UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SEGURIDAD DE REDES Y ENCRIPTACIÓN DE DATOS

INFORME DE SUFICIENCIA

PARA OPTAR EL TITULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

YOVANA MERY VIVAS INGA

PROMOCIÓN
1999-I
LIMA-PERÚ
2002

Dedico este trabajo a:
Mis padres, por su ejemplo, inspiración plena
de lucha y sacrificio,
Y a mis hermanas, por hacer en mi vida un
mar de alegrías

SEGURIDAD DE REDES Y ENCRIPTACIÓN DE DATOS

SUMARIO

El objetivo de este informe es dar a conocer el impacto de la seguridad sobre las plataformas de información electrónica, así como también informar acerca de las formas que existen para prevenir los ataques implementando las herramientas necesarias sobre un sistema o una red de datos.

En el capitulo I se ofrece una visión general de las formas de ataques a diferentes niveles de la comunicación, a su vez proporciona las opciones para evitar ser victimas potenciales.

El capitulo II se introduce la criptografía, define varios términos y se describen protocolos criptográficos. Estos protocolos van desde los más simples hasta los complejos.

El capitulo III se discute el manejo de claves, como generarlas, guardarlas y disponer de ellas. El manejo de la clave es la parte más difícil de la criptografía. Se discute diferentes tipos de algoritmos y se muestra asimismo las ventajas y desventajas de cada una.

El capitulo IV proporciona el respaldo matemático. Se detalla el funcionamiento de los algoritmos más conocidos, su seguridad y algunas variantes. También se discuten las funciones de un solo sentido.

El capítulo V discute algunas de las implementaciones de los algoritmos y protocolos detallados en capítulos anteriores y a su vez muestra algunos detalles legales concernientes a la criptografía.

PRÓLOGO

Cuando un árbol cae en el bosque y no hay nadie cercano que lo escuche, de todos modos esta caída produce un sonido. Pero si una red de computadoras tiene una vulnerabilidad de seguridad y nadie sabe de la misma, ¿es insegura? Es insegura para aquellos que conocen la vulnerabilidad. Si nadie esta informado acerca de la vulnerabilidad, es decir, no ha sido literalmente descubierta, la red es segura. Si una persona descubre la vulnerabilidad, la red será insegura solo para esta persona y segura para cualquier otro individuo. En conclusión, la vulnerabilidad siempre existe, aun cuando nadie la conoce. Cuando se publica la vulnerabilidad no se hace insegura a la red, quizás se incremente la probabilidad que exista un ataque contra la vulnerabilidad pero también se incrementa la probabilidad de que los administradores de sistemas se puedan proteger de las mismas. En este sentido la criptografía se utiliza para de alguna manera esconder la información transmitida. La literatura de la criptografía tiene una

historia curiosa. El secreto y la confidencialidad siempre ha jugado un rol central, pero hasta la primera guerra mundial, los desarrollos importantes en este campo aparecían muy esporádicamente. A fines de 1918 apareció uno de los papers sobre criptoanálisis más influyentes del siglo veinte, la monografía de William F. Friedman: El índice de la coincidencia y sus aplicaciones en criptografía. Después de la guerra sin embargo, las organizaciones militares de las grandes potencias trabajaban en secreto, en este tiempo se hicieron avances fundamentales en criptografía. En nuestros días la criptografía se ve limitada quizás por la falta de procesamiento suficiente de las computadoras, pero se ha convertido en una pieza fundamental dentro de la transmisión de datos.

CAPÍTULO I

SEGURIDAD: GENERALIDADES

1.1 RECOPILACIÓN DE INFORMACIÓN DE SISTEMAS VULNERABLES

En el campo de la computación mas conocido como "footprinting", es cuando el atacante se informa acerca de los métodos de seguridad utilizado por la organización que tenga por objeto para su ataque. Hay varias formas de efectúar footprinting pero la mas común consiste en que el atacante se informe sobre tecnologías utilizadas por la organización que tienen relación con números IPs, servidores de correo, acceso remoto, intranet, extranet.

1.1.1 Recopilación de datos de victimas potenciales mediante Internet

A continuación algunos de los puntos que se recoleccionan como data cuando se hace footprinting, a nivel de internet: nombre del dominio, bloques de red, especificación de direcciónes IPs de sistemas que se pueden conocer identificando los servicios TCP y

UDP, los tipos de servidores y equipos que trabajan en la red (SPARC o Windows), mecanismos de cotrol de acceso (pueden ser listas de acceso), sistemas IDS, sistemas de numeración. Hay varias maneras de hacer el seguimiento y obtener información con respecto a una determinada organización y/o dominio, estos se ejecutan mediante varios pasos, como se detallá a continuación:

- Determinar el objetivo del rastreo y hasta que nivel de inseguridad se puede acceder.

- Informarse sobre la enumeración de red, como se sabe cada red tiene determinados IPs ya sean estos públicos o privados. Esto se puede hacer utilizando requerimientos (queries) básicos (como el "whois") desde una máquina con plataforma unix o windows.

- A continuación hay que averiguar el estado del DNS de la corporación o de su operador local.

1.2 RASTREO DE VULNERABILIDADES A SISTEMAS INDEPENDIENTES O REDES

Con footprinting lo que se hace básicamente es tomar conocimiento sobre la arquitectura de red y rangos de números IPs. Sin embargo si se quiere ir mas allá de eso y saber cuales son los sistemas que están funcionando y además que tipos de servicios ofrecen las redes a las que se quiere accesar se debe hacer el "scanning", ya que no es seguro que si bien hay un número IP publicado en el DNS se puede acceder a este a través de

internet. En conclusión, Scanning es el segundo paso en cuanto a reunión de información se refiere.

1.2.1 Barridos de Ping a la red

Uno de los pasos mas básicos es hacer barridos de ping a un determinado rango de IPs para determinar especificamente que sistemas están funcionando.

UNO (ICMP ECHO) → llega el ICMP ECHO

Espera ← envía el ICMP ECHO REPLY

En internet se tienen disponibles, en versión freeware, varias herramientas para ejecutar ping-sweeps, tanto en Windows como en UNIX, como:

- fping
- gping
- nmap
- WS_PingProPack

Si bien efectúar un ping a cada numero IP de una red es confiable, si tenemos una red con gran cantidad de sistemas este método no es práctico.

1.2.2 Pedidos de paquetes de protocolo ICMP

Los barridos de ping que se vió en el punto anterior es solo una de las formas de informarse en cuanto se refiere a información ICMP, se puede reunir todo tipo de información valiosa del sistema tan solo enviándole un paquete ICMP. Por ejemplo, con la herramienta de unix "icmpquery" o "icmpush" se puede solicitar la hora a los sistemas. O también se puede pedir la mascara de red (netmask) de un sistema en particular tan solo con el mensaje ICMP tipo 17 (ADDRESS MASK REQUEST), es importante conocer el netmask de una determinada red ya que esto nos permite determinar cuantas subnets se están usanndo, y se pueden orientar los ataques a todos los subnets que estén siendo usados y evitar los broadcasts. El modo de prevenir de estos queries es restringiendo los tipos de ICMP que se dan como información en los routers de borde de la red en cuestión. Por lo menos se debe restringir los TIMESTAMP (ICMP type 13) y ADDRESS MASK (ICMP type 17).

1.2.3 Prueba de respuesta de los puertos de una red

Hasta ahora se han identificado sistemas que están activos tan solo con barridos de ping ICMP o TCP, pero también se puede hacer un "port scanning" de cada uno de estos sistemas. Port scanning es el proceso de conectarse a puertos TCP o UDP en sistemas individuales para determinar que servicios están corriendo o si están en un estado de LISTENING. Cuando se identifican los puertos que estén en listening podemos deducir que tipo de servidor es y que aplicaciones corre determinado sistema. Los servicios activos que están en "listening" podrían dar acceso no autorizado cuando tengan

vulnerabilidades o estén corriendo aplicaciones que no son seguras. A continuacion se mostraran las herramientas más comunes de portscanning y técnicas utilizadas que muestran como evitarlos.

1.2.4 Tipos de pruebas de servicio para identificar vulnerabilidades

Existen varias técnicas para probar los servicios activos o puertos virtuales que estarían en condiciones de responder, a esto se le llama port-scanning:

A. Conección TCP

Se conecta al puerto de la víctima y ejecuta un saludo completo de tres pasos (handshake): SYN, SYN/ACK, y ACK. Este tipo de port scan es detectado facilmente por el sistema que esta siendo escaneado.

B. Prueba mediante el paquete TCP SYN

Es conocida también como "half-open scanning" ya que no se efectúa una comunicación propiamente dicha. Si un SYN/ACK se recibe del puerto del host scaneado, se deduce que esta en estado de "listening". Si se recibe un RST/ACK también se sabe que este puerto esta en "not listening". Finalmente el sistema que esta atacando envía un RST/ACK de modo que nunca se establece una sesion real. Esta técnica es más peligrosa que la anterior ya que no se puede detectar tan facilmente.

C. Prueba con el paquete TCP FIN

Esta técnica envía un paquete FIN al puerto víctima. Basado en el RFC 793, el sistema víctima deberia enviar un RST para todos los puertos inactivos. Y no responde nada si el puerto esta en estado de "listening".Esta técnica solo se puede utilizar en los sistemas UNIX.

D. Pruebas con el paquete TCP Xmas Tree scan

Esta técnica envía un paquete FIN, URG y PUSH al puerto víctima. Al igual que el anterior se debe recibir un RST en caso de que el puerto este inactivo.

E. Prueba con el paquete TCP Null

Es similar al anterior, con la diferencia que este scan hace que todos los flags del sistema víctima se anulen, si se recibe un RST el puerto esta inactivo.

F. Prueba con el paquete ACK del protocolo TCP

Se usa para saber cuales son las reglas de filtro de un firewall. Con este scan se puede determinar si un firewall es simplemente un filtro de paquetes permitiendo determinadas conexiones o un firewall con técnicas de filtro complejas.

G. Pruebas mediante protcolo TCP para Windows

Mediante esta técnica se puede detectar puertos abiertos estén o no estén filtrados.

Ya que se aprovecha de una anomalia en la forma en que el TCP reporta el tamaño de memoria disponible (windows).

H. Pruebas con el protocolo TCP RPC

Esta técnica se usa solo en sistemas UNIX y es útil para detectar e identificar puertos que estén haciendo llamadas de procedimiento remotos (RPC), sus programas asociados y números de versión.

I. Pruebas con protocolo UDP

Esta técnica envía un paquete UDP al puerto víctima. Si el puerto responde con un "ICMP port unreachable", el puerto esta inactivo. Si no se recibe nada se puede deducir que el puerto esta activo, UDP es conocido como un protocolo no confiable, en consecuencia, la fiabilidad de este método esta superditado a condiciónes externas.

1.2.5 Identificando servicios TCP y UDP activos

En un procedimiento de port-scanning la herramienta utilizada juega un papel crítico, se menciona a continuación los más efectivos y conocidos:

A. Herramienta "Strobe"

Es una herramienta de scanning TCP, como utiliza protocolo TCP, los datos que se obtienen con esta herramienta son confiables, sin embargo la desventaja es que solo se

pueden poolear los puertos TCP, no los puertos UDP. Asimismo al tratarse de comunicación TCP, el sistema escaneado puede detectarlo fácilmente.

B. Herramienta "Udp_scan"

Como Strobe solo cubre lo que respecta a puertos TCP, se puede usar udp_scan, originalmente SATAN (Security Administrator Tool for Analyzing Networks).

C. Herramienta "Netcat" ó "NC"

Es una herramienta tan eficaz que es llamada tambien la navaja suiza de las herramientas de seguridad. NC utiliza, por defecto el escaneo a puertos tcp, sin embargo si se escpecifica la opción –u efectua un escaneado UDP. El reporte que envia el nc tiene el siguiente formato:

```
[server1] nc –v –z –w2 214.124.13.1 1-140
[214.124.13.1] 139 (?) open
[214.124.13.1] 135 (?) open
[214.124.13.1] 110 (pop-3) open
[214.124.13.1] 106 (?) open
[214.124.13.1] 81 (?) open
[214.124.13.1] 80 (http) open
[214.124.13.1] 79 (finger) open
[214.124.13.1] 53 (domain) open
[214.124.13.1] 42 (?) open
[214.124.13.1] 25 (smtp) open
[214.124.13.1] 21 (ftp) open
```

Fig. 1.1 Muestra de como se ejecuta el programa en una PC

Donde, -v habilita el modo verbose, la opción -z habilita el modo I/O cero que se usa para el escaneo de puertos, la opción -w2 especifica un timeout para cada puerto.

D. Programa de mapeo de red: Network Mapper (nmap)

Esta herramienta es la más eficiente hasta el momento, además de tener capacidad de efectuar escaneo de puertos a nivel TCP y UDP, también muestra la opción de que el usuario escoja la técnica de escaneo a utilizar.

Ademas con nmap se puede escanear los puertos de los sistemas de una red en una sola línea de comando, con la opción –oN guarda el resultado en un archivo ascci. Ejm:

server1# nmap –sF 216.200.12.0/24 –oN archivo_de_salida.

Supongamos que la red que se quiere escasear esta atrás de un firewall que se comporta como un filtro de paquetes. Podemos utilizar la opción –f con la que se envian los paquetes fragmentados de modo que el firewall no lo reconozca.

1.2.6 Prueba de puertos para sistemas Windows

En la parte anterior se detallan herramientas solo para la plataforma UNIX, las siguientes son las herramientas mas conocidas en windows:

A. Programa "NetScan Tools Pro 2000"

Esta herramienta no es freeware pero se tiene la versión anterior a esta que es la versión 4. Es versátil y ofrece varias herramientas de exploración de red, que incluyen nslookup

y dig con axfr, whois, barridos de ping , escaneo de la tabla de nombres del NetBIOS, SNMP walks, etc.

B. Programa "Super Scan"

Esta herramienta es freeware, al igual que NetScan Tools, es bastante versátil en cuanto a la especificación de puertos y de IPs evaluados.

C. Programa "IPEye"

IPEye ejecuta escaneo de puertos en diferentes tecnicas, como SYN, FIN y Xmas desde la línea de comandos de Windows, la limitación de esta herramienta es que corre solamente wn Win2000, y escanea un host por vez.

1.3 ENUMERACIÓN

Suponiendo que efectuando el "footprinting" y "scanning" no se ha logrado ingresar al sistema, el siguiente paso es identificar los usuarios o recursos del sistema que estén compartidos y que no estén debidamente protegidos. Hay varias maneras de extraer una cuenta válida o nombres de recursos exportados del sistema, a este proceso se le llama "enumeración". Entre el tipo de información que se puede obtener mediante la enumeración se encuentran los siguientes:

- Recursos de red y recursos compartidos de red.
- Usuarios y grupos.

- Aplicaciones y banners (saludos de los equipos).

Las técnicas utilizadas en enumeración dependen del sistema operativo, el proceso de enumeración se basa en información conseguida previamente con el escaneo de puertos, que se analizó en el punto anterior.

1.3.1 Enumeración en Windows NT/2000

Windows es conocido como un sistema operativo que entrega información a intrusos remotos. Esto se debe principalmente al CIFS/SMB (Common Internet File System/Server Message Block, y los protocolos de transporte de NetBIOS, sobre la que los servicios de red fuertemente dependientes. Sin embargo Windows 2000 tiene la capacidad de correr TCP/IP sin NetBIOS. Desde la primera versión de Windows, hasta nuestros días, se venden diversos CDs con software adicional al sistema operativo, uno de estos softwares es el Windows NT Resource Kit. Este software contiene consta de una colección de utilidades potentes, sin embargo también puede ser utilizado por intrusos para tener acceso a información valiosa, de modo que en algunos circulos este kit recibió el apelativo de "The Windows NT Hacking Kit".

- **Sesiones no validas o Null**

Los standards de NetBIOS y CIFS/SMB incluyen APIs que devuelven información valiosa acerca de una máquina a través del puerto 139 TCP, aun cuando el usuario no esta autenticado. El primer paso para acceder a estos APIs remotamente es crear una

conexión no autenticada al sistema NT/2000 utilizando el comando llamado "null-session", asumiendo, de un port-scan previo, que el puerto TCP 139 este activo:

net use \\192.134.23.5\IPC$ "" /u:" "

La sintaxis precedente conecta los procesos de comunicación escondidos "share" (IPC$) a la dirección 192.134.23.5 como un usuarios anónimo (/u:" ") con password nulo (""). Si se logra establecer esta conexión, el atacante ahora tiene un canal abierto sobre el que puede intentar todas las técnicas que se detallaran mas adelante para informarse sobre: ubicación en la red, usuarios, grupos, claves de registro, etc. Para limitar el acceso a lo información NetBIOS a usuarios no autenticados se deben filtrar los puertos TCP y UDP desde 135 hasta 139 en el perímetro de la red. Además se puede deshabilitar NetBIOS sobre TCP/IP en hosts con plataforma NT

1.3.2 Enumeración de recursos de red de NT/2000

Lo primero que un atacante intentará en una red NT/2000 es averiguar que servicios existen. Primero intentará hacer enumeración de recursos del NetBIOS y luego enumeración de servicios TCP/IP que conmunmente se ofrecen en este tipo de sistemas.

A. Enumerando dominios NT/2000 con net view

El comando net view es una herramienta que lista los dominios disponibles en la red, por ejemplo:

C:\>net view /domain

Domain

APACHE

TOGETHER

NAZCAR

Estos son dominios de la red en donde se ejecuta el comando. De la siguiente forma se listan las máquinas que pertenecen al dominio APACHE:

C:\>net view /domain:apache

B. Obtener la tabla de nombres del NetBIOS con nbstat y nbtscan

Con el comando nbtstat se obtiene la tabla de NetBIOS de cualquier sistema remoto.

C:\>nbtstat –A 218.18.2.30

La desventaja del ndtstat es que solo se puede obtener información de una máquina a la vez. Para esto podemos utilizar el software, disponible en la red, "nbtscan" que es como hacer simultaneamente un nbtstat a todas las maquinas de una red:

C:\> nbtscan 218.18.2.0/24

1.3.3 Enumeración con protocolo SNMP de NT/2000

Si se tiene habilitado algún agente SNMP en la red y no se ha cambiado la comunidad por defecto de todos los equipos: "public", entonces facilmente se puede obtener información por este medio, utilizando la herramienta SNMP "snmputil":

C:\>snmputil walk 200.34.6.12 public .1.3.6.1.4.1.77.1.2.25

Donde, .1.3.6.1.4.1.77.1.2.25 es el "object identifier" OID, que es una rama de todos los Management Information Base MIBs que se definen en el protocolo SNMP, en la tabla 1.1 se especifican los MIBs mas solicitados por los atacantes a los sistemas víctimas:

SNMP MIB iso.org.dod.internet.private.enterprises.lanmanager.lanmgr2	Información
.server-svSvcTable.svSvcEntry.svSvcName	Servicios habilitados
.server-svShareTable.svShareEntry.svShareName	Share names
.server-svShareTable.svShareEntry.svSharePath	Share directorios
.server-svShareTable.svShareEntry.svShareComment	Comentarios en share
.server-svUserTable.svUserEntry.svUserName	Nombres de usuarios
.domain.domPrimaryDomain	Nombres del dominio

Tabla 1.1

1.3.4 Enumeración de usuarios y grupos de trabajo en NT/2000

La herramienta "enum" levanta sesiones null en el sistema remoto y extrae la información, esto se puede ver en el ejmplo siguiente:

C:\> enum –U –d –P –L –c 123.53.25.6

Con este comando se obtiene información sobre los usuarios, administradores, grupos, dominios y servicios de la maquina con el numero IP especificado.

1.3.5 Enumeración en UNIX

En cuanto a UNIX, la enumeración no es tan fácil como en Windows, sin embargo tiene sus puntos débiles en varias aplicaciones como Remote Procedure Call (RPC), Network Information Systems (NIS) y Network File System (NFS). Las tecnicas detalladas a continuación se basan en información previamente obtenida con los port scans que se estudió en el punto anterior.

- **Recursos de Red en UNIX y Enumeración compartida.**

Supongamos que un sistema UNIX esta compartiendo directorios de red, una manera práctica de obtener estos directorios, es con el comando "showmount". Supongamos que de un port scan previo se verificó que el puerto 2049 (NFS), como en la Fig. 1.2.

Este es el comportamiento por defecto del NFS ya que se trata precisamente de compartir archivos, sin embargo para filtrar estos requerimientos desde redes externas se deben filtrar los puertos 2049, ademas los requerimientos de NFS pueden ser registrados, esto es util para estar al tanto de probables ataques. Adicional a NFS, también se tiene la herramienta Samba que brinda servicios de red y de impresion

similares a clientes SMB. SMB (Server Message Block) forma parte de las herramientas de red de Windows. Samba es freeware y tiene diversos parametros que permiten configurarlo de forma segura, pero si no se configura convenientemente se pueden formar huecos de seguridad que pueden ser explotados por atacantes.

```
Showmount –e 200.21.234.19
export list for 200.21.234.19
/freeware
/todos
/export/home
```

Fig. 1.2 Muestra de cómo se ejecuta el programa en la PC

- **Enumeración de usuarios y grupos**

El comando más común para preguntar a un dominio o sistema remoto cuales son sus usuarios activos es el finger, que corre en el puerto 79, para evitar este tipo de intrusión se debe filtrar este puerto en los servidores. Se ejecuta de la siguiente forma:

root# finger –l @impa.br

root# finger 0@126.34.74.1

Como respuesta a estos comandos se muestra información sobre el usuario conectado a la máquina, el tiempo de cada sesión, shell utilizados, los comentarios del username, etc. Igualmente peligrosos, son los comandos rusers, y rwho.

Otra técnica clásica de enumeración son los comandos SMTP de aplicaciones de correo electrónico. SMTP brinda dos comandos que permite la enumeración de usuarios, uno es VRFY que permite verificar el nombre de usuarios permitidos dentro del servidor de correos, y EXPN que revela la dirección real de entrega de correos de usuarios que tienen aliases.

1.4 ATAQUES A SISTEMAS OPERATIVOS

1.4.1 Ataques a Windows 95/98 Y ME

Este sistema operativo es fácil de usar y configurar, la desventaja es que no es precisamente un sistema operativo diseñado para garantizar seguridad.

A. Ataques remotos

Este tipo de técnica se compone de cuatro categrias: conexión directa a recursos compartidos, instalación de demonios de salida del servidor, abuso de vulnerabilidades conocidas de las aplicaciones del servidor y denegación de servicios. Los tres primeros requiren que el administrador de este sistema operativo no tenga el sistema debidamente asegurado y configurado, por consiguiente se previenen facilmente.

B. Conexión directa a recursos compartidos

Hay tres mecanismos que este sistema operativo ofrece para tener acceso directo al sistema: compartir archivos e impresora, el servidor adicional de dial-up y manipulación remota del registro.

- **Archivos compartidos**

Existen varias tecnicas nombradas en la parte de scanning y enumeración que detallan como aprovecharse de los recursos compartidos de sistema. Una de estas herramientas se llama Legion, que ademas tambien tiene capacidad para llevar a cabo ataques de password. Legion tambien tiene una herramienta "BF" que adivina los passwords desde un texto plano, rapidamente los prueba y entrega los resultados que estan correctos. El nivel de daño que puede causar una intrusión de este tipo depende en el directorio al que se pueda entrar, si se trata de directorios con archivos de sistema, el atacante puede colocar agregar ejecutables en el archivo %systemroot%\Start Menu\Programs\Startup, y la proxima vez que se reinicie la maquina, este archivo se ejecutará.

- **Ataque remoto al registro de Windows**

A diferencia de Windows NT, en este caso no esta habilitado el acceso remoto al registro. Sin embargo, es posible que el Microsoft Remote Registry Service este instalado (que se encuentre en \admin\nettools\remotreg). Este Remote Registry Service requiere de seguridad a nivel de usuario para que sea habilitado, y por consiguiente también requiere de un nombre de usuario validado para el acceso.

C. Atques del tipo de "backdoor" y los llamados "troyanos"

Asumiendo que no este habilitado todo lo anterior en el servidor, esto no es una garantia de que el sistema este asegurado. Ya que si no hay ninguna manera de encontrar una herramienta de administración remota instalado en el sistema victima, entonces solo

falta instalar uno. A continuación se mostrará dos de los programas más populares que circulan por internet.

- **Orificios negros**

También llamados Back Orifice (BO), según sus creadores es una herramienta de administración remota, y es freeware en el internet, BO permite un control remoto semi completo del windows, incluyendo la habilidad para adicionar y borrar claves de registro, reiniciar el sistema, enviar y recibir archivos, ver passwords del cache, crear procesos y compartir archivos. BO puede ser configurado para instalar y correr el mismo bajo cualquier nombre de archivo.

Este programa añade una linea de ejecución al archivo "RunServices" de modo que se autohabilitet el blackdoor cada vez que el sistema se reinicie. El puerto que escuche este programa es el UDP 31337. La configuración por defecto de BO segunda versión, es que escuche en el puerto TCP 54320 o UDP 54321, y se copia asimismo a un archivo llamado umgr32.exe en el directorio de %systemroot%.

1.4.2 Ataques a Windows NT

Existen varias tecnicas para hacer efectivo un ataque al sistema operativo Windows NT:

A. Privilegios del administrador

En este sistema operativo el administrador es el único que tiene los privilegios como para modificar archivos de sistema, hacer comandos remotos, acceder a información del sistema, etc. de modo que si hay algun intruso que consigue logearse en la maquina, si no es administrador, sera poco o nada lo que puede hacer, por esto lo que hacen los atacantes es tratar de averiguar los passwords de los administradores:

- **Adivinando los passwords remotos**

En el caso de que este habilitado el servicio de Sesion de NetBIOS en NT, la forma mas efectiva de romper la seguridad es tratar de adivinar el password del sistema, para esto previamente a este paso ya se hizo el estudio del footprinting, escaning y enumeración, donde se obtienen los nombres de los usuarios validos del sistema. Una manera sencilla y discreta de adivinar el password del administrador es con la linea de comando:

C:\> net use \\10.123.33.18 \IPC$ * /user:Administrator

Para evitar que alguien ingrese al sistema mediante este método, se pueden instalar las herramientas Passprop o Passfilt, que permiten que si el administrador falla, por ejemplo, 5 veces el password, la cuenta del administrador esta bloqueada.

- **Interceptación del intercambio de password de red**

Si adivinar el password es demasiado, entonces lo que se puede hacer es interceptar la linea por donde pasa el intercambio de credenciales de los usuarios para logearse al servidor y luego hacer lo mismo para tener acceso al sistema. Esta tarea puede llevarse a cabo utilizando cualquier sniffer, pero hay una herramienta que es utilizada especialmente para esto, l0phtcrack.

- **Denegacion de servicios y sobrecargas**

Windows NT se caracteriza por ser un sistema relativamente seguro, y se han revelado muy pocos huecos que los atacantes podrian intentar abusar. Uno de estos huecos es la conocida sobrecarga de memoria o buffer, esto ocurre cuando los programas no revisan apropiadamente el tamaño de los datos de entrada. Este tipo de ataque se da en dos situaciones: cuando hay control local o remoto del sistema. La sobrecarga local requiere del control de la consola, pero las sobrecargas remotas son mucho mas peligrosas, ya que estas no necesitan que el atacante se logee al sistema y puede ser hecha desde cualquier parte de la red.

- **Escalando privilegios**

Supongamos que un atacate solo pudo logearse aun sistema como un usuario, y no consigue el password del administrador, esto no es un problema muy critico para el atacante ya que hay herramientas que permiten escalar privilegios y hacer que el usuario

pueda tener acceso de administrador. Hay varias tecnicas para escalar al privilegio de administrador, estos se pueden cargar desde locaciones remotas y la consola.

- Extrayendo información

Si algun intruso encuentra una cuenta de usuario, su unica opcion real es identificar mayor información para poder tener acceso de root, esto se logra siguiendo los pasos de enumeración detallados en un capítulo anterior. Asi combinando tanta información del sistema como sea posible los atacantes pueden identificar el acceso a los directorios mas criticos. Unas herramientas para esto son NTRK servinfo, utilizar "Find" para encrontrar archivos de programas con extensión .bat que contengan la palabra "password", también el "regdmp" prueba accesos a archivos de registro.

- Utilizando la herramienta "Getadmin"

Es un pequeño programa que aumenta un usuario perteneciente al grupo de Adminstrator en el sistema. La desventaja de este programa es que tiene que ser ejecutado en la misma maquina localmente. Para evitar ser victima de este tipo de ataque basta con instalar el parche de seguridad de esta versión de windows, Service Pack.

- Utilizando herramienta "Sechole"

Es similar a "getadmin", pero coloca al nuevo usuario en el grupo de "Domain Admin". Tambien tiene que ejecutarse localmente, pero si el sistema esta corriendo IIS, sechole puede ser ejecutado remotamente.

Para evitar este tipo de ataque no se debe permitir acceso de escritura a los directorios ejecutables del servidor web. Como se puede ver de lo anterior, no es tan facil acceder a privilegios de administrador de un sistema, a menos que el sistema no este correctamente configurado y actualizado o que el atacante tenga suficente acceso local al sistema, es decir debe tener por lo menos una cuenta de usuario.

1.5 ATAQUES AL SISTEMA UNIX

Hay básicamente dos formas de entrar a un sistema UNIX, una es mediante el acceso remoto y otra con el acceso físico al sistema, es decir, conociendo un acceso ya sea de ftp, tftp, telnet, etc desde cualquier parte de la red.

1.5.1 Acceso remoto

El acceso remoto esta compuesto de acceso de red o acceso a otro canal de comunicación como módems de dial-in. A continuación centraremos los conceptos relacionados con acceso a sistemas unix vía TCP/IP. Hay principalmente tres métodos para amenazar remotamente la seguridad de un sistema UNIX:

- Abusando de un servicio del sistema (Ej. TCP/UDP)
- Ruteo hacia un sistema UNIX que esta dando seguridad entre dos o mas redes.
- Ataques de ejecución remota iniciadas por un usuario (por ejemplo: sitios Web hostiles, emails de caballos troyanos, etc).

A. Ataques de Fuerza

También conocidos como Brute Forve Atack, Es una de las maneras más efectivas de tener acceso a un sistema UNIX. Un ataque de fuerza no es mas que adivinar una combinación de usuario/password en un servicio que permita acceder al sistema, los tipos de servicio mas comunes que pueden ser atacados de esta manera son:

- Telnet
- File Transfer Protocol (FTP)
- Los comandos R (rlogin, rsh, etc)
- Secure Shell (ssh)
- Nombres de comunidades SNMP
- Post Office Protocol (POP)
- HyperText Transport Protocol (HTTP/HTTPS)

La forma de ubicar a usuarios validados dentro de un sistema UNIX ya se explicó en el capítulo de enumeración, existen diversos servicios como finger, rusers y sendmail que son muy útiles en este proceso. La mayoría de los passwords se adivinan utilizando herramientas automatizadas y creadas especialmente para eso, entre las que se incluyen las siguientes:

- Brutus
- Brute web.c

- Pop.c
- Middlefinger
- Teenet

B. Ataques de data

Los ataques de data se ejecutan enviando data a un servicio activo para causar resultados no deseados o no esperados. Por supuesto estos "resultados no deseados o no esperados" son subjetivos y depende de si se es el atacante o la persona que administra el servicio. Los ataques de datos se dividen en dos categorías como ataque de overflow (saturación) de memoria o ataque de validación de entrada.

- **Ataques de saturación de memoria**

La saturación de buffer o memoria ocurre cuando un usuario o un proceso intenta colocar mas data en la memora de lo que fue originalmente permitido. Este tipo de comportamiento esta asociado con funciones especificas del C como strcpy (), strcat (), sprintf (), entre otras. Un estado de memoria saturada causa normalmente una violación de segmentación. Sin embargo, este tipo de comportamiento puede ser abusado para obtener acceso al sistema victima. En el siguiente ejemplo se detalla como es el proceso de este tipo de ataque:

Supongamos que se tiene un buffer predeterminado de 128 bytes, asumamos que este buffer define la cantidad de data que se puede almacenar como entrada mediante el

comando VRFY de sendmail. Asumamos también que sendmail esta corriendo con privilegios de root, lo cual ocurre frecuentemente. Entonces si alguien se conecta al demonio de sendmail y envía un bloque de data que tenga unas 1000 "g" al comando VRFY en vez de un nombre de usuario, saturaría este servicio.

Echo "vrfy 'perl –e 'print "g" x 1000'"" | nc *www.sistemavictima.com* 25

Como consecuencia la memoria se satura ya que esta configurada para albergar a lo mas 128 bytes y no 1000, esto causa denegación de servicios del sendmail y el demonio del sendmail se puede colgar. Esto es aun más peligroso si el sistema victima es el mismo que activa este comando.

