SEGURIDAD

DE LOS

SISTEMAS

DISTRIBUIDOS

INTRODUCCIÓN

La computación desde sus inicios ha sufrido muchos cambios, desde los grandes ordenadores que permitían realizar tareas en forma limitada y de uso un tanto exclusivo de organizaciones muy selectas, hasta los actuales ordenadores ya sean personales o portátiles que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona.

Los mayores cambios se atribuyen principalmente a dos causas, que se dieron desde las décadas de los setenta:

1. El desarrollo de los microprocesadores, que permitieron reducir en tamaño y costo a los ordenadores y aumentar en gran medida las capacidades de los mismos y su acceso a más personas.
2. El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar ordenadores con posibilidad de transferencia de datos a alta velocidad.

Es en este contexto que aparece el concepto de "Sistemas Distribuidos" que se ha popularizado tanto en la actualidad y que tiene como ámbito de estudio las redes como por ejemplo: Internet, redes de teléfonos móviles, redes corporativas, redes de empresas, etc.

En consecuencia, el presente trabajo que lleva el título de "Seguridad de los Sistemas Distribuidos", tiene como principal objetivo: "describir panorámicamente los aspectos relevantes que están involucrados en los Sistemas Distribuidos".

En los últimos años el tema de la seguridad en las redes de computadores se ha tornado en un asunto de primera importancia dado el incremento de prestaciones de las mismas, así como la imparable ola de ataques o violaciones a las barreras de acceso a los sistemas implementados en aquellas. Los "incidentes de seguridad" reportados continúan creciendo cada vez a un ritmo más acelerado, a la par de la masificación del Internet y de la complejidad del software desarrollado. Este texto pretende presentar brevemente algunas recomendaciones dirigidas a los administradores e implementadores de redes informáticas, lo he preparado a modo de síntesis breve a partir de diversos materiales difuminados en Internet y material brindado por el profesor de la cátedra. No pretendo hacer un listado exhaustivo de las responsabilidades del administrador ni explicar detalles acerca de la implementación de las recomendaciones, aunque sugiero algunos puntos de partida para esto.

SISTEMAS DISTRIBUIDOS

Desde el inicio de la era de la computadora moderna (1945), hasta cerca de 1985, solo se conocía la computación centralizada. A partir de la mitad de la década de los ochentas aparecen dos avances tecnológicos fundamentales:

Desarrollo de microprocesadores poderosos y económicos con arquitecturas de 8, 16, 32 y 64 bits.

Desarrollo de redes de área local (LAN) de alta velocidad, con posibilidad de conectar cientos de máquinas a velocidades de transferencia de millones de bits por segundo (mb/seg).

Aparecen los sistemas distribuidos, en contraste con los sistemas centralizados. Los sistemas distribuidos necesitan un software distinto al de los sistemas centralizados. Los S. O. para sistemas distribuidos han tenido importantes desarrollos pero todavía existe un largo camino por recorrer.

Los usuarios pueden acceder a una gran variedad de recursos computacionales:

De hardware y de software.

Distribuidos entre un gran número de sistemas computacionales conectados.

Un importante antecedente de las redes de computadoras lo constituye Arpanet, iniciada en 1968 en los EE. UU.

Que son los Sistemas Distribuidos?:

Son sistemas cuyos componentes hardware y software, que están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor.

Características de los Sistemas Distribuidos:

Concurrencia.- Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

Carencia de reloj global.- Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, está más bien distribuida a los componentes.

Fallos independientes de los componentes.- Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.

Evolución de los Sistemas Distribuidos:

Procesamiento central (Host).- Al comienzo se usaba el Procesamiento Central (Host); fue uno de los primeros modelos de ordenadores interconectados, llamados centralizados, donde todo el procesamiento de la organización se llevaba a cabo en una sola computadora, normalmente un Mainframe, y los usuarios empleaban sencillos ordenadores personales.

Los problemas de este modelo eran que cuando la carga de procesamiento aumentaba se tenía que cambiar el hardware del Mainframe, lo cual es más costoso que añadir más computadores personales clientes o servidores que aumenten las capacidades.

El otro problema que surgió son las modernas interfases gráficas de usuario, las cuales podían conllevar a un gran aumento de tráfico en los medios de comunicación y por consiguiente podían colapsar.

Grupo de Servidores.- Otro modelo que entró a competir con el anterior, también un tanto centralizado, eran un grupo de ordenadores actuando como servidores, normalmente de archivos o de impresión, poco inteligentes para un número de Minicomputadores que hacen el procesamiento conectados a una red de área local.

Los problemas de este modelo: podría generarse una saturación de los medios de comunicación entre los servidores poco inteligentes y los minicomputadores, por ejemplo cuando se solicitaban archivos grades por varios clientes a la vez, podían disminuir en gran medida la velocidad de transmisión de información.

La Computación Cliente Servidor.- Este modelo, que predomina en la actualidad, permite descentralizar el procesamiento y recursos, sobre todo, de cada uno de los servicios y de la visualización de la Interfaz Gráfica de Usuario. Esto hace que ciertos servidores estén dedicados solo a una aplicación determinada y por lo tanto ejecutarla en forma eficiente.

Cliente-Servidor

Es el sistema donde el cliente es una máquina que solicita un determinado servicio y se denomina servidor a la máquina que lo proporciona. Los servicios pueden ser:

- Ejecución de un determinado programa.
- Acceso a un determinado banco de información.
- Acceso a un dispositivo de hardware.

Es un elemento primordial, la presencia de un medio físico de comunicación entre las máquinas, y dependerá de la naturaleza de este medio la viabilidad del sistema.

Categorías de Servidores:

Servidores de archivos.- Proporciona archivos para clientes. Si los archivos no fueran tan grandes y los usuarios que comparten esos archivos no fueran muchos, esto sería una gran opción de almacenamiento y procesamiento de archivos. *El cliente solicita los archivos y el servidor los ubica y se los envía.*

Servidores de Base de Datos.- Son los que almacenan gran cantidad de datos estructurados, se diferencian de los de archivos pues la información que se envía está ya resumida en la base de datos. Ejemplo: *El Cliente hace una consulta, el servidor recibe esa consulta (SQL) y extrae solo la información pertinente y envía esa respuesta al cliente.*

Servidores de Software de Grupo.- El software de grupo es aquel, que permite organizar el trabajo de un grupo. El servidor gestiona los datos que dan soporte a estas tareas. Por ejemplo: almacenar las listas de correo electrónico. *El Cliente puede indicarle, que se ha terminado una tarea y el servidor se lo envía al resto del grupo.*

Servidores WEB.- Son los que guardan y proporcionan Páginas HTML. *El cliente desde un browser o link hace un llamado de la página y el servidor recibe el mensaje y envía la página correspondiente.*

Servidores de correo.- Gestiona el envío y recepción de correo de un grupo de usuarios (el servidor no necesita ser muy potente). El servidor solo debe utilizar un protocolo de correo.

Servidor de objetos.- Permite almacenar objetos que pueden ser activados a distancia. *Los clientes pueden ser capaces de activar los objetos que se encuentran en el servidor.*

Servidores de impresión.- Gestionan las solicitudes de impresión de los clientes. *El cliente envía la solicitud de impresión, el servidor recibe la solicitud y la ubica en la cola de impresión, ordena a la impresora que lleve a cabo las operaciones y luego avisa a la computadora cliente que ya acabo su respectiva impresión.*

Servidores de aplicación.- Se dedica a una única aplicación. Es básicamente una aplicación a la que pueden acceder los clientes.

Componentes de Software:

Se distinguen tres componentes básicos de software:

Presentación.- Tiene que ver con la presentación al usuario de un conjunto de objetos visuales y llevar a cabo el procesamiento de los datos producidos por el mismo y los devueltos por el servidor.

Lógica de aplicación.- Esta capa es la responsable del procesamiento de la información que tiene lugar en la aplicación.

Base de datos.- Esta compuesta de los archivos que contienen los datos de la aplicación.

Arquitecturas Cliente / Servidor:

A continuación mostramos las arquitecturas cliente-servidor más populares:

Arquitectura Cliente-Servidor de Dos Capas: Consiste en una capa de presentación y lógica de la aplicación; y la otra de la base de datos. Normalmente esta arquitectura se utiliza en las siguientes situaciones:

- Cuando se requiera poco procesamiento de datos en la organización.
- Cuando se tiene una base de datos centralizada en un solo servidor.
- Cuando la base de datos es relativamente estática.
- Cuando se requiere un mantenimiento mínimo.

Arquitectura Cliente-Servidor de Tres Capas: Consiste en una capa de la presentación, otra capa de la lógica de la aplicación y otra capa de la base de datos. Normalmente esta arquitectura se utiliza en las siguientes situaciones:

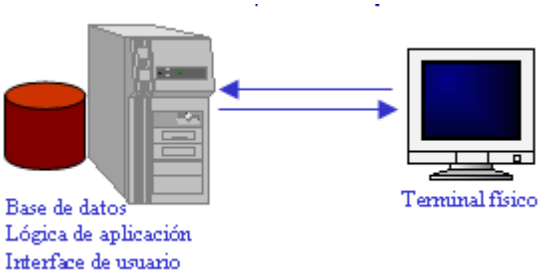
- Cuando se requiera mucho procesamiento de datos en la aplicación.
- En aplicaciones donde la funcionalidad este en constante cambio.
- Cuando los procesos no están relativamente muy relacionados con los datos.
- Cuando se requiera aislar la tecnología de la base de datos para que sea fácil de cambiar.
- Cuando se requiera separar el código del cliente para que se facilite el mantenimiento.

Está muy adecuada para utilizarla con la tecnología orientada a objetos.

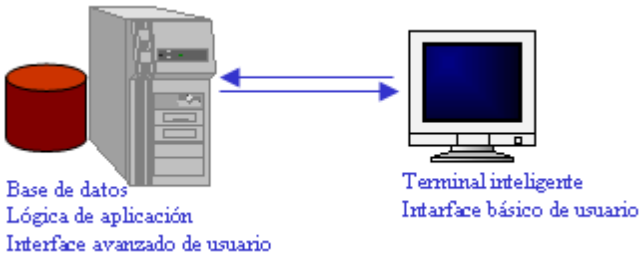
Clasificación de los Sistemas Cliente Servidor:

A continuación mostramos la clasificación de de los sistemas cliente/servidor de acuerdo al nivel de abstracción del servicio que ofrecen:

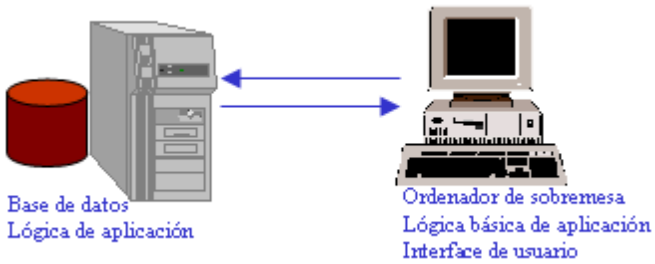
Representación Distribuida: La interacción con el usuario se realiza en el servidor, el cliente hace de pasarela entre el usuario y el servidor.



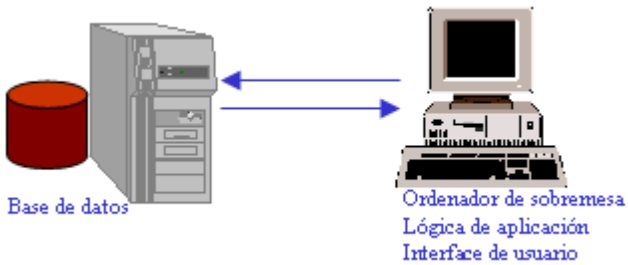
***Representación Remota*:** La lógica de la aplicación y la base de datos se encuentran en el servidor. El cliente recibe y formatea los datos para interactuar con el usuario.



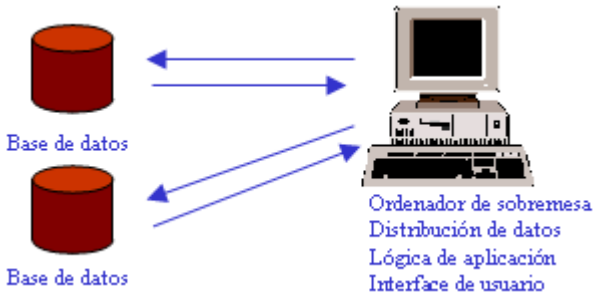
***Lógica Distribuida*:** El cliente se encarga de la interacción con el usuario y de algunas funciones triviales de la aplicación. Por ejemplo controles de rango de campos, campos obligatorios, etc. Mientras que el resto de la aplicación, junto con la base de datos, están en el servidor.



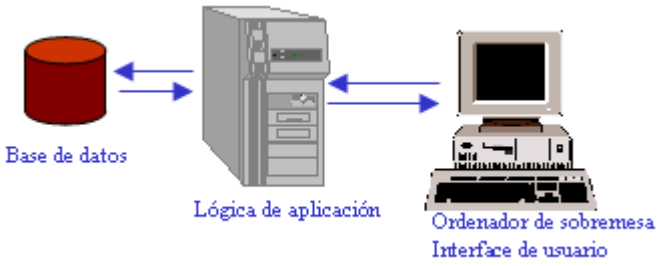
***Gestión Remota de Datos*:** El cliente realiza la interacción con el usuario y ejecuta la aplicación y el servidor es quien maneja los datos.



***Base de Datos Distribuidas*:** El cliente realiza la interacción con el usuario, ejecuta la aplicación, debe conocer la topología de la red, así como la disposición y ubicación de los datos. Se delega parte de la gestión de la base de datos al cliente.



***Cliente Servidor a Tres Niveles*:** El cliente se encarga de la interacción con el usuario, el servidor de la lógica de aplicación y la base de datos puede estar en otro servidor.



Protocolo:

Es un conjunto bien conocido de reglas y formatos que se utilizan para la comunicación entre procesos que realizan una determinada tarea. Se requieren dos partes:

- Especificación de la secuencia de mensajes que se han de intercambiar.
- Especificación del formato de los datos en los mensajes.

Un protocolo permite que componentes heterogéneos de sistemas distribuidos puedan desarrollarse independientemente, y por medio de módulos de software que componen el protocolo, haya una comunicación transparente entre ambos componentes. Es conveniente mencionar que estos componentes del protocolo deben estar tanto en el receptor como en el emisor.

