  $client->free();
}
```

En nuestro caso el servidor espera hasta que el cliente establece una conexión, los valores del socket son instanciados en la variable $client.

```
$client = $server->accept;
$client1->peerhost();
$client1->peerport();
```

La conexión de los sockets en el programa servidor, una vez que se ha establecido la conexión con el programa cliente puede verse en la figura 4.3.5.1.

```
dana@ubuntu:-$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:45948
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:36818
```

Figura 4.3.5.1 Conexión sockets.

Una vez que el socket establece un vínculo entre el servidor y el cliente y que los valores de la conexión están almacenados en la variable $client, para el envío y recepción de la información se utilizarán las funciones correspondientes.

Función:

send() - Envía datos a un socket conectado. Regresa el número de bytes enviados o el error si ocurre.

Sinopsis:

send($buf [, $flags])

En nuestro caso, se envía por el socket la información que ya ha sido cifrada.

```
$client->send($ciphertext);
```

$buf: Contiene la información que será enviada. En nuestro caso está almacenada en la variable $ciphertext.

Función:

recv() - Recibe datos de un socket conectado. Regresa el número de bytes recibidos o el error si ocurre.

Sinopsis:

recv($buf, $len [, $flags])

En nuestro caso, se recibe por el socket la información cifrada.

```
$client->recv($ciphertext,256);
```

$buf: Variable en la que se almacena la información que está siendo enviada por el socket, en nuestro caso se almacena en la variable $ciphertext.

$len: Define el número de bytes que serán recibidos, en nuestro caso serán bloques de 256 bytes.

En la figura 4.3.5.2 puede observarse cómo el servidor envía un mensaje al cliente y cómo recibe un mensaje del cliente.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:51019
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:38100



Enviando la llave publica (llave.pub)
Llave publica enviada

Esperando la llave de sesion (llave.ses)
Llave de sesion recibida

Descifrado llave de sesion
Llave de sesion descifrada

        INICIANDO COMUNICACI◆N

hola!!

Cliente: aloooo
```

Figura 4.3.5.2. Comunicación en el servidor

En la figura 4.3.5.3 puede observarse como el cliente envía un mensaje al servidor y como recibe un mensaje del servidor.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto/cliente$ perl ComunicacionCliente.pl 127.0.0.1


Esperando la llave publica (llave.pub)
Llave de publica recibida

Validando llave Publica

Corresponden las llaves

Continuando con la comunicacion

Generando la llave simple (llave.sim)
Llave simple generada

Generando la llave de sesion (llave.ses)
Llave de sesion generada

Enviando la llave de sesion (llave.ses)
Llave de sesion enviada

        INICIANDO COMUNICACI◆N


Servidor: hola!!
aloooo
```

Figura 4.3.5.3. Comunicación en el cliente.

4.4 INTEGRACIÓN, PRUEBAS Y MANTENIMIENTO DE LA APLICACIÓN

A la técnica sistemática para construir la estructura del programa se conoce como integración, es decir, la unión de módulos para detectar errores asociados con la interacción, por esta razón las pruebas y el mantenimiento son parte medular del modelo lineal secuencial de la aplicación, el cual fue descrito anteriormente en el presente capítulo.

Con la finalidad de descubrir defectos y mejorar la calidad de la aplicación se realizan pruebas, entendiendo como defecto una deficiencia que resulta del proceso de desarrollo del sistema.

Las pruebas de software cliente/servidor se produce en tres niveles diferentes:

- Las aplicaciones de cliente individuales se prueban de modo desconectado ya que el funcionamiento del servidor y de la red subyacente no se consideran.

- Las aplicaciones de software de cliente y del servidor asociado se prueban al unísono, pero no se ejercitan específicamente las operaciones de red.

- Se prueba la arquitectura completa de cliente/servidor, incluyendo el rendimiento y funcionamiento de la red.

Aún cuando se efectúen muchas clases distintas de pruebas en cada uno de los niveles de detalle anteriores, es frecuente encontrar los siguientes enfoques de pruebas para aplicaciones cliente/servidor.
Durante el desarrollo de la aplicación, aplicamos las siguientes pruebas:

4.4.1 Pruebas de función de la aplicación

Se prueba la coordinación y las funciones de administración de datos del servidor. También se considera el rendimiento del servidor (tiempo de respuesta y transporte de datos en general).

En nuestro caso se probó la conexión de los sockets, como primer punto cuando el servidor se encuentra esperando la petición del cliente, como se muestra en la figura 4.4.1.1 y como segundo punto cuando el cliente hace la petición y se establece el socket, como se muestra en la figura 4.4.1.2.