En vez de enviar 1000 veces la letra "g" al comando VRFY, los atacantes enviaran códigos específicos que saturaran el buffer y ejecutaran el comando /bin/sh. Como el sendmail esta corriendo como root, los atacantes tendrán acceso instantáneo al root, para que el sendmail ejecute el /bin/sh por si mismo, se envía un código especial de assembler conocido como eggs al comando VRFY como parte de la cadena de caracteres utilizada para saturar la memoria.

Hay recordar que el código assembler es dependiente del sistema operativo y de la arquitectura del sistema. Una saturación de memoria para el solaris X86 que correo

sobre CPU Intel es completamente diferente a uno para Solares corriendo sobre un sistema SPARC. La siguiente lista ilustra un egg o código assembler para linux X86:

char shellcode [] =

"\xeb\xlf\x5e\x89\x76 .../xff\xff\xff/bin/sh" ";

Los ataques de saturación de memoria son extremadamente peligrosos, es extremadamente difícil crear un egg efectivo. Sin embrago, la mayoría de los eggs dependientes del sistema han sido ya creados y están disponibles en Internet.

- **Ataques de validación de entrada**

Un ataque de validación de entrada ocurre cuando:

- Un programa falla en reconocer sintacticamente entradas incorrectas.
- Un modulo acepta una entrada extraña
- Un modulo falla en manejar campos de entrada faltantes.
- Ocurre un error de correlación de campo-valor

El PHF es un script en CGI que vino en un inicio con la versión standard del servidor Web Apache. Este programa no dividía convenientemente la entrada que recibía. La versión original del script PHF aceptaba un carácter de nueva línea (%0a) y ejecutaba

comandos subsecuentes con los privilegios del usuario que levanto el demonio del servidor Web. El abuso original de PHF fue el siguiente:

/cgi-bin/phf?Qlaias=x%a/bin/car%20/etc/passwd

Con el comando anterior se podía ver el archivo /etc/passwd, con lo que se podía identificar los usuarios del servidor Web, así como los passwords encriptados, asumiendo que el archivo de password no estaba disponible, un atacante mas sofisticado podía modificar este archivo y tener acceso a un shell del sistema.

C. Telnet reverso y back channels

El aplicativo Xterm es un buen comienzo para los atacantes, se define back channel como un mecanismo donde el canal de comunicación se origina desde el sistema victima en vez de originarse del sistema atacante. Supongamos que los sistemas victimas están totalmente protegidos salvo los puertos 80 y 443, entonces el ataque debe comenzar desde el sistema UNIX y debe dirigirse hacia la maquina del atacante.

Una de las formas de hacer que haya un telnet reverso es el telnet reverso, es llamado así por que la conexión empieza en el sistema creando un canal de llegada hacia el atacante. Un cliente telnet es típicamente instalado en los servidores UNIX. Para ejecutar un telnet reverso, se necesita listar todas las utilidades de netcat o nc. Se debe ejecutar en el

sistema atacante estos dos comandos en dos ventanas diferentes para recibir el telnet reverso:

Instruso# nc –l –n –v –p 80

Instruso# nc –l –n –v –p 25

Para evitar estos ataques no debe haber servicios como HTTPD o sendmail en los puertos 80 o 25. Si uno de estos puertos estuviera utilizado, hay que matarlo con el comando kill enviado desde nc. Los dos comandos nc escuchan en los puertos 25 y 80 a través de –l y –p, el modo verbose –v y no resuelve las direcciones IP a hostnames –n.

Ahora debemos ejecutar los comandos de telnet al puerto 80 y 25 desde el sistema remoto, esto se puede hacer abusando del PHF.

1.5.2 Acceso Local

Cuando el atacante tiene un shell interactivo se considera que tiene acceso local al sistema. Aun cuando es posible tener el acceso a root desde una vulnerabilidad remota, los atacantes frecuentemente utilizan un acceso a usuario primero. De este modo los atacantes pueden escalar privilegios hasta llegar al root. Esto es conocido como "escalar privilegios", la dificultad del escalamiento depende de la configuración del sistema y del sistema operativo.

A. Vulnerabilidades en la composición de passwords

Cuando se hacen pruebas de todas las combinaciones de los caracteres para adivinar el password de root se esta hablando de un ataque de fuerza "brute force attack". Romper un password comúnmente se conoce como un ataque de diccionario (automated dictionary attack). Mientras que el ataque de fuerza es un ataque activo, para romper el password solamente se necesitan los archivos de /etc/passwd y de /etc/shadow. Cuando se rompe un password el atacante trata de adivinar aleatoriamente el password adivinándolo y luego lo compara con el password encriptado de los archivos /etc/passwd o shadow. Adivinar no es realmente complicado, ya que si se conocen unos pocos caracteres del password se pueden deducir los otros, ya que como se sabe el algoritmo para los passwords es el DES, del que se hablará mas adelante.

El software mas popular para romper passwords es el "Crack", el cual es muy practico solo es necesario darle los archivos como entrada y el programa entrega el password de cada usuario del sistema.

B. Ataques de descriptor de archivos

Los descriptotes de archivo son enteros no negativos que el sistema usa para mantener un registro los archivos en vez de utilizar sus nombres de archivos conocidos. Por convención los descriptores de archivo 0, 1 y 2 tienen uso reservado para la entrada estándar, la salida estándar y el error estándar respectivamente.

1.6 TÉCNICAS AVANZADAS DE ATAQUES

1.6.1 Ataque de denegación de servicios

Los ataques de denegación de servicios son peligrosos, en este tema se trata de varias herramientas que los hackers utilizan para causar caos y pérdidas de servicio en el Internet en los últimos años.

En esencia, un ataque de DoS (Denial of Service) desestabiliza o deniega el servicio completamente a usuarios, redes, sistemas, etc. legalmente autorizados.

En setiembre de 1996 un operador ISP (PANIX) quedó bloqueado por una semana por uno de estos ataques, los resultados fueron 6000 usuarios y 1000 compañias sin servicio de internet. Lo más alarmante de este gran ataque fue que se aprovecho una debilidad inherente del protocolo del internet (TCP/IP) y la manera como los sistemas manejan los requerimientos de SYN. Esta situacion se tornó más crítica ya que el atacante mostraba una dirección fuente falsa para esconder su identidad. Estos ataques fueron extremadamente dificiles de seguirles la pista para identificar a los perpetradores.

A. Tipos de ataques de DoS

Es más fácil interrumpir una operación de red o sistema que realmente tener acceso a la misma. El protocolo TCP/IP fue inicialmente diseñado para ser utilizado entre una comúnidad abierta y de mutua confianza, y por lo tanto las versiones actuales del protocolo tienen defectos inherentes. Además, muchos sistemas operativos y dispositivos de red tienen defectos en sus memorias de red que debilitan su habilidad

para resistir a los ataques de DoS. A continuación se veran los cuatro tipos de ataques de DoS:

- **Consumo de ancho de banda**

La forma más insidiosa de ataques de DoS son los ataques que consumen ancho de banda. En resumen, los atacantes consumen todo el ancho de banda disponible de una red en particular. Esto puede suceder en una red local, pero es mucho mas frecuente que los atacantes consuman los recursos remotamente. Hay dos escenarios básicos para el ataque:

- Escenario 1:

Los atacantes son capaces de inundar la conexión de red de la víctima por que tienen mayor ancho de banda disponible. Un escenario probable podria ser de alguien que tenga una conexión T1 (1,544 Mbps) o mas rapido y que inunde una conexión de 64 Kbps o 128 kbps.

- Escenario 2:

Los atacantes "amplifican" su ataque DoS empleando diversos sites para inundar la conexión de red de la víctima. De esta manera alguien que tiene una conexión de 64Kbps puede saturar completamente una conexión T3 (45 – Mbps de acceso). Esto es posible utilizando otros sites para amplificar el ataque DoS. Alguien que tiene ancho de banda limitado puede recolectar hasta 100 Mbps de ancho de banda, utilizando técnicas

de recolección de ancho de banda que son relativamente sencillos de implementar. Como se sabe el tráfico ICMP es peligroso. Mientras ICMP sirve para diagnósticos de acceso, además puede ser objeto de abuso y es el "proyectil" utilizado para ataques de consumo de ancho de banda.

- **Privación de recursos (Resource Starvation)**

El ataque de privación-de-recursos se diferencia del anterior ataque en que este se concentra en consumir los recursos del sistema y no los recursos de red. Generalmente, este consiste en consumir los recursos de sistema como abuso de CPU, memoria, quotas de archivos de sistema, u otros procesos de sistema. Con frecuencia, los atacantes tienen administración ilegítima de una gran cantidad de recursos de sistemas. Finalmente este tipo de ataque se traduce en que los sistemas colapsan, se llenan los sistemas o los procesos se cuelgan.

- **Defectos de programación (Programming Flaws)**

Son fallas de una aplicación o sistemas operativos o chips lógicos implantados para manejar condiciónes escepcionales. Estas condiciónes normalmente resultan cuando un usuario envía data involuntariamente a algún elemento vulnerable. Muchas veces los atacantes envían a sus objetivos paquetes extraños que no cumplen con ningun RFC para determinar si la capacidad de almacenamiento de esta red manejara esta escepcion o si resultará en un pánico del kernel y en un colapso completo del sistema. Como se sabe no existe ningún programa, sistema operativo o CPU libre de bugs.

- **Ataques de routing y a resolvedores de dominios de redes**

Un ataque de DoS basado en routing implica que el atacante pueda manipular entradas de tablas de routing para denegar servicios legítimos a sistemas o redes. La mayoria de los protocolos de red como el protocolo RIPvl y BGPv4 no tienen casi ninguna autenticación. Esto brinda un escenario perfecto para que los atacantes alteren las rutas legitimas, con una dirección IP fuente falsa, para crear una condición de DoS. Las víctimas de este tipoo de ataques tendrán su tráfico ruteado hacia la red del atacante o hacia un hueco negro, una red que no existe.

Los ataques de DoS sobre DNSs son tan problematicos como los ataques basados en routing. La mayoria de estos ataques consiste en convencer a los servidores DNSs de almacenar direcciónes falsas.

1.6.2 Ataques genéricos de DoS

Algunos ataques DoS son capaces de afectar sistemas de diferentes tipos, a estos se les llaman "genericos". Generalmente, estos ataques están dentro del grupo de "consumo de ancho de banda" o "privación de recursos".

Un elemento común de este tipo de ataques son las manipulaciones de protocolos. Si un protocolo como el ICMP se manipulara para propositos nefastos, tendría la capacidad de afectar varios sistemas simultaneamente. Por ejemplo, los atacantes pueden utilizar bombas de email para enviar miles de mensajes de email a sistemas víctimas en un

intento de consumir ancho de banda asi como de agotar los recursos de red en el servidor email. El virus Melissa, en realidad es un gusano, no fue diseñado para hacer un ataque DoS, pero se pudo comprobar como una oleada de mensajes email puede llevar a un servidor email a una interrupción de servicios. Los siguientes son los ataques más relevantes.

A. Ataque tipo "Smurf"

El ataque smurf es uno de los mas temibles debido a los efectos de amplificación del ataque. El efecto de amplificación es el resultado de enviar un requerimiento de ping broadcast directo a una red de sistemas que responderán a este requerimiento. Un requerimiento de ping broadcast puede ser enviado a una dirección de red o a una dirección broadcast de red y requiere de un dispositivo que pueda pasar el broadcast de capa 3 a capa 2. Si asumimos que esta red tiene una clase C standard, la dirección de red será .0, mientras que la dirección de broadcast sera .255. Los broadcast directos se usan para propósitos de diagnóstico y ver que sistema está funcionando, asi se evita hacer ping a cada dirección de la red.

Un ataque smurf se aprovecha de los broadcasts directos y requiere de un mínimo de tres elementos: el atacante, la red amplificadora, y la víctima. El atacante envía paquetes ICMP ECHO con una dirección IP fuente falsa, hacia la dirección broadcast de la red amplificadora. La dirección fuente de los paquetes se falsifica para que parezca que es la dirección de la víctima, y que esta es la que ha iniciado el requerimiento de ICMP

ECHO. Luego se inicia el ataque. Ya que el paquete ECHO fue enviado a la dirección broadcast, todos los sistemas en la red responderan hacia la víctima.

Hay una variante de este ataque llamado ataque Fraggle, que es basicamente un ataque smurf pero en vez de paquetes ICMP se utilizan paquetes UDP. Los atacantes usan paquetes UDP con direcciónes fuente falsas a la dirección broadcast de la red amplificadora, puerto 7 (echo)

Para prevenir este tipo de ataques y evitar ser utilizados como una herramienta amplificadora, se deberia deshabilitar el broadcast en el router de borde. Para routers Cisco, se debe usar el comando "no ip directed-broadcast" para esto, ya que en el IOS 12 este está habilitado por default. Adicionalmente, los sistemas operativos pueden ser configurados para que descarte los paquetes ECHO ICMP.

En solaris, adicionar la sgte. Linea a /etc/rc2.d/S69inet:
add –set /dev/ip ip_respond_to_echo_broadcast 0.

B. Ataque de tipo "SYN Flood"

Antes de que el ataque Smurf se volviera tan conocido, el ataque SYN era el ataque de DoS disponible más devastador. Esto funciona de la sgte. forma: cuando se establece una conexión TCP, hay un proceso de tres pasos, bajo circunstancias normales un paquete SYN se envía desde un puerto específico del sistema A un puerto especifico que

esta en estado LISTEN en el sistema B. En este punto, el sistema B esta en estado SYNC_RECV. Luego el sistema B intentará enviar un paquete de SYNC/ACK al sistema A. Si todo sigue bien, el sistema A enviará un paquete ACK, y la conexión estará en estado ESTABLISHED.

El problema es que la mayoria de los sitemas aloja un número finito de recursos cuando se establece una conexión completa o incompleta. Mientras que la mayoria de los sistemas puede mantener cientos de conexiones concurrentes a un mismo puerto (por ejemplo el puerto 80), solo es necesario una doceno de conexiones o requerimientos de conexiones potenciales para agotar los recursos que se encargan de establecer una conexión.

Para evitar este tipo de ataque, antes que todo se puede detectar si estamos bajo este tipo de ataque ejecutanto "netstat". Si se observan varias conexiónes en SYN_RECV, puede indicar que hay un ataque SYN en progreso, pero se puede evitar de la siguiente forma:

- Incrementar el tamaño de colas de conexión (queue)
- Disminuir el periodo de timeout para establecimiento de conexión.
- Emplear los parches de los proveedores para detectar y evitar ataques SYN potenciales.
- Emplear IDSs.

C. Ataques a DNSs

En 1997 se alerto sobre algunas debilidades del BIND, que es el software mas utilizado para implementar un DNS. Las versiónes antiguas del BIND grababan la informacion falsa cuando el demonio del BIND se ejecutaba.

D. Ataques de DoS distribuidos - DDoS

Ocurre cuando alguien usa algun software disponible para enviar una rafaga de paquetes hacia la red o host destino intentando saturar sus recursos. Pero en el caso de DoS distribuidos, la fuente del ataque proviene de múltiples lugares. Y la única forma de crear este escenario es comprometiendo sistemas de computadores existentes en el internet.

El primer paso de cualquier ataque DdoS es obtener el acceso a tantos sistemas como sea posible. Este ataque tan siniestro se ejecuta con scripts adaptados para identificar los sistemas vulnerables. A continuación se muestra herramientas para efectuar estos ataques:

- **Ataques de Flujos de red - Tribe Flood Network (TFN)**

Fue el primer software de DDoS, para plataformas UNIX. TFN tiene los componentes de cliente y servidor, para iniciar un ataque, se tiene la opción de que el atacante pueda instalar la parte servidor en un sistema remoto solo con un simple comando en el servidor cliente.

- **Ataque tipo "Trinoo"**

Es similar a TFN, se activa con una sesion de programa de control remoto (cliente) hacia el master y envía instrucciones a los demonios del master para hacer el ataque. La comunicación entre el cliente y el master es sobre el puerto TCP 27665, y normalmente requiere el password "betalmostdone". La comunicación entre el master y el sevidor es sobre el puerto UDP 27444. La comunicación de vuelta para el master es por el puerto UDP estático 31335.

Para detectar este tipo de ataques existen mecanismos de detección, como el DDOSPing. Para prevenir lo unico necesario es que el servidor UNIX debe estar previamente protegido.

- **Ataque tipo "Stacheldraht"**

El Stacheldraht combina las propiedades de Trinoo con las de TFN para formar una herramiente de destrucción que incluye las sesiones telnet encriptadas entre los esclavos y los masters. Con esto el atacante puede cegar los sistemas IDS. Stacheldraht también ataca con ICMP, UDP y SYN, y ataques multiplos de Smurf. Para la comunicación entre el cliente y el server, Stacheldraht usa una combinación de paquetes TCP y ICMP. La encriptación utilizada entre el cliente y el servidor emplea un algoritmo de encriptación simetrica. También tiene la capacidad de hacer un updrage de los componentes del servidor utilizados mediante el comando rcp.

Para detectar este ataque se puede utilizar el DDOSPing. Pero para prevenir hay que asegurarse que los sistemas no puedan ser utilizados como zombies, esto implica implementar la limitación de servicios, parches.

1.6.3 Dispositivos de red

En la realidad, la típica red lan o wan no es completamente segura. En la mayoria de los casos, administrar la red significa estar al tanto del tráfico cursante como los emails, datos financieros, redirección de tráfico a sistemas no autorizados, el uso de la tecnología VPN, etc.

A. Descubrimiento de la Red

Los atacantes suelen comenzar por esta etapa con un port scanning, identificando puertos abiertos, luego aprenderán cuales son los mensajes enviados por estos equipos y finalmente la enumeración IP que manejan, para esto podría ser utilizado el netcat.

Si esta habilitado el puerto UDP (161), se utilizará el SNMP para identificar las debilidades reales.

- **Detección**

Un port scanning se puede efectuar con una variedad de herramientas, como traceroute, netcat, nmap ó SuperScan.

Traceroute: Usando esta herramienta se puede determinar los ips de los routers que hay entre dos redes. Es muy útil en el proceso de identificación del objetivo.

Port Scanning: En este caso si se estuviera tratando con routers Cisco, los puertos TCP comunes con 1-25,80,512-515,2001,4001,6001 y 9001. Después de identificar la lista de puertos con SuperScan o nmap, se puede scannear la red para identificar dispositivos Cisco.

Identificación del Sistema Operativo: para confirmar si estamos frente al un dispositivo Cisco, utilizamos nmap (nmap –O –p13 –n 10.101.2.1).

- **Protocolo SNMP**

El Simple Network Management Protocol es un protocolo diseñado para ayudar a los administradores a manejar sus dispositivos de red. La versión original tiene solo un mecanismo de seguridad: passwords, conocidos comunmente como nombre de la comunidad. En la segunda versión se usa un algoritmo aleatorio llamado mensaje digest v5 (MD5) para autenticar las transmisiones entre los servidores SNMP y los agentes. MD5, verifica la integridad de las comunicaciones y su origen. Hay dos tipos de comunidades SNMP: comunidad de solo lectura (*read*) o comunidad de lectura y escritura (*read/write*). La comunidad de solo lectura permite ver los detalles de configuración de los dispositivos de red, pero la comunidad de lectura/escritura permite que el administrador (o atacante) pueda escribir información hacia el dispositivo. Por ejemplo se puede cambiar el nombre del contacto del sistema con un simple comando:

snmpset 10.4.122.4 private .1.3.6.1.2.1.1 s Nuevo_contacto

Para evitar este tipo de intrusión se debe primero que todo cambiar el nombre de la comunidad por defecto que viene en los equipos, private y public son los nombres por defecto de las comunidades de los dispositivos de red. Además se debe configurar listas de acceso de SNMP, puertos 160 y 161 para solo dar acceso a este tipo de requerimientos solo a los sistemas de administración.

1.7 ATAQUES A PÁGINAS DE INTERNET

1.7.1 Búsqueda exhaustiva de la página Web

Esto es muy similar a la parte de footprinting del primer capítulo, donde se busca exhaustivamente cada pagina de un site, para averiguar todo sobre las paginas Web ahí publicadas, que comúnmente están comentados en los códigos fuentes de las paginas como direcciones de email, números de teléfonos, código en JavaScriot, etc. Para sites grandes y con muchos links hay varias herramientas que hacen esta búsqueda mas fácil, estos son scripts en perl o también el Teleport Pro que se puede copiar todo un site en la maquina del atacante para que este pueda luego revisarlo en detalle.

1.7.2 Encontrando vulnerabilidades conocidas

Para prevenir el hackeo a un site Web, el administrador deberá correr siempre los scripts de búsqueda de debilidades y verificar que el site esta completamente seguro, a

continuación se especifican los scripts mas conocidos, se pueden encontrar mas scripts que busquen vulnerabilidades en la Web.

A. Ataque al archivo Phfscan.c

Este programa ubica los servidores que tienen la vulnerabilidad del PHF que se detallo en capítulos atrás, esto permite que los atacantes ejecuten cualquier comando localmente como usuarios del servidor Web, lo más común es que el atacante copie el archivo /etc/passwd. Para ejecutar este programa: "gcc phfscan.c o phfscan". Con esto el programa ubica de una lista de servidores Web aquellos que son vulnerables.

B. Ataque al archivo Cgiscan.c

Esta es otra utilidad para escanear vulnerabilidades conocidas como PHF, count.cgi, test-cgi, PHP, handler, webdist.cgi, etc. este programa busca las vulnerabilidades en el directorio por defecto (http://webserver/cgi-bin/).

C. Errores en los programas (scripts): Vulnerabilidades

Los ataques de este tipo utilizan programas de Common Gateway Interface (CGI). Active Server Pages (ASP), y Cold Fusion Markup Languange (CFML) de un desarrollador Web o una falla de venta. El problema surge de la inadecuada programación de la entrada en un script en particular. Sin la debida validación, es posible que los atacantes puedan ingresar un carácter en particular, con un comando local como parámetro y hacer que el servidor Web lo ejecute localmente.

1.7.4 Sobrecarga del Buffer o memoria

Las sobrecargas de buffer son una grieta en la plataforma de la seguridad UNIX, las siguientes vulnerabilidades dan una idea de cómo los atacantes explotan las sobrecargas de buffer remotamente.

A. Vulnerabilidad de tipos PHP

Adicionalmente a la vulnerabilidad PHP que tiene que ver con la entrada de datos que permiten ejecutar un comando remotamente al Web server, existe la vulnerabilidad conocida como sobrecarga de buffer en el php.cgi2.0beta10. El problema ocurre cuando los atacantes envían una cadena larga dentro de la función del FixFilename () y sobrescribe la pila de la maquina, permitiendo que un código arbitrario se ejecute en el sistema local.

B. Vulnerabilidad www.count.cgi

Es un contador Web muy popular. La vulnerabilidad permite que un atacante remoto ejecute remotamente cualquier codificación en el sistema local.

CAPÍTULO II

PROTOCOLOS DE ENCRIPTACIÓN

2.1 FUNDAMENTOS

2.1.1 Terminología

A. Emisor y Receptor

En este caso es un emisor que quiere enviar un mensaje de una manera segura de modo que alguien que tenga acceso al medio de transmisión no pueda leer el mensaje.

B. Mensajes y Encriptación

El mensaje es un "texto plano". El proceso para cambiar el mensaje y no sea entendible se llama encriptación. Un mensaje encriptado es un "texto cifrado". El proceso por el que se regresa este "texto cifrado" a "texto plano" es llamado desencriptación.

Criptografía es la ciencia que mantiene los mensajes seguros y las personas que lo practican se llaman encriptadores. Las personas que descifran los textos cifrados son

llamados criptoanalistas, la rama de la matemática que se dedica al criptoanalisis es la criptología.

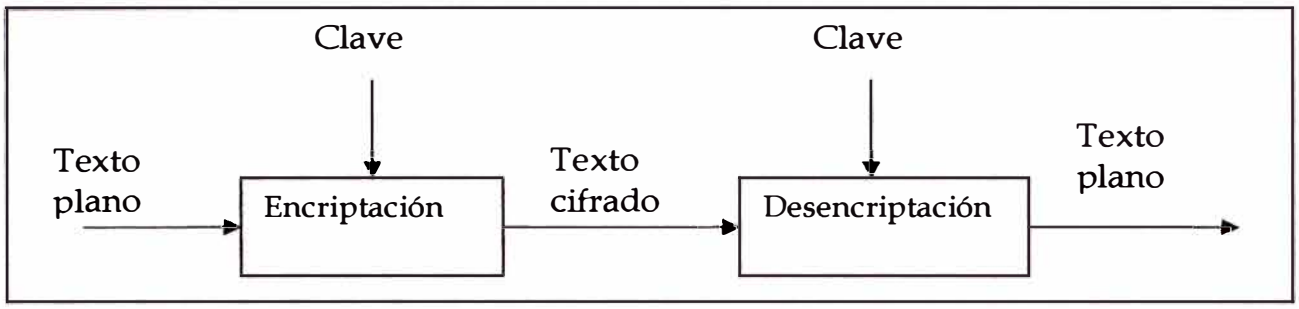


Fig. 2.1 Diagrama de comunicación con la encriptación

C. Autenticidad, Integridad y No repudiación.

La autenticidad consiste en que el receptor pueda identificar al emisor del mensaje, sin dejar que un atacante pueda pasar como el emisor.

La integridad consiste en que la información no puede ser alterada en el transcurso de ser enviada. La integridad se puede solucionar con técnicas criptográficas, particularmente con procesos simétricos o asimétricos. Las técnicas necesarias para verificar la autenticidad tanto de personas como de mensajes usan la mas conocidad aplicación de la criptografía asimétrica que es la firma digital. Para autenticar mensajes se usa la criptografía simétrica.

Mediante la no repudiación un emisor no podría negar falsamente que ha enviado un mensaje. Estos tres conceptos son los requisitos básicos para una interacción segura.

D. Algoritmos simétricos y asimetricos

Un algoritmo criptográfico, también llamdo un texto cifrado, es una función matemática usada para la encriptación y desencriptación. Si la seguridad de un algoritmo se basa en la confidencialidad de la forma en que el algoritmo trabaja, se le llama algoritmo restringido. Estos algoritmos son peligrosos cuando se utilizan entre un gran grupo de gente, sin embargo son bastante utilizados en aplicaciones que requieren poca seguridad.

La criptografía moderna resuelve el problema con una clave (key), la cual puede consistir en varios valores, el rango de estos valores es llamado keyspace, en los casos que hay una clave, no interesa si hay alguien que tenga el algoritmo, si no que no descifre a la clave. De este modo tenemos un criptosistema que consiste en todos los posibles "textos planos", "textos cifrados" y "keys". Resumiendo tenemos:

- **Algoritmos simétricos**

Se refiere al conjunto de métodos que permiten tener una comunicación segura siempre y cuando previamente se haya intercambiado la clave entre las partes, a esta clave se le llama clave simétrica. Son algoritmos en los que la clave de encriptación se puede deducir de la clave de desencriptación y viceversa. Este algoritmo requiere que el emisor y el receptor se pongan de acuerdo en la clave que van a utilizar antes de empezar la comunicación. La seguridad de esta comunicación depende en la confidencialidad (no publicación) de esta clave.

Hay dos tipos de algoritmos simétricos: los que operan cada vez que hay un bit o byte llamado "algoritmo de cadena" o "algoritmo stream" y otros que operan sobre un grupo de bits o sobre bloques llamado "algoritmo de bloques".

- **Algoritmos de clave pública (public key)**

También se les llama algoritmos asimétricos, en este caso existen claves diferentes para la encriptación y la desencriptación además ninguna de las claves se puede deducir de la otra, aquí se permite que la clave de encriptación sea pública. En conclusión, la clave de encriptación es la clave pública y la clave de desencriptación es llamada clave privada o clave secreta.

E. Criptoanalisis

Como se mencionó anteriormente el criptoanalasis consiste en recuperar el texto plano de un mensaje encriptado sin tener acceo a la clave, un criptoanalisis exitoso puede recuperar la clave o el mensaje original, esto demuestra que se pueden encontrar debilidades en un criptosistema. La pérdida de la clave a través de medios no criptoanaliticos es llamado un compromiso. A un intento de criptoanalisis se le llama ataque.

Pongamos el siguiente criptosistema: hay cuatro tipos de ataques criptoanaliticos, en todos ellos se asume que el criptoanalista conoce completamente el algoritmo utilizado:

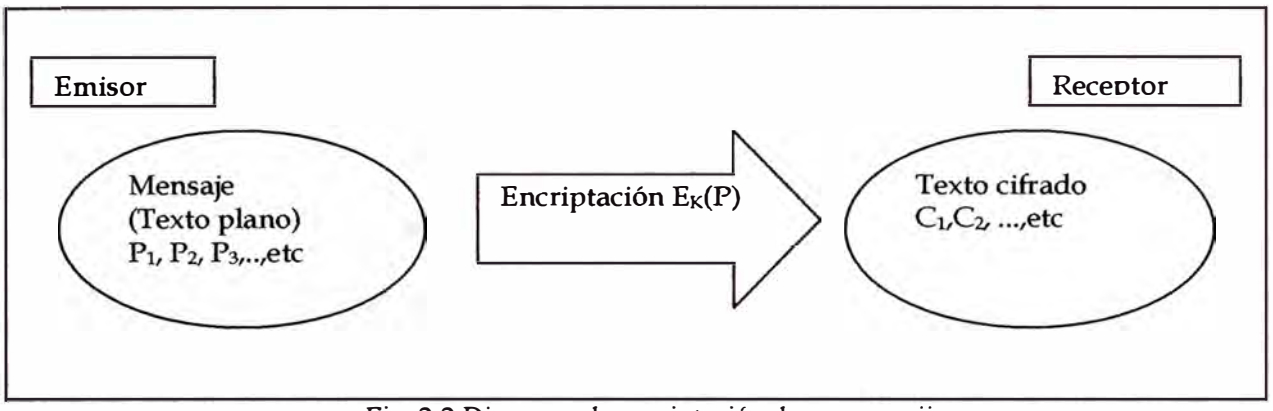


Fig. 2.2 Diagrama de encriptación de un mensajje

- **Ataque de texto cifrado unicamente**

O también "ciphertext-only", consiste en que el criptoanalista tiene el texto cifrado de varios mensajes, los que han sido encriptados utilizando el mismo algoritmo de encriptación. Lo que hace luego es recuperar el texto plano de tantos mensajes como le sea posible, o recuperar la clave (o claves) de encriptación. Ejm:

Datos: $C_1 = E_k (P_1)$, $C_2 = E_k (P_2)$

Se deduce: P_1, P_2, etc. ó K

- **Ataque de texto plano conocido**

También llamado "known-plaintext attack", en este caso el criptoanalista conoce el texto plano de algunos mensajes y su texto cifrado, solamente restaria deducir la clave.

Datos: P_1, $C_1 = E_k (P_1)$; P_2 ,$C_2 = E_k (P_2)$ ...

Se deduce: K ó un algoritmo para inferir P_{i+1} de $C_{i+1} = E_k (P_{i+1})$

- **Ataque de texto plano seleccionado**

También conocido como "chosen plaintext attack", es más peligroso que los anteriores por que el atacante puede escoger los textos planos especificos y sus correspondientes textos cifrados, que le hacen mas fácil deducir la llave.

Datos: P_1, $C_1 = E_k (P_1)$; P_2 ,$C_2 = E_k (P_2)$ ... donde el criptoanalista escoge P_1, P_2, P_3, ...

Se deduce: K ó un algoritmo para inferir P_{i+1} de $C_{i+1} = E_k (P_{i+1})$

- **Ataque de texto plano seleccionado y adoptivo**

Es un caso especial del ataque anterior, en este caso además de escoger el texto plano, el atacante puede modificar el texto plano de modo que le permita acceder a la clave con mayor rapidez.

F. Seguridad de los algoritmos

Los algoritmos tienen diferentes grados de seguridad, y esto depende de que tan difícil sea de descifrar. Si el costo requerido para descifrar un algoritmo es mayor que el valor de la data encriptada entonces se puede considerar al algoritmo como probablemente seguro. A continuación se definen las categorías para descifrar o romper un algoritmo:

- **Break total**

El criptoanalista encuentra la clave K, como $D_k(C)=P$.

- **Deducción global**

El criptoanlaista encuentra un algoritmo alterno, A, que sea equivalente a la clave sin tener conociemiento de esta.

- **Deducción local**

El criptoanalista encuentra el texto plano de un texto cifrado interceptado.

- **Deducción de información**

El criptoanalista obtiene información sobre el texto plano o la clave, y esta información puede ayudarlo a encontrar algunos bits de la clave, o la forma del texto plano.