Ejemplos de Protocolos Usados en los Sistemas Distribuidos:

IP: Protocolo de Internet: Protocolo de la capa de Red, que permite definir la unidad básica de transferencia de datos y se encarga del direccionamiento de la información, para que llegue a su destino en la red.

TCP: Protocolo de Control de Transmisión: Protocolo de la capa de Transporte, que permite dividir y ordenar la información a transportar en paquetes de menor tamaño para su transporte y recepción.

HTTP: Protocolo de Transferencia de Hipertexto: Protocolo de la capa de aplicación, que permite el servicio de transferencia de páginas de hipertexto entre el cliente WEB y los servidores.

SMTP: Protocolo de Transferencia de Correo Simple: Protocolo de la capa de aplicación, que permite el envío de correo electrónico por la red.

POP3: Protocolo de Oficina de Correo*:* Protocolo de la capa de aplicación, que permite la gestión de correos en Internet, es decir, le permite a una estación de trabajo recuperar los correos que están almacenados en el servidor.

Conceptos de Hardware:

Todos los sistemas distribuidos constan de varias cpu, organizadas de diversas formas, especialmente respecto de:

- La forma de interconectarlas entre sí.
- Los esquemas de comunicación utilizados.

Existen diversos esquemas de clasificación para los sistemas de cómputos con varias cpu:

- Uno de los más conocidos es la "Taxonomía de Flynn": Considera como características esenciales el número de flujo de instrucciones y el número de flujos de datos.

La clasificación incluye equipos **SISD, SIMD, MISD** y **MIMD.**

- **SISD** (Single Instruction Single Data: un flujo de instrucciones y un flujo de datos): Poseen un único procesador.
- **SIMD** (Single Instruction Multiple Data: un flujo de instrucciones y varios flujos de datos): Se refiere a ordenar procesadores con una unidad de instrucción que: Busca una instrucción. Instruye a varias unidades de datos para que la lleven a cabo en paralelo, cada una con sus propios datos. Son útiles para los cómputos que repiten los mismos cálculos en varios conjuntos de datos.
- **MISD** (Multiple Instruction Single Data: un flujo de varias instrucciones y un solo flujo de datos): No se presenta en la práctica.
- **MIMD** (Multiple Instruction Multiple Data: un grupo de computadoras independientes, cada una con su propio contador del programa, programa y datos): **Todos los sistemas distribuidos son de este tipo.**

Un avance sobre la clasificación de Flynn incluye la división de las computadoras MIMD en dos grupos:

1. **Multiprocesadores:** poseen memoria compartida: Los distintos procesadores comparten el mismo espacio de direcciones virtuales.
2. **Multicomputadoras:** no poseen memoria compartida: Ej.: grupo de PC conectadas mediante una red.

Cada una de las categorías indicadas se puede clasificar según la arquitectura de la red de interconexión en:

- **Esquema de bus:** Existe una sola red, bus, cable u otro medio que conecta todas las máquinas: Ej.: la televisión por cable.
- **Esquema con conmutador:** No existe una sola columna vertebral de conexión: Hay múltiples conexiones y varios patrones de conexionado. Los mensajes de mueven a través de los medios de conexión. Se decide explícitamente la conmutación en cada etapa para dirigir el mensaje a lo largo de uno de los cables de salida. Ej.: el sistema mundial telefónico público.

Otro aspecto de la clasificación considera el acoplamiento entre los equipos:

- **Sistemas fuertemente acoplados:** El retraso al enviar un mensaje de una computadora a otra es corto y la tasa de transmisión es alta. Generalmente se los utiliza como sistemas paralelos.
- **Sistemas débilmente acoplados:** El retraso de los mensajes entre las máquinas es grande y la tasa de transmisión es baja. Generalmente se los utiliza como sistemas distribuidos.

Generalmente los multiprocesadores están más fuertemente acoplados que las multicomputadoras.

Conceptos de los Sistemas Distribuidos:

Las características principales responsables de la utilidad de los sistemas distribuidos son seis. Se trata de comparición de recursos, apertura (openness), concurrencia, escalabilidad, tolerancia a fallos y transparencia.

Comparición de Recursos;

El término 'recurso' es bastante abstracto, pero es el que mejor caracteriza el abanico de entidades que pueden compartirse en un sistema distribuido. El abanico se extiende desde componentes hardware como discos e impresoras hasta elementos software como ficheros, ventanas, bases de datos y otros objetos de datos.

La idea de compartición de recursos no es nueva ni aparece en el marco de los sistemas distribuidos. Los sistemas multiusuario clásicos desde siempre han provisto compartición de recursos entre sus usuarios. Sin embargo, los recursos de una computadora multiusuario se comparten de manera natural entre todos sus usuarios. Por el contrario, los usuarios de estaciones de trabajo monousuario o computadoras personales dentro de un sistema distribuido no obtienen automáticamente los beneficios de la compartición de recursos.

Los recursos en un sistema distribuido están físicamente encapsulados en una de las computadoras y sólo pueden ser accedidos por otras computadoras mediante las comunicaciones (la red). Para que la compartición de recursos sea efectiva, ésta debe ser manejada por un programa que ofrezca un interfaz de comunicación permitiendo que el recurso sea accedido, manipulado y actualizado de una manera fiable y consistente. Surge el término genérico de **gestor de recursos**.

Un gestor de recursos es un modulo software que maneja un conjunto de recursos de un tipo en particular. Cada tipo de recurso requiere algunas políticas y métodos específicos junto con requisitos comunes para todos ellos. Éstos incluyen la provisión de un esquema de nombres para cada clase de recurso, permitir que los recursos individuales sean accedidos desde cualquier localización; la traslación de nombre de recurso a direcciones de comunicación y la coordinación de los accesos concurrentes que cambian el estado de los recursos compartidos para mantener la consistencia.

Un sistema distribuido puede verse de manera abstracta como un conjunto de gestores de recursos y un conjunto de programas que usan los recursos. Los usuarios de los recursos se comunican con los gestores de los recursos para acceder a los recursos compartidos del sistema. Esta perspectiva nos lleva a dos modelos de sistemas distribuidos: el modelo **cliente-servidor** y el modelo **basado en objetos**.

Apertura (opennesss):

Un sistema informático es abierto si el sistema puede ser extendido de diversas maneras. Un sistema puede ser abierto o cerrado con respecto a extensiones hardware (añadir periféricos, memoria o interfaces de comunicación, etc...) o con respecto a las extensiones software (añadir características al sistema operativo, protocolos de comunicación y servicios de compartición de recursos, etc...). La apertura de los sistemas distribuidos se determina primariamente por el grado hacia el que nuevos servicios de compartición de recursos se pueden añadir sin perjudicar ni duplicar a los ya existentes.

Básicamente los sistemas distribuidos cumplen una serie de *características*:

- Los interfaces software clave del sistema están claramente especificados y se ponen a disposición de los desarrolladores. En una palabra, los interfaces se hacen públicos.
- Los sistemas distribuidos abiertos se basan en la provisión de un mecanismo uniforme de comunicación entre procesos e interfaces publicados para acceder a recursos compartidos.

Los sistemas distribuidos abiertos pueden construirse a partir de hardware y software heterogéneo, posiblemente proveniente de vendedores diferentes. Pero la conformidad de cada componente con el estándar publicado debe ser cuidadosamente comprobada y certificada si se quiere evitar tener problemas de integración.

Concurrencia:

Cuando existen varios procesos en una única maquina decimos que se están ejecutando concurrentemente. Si el ordenador está equipado con un único procesador central, la concurrencia tiene lugar entrelazando la ejecución de los distintos procesos. Si la computadora tiene N procesadores, entonces se pueden estar ejecutando estrictamente a la vez hasta N procesos.

En los sistemas distribuidos hay muchas maquinas, cada una con uno o mas procesadores centrales. Es decir, si hay M ordenadores en un sistema distribuido con un procesador central cada una entonces hasta M procesos estar ejecutándose en paralelo.

En un sistema distribuido que está basado en el modelo de compartición de recursos, la posibilidad de ejecución paralela ocurre por dos razones:

1. Muchos usuarios interactúan simultáneamente con programas de aplicación.
2. Muchos procesos servidores se ejecutan concurrentemente, cada uno respondiendo a diferentes peticiones de los procesos clientes.

El caso (1) es menos conflictivo, ya que normalmente las aplicaciones de interacción se ejecutan aisladamente en la estación de trabajo del usuario y no entran en conflicto con las aplicaciones ejecutadas en las estaciones de trabajo de otros usuarios.

El caso (2) surge debido a la existencia de uno o más procesos servidores para cada tipo de recurso. Estos procesos se ejecutan en distintas máquinas, de manera que se están ejecutando en paralelo diversos servidores, junto con diversos programas de aplicación. Las peticiones para acceder a los recursos de un servidor dado pueden ser encoladas en el servidor y ser procesadas secuencialmente o bien pueden ser procesadas varias concurrentemente por múltiples instancias del proceso gestor de recursos. Cuando esto ocurre los procesos servidores deben sincronizar sus acciones para asegurarse de que no existen conflictos. La sincronización debe ser cuidadosamente planeada para asegurar que no se pierden los beneficios de la concurrencia.

Escalabilidad:

Los sistemas distribuidos operan de manera efectiva y eficiente a muchas escalas diferentes. La escala más pequeña consiste en dos estaciones de trabajo y un servidor de ficheros, mientras que un sistema distribuido construido alrededor de una red de área local simple podría contener varios cientos de estaciones de trabajo, varios servidores de ficheros, servidores de impresión y otros servidores de propósito específico. A menudo se conectan varias redes de área local para formar **internetworks**, y éstas podrían contener muchos miles de ordenadores que forman un único sistema distribuido, permitiendo que los recursos sean compartidos entre todos ellos.

Tanto el software de sistema como el de aplicación no deberían cambiar cuando la escala del sistema se incrementa. La necesidad de escalabilidad no es solo un problema de prestaciones de red o de hardware, sino que esta íntimamente ligada con todos los aspectos del diseño de los sistemas distribuidos. El diseño del sistema debe reconocer explícitamente la necesidad de escalabilidad o de lo contrario aparecerán serias limitaciones.

La demanda de escalabilidad en los sistemas distribuidos ha conducido a una filosofía de diseño en que cualquier recurso simple -hardware o software-

puede extenderse para proporcionar servicio a tantos usuarios como se quiera. Esto es, si la demanda de un recurso crece, debería ser posible extender el sistema para darle servicio. Por ejemplo, la frecuencia con la que se accede a los ficheros crece cuando se incrementa el número de usuarios y estaciones de trabajo en un sistema distribuido. Entonces, debe ser posible añadir ordenadores servidores para evitar el cuello de botella que se produciría si un solo servidor de ficheros tuviera que manejar todas las peticiones de acceso a los ficheros. En este caso el sistema deberá estar diseñado de manera que permita trabajar con ficheros replicados en distintos servidores, con las consideraciones de consistencias que ello conlleva.

Cuando el tamaño y complejidad de las redes de ordenadores crece, es un objetivo primordial diseñar software de sistema distribuido que seguirá siendo eficiente y útil con esas nuevas configuraciones de la red. Resumiendo, el trabajo necesario para procesar una petición simple para acceder a un recurso compartido debería ser prácticamente independiente del tamaño de la red. Las técnicas necesarias para conseguir estos objetivos incluyen el uso de datos replicados, la técnica asociada de caching, y el uso de múltiples servidores para manejar ciertas tareas, aprovechando la concurrencia para permitir una mayor productividad.

Tolerancia a Fallos:

Los sistemas informáticos a veces fallan. Cuando se producen fallos en el software o en el hardware, los programas podrían producir resultados incorrectos o podrían pararse antes de terminar la computación que estaban realizando. El diseño de sistemas tolerantes a fallos se basa en dos cuestiones, complementarias entre sí: Redundancia hardware (uso de componentes redundantes) y recuperación del software (diseño de programas que sean capaces de recuperarse de los fallos).

En los sistemas distribuidos la redundancia puede plantearse en un grano mas fino que el hardware, pueden replicarse los servidores individuales que son esenciales para la operación continuada de aplicaciones críticas.

La recuperación del software tiene relación con el diseño de software que sea capaz de recuperar (roll-back) el estado de los datos permanentes antes de que se produjera el fallo.

Los sistemas distribuidos también proveen un alto grado de disponibilidad en la vertiente de fallos hardware. La disponibilidad de un sistema es una medida de la proporción de tiempo que esta disponible para su uso. Un fallo simple en una máquina multiusuario resulta en la no disponibilidad del sistema para todos los usuarios. Cuando uno de los componentes de un sistema distribuidos falla, solo se ve afectado el trabajo que estaba realizando el componente averiado. Un usuario podría desplazarse a otra estación de trabajo; un proceso servidor podría ejecutarse en otra máquina.

Transparencia:

La transparencia se define como la ocultación al usuario y al programador de aplicaciones de la separación de los componentes de un sistema distribuido, de manera que el sistema se percibe como un todo, en vez de una colección de componentes independientes. La transparencia ejerce una gran influencia en el diseño del software de sistema.

El manual de referencia RM-ODP identifica ocho formas de transparencia. Estas proveen un resumen útil de la motivación y metas de los sistemas distribuidos. Las transparencias definidas son:

- **Transparencia de Acceso:** Permite el acceso a los objetos de información remotos de la misma forma que a los objetos de información locales.
- **Transparencia de Localización:** Permite el acceso a los objetos de información sin conocimiento de su localización.
- **Transparencia de Concurrencia:** Permite que varios procesos operen concurrentemente utilizando objetos de información compartidos y de forma que no exista interferencia entre ellos.
- **Transparencia de Replicación:** Permite utilizar múltiples instancias de los objetos de información para incrementar la fiabilidad y las prestaciones sin que los usuarios o los programas de aplicación tengan por que conoces la existencia de las replicas.
- **Transparencia de Fallos:** Permite a los usuarios y programas de aplicación completar sus tareas a pesar de la ocurrencia de fallos en el hardware o en el software.
- **Transparencia de Migración:** Permite el movimiento de objetos de información dentro de un sistema sin afectar a los usuarios o a los programas de aplicación.
- **Transparencia de Prestaciones.** Permite que el sistema sea reconfigurado para mejorar las prestaciones mientras la carga varía.
- **Transparencia de Escalado:** Permite la expansión del sistema y de las aplicaciones sin cambiar la estructura del sistema o los algoritmos de la aplicación.