```
dana@ubuntu:-$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112
```

Figura 4.4.1.1 Servidor en espera de una petición.

```
dana@ubuntu:-$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:45948
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:36818
```

Figura 4.4.1.2 Servidor cuando un cliente ha solicitado una petición.

Además del envío y recepción de archivos entre el cliente y el servidor, como se muestra en las figuras 4.4.1.3 y 4.4.1.4 ya que es necesario que el servidor y el cliente se sincronicen para enviar y recibir la información en el momento correcto.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:51019
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:38100



Enviando la llave publica (llave.pub)
Llave publica enviada

Esperando la llave de sesion (llave.ses)
Llave de sesion recibida
```

Figura 4.4.1.3. Envío/recepción archivos en el servidor.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto/cliente$ perl ComunicacionCliente.pl 127.0.0.1


Esperando la llave publica (llave.pub)
Llave de publica recibida
```

Figura 4.4.1.4. Envío/recepción archivos en el cliente.

4.4.2 Pruebas de transacciones

Se crea una serie de pruebas adecuada para comprobar que todas las clases de transacciones se procesen de acuerdo con los requisitos. Las transacciones hacen especial hincapié en la corrección de procesamiento y también en los temas de rendimiento.

En nuestro caso fue probada como transacción la validación de la clave pública, enviada por el servidor, en el cliente. Como se muestra en la figura 4.4.2.1.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto/cliente$ perl ComunicacionCliente.pl 127.0.0.1


Esperando la llave publica (llave.pub)
Llave de publica recibida

Validando llave Publica

Corresponden las llaves

Continuando con la comunicacion
```

Figura 4.4.2.1. Validación de las claves.

Adicionalmente, se realizaron pruebas con la generación y el cifrado de la clave simple en el cliente, como se muestra en la figura 4.4.2.2.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto/cliente$ perl ComunicacionCliente.pl 127.0.0.1


Esperando la llave publica (llave.pub)
Llave de publica recibida

Validando llave Publica

Corresponden las llaves

Continuando con la comunicacion

Generando la llave simple (llave.sim)
Llave simple generada

Generando la llave de sesion (llave.ses)
Llave de sesion generada
```

Figura 4.4.2.2. Generación y cifrado de la clave simple.

Además de pruebas con el descifrado de la clave de sesión en el servidor, como se muestra en la figura 4.4.2.3.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:51019
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:38100



Enviando la llave publica (llave.pub)
Llave publica enviada

Esperando la llave de sesion (llave.ses)
Llave de sesion recibida

Descifrado llave de sesion
Llave de sesion descifrada
```

Figura 4.4.2.3. Descifrado de la clave de sesión.

4.4.3 Pruebas de comunicaciones a través de la red

Estas pruebas verifican que la comunicación entre los nodos de la red se produzca correctamente, y que el paso de mensaje, las transacciones y el tráfico de red relacionado tenga lugar sin errores. Aquí es donde se efectúan las pruebas de seguridad de la red como parte de esta actividad de prueba.

Las pruebas se realizan para detectar errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

En nuestro caso, la comunicación entre el cliente y el servidor es primordial en la mayoría de los pasos de la aplicación como se muestra en las figuras 4.4.3.1 para el servidor y en la figura 4.4.3.2 para el cliente.

```
dana@ubuntu:~/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto$ perl ComunicacionServer.pl

Servidor escuchando     Puerto: 11111
Servidor enviando       Puerto: 11112

Conexion establecida (Escuchando)       Host: 127.0.0.1 Puerto:51019
Conexion establecida (Enviando) Host: 127.0.0.1 Puerto:38100



Enviando la llave publica (llave.pub)
Llave publica enviada

Esperando la llave de sesion (llave.ses)
Llave de sesion recibida

Descifrado llave de sesion
Llave de sesion descifrada

        INICIANDO COMUNICACI?N

hola!!

Cliente: aloooo
```