Un algoritmo se cosidera seguro incondicionalmente si es que sin importar cuanto texto cifrado el criptoanalista tenga, no haya suficiente información para recuperar el texto plano. En general, la complejidad de un ataque depende del mínimo de factores necesarios para romper un algoritmo, como operaciones con computadores, mientras no se tengan al momento ni en un futuro cercano herramientas necesarias para romper el algoritmo.

2.1.2 Esteanografía

Sirve para esconder mensajes secretos en otros mensajes, de manera que se oculte la existencia del secreto en si. Generalmente, el emisor escribe escribe un mensaje incoherente y luego oculta un mensaje secreto en el mismo medio, este puede ser una

hoja de papel. En un principio se hacia esténografía con tinta invisible, marca en algunos caracteres de un mensaje externo, etc, últimamente se han ocultado mensajes secretos utilizado los bits de imágenes graficas. Ya que si se cambia el bit menos significativo de cada byte de una imagen esta no cambia a simple vista.

2.1.3 Digitos y cifras de substitución y transposición

Antes de las computadoras, la criptografía consistía en algoritmos basados en caracteres. Los algoritmos criptográficos substituían un carácter por otro o transponía caracteres con otro. Hoy en día la filosofía de la criptografía no ha cambiado pero si se ha hecho mas compleja.

A. Substitución de cifras

Es una en que cada carácter de un texto plano es substituido por otro carácter en el texto cifrado. El receptor invierte la substitución en el texto cifrado para recuperar el texto plano. En la criptografía clásica, hay cuatro tipos de substitución de cifras:

- Sustitución simple de cifras o cifrado monoalfabetico.

- Substitución homofónica de cifras, es como una substitución simple de cifras, excepto que un solo carácter puede cambiarse por diferentes caracteres diferentes. Cifrado(a) = 3 ó 5t ó R, etc.

- Substitución poligrámica de cifras, es una en la que un bloque de caracteres son encriptados en otro grupo específico de caracteres. Ej. Cifrado (AMB) = NUF.

- Substitución poli alfabética de cifras, esta hecha de múltiples substituciones de cifrado. Por ejemplo, pueden hacerse unas cinco substituciones simples.

Una programa de encriptación fácilmente encontrado en UNIX el ROT13, donde se corre 13 caracteres del abecedario al texto original.

B. Transposición de cifras

Es tan simple como lo indica su nombre, el texto plano permanece como tal pero el orden de los caracteres se cambia.

2.1.4 Algoritmos de Computadoras

Hay muchos algoritmos criptográficos de computadoras, sin embargo los más comunes son:

- DES. Data Encription Standard, es el más popular. Es un algoritmo simétrico, ya que se utiliza la misma llave para encriptación y desencriptación.

- RSA. Llamada así en honor a sus creadores, Rivest, Shamir y Adleman, es el algoritmo de llave pública más popular. Se usa en firmas digitales y en encriptación.

- DSA. Digital Signatura Algorithm, es un algoritmo de clave pública. No puede ser utilizado para encriptación pero si para firmas digitales.

2.2 BLOQUES DE CONSTRUCCIÓN DE PROTOCOLOS

2.2.1 Introducción a los protocolos

La criptografía resuelve problemas que involucran confidencialidad, autenticación, integridad y atacantes. Se puede aprender mucho de técnicas y algoritmos de criptografía pero estas son teóricas a menos que puedan resolver un problema.

Un protocolo es una serie de pasos, que involucra a dos o más partes, diseñados para cumplir con un objetivo. Un protocolo criptográfico es para prevenir o detectar robo o modificación de información entre dos partes. Protocolos de arbitración, es el que se encarga de hacer que las dos partes en una comunicación acepten la información entregada como verdadera.

Ataques contra protocolos, los ataques criptográficos pueden ser contra protocolos, hay varios tipos de ataques, el ataque pasivo, cuando alguien externo puede acceder al protocolo, este tipo de ataque corresponde al ataque de solo texto cifrado. Por otro lado el atacante puede intentar modificar el protocolo para su propia ventaja, a esto se le llama un ataque activo.

2.2.2 Comunicación utilizando criptografía simétrica

Veamos a continuación los pasos que se dan en una comunicación utilizando criptografía simétrica:

Donde A es el emisor y B el receptor:

Paso 1: A y B acuerdan un determinado criptosistema.

Paso 2: A y B acuerdan utilizar una clave.

Paso 3: A toma su texto plano y lo encripta utilizando su algoritmo y la clave, dando como resultado el mensaje cifrado.

Paso 4: A envía el mensaje cifrado a B

Paso 5: B desencripta el mensaje cifrado con el mismo algoritmo y la misma clave que utilizó A.

Este protocolo asume que A y B confien entre sı. Además, tiene los siguientes problemas:

- Las claves deben ser distribuidas en secreto. Son tan valiosos como los mensajes que ellos encriptan, ya que si alguien conoce la clave automáticamente puede conocer los mensajes.
- Si se compromete una clave (robada, deducida, extorsionada), luego cualuier atacante puede desencriptar los mensajes con esta clave.
- Asumiendo que se utiliza diferentes claves para cada par de usuarios en una red, el número total de claves se incrementaria a medida que aumenten los usuarios.

2.2.3 Funciones de uno solo sentido: One way

Estas funciones no son protocolos propiamente dichos, pero las funciones de one-way son bloques de construcción para la mayoría de protocolos de seguridad. Son relativamente fáciles de generar en computadores. Es decir, dado x se puede deducir fácilmente f(x), pero no viceversa.

2.2.4 Funciones aleatorias de un solo sentido: One way hash

Son usadas principalmente para resolver el problema de la integridad de mensajes, al igual que la autenticidad de mensajes y de su origen. También llamado funciones de compresión, funciones de contracción, huella digital, resumen criptográfico, mensaje de integridad (MIC), y código de detección de modificación (MDC). Es otro bloque de construcción para diferentes códigos. Una función hash es también utilizada para la firma digital, ya que los documentos a firmar generalmente son demasiado grandes, la función hash les asocia una cadena de longitud de 160 bits, por ejemplo, que los hace más manejables para el propósito de la firma digital.

La función hash es una función matemática que toma una cadena de tamaño variable como entrada (pre-imagen) y la convierte en una cadena de caracteres de tamaño establecido (hash value). El objetivo aquí es hacer como una huella digital de la pre-imagen: generar un valor que indique si lo que se tiene supuestamente como pre-imagen, corresponde al valor de hash que se tiene. Este valor es único, es decir no se pueden

tener dos mensajes diferentes con el mismo valor de hash. Esta función es pública y su seguridad se basa en que es una función de solo una dirección, en otras palabras no puede ser desencriptada, sirve para verificación.

2.2.5 Comunicación utilizando criptografía de clave pública

Consiste en usar dos claves diferentes, una pública y otra privada, es difícil deducir la clave privada de la clave pública. Cualquiera que tenga la clave pública puede encriptar un mensaje pero no puede desencriptarlo. Solo lo puede descencriptar la persona con la clave privada. Aquí los pasos principales de este protocolo:

Paso 1: A y B acuerdan utilizar una clave pública en su criptosistema

Paso 2: B envía a A su clave publica.

Paso 3: A encripta su msg usando la clave publica de B y se la envía a B.

Paso 4: B desencripta el msg de A con su propia clave privada.

Una variante de esta comunicación es el criptosistema hibrido. Donde en la mayoria de implementaciones prácticas la criptografía de clave pública se usa para asegurar y distribuir claves de sesión, trabaja de la siguiente forma:

Paso 1: B envía a A su clave pública.

Paso 2: A genera una clave de sesion aleatoria K, encripta esta clave utilizando la clave publica de B y se la envía a B: $E_b(K)$

Paso 3: B desencripta el mensaje de A utilizando su clave privada para recuperar la clave de sesión: $D_b(E_b(K))= K$.

Paso 4: A y B encriptan su comunicación con la clave de sesion K.

2.2.6 Firmas digitales

Estas son algunas características de una firma:

- Tiene que ser autentica y esta convence al receptor que el emisor verdaderamente firmó este mensaje.

- No se puede utilizar nuevamente la misma firma, ya que esta es parte del documento, y la firma no puede sacarse de un documento para utilizarla en otro.

Después que se ha firmado un documento este no puede ser alterado.

A. Firma de documentos con un Criptosistema Simetrico y un Árbitro.

Consiste en los siguientes pasos, donde A es el emisor, B el receptor y M es el elemento arbitrador.

Paso 1: A encripta su mensaje a B con K y lo envía a C.

Paso 2: M desencripta el mensaje con K_a.

Paso 3: M toma el mensaje desencriptado y una confirmación de que recibió el mensaje de A, y lo encripta con K_b.

Paso 4: M envía el mensaje encriptado a B

Paso 5: B desencripta el mensaje con K_b. Ahora leerá ambos mensajes el de A y el mensaje de certificación por parte de M.

La pregunta ahora es como sabe M que el mensaje es de A y no de otra persona, M infiere esto de la encriptación del mensaje. Ya que solo M y A comparten la clave secreta de A (K_a).

Ahora veamos si B quiesiera mostrar el mensaje enviado por A a C, tendria que recurrir de todos modos a M:

Paso 1: B toma el mensaje de A y la certificación de M y lo encripta con K_b.

Paso 2: M desencripta el mensaje con K_b

Paso 3: M verifica su base de datos y certifica que el mensaje original es de A.

Paso 4: M reencripta el mensaje con la clave secreta que comparte con C, K_c.

Paso 5: C desencripta el mensaje con K_c. Ahora C puede leer el mensaje de A y la certificación de M.

Este protocolo es relativamente seguro pero consume mucho el procesamiento de M, ya que tiene que encriptar y desencriptar a cada momento. Además si alguien atacara el

programa utilizado por M, seria un caos ya que podría modificar todos los K's y todos los mensajes entre usuarios serían rechazados. Es decir este protocolo este bien propuesto teóricamente pero en la práctica solo trabaja para poco tráfico de red.

B. Firma de documentos con criptografía pública

Hay algoritmos de clave pública que pueden ser usados para firmas digitales. En algunos algoritmos, por ejemplo en RSA, puede utilizarse la clave pública o la clave privada para encriptación. Se puede encriptar un documento con la clave privada y se tiene una firma digital segura. En otros casos, por ejemplo DSA, hay un algoritmo separado exclusivamente para firmas digitales que no puede usarse para encriptación. En si el protocolo es simple:

Paso 1: A encripta el documento con su clave secreta.

Paso 2: A envía la firma a B

Paso 3: B desencripta el documento con la clave publica de A, y así verifica la firma.

Es protocolo es mejor que el anterior ya que no se requiere de ningún medio externo, como M. Así mismo esta firma digital también contiene a veces marcadores de tiempo (timestamps), de lo contrario B podría utilizar la firma de A mas de una vez.

C. Firma de documentos con criptografía pública y funciones de one-way hash.

En implementaciones prácticas, los algoritmos de clave pública son con frecuencia ineficientes para firmar documentos grandes. Para ahorrar tiempo, los protocolos de firmas digitales son implementados con funciones de one-way hash. Es decir A, en vez de firmar el documento, firma el hash del documento. Es este protocolo, se tiene que concordar en la función de one-way hash y el algoritmo de firma digital de antemano.

Paso 1: A produce un documento de one-way hash.

Paso 2: A encripta el hash con su clave privada, de esta manera firmando el documento.

Paso 3: A envía el documento y la firma del hash a B.

Paso 4: B produce un hash one-way con el documento que A envió. Luego, utilizando el algoritmo de firma digital, desencripta el hash firmado con la clave publica de A. Si el hash firmado concuerda con el hash generado, la comunicación es valida.

Con este protocolo la velocidad de transmisión aumenta drásticamente, además la probabilidad de que hayan 2 firmas encriptadas iguales es infima, además se tienen otras ventajas como que la firma puede permanecer separada del documento, y los requerimientos de espacio de memoria del receptor para la firma y el documento son mas pequeños.

D. Algoritmos y su terminología

Hay varios algoritmos de firmas digitales. Todas son algoritmos de clave pública con información secreta para firmar documentos e información publica para verificar las firmas. El proceso de firma es llamado "encriptación con clave privada" y el proceso de verificación es llamado "desencriptación con una clave pública". Sin embargo esto es válido únicamente para el algoritmo de RSA. Ya que los algoritmos tienen diferentes implementaciones, y estos pueden ser utilizados para firmas digitales pero no para encriptación. En lo sucesivo se tendrá lo siguiente:

Firma del mensaje M con clave privada: S_k (M)

Verificación de una firma con la clave pública: V_k (M)

El protocolo en si, por el que el receptor de un mensaje se convence de la identidad del emisor y la integridad del mensaje, es llamado autenticación.

E. No repudiación de firmas digitales

Se llama repudiación cuando por ejemplo A pueda engañar son su firma digital y no se puede hacer nada respecto a eso. Primero, A firma el documento normalmente, y dice que no lo hizo, luego publica anónimamente su clave privada, y la pierde en un lugar publico, finalmente dice que su firma ha sido comprometida y que otros están utilizando su firma simulando ser A, de este modo A rechaza todos los documentos que firmó.

2.2.7 Firmas digitales con encriptación

En este caso se combinan las firmas digitales con la criptografía de clave pública desarrollando un protocolo que ofrezca la seguridad de la encriptación y también la autenticidad de las firmas digitales.

Paso 1: A firma el mensaje M con su clave privada

$$S_a\,(M)$$

Paso 2: A encripta el mensaje firmado con la clave publica de B y la envía a B

$$E_b\,(S_a\,(M))$$

Paso 3: B desencripta el mensaje con su clave privada:

$$D_b(E_b(S_a(M)) = S_a(M)$$

Paso 4: B verifica con la clave pública de A y recupera el mensaje

$$V_a\,(S_a\,(M)) = M$$

Este protocolo obviamente es mas seguro que solo firmar el mensaje, es conveniente utilizar marcadores de tiempo también para que no se pueda utilizar una firma más de una vez.

2.3 PROTOCOLOS BÁSICOS

2.3.1 Intercambio de clave

Una técnica común de criptografía es encriptar cada conversación individual con una clave diferente. Esta es llamada una clave de sesión, por que se usa para una sesión de

comunicación en particular. Como hacer que esta clave de sesión llegue a manos de los implicados en la sesión es una situación complicada.

A. Intercambio de clave con criptografía simétrica

Este protocolo asume que A y B, comparten una clave, cada uno comparte una clave secreta con el Centro de Distribución de Claves, que llamaremos K, esta clave debe ser puesta al comienzo del protocolo.

Paso 1: A se comunica con K y le pide una clave de sesión para comunicarse con B.

Paso 2: K genera una clave de sesión aleatoria. Luego encripta dos copias de la clave, una con la clave de A y otra con la clave de B. L envía ambas copias a A.

Paso 3: A desencripta su copia con la clave de sesión.

Paso 4: A envía a B su copia de la clave de sesión.

Paso 5: B desencripta su copia de la clave de sesión.

Paso 6: B y A usan su clave de sesión para comunicarse en forma segura.

Este protocolo confía completamente en K, lo que no es muy recomendable, además se generaria un cuello de botella en K en caso de existir un tráfico considerable en la red.

B. Intercambio clave con criptografia de clave publica

Esto hace el protocolo de intercambio de claves más fácil de manejar:

Paso 1: A obtiene la clave pública de B de K.

Paso 2: A genera una clave de sesión aleatoria, la encripta usando la clave publica de B y la envía a B.

Paso 3: B desencripta el mensaje de A usando su clave privada.

Paso 4: luego A y B encriptan su comunicación utilizando la misma clave.

C. Atacante situado en medio de la comunicación

Supongamos que M quiere interceptar la comunicación, ya que no solo puede escuchar el mensaje entre A y B, también puede modificar y borrar mensajes y además generar otros mensajes totalmente nuevos. M puede imitar a B cuando envía mensajes a A, y viceversa, esto puede suceder de la siguiente forma:

Paso 1: A envía a B su clave pública. M intercepta su clave y envía a B la clave de M misma.

Paso 2: B envía a A su clave publica. M intercepta la clave y envía su propia clave.

Paso 3: Cuando A envía el mensaje de B, encriptado en la clave publica de B, en realidad estará enviándolo en la clave de M y viceversa.

En este tipo de ataque A ni B pueden saber que hay alguien en el medio físico en que se comunican.

D. Protocolo de enclavamiento

El protocolo de enclavamiento o interlock tiene una manera de prevenir el ataque anterior:

Paso 1: A envía su clave pública a B

Paso 2: B envía su clave pública a A.

Paso 3: A encripta su mensaje usando la clave publica de B. Envía la mitad del mensaje encriptado a B.

Paso 4: B encripta el mensaje usando la clave publica de A. B envía la mitad del mensaje encriptado a A.

Paso 5: A envía la otra mitad del mensaje encriptado a B.

Paso 6: B junta las dos mitades de A y las desencripta con su clave privada. B envía la otra mitad de su mensaje encriptado a A.

Paso 7: A pone las dos mitades del mensaje de B juntos y las desencripta con su clave privada.

Lo más resaltante es que la mitad del mensaje no tiene importancia sin su otra mitad; y asi no puede ser desencriptado. Y B no puede leer ninguno de los msgs de A hasta que ocurra el paso 6.

E. Intercambio clave con firmas digitales

Cuando se implementa firmas digitales también se evita el tipo de ataque del medio de la comunicación. En este caso M firma las claves públicas de A y B. Cuando A y B reciben las claves, cada uno verifica la firma de M. Ahora ellos conocen a quien pertenece la clave pública de la otra persona. En este protocolo no se tiene tan comprometido a M como en el de intercambio de claves con criptografía simétrica, ya que lo máximo que se podría comprometer a M es obtener la clave privada de M, esto no le permite desencriptar nada solo firmar falsamente.

F. Transmisión simultanea de claves y mensajes

A y B necesitan completar el protocolo de intercambio de claves antes de intercambiar sus mensajes. En este caso, A envía un mensaje M a B, sin ningún protocolo previo de intercambio de claves:

Paso 1: A genera una clave de sesión aleatoria K, encripta M usando K:

$$E_K (M)$$

Paso 2: A obtiene la clave pública de B de la base de datos.

Paso 3: A encripta K con la clave publica de B:

$$E_B (K)$$

Paso 4: A envía envía el mensaje encriptado y la clave de sesión encriptada a B:

$$E_K (M), E_B (K)$$

Paso 5: B desencripta la clave de sesión de A, utilizando su propia clave privada.

Paso 6: B desencripta el mensaje de A usando su clave de sesión.

Este sistema híbrido muestra como la criptografía de clave pública es frecuentemente utilizada para sistemas de comunicaciones. Además se puede combinar con marcadores de tiempo, firmas digitales y otros protocolos de seguridad.

G. Broadcast de claves y mensajes

Esto se usa cuando A necesita enviar un mensaje a varias personas a la vez. En este ejemplo. A enviara un mensaje a B, C y D.

Paso 1: A genera una clave de sesión aleatoria K, y encripta M con K.

E_K (M)

Paso 2: A obtiene la clave pública de B, C y D.

Paso 3: A encripta K con la clave publica de B, y con la clave publica de C y D.

E_B (K), E_C (K) y E_D (K)

Paso 4: A envía el mensaje encriptado y las claves encriptadas a todos los que quieran recibirlos:

E_B (K), E_C (K) y E_D (K) y E_K(M)

Paso 5: Solo B, C y D pueden desencriptar la clave K, cada uno utilizando su propia clave privada.

Paso 6: Solo B, C y D pueden desencriptar el mensaje de A usando K.

2.3.2 Autenticación

Es la forma en que un sistema reconoce que un usuario es quien dice ser, en el caso de las PCs por ejemplo se ingresa un login y un password.

A. Autenticación utilizando funciones de one-way

Mediante este tipo autenticación se establece que un host no debería conocer el password, solo debería saber y diferenciar si un password es valido o no. Esto es sencillo mediante este método, así en vez de mostrar passwords, el host graba funciones de one-way como passwords.

Paso 1: A envía su password al host.

Paso 2: El host ejecuta una función de one-way sobre el password.

Paso 3: El host comprar el resultado de la función de one-way con el valor el previamente almacenado.

B. Ataques tipo diccionario

De todos modos un archivo con passwords encriptados es vulnerable, ya que un atacante podría compilar una lista de 1000000 palabras comunes, y las procesa por la función de one-way, y haciendo una comparación entre sus resultados y el archivo de passwords puede deducir los passwords verdaderos, a esto se le llama un ataque diccionario. Para contrarrestar esto se adicionó el procedimiento de Salto, que es una cadena aleatoria que se concatena con los passwords antes de pasarlos por la función de one-way hash, en

este caso se almacena en la base de datos del host la cadena de SALT y la función de one-way hash.

C. Programa de autenticación SKEY

Es un programa de autenticación que confía en las funciones de one-way hash para su seguridad. Es fácil de explicar.

Por ejemplo, A ingresar un número aleatorio H, la computadora calcula f(H), f(f(H)), f(f(f(H))), etc, alrededor de 100 veces. A estos números los llama X_1, X_2, X_3,..., X_{100}. La computadora luego le entrega a A esta lista de números. La computadora luego guarda el X_{101} en su base de datos de usuarios para A. Luego la primera vez que A quiera ingresar a la PC tendra que ingresar su nombre y X_{100}, la computadora ejecuta $f(X_{100})$ que debe conincidir con X_{101} que tiene en su base de datos, luego borra el X_{101}, la segunda vez que A quiera ingresar la PC le pedirá el X_{99} y efectuará el $f(X_{99})$ y lo compara con el valor de X_{100} que tiene y asi sucesivamente.

D. Autenticación utilizando criptografía de clave publica.

El protocolo de autenticación de función de one-way aun con la característica de SALT, tiene problemas de seguridad. Cuando A envía su password al host, cualquiera que tenga acceso al medio físico por donde pasa su mensaje puede leer el password.

La criptografía de clave pública puede resolver este problema. El host mantiene un archivo de cada clave pública del usuario, todos los usuarios mantienen sus propias claves privadas. Así cuando alguien se logea, el protocolo procede como sigue:

Paso 1: El host envía a A una cadena aleatoria.

Paso 2: A encripta la cadena con su clave privada y la devuelve al host, con su nombre.

Paso 3: el host busca la clave pública de A en su base de datos y desencripta el mensaje usando esa clave publica.

Paso 4: si la cadena desencriptada concuerda con el que envió el host a A en el paso Paso 1:, A accede al sistema.

En esta comunicación nadie tiene acceso a la clave privada de A.

2.3.3 Autenticación e intercambio de claves

Si A y B quieren comunicarse desde diferentes puntos de red, y quieren conversa de forma segura, estos protocolos combinan la autenticación con el intercambio de claves para resolver el problema general de seguridad.

En adelante se utilizaran los siguientes símbolos para denotar a los elementos componentes que participan en los protocolos:

A: usuario A

B: usuario B

E_A: Encriptación con una clave que se comparte con A

E_B: Encriptación con una clave que se comparte con B

K: Clave de sesión aleatoria

L: Tiempo de vida

R_A, R_B: Número aleatorio elegido por A y B respectivamente.

A. Protocolo de Wide-Mouth Frog

Este es probablemente el protocolo más sencillo de claves simétricas, B y A comparten una clave secreta con el distribuidor de claves T. Las claves son utilizadas unicamente para distribución de claves pero no para encriptar ningún mensaje real entre los usuarios:

Paso 1: A concatena un marcador de tiempo, el nombre de B y una clave de sesión aleatoria y encripta todo el mensaje con la clave que comparte con T. A envía esto a T con su nombre de usuario:

$$A, E_A (T_A, B, K)$$

Paso 2: T desencripta el mensaje de A. Luego concatena un nuevo marcador de tiempo, el nombre de A, y la clave de sesión aleatoria. T encripta todo el mensaje con la clave que comparte con B, y la envía a B.

En este protocolo se asume que A es suficientemente competente como para generar buenas claves de sesión.

B. Protocolo Yohalom

En este protocolo A y B comparten una clave secreta con T.

Paso 1: A concatena su nombre y un número aleatorio y se lo envía a B.

A, R_A

Paso 2: B concatena el nombre de A, el número aleatorio de A, y su propio número aleatorio, y lo encripta con la clave que comparte con T. B luego lo envía a T con su nombre:

$B, E_B(A, Ra, Rb)$

Paso 3: T genera dos mensajes. El primero consiste en el nombre de B, una clave de sesion aleatoria, el número aleatorio de A y el número aleatorio de B, todo encriptado con la clave que comparte con A. El segundo número consiste en el nombre de A y la clave de sesion aleatoria, encriptada con la clave que comparte con B. T envía estos dos msgs a A:

Primer mensaje: $E_A (B, K, R_A, R_B)$ y

Segundo mensaje: $E_B(A, K)$

Paso 4: A desencripta el primer mensaje, extrae K y confirma que R_A, tiene el mismo valor que en el paso Paso 1:. A envía dos mensajes a B: el primero es el mensaje recibido de T, que contiene la clave encriptada de B. La segunda es R_B encriptado con la clave de sesion:

Primer mensaje: $E_B(A, K)$ y

Segundo mensaje: $E_K (R_B)$

Paso 5: B desencripta el mensaje con su clave privada, extrae K y confirma que R_B es el mismo que envio en el paso (2).

C. Protocolo Needham-Schroeder

Llamado así en honor a sus creadores, se ejecuta mediante los siguientes pasos:

Paso 1: A envía un mensaje a T que consiste en su nombre, el nombre de B y un número aleatorio.

$$A, B, R_A$$

Paso 2: T genera una clave de sesión aleatoria. Encripta el mensaje que consiste en la clave de sesión aleatoria y el nombre de A con la clave secreta que comparte con B. Luego encripta el valor aleatorio de A, el nombre de B, la clave y el mensaje encriptado con la clave secreta que comparte con A. Finalmente envía el mensaje a A con su clave pública:

$$E_A(R_A, B, K, E_B(K,A))$$

Paso 3: A desencripta el mensaje y extrae K. A confirma que R_A es el mismo valor que el que envió a T en el paso (1). Luego A envía a B el mensaje que T encriptó:

$$E_B(K, A)$$

Paso 4: B desencripta el mensaje y extrae K. Luego B genera otro valor aleatorio, R_B. B encripta el mensaje con K y lo envía a A.

$$E_K(R_B)$$

Paso 5: A desencripta el mensaje con K. Luego genera R_B -1 y lo encripta con K. Luego A envía el mensaje a B.

$$E_K (R_B -1)$$

Paso 6: B desencripta el mensaje con K y verifica que el valor es R_B -1

Esto es para prevenir ataques tipo "reply". En este ataque cualquiera que pueda obtener los mensajes anteriores puede usarlos después para engañar al protocolo, pero esto sería contrarestado con la presencia de R_B.

La inseguridad de este protocolo es que K puede ser usado más de una vez y si un K antiguo llega a manos de un atacante, todo lo que tendría que hacer es grabar el mensaje de A hacia B en el paso (3).

D. Protocolo Otway-Rees

Esto se usa cuando A necesita enviar un mensaje a varias personas a la vez. En este ejemplo. A enviará un mensaje a B, C y D.

Paso 1: A genera un mensaje que consiste de un número de índice, su nombre, el nombre de B y un número aleatorio. Todo esto lo encripta con una clave que comparte con T. A envía este mensaje a B con el número de índice, su nombre y el nombre de B:

$$I,A,B,E_A (R_A,I,A,B)$$

Paso 2: B genera un mensaje que consiste de un numero aleatorio, el numero de índice, el nombre de A, y el de B, todo esto se encripta en la clave que comparte con T. Todo

esto B envía a T, adicionalmente con el mensaje encriptado de A, el número de índice, el nombre de A y de B.

$$I,A,B,E_A\ (R_A,I,A,B),\ E_B\ (R_B,I,A,B)$$

Paso 3: T genera una clave aleatoria de sesión. Luego T crea dos mensajes. Uno es el número de A y su clave de sesión, encriptado en la clave que comparte T con A. El segundo mensaje es el número aleatorio de B y la clave de sesión, todo esto encriptado en la clave que comparten T y B. T envía estos dos mensajes más el número de índice a B: $I,E_A\ (R_A,K),\ E_B\ (R_B,K)$

Paso 4: B envía a A el mensaje encriptado en su clave, con el número de índice:

$$I,\ EA\ (R_A,\ K)$$

Paso 5: A desencripta el mensaje para recuperar la clave y el numero aleatorio. A confirma que ambos números no han cambiado en el protocolo.

Asumiendo que todos los números aleatorios coinciden y que el numero de índice no ha cambiado en toda la comunicación, entonces A y B están ahora convencidos de la identidad de cada uno, además tienen una clave secreta con que comunicarse.

Además se tienen los siguientes protocolos que cumplen con el intercambio de claves y la autenticación: Kerberos, Neuman-Stubblebine, DASS, Denning Sacco, WooLam, etc.

2.3.4 Criptografía de claves públicas para múltiples claves

La criptografia de clave publica usa dos claves. Un mensaje encriptado con una clave puede ser desencriptada con otra clave. Normalmente una clave es la privada y la otra es la clave publica. Sin embargo asumamos que A tiene una de estas claves y B tiene la

otra, ahora A puede encriptar un mensaje que solo B puede desencriptarlo y B encripta un mensaje que solo A pueda leerlo.

Ahora imaginemos una varianrw sw la criptografia de clave publica con tres claves: K_A, K_B, y K_C, distribuidas de la siguiente forma:

A..... K_A

B..... K_B

C..... K_C

D..... K_A y K_B

E..... K_B y K_C

F..... K_C y K_A

Esto se puede extender a n claves. Si un determinado grupo de claves se usa para encriptar el mensaje, entonces las otras claves se requieren para desencriptar el mensaje.

2.3.5 División de la clave secreta

Es un método en el que un mensaje se divide en varias partes de modo que las partes por si solas no significan nada ni pueden desencriptarse. En el esquema mas simple se divide un mensaje entre dos personas:

Paso 1: T genera una cadena de bits aleatoria R, del mismo tamaño que el mensaje M.

Paso 2: T hace un XOR entre M y R y genera S.

$$M \Theta R = S$$

Paso 3: T entrega R a A y S a B.

Para reconstruir el mensaje, A y B solo tienen que unir los mensajes que recibieron de T y efectuar un XOR entre ellos:

$$R \Theta S = M$$

Sin embargo, este protocolo tienen un problema, si cualquiera de las piezas finales se pierde y no se tiene acceso a T, ya no se puede recuperar el mensaje.

2.4 PROTOCOLOS INTERMEDIOS

2.4.1 Servicion de marcadores de tiempo (timestamping)

Es un poco complicado marcar un documento digital con la ultima fecha de creación o modificación, es decir el documento puede ser copiado y modificado indefinidamente sin que alguien se de cuenta. El protocolo de marcador de fecha debe cumplir al menos con las siguientes características:

- La data en si debe ser marcada con la fecha, sin importar el medio físico o en el que reside.
- Debe ser imposible cambiar un solo bit del documento sin que ese cambio se registre.
- Debe ser imposible marcar un documento con una fecha diferente a la real.

A. Solucion utilizando parte neutral:

Este protocolo necesita un árbitro T, que ejecute el marcado de fecha, de la siguiente forma:

Paso 1: A transmite una copia del documento a T.

Paso 2: T graba la fecha en que recibió el documento es su base de datos y hace una copia de seguridad del documento.

Este protocolo funciona pero tiene tres grandes desventajas: primero que no hay privacidad ya que A tiene que entregar una copia del documento siempre, segundo que la base de datos de T crece indiscriminadamente al tener que tener una copia de seguridad de todos los documentos, y tercero que el protocolo es totalmente dependiente de T ya que si T tuviera una pérdida en su base de datos central automáticamente invalidaría todos los marcadores de fecha de los documentos que tenia. Además A y T pueden confabular para que el documento tenga una fecha diferente a la real.

B. Solución mejorada utilizando parte neutral:

Los problemas detectados en el punto anterior se pueden resolver utilizando las funciones de one-way hash y firmas digitales:

Paso 1: A produce un hash de one-way del documento.

Paso 2: A transmite a T el hash.

Paso 3: T marca la fecha y hora en que recibio el hash en el hash y hace una firma digital del resultado.

Paso 4: T envía el hash firmado con el marcador de tiempo a A.

Con esto T ya no tiene que copiar todo el documento en su base de datos, sin embargo aquí todavía A y T pueden ponerse de acuerdo para poner una fecha diferente.