Las dos más importantes son las **transparencias de acceso y de localización;** su presencia o ausencia afecta fuertemente a la utilización de los recursos distribuidos. A menudo se las denomina a ambas transparencias de red. La **transparencia de red** provee un grado similar de anonimato en los recursos al que se encuentra en los sistemas centralizados.

Middleware:

El software distribuido requerido para facilitar las interacciones cliente-servidor se denomina middleware. El acceso transparente a servicios y recursos no locales distribuidos a través de una red se provee a través del middleware, que sirve como marco para la comunicación entre las porciones cliente y servidor de un sistema.

El middleware define: el API que usan los clientes para pedir un servicio a un servidor, la transmisión física de la petición vía red, y la devolución de

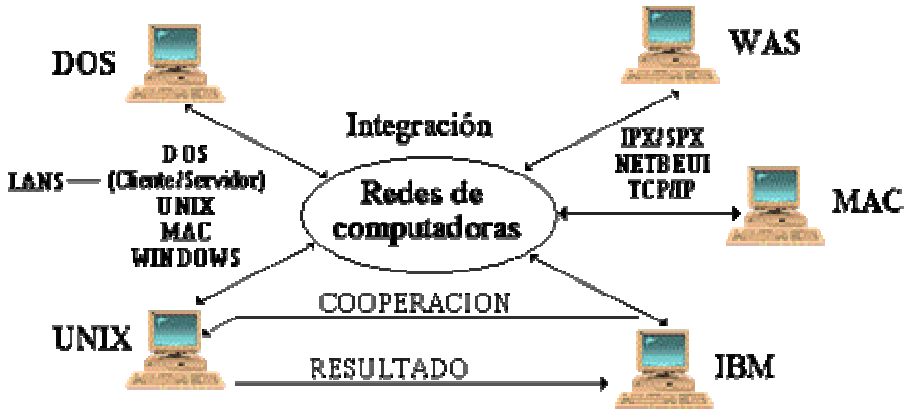
resultados desde el servidor al cliente. Ejemplos de middleware estándar para dominios específicos incluyen: ODBC, para bases de datos, Lotus para groupware, HTTP y SSL para Internet y CORBA, DCOM y JAVA RMI para objetos distribuidos.

El middleware fundamental o genérico es la base de los sistemas cliente-servidor. Los servicios de autentificación en red, llamadas a procedimiento remoto, sistemas de ficheros distribuidos y servicios de tiempo en red se consideran parte del middleware genérico. Este tipo de middleware empieza a ser parte estándar de los sistemas operativos modernos como Windows NT. En sistemas donde no se disponga deberá recurrirse a middlware del tipo OSD DCE (Distributed Computing Environment). El middleware específico para un dominio complementa al middlware genérico de cara a aplicaciones mucho más específicas.

El protocolo de comunicaciones más usado por el middlware, tanto genérico como específico, es TCP/IP. Esto se debe a su amplia difusión en todos los sistemas operativos del mercado y en especial en los ordenadores personales.

Factores que Han Afectado el Desarrollo de los Sistemas Distribuidos:

1. Avances Tecnológicos.
2. Nuevos requerimientos.
3. Globalización.
4. Aspectos Externos (Culturales, Políticos, Económicos).
5. Integración.



Ventajas y Desventajas de los Sistemas Distribuidos

Ventajas:

- Procesadores más poderosos y a menos costos.

- Desarrollo de Estaciones con más capacidades.
- Las estaciones satisfacen las necesidades de los usuarios.
- Uso de nuevas interfaces.
- Avances en la Tecnología de Comunicaciones.
- Disponibilidad de elementos de Comunicación.
- Desarrollo de nuevas técnicas.
- Comparición de Recursos.
- Dispositivos (Hardware).
- Programas (Software).
- Eficiencia y Flexibilidad.
- Respuesta Rápida.
- Ejecución Concurrente de procesos (En varias computadoras).
- Empleo de técnicas de procesamiento distribuido.
- Disponibilidad y Confiabilidad.
- Sistema poco propenso a fallas (Si un componente falla no afecta a la disponibilidad del sistema).
- Mayores servicios que elevan la funcionalidad (Monitoreo, Telecontrol, Correo Eléctrico, Etc.).
- Crecimiento Modular.
- Es inherente al crecimiento.
- Inclusión rápida de nuevos recursos.
- Los recursos actuales no afectan.

Desventajas:

- Requerimientos de mayores controles de procesamiento.
- Velocidad de propagación de información (Muy lenta a veces).
- Servicios de replicación de datos y servicios con posibilidades de fallas.
- Mayores controles de acceso y proceso (Commit).
- Administración más compleja.
- Costos.

Debido a ellas surge la necesidad de proteger la integridad y la privacidad de la información, y otros recursos que pertenecen a individuos y organizaciones, se conjuga ambos mundos: el físico y el digital. Nace, como es lógico, de la necesidad de compartir recursos. En el mundo físico, las organizaciones adoptan *políticas de seguridad* para poder compartir recursos dentro de unos límites especificados. Las políticas de seguridad se hacen cumplir con la ayuda de los *mecanismos de seguridad.*

En el mundo electrónico, la distinción entre políticas de seguridad y los mecanismos también es importante: sin ella, seria difícil determinar si un sistema particular es seguro. Las políticas de seguridad son independientes de la tecnología empleada, así como el instalar una cerradura en una puerta no garantiza la seguridad del edificio a menos que haya una política de uso (por ejemplo. que la puerta esté cerrada cuando no esté vigilada). Los mecanismos

de seguridad que describiré no garantizan, por sí mismos, la seguridad de un sistema.

Evolución de las Necesidades de Seguridad

	1965-75	*1975-89*	*1990-99*	*Actualmente*
Plataformas	Computadores multiusuario de tiempo compartido	Sistemas distribuidos basados en redes locales	Internet, servicios de área extensa	Internet + dispositivos móviles
Recursos compartidos	Memoria, archivos	Servicios locales (p. ej.: NFS), redes locales	e-mail, lugares web, comercio Internet	Objetos distribuidos, código móvil
Requisitos de seguridad	Identificación y autenticación de usuario	Protección de servicios	Seguridad robusta para transacciones comerciales	Control de acceso para objetos individuales, código móvil seguro
Entorno de gestión de la seguridad	Autoridad única, base de datos de autorización única (p. ej.: /etc/passwd)	Autoridad única, delegación, bases de datos de autorización replicadas (p. e).: NIS)	Muchas autoridades, sin autoridad en la red, en general	Autoridades por actividad, grupos con responsabilidades compartidas

A continuación describiremos los mecanismos que permiten hacer cumplir las políticas de seguridad en los sistemas distribuidos.

La distinción entre políticas de seguridad y mecanismos de seguridad es de utilidad cuando se diseñan sistemas seguros, pero no es fácil estar seguro de que cierto conjunto de mecanismos de seguridad implementan completamente las políticas de seguridad deseadas. Este es un modelo de seguridad diseñado para ayudar en el análisis de las amenazas de seguridad potenciales en un sistema distribuido.

Resumiendo el modelo de seguridad como sigue:

- Los procesos encapsulan recursos y acceden a comunicarse con los clientes a través de sus interfaces. Los usuarios u otros procesos pueden estar autorizados para operar sobre los recursos y estos deben estar protegidos contra accesos no autorizados.
- Los procesos interactúan en la red, que es compartida. Los enemigos o atacantes pueden acceder a la red y podrán copiar, leer o introducir mensajes arbitrarios, dirigidos hacia cualquier destino y simular que provienen de cualquier otro.

Este modelo de seguridad identifica las características de los sistemas de seguridad expuestos a ataques.

La Emergencia de la Criptografía en el Dominio Público:

La criptografía proporciona la base para la mayoría de los sistemas de seguridad de los computadores. La criptografía tiene una larga y fascinante historia. La necesidad de comunicaciones militares seguras y la

correspondiente necesidad del enemigo de interceptarlas y desencriptarlas ha fomentado la inversión de mucho esfuerzo intelectual, por parte de los mejores cerebros matemáticos de cada época.

Pero sólo en tiempos recientes la criptografía emerge de la trastienda en la que fue puesta por la clase dirigente de políticos y militares que solían controlar su desarrollo y su aplicación. Hoy en día es un tema de investigación abierta y con una comunidad de investigadores amplia y muy activa.

La reciente apertura es, en su mayor medida, resultado del importante crecimiento del interés en las aplicaciones no militares de la criptografía y los requisitos de seguridad de los sistemas de computadores distribuidos. Esto desembocó en la existencia, por primera vez, de una comunidad autosuficiente de criptógrafos aparte del entorno militar.

Irónicamente, esta apertura al público de la criptografía ha traído consigo un mayor avance de las técnicas criptográficas, su resistencia a los ataques criptoanalíticos y la comodidad con la que se despliegan las medidas criptográficas. La criptografía de clave pública es fruto de esta apertura. Un ejemplo más, el algoritmo de encriptación estándar DES fue inicialmente un secreto militar. Su eventual publicación y los esfuerzos exitosos para romperlo han traído consigo el desarrollo de algoritmos de encriptación de clave secreta mucho más resistentes.

Otro producto secundario útil ha sido el desarrollo de una terminología y aproximación común. Un ejemplo de esto último es la adopción de un conjunto de nombres familiares para los protagonistas (principales) involucrados en las transacciones que hay que asegurar. El uso de nombres familiares para los principales y los atacantes ayuda a aclarar y acercar al mundo las descripciones de los protocolos de seguridad y los potenciales ataques sobre ellos, lo que supone un paso importante hacia la identificación de sus debilidades.

Alice	Primer participante.
Bob	Segundo participante.
Carol	Otro participante en los protocolos a tres o cuatro bandas.
Dave	Participante en protocolos a cuatro bandas.
Eve	Fisgón.
Mallory	Atacante malevolente.
Sara	Un servidor.

Nombres familiares (del mundo anglosajón) para los protagonistas de los protocolos de seguridad.

Amenazas y Ataques

En la mayoría de los tipos de redes locales es fácil construir un programa sobre un computador conectado para que obtenga copias de los mensajes transmitidos entre computadores. Otras amenazas son más sutiles; un programa podría situarse a sí mismo en lugar del auténtico servidor de archivos y así obtener copias de información confidencial que los clientes, inconscientemente, envían para su almacenamiento.

Además del peligro de daño de información pueden aparecer reclamaciones fraudulentas contra el propietario de un sistema que no sea demostrablemente seguro. Para evitarlo, el propietario debe desacreditar la reclamación mostrando que el sistema es seguro contra tales violaciones, o produciendo un registro histórico de todas las transacciones. Un ejemplo es el *débito fantasma* en los cajeros automáticos. La mejor respuesta de un banco es proporcionar un registro de la transacción firmado digitalmente por el titular de la cuenta, que no pueda ser falsificado.

La principal meta de la seguridad es restringir el acceso a la información y los recursos de modo que sólo tengan acceso aquellos que estén autorizados.

Las amenazas de seguridad se dividen en tres clases:

Fuga — la adquisición de información por receptores no autorizados.

Alteración — la modificación no autorizada de información.

Vandalismo — interferencia en el modo de operación adecuado de un sistema, sin ganancia para el responsable.

Los ataques en los sistemas distribuidos dependen de la obtención de acceso a los canales de comunicación. Los métodos de ataque pueden clasificarse en función del modo en que se abusa del canal:

Fisgar - obtener copias sin autorización.

Suplantar — enviar o recibir mensajes utilizando la identidad de otro sin su autorización.

Alterar mensajes — interceptar mensajes y alterar sus contenidos antes de pasarlos al receptor.

Reenviar — almacenar mensajes interceptados y enviarlos más tarde.

Denegación de servicio — desbordar un canal o recurso para impedir que otros accedan a él.

Los ataques victoriosos dependen del descubrimiento de agujeros en la seguridad de los sistemas y estos problemas son comunes en los sistemas de hoy.

Cuando se diseñó Internet y los sistemas conectados a ella, la seguridad no era una prioridad.

La incorporación de medidas de seguridad requiere ser cuidadoso con la etapa de diseño.

Nos hemos concentrado en los ataques a los sistemas distribuidos que nacen de la exposición de sus canales de comunicación y sus interfaces. Los mecanismos de seguridad no pueden protegernos contra una clave de acceso mal elegida o custodiada. Pero para sistemas que incluyan programas móviles y sistemas cuya seguridad sea sensible a la fuga de información, hay más ataques.

Fugas de Información: Si pudiera observarse la sucesión de mensajes en la comunicación entre dos procesos, sería posible vislumbrar información importante aún de su sola existencia. Hay muchas formas sutiles de fugas de información, algunas maliciosas y otras que son consecuencia de errores inadvertidos. El potencial de las fugas aparece cuando se pueden observar los resultados de un cómputo. La aproximación empleada es la asignación de niveles de seguridad a la información y los canales, y analizar el flujo de información hacia los canales con el objetivo de asegurar que la información de alto nivel no fluya hacia los canales de bajo nivel.

Seguridad de las Transacciones Electrónicas

Muchas aplicaciones de comercio y demás implican transacciones que dependen de la seguridad, como ser:

- E-mail: hay muchos usos del correo en que los mensajes deben ser confidenciales (como enviar un número de tarjeta de crédito).
- Compra de bienes y servicios: estas transacciones son usuales. Los compradores seleccionan bienes y pagan por ellos empleando el Web, luego le son enviados por un mecanismo de reparto. Transacciones bancarias: los bancos electrónicos ofrecen a los usuarios todos los servicios que proporcionan los bancos convencionales.
- Micro-transacciones: Internet se presta a proporcionar pequeñas cantidades de información y otros servicios hacia sus clientes. Por ejemplo, el acceso a la mayoría de las páginas web no exige ningún pago, pero el desarrollo del Web como un medio de publicación de alta calidad seguramente depende de hasta qué punto los proveedores de información puedan obtener beneficio de los clientes de esta información.

Las transacciones como éstas sólo se pueden realizar de modo seguro cuando se encuentran protegidas contra la revelación de los códigos de crédito durante la transmisión, y contra un vendedor fraudulento que obtenga un pago sin intención de proveer bien alguno.

Una política de seguridad sensata para vendedores y compradores de Internet exige los siguientes requisitos:

- Autenticación del vendedor al comprador.
- Mantenimiento del número de tarjeta de crédito y otros detalles del comprador bajo secreto, y asegurar que se transmiten de forma inalterada del comprador al vendedor.
- Si los bienes se encuentran en una forma útil para su descarga, asegurar que su contenido llega al comprador sin alteración y sin ser desvelados a terceras partes.

