Emoticono	Significado	Emoticono	Significado	Emoticono	Significado
:-)	Estoy feliz	=l:-)	Abe Lincoln	:+)	Nariz grande
:-(	Estoy triste/enojado	=):-)	Tío Sam	:-))	Realmente feliz
:-I	Estoy apático	*<:-)	Santa Claus	:-{)	Con bigote
;-)	Estoy guiñando un ojo	<:-(	Bobo	#:-)	Peinado yuppy
:-(O)	Estoy gritando	(-:	Zurdo	8-)	Con gafas
:-(*)	Estoy vomitando	:-)X	Hombre con moño	C:-)	Inteligente

Figura 7-6. Algunos emoticonos. No estarán en el examen final :-)

contenía la dirección del destinatario. A medida que pasó el tiempo, las limitaciones de este enfoque se hicieron obvias. Algunas de las quejas eran:

1. El envío de un mensaje a un grupo de personas era laborioso. Los administradores con frecuencia necesitaban esta facilidad para enviar memorandos a todos sus subordinados.
2. Los mensajes no tenían estructura interna, lo que dificultaba el proceso por computadora. Por ejemplo, si un mensaje reenviado se incluía en el cuerpo de otro mensaje, era difícil extraer la parte reenviada del mensaje recibido.
3. El originador (remitente) nunca sabía si el mensaje había llegado o no.
4. Si alguien planeaba ausentarse por un viaje de negocios durante varias semanas y quería que todo el correo entrante fuera manejado por su secretaria, esto no era fácil de arreglar.
5. La interfaz de usuario estaba mal integrada al sistema de transmisión, pues requería que el usuario primero editara un archivo, luego saliera del editor e invocara el programa de transferencia de archivos.
6. No era posible crear y enviar mensajes que contuvieran una mezcla de texto, dibujos, facsímiles y voz.

A medida que se acumuló experiencia, se propusieron sistemas de correo electrónico más elaborados. En 1982, se publicaron las propuestas de correo electrónico de ARPANET como el RFC 821 (protocolo de transmisión) y el RFC 822 (formato de mensaje). Las revisiones menores, los RFCs 2821 y 2822, se han vuelto estándares de Internet, pero todas las personas aún se refieren al correo electrónico de Internet como el RFC 822.

En 1984, el CCITT redactó la recomendación X.400. Después de dos décadas de competencia, los sistemas de correo electrónico basados en el RFC 822 se utilizan ampliamente, mientras que aquellos basados en X.400 han desaparecido. El hecho de que un sistema creado por un grupo de estudiantes de licenciatura en ciencias de la computación haya vencido a un estándar internacional que estaba fuertemente respaldado por todos los PTTs en el mundo, muchos gobiernos y una parte significativa de la industria de la computación nos recuerda a la historia bíblica de David y Goliath.

La razón del éxito del RFC 822 no es que sea tan bueno, sino que X.400 estaba tan pobremente diseñado y era tan complejo que nadie podía implementarlo bien. Dada la opción entre un sistema de correo electrónico basado en el RFC 822 simple, pero funcional, y un sistema de correo electrónico X.400 supuestamente maravilloso, aunque complejo, la mayoría de las organizaciones eligen el primero. Tal vez aquí haya una lección. En consecuencia, nuestro análisis del correo electrónico se concentrará en el sistema de correo electrónico de Internet.

7.2.1 Arquitectura y servicios

En esta sección ofreceremos un panorama de lo que pueden hacer los sistemas de correo electrónico y la manera en que están organizados. Normalmente consisten en dos subsistemas: los **agentes de usuario,** que permiten a la gente leer y enviar correo electrónico, y los **agentes de transferencia de mensajes,** que mueven los mensajes del origen al destino. Los agentes de usuario son programas locales que proporcionan un método basado en comandos, en menús o en una interfaz gráfica para interactuar con el sistema de correo electrónico. Los agentes de transferencia de mensajes son por lo común **demonios** (***daemons***) del sistema que operan en segundo plano y mueven correo electrónico a través del sistema.

Por lo general, los sistemas de correo electrónico desempeñan cinco funciones básicas, como se describe a continuación.

La **redacción** se refiere al proceso de crear mensajes y respuestas. Aunque es posible utilizar cualquier editor de texto para el cuerpo del mensaje, el sistema mismo puede proporcionar asistencia con el direccionamiento y los numerosos campos de encabezado que forman parte de cada mensaje. Por ejemplo, al contestar un mensaje, el sistema de correo electrónico puede extraer la dirección del remitente e insertarla en forma automática en el lugar adecuado de la respuesta.

La **transferencia** se refiere a mover mensajes del remitente al destinatario. En gran medida, esto requiere establecer una conexión con el destino o alguna máquina intermedia, enviar el mensaje y liberar la conexión. El sistema de correo electrónico debe hacer esto en forma automática, sin molestar al usuario.

La **generación del informe** tiene que ver con indicar al remitente lo que ocurrió con el mensaje: ¿se entregó, se rechazó o se perdió? Existen muchas aplicaciones en las que la confirmación de la entrega es importante y puede incluso tener una aplicación legal (“Bien, Su Señoría, mi sistema de correo electrónico no es muy confiable, por lo que creo que la citación electrónica se perdió en algún lado”).

La **visualización** de los mensajes de entrada es necesaria para que la gente pueda leer su correo electrónico. A veces se requiere cierta conversión o debe invocarse un visor especial, por ejemplo, si el mensaje es un archivo PostScript o voz digitalizada. Algunas veces también se intentan conversiones y formateo sencillos.

La **disposición** es el paso final y tiene que ver con lo que el destinatario hace con el mensaje una vez que lo recibe. Las posibilidades incluyen tirarlo antes de leerlo, desecharlo después de leerlo, guardarlo, etcétera. También debe ser posible recuperar y releer mensajes guardados, reenviarlos o procesarlos de otras maneras.

Además de estos servicios básicos, la mayoría de los sistemas de correo electrónico, especialmente los corporativos internos, proporcionan una gran variedad de características avanzadas. Brevemente mencionaremos algunas de éstas. Cuando la gente se muda, o cuando está lejos durante cierto tiempo, podría querer el reenvío de su correo electrónico, por lo que el sistema debe ser capaz de hacer esto de manera automática.

La mayoría de los sistemas permite a los usuarios crear **buzones de correo** para almacenar el correo entrante. Se requieren comandos para crear y destruir buzones de correo, inspeccionar el contenido de los buzones, insertar y borrar mensajes de los buzones, etcétera.

Los gerentes corporativos con frecuencia necesitan enviar un mensaje a cada uno de sus subordinados, clientes o proveedores. Esto da pie al concepto de **lista de correo,** que es una lista de direcciones de correo electrónico. Cuando se envía un mensaje a la lista de correo, se entregan copias idénticas a todos los de la lista.

Otras características avanzadas son copias ocultas, correo electrónico de alta prioridad, correo electrónico secreto (es decir, encriptado), destinatarios alternos si el primario no está disponible y la capacidad de que las secretarias manejen el correo electrónico de su jefe.

En la actualidad, el correo electrónico se usa ampliamente en la industria para la comunicación intracompañía. Permite que los empleados remotos cooperen en proyectos complejos, incluso si están en otros husos horarios. Al eliminar la mayoría de las señales asociadas al rango, edad y género, los debates realizados por correo electrónico tienden a enfocarse principalmente en las ideas, no en el *status* corporativo. Con el correo electrónico, una idea brillante de un estudiante becario puede tener más impacto que una idea tonta de un vicepresidente ejecutivo.

Una idea clave de todos los sistemas modernos de correo electrónico es la distinción entre el **sobre** y su contenido. El sobre encapsula el mensaje; contiene toda la información necesaria para transportar el mensaje, como dirección de destino, prioridad y nivel de seguridad, la cual es diferente del mensaje mismo. Los agentes de transporte del mensaje usan el sobre para enrutar, al igual que la oficina postal.

El mensaje dentro del sobre contiene dos partes: el **encabezado** y el **cuerpo.** El encabezado contiene información de control para los agentes de usuario. El cuerpo es por completo para el destinatario humano. Los sobres y mensajes se ilustran en la figura 7-7.

7.2.2 El agente de usuario

Los sistemas de correo electrónico tienen dos partes básicas, como hemos visto: los agentes de usuario y los agentes de transferencia de mensajes. En esta sección veremos a los agentes de usuario. Un agente de usuario normalmente es un programa (a veces llamado lector de correo) que acepta una variedad de comandos para redactar, recibir y contestar los mensajes, así como para manipular los buzones de correo. Algunos agentes de usuario tienen una interfaz elegante operada por menús o por iconos que requiere un ratón, mientras que otros esperan comandos de un carácter desde el teclado. Funcionalmente, ambos son iguales. Algunos sistemas son manejados por menús o por iconos, pero también tienen métodos abreviados de teclado.

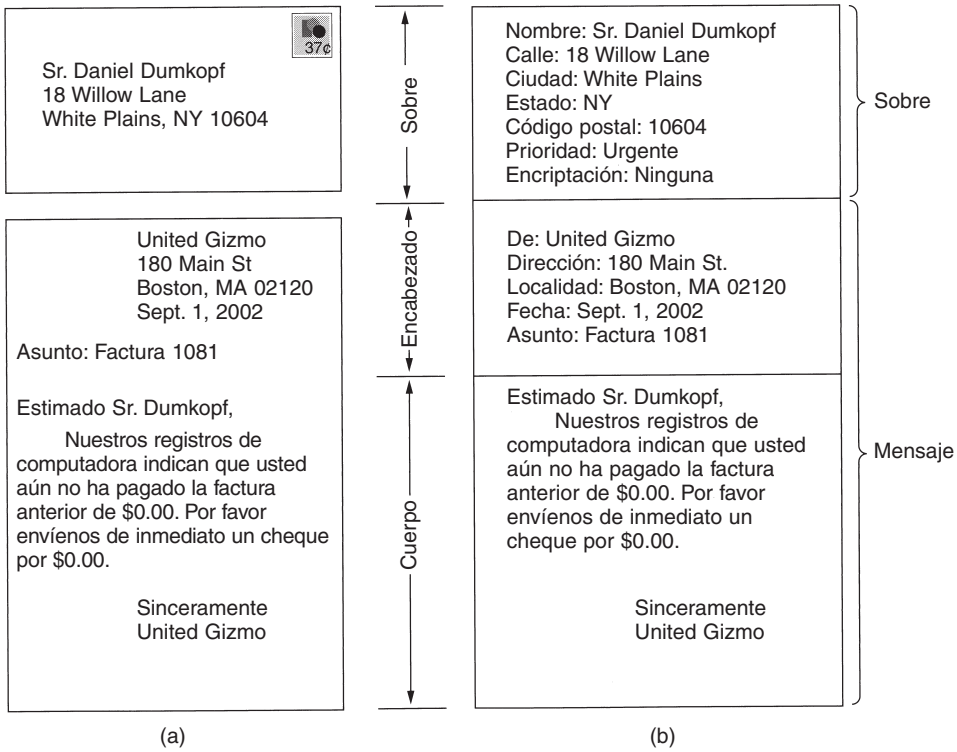


Figura 7-7. Sobres y mensajes. (a) Correo postal. (b) Correo electrónico.

Envío de correo electrónico

Para enviar un mensaje de correo electrónico, el usuario debe proporcionar el mensaje, la dirección de destino y, posiblemente, algunos otros parámetros. El mensaje puede producirse con un editor de texto independiente, un programa de procesamiento de texto o, posiblemente, un editor de texto incorporado en el agente de usuario. La dirección de destino debe estar en un formato que el agente de usuario pueda manejar. Muchos agentes de usuario esperan direcciones DNS de la forma *usuario@dirección-dns*. Dado que las hemos estudiado antes, no repetiremos ese material aquí.

Sin embargo, vale la pena indicar que existen otras formas de direccionamiento. En particular, las direcciones X.400 tienen un aspecto radicalmente diferente del de las direcciones DNS; se componen de pares *atributo* = *valor*, separadas por diagonales, por ejemplo,

/C=US/ST=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Esta dirección especifica un país, estado, localidad, dirección personal y nombre común (Ken Smith). Son posibles muchos otros atributos, de modo que puede enviarse correo electrónico a alguien cuyo nombre no se conoce, siempre y cuando se conozca un número suficiente de los otros

atributos (por ejemplo, compañía y puesto). Aunque los nombres X.400 son considerablemente menos convenientes que los DNS, la mayoría de los sistemas de correo electrónico tienen alias (algunas veces llamados sobrenombres) que permiten que los usuarios introduzcan o seleccionen un nombre de una persona y obtengan la dirección de correo electrónico correcta. En consecuencia, incluso con direcciones X.400, por lo general no es necesario teclear realmente estas extrañas cadenas.

La mayoría de los sistemas de correo electrónico manejan listas de correo, por lo que un usuario puede enviar el mismo mensaje a un grupo de personas con un solo comando. Si la lista de correo se mantiene localmente, el agente usuario puede sencillamente enviar un mensaje por separado a cada destinatario. Sin embargo, si la lista se mantiene de manera remota, los mensajes se multiplicarán ahí. Por ejemplo, si un grupo de observadores de pájaros tiene una lista de correo llamada *birders* instalada en *meadowlark.arizona.edu,* entonces cualquier mensaje enviado a *birders-@meadowlark.arizona.edu* se enrutará a la Universidad de Arizona y se multiplicará ahí para producir mensajes individuales para todos los miembros de la lista de correo, sin importar el lugar del mundo en el que estén. Los usuarios de esta lista de correo no pueden saber que es una lista de correo. Podría ser simplemente el buzón de correo personal del profesor Gabriel O. Birders.

Lectura del correo electrónico

Por lo común, cuando se inicia un agente de usuario, buscará en el buzón del usuario el correo electrónico recibido, antes de presentar otra cosa en la pantalla. Después puede anunciar la cantidad de mensajes en el buzón o presentar un resumen de una línea de cada uno y esperar un comando.

Como ejemplo del funcionamiento de un agente de usuario, veamos una situación típica de correo. Después de iniciar el agente de usuario, el usuario solicita un resumen de su correo electrónico. A continuación aparece una pantalla como la de la figura 7-8. Cada línea hace referencia a un mensaje. En este ejemplo, el buzón contiene ocho mensajes.

#	Banderas	Bytes	Remitente	Asunto
1	K	1030	asw	Cambios a MINIX
2	KA	6348	trudy	No todas las Trudies son molestas
3	K F	4519	Amy N. Wong	Solicitud de información
4		1236	bal	Bioinformática
5		104110	kaashoek	Material sobre las redes de igual a igual
6		1223	Frank	Re: Revisará una propuesta de otorgamiento
7		3110	guido	Nuestro documento ha sido aceptado
8		1204	dmr	Re: Visita de mis estudiantes

Figura 7-8. Ejemplo de exhibición del contenido de un buzón.

Cada línea de la pantalla contiene varios campos extraídos del sobre o del encabezado del mensaje correspondiente. En un sistema sencillo de correo electrónico, la selección de campos a presentar está incorporada en el programa. En uno más refinado, el usuario puede especificar los campos a presentar proporcionando un **perfil de usuario,** un archivo que describe el formato de

presentación. En este ejemplo, el primer campo es el número de mensaje. El segundo campo, *Banderas*, puede contener una *K*, lo que significa que el mensaje no es nuevo, sino que ha sido leído previamente y guardado en el buzón; una *A*, lo que quiere decir que el mensaje ya ha sido contestado; y/o una *F*, lo que indica que ese mensaje ha sido reenviado a alguien más. También son posibles otras banderas.

El tercer campo indica la longitud del mensaje y el cuarto indica quién envió el mensaje. Puesto que este campo simplemente se extrae del mensaje, puede contener nombres de pila, nombres completos, iniciales, claves de acceso o cualquier otra cosa que el remitente decida poner ahí. Por último, el campo de *Asunto* da un resumen breve del tema del mensaje. La gente que no incluye un campo de *Asunto* con frecuencia descubre que las respuestas a su correo electrónico tienden a no recibir la prioridad más alta.

Una vez que se han visualizado los encabezados, el usuario puede realizar cualquiera de diversas acciones, como desplegar o eliminar el mensaje. Los sistemas más antiguos se basaban en texto y típicamente utilizaban comandos de un carácter para realizar estas tareas, como T (teclear mensaje), A (responder a mensaje), D (borrar mensaje) y F (reenviar mensaje). Un argumento especificaba el mensaje en cuestión. Los sistemas más recientes utilizan interfaces gráficas. Por lo general, el usuario selecciona un mensaje con el ratón y después hace clic en un icono para escribir, responder a él, eliminarlo o reenviarlo.

El correo electrónico ha recorrido un largo camino desde los días cuando era simplemente transferencia de archivos. Los agentes de usuario refinados hacen posible manejar un gran volumen de correo electrónico. Para las personas que reciben y envían miles de mensajes de correo electrónico al año, tales herramientas son invaluables.

7.2.3 Formatos de mensaje

Pasemos ahora de la interfaz de usuario al formato de los mensajes de correo electrónico mismos. Primero veremos el correo electrónico ASCII básico usando el RFC 822. Después de eso, veremos las extensiones multimedia del RFC 822.

RFC 822

Los mensajes consisten en un sobre primitivo (descrito en el RFC 821), algunos campos de encabezado, una línea en blanco y el cuerpo del mensaje. Cada campo de encabezado consiste (lógicamente) en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos (:) y, para la mayoría de los campos, un valor. El RFC 822 es un estándar viejo y no distingue claramente los campos del sobre de los de encabezado. Aunque se revisó en el RFC 2822, no fue posible restaurarlo por completo debido a su uso amplio. En el uso normal, el agente de usuario construye un mensaje y lo pasa al agente de transferencia de mensajes, quien después usa algunos campos del encabezado para construir el sobre, una mezcla un tanto anticuada de mensaje y sobre.

En la figura 7-9 se listan los principales campos de encabezado relacionados con el transporte del mensaje. El campo *To:* indica la dirección DNS del destinatario primario. Está permitido tener

varios destinatarios. El campo *Cc:* indica la dirección de los destinatarios secundarios. En términos de entrega, no hay diferencia entre los destinatarios primarios y los secundarios. Es una diferencia por entero psicológica que puede ser importante para los participantes, pero que no lo es para el sistema de correo. El término *Cc:* (copia al carbón) es un poco anticuado, puesto que las computadoras no usan papel carbón, pero está muy establecido. El campo *Bcc:* (copia oculta) es como el campo *Cc:*, excepto que esta línea se borra de todas las copias enviadas a los destinatarios primarios y secundarios. Esta característica permite a la gente mandar copias a terceros sin que los destinatarios primarios y secundarios lo sepan.

Encabezado	Significado
To:	Direcciones de correo electrónico de los destinatarios primarios
Cc:	Direcciones de correo electrónico de los destinatarios secundarios
Bcc:	Direcciones de correo electrónico para las copias ocultas
From:	Persona o personas que crearon el mensaje
Sender:	Dirección de correo electrónico del remitente
Received:	Línea agregada por cada agente de transferencia en la ruta
Return-Path:	Puede usarse para identificar una ruta de regreso al remitente

Figura 7-9. Campos de encabezado RFC 822 relacionados con el transporte de mensajes.

From: y *Sender:* indican quién escribió y envió el mensaje, respectivamente; pueden ser diferentes. Por ejemplo, un ejecutivo de negocios puede escribir un mensaje, pero su secretaria podría ser la que lo transmita. En este caso, el ejecutivo estaría listado en el campo *From:* y la secretaria en el campo *Sender:*. El campo *From:* es necesario, pero el campo *Sender:* puede omitirse si es igual al campo *From:*. Estos campos son necesarios en caso de que el mensaje no pueda entregarse y deba devolverse al remitente.

Se agrega una línea que contiene *Received:* por cada agente de transferencia de mensajes en el camino. La línea contiene la identidad del agente, la fecha y hora de recepción del mensaje, y otra información que puede servir para encontrar fallas en el sistema de enrutamiento.

El campo *Return-Path:* es agregado por el agente de transferencia de mensajes final y estaba destinado para indicar la manera de regresar al remitente. En teoría, esta información puede tomarse de todos los encabezados *Received:* (con excepción del nombre del buzón del remitente), pero pocas veces se llena de esta manera y por lo general contiene sólo la dirección del remitente.

Además de los campos de la figura 7-9, los mensajes RFC 822 también pueden contener una variedad de campos de encabezado usados por los agentes de usuario o los destinatarios. En la figura 7-10 se listan los más comunes. La mayoría de ellos autoexplicativos, por lo que no los veremos en detalle.

El campo *Reply-To:* a veces se usa cuando ni la persona que redactó el mensaje ni la que lo envió quieren ver la respuesta. Por ejemplo, un gerente de *marketing* escribe un mensaje de correo electrónico para informar a los clientes sobre un producto nuevo. El mensaje lo envía una

Encabezado	Significado
Date:	Fecha y hora de envío del mensaje
Reply-To:	Dirección de corre electrónico a la que deben enviarse las contestaciones
Message-Id:	Número único para referencia posterior a este mensaje
In-Reply-To:	Identificador del mensaje al que éste responde
References:	Otros identificadores de mensaje pertinentes
Keywords:	Claves seleccionadas por el usuario
Subject:	Resumen corto del mensaje para desplegar en una línea

Figura 7-10. Algunos campos usados en el encabezado de mensaje RFC 822.

secretaria, pero el campo *Reply-To:* indica el jefe del departamento de ventas, quien puede contestar preguntas y surtir pedidos. Este campo también es útil cuando el emisor tiene dos cuentas de correo electrónico y desea que la respuesta llegue a su otra cuenta.

El documento RFC 822 indica explícitamente que los usuarios pueden inventar encabezados nuevos para uso privado, siempre y cuando comiencen con la cadena *X-*. Se garantiza que no habrá encabezados futuros que usen nombres que comiencen con *X-*, a fin de evitar conflictos entre los encabezados oficiales y los privados. A veces algunos estudiantes universitarios, pasándose de listos incluyen campos como *X-Fruta-del-Día:* o *X-Enfermedad-de-la-Semana:,* que son legales, aunque no muy ilustrativos.

Tras los encabezados viene el cuerpo del mensaje. Los usuarios pueden poner aquí lo que les venga en gana. Algunos terminan sus mensajes con firmas complicadas, incluidas caricaturas sencillas en ASCII, citas de autoridades mayores y menores, pronunciamientos políticos y renuncias de responsabilidades de todo tipo (por ejemplo, la corporación ABC no es responsable de mis opiniones; ni siquiera las puede entender).

MIME—Extensiones Multipropósito de Correo Internet

En los primeros días de ARPANET, el correo electrónico consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En tal entorno, el RFC 822 hacía todo el trabajo: especificaba los encabezados, pero dejaba el contenido en manos del usuario. Hoy día, en la red mundial de Internet, este método ya no es adecuado. Los problemas incluyen envío y recepción de:

1. Mensajes en idiomas con acentos (por ejemplo, español, francés y alemán).
2. Mensajes en alfabetos no latinos (por ejemplo, hebreo y ruso).
3. Mensajes en idiomas sin alfabetos (por ejemplo, chino y japonés).
4. Mensajes que no contienen texto (por ejemplo, audio y vídeo).

Se propuso una solución en el RFC 1341 y se actualizó en los RFCs 2045-2049. Esta solución, llamada **MIME (Extensiones Multipropósito de Correo Internet),** se usa ampliamente. A continuación la describiremos. Para información adicional sobre MIME, vea dichos RFCs.

La idea básica de MIME es continuar usando el formato RFC 822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. Al no desviarse del 822, los mensajes MIME pueden enviarse usando los programas y protocolos de correo electrónico existentes. Todo lo que tiene que cambiarse son los programas emisores y receptores, lo que pueden hacer los usuarios mismos.

MIME define cinco nuevos encabezados de mensaje, como se muestra en la figura 7-11. El primero de éstos simplemente indica al agente de usuario receptor del mensaje que está tratando con un mensaje MIME, así como la versión del MIME que usa. Se considera que cualquier mensaje que no contenga un encabezado *MIME-Version:* es un mensaje de texto normal en inglés, y se procesa como tal.

Encabezado	Significado
MIME-Version:	Identifica la versión de MIME
Content-Description:	Cadena de texto que describe el contenido
Content-Id:	Identificador único
Content-Transfer-Encoding:	Cómo se envuelve el mensaje para su transmisión
Content-Type:	Naturaleza del mensaje

Figura 7-11. Encabezados RFC 822 agregados por MIME.

El encabezado *Content-Description:* es una cadena ASCII que dice lo que está en el mensaje. Este encabezado es necesario para que el destinatario sepa si vale la pena descodificar y leer el mensaje. Si la cadena dice: "Foto del jerbo de Bárbara" y la persona que recibe el mensaje no es un gran fanático de los jerbos, el mensaje probablemente será descartado en lugar de descodificado para dar una foto a color de alta definición.

El encabezado *Content-Id:* identifica el contenido; usa el mismo formato que el encabezado estándar *Message-Id:*.

Content-Transfer-Encoding: indica la manera en que está envuelto el cuerpo para su transmisión a través de una red donde se podría tener problemas con la mayoría de los caracteres distintos de las letras, números y signos de puntuación. Se proporcionan cinco esquemas (más un escape hacia nuevos esquemas). El esquema más sencillo es simplemente texto ASCII. Los caracteres ASCII usan 7 bits y pueden transportarse directamente mediante el protocolo de correo electrónico siempre y cuando ninguna línea exceda 1000 caracteres.

El siguiente esquema más sencillo es lo mismo, pero utiliza caracteres de 8 bits, es decir, todos los valores de 0 a 255. Este esquema de codificación viola el protocolo (original) del correo electrónico de Internet, pero es usado por algunas partes de Internet que implementan ciertas extensiones al protocolo original. Mientras que la declaración de la codificación no la hace legal,

hacerla explícita puede cuando menos explicar las cosas cuando algo sucede mal. Los mensajes que usan codificación de 8 bits deben aún adherirse a la longitud máxima de línea estándar.

Peor aún son los mensajes que usan codificación binaria. Éstos son archivos binarios arbitrarios que no sólo utilizan los 8 bits, sino que ni siquiera respetan el límite de 1000 caracteres por línea. Los programas ejecutables caen en esta categoría. No se da ninguna garantía de que los mensajes en binario llegarán correctamente, pero mucha gente los envía de todos modos.

La manera correcta de codificar mensajes binarios es usar **codificación base64,** a veces llamada **armadura ASCII.** En este esquema se dividen grupos de 24 bits en unidades de 6 bits, y cada unidad se envía como un carácter ASCII legal. La codificación es "A" para 0, "B" para 1, etcétera, seguidas por las 26 letras minúsculas, los 10 dígitos y, por último, + y / para el 62 y 63, respectivamente. Las secuencias == e = se usan para indicar que el último grupo contenía sólo 8 o 16 bits, respectivamente. Los retornos de carro y avances de línea se ignoran, por lo que puede introducirse a voluntad para mantener la línea lo suficientemente corta. Puede enviarse un texto binario arbitrario usando este esquema.

En el caso de mensajes que son casi completamente ASCII, pero con algunos caracteres no ASCII, la codificación base 64 es algo ineficiente. En su lugar se usa una codificación conocida como **codificación entrecomillada imprimible.** Simplemente es ASCII de 7 bits, con todos los caracteres por encima de 127 codificados como un signo de igual seguido del valor del carácter en dos dígitos hexadecimales.

En resumen, los datos binarios deben enviarse codificados en base 64 o en forma entrecomillada imprimible. Cuando hay razones válidas para no usar uno de estos esquemas, es posible especificar una codificación definida por el usuario en el encabezado *Content-Transfer-Encoding:*.

El último encabezado mostrado en la figura 7-11 en realidad es el más interesante. Especifica la naturaleza del cuerpo del mensaje. En el RFC 2045 hay siete tipos definidos, cada uno de los cuales tiene uno o más subtipos. El tipo y el subtipo se separan mediante una diagonal, como en

```
Content-Type: video/mpeg
```

El subtipo debe indicarse de manera explícita en el encabezado; no se proporcionan valores predeterminados. La lista inicial de tipos y subtipos especificada en el RFC 2045 se da en la figura 7-12. Se han agregado muchos otros desde entonces, y se agregan entradas nuevas todo el tiempo, a medida que surge la necesidad.

Veamos brevemente la lista de tipos. El tipo *text* es para texto normal. La combinación *text/plain* es para mensajes ordinarios que pueden visualizarse como se reciben, sin codificación ni ningún procesamiento posterior. Esta opción permite el transporte de mensajes ordinarios en MIME con sólo unos pocos encabezados extra.

El subtipo *text/enriched* permite la inclusión de un lenguaje de marcado sencillo en el texto. Este lenguaje proporciona una manera independiente del sistema para indicar negritas, cursivas, tamaños en puntos menores y mayores, sangrías, justificaciones, sub y superíndices, y formatos sencillos de página. El lenguaje de marcado se basa en SGML, el lenguaje estándar genérico de

Tipo	Subtipo	Descripción
Texto	Plano	Texto sin formato
	Enriquecido	Texto con comandos de formato sencillos
Imagen	Gif	Imagen fija en formato GIF
	Jpeg	Imagen fija en formato JPEG
Audio	Básico	Sonido
Vídeo	Mpeg	Película en formato MPEG
Aplicación	Octet-stream	Secuencia de bytes no interpretada
	Postscript	Documento imprimible en PostScript
Mensaje	Rfc822	Mensaje MIME RFC 822
	Parcial	Mensaje dividido para su transmisión
	Externo	El mensaje mismo debe obtenerse de la red
Multipartes	Mezclado	Partes independientes en el orden especificado
	Alternativa	Mismo mensaje en diferentes formatos
	Paralelo	Las partes deben verse en forma simultánea
	Compendio	Cada parte es un mensaje RFC 822 completo

Figura 7-12. Tipos y subtipos MIME definidos en el RFC 2045.

etiquetas que también se utiliza como la base del HTML de World Wide Web. Por ejemplo, el mensaje

Ha llegado el <bold> momento </bold>, dijo la <italic> morsa </italic>...

se presentaría como

Ha llegado el **momento,** dijo la *morsa*...

Es responsabilidad del sistema receptor seleccionar la presentación adecuada. Si hay negritas y cursivas disponibles, pueden usarse; de otra manera, pueden usarse colores, parpadeos, vídeo inverso, etcétera, como énfasis. Los diferentes sistemas pueden hacer selecciones distintas, lo cual hacen en realidad.

Cuando Web se volvió popular, se agregó un nuevo subtipo *texto/html* (en el RFC 2854) para permitir que las páginas Web se enviaran en el correo electrónico del RFC 822. En el RFC 3023 se define un subtipo para el lenguaje de marcado extensible, *texto/xml*. Analizaremos HTML y XML más adelante en este capítulo.

El siguiente tipo MIME es *imagen*, que se usa para transmitir imágenes fijas. Hoy día se usan muchos formatos para almacenar y transmitir imágenes, tanto con compresión como sin ella. Dos de éstos, GIF y JPEG, están integrados en la mayoría de los navegadores, pero también existen muchos otros, los cuales se han agregado a la lista original.

Los tipos *audio* y *vídeo* son para sonido e imágenes en movimiento, respectivamente. Observe que *vídeo* incluye sólo la información visual, no la pista de sonido. Si se debe transmitir una película con sonido, tal vez sea necesario transmitir las partes de vídeo y de audio por separado, dependiendo del sistema de codificación usado. El primer formato de vídeo definido fue el diseñado por el grupo de modesto nombre MPEG (Grupo de Expertos en Imágenes en Movimiento), pero desde entonces se han agregado otros. Además de *audio/basic*, se agregó un nuevo tipo de audio, *audio/mpeg*, en el RFC 3003 para permitir que las personas pudieran enviar archivos de audio MP3.

El tipo *aplicación* es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos. Un *octet-stream* simplemente es una secuencia de bytes no interpretados. Al recibir una de tales cadenas, un agente de usuario probablemente debería presentarla en pantalla sugiriendo al usuario que lo copie en un archivo y solicitándole un nombre de archivo. El procesamiento posterior es responsabilidad del usuario.

Otro subtipo definido es *postscript,* que se refiere al lenguaje PostScript producido por Adobe Systems y ampliamente usado para describir páginas impresas. Muchas impresoras tienen intérpretes PostScript integrados. Aunque un agente de usuario simplemente puede llamar a un intérprete PostScript externo para visualizar los archivos PostScript entrantes, hacerlo no está exento de riesgos. PostScript es un lenguaje de programación completo. Con el tiempo necesario, una persona lo bastante masoquista podría escribir un compilador de C o un sistema de administración de base de datos en PostScript. El despliegue de un mensaje PostScript entrante puede hacerse ejecutando el programa PostScript contenido en él. Además de desplegar algo de texto, este programa puede leer, modificar o borrar los archivos del usuario, así como tener otros efectos secundarios desagradables.

El tipo *mensaje* permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, por ejemplo, correo electrónico. Cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior, debe usarse el subtipo *rfc822.*

El subtipo *parcial* hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado (por ejemplo, si el mensaje encapsulado es demasiado grande). Los parámetros hacen posible reensamblar en forma correcta todas las partes en el destino en el orden correcto.

Por último, el subtipo *externo* puede usarse para mensajes muy grandes (por ejemplo, películas en vídeo). En lugar de incluir el archivo MPEG en el mensaje, se da una dirección FTP y el agente de usuario del receptor puede obtenerlo a través de la red al momento que se requiera. Esta característica es especialmente útil cuando se envía una película a una lista de correo, y sólo se espera que unos cuantos la vean (imagine correo electrónico chatarra que contenga vídeos de propaganda).

El último tipo es *multiparte*, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados. El subtipo *mezclado* permite que cada parte sea diferente, sin ninguna estructura adicional impuesta. Muchos programas de correo electrónico permiten que el usuario agregue uno o más archivos adjuntos a un mensaje de texto. Estos archivos adjuntos se envían mediante el tipo *multiparte*.

A diferencia del tipo *multiparte*, el subtipo *alternativa* permite que el mismo mensaje se incluya varias veces, pero expresado en dos o más medios diferentes. Por ejemplo, un mensaje puede enviarse como ASCII común, como texto enriquecido (*enriched*) y como PostScript. Un agente de usuario adecuadamente diseñado que reciba uno de tales mensajes lo presentará en PostScript de ser posible. La segunda posibilidad sería como texto enriquecido. Si ninguna de las dos fuera posible, se presentaría el texto ASCII normal. Las partes deben ordenarse de la más sencilla a la más compleja para ayudar a que los receptores con agentes de usuario pre-MIME puedan encontrarle algún sentido al mensaje (por ejemplo, incluso un usuario pre-MIME puede leer texto ASCII común).

El subtipo *alternativa* también puede servir para varios lenguajes. En este contexto, puede pensarse en la piedra Rosetta como uno de los primeros mensajes *multiparte/alternativa.*

En la figura 7-13 se muestra un ejemplo multimedia. Aquí se transmiten una felicitación de cumpleaños como texto y como canción. Si el receptor tiene capacidad de audio, el agente de usuario traerá el archivo de sonido, *birthday.snd,* y lo ejecutará. Si no, se presenta la letra en la pantalla en absoluto silencio. Las partes están delimitadas por dos guiones seguidos de una cadena (generada por software) especificada en el parámetro *boundary.*

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: La Tierra da vuelta al Sol un número entero de veces

Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched

Feliz cumpleaños a ti
Feliz cumpleaños a ti
Feliz cumpleaños, querida <bold> Carolyn </bold>
Feliz cumpleaños a ti

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
      access-type="anon-ftp";
      site="bicycle.abcd.com";
      directory="pub";
      name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Figura 7-13. Mensaje multiparte que contiene alternativas de texto enriquecido y audio.

Observe que el encabezado *Content-Type* aparece en tres lugares en este ejemplo. En el nivel superior, el encabezado indica que el mensaje tiene varias partes. En cada parte, el encabezado indica el tipo y el subtipo de la parte. Por último, en el texto de la segunda parte, el encabezado es necesario para indicarle al agente de usuario la clase de archivo externo que debe conseguir. Para indicar esta pequeña diferencia de uso, hemos usado letras en minúscula, aunque los encabezados no hacen distinciones entre mayúsculas y minúsculas. De la misma manera, se requiere el encabezado *content-transfer-encoding* para cualquier cuerpo externo que no esté codificado como ASCII de 7 bits.