Figura 4.4.3.1. Comunicación en el servidor.

```
dana@ubuntu:-/ChatComunicacionOpenSSL/RESPALDO CRYPT/proyecto/cliente$ perl ComunicacionCliente.pl 127.0.0.1

Esperando la llave publica (llave.pub)
Llave de publica recibida

Validando llave Publica

Corresponden las llaves

Continuando con la comunicacion

Generando la llave simple (llave.sim)
Llave simple generada

Generando la llave de sesion (llave.ses)
Llave de sesion generada

Enviando la llave de sesion (llave.ses)
Llave de sesion enviada

        INICIANDO COMUNICACI[illegible]N


Servidor: hola!!
aloooo
```

Figura 4.4.3.1 Comunicación en el cliente

4.4.4 Mantenimiento

El software indudablemente sufre cambios después de ser puesto en producción. Se producen cambios porque se encuentran errores, debido a que el software debe acoplarse a los cambios de su entorno, como lo puede ser por un nuevo sistema operativo, librerías nuevas, etc., o porque el cliente requiere mejoras funcionales o de rendimiento.

Tipos de mantenimiento.

Un aspecto importante en el diseño y desarrollo de sistemas es el mantenimiento, que también es parte del modelo lineal secuencial empleado para el desarrollo de la aplicación.

Una vez finalizada la fase de desarrollo e implantación del sistema, es imprescindible garantizar el mantenimiento del mismo. Los programas deben ser actualizados con el tiempo, ya que las reglas de negocio pueden variar o es necesario adaptarse al entorno o a los cambios en los sistemas tecnológicos.

Existen diferentes tipos de mantenimiento aplicables al sistema:

• Mantenimiento preventivo. Es la actividad proactiva en la cual se realizan modificaciones a la aplicación para mejorar la estabilidad, escalabilidad y confiabilidad en la operación. También es útil para proporcionar bases seguras sobre las que podrán implementarse mejoras posteriores.

• Mantenimiento correctivo. Es el que se realiza cuando ocurre un error en la operación del sistema.

• Mantenimiento adaptativo. Se presenta cuando se generan cambios en los requerimientos de tal manera que cumpla con nuevas especificaciones. Estos requerimientos también pueden ser cambios en las plataformas de hardware o sistemas operativos.

• Mantenimiento perfectivo. Se realiza cuando existe la necesidad de optimización de procesos, sin que cambien forzosamente los requerimientos. La especificación permite entender claramente el impacto de los cambios de manera que estos se implanten confiadamente.

Podemos ver la importancia que tiene cada uno de éstos para el funcionamiento adecuado del sistema, por lo que se debe de crear un plan para ejecutarlos de acuerdo con las necesidades del negocio y nuevos estándares, por si se llegara a descubrir alguna vulnerabilidad en el algoritmo empleado.

Madurez de la aplicación.

El estándar IEEE 982.1 sugiere un índice de madurez de software que proporciona una indicación de la estabilidad de un producto software (basada en los cambios que ocurren con dada versión del producto). El índice de madurez del software (IMS) se calcula de la siguiente manera:

IMS= [MT – (Fa + Fc+ Fd)] / MT

Donde:

MT= número de módulos en la versión actual.
FC= número de módulos en la versión actual que se han cambiado.
Fa= número de módulos en la versión actual que se han añadido.
Fd= número de módulos de la versión anterior que se han borrado en la versión actual.
A medida que el IMS se aproxima a 1, el producto se empieza a estabilizar. El IMS puede utilizarse también como métrica para la planificación de las actividades de mantenimiento del software.

4.5 AUDITORÍA DE LA APLICACIÓN

Se conoce como al proceso de recopilar, agrupar y evaluar evidencias para determinar si un sistema de información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización, utiliza eficientemente los recursos y cumple con las leyes y regulaciones establecidas. También permiten detectar de forma sistemática el uso de los recursos y los flujos de información dentro de una organización y determinar qué información es crítica para el cumplimiento de su misión y objetivos,

identificando necesidades, duplicidades, etc., que obstaculizan flujos de información eficientes.