Las necesidades de seguridad de las transacciones bancarias que emplean una red abierta son similares a las de las transacciones de compra, con el titular de la cuenta y el banco como vendedor, aunque hay necesidad de:

- Autenticar la identidad del titular de la cuenta hacia el banco antes de darle acceso a su cuenta.
- En esta situación es importante para el banco estar seguro de que el titular de la cuenta no pueda negar haber participado en una transacción. A esto se le da el nombre de no repudio.
- El comercio en Internet es una aplicación importante de las técnicas de seguridad, pero no es ciertamente la única. Es una necesidad cuando quiera que dos computadores sean utilizados por individuos u organizaciones para almacenar y comunicar información importante.

Diseño de Sistemas Seguros:

Debemos diferenciar las tareas específicas de un diseñador de sistemas seguros y de un programador. El objetivo del diseñador es excluir todos los posibles ataques y agujeros. La situación es análoga a la del programador cuyo principal objetivo es excluir todos los errores de su programa. En ningún caso existe un método concreto para asegurar las metas durante el diseño. Cada uno diseña con los mejores estándares disponibles y aplica un análisis informal y comprobaciones. Una vez que un diseño está completo, una opción es la validación formal. La seguridad trata de evitar los desastres y minimizar los contratiempos. Cuando se diseña para seguridad es necesario pensar siempre en lo peor.

Para demostrar la validez de los mecanismos de seguridad, empleados en un sistema, los diseñadores deben construir, en primer lugar, una lista de amenazas y probar que cada una de ellas se puede prevenir mediante los mecanismos empleados como por ej. Un histórico de seguridad.

Un histórico de seguridad contendrá una secuencia de registros fechados de las acciones de los usuarios. Como mínimo, los registros incluirán la identidad del principal, la operación realizada), la identidad del objeto sobre el que se opera y la fecha y hora.

Donde se sospeche que pudiera haber violaciones concretas, los registros pueden contener información a mayores para incluir la utilización de los recursos físicos (ancho de banda de red, periféricos), o disparar un procedimiento histórico especial de operaciones sobre objetos concretos.

Posteriormente se puede efectuar un análisis de carácter estadístico o bien basado en búsquedas. Incluso aunque no se sospeche de alguna violación, las técnicas estadísticas permitirán comparar registros a lo largo del tiempo para descubrir tendencias o cualquier suceso inusual.

El diseño de sistemas seguros es un ejercicio de balance entre los costos y las amenazas ya que:

- Su uso acarrea un costo (en esfuerzo computacional y uso de la red). Los costos deben compensar la amenaza.
- Unas especificaciones de medidas de seguridad inapropiadas podrían impedir a los usuarios legítimos el realizar ciertas acciones necesarias.

Premisas del peor caso posible y guías de diseño

Las interfaces están desprotegidas: Los sistemas distribuidos se componen, de procesos que ofrecen servicios, que comparten información. Sus interfaces de comunicación son necesariamente abiertas (para permitir el acceso a nuevos clientes), un atacante puede enviar un mensaje a cualquier interfaz.

Las redes son inseguras: Puede falsificarse la fuente de cualquier mensaje

Límite en el tiempo de vida y el alcance de cada secretor: Cuando se genera por primera vez una clave secreta, podemos confiar en que no se encuentra comprometida. Cuando más la usemos y más ampliamente se conozca, mayor será el riesgo. La utilización de secretos como las contraseñas y las claves secretas, compartidas deberían, tener una caducidad y un alcance de uso restringido.

Los algoritmos y el código de los programas están disponibles para los atacantes: Cuanto mayor y más ampliamente se difunde un secreto, mayor es el riesgo de que se descubra. Los algoritmos de encriptación secreta son totalmente .inadecuados para los entornos de red de gran escala de hoy en día. La mejor práctica es publicar los algoritmos empleados para la encriptación y la autentificación, confiándose solamente en el secreto de las claves criptográficas. Abriendo los algoritmos al público el escrutinio de terceras partes nos ayudará a garantizar el que éstos sean suficientemente fuertes.

Los atacantes tienen acceso a suficientes recursos: El costo de la potencia de cálculo decrece con rapidez. Deberíamos presuponer que los atacantes tendrán acceso a los computadores más grandes y más potentes que se puedan proyectar durante la vida útil de un sistema, y aun así añadir unos cuantos órdenes de magnitud más para contemplar desarrollos inesperados.

Minimícese la base de confianza: Las porciones de un sistema responsable de la implementación de seguridad y todos los componentes hardware y software sobre los que descanse, deben ser confiables; a esto se le denomina base de computación confiable. Cualquier defecto o error de programación en esta base de confianza puede producir debilidad en el sistema, de modo que debiéramos tender a minimizar su tamaño.

Criptografía:

La encriptación es el proceso de codificación de un mensaje de forma que queden ocultos sus contenidos.

La criptografía moderna incluye algunos algoritmos seguros de encriptación y desencriptación de mensajes.

Todos ellos se basan en el uso de ciertos secretos llamados claves.

Una clave criptográfica es un parámetro empleado en un algoritmo de encriptación de manera que no sea reversible sin el conocimiento de una clave.

Hay dos clases principales de algoritmos de encriptación de uso general:

- La primera emplea *claves secretas compartidas*: donde el emisor y el receptor deben compartir el conocimiento de una clave y ésta no debe ser revelada a ningún otro.
- La segunda emplea *pares de claves pública / privada*: donde el emisor de un mensaje emplea una clave pública, difundida previamente por el receptor, para encriptar el mensaje. El receptor emplea la clave privada correspondiente para desencriptar el mensaje. A pesar de que una multitud de principales pudiera examinar la clave pública, solamente el receptor puede desencriptar el mensaje, gracias a su clave privada. Los algoritmos de encriptación de clave pública requieren usualmente de 100 a 1.000 veces más potencia de procesamiento que los algoritmos de clave secreta, aunque hay situaciones en las que su conveniencia compensa esta desventaja.

Notación Criptográfica

KA: Clave secreta de Alice.

KB: Clave secreta de Bob.

KAB: Clave secreta compartida por Alice y Bob.

KApriv: Clave privada de Alice (solo conocida por Alice).

KApub: Clave pública de Alice (publicada por Alice para lectura de cualquiera).

{M} k: Mensaje M encriptado con la clave K.

{M} k: Mensaje M firmado con la clave K.

Algoritmos Criptográficos:

Un mensaje puede encriptarse mediante la aplicación por el emisor de alguna regla que transforme el *texto en claro* del mensaje a un *texto cifrado o*

criptograma. El receptor debe conocer la regla inversa para transformar el testo cifrado en el texto original.

La transformación de encriptación se define mediante dos elementos, una *función E* y una *clave K.* El mensaje resultante encriptado se escribe $\{M\}_k$.*E (K,M)* = $\{M\}_k$

La función de encriptación *E* define un algoritmo que transforma los datos de texto en datos en-criptados al combinarlos con la clave y transformándolos de un modo que depende fuertemente del valor de la clave. Podemos pensar en un algoritmo de encriptación como en una especificación de una familia grande de funciones, de la que seleccionamos un miembro particular mediante una clave dada. La desencriptación se lleva a cabo empleando una función inversa D, que también toma como parámetro una clave. Para la encriptación de clave secreta, la clave de desencriptado es la misma que la de encriptado: *D(K, E(K, M))=M*

Debido este uso simétrico de las claves, a menudo se habla de la criptografía de clave secreta como *criptografía simétrica,* mientras que la criptografía de clave pública se denomina *asimétrica* debido a que las claves empleadas para el encriptado.

***Algoritmos simétricos*:** Si eliminamos de la consideración el parámetro de la clave y definimos $F_k([M])$ = *E(K, M),* una propiedad de las funciones de encriptación robustas es que $F_k([M])$ sea relativamente fácil de calcular, mientras que la inversa, $F_k^{-1}([M])$ sea tan difícil de calcular que no sea factible. Tales funciones se las conoce como funciones de un solo sentido.

La efectividad de cualquier método para encriptar información depende del uso de una función de encriptación F_k que posea esta propiedad de irreversibilidad. Es esta propiedad la que protege a *M* de ser descubierto dado $\{M\}_k$.

***Algoritmos asimétricos*:** Cuando se emplea un par de claves pública / privada, las funciones de un solo sentido se explotan de otra forma. La base de todos los esquemas es la existencia de *funciones de puerta falsa.* Una función de puerta falsa es una función de un solo sentido con una salida secreta: es fácil calcularla en una dirección pero impracticable calcular su inversa a menos que se conozca un secreto.

El par de claves necesario para los algoritmos asimétricos se deriva de una raíz común. Para el algoritmo RSA, la raíz es un par de números primos muy grandes que se eligen arbitrariamente.

***Cifradores de bloque*:** La mayoría de los algoritmos de encriptación operan sobre bloques de datos de tamaño fijado; 64 bits es un tamaño de bloque popular. Cada mensaje se subdivide en bloques, el último bloque se rellena hasta la longitud estándar si fuera necesario y cada bloque se encripta independientemente. El primer bloque está listo para ser transmitido tan pronto como haya sido encriptado.

Para un simple cifrador de bloque, el valor de cada bloque del criptograma no depende de los bloques precedentes. Esto constituye una debilidad, dado que un atacante puede reconocer patrones repetidos e inferir su relación con el texto en claro. Tampoco se garantiza la integridad de los mensajes; debido a esto la mayoría de los algoritmos de cifrado de bloque emplean un encadenamiento de bloques de cifrado *(cipher block chaining,* CBC).

***Encadenamiento de bloques de cifrado**:* En el modo de encadenamiento de bloques de cifrado, cada bloque de texto en claro se combina con el bloque de criptograma precedente empleando la operación 0-exclusivo (XOR) antes de encriptarlo. En el desencriptado, el bloque se desencripta y se efectúa una operación XOR con el bloque encriptado precedente obteniendo el nuevo bloque de texto en claro. Esto es posible porque la operación XOR es *idempotente* (dos aplicaciones sucesivas producen el mismo valor).

Con el CBC se pretende impedir que dos porciones idénticas de texto en claro deriven en dos porciones de código encriptado idénticas. Para prevenir esto, necesitamos insertar un trozo de texto en claro diferente a la cabeza de cada mensaje. Tal texto se denomina *vector de iniciación.* Un número conteniendo la fecha y la hora del mensaje sirve bien como vector de iniciación, y fuerza a que cada mensaje comience con un bloque de texto en claro diferente.

El uso del modo CBC se restringe a la encriptación de datos que se transmiten a lo largo de una conexión fiable. El desencriptado fallará si se pierde cualquiera de los bloques del criptograma dado que el proceso de desencriptado será incapaz de desencriptar bloques a partir de ese momento.

Usos de la Criptografía

La criptografía juega tres papeles principales en la implementación de los sistemas seguros:

***Secreto e integridad*:** se emplea para mantener el secreto y la integridad de la información dondequiera que pueda estar expuesta a ataques potenciales.

Este uso se corresponde con su papel tradicional en las actividades militares y de inteligencia.

Se explota el hecho de que un mensaje encriptado con una clave de encriptación particular sólo puede ser desencriptado por un receptor que conozca la correspondiente clave de desencriptación. Así se conseguirá el secreto del mensaje encriptado en la medida de que la clave de desencriptado no esté *comprometida* y a condición de que el algoritmo de encriptación sea suficientemente fuerte como para derrotar cualquier posible intento de romperlo.

También preserva la integridad de la información encriptada, supuesto que se incluya algo de información redundante, de la misma forma en que se incluyen y comprueban las sumas de chequeo (checksums).

Autenticación: La criptografía se emplea como base para los mecanismos para autenticar la comunicación entre pares de principales.

Un principal que desencripta un mensaje con éxito empleando una clave particular puede presuponer que el mensaje es auténtico si contiene una suma de chequeo correcta o, si se emplea el modo de encriptación de encadenamiento de bloques.

Éstos podrán inferir que el emisor del mensaje estaba en posesión de la clave de encriptación correspondiente y entonces deducir la identidad del emisor, siempre qué la clave sólo sea conocida por las dos partes.

En resumen, si las claves se mantienen en secreto, una desencriptación con éxito da fe de que el mensaje desencriptado proviene de un emisor concreto.

***Firmas digitales*:** Ésta emula el papel de las firmas convencionales, verificando a una tercera parte que un mensaje o un documento es una copia inalterada producida por el firmante.

Las técnicas de firmas digitales se basan en una modificación irreversible sobre el mensaje o el documento de un secreto que únicamente conoce el firmante.

Esto se puede lograr encriptando el mensaje; o mejor, una forma comprimida del mensaje denominada resumen (digest), empleando una clave sólo conocida por el firmante.

Un resumen es un valor de longitud fija calculado mediante la aplicación de una función de resumen segura.

Un resumen seguro es similar a una función de chequeo, con la diferencia de que es muy difícil que dos mensajes diferentes produzcan el mismo resumen.

El resumen resultante encriptado actúa como una firma que acompaña el mensaje.

La firma se suele encriptar mediante un método de clave pública: el firmante genera una firma con su clave privada; la firma puede ser desencriptada por el receptor utilizando la correspondiente clave pública.

Hay un requerimiento adicional: el verificador de la firma debe asegurarse de que la clave pública que emplea es la que realmente pretende el firmante.

Certificados:

Un certificado digital es un documento que contiene una sentencia (generalmente corta) firmada por un principal. Estos pueden emplearse para establecer la autenticidad de muchos tipos de enunciados.

Para que los certificados sean útiles se requieren dos cosas:

- Un formato estándar y una representación para ellos de modo que los emisores de certifica dos y los usuarios de certificados puedan construirlos e interpretarlos.

- Un acuerdo sobre la forma en que se construyen las cadenas de certificados, y en particular la noción de autoridad fiable.

Uno de los problemas que aparecen con los certificados es la dificultad al elegir una autoridad fiable de donde pueda arrancar una cadena de autenticaciones. La elección de una autoridad dependerá del objetivo para el que se necesite el certificado. Otros problemas nacen del riesgo de comprometer (desvelar) las claves privadas y de la longitud permisible de una cadena de certificación, cuanto más larga mayor es el riesgo de un eslabón débil.