Regresando a los subtipos para mensajes multiparte, existen dos posibilidades más. El subtipo *paralelo* se usa cuando todas las partes deben "verse" de manera simultánea. Por ejemplo, las películas con frecuencia tienen un canal de audio y uno de vídeo. Las películas son más efectivas si estos dos canales se producen en paralelo, en lugar de consecutivamente.

Por último, el subtipo *compendio* se usa cuando se empacan muchos mensajes juntos en un mensaje compuesto. Por ejemplo, algunos grupos de discusión de Internet recolectan mensajes de los subscriptores y luego los envían como un solo mensaje *multiparte/compendio*.

7.2.4 Transferencia de mensajes

El sistema de transferencia de mensajes se ocupa de transmitir del remitente al destinatario. La manera más sencilla de hacer esto es establecer una conexión de transporte de la máquina de origen a la de destino y sencillamente transferir el mensaje. Después de examinar la manera en que se hace normalmente esto, estudiaremos algunas situaciones en las que no funciona esto, y lo que puede hacerse al respecto.

SMTP—Protocolo Simple de Transporte de Correo

En Internet, el correo electrónico se entrega al hacer que la máquina de origen establezca una conexión TCP con el puerto 25 de la máquina de destino. Escuchando en este puerto está un demonio de correo electrónico que habla con el **SMTP** (**Protocolo Simple de Transporte de Correo**)**.** Este demonio acepta conexiones de entrada y copia mensajes de ellas a los buzones adecuados. Si no puede entregarse un mensaje, se devuelve al remitente un informe de error que contiene la primera parte del mensaje que no pudo entregarse.

SMTP es un protocolo ASCII sencillo. Después de establecer la conexión TCP con el puerto 25, la máquina emisora, operando como cliente, espera que la máquina receptora, operando como servidor, hable primero. El servidor comienza enviando una línea de texto que proporciona su identidad e indica si está preparado o no para recibir correo. Si no lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar correo electrónico, el cliente anuncia de quién proviene el mensaje, y a quién está dirigido. Si existe el destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. A continuación el cliente envía el mensaje y el servidor confirma su recepción. Por lo general, no se requieren sumas de verificación porque TCP proporciona un flujo de bytes confiable. Si hay más correo electrónico, se envía ahora. Una vez que todo el correo electrónico ha sido intercambiado en ambas direcciones, se libera la conexión. En la

figura 7-14 se muestra un diálogo de ejemplo para el envío del mensaje de la figura 7-13, incluidos los códigos numéricos usados por SMTP. Las líneas enviadas por el cliente se marcan con *C:*; aquellas enviadas por el servidor se marcan con *S:*.

Pueden ser útiles algunos comentarios sobre la figura 7-14. El primer comando del cliente es ciertamente *HELO.* De las dos abreviaturas de cuatro caracteres de *HELLO,* ésta tiene numerosas ventajas sobre su competidora. La razón por la que los comandos tienen que ser de cuatro caracteres se ha perdido en las brumas del tiempo.

En la figura 7-14 el mensaje se envía a un solo destinatario, por lo que se usa un solo comando *RCPT*. Se permiten tales comandos para enviar un solo mensaje a destinatarios múltiples; se confirma la recepción de cada uno o se rechaza de manera individual. Incluso si se rechazan algunos destinatarios (porque no existen en el destino), el mensaje puede enviarse a los demás.

Por último, aunque la sintaxis de los comandos de cuatro caracteres del cliente se especifica con rigidez, la sintaxis de las respuestas es menos rígida. Sólo cuenta el código numérico. Cada implementación puede poner la cadena que desee después del código.

Para obtener una mejor idea de cómo funcionan SMTP y algunos otros protocolos descritos en este capítulo, pruébelos. En todos los casos, primero vaya a una máquina conectada a Internet. En un sistema UNIX, en una línea de comando, escriba:

```
telnet mail.isp.com 25
```

y sustituya el nombre DNS de su servidor de correo ISP por *mail.isp.com*. En un sistema Windows, haga clic en Inicio | Ejecutar (Start | Run) y después escriba el comando en el cuadro de diálogo. Este comando establecerá una conexión telnet (es decir, TCP) con el puerto 25 de esa máquina. El puerto 25 es el puerto SMTP (vea la figura 6-27 en la que se listan algunos puertos comunes). Probablemente obtendrá una respuesta parecida a lo siguiente:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Las primeras tres líneas son de telnet y le indican lo que están haciendo. La última línea es del servidor SMTP de la máquina remota y anuncia su disponibilidad de hablar con usted y aceptar correo electrónico. Para ver cuáles comandos acepta, escriba

```
HELP
```

De aquí en adelante, es posible una secuencia de comandos como la que se muestra en la figura 7-14, a partir del comando *HELO* del cliente.

Vale la pena mencionar que no es un accidente el uso de líneas de texto ASCII para los comandos. La mayoría de los protocolos de Internet funcionan de esta manera. El uso de texto ASCII hace que los protocolos sean fáciles de probar y depurar. Pueden probarse emitiendo comandos de manera manual, como vimos anteriormente, y las copias de los datos a pantalla o impresora son fáciles de leer.

```
              S: 220 servicio SMTP xyz.com listo
C: HELO abcd.com
              S: 250 xyz.com dice hola a abcd.com
C: MAIL FROM:<elinor@abcd.com>
              S: 250 emisor ok
C: RCPT TO:<carolyn@xyz.com>
              S: 250 receptor ok
C: DATA
              S: 354 envía correo; termina con una línea únicamente con "."
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: La Tierra da vuelta al Sol un número entero de veces
C:
C: Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Feliz cumpleaños a ti
C: Feliz cumpleaños a ti
C: Feliz cumpleaños, querida <bold> Carolyn </bold>
C: Feliz cumpleaños a ti
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:            access-type="anon-ftp";
C:            site="bicycle.abcd.com";
C:            directory="pub";
C:            name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
              S: 250 mensaje aceptado
C: QUIT
              S: 221 xyz.com cerrando conexión
```

Figura 7-14. Transferencia de un mensaje de *elinor@abc.com* a *carolyn@xyz.com*.

Aunque el protocolo SMTP está bien definido, pueden surgir algunos problemas. Uno se relaciona con la longitud del mensaje. Algunas implementaciones más viejas no pueden manejar mensajes mayores que 64 KB. Otro problema se relaciona con las terminaciones de temporizador. Si el cliente y el servidor tienen temporizadores diferentes, uno de ellos puede terminar mientras

que el otro continúa trabajando, terminando en forma inesperada la conexión. Por último, en contadas ocasiones, pueden dispararse tormentas de correo infinitas. Por ejemplo, si el *host* 1 contiene la lista de correo *A* y el *host* 2 contiene la lista de correo *B* y cada lista contiene una entrada para la otra lista, entonces un mensaje enviado a cualquiera de las listas generará una cantidad sin fin de tráfico de correo electrónico, a menos que alguien lo verifique.

Para superar algunos de estos problemas, se ha definido el SMTP extendido (**ESMTP**) en el RFC 2821. Los clientes que deseen usarlo deben enviar inicialmente un mensaje *EHLO*, en lugar de *HELO*. Si el saludo se rechaza, esto indica que el servidor es un servidor SMTP normal, y el cliente debe proceder de la manera normal. Si se acepta el *EHLO*, entonces se permiten los comandos y parámetros nuevos.

7.2.5 Entrega final

Hasta ahora hemos supuesto que todos los usuarios trabajan en máquinas capaces de enviar y recibir correo electrónico. Como vimos, el correo electrónico se entrega al hacer que el emisor establezca una conexión TCP con el receptor y después que envíe el correo electrónico a través de ella. Este modelo funcionó bien por décadas cuando todos los *hosts* ARPANET (y más tarde Internet) se pusieron, de hecho, en línea todo el tiempo para aceptar conexiones TCP.

Sin embargo, con el advenimiento de personas que acceden a Internet llamando a su ISP por medio de un módem, ese modelo dejó de usarse. El problema es el siguiente: ¿qué sucede cuando Elena desea enviar correo electrónico a Carolina y esta última no está en línea en ese momento? Elena no puede establecer una conexión TCP con Carolina y, por lo tanto, no puede ejecutar el protocolo SMTP.

Una solución es que un agente de transferencia de mensajes en una máquina ISP acepte correo electrónico para sus clientes y lo almacene en sus buzones en una máquina ISP. Puesto que este agente puede estar en línea todo el tiempo, el correo electrónico puede enviarse las 24 horas del día.

POP3

Desgraciadamente, esta solución genera otro problema: ¿cómo obtiene el usuario el correo electrónico del agente de transferencia de mensajes del ISP? La solución a este problema es crear otro protocolo que permita que los agentes de transferencia de usuarios (en PCs cliente) contacten al agente de transferencia de mensajes (en la máquina del ISP) y que el correo electrónico se copie desde el ISP al usuario. Tal protocolo es **POP3** (**Protocolo de Oficina de Correos Versión 3**), que se describe en el RFC 1939.

En la figura 7-15(a) se presenta la situación más común (en la que tanto el emisor como el receptor tienen una conexión permanente a Internet). En la figura 7-15(b) se ilustra la situación en la que el emisor está (actualmente) en línea pero el receptor no.

POP3 inicia cuando el usuario arranca el lector de correo. Éste llama al ISP (a menos que ya haya una conexión) y establece una conexión TCP con el agente de transferencia de mensajes en

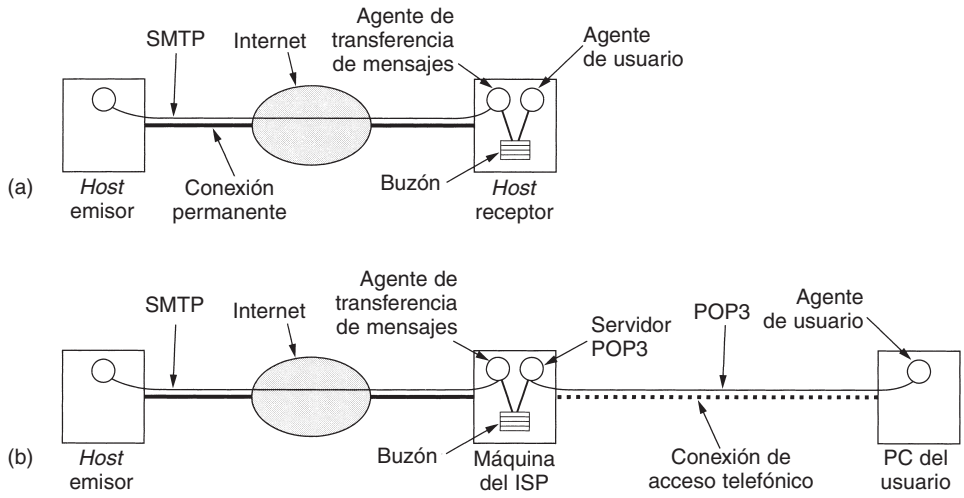


Figura 7-15. (a) Envío y recepción de correo cuando el receptor tiene una conexión permanente a Internet y el agente de usuario se ejecuta en la misma máquina que el agente de transferencia de mensajes. (b) Lectura de correo cuando el receptor tiene una conexión de acceso telefónico a un ISP.

el puerto 110. Una vez que se ha establecido la conexión, el protocolo POP3 pasa por tres estados en secuencia:

1. Autorización.
2. Transacciones.
3. Actualización.

El estado de autorización tiene que ver con el inicio de sesión por parte del usuario. El estado de transacción se relaciona con el hecho de que el usuario colecte los mensajes de correo electrónico y los marque para eliminación desde el buzón. El estado de actualización se encarga de que los mensajes de correo electrónico se eliminen realmente.

Este comportamiento se puede observar tecleando algo como lo siguiente:

```
telnet mail.isp.com 110
```

donde *mail.isp.com* representa el nombre DNS del servidor de correo de su ISP. Telnet establece una conexión TCP con el puerto 110, en el que escucha el servidor POP3. Al aceptar la conexión TCP, el servidor envía un mensaje ASCII que indica que está presente. Por lo general, este mensaje comienza con +OK seguido de un comentario. En la figura 7-16 se muestra un escenario de ejemplo, comenzando después de que se ha establecido la conexión TCP. Al igual que antes, las líneas marcadas con *C:* provienen del cliente (usuario) y las marcadas con *S:* provienen del servidor (agente de transferencia de mensajes en la máquina del ISP).

```
                S: +OK servidor POP3 listo
C: USER carolina
                S: +OK
C: PASS vegetales
                S: +OK inicio de sesión exitoso

C: LIST
                S: 1 2505
                S: 2 14302
                S: 3 8122
                S: .
C: RETR 1
                S: (envía mensaje 1)
C: DELE 1
C: RETR 2
                S: (envía mensaje 2)

C: DELE 2
C: RETR 3
                S: (envía mensaje 3)
C: DELE 3
C: QUIT
                S: +OK servidor POP3 desconectándose
```

Figura 7-16. Uso de POP3 para obtener tres mensajes.

Durante el estado de autorización, el cliente envía su nombre de usuario y después su contraseña. Después de un inicio de sesión exitoso, el cliente puede enviar el comando *LIST*, que causa que el servidor liste el contenido del buzón, un mensaje por línea, y que dé la longitud de ese mensaje. La lista se termina mediante un punto.

A continuación el cliente puede recuperar los mensajes utilizando el comando *RETR* y los puede marcar para eliminarlos con *DELE*. Cuando todos los mensajes se han recuperado (y posiblemente se han marcado para eliminación), el cliente emite el comando *QUIT* para terminar el estado de transacción y entrar al de actualización. Cuando el servidor ha eliminado todos los mensajes, envía una respuesta y termina la conexión TCP.

Si bien es cierto que el protocolo POP3 soporta la capacidad de descargar un mensaje específico o un conjunto de mensajes y dejarlos en el servidor, la mayoría de los programas de correo electrónico simplemente descarga todo y vacía el buzón. Este comportamiento significa que en la práctica, la única copia está en el disco duro del usuario. Si se cae, es posible que se pierda de manera permanente todo el correo electrónico.

Resumamos brevemente la forma en que funciona el correo electrónico para los clientes ISP. Elena utiliza un programa de correo electrónico (es decir, un agente de usuario) para crear un mensaje para Carolina y hace clic en un icono para enviarlo. El programa de correo electrónico entrega el mensaje al agente de transferencia de mensajes del *host* de Elena. Dicho agente ve que el mensaje está dirigido a *carolyn@xyz.com* por lo que utiliza DNS para buscar el registro *MX* de *xyz.com* (donde *xyz.com* es el ISP de Carolina). Esta consulta regresa el nombre del DNS del servidor de correo de *xyz.com*. El agente de transferencia de mensajes ahora busca la dirección IP de

esta máquina utilizando nuevamente DNS, por ejemplo, *gethostbyname*. Después establece una conexión TCP con el servidor SMTP en el puerto 25 de esta máquina. A continuación utiliza una secuencia de comandos SMTP análoga a la de la figura 7-14 para transferir el mensaje al buzón de Carolina y termina la conexión TCP.

A su debido tiempo, Carolina arranca su PC, se conecta a su ISP e inicia su programa de correo electrónico. Éste establece una conexión TCP con el servidor POP3 en el puerto 110 de la máquina servidora de correo del ISP. Por lo general, el nombre del DNS o la dirección IP de esta máquina se configura cuando se instala el programa de correo electrónico o cuando se realiza la suscripción al ISP. Una vez que se ha establecido la conexión TCP, el programa de correo electrónico de Carolina se ejecuta en el protocolo POP3 para obtener el contenido del buzón a su disco duro utilizando comandos similares a los de la figura 7-16. Una vez que se ha transferido todo el correo electrónico, se libera la conexión TCP. De hecho, la conexión con el ISP también puede terminarse ahora, puesto que todo el correo electrónico se encuentra en el disco duro de la máquina de Carolina. Por supuesto, para enviar una respuesta, será necesaria otra vez la conexión con el ISP, por lo que generalmente dicha conexión no se termina después de obtener el correo electrónico.

IMAP

Para un usuario que tiene una cuenta de correo electrónico con un ISP que siempre se accede desde una PC, POP3 es adecuado y se utiliza ampliamente debido a su sencillez y robustez. Sin embargo, es un axioma de la industria de la computación el hecho de que siempre que algo funciona bien, alguien comienza a pedir más características (y a obtener más errores). Eso también sucedió con el correo electrónico. Por ejemplo, muchas personas tienen una sola cuenta de correo electrónico en el trabajo o en la escuela y desean accederla desde el trabajo, desde la PC de su casa, desde su computadora portátil cuando están en viaje de negocios y desde algún cibercafé cuando se encuentran de vacaciones. Aunque POP3 permite esto, debido a que descarga todos los mensajes almacenados en cada contacto, el resultado es que los mensajes de correo electrónico del usuario quedan esparcidos rápidamente en múltiples máquinas, más o menos de manera aleatoria, y algunos de ellos ni siquiera en la máquina del usuario.

Esta desventaja dio lugar a un protocolo de entrega final alternativo, **IMAP (Protocolo de Acceso a Mensajes de Internet),** que se define en el RFC 2060. A diferencia de POP3, que asume básicamente que el usuario vaciará el buzón de cada contacto y trabajará sin conexión después de eso, IMAP supone que todo el correo electrónico permanecerá en el servidor de manera indefinida en múltiples buzones de correo. IMAP proporciona mecanismos de gran alcance para leer mensajes o incluso partes de un mensaje, una característica útil cuando se utiliza un módem lento para leer parte del texto de un mensaje dividido en varios fragmentos y que tiene archivos adjuntos grandes de audio y vídeo. Debido a que la suposición más razonable es que los mensajes no se transferirán a la computadora del usuario para un almacenamiento permanente, IMAP proporciona mecanismos para crear, destruir y manipular múltiples buzones en el servidor. De esta forma, un usuario puede mantener un buzón para cada uno de sus contactos y colocar ahí mensajes de la bandeja de entrada después de que se han leído.

IMAP tiene muchas características, como la capacidad de dirigir correo no por número de llegada, como se hace en la figura 7-8, sino utilizando atributos (por ejemplo, dame el primer mensaje

de Juan). A diferencia de POP3, IMAP también puede aceptar correo saliente para enviarlo al destino, así como entregar correo electrónico entrante.

El estilo general del protocolo IMAP es similar al de POP3, como se muestra en la figura 7-16, excepto que hay docenas de comandos. El servidor IMAP escucha en el puerto 143. En la figura 7-17 se muestra una comparación de POP3 e IMAP. Sin embargo, se debe mencionar que no todos los ISPs ni todos los programas de correo electrónico soportan ambos protocolos. Por lo tanto, cuando se elige un programa de correo electrónico, es importante averiguar qué protocolos soporta y asegurarse de que el ISP soporte por lo menos uno de ellos.

Característica	POP3	IMAP
En dónde se define el protocolo	RFC 1939	RFC 2060
Puerto TCP utilizado	110	143
En dónde se almacena el correo electrónico	PC del usuario	Servidor
En dónde se lee el correo electrónico	Sin conexión	En línea
Tiempo de conexión requerido	Poco	Mucho
Uso de recursos del servidor	Mínimo	Amplio
Múltiples buzones	No	Sí
Quién respalda los buzones	Usuario	ISP
Bueno para los usuarios móviles	No	Sí
Control del usuario sobre la descarga	Poco	Mucho
Descargas parciales de mensajes	No	Sí
¿Es un problema el espacio en disco?	No	Con el tiempo podría serlo
Sencillo de implementar	Sí	No
Soporte amplio	Sí	En crecimiento

Figura 7-17. Comparación entre POP3 e IMAP.

Características de entrega

Independientemente de si se utiliza POP3 o IMAP, muchos sistemas proporcionan ganchos para procesamiento adicional de correo electrónico entrante. Una característica especialmente valiosa para muchos usuarios de correo electrónico es la capacidad de establecer **filtros.** Éstas son reglas que se verifican cuando llega el correo electrónico o cuando se inicia el agente de usuario. Cada regla especifica una condición y una acción. Por ejemplo, una regla podría decir que cualquier mensaje del jefe va al buzón número 1, cualquier mensaje de un grupo seleccionado de amigos va al buzón número 2, y cualquier mensaje que contenga en la línea Asunto ciertas palabras objetables se descarta sin ningún comentario.

Algunos ISPs proporcionan un filtro que clasifica de manera automática el correo electrónico como importante o como publicidad no deseada (correo basura) y almacena cada mensaje en el buzón correspondiente. Por lo general, tales filtros funcionan verificando primero si el origen es un *spammer* (persona que envía correo basura) conocido. A continuación, por lo general examinan la

línea de asunto. Si cientos de usuarios han recibido un mensaje con la misma línea de asunto, probablemente sea correo basura. También se utilizan otras técnicas para la detección de correo basura.

Otra característica de entrega que por lo general se proporciona es la capacidad de reenviar (de manera temporal) correo electrónico entrante a una dirección diferente, la cual puede ser una computadora manejada por un servicio de localización comercial, el cual a continuación localiza al usuario por radio o satélite y le indica que tiene un mensaje en su localizador.

Otra característica común de la entrega final es la capacidad de instalar un **demonio de vacaciones.** Éste es un programa que examina cada mensaje entrante y envía al emisor una respuesta grabada como

```
Hola. Estoy de vacaciones. Regresaré el 24 de agosto. Tenga un buen fin de semana.
```

Tales respuestas también pueden especificar cómo manejar asuntos urgentes en el ínterin, las personas a quienes se puede acudir en caso de problemas específicos, etcétera. La mayoría de los demonios de vacaciones lleva un registro de a quién le ha enviado respuestas grabadas y se abstiene de enviar a la misma persona una segunda respuesta. Los buenos demonios también verifican si los mensajes entrantes se enviaron a una lista de correo, y de ser así, no envían una respuesta grabada. (Las personas que envían mensajes a listas de correo grandes durante el verano probablemente no desean obtener cientos de respuestas que detallen los planes de vacaciones de todos los demás.)

El autor una vez se topó con una forma extrema de procesamiento de correo cuando envió un mensaje de correo electrónico a una persona que se jactaba de recibir 600 mensajes al día. Su identidad no se revelará aquí, por miedo a que la mitad de los lectores de este libro también le envíen a él correo electrónico. Llamémoslo Juan.

Juan ha instalado un robot de correo electrónico que verifica cada mensaje entrante para ver si proviene de un nuevo remitente. De ser así, regresa una respuesta grabada que explica que Juan ya no puede leer personalmente todo su correo electrónico. En su lugar, ha producido un documento de FAQs (preguntas más frecuentes) personales en el que responde muchas preguntas que generalmente se le formulan. Es común que los grupos de noticias tengan FAQs, no personas.

La FAQ de Juan da su dirección, fax y números telefónicos e indica cómo contactar a su compañía. Explica cómo contratarlo como orador y describe en dónde obtener sus ponencias y otros documentos. También proporciona apuntadores a software que ha escrito, a conferencias que está impartiendo, a un estándar del que es editor, etcétera. Tal vez esta estrategia sea necesaria, pero podría ser que una FAQ personal sea el último símbolo de *status*.

Correo de Web

Un tema final que vale la pena mencionar es el correo de Web. Algunos sitios Web, como Hotmail y Yahoo, proporcionan servicio de correo electrónico a cualquiera que lo desee. Funcionan como se menciona a continuación. Tienen agentes de transferencia de mensajes normales que escuchan el puerto 25 para conexiones SMTP entrantes. Para contactar, digamos, Hotmail, usted tiene que adquirir su registro *MX* de DNS, por ejemplo, tecleando

```
host –a –v hotmail.com
```

en un sistema UNIX. Suponga que el servidor de correo se llama *mx10.hotmail.com*, por lo que al escribir

```
telnet mx10.hotmail.com 25
```

puede establecer una conexión TCP a través de la cual pueden enviarse comandos SMTP de la forma usual. Hasta ahora, nada es inusual, excepto que estos servidores enormes por lo general están ocupados, de modo que tal vez se necesiten varios intentos para que se acepte una conexión TCP.

La parte interesante es la forma en que se entrega el correo electrónico. Básicamente, cuando el usuario va a la página Web de correo electrónico, se despliega un formulario en el que se pide al usuario que introduzca un nombre de usuario y una contraseña. Cuando el usuario hace clic en Registrarse, el nombre de usuario y la contraseña se envían al servidor, quien los valida. Si el inicio de sesión es exitoso, el servidor encuentra el buzón del usuario y construye una lista similar a la de la figura 7-8, sólo que formateada como página Web en HTML. Esta página Web se envía al navegador para su despliegue. En muchos de los elementos de la página se puede hacer clic, por lo que los mensajes se pueden leer, eliminar, etcétera.

7.3 WORLD WIDE WEB

World Wide Web es un armazón arquitectónico para acceder a documentos vinculados distribuidos en miles de máquinas de toda Internet; en diez años, pasó de ser una manera de distribuir datos sobre física de alta energía a la aplicación que millones de personas piensan que es "Internet". Su enorme popularidad se deriva del hecho de que tiene una interfaz gráfica atractiva que es fácil de usar por los principiantes y proporciona un enorme cúmulo de información sobre casi cualquier tema concebible, desde aborígenes hasta zoología.

Web (también conocida como **WWW**) comenzó en 1989 en el CERN, el Centro Europeo de Investigación Nuclear. El CERN tiene varios aceleradores en los que los científicos de los países europeos participantes llevan a cabo investigaciones sobre física de partículas. Estos equipos con frecuencia tienen miembros de media docena de países o más. La mayoría de los experimentos son altamente complejos, y requieren años de planeación adelantada y construcción de equipo. Web surgió de la necesidad de lograr que estos grandes grupos de investigadores dispersos internacionalmente colaboren usando un conjunto siempre cambiante de informes, planos, dibujos, fotos y otros documentos.

La propuesta inicial de una red de documentos vinculados surgió del físico del CERN Tim Berners-Lee en marzo de 1989. El primer prototipo (basado en texto) estaba en operación 18 meses después. En diciembre de 1991 se hizo una demostración pública en la conferencia Hypertext '91 en San Antonio, Texas.

Esta demostración y su publicidad acompañante captó la atención de otros investigadores, lo que llevó a Marc Andreessen de la Universidad de Illinois a comenzar el desarrollo del primer navegador gráfico, Mosaic. Se liberó en febrero de 1993. Mosaic fue tan popular que un año más tarde, Andreessen formó su propia compañía, Netscape Communications Corp., cuya meta era desarrollar clientes, servidores y otro tipo de software Web. Cuando Netscape se liberó en 1995, los

inversionistas, creyendo que éste era el siguiente Microsoft, pagaron 1500 millones de dólares por las acciones. Esta transacción fue muy sorprendente porque la compañía sólo tenía un producto, operaba profundamente en la red y había anunciado que no esperaba obtener utilidades en un futuro previsible. Durante los siguientes tres años, Netscape Navigator y Microsoft Internet Explorer sostuvieron una "guerra de navegadores", cada uno tratando frenéticamente de agregar más características (y, por ende, más errores) que el otro. En 1998, America Online compró Netscape Communications Corp. por 4,200 millones de dólares, terminando con la breve vida de Netscape como compañía independiente.

En 1994, el CERN y el MIT firmaron un acuerdo para establecer el **World Wide Web Consortium** (W3C), una organización dedicada al desarrollo de Web, la estandarización de protocolos y el fomento de interoperabilidad entre los sitios. Berners-Lee se convirtió en el director. Desde entonces, cientos de universidades y compañías se han unido al consorcio. Aunque hay más libros sobre Web de los que pueden contarse, el mejor lugar para recibir información actualizada sobre Web es (naturalmente) la Web misma. La página de inicio del consorcio puede encontrarse en *http://www.w3.org.* Los lectores interesados pueden encontrar ahí vínculos con páginas que cubren todos los documentos y actividades del consorcio.

7.3.1 Panorama de la arquitectura

Desde el punto de vista del usuario, Web consiste en un enorme conjunto de documentos a nivel mundial, generalmente llamados **páginas Web.** Cada página puede contener vínculos (apuntadores) a otras páginas relacionadas en cualquier lugar del mundo. Los usuarios pueden seguir un vínculo haciendo clic en él, lo que los lleva a la página apuntada. Este proceso puede repetirse de manera indefinida. La idea de hacer que una página apunte a otra, lo que ahora se conoce como **hipertexto,** fue inventada por un profesor visionario de ingeniería eléctrica del MIT, Vannevar Bush, en 1945, mucho antes de que se inventara Internet.

Las páginas se ven mediante un programa llamado **navegador;** Internet Explorer y Netscape son dos de los navegadores más populares. El navegador obtiene la página solicitada, interpreta el texto y los comandos de formateo que contienen, y despliega la página, adecuadamente formateada, en la pantalla. En la figura 7-18(a) se da un ejemplo. Al igual que en muchas páginas Web, ésta comienza con un título, contiene cierta información y termina con la dirección de correo electrónico del encargado de mantenimiento de la página. Las cadenas de texto que son vínculos a otras páginas, llamadas **hipervínculos,** se resaltan, ya sea mediante subrayado, presentación en un color especial, o ambas cosas. Para seguir un vínculo, el usuario coloca el cursor en el área resaltada (usando el ratón o las teclas de flecha) y la selecciona (haciendo clic con un botón del ratón u oprimiendo ENTRAR). Aunque existen algunos navegadores no gráficos, como Lynx, no son tan comunes como los navegadores gráficos, por lo que nos concentraremos en estos últimos. También se están desarrollando navegadores basados en voz.

Los usuarios con curiosidad respecto al Departamento de psicología animal pueden aprender más sobre él haciendo clic en su nombre (subrayado). A continuación el navegador trae la página

BIENVENIDOS A LA PÁGINA DE INICIO DE WWW DE LA UNIVERSIDAD DE EAST PODUNK

- Información de la Universidad
  - Información sobre inscripciones
  - Mapa de las instalaciones
  - Indicaciones para llegar a las instalaciones
  - El cuerpo estudiantil de la UEP
- Departamentos académicos
  - Departamento de psicología animal
  - Departamento de estudios alternativos
  - Departamento de cocina microbiótica
  - Departamento de estudios no tradicionales
  - Departamento de estudios tradicionales

Webmaster@eastpodunk.edu

(a)

DEPARTAMENTO DE PSICOLOGÍA ANIMAL

- Información sobre posibles carreras
- Personal
  - Miembros del profesorado
  - Estudiantes de postgrado
  - Personal no académico
- Proyectos de investigación
- Puestos disponibles
- Nuestros cursos más populares
  - Manejo de herbívoros
  - Administración de caballos
  - Negociando con su mascota
  - Construcción de perreras amables con el usuario
- Lista completa de cursos

Webmaster@animalpsyc.eastpodunk.edu

(b)

Figura 7-18. (a) Página Web. (b) Página accedida al hacer clic en Departamento de psicología animal.

a la que está vinculado el nombre y la despliega, como se muestra en la figura 7-18(b). También es posible hacer clic en los elementos subrayados que se muestran en la figura para desplegar otras páginas, y así se puede continuar. La página nueva puede estar en la misma máquina que la primera, o en otra máquina del otro lado del mundo. El usuario no puede saberlo. El navegador es quien realiza la obtención de páginas sin ayuda del usuario. Si éste llega a regresar a la página principal,

los vínculos ya seguidos tal vez aparezcan con un subrayado punteado (y posiblemente con otro color) para distinguirlos de los vínculos que no se han seguido. Observe que hacer clic en la línea de *Información de la Universidad* de la página principal no tiene ningún efecto; no está subrayada, lo que significa que simplemente es texto y no está vinculada con otra página.

En la figura 7-19 se muestra el modelo básico de la forma en que funciona Web. Aquí, el navegador despliega una página Web en la máquina cliente. Cuando el usuario hace clic en una línea de texto que está vinculada a una página del servidor *abcd.com*, el navegador sigue el hipervínculo enviándole un mensaje al servidor *abcd.com* en el que le solicita la página. Cuando ésta llega, se despliega. Si contiene un hipervínculo a una página del servidor *xyz.com* en el que se hace clic, el navegador a continuación envía un mensaje a dicha máquina solicitando esa página, y así de forma indefinida.

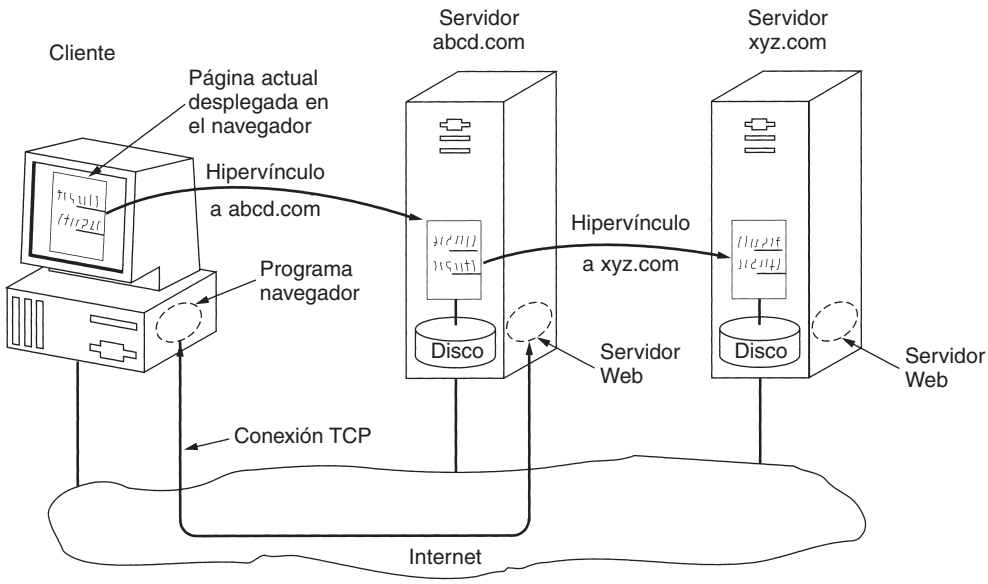


Figura 7-19. Las partes del modelo Web.

El cliente

Ahora examinemos el cliente de la figura 7-19 con mayor detalle. En esencia, un navegador es un programa que puede desplegar una página Web y atrapar los clics que se hacen en los elementos de la página desplegada. Cuando se selecciona un elemento, el navegador sigue el hipervínculo y obtiene la página seleccionada. Por lo tanto, el hipervínculo incrustado necesita una manera de nombrar cualquier página que se encuentre en Web. Las páginas se nombran utilizando **URLs (Localizadores Uniformes de Recursos).** Un URL típico es

http://www.abcd.com/productos.html

Analizaremos los URLs más adelante en este capítulo. Por el momento, es suficiente saber que un URL tiene tres partes: el nombre del protocolo (*http*), el nombre DNS de la máquina donde se localiza la página (*www.abcd.com*) y (por lo general) el nombre del archivo que contiene la página (*productos.html*).

Cuando un usuario hace clic en un hipervínculo, el navegador lleva a cabo una serie de pasos para obtener la página a la que se está apuntado. Suponga que un usuario está navegando Web y encuentra un vínculo sobre telefonía de Internet que apunta a la página de inicio de la ITU, que es *http://www.itu.org/home/index.html*. Listemos los pasos que se dan cuando se selecciona este vínculo.

1. El navegador determina el URL (enviando lo que se seleccionó).
2. El navegador pide al DNS la dirección IP de *www.itu.org*.
3. DNS responde con 156.106.192.32.
4. El navegador realiza una conexión TCP con el puerto 80 en 156.106.192.32.
5. Después envía un mensaje en el que solicita el archivo */home/index.html*.
6. El servidor *www.itu.org* envía el archivo */home/index.html*.
7. Se libera la conexión TCP.
8. El navegador despliega todo el texto de */home/index.html*.
9. El navegador obtiene y despliega todas las imágenes del archivo.

Muchos navegadores despliegan en una línea de estado, que se encuentra en la parte inferior de la pantalla, qué paso están ejecutando actualmente. De esta manera, cuando el desempeño es pobre, el usuario puede ver si se debe a que el DNS o el servidor no están respondiendo, o simplemente hay congestionamiento en la red durante la transmisión de la página.