C. Protocolo de Enlace ó Linking:

Una manera de resolver el problema anterior es hacer un enlace entre el marcador de tiempo de A con otros marcadores de tiempo generados anteriormente por T. Estos marcadores de tiempo probablemente hayan sido generados por otros usuarios diferentes de A. Supongamos de N_A es el nombre de A, y el hash que A quiere que marquen con la fecha es H_n, y la marca de fecha anterior es T_{n-1}, el protocolo funciona de la siguiente forma:

Paso 1: A envía a T: H_n y N_A

Paso 2: T envía a A:

$T_n = S_K(n, N_A, H_n, t_n, I_{n-1}, H_{n-1}, T_{n-1}, L_n)$

Donde L_n es el resultado del hash de la siguiente información:

$L_n = H(I_{n-1}, H_{n-1}, T_{n-1}, L_{n-1})$

S_K indica que el mensaje es firmado con la clave privada de T. A se identifica 1como la que hizo el pedido de firma. El parámetro n indica la secuencia del pedido de marcado de tiempo, es el enesimo timestamp que T certifica.

Paso 3: Después de que T marca el siguiente documento, envía a A la identificación del creador de ese documento: I_{n+1}.

En este protocolo se elimina la probabilidad de que A y T puedan ponerse de acuerdo para colocar una fecha diferente, sin embargo si el documento de A es cuestionado, A tendra que recurrir al dueño del docuemnto que fue marcado con la fecha antes y después del suyo.

2.4.2 Canal subliminal

Canal subliminal, es un canal mediante el cual se tiene un medio que de antemano se sabe que no será privado y que desde antes de ser enviado y recepcionado sera revisado por una parte tercera. Este canal subliminal puede ir dentro del algoritmo normal de firma digital. Asi, el mensaje subliminal esta escondido en lo aparentemente es la firma digital, esta es una forma de ofuscación. Para esto, el algoritmo de firma de canal subliminal debe ser indistinguible de un algoritmo de firma digital ordinario para los individuos que quieran descifrar el mensaje. En general el protocolo trabaja como sigue:

Paso 1: A genera un mensaje inocuo, algo asi como aleatorio.

Paso 2: A, usando una clave secreta compartida con B, A firma el mensaje inocuo de manera que esconde su mensaje subliminal en la firma.

Paso 3: A envía este mensaje firmado a B por medio de W, donde W quiere ver exactamente el mensaje de A,

Paso 4: W lee el mensaje inocuo de A y chequea la firma, como no encuentra nada raro, pasa la información a B.

Paso 5: B revisa la firma del mensaje inocuo, confirma que el mensaje viene de A.

Paso 6: B ignora el mensaje inocuo y usando la clave secreta que comparte con A, extrae el mensaje subliminal.

La aplicación más obvia de estos mensajes es en una red de espionaje, donde todos envían y reciben mensajes firmados,

2.4.3 Firmas digitales no denegables

Las firmas digitales normales pueden ser copiadas exactamente. Algunas veces esta propiedad es útil. Como en la diseminación de anuncios públicos. Pero en otras puede ser un problema. Las firmas no denegables son apropiadas para las tareas que requieran que el distribuidor de un documento no pueda denegar que es el emisor o creador de un documento infectado con algún virus. Del mismo modo que una firma normal, la firma no denegable depende del documento firmado y la clave privada del firmante. Pero a diferencia de la firma normal, una firma no denegable no puede ser verificada sin el consentimiento del firmante. Sin embargo, se podría llamar a estas firmas como "firmas

no transferibles", ya que el firmante esta forzado a reconocer o denegar la firma. Los pasos a seguir en este proceso serian:

Paso 1: A se presenta a B con una firma.

Paso 2: B genera un número aleatorio y lo envía a A.

Paso 3: A hace el cálculo usando el número aleatorio y su clave privada y envía a B el resultado. A puede hacer este cálculo solo si la firma es válida.

Paso 4: B confirma esto.

Además hay un protocolo adicional que se encarga de que A pueda probar que no firmo un documento, y que no pueda negar falsamente una firma. Este protocolo no es perfecto, pero si es muy utilizado en diversas aplicaciones.

2.4.4 Firmas diseñadas para confirmación de mensajes

Supongamos que A tiene una compañía con un software que tiene mucho éxito, tanto que A pasa mas tiempo verificando firmas no denegables que haciendo nuevas características del software. A debería asignar una persona en particular, llamada C, para que se encargue de la verificación de la firma. Así, A podría firmar documentos con un protocolo no denegable. Pero las verificaciones serian manejadas completamente por C. Básicamente esto es posible con este protocolo. A puede firmar un documento de modo que B se convenza de que la firma es válida, pero que él no pueda convencer a una tercera parte, al mismo tiempo A puede designar a C para que sea la futura confirmadora

de su firma. Para esto A solo tendría que usar la clave publica de C. C puede aun verificar la firma de A si A no esta disponible. De este modo A se libra de tener que confirmar ella misma su firma, y quien hace todo el trabajo es C. C puede ser una oficina oficial de marcas registradas o una agencia de gobierno o un host, etc.

2.4.5 Firmas de Proxy

El protocolo anterior permite que el firmante designe a cualquier otro individuo para que pueda verificar su firma. Supongamos que el firmador, en este caso A, no pueda firmar por razones desconocidas, ¿como podria darle a B el poder de que él firma mensajes por A sin tener que darle su clave secreta? , esto se hace a través de las firmas de proxys, A puede darle un proxy a B, de modo que tenga las siguientes propiedades:

Distinguibilidad, cualquiera puede distinguir las firmas de proxy de las firmas normales.

Que no pueda ser forzado, solo el firmante original y el firmador proxy asignado, pueden crear una firma de proxy valida.

Desviación de la firma de proxy, un proxy firmador no puede crear una firma de proxy válida que no sea detectada como una firma de proxy.

Pueda verificarse.

Identificable, el firmante original puede determinar la identidad del proxy firmante de la firma del proxy.

No pueda ser denegado.

En algunos casos, se requiere una forma de identificación mas robusta, por ejemplo, que nadie pueda determinar la identidad del proxy que firma con solo ver la firma del mismo.

2.4.6 Firmas grupales

Tiene las siguientes propiedades:

Solo los miembros de un grupo pueden firmar los mensajes.

El receptor de la firma puede verificar que esta es una firma válida del grupo.

El receptor de la firma no puede determinar que miembro del grupo la firma.

En el caso de desacuerdo, la firma puede ser "abierta" para revelar la identidad del firmante.

Hay una variente de este tipo de firma, a esta se le llama:

Firma de grupo con una arbitro de confianza (Trusted Arbitrator):

Paso 1: T genera un grupo de pares de claves publicas y privadas y le da a cada miembro del grupo una lista de las claves privadas obviamente diferentes, y cada miembro recibe m pares de claves, o sea si hay n miembros, hay n*m claves en total.

Paso 2: T publica la lista de todas las claves públicas del grupo, pero T no dice a quien le pertenece que clave.

Paso 3: Cuando los miembros de un grupo quieren firmar un documento, cada persona escoge una clave del grupo de claves que tiene disponible.

Paso 4: Si alguien quiere verificar que la firma pertenece al grupo, mira la lista completa para las correspondientes claves públicas y verifica la firma.

Paso 5: Si hubiera una disputa, T sabe que clave publica le corresponde a que miembro.

El problema con este protocolo es que requiere de una tercera y confiable parte . T conoce las claves privadas de cada miembro y puede forzar firmas.

2.4.7 Procesamiento con data encriptada

Supongamos que A quiere saber la solución de alguna función f(x), para un valor particular de x. Pero por alguna razón desconocida no esta en capacidad de hacerlo, de modo que se lo encarga a B, lo malo es que B sabrá finalmente que es x. Para evitar esto se usa el procesamiento con data encriptada.

2.5 PROTOCOLOS AVANZADOS

2.5.1 Pruebas de conocimiento

Esto es cuando A tiene cierta información y tiene que probar lo que sabe a B, la desventaja es que si entrega la información a B, luego B sabrá la información de A, y podria divulgarlo, para evitar esto, un elemento confiable V, se encargaría de verificar, esto lo haría haciendo unas preguntas a A, que verificarían que A tiene esa información, pero sin que A entregue la información.

A. Protocolo básico de conocimiento.

Este protocolo se puede explicar utilizando un laberinto, como el de la Fig. 2.3, es decir, alguien que sabe la forma de abrir la puerta entre C y D, nadie más puede abrir esa

puerta. P sabe como abrir esta puerta y quiere demostrar que lo sabe a V. Esta es la manera como lo demuestra:

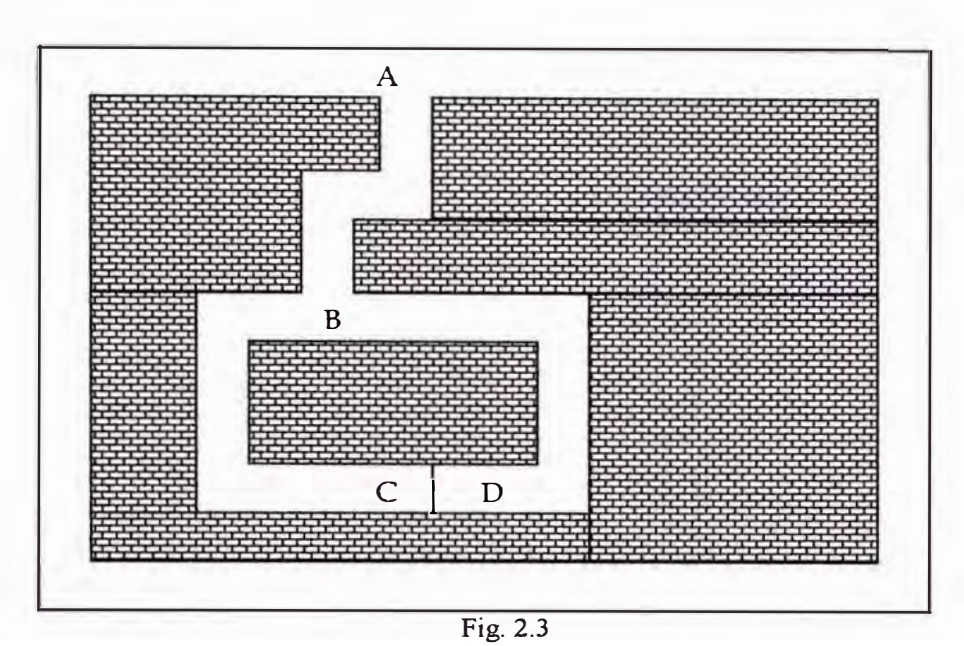


Fig. 2.3

Paso 1: V se queda en el punto A.

Paso 2: P va hacia el punto C o D.

Paso 3: Luego que P desaparece, V camina hasta el punto B.

Paso 4: V le pide a P que salga ya sea por el pasaje C y D.

Paso 5: P abre la puerta secreta.

Paso 6: Se demuestra que P tiene las claves de esa puerta, este procedimiento se repite n veces.

Sin embargo, V no puede convencer a otros individuos, de la veracidad de estas pruebas, ya que cabe la posibilidad de que V y P se pongan anticipadamente de acuerdo.

B. Isomorfismo gráfico:

El concepto de isomorfismo gráfico se puede explicar en base a la teoría gráfica. Una grafica es una red de líneas conectando diferentes puntos, si dos graficas son idénticas excepto por el nombre de los puntos, son llamadas **isomorfas**. Es extremadamente difícil verificar si dos graficas grandes son isomorfas. Asumamos que P sabe del isomorfismo entre las graficas G_1 y G_2, el siguiente protocolo demuestra a V que P tiene esta información:

Paso 1: P aleatoriamente permuta G_1 para producir otra grafica, H, que es isomorfa a G_1. Ya que P sabe del isomorfismo entre H y G_1, también sabe del isomorfismo entre H y G_2. Para cualquier otro individuo, encontrar un isomorfismo entre G_1 y H o entre G_2 y H es tan difícil como encontrarlo entre G_1 y G_2.

Paso 2: P envía H a V.

Paso 3: V pregunta a P:

1. Probar que H y G_1 son isomorfas; o
2. Probar que H y G_2 son isomorfas.

Paso 4: P responde. Luego, respectivamente:

1. Prueba que H y G_1 son isomorfas, sin probar que H y G_2 son isomorfas.
2. Prueba que H y G_2 son isomorfas, sin probar que H y G_1 son isomorfas.

Paso 5: P y V repiten los pasos anteriores n veces.

Ahora si P no supiera del isomorfismo entre G_1 y G_2, no podría crear una grafica H que sea isomorfa a ambas.

C. Ciclo Hamiltoniano

Un ciclo hamiltoniano es un camino continuo a lo largo de las líneas de una grafica que pasa por cada punto exactamente una vez, encontrar el ciclo hamiltoniano es una tarea complicada. Supongamos que P sabe el ciclo Hamiltoniano de la grafica H, y quiere demostrar que lo sabe sin tener que entregárselo a V. V conoce G, pero no el ciclo hamiltonaniano. Se siguen los siguientes pasos:

Paso 1: P aleatoriamente permuta G. Mueve los puntos y cambia sus nombres para hacer una nueva grafica H. Ya que H y G son topologicamente isomorfas, si P sabe el ciclo hamiltoniano de G fácilmente conocerá el de H, Luego encripta H para obtener H'.

Paso 2: P le da a V una copia de H'.

Paso 3: V le pide a P, que:

1. Pruebe que H' es una encriptación de alguien isomorfo de G, o
2. Mostrar a P un ciclo hamiltoniano para H.

Paso 4: P compila, y luego:

1°. Prueba que H' es una encriptación de un isomorfo de G, revelando las permutaciones y desencriptandolo todo, sin mostrar un ciclo hamiltoniano ni para H, ni a G.

2°. Muestra un ciclo hamiltoniano de H desencriptando solo las líneas que constituyen un ciclo hamiltoniano, sin dar a conocer que G y H son topologicamente isomorfas.

Paso 5: P y V repiten los pasos 1 a 4 n veces.

2.5.2 Pruebas de conocimiento de identidad

Este mecanismo se trata de probar la identidad de un individuo con propiedades natas y evidentes del mismo, sin ningún método físico. Sin embargo existen ciertos abusos de este método: supongamos que A juega con dos jugadores de ajedrez a la vez, X y Y. Donde A juega con Y con las mismas jugadas que hace X. Este tipo de fraude trabaja mediante lo siguiente, X le prueba a A que es X, utilizando esto A, al mismo tiempo prueba a Y que es X.

2.5.3 Firmas ciegas

Una característica esencial de los protocolos de firmas digitales es que el firmante conoce que es lo que esta firmando, excepto cuando se necesita absoluta reserva. Hay métodos por los que el firmante puede saber parcialmente, no en su totalidad, lo que esta firmando.

A. Firmas completamente ciegas

Supongamos que B es un notario público, A quiere que B firme un documento, pero no quiere que B vea que documento es. B no quiere saber que tiene el documento de A, solo

quiere certificar que lo notarió en cierto momento. Esto se da de la siguiente manera:

Paso 1: A multiplica el documento original por un factor aleatorio. Este valor es llamado ***factor blinding***.

Paso 2: A envía el documento alterado a B.

Paso 3: B firma el documento alterado.

Paso 4: A divide la firma entre el factor blinding, dando como resultado solamente la firma de su propio documento.

CAPÍTULO III

TÉCNICAS DE CRIPTOGRAFÍA

3.1 TAMAÑO ADECUADO DE LA CLAVE

3.1.1 Tamaño de clave simétrica

La seguridad de un criptosistema simétrico, reside en primer, y más importante lugar, en la resistencia del algoritmo y en la longitud de la clave, lo último es es mas fácil de demostrar. Asumamos que la resistencia del algoritmo es perfecta. Esto extremadamente difícil de lograr en la práctica, decir "perfecto" se refiere a que no hay mejor manera de romper el criptosistema que intentar descifrarla con cada clave posible, esto se conoce como ataque tipo brute-force.

Para iniciar este ataque, el criptoanalista necesita una pequeña cantidad de texto cifrado y su texto plano correspondiente; un ataque tipo brute-force es un ataque de "texto plano conocido". Para un bloque cifrado, el atacante necesitará un bloque de texto cifrado y el

plano texto correspondiente: generalmente 64 bits. Al tener acceso al texto cifrado, el atacante podrá verificar que tipo de formato tiene este, ya sea un archivo de Word o la cabecera de un mensaje electrónico o un directorio de UNIX. Todos estos formatos tienen algunos bytes predefinidos, de modo que el atacante no necesita acceder a grandes cantidades de texto cifrado para llevar a cabo su ataque. Calcular la complejidad de un ataque de tipo brute-force es sencillo. Si la clave es de 8 bits, hay 2^8 o 256 posibles claves. Por lo tanto, se necesitan a mas 256 intentos para encontrar la clave correcta. Paro si la clave es de 64 bits, tomara 2^{64}, o sea será casi imposible intentar encontrar la clave procesando con una sola PC en un tiempo pequeño. Sin embargo, un algoritmo debe ser tan seguro que no haya otra forma de romperlo que con un ataque de brute-force.

A. Redes neuronales

Las redes neuronales no son tan útiles para los criptoanalistas, primeramente por la forma del espacio de solución. Las redes neuronales trabajan mejor con problemas que tienen una continuidad de soluciones. Esto permite que la red neuronal aprenda, proponiendo ideas mejores cada vez. Por el contrario romper un algoritmo no es algo que haya que aprender de la solución anterior. En conclusión, las redes neuronales no trabajan bien para lo que se refiere a romper un algoritmo.

B. Viruses

La mayor dificultad en hacer que miles de computadoras empiecen a trabajar en un ataque de "brute-force" es convenciendo a los miles de administradores que participen. Hay varias formas de lograr esto, ya sea entrando y craqueando las computadoras, pero esto consume mucho tiempo, otra forma muy común es expandiendo un virus que consista en el programa de cracking. El atacante escribe y expande un virus, este virus no formatea el disco duro ni borra archivos, lo que hace es formar parte de un criptoanalisis a otro sistema cuando la pc este desocupada.

3.1.2 Tamaño de clave pública

Cuando se multiplican dos números primos grandes se esta efectuando una función de one-way, es fácil multiplicar los números y obtener un productos pero difícil de factorizar este producto y recuperar los dos números primos grandes. La criptografía de clave pública utiliza esta idea para hacer una función de one-way con trap-door. En realidad, auque no es cierto, factorizar se considera como un problema difícil.

Los algoritmos de encriptación de claves públicas de hoy en día se basan en la dificultad de factorizar grandes números primos, estos algoritmos son también susceptibles a ataques de tipo brute-force, pero de diferente manera, ya que para romper estos algoritmos no hay que intentar cada clave posible, en vez de esto hay que tratar de factorizar el número, el inconveniente es que si el número es muy pequeño no se tiene suficiente seguridad y si el número es lo suficientemente grande, se tiene seguridad contra la capacidad de los procesadores de hoy en día hasta un futuro cercano

proyectado, en la tabla 3.1 se analizan cuanto tomharía factorizar números de varios tamaños.

Tamaño recomendado de claves de claves-públicas (en bits)			
Año	Vs. Individuos	Vs. Corporaciones	Vs. Gobiernos
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1538	2048
2015	1536	2048	2048

Tabla 3.1

3.1.3 Comparación entre la resistencia de las claves simétricas y públicas

Cualquier sistema siempre es atacado por su parte mas débil, si se está diseñando un sistema que utiliza criptografía de claves simétrica y criptografía publica, los tamaños de cada tipo de criptografía debe escogerse de modo que sea igualmente difícil de atacar el sistema por cualquiera de los dos mecanismos.

Tamaños de claves simétricas y de clave-pública con resistencia similar hacia una ataque de brute-force	
Tamaño de clave simétrica	Tamaño de clave de clave-pública
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

Tabla 3.2

Ya que no tiene sentido utilizar un algoritmo simétrico con una clave de 128 bits junto a un algoritmo de clave pública con una clave de 386 bits. En la siguiente tabla se lista módulos de clave pública cuya dificultad de factorizacion se iguala a la dificultad de hacer un ataque de frute-force a las claves simétricas. En general, se debe escoger una resistencia de clave pública que sea más segura que la resistencia de clave simétrica. Las claves públicas generalmente están más expuestas, y son utilizadas para proteger más información.

3.1.4 Ataques contra funciones de un solo sentido: one-way hash

Hay dos formas de atacar en contra de una función hash:

- La forma mas obvia, dado el hash de un mensaje H (M), un adversario querría crear un documento M' de modo que H (M') = H (M).
- Este ataque es más elaborado. Un adversario atacante deberá encontrar dos mensajes aleatorios, M y M' de modo que se cumpla que H (M)=H (M'), esto es una colisión y es un ataque mas fácil que el anterior.

La paradoja del cumpleaños es un problema clásico de la estadística, cuantas personas deben están en una habitación para que exista la probabilidad de que una de ellas compartan el mismo día de cumpleaños con alguien externo?, la respuesta es 253. Pero si la pregunta es cuantas personas debe haber para que exista la probabilidad de que algún par de estas personas compartan el mismo día de cumpleaños, solo 23. Con solo 23 personas en una habitación, hay todavía 253 pares de personas en una habitación. Se

explicó lo anterior por que esta paradoja se puede comparar a los ataques, ya que encontrar una persona de un cumpleaños especifico se asemeja al primer ataque y encontrar dos personas que compartan un cumpleaños es similar al segundo ataque, comúnmente llamado ataque de cumpleaños (birthday attack).

3.2 MANEJO Y ADMINISTRACIÓN DE LA CLAVE

La administración de claves es la parte más difícil de la criptografía. Diseñar algoritmos y protocolos criptográficos seguros no es fácil, pero mantener la clave en secreto es mucho más difícil.

Los criptoanalistas frecuentemente atacan los criptosistemas simétricos y de clave publica mediante la administración de la clave.

3.2.1 Generación de claves

La seguridad de un algoritmo se basa en su clave. Si se esta utilizando un proceso débil criptográficamente hablando para generar una clave, en consecuencia todo el sistema será débil. DES tiene una clave de 56 bits.

Número de claves disponibles					
	4-Byte	5-Byte	6-Byte	7-Byte	8-Byte
Letras minúsculas (26)	460000	$1.2*10^{7}$	$3.1*10^{8}$	$8.0*10^{9}$	$2.1*10^{11}$
Letras minúsculas y dígitos 36	1700000	$6.0*10^{7}$	$2.2*10^{9}$	$7.8*10^{10}$	$2.8*10^{12}$
Caracteres alfanuméricos 62	$1.5 * 10^{7}$	$9.2*10^{8}$	$5.7*10^{10}$	$3.5*10^{12}$	$2.2*10^{14}$
Caracteres imprimibles 95	$8.1*10^{7}$	$7.7*10^{9}$	$7.4*10^{11}$	$7.0*10^{13}$	$6.6*10^{15}$
Caracteres ASCII 128	$2.7*10^{8}$	$3.4*10^{10}$	$4.4*10^{12}$	$5.6*10^{14}$	$7.2*10^{16}$
Caracteres ASCII de 8 bits 256	$4.3*10^{9}$	$1.1*10^{12}$	$2.8*10^{14}$	$7.2*10^{16}$	$1.8*10^{19}$

Tabla 3.3

Si se implementa adecuadamente, cualquier cadena de 56 bits podría ser la clave, hay 256 posibles claves. Ahora tengamos en cuenta que hay programas que solo aceptan claves ASCII, de este modo fuerzan que el bit mas significativo de cada byte sea cero, así en un ataque del tipo de brute-force se tendrá los parámetros especficados en la tabla 3.3 si se prueban un millón de claves por segundo.

Hay también un tipo de ataque llamado "ataque de diccionario" por que tiene entre sus primeras posibilidades las palabras más comunes, que incluyen:

- nombres de hombres y mujeres , unos 16000
- lugares
- nombres de gente famosa
- títulos, personajes y lugares de historias de ciencia ficción o películas.
- Criaturas míticas.
- Deportes
- Series conocidas como "lumamijuvisado" o efmamjj..., etc.
- Palabras comunes y conocidas de todos los idiomas.

Un ataque de diccionario es mucho más poderoso cuando es utilizado contra un grupo de usuarios, ya que la probabilidad de que alguien de este grupo tenga un password relativamente sencillo es muy grande.

3.3 MODOS Y TIPOS DE ALGORITMOS

Hay básicamente dos tipos de algoritmos simétricos, los cifrados por bloques y los cifrados por cadenas de caracteres (strings). Los algoritmos cifrados por bloques operan

sobre bloques de texto plano o cifrado, estos son usualmente de 64 bits, sin embargo a veces es mayor. Los cifrados por cadenas, operan sobre arreglos de texto plano y texto cifrado. Con el cifrado por bloques, el mismo bloque de texto plano se encriptaría siempre en el mismo bloque cifrado, si se utiliza la misma clave. En cambio con el cifrado por cadenas, el mismo texto plano de bits o bytes se encriptaría a un bit o byte diferente cada vez que este se encripte. Un modo criptográfico normalmente combina un cifrado básico, algún tipo de realimentación, y algunas operaciones simples.

3.3.1 Modo del libro de código electrónico (ECB)

Es el modo mas obvio de utilizar un cifrado por bloques: un bloque de texto plano se encripta formando un bloque de texto cifrado. Ya que el mismo bloque de texto plano siempre se encriptará al mismo bloque de texto cifrado, es teóricamente posible crear un libro de códigos de textos planos y su correspondiente texto cifrado, si el tamaño del bloque es de 64 bits, el libro de códigos tendrá 2^{64} entradas. El problema con el modo ECB es que si un atacante tiene el texto plano o cifrado de varios mensajes, puede empezar a completar el libro de códigos perteneciente a este cifrado sin saber la clave del código. En situaciones más reales, hay fragmentos de mensajes que se tienden a repetir, esto es cuando mensajes diferentes tienen secuencias de bits comunes, por ejemplo los mensajes generados por computadoras como el correo electrónico tiene estructuras regulares y conocidas. Esta vulnerabilidad es mayor al comienzo y fin de los mensajes, donde las cabeceras conocidas contienen información sobre el emisor, receptor, fecha y demás datos importantes.

3.3.2 Modo de concatenamiento de bloques cifrados (CBC)

Tambien llamado chaining, adiciona el mecanismo de realimentación al cifrado por bloques, que se reduce en los siguiente: el resultado de la encriptación del bloque previo se realimenta en la encriptación del bloque siguiente. En el modo CBC, el texto plano se opera en XOR con el bloque cifrado previo antes de encriptarlo.

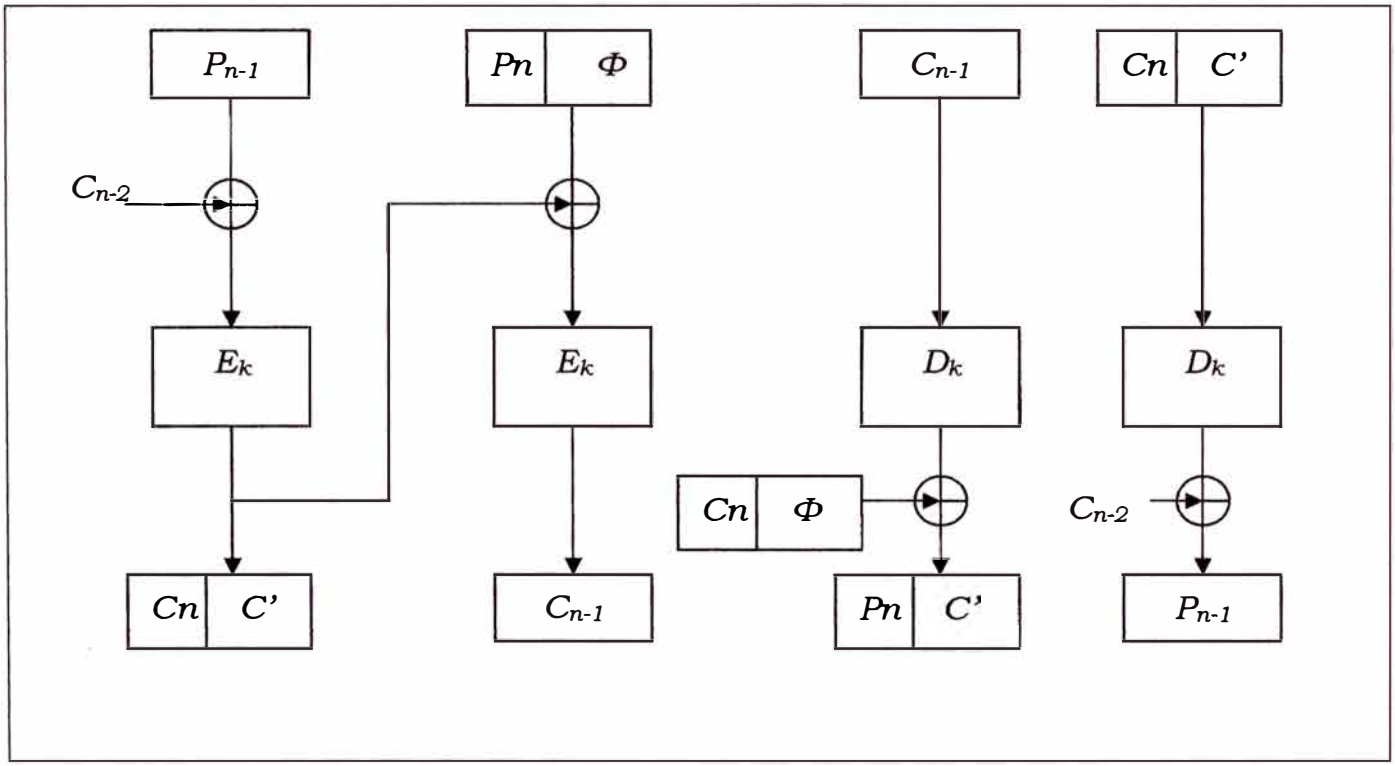


Fig. 3.1

La Fig. 3.1 muestra claramente el método utilizado aquí. De esta forma el modo CBC fuerza a que dos bloques planos idénticos se encripten en texto cifrado diferentes, sin embargo los textos cifrados en conjunto de dos mensajes idénticos completos serán idénticos, de esta forma para prevenir que un atacante pueda deducir de los mensajes comunes que casi siempre empiezan con "From:" se implementó un bloque aleatorio

llamado "Vector de inicialización – IV", donde IV no tiene ningún significado en especial, se genera aleatoriamente.

La desventaja de este código es que si ocurriera algún error en el bloque de texto plano, este error se propaga a través de todos los textos cifrados a partir del primer texto errado, y como se sabe un error de bit fácilmente resulta de un medio de transmisión con ruido o interferencia.

3.3.3 Cifrado por cadenas de caracteres

El modo de cifrado por cadenas convierte de texto plano a cifrado pero operando 1 bit a la vez. La implementación más simple de un cifrado por cadena se muestra en la Fig.3.2:

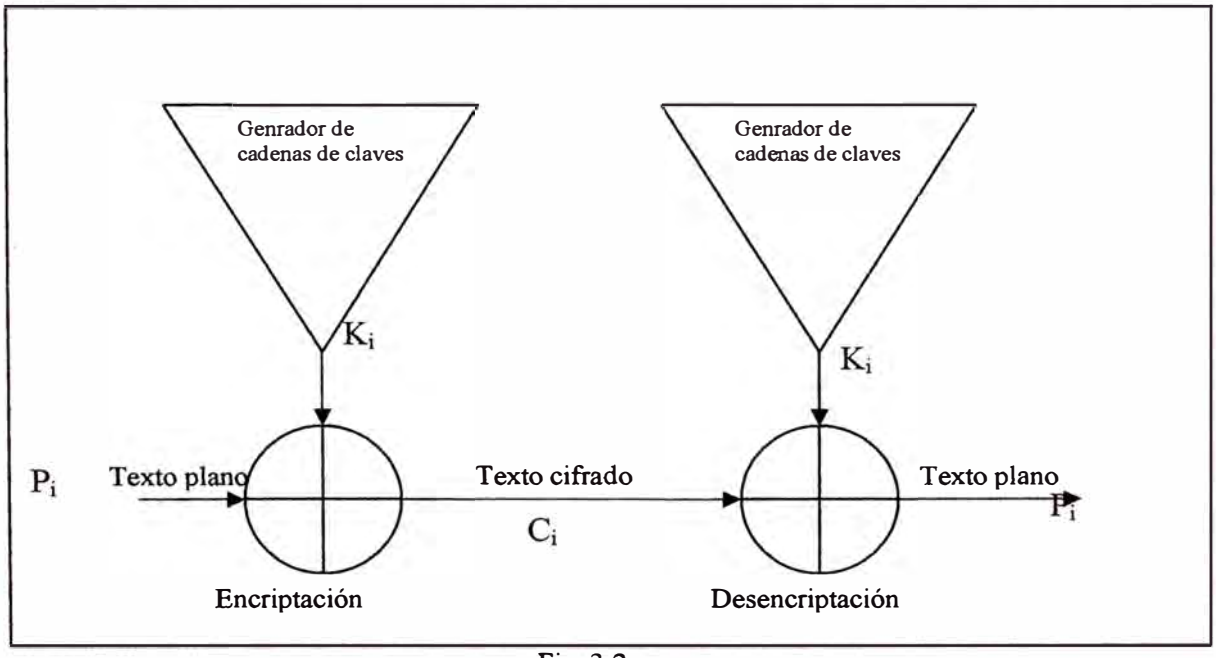


Fig. 3.2

Un generador de claves entrega como salidas, cadenas de bits, k_1, k_2, k_3, etc. Estas cadenas de claves se pasa por un XOR con las cadenas de bits de textos planos p_1, p_2, como se muestra en la figura de modo que: $c_i = p_i \Theta k_i$

Para desencriptar será necesario efectuar: $p_i = c_i \Theta k_i$

La desventaja es que el sistema de seguridad de este modo depende por completo del generador de claves.