Control de Acceso:

Históricamente, la protección de los recursos de los sistemas distribuidos se implanta mayormente en el servicio concreto que queremos proteger. Los servidores reciben mensajes con peticiones de la forma *<op, principal, recurso>,* donde *op* es la operación solicitada, *principal* es una identidad o un conjunto de credenciales del principal que realiza la petición y *recurso* identifica el recurso sobre el que se aplica la operación. El servidor debe, en primer lugar, comprobar la autenticidad del mensaje de petición y las credenciales del principal y después aplicar el control de acceso, rehusando cualquier petición para la cual el principal solicitante no tenga los derechos de acceso pertinentes para realizar la operación requerida sobre el recurso especificado.

En sistemas distribuidos orientados al objeto puede haber muchos tipos de objeto a los cuales habrá que aplicar el control de acceso, y las decisiones son siempre específicas para cada aplicación.

Las decisiones de control de acceso se dejan usualmente al código de nivel de aplicación, pero se proporciona un soporte genérico para gran parte de la mecánica que soporta las decisiones. Esto incluye la autenticación de los principales, la firma y autenticación de las peticiones, y la administración de credenciales y datos de derechos de acceso.

Dominios de protección: Un dominio de protección es un entorno de ejecución compartido por un conjunto de procesos: contiene un conjunto de pares < *recurso, derechos>,* que listan los recursos que pueden ser accedidos por todos los procesos en ejecución dentro del dominio y especificando las operaciones permitidas sobre cada recurso. Un dominio de protección se asocia generalmente con un principal dado; cuando un usuario entra en el sistema, se comprueba su identidad y se crea un dominio de protección para todos los derechos de acceso que posee el principal. Conceptualmente, el dominio incluye todos los derechos que posea el principal, incluyendo cualquier derecho que ella adquiera por el hecho de pertenecer a varios grupos. Un dominio de protección es sólo una abstracción. En los sistemas distribuidos se emplean usualmente dos implementaciones alternativas.

Habilitaciones *(capabilities):* Cada proceso, según el dominio en que esté ubicado, aloja un conjunto de habilitaciones. Una habilitación es un valor binario que actúa como una clave de acceso permitiendo al que lo posee acceder a ciertas operaciones sobre un recurso especificado. Para su uso en los sistemas distribuidos, donde las habilitaciones deben ser infalsificables, toman la forma de:

- *Identificador del recurso:* Un identificador único para el recurso objeto.
- *Operaciones:* Una lista de operaciones permitidas sobre el recurso.
- *Código de autenticación:* Una firma digital que, hace la habilitación infalsificable.

Cuando se emplean habilitaciones, las peticiones de los clientes tienen la forma < *op, idUsuario, habilitación* >. Se incluye en ellas una habilitación para acceder al recurso en lugar de un simple identificador, lo que da al servidor una prueba inmediata de que el cliente está autorizado a acceder al recurso identificado por la habilitación con las operaciones especificadas por ésta. Una comprobación de control de acceso sobre una petición acompañada por una habilitación involucra solo la validación de la habilitación y una comprobación de que la operación solicitada está en el conjunto permitido por la habilitación. Esta característica es la mayor ventaja de las habilitaciones, que constituyen una clave de acceso autocontenida, tal y como una llave física para una cerradura es una clave para acceder al edificio protegido por la cerradura.

Las *habilitaciones* comparten dos inconvenientes de las llaves de una cerradura física:

- **Robo de llaves:** cualquiera que posea la llave de un edificio puede utilizarla para tener acceso a él, sea o no un poseedor autorizado de la llave; ésta pudiera haber sido robada u obtenida de alguna forma fraudulenta.
- **El problema de la revocación:** la calificación para poseer una llave cambia con el tiempo. Por ejemplo, el poseedor pudiera dejar de ser un empleado o el propietario del edificio, pero podría seguir reteniendo la llave, o una copia de ella, y utilizarla de manera no autorizada.

Quedan claros los problemas análogos para las *habilitaciones*:

- Las habilitaciones podrían, debido a un descuido o como resultado de un ataque a la confidencialidad, caer en manos de otros principales que no son para quienes fueron emitidas. Si ocurre esto, los servidores quedan a merced de ser utilizados ilícitamente.
- Resulta difícil cancelar las habilitaciones. El estado del portador pudiera cambiar y sus derechos.
- Listas de control de acceso: Se almacena una lista con cada recurso, con una entrada de la forma < *dominio, operaciones* > para cada dominio que tenga acceso al recurso y especificando las operaciones permitidas al dominio. Un dominio puede especificarse mediante un identificador de un principal o puede ser una expresión susceptible de ser utilizada para determinar la pertenencia del principal al dominio.

Este es el esquema adoptado en la mayoría de los sistemas de archivos, incluyendo UNIX y Windows NT, donde se asocia a cada archivo un conjunto de bits de permisos de acceso, y los dominios a los cuales se ceden los permisos se definen por medio de una referencia a la información de pertenencia almacenada con cada archivo.

Las peticiones a los servidores son de la forma <op. principal, recurso>, el servidor identifica el principal y comprueba si la operación solicitada está incluida en la entrada del principal en la lista de control de acceso del recurso relevante.

Las firmas, digitales, las credenciales y los certificados de clave pública proporcionan las bases criptográficas para el control de acceso seguro. Los canales seguros ofrecen beneficios de prestaciones, posibilitando el manejo de múltiples peticiones sin necesidad de comprobar los principales y las credenciales reiteradamente

Credenciales:

Las credenciales son un conjunto de evidencias presentadas por un principal cuando pide acceso a un recurso. En el caso más simple, un certificado de una autoridad relevante afirmando la identidad del principal es suficiente, y ésta podrá emplearse para comprobar los permisos del principal en una lista de control de. A menudo esto es todo lo que se necesita o se proporciona, pero el concepto puede generalizarse para tratar con muchos requisitos más sutiles.

No es conveniente pedir que los usuarios contacten con el sistema y se autentiquen ellos mismos cada vez que se requiere su autoridad para realizar una operación sobre un recurso protegido. En su lugar, se introduce la noción de credencial que *habla por* un principal. Así un certificado de clave pública de un usuario habla por ese usuario; cualquier proceso que reciba una petición autenticada con la clave privada del usuario puede asumir que la petición fue enviada por ese usuario.

Las credenciales basadas en funciones parecen especialmente útiles en el diseño de esquemas de control de acceso prácticos. Los conjuntos de credenciales basadas en funciones se definen para las organizaciones o para tareas cooperativas, y los derechos de acceso de nivel de aplicación se construyen con referencia a ellas. Se pueden asignar funciones o roles a principales específicos mediante la generación de un certificado de rol que asocia un principal con un rol determinado en una tarea u organización específica.

Delegación: Una forma particularmente útil de credencial es aquella que permite a un principal, o proceso actuando para un principal, realizar una acción con la autoridad de otro principal. Una necesidad de delegación puede aparecer en cualquier situación donde un servicio necesite acceder a un recurso protegido para completar una acción en representación de su cliente.

Se puede conseguir la delegación utilizando un certificado de delegación o una habilitación. El certificado está firmado por el principal solicitante y autoriza a otro principal (el servidor de impresión en nuestro ejemplo) para acceder a un recurso con nombre (el archivo que hay que imprimir). En los sistemas que lo soportan, las habilitaciones pueden lograr el mismo resultado sin necesidad de identificar a los principales; se puede pasar una habilitación para acceder a un recurso en la petición al servidor. La habilitación es un conjunto codificado e infalsificable de derechos de acceso al recurso.

Cuando se delegan derechos, es común restringirse a un subconjunto de los derechos que posee el principal que lo emite, de este modo el principal delegado no podrá abusar de ellos.

Cortafuegos:

Con ellos se protege una intranet, se realizan acciones de filtrado en las comunicaciones entrantes y salientes.

Los cortafuegos producen un entorno de comunicación local en el que se intercepta toda comunicación externa. Los mensajes se reenvían al recipiente local final sólo para las comunicaciones que estén autorizadas explícitamente.

Se puede controlar el acceso a las redes internas mediante cortafuegos, pero el acceso a los servicios públicos a Internet no puede restringirse porque su objetivo es ofrecer éstos a un amplio conjunto de usuarios. El empleo de cortafuegos no ofrece protección contra los ataques desde el interior de una organización, y es ciertamente tosco en el control del acceso externo. En consecuencia existe una necesidad de mecanismos de seguridad de un grano más fino, que permitan a los usuarios individuales compartir información con otros usuarios seleccionados sin comprometer la privacidad y la integridad.

Los cortafuegos no son particularmente útiles contra ataques de denegación de servicios, basado en la suplantación de direcciones IP. El problema es que el flujo de mensajes generado por tales ataques sobrepasa las capacidades de cualquier elemento de defensa singular como un cortafuego. Cualquier remedio contra las riadas de mensajes debiera aplicarse corriente arriba del objetivo.

Los remedios basados en el empleo de mecanismos de calidad de servicio para restringir el flujo de mensajes desde la red hasta un nivel que el objetivo pueda manejar parece ser que prometen.

Algoritmos de Clave Secreta (Simétricos):

Describiremos a continuación tres de ellos.

El primero, TEA, se ha escogido por la simplicidad de su diseño e implementación. Se continua la discusión con DES e IDEA aunque en menor detalle. DES ha sido un estándar nacional de los EE.UU. aunque su interés es histórico dado que sus claves de 56 bits son demasiado reducidas para resistir un ataque por fuerza bruta con el hardware actual. IDEA emplea una clave de

128 bits y es, probablemente, el algoritmo de encriptación simétrico de bloques más efectivo.

TEA. Los principios de diseño para los algoritmos simétricos que se delinearon anteriormente se ven ilustrados en *Tiny Encryption Algorithm* (TEA, pequeño algoritmo de encriptación).

Función de encriptación de TEA:

```
Void encripta (unsigned long k [ ], unsigned long texto [ ]{
    unsigned long  y = texto [0], z = texto [1];
    unsigned long delta = 0x 9e3779v9, suma = 0; int n;
    for ( n = 0 ; n < 32; n ++){
      suma + = delta;
      y + = (( z << 4) + k [0]) ^ ( z + suma) ^ (( z >> 5) + ( k [1]);
      z + = (( y << 4) + k [2]) ^ (y + suma ) ^ (( y >> 5 ) + (k [3]);
    }
    texto [0] = y; texto [1] = z;
}
```

El algoritmo TEA emplea vueltas de sumas enteras, XOR y desplazamientos lógicos de bits, para obtener la difusión y confusión de los patrones de bits en el texto en claro. El texto en claro es un bloque de 64 bits representado como dos enteros de 32 bits en el vector de texto [ ]. La clave tiene 128 bits representada como cuatro enteros de 32 bits. Esta clave es segura contra los ataques de fuerza bruta.

En cada una de las 32 etapas, se combinan repetidamente las dos mitades del texto con porciones desplazadas de la clave y entre sí en las líneas 5 y 6 el empleo de XOR sobre porciones del texto desplazada introduce confusión y el desplazamiento e intercambio de las dos porciones del texto introduce difusión. La constante *delta* que se combina con cada porción del texto en cada ciclo para oscurecer la clave en caso de que fuera revelada por una sección de texto que no variará. La función de desencriptado es la inversa de función de encriptación.

Función de desencriptación de TEA

```
Void encripta (unsigned long k [ ], unsigned long texto [ ]{
    unsigned long  y = texto [0], z = texto [1];
```

```
unsigned long delta = 0x 9e3779v9, suma = 0; int n;
for ( n = 0 ; n < 32; n ++){
  suma + = delta;
  y + = (( z << 4) + k [2]) ^ ( z + suma) ^ (( z >> 5) + ( k [3]);
  z + = (( y << 4) + k [0]) ^ (y + suma ) ^ (( y >> 5 ) + (k [1]);
}
texto [0] = y; texto [1] = z;
}
```

DES: El Estandar de Encriptación de Datos *(Data Encryption Standar)*, usado para aplicaciones gubernamentales y de negocios. En este estándar la función de encriptación proyecta un texto en claro de 64 bits usando una clave de 56 bits. El algoritmo tiene 16 etapas dependientes de claves conocidas como *Vueltas* en las que el dato a encriptar se rota bit a bit un número de veces que depende de la clave y tres transposiciones no dependientes de la clave. Sobre los computadores de los años setenta y ochenta, el algoritmo suponía un coste computacional no despreciable, y consecuentemente fue implementado en hardware VLSI con su consiguiente aplicación en interfaces de red y demás dispositivos de comunicación.

A pesar de que aún se emplea en muchas aplicaciones comerciales y de otro tipo, DES, en su forma básica, debe considerarse obsoleta para la protección de aquello que no sea información de bajo interés. En la práctica se utiliza un sistema conocido como *triple-DES* (o 3DES). Este método conlleva la aplicación repetida de DES con dos claves K_1 y K_2: $E_{3DES}(K_1, K_2, M) = E_{DES}(K_1, E_{DES}(K_2, E_{DES}(K_1, M)))$

Esto proporciona una resistencia contra los ataques por fuerza bruta equivalente a una longitud de clave de 112 bits, proporcionando resistencia suficiente para un futuro previsible a corto plazo, aunque tiene el problema menor de mayores requerimientos de cálculo, como resultado de la aplicación repetida de un algoritmo que aisladamente ya es lento para los estándares modernos.

IDEA. El Algoritmo de Encriptación de Datos Internacional *(International Data Encription Algorithm,* IDEA) se desarrolló a comienzos de los años noventa como sucesor de DES. Como TEA, emplea una clave de 128 bits para encriptar bloques de 64 bits. El algoritmo se basa en el álgebra de grupos y tiene ocho vueltas XOR, suma módulo 2^{16} y multiplicación. Tanto DES como IDEA, emplean una misma función para la encriptación y desencriptación: una propiedad útil para los algoritmos que han de implementarse en hardware.

El IDEA realiza la encriptación y desencriptación a una velocidad tres veces superior que la de DES.

AES. En 1997, el Instituto Nacional para los Estándares y la Tecnología (NIST) publicó una invitación para remitir propuestas de un algoritmo seguro y eficiente que seria adoptado como nuevo Estándar de Encriptación Avanzada (Advanced Encryption Standard, AES).

La comunidad de investigación sobre criptografía remitió quince algoritmos, como respuesta a la invitación inicial para el AES. Tras una intensa inspección técnica se seleccionaron cinco de ellos para la siguiente fase de evaluación. Todos los candidatos soportaban claves de 128, 192 y 256 bits, siendo todos ellos de altas prestaciones. La evaluación concluyó en mayo del año 2000, cuando se seleccionó un estándar preliminar.