Para poder desplegar la nueva página (o cualquiera), el navegador tiene que entender su formato. Para permitir que todos los navegadores entiendan todas las páginas Web, éstas se escriben en un lenguaje estandarizado llamado HTML, el cual las describe. Más adelante en este capítulo analizaremos esto con detalle.

Aunque un navegador es básicamente un intérprete HTML, la mayoría de los navegadores tiene varios botones y características para facilitar la navegación en Web. La mayoría tiene un botón para regresar a la página anterior, uno para ir a la siguiente (el cual sólo funciona una vez que el usuario ha regresado de esa página) y uno para ir directamente a la página de inicio del usuario. La mayoría de los navegadores tiene un botón o elemento para establecer un marcador y otro para desplegar la lista de marcadores, lo que hace posible volver a visitar cualquiera de ellos con sólo algunos clics del ratón. Las páginas también pueden guardarse en disco o imprimirse. Por lo general, hay varias opciones disponibles para controlar el diseño de la pantalla y establecer varias preferencias de usuario.

Además de tener texto ordinario (que no está subrayado) e hipertexto (texto subrayado), las páginas Web también pueden tener iconos, dibujos de líneas, mapas y fotografías. Cada uno de

éstos puede (opcionalmente) vincularse a otra página. Hacer clic en alguno de estos elementos causa que el navegador obtenga la página vinculada y la despliegue en pantalla, al igual que al hacer clic en el texto. En el caso de imágenes como fotos y mapas, la página que se obtenga a continuación depende de en cuál parte de la imagen se hizo clic.

No todas las páginas contienen HTML. Una página puede consistir en un documento con formato PDF, un icono con formato GIF, una fotografía con formato JPEG, una canción con formato MP3, un vídeo con formato MPEG, o cualquiera de los cientos de los otros tipos de archivos. Puesto que las páginas HTML estándar pueden vincular cualquiera de éstos, el navegador tiene un problema cuando encuentra una página que no puede interpretar.

En lugar de agrandar cada vez más los navegadores incorporándoles intérpretes para una colección creciente de tipos de archivos, la mayoría de los navegadores ha elegido una solución más general. Cuando un servidor regresa una página, también regresa alguna información adicional acerca de ella. Dicha información incluye el tipo MIME de la página (vea la figura 7-12). Las páginas del tipo *text/html* se despliegan de manera directa, como las páginas de algunos otros tipos integrados. Si el tipo MIME no es de los integrados, el navegador consulta su tabla de tipos MIME que le indique cómo desplegar la página. Esta tabla asocia un tipo MIME con un visor.

Hay dos posibilidades: *plug-ins* y aplicaciones auxiliares. Un ***plug-in*** (conector) es un módulo de código que el navegador obtiene de un directorio especial del disco y lo instala como una extensión de sí mismo, como se ilustra en la figura 7-20(a). Debido a que los *plug-ins* se ejecutan dentro del navegador, tienen acceso a la página actual y pueden modificar su apariencia. Después de que el *plug-in* ha hecho su trabajo (por lo general después de que el usuario se ha movido a una página Web diferente), se elimina de la memoria del navegador.

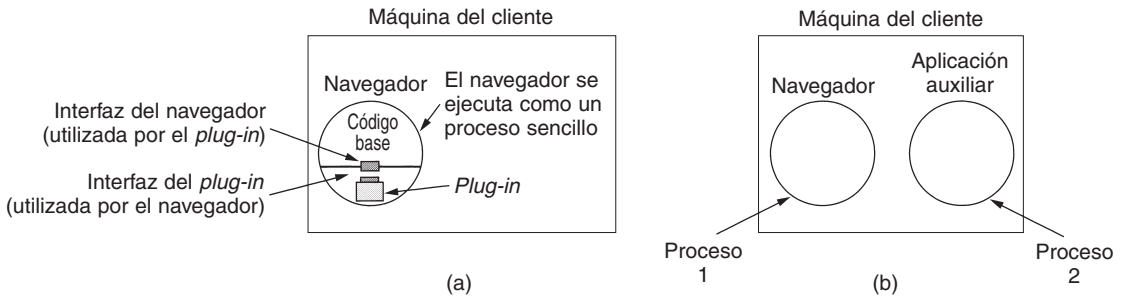


Figura 7-20. (a) Un *plug-in* de navegador. (b) Una aplicación auxiliar.

Cada navegador tiene un conjunto de procedimientos que todos los *plug-ins* tienen que implementar a fin de que el navegador pueda llamarlos. Por ejemplo, generalmente hay un procedimiento que el código base del navegador llama para proporcionar datos que desplegar al *plug-in*. Este conjunto de procedimientos es la interfaz del *plug-in* y es específico del navegador.

Además, pone a disposición del *plug-in* un conjunto de sus propios procedimientos, que proporcionan servicios a los *plug-ins*. Los procedimientos típicos de la interfaz del navegador son para asignar y liberar memoria, desplegar un mensaje en la línea de estado del navegador y consultar al navegador sobre los parámetros.

Para poder utilizar un *plug-in*, primero debe instalarse. El proceso de instalación común consiste en que el usuario vaya al sitio Web del *plug-in* y descargue un archivo de instalación. En Windows, por lo general este archivo es un zip que se extrae en forma automática y que tiene la extensión *.exe*. Cuando se hace doble clic en dicho archivo, se ejecuta un pequeño programa insertado al frente del archivo zip. Este programa extrae el *plug-in* y lo copia en el directorio de *plug-ins* del navegador. A continuación hace las llamadas apropiadas para registrar el tipo MIME del *plug-in* y para asociarlo con éste. En UNIX, el instalador es por lo general una secuencia de comandos (*script*) de shell que maneja la copia y el registro.

La otra forma de ampliar un navegador es utilizar una **aplicación auxiliar.** Ésta es un programa completo que se ejecuta como un proceso independiente [vea la figura 7-20(b)]. Debido a esto no ofrece interfaz con el navegador y no utiliza los servicios de éste. En su lugar, por lo general simplemente acepta el nombre de un archivo de trabajo en el que se ha almacenado el archivo de contenido, abre dicho archivo y despliega su contenido. Por lo general, las aplicaciones auxiliares son programas grandes que existen independientemente del navegador, como Acrobat Reader de Adobe para desplegar archivos PDF o Microsoft Word. Algunos programas (como Acrobat) tienen un *plug-in* que invoca a la aplicación auxiliar misma.

Muchas aplicaciones auxiliares utilizan el tipo MIME *aplicación*. Se ha definido un número considerable de subtipos, por ejemplo, *aplicación*/*pdf* para los archivos PDF y *aplicación*/*msword* para archivos de Word. De esta manera, un URL puede apuntar en forma directa a un archivo PDF o Word, y cuando el usuario hace clic en él, se inician automáticamente Acrobat o Word y se les proporciona el mismo nombre de un archivo de trabajo que contiene los datos a desplegar. En consecuencia, es posible configurar a los navegadores para manejar un número virtualmente ilimitado de tipos de documento sin tener que modificar el navegador. Los servidores Web modernos con frecuencia están configurados con cientos de combinaciones de tipos/subtipos y se agregan más cada vez que se instala un nuevo programa.

Las aplicaciones auxiliares no están restringidas a utilizar el tipo MIME *aplicación*. Por ejemplo, Adobe Photoshop utiliza *imagen/x-photoshop* y RealOne Player tiene la capacidad de manejar *audio/mp3*.

En Windows, cuando se instala un programa en la computadora, ésta registra los tipos MIME que necesita manejar. Este mecanismo causa un conflicto cuando múltiples visores están disponibles para algún subtipo, como *video/mpg*. Lo que sucede es que el último programa que se registra sobrescribe las asociaciones existentes (tipo MIME, aplicación auxiliar) con las que requiere para sí mismo. Como consecuencia, instalar un nuevo programa podría cambiar la forma en que un navegador maneja los tipos existentes.

En UNIX, este proceso de registro por lo general no es automático. El usuario tiene que actualizar de manera manual ciertos archivos de configuración. Este método significa más trabajo pero menos sorpresas.

Los navegadores también pueden abrir archivos locales, en lugar de obtenerlos de los servidores Web remotos. Debido a que los archivos locales no tienen tipos MIME, el navegador necesita alguna manera para determinar cuál *plug-in* o aplicación auxiliar utilizar para los tipos que no sean sus tipos integrados, como *texto/html* e *imagen/jpeg*. Para manejar archivos locales, las aplicaciones auxiliares pueden asociarse con una extensión de archivo, así como con un tipo MIME. Con

la configuración estándar, *foo.pdf* se abrirá en Acrobat y *bar.doc* se abrirá en Word. Algunos navegadores utilizan el tipo MIME, la extensión de archivo e incluso información tomada del archivo mismo para adivinar el tipo MIME. En particular, Internet Explorer se apoya más fuertemente en la extensión de archivo que en el tipo MIME, cuando puede.

Aquí también pueden surgir conflictos debido a que muchos programas están dispuestos, de hecho ansiosos, a manejar, digamos, *.mpg*. Durante la instalación, los programas que están diseñados para los profesionales por lo general despliegan casillas de verificación para los tipos MIME y las extensiones que pueden manejar a fin de permitir que el usuario seleccione los apropiados y, de esta forma, no sobrescriba las asociaciones existentes por accidente. Los programas dirigidos al mercado consumidor asumen que el usuario no tiene idea de lo que es un tipo MIME y simplemente toman todos los que pueden sin importarles lo que los programas instalados anteriormente hayan hecho.

La capacidad de ampliar el navegador con un número significativo de tipos nuevos es conveniente pero también puede generar problemas. Cuando Internet Explorer obtiene un archivo con la extensión *exe*, sabe que este archivo es un programa ejecutable y, por lo tanto, no tiene una aplicación auxiliar. La acción obvia es ejecutar el programa. Sin embargo, esto podría ser un enorme hoyo de seguridad. Todo lo que un sitio Web malicioso tiene que hacer es producir una página Web con imágenes de, digamos, estrellas de cine o héroes de deportes, todo lo cual se vincula con un virus. Un solo clic en una imagen podría causar la obtención de un programa ejecutable potencialmente hostil, y su ejecución en la máquina del usuario. Para evitar huéspedes no deseados como éstos, es posible configurar Internet Explorer para que sea selectivo al ejecutar automáticamente programas desconocidos, pero no todos los usuarios saben cómo manejar la configuración.

En UNIX puede ocurrir un problema análogo con las secuencias de comandos de shell, pero eso requiere que el usuario instale de manera consciente el shell como una aplicación auxiliar. Por fortuna, esta instalación es tan complicada que nadie podría realizarla por accidente (y pocas personas pueden hacerlo de manera intencional).

El servidor

Basta de hablar sobre el cliente. Ahora echemos un vistazo al servidor. Como vimos anteriormente, cuando el usuario teclea un URL o hace clic en una línea de hipertexto, el navegador lo analiza e interpreta la parte entre *http://* y la siguiente diagonal como un nombre DNS a buscar. Una vez que el navegador tiene la dirección IP del servidor, establece una conexión TCP con el puerto 80 de ese servidor. A continuación envía un comando que contiene el resto del URL, que es el nombre del archivo que se encuentra en ese servidor. Éste regresa el archivo para que el navegador lo despliegue.

Como una primera aproximación, un servidor Web es similar al de la figura 6-6. A éste, al igual que a un servidor Web real, se le proporciona el nombre del archivo a buscar y regresar. En ambos casos, los pasos que da el servidor en su ciclo principal son:

1. Acepta una conexión TCP de un cliente (un navegador).
2. Obtiene el nombre del archivo solicitado.

3. Obtiene el archivo (del disco).
4. Regresa el archivo al cliente.
5. Libera la conexión TCP.

Los servidores Web actuales tienen más características, pero en esencia, esto es lo que hacen.

El problema de este diseño es que cada solicitud requiere un acceso al disco para obtener el archivo. El resultado es que el servidor Web no puede atender más solicitudes por segundo que accesos al disco. Un disco SCSI de alta calidad tiene un tiempo de acceso promedio de aproximadamente 5 mseg, lo que limita al servidor a lo mucho 200 solicitudes/seg, o menos si se tienen que leer con frecuencia archivos grandes. Para un sitio Web grande, esta cifra es muy baja.

Una mejora obvia (utilizada por todos los servidores Web) es mantener un caché en la memoria de los *n* archivos más recientemente utilizados. Antes de ir al disco para obtener un archivo, el servidor verifica el caché. Si el archivo está ahí, se puede proporcionar directamente desde memoria, con lo que se elimina el acceso al disco. Aunque el almacenamiento en caché efectivo requiere una gran cantidad de memoria principal y tiempo de procesamiento extra para verificar el caché y manejar su contenido, el ahorro de tiempo justifica la sobrecarga y costo implícitos.

El siguiente paso para construir un servidor más rápido es hacerlo de múltiples subprocesos. En un diseño, el servidor consiste en un módulo de *front end* que acepta todas las solicitudes entrantes y *k* módulos de procesamiento, como se muestra en la figura 7-21. Los $k + 1$ subprocesos pertenecen al mismo proceso por lo que todos los módulos de procesamiento tienen acceso al caché dentro del espacio de direcciones del proceso. Cuando llega una solicitud, el *front end* la acepta y construye un registro corto que la describe. Después entrega el registro a uno de los módulos de procesamiento. En otro diseño posible, se elimina el *front end* y cada módulo de procesamiento trata de adquirir sus propias solicitudes, pero se necesita un protocolo de bloqueo para evitar conflictos.

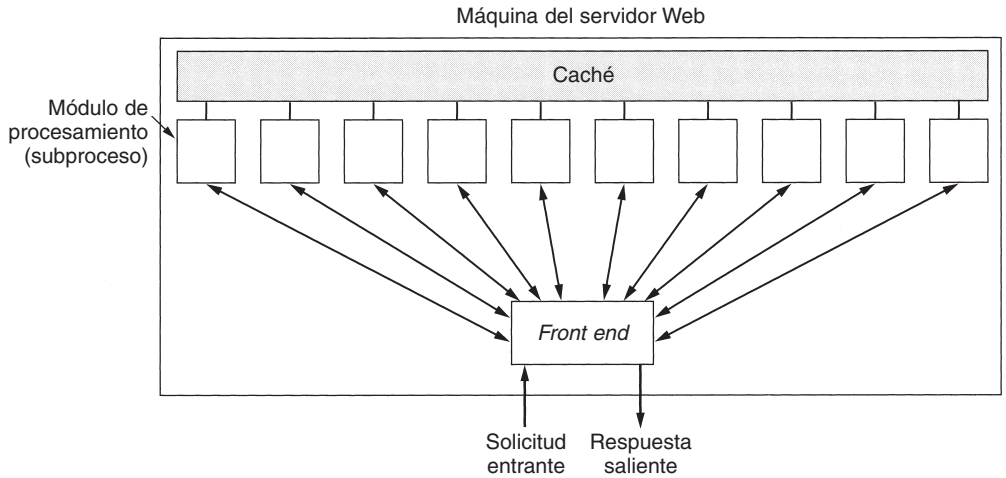


Figura 7-21. Un servidor Web con múltiples subprocesos con un *front end* y módulos de procesamiento.

El módulo de procesamiento primero verifica el caché para ver si el archivo solicitado está ahí. De ser así, actualiza el registro para incluir un apuntador al archivo del registro. Si no está ahí, el módulo de procesamiento inicia una operación de disco para leerlo en el caché (posiblemente descartando algunos otros archivos en caché para hacerle espacio). Cuando el archivo llega del disco, se coloca en el caché y también se regresa al cliente.

La ventaja de este esquema es que mientras que uno o más módulos de procesamiento están bloqueados esperando a que termine una operación del disco (y, por lo tanto, no consumen tiempo de CPU), otros módulos pueden estar trabajando activamente en otras solicitudes. Por supuesto, para obtener cualquier mejora real sobre el modelo de un solo subproceso, es necesario tener múltiples discos, a fin de que más de un disco pueda estar ocupado al mismo tiempo. Con *k* módulos de procesamiento y *k* discos, la velocidad real de transporte puede ser *k* veces mayor que con el servidor de un solo subproceso y un disco.

En teoría, un servidor de un solo subproceso y *k* discos también puede ganar un factor de *k*, pero el código y la administración son mucho más complicados puesto que las llamadas de sistema READ de bloqueo normal no se pueden utilizar para acceder al disco. Con un servidor de múltiples subprocesos se pueden utilizar puesto que READ sólo bloquea el subproceso que hizo la llamada, no todo el proceso.

Los servidores Web modernos hacen más que sólo aceptar y regresar nombres de archivos. De hecho, el procesamiento real de cada solicitud puede ser muy complicado. Por esta razón, en muchos servidores cada módulo de procesamiento realiza una serie de pasos. El *front end* pasa cada solicitud entrante al primer módulo disponible, que después la transporta mediante algunos de los siguientes pasos, dependiendo de los que sean necesarios para esa solicitud en particular.

1. Resuelve el nombre de la página Web solicitada.
2. Autentica al cliente.
3. Realiza control de acceso en el cliente.
4. Realiza control de acceso en la página Web.
5. Verifica el caché.
6. Obtiene del disco la página solicitada.
7. Determina el tipo MIME que se incluirá en la respuesta.
8. Se encarga de diversos detalles.
9. Regresa la respuesta al cliente.
10. Realiza una entrada en el registro del servidor.

El paso 1 es necesario porque la solicitud entrante tal vez no contenga el nombre real del archivo como una cadena literal. Por ejemplo, considere el URL *http://www.cs.vu.nl*, que tiene un nombre de archivo vacío. Tiene que expandirse a algún nombre de archivo predeterminado. Además, los navegadores modernos pueden especificar el lenguaje predeterminado del usuario (por ejemplo,

italiano o inglés), lo cual posibilita que el servidor seleccione una página Web en ese lenguaje, si está disponible. En general, la expansión de nombres no es tan trivial como podría parecer a primera vista, debido a una variedad de convenciones acerca de la asignación de nombres de archivos.

El paso 2 consiste en verificar la identidad del cliente. Este paso es necesario para las páginas que no están disponibles para el público en general. Más adelante en este capítulo analizaremos una forma de hacerlo.

El paso 3 verifica si hay restricciones con respecto a si la solicitud se puede satisfacer a partir de la identidad y ubicación del cliente. El paso 4 verifica si hay restricciones de acceso asociadas con la página misma. Si cierto archivo (por ejemplo, *.htaccess*) se encuentra en el directorio donde se localiza la página deseada, ésta puede prohibir que dominios particulares accedan al archivo; por ejemplo, tal vez sólo permita que usuarios que están dentro de la compañía accedan al archivo.

Los pasos 5 y 6 consisten en obtener la página. El paso 6 necesita poder manejar múltiples lecturas de disco al mismo tiempo.

El paso 7 consiste en determinar el tipo MIME a partir de la extensión del archivo, de las primeras palabras del archivo, de un archivo de configuración y, posiblemente, de otros recursos. El paso 8 tiene que ver con una variedad de tareas, como la construcción de un perfil de usuario o la recolección de ciertas estadísticas.

El paso 9 tiene que ver con el lugar al que se envía el resultado, y el paso 10 crea una entrada en el registro del sistema para propósitos administrativos. Tales registros pueden examinarse en busca de información valiosa acerca del comportamiento de los usuarios, como el orden en que acceden las páginas.

Si llegan demasiadas solicitudes cada segundo, la CPU no será capaz de manejar la carga de procesamiento, sin importar cuántos discos se utilicen en paralelo. La solución es agregar más nodos (computadoras), posiblemente con discos replicados para evitar que los discos se vuelvan el siguiente cuello de botella. Esto lleva al modelo de **granja de servidores** de la figura 7-22. Un *front end* aún acepta solicitudes entrantes pero las distribuye en múltiples CPUs en lugar de en múltiples subprocesos para reducir la carga en cada computadora. Las máquinas individuales podrían contar con múltiples subprocesos y canales como en el esquema anterior.

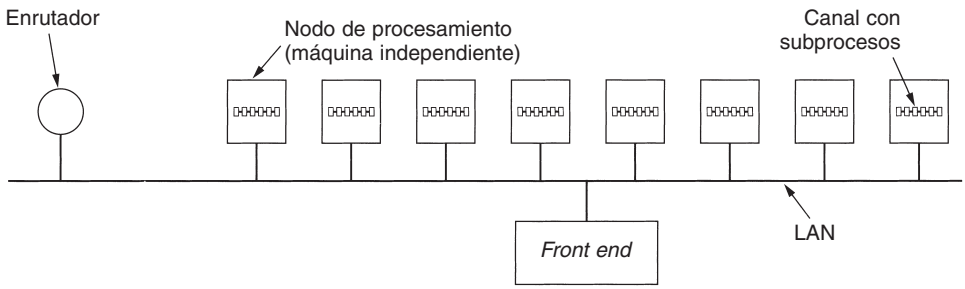


Figura 7-22. Granja de servidores.

Un problema con las granjas de servidores es que no hay caché compartido debido a que cada nodo de procesamiento tiene su propia memoria —a menos que se utilice un multiprocesador

de memoria compartida costoso. Una forma de medir esta pérdida de rendimiento es hacer que un *front end* registre el lugar a donde envía cada respuesta y que envíe las solicitudes subsecuentes de la misma página al mismo nodo. Hacer esto ocasiona que cada nodo sea un especialista en ciertas páginas a fin de que el espacio en caché no se desperdicie al tener cada archivo en cada caché.

Otro problema con las granjas de servidores es que la conexión TCP del cliente termine en el *front end*, por lo que la respuesta puede pasar a través del *front end*. Esta situación se ilustra en la figura 7-23(a), donde la solicitud (1) y la respuesta (4) pasan a través del *front end*. Para solucionar este problema, algunas veces se utiliza un truco llamado **transferencia TCP** (**TCP handoff**)**.** Con ésta, el punto final de la conexión TCP se pasa al nodo de procesamiento a fin de que pueda contestar directamente al cliente, que se muestra como (3) en la figura 7-23(b). Esta transferencia se realiza de tal forma que es transparente para el cliente.

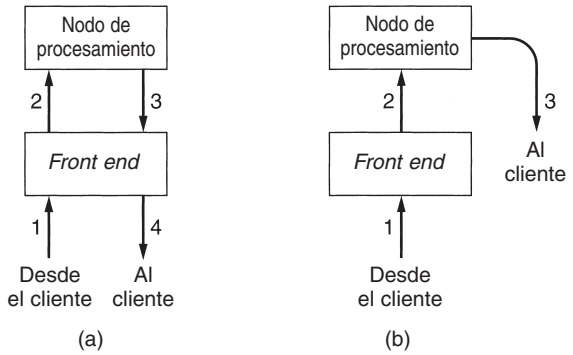


Figura 7-23. (a) Secuencia normal de mensajes solicitud-respuesta. (b) Secuencia cuando se utiliza la transferencia TCP.

URLs—Localizadores Uniformes de Recursos

Hemos dicho repetidamente que las páginas Web pueden contener apuntadores a otras páginas Web. Ahora es tiempo de ver la implementación de estos apuntadores. Cuando se creó Web, de inmediato fue evidente que para que una página Web apuntara a otra se requerían mecanismos para nombrar y localizar las páginas. En particular, había tres preguntas que debían contestarse antes de poder presentar visualmente una página:

1. ¿Cómo se llama la página?
2. ¿Dónde está la página?
3. ¿Cómo se puede acceder a la página?

Si cada página tuviera asignado de alguna manera un nombre único, no habría ambigüedad al identificar las páginas. No obstante, el problema no estaría resuelto. Considere un paralelismo

entre gente y páginas. En Estados Unidos casi todos tienen un número de seguro social, que es un identificador único, puesto que no hay dos personas que tengan el mismo. Sin embargo, armados sólo con el número de seguro social, no hay manera de encontrar la dirección del dueño y, ciertamente, tampoco de saber si se debe escribir a la persona en inglés, español o chino. Web tiene básicamente los mismos problemas.

La solución escogida identifica las páginas de una manera que resuelve los tres problemas a la vez. A cada página se le asigna un **URL** (**Localizador Uniforme de Recursos**) que sirve efectivamente como nombre mundial de la página. Los URLs tienen tres partes: el protocolo (también llamado **esquema**), el nombre DNS de la máquina en donde se encuentra la página y un nombre local que indica de manera única la página específica (por lo general, sólo un nombre de archivo de la máquina en que reside). Por ejemplo, el sitio Web del departamento del autor contiene varios vídeos sobre la universidad y la ciudad de Ámsterdam. El URL de la página de los vídeos es

```
http://www.cs.vu.nl/video/index-en.html
```

Este URL consta de tres partes: el protocolo (*http*), el nombre DNS del *host* (*www.cs.vu.nl*) y el nombre del archivo (*video/index-en.html*), con cierta puntuación que separa las partes. El nombre de archivo es una ruta relativa al directorio Web predeterminado en *cs.vu.nl*.

Muchos sitios cuentan con ciertos atajos para los nombres de archivos. En muchos sitios, un nombre de archivo nulo tiende a ser de manera predeterminada la página de inicio de la organización. Por lo general, cuando el archivo nombrado es un directorio, esto implica un archivo nombrado *index.html*. Por último, *~user/* podría tener una correspondencia con el directorio WWW de *user* y con el archivo *index.html* de ese directorio. Por tanto, la página de inicio del autor puede encontrarse en

```
http://www.cs.vu.nl/~ast/
```

aunque el nombre de archivo real es *index.html* en un directorio predeterminado.

Ahora debe quedar clara la manera en que funciona el hipertexto. Para que pueda hacerse clic en una parte del texto, el redactor de la página debe proporcionar dos elementos de información: el texto en el que se puede hacer clic y el URL de la página a la que se debe ir si se hace clic en dicho texto. Explicaremos la sintaxis del comando más adelante en este capítulo.

Al seleccionarse el texto, el navegador busca el nombre del *host* usando DNS. Armado ahora con la dirección IP del *host,* el navegador establece una conexión TCP con el *host*, y envía por esa conexión el nombre de archivo usando el protocolo especificado. Lotería. La página regresa.

Este esquema de URL es abierto en el sentido de que es directo hacer que los navegadores utilicen múltiples protocolos para obtener diferentes tipos de recursos. De hecho, se han definido los URLs de varios otros protocolos comunes. En la figura 7-24 se listan formas ligeramente simplificadas de los más comunes.

Examinemos con brevedad la lista. El protocolo *http* es el lenguaje nativo de Web, el que hablan los servidores Web. **HTTP** significa **Protocolo de Transferencia de Hipertexto.** Lo examinaremos con mayor detalle, más adelante en este capítulo.

Nombre	Usado para	Ejemplo
http	Hipertexto (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Archivo local	file:///usr/suzanne/prog.c
news	Grupo de noticias	news:comp.os.minix
news	Artículo	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Envío de correo electrónico	mailto:JohnUser@acm.org
telnet	Inicio de sesión remota	telnet://www.w3.org:80

Figura 7-24. Algunos URLs comunes.

El protocolo *ftp* se usa para acceder a archivos mediante FTP, el protocolo de transferencia de archivos de Internet. FTP tiene más de dos décadas de existencia y está bien atrincherado. Numerosos servidores FTP de todo el mundo permiten que gente de cualquier lugar de Internet establezca una sesión y descargue los archivos colocados en el servidor FTP. Web no cambia esto; simplemente hace más sencilla la obtención de archivos mediante FTP, puesto que FTP tiene una interfaz un tanto arcaica (pero es más poderoso que HTTP; por ejemplo, permite que un usuario en la máquina *A* transfiera un archivo de la máquina *B* a la máquina *C*).

Es posible acceder a un archivo local como página Web, ya sea usando el protocolo *file* o, más sencillamente, con sólo nombrarlo. Este enfoque es parecido a usar FTP pero no requiere un servidor. Por supuesto que sólo funciona con archivos locales.

Mucho antes de que existiera Internet, ya existía el sistema de noticias USENET, el cual consiste en 30,000 grupos de noticias en los que millones de personas discuten una variedad de temas publicando y leyendo artículos relacionados con ese tema. El protocolo de noticias puede utilizarse para llamar a un artículo de noticias como si fuera una página Web. Esto significa que un navegador Web es un lector de noticias. De hecho, muchos navegadores tienen botones o elementos de menú para que la lectura de noticias USENET sea más fácil que utilizar los lectores de noticias estándar.

Se soportan dos formatos para el protocolo *news.* El primero especifica un grupo de noticias y puede usarse para obtener una lista de artículos de un sitio de noticias preconfigurado. El segundo requiere que se indique el identificador de un artículo específico, en este caso *AA0134223112@cs.utah.edu.* A continuación el navegador obtiene el artículo dado de su sitio de noticias preconfigurado usando **NNTP (Protocolo de Transferencia de Noticias en Red).** No estudiaremos NNTP en este libro, pero se basa ligeramente en SMTP y tiene un estilo similar.

El protocolo *gopher* corresponde al sistema Gopher, que se diseñó en la Universidad de Minnesota y fue bautizado por los equipos atléticos de la escuela, los Golden Gophers (también es una expresión en jerga que significa "*go for*", es decir, ve y trae). Gopher es más antiguo que Web por varios años. Se trata de un esquema de recuperación de información, conceptualmente parecido a

la Web misma, pero que sólo maneja texto y no imágenes. Ahora es esencialmente obsoleto y no se utiliza con mucha frecuencia.

Los dos últimos protocolos en realidad no obtienen páginas Web, y no son reconocidos por todos los navegadores, pero de todas maneras son útiles. El protocolo *mailto* permite a los usuarios enviar correo electrónico desde un navegador Web. El procedimiento es hacer clic en el botón OPEN y especificar un URL que consista en *mailto:* seguido de la dirección de correo electrónico del destinatario. La mayoría de los navegadores responderá iniciando un programa de correo electrónico con la dirección y algunos de los campos de encabezado ya llenos.

El protocolo telnet sirve para establecer una conexión en línea con una máquina remota. Se utiliza de la misma manera que el programa telnet, lo cual no es sorprendente, puesto que la mayoría de los navegadores simplemente llama al programa telnet como aplicación auxiliar.

En pocas palabras, los URLs se han diseñado no sólo para permitir que los usuarios naveguen por Web, sino para que también utilicen FTP, noticias, Gopher, correo electrónico y telnet; de este modo los programas especializados de interfaz de usuario para esos otros servicios son innecesarios pues casi todos los accesos a Internet se integran en un solo programa, el navegador Web. Si no fuera por el hecho de que este esquema fue diseñado por un investigador de física, podría pensarse fácilmente que lo creó el departamento de publicidad de una compañía de software.

A pesar de todas estas agradables propiedades, el uso creciente de Web ha sacado a la luz una debilidad inherente del esquema URL. Un URL apunta a un *host* específico. En el caso de páginas a las que se hace referencia constante, sería deseable tener varias copias muy distantes, para reducir el tráfico de la red. El problema es que los URLs no proporcionan ninguna manera de referirse a una página sin decir de manera simultánea dónde está. No hay manera de decir: "quiero la página xyz y no me importa de dónde la traigas". Para resolver este problema y hacer posible la duplicación de páginas, la IETF está trabajando en un sistema de **URNs** (**Nombres Universales de Recursos**)**.** Un URN puede considerarse como un URL generalizado. Este tema aún es el objetivo de la investigación, aunque en el RFC 2141 se da una sintaxis propuesta.

Sin estado y cookies

Como hemos visto en varias ocasiones, Web básicamente no tiene estado. No existe el concepto de inicio de sesión. El navegador envía una solicitud a un servidor y obtiene un archivo. A continuación el servidor olvida que ha visto alguna vez a ese cliente en particular.

Al inicio, cuando Web sólo se utilizaba para recuperar documentos disponibles públicamente, este modelo era adecuado. Pero a medida que Web comenzó a adquirir otras funciones, surgieron problemas. Por ejemplo, algunos sitios Web requieren que los clientes se registren (y, con probabilidad, que paguen dinero) para poder utilizarlos. Esto da lugar a la pregunta de cómo los servidores pueden distinguir entre las solicitudes de usuarios registrados y las demás. Un segundo ejemplo es el comercio electrónico. Si un usuario navega en una tienda electrónica y coloca artículos en su carrito de compras de vez en cuando, ¿de qué manera el servidor mantiene un registro del contenido del carrito? Un tercer ejemplo son los portales Web personalizados, como Yahoo.

Los usuarios pueden establecer una página de inicio detallada que contenga sólo la información que desean (por ejemplo, sobre sus acciones y sus equipos de deportes favoritos), pero, ¿de qué manera puede el servidor desplegar la página correcta si no sabe quién es el usuario?

A primera vista, uno podría pensar que los servidores podrían seguir la pista de los usuarios al ver sus direcciones IP. Sin embargo, esta idea no funciona. Primero que nada, muchos usuarios trabajan en computadoras compartidas, especialmente en compañías, y la dirección IP simplemente identifica a la computadora, no al usuario. Segundo, y todavía peor, muchos ISPs utilizan NAT, por lo que todos los paquetes salientes de todos los usuarios tienen la misma dirección IP. Desde el punto de vista del servidor, los miles de clientes de los ISPs utilizan la misma dirección IP.

Para resolver este problema, Netscape diseñó una técnica muy criticada llamada **cookies.** El nombre se deriva de la antigua jerga de programación en la que un programa llama a un procedimiento y obtiene algo similar que tal vez tenga que presentar posteriormente para conseguir que se realice algún trabajo. En este sentido, un descriptor de archivos UNIX o un identificador de objetos de Windows puede considerarse como una *cookie*. Las *cookies* se formalizaron posteriormente en el RFC 2109.

Cuando un cliente solicita una página Web, el servidor puede proporcionar información adicional junto con la página solicitada. Esta información puede incluir una *cookie*, que es un pequeño archivo (o cadena, de a lo mucho 4 KB). Los navegadores almacenan *cookies* ofrecidas en un directorio de *cookies* en el disco duro de la máquina del cliente, a menos que el usuario las haya deshabilitado. Las *cookies* son simplemente archivos o cadenas, no programas ejecutables. En principio, una *cookie* puede contener un virus, pero puesto que las *cookies* se tratan como datos, no hay una forma oficial de que los virus se ejecuten realmente y hagan daño. Sin embargo, siempre es posible que algún *hacker* saque provecho de un error de un navegador para causar la activación de dicho virus.

Una *cookie* puede contener hasta cinco campos, como se muestra en la figura 7-25. El *dominio* indica de dónde viene la *cookie*. Se supone que los navegadores verifican que los servidores no mientan acerca de su dominio. Cada dominio puede almacenar hasta 20 *cookies* por cliente. La *ruta* es la ruta en la estructura del directorio del servidor que identifica qué partes del árbol de archivos del servidor podrían utilizar la *cookie*. Por lo general es /, lo que significa el árbol completo.

Dominio	Ruta	Contenido	Expira	Seguro
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Sí
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

Figura 7-25. Algunos ejemplos de *cookies*.

El campo *Contenido* toma la forma *nombre* = *valor*. Tanto *nombre* como *valor* pueden ser lo que el servidor desee. Este campo es donde se almacena el contenido de la *cookie*.

El campo *Expira* especifica cuándo caduca la *cookie*. Si este campo está ausente, el navegador descarta la *cookie* cuando sale. Tal *cookie* se conoce como ***cookie* no persistente.** Si se proporciona

una hora y una fecha, se dice que la *cookie* es **persistente** y se mantiene hasta que expira. Las fechas de expiración se dan en la hora del Meridiano de Greenwich. Para eliminar una *cookie* del disco duro de un cliente, un servidor simplemente la envía nuevamente, pero con una fecha caducada.

Por último, el campo *Seguro* puede establecerse para indicar que el navegador podría simplemente regresar la *cookie* a un servidor seguro. Esta característica se utiliza para comercio electrónico, actividades bancarias y otras aplicaciones seguras.

Ya vimos cómo se adquieren las *cookies*, pero, ¿cómo se utilizan? Justo antes de que un navegador solicite una página a un sitio Web, verifica su directorio de *cookies* para ver si el dominio al que está solicitando la página ya colocó alguna *cookie*. De ser así, todas las *cookies* colocadas por ese dominio se incluyen en el mensaje de solicitud. Cuando el servidor las obtiene, puede interpretarlas de la forma que desee.