Auditar consiste principalmente en estudiar los mecanismos de control que están implantados en una empresa u organización, determinando si los mismos son adecuados y cumplen unos determinados objetivos o estrategias, estableciendo los cambios que se deberían realizar para la consecución de los mismos.

Los objetivos de la auditoría Informática son:

- El control de la función informática.
- El análisis de la eficiencia de los sistemas informáticos.
- La verificación del cumplimiento de la normativa en este ámbito.
- La revisión de la eficaz administración de los recursos informáticos.

Dentro de la auditoría informática destacan los siguientes tipos (entre otros):

- Auditoría de la gestión: la contratación de bienes y servicios, documentación de los programas, etc.
- Auditoría legal del Reglamento de Protección de Datos: Cumplimiento legal de las medidas de seguridad exigidas por el Reglamento de desarrollo de la Ley Orgánica de Protección de Datos.
- Auditoría de los datos: Clasificación de los datos, estudio de las aplicaciones y análisis de los flujogramas.
- Auditoría de las bases de datos: Controles de acceso, de actualización, de integridad y calidad de los datos.
- Auditoría de la seguridad: Referidos a datos e información verificando disponibilidad, integridad, confidencialidad, autenticación y no repudio.
- Auditoría de la seguridad física: Referido a la ubicación de la organización, evitando ubicaciones de riesgo, y en algunos casos no revelando la

situación física de ésta. También está referida a las protecciones externas (arcos de seguridad, CCTV, vigilantes, etc.) y protecciones del entorno.

- Auditoría de la seguridad lógica: Comprende los métodos de autenticación de los sistemas de información.
- Auditoría de las comunicaciones. Se refiere a la auditoría de los procesos de autenticación en los sistemas de comunicación.

Auditoría de la seguridad en producción: Frente a errores, accidentes y fraudes.

Análisis de Bitácoras: se hace la revisión de las diferentes bitácoras como son: bitácoras de fallas de equipo, bitácoras de procesos ejecutados, bitácoras de accesos no autorizados, bitácoras de uso de equipo, entre otras. Aquí se puede ver cómo los recursos han sido utilizados y detectar parámetros de uso o desviaciones en cuanto a los procedimientos y políticas de la empresa.

Para el sistema, se requiere se registre la fecha y hora, IP, usuario y acción realizada para trazabilidad del lado del cliente y del lado del servidor con la finalidad de cumplir con la característica de no repudio de la información.

CONCLUSIONES

CONCLUSIONES

El sistema desarrollado proporciona un medio de comunicación seguro que reduce el riesgo de la pérdida de confidencialidad de la información transmitida entre los usuarios y el servidor de la red.

Con la implementación de este sistema se garantiza la privacidad de los datos personales en la transmisión de la información, lo cual es requerido por la ley federal de protección de datos personales en posesión de los particulares.

Las herramientas utilizadas durante el desarrollo de este proyecto son en su mayoría de código abierto por lo que se logró una reducción significativa en los costos de desarrollo e implementación del sistema.

El método de cifrado híbrido que se implementó en el sistema proporciona los beneficios de un algoritmo simétrico como la velocidad y menor complejidad; y la confiablidad de un algoritmo asimétrico utilizado para la transmisión de la llave de sesión.

El seguir una metodología al pie de la letra, nos permitió resolver el problema de una manera sistemática.

REFERENCIAS

REFERENCIAS

Libros

Espina, E., Guel, J. (2005). *Ataques y métodos de intrusión*. México: Dirección General de Servicios de Cómputo Académico.

Gantz, J., Rochester, J. (2009). *Pirates of the Digital Millennium: How the Intellectual Property Wars Damage Our Personal Freedoms, Our Jobs, and the World Economy*. Financial Times Management.

Harris, S. (2008). *CISSP All-in-One Exam Guide (4ª ed.)*. Estados Unidos: McGraw-Hill.

Harris, S., Harper, A., Eagle, C., Ness, J., Lester, M. (2005). ***Gray*** *Hat Hacking: The Ethical Hacker's Handbook*. Estados Unidos: McGraw-Hill.