Algoritmos de Clave Pública (Asimétricos):

Hasta la fecha sólo se han desarrollado unos pocos esquemas prácticos de clave pública. Dependen del uso de funciones de puerta falsa de números grandes para producir las claves. Las claves *Ke* y *Kd* son números muy grandes, y la función de encriptación realiza una operación, con una exponenciación de *M,* usando una de ellas. La desencriptación es una función similar usando la otra clave. Si la exponenciación usa una aritmética modular, puede demostrarse que el resultado es el mismo que el valor original de *M;* es decir:

$$D (Ke\ E (Ke, M)) = M$$

Un principal que desee participar en una comunicación segura con otros confecciona un par de claves K_e, *y* K_d y guarda en secreto la clave de desencriptación K_d. La clave de encriptación K_e puede publicarse para cualquiera que desee comunicar; la que podrá verse como parte de una función de encriptación de un sentido E, y la clave de desencriptación K_d es el conocimiento secreto que permite al principal *p* invertir la encriptación. Cualquiera que posea K_e puede encriptar mensajes $\{M\}_{ke}$, pero solamente el principal que posea el secreto K_d puede operar la puerta falsa.

El algoritmo RSA es ciertamente el algoritmo de clave pública más conocido que lo describiremos a continuación:

RSA. (Rivest, Snamir y Adelman) el diseño para el encriptador de clave pública, se basa en el uso del producto de dos números primos muy grandes (mayores que 10^100), la determinación de los factores primos de números tan grandes es computacionalmente imposible de calcular.

A es objeto de extensas investigaciones, no se encontrándose flaquezas en él hasta el momento, por lo que se lo usa ampliamente.

Bosquejo del método RSA

Para encontrar las claves *e, d:*

1. Elíjanse dos números primos grandes, *P* y *Q* (mayores que10^100), y fórmese

$$N = P \times Q$$

$$Z = (P- 1) \times (Q- 1)$$

2. Para *d* elíjase cualquier número primo con relación a Z (esto es, que *d* no tenga factores en común con Z).

Ilústreme los cálculos correspondientes empleando valores pequeños de *P* y *Q*

$$P = 13 \; . \; Q = 17 \rightarrow \; N = 221 \; . \; Z = 192$$

$$d = 5$$

3. Para encontrar *e* resuélvase la ecuación: $e \times d = 1 \; mod \; Z$

Esto es, *e* x *d* es el número más pequeño divisible por *d* en la serie Z + 1, 2Z + 1, 3Z + 1,

Para encriptar el texto en claro se divide en bloques iguales de longitud *k* bits donde 2 ^ k < N (el valor numérico de un bloque es siempre menor que *N;* en aplicaciones prácticas, *k* está generalmente en el rango de 512 a 1.024).

La función de encriptación de un bloque de texto en claro *M* es:

$$E' (e, N, M) = M \text{ ^e } \textbf{mod N}$$

Para desencriptar un bloque de texto encriptado c para producir el bloque de texto en claro original es:

$$D' (d, N, c) = c\text{^ } d \; mod \; N$$

S e probó que *E'* y *D'* son inversas mutuas (esto es, que $E' (D' (x)) = D' (E' (x)) = x)$ para cualquier valor *P* en el rango $0 \leq P \leq$ N.

Se podría intentar desencriptar un mensaje desconocido encriptando exhaustivamente secuencias arbitrarias de bits hasta que concuerden con el mensaje objetivo. Este ataque se conoce como *ataque de texto en claro escogido,* y se vence asegurando que todos los mensajes son más largos que la longitud de la clave, de forma que el ataque por fuerza bruta sea menos factible que un ataque directo sobre la clave.

Algoritmos de Curvas Elípticas:

Un algoritmo puede generar pares de claves pública / privada basándose en las propiedades de las curvas elípticas. Las claves de derivan de una rama diferente de las matemáticas, y a diferencia de RSA su seguridad no depende de la dificultad de la factorización de números grandes. Las claves cortas son seguras, y los requisitos de procesamiento para la encriptación y la desencriptación son menores.

Podrían ser adoptados más ampliamente en el futuro, especialmente en sistemas como los que incorporan los dispositivos móviles, con recursos de procesamiento limitados.

Protocolos Criptográficos Híbridos:

La criptografía de clave pública es apropiada para el comercio electrónico porque no hay necesidad de un mecanismo de distribución segura de claves.

La criptografía de clave pública demanda costos elevado para la encriptación incluso de mensajes de tamaño medio como los que se encuentran habitualmente en el comercio electrónico.

La solución adoptada en los sistemas distribuidos de gran escala es el empleo de un esquema de encriptación híbrido en el que se emplea criptografía de clave pública para autentificar cada parte y para encriptar un intercambio de claves secretas, que se emplearán para toda la comunicación subsiguiente.

Firmas Digitales:

Una firma digital robusta es un requisito esencial para los sistemas seguros. Se las necesita para certificar ciertos trozos de información, por ejemplo para proporcionar enunciados dignos de confianza que relacionan identidades de usuarios con claves públicas.

Las firmas manuscritas necesitan verificar que éste es:

- **Autentico:** convence al receptor de que el firmante firmó deliberadamente el documento y que la firma no ha sido alterado por nadie.
- **Infalsificable:** aporta la prueba de que el firmante firmó el documento.
- **No repudiable:** el firmante no puede negar de forma creíble que el documento fue por él.

Las propiedades de los documentos digitales almacenados en archivos o mensajes son completamente diferentes de las de los documentos en papel. Los documentos digitales son trivialmente fáciles de generar, copiar y alterar. La simple adición de la identidad del emisor en forma de cadena de texto, una fotografía o una imagen manuscrita, no tiene valor alguno como medio de verificación.

Lo que se requiere es un medio de enlazar una identidad de un firmante a la secuencia completa de bits que representa un documento. Con esto cumplimos el primer requisito anterior, de autenticidad. Como con las firmas manuscritas, la fecha del documento no está garantizada por la sola firma. El destinatario de un documento firmado sólo sabe que el documento fue firmado con anterioridad a la recepción.

Se puede considerar que un documento con firma digital es más robusto frente a la falsificación que un escrito a mano.

Firmas Digitales Con Claves Públicas:

La criptografía de clave pública se adapta particularmente bien a la generación de firmas digitales dado que es relativamente simple y no requiere ninguna comunicación entre el destinatario de un documento firmado y el firmante.

1) A genera un par de claves K_{pub} y K_{pri} y publica la calve K_{pub} situándola en una ubicación bien conocida.

2) A calcula el resumen de M,H(M) empleando una función de dispersión segura H acordada y la encripta utilizando la calve privada K_{pri} para producir la firma $S = (H(M))\ K_{pri}$

3) A envía el mensaje firmado $(M)_K = M, S$ a B

B desencripta S empleando K_{pub} y calcula el resumen de M, H(M). si concuerda, la firma es valida.

Funciones resumen. Las funciones resumen se denominan también funciones de dispersión seguras y se denotan con H(M). Éstas deben diseñarse cuidadosamente para asegurar que H(M) sea diferente de H(M') para todos los probables pares de mensajes M y M'.

Firmas Digitales Con Claves Secretas, Mac:

No hay ninguna razón técnica por la que un algoritmo de encriptación de clave secreta no pueda usarse para encriptar una firma digital, pero existen algunos problemas:

- El firmante debe conseguir que el verificador reciba la clave secreta empleada para firmar de modo seguro.
- Debe ser necesario poder verificar una firma en varios contextos diferentes y en momentos diferentes.
- El descubrimiento de una clave secreta empleada para una firma es poco deseable puesto que debilita la seguridad de las firmas realizadas con ellas, podrían falsificarse una firma por alguien que posea la clave y que no sea su propietario.

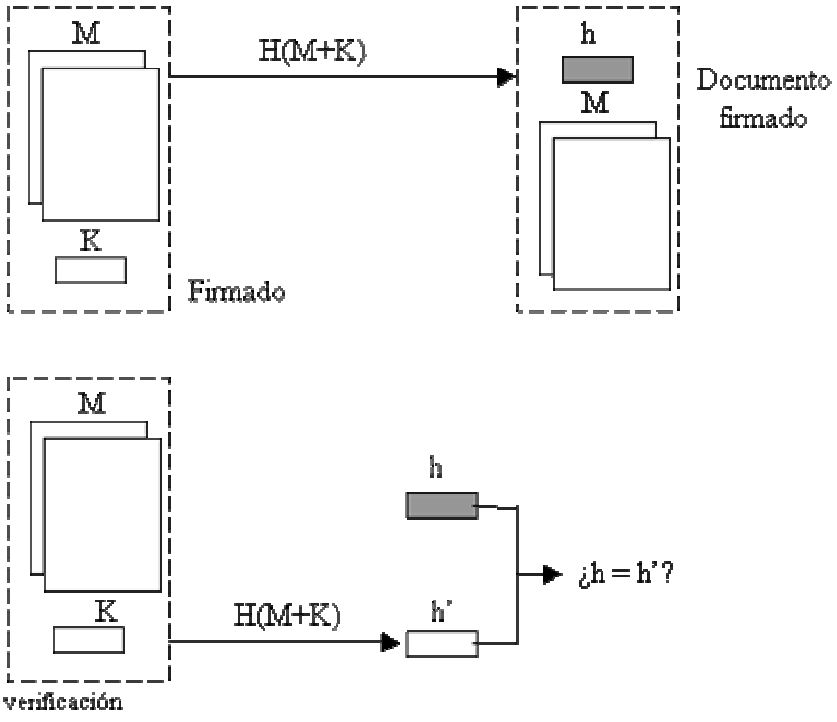
Existe una excepción cuando se utiliza un canal seguro para transmitir los mensajes desencriptados pero subsiste la necesidad de verificar la autenticidad de los mensajes. Estas firmas se denominan **códigos de autenticidad de mensajes (MAC)** para reflejar su objetivo más limitado: autentican la comunicación entre pares de principales basándose en un secreto compartido.

El siguiente método ilustra la técnica de firmado de bajo coste basada en claves secretas que da seguridad suficiente para muchos propósitos:

1) A genera una clave aleatoria K para firmar, y la distribuye utilizando canales seguros a uno o más principales que necesitan autenticar los mensajes recibidos de A.

2) Para cualquier documento M que A desee firmar. A concatena M con K , calcula el resumen del resultado : h = H (m + k) y envía el documento firmado (M)k = M , h a cualquiera que desee verificar la firma (el resumen h es un MAC). K no queda comprometida por la publicación de h, dado que la función de diversión oscurece totalmente su valor.

3) El receptor, B, concatena la clave secreta K con el documento recibido M y calcula el resumen h´= H (M + K). La firma es valida si h = h´.



Funciones De Resumen Seguro:

Hay muchas formas de producir un patrón de bits de longitud fija que caracterice un mensaje o documento de longitud arbitraria. Quizás la más simple es utilizar repetidamente la operación XOR para combinar trozos de tamaño fijo del documento fuente. Tal función se emplea a menudo en

protocolos de comunicación para producir un patrón de dispersión de longitud fija para caracterizar un mensaje con fines de detección de errores; pero este sistema es inadecuado como base de un esquema de firma digital.

Una función de resumen segura *h = H(Af)* debería poseer las siguientes propiedades:

- Dado *M,* debe ser fácil calcular *h.*
- Dado *h,* debe ser extremadamente difícil calcular *M.*
- Dado *M,* debe ser extremadamente difícil encontrar otro mensaje *Af,* tal que //(M) = *H(M').*

Tales funciones se denominan también *funciones de dispersión de un solo sentido.* La propiedad 3 requiere una característica adicional: incluso aunque conozcamos que la aplicación de la función de dispersión no dé un valor único (porque el resumen es una transformación de reducción de información), necesitamos asegurarnos que un atacante, dado un mensaje *M* que produce un valor *h,* no pueda descubrir otro mensaje *M'* que también produzca *h*. Si un atacante *pudiera* hacer esto, podría falsificar un documento firmado *M'* sin tener que conocer la clave de la firma, simplemente copiando la firma del documento firmado *M* y añadiéndosela a *M'.*

Desde luego, el conjunto de mensajes que se aplican sobre el mismo valor está restringido y el atacante lo tendría difícil para producir una falsificación con sentido, pero con paciencia podría hacerse, de modo que hay que prevenirlo. La posibilidad de conseguirlo aumenta considerablemente si se emplea el denominado *(ataque del cumpleaños (birdrthday attack):*

Si nuestros valores de dispersión tienen un tamaño de 64 bits, sólo necesitamos 2" versiones de *M* y *M'* por término medio. Es demasiado poco para quedar conforme. Al menos necesitamos que nuestros valores de dispersión tengan 128 bits para alejar el peligro de este ataque.

El ataque descansa sobre la paradoja estadística conocida como la paradoja del cumpleaños: la probabilidad de encontrar un par idéntico en un conjunto dado es mucho mayor que la de encontrar la pareja para un individuo dado.

Para satisfacer las propiedades indicadas anteriormente hay que diseñar la función de resumen segura cuidadosamente. Las operaciones al nivel de bit empleadas y su secuenciación son similares a las que se encuentran en la criptografía simétrica, pero en este caso las operaciones no tienen por qué preservar la información, dado que la función no tiene por qué ser reversible en absoluto. De modo que una función de resumen segura puede emplear cualquier operación aritmética y lógica al nivel de bit. La longitud del texto fuente se incluye habitualmente entre los datos resumidos.

Dos funciones de resumen ampliamente utilizadas son el algoritmo MD5 (llamado así porque es el quinto de una serie de algoritmos de resumen desarrollados por Ron Rivest) y el SHA *(Secure Hash Algoritm)* adoptado para su estandarización por el NIST.

MD5. El algoritmo MD5 emplea cuatro vueltas, cada una aplicando una de cuatro funciones no lineales a cada uno de los dieciséis segmentos de 32 bits de un bloque de texto fuente de 512 bits. El resultado es un resumen de 128 bits. MD5 es uno de los algoritmos más eficientes de que se dispone hoy en día.

SHA. Es un algoritmo que produce un resumen de 160 bits. Se basa en el algoritmo de Rivest MD4 (similar a MD5) con algunas operaciones adicionales. Es sustancialmente más lento que MD5, pero el resumen de 160 bits ofrece una mayor seguridad contra los ataques por fuerza bruta y del cumpleaños.