Examinemos algunos usos posibles para las *cookies*. En la figura 7-25, la primera *cookie* fue establecida por *toms-casino.com* y se utiliza para identificar al cliente. Cuando éste inicia una sesión la siguiente semana para despilfarrar más dinero, el navegador envía la *cookie* de forma que el servidor sepa quién es. Una vez que el servidor cuenta con el ID del cliente, puede buscar el registro de éste en una base de datos y utilizar esta información para construir una página Web apropiada para desplegar. Dependiendo de los hábitos conocidos de apuestas del cliente, esta página podría consistir en una mano de póquer, un listado de las carreras de caballos del día o una máquina tragamonedas.

La segunda *cookie* proviene de *joes-store.com*. El escenario aquí es un cliente que está vagando por una tienda en busca de cosas buenas que comprar. Cuando dicho cliente encuentra una ganga y hace clic en ella, el servidor crea una *cookie* que contiene el número de elementos y el código del producto y la regresa al cliente. Conforme el cliente vaga por la tienda, la *cookie* se regresa cada vez que se solicita una página. Conforme se acumulan más compras, el servidor las agrega a la *cookie*. En la figura, el carrito contiene tres elementos, el último de los cuales se requiere dos veces. Por último, cuando el cliente hace clic en PASAR A LA CAJA, la *cookie*, que ahora contiene la lista completa de compras, se envía junto con la solicitud. De esta forma el servidor sabe exactamente qué ha comprado.

La tercera *cookie* es para un portal Web. Cuando el cliente hace clic en un vínculo que lo lleva al portal, el navegador envía la *cookie*. Ésta le indica al portal que construya una página que contenga los precios de las acciones de Sun Microsystems y Oracle, así como los resultados de fútbol de los Jets de Nueva York. Puesto que una *cookie* puede tener un tamaño de hasta 4 KB, hay demasiado espacio para incluir preferencias más detalladas, como encabezados de los periódicos, el clima local, ofertas especiales, etcétera.

Las *cookies* también pueden utilizarse para el beneficio del servidor. Por ejemplo, suponga que un servidor desea llevar el registro de cuántos visitantes únicos ha tenido y cuántas páginas miró cada uno antes de dejar el sitio. Cuando llega la primera solicitud, no hay *cookie* acompañante, por lo que el servidor regresa una *cookie* que contiene *Counter = 1*. Los clics subsecuentes en ese sitio regresarán la *cookie* al servidor. Cada vez el contador se incrementa y se regresa al cliente. Al llevar un registro de los contadores, el servidor puede ver cuántas personas salieron del sitio después de ver la primera página, cuántos vieron dos páginas, y así sucesivamente.

Las *cookies* también han sido objeto de malos usos. En teoría, se supone que las *cookies* sólo deben regresar al sitio original, pero los *piratas informáticos* han aprovechado varios errores de los navegadores para capturar las *cookies* que no son para ellos. Desde que algunos de los sitios de comercio electrónico colocaron números de tarjetas de crédito en las *cookies*, los intentos de abuso han sido más evidentes.

Un uso controversial de las *cookies* es colectar de manera secreta información sobre los hábitos de navegación en Web de los usuarios. Funciona como se explica a continuación. Una agencia de publicidad, digamos, Sneaky Ads, contacta sitios Web grandes y paga a los dueños una cuota por colocar anuncios de los productos de sus clientes corporativos. En lugar de dar al sitio un archivo GIF o JPEG para que lo coloque en cada página, le proporciona un URL para dicho propósito. Cada URL que dicha agencia maneja contiene un número único en la parte del archivo, como

http://www.sneaky.com/382674902342.gif

Cuando un usuario visita por primera vez una página, *P*, que contiene un anuncio de éstos, el navegador obtiene el archivo HTML. A continuación el navegador inspecciona el archivo HTML y ve el vínculo al archivo de imagen en *www.sneaky.com*, por lo que envía un mensaje en el que solicita la imagen. Se regresa un archivo GIF que contiene un anuncio, junto con una *cookie* que contiene un ID de usuario único, 3627239101 en la figura 7-25. Sneaky registra el hecho de que el usuario con este ID visitó la página *P*. Esto es fácil de hacer puesto que se hace referencia al archivo solicitado (*382674902342.gif*) sólo en la página *P*. Por supuesto, el anuncio real puede aparecer en miles de páginas, pero cada vez con un nombre de archivo diferente. Sneaky probablemente cobre un par de centavos al fabricante del producto cada vez que envíe el anuncio.

Más tarde, cuando el usuario visite otra página Web que contenga cualquiera de los anuncios de Sneaky, después de que el navegador ha obtenido el archivo HTML del servidor, ve el vínculo a, digamos, *http://www.sneaky.com/493654919923.gif* y solicita ese archivo. Puesto que ya tiene una *cookie* del dominio *sneaky.com*, el navegador incluye una *cookie* de Sneaky que contiene el ID del usuario. Sneaky ahora conoce una segunda página que el usuario ha visitado.

A su debido tiempo, Sneaky puede construir un perfil completo de los hábitos de navegación del usuario, aunque éste nunca ha hecho clic en ninguno de los anuncios. Por supuesto, aún no tiene el nombre del usuario (aunque tiene su dirección IP, lo cual debe ser suficiente para deducir el nombre a partir de otras bases de datos). Sin embargo, si el usuario alguna vez proporciona su nombre a algún sitio que coopere con Sneaky, estará disponible un perfil completo junto con un nombre para venderlo a quien desee comprarlo. La venta de esta información puede ser lo suficientemente rentable para que Sneaky coloque más anuncios en más sitios Web y, por lo tanto, para que colecte más información. La parte más insidiosa de este negocio es que la mayoría de los usuarios desconocen por completo esta recolección de información y tal vez piensen que están a salvo porque no hacen clic en ninguno de los anuncios.

Y si Sneaky desea ser supersneaky, el anuncio no necesita ser el clásico. Un “anuncio” que conste de un color de fondo de un solo píxel (por lo que es invisible), funciona exactamente de la misma forma: requiere que el navegador obtenga la imagen gif de 1×1 píxeles y le envíe todas las *cookies* que se originan en el dominio del píxel.

Para mantener algo de su privacidad, algunos usuarios configuran sus navegadores para que rechacen las *cookies*. Sin embargo, esto puede darles problemas con los sitios Web legítimos que utilizan *cookies*. Para resolver este problema, algunas veces los usuarios instalan software que elimina *cookies*. Éstos son programas especiales que inspeccionan cada *cookie* entrante al momento del arribo y la aceptan o descartan, dependiendo de las opciones que el usuario haya establecido (por ejemplo, sobre qué sitios Web pueden ser confiables). Esto le da al usuario un control detallado sobre cuáles *cookies* se aceptan y cuáles se rechazan. Los navegadores modernos, como Mozilla (*www.mozilla.org*), han elaborado controles de usuario sobre *cookies* integradas.

7.3.2 Documentos Web estáticos

La base de Web es la transferencia de páginas Web desde el servidor al cliente. En la forma más simple, las páginas Web son estáticas, es decir, son simplemente archivos que se encuentran en algún servidor esperando a ser recuperados. En este sentido, incluso un vídeo es una página Web estática porque es sólo un archivo. En esta sección examinaremos en detalle las páginas Web estáticas. En la siguiente examinaremos el contenido dinámico.

HTML—Lenguaje de Marcado de Hipertexto

En la actualidad las páginas Web se escriben en un lenguaje llamado **HTML** (**Lenguaje de Marcado de Hipertexto**). HTML permite a los usuarios producir páginas Web que incluyen texto, gráficos y apuntadores a otras páginas Web. HTML es un lenguaje de marcado que sirve para describir cómo se van a formatear los documentos. El término "marcado" proviene de la época en que los correctores de estilo realmente marcaban los documentos para indicar a la imprenta —en aquellos tiempos, un humano— qué fuentes utilizar, y cosas por el estilo. Por lo tanto, los lenguajes de marcado contenían comandos explícitos para formatear. Por ejemplo, en HTML, significa **iniciar modo en negritas y** significa abandonar modo en negritas. La ventaja de un lenguaje de marcado sobre uno con marcado no explícito es que escribir un navegador para él es directo: el navegador simplemente tiene que entender los comandos de marcado. TeX y troff son ejemplos de otros lenguajes de marcado bien conocidos.

Al integrar todos los comandos de marcado dentro de cada archivo HTML y al estandarizarlos, se hace posible que cualquier navegador Web lea y reformatee cualquier página Web. La capacidad de reformatear las páginas Web tras su recepción es crucial porque una página pudo haberse producido en una ventana de 1600 × 1200 con colores de 24 bits pero tal vez se vaya a desplegar en una de 640 × 320 con colores de 8 bits.

A continuación daremos una breve introducción al HTML, sólo para dar una idea de lo que es. Aunque ciertamente es posible escribir documentos HTML con cualquier editor estándar (y mucha gente lo hace), también es posible usar editores HTML especiales o procesadores de texto que pueden hacer la mayoría del trabajo (pero que por lo mismo dan al usuario menos control sobre todos los detalles del resultado final).

Una página Web consiste en un encabezado y un cuerpo encerrado entre **etiquetas** (comandos de formateo) <html> y </html>, aunque la mayoría de los navegadores no se quejan si faltan estas etiquetas. Como puede verse en la figura 7-26(a), el encabezado está delimitado por las etiquetas <head> y </head>, y el cuerpo, por las etiquetas <body> y </body>. Los comandos dentro de las etiquetas se llaman **directivas.** La mayoría de las etiquetas HTML tiene este formato, es decir, <algo> para marcar el comienzo de algo, y </algo> para marcar su fin. La mayoría de los navegadores tienen un elemento de menú VIEW SOURCE o algo parecido. Su selección presenta el código fuente HTML de las páginas actuales, en lugar de su salida formateada.

Las etiquetas pueden escribirse tanto en minúsculas como en mayúsculas. Por lo tanto, <head> y <HEAD> significan la misma cosa, pero las nuevas versiones del estándar requieren el uso de minúsculas. La distribución real del documento HTML no es importante. Los analizadores ignoran los espacios extra y los retornos de carro, puesto que tienen que formatear el texto para acomodarlo en el área de presentación. En consecuencia, pueden agregar espacios virtuales a voluntad para hacer más legibles los documentos HTML, algo que la mayoría necesita con urgencia. Como consecuencia, no pueden usarse líneas en blanco para separar párrafos, puesto que simplemente se ignoran. Se requiere una etiqueta específica.

Algunas etiquetas tienen parámetros (nombrados), llamados **atributos.** Por ejemplo,

```
<img src="abc" alt="foobar">
```

es una etiqueta, <img>, que tiene el parámetro *SRC* establecido a *abc* y el parámetro *alt* a *foobar*. Para cada etiqueta, el estándar HTML da una lista de los parámetros permitidos, si los hay, y lo que significan. Puesto que cada parámetro se nombra, el orden en que se dan los parámetros no es de importancia.

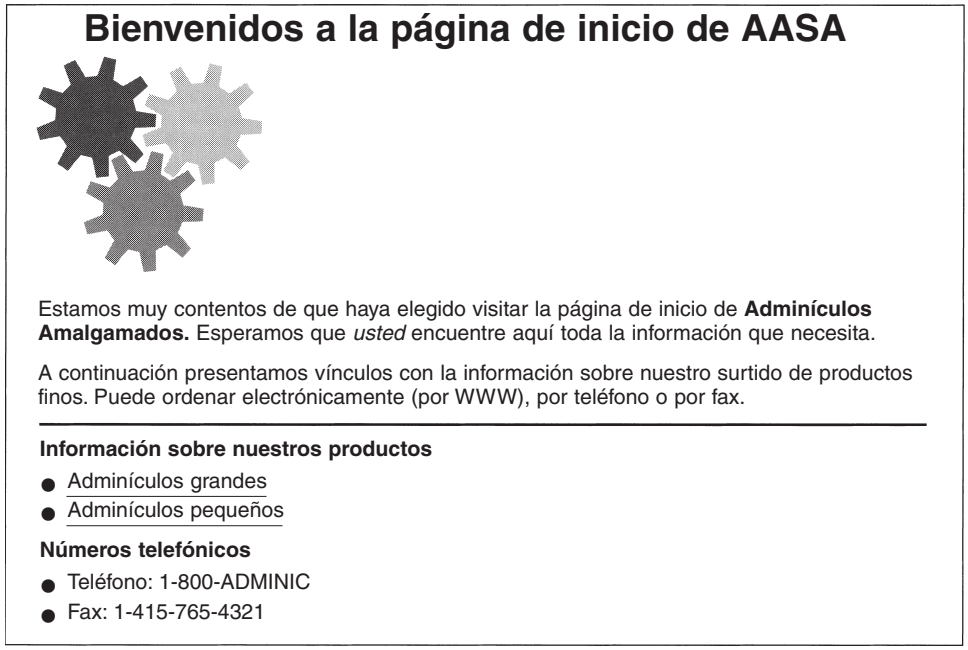
Técnicamente, los documentos HTML se escriben en el conjunto de caracteres Latin-1 del ISO 8859-1, pero para los usuarios cuyos teclados sólo reconocen ASCII, hay secuencias de escape para los caracteres especiales, como è. La lista de caracteres especiales se proporciona en el estándar. Todos estos caracteres comienzan con un signo & y terminan con un punto y coma. Por ejemplo,   produce un espacio, è produce è y é produce é. Puesto que <, >, y & tienen significados especiales, pueden expresarse sólo con sus secuencias de escape, < > y &, respectivamente.

El elemento principal del encabezado es el título, delimitado por <title> y </title>, pero pueden estar presentes también ciertos tipos de metainformación. El título mismo no se representa en la página. Algunos navegadores lo usan para rotular la ventana de la página.

Veamos ahora algunas de las otras características mostradas en la figura 7-26. Todas las etiquetas usadas en esa figura y algunas más se presentan en la figura 7-27. Los encabezados se generan con una etiqueta <h*n*>, donde *n* es un dígito del intervalo 1 a 6. Por lo tanto, <h1> es el encabezado más importante y <h6> es el menos importante. Es responsabilidad del navegador presentarlas de manera adecuada en la pantalla. Comúnmente, los encabezados de número menor se presentarán con una fuente más grande y gruesa. El navegador también puede seleccionar colores distintos para cada nivel de encabezado. Por lo común, los encabezados <h1> son grandes y en negritas, con cuando menos una línea en blanco arriba y abajo. En contraste, los encabezados <h2> tienen una fuente más pequeña y con menos espacio arriba y abajo.

```
<html>
<head> <title> ADMINÍCULOS AMALGAMADOS, S.A. </title> </head>
<body> <h1> Bienvenidos a la página de inicio de AASA </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
Estamos muy contentos de que haya elegido visitar la página de inicio de <b>
Adminículos Amalgamados </b>. Esperamos que <i> usted </i> encuentre aquí toda
la información que necesita.
<p> A continuación presentamos vínculos con la información sobre nuestro
surtido de productos finos. Puede ordenar electrónicamente (por WWW), por
teléfono o por fax. </p>
<hr>
<h2> Información sobre nuestros productos </h2>
<ul>
  <li> <a href="http://widget.com/products/big"> Adminículos grandes </a>
  <li> <a href="http://widget.com/products/little"> Adminículos pequeños </a>
</ul>
<h2> Números telefónicos </h2>
<ul>
  <li> Teléfono: 1-800-ADMINIC
  <li> Fax: 1-415-765-4321
</ul>
</body>
</html>
```

(a)



(b)

Figura 7-26. (a) HTML para una página Web de ejemplo. (b) Página con formato.

Las etiquetas <b> e <i> se usan para entrar en el modo de negritas y el de cursivas, respectivamente. Si el navegador no es capaz de presentar negritas y cursivas debe usar algún otro método para mostrarlas (por ejemplo, un color diferente para cada una o tal vez vídeo inverso).

HTML proporciona varios mecanismos para hacer las listas, entre ellas las listas anidadas. Las listas inician con <ul> u <ol>, y <li> marca el inicio de los elementos en ambos casos. La etiqueta <ul> comienza una lista desordenada. Los elementos individuales, que se marcan con la etiqueta <li> en el origen, aparecen con viñetas (•) al principio. Una variante de ese mecanismo es <ol>, que es para listas ordenadas. Cuando se usa esta etiqueta, los elementos <li> son numerados por el navegador. Aparte del uso de diferentes etiquetas de inicio y fin, <ul> y <ol> tienen la misma sintaxis y resultados parecidos.

Las etiquetas
, y <hr> indican límites entre secciones del texto. El formato preciso se puede determinar por la hoja de estilo (vea más adelante) asociada a la página. La etiqueta
 simplemente fuerza una línea nueva. Por lo común, los navegadores no insertan una línea en blanco tras una
. En contraste, comienza un párrafo y podría, por ejemplo, insertar una línea en blanco y posiblemente una sangría. (En teoría, existe para marcar el fin del párrafo, pero pocas veces se usa; la mayoría de los autores de HTML ni siquiera sabe que existe.) Por último, <hr> fuerza una nueva línea y dibuja una línea horizontal a lo ancho de la pantalla.

HTML permite la inclusión de imágenes en línea en una página Web. La etiqueta <img> especifica que se cargará una imagen en la posición actual de la página; puede tener varios parámetros. El parámetro *src* indica el URL de la imagen. El estándar HTML no especifica los formatos gráficos permitidos. En la práctica, todos los navegadores soportan archivos GIF y JPEG. Los navegadores pueden soportar otros formatos, pero esta extensión es un arma de doble filo. Si un usuario está acostumbrado a un navegador que reconoce, digamos, archivos BMP, puede incluir éstos en sus páginas Web y luego sorprenderse de que otros navegadores simplemente ignoren todo su maravilloso arte.

Otros parámetros de <img> son *align*, que controla la alineación de la imagen con respecto a la línea base del texto (*top, middle, bottom*), *alt*, que proporciona texto a usar en lugar de la imagen cuando el usuario ha inhabilitado las imágenes, e *ismap*, un indicador de que la imagen es un mapa activo (es decir, un elemento en el que se puede hacer clic).

Por último, llegamos a los hipervínculos, que utilizan las etiquetas <a> (ancla) y . Al igual que <img>, <a> tiene varios parámetros, entre los que se encuentran *href* (el URL) y *name* (el nombre del hipervínculo). El texto entre <a> y se despliega. Si se selecciona, se sigue el hipervínculo a una nueva página. También se permite poner una imagen <img> ahí, en cuyo caso hacer clic en la imagen también activa el hipervínculo.

Como ejemplo, considere el siguiente fragmento HTML:

```
<a href="http://www.nasa.gov"> Página de inicio de la NASA </a>
```

Cuando se presenta una página con este fragmento, lo que aparece en la ventana es

Página de inicio de la NASA

Etiqueta	Descripción
<html>...</html>	Declara que la página Web está descrita en HTML
<head>...</head>	Delimita el encabezado de la página
<title>...</title>	Delimita el título (no se presente en la página)
<body>...</body>	Delimita el cuerpo de la página
<h*n*>...</h*n*>	Delimita un encabezado de nivel *n*
...	Pone...en negritas
...	Pone...en cursivas
...	Centra...en la página horizontalmente
...	Corchetes de una lista desordenada (con viñetas)
...	Corchetes de una lista numerada
...	Corchetes de un elemento de una lista ordenada o numerada
 	Obliga salto de línea aquí
<p>	Inicia un párrafo
<hr>	Inserta una regla horizontal
<img src="...">	Carga una imagen aquí
...	Define un hipervínculo

Figura 7-27. Selección de etiquetas HTML comunes. Algunas tienen parámetros adicionales.

Si el usuario hace clic en este texto, el navegador de inmediato trae la página cuyo URL es *http://www.nasa.gov* y la presenta.

Como segundo ejemplo, considere

```
<a href="http://www.nasa.gov"> <img src="shuttle.gif" alt="NASA"> </a>
```

Al presentarse, esta página muestra una fotografía (por ejemplo, del transbordador espacial). Al hacer clic en la foto se pasa a la página de inicio de la NASA, igual que al hacer clic en el texto subrayado del ejemplo anterior. Si el usuario inhabilita la presentación automática de imágenes, el texto NASA se presentará donde iría la fotografía.

La etiqueta <a> puede aceptar un parámetro *name* para plantar un hipervínculo, de modo que pueda hacerse referencia a él desde la página. Por ejemplo, algunas páginas Web comienzan con un índice de materias en el que se puede hacer clic. Al hacer clic en un elemento de dicho índice, el usuario salta a la sección correspondiente de la página.

HTML sigue evolucionando. HTML 1.0 y HTML 2.0 no tenían tablas, pero éstas se agregaron en HTML 3.0. Una tabla HTML consiste en una o más filas, cada una de las cuales contiene una o más **celdas.** Éstas pueden contener una gama amplia de material, como texto, figuras, iconos, fotografías e incluso otras tablas. Es posible combinar las celdas a fin de que, por ejemplo, un encabezado pueda abarcar varias columnas. Los autores de páginas tienen control limitado sobre el diseño —incluyendo la alineación, los estilos de los bordes y los márgenes de las celdas—, pero los navegadores tienen la palabra final al representar las tablas.

En la figura 7-28(a) se lista una definición de tabla HTML y en la figura 7-28(b) se muestra una posible representación. Este ejemplo simplemente muestra algunas de las características básicas de las tablas HTML. Las tablas se inician mediante la etiqueta <table>. Es posible proporcionar información adicional para describir las propiedades generales de la tabla.

La etiqueta <caption> puede utilizarse para proporcionar el título de una figura. Cada fila comienza con una etiqueta <tr> (fila de tabla). Las celdas individuales se marcan como <th> (encabezado de tabla) o <td> (datos de tabla). La distinción se realiza para permitir que los navegadores utilicen diferentes representaciones para ellas, como lo hemos hecho en este ejemplo.

En la tablas también se permiten varios atributos. Incluyen formas de especificar alineaciones de celda horizontales y verticales, texto justificado dentro de una celda, bordes, grupos de celdas, unidades y más.

En HTML 4.0 se agregaron nuevas características. Éstas incluyen características de accesibilidad para usuarios discapacitados, incrustación de objetos (una generalización de la etiqueta <img> de manera que otros objetos también puedan ser incrustados en las páginas), soporte para lenguajes de secuencias de comandos (para permitir contenido dinámico), y más.

Cuando un sitio Web es complejo, y consiste en muchas páginas creadas por diversos autores que trabajan para la misma compañía, con frecuencia es deseable tener una forma de evitar que páginas diferentes tengan apariencias distintas. Este problema puede resolverse utilizando **hojas de estilo.** Con éstas, las páginas individuales ya no utilizan más estilos físicos, como negritas e itálicas. En su lugar, los autores de páginas utilizan estilos lógicos, como <dn> (define), (énfasis débil), (énfasis fuerte) y <var> (variables de programa). Los estilos lógicos se definen en la hoja de estilo, que se indica al principio de cada página. De esta manera, todas las páginas tienen el mismo estilo, y si el Webmaster decide cambiar de itálicas de 14 puntos en azul a negritas de 18 puntos en rosa, todo lo que necesita es cambiar una definición para convertir todo el sitio Web. Una hoja de estilo se puede comparar con un archivo #include de un programa C: al cambiar una definición de macro ahí, se cambia esa definición de todos los archivos de programa que incluyen el encabezado.

Formularios

HTML 1.0 básicamente era de un solo sentido. Los usuarios podían llamar las páginas desde los proveedores de información, pero era difícil enviar información en el otro sentido. A medida que más y más organizaciones comerciales comenzaron a utilizar Web, creció la demanda del tráfico de dos vías. Por ejemplo, muchas compañías querían tener la capacidad de tomar pedidos de productos mediante sus páginas Web, los proveedores de software querían distribuir software por medio de Web y hacer que sus clientes llenaran sus tarjetas de registro electrónicamente, y las compañías que ofrecían búsquedas en Web querían que sus clientes tuvieran la capacidad de indicar claves de búsqueda.

Estas demandas condujeron a la inclusión de **formularios** a partir de HTML 2.0. Los formularios contienen cuadros o botones que permiten a los usuarios proporcionar información o tomar decisiones, y después enviar dicha información al dueño de la página. Los formularios utilizan la etiqueta <input> para este propósito. Esta etiqueta tiene una variedad de parámetros para determinar

```
<html>
<head> <title> Una página de ejemplo con una tabla </title> </head>
<body>
<table border=1 rules=all>
<caption> Algunas diferencias entre las versiones de HTML </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th>Elemento <th>HTML 1.0 <th>HTML 2.0 <th>HTML 3.0 <th>HTML 4.0 </tr>
<tr> <th> Hipervínculos <td> x <td> x <td> x <td> x </tr>
<tr> <th> Imágenes <td> x <td> x <td> x <td> x </tr>
<tr> <th> Listas <td> x <td> x <td> x <td> x </tr>
<tr> <th> Mapas activos e imágenes <td>   <td> x <td> x <td> x </tr>
<tr> <th> Formularios <td>   <td> x <td> x <td> x </tr>
<tr> <th> Ecuaciones <td>   <td>   <td> x <td> x </tr>
<tr> <th> Barras de herramientas <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tablas <td>   <td>   <td> x <td> x </tr>
<tr> <th> Características de accesibilidad <td>   <td>   <td>  
     <td> x </tr>
<tr> <th> Incrustación de objetos <td>   <td>   <td>  
     <td> x </tr>
<tr> <th> Secuencias de comandos <td>   <td>   <td>   <td> x
     </tr>
</table>
</body>
</html>
```

(a)

Algunas diferencias entre las versiones de HTML

Elemento	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Hipervínculos	X	X	X	X
Imágenes	X	X	X	X
Listas	X	X	X	X
Mapas activos e imágenes		X	X	X
Formularios		X	X	X
Ecuaciones			X	X
Barras de herramientas			X	X
Tablas			X	X
Características de accesibilidad				X
Incrustación de objetos				X
Secuencias de comandos				X

Figura 7-28. (a) Una tabla HTML. (b) Una posible representación de esta tabla.

el tamaño, la naturaleza y el uso del cuadro presentado. Los formularios más comunes constan de campos en blanco para aceptar texto del usuario, casillas de verificación que pueden marcarse, mapas activos y botones de envío. El ejemplo de la figura 7-29 ilustra algunas de estas opciones.

Comencemos nuestro estudio de los formularios repasando este ejemplo. Como todos los formularios, éste se encierra entre las etiquetas <form> y </form>. El texto no delimitado por etiquetas simplemente se despliega. En un formulario se permiten todas las etiquetas normales (por ejemplo, <b>). Se usan tres tipos de cuadros de entrada en este formulario.

El primer cuadro de entrada está después del texto "Nombre". Dicho cuadro tiene 46 caracteres de longitud y espera que el usuario escriba una cadena, que luego se almacenará en la variable *customer* para procesarse posteriormente. La etiqueta <p> indica al navegador que despliegue el texto y los cuadros subsiguientes en otra línea, aunque haya espacio en la línea actual. Usando <p> y otras etiquetas de distribución, el autor de la página puede controlar la manera en que se ve el formulario en la pantalla.

La siguiente línea del formulario —la cual tiene 40 caracteres de longitud— solicita la dirección del usuario, también en una línea individual. Luego se encuentra una línea que solicita la ciudad, el estado y el país. Aquí no se usan etiquetas <p> entre los campos, por lo que el navegador desplegará todos en la misma línea, si caben. En lo que concierne al navegador, este párrafo simplemente contiene seis elementos: tres cadenas que alternan con tres cuadros. Todo esto lo despliega de manera lineal de izquierda a derecha, pasando a una línea nueva cada vez que la línea actual ya no puede contener el siguiente elemento. Por lo tanto, es concebible que en una pantalla de 1600×1200 las tres cadenas y sus cuadros correspondientes aparecerán en la misma línea, pero en una pantalla de 1024×768 tal vez se dividan en dos líneas. En el peor caso, la palabra "País" está al final de una línea y su cuadro al principio de la siguiente.

La siguiente línea solicita el número de tarjeta de crédito y su fecha de expiración. La transmisión de números de tarjeta de crédito por Internet sólo debe hacerse cuando se han tomado las medidas de seguridad adecuadas. En el capítulo 8 analizaremos algo de esto.

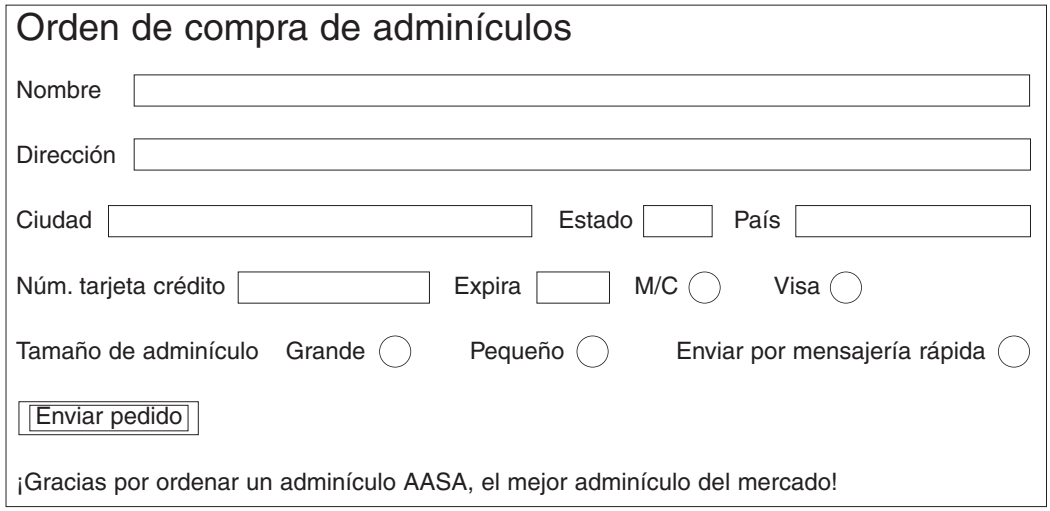
A continuación de la fecha de expiración encontramos una característica nueva: botones de opción. Éstos se usan cuando debe tomarse una decisión entre dos o más alternativas. Aquí el modelo es un radio de automóvil con media docena de botones para elegir estaciones. El navegador presenta estos botones de forma que permite que el usuario los seleccione y deseleccione haciendo clic en ellos (o usando el teclado). Al hacer clic en uno de los botones, se desactivan todos los demás del mismo grupo. La presentación visual depende del navegador. El tamaño del adminículo también usa dos botones de opción. Los dos grupos se distinguen por su campo *name*, no por el alcance estático usando algo como <radiobutton>...</radiobutton>.

Los parámetros de *value* sirven para indicar el botón de opción en el que se ha hecho clic. Dependiendo de las opciones de tarjeta de crédito que haya seleccionado el usuario, a la variable *cc* se le asignará la cadena "mastercard" o la cadena "visacard".

Después de los dos grupos de botones de opción, llegamos a la opción de envío, representada por un cuadro de tipo *checkbox*, que puede estar activo o inactivo. A diferencia de los botones de opción, de los que sólo se puede seleccionar uno del grupo, cada cuadro de tipo *checkbox* puede estar activo o inactivo, independientemente de los demás. Por ejemplo, al ordenar una pizza mediante la página Web de Electropizza, el usuario puede seleccionar sardinas *y* cebolla *y* piña

```
<html>
<head> <title> FORMULARIO DE PEDIDO DE CLIENTES AASA </title> </head>
<body>
<h1> Orden de compra de adminículos </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Nombre <input name="customer" size=46> </p>
<p> Dirección <input name="address" size=40> </p>
<p> Ciudad <input name="city" size=20> Estado <input name="state" size =4>
País <input name="country" size=10> </p>
<p> Núm. tarjeta crédito <input name="cardno" size=10>
Expira <input name="expires" size=4>
mastercard <input name="cc" type=radio value="mastercard">
visacard <input name="cc" type=radio value="visacard"> </p>
<p> Tamaño de adminículo Grande <input name="product" type=radio
    value="expensive">
Pequeño <input name="product" type=radio value="cheap">
Enviar por mensajería rápida <input name="express" type=checkbox> </p>
<p><input type=submit value="Enviar pedido"> </p>
¡Gracias por ordenar un adminículo AASA, el mejor adminículo del mercado!
</form>
</body>
</html>
```

(a)



(b)

Figura 7-29. (a) Código HTML para un formulario de pedido. (b) Página formateada.

(si se atreve), pero no puede seleccionar pequeña *y* mediana *y* grande para la misma pizza. Los ingredientes de la pizza se representarían con tres cuadros independientes del tipo *checkbox*, mientras que el tamaño de la pizza sería un grupo de botones de opción.

Como nota, en las listas muy largas en las que debe hacerse una selección, los botones de opción son poco prácticos. Por lo tanto, se proporcionan las etiquetas <select> y </select> para delimitar una lista de alternativas, pero con la semántica de los botones de opción (a menos que se dé el parámetro *multiple*, en cuyo caso la semántica es la de las casillas de verificación). Algunos navegadores presentan los elementos entre <select> y </select> como menú desplegable.

Hemos visto dos de los tipos integrados de la etiqueta <input>: *radio* y *checkbox*. De hecho, ya hemos visto también un tercero: *text*. Como este tipo es el predeterminado, no nos molestamos en incluir el parámetro *type = text*, pero podríamos haberlo hecho. Otros dos tipos son *password* y *textarea*. Un cuadro *password* es igual a uno *text*, excepto que los caracteres no se despliegan cuando se escriben. Un cuadro *textarea* es igual a un cuadro *text*, excepto que puede contener varias líneas.

Regresando al ejemplo de la figura 7-29, ahora nos topamos con un ejemplo de botón de envío. Al hacer clic en él, la información del usuario introducida en el formulario se envía a la máquina que proporcionó dicho formulario. Al igual que los demás tipos, *submit* es una palabra reservada que el navegador entiende. La cadena *value* es la etiqueta del botón, la cual se despliega. Todos los cuadros pueden tener valores; únicamente necesitamos esa característica aquí. En el caso de los cuadros *text*, el contenido del campo *value* se presenta junto con el formulario, pero el usuario puede modificarlo o borrarlo. Los cuadros *checkbox* y *radio* también pueden inicializarse, pero con un campo llamado *checked* (puesto que *value* simplemente da el texto, pero no indica una selección preferida).