Lopez, A., Novo, A. (2000). *Protocolos de Internet Diseño e implementación en sistemas UNIX*. Colombia: Alfaomega.

Malik, S. (2003). *Network Security Principles and Practices*. Estados Unidos: Cisco Press.

Northcutt, S., Novak J. (2001). *Guía Avanzada Detección de Intrusos* (2ª ed.). Madrid: Pearson Education.

Varela, R., Granados, G. (2006). *Criptografía*. México: Dirección General de Servicios de Cómputo Académico.

Vicente, C. (2006). *Seguridad en Redes UNIX*. México: Dirección General de Servicios de Cómputo Académico.

Zaragoza, F., Alavez, S., Becerril, I. (2005). *Introducción a la seguridad en Unix*. México: Dirección General de Servicios de Cómputo Académico.

Páginas de internet

Ley Federal de Protección de Datos Personales en Posesión de los Particulares. (2010). Obtenida el 6 de noviembre de 2010, Secretaría de Gobernación, Diario Oficial de la Federación: http://dof.gob.mx/ley-reg.php.

The Perl language interpreter (n.d.). Obtenida el 9 de octubre de 2010, Perl Programming Documentation: http://perldoc.perl.org.

Warning of webmail wi-fi hijack. (2007). Obtenida el 1 de junio de 2010, BBC News, Technology: http://news.bbc.co.uk/2/hi/technology/6929258.stm.

GLOSARIO

GLOSARIO

AES (Advanced Encryption Standard): Algoritmo de cifrado simétrico, diseñado como reemplazo para el algoritmo DES. También conocido como Rijndael, adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001.

Amenaza: Todas aquellas acciones que podrían explotar una vulnerabilidad y causar una violación de la seguridad de los sistemas, esto es su confidencialidad, integridad y/o disponibilidad.

ARP Spoofing: Suplantación de identidad por falsificación de la tabla de resolución de direcciones de protocolos (ARP), cuando se realiza la traducción de una dirección MAC a una dirección IP.

Asociación de Seguridad (SA, security association): Es el establecimiento de atributos de seguridad compartidos entre dos entidades de la red para proporcionar una comunicación segura.

Atacante: Persona que actúa contra algo para destruirlo.

Base de Datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Clave: Dato utilizado para verificar que alguien está autorizado para acceder a un servicio o un sistema.

Confidencialidad: Propiedad de prevenir la divulgación de información a personas o sistemas no autorizados.

Cookie: Fragmento de información que se almacena en el disco duro del visitante de una página web a través de su modo a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas.

Criptografía: Método de almacenar y transmitir datos en una forma que únicamente los destinatarios pueden leer y procesar. Es considerada como la ciencia de proteger la información mediante codificación en un formato no entendible.

Criptosistema (Sistema Criptográfico): Colección de transformaciones de texto claro en texto cifrado y viceversa, en la que la transformación o transformaciones que se han de utilizar son basadas en claves. Las transformaciones son definidas normalmente por un algoritmo matemático.

Disponibilidad: Característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella: personas, procesos o aplicaciones.

DNS Spoofing: Suplantación de identidad por nombre de dominio, al realizar una consulta al DNS para la resolución de una IP por un nombre de dominio y viceversa.

DNS: Sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. Este sistema asocia información variada con nombres de dominios asignados a cada uno de los participantes. Su función más importante, es traducir (resolver) nombres

inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red.

DSA (Digital Signature Algorithm, Algoritmo de Firma digital): Es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales. Este algoritmo, sirve para firmar y no para cifrar información.

FTP (File Transfer Protocol, Protocolo de Transferencia de Archivos): Protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

Handshaking: Proceso automatizado de negociación que establece de forma dinámica los parámetros de un canal de comunicaciones establecido entre dos entidades antes de que comience la comunicación normal por el canal. De ello se desprende la creación física del canal y precede a la transferencia de información normal.

Impacto: Impresión o efecto muy intenso que causa algo o alguien sobre otra cosa o persona.

Índice de madurez del software (IMS): Proporciona una indicación de la estabilidad de un producto software (basada en los cambios que ocurren con dada versión del producto). A medida que el IMS se aproxima a 1, el producto se empieza a estabilizar. El IMS puede utilizarse también como métrica para la planificación de las actividades de mantenimiento del software.