3.3.4 Cifrado de cadenas de autosincronismo

Para el cifrado de cadenas autosincronizada, cada bit de clave de cadena es una función o un número predeterminado proveniente de los bits del texto cifrado anterior. Este concepto se puede observar mejor en la figura siguiente. La complejidad criptográfica se encuentra en la salida de la función, que procesa su estado interno y genera un bit de clave de cadena. Como el estado interno depende totalmente de los n anteriores bits cifrados, la desencriptacion del generador de cadenas de claves automáticamente sincronizará con el generador de cadenas de claves después de recibir n bits de texto cifrado.

Este modo también es vulnerable a ataques, por ejemplo si alguien que tiene acceso al medio de transmisión graba algunos bits cifrados, luego de un tiempo substituye esta grabación con el tráfico del momento, después de un momento el receptor sincronizará

con la data enviada por el atacante y la empezará a desencriptar como si fuera data normal, y el receptor no tiene modo de saber que este texto no es verdadero a menos que se utilicen marcadores de tiempos.

3.3.5 Modo cifrado realimentado

También llamado "cipher-feedback mode", los bloques cifrados también pueden ser implementados como cifrados de cadena autosincronizados. Con el modo CBC, la encriptación no empieza hasta que se reciba un bloque de datos completo. Este es un problema es algunas aplicaciones de red.

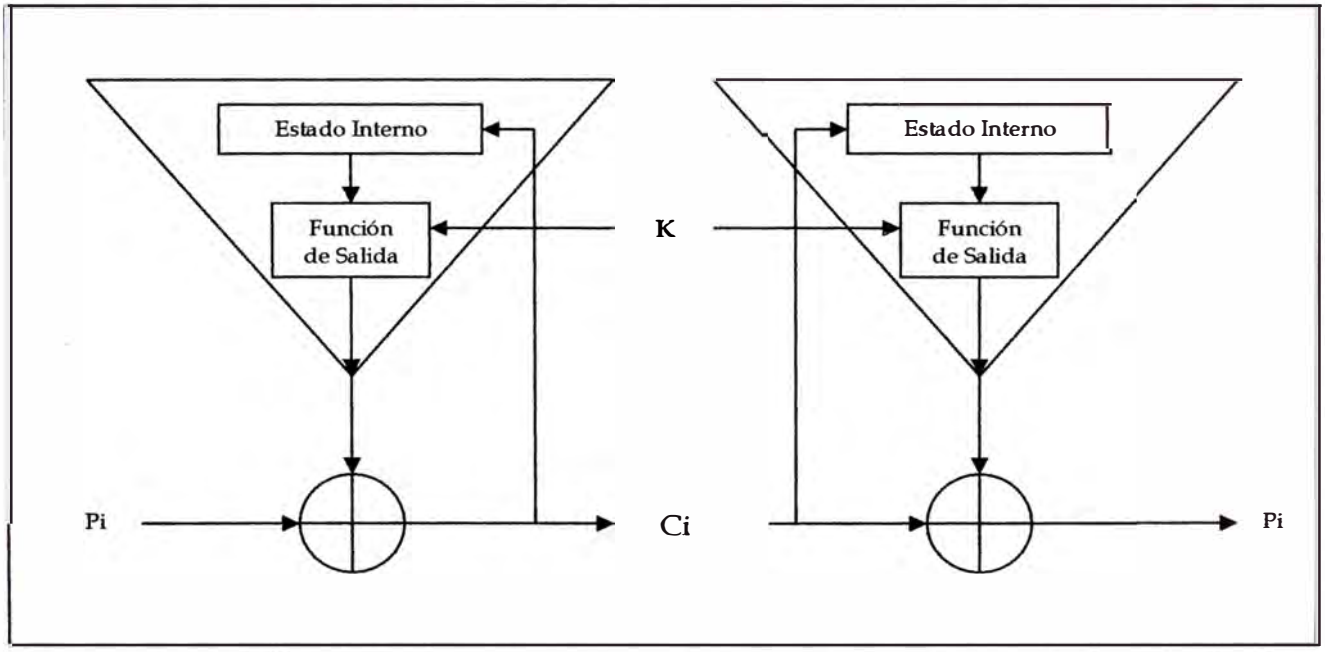


Fig. 3.3 Generador de clave auto sincronizado

En una ambiente de red seguro, por ejemplo, un terminal debe ser capaz de transmitir cada carácter al host. En el modo CFB la data puede ser encriptada en unidades menores

al tamaño de un bloque. En la Fig. 3.4 se muestra un modo de CFB de 8 bits trabajando con un algoritmo de bloque de 64 bits.

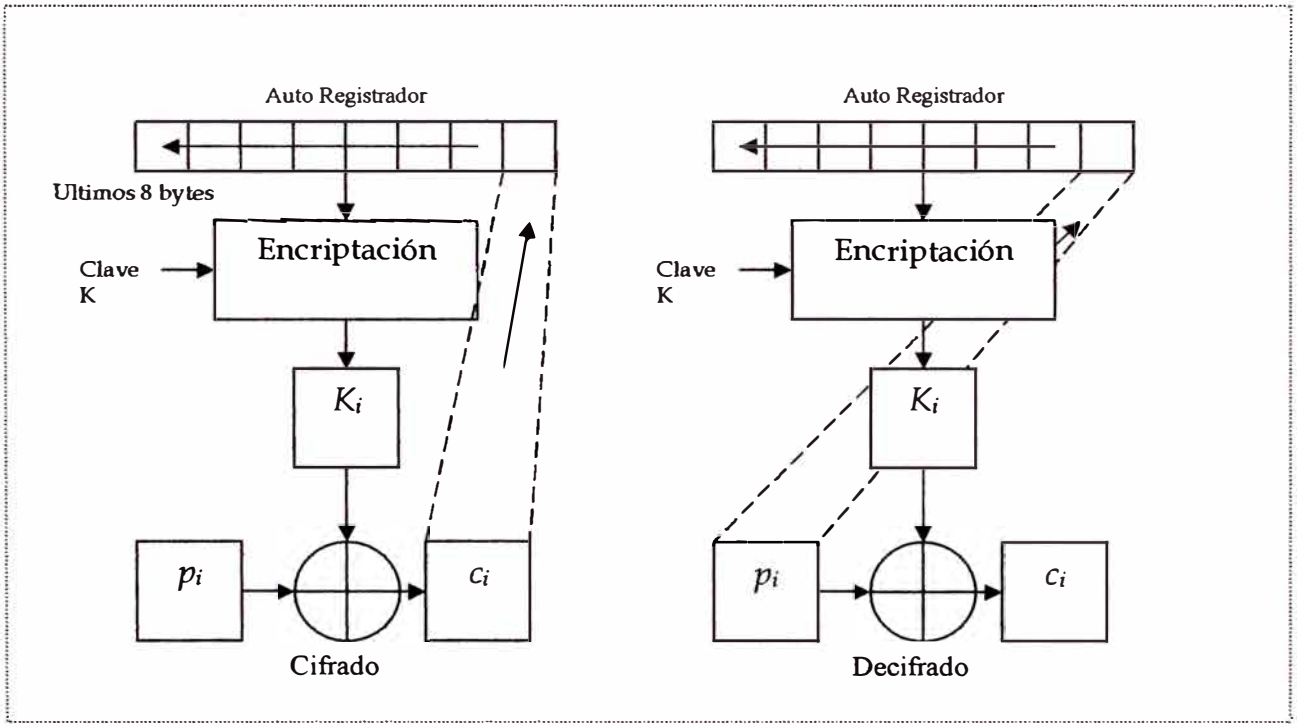


Fig. 3.4 Modo CFB

Un algoritmo de bloques en modo CFB opera en una cola el tamaño del bloque entrante. Inicialmente, la cola de datos se determina con un Vector de inicialización (IV), como en el modo CBC. La cola se encripta y los 8 bits de la izquierda del resultado se aplica un XOR con los primeros 8 bits de caracteres de texto plano para convertirse en los primeros 8 bits de caracteres del texto cifrado.

Este carácter puede ahora ser transmitido. Los mismos 8 bits son también movidos a las ocho posiciones mas a la derecha de la cola, y todos los otros bits son movidos ocho bits

a la izquierda. La desencriptacion es el reverso de este proceso. Un error de texto cifrado afecta al correspondiente bit de texto plano y a todo el bloque siguiente, los errores de sincronización de un tamaño de bloque completo son recuperables.

3.3.6 Modo de salida realimentada

También llamado Output-feedback (OFB) es un método mediante el que se procesa un bloque de texto cifrado como una cadena cifrada sincronizada. Es similar al modo CFB, excepto que los n bits del bloque del resultado previo se mueven hacia la posición más a la derecha de la cola, como se observa en la Fig. 3.5. La desencriptacion es el reverso de este proceso.

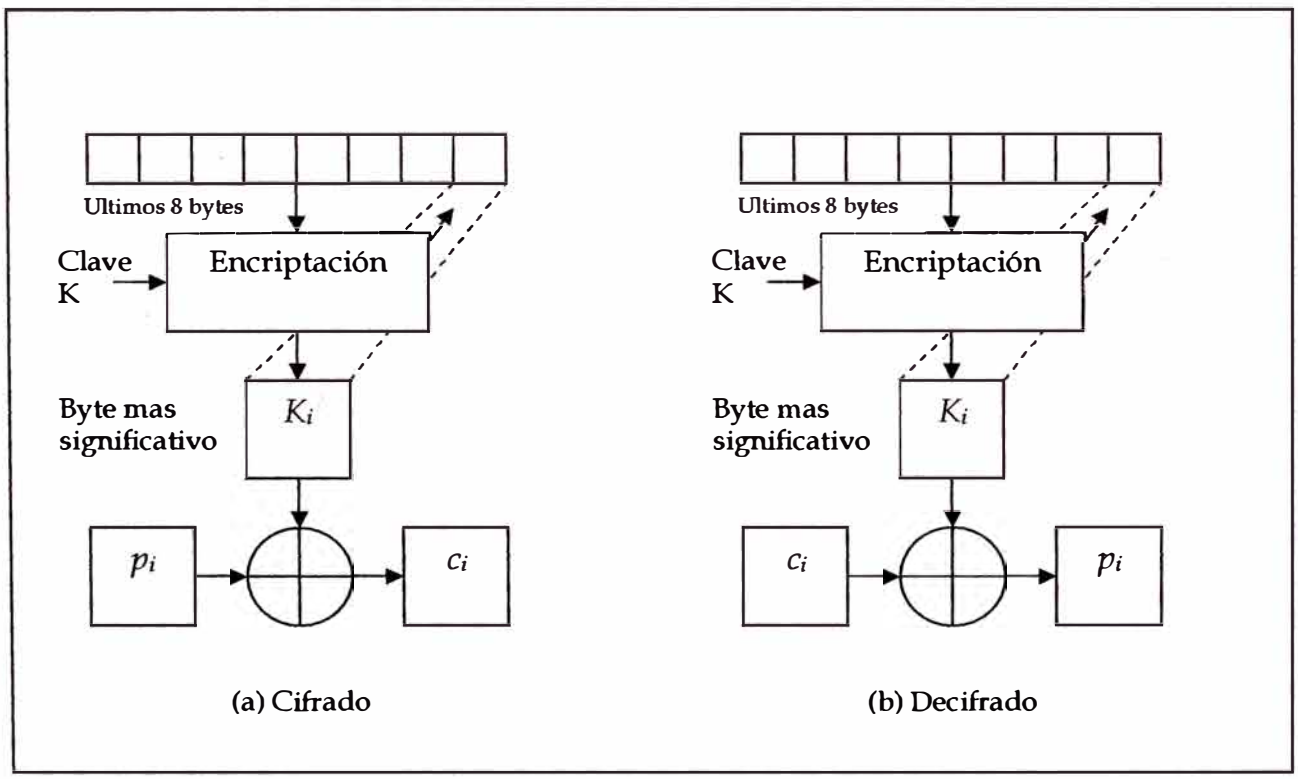


Fig.3.5 Modo de salida realimentada

3.4 USO DE ALGORITMOS

Si se analiza la seguridad, seguridad de datos, seguridad de comunicaciones, seguridad de la información, todo esto es una cadena, ya que la seguridad de todo el sistema es tan fuerte como la parte más insegura del mismo. Todo tiene que ver con seguridad, los algoritmos criptográficos, los protocolos, la administración de la clave, etc. Si los algoritmos son perfectos pero el generador de claves es predecible, entonces cualquier hábil criptoanalista atacará el sistema a través de la generación de números aleatorios.

La criptografía es solo una parte de la seguridad total, y frecuentemente una parte muy pequeña. Es la matemática de hacer seguro un sistema lo cual es diferente de hacer un sistema realmente seguro. Aun la NASA ha admitido que la mayoría de fallas de seguridad en esta área de interés se deben a fallas de implementación, y no a fallas en algoritmos y protocolos, en estos casos no importó cuan buena era la criptografía, si no todo el criptosistema en si.

3.4.1 Seleccionando el algoritmo adecuado

Cuando se debe evaluar parámetros para escoger el algoritmo adecuado, se tienen varias alternativas:

Se puede escoger un algoritmo publicado, basándose en la creencia de que un algoritmo publicado ha pasado por el escrutinio de varios criptógrafos, si nadie lo ha roto todavía, entonces debe ser bueno.

Se puede confiar en un programador basándose en que un programador conocido tiene una reputación a mantener y que es poco probable que arriesgue su reputación vendiendo equipos o programas con algoritmos inferiores.

Se puede confiar en un consultor privado basándose en que un consultor imparcial esta mejor equipado para hacer una evaluación confiable de diferentes algoritmos.

Se puede confiar en el gobierno, basándose en que el gobierno es totalmente confiable y no engañaría a sus ciudadanos.

Se puede escribir uno mismo los algoritmos, basándose en las habilidades criptográficas.

Cualquiera de estas alternativas es factible pero la primera es la más apropiada.

3.4.2 Criptografía pública versus criptografía simétrica

El debate sobre si es mejor utilizar la criptografía simétrica o publica nació desde que se invento la criptografía pública. El debate asume que los dos tipos de criptografía pueden ser comparados en iguales modos, sin embargo esto no es así, ya que se basan en diferentes tipos de filosofías, la criptografía simétrica es mejor para encriptas data, es más rápida y no es susceptible a ataques de texto cifrado escogido. La criptografía de

clave pública puede hacer cosas que no puede hacer la criptografía simétrica, además es mejor para el manejo de las claves y varios protocolos de seguridad.

3.4.3 Encriptando canales de comunicación

Este es el típico problema ya visto anteriormente, entre A y B, si A quiere enviar un mensaje seguro a B, A encripta el mensaje. En teoría la encriptación toma lugar en cualquier capa del modelo de comunicaciones de la capa OSI. En la práctica, la encriptación toma lugar ya sea en los niveles mas bajos (capas uno y dos) o en las capas mas altas. Si toma lugar en las capas mas bajas, se llama encriptación link-by-link, y si se da en las capas altas se llama encriptación end-to-end. Cada uno tiene sus propias ventajas y desventajas. En la tabla 3.4 se muestra los diferentes tipos de algoritmos y sus propiedades:

Clases de algoritmos				
Algoritmo	Confidencialidad	Autenticación	Integridad	Administración de clave
Algoritmo de encriptación simétrica	Si	No	No	Si
Algoritmo de encriptación de clave publica	Si	No	No	Si
Algoritmos de firma digital	No	Si	Si	No
Algorimos de key-agreement	Si	Opcional	No	Si
Funciones de one-way hash	No	No	Si	No
Mensajes de código de autenticación	No	Si	Si	No

Tabla 3.4

A. Encriptación Link-by-Link

El lugar más fácil para añadir la encriptación es la capa física, como se muestra en la figura 3.6. Las interfaces con la capa física son generalmente estandarizadas y es fácil

conectar dispositivos de encriptación en este nivel. Estos dispositivos hardware encriptan toda la data que pasa a través de ellos, incluyendo, data, información de routing, e información de protocolo. Se pueden utilizar en cualquier tipo de enlace de comunicación digital. Este tipo de encriptación es efectiva, ya que todo esta encriptado, un criptoanalista no puede obtener información acerca de la estructura de la información, y no tiene idea de la quien esta hablando a quien, que tan largos son los mensajes, etc. Esto es llamado "seguridad de traffic-flow", porque no solo se prohíbe el acceso a la información al enemigo si no que también de donde y cuanta información se pasa a través del enlace.

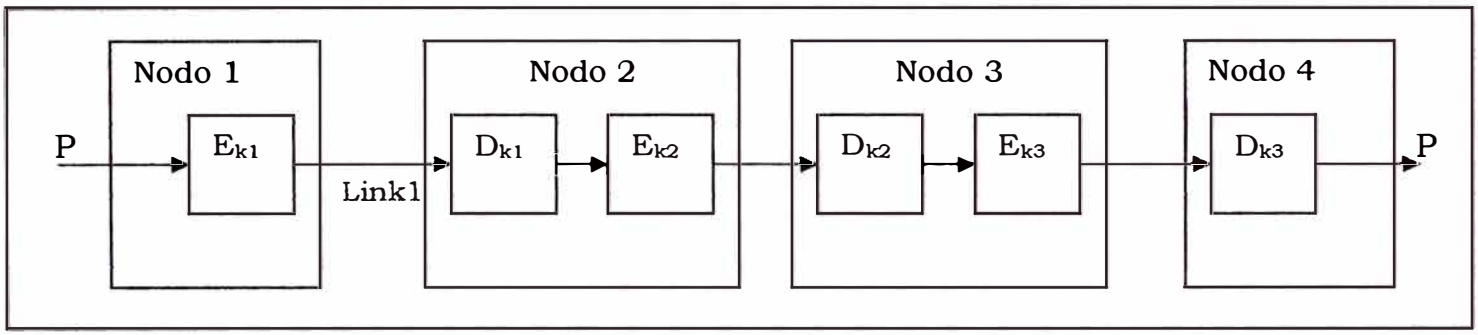


Fig. 3.6

Como se puede ver la seguridad no depende de ninguna técnica de manejo de tráfico. La administración de la clave es también simple, solo los dos puntos finales de una línea necesitan saber la clave común, pueden cambiar su clave en cualquier momento independientemente de la topología o condiciones de la red.

El gran problema con la encriptación en la capa física es que cada enlace físico debe ser encriptado, si se deja cualquier enlace sin encriptar se pone en riesgo la seguridad de

toda la red. Si la red es grande, el costo puede rápidamente volverse enorme para este tipo de encriptación.

B. Encriptación end-to-end

Este otro método consiste en poner el equipo de encriptación entre la capa de red y la capa de transporte. El dispositivo de encriptación debe entender la data de acuerdo a los protocolos inferiores y encriptar solo las unidades de dato de transporte, las cuales son luego combinadas con información de routing no encriptada y enviada a las capas mas bajas para la transmisión.

Este método evita el problema de encriptación/desencriptación en la capa física. La data permanece encriptada hasta que llegue a su destino final, ya que se trata de encriptación end-to-end. El problema principal con la encriptación end-to-end es que la información de routing para la data no esta encriptada, y un buen criptoanalista puede averiguar quien esta hablando con quien, en que momento y la cantidad de tiempo, sin siquiera tener el contenido de las conversaciones. El manejo de claves es también más difícil que el método anterior, ya que los usuarios individuales deben estar seguros de que tiene una clave común. La mayor desventaja de este tipo de encriptación es que permite el análisis de tráfico. El análisis de tráfico es el análisis de mensajes encriptados.

3.4.4 Encriptación en hardware versus encriptación en software

A. Encriptación hardware.

Hasta hace poco, todos los productos de encriptación estaban en forma de hardware especializado. Estas cajas negras de encriptación/desencriptación unían mediante una línea de comunicación y encriptaban toda la data que atravesaba la línea. Aunque la encriptación de software se vuelve más común y conocida hoy en día, el hardware es todavía el más aceptado, confiable y preferido por las aplicaciones militares e importantes corporaciones comerciales. La más importante ventaja es la rapidez, además la encriptación ocupa casi todo el procesamiento de una PC, así que es preferible que exista un chip que se encargue exclusivamente de esta tarea. La segunda razón es la seguridad, una algoritmo de encriptación corriendo en una computadora común de la red no tiene protección física, finalmente el hardware es mas fácil de instalar. Las aplicaciones de este tipo de encriptación se pueden encontrar en teléfonos, faxes, módems seguros, etc. que tiene un chip adicional que encripta la comunicación que sale de cada uno de estos aparatos.

B. Encriptación software

Se puede implementar un algoritmo de encriptación en software. Las desventajas con la rapidez, el costo, la facilidad de modificación. Las ventajas son la flexibilidad y la portatibilidad, fácil de usar y fácil de mejorar cada vez que se tengan nuevas versiones. Los algoritmos son escritos mayormente en lenguaje C. Los programas de encriptación son populares y están disponibles para casi todos los sistemas operativos. Estos están

orientados a proteger archivos individuales, el usuario generalmente tiene que encriptar y desencriptar manualmente los archivos específicos.

3.4.5 Compresión, codificación y encriptación

Cuando se utilizan los algoritmos de compresión con los algoritmos de encriptación se opera bajo las siguientes condiciones: primeramente, el criptoanálisis se basa en sacar ventaja de las redundancias en el texto plano; comprimir a un archivo antes de la encriptación reduce estas redundancias, adicionalmente, las encriptación consume tiempo, la compresión de un archivo después de la encriptación ahorra tiempo a todo el proceso.

En conclusión, lo importante es comprimir antes de encriptar, si el algoritmo de encriptación es bueno el texto cifrado no será comprensible, y parecerá a data aleatoria. Esto se observa en la Fig. 3.7.

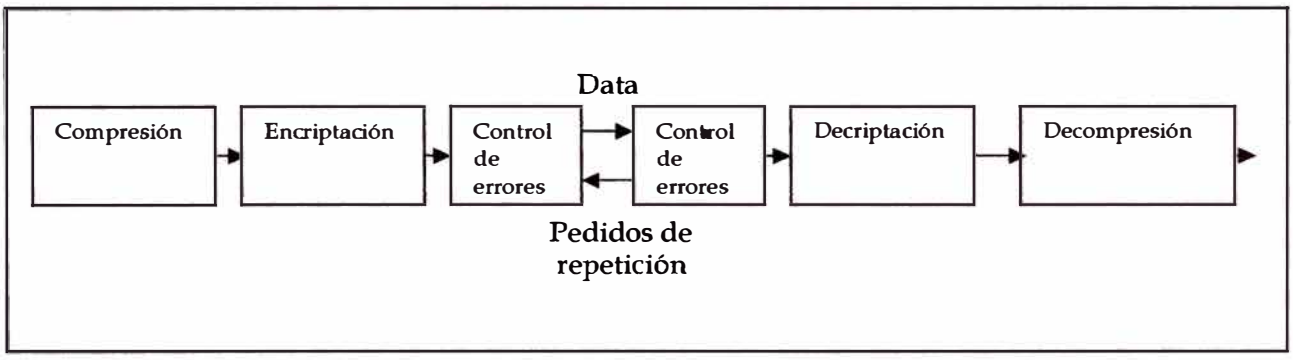


Fig. 3.7

CAPÍTULO IV

ALGORITMOS DE CRIPTOGRAFÍA

4.1 CONCEPTOS MATEMÁTICOS

4.1.1 Teoría de la información

La teoria moderna de la información fue públicada por primera vez en 1948 por Claude Shannon, a continuación se mostrará un esquema algunas de sus ideas más importantes.

a. Entropía e incertidumbre

La teoria de la información define como la cantidad de información en un mensaje la cantidad minima de bits necesarios para codificar el significado de este mensaje. La cantidad de información de un mensaje M se mide por la entropía del mensaje, es denotado como H(M). La entropía de un mensaje también mide su incertidumbre, ya que la incertidumbre es el número de bits de texto plano necesarios a recuperarse cuando el mensaje es encriptado.

b. Seguridad de un criptosistema

Para cualquier lenguaje dado, la velocidad del lenguaje es: $r = H(M)/N$

en la que N es la longitud del mensaje.

c. Seguridad de un criptosistema

Shanon definió un modelo matemático preciso de lo que significa para un criptosistema ser seguro. El objetivo de un criptoanalista o atacante es determinar la clave K, y el texto plano P, o ambos. Sin embargo, el atacante puede conformarse con alguna información probabilística de P: ya sea audio digitalizado, texto alemán, data escrita o cualquier otra cosa. En un criptoanálisis más real, el criptoanalista tiene alguna información probabilística acerca de P aun antes de que este empiece, probablemente el atacante conozca el lenguaje del texto plano, o la identidad del emisor, etc.

d. Distancia única

Para un mensaje de tamaño n, el número de claves diferentes que se descifraran en un mensaje de texto cifrado por algún texto plano entendible en el mismo lenguaje que el texto plano original (como cadenas de texto en español) se puede ver mediante la siguiente fórmula:

$$2^{H(K) - nD} - 1$$

Shannon definió como la distancia única, U, también llamado punto de unicidad como una aproximación de la cantidad de texto cifrado de modo que se cumpla que la suma de la información real (entropía) en el texto plano correspondiente mas la entropía de la clave de la encriptación iguale el numero de bits usados en el texto cifrado.

e. Confusión y Difusión

Las dos técnicas básicas para esconder las redundancias en un mensaje de texto plano son, de acuerdo con Shannon, la confusión y la difusión: La confusión esconde la relación entre el texto plano y el texto cifrado. Así se frustran los intentos de estudiar el texto cifrado buscando redundancias y modelos estadísticos. La forma más fácil de hacer esto es mediante la sustitución. La difusión disipa la redundancia del texto plano expandiéndola sobre todo el texto cifrado. Un atacante que busque las redundancias no las encontrará tan fácilmente. La forma mas practica de efectuar la difusión es mediante la transposición.

4.1.2 Teoría de complejidad

La teoría compleja proporciona una metodología para analizar la complejidad computacional de diferentes técnicas y algoritmos criptográficos. Compara los algoritmos de criptografía y las técnicas y determina su nivel de seguridad. La teoría de la información sostiene que se pueden romper todos los algoritmos criptográficos. La teoría compleja determina si estos algoritmos pueden ser rotos en un tiempo de ataque

prudencial con las herramientas de procesamiento que se tienen disponibles en la actualidad.

a. Complejidad de los algoritmos

La complejidad del algoritmo se determina por la potencia de procesamiento de computadores necesario para romper este algoritmo. La complejidad computacional de un algoritmo se mide frecuentemente con dos variables: T (para la complejidad en el tiempo) y S (complejidad en espacio, o requerimientos de memoria). T y S son comúnmente expresados como función de n, donde n es el tamaño de la entrada. Adicionalmente existen otras variables para medir la complejidad: como el número de bits aleatorios, el ancho de banda de comunicación, la cantidad de data, etc.

Generalmente la complejidad computacional de un algoritmo se expresa en la notación llamada "big O": orden de magnitud de complejidad computacional. Por ejemplo, si la complejidad de tiempo de un algoritmo es $4n^2+7n+12$, entonces la complejidad computacional es del orden de n^2 expresado O (n^2).

Generalmente, los algoritmos se clasifican en base a su complejidad tiempo o espacio. Un algoritmo es constante si su complejidad es independiente de n: *O Paso 1:*. Un algoritmo es lineal, si su complejidad es *O(n)*. También pueden ser cuadráticos, cúbico, etc. todos estos algoritmos son polinomiales. Los tipos de algoritmos que tienen complejidad de tiempo polinomiales son llamados algoritmos polinomiales en el tiempo.

Los algoritmos exponenciales cuyas complejidades son $O(t^{f(n)})$, donde t es una constante mayor que 1 y $f(n)$ es alguna función polinomial de n.

La complejidad de tiempo de un algoritmo se diferencia notablemente a medida que crece n. La siguiente tabla muestra los tiempos de procesamiento para diferentes clases de algoritmos en la que n es igual a un millón. En la tabla 4.1 se ignora a las constantes, pero se muestra también por que es práctico ignorarlas.

Tiempos de procesamiento de diferentes tipos de algoritmos			
Clase	Complejidad	# de operaciones para $n=10^6$	Tiempo a 10^6 del sistema operativo
Constantes	O(1)	1	1 useg.
Lineales	O(n)	10^6	1 seg.
Cuadraticos	$O(n^2)$	10^{12}	11,6 dias.
Cubicos	$O(n^3)$	10^{18}	32000 años.
Exponenciales	$O(2^n)$	10^{301030}	10^{301006} veces la edad del universo.

Tabla 4.1

Asumiendo que la unidad de tiempo del computador es de un microsegundo, la computadora puede procesar un algoritmo constante en un microsegundo, un algoritmo lineal en un segundo, y un cuadrático en 11,6 días, ejecutar un algoritmo exponencial es fútil por el tiempo que demora.

b. Complejidad de problemas

La teoría de la complejidad también clasifica la complejidad inherente de los problemas, no solo la complejidad de un algoritmo en particular. La teoría busca el mínimo tiempo y espacio requerido para resolver el problema mas complejo en una computadora teórica

comúnmente conocida como maquina Turing. Una maquina Turing es una maquina de estados finito con una cantidad de memoria infinita, en otras palabras la maquina Turing es un modelo real de computación.

Los problemas que pueden ser resueltos con algoritmos polinomiales en el tiempo son llamados tractable (tratables), ya que por lo general, dadas las variables lógicas de entrada pueden ser solucionados en una cantidad razonable de tiempo. Los problemas que no pueden ser resueltos en base a polinomios de tiempo son llamados intractables, porque el cálculo de su solución se vuelve rápidamente inmanejable. Los problemas intractables son aquellos que solo pueden ser resueltos con algoritmos que son superpolinomiales. Adicionalmente están los problemas undecidables, porque no se sabe con que tipo de algoritmo pueden ser resueltos, sin importar la complejidad en el tiempo.

4.1.3 Teoría de números

A continuación se describe a grandes rasgos algunos conceptos de teorias de números que se aplican en la criptografía.

a. Aritmetica modular

Es básicamente el clásico problema de módulo, ejm:

$47 = 23 \pmod{24}$

Es decir, $a \equiv b \ (mod\ n)$ si $a = b+kn$ para algun entero k. Si a no es negativo y b es un múmero entre 0 y n, se puede decir que b es el resto de a cuando se divide entre n. Algunas veces, b es llamado residuo de a, módulo n, también a es llamado congruente a b, módulo n. La aritmética modular tiene las mismas propiedades que la aritmética clásica, es conmutativa, asociativa, y distributiva.

(a+b) mod n = ((a mod n) + (b mod n)) mod n

(a-b) mod n = ((a mod n) - (b mod n)) mod n

(a*b) mod n = ((a mod n) * (b mod n)) mod n

(a* (b + c)) mod n = (((a*b) mod n) + ((a*c) mod n)) mod n

En criptografía se utiliza frecuentemente la operación "mod n". La aritmética modular es también una operación fácil para los procesadores de las computadoras.

b. Números primos

Los números primos son enteros cuyos únicos factores son el 1 y el mismo número. Hay infinitos números primos, las criptografía, especialmente la criptografía de clave pública utiliza números primos grandes (de 512 bits ó mas).

c. Mayor divisor común

Dos números son primos relativamente cuando no comparten ningún factor en común, solo 1, es decir, el mayor divisor común de dos números a y n es 1, esto se denota como:

$$gcd\,(a,n) = 1$$

Ejm. Los números 15 y 28 son primos relativos. Una forma de identificar el mayor divisor común de dos números en por el algoritmo euclidiano.