Cuando el usuario hace clic en el botón de envío, el navegador empaca la información recolectada en una línea larga y la regresa al servidor para que la procese. El & se utiliza para separar campos y + para representar un espacio. Para nuestro formulario de ejemplo, la línea podría parecerse al contenido de la figura 7-30 (aquí se divide en tres líneas porque la página no es lo suficientemente ancha):

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&
product=cheap&express=on
```

Figura 7-30. Una posible respuesta del navegador al servidor con información introducida por el usuario.

La cadena regresaría al servidor como una línea, no tres. Si no se selecciona un *checkbox*, se omite de la cadena. Es responsabilidad del servidor darle sentido a esta cadena. Más adelante en este capítulo analizaremos cómo puede hacerse esto.

XML y XSL

HTML, con o sin formularios, no proporciona estructura alguna para las páginas Web. Además, mezcla el contenido con el formato. Conforme el comercio electrónico y otras aplicaciones se vuelven más comunes, hay una necesidad cada vez mayor de dar estructura a las páginas Web y de separar el contenido del formato. Por ejemplo, un programa que busca en Web el mejor precio de algún libro o CD necesita analizar muchas páginas Web en busca del título y precio del elemento. Con las páginas Web en HTML, es muy difícil que un programa averigüe en dónde están el título y el precio.

Por esta razón, el W3C ha mejorado el HTML para permitir que las páginas Web tengan estructura para su procesamiento automatizado. Para este propósito se han desarrollado dos nuevos lenguajes. El primero, **XML (Lenguaje de Marcado Extensible),** describe el contenido Web de una forma estructurada y el segundo, **XSL (Lenguaje de Hojas de estilo Extensible),** describe el formato independientemente del contenido. Estos dos lenguajes son temas extensos y complicados, por lo que nuestra breve introducción sólo rasca la superficie, pero debe darle una idea de cómo funcionan.

Considere el documento XML de ejemplo que se muestra en la figura 7-31. Define una estructura llamada book_list, que es una lista de libros. Cada libro tiene tres campos: el título, autor y año de publicación. Estas estructuras son muy sencillas. Está permitido tener estructuras con campos repetidos (por ejemplo, varios autores), opcionales (como el título del CD-ROM incluido) y alternativos (por ejemplo, el URL de una librería si el libro está en venta o uno de un sitio de subastas si se encuentra ahí debido a que es el último ejemplar).

En este ejemplo, cada uno de los tres campos es una entidad indivisible, pero sí está permitido dividirlos aún más. Por ejemplo, el campo de autor pudo haberse hecho como se muestra a continuación para proporcionar un control detallado sobre la búsqueda y el formato:

```
<author>
   <first_name> Andrew </first_name>
   <last_name> Tanenbaum </last_name>
</author>
```

Cada campo puede dividirse en subcampos y en subsubcampos, y así sucesivamente.

Todo lo que hace el archivo de la figura 7-31 es definir una lista de libros que contiene tres libros. No dice nada sobre cómo desplegar la página Web en la pantalla. Para proporcionar la información de formato, necesitamos un segundo archivo, *book_list.xsl*, que contiene la definición XSL. Este archivo es una hoja de estilo que indica cómo desplegar la página. (Hay alternativas a las páginas de estilo, como una forma de convertir XML en HTML, pero están más allá del alcance de este libro.)

En la figura 7-32 se muestra un archivo XSL para formatear el de la figura 7-31. Después de algunas declaraciones necesarias, entre ellas el URL del estándar XSL, el archivo contiene etiquetas que comienzan con <html> y <body>. Éstas definen el inicio de la página Web, de la forma

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>

<book>
  <title> Redes de computadoras, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2003 </year>
</book>

<book>
  <title> Sistemas operativos modernos, 2/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 2001 </year>
</book>

<book>
  <title> Organización estructurada de computadoras, 4/e </title>
  <author> Andrew S. Tanenbaum </author>
  <year> 1999 </year>
</book>

</book_list>
```

Figura 7-31. Una página Web simple en XML.

común. Después está la definición de una tabla, que incluye los encabezados para las tres columnas. Observe que además de las etiquetas <th> también hay etiquetas </th>, algo con lo que no nos molestaremos por el momento. Las especificaciones XML y XSL son mucho más estrictas que la especificación HTML. Dichas especificaciones declaran que es obligatorio rechazar archivos incorrectos sintácticamente, aunque el navegador pueda determinar lo que quiso decir el diseñador Web. Un navegador que acepta un archivo XML o XSL incorrecto sintácticamente y repara los errores él mismo no se apega a las especificaciones y se rechazará en una prueba de conformidad. Sin embargo, los navegadores pueden determinar con precisión el error. Esta medida algo draconiana es necesaria para tratar con el inmenso número de páginas Web mal diseñadas que existen actualmente.

La instrucción

```
<xsl:for-each select="book_list/book">
```

es análoga a una instrucción for de C. Causa que el navegador itere por el cuerpo del ciclo (el cual termina con <xsl:for-each>), una iteración por cada libro. Cada iteración envía cinco líneas: <tr>, el título, autor y año, y </tr>. Después del ciclo, se envían las etiquetas de cierre </body> y </html>. El

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0">
<xsl:template match="/">
<html>
<body>
<table border="2">
 <tr>
  <th> Título</th>
  <th> Autor</th>
  <th> Año </th>
 </tr>

 <xsl:for-each select="book_list/book">
 <tr>
  <td> <xsl:value-of select="title"/> </td>
  <td> <xsl:value-of select="author"/> </td>
  <td> <xsl:value-of select="year"/> </td>
 </tr>
 </xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Figura 7-32. Una hoja de estilo XSL.

resultado de que el navegador interprete esta hoja de estilo es el mismo que si la página Web contuviera la tabla en línea. Sin embargo, en este formato los programas pueden analizar el archivo XML y encontrar fácilmente libros publicados después del 2000, por ejemplo. Vale la pena enfatizar que aunque nuestro archivo XSL contenía una clase de ciclo, las páginas Web en XML y XSL aún son estáticas, al igual que las páginas HTML, debido a que simplemente contienen instrucciones para que el navegador despliegue la página, tal como lo hacen las páginas HTML. Por supuesto, para utilizar XML y XSL, el navegador debe tener la capacidad de interpretarlos, aunque la mayoría de los navegadores sí la tienen. Aún no es claro si XSL quitará el control a las hojas de estilo tradicionales.

No hemos mostrado cómo hacer esto, pero XML permite que el diseñador del sitio Web realice archivos de definición en los que las estructuras se definen por adelantado. Estos archivos pueden incluirse, por lo que pueden utilizarse para construir páginas Web complejas. Para obtener información adicional sobre ésta y otras características de XML y XSL, vea alguno de los muchos libros escritos sobre ellos. Dos ejemplos son (Livingston, 2002, y Williamson, 2001).

Antes de terminar el análisis de XML y XSL, vale la pena hacer algunos comentarios sobre la batalla ideológica entre el consorcio de WWW y la comunidad de diseñadores Web. El

objetivo original de HTML era especificar la *estructura* del documento, no su *apariencia*. Por ejemplo,

```
<h1> Fotos de Deborah </h1>
```

indica al navegador que resalte el encabezado, pero no dice nada sobre el tipo, color ni tamaño de fuente. Eso se dejó al navegador, el cual conocía las propiedades del despliegue (por ejemplo, con cuántos píxeles contaba). Sin embargo, muchos diseñadores de páginas Web deseaban control absoluto sobre la forma en que aparecían las páginas, por lo que se agregaron nuevas etiquetas al HTML para controlar la apariencia, como

```
<font face="helvetica" size="24" color="red"> Fotos de Deborah </font>
```

Además se agregaron formas para controlar la posición precisa en la pantalla. El problema con este enfoque es que no es portable. Aunque una página puede desplegarse perfectamente en el navegador en el que se desarrolló, en otro navegador u otra versión del mismo navegador o a una resolución de pantalla diferente, puede ser un problema. XML era en parte un intento por regresar al objetivo original de especificar sólo la estructura, no la apariencia de un documento. Sin embargo, XSL también se proporciona para manejar la apariencia. Sin embargo, ambos formatos pueden utilizarse de manera inadecuada. Puede apostarlo.

Además de describir páginas Web, XML también sirve para otros propósitos. Por ejemplo, se puede utilizar como lenguaje para la comunicación entre programas de aplicación. En particular, **SOAP** (**Protocolo Simple de Acceso a Objetos**) es una forma de realizar RPCs entre aplicaciones en forma independiente del lenguaje y de la aplicación. El cliente construye la solicitud como un mensaje XML y lo envía al servidor, utilizando el protocolo HTTP (descrito a continuación). El servidor envía una respuesta como un mensaje XML formateado. De esta manera, las aplicaciones de plataformas heterogéneas pueden comunicarse.

XHTML—Lenguaje de Marcado de Hipertexto Extendido

HTML sigue evolucionando para satisfacer las nuevas demandas. Muchas personas de la industria sienten que en el futuro la mayoría de los dispositivos habilitados para Web no serán PCs, sino dispositivos tipo PDA de mano inalámbricos. Estos dispositivos tienen memoria limitada para navegadores grandes llenos de heurística que tratan de alguna manera de lidiar con las páginas Web incorrectas sintácticamente. Por lo tanto, el siguiente paso después de HTML 4 es un lenguaje muy exigente. Se llama **XHTML** (**Lenguaje de Marcado de Hipertexto Extendido**) y no HTML 5 porque es esencialmente HTML 4 reformulado en XML. Con esto queremos decir que las etiquetas como <h1> no tienen significado intrínseco. Para obtener el efecto HTML 4, se necesita una definición en el archivo XSL. XHTML es el nuevo estándar Web y se debe utilizar para todas las páginas Web nuevas a fin de alcanzar la máxima portabilidad entre plataformas y navegadores.

Hay seis diferencias mayores y una variedad de diferencias menores entre XHTML y HTML 4. Veamos primero las principales diferencias. Primero, las páginas XHTML y los navegadores

deben apegarse estrictamente al estándar. No más páginas Web de mala calidad. Esta propiedad se heredó de XML.

Segundo, todas las etiquetas y los atributos deben estar en minúsculas. Las etiquetas como <HTML> no son válidas en XHTML. Ahora el uso de etiquetas como <html> es obligatorio. De manera similar, <img SRC="ima001.jpg"> también está prohibido porque contiene un atributo en mayúsculas.

Tercero, ahora se requiere el uso de etiquetas de cierre, incluso para </p>. Las etiquetas que no tienen etiqueta de cierre natural, como
, <hr> e <img>, deben tener una diagonal antes del paréntesis angular de cierre ">", por ejemplo

```
<img src="ima001.jpg" />
```

Cuarto, los atributos deben estar encerrados entre comillas. Por ejemplo,

```
<img SRC="ima001.jpg" height=500 />
```

ya no se permite. El número 500 debe estar encerrado entre comillas, de la misma forma que el nombre del archivo JPEG, aunque 500 sea un número.

Quinto, las etiquetas deben estar anidadas de manera apropiada. En el pasado, el anidamiento correcto no era requerido siempre y cuando el estado final alcanzado fuera el correcto. Por ejemplo,

```
<center> <b> Fotos de vacaciones </center> </b>
```

era legal. En XHTML no lo es. Las etiquetas deben cerrarse en el orden inverso en el que se abrieron.

Sexto, cada documento debe especificar su tipo de documento. Este criterio se aplicó en la figura 7-32. Para un análisis de todos los cambios, mayores y menores, vea *www.w3.org.*

7.3.3 Documentos Web dinámicos

Hasta ahora el modelo que hemos utilizado es el de la figura 6-6: el cliente envía un nombre de archivo al servidor, el cual regresa el archivo. En los primeros días de Web, todo el contenido era, de hecho, estático como éste (sólo archivos). Sin embargo, en los años recientes, cada vez más contenido es dinámico, es decir, se genera a solicitud, en lugar de almacenarlo en disco. La generación de contenido puede suceder ya sea en el servidor o en el cliente. A continuación examinaremos cada uno de estos casos.

Generación de páginas Web dinámicas en el servidor

Para ver por qué es necesaria la generación de contenido en el servidor, considere el uso de formularios, como se describió anteriormente. Cuando un usuario llena un formulario y hace clic en el botón de envío, se envía un mensaje al servidor con el contenido del formulario, junto con los campos que el usuario llenó. Este mensaje no es el nombre de un archivo a regresar. Lo que se necesita es proporcionar el mensaje a un programa o a una secuencia de comandos para que lo procesen. Por lo general, el procesamiento involucra el uso de la información proporcionada por el usuario para buscar un registro en una base de datos del disco del servidor y generar una página HTML personalizada para regresarla al cliente. Por ejemplo, en una aplicación de comercio

electrónico, después de que el usuario hace clic en *PASAR A LA CAJA*, el navegador regresa la *cookie* con el contenido del carrito de compras, pero es necesario invocar a algún programa o secuencia de comandos en el servidor para procesar la *cookie* y generar una página HTML en respuesta. La página HTML podría desplegar un formulario que contenga la lista de los elementos del carrito y la última dirección de envío conocida del usuario junto con una respuesta para verificar la información y especificar la forma de pago. En la figura 7-33 se muestran los pasos necesarios para procesar la información de un formulario HTML.

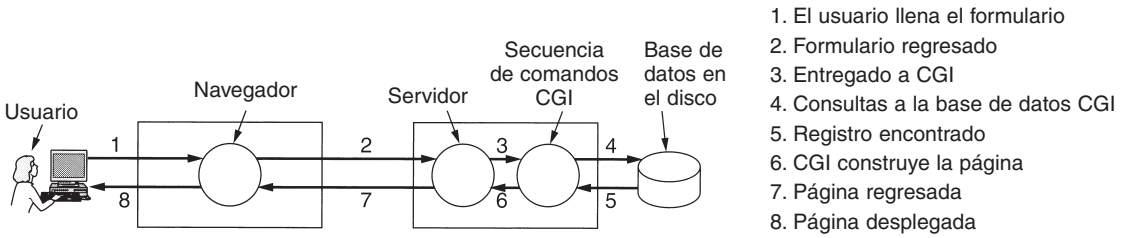


Figura 7-33. Pasos en el procesamiento de información de un formulario HTML.

La forma tradicional de manejar formularios y otras páginas Web interactivas es un sistema llamado **CGI** (**Interfaz de Puerta de Enlace Común**)**.** Es una interfaz estandarizada para permitir que los servidores Web hablen con los programas *back-end* y las secuencias de comandos puedan aceptar datos de entrada (por ejemplo, de formularios) y generar en respuesta páginas HTML. Por lo general, estos *back-ends* son secuencias de comandos escritas en el lenguaje Perl porque las secuencias de comandos de Perl son más fáciles y rápidas de escribir que los programas (si usted sabe programar en Perl). Por convención, se encuentran en un directorio llamado *cgi-bin*, que se muestra en el URL. Algunas veces se utiliza Python en lugar de Perl.

Como ejemplo de la forma en que funciona CGI, considere el caso de un producto de la compañía Productos Verdaderamente Geniales que viene sin una tarjeta de registro de garantía. En su lugar, se le dice al cliente que vaya a *www.pvg.com* para registrarse en línea. En esa página hay un hipervínculo que dice

Haga clic aquí para registrar su producto

Este vínculo apunta a una secuencia de comandos de Perl, digamos, *www.pvg.com/cgi-bin/reg.perl*. Cuando esta secuencia de comandos se invoca sin parámetros, regresa una página HTML que contiene el formulario de registro. Cuando el usuario llena el registro y hace clic en el botón de envío, se regresa un mensaje a esta secuencia de comandos que contiene los valores introducidos y que utiliza el estilo de la figura 7-30. A continuación la secuencia de comandos de Perl analiza los parámetros, crea una entrada en la base de datos para el nuevo cliente y regresa una página HTML que proporciona un número de registro y un número telefónico para el escritorio de ayuda. Ésta no es la única forma de manejar formularios, pero sí es común. Hay muchos libros acerca de cómo crear secuencias CGI y cómo programar en Perl. Algunos ejemplos son (Hanegan, 2001; Lash, 2002, y Meltzer y Michalski, 2001).

Las secuencias de comandos CGI no son la única forma de generar contenido dinámico en el servidor. Otra forma común es incrustar pequeñas secuencias de comandos dentro de páginas HTML y hacer que el servidor mismo sea quien las ejecute para generar la página. Un lenguaje popular para escribir estas secuencias de comandos es **PHP** (**Preprocesador de Hipertexto**)**.** Para utilizarlo, el servidor tiene que entender PHP (de la misma forma que un navegador tiene que entender XML para interpretar páginas Web escritas en XML). Por lo general, los servidores esperan páginas Web que contienen PHP para tener una extensión *php* en lugar de *html* o *htm*.

En la figura 7-34 se ilustra una pequeña secuencia de comandos PHP; debe funcionar con cualquier servidor que tenga instalado PHP. Contiene HTML normal, excepto por la secuencia de comandos PHP dentro de la etiqueta `<?php ... ?>`. Lo que hace es generar una página Web que indica lo que sabe sobre el navegador que la invoca. Normalmente, los navegadores envían alguna información junto con su solicitud (y cualesquier *cookies*) y esta información se coloca en la variable *HTTP_USER_AGENT*. Cuando este listado se coloca en un archivo *test.php* en el directorio WWW de la compañía ABCD, al escribir el URL *www.abcd.com/test.php* se producirá una página Web que indica al usuario cuál navegador, lenguaje y sistema operativo está utilizando.

```
<html>
<body>

<h2> Esto es lo que sé de ti </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

Figura 7-34. Una página HTML de ejemplo con PHP incrustado.

PHP es especialmente bueno para manejar formularios y es más sencillo que utilizar una secuencia de comandos CGI. Como ejemplo de cómo funciona con formularios, considere la figura 7-35(a). Ésta contiene una página normal HTML con un formulario en ella. Lo único inusual es la primera línea que especifica que se va a invocar el archivo *action.php* para manejar los parámetros después de que el usuario ha llenado y emitido el formulario. La página despliega dos cuadros de texto, uno que solicita un nombre y otro que solicita una edad. Una vez que se han llenado los cuadros de texto y se ha enviado el formulario, el servidor analiza la cadena de tipo regresada de la figura 7-30, colocando el nombre en la variable *nombre* y la edad en la variable *edad*. Después comienza el procesamiento del archivo *action.php*, que se muestra como respuesta en la figura 7-35(b). Durante el procesamiento de este archivo, se ejecutan los comandos PHP. Si el usuario introdujo “Barbara” y “24” en los cuadros, el archivo HTML regresado será el que se muestra en la figura 7-35(c). Por lo tanto, el manejo de formularios con PHP se vuelve muy sencillo.

Aunque PHP es fácil de utilizar, realmente es un lenguaje de programación poderoso diseñado para interactuar entre Web y una base de datos del servidor. Tiene variables, cadenas, arreglos y la mayoría de las estructuras de control que se encuentran en C, pero la E/S es mucho más poderosa que *printf*. PHP es código *open source* y está disponible de manera gratuita.

```
<html>
<body>
<form action="action.php" method="post">
<p> Por favor introduzca su nombre: <input type="text" name="name"> </p>
<p> Por favor introduzca su edad: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Respuesta </h1>
Hola <?php echo $name; ?>.
Predicción: el siguiente año tendrá <?php echo $age + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Respuesta: </h1>
Hola Barbara.
Predicción: el siguiente año tendrá 25
</body>
</html>
```

(c)

Figura 7-35. (a) Una página Web que contiene un formulario. (b) Una secuencia de comandos PHP para manejar la salida del formulario. (c) Salida de la secuencia de comandos PHP cuando las entradas son "Barbara" y 24, respectivamente.

Se diseñó específicamente para trabajar bien con Apache, que también es código abierto y es el servidor Web más ampliamente utilizado del mundo. Para mayor información sobre PHP, vea (Valade, 2002).

Ya vimos dos formas diferentes para generar páginas HTML dinámicas: las secuencias de comandos CGI y el PHP incrustado. También hay una tercera técnica, llamada **JSP** (**Java Server Pages**), que es similar a PHP, excepto que la parte dinámica se escribe en el lenguaje de programación Java en lugar de en PHP. Las páginas que utilizan esta técnica tienen la extensión de archivo *jsp*. Una cuarta técnica, **ASP** (**Active Server Pages**), es la versión de Microsoft de PHP y JavaServer Pages. Para generar el contenido dinámico utiliza el lenguaje de secuencias de comandos propietario de Microsoft, Visual Basic Script. Las páginas que utilizan esta técnica tienen la extensión *asp*. Por lo general, la selección entre *PHP*, *JSP* y *ASP* tiene que ver más con políticas

(código abierto contra Sun contra Microsoft) que con la tecnología, puesto que estos tres lenguajes son más o menos similares.

La colección de tecnologías para generar contenido al vuelo algunas veces se llama **HTML dinámico.**

Generación de páginas Web dinámicas en el cliente

Las secuencias de comandos de CGI, PHP, JSP y ASP resuelven el problema de manejar formularios e interacciones con bases de datos en el servidor. Pueden aceptar información entrante de formularios, buscar información en una o más bases de datos y generar páginas HTML con los resultados. Lo que ninguno de ellos puede hacer es responder a los movimientos del ratón o interactuar de manera directa con los usuarios. Para esto es necesario tener secuencias de comandos incrustadas en páginas HTML que se ejecuten en la máquina cliente y no en el servidor. Comenzando con HTML 4.0, tales secuencias de comandos son posibles mediante la etiqueta <script>. El lenguaje de secuencias de comandos más popular para el cliente es **JavaScript,** por lo que ahora le daremos un vistazo.

JavaScript es un lenguaje de secuencias de comandos, inspirado por algunas ideas del lenguaje de programación Java. Definitivamente éste no es Java. Al igual que otros lenguajes de secuencias de comandos, es un lenguaje de muy alto nivel. Por ejemplo, en una sola línea de JavaScript es posible desplegar un cuadro de diálogo, esperar entrada de texto y almacenar la cadena resultante en una variable. Las características de alto nivel como éstas hacen que JavaScript sea ideal para diseñar páginas Web interactivas. Por otro lado, el hecho de que no está estandarizado y de que tiene más mutaciones que una mosca de fruta atrapada en una máquina de rayos X, hace extremadamente difícil escribir programas de JavaScript que funcionen en todas las plataformas, pero tal vez algún día se estabilice.

Como ejemplo de un programa de JavaScript, considere el de la figura 7-36. Al igual que el de la figura 7-35(a), despliega un formulario que pide un nombre y una edad, y después calcula cuántos años tendrá una persona el siguiente año. El cuerpo es casi el mismo que el del ejemplo de PHP, la diferencia principal es la declaración del botón de envío y su instrucción de asignación. Ésta indica al navegador que invoque la secuencia de comandos *response* mediante un clic del botón y que la pase al formulario como un parámetro.

Lo que es completamente nuevo aquí es la declaración de la función *response* de JavaScript del encabezado del archivo HTML, un área que se reserva normalmente para títulos, colores de fondo, etcétera. Esta función extrae el valor del campo *name* del formulario y lo almacena como una cadena en la variable *person*. También extrae el valor del campo *age*, lo convierte en un entero mediante el uso de la función *eval*, le agrega 1 y almacena el resultado en *years*. A continuación abre un documento para salida, utiliza el método *writeln* para realizar cuatro escrituras en dicho documento y, por último, lo cierra. Este documento es un archivo HTML, como puede advertirse a partir de las etiquetas HTML. A continuación el navegador despliega el documento en la pantalla.

Es muy importante hacer notar que si bien las figuras 7-35 y 7-36 son muy similares, se procesan de manera totalmente diferente. En la figura 7-35, después de que el usuario hace clic en el botón de envío, el navegador colecta la información en una cadena larga como la de la figura 7-30

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
  var person = test_form.name.value;
  var years = eval(test_form.age.value) + 1;
  document.open();
  document.writeln("<html> <body>");
  document.writeln("Hola" + person + ".<br>");
  document.writeln("Predicción: el siguiente año usted tendrá" +
    años + ".");
  document.writeln("</body> </html>");
  document.close();
}
</script>
</head>

<body>
<form>
Por favor, introduzca su nombre: <input type="text" name="name">
<p>
Por favor, introduzca su edad: <input type="text" name="age">
<p>
input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Figura 7-36. Uso de JavaScript para procesar un formulario.

y la regresa al servidor que envió la página. Éste ve el nombre del archivo PHP y lo ejecuta. La secuencia de comandos de PHP produce una nueva página HTML y ésta se regresa al navegador para que la despliegue. En la figura 7-36, cuando se hace clic en el botón de envío, el navegador interpreta una función de JavaScript que está contenida en la página. Todo el trabajo se lleva a cabo de manera local, dentro del navegador. No hay contacto con el servidor. Debido a esto, el resultado se despliega casi instantáneamente, mientras que con PHP, puede haber un retraso de algunos segundos antes de que el HTML resultante llegue al cliente. En la figura 7-37 se ilustra la diferencia entre las secuencias de comandos del servidor y las del cliente, además de los pasos involucrados. En ambos casos, los pasos numerados comienzan después de que se despliega el formulario. El paso 1 consiste en aceptar la entrada del usuario. A continuación se realiza el procesamiento de la entrada, lo cual es diferente en los dos casos.

Esta diferencia no significa que JavaScript sea mejor que PHP. Sus usos son completamente diferentes. PHP (y, por ende, JSP y ASP) se utiliza cuando es necesaria la interacción con una base de datos remota. JavaScript se utiliza cuando la interacción es con el usuario en la máquina

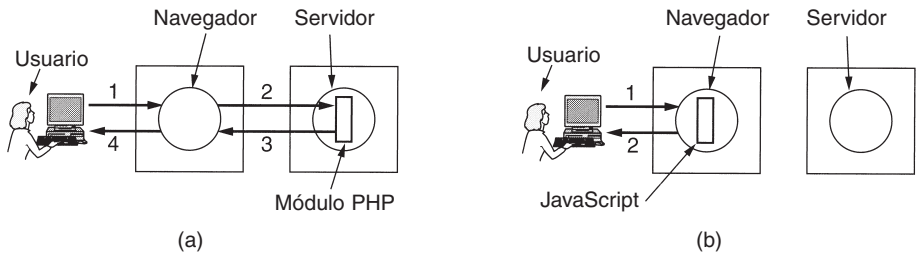


Figura 7-37. (a) Secuencias de comandos en el servidor con PHP. (b) Secuencias de comandos en el cliente con JavaScript.

cliente. Ciertamente es posible (y común) tener páginas HTML que utilicen PHP y JavaScript, aunque éstas no pueden hacer el mismo trabajo ni poseer, por supuesto, el mismo botón.

JavaScript es un lenguaje de programación completo, con todo el poder de C o Java. Tiene variables, cadenas, arreglos, objetos, funciones y todas las estructuras de control comunes. También tiene una gran cantidad de características específicas para páginas Web, entre las que se encuentran la capacidad para administrar ventanas y marcos, establecer y obtener *cookies*, manejar formularios e hipervínculos. En la figura 7-38 se proporciona un programa JavaScript que utiliza una función recursiva.

```
<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
  function factorial(n) {if (n == 0) return 1; else return n * factorial
    (n - 1);}
  var r = eval(test_form.number.value);    // r = escribe un argumento
  document.myform.mytext.value = "Aquí están los resultados.\n";
  for (var i = 1; i <= r; i++)             // imprime una línea de 1 a r
    document.myform.mytext.value += (i + "! = " + factorial(i) + "\n");
}
</script>
</head>

<body>
<form name="myform">
Please enter a number: <input type="text" name="number">
<input type="button" value="calcular la tabla de factoriales" onclick="
  response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>
```

Figura 7-38. Un programa JavaScript para calcular e imprimir factoriales.

JavaScript también puede rastrear el movimiento del ratón sobre objetos de la pantalla. Muchas páginas Web de JavaScript tienen la propiedad de que cuando el cursor del ratón se mueve sobre algún texto o imagen, algo pasa. Con frecuencia la imagen cambia o aparece de repente un menú. Este tipo de comportamiento es fácil de programar en JavaScript y produce páginas Web animadas. En la figura 7-39 se muestra un ejemplo.

```
<html>
<head>
<script language="javascript" type="text/javascript">
if (!document.myurl) document.myurl = new Array();
document.myurl[0] = "http://www.cs.vu.nl/ast/im/gato.jpg";
document.myurl[1] = "http://www.cs.vu.nl/ast/im/perro.jpg";
document.myurl[2] = "http://www.cs.vu.nl/ast/im/conejo.jpg";
function pop(m) {
  var urx = "http://www.cs.vu.nl/ast/im/gato.jpg";
  popupwin = window.open(document.myurl[m],"mywind","width=250,
    height=250");
}
</script>
</head>

<body>
<p> <a href="#" onMouseover="pop(0); return false;" > Gato </a> </p>
<p> <a href="#" onMouseover="pop(1); return false;" > Perro </a> </p>
<p> <a href="#" onMouseover="pop(2); return false;" > Conejo </a> </p>
</body>
</html>
```

Figura 7-39. Una página Web interactiva que responde al movimiento del ratón.

JavaScript no es la única forma de crear páginas Web altamente interactivas. Otro popular método es a través de **subprogramas** (*applets*). Éstos son pequeños programas de Java que se han compilado en instrucciones de máquina para una computadora virtual llamada **JVM** (**Máquina Virtual de Java**). Los subprogramas pueden incrustarse en páginas HTML (entre <applet> y </applet>) e interpretarse mediante navegadores con capacidad JVM. Debido a que los subprogramas de Java se interpretan en lugar de ejecutarse directamente, el intérprete de Java puede evitar que causen daños. Por lo menos en teoría. En la práctica, los escritores de subprogramas han encontrado un flujo infinito de errores en las bibliotecas de E/S de Java de los cuales aprovecharse.

La respuesta de Microsoft a los subprogramas de Java de Sun fue permitir que las páginas Web contuvieran **controles ActiveX,** que son programas compilados para lenguaje de máquina Pentium y ejecutados en el hardware. Esta característica los hace inmensamente rápidos y más flexibles que los subprogramas interpretados de Java porque pueden hacer cualquier cosa que un programa pueda hacer. Cuando Internet Explorer ve un control ActiveX en una página Web, lo descarga, verifica su identidad y lo ejecuta. Sin embargo, la descarga y ejecución de programas extraños aumenta los problemas de seguridad, lo cual veremos en el capítulo 8.

Puesto que casi todos los navegadores pueden interpretar los programas de Java y JavaScript, un diseñador que desee crear una página Web altamente interactiva puede elegir por lo menos de entre dos técnicas, y si la portabilidad a múltiples plataformas no es importante, también puede elegir ActiveX. Como regla general, los programas de JavaScript son fáciles de escribir, los subprogramas de Java se ejecutan muy rápido y los controles ActiveX se ejecutan todavía más rápido. Además, puesto que todos los navegadores implementan la misma JVM pero no todos implementan la misma versión de JavaScript, los subprogramas de Java son más portables que los programas de JavaScript. Para mayor información sobre JavaScript, hay muchos libros, cada uno con muchas páginas (por lo general, más de 1000). Algunos ejemplos son (Easttom, 2001; Harris, 2001, y McFedries, 2001).

Antes de dejar el tema del contenido dinámico Web, resumamos brevemente lo que hemos visto hasta ahora. Las páginas Web pueden generarse en la marcha mediante varias secuencias de comandos en la máquina servidora. Una vez que el navegador las recibe, las trata como páginas HTML normales y simplemente las despliega. Las secuencias de comandos pueden escribirse en Perl, PHP, JSP o ASP, como se muestra en la figura 7-40.

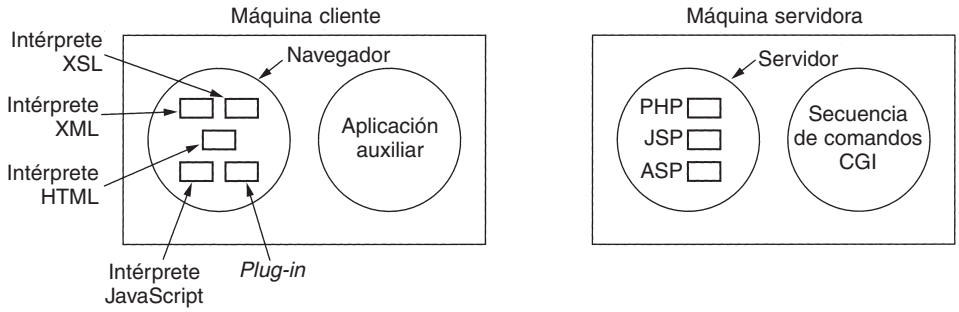


Figura 7-40. Las diversas formas de generar y desplegar contenido.

La generación de contenido dinámico también es posible en el cliente. Es posible escribir en XML las páginas Web y después convertirlas en HTML de acuerdo a un archivo XSL. Los programas de JavaScript pueden realizar cálculos arbitrarios. Por último, es posible utilizar los *plug-ins* y las aplicaciones auxiliares para desplegar contenido en varios formatos.

7.3.4 HTTP—Protocolo de Transferencia de Hipertexto

El protocolo de transferencia utilizado en World Wide Web es **HTTP** (**Protocolo de Transferencia de Hipertexto**)**.** Especifica cuáles mensajes pueden enviar los clientes a los servidores y qué respuestas obtienen. Cada interacción consiste en una solicitud ASCII, seguida por una respuesta tipo MIME del RFC 822. Todos los clientes y servidores deben obedecer este protocolo. Se define en el RFC 2616. En esta sección veremos algunas de sus propiedades más importantes.

Conexiones

La forma común en que un navegador contacta a un servidor es estableciendo una conexión TCP con el puerto 80 de la máquina del servidor, aunque este procedimiento no se requiere formalmente. El valor de utilizar TCP es que ni los navegadores ni los servidores tienen que preocuparse por los mensajes largos, perdidos o duplicados, ni por las confirmaciones de recepción. En la implementación de TCP se manejan todos estos aspectos.

En HTTP 1.0, una vez que se establecía la conexión, se enviaba una solicitud y se obtenía una respuesta. Después se liberaba dicha conexión. Este método era adecuado en un mundo en el que una página Web típica consistía por completo de texto HTML. Sin embargo, años más tarde la página Web promedio contenía una gran cantidad de iconos, imágenes, entre otras cosas, por lo que establecer una conexión TCP para transportar un solo icono era un método muy costoso.

Debido a lo anterior se diseñó HTTP 1.1, que soporta **conexiones persistentes.** Con ellas, es posible establecer una conexión TCP, enviar una solicitud y obtener una respuesta, y después enviar solicitudes adicionales y obtener respuestas adicionales. Al repartir la configuración y terminación de sesiones de TCP en múltiples solicitudes, la sobrecarga resultante de TCP es mucho menor por solicitud. También es posible enviar solicitudes en canalización, es decir, enviar la solicitud 2 antes de que la respuesta a la solicitud 1 haya llegado.

Métodos

Aunque HTTP se diseñó para utilizarlo en Web, se ha hecho intencionalmente más general que lo necesario con miras a las aplicaciones orientadas a objetos futuras. Por esta razón, se soportan otras operaciones, llamadas **métodos,** diferentes a las de solicitar una página Web. Esta generalidad es lo que permite que SOAP exista. Cada solicitud consiste en una o más líneas de texto ASCII, y la primera palabra de la primera línea es el nombre del método solicitado. En la figura 7-41 se listan los métodos integrados. Para acceder a objetos generales, también están disponibles métodos adicionales específicos de objetos. Los nombres son sensibles a mayúsculas y minúsculas, por lo que *GET* es un método válido y *get* no lo es.

El método *GET* solicita al servidor que envíe la página (con lo cual queremos decir objeto, en el caso más general, pero en la práctica se trata normalmente tan sólo de un archivo). La página está codificada de forma adecuada en MIME. La mayoría de las solicitudes a servidores Web son *GET*s. La forma común de *GET* es