Empleo de un algoritmo de encriptación para obtener un resumen. Es posible utilizar un algoritmo de encriptación simétrico para producir un resumen seguro. En este caso, la clave debería ser pública de modo que el algoritmo pudiera ser aplicado por cualquiera que deseara verificar una firma digital. El algoritmo de encriptado se emplea en modo CBC, y el resumen es el resultado de combinar el penúltimo valor CBC con el bloque final encriptado.

Estándares de Certificación y Autoridades de Certificación:

X.509 es el formato estándar de certificación más ampliamente usado [CCITT 1988b] es parte del estándar X.500 para la construcción de directorios globales de nombres y atributos.

La estructura y contenido de un certificado X.509 se enlaza una clave pública a una entidad con nombre denominada *sujeto.* El enlace está en la firma, que es emitida por otra entidad denominada *emisora.* El certificado tiene un *período de validez,* que es definido mediante dos fechas. Los contenidos etiquetados como < Nombre Diferenciado > se refieren al nombre de una persona, organización o cualquier otra entidad junto con suficiente información contextual para darlo por único.

Formato de Certificado X.509.

Sujeto: Nombre Diferenciado.

Clave pública *Emisor:* Nombre Diferenciado.

Firma *Período de validez:* Fecha inicio. Fecha expiración.

Información administrativa: Versión, Número de Serie.

Información añadida: Etc.

Procedimiento de verificación de dos pasos para cualquier certificado X.509:

1. Obtener el certificado de clave pública del emisor (una autoridad de certificación) desde una fuente fiable.

2. Validar la firma.

Aproximación SPKI. La aproximación X.509 se basa en la unicidad global de los nombres diferenciados. Ya se ha indicado que esta meta es inalcanzable dado que no refleja la realidad actual legal y de la práctica comercial en la que no se presupone la unicidad de la identidad de los individuos si no se hace referencia a otras personas y organizaciones. Esto se pone en evidencia, por ejemplo, en el uso de un permiso de conducción o una carta de un banco para autenticar el nombre y dirección de un individuo (un nombre a secas puede estar repetido si consideramos la población mundial). Esto nos lleva a cadenas de verificación más largas, puesto que hay muchos emisores posibles de certificados de clave pública, y sus firmas deben validarse a través de una cadena de verificación que nos lleve a alguien conocido en quien confíe el principal que realiza la verificación. A pesar de ello la verificación resultante suena más convincente, y muchos de los pasos de esta cadena pueden guardarse para acortar el proceso en otras ocasiones.

Los argumentos de arriba son la base de las propuestas recientemente desarrolladas para la Infraestructura de Clave Pública Simple *(Simple Public-key Infrastructure,* SPKI) Éste es un esquema para la creación y administración de conjuntos de certificados públicos. Posibilita el procesamiento de cadenas de certificaciones empleando inferencia lógica para producir certificados derivados.

Prestaciones de los Algoritmos Cripotográficos:

	Tamaño de clave / tamaño de dispersión(bits)	Velocidad extrapolada (kbytes/sec)	**PRB optimizado (kbytes/s)**
TEA	**128**	**700**	**----**
DES	**56**	**350**	**7746**
Triple-DES	**112**	**120**	**2842**
IDEA	**128**	**700**	**4469**
RSA	**512**	**7**	**----**
RSA	**2048**	**1**	**----**
MD5	**128**	**1740**	**62425**
SHA	**160**	**750**	**25162**

Prestaciones de los algoritmos de encriptación y resúmenes seguros

En el cuadro que vemos arriba se puede observar el tamaño de la clave y la velocidad de los algoritmos de encriptación y las funciones de resumen seguras.

Se dan dos medidas de seguridad:

1. ***Velocidad Extrapolada:*** basada en cifras de Schneier excepto para TEA, que se basan en Wheeler y Needham.
2. ***PRB optimizado:*** basada en la publicidad de Preneel y otros.

En ambos casos se han ajustado las cifras teniendo en cuenta los avances de los procesadores. Los números pueden tomarse como estimaciones aproximadas sobre un procesador Pentium II a 330 Mhz.

La longitud de las claves encriptadas proporciona una indicación del coste computacional de un ataque por fuerza bruta sobre la clave. La autentica resistencia de los algoritmos criptográficos es mucho mas difícil de evaluar.

La diferencia entre las dos columnas de prestaciones proviene del hecho de que las cifras dada por Schneier están basadas en el código C, sin ningún intento de optimizar el código. Y las del PRB son el resultado de un esfuerzo para producir implementaciones propietarias, optimizadas, de los algoritmos en el lenguaje ensamblador. Preneel y otros presentan una discusión útil sobre la resistencia y prestaciones de los algoritmos simétricos.

Aplicaciones de la Criptografía y Obstáculos Políticos:

Los algoritmos sobresalen alrededor de los años ochenta y noventa, cuando comienzan a utilizarse las redes de computadores con objetivos comerciales, por esto se hace evidente las necesidades de seguridad naciendo así la criptografía (arte de escribir con clave o de modo enigmático).

Pero esta necesidad emergente trajo aparejado el surgimiento de grandes obstáculos políticos como lo que impuso la Agencia de Seguridad Nacional *(National Security Agency,* NSA) al precisar que este mecanismo criptográfico al estar disponible para otras naciones, volvía vulnerable cualquier comunicación secreta por objetivos de inteligencia militar; y otra de las oposiciones también dependiente del gobierno de los EE.UU. mas precisamente por la Oficina Federal de Investigación *(Federal Bureau of Investigation,* FBI) trataba sobre asegurar que sus agentes pudieran tener acceso privilegiado a las claves empleadas por todas las organizaciones privadas e individuos de los EE.UU. con el objetivo de poder ejercer la ley.

Como conclusión el software criptográfico se clasificó como armamento y quedaba sujeto a estrictas restricciones de exportación lo que llevo a grandes compañías de software de los EE.UU. a protestar ante los mismos ya que estas restricciones inhibían la exportación de cierto software como los visualizadores, logrando reformar las restricciones de modo que se permitía la exportación de código que utilizara claves de no más de 40 bits.

La situación actual es que el software que implementa la mayoría de los algoritmos criptográficos ha estado disponible por todo el mundo durante años, impreso y en línea, en versiones comerciales y gratuitas Por ejemplo el programa llamado PGP *(Pretty Good Privacy)*, que es parte de una campaña técnica y política para asegurar que la disponibilidad de métodos criptográficos no se encuentre controlada por el gobierno de los EE.UU. (otorgaba cierto grado de privacidad al usuario).

El gobierno de los EE.UU. tomo conciencia del daño que causaba (con las políticas N.S.A.) a la industria de computadores y en enero del año 2000 introduce una nueva política que permitió a los vendedores de software de los EE.UU. exportar productos software que incorporara encriptación de hasta 64 bits y hasta 1.024 en el caso de claves públicas para su uso en firmas e intercambios de claves.

Casos de Estudio: Needham-Schroeder, Kerberos, SSL y Millicent

Los protocolos de autenticación publicados Needham y Schroeder [1978] son el núcleo de muchas técnicas de seguridad siendo uno de los más importantes entre sus sistemas de protocolo el Kerberos que fue diseñado para proporcionar autenticación entre clientes y servidores en redes.

Existen dos casos de estudio que describen protocolos de seguridad en el nivel de aplicación importante para el comercio electrónico:

- El de Capa de Sockets Segura *(Secure Sockets Layer,* SSL) diseñado para cumplimentar la necesidad de transacciones seguras (soportada por la mayoría de los visualizadores y servidores web)
- El Millicent, diseñado específicamente para las necesidades de un método de pago para micro-transacciones.

El Protocolo de Autenticación de Needham y Schroeder:

Fue empleado debido a una necesidad urgente de mejores formas de administrar la seguridad de las redes locales.

En la misma publicación, Needham y Schroeder también confeccionan un protocolo basado en el uso de claves públicas para autenticación y distribución de claves y que no depende de la existencia de servidores de claves seguros, resultando el más adecuado para su empleo en redes con muchos dominios de administración independientes, como Internet.

La propuesta autentica de estos dos creadores se basa en un *servidor de autenticación* que proporciona claves secretas a sus clientes en forma segura.

El trabajo del servidor de autenticación es proporcionar una forma segura por la que pares de procesos obtengan claves compartidas. Para hacer esto, debe comunicarse con sus clientes usando mensajes encriptados.

Needham y Schroeder con claves secretas. El protocolo se describe para dos procesos arbitrarios A y B, pero en sistemas cliente-servidor, A es cualquier cliente que inicie una secuencia de solicitudes hacia algún servidor B. La clave le viene dada a A de dos formas, una que A puede usar para encriptar los mensajes que envíe a B y otra que pueda transmitir de modo seguro a B. (La última se encuentra encriptada en una clave que es conocida por B pero no por A, de modo que B puede desencriptarla y no se compromete durante la transmisión.)

El servidor de autenticación S mantiene una tabla que contiene un nombre y una clave secreta para cada principal conocido por el sistema.

Nunca se muestra a terceras partes y se transmite por la red como máximo una sola vez, cuando se genera.

Una clave secreta sería el equivalente de la contraseña *(password)* que se emplea para autenticar usuarios en los sistemas centralizados. Para los principales humanos, el nombre conocido por el servicio de autenticación es su <<nombre de usuario>> y la clave secreta es su password.

El protocolo se basa en la generación y transmisión de tickets por el servidor de autenticación, que son mensaje encriptado que contiene una clave para su uso en la comunicación.

El servidor de autenticación es S. *NA* Y *NB* son *ocasiones.* Una ocasión es un valor entero que se añade a un mensaje para demostrar su frescura. Las ocasiones se utilizan una sola vez y se generan bajo demanda únicos mensajes que han sido enviados conteniendo *KAB* fueron encriptados en la clave secreta de A o en la de B.

Existe una debilidad en este protocolo, ya que un intruso puede obtener información de un descuido y utilizarla para iniciar un intercambio posterior con B, suplantando a A, para que no ocurra este ataque debe comprometerse un antiguo valor de KAB, en terminología de hoy en día. Esta posibilidad no fue incluida por Needham y Schroeder en su lista de amenaza cuando debían hacerlo, pero esta debilidad se puede remediar añadiendo una ocasión o una marca temporal. Solución adoptada por Kerberos.

ENCABEZADO	MENSAJE	ANOTACIONES
1. A → S:	**A, B, N_A**	**A solicita a S una clave para comunicarse con B.**
2. S → A:	**$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{KB}\}_{KA}$**	**S retorna un mensaje encriptado en la clave secreta de A, que contiene una clave nueva K_{AB}, y un <<ticket>> encriptado en la clave secreta de B. La ocasión N_A demuestra que el mensaje fue enviado en respuesta a la anterior. A cree que S envió el mensaje porque solo S conoce la clave secreta de A.**
		A envía el <<ticket>> a B.
3. A → B:	**$\{K_{AB}, A\}_{KB}$**	**B desencripta el ticket y usa la nueva clave KAB para encriptar otra ocasión NB.**
4. B → A:	**$\{N_B\}_{KAB}$**	**A demuestra a B que fue el emisor del mensaje anterior devolviendo una transformación acordada sobre NB.**
5. A → B:	**$\{N_B - 1\}_{KAB}$**	

Protocolo de autenticación de clave secreta de Needham – Schroeder

Kerberos:

Kerberos se desarrolló en el MIT en los años ochenta, para proporcionar un catalogo de medios de autenticación y seguridad para su uso en la red de computación del campus en el MIT y otras intranets. Ha soportado varias revisiones y mejoras, de las cuales surgió entre otras la versión 5 de Kerberos (1994), la cual esta en vía de ser un estándar Internet y es usada hoy en día por muchas compañías y universidades. El código fuente de la implementación de Kerberos está disponible desde el MIT, en el Entorno de Computación Distribuida de OSF y en el sistema operativo Windows 2000 como servicio de autenticación como servicio de autenticación por defecto. Se ha propuesto su

extensión para incorporar el uso de certificados de clave pública para la autenticación inicial de principales.

Kerberos trata con tres clases de objetos de segundad:

***Ticket*:** una palabra enviada a un cliente por el servicio de concesión de tickets para su presentación a un servidor particular, y que verifica que el emisor se ha autenticado recientemente frente a Kerberos. Los tickets incluyen un tiempo de expiración y una clave de sesión generada en este momento para su uso por el cliente y el servidor.

***Autenticación*:** una palabra construida por un cliente y enviada al servidor para demostrar la identidad del usuario y la actualidad de cualquier comunicación con un servidor. Un autenticador sólo puede usarse una vez. Éste contiene el nombre del cliente y una marca temporal y se encripta con la clave de sesión apropiada.

Clave de sesión: una clave secreta generada aleatoriamente por Kerberos y enviada a un cliente para su uso en la comunicación con un servidor particular; la clave de sesión se emplea para encriptar la comunicación con aquellos servidores que la pidan y para encriptar todos los autenticadores.

Los procesos clientes deben poseer un ticket y una clave de sesión para cada servidor que empleen. La mayoría de los tickets se conceden a los clientes con un periodo de vida de varias horas, de modo que puedan utilizarse para interactuar con un servidor particular hasta que expiran.

Un servidor Kerberos es conocido como Centro de Distribución de Claves, ofrece un Servicio de Autenticación y un Servicio de Concesión de Tickets. Al presentarse al sistema, los usuarios son autenticados por el SA, y el proceso del cliente que actúa en representación del usuario recibe un ticket de concesión de tickets y una clave de sesión para comunicarse con el SCT. Posteriormente el proceso cliente original puede usar el ticket de concesión de tickets y las claves de sesión para los servicios especificados desde el SCT.

El Kerberos tiene muy en cuenta los valores del tiempo (fecha y hora) usando como ocasiones. Esto sirve a dos objetivos:

- Protegerse de la repetición de mensajes antiguos o rechazar los viejos tickets encontrados en algún lugar de la memoria de máquina.
- Aplicar un tiempo de vida a los tickets, permitiendo que el sistema revoque los derechos de acceso de usuarios cuando, por ejemplo, deja de ser usuarios autorizados del sistema.