Por ejemplo el siguiente algoritmo es para dos números *x* e *y*, entrega como salida el gcd (greatest common divisor) = *m*:

```
/* la salida es el m: máximo común divisor de x e y/
int mcd (int x, int y)
{       int m ;
        si (x<0)
                x = -x
        si (y<0)
                y = -y
        si (x+y=0)
                ERROR
        m = y
        while(x>0)
                {m = x
                x = y mod x
                y = m }
        respuesta m ;}
```

d. El módulo inverso de un número

El módulo inverso de un número es mucho mas difícil de resolver que un módulo normal. En algunos casos existe la solución y otras no. Se trata de encontrar *x* para que se cumpla con:

$$1 = (a * x) \bmod n \quad \text{ó} \quad a^{-1} \equiv x \pmod{n}$$

En general $a^{-1} \equiv x \pmod{n}$ tiene una solución única si *a* y *n* son primos relativos. Si *a* y *n* no son primos relativos, entonces $a^{-1} \equiv x \pmod{n}$ no tiene solución. Si *n* es un número primo, entonces cada número desde *1* hasta *n-1* es relativamente primo a *n* y tiene exactamente un módulo inverso *n* en este rango. Para averiguar el módulo inverso de un número *módulo n*, también se puede utilizar el algoritmo euclidiano extendido.

4.1.4 Factorización y generación de números primos

Factorizar significa encontrar los factores primos de un número. La operación en sí de factorizacion es simple pero consume tiempo cuando se trata de grandes números. Actualmente el mejor algoritmo de factorizacion es el de Number Field Sieve (NFS) el cual es el algoritmo mas rapido para números de mas de 110 dígitos. Cuando fue propuesto inicialmente no era considerado práctico, pero en los ultimos años se han hecho varios upgrades a este algoritmo.

En cuanto a generación de números primos, los algoritmos de clave pública necesitan multiples números primos, en general el algoritmo para crear primos es crear un número aleatoriamente y luego ver si es primo o no, para esto hay varios métodos entre los más eficientes esta el de Rabin-miller, consisten básicamente en :

- Escoger un numero aleatoriamente a evaluar p. Calcular b, donde b es el numero de veces que 2 divide p-1, luego calcular m, de modo que $p = 1+2^b m$
- Escoger un número aleatorio a, de modo que a sea menor que p.
- Establecer $j = 0$ y $z = a^m \bmod p$.
- Si $z = 1$, o si $z = p-1$, entonces p pasa el test y es primo.
- Si $j > 0$ y $z = 1$, p no es primo.
- Hacer $j = j+1$. Si $j < b$ y $z \neq p-1$, hacer $z = z^2 \bmod p$ y retroceder al paso (4). Si $z = p-1$, luego p pasa el test y es primo.
- Si $j = b$ y $z \neq p-1$, entonces p no es primo.

4.2 ALGORITMO DE ENCRIPTACIÓN ESTANDAR DE DATOS

4.2.1 Introducción

El nombre real es Data Encryption Standard DES, conocido como el Data Encryption Algorithm (DEA) por ANSI y como DEA-1 por el ISO. Este algoritmo se ha mantenido como standard por 20 años, a pesar de su antigüedad, el algoritmo nunca ha sido roto (a excepción de una implementación que no es práctica).

El desarrollo del algoritmo se produjo como resultado de un concurso público convocado por el Nacional Bureau of Standard (hoy el Nacional Institute of Standards and Technology, NIST) donde se elegiría un estándar de encriptación para la seguridad de documentos oficiales se especificaron los siguientes criterios como requisitos al algoritmo:

- El algoritmo debería brindar un alto nivel de seguridad.
- El algoritmo debería ser completamente especificado y fácil de entender.
- La seguridad del algoritmo debería residir en la clave; la seguridad no debería depender del secreto del algoritmo.
- El algoritmo debe ser disponible a todos los usuarios.
- El algoritmo debe ser adaptable para el uso en diversas aplicaciones.
- El algoritmo debe ser económicamente implementable en dispositivos electrónico.
- El algoritmo debería ser eficiente para usar.
- El algoritmo debe ser capaz de ser validado-
- El algoritmo debe ser exportable.

Después de 4 años de concurso, fue ganado por el grupo de investigadores que había generado el algoritmo LUCIFER en 1970. Aunque el algoritmo DES era algo complicado. Usaba únicamente operaciones lógicas simples en pequeños grupos de bits y podía ser implementado eficientemente en hardware.

Las validaciones de las implementaciones en hardware y software basadas en DES se llevaban a cabo por NIST que asegura que la implementación sería de acuerdo al estándar, esto fue hasta 1994 cuando se prohibieron las implementaciones en software.

4.2.2 Descripción del DES

DES es un bloque cifrado, el algoritmo encripta la data en bloques de 64 bits. La entrada al algoritmo es un bloque de 64 bits de un texto plano y produce una salida un bloque de 64 bits de texto cifrado. Des es también un algoritmo simétrico: el mismo algoritmo y clave son utilizadas para encriptación y desencriptacion, a excepción de diferencias en el intercambio de claves.

El tamaño de la clave es de 56 bits, la clave normalmente consiste en un número de 64 bits, pero los otros 8 son bits de paridad (los bits de paridad son los bits menos significativo en los bytes de la clave). La clave puede ser cualquier número de 56 bits y puede ser cambiado en cualquier momento. En el nivel más simple, el algoritmo no es más que una combinación de dos técnicas básicas de encriptación: confusión y difusión. El bloque de construcción fundamental de DES es una simple combinación de estas técnicas (una sustitución seguida por una permutación) sobre un texto, basada en la clave. Esto es conocido como un round. DES tiene 16 rounds; y aplica las misma combinación de técnicas en el texto plano 16 veces. Como se muestra en la Fig. 4.1.

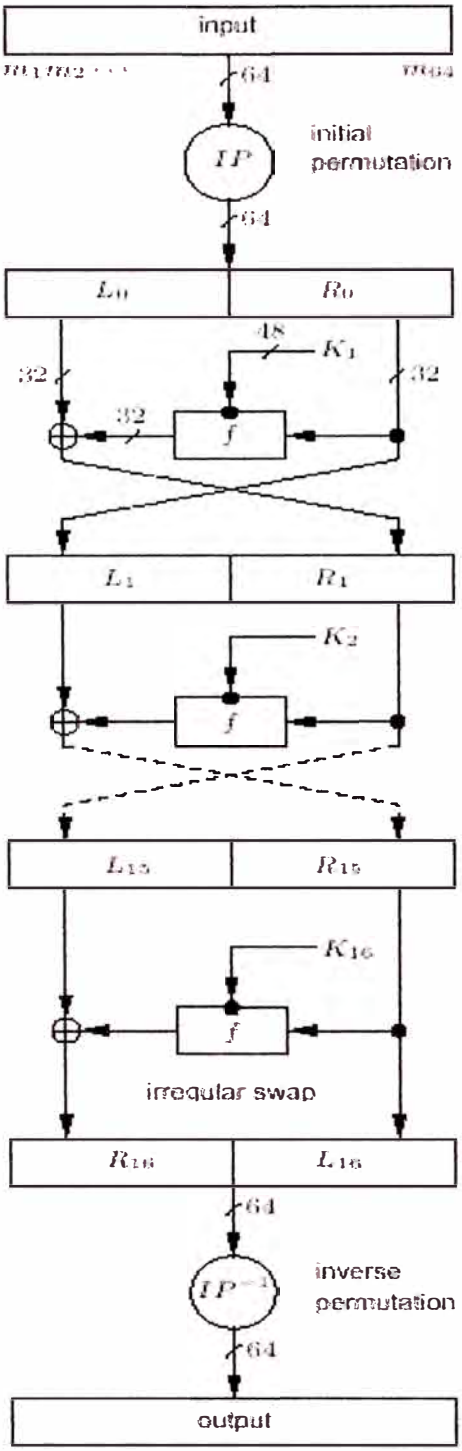

input
64
initial
permutation
IP
64
L_0
R_0
48
K_1
32
32
32
f
L_1
R_1
K_2
f
L_{15}
R_{15}
K_{16}
f
irregular swap
R_{16}
L_{16}
64
inverse
permutation
IP^{-1}
64
output

Fig. 4.1

El algoritmo usa solo operaciones estándares aritméticas y lógicas en números de 64 bits a mas, razón por la que fue fácilmente implementado en la tecnología hardware de la década del 70. La naturaleza repetitiva del algoritmo hace que sea ideal para un chip de propósito especial. Las implementaciones iniciales en software eran algo torpes, pero últimamente han mejorado notablemente. DES opera sobre bloques de 64 bits de texto plano. Cada trozo de 64 bits de los datos se desordena según un modelo fijo llamado permutación inicial: IP (Inicial Permutation). Luego se divide en dos mitades de 32 bits cada uno, este bloque de 32 bits pasa por 16 iteraciones. Este algoritmo se repite 16 veces, llamada Función f, en la que la data es combinada con la clave, después de la vuelta 16, las mitades, de 32 bits cada una, se juntan, y el algoritmo termina con una permutación final (que es la inversa de la permutación inicial IP).

a. Descripción de un round del algoritmo

En cada round el algoritmo ejecuta cuatro operaciones, primeramente selecciona 48 bits de los 56 de la clave, los 32 bits de la derecha del texto plano permutado se expande a 48 bits por una permutación de expansión, se combina por medio de un XOR con los 48 bits de la clave que han sido previamente permutados, este resultado se envía a través de 8 S-boxes produciendo 32 nuevos bits, que se permutan otra ves. Estas cuatro operaciones forman en conjunto la Función f. La salida de la función f se combina por un XOR con la mitad de la izquierda.

El resultado de estas operaciones se convierte en la nueva parte derecha y la antigua parte derecha se convierte en la nueva parte izquierda para empezar un nuevo round. De esta manera se repiten 16 veces, formando los 16 rounds del DES. El diagrama de un round se observa en la Fig. 4.2. De la Fig. 4.2 se puede ver que si en la iteración enesima, Li y Ri son las mitades izquierdas y derechas de Bi, y Ki es la clave de 48 bits para la iteración i, y f es la función que hace la sustitución y la permutacion y hace un XOR con la clave, luego un round se puede expresar de la siguiente forma:

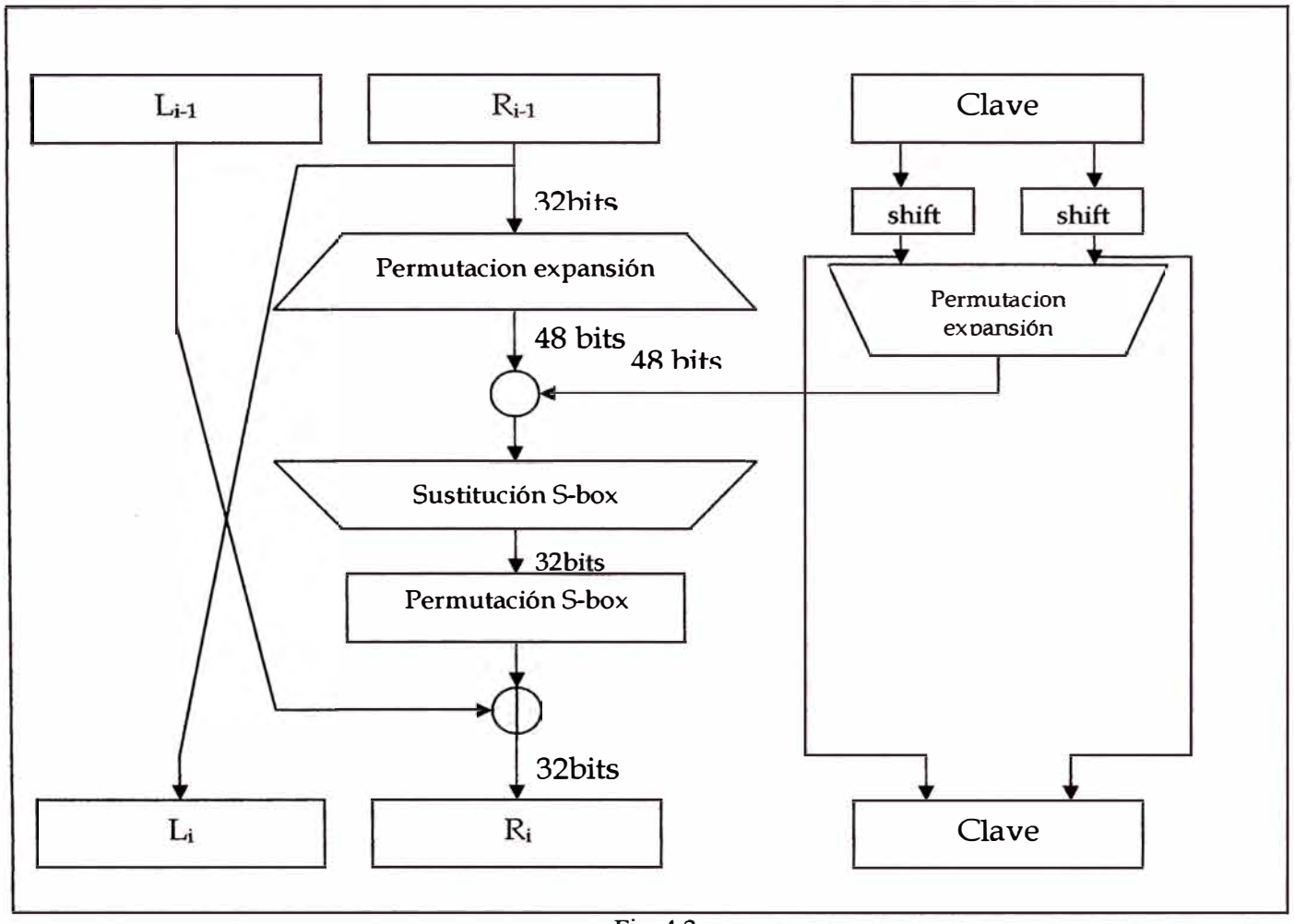


Fig. 4.2

$$L_i = R_{i-1} \quad \rightarrow \quad R_i = L_{i-1}\, f(R_{i-1}, K_i)$$

b. Permutación inicial

La permutación inicial ocurre antes de la primera iteración o round; se traspone el bloque de entrada como se describe en la tabla 4.2, esta tabla, como todas las otras tablas que utiliza DES, debe ser leída de izquierda a derecha, de arriba hacia abajo. Por ejemplo la permutación inicial mueve el bit 58 del texto plano a la posición 1, el bit 50 a la posición 2 , el bit 42 a la posición 3, etc. la permutación inicial y si correspondiente inverso al final de todo el algoritmo no compromete la seguridad de DES.

Permutación Inicial															
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	59	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Tabla 4.2

c. Transformación de la clave

Inicialmente la clave de 64 bits de DES se reduce a una clave de 56 bits ignorando cada bit 8 por que se trata del bit de paridad de cada byte. Esto se describe en la tabla 4.3:

Permutación de la clave													
57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Tabla 4.3

Luego se genera una sub clave diferente de 48 bits, para cada round de los 16 que conforman todo el algoritmo DES. Estas subclaves Ki, se determinan de la siguiente manera, primero los 56 bits se dividen en dos mitades de 28 bits cada una, luego las mitades son circularmente transportadas (también llamada elevación) hacia la derecha por uno o dos bits, dependiendo del número de round. Esta elevación se muestra en la tabla 4.4.

Número de bits elevados en la clave por round																
Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Número	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabla 4.4

Después de ser elevados los bits de las claves, se selecciona 48 de los 56 bits. Esta operación se llama permutación compresión porque permuta el orden de los 56 bits y luego selecciona un subconjunto de ellos. Finalmente la salida de toda esta operación son 48 bits. En la tabla 4.5 se define la permutación compresión. Por ejemplo, el bit en la posición 33 se mueve a la posición 35 de la salida y el bit en la posición 18 de la clave elevada se ignora. Debido a la elevación, se usa siempre un subconjunto diferente de la clave en la clave correspondiente a cada vuelta. Cada bit se usa aproximadamente en 14 de las 16 sub claves, sin embargo no todos los bits son utilizados la misma cantidad de veces.

Permutación compresión											
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	43	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Tabla 4.5

d. La permutación expansión

Esta operación expande la mitad derecha de la data Ri, de 32 bits a 48 bits, esta operación es llamada permutación expansión porque cambia el orden de los bits y también repite ciertos bits. Esta operación tiene dos propósitos: hace que la mitad derecha tenga el mismo tamaño que la clave para poder efectuar la operación XOR y entrega un resultado más grande que puede ser comprimido durante la operación de sustitución. Sin embargo también tiene un propósito criptográfico, porque al permitir que un bit afecte dos sustituciones, la dependencia de los bits de salida en los bits de la entrada se hace mayor.

En la figura 4.3 se define la permutación expansión (a veces llamada E-box) para cada bloque de entrada de 4 bits, el primero y el cuarto bit representan cada uno dos bits del bloque de la salida, mientras que el segundo y tercer bit representan cada uno un bit de la salida.

En la tabla 4.6 se muestra que posiciones corresponden a cada posición de entrada, por ejemplo en la posición 3 de bit en el bloque de entrada se mueve hacia la posición 4 del bloque de salida, y la posición 21 del bloque de entrada se mueve a la posición 20 y 32 del bloque de salida.

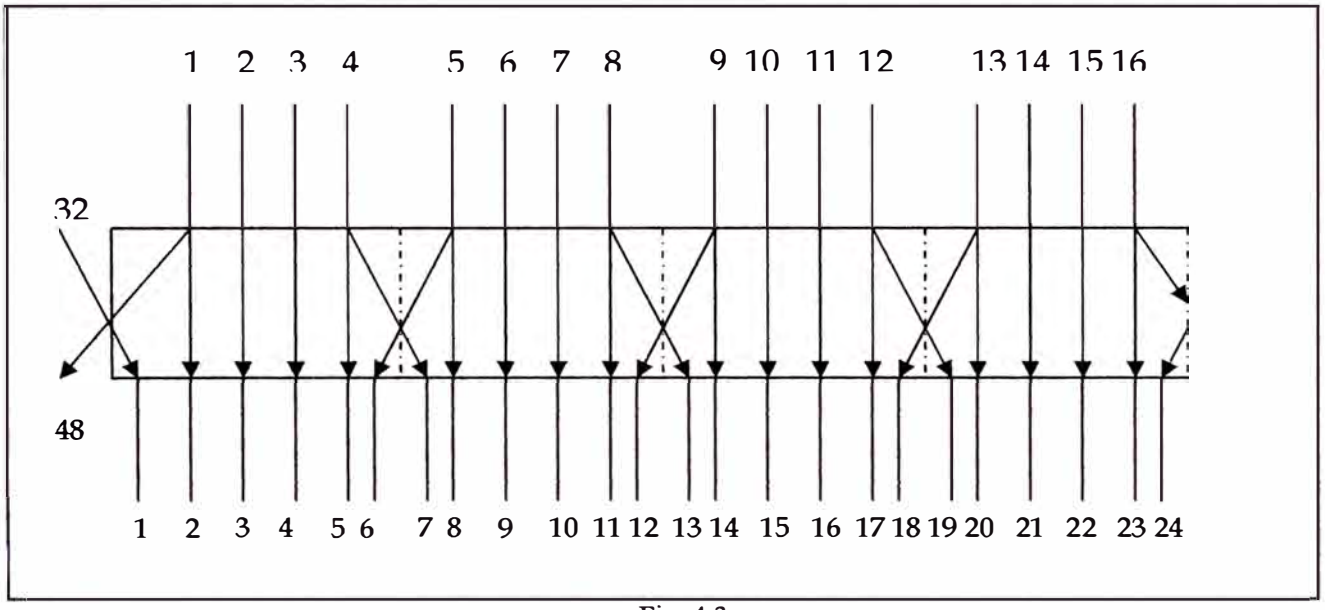


Fig. 4.3

permutación expansion											
32	1	2	3	4	5	4	5	6	7	8	9
8	6	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	7

Tabla 4.6

e. La caja S de sustitución: S-Box

Después de que la clave comprimida pasa por un XOR con el bloque expandido, el resultado de 48 bits pasa por una operación de sustitución, las sustituciones son

efectuadas por ocho cajas de sustitución o S-boxes. Cada S-box tiene 6 bits de entrada y 4 bits de salida, y hay 8 S-box diferentes. Los 48 bits son divididos bloques de 6 bits cada uno. Cada bloque separado opera en un S-box diferente. Como se muestra en la figura 4.4:

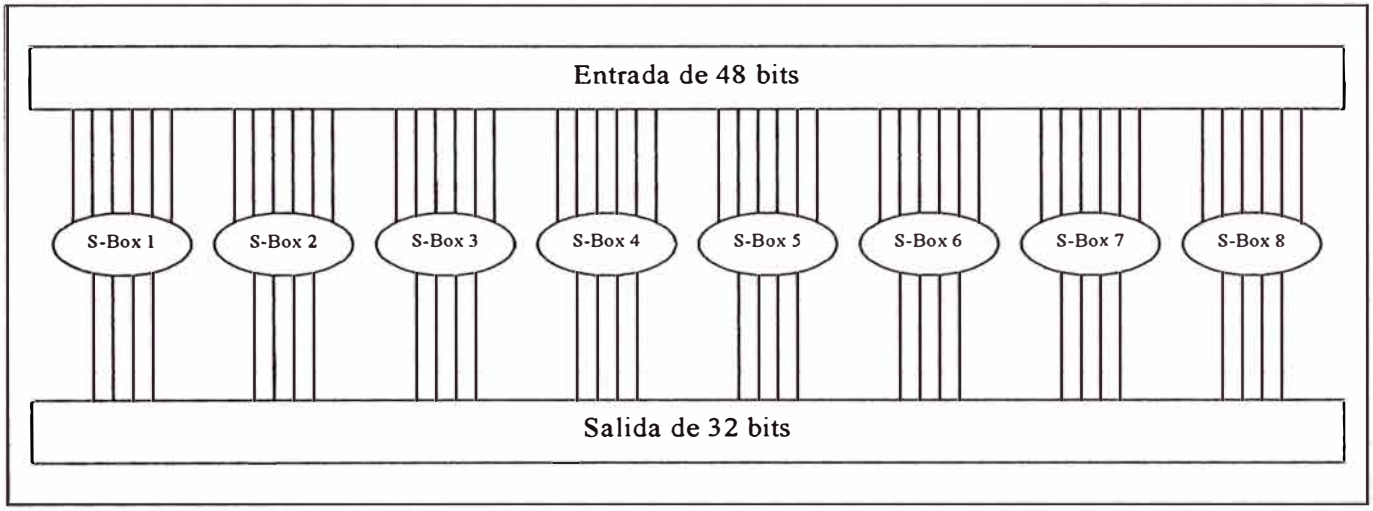


Fig. 4.4

Cada caja S-box es una tabla de 4 filas y 16 columnas. Cada entrada de la caja es un número de 4 bits. Los 6 bits de entrada del S-box especifican bajo que numero de fila y columna se debe buscar en la salida, a continuación el diagrama correspondiente al primer S-box, cabe recalcar que cada S-box es diferente:

Diagrama del S-box 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabla 4.7

Los bits de entrada especifican una entrada en el S-box en una manera muy particular, considerando las entradas de S-box: b_1 b_2, b_3, b_4, b_5 y b_6. Los bits b_1 y b_6 se combinan

para formar un número de 2 bits, desde el 0 al 3, que corresponden a una fila de la tabla. Los bits del medio desde el b_2 al b_5 se combinan para formar un número de 4 bits, desde el 0 al 15 que corresponde a la columna en la tabla 4.7.

f. La caja B de permutación: P-Box

Los 32 bits de salida de la caja de sustitución S-box se permuta de acuerdo a la caja P-box. Esta permutación mapea cada bit de entrada en una posición de salida; no se usa ningún bit dos ves ni se ignora ningún bit, a esto se le llama permutación directa o solo permutación, la tabla 4.8 muestra la posición a la que se mueve cada bit. Por ejemplo el bit 21 se mueve al 4 mientras el 4 se mueve al bit 31.

permutación P-Box															
16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Tabla 4.8

Finalmente la permutación P-Box se opera mediante XOR con la mitad de la parte izquierda del bloque de 64 bits. Luego las mitades izquierda y derecha se intercambian y empieza otro round.

g. Desencriptando DES

Después de todas las sustituciones, permutaciones, XORs, y rounds se puede creer que la desencriptacion del algoritmo es completamente diferente y difícil como el algoritmo de encriptación. Por el contrario, estas operaciones fueron seleccionadas

deliberadamente para cumplir con una propiedad muy útil: el mismo algoritmo trabaja para la encriptación y desencriptacion, Con DES es posible usar la misma función para encriptar y desencriptar un bloque. La única diferencia es que las claves deben ser utilizadas en orden inverso. Es decir, si las claves de encriptación fueron K_1, K_2, K_3..., K_{16} entonces las claves de desencriptacion serán K_{16}, K_{15},...., K_2, K_1.

h. Modos de DES

DES tiene cuatro modos de operación: ECB, CBC, OFB y CFB, los modos de operación ya se describieron en un capítulo anterior, los estándares ANSI especifican los modos ECB y CBC para la encriptación y los CBC y CFB para autenticación.

4.2.3 Criptoanálisis diferencial y de relación de claves

a. Criptoanálisis diferencial

El criptoanálisis diferencial es un nuevo método de criptoanálisis, utilizando este método se encontró un ataque del tipo de texto escogido contra DES que fue mas eficiente que un brute force. El criptoanálisis diferencial se centra específicamente en pares de texto cifrado: que son pares de texto cifrado cuyos textos planos tienen diferencias particulares. Analiza la evolución de estas diferencias a medida que los textos planos se propagan a través de los rounds de DES cuando se encriptan con la misma clave. Simplemente, se escogen pares de texto plano con una diferencia predeterminada. Los dos textos planos pueden ser ubicados aleatoriamente, siempre y cuando satisfagan condiciones de diferencias particulares. Luego, utilizando las diferencias en los textos

cifrados resultantes, se utilizan diferentes probabilidades para diferentes claves. A medida que se analicen más y más textos cifrados, se ubica una clave como la más probable.

b. Criptoanálisis de relación de claves

Como sabemos del capítulo de "descripción de des" se sabe que la clave DES se cambia de dos lugares de bits dependiendo en cada vuelta, excepto para las vueltas 1, 2, 9 y 16 que se cambia un bit. El criptoanálisis de clave relacionada es similar al criptoanálisis diferencial, pero este último examina la diferencia entre claves. Este ataque es diferente de cualquier ataque discutido anteriormente: el criptoanalista escoge una relación entre un par de claves, pero no sabe cuales son las claves. La data se encripta con ambas claves. En la versión de ataque con texto plano conocido, el criptoanalista conoce el texto plano y el texto cifrado de la data encriptada con ambas claves, en el ataque de texto plano escogido, el criptoanalista llega aun a escoger el texto encriptado con las dos claves.

4.2.4 Variantes de DES

a. DES Múltiple

Algunas implementación de de DES utilizan un triple DES. Como DES no es un grupo, el texto cifrado resultante es mucho mas difícil de descrifrar cuando se ejecuta una búsqueda exhaustiva, es necesario 2^{112} intentos en vez de los 2^{56} intentos necesarios para violar el algoritmo DES original.

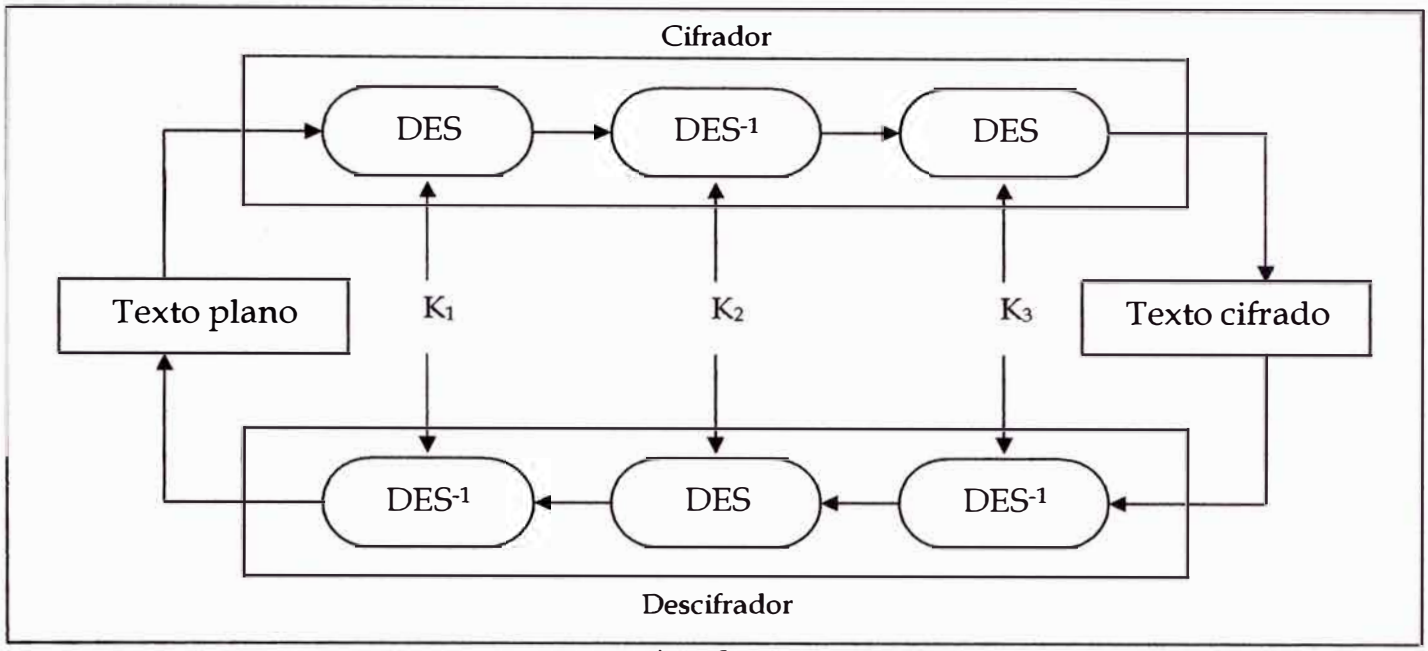


Fig. 4.5

b. DES con diferentes sub claves independientes.

Esta es otra variación donde se usa una sub clave diferente para cada round, en vez de generarlas desde una sola clave de 56 bits, ya que la clave de 48 bits se utiliza en cada una de las 16 vueltas, esto quiere decir que el tamaño de la clave para esta variante es de 768 bits. Esta variante incrementaría drásticamente la dificultad de un ataque de brute-force contra el algoritmo. Este ataque tendría una complejidad de 2^{768}.

4.3 FUNCIONES DE UN SOLO SENTIDO

4.3.1 Introducción

Una función de one-way hash, H (M), opera sobre un mensaje de tamaño arbitrario, M. El resultado es un valor hash de tamaño determinado previamente, h.

h = H (M), donde h es de tamaño m

Hay muchas funciones que pueden tener una entrada de tamaño arbitrario y entregar una salida de tamaño predeterminada, pero las funciones de one-way hash tienen características adicionales que las hacen que sean de one-way: Dado M, es fácil de procesar h. Dado h, es difícil de procesar M de modo que H (M) =h. Dado M, es difícil de encontrar otro mensaje M', que cumpla con H (M)=H (M'). El objetivo central de este tipo de funciones es garantizar una "huella digital" única de M. En algunas aplicaciones, la función de one-way no es suficiente, se necesita requisitos adicionales conocidos como "resistencia a colisiones".

a. Tamaño de las funciones de one-way hash

Las funciones hash de 64 bits son demasiado pequeños para hacer frente a un ataque tipo "aniversario". La mayoría de funciones de one-way hash producen 128 bits. Esto fuerza a cualquiera que intenta usar el ataque aniversario a intentar unos 2^{64} documentos para intentar que se encuentren dos con el mismo resultado de hash, esto no es suficientemente confiable en términos de seguridad.

b. Descripción de las funciones one-way hash.

Las funciones de one-way hash se construyen en base a la idea de una función de compresión. Esta función de one-way entrega un valor hash de tamaño n y tiene una

entrada de tamaño (mayor que n) m. Las entradas a la función de compresión son un bloque del mensaje y la salida de los bloques de texto anteriores. La salida es un arreglo (hash) de todos los bloques hasta el último punto. Este es el arreglo del bloque M_i:

$$h_i = f(M_i, h_{i-1})$$

Este valor hash, acompañado de todos los bloques de mensajes, se transforma en la próxima entrada de la función de compresión. De modo que todo el mensaje es el hash (o arreglo) del último bloque. La pre-imagen debería contener algún tipo de representación binaria del tamaño de todo el mensaje. Esta técnica cubre un problema potencial de seguridad que es de diversos mensajes con diferentes tamaños podrían entregar el mismo hash de salida.