Integridad: Propiedad que busca mantener los datos libres de modificaciones no autorizadas.

IP Spoofing: Suplantación de la dirección IP origen, directamente en los paquetes de la red.

IP: Etiqueta numérica que identifica, de manera lógica y jerárquica, a un interfaz de un dispositivo dentro de una red que utilice el protocolo IP, que corresponde al nivel de red del protocolo TCP/IP.

IPsec (Internet Protocol Security, Protocolo de Internet Seguro): Conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos.

Magerit: Metodología de análisis y administración de riesgos de los sistemas de tecnologías de información para realizar un análisis de riesgos y recomendar los controles necesarios para su minimización.

Mail Spoofing: Suplantación de la dirección de correo electrónico de otras personas o entidades.

Media access control (MAC): Identificador de 48 bits que corresponde de forma única a una tarjeta o dispositivo de red. Se conoce también como dirección física.

Middleware: Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

Normas NMX: Las Normas Mexicanas, creadas en el Artículo 3 Fraccción X de la Ley Federal sobre Metrología y Normalización.
“Articulo 3, Fracción X. Norma mexicana: la que elabore un organismo nacional de normalización, o la Secretaría, en los términos de esta Ley, que prevé para un uso común y repetido reglas, especificaciones, atributos, métodos de prueba, directrices, características o prescripciones aplicables a un producto, proceso,

instalación, sistema, actividad, servicio o método de producción u operación, así como aquellas relativas a terminología, simbología, embalaje, marcado o etiquetado."

Normas NOM: La Norma Oficial Mexicana, creadas en el Art. 3 Frac. XI de la Ley Federal sobre Metrología y Normalización.
"Articulo. 3, Fracción XI. Norma Oficial Mexicana: la regulación técnica de observancia obligatoria expedida por las dependencias competentes, conforme a las finalidades establecidas en el artículo 40, que establece reglas, especificaciones, atributos, directrices, características o prescripciones aplicables a un producto, proceso, instalación, sistema, actividad, servicio o método de producción u operación, así como aquellas relativas a terminología, simbología, embalaje, marcado o etiquetado y las que se refieran a su cumplimiento o aplicación."

Recursos Humanos (RH): Función dentro de una organización que se ocupa de seleccionar, contratar, formar, emplear y retener a los colaboradores.

RSA: Algoritmo de clave pública desarrollado en 1977, válido tanto para cifrar como para firmar digitalmente. Actualmente sigue siendo el más utilizado en su tipo.

Session fixation: Sucede cuando un atacante fija un identificador de sesión a un usuario conocido, enviándole una liga que contiene ese identificador de sesión particular y esperando a que el usuario se autentique en la sesión.

Session hijacking: Sucede cuando un atacante utiliza el sniffing de los paquetes de la red para robar la cookie de sesión, de una sesión válida para obtener acceso no autorizado a la información transmitida durante la sesión.

Session sidejacking: Vulnerabilidad en sitios web que solamente cifran la autenticación de la sesión pero no cifran el resto de la sesión, lo que permite a un atacante obtener mediante el sniffer los paquetes de red que contienen la cookie de sesión.

Simplex: Tipo de conexión que sólo permite la transmisión en un sentido.

Sniffer: Programa y/o dispositivo de red que monitorea toda la información que es transmitida a través de una red de computadoras.

SSH (Secure Shell, Intérprete de órdenes seguro): Nombre del protocolo de comunicación utilizado para acceder a máquinas remotas a través de una red. Permite manejar un equipo mediante un intérprete de comandos.

SSL (Secure Sockets Layer, protocolo de capa de conexión segura): Protocolo que proporciona autenticación y privacidad de la información mediante el uso de algoritmos criptográficos. Generalment el servidor es autenticado mientras que el cliente se mantiene sin autenticar.

TCP/IP: Conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

Vulnerabilidad: Debilidad que puede explotarse para causar pérdida o daño al sistema.

Web Spoofing: Suplantación de una página web real en donde se encamina la conexión de una víctima a través de una página falsa hacia otras páginas maliciosas.