```
GET nombredearchivo HTTP/1.1
```

donde *nombredearchivo* es el recurso (archivo) que se va a obtener y 1.1 es la versión del protocolo que se está utilizando.

El método *HEAD* simplemente solicita el encabezado del mensaje, sin la página real. Este método puede utilizarse para obtener la fecha de la última modificación de la página, para colectar información para propósitos de indización o simplemente para proporcionar un URL para validación.

Método	Descripción
GET	Solicita la lectura de una página Web
HEAD	Solicita la lectura del encabezado de una página Web
PUT	Solicita el almacenamiento de una página Web
POST	Inserta algo a un recurso con nombre (por ejemplo, una página Web)
DELETE	Elimina la página Web
TRACE	Repite la solicitud entrante
CONNECT	Reservado para uso futuro
OPTIONS	Consulta ciertas opciones

Figura 7-41. Los métodos de solicitud HTTP integrados.

El método *PUT* es lo inverso a *GET:* en lugar de leer la página, la escribe. Este método posibilita la construcción de una colección de páginas Web en un servidor remoto. El cuerpo de la solicitud contiene la página. Puede codificarse utilizando MIME, en cuyo caso las líneas que siguen al método *PUT* podrían incluir encabezados *Content-Type* y de autenticación, para probar que el invocador tiene permiso de realizar la operación solicitada.

El método *POST* es similar a *PUT*. También transporta un URL, pero en lugar de reemplazar los datos existentes, los nuevos datos se "insertan" en él en algún sentido generalizado. Enviar un mensaje a un grupo de noticias o agregar un archivo a un sistema de boletín de noticias son ejemplos de agregar en este contexto. En la práctica, ni *PUT* ni *POST* se utilizan mucho.

DELETE hace lo que usted espera: elimina la página. Al igual que con *PUT*, la autenticación y el permiso juegan un papel importante aquí. *DELETE* no siempre tendrá éxito, puesto que aunque el servidor HTTP remoto esté dispuesto a eliminar la página, el archivo subyacente puede tener un modo que prohíba al servidor HTTP modificarla o eliminarla.

El método *TRACE* es para la depuración. Indica al servidor que regrese la solicitud. Este método es útil cuando las solicitudes no se están procesando de manera correcta y el cliente desea saber cuál solicitud ha obtenido realmente el servidor.

El método *CONNECT* no se utiliza en la actualidad. Está reservado para uso futuro.

El método *OPTIONS* proporciona una forma para que el cliente consulte al servidor sobre sus propiedades o las de un archivo específico.

Cada solicitud obtiene una respuesta que consiste en una línea de estado, y posiblemente de información adicional (por ejemplo, toda o parte de una página Web). La línea de estado contiene un código de estado de tres dígitos que indica si la solicitud fue atendida, y si no, por qué. El primer dígito se utiliza para dividir las respuestas en cinco grupos mayores, como se muestra en la figura 7-42. En la práctica, los códigos 1xx no se utilizan con frecuencia. Los códigos 2xx significan que la solicitud se manejó de manera exitosa y que se regresa el contenido (si hay alguno). Los códigos 3xx indican al cliente que busque en otro lado, ya sea utilizando un URL diferente o en su propio caché (que se analiza posteriormente). Los códigos 4xx significan que la solicitud

Código	Significado	Ejemplos
1xx	Información	100 = el servidor está de acuerdo en manejar la solicitud del cliente
2xx	Éxito	200 = la solicitud es exitosa; 204 = no hay contenido
3xx	Redirección	301 = página movida; 304 = la página en caché aún es válida
4xx	Error del cliente	403 = página prohibida; 404 = página no encontrada
5xx	Error del servidor	500 = error interno del servidor; 503 = trata más tarde

Figura 7-42. Los grupos de respuesta del código de estado.

falló debido a un error del cliente, por ejemplo una solicitud inválida o una página no existente. Por último, los errores 5xx significan que el servidor tiene un problema, debido a un error en su código o a una sobrecarga temporal.

Encabezados de mensaje

A la línea de solicitud (por ejemplo, la línea con el método *GET*) le pueden seguir líneas adicionales que contienen más información. Éstas se llaman **encabezados de solicitud.** Esta información puede compararse con los parámetros de una llamada a procedimiento. Las respuestas también pueden tener **encabezados de respuesta.** Algunos encabezados pueden utilizarse en cualquier dirección. En la figura 7-43 se muestra una selección de los más importantes.

El encabezado *User-Agent* permite que el cliente informe al servidor sobre su navegador, sistema operativo y otras propiedades. En la figura 7-34 vimos que el servidor tenía mágicamente esta información y podía producirla a solicitud en una secuencia de comandos PHP. El cliente utiliza este encabezado para proporcionarle la información al servidor.

Los cuatro encabezados *Accept* indican al servidor lo que el cliente está dispuesto a aceptar en caso de que tenga un repertorio limitado de lo que es aceptable. El primer encabezado especifica los tipos MIME aceptados (por ejemplo, text/html). El segundo proporciona el conjunto de caracteres (por ejemplo, ISO-8859-5 o Unicode-1-1). El tercero tiene que ver con métodos de compresión (por ejemplo, gzip). El cuarto indica un idioma natural (por ejemplo, español). Si el servidor tiene la opción de páginas, puede utilizar esta información para proporcionar la que el cliente esté buscando. Si es incapaz de satisfacer la solicitud, se regresa un código de error y la solicitud falla.

El encabezado *Host* da nombre al servidor. Este encabezado se toma del URL y es obligatorio. Se utiliza porque algunas direcciones IP pueden proporcionar múltiples nombres DNS y el servidor necesita una forma de indicar a cuál *host* entregarle la solicitud.

El encabezado *Authorization* es necesario para páginas que están protegidas. Por ejemplo, tal vez el cliente debe probar que tiene permiso para ver la página solicitada. Este encabezado se utiliza en estos casos.

Aunque las *cookies* se tratan en el RFC 2109 en lugar de en el RFC 2616, también tienen dos encabezados. Los clientes utilizan el encabezado *Cookie* para regresar al servidor una *cookie* que fue enviada previamente por alguna máquina del dominio del servidor.

Encabezado	Tipo	Contenido
User-Agent	Solicitud	Información acerca del navegador y su plataforma
Accept	Solicitud	El tipo de páginas que el cliente puede manejar
Accept-Charset	Solicitud	Los conjuntos de caracteres que son aceptables para el cliente
Accept-Encoding	Solicitud	Las codificaciones de página que el cliente puede manejar
Accept-Language	Solicitud	Los idiomas naturales que el cliente puede manejar
Host	Solicitud	El nombre DNS del servidor
Authorization	Solicitud	Una lista de las credenciales del cliente
Cookie	Solicitud	Regresa al servidor una cookie establecida previamente
Date	Ambas	Fecha y hora en que se envió el mensaje
Upgrade	Ambas	El protocolo al que el emisor desea cambiar
Server	Respuesta	Información acerca del servidor
Content-Encoding	Respuesta	Cómo se codifica el contenido (por ejemplo, gzip)
Content-Language	Respuesta	El idioma natural utilizado en la página
Content-Length	Respuesta	La longitud de la página en bytes
Content-Type	Respuesta	El tipo MIME de la página
Last-Modified	Respuesta	La fecha y hora de la última vez que cambió la página
Location	Respuesta	Un comando para que el cliente envíe su solicitud a cualquier otro lugar
Accept-Ranges	Respuesta	El servidor aceptará solicitudes de rango de bytes
Set-Cookie	Respuesta	El servidor desea que el cliente guarde una cookie

Figura 7-43. Algunos encabezados de mensaje HTTP.

El encabezado *Date* puede utilizarse en las dos direcciones y contiene la fecha y la hora en la que se envió el mensaje. El encabezado *Upgrade* se utiliza para facilitar la transición a una versión futura (posiblemente incompatible) del protocolo HTTP. Permite que el cliente anuncie lo que puede soportar y que el servidor afirme lo que está utilizando.

Ahora veamos los encabezados que son utilizados exclusivamente por el servidor en respuesta a las solicitudes. El primero, *Server*, permite que el servidor indique quién es y algunas de sus propiedades, si lo desea.

Los siguientes cuatro encabezados, los que comienzan con *Content-*, permiten que el servidor describa propiedades de la página que está enviando.

El encabezado *Last-Modified* indica cuándo se modificó la página por última vez. Este encabezado juega un papel importante en el almacenamiento en caché de la página.

El servidor utiliza el encabezado *Location* para informar al cliente que debe tratar con un URL diferente. Éste puede utilizarse si la página se movió o para permitir que múltiples URLs hagan referencia a la misma página (posiblemente en servidores diferentes). También se utiliza para compañías que tienen una página Web principal en el dominio *com*, pero que redirigen a los clientes a una página nacional o regional con base en su dirección IP o idioma preferido.

Si una página es muy grande, tal vez un cliente pequeño no la quiera toda de una vez. Algunos servidores aceptan solicitudes de rangos de bytes, por lo que la página puede obtenerse en múltiples unidades pequeñas. El encabezado *Accept-Ranges* indica si el servidor está dispuesto a manejar este tipo de solicitud parcial de páginas.

El segundo encabezado de *cookie*, *Set-Cookie*, es la forma en que los servidores envían *cookies* a los clientes. Se espera que el cliente guarde la *cookie* y la regrese al servidor en solicitudes subsiguientes.

Uso de ejemplo de HTTP

Debido a que HTTP es un protocolo ASCII, es muy fácil que una persona en una terminal (a diferencia de un navegador) hable de manera directa con los servidores Web. Todo lo que se necesita es una conexión TCP con el puerto 80 del servidor. Se alienta a los lectores a que prueben este escenario personalmente (de preferencia desde un sistema UNIX, porque algunos otros sistemas no regresan el estado de la conexión). La siguiente secuencia de comandos lo hará:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org

close
```

Esta secuencia de comandos inicia una conexión telnet (es decir, TCP) con el puerto 80 en el servidor Web de IETF, *www.ietf.org*. El resultado de la sesión se redirige a un archivo de *registro* para una inspección posterior. A continuación se encuentra el comando *GET* que nombra al archivo y al protocolo. La siguiente línea es el encabezado *Host* obligatorio. La línea en blanco también se requiere. Indica al servidor que ya no hay más encabezados de solicitud. El comando *close* indica al programa que termine la conexión.

El *registro* puede inspeccionarse utilizando cualquier editor. Debe comenzar de manera similar al listado de la figura 7-44, a menos que IETF haya cambiado recientemente.

Las primeras tres líneas se envían a la salida desde el programa telnet, no desde el sitio remoto. La línea que comienza con HTTP/1.1 es la respuesta de IETF mediante la cual indica que está dispuesto a hablar HTTP/1.1 con usted. A continuación se encuentran algunos encabezados y después el contenido. Ya vimos todos los encabezados excepto *ETag,* que es un identificador de página único que se relaciona con el almacenamiento en caché, y *X-Pad,* que es no estándar y que tal vez sea una solución para algún navegador con errores.

7.3.5 Mejoras de desempeño

La popularidad de Web ha sido casi su destrucción. Los servidores, enrutadores y las líneas con frecuencia están sobrecargados. Muchas personas han comenzado a llamar a WWW World Wide Wait. Como consecuencia de estos retardos interminables, los investigadores han desarrollado varias técnicas para mejorar el desempeño. A continuación examinaremos tres de ellas: el almacenamiento en caché, la replicación del servidor y las redes de entrega de contenido.

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes
Content-Length: 3211
Content-Type: text/html
X-Pad: avoid browser bug

<html>
<head>
<title>Página IETF RFC</title>

<script language="javascript">
function url() {
var x = document.form1.number.value
if (x.length == 1) {x = "000" + x }
if (x.length == 2) {x = "00" + x }
if (x.length == 3) {x = "0" + x }
document.form1.action = "/rfc/rfc" + x + ".txt"
document.form1.submit
}
</script>

</head>
```

Figura 7-44. El inicio de la salida de *www.ietf.org/rfc.html.*

Almacenamiento en caché

Una forma muy sencilla de mejorar el desempeño es guardar las páginas que han sido solicitadas en caso de que se utilicen nuevamente. Esta técnica es especialmente efectiva con páginas que se visitan mucho, como *www.yahoo.com* y *www.cnn.com*. La técnica de guardar páginas para uso posterior se conoce como **almacenamiento en caché.** El procedimiento común es que algún proceso, llamado **proxy,** mantenga el caché. Para utilizar el almacenamiento en caché, un navegador puede configurarse para que todas las solicitudes de páginas se le hagan a un *proxy* en lugar de al servidor real de la página. Si el *proxy* tiene la página, la regresa de inmediato. De lo contrario, la obtiene del servidor, la agrega al caché para uso posterior y la envía al cliente que la solicitó.

Dos preguntas importantes relacionadas con el almacenamiento en caché son las siguientes:

1. ¿Quién debe realizar el almacenamiento en caché?
2. ¿Por cuánto tiempo deben almacenarse en caché las páginas?

Hay diversas respuestas para la primera pregunta. Las PCs individuales con frecuencia ejecutan los *proxies* a fin de que éstos puedan buscar con rapidez las páginas previamente visitadas. En una LAN de una compañía, el *proxy* con frecuencia es una máquina compartida por todas las máquinas de la LAN, por lo que si un usuario busca cierta página y luego otro usuario de la misma LAN desea la misma página, ésta se puede obtener del caché del *proxy*. Muchos ISPs también ejecutan *proxies*, a fin de que todos sus clientes puedan tener un acceso más rápido. Con frecuencia, todos estos cachés funcionan al mismo tiempo, por lo que las solicitudes primero van al *proxy* local. Si éste falla, consulta al *proxy* de la LAN. Si éste también falla, prueba el *proxy* del ISP. Este último debe tener éxito, ya sea desde su caché, un caché de nivel superior o desde el servidor mismo. Un esquema que involucra el acceso en secuencia de múltiples cachés se conoce como **almacenamiento en caché jerárquico.** En la figura 7-45 se ilustra una posible implementación.

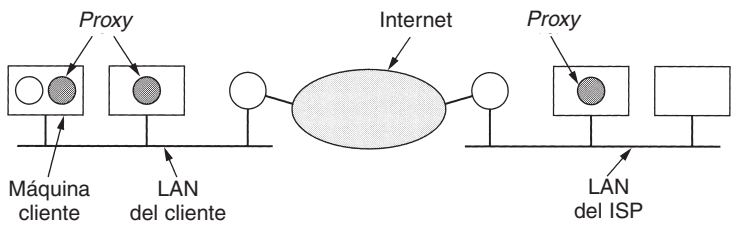


Figura 7-45. Almacenamiento en caché jerárquico con tres *proxies*.

Determinar el tiempo que deben permanecer las páginas en caché es un poco difícil. Algunas páginas no deberían almacenarse en caché. Por ejemplo, una página que contiene los precios de las 50 acciones más activas cambia cada segundo. Si se almacenara en caché, un usuario que obtuviera una copia del caché obtendría datos **obsoletos.** Por otro lado, una vez que el mercado de valores cierre, esa página permanecerá válida por algunas horas o días, hasta que inicie la siguiente sesión del mercado. Por lo tanto, el almacenamiento en caché de una página podría variar con el tiempo.

El elemento clave para determinar cuándo expulsar una página del caché es qué tanta obsolescencia están dispuestos a aceptar los usuarios (debido a que las páginas en caché se almacenan en el disco, la cantidad de espacio consumido por lo general no es un problema). Si un *proxy* elimina páginas con rapidez, raramente regresará una página obsoleta pero tampoco será muy efectivo (es decir, tendrá una tasa baja de coincidencias). Si mantiene las páginas por mucho tiempo, tal vez tendrá una tasa alta de coincidencias pero a expensas de regresar con frecuencia páginas obsoletas.

Hay dos métodos para tratar este problema. El primero utiliza una heurística para adivinar cuánto tiempo se mantendrá cada página. Una común es basar el tiempo de almacenamiento en el encabezado *Last-Modified* (vea la figura 7-43). Si una página se modificó hace una hora, se

mantendrá en caché por una hora. Si se modificó hace un año, obviamente es una página muy estable (por ejemplo, una lista de los dioses de las mitologías griega y romana), por lo que se puede almacenar en caché por un año y esperar razonablemente que no cambie durante dicho tiempo. Si bien esta heurística con frecuencia funciona bien en la práctica, regresa páginas obsoletas de vez en cuando.

El otro método es más costoso, pero elimina la posibilidad de obtener páginas obsoletas debido a que utiliza características especiales del RFC 2616 que tienen que ver con el manejo del caché. Una de las características más útiles es el encabezado de solicitud *If-Modified-Since*, que puede ser enviado por un *proxy* a un servidor. Especifica la página que el *proxy* desea y la fecha en que ésta fue modificada por última vez (a partir del encabezado *Last-Modified*). Si la página no se ha modificado desde entonces, el servidor regresa un mensaje corto *Not Modified* (el código de estado 304 de la figura 7-42), el cual indica al *proxy* que utilice la página en caché. Si ésta ha sido modificada, se regresa la nueva página. Aunque este método siempre requiere un mensaje de solicitud y uno de respuesta, este último será muy corto cuando la entrada en caché aún sea válida.

Estos dos métodos pueden combinarse con facilidad. Para la primera ΔT después de obtener la página, el *proxy* simplemente la regresa a los clientes que la solicitan. Después de que la página ha existido por algún tiempo, el *proxy* utiliza mensajes *If-Modified-Since* para verificar qué tan actualizada está. Elegir ΔT involucra invariablemente algún tipo de heurística, dependiendo de cuándo fue la última modificación de la página.

Las páginas Web con contenido dinámico (por ejemplo, generado por una secuencia de comandos PHP) nunca deben almacenarse en caché debido a que los parámetros pueden ser diferentes la siguiente vez. Para manejar éste y otros casos, hay un mecanismo general para que un servidor instruya a todos los *proxies* de la ruta hacia el cliente que no utilicen otra vez la página actual sin antes verificar si está actualizada. Este mecanismo también puede utilizarse para cualquier página que se espera que cambie rápidamente. En el RFC 2616 también se define una variedad de otros mecanismos de control de caché.

Otro enfoque para mejorar el desempeño es el almacenamiento en caché proactivo. Cuando un *proxy* obtiene una página de un servidor, puede inspeccionarla para ver si tiene hipervínculos. De ser así, puede emitir solicitudes a los servidores correspondientes para precargar el caché con las páginas a las que apuntan dichos hipervínculos, en caso de que éstas se necesiten. Esta técnica puede reducir el tiempo de acceso en las solicitudes subsiguientes, pero también puede inundar las líneas de comunicación con páginas que nunca se solicitan.

Claramente, el almacenamiento en caché de Web es todo menos trivial. Se puede decir mucho más sobre él. De hecho, se han escrito libros completos sobre él, por ejemplo (Rabinovich y Spatscheck, 2002, y Wessels, 2001). Sin embargo, es tiempo de que pasemos al siguiente tema.

Replicación del servidor

El almacenamiento en caché es una técnica en el cliente para mejorar el desempeño, pero también existen técnicas en el servidor. El método más común que los servidores utilizan para mejorar el desempeño es replicar su contenido en múltiples ubicaciones separadas considerablemente. Esta técnica a veces se conoce como **espejeo** (*mirroring*).

Un uso típico del espejeo es para que la página principal de una compañía contenga algunas imágenes e hipervínculos para, digamos, los sitios Web regionales este, oeste, norte y sur de dicha compañía. A continuación el usuario hace clic en el más cercano para acceder a ese servidor. De ahí en adelante, todas las solicitudes van al servidor seleccionado.

Los sitios espejo por lo general son estáticos. La compañía decide dónde colocar los espejos, arregla un servidor en cada región y coloca el contenido en cada ubicación (posiblemente omitiendo los quitanieve del sitio de Miami, y las toallas de playa del sitio de Anchorage). La elección de sitios por lo general permanece estable por meses o años.

Desafortunadamente, Web tiene un fenómeno conocido como **flash crowds** (*aglomeración instantánea*) en el que un sitio Web que era un lugar alejado, desconocido y no visitado de repente se vuelve el centro del universo. Por ejemplo, hasta el 6 de noviembre de 2000, el sitio Web de la Secretaría del Estado de Florida, *www.dos.state.fl.us*, se encontraba tranquilamente proporcionando información sobre las reuniones del gabinete del Estado de Florida e instrucciones de cómo convertirse en un notario en Florida. Pero el 7 de noviembre, cuando la presidencia de los Estados Unidos dependía repentinamente de unos cuantos miles de votos controvertidos en un puñado de condados de Florida, se convirtió en uno de los cinco sitios más visitados del mundo. No es necesario decir que no pudo manejar la carga y casi muere en el intento.

Lo que se necesita es una forma para que un sitio Web que de repente note un incremento masivo en el tráfico, se clone automáticamente a sí mismo en tantas ubicaciones como sea necesario y mantenga funcionando a esos sitios hasta que pase la tormenta, en cuyo momento deberá desactivar muchos o todos ellos. Para tener esta capacidad, un sitio necesita un acuerdo por adelantado con alguna compañía que posea muchos sitios de hospedaje, que indique que puede crear réplicas bajo demanda y que debe pagar por la capacidad que realmente utilice.

Una estrategia aún más flexible es crear réplicas dinámicas por página dependiendo de dónde viene el tráfico. Alguna investigación sobre este tema se reporta en (Pierre y cols., 2001, y Pierre y cols., 2002).

Redes de entrega de contenido

La brillantez del capitalismo es que alguien ha descubierto cómo hacer dinero con World Wide Wait. Funciona como se muestra a continuación. Las compañías llamadas **CDN**s (**redes de entrega de contenido**) hablan con proveedores de contenido (sitios de música, periódicos y otros que desean que su contenido esté disponible rápida y fácilmente) y ofrecen entregar su contenido a los usuarios finales de manera eficiente a cambio de una cuota. Después de que se firma el contrato, el dueño del contenido da a la CDN el contenido de su sitio Web para que lo preprocese (lo que se analizará más adelante) y después lo distribuya.

A continuación la CDN habla con una gran cantidad de ISPs y ofrece pagarles bien a cambio de que le permitan colocar en sus LANs un servidor manejado de manera remota lleno de contenido valioso. Ésta no es sólo una fuente de ingresos, sino que también proporciona a los clientes de los ISPs excelente tiempo de respuesta para obtener el contenido de la CDN, lo que da al ISP una ventaja competitiva sobre otros ISPs que no han tomado dinero gratis de la CDN. Bajo estas condiciones, firmar con una CDN es una decisión que el ISP no necesita pensar. Como

consecuencia, las CDNs más grandes tienen más de 10,000 servidores distribuidos por todo el mundo.

Con el contenido replicado en miles de sitios alrededor del mundo, hay claramente un gran potencial para mejorar el rendimiento. Sin embargo, para que esto funcione, debe haber una forma para redirigir la solicitud del cliente al servidor más cercano de la CDN, de preferencia uno colocado en el ISP del cliente. Además, esta redirección se debe hacer sin modificar el DNS o cualquier otra parte de la infraestructura estándar de Internet. A continuación se da una descripción ligeramente simplificada de cómo lo hace Akamai, la CDN más grande.

Todo el proceso inicia cuando el proveedor de contenido entrega a la CDN su sitio Web. A continuación la CDN ejecuta cada página a través de un preprocesador que reemplaza todos los URLs con URLs modificados. El modelo funcional detrás de esta estrategia es que el sitio Web del proveedor de contenido conste de muchas páginas que sean pequeñas (sólo texto HTML), pero que éstas por lo general tengan vínculos a archivos más grandes, como imágenes, audio y vídeo. Las páginas HTML modificadas se almacenan en el servidor del proveedor de contenido y se obtienen de la forma común; las imágenes, el audio y el vídeo se encuentran en los servidores de la CDN.

Para ver cómo funciona realmente este esquema, considere la página Web de Furry Video que se muestra en la figura 7-46(a). Después de que preprocesa, se transforma en la que se muestra en la figura 7-46(b) y se coloca en el servidor de Furry Video como *www.furryvideo.com/index.html*.

Cuando el usuario teclea el URL *www.furryvideo.com*, el DNS regresa la dirección IP del sitio Web de Furry Video, permitiendo que la página principal (HTML) se obtenga de la forma común. Cuando se hace clic en alguno de los hipervínculos, el navegador pide al DNS que busque *cdn-server.com*, lo cual hace. A continuación, el navegador envía una solicitud HTTP a esta dirección IP, esperando obtener un archivo MPEG.

Eso no sucede porque *cdn-server.com* no alberga ese contenido. En su lugar, es el servidor HTTP falso de la CDN. Éste examina el nombre del archivo y del servidor para averiguar cuál página de cuál proveedor de contenido se necesita. También examina la dirección IP de la solicitud entrante y la busca en su base de datos para determinar en dónde es probable que esté el usuario. Una vez que obtiene esta información determina cuál de los servidores de contenido de la CDN puede dar el mejor servicio al usuario. Esta decisión es difícil porque el que esté más cerca geográficamente puede no ser el que esté más cerca en términos de topología de red, y éste puede estar muy ocupado en ese momento. Después de tomar una decisión, *cdn-server.com* regresa una respuesta con el código de estado 301 y un encabezado *Location* en el que se da el URL del archivo del servidor de contenido de la CDN que esté más cerca del cliente. Para este ejemplo, asumamos que el URL es *www.CDN-0420.com/furryvideo/osos.mpg*. A continuación el navegador procesa este URL de la forma usual para obtener el archivo MPEG real.

La figura 7-47 ilustra los pasos involucrados. El primer paso busca a *www.furryvideo.com* para obtener su dirección IP. Después de eso, la página HTML se puede obtener y desplegar de la forma común. La página contiene tres hipervínculos al *cdn-server* [vea la figura 7-46(b)]. Cuando se hace clic en el primero, su dirección DNS se busca (paso 5) y se regresa (paso 6). Cuando se envía a *cdn-server* un mensaje en el que se solicita *osos.mpg* (paso 7), se le indica al cliente que vaya a *CDN-0420.com* (paso 8). Cuando dicho cliente hace lo que se le indica (paso 9), se le da el archivo del caché del *proxy* (paso 10). La propiedad que hace que todo este mecanismo

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Lista de productos de Furry Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href="osos.mpg"> Los osos hoy </a> <br>
<a href="conejos.mpg"> Conejos graciosos </a> <br>
<a href="ratones.mpg"> Ratones simpáticos </a> <br>
</body>
</html>
```

(a)

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Lista de productos de Furry Video </h1>
<p> Haga clic abajo para obtener muestras gratis. </p>