Aplicación de Kerberos:

Kerberos se desarrolló para su uso en el proyecto Atenía en el MIT; una infraestructura de computación en red por todo el campus. El entorno es tal que no es posible presuponer ni la confianza en los clientes ni la seguridad de la red y las máquinas que ofrecen servicios de red; por ejemplo, las estaciones de

trabajo no están protegidas por la instalación de software desarrollado por los usuarios, y las máquinas servidoras (diferentes al servidor Kerberos) no están aseguradas contra una interferencia física con su configuración software.

Kerberos proporciona virtualmente toda la seguridad del sistema Atenía. Se emplea para autenticar usuarios y otros principales. La mayoría de los servidores que operan en la red solicitan un ticket de cada cliente al comienzo de cada interacción cliente-servidor. En éstos se incluyen, el almacenamiento de archivos, correo electrónico, etc. Las claves de acceso de los usuarios son conocidas solo por cada uno de ellos y por el servicio de autenticación Kerberos. Los servicios tienen claves secretas conocidas solo para Kerberos y los servidores que proporcionan el servicio.

Entrada en el Sistema con Kerberos:

Cuando un usuario entra en una estación de trabajo, el programa de bienvenida envía el nombre del usuario al servicio de autenticación de Kerberos. Si el usuario es conocido al servicio de autenticación éste responde con una clave de sesión y una ocasión encriptada en una clave de acceso y un ticket para el SCT. El programa de bienvenida, ahora, intenta desencriptar la clave de sesión y la ocasión empleando la clave de acceso que deberá introducir el usuario. Si la clave de acceso es correcta, el programa de bienvenida comprueba la ocasión y almacena la clave de sesión junto al ticket para ser usado cuando se comunique con el SCT. En este momento, el ticket ya sirve para autenticar al usuario; se inicia la sesión del usuario en la estación del sistema. La clave del usuario nunca se expone en la red y se borra de la memoria tan pronto como se ha empleado.

Acceso a los Servidores en Kerberos:

Cuando un programa en ejecución solicita el acceso a un nuevo servicio, pide un ticket para este servicio al servicio de concesión de tickets. Por ejemplo, cuando un usuario desea acceder a un computador remoto, el mandato *rlogin* de la estación del usuario obtendrá un ticket del servicio de concesión de ticket de Kerberos para acceder al servicio de red *rlogind.* El programa del mandato *rlogin* envía el ticket, junto con un nuevo autenticador. El programa *rlogind* desencripta el ticket con la clave secreta del servicio *rlogin* y comprueba la validez del ticket.

Implementación de Kerberos: Se emplea el algoritmo de encriptación DES, pero se implementa en un módulo separado que puede reemplazarse fácilmente.

El servicio Kerberos es escalable; el mundo se divide en dominios de autoridades de autenticación separados, denominados *esferas (realms),* cada una con su propio servidor Kerberos. La mayoría de los principales están registrados en una sola esfera, pero los servidores de concesión de ticket de Kerberos están registrados en todas las esferas. Dentro de un dominio, puede haber varios servidores de autenticación, cada uno de los cuales tiene copias de la misma base de datos de autenticación. La base de datos de autenticación

se replica mediante una simple técnica maestro-esclavo. Las actualizaciones se aplican a la copia maestra por un único servicio de Administración de Base de Datos Kerberos *(Kerberos Datábase Management,* KDBM) que se ejecuta sólo en la máquina maestra. El KDBM maneja las solicitudes de cambio de claves de acceso de los usuarios y las peticiones de los administradores para añadir o borrar principales y para cambiar sus claves de acceso.

Para hacer transparente este esquema a los usuarios, el tiempo de vida de los tickets TGS debería ser tan largo como la sesión de usuario más larga posible, dado que el uso de un ticket expirado trae consigo el rechazo de las solicitudes de servicio.

Seguridad de Transacciones Electrónicas con Sockets Seguros:

SSL está soportada en la mayoría de los visualizadores y se usa ampliamente en el comercio en Internet.

Sus características más importantes son:

***Encriptacion y algoritmos de encriptación negociables*:** SSL se ha diseñado para que puedan negociarse los algoritmos de encriptación y autenticación entre los procesos al principio y al final de la conexión durante el saludo *(handsake)* inicial. Pudiera ocurrir que no tuvieran suficientes algoritmos en común, y en este caso el intento de conexión fallaría.

***Autoarranque de la comunicación segura*:** Para cumplir con la necesidad de comunicación segura sin una negociación previa y sin ayuda de tercera partes, el canal seguro emplea una comunicación no encriptada en los intercambios iniciales después criptografía de clave pública y finalmente se conmuta a criptografía de clave privada una vez que se ha establecido una clave de secreta compartida El cambio a cada etapa es opcional y viene precedido por una negociación

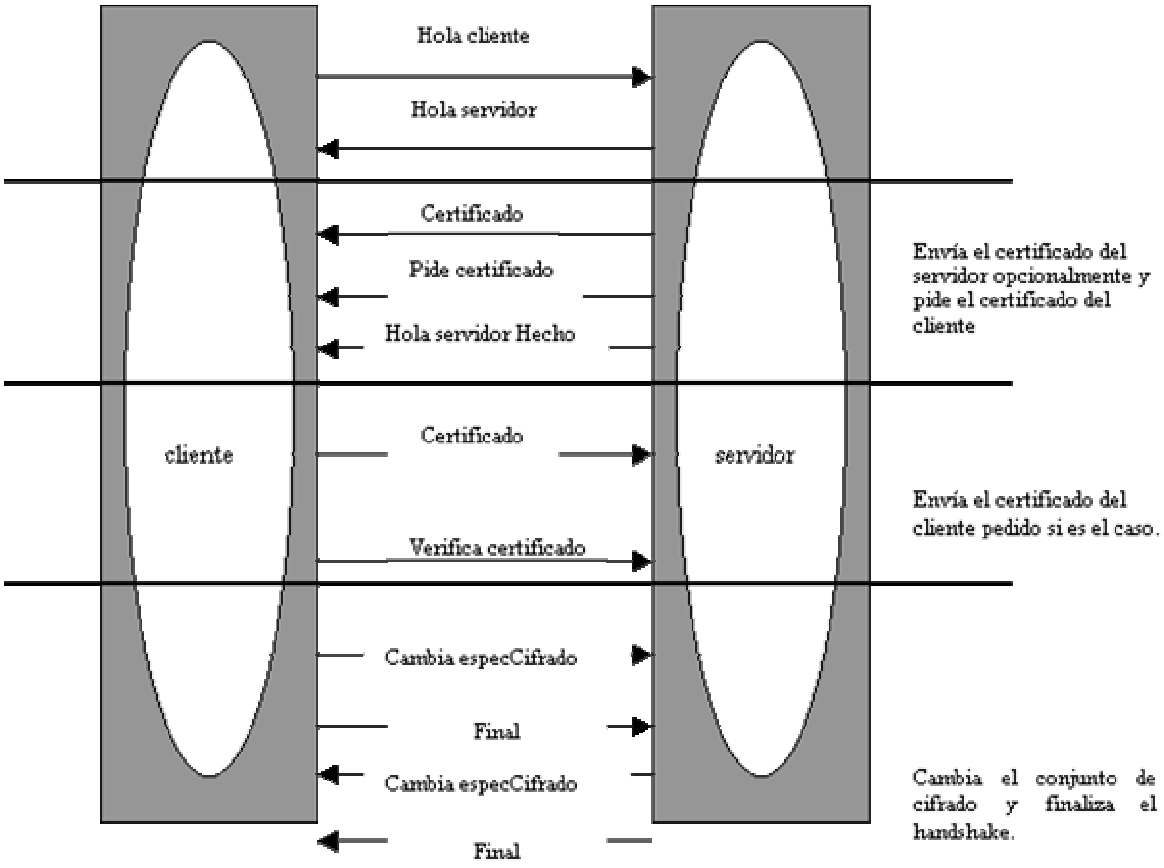
Consiguientemente el canal seguro es completamente configurable, permitiendo que la comunicación en ambas direcciones este incriptada y autenticada

SSL consta de dos capas, una capa de Protocolo de Registro SSL, que implementa un canal seguro encriptando y autenticando mensajes transmitidos a través de cualquier protocolo orientado a conexión, y una capa de handsake, conteniendo el protocolo de handsake y otros dos protocolos relacionados que establecen y mantienen una sesión SSL (esto es, un canal seguro) entre un cliente y un servidor.

El protocolo de registro de SSL es una capa de nivel de sesión; puede emplearse para transportar datos del nivel de aplicación de modo transparente entre un par de procesos mientras garantiza su privacidad, integridad y autenticidad. En SSL hay opciones para que los participantes de la comunicación escojan si desplegar o no la desencriptación y autenticación de

los mensajes en cada dirección. A cada sesión segura se le da un identificador y cada participante puede almacenar los identificadores de sesión en una caché para su uso subsiguiente, evitando la sobrecarga de establecer una nueva sesión cuando se requiere otra sesión segura con el mismo compañero.

SSL se usa ampliamente para añadir una capa de comunicación segura a los protocolos del nivel de aplicación existentes. Su uso más extendido es, probablemente, la seguridad de las interacciones HTTP. El uso del prefijo de



protocolo *https:* en un URL inicia el establecimiento de un canal seguro SSL entre un navegador y un servidor web. También se ha empleado para proporcionar implementaciones seguras de Telnet, FTP y muchos otros protocolos de aplicación. SSL se utiliza en aplicaciones que requieran canales seguros, con interfaces de programación para CORBA y Java.

El handshake se realiza sobre una conexión existente. Comienza con texto en claro y establece una sesión SSL intercambiando las opciones y parámetros acordados que se necesitan para realizar la encriptación y la autenticación. La secuencia de handshake varía dependiendo de si se requiere, o no, autenticación de cliente y servidor.

Protocolo de Handshake de SSL

SSL soporta una variedad de opciones de funciones criptográficas. El nombre colectivo que recibe es *catálogo de cifrado.* Un catálogo de cifrado incluye una elección única para cada una de las características.

Se suelen encontrar ya cargadas cierta variedad usual de catálogos de cifrado, con identificadores estándar tanto en el cliente como en el servidor. Durante el handshake, el servidor ofrece al cliente una lista de los identificadores de catálogos de cifrado disponibles, y el cliente responde seleccionando uno de ellos. En esta etapa también deben acordar (opcionalmente) un método de compresión y un valor de arranque aleatorio para las funciones de encriptación de bloque CBC.

El Protocolo de registro de seguridad de SSL muestra la operación del protocolo de registro. Un mensaje a transmitir se fragmenta inicialmente en bloques de un tamaño manejable, y opcionalmente después se comprimen. La compresión no es estrictamente una característica de la comunicación segura, pero se incorpora aquí para sacar provecho del procesamiento masivo de datos que se genera en los algoritmos de encriptación y firma digital. En otras palabras, dentro de la capa SSL de registro se puede establecer conjuntamente un flujo de transformaciones sobre los datos mucho más eficiente que si se realizaran individualmente.

Las transformaciones de encriptado y autenticación de mensajes (MAC) lanzan los algoritmos especificados en el catálogo de cifrado acordado. Finalmente, el bloque firmado y encriptado se transmite al destinatario mediante la conexión TCP asociada, donde se invierten las transformaciones para producir el bloque de datos originales.

SSL proporciona una implementación práctica de un esquema de encriptación híbrido con autenticación e intercambio de claves basado en claves públicas. Dado que los cifradores se negocian en la etapa de handshake, no depende de la disponibilidad de ningún algoritmo en particular. Tampoco depende de servicios de seguridad en el momento de establecer la sesión. El único requisito es que los certificados de clave pública enviados por una autoridad sean reconocidos por ambas partes.

Conclusión:

Los ataques a la seguridad son partes de la realidad de los sistemas distribuidos. Es esencial proteger los canales e interfaces de comunicaciones de cualquier sistema que trate con información que sea susceptible de ser atacada.

- Los sistemas distribuidos abarcan una cantidad de aspectos considerables, por lo cual su desarrollo implica mucha complejidad.
- Existen ciertos aspectos que requieren extremo cuidado al desarrollarse e implantarse como el manejo de fallos, el control de la concurrencia, etc.
- Existen muchos temas de investigación relacionados con los sistemas distribuidos.
- Se nota también que muchas tecnologías están en constante desarrollo y maduración, lo cual implica un minucioso estudio previo de muchos factores antes de apostar por alguna tecnología en especial.

Entonces podemos decir que más que un problema de tecnología, la seguridad en la transmisión de la información por la Red se debe a la falta de cultura de las organizaciones y de las personas que la integran. El eslabón más débil de esta cadena en la seguridad la constituye el humano y no el tecnológico, lo cual destaca la importancia de tener una cultura de seguridad, porque no existe en muchas empresas un responsable de la seguridad. A todos los usuarios se les debe divulgar las políticas de seguridad, además de hacer constantes auditorias para controlar que sean las adecuadas al momento que vive la empresa.

Lo que se necesita no es solamente prevenir un ataque en la seguridad, sino ser capaces de detectar y responder a esta agresión mientras ocurre y reaccionar ante la misma. Es importante destacar que no existe un control de seguridad único, sino que las empresas deben contar con diversas capas de seguridad en todos los niveles de su información para poder así detectar el problema en algunos de estos puntos antes de que llegue a la información crucial.

Bibliografía:

George Coulouris. *Sistemas Distribuidos*. Tercera Edición. Addison Wesley. Madrid. 2001.

Coulouris Dolumore. Sistemas Distribuidos. Capítulo 7. Páginas (235 - 289). 3º Edición.

Roger S. Presuman. Ingeniería de Software. Quinta Edición. McGraw-Hill Interamericana. Madrid. 2002.

David Zurdo Saiz, Alejandro Sicilia Burgoa, Fernando Acevedo Quero. *Guía Rápida de Internet*. Paraninfo. Madrid. 1997.

http://fmc.axarnet.es/redes/tema_04.htm (Sistemas Operativos).

http://www.fortunecity.es/sopa/chinchulines/812/informacion/noscs.htm (Sistemas Operativos).

http://dmi.uib.es/~bbuades/sistdistr/sld007.htm (Sistemas Distribuidos).

http://members.fortunecity.es/lrmdl/SO7.htm#VSDRC (Sistemas Distribuidos).

http://sacbeob.8m.com/tutoriales/bddistribuidas/ (Base de Datos Distribuidas).

http://pdf.rincondelvago.com/bases-de-datos-distribuidas.html (Base de Datos Distribuidas).

http://www-lt.ls.fi.upm.es/sistemas_dist/Introduccion.pdf (Sistemas Distribuidos).