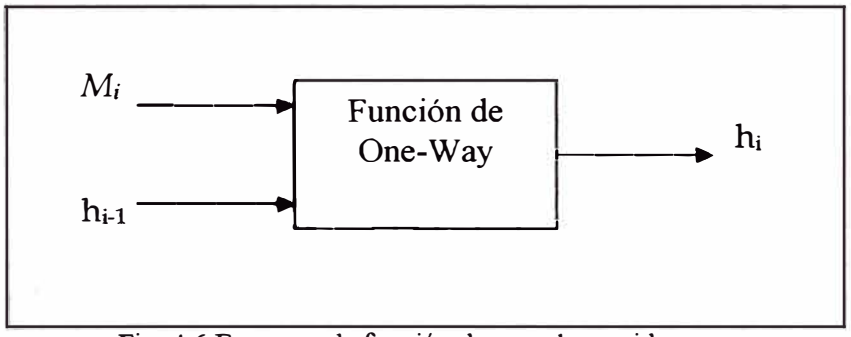


Fig. 4.6 Esquema de función de un solo sentido

4.3.2 Funciona SNEFRU

Esta función entrega como resultado arreglos de 128 bits o de 256 bits para cualquier tamaño que sea el mensaje de entrada. Primero el mensaje se divide en particiones, cada uno de (512-m) bits, donde la variable m es el tamaño del valor de hash. Si la salida es un valor hash de 128 bits, entonces las particiones son cada uno de 384 bits. Si la salida

es un hash de 256 bits, luego las particiones son cada uno de 256 bits. La esencia del algoritmo es la función H, que puede hacer un hash sobre un bloque de 512 bits y transformarlo en un valor de m bits. Los primeros m bits de la salida de H son el hash del bloque; el resto de bits de la salida se descarta. El siguiente bloque se agrega al hash del bloque previo y todo esto pasa nuevamente por la función H para formar un nuevo hash. Después del último bloque, los primeros m bits se agregan a la representación binaria del tamaño del mensaje y pasan nuevamente por la función H para producir el último hash.

La función H se basa en E, que es una función reversible de bloques de texto cifrado que opera sobre bloques de 512 bits. H es el resultado de la operación de XOR entre los últimos m bits de la salida de E y los primeros m bits de la entrada de E. La seguridad de Snefru reside en la función E, que hace que la data se vuelva aleatoria en varios pasos. Cada paso se compone de 64 vueltas de en la que se hacen los pasos para volverla aleatoria. En cada vuelta o round se usaba un byte diferente de la data como entrada a la caja S, y la salida de la caja S se opera en un XOR con las dos palabras contiguas del mensaje.

4.3.3 Función N-Hash

N-hash utiliza bloques de mensajes de 128 bits, ejecuta una transformación aleatoria complicada y produce un hash de 128 bits. El hash de cada bloque de 128 bits es el resultado de una función sobre el bloque mismo y el hash del bloque previo:

$H_0 = I$, donde I es un valor inicial aleatorio.

$H_i = g(M_i, H_{i-1}) \Theta M_i \Theta H_{i-1}$

El hash del mensaje completo es el hash del último bloque del mensaje. El valor inicial, I, puede ser cualquier valor determinado por el usuario. La función g es complicada, se intercambian las mitades (de 64 bits cada una) de los 128 bits del hash bloque anterior del mensaje H_{i-1}, luego esto se opera en un XOR con un mensaje de puros 0s o 1s de 128 bits y luego se opera otra vez con un XOR con el mensaje actual Mi. Este valor pasa luego por N etapas de procesos. La otra entrada de la etapa de procesamiento es el valor previo de hash operado con un XOR con uno de los ocho valores binarios constantes.

4.3.4 Función MD4

MD4 se refiere a Message Digest, el algoritmo produce un hash de 128 bits, para cualquiera que sea el tamaño del mensaje de entrada. El algoritmo fue diseñado de modo que cumpla lo siguiente:

Seguridad. Que sea, bajo el punto de vista de procesamiento, improbable encontrar dos mensajes que produzcan el mismo valor de hash, es decir el mejor y único ataque para este algoritmo seria el de brute force.

Seguridad Directa. Que la seguridad de MD4 no tenga su base en presunciones, por ejemplo la dificultad de factorización.

Velocidad. MD4 es apropiado para implementaciones de software de grandes velocidades, se basa en un simple conjunto de manipulaciones de bits sobre 32bits.

Después de que el algoritmo fuera conocido, se descifró satisfactoriamente dos de sus tres vueltas o rounds. Adicionalmente, otro criptoanalista, Ralph Merkle, criptoanalizó las primeras dos vueltas. Aun cuando estos ataques no podían extenderse a todo el algoritmo, los diseñadores de este algoritmo crearon uno más seguro MD5.

4.3.5 Función MD5

Es una versión mejorada del MD4, pero a su vez mas compleja, es similar en diseño y también produce un hash de 128 bits.

a. Descripción de MD5

Después del procesamiento inicial. MD5 procesa el texto de entrada en bloques de 512 bits, divididos en 16 sub-bloques de 32 bits. La salida del algoritmo es un grupo de cuatro bloques de 32 bits, que se concatenan para formar una sola salida de 128 bits.

Primero, el mensaje se acomoda de modo que sea 64 bits menor a un múltiplo de 512. Esto se hace agregando un solo bit de valor 1 al final del mensaje, seguido por tantos

ceros como sean requeridos. Luego, se agrega al resultado una representación de 64 bits. Estos dos pasos sirven para hacer que el tamaño del mensaje sea un múltiplo exacto de 512, mientras se asegura que dos mensajes diferentes no coincidan en el mismo resultado. Se inicializan cuatro variables de 32 bits:

A = 0x01234567

B = 0x89abcdef

C = 0xfedcba98

D = 0x76543210

Estos son llamados variables de concatenación. Ahora, empieza el loop principal del algoritmo, este loop continua tantas veces como bloques de 512 bits hay en el mensaje. Las cuatro variables se copian a diferentes variables. Se copian las cuatro variables a diferentes variables: a toma el valor de A, b el de B, c el de C y d el de D.

El loop principal tiene cuatro rounds, todos muy similares. Cada round utiliza una operación diferente 16 veces, cada operación consta de una función no lineal de tres variables de los cuatro: A, B, C y D. Luego agrega el resultado a la cuarta variable, un sub-bloque de texto y una constante. Luego el resultado rota hacia la derecha un número variable de bits y adiciona el resultado a una de las variables, a, b, c o d. Finalmente el resultado reemplaza uno de a, b, c o d. Como en las siguientes figuras:

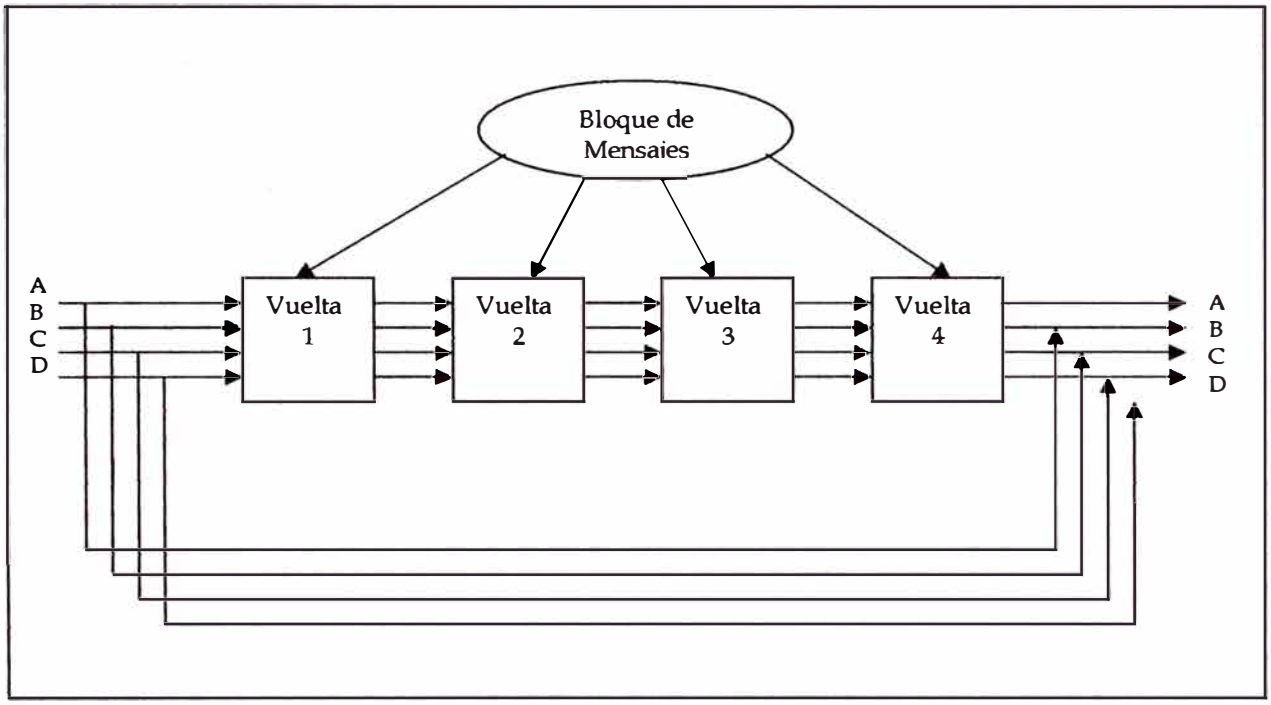


Fig. 4.7

Hay cuatro funciones no lineales, cada una se utiliza en una operación (uno diferente para cada round)

$F(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$

$G(X, Y, Z) = (X \wedge Y) \vee (Y \wedge (\neg Z))$

$H(X, Y, Z) = X \Theta Y \Theta Z$

$I(X, Y, Z) = Y \Theta (X \vee (\neg Z))$

Donde:

Θ = XOR,

$\wedge$ = AND,

$\vee$ = OR, y

$\neg$ = NOT.

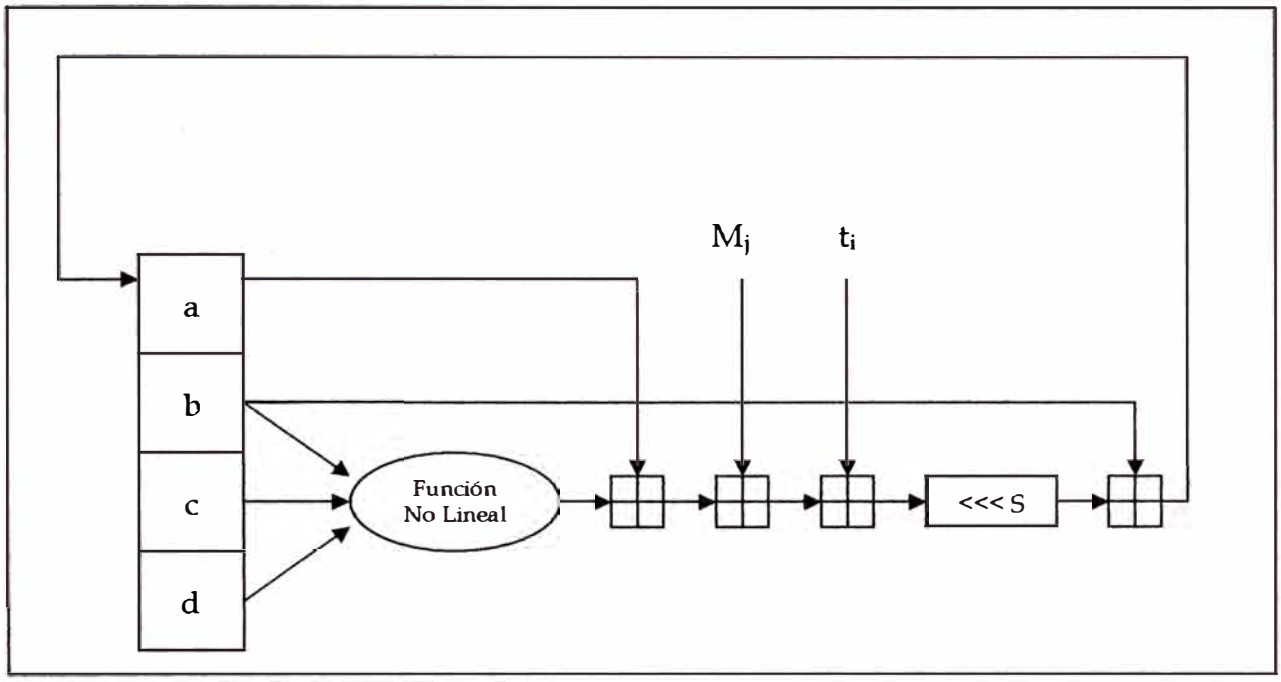


Fig. 4.8

Estas funciones se designan de modo que si los bits correspondientes de X, Y y Z sean independientes, entonces cada bit del resultado será también independiente. La función F tiene la siguiente condicional de bit: si X luego Y también Z. La función H es el operador del bit de paridad.

4.3.6 Función MD2

MD2 es otra función de hash de one-way de 128 bits, al igual que MD5, se utiliza en protocolos PEM (privacy enhanced mail). La seguridad de MD2 depende de una permutación aleatoria de bytes. Esta permutación esta predefinida y depende de los dígitos de л. La permutación es S_0, S_1, S_2, S_3,..., S_{255}. Para generar un hash del mensaje M, se siguen los siguientes pasos:

Paso 1: Rellenar el mensaje con i bytes de valor i de modo que la longitud del mensaje resultante es un múltiplo de 16 bytes.

Paso 2: Agregar un checksum de 16 bytes al mensaje.

Paso 3: Inicializar un bloque de 48 bytes: X_0, X_1, X_2,...., X_{47}. Los primeros 16 bytes de X se hacen igual a 0, los segundos 16 bytes de X serán los primeros 16 bytes del mensaje, los terceros 16 bytes de X son el resultado de un XOR entre los primeros 16 bytes de X con los 16 bytes de X.

Paso 4: Este es el algoritmo de la función de compresión:

t = 0

Para j = 0 hasta 17

Para k = 0 hasta 47

$t = X_K$ XOR S_t

$X_K = t$

t = (t+j) mod 256

Paso 5: Luego se hacen que los segundos 16 bytes de X sean los segundos 16 bytes del mensaje, luego los terceros 16 bytes de X son el resultado de un XOR entre los primeros 16 bytes de X y los segundos 16 bytes de X. Se ejecuta otra vez el paso 4. Se repiten los pasos 5 y 4 cada 16 bytes del mensaje.

Paso 6: La salida son los primeros 16 bytes de X.

Aunque no se han encontrado huecos ni debilidades en el algoritmos MD2, la principal desventaja es que es un algoritmo lento en comparación a otras funciones hash.

4.3.7 Algoritmo hash seguro (SHA)

Este algoritmo produce una salida de 160 bits, a cualquiera que sea el tamaño de mensaje de entrada. La descripción de la operación del SHA es la siguiente: primero el mensaje se acomodado con la condición de que su tamaño sea múltiplo de 512 bits. Luego se rellena de la misma manera que MD5: primero un 1 y luego tantos ceros como sea necesario para hacerla 64 bits menos de un múltiplo de 512, y finalmente una representación de 64 bits del tamaño del mensaje antes de ser modificado.

Las cinco variables de 32 bits (MD5 tiene cuatro variables) se inicializan de la siguiente manera:

A = 0x67452301 B = 0xefcdab89 C = 0x98badcfe

D = 0x10325476 E = 0xc3d2elf0

Luego empieza el loop principal del algoritmo. Se procesa el mensaje de 512 bits en 512 bits hasta que termine el mensaje. Primero las primeras cinco variables se copian en diferentes variables: A se copia en a, B en b, y así hasta E en e. El loop principal tiene cuatro vueltas o rounds de 20 operaciones cada una. Cada operación ejecuta una función no lineal sobre tres de las cinco variables a, b, c, d o e y luego ejecuta una rotación y adicionamiento similar a MD5.

Las funciones no lineales del SHA son:

$f_t(X, Y, Z) = (X \Lambda Y)\ V\ ((\neg X)\ \Lambda Z)$, para t = 0 hasta 19.

$f_t(X, Y, Z) = X \Theta Y \Theta Z$, para t = 20 hasta 39.

$f_t(X, Y, Z) = (X \Lambda Y) V (X \Lambda Z) V (Y \Lambda Z)$, para t = 40 hasta 59.

$f_t(X, Y, Z) = X \Theta Y \Theta Z$, para t = 60 hasta 79.

Se utilizan cuatro constantes en el algoritmo:

$K_t = 0x5a827999$, para t = 0, hasta 19. $K_t = 0x6ed9eba1$, para t = 20, hasta 39.

$K_t = 0x8f1bbcdc$, para t = 40, hasta 59. $K_t = 0xca62c1d6$, para t = 60, hasta 79.

El bloque de mensajes se transforma de 16 palabras de 32 bits (M_0 a M_{15}) a 80 palabras de 32 bits (W0 a W79) utilizando el mismo algoritmo:

$W_t = M_t$, para t = 0 hasta 15

$W_t = (W_{t-3} \Theta W_{t-8} \Theta W_{t-14} \Theta W_{t-16}) <<< 1$, para t = 16 hasta 79.

Si t es el número de operación (desde 0hasta 79), W_t representa el t-avo sub-bloque del mensaje expandido, y "<<< s" representa un cambio de lugar izquierdo circular de los s bits, luego el loop principal tendrá el siguiente algoritmo:

Para t = 0 hasta 79.

$TEMP = (a <<< 5) + f_t (b, c, d) + e + W_t + K_t$

e=d → d=c

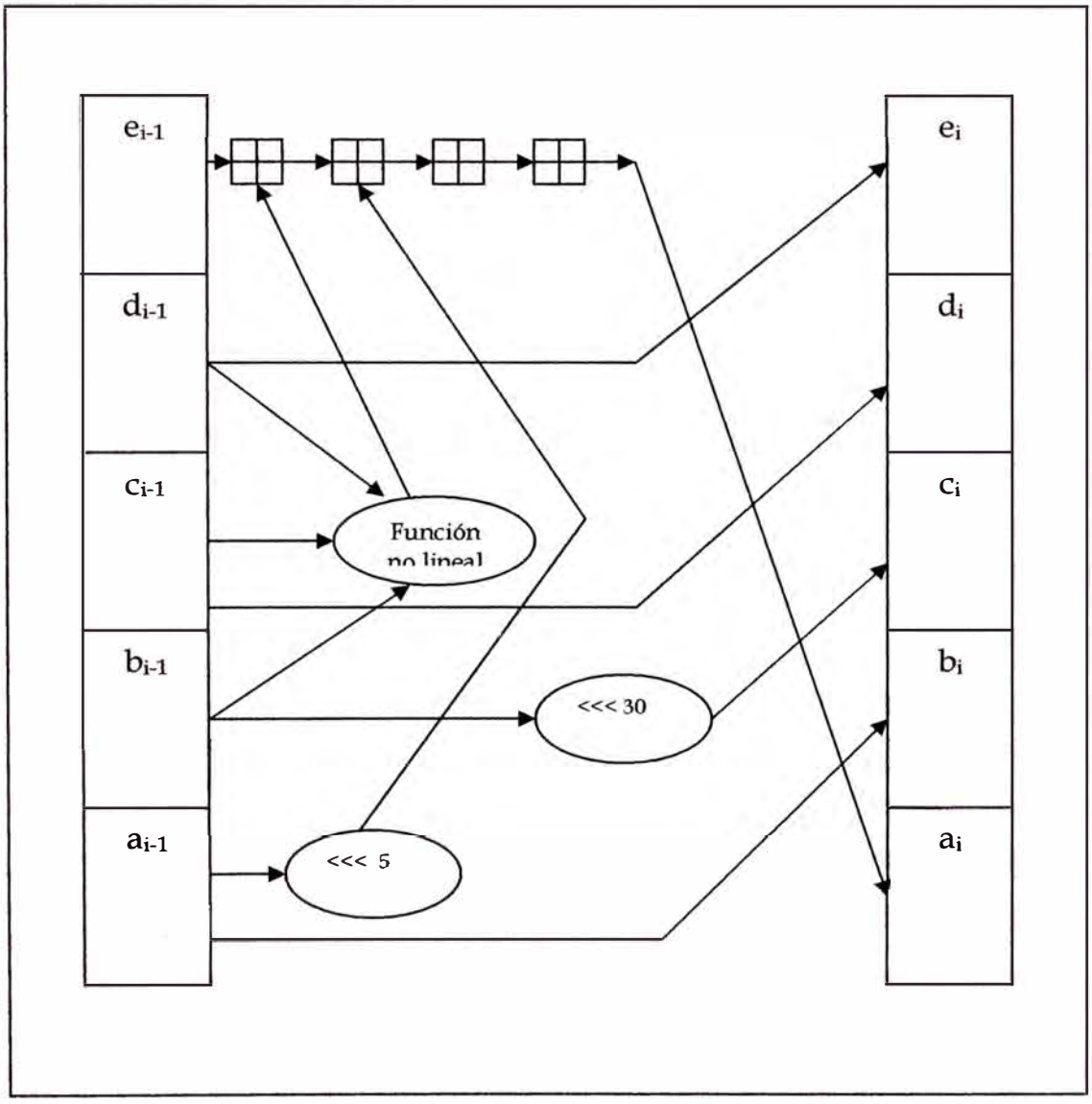
c=b <<< 30

b=a

e=TEMP

En la figura 4.7 se muestra una de las operaciones.

Fig. 4.7



SHA es muy similar a MD4, pero el valor de salida tiene 160 bits, los cambios principales son la adición de una transformación expandida, un round o vuelta extra, y

una mejor aplicación del "efecto de avalancha". Además no existen ataques criptográficos registrados contra SHA, ya que produce una salida de 160 bits es mas resistente a los ataques de brute-force que todos los algoritmos de 128 bits.

4.3.8 Funciones de un solo sentido utilizando algoritmos simétricos de bloques

Es posible utilizar algoritmos simétricos de bloques como funciones de one-way hash. La idea es que si el algoritmo de bloques es seguro, entonces la función de one-way hash también será segura. El método mas obvio es encriptar un mensaje con el algoritmo en modo CBC o CFB, una clave predeterminada, y el vector de inicialización (IV), luego el ultimo bloque de texto cifrado será el valor hash. En resumen este método utiliza en bloque de mensaje como clave y el valor hash previo como entrada y el valor actual de hash como la salida. La función hash propuesta real es un poco más complicada, el tamaño de bloque es frecuentemente el tamaño de bloque. Varios esquemas se diseñan con un hash que es dos veces el tamaño del bloque, ya que la mayoría de los algoritmos de bloques es de 64 bits.

a. Esquemas donde la longitud del hash es el mismo que el tamaño de bloque

El esquema general es como sigue:

$H_0 = I_H$, donde I_H es un valora inicial aleatorio.

$H_i = E_A (B) \Theta C$

Donde A, B y C, pueden ser M_i, H_{i-1}, $(M \Theta H_{i-1})$ o una constante (puede ser 0). H_0 es algún valor inicial aleatorio: I_H. Este mensaje se divide en trozos de tamaño de bloques

M_i, y se procesa cada bloque individualmente. Además esos mensajes individuales pasan por algún tipo de reforzamiento, ya sea MD5 o SHA. De este modo 3 variables diferentes pueden tomar cada una uno de cuatro valores posibles, por lo tanto que hay 64 posibles esquemas de este tipo.

4.4 ALGORITMOS DE CLAVE PÚBLICA

4.4.1 Introducción

Desde 1976, se han propuesto diversos algoritmos criptográficos de clave pública, muchos de estos son inseguros, y algunos que son seguros son demasiado imprácticos. Ya que tienen una clave muy grande o el texto cifrado es muy grande para el texto plano. De todos los algoritmos, algunos son apropiados para distribución de claves, otros para encriptación. Y otros solo para firmas digitales, solo 3 algoritmos trabajan bien para encriptación y firmas digitales, estos son: RSA, Legal y Rabin. Estos algoritmos son lentos, encriptan y desencriptan la data mucho mas lentamente que los algoritmos simétricos, usualmente es mucho más lento y en consecuencia imprácticos para soportar encriptación de grandes cantidades de data.

- **Seguridad de algoritmos de clave pública**

Si algún atacante tiene acceso a la clave publica, puede escoger cualquier mensaje para encriptar. Esto quiere decir que el criptoanalista, dado $C=E_K(P)$, puede adivinar el valor de P, y fácilmente verificar esta respuesta. Este es un problema serio si los mensajes de texto plano son pocos como para permitir una búsqueda exhaustiva, pero puede ser

solucionado adicionando a los mensajes una cadena de bits aleatorios. Esto hace que textos planos idénticos se encripten en mensajes diferentes. Los algoritmos de clave publica están diseñados para resistir ataques de texto plano escogido, su segundad se basa en la dificultad de deducir la clave secreta de la clave publica y en la dificultad de deducir el texto plano del texto cifrado. Sin embargo, la mayoría de los algoritmos de clave pública son particularmente susceptibles a los ataques de texto cifrado escogido.

En sistemas donde la operación de firmas digitales es la inversa de la operación de encriptación, este ataque es imposible de prevenir a menos que se utilicen diferentes claves para encriptación y firmas. En conclusión, es importante observar el sistema en su totalidad, y no solo las partes individuales. Los mejores protocolos de clave publica se diseñan de modo que las partes componentes no puedan desencriptar mensajes generados por otras partes.

4.4.2 Algoritmos Knapsack

El primer algoritmo para encriptación generalizada con clave pública fue el algoritmo knapsack. Inicialmente solo se podía utilizar para encriptación, este algoritmo obtienen su seguridad del problema de knapsack, un problema de NP-completo. Aunque posteriormente se demostró que este algoritmo era inseguro, se detallará su proceso para ver como el problema de NP se puede aplicar a la criptografía de clave pública.

El problema de knapsack es el siguiente: dada una pila de términos, cada uno con pesos diferentes, es posible colocar algunos de estos términos en una bolsa de modo que la

bolsa tenga un peso determinado?, en otras palabras: dado M_1, M_2, ..., M_n, y una suma S, hay que encontrar los valores de b_i , de modo que:

$$S= b_1M_1 + b_2M_2 + b_3M_3 +..... + b_nM_n$$

Los valores de b_i pueden ser cero o unos. Un uno indica que los ítems están dentro de la bolsa y un cero indica que no. El tiempo requerido para resolver este problema crece exponencialmente con el número de elementos dentro de la bolsa.

La idea central del algoritmo es codificar un mensaje como una solución de una serie de problemas knapsack. Un bloque de texto plano igual en tamaño al numero de elementos en la pila seleccionaría los elementos dentro de la bolsa (bits de texto plano correspondiente a los valores b), y el texto cifrado podría ser la suma resultante. La tabla 4.9 muestra texto plano encriptado con un problema clásico de knapsack.

Texto plano :	1 1 1 0 0 1	0 1 0 1 1 0	0 0 0 0 0 0	0 1 1 0 0 0
Knapsack:	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20
Texto cifrado:	1+5+6+20=	1+11+14=	0=	5+6=
	32	39	0	11

Tabla 4.9

En realidad hay dos problemas clásicos de knapsack, uno se puede resolver en tiempo lineal y el otro no. El problema knapsack fácil se puede modificar para convertirlo en el problema difícil de knapsack. La clave pública es el knapsack difícil, la cual se puede

utilizar fácilmente para encriptar pero no para la desencriptacion de mensajes. La clave privada corresponde al problema fácil de knapsack.

El problema fácil de knapsack consiste es que si una lista de los pesos es una secuencia creciente, luego el problema es fácil de resolver. Una secuencia creciente es una en la que cada nuevo término es mayor que la suma de todos los anteriores. Ej [1, 3, 6, 13, 27,52]. La solución a este problema es fácil de encontrar. Los problemas difíciles son los que corresponden a knapsack normales, ya que no tienen algoritmo de solución, la única forma de determinar que elementos están dentro de la bolsa es metódicamente probar posibles soluciones hasta que se encuentre finalmente con el correcto.

- **Creación de la clave publica a partir de la clave privada**

El algoritmo trabaja de la siguiente manera: para obtener una secuencia de un problema knapsack normal, tomar la secuencia de un problema knapsack creciente, por ejemplo [2, 5, 9, 20, 41, 83], y multiplicar por un numero n al resultado efectuar un mod con un numero m. Los módulos serian números mayores que la suma de todos los numero en la secuencia, por ejemplo 179 (>2+5+9+20+41+83). El número multiplicador tampoco tendrá que tener factores en común con los módulos, ejemplo, 31. Para este ejemplo, la secuencia normal knapsack sería:

2 * 31 mod 179 = 62

5 * 31 mod 179 = 155

9 * 31 mod 179 = 100

$20^* 31 \bmod 179 = 83$

$41^* 31 \bmod 179 = 18$

$83^* 31 \bmod 179 = 67$

Luego el knapsack normal tendría la secuencia: {62, 155, 100, 83, 18,67}. La secuencia creciente de knapsack es la clave privada, y la secuencia normal es la clave pública.

4.4.3 Función RSA

RSA es el mas fácil de entender e implementar de todos los algoritmos de encriptación y firmas digitales, también es el mas popular. Se llama RSA debido a sus tres inventores (Rivest, Shamir y Adleman), el algoritmo ha resistido años de criptoanalisis intensivo, hasta el momento no se ha logrado violar el algoritmo. Sin embargo el criptoanálisis no prueba ni desaprueba la seguridad de RSA, solo establece un nivel de certificación aceptable al algoritmo.

La seguridad de RSA se basa en la dificultad de factorizar grandes números. Las claves públicas y privadas son funciones de un par de números primos grandes (de 100 a 200 dígitos). Así que recuperar el texto de la clave pública y el texto cifrado se considera equivalente a factorizar el producto de los dos números primos. Para generar las dos claves, se selecciona aleatoriamente dos números primos grandes p y q. Para seguridad se pueden escoger ambos que tengan el mismo tamaño de dígitos. Luego se multiplican:

$n = pq$

Luego aleatoriamente se selecciona una clave de encriptación e, de modo que e y (p-1) (q-1) sean primos relativamente. Finalmente se utiliza el método de algoritmo euclidiano extendido para procesar la clave de desencriptacion d, de modo que:

$ed = 1 \bmod (p-1)(q-1)$

Nota: si $x = (1 \bmod n)$, entonces se cumple que $x = nk + 1$, donde k es el máximo entero positivo que cumpla con esa condición.

En otras palabras: $d = e^{-1} \bmod ((p-1)(q-1))$

Notar que d y n son también primos relativos. Los números e y n son la clave publica, el numero d es la clave privada. En este punto ya no se necesitan los números primos p y q. Estos números se descartan y nunca serán revelados.

Así, para encriptar un mensaje m, primero se divide en bloques de numéricos mas pequeños que n, es decir si p y q tienen 100 dígitos cada una, entonces n tendrá aproximadamente 200 dígitos, entonces cada bloque de mensaje m_i, tendrá 200 dígitos. El mensaje encriptado, c, estará compuesto de bloques de mensajes, c_i, de aproximadamente el mismo tamaño. La formula de encriptación es simplemente:

$c_i = m_i^e \bmod n$

Para desencriptar se tendrá que efectuar en cada bloque encriptado c_i y efectuar:

$m_i = c_i^d \bmod n$

a. Velocidad del RSA

Se han implementado diferentes chips con el algoritmo de RSA. Pero en hardware el RSA es aproximadamente 1000 veces mas lento que el DES, la implementación VLSI hardware mas rápida para RSA con un modulo de 512 bits tiene una salida de 64 kilobits por segundo. En software DES es aproximadamente 100 veces mas rápido que RSA.

b. Seguridad de RSA

La seguridad del RSA depende directamente de la dificultad en factorizar los grandes números primos que se utilizan en el algoritmo. Es también posible atacar RSA adivinando el valor de (p-1) (q-1) pero es tan difícil como la factorizacion.

Algunos ataques trabajan contra la implementación de RSA. Estos ataques no son contra el propio algoritmo si no contra el protocolo. Esto se puede dar del modo siguiente:

Supongamos que existe un intruso que tiene acceso al medio de transmisión de datos, si este intruso llega a interceptar un mensaje en texto cifrado enviado por A, detonado por c, encriptado con RSA con la clave publica de A. Si este intruso quiere leer el mensaje, matemáticamente quiere saber m, en el que: $m = c^d$

Para recuperar m, primero selecciona un numero aleatorio r, r es menor que n. Toma la clave publica e de A. Luego procesa:

$$x = r^e \bmod n$$

$$y = xc \bmod n$$

$$t = r^{-1} \bmod n$$

Si $x = r^e \bmod n$, entonces $r = x^d \bmod n$.

Ahora lo que puede hacer el intruso es hacer que A firme y con su clave privada, y luego desencripta y. Luego A envía al intruso:

$u = y^d \bmod n$

Ahora el intruso opera de la siguiente forma:

$tu \bmod n = r^{-1}y^d \bmod n = r^{-1} x^d c^d \bmod n = c^d \bmod n = m$

Finalmente el intruso consigue acceder a m.

4.4.4 Función Pohlig-Hellman

Es similar a RSA, y no es un algoritmo simetrico, ya que se utilizan diferentes claves para encriptación y desencriptacion. No es un esquema de clave pública, ya que las claves son deducibles fácilmente una de otra; las claves de encriptación y desencriptacion deben permanecer en secreto.

Como en RSA:

$$C = P^e \bmod n$$

$$P = C^d \bmod n$$

Donde

$$ed \equiv 1$$

la diferencia fundamental con RSA es que no deben ser primos grandes, debe permanecer parte de una clave secreta. Si alguien tuviera *e* y *n*, podrían calcular *d*. Sin conocimiento de *e* o *d*, un atacante potencial tendría que calcular:

$$e = \log_P C \bmod n.$$

4.4.5 Función Rabin

La seguridad de este algoritmo se basa en la dificultad de encontrar las raíces cuadradas de un número compuesto. Este problema es equivalente a factorizacion, a continuación un ejemplo de implementación de este esquema. Primero se escogen dos primos, p q, ambos congruentes a 3 mod 4. Estos primos son la clave secreta. El producto n = pq es la clave publica. Para encriptar un mensaje M (donde M debe ser menor que n), simplemente computar: $C = M^2 \bmod n$.

La desencriptacion del mensaje es fácil, ya que el receptor conoce p y q, utilizando el teorema del residuo: $m_1 = C^{(p+1)/4} \bmod p$. $m_2 = (p - C^{(p+1)/4}) \bmod p$. $m_3 = C^{(p+1)/4} \bmod q$. $m_4 = (q - C^{(p+1)/4}) \bmod q$

Luego escoger un entero a y b de modo que cumpla:

$a = q\ (q^{-1} \bmod p)$ $\qquad$ $b = p\ (p^{-1} \bmod q)$

Las cuatro posibles soluciones serán:

$M_1 = (am_1 + bm_3) \bmod n$ $\qquad$ $M_2 = (am_1 + bm_4) \bmod n$

$M_3 = (am_2 + bm_3) \bmod n$ $\qquad$ $M_4 = (am_2 + bm_4) \bmod n$

Uno de los mensajes M_1, M_2, M_3 o M_4, es M.

4.4.6 Función ElGamal

Esto se puede utilizar para firmas digitales y encriptación, toma su seguridad de la dificultad de calcular logaritmos discretos en un campo finito. Para generar un par de claves, primero hay que escoger un primo, p, y dos números aleatorios g y x, como que g y x son menores que p. Luego se pueden calcular:

$y = g^x \bmod p$

La clave publica es y, g y p. Donde g y p pueden compartirse entre un grupo de usuarios. La clave privada es x.

a. Firmas ElGamal

Para la firma de un mensaje M. Primero se escoge aleatoriamente un numero, k, de modo que k sea primo relativamente con (p-1).

Luego se ejecuta: $a = g^k \bmod p$. Luego se utiliza el método del algoritmo euclidiano para resolver b en la siguiente ecuación: $M = (xa + kb) \bmod (p-1)$.

La firma es el par a y b. El valor aleatorio, k, debe permanecer en secreto. Cada firma o encriptación ElGamal requiere de un valor de k diferente, y el valor se debe escoger aleatoriamente

b. Encriptación ElGamal

Para encriptar mensajes se procede con una modificación de ElGamal. Para encriptar un mensaje M primero se escoge aleatoriamente k, de modo que k sea primo relativamente con p-1. Luego calcular:

$a = g^k \bmod p$

$b = y^k M \bmod p$

El texto cifrado consiste en el par de a y b. Hay que especificar que el texto cifrado es dos veces el tamaño del texto plano.

4.5 ALGORITMOS DE FIRMAS DIGITALES CON CLAVE PÚBLICA

4.5.1 Algoritmo de firma digital (DSA)

DSA es una variante de loa algoritmos de firmas de Schnorr y ElGamal. El algoritmo utiliza los siguientes parámetros:

p = un numero primo de L bits de tamaño, donde L puede estar en el rango entre 512 a 1024 y es múltiplo de 64.

q = un factor primo de p-1 de 160 bits.

$g = h^{(p-1)/q} \bmod p$, donde h es cualquier numero menor que p-1 de modo que $h^{(p-1)/q} \bmod p$ sea mayor que 1.

x = un numero menor que q.

$y = g^x \bmod p$.

El algoritmo también utiliza una función de one-way hash: H (m). El standard de este algoritmo utiliza la función de Secure Hash Algoritm. Los tres primero parámetros, p, q y g son públicos y se pueden compartir en la red entre los usuarios. La clave privada es x y la clave publica es y. Para firmar un mensaje, m, se siguen los siguientes pasos:

- A genera un numero aleatorio k, menor que q.

- A genera lo siguiente:

$r = (gk \bmod p) \bmod q$ $\qquad$ $s = (k^{-1}(H(m)+xr)) \bmod q$

Los parámetros r y s son la firma de A, estas son enviadas a B.

- B verifica la firma, ejecutando lo siguiente:

$w = s^{-1} \bmod q$

$u_1 = (H(m) * w) \bmod q$

$u_2 = (rw) \bmod q$

$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$

Si $v = r$, entonces B concluye que la firma ha sido correctamente verificada.

4.5.2 Variantes de la función DSA

Esta variante requiere menos procesamiento para la firma ya que no es necesario calcular k^{-1}. Todos los parámetros son como en DSA. Para firmar un mensaje, m, A genera dos números aleatorios, k y d, ambos menores que q. La firma es:

$r = (g^k \bmod p) \bmod q$ $\quad s = (H(m) + xr) * d \bmod q$ $\quad t = kd \bmod q$

B verifica la firma de la siguiente manera:

$w = t/s \bmod q$ $\quad u_1 = (H(m) + w) \bmod q$ $\quad u_2 = (rw) \bmod q$

Luego, si se cumple que $r = ((g^{u1} * g^{u2}) \bmod p) \bmod q$, entonces queda confirmada la firma. La siguiente variante hace que sea más fácil la parte de la verificación de la firma. Todos los parámetros son como en DSA. Para firmar un mensaje, m, A genera un numero aleatorio, k, menor que q. La firma es:

$r = (g^k \bmod p) \bmod q$ $\quad s = k * (H(m) + xr)^{-1} \bmod q$

B verifica la firma de la siguiente forma:

$u_1 = (H(m) + s) \bmod q$ $\qquad\qquad$ $u_2 = (sr) \bmod q$

Luego, si $r = ((g^{u1} * y^{u2}) \bmod p) \bmod q$, lego se verifica la firma.

4.5.3 Algoritmo de firma Algost

Esta es una firma Standard, el algoritmo es muy similar a DSA, pero usa los siguientes parámetros:

p = un numero primo, cuyo tamaño sea entre 509 y 512 bits, o entre 1020 y 1024 bits.

q = un factor primo de p-1, cuyo tamaño este entre 254 a 256 bits.

a = cualquier numero menor que p-1, de modo que $a^q \bmod p = 1$.

x = un numero menor que q.

$y = a^x \bmod p$.

Este algoritmo también hace uso de la función de one-way hash: H(x). El Standard especifica que la función de one-way has sea del tipo GOST. Los primeros tres algoritmos p, q y a, son públicos. La clave privada es x, la clave publica es y. Para firmar un mensaje m enviado por A hacia B, se siguen los siguientes pasos:

Paso 1: A genera un numero aleatorio k, menor que q.

Paso 2: A genera lo siguiente:

$$r = (a^k \bmod p) \bmod q$$

$$s = (xr + k(H(m))) \bmod q$$

Paso 3: B verifica la firma calculando lo siguiente:

$v = H(m)^{q-2} \bmod q$

$z_1 = (sv) \bmod q$

$z_2 = ((q-r) * v) \bmod q$

$u = ((a^{z1} * y^{z2}) \bmod p) \bmod q$

Si u=r, entonces se verifica la firma.

4.6 ESQUEMAS DE IDENTIFICACIÓN

4.6.1 Esquemas de identificación Beige-Fiat-Shamir

a. Esquema de identificación simplificado Feige-Fiat-Shamir

Antes de escoger las claves privadas, el arbitro (parte neutral del protocolo) escoge un numero aleatorio, n, que es el producto de dos números primos grandes. En la práctica, n debería ser al menos de 512 bits y cercano a 1024 bits. Para generar la clave publica y privada, el árbitro selecciona un número, v, donde v es un residuo cuadrático mod n. Es decir, v cumple con una solución para la ecuación: $x^2 \equiv v \pmod n$ además ($v^{-1} \bmod n$) existe, esta seria la clave publica. Luego se calcula s, como el valor mas pequeño que cumple con $s \equiv \text{sqrt}(v^{-1}) \pmod n$. Esta seria la clave privada.

Con los datos anteriores, el protocolo procede de la siguiente forma:

Paso 1: A escoge un r aleatorio, donde r es menor que n. Luego ejecuta $x = r^2 \bmod n$, envía x a B.

Paso 2: B envía a X un bit aleatorio, b.

Paso 3: Si b=0, luego A envía r a B. Si b=1, luego A envía a B y = r*s mod n.

Paso 4: Si b=0, B verifica que $x = r^2$ mod n, ya que A conoce sqrt(x). si b=1, B verifica que $x=y^2 * v$ mod n, ya que B conoce $sqrt(v^{-1})$.

Este esta operación se repite t veces hasta que B se convenza que A conoce s.

b. Esquema de identificación Feige-Fiat-Shamir

Se puede aumentar el número de acreditaciones por loop y reducir las interacciones entre A y B. Primero se genera n como en el método anterior, como el producto de dos números primos grandes. Para generar las claves publicas y privadas de A, primero escoger k números diferentes: $v_1, v_2,\ldots, v_k$, donde cada v_i es un residuo cuadrático mod n. en otras palabras, escoger vi de modo que la ecuación $x^2 = v_i$ mod n tenga una solución y que v_i^{-1} mod n exista. Esta cadena, $v_1, v_2, \ldots, v_k$, es una clave publica. Luego calcular el s_i mas pequeño de modo que $s_i = sqrt\ (v_i^{-1})$ mod n. esta cadena $s_1, s_2, \ldots, s_k$, es la clave privada. El protocolo es de la siguiente manera:

Paso 1: A escoge un numero aleatorio r, donde r es menor que n. luego ejecuta $x=r^2$ mod n, y envía x a B.

Paso 2: B envía a A una cadena binaria aleatoria: $b_1, b_2, \ldots.,b_k$.

Paso 3: A ejecuta $y = r * (s_1^{b1} * s_2^{b2} * \ldots. * s_k^{bk})$ mod n.

Paso 4: B verifica que $x= y^2 * (v_1^{b1} * v_2^{b2} * \ldots. * v_k^{bk})$ mod n.

A y B repitan esto t veces hasta que B se convenza de que A conoce $s_1, s_2, \ldots, s_k$.

c. Esquema de firma de Fiat-Shamir

La principal ventaja de esta firma sobre la firma RSA es la rapidez, Fiat-Shamor requiere solo del 1 a 4 por ciento de de la multiplicación modular RSA.

El inicio es el mismo que el esquema de identificación, escoger n como un producto de dos numero primos grandes. Se genera la clave publica, $v_1, v_2, \ldots, v_k$, y la clave privada $s_1, s_2, \ldots, s_k$, de modo que $s_i = \text{sqrt}(v_i^{-1}) \bmod n$.

Paso 1: A escoge t enteros aleatorios entre 1 y n: $r_1, r_2, \ldots, r_t$, y ejecuta $x_1, x_2, \ldots, x_t$ de modo que $x_i = r_i^2 \bmod n$.

Paso 2: A combina la concatenación del mensaje y la cadena de x_is para generar la cadena de bits: H (m, $x_1, x_2, \ldots$ xt). A utiliza los primeros k*t bits de esta cadena como valores de b_{ij}, donde i va de 1 a t y j va de 1 a k.

Paso 3: A calcula $y_1, y_2, \ldots, y_t$, donde: $y_i = r_i * (s_1^{bi1} * s_2^{bi2} * \ldots * s_k^{bik}) \bmod n$.

Paso 4: A envía a B a m , todos los valores de bit de $b_{i,j}$ y todos los valores de y_i. B ya tiene la clave publica de A: $v_1, v_2, \ldots, v_k$.

Paso 5: B calcula $z_1, z_2, \ldots, z_t$, donde:

$Zi = yi2 * (v_1^{bi1} * v_2^{bi2} * \ldots * v_k^{bik}.) \bmod n$.

Paso 6: B verifica que los primeros k*t bits de H (m, z1, z2,..., zt) son los valores $b_{i,j}$ que A envió previamente.

4.6.2 Esquemas de identificación Guillou-Quiquarter

El protocolo anterior minimiza el procesamiento incrementando el número de iteraciones y también el número de acreditaciones por cada iteración. Pero esto no es practico para algunas implementaciones como los smart cards (o tarjetas inteligentes o tarjetas electrónicas). En respuesta a esto se desarrolla un algoritmo de identificación más apropiado para estas aplicaciones en la que los intercambios entre ambas partes y las acreditaciones paralelas en cada intercambio se mantengan en lo mínimo: dando como resultado un solo intercambio de una acreditación en cada prueba. Para el mismo nivel de seguridad, el procesamiento que necesita el método de Guillou-Quisper es mayor que el de Feige-Fiat-Shamir en un factor de tres. Además este algoritmo de identificación puede convertirse a algoritmo de firma digital.

Supongamos que A es un smart card que quiere probar su identidad a B. la identidad de A consiste en una serie de credenciales entre las que se encuentran el nombre de la tarjeta, el periodo de validez, un numero de cuenta de banco, y cualquier otra garantía de aplicaciones. A toda esta cadena de bits llamémosla J, esto es análoga a la clave pública. Cualquier otra información pública, que se comparte entre todas las As que podrían usar esta aplicación, sería un exponente n y un modulo n, donde n es el producto de dos números primos secretos. La clave privada es P, donde $JP^V \equiv 1 \pmod{n}$.

A envía sus credenciales a B, estas son J. para demostrarle a B que estas credenciales con de A, tiene que convencer a B que conoce P. A continuación el protocolo:

Paso 1: A toma in entero cualquiera r, donde r esta entre 1 y n-1. Luego ejecuta $T = r^v$ mod n y lo envía a B. A toma in entero cualquiera r, donde r esta entre 1 y n-1. Luego ejecuta T = r.

Paso 2: B selección un numero cualquiera, d, de modo que este entre cero y v-1. B envía d a A.

Paso 3: A ejecuta $D = rP^d$ mod n, este valor lo envía a B.

Paso 4: B ejecuta $T' = D^v J^d$ mod n. Si $T \equiv T'$ (mod n), entonces la autenticación esta comprobada.

4.6.3 Esquema de firma de Guillou-Quisper

Esta identificación se puede convertir en un esquema de firma, también apropiada para implementaciones de tarjetas inteligentes electrónicas. La forma de acordar cuales serán las claves pública y privada es la misma que en la explicación anterior. El protocolo es como sigue:

Paso 1: A toma un entero cualquiera, r, pero que cumpla con estar entre 1 y n-1. Luego ejecuta $T = r^v$ mod n.

Paso 2: A ejecuta d = H (M, T), donde M es el mensaje que esta firmando y H(x) es una función de tipo de one-way hash. El valor de d debe estar entre 0 y v-1. Si la salida de la función hash no esta dentro de este rango, se debe reducir mediante la operación "modulo v".

Paso 3: A ejecuta $D = rP^d \bmod n$. La firma consiste del mensaje M y dos valores que se calculó previamente: d y D, y las credenciales J. A envía esta firma a B.

Paso 4: B comprueba calculando $T' = D^v J^d \bmod n$. Luego calcula d'=H (M, T'). Si d = d', entonces B concluye que A tiene la clave privada, P, y la firma es válida.

a. Esquemas de identificación Schnorr

La seguridad de este algoritmo se basa en la dificultad del cálculo de logaritmos discretos. Para generar un par de claves, se toman dos números primos, p y q, donde q es un factor primo de p-1. Luego, se escoge un número a diferente de 1, a condición que cumpla con $aq \equiv 1 \pmod p$. Todos estos números pueden ser comunes a un grupo de usuarios y se publican libremente.

Para generar un par de claves publica y privada en particular, se selecciona un número menor que q. Este seria la clave privada, s. Luego se calcula $v = a^{-s} \bmod p$. Donde v es la clave pública.

b. Protocolo de autenticación

Paso 1: A escoge un número cualquiera r, menor que q, luego calcula $x = a^r \bmod p$.

Paso 2: A envía x a B.

Paso 3: B envía a A un numero cualquiera e, cuyo valor esta entre 0 y 2^t-1.

Paso 4: A calcula $y = (r + se) \bmod q$, luego envía y a B.

Paso 5: B verifica que $x = a^y v^e \bmod p$.

En este protocolo la seguridad depende del parámetro t. la dificultas de quebrar el algoritmo es de 2^t. Se recomienda que p tenga alrededor de 512 bits, q 140 bits y t 72.

4.6.4 Conversión de esquemas de identificación a esquemas de firmas

Hay un método estándar para convertir un esquema de identificación en un esquema de firma: remplazar a B con una función de one-way hash. El mensaje no pasa por la función hasta ser firmado por A; en vez de esto la función se adiciona en el algoritmo de firma. Teóricamente, esto se puede hacer con cualquier sistema de identificación.

4.7 ALGORITMOS DE INTERCAMBIO DE CLAVES

4.7.1 Algoritmo Diffie-Hellman

La seguridad de este algoritmo se basa en la dificultad del cálculo de logaritmos discretos en un campo finito. Este algoritmo se utiliza para generar una clave secreta, pero no para encriptar o desencriptar mensajes. El proceso del algoritmo comienza cuando las partes, A y B seleccionan dos números primos grandes, n y g, estos dos números no son secretos. Luego:

Paso 1: A selecciona un número grande entero cualquiera x, envía x a B.

Paso 2: B selecciona otro número grande cualquiera, y, envía y a A.

Paso 3: A calcula:

$$k = Y^x \bmod n$$

Paso 4: B calcula

$k'=X^Y \bmod n$

Donde, k y k' son iguales a $g^{xy} \bmod n$. nadie puede calcular estre valor, ya que solo podrian tener acceso a n, g, X e Y. A menos que calculen el logaritmo discreto y recuperen x o y. Finalmente, la clave secreta es k.

4.7.2 Protocolo de estación a estación

El algoritmo anterior es vulnerable a ataques de interceptación de mensajes en el medio de transmisión del mismo. Una forma de prevenir esto es hacer que A y B firmen sus mensajes entre ellos. Este protocolo asume que A tiene el certificado con la clave pública de B y que B tiene el certificado con la clave pública de A. La forma en que como generan una clave secreta k es la siguiente:

Paso 1: A genera un número cualquiera, x, y lo envía a B.

Paso 2: B genera un número cualquiera, y. Utilizando el algoritmo anterior calcula su clave secreta basada en x e y:k. B firma x e y, además encripta la firma utilizando k. luego envía con y, a A:

$y, E_K (S_B(x,y))$

Paso 3: A también calcula k. lego desencripta k. A desencripta el resto del mensaje de de B y verifica la firma. Luego envía a B, una firma que consiste en x e y, encriptado en la clave compartida:

$E_K (S_A(x, y))$

Paso 4: B desencripta el mensaje y verifica la firma de A.

4.7.3 Intercambio encriptado de claves

Tambien llamado protocolo Encrypted Key Exchange (EKE), proporciona seguridad y autenticación en redes de computadoras, utiliza criptografía simétrica y de clave pública de una forma nueva: una clave secreta común se utiliza para encriptar una clave pública generada aleatoriamente.

El protocolo procede de la siguiente forma: A y B comparten un password común, P. Utilizando este protocolo, pueden autenticarse uno al otro y generar una clave común de sesion, K.

Paso 1: A genera una par cualquiera de claves pública/privada. A encripta la clave pública, K', utilizando un algoritmo simétrico y P como clave: E_P (K'), luego lo envía a B:

A, E_P (K')

Paso 2: B conoce P. B desencripta el mensaje para obtener K'. Luego, B genera una sesión aleatoria, K, y encripta con la clave pública y lo encripta con clave pública que recibió anteriormente de A, y luego encripta el total con P. Envía esto a A:

E_P ($E_{K'}$ (K))

Paso 3: A desencripta el mensaje para obtener K. Luego A genera un cadena cualquiera, R_A, lo encripta con K y lo envía a B.

E_K (R_A)

Paso 4: B desencripta el mensaje y obtiene R_A. Luego B genera otra cadena aleatoria, R_B, encriptada con, envía el resultado a A.

$E_K (R_A, R_B)$

Paso 5: A desencripta el mensaje y obtiene R_A y R_B. Asumiendo que el R_A que recibió de B es el mismo al que envió a B en el paso (3), luego A encripta R_B con K y lo envía a B.

$E_K (R_B)$

Paso 6: B desencripta el mensaje y obtiene R_B. En caso de que el R_B que recibió de A es el mismo que B envió a A en el paso (4), el protocolo esta completo. Ambas partes ahora se comunican utilizando K como la clave de sesión.

CAPÍTULO V

CÓDIGO FUENTE DE ALGORITMOS

5.1 EJEMPLOS DE IMPLEMENTACIÓN

Muchas veces los criptosistemas diseñados no han sido aplicables a la práctica, en algunos casos por que se requería demasiado ancho de banda, o porque era muy lento, etc. A continuación los protocolos mas conocidos.

5.1.1 Protocolo PGP

PGP (Pretty Goog Privacy) es un programa de seguridad para correo electrónico, es freeware. Utiliza IDEA para encriptación de datos, RSA para el manejo de la clave y firmas digitales y MD5 como una función hash.

Los certificados del sistema de correo PGP funcionan mediante niveles de confianza. El sistema seria ideal si todos los certificados llegaran firmados por una persona a la que se le ha comprobado la identidad mediante su clave pública, pero esto no se cumple así.

Además, una clave sin certificado solo es confiable si se transmite personalmente o por medio de un medio de comunicación pública. El teléfono y la Web no son vías seguras. En PGP se asigna dos niveles de confianza a cada clave pública de la base de datos, cada usuario genera y distribuye su propia clave, así se crea una comunidad interconectada de usuarios PGP, los niveles de seguridad son:

a. Confianza propia

La confianza en la clave pública del usuario calculada según el procedimiento por el que ha llegado:

- Directamente: de confianza máxima.
- Por certificado: dependiendo de la firma de una tercera persona

b. Confianza para firmar certificados

Una clave pública puede tener una confianza propia muy alta porque ha llegado por un sistema seguro, pero puede ser que no se pueda confiar en las firmas de certificados de este usuario, porque firma a cualquiera sin confirmar su procedencia.

Los mensajes encriptados con PGP tienen implementada la seguridad por capas. Lo único que un criptoanalista puede conocer de un mensaje encriptado es quien en es que recibe el mensaje, y esto es asumiendo que el criptoanalista conoce la clave del receptor.

5.1.2 Protocolo PEM

PEM (Privacy Enhanced Mail) es el protocolo que proporciona correos electrónicos seguros en el Internet, es el más común actualmente. Este protocolo permite efectuar la encriptación, autenticación, integridad del mensaje y manejo de las claves. Los procedimientos de PEM son compatibles con un amplio rango de procedimientos de manejo de clave, incluyendo esquemas simétricos y de clave pública para encriptación de data y encriptación de claves. La criptografía simétrica es utilizada para encriptación de mensajes de texto. Los algoritmos de de funciones de hash se utilizan para la integridad del mensaje. PEM soporta solo ciertos algoritmos, pero permite que diferentes tipos de algoritmos. Los mensajes se encriptan con DES en el modo CBC. La autenticación, la proporciona el Message Integrity Check (MIC), este proceso utiliza el MD2 o MD5. El manejo de claves puede ser mediante el DES en modo ECB o mediante el algoritmo T-DES. PEM también soporta certificados de clave pública para el manejo de la clave, utilizando el algoritmo RSA y el estándar X.509 para la estructura del certificado.

5.1.3 Protocolo Kerberos

Es un protocolo de autenticación diseñado para protocolos TCP/IP. El servicio kerberos actual como un árbitro confiable. Kerberos garantiza una autenticación de red segura, permitiendo que una persona acceda a diferentes máquinas en la red. Kerberos se basa en la criptografía simétrica (DES de preferencia, pero también permite otros algoritmos).

Este protocolo comparte una clave secreta con cada entidad de toda la red y el conocimiento de esta clave secreta es equivalente a la prueba de identidad.

En el modelo Kerberos has dos tipos de entidades, clientes y servidores. Los clientes pueden ser los usuarios, o programas software independientes que necesiten efectuar tareas como bajar archivo de Internet, enviar mensajes, acceder a base de datos, acceder a impresoras, etc. De este modo Kerberos guarda la base de datos de los clientes y sus respectivas claves secretas, debido a esto Kerberos puede crear mensajes que convencen a una entidad de la identidad de otra entidad de la red.

Kerberos trabaja de la siguiente manera: un cliente pide un ticket a kerberos para el servicio de Ticket-Granting (TGS). Este ticket se envía al cliente, encriptado en la clave secreta del cliente. Para usar un servicio en particular, el cliente solicita al TGS un ticket para este servicio. Luego el cliente presenta este ticket conjuntamente con el autenticador al servidor, si todo esta bien el servidor le da acceso al cliente al servicio requerido.

a. Credenciales

Kerberos utiliza dos tipos de credenciales: tickets y autenticadores. Un ticket es para demostrar al servidor de forma segura la identidad del portador del ticket, también contiene información que el servidor utiliza para asegurar que el cliente que tiene el

ticket es el mismo que lo solicito al TGS. El autenticador es un parámetro adicional que se presenta con el ticket. Un ticket Kerberos tiene la siguiente forma:

$$T_{c,s} = s, \{ c, a, v, K_{c,s} \} K_s$$

Donde:

c = cliente

s = servidor

a = dirección de red del cliente

v = tiempo inicio y fin de validación del ticket

$K_{c,s}$ = clave de sesión para c y s

K_s = clave secreta de s.

$T_{c,s}$ = el ticket de c para usar con s.

Esta información se encripta con la clave secreta del servidor. Una vez que el cliente recibe este ticket, puede utilizarlo varias veces con el mismo servidor hasta que el ticket expire. El autenticador Kerberos tiene los siguientes componentes:

$A_{c,s} = \{c, t, clave\} K_{c,s}$ Donde:

$A_{c,s}$ = autenticador de c a s t = marcador de tiempo

El cliente lo genera cada vez que quiere utilizar el servicio del servidor. El autenticador contiene el nombre del cliente, un marcador de tiempo, y opcionalmente, una clave de sesión adicional , todo esto encriptado con la clave de sesión compartida entre el cliente y el servidor.

5.2 POLÍTICAS

A continuación los organismos multinacionales relacionados con la encriptación y seguridad de los algoritmos.

5.2.1 La Agencia de Seguridad Nacional

NSA (National Security Agency) encabeza la investigación en criptografía, entre sus objetivos esta el diseño de algoritmos seguros para proteger las comunicaciones de EEUU y el diseño de técnicas de criptoanálisis para acceder a información que no es de EEUU.

El NSA utiliza su poder para restringir la disponibilidad pública hacía software de criptografía, de este modo evita que enemigos nacionales utilicen métodos de encriptación demasiado complicados como para que la NSA no lo pueda quebrar.

5.2.2 Instituto Nacional de Estándares y Tecnología

El NIST (National Institute of Standards and Technology) es una división del departamento de comercio de los EEUU, anteriormente llamado NBS (Nacional Bureau of Standards), NIST promueve, a través de su Laboratorio de Sistemas de Informática (CSL), standards de protocolos libres y conocidos e interoperatividad que contribuirá con el desarrollo de las industrias de informática. Con este fin, NIST pública guías de usuarios y estándares que espera que se adopten en todos los sistemas de computadoras de los Estados Unidos. Estos estándares oficiales se conocen como FIPS (Federal Information Processing).

El NIST esta a cargo de garantizar la seguridad de información sensible (que sin embargo no es información clasificada) en los sistemas de computadoras del gobierno. Existe un Acta del congreso que autoriza a NIST a trabajar con otras agencias de gobierno e industria privada para evaluar estándares tecnológicos concernientes a seguridad.

5.2.3 Importación y exportación de criptografía

Los países pueden tener reglas concernientes a este tema e ignorarlas o pueden no tener reglas pero establecer restricciones de importación o exportación.

Veamos algunos ejemplos:

- Australia requiere un certificado de importación para el software criptográfico.
- Según los EEUU, la criptografía es considerada como municiones, esto significa que su uso esta bajo las mismas reglas de un misil o un tanque. De modo que si se vende criptografía sin la debida licencia de exportación.
- Canadá no tiene controles para la importación, pero los controles para exportación son similares a los de Estados Unidos. Los requerimientos de exportación de Canadá pueden estar sujetos a restricción si estos están incluidos en una lista llamada Export Control List.
- Francia no tiene reglas especiales de importación o exportación de criptografía, sin embargo tienen regulaciones con respecto a la venta y uso de criptografía dentro de su país. Todos los productos deben estar certificados, ya sea cumpliendo una

especificaciones públicas, o la especificaciones de la compañía propietaria deben entregarse al gobierno. El gobierno puede también pedir unos ejemplares de muestra para su uso propio. De modo que las compañías deben tener licencia para vender criptografía, esta licencia especifica el producto y los usuarios también deben tener licencia de compra y uso de criptografía.

Alemania requiere una licencia para exportar criptografía. Ellos mantienen control específico sobre el software de criptografía dentro de su país.

5.2.4 Patentes

Los algoritmos criptográficos pueden ser patentados en la mayoría de los países, IBM patento el DES. IDEA esta patentado. Casi todos los algoritmos de clave pública están patentados. El NIST tiene la patente para el DSA. Algunas patentes han sido interpuestas por el NSA, esto quiere decir que el inventor en vez de una patente obtiene un orden de confidencialidad y se le prohíbe discutir su invento con ninguna persona.

CONCLUSIONES

1. Después de escribir este informe se han dado a conocer diferentes puntos de partida para el tema de la seguridad, enfocandolo desde el nivel de programas aplicativos hasta el nivel de transmisión de datos. Esto nos enseña que mientras más actualizados e informados estemos con respecto a potenciales vulnerabilidades de los protocolos y programas que estemos utilizando mejor será nuestra respuesta o reacción frente a ellas.

2 Tengamos claro que la metodología de un atacante para ingresar a un sistema determinado toma el siguiente camino: footprintong, scanning y enumeración, una vez logrado el objetivo proceden con otros métodos que van desde la instalación de troyanos hasta la infección con virus del sistema víctima

3. En cuanto a los diferentes algoritmos de encriptación detallados, existe un algoritmo apropiado para cada aplicación dados los requerimientos de la misma,

adicionalmente existen algoritmos que no pueden todavía ser implementados ya que requieren de procesadores con características que aun no existen.

4. El aspecto más peligroso de la criptografía es que es medible, es decir, sabiendo el protocolo que se está utilizando, existe un factor que podriamos considerar como "trabajo" necesario para hacer romper el sistema de criptoanalisis, sin embargo este factor muchas veces no es posible alcanzar dadas las limitaciones actuales de velocidad de procesmiento de los sistemas, pero esto no asegura que en un futuro cercano los algoritmos considerados "seguros" por el momento continuen siendolo. Además, no hay un procedimiento específico para hacer segura una red o sistema, simplemente es una compleja suma de procesos que hacen que el sistema sea mucho mas difícil de ingresar para el atacante o pirata cibernético que para un usuario legítimo.

BIBLIOGRAFÍA

1. Scambray, Joel; MacClure, Stuart; Kurtz, George. "Hacking Exposed" Ed. MC Graw-Hill, 2001.

2. Schneier, Bruce. "Applied Cryptography" Ed. John Wiley, 1998.

3. Cohen, Frederick "Protection and Security on the Information Superhighway" Ed. MC Graw-Hill, 1994.

4. Hatch, Brian. "Hacking Secrets" Ed. John Wiley, 2001.

5. Menezes, Alfred; Van Oorschot, Paul. "Handbook of Applied Cryptography" Ed. CRC Press, 1996.

6. Northcutt, Stephen; Novak, Judy. "Network Intrusion Detection" Ed. New Riders Publishing, 2002.

7. Goldreich, Oded. "Foundations of Cryptography: Basic Tools" Ed. Cambridge University Press, 2001

8. Schiffman, Mike. "Hacker's Challenge: Test Your Incident Response Skills Using 20 Scenarios" Ed. MC Graw-Hill, 2001.

9. Criptography research: www.cryptography.com