<a href="http://cdn-server.com/furryvideo/osos.mpg"> Los osos hoy </a> <br>
<a href="http://cdn-server.com/furryvideo/conejos.mpg"> Conejos
   graciosos </a> <br>
<a href="http://cdn-server.com/furryvideo/ratones.mpg"> Ratones
   simpáticos </a> <br>
</body>
</html>
```

(b)

Figura 7-46. (a) Página Web original. (b) La misma página después de la transformación.

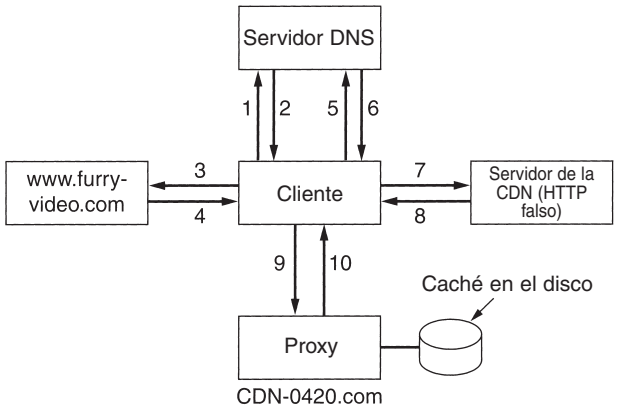
funcione es el paso 8, el servidor HTTP falso que redirige al cliente a un *proxy* de la CDN que está cerca del cliente.

El servidor de la CDN al que se redirige al cliente por lo general es un *proxy* con un caché grande precargado con el contenido más importante. Sin embargo, si alguien pide un archivo que no se encuentra en el caché, se obtiene del servidor real y se coloca en el caché para uso posterior. Al hacer que el servidor de contenido sea un *proxy* en lugar de una réplica, la CDN tiene la capacidad de negociar un tamaño de disco, tiempo de precarga y diversos parámetros de desempeño.

Para obtener mayor información sobre las redes de entrega de contenido, vea (Hull, 2002, y Rabinovich y Spatscheck, 2002).

7.3.6 La Web inalámbrica

Hay un interés considerable en los dispositivos pequeños portables capaces de acceder a Web a través de un enlace inalámbrico. De hecho, ya se han tomado los primeros pasos tentativos en esa dirección. Sin duda habrá muchos cambios en esta área en los años venideros, pero aún vale



1. Se busca www.furryvideo.com
2. Se regresa la dirección IP de Furry
3. Se solicita la página HTML desde Furry
4. Se regresa la página HTML
5. Después de hacer clic, se busca cdn-server.com
6. Se regresa la dirección IP del servidor de la CDN
7. Se pide al servidor de la CDN el archivo osos.mpg
8. Se le indica al cliente que redirija a CDN-0420.com
9. Se solicita el archivo osos.mpg
10. Se regresa el archivo en caché osos.mpg

Figura 7-47. Pasos de la búsqueda de un URL cuando se utiliza una CDN.

la pena examinar algunas de las ideas actuales relacionadas con la Web inalámbrica para ver en dónde estamos y hacia dónde podríamos dirigirnos. Nos enfocaremos en los dos primeros sistemas Web inalámbricos de área amplia: WAP e i-mode.

WAP—Protocolo de Aplicaciones Inalámbricas

Una vez que Internet y los teléfonos móviles se volvieron comunes en todos los lugares, no tomó mucho tiempo para que alguien tuviera la idea de combinarlos en un teléfono móvil con una pantalla integrada para acceder de manera inalámbrica al correo electrónico y a Web. Ese "alguien" en este caso fue un consorcio que inicialmente estaba encabezado por Nokia, Ericsson, Motorola y phone.com (anteriormente Unwired Planet) y que ahora tiene una gran cantidad de miembros. El sistema se llama **WAP (Protocolo de Aplicaciones Inalámbricas).**

Un dispositivo WAP puede ser un teléfono móvil mejorado, un PDA o una computadora *notebook* sin ninguna capacidad de voz. La especificación acepta a todos ellos y a otros más. La idea básica es utilizar la infraestructura existente digital inalámbrica. Los usuarios pueden literalmente llamar a una puerta de enlace WAP a través del enlace inalámbrico y enviarle solicitudes de páginas Web. A continuación dicha puerta de enlace verifica su caché para ver si tiene la página solicitada. Si la tiene, la envía; si no la tiene, la obtiene a través de la Internet alámbrica. En esencia, esto significa que WAP 1.0 es un sistema de conmutación de circuitos con un cargo de conexión por minuto relativamente alto. En resumen, a las personas no les gustó acceder a Internet en una pequeña pantalla y pagar por minuto, por lo que WAP fue un fracaso (aunque también habían otros problemas). Sin embargo, parecía que WAP y su competidor, i-mode (que se analizará más adelante), convergían en una tecnología similar, por lo que WAP 2.0 sí podría ser un éxito. Puesto que WAP 1.0 fue el primer intento de una Internet inalámbrica, vale la pena describirla por lo menos brevemente.

WAP es en esencia una pila de protocolos para acceder a Web, pero está optimizada para conexiones de ancho de banda bajo que utilizan dispositivos inalámbricos que tienen una CPU lenta, poca memoria y una pantalla pequeña. Estos requerimientos son muy diferentes de aquellos del escenario de PC de escritorio estándar, lo que provoca algunas diferencias de protocolos. En la figura 7-48 se muestran las capas.

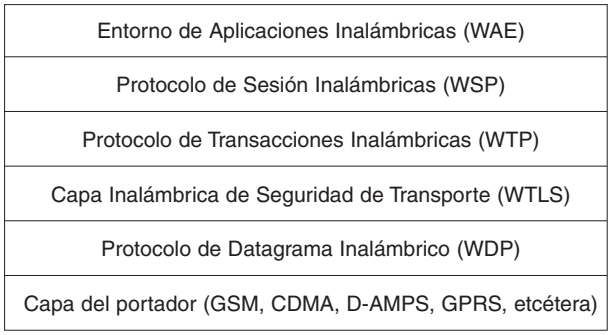


Figura 7-48. La pila de protocolos WAP.

La capa inferior soporta todos los sistemas existentes de teléfonos móviles, incluyendo GSM, D-AMPS y CDMA. La tasa de datos de WAP 1.0 es de 9600 bps. Encima de esta capa se encuentra el protocolo de datagramas, **WDP (Protocolo de Datagrama Inalámbrico),** que es esencialmente UDP. A continuación se encuentra una capa para seguridad, obviamente necesaria en un sistema inalámbrico. WTLS es un subgrupo de la SSL de Netscape, la cual veremos en el capítulo 8. Arriba de la capa anterior se encuentra la capa de transacciones, la cual maneja de manera confiable o no confiable las solicitudes y respuestas. Esta capa reemplaza a TCP, que no se utiliza a través del enlace de aire por razones de eficiencia. A continuación se encuentra una capa de sesión, que es similar a HTTP/1.1 pero tiene algunas restricciones y extensiones para propósitos de optimización. En la parte superior se encuentra un micronavegador (WAE).

Además del costo, el otro aspecto que sin duda daña la aceptación de WAP es el hecho de que no utiliza HTML. En su lugar, la capa WAE utiliza un lenguaje de marcado llamado **WML (Lenguaje de Marcado Inalámbrico),** que es una aplicación de XML. Como consecuencia, en principio, un dispositivo WAP sólo puede acceder aquellas páginas que se han convertido a WML. Sin embargo, debido a que esto restringe el valor de WAP, la arquitectura exige un filtro al vuelo de HTML a WML para incrementar el conjunto de páginas disponibles. Esta arquitectura se ilustra en la figura 7-49.

Con toda justicia, WAP probablemente estaba adelantado a su época. Cuando se inició WAP por primera vez, XML ya era muy conocido fuera del W3C por lo que la prensa anunció su lanzamiento como **WAP NO UTILIZA HTML.** Un encabezado más preciso habría sido: **WAP YA UTILIZA EL NUEVO ESTÁNDAR HTML.** Pero una vez hecho el daño, fue difícil repararlo y WAP 1.0 nunca tuvo popularidad. Analizaremos WAP después de echar un vistazo a su mayor competidor.

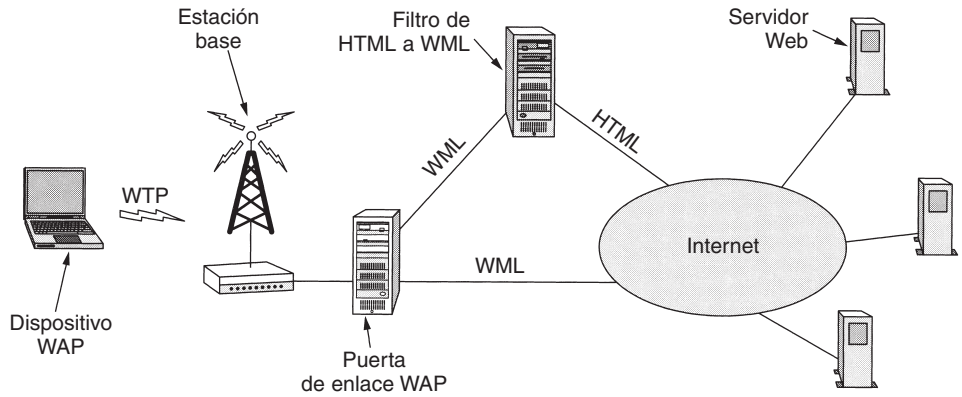


Figura 7-49. La arquitectura de WAP.

I-mode

Mientras un consorcio de múltiples industrias de fabricantes de telecomunicaciones y compañías de computadoras estaba ocupado elaborando con dificultad un estándar abierto utilizando la versión más avanzada disponible de HTML, en Japón se estaban realizando otros desarrollos. Una mujer japonesa, Mari Matsunaga, inventó un método diferente para la Web inalámbrica llamado **i-mode** (***information-mode***)**.** Mari convenció al subsidiario inalámbrico del primer monopolio de telefonía japonesa de que su método era correcto, y en febrero de 1999 NTT DoCoMo (literalmente: Japanese Telephone and Telegraph Company a donde quiera que vaya) lanzó el servicio en Japón. En los siguientes tres años tuvo cerca de 35 millones de suscriptores japoneses, quienes podían acceder aproximadamente 40,000 sitios Web de i-mode especiales. También tenía la admiración de las compañías de telecomunicaciones por su éxito financiero, especialmente debido a que parecía que WAP no iba a ningún lado. A continuación echaremos un vistazo a lo que es i-mode y cómo funciona.

El sistema i-mode tiene tres componentes principales: un nuevo sistema de transmisión, un nuevo microteléfono y un nuevo lenguaje para el diseño de páginas Web. El sistema de transmisión consiste en dos redes separadas: la red de teléfonos móviles de conmutación de circuitos existente (un tanto similar a D-AMPS) y una nueva red de conmutación de paquetes construida específicamente para el servicio i-mode. El modo de voz utiliza la red de conmutación de circuitos y se cobra por minuto de tiempo de conexión. i-mode utiliza la red de conmutación de paquetes y siempre está activo (al igual que ADSL o el cable), por lo que no se cobra una tarifa por tiempo de conexión. En su lugar, se cobra por paquete enviado. En la actualidad no es posible utilizar las dos redes al mismo tiempo.

Los microteléfonos se parecen a los teléfonos móviles, pero además tienen una pantalla pequeña. NTT DoCoMo promociona los dispositivos i-mode como teléfonos móviles mejorados y no como terminales Web inalámbricas, aunque esto es precisamente lo que son. De hecho, probablemente la mayoría de los clientes no están concientes de que están en Internet. Consideran a sus

dispositivos i-mode como teléfonos móviles con servicios mejorados. Al apegarnos a la idea de que el modelo de i-mode es un servicio, los microteléfonos no pueden ser programados por usuarios, aunque contienen el equivalente de una PC de 1995 y probablemente podrían ejecutar Windows 95 o UNIX.

Cuando se hace la conmutación del microteléfono de i-mode, se presenta al usuario una lista de categorías de los servicios aprobados oficialmente. Hay cerca de 1000 servicios divididos en 20 categorías. Cada uno, que en realidad es un pequeño sitio Web i-mode, es ejecutado por una compañía independiente. La categoría principal del menú oficial incluye correo electrónico, noticias, clima, deportes, juegos, compras, mapas, horóscopos, entretenimiento, viajes, guías regionales, sonidos de teléfono, recetas, casinos, sistema bancario doméstico y valores de la bolsa. El servicio está dirigido a adolescentes y a personas en sus 20s, quienes tienden a amar los cosas electrónicas, especialmente si vienen en colores de moda. El simple hecho de que cerca de 40 compañías estén vendiendo sonidos de teléfono significa algo. La aplicación más popular es el correo electrónico, que permite mensajes de hasta 500 bytes y, por lo tanto, se considera como una gran mejora en comparación con SMS (Servicio de Mensajes Cortos) que permite mensajes de hasta 160 bytes. Los juegos también son populares.

También hay cerca de 40,000 sitios Web i-mode, pero tienen que accederse escribiendo su URL, en lugar de seleccionarlos de un menú. En este sentido, la lista oficial es como un portal de Internet que permite que otros sitios Web se accedan haciendo clic en un hipervínculo en lugar de escribir un URL.

NTT DoCoMo controla en forma estricta los servicios oficiales. Para ser aceptado en la lista, un servicio debe cumplir varios criterios publicados. Por ejemplo, un servicio no debe ser una mala influencia para la sociedad, los diccionarios japonés-inglés deben tener suficientes palabras, los servicios que proporcionan sonidos de teléfono deben agregar con frecuencia nuevos sonidos y ningún sitio debe alentar el comportamiento anormal o decir algo malo de NTT DoCoMo (Frengle, 2002). Los 40,000 sitios de Internet pueden hacer lo que quieran.

El modelo de negocios de i-mode es tan diferente del de la Internet convencional que vale la pena explicarlo. La cuota básica de suscripción a i-mode es de algunos dólares mensuales. Puesto que hay un cargo por cada paquete recibido, la suscripción básica incluye un pequeño número de paquetes. De manera alterna, el cliente puede elegir una suscripción con más paquetes gratis, en la que el cargo por paquete desciende bruscamente conforme avanza de 1 MB hasta 10 MB por mes. Si los paquetes gratis se acaban a la mitad del mes, es posible comprar paquetes adicionales en línea.

Para utilizar un servicio, debe suscribirse a él, lo que se logra con sólo hacer clic en él e introduciendo su código PIN. La mayoría de los servicios oficiales cuesta cerca de $1 a $2 mensuales. NTT DoCoMo agrega el cargo al recibo telefónico y da el 91% de tal cargo al proveedor del servicio, y se queda con el 9%. Si un servicio no oficial tiene un millón de clientes, tiene que enviar mensualmente un millón de facturas de (aproximadamente) $1 cada una. Si ese servicio se vuelve oficial, NTT DoCoMo maneja la facturación y sólo transfiere $910,000 mensualmente a la cuenta bancaria del servicio. No tener que manejar la facturación es un gran incentivo para que un proveedor de servicios se vuelva oficial, lo cual genera más ingresos para NTT DoCoMo. Además, el hecho de ser oficial lo coloca en el menú inicial, que hace que su sitio sea más fácil

de encontrar. El recibo telefónico del usuario incluye llamadas telefónicas, cargos por suscripción a i-mode, cargos por suscripción a servicios y paquetes extra.

A pesar de su gran éxito en Japón, aún no es claro si tendrá éxito en Estados Unidos y en Europa. De alguna forma, las circunstancias japonesas son diferentes a las de Occidente. Primero, la mayoría de los clientes potenciales occidentales (por ejemplo, adolescentes, universitarios y personas de negocios) ya tienen en casa una PC con una gran pantalla y, seguramente, con una conexión a Internet con una velocidad de por lo menos 56 kbps, y con frecuencia mucho más. En Japón pocas personas tienen en casa una PC conectada a Internet, en parte debido a la falta de espacio, pero también debido a los exorbitantes cargos de NTT por los servicios locales telefónicos (aproximadamente $700 por instalar una línea y $1.50 la hora por llamadas locales). Para la mayoría de los usuarios, i-mode es la única conexión a Internet.

Segundo, las personas occidentales no están acostumbradas a pagar $1 mensual por acceder al sitio Web de CNN, $1 mensual por acceder al sitio Web de Yahoo, $1 mensual por acceder el sitio Web de Google, etcétera, sin mencionar la cuota por MB descargados. En la actualidad la mayoría de los proveedores de Internet occidentales cobran una cuota mensual sin importar el uso real, principalmente como respuesta a las exigencias del cliente.

Tercero, para muchos japoneses el horario de cuota normal es mientras se trasladan en tren o en el metro. En Europa se trasladan por tren menos personas que en Japón, y en Estados Unidos casi nadie lo hace. Usar i-mode en casa junto a la computadora con monitor de 17 pulgadas, conexión ADSL de 1 Mbps, y todos los megabytes que quiera gratis no tiene mucho sentido. Sin embargo, nadie predijo la inmensa popularidad de los teléfonos móviles, por lo que i-mode aún podría encontrar un lugar en Occidente.

Como mencionamos anteriormente, los microteléfonos de i-mode utilizan la red de conmutación de circuitos existente para voz y una nueva red de conmutación de paquetes para datos. La red de datos se basa en CDMA y transmite paquetes de 128 bytes a 9600 bps. En la figura 7-50 se muestra un diagrama de la red. Los microteléfonos hablan **LTP** (**Protocolo de Transporte de Carga Ligera**) a través del enlace de aire hacia una puerta de enlace de conversión de protocolo. Dicha puerta de enlace tiene una conexión de fibra óptica de banda ancha con el servidor i-mode, que está conectado a todos los servicios. Cuando el usuario selecciona un servicio del menú oficial, la solicitud se envía al servidor i-mode, que almacena en caché la mayoría de las páginas para mejorar el desempeño. Las solicitudes a sitios que no están en el menú oficial ignoran el servidor i-mode y van directamente a través de Internet.

Los microteléfonos actuales tienen CPUs que se ejecutan aproximadamente a 100 MHz, algunos megabytes de memoria ROM (flash ROM), tal vez 1 MB de RAM y una pequeña pantalla integrada. i-mode requiere que la pantalla sea de por lo menos 72×94 píxeles, pero algunos dispositivos de alta calidad pueden tener hasta 120×160 píxeles. Por lo general, las pantallas tienen colores de 8 bits, que permiten 256 colores. Esto no es suficiente para fotografías pero es adecuado para dibujos de líneas y caricaturas sencillas. Puesto que no hay ratón, la navegación en pantalla se realiza con las teclas de flecha.

En la figura 7-51 se muestra la estructura del software. La capa inferior consiste en un sistema operativo en tiempo real sencillo para controlar el hardware. Después se encuentra el módulo para realizar la comunicación de red, que utiliza el protocolo LTP propietario de NTT DoCoMo.

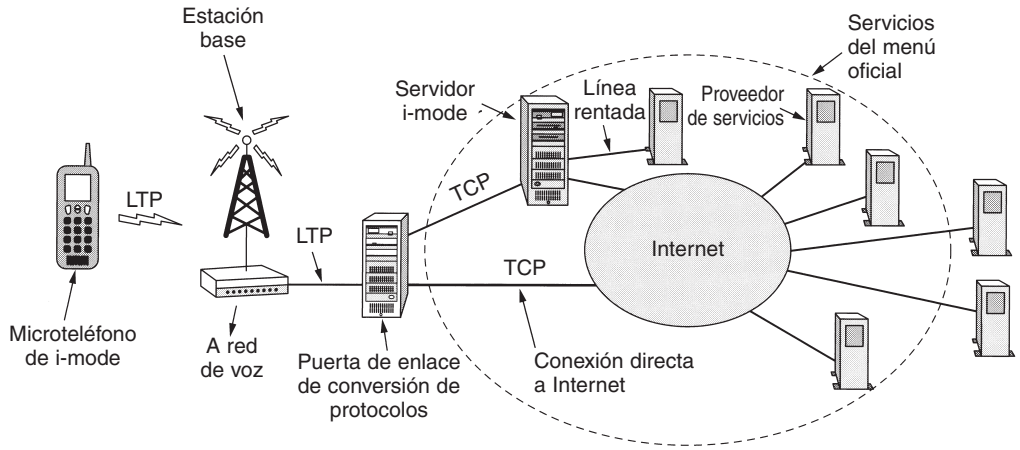


Figura 7-50. Estructura de la red de datos de i-mode que muestra los protocolos de transporte.

Arriba de eso se encuentra un administrador de ventanas sencillo que maneja texto y gráficos simples (archivos GIF). Debido a que las pantallas por lo mucho son de 120×160 píxeles, no hay mucho que manejar.

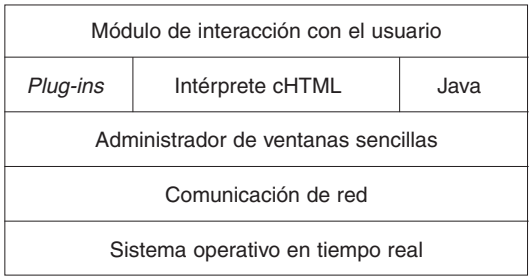


Figura 7-51. Estructura del software de i-mode.

La cuarta capa contiene el intérprete de la página Web (es decir, el navegador). i-mode no utiliza el HTML completo, sino un subconjunto de él, llamado **cHTML** (**compact HTML, HTML compacto**), que se basa ligeramente en HTML 1.0. Esta capa también permite aplicaciones auxiliares y *plug-ins*, al igual que lo hacen los navegadores de las PCs. Una aplicación auxiliar estándar es un intérprete de una versión ligeramente modificada de JVM.

En la parte superior se encuentra un módulo de interacción con el usuario, el cual maneja la comunicación con el usuario.

Ahora demos un vistazo más de cerca al cHTML. Como dijimos, es parecido a HTML 1.0, con algunas omisiones y algunas extensiones para utilizarlo en microteléfonos móviles. Se emitió al W3C para su estandarización, pero éste mostró poco interés en él, por lo que es probable que permanezca como un producto propietario.

Se permite la mayoría de las etiquetas básicas HTML, entre ellas <html>, <head>, <title>, <body>, <h*n*>, , <ul>, <ol>, <menu>, <li>,
, <p>, <hr>, <img>, <form> e <input>. Las etiquetas <b> e <i> no están permitidas.

La etiqueta <a> se permite para enlazar páginas, pero con el esquema adicional *tel* para marcar números telefónicos. En un sentido, *tel* es análogo a *mailto*. Cuando se selecciona un hipervínculo que utiliza *mailto*, el navegador despliega un formulario para enviar correo electrónico al destino nombrado en el vínculo. Cuando se selecciona un hipervínculo que utiliza el esquema *tel*, el navegador marca el número telefónico. Por ejemplo, una libreta de direcciones podría tener fotos sencillas de varias personas. Cuando se seleccione una de ellas, el microteléfono llamará a la persona correspondiente. El RFC 2806 analiza los URLs telefónicos.

El navegador cHTML está limitado de otras formas. No soporta JavaScript, tramas, hojas de estilo, colores de fondo o imágenes de fondo. Tampoco soporta imágenes JPEG debido a que tardan mucho en descomprimirse. Los subprogramas de Java están permitidos, pero están (actualmente) limitados a 10 KB debido a la velocidad lenta de transmisión a través del enlace de aire.

Aunque NTT DoCoMo eliminó algunas etiquetas HTML, también agregó otras más. La etiqueta <blink> hace que el texto se encienda y se apague. Aunque parece inconsistente prohibir la etiqueta <b> (sobre la base de que los sitios Web no deben manejar la apariencia) y después agregar la etiqueta <blink>, que se refiere solamente a la apariencia, lo hicieron. <marquee> es otra de las etiquetas nuevas; ésta desplaza el contenido en la pantalla de forma parecida a un teletipo.

Otra característica nueva es el atributo *align* de la etiqueta
. Es necesario debido a que con una pantalla de, por lo general, 6 filas de 16 caracteres, hay un gran riesgo de que las palabras se partan a la mitad, como se muestra en la figura 7-52(a). *Align* ayuda a reducir este problema a fin de obtener algo muy parecido a lo que se muestra en la figura 7-52(b). Es interesante mencionar que los japoneses no tienen el problema de que sus palabras se dividan en diversas líneas. Para el texto kanji, la pantalla se divide en una cuadrícula rectangular de celdas de 9×10 píxeles o 12×12 píxeles, dependiendo de la fuente soportada. Cada celda puede contener exactamente un carácter kanji, que es el equivalente de una palabra en inglés. En japonés están permitidos los saltos de líneas entre palabras.

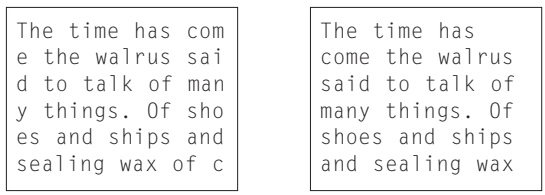


Figura 7-52. Lewis Carroll conoce una pantalla de 16×6.

Aunque el lenguaje japonés tiene decenas de miles de kanji, NTT DoCo-Mo inventó otros 166, llamados **emoji,** pictogramas parecidos a las caritas que se muestran en la figura 7-6. Incluyen símbolos para los signos astrológicos, cerveza, hamburguesa, parque de diversiones, cumpleaños, teléfono

móvil, perro, gato, Navidad, corazón roto, beso, estado de ánimo, dormilón y, claro, uno que significa bonito.

Otro nuevo atributo es la capacidad de permitir que los usuarios seleccionen hipervínculos mediante el teclado, una propiedad claramente importante en una computadora sin ratón. En la figura 7-53 se muestra el archivo cHTML que contiene un ejemplo de cómo se utiliza este atributo.

```
<html>
<body>
<h1> Seleccione una opción </h1>
<a href="messages.chtml" accesskey="1"> Verifique el correo de voz </a> <br>
<a href="mail.chtml" accesskey="2"> Verifique el correo electrónico </a> <br>
<a href="games.chtml" accesskey="3"> Ejecute un juego </a>
</body>
</html>
```

Figura 7-53. Un ejemplo de un archivo cHTML.

Auque el cliente está algo limitado, el servidor i-mode es una computadora completamente equipada, con todas las características comunes. Soporta CGI, Perl, PHP, JSP, ASP y todo lo demás que normalmente soportan los servidores Web.

En la figura7-54 se muestra una breve comparación de cómo están implementados WAP e i-mode en los sistemas de primera generación. Aunque algunas de las diferencias parecen pequeñas, por lo general son importantes. Por ejemplo, las personas de 15 años no tienen tarjetas de crédito, por lo que poder comprar productos a través del comercio electrónico y cargarlos al recibo telefónico marcan una gran diferencia en su interés en el sistema.

Para mayor información acerca de i-mode, vea (Frengle, 2002, y Vacca, 2002).

Web inalámbrica de segunda generación

Se suponía que WAP 1.0, basado en estándares internacionales reconocidos, sería una herramienta seria para la gente de negocios en movimiento. Fracasó. I-mode era un juguete electrónico para los adolescentes japoneses que utilizaban un todo propietario. Fue un gran éxito. ¿Qué sucedió a continuación? Cada lado aprendió algo de la primera generación de la Web inalámbrica. El consorcio WAP aprendió que el contenido importa. No tener un gran número de sitios Web que hablan su lenguaje de marcado es fatal. NTT DoCoMo aprendió que un sistema propietario cerrado, estrechamente enlazado con microteléfonos y con la cultura japonesa no es un buen producto de exportación. La conclusión que ambos lados aprendieron es que para convencer a una gran cantidad de sitios Web para que coloquen su contenido en el formato de usted, es necesario tener un lenguaje de marcado estable y abierto que sea aceptado de manera universal. Las guerras de los formatos no son buenas para los negocios.

Ambos servicios están cerca de entrar a la tecnología de la segunda generación de la Web inalámbrica. WAP 2.0 existió primero, por lo que lo utilizaremos para nuestro ejemplo. WAP 1.0

Característica	WAP	I-mode
Qué es	Pila de protocolos	Servicio
Dispositivo	Microteléfono, PDA, notebook	Microteléfono
Acceso	Marcado telefónico	Siempre activo
Red subyacente	Conmutación de circuitos	Dos: circuitos + paquetes
Tasa de datos	9600 bps	9600 bps
Pantalla	Monocroma	A color
Lenguaje de marcado	WML (aplicación XML)	cHTML
Lenguaje de secuencia de comandos	WMLscript	Ninguno
Cargos por uso	Por minuto	Por paquete
Pago por compras	Tarjeta de crédito	Recibo telefónico
Pictogramas	No	Sí
Estandarización	Estándar abierto del foro WAP	Propietario NTT DoCoMo
En dónde se utiliza	Europa, Japón	Japón
Usuario típico	Hombre de negocios	Personas jóvenes

Figura 7-54. Comparación entre WAP de primera generación e i-mode.

obtuvo algunas cosas correctas, y éstas aún funcionan. Por ejemplo, WAP se puede transportar en una variedad de redes. La primera generación utilizó las redes de conmutación de circuitos, pero las redes de conmutación de paquetes siempre fueron una opción y aún lo siguen siendo. Es posible que los sistemas de segunda generación utilicen la conmutación de paquetes, por ejemplo, GPRS. Además, WAP estaba destinado inicialmente para soportar una amplia variedad de dispositivos, desde teléfonos móviles hasta poderosas computadoras, y todavía lo está.

WAP 2.0 también tiene algunas nuevas características. Las más significativas son:

1. Modelo *push* (de actualización automática) y modelo *pull* (de recepción automática).
2. Soporte para integrar la telefonía en las aplicaciones.
3. Mensajería multimedia.
4. Inclusión de 264 pictogramas.
5. Interacción con un dispositivo de almacenamiento.
6. Soporte en el navegador para *plug-ins*.

El modelo *pull* es bien conocido: el cliente solicita una página y la obtiene. El modelo *push* soporta el arribo de datos sin que se le solicite, como una retroalimentación continua de la información de la bolsa o alertas de tráfico.

La voz y los datos están comenzando a combinarse, y WAP 2.0 los soporta en una variedad de formas. Vimos un ejemplo de esto anteriormente con la capacidad de i-mode de llamar a un

número telefónico mediante un hipervínculo de icono o fragmento de texto de la pantalla. Además del correo electrónico y la telefonía, se soporta la mensajería multimedia.

La gran popularidad del emoji de i-mode estimuló al consorcio WAP a inventar 264 de sus propios emoji. Las categorías incluyen animales, aplicaciones, vestido, estados de ánimo, comida, cuerpo humano, género, mapas, música, plantas, deportes, fechas, herramientas, vehículos, armas y clima. Es interesante que el estándar sólo nombra cada pictograma; no proporciona el mapa de bits real, probablemente por miedo a que la representación de "adormilado" o "abrazo" de algunas culturas podría ser insultante para otras. I-mode no tuvo ese problema debido a que estaba destinado para un solo país.

El hecho de proveer una interfaz para almacenamiento no significa que cada teléfono WAP 2.0 tendrá un disco duro grande. La memoria ROM también es un dispositivo de almacenamiento. Una cámara inalámbrica con capacidad WAP podría utilizar la memoria ROM para almacenamiento temporal de imágenes antes de emitir la mejor imagen a Internet.

Por último, los *plug-ins* pueden extender las capacidades del navegador. También se proporciona un lenguaje de secuencias de comandos.

En WAP 2.0 también hay diversas diferencias técnicas. Las dos más significativas tienen que ver con la pila de protocolos y con el lenguaje de marcado. WAP 2.0 continúa soportando la antigua pila de protocolos de la figura 7-48, pero también soporta la pila estándar de Internet con TCP y HTTP/1.1. Sin embargo, se realizaron cuatro pequeños (pero compatibles) cambios (para simplificar el código) a TCP: (1) uso de una ventana fija de 64 KB, (2) inicio rápido, (3) una MTU máxima de 1500 byte y (4) un algoritmo de retransmisión ligeramente diferente. TLS es el protocolo de seguridad de la capa de transporte estandarizado por IETF; lo examinaremos en el capítulo 8. Muchos dispositivos iniciales probablemente contendrán ambas pilas, como se muestra en la figura 7-55.

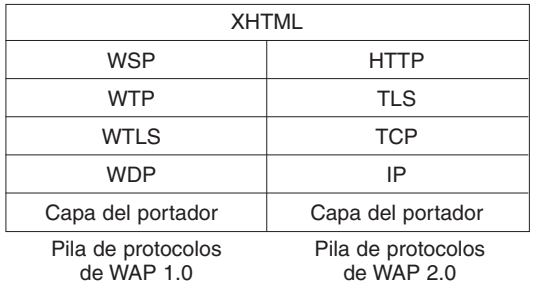


Figura 7-55. WAP 2.0 soporta dos pilas de protocolos.

La otra diferencia técnica con respecto de WAP 1.0 es el lenguaje de marcado. WAP 2.0 soporta XHTML Basic, que está diseñado para dispositivos inalámbricos pequeños. Debido a que NTT DoCoMo también está de acuerdo en soportar este subconjunto, los diseñadores de sitios Web pueden utilizar este formato y saber que sus páginas funcionarán en la Internet fija y en todos los dispositivos inalámbricos. Estas decisiones terminarán con las guerras de formatos de lenguaje de marcado que impiden el crecimiento de la industria de la Web inalámbrica.

Tal vez sean necesarias unas palabras acerca de XHTML Basic. Está diseñado para teléfonos móviles, televisiones, PDAs, máquinas vendedoras (refrescos, golosinas), localizadores, carros, máquinas tragamonedas y de juego e, incluso, relojes. Por esta razón, no soporta hojas de estilo, secuencias de comandos o tramas, pero sí soporta la mayoría de las etiquetas estándar. Están agrupadas en 11 módulos. Algunos son requeridos; algunos son opcionales. Todos están en XML. En la figura 7-56 se listan los módulos y algunas etiquetas de ejemplo. No explicaremos todas las etiquetas de ejemplo, pero puede encontrar mayor información en *www.w3.org*.

Módulo	¿Requerido?	Función	Etiquetas de ejemplo
Estructura	Sí	Estructura de documentos	body, head, html, title
Texto	Sí	Información	br, code, dfn, em, h*n*, kbd, p, strong
Hipertexto	Sí	Hipervínculos	a
Lista	Sí	Listas de elementos	dl, dt, dd, ol, ul, li
Formularios	No	Formularios de relleno	form, input, label, option, textarea
Tablas	No	Tablas rectangulares	caption, table, td, th, tr
Imagen	No	Imágenes	img
Objeto	No	Subprogramas, mapas, etcétera	object, param
Metainformación	No	Información extra	meta
Vínculo	No	Similar a <a>	link
Base	No	Punto de inicio de URL	base

Figura 7-56. Los módulos y etiquetas de XHTML Basic.

A pesar del acuerdo del uso de XHTML Basic, una amenaza asecha a WAP e i-mode: 802.11. Se supone que la Web inalámbrica de segunda generación se debe ejecutar a 384 kbps, una velocidad mucho mayor que los 9600 bps de la primera generación, pero mucho menor que los 11 Mbps o 54 Mbps ofrecidos por 802.11. Por supuesto, 802.11 no está en todos lados, pero conforme más restaurantes, hoteles, tiendas, compañías, aeropuertos, estaciones de autobús, museos, universidades, hospitales y otras organizaciones decidan instalar estaciones base para sus empleados y clientes, tal vez haya mayor cobertura en las áreas urbanas y las personas estén dispuestas a caminar unas cuadras para sentarse en un restaurante de comida rápida con capacidad de 802.11 para tomarse una tasa de café y enviar correo electrónico. Tal vez los negocios coloquen de manera rutinaria logos de 802.11 junto a los que muestran las tarjetas de crédito que dichos negocios aceptan, y por la misma razón: para atraer a los clientes. Los mapas de la ciudad (descargables, naturalmente) podrían mostrar con color verde las áreas cubiertas y en color rojo las áreas sin cobertura, de manera que las personas puedan vagar de estación base a estación base, al igual que los nómadas se trasladan de oasis a oasis en el desierto.

Aunque los restaurantes de comida rápida puedan instalar rápidamente estaciones base 802.11, tal vez los granjeros no puedan hacerlo, por lo que la cobertura será desigual y limitada a las áreas del centro de la ciudad, debido al rango limitado de 802.11 (unos cuantos cientos de metros por lo mucho). Esto podría llevar a dispositivos inalámbricos de modo dual que utilicen 802.11 si pueden captar una señal y regresar a WAP si no pueden hacerlo.

7.4 MULTIMEDIA

La Web inalámbrica es un desarrollo nuevo y excitante, pero no es el único. Para muchas personas, la multimedia es el Santo Grial de las redes. Cuando se menciona la palabra, los *propeller heads* y las demandas legales comienzan a babear como si vieran un gran banquete. El primero ve retos técnicos inmensos para proporcionar a cada casa vídeo (interactivo) bajo demanda. El último ve en ello ganancias inmensas. Debido a que la multimedia requiere un alto ancho de banda, hacer que funcione en conexiones fijas es difícil. Incluso el vídeo de calidad VHS a través de sistemas inalámbricos está a algunos años de distancia, por lo que nos enfocaremos en los sistemas cableados.

Literalmente, multimedia son dos o más medios. Si el editor de este libro quisiera unirse a la euforia actual por la multimedia, podría anunciar que el libro usa tecnología multimedia. A fin de cuentas, contiene dos medios: texto y gráficos (las figuras). No obstante, cuando la mayoría de la gente habla de multimedia, por lo general se refiere a la combinación de dos o más **medios continuos,** es decir, medios que tienen que ejecutarse durante cierto intervalo de tiempo bien definido, generalmente con alguna interacción con el usuario. En la práctica, por lo común los dos medios son audio y vídeo, es decir, sonido más imágenes en movimiento.

Sin embargo, muchas personas con frecuencia también se refieren al audio puro, como la telefonía de Internet o la radio en Internet, como multimedia, pero no lo es. Realmente, un mejor término es **medios de flujo continuo,** pero también consideraremos el audio en tiempo real como multimedia. En las siguientes secciones analizaremos la forma en que las computadoras procesan el audio y el vídeo, cómo están comprimidos y algunas aplicaciones de red de estas tecnologías. Para un análisis (de tres volúmenes) sobre la multimedia en red, vea (Steinmetz y Nahrstedt, 2002; Steinmetz y Nahrstedt, 2003a, y Steinmetz y Nahrstedt, 2003b).

7.4.1 Introducción al audio digital

Una onda de audio (sonido) es una onda acústica (de presión) de una dimensión. Cuando una onda acústica entra en el oído, el tímpano vibra, causando que los pequeños huesos del oído interno vibren con él, enviando pulsos nerviosos al cerebro. El escucha percibe estos pulsos como sonido. De manera parecida, cuando una onda acústica incide en un micrófono, éste genera una señal eléctrica, que representa la amplitud del sonido como una función del tiempo. La representación, procesamiento, almacenamiento y transmisión de tales señales de audio es una parte principal del estudio de los sistemas multimedia.

La gama de frecuencias perceptibles por el oído humano va de 20 Hz a 20,000 Hz, aunque algunos animales, principalmente los perros, pueden escuchar frecuencias más altas. El oído escucha de manera logarítmica, por lo que la relación entre dos sonidos de amplitudes A y B se expresa convencionalmente en **dB** (**decibeles**) de acuerdo con la fórmula

$$\text{dB} = 10 \log_{10} (A/B)$$

Si definimos como 0 dB el límite inferior de la audibilidad (una presión de unas 0.0003 dinas/cm^2) para una onda senoidal de 1 kHz, una conversación ordinaria es de unos 50 dB y el umbral del dolor es de 120 dB, lo que representa una gama dinámica de un factor de un millón.

El oído es sorprendentemente sensible a variaciones de sonido que duran apenas unos milisegundos. En cambio, el ojo no nota cambios en el nivel de luz que duran unos cuantos milisegundos. El resultado de esta observación es que fluctuaciones de apenas unos cuantos milisegundos durante una transmisión multimedia afectan la calidad del sonido percibido más que a la calidad de la imagen percibida.

Las ondas de audio pueden convertirse a una forma digital mediante un **ADC** (**convertidor analógico a digital**). Un ADC toma un voltaje eléctrico como entrada y genera un número binario como salida. En la figura 7-57(a) se muestra un ejemplo de onda senoidal. Para representar esta señal de manera digital, simplemente la muestreamos cada ΔT segundos, como lo muestra la altura de las barras de la figura 7-57(b). Si una onda de sonido no es una onda senoidal pura, sino una superposición de ondas senoidales en las que la componente de más alta frecuencia es *f,* entonces el teorema de Nyquist (vea el capítulo 2) establece que es suficiente tomar muestras a una frecuencia 2*f*. Muestrear a una frecuencia mayor no tiene ningún valor, porque no están presentes las frecuencias mayores que serían detectadas por dicho muestreo.

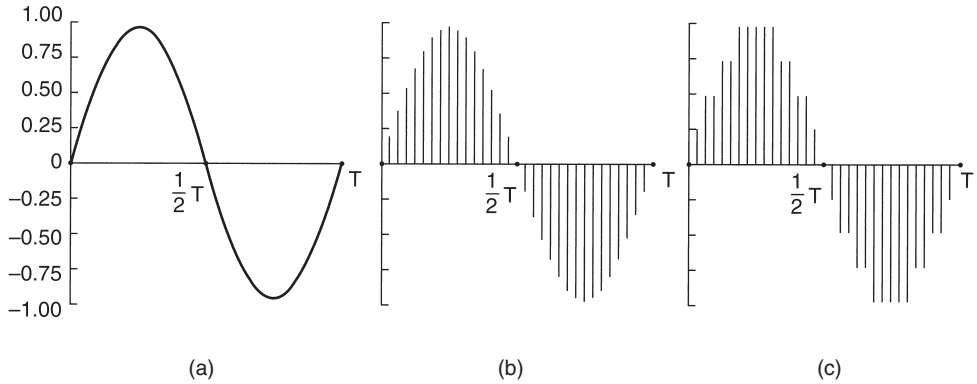


Figura 7-57. (a) Onda senoidal. (b) Muestreo de la onda senoidal. (c) Cuantización de las muestras a 4 bits.

Las muestras digitales nunca son exactas. Las muestras de la figura 7-57(c) sólo permiten nueve valores, de −1.00 a + 1.00 en incrementos de 0.25. Una muestra de 8 bits permitirá 256 valores diferentes. Una muestra de 16 bits permitirá 65,536 valores diferentes. El error introducido por la cantidad finita de bits por muestra se llama **ruido de cuantización.** Si éste es demasiado grande, el oído lo detecta.

Dos ejemplos bien conocidos de sonido muestreado son el teléfono y los discos compactos de audio. La modulación de código de pulso, como la usada en el sistema telefónico, emplea muestras de 8 bits, 8000 veces por segundo. En Norteamérica y Japón, 7 bits son para datos y 1 para control; en Europa, los 8 bits son para datos. Este sistema da una tasa de datos de 56,000 bps o 64,000 bps. Con sólo 8000 muestras/seg, las frecuencias por arriba de 4 kHz se pierden.

Los CDs de audio son digitales, con una tasa de muestreo de 44,100 muestras/seg, suficientes para capturar frecuencias de hasta 22,050 Hz, lo que es bueno para la gente, malo para los perros. Cada una de las muestras tiene 16 bits, y es lineal dentro de la gama de amplitudes. Observe que las muestras de 16 bits permiten sólo 65,536 valores diferentes, aunque la gama dinámica del oído es de aproximadamente 1 millón si se mide en pasos del tamaño del sonido audible más pequeño. Por lo tanto, el uso de sólo 16 bits por muestra genera ruido de cuantización (aunque no se cubre la gama dinámica completa; se supone que los CDs no deben lastimar). Con 44,100 muestras/seg de 16 bits cada una, un CD de audio necesita un ancho de banda de 705.6 kbps para monofónico y 1.411 Mbps para estéreo. Si bien esto es menos de lo que necesita el vídeo (vea más adelante), aun así se requiere un canal T1 completo para transmitir en tiempo real sonido estéreo de calidad CD.

Las computadoras pueden procesar con facilidad mediante software el sonido digital. Existen docenas de programas para que las computadoras personales permitan que los usuarios graben, desplieguen, editen, mezclen y almacenen ondas de sonido de múltiples fuentes. En la actualidad, casi toda la grabación y edición profesional de sonido es digital.

La música, por supuesto, es simplemente un caso especial del audio general, pero es importante. Otro caso especial muy importante es la voz. La voz humana tiende a estar en el rango de 600 a 6000 Hz. La voz se compone de vocales y consonantes, las cuales tienen propiedades diferentes. Las vocales se producen cuando el tracto vocal está libre, produciendo resonancias cuya frecuencia fundamental depende del tamaño y de la forma del sistema vocal y de la posición de la lengua y mandíbula de quien habla. Estos sonidos son casi periódicos en intervalos de aproximadamente 30 mseg. Las consonantes se producen cuando el tracto vocal está bloqueado parcialmente. Estos sonidos son menos regulares que las vocales.

Algunos sistemas de transmisión y generación de voz utilizan modelos del sistema vocal para reducir la voz a unos cuantos parámetros (por ejemplo, los tamaños y las formas de diversas cavidades), en lugar de simplemente muestrear la forma de onda de la voz. Sin embargo, la forma en que funcionan estos codificadores de voz está más allá del alcance de este libro.

7.4.2 Compresión de audio

El audio con calidad de CD requiere un ancho de banda de transmisión de 1.411 Mbps, como acabamos de ver. Claramente, la compresión sustancial se necesita para hacer que la transmisión a través de Internet sea práctica. Por esta razón, se han desarrollado varios algoritmos de compresión de audio. Tal vez el más popular es el audio MPEG, que tiene tres capas (variantes), de las cuales **MP3** (**capa de audio 3 de MPEG**) es la más poderosa y mejor conocida. En Internet hay cantidades considerables de música en formato MP3, no todos legales, lo que ha resultado en varias demandas de los artistas y propietarios de derechos de autor. MP3 pertenece a la porción de audio del estándar de compresión de vídeo de MPEG. Más adelante en este capítulo analizaremos la compresión de vídeo; por ahora veremos la compresión de audio.

La compresión de audio se puede realizar de una de dos formas. En la **codificación de forma de onda** la señal se transforma de manera matemática en sus componentes de frecuencia mediante

una transformación de Fourier. La figura 2-1(a) muestra una función de ejemplo de tiempo y sus amplitudes de Fourier. Por lo tanto, la amplitud de cada componente se codifica en una forma mínima. El objetivo es reproducir la forma de onda de manera precisa en el otro extremo utilizando los menos bits posibles.

La otra forma, **codificación perceptual,** aprovecha ciertas fallas del sistema auditivo humano para codificar una señal a fin de que suene de la misma forma para un escucha, aunque dicha señal luzca de manera diferente en un osciloscopio. La codificación perceptual se basa en la ciencia de **psicoacústica** —cómo perciben las personas un sonido. MP3 se basa en la codificación perceptual.

La propiedad clave de la codificación perceptual es que algunos sonidos pueden **enmascarar** otros sonidos. Imagine que está difundiendo un concierto de flauta en vivo en un día caluroso de verano. De repente, un grupo de trabajadores que está cerca enciende sus martillos perforadores y comienza a romper la calle. Ya nadie puede escuchar la flauta. Sus sonidos han sido enmascarados por los de los martillos perforadores. Para propósitos de transmisión, ahora es suficiente con codificar sólo la banda de frecuencia utilizada por los martillos perforadores, pues de cualquier forma las personas no pueden escuchar la flauta. A esto se le conoce como **enmascaramiento de frecuencia** —la capacidad que tiene un sonido fuerte en una banda de frecuencia de ocultar un sonido más suave en otra banda de frecuencia, el cual podría ser audible si el sonido fuerte no estuviera presente. De hecho, incluso después de que los martillos perforadores pararan, la flauta no se escucharía por un periodo corto debido a que el oído reduce su ganancia cuando los martillos comienzan y toma un tiempo finito para aumentarlo nuevamente. Este efecto se conoce como **enmascaramiento temporal.**

Para hacer que estos efectos sean más cuantitativos, imagine el experimento 1. Una persona en un salón silencioso se pone unos audífonos que están conectados a la tarjeta de sonido de una computadora. Ésta genera una onda senoidal pura a 100 Hz, pero incrementa la potencia de manera gradual. Se le indica a la persona que pulse una tecla cuando escuche el tono. La computadora graba el nivel de potencia actual y después repite el experimento a 200 Hz, 300 Hz y demás frecuencias hasta el límite del oído humano. Cuando se calcula un promedio a partir de varias personas, un gráfico de registro a registro de cuánta potencia se necesita para que un tono sea audible luce como el que se muestra en la figura 7-58(a). Una consecuencia directa de esta curva es que nunca es necesario codificar ninguna frecuencia cuya potencia esté por debajo del umbral de audibilidad. Por ejemplo, si la potencia de 100 Hz fuera 20 dB en la figura 7-58(a), se podría omitir de la salida sin ninguna pérdida perceptible de calidad debido a que 20 dB a 100 Hz están debajo del nivel de audibilidad.

Ahora considere el experimento 2. La computadora realiza otra vez el experimento 1, pero esta vez con una onda senoidal de amplitud constante a, digamos, 150 Hz, superimpuesta en la frecuencia de prueba. Lo que descubrimos es que se incrementa el umbral de audibilidad para las frecuencias que están cerca de 150 Hz, como se muestra en la figura 7-58(b).

La consecuencia de esta nueva observación es que al mantener un registro de cuáles señales están siendo enmascaradas por señales más poderosas de bandas de frecuencia cercanas, podemos omitir más y más frecuencias en la señal codificada, lo que ahorra bits. En la figura 7-58, la señal a 125 Hz se puede omitir por completo de la salida y nadie notará la diferencia. Aunque una

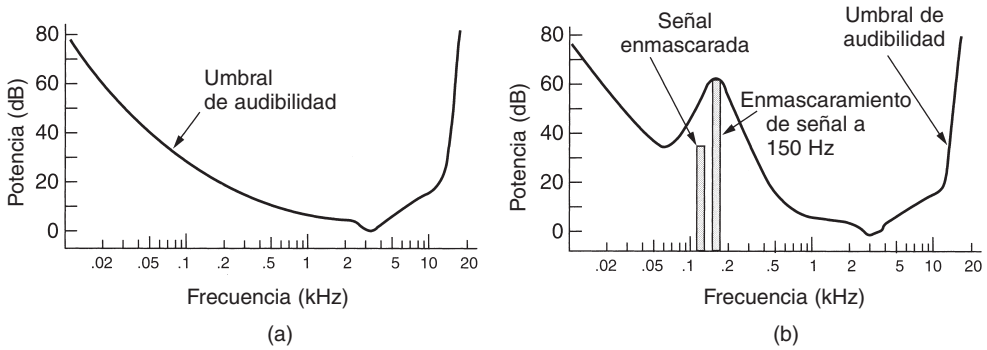


Figura 7-58. (a) Umbral de audibilidad como una función de frecuencia. (b) Efecto de enmascaramiento.

señal poderosa se detenga en alguna banda de frecuencia, si conocemos sus propiedades de enmascaramiento temporal, podemos continuar omitiendo las frecuencias enmascaradas por algún intervalo de tiempo mientras el oído se recupera. La esencia de MP3 es transformar mediante un análisis de Fourier el sonido para obtener la potencia de cada frecuencia y después transmitir sólo las frecuencias no enmascaradas, codificando éstas con los menos bits posibles.

Con esta información como base, ahora podemos ver cómo se realiza la codificación. La compresión de audio se realiza muestreando la forma de onda a 32, 44.1 o 48 kHz. El muestreo puede realizarse en uno de dos canales, en cualquiera de cuatro configuraciones:

1. Monofónica (un solo flujo de entrada).
2. Monofónica dual (por ejemplo, una pista sonora en inglés y una en japonés).
3. Estéreo separado (cada canal se comprime por separado).
4. Estéreo unido (redundancia intercanal completamente explotada).

Primero se elige la tasa de bits de salida. MP3 puede comprimir un CD de estéreo de rock and roll a 96 kbps con una pérdida de calidad apenas perceptible, incluso para los fanáticos del rock and roll que no tienen pérdida del oído. Para un concierto de piano, se necesitan por lo menos 128 kbps. Esto difiere porque la relación señal a ruido para el rock and roll es más alta que la de un concierto de piano (en el sentido de la ingeniería). También es posible elegir tasas de salida más bajas y aceptar alguna pérdida en la calidad.

Después, las muestras se procesan en grupos de 1152 (aproximadamente 26 mseg). Cada grupo primero se pasa a través de 32 filtros digitales para obtener 32 bandas de frecuencia. Al mismo tiempo, la entrada se coloca en un modelo psicoacústico para determinar las frecuencias enmascaradas. A continuación, cada una de las 32 bandas de frecuencia se transforman aún más para proporcionar una resolución espectral más fina.

En la siguiente fase, los bits disponibles se dividen entre las bandas; la mayoría de los bits se asignan a las bandas con la mayor potencia espectral no enmascarada, a las bandas no enmascaradas con menos potencia espectral se les asignan muy pocos bits y a las bandas enmascaradas no se les

asignan bits. Por último, los bits se codifican mediante la codificación de Huffman, que asigna códigos cortos a números que aparecen frecuentemente, y códigos largos a aquellos que no ocurren con frecuencia.

En realidad hay mucho más que decir. También se utilizan varias técnicas para la reducción del ruido, el suavizado y la explotación de la redundancia de intercanal, si es posible, pero éstas están más allá del alcance de este libro. Una introducción matemática a este proceso se proporciona en (Pan, 1995).

7.4.3 Audio de flujo continuo

Ahora movámonos de la tecnología del audio digital a tres de sus aplicaciones de red. La primera es el audio de flujo continuo, es decir, escuchar el sonido a través de Internet. A esto también se le conoce como música bajo demanda. En las siguientes dos veremos la radio en Internet y la voz sobre IP, respectivamente.

Internet está lleno de sitios Web de música, muchos de los cuales listan títulos de canciones en los que los usuarios pueden hacer clic para reproducir esas canciones. Algunos de éstos son sitios gratuitos (por ejemplo, las nuevas bandas que buscan publicidad); otros requieren un pago a cambio de la música, aunque éstos con frecuencia también ofrecen algunas muestras gratis (por ejemplo, los primeros 15 seg de una canción). En la figura 7-59 se muestra la forma más directa para hacer que se reproduzca la música.

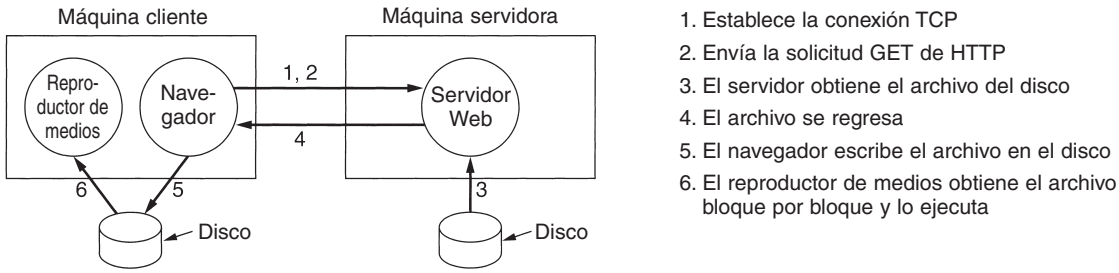


Figura 7-59. Una forma directa de implementar en una página Web música en la que se pueda hacer clic.

El proceso inicia cuando el usuario hace clic en una canción. A continuación el navegador entra en acción. El paso 1 consiste en que éste establezca una conexión TCP con el servidor Web con el que la canción está vinculada. El paso 2 consiste en enviar una solicitud *GET* en HTTP para pedir la canción. A continuación (pasos 3 y 4), el servidor obtiene la canción (que es simplemente un archivo en MP3 o en algún otro formato) del disco y la regresa al navegador. Si el archivo es más grande que la memoria del servidor, tal vez obtenga y envíe la música un bloque a la vez.

El navegador investiga, mediante el tipo MIME, por ejemplo, *audio/mp3* (o la extensión de archivo), cómo se supone que debe desplegar el archivo. Por lo general, habrá una aplicación auxiliar, como RealOne Player, el Reproductor de Windows Media o Winamp, asociado con este tipo de archivos. Debido a que la forma usual de que el navegador se comunique con una aplicación

auxiliar es escribir el contenido en un archivo de trabajo, primero guardará en el disco todo el archivo de música como un archivo de trabajo (paso 5). Después iniciará el reproductor de medios y pasará el nombre del archivo de trabajo. En el paso 6, el reproductor de medios comienza a obtener y a reproducir la música bloque por bloque.

Al principio, este enfoque es correcto y reproducirá la música. El único problema es que la canción completa debe transmitirse a través de la red antes de que comience la música. Si la canción mide 4 MB (un tamaño típico para una canción MP3) y el módem es de 56 kbps, el usuario obtendrá casi 10 minutos de silencio mientras la canción se descarga. No a todos los amantes de la música les gusta esta idea. Especialmente debido a que la siguiente canción también iniciará después de 10 minutos de descarga, y así sucesivamente.

Para resolver este problema sin cambiar la forma en que funciona el navegador, los sitios de música han adoptado el siguiente esquema. El archivo vinculado al título de la canción no es el archivo de música real. En su lugar, es lo que se llama un **metaarchivo,** que es un archivo muy pequeño que sólo nombra a la música. Un metaarchivo típico podría ser una sola línea de texto ASCII y podría lucir como lo siguiente:

```
rtsp://joes-audio-server/song-0025.mp3
```

Cuando el navegador obtiene el archivo de una línea, lo escribe en el disco en un archivo de trabajo, inicia el reproductor de medios como una aplicación auxiliar, y le entrega el nombre del archivo de trabajo, como es usual. A continuación el reproductor de medios lee dicho archivo y ve que contiene un URL. Enseguida contacta al servidor *joes-audio-server* y le pide la canción. Observe que el navegador ya no está en el ciclo.

En muchos casos, el servidor nombrado en el metaarchivo no es el mismo que el servidor Web. De hecho, por lo general ni siquiera es un servidor HTTP, sino un servidor de medios especializado. En este ejemplo, el servidor de medios utiliza **RTSP (Protocolo de Transmisión en Tiempo Real),** como se indica en el nombre de esquema *rtsp*. Éste se describe en el RFC 2326.

El reproductor de medios tiene cuatro trabajos principales:

1. Administrar la interfaz de usuario.
2. Manejar los errores de transmisión.
3. Descomprimir la música.
4. Eliminar la fluctuación.

En la actualidad, la mayoría de los reproductores de medios tienen una interfaz de usuario brillante que algunas veces simula una unidad de estéreo, con botones, palancas, barras deslizantes y despliegues visuales. Por lo general hay paneles frontales intercambiables, llamados máscaras (*skins*), que el usuario puede colocar en el reproductor. El reproductor de medios tiene que manejar todo esto e interactuar con el usuario.

Su segundo trabajo es tratar con los errores. La transmisión de música en tiempo real raramente utiliza TCP porque un error y una retransmisión podrían introducir un hueco demasiado grande en la música. En su lugar, la transmisión real por lo común se realiza con un protocolo como RTP,

el cual estudiamos en el capítulo 6. Al igual que la mayoría de los protocolos en tiempo real, la capa de RTP se encuentra encima de UDP, por lo que los paquetes pueden perderse. El reproductor es quien tiene que tratar esto.

En algunos casos, la música se intercala para facilitar el manejo de errores. Por ejemplo, un paquete podría contener 220 muestras de estéreo, cada una con un par de números de 16 bits, lo que normalmente está bien para 5 mseg de música. Pero tal vez el protocolo envíe todas las muestras impares en un intervalo de 10 mseg en un paquete, y todas las muestras pares en el siguiente. Por lo tanto, un paquete perdido no representa un hueco de 5 mseg en la música, sino la pérdida de cualquier otra muestra durante 10 mseg. Esta pérdida puede manejarse fácilmente haciendo que el reproductor de medios realice una interpolación mediante las muestras anterior y posterior para estimar el valor faltante.

En la figura 7-60 se ilustra el uso de intercalación para la recuperación de errores. Cada paquete contiene las muestras de tiempo alternadas durante un intervalo de 10 mseg. En consecuencia, perder el paquete 3, como se muestra, no crea un hueco en la música, sólo reduce la resolución temporal por algún tiempo. Los valores faltantes pueden interpolarse para proporcionar música continua. Este esquema particular sólo funciona con el muestreo sin comprimir, pero muestra la forma en que un código adecuado puede hacer que un paquete perdido signifique menos calidad en lugar de un hueco de tiempo. Sin embargo, el RFC 3119 proporciona un esquema que funciona con el audio comprimido.

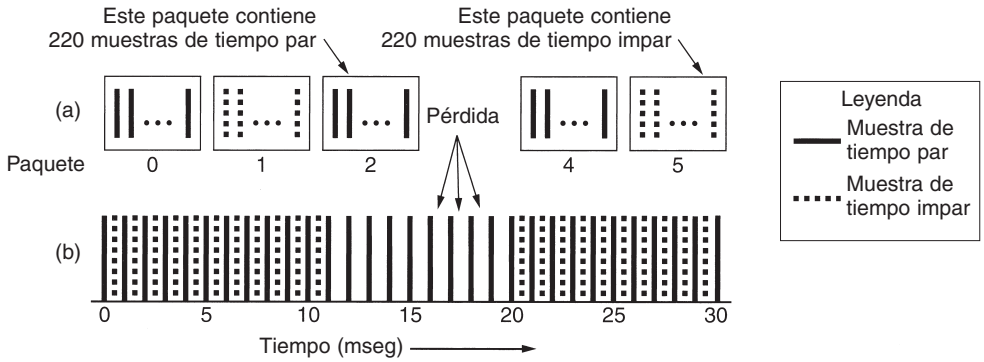


Figura 7-60. Cuando los paquetes transportan muestras alternas, la pérdida de un paquete reduce la resolución temporal en lugar de crear un hueco en el tiempo.

El tercer trabajo del reproductor de medios es descomprimir la música. Aunque esta tarea es intensa para la computadora, es muy directa.

El cuarto trabajo es eliminar la fluctuación, el veneno de todos los sistemas en tiempo real. Todos los sistemas de audio de flujo continuo inician almacenando en el búfer aproximadamente de 10 a 15 seg de música antes de comenzar a reproducir, como se muestra en la figura 7-61. Idealmente, el servidor continuará llenando el búfer a la tasa exacta a la que el reproductor de medios lo vacía, aunque en realidad esto no podría suceder, por lo que la retroalimentación en el ciclo podría ser útil.

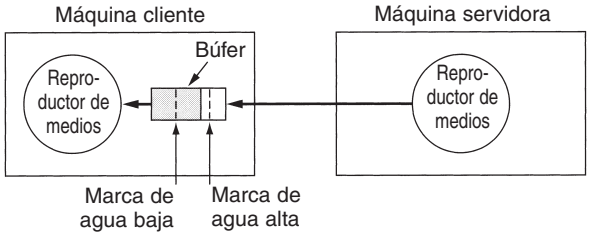


Figura 7-61. El reproductor de medios almacena en el búfer la entrada del servidor de medios y reproduce desde el servidor en lugar de hacerlo directamente desde la red.

Se pueden utilizar dos métodos para mantener lleno el búfer. Con un **servidor *pull*** (de recepción automática), siempre y cuando haya espacio en el búfer para otro bloque, el reproductor de medios simplemente sigue enviando al servidor mensajes en los que le solicita un bloque adicional. Su objetivo es mantener el búfer lo más lleno posible.

La desventaja de un servidor *pull* son todas las solicitudes de datos innecesarias. El servidor sabe que ha enviado el archivo completo, de modo que, ¿por qué el reproductor sigue enviando solicitudes? Por esta razón, raramente se utiliza.

Con un **servidor *push*** (de actualización automática), el reproductor de medios envía una solicitud *PLAY* y el servidor simplemente continúa enviándole datos. Aquí hay dos posibilidades: el servidor de medios se ejecuta a la velocidad normal de reproducción o se ejecuta más rápido. En ambos casos, algunos datos se almacenan en el búfer antes de que inicie la reproducción. Si el servidor se ejecuta a la velocidad normal de reproducción, los datos que provengan de él se agregan al final del búfer y el reproductor elimina los datos del frente del búfer para reproducirlos. Siempre y cuando todo funcione a la perfección, la cantidad de datos en el búfer permanece constante. Este esquema es sencillo debido a que no se necesitan mensajes de control en ninguna dirección.

El otro método *push* es hacer que el servidor envíe datos a una velocidad mayor que la necesaria. La ventaja aquí es que si no se puede garantizar que el servidor se ejecute a una tasa regular, tiene la oportunidad de reponerse si se queda atrás. Sin embargo, un problema aquí son los desbordamientos de búfer potenciales si el servidor puede enviar datos con más rapidez que con la que se consumen (y debe poder hacer esto para evitar los huecos).

La solución es que el reproductor de medios defina una **marca de agua baja** y una **marca de agua alta** en el búfer. Básicamente, el servidor sólo envía datos hasta que el búfer llega a la marca de agua alta. A continuación el reproductor de medios le indica que haga una pausa. Puesto que los datos continuarán llegando hasta que el servidor obtenga la solicitud de pausa, la distancia entre la marca de agua alta y el final del búfer tiene que ser mayor que el producto del retardo de ancho de banda de la red. Después de que el servidor se detenga, el búfer comenzará a vaciarse. Cuando llegue a la marca de agua baja, el reproductor de medios indicará al servidor de medios que comience de nuevo. La marca de agua baja tiene que colocarse de manera que la subutilización de búfer no ocurra.

Para operar un servidor *push*, el reproductor de medios necesita un control remoto para él. RTSP, que se define en el RFC 2326, proporciona el mecanismo para que el reproductor controle al servidor. No proporciona el flujo de datos, que por lo general es RTP. En la figura 7-62 se listan los principales comandos que proporciona RTSP.

Comando	Acción del servidor
DESCRIBE	Lista los parámetros de los medios
SETUP	Establece un canal lógico entre el reproductor y el servidor
PLAY	Comienza a enviar los datos al cliente
RECORD	Comienza a aceptar los datos del cliente
PAUSE	Detiene de manera temporal el envío de datos
TEARDOWN	Libera el canal lógico

Figura 7-62. Los comandos RTSP del reproductor al servidor.

7.4.4 Radio en Internet

Una vez que pudo ser posible difundir audio a través de Internet, las estaciones de radio comerciales tuvieron la idea de transmitir su contenido a través de Internet, así como a través de aire. No mucho tiempo después, las estaciones de radio universitarias comenzaron a colocar su señal en Internet. Después, los *estudiantes* comenzaron sus propias estaciones de radio. Con la tecnología actual, casi cualquier persona puede iniciar una estación de radio. El área de la radio de Internet es muy nueva y se encuentra en un proceso de cambio, pero vale la pena decir un poco más.

Hay dos métodos generales para la radio en Internet. En el primero, los programas se graban con anterioridad y se almacenan en disco. Los escuchas pueden conectarse a los archivos de la estación de radio y obtener y descargar cualquier programa para escucharlo. De hecho, esto es exactamente lo mismo que el audio de flujo continuo que acabamos de analizar. También es posible almacenar cada programa justo después de que se transmite en vivo, por lo que el archivo sólo está atrasado, digamos, media hora, o menos con respecto de la transmisión en vivo. Este método tiene las ventajas de que es fácil de llevar a cabo, de que las demás técnicas que hemos visto hasta aquí también lo son y de que los escuchas pueden elegir de entre todos los programas del archivo.

El otro método es difundir el contenido en vivo a través de Internet. Algunas estaciones transmiten a través de aire y a través de Internet de manera simultánea, pero cada vez hay más estaciones de radio que sólo transmiten a través de Internet. Algunas de las técnicas que se aplican al audio de flujo continuo también se aplican a la radio en vivo por Internet, pero también hay algunas diferencias clave.

Un punto que es el mismo es la necesidad de almacenar en el búfer del usuario para disminuir la fluctuación. Al colectar 10 o 15 segundos de radio antes de comenzar la reproducción, el audio puede escucharse bien, aunque suceda fluctuación sustancial a través de la red. En tanto todos los paquetes lleguen antes de que se necesiten, no importa cuándo lleguen.

Una diferencia clave es que el audio de flujo continuo puede enviarse a una tasa mayor que la de reproducción, puesto que el receptor puede detenerla cuando se llegue a la marca de agua alta. Potencialmente, esto le da el tiempo para retransmitir los paquetes perdidos, aunque esta estrategia no es muy común. En contraste, la radio en vivo siempre se difunde a la tasa exacta a la que se genera y a la que se reproduce.

Otra diferencia es que una estación de radio por lo general tiene cientos o miles de escuchas simultáneos mientras que el audio de flujo continuo es de punto a punto. Bajo estas circunstancias, la radio en Internet debería utilizar multidifusión con los protocolos RTP/RTSP. Ésta es claramente la forma más eficiente de operar.

En la actualidad, la radio en Internet no funciona así. Lo que sucede realmente es que el usuario establece una conexión TCP con la estación y la alimentación se envía a través de una conexión TCP. Por supuesto, esto crea varios problemas, como que el flujo se detenga cuando la ventana esté llena, que los paquetes perdidos expiren y se retransmitan, etcétera.

Hay tres razones por las que se utiliza la unidifusión TCP en lugar de la multidifusión RTP. La primera es que pocos ISPs soportan la multidifusión, por lo que no es una opción práctica. La segunda es que RTP es menos conocido que TCP y las estaciones de radio por lo general son pequeñas y tienen poca experiencia en computación, por lo que es más fácil que utilicen un protocolo que conocen bien y que es soportado por todos los paquetes de software. La tercera es que muchas personas escuchan la radio en Internet en su trabajo, lo que en la práctica significa detrás de un *firewall*. La mayoría de los administradores de sistemas configura su *firewall* para proteger su LAN contra visitantes no bienvenidos. Por lo general, tales administradores permiten conexiones TCP del puerto remoto 25 (SMTP para correo electrónico), paquetes UDP del puerto remoto 53 (DNS) y conexiones TCP del puerto remoto 80 (HTTP para Web). Casi todo lo demás podría bloquearse, incluyendo RTP. Por lo tanto, la única forma de obtener la señal de radio a través del *firewall* es que el sitio Web pretenda ser un servidor HTTP, por lo menos ante el *firewall*, y que utilice servidores HTTP, los cuales hablan TCP. Aunque estas medidas severas proporcionan un mínimo de seguridad, por lo general obligan a las aplicaciones multimedia a que utilicen modos de operación drásticamente menos eficientes.

Puesto que la radio en Internet es un medio nuevo, las guerras de los formatos están en su apogeo. RealAudio, Windows Media Audio y MP3 están compitiendo de manera agresiva en este mercado para ser el formato dominante para la radio en Internet. Un recién llegado es Vorbis, que técnicamente es similar a MP3 pero es código abierto y tiene las diferencias suficientes como para no utilizar las patentes en la que se basa MP3.

Una estación de radio típica de Internet tiene una página Web que lista su agenda, información sobre sus DJs y anunciadores, y muchos anuncios. También hay varios iconos que listan los formatos de audio que la estación soporta (o simplemente ESCUCHAR AHORA si sólo soporta un formato). Estos iconos o ESCUCHAR AHORA están vinculados con metaarchivos del tipo que analizamos anteriormente.

Cuando un usuario hace clic en uno de estos iconos, se envía el pequeño metaarchivo. El navegador utiliza su tipo MIME o extensión de archivo para determinar la aplicación auxiliar apropiada (es decir, el reproductor de medios) para el metaarchivo. A continuación escribe el metaarchivo en un archivo de trabajo en disco, inicia el reproductor de medios y le entrega el nombre del archivo de trabajo. El reproductor de medios lee dicho archivo, ve el URL que contiene (por lo general, con un esquema *http* en lugar de *rtsp* para solucionar el problema del *firewall* y porque algunas aplicaciones populares de multimedia funcionan de esa manera), contacta al servidor y comienza a actuar como un radio. Además, el audio sólo tiene un flujo, por lo que *http* funciona, pero para el vídeo, que por lo menos tiene dos flujos, *http* no funciona y lo que realmente se necesita es algo como *rtsp*.

Otro desarrollo interesante en el área de la radio en Internet es un arreglo en el que cualquiera, incluso un estudiante, puede establecer y operar una estación de radio. Los componentes principales se ilustran en la figura 7-63. La base de la estación es una PC ordinaria con una tarjeta de sonido y un micrófono. El software consiste en un reproductor de medios, como Winamp o Freeamp, con un *plug-in* para la captura de audio y un codec para el formato de salida seleccionado, por ejemplo, MP3 o Vorbis.

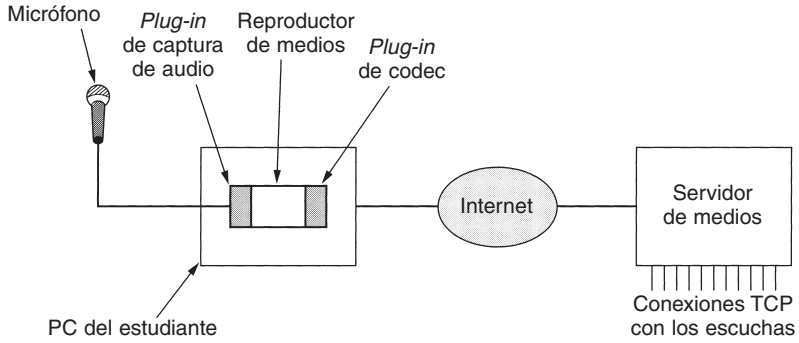


Figura 7-63. Una estación de radio estudiantil.

A continuación el flujo de audio generado por la estación se alimenta a través de Internet hacia un servidor grande, el cual lo distribuye a una gran cantidad de conexiones TCP. Por lo general, el servidor soporta muchas estaciones pequeñas. También mantiene un directorio de las estaciones que tiene y de lo que está actualmente en el aire en cada una. Los escuchas potenciales van al servidor, seleccionan una estación y obtienen una alimentación TCP. Hay paquetes de software comercial para manejar todas las piezas y paquetes de código abierto, como *icecast*. También hay servidores que están dispuestos a manejar la distribución a cambio de una cuota.

7.4.5 Voz sobre IP

En sus inicios, el sistema telefónico conmutado público se utilizaba principalmente para el tráfico de voz con un poco de tráfico de datos por aquí y por allá. Pero el tráfico de datos creció y creció, y aproximadamente en 1999, la cantidad de bits de datos movidos igualó a la de bits de voz (debido a que la voz está en PCM en las troncales, puede medirse en bits/seg). En el 2002, el volumen del tráfico de datos era mayor que el volumen del tráfico de voz y continúa creciendo de manera exponencial, mientras que el crecimiento del tráfico de voz casi era plano (con un crecimiento anual de 5%).

Como consecuencia de estas cifras, muchos operadores de redes de conmutación de paquetes de repente se interesaron en transportar voz a través de sus redes de datos. La cantidad de ancho de banda adicional requerida para voz es minúscula debido a que las redes de paquetes están dimensionadas para el tráfico de datos. Sin embargo, probablemente el recibo telefónico de la persona promedio sea más grande que su cuenta de Internet, por lo que los operadores de redes de datos

vieron la telefonía de Internet como una forma de ganar una gran cantidad de dinero adicional sin tener que colocar una nueva fibra en el piso. De esta forma nació la **telefonía de Internet** (también conocida como **voz sobre IP**).

H.323

Lo que a todos les quedó claro desde el principio fue que si cada fabricante diseñaba su propia pila de protocolos, tal vez el sistema nunca funcionaría. Para evitar este problema, un grupo de personas interesadas se unieron bajo la protección de la ITU para trabajar en estándares. En 1996 la ITU emitió la recomendación **H.323** titulada "Sistemas Telefónicos Visuales y Equipo para Redes de Área Local que Proporcionan una Calidad de Servicio No Garantizada". Sólo la industria telefónica podría pensar en tal nombre. La recomendación se revisó en 1998, y esta H.323 revisada fue la base de los primeros sistemas de telefonía de Internet ampliamente difundidos.

H.323 es más una revisión arquitectónica de la telefonía de Internet que un protocolo específico. Hace referencia a una gran cantidad de protocolos específicos para codificación de voz, establecimiento de llamadas, señalización, transporte de datos y otras áreas, en lugar de especificar estas cosas en sí. El modelo general se ilustra en la figura 7-64. En el centro se encuentra una **puerta de enlace** que conecta Internet con la red telefónica. Dicha puerta de enlace habla los protocolos H.323 en el lado de Internet y los protocolos PSTN en el lado del teléfono. Los dispositivos de comunicación se llaman **terminales.** Una LAN podría tener un **gatekeeper,** el cual controla los puntos finales bajo su jurisdicción, llamados **zona.**

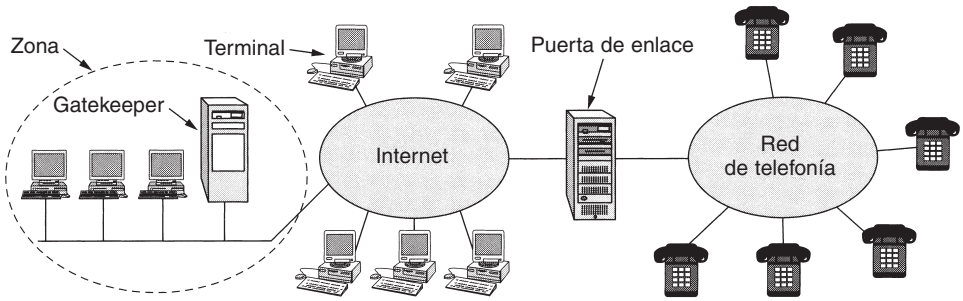


Figura 7-64. El modelo arquitectónico H.323 de la telefonía de Internet.

Una red telefónica necesita una cantidad de protocolos. Para empezar, hay un protocolo para codificar y decodificar voz. El sistema PCM que estudiamos en el capítulo 2 está definido en la recomendación **G.711** de la ITU. Codifica un solo canal de voz muestreando 8000 veces por segundo con una muestra de 8 bits para proporcionar voz descomprimida a 64 kbps. Todos los sistemas H.323 deben soportar G.711. Sin embargo, también se permiten otros protocolos de compresión de voz (aunque no son requeridos). Utilizan diferentes algoritmos de compresión y realizan diferentes intercambios entre calidad y ancho de banda. Por ejemplo, **G.723.1** toma un

bloque de 240 muestras (30 mseg de voz) y utiliza la codificación predictiva para reducirlo ya sea a 24 o a 20 bytes. Este algoritmo proporciona una tasa de salida ya sea de 6.4 o 5.3 kbps (factores de compresión de 10 y 12), respectivamente, con poca pérdida en la calidad percibida. También se permiten otros codecs.

Puesto que están permitidos múltiples algoritmos de compresión, se necesita un protocolo para permitir que las terminales negocien cuál van a utilizar. Este protocolo se llama **H.245.** También negocia otros aspectos de la conexión, como la tasa de bits. RTCP es necesario para el control de los canales RTP. También se necesita un protocolo para establecer y liberar conexiones, proporcionar tonos de marcado, emitir sonidos de marcado y el resto de la telefonía estándar. Aquí se utiliza **Q.931** de la ITU. Las terminales necesitan un protocolo para hablar con el *gatekeeper* (si está presente). Para este propósito se utiliza **H.225.** El canal PC a *gatekeeper* manejado por este protocolo se conoce como canal **RAS** (**Registro/Admisión/Estado**). Este canal permite terminales para unirse y dejar la zona, solicitar y regresar ancho de banda, y proporcionar actualizaciones de estado, entre otras cosas. Por último, se necesita un protocolo para la transmisión real de datos. RTP se utiliza para este propósito. Es manejado por RTCP, como es usual. La posición de todos estos protocolos se muestra en la figura 7-65.

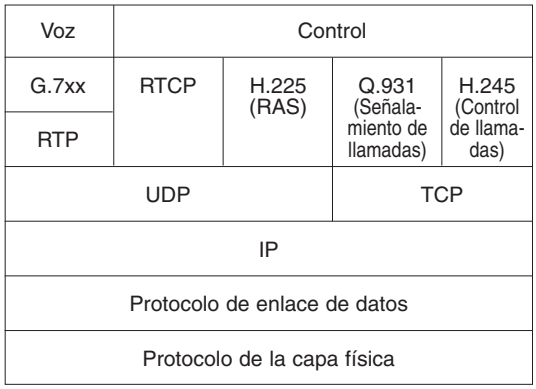


Figura 7-65. La pila de protocolos H.323.

Para ver cómo se ajustan estos protocolos, considere el caso de una terminal de PC en una LAN (con un *gatekeeper*) que llama a un teléfono remoto. La PC primero tiene que descubrir al *gatekeeper*, por lo que difunde un paquete UDP de descubrimiento de *gatekeeper* al puerto 1718. Cuando el *gatekeeper* responde, la PC se aprende la dirección IP de éste. Ahora la PC se registra con el *gatekeeper* enviándole un mensaje RAS en un paquete UDP. Después de que es aceptada, la PC envía al *gatekeeper* un mensaje de admisión RAS solicitando ancho de banda. Sólo después de que se ha proporcionado el ancho de banda, se puede establecer la llamada. La idea de solicitar ancho de banda con anticipación es permitir que el *gatekeeper* limite el número de llamadas para evitar sobrescribir la línea saliente para ayudar a proporcionar la calidad de servicio necesaria.

La PC ahora establece una conexión TCP con el *gatekeeper* para comenzar el establecimiento de la llamada. Este establecimiento utiliza los protocolos de red telefónicos existentes, que están orientados a la conexión, por lo que se necesita TCP. En contraste, el sistema telefónico no tiene nada parecido a RAS para permitir que los teléfonos anuncien su presencia, por lo que los diseñadores de H.323 eran libres de utilizar UDP o TCP para RAS, y eligieron el UDP que genera menor información adicional.

Ahora que tiene asignado ancho de banda, la PC puede enviar un mensaje *SETUP* de Q.931 a través de la conexión TCP. Este mensaje especifica el número del teléfono al que se está llamando (o la dirección IP y el puerto, si se está llamando a una computadora). El *gatekeeper* responde con un mensaje *CALL PROCEEDING* de Q.931 para confirmar la recepción de la solicitud. Enseguida el *gatekeeper* reenvía un mensaje *SETUP* a la puerta de enlace.

A continuación, la puerta de enlace, que es mitad computadora y mitad conmutador de teléfono, realiza una llamada telefónica ordinaria al teléfono deseado (ordinario). La oficina central a la que está anexado el teléfono hace que suene el teléfono al que se hace la llamada y también regresa un mensaje *ALERT* de Q.931 para indicar a la PC invocadora que ya se ha emitido el sonido. Cuando la persona en el otro extremo levanta el teléfono, la oficina central regresa un mensaje *CONNECT* de Q.931 para indicar a la PC que tiene una conexión.

Una vez que se ha establecido la conexión, el *gatekeeper* ya no está en el ciclo, aunque la puerta de enlace sí lo está, por supuesto. Los paquetes subsiguientes ignoran al *gatekeeper* y van directo a la dirección IP de la puerta de enlace. En este punto, simplemente tenemos un tubo que se ejecuta entre los dos lados. Ésta es tan sólo una conexión de capa física para mover bits, nada más. Ninguno de los lados sabe nada del otro.

A continuación se utiliza el protocolo H.245 para negociar los parámetros de la llamada. Utiliza el canal de control H.245, que siempre está abierto. Cada lado inicia anunciando sus capacidades, por ejemplo, si puede manejar vídeo (H.323 puede manejar vídeo) o llamadas de conferencia, qué codecs soporta, etcétera. Una vez que cada lado sabe lo que el otro puede manejar, se establecen dos canales de datos unidireccionales y a cada uno se le asigna un codec y otros parámetros. Puesto que cada lado puede tener equipo diferente, es posible que los codecs en los canales hacia adelante e inverso sean diferentes. Una vez que se terminan todas las negociaciones, puede comenzar el flujo de datos utilizando RTP. Tal flujo se maneja mediante RTCP, que juega un papel en el control de congestionamiento. Si el vídeo está presente, RTCP maneja la sincronización de audio/vídeo. En la figura 7-66 se muestran los diversos canales. Cuando cualquiera de las dos partes cuelga, se utiliza el canal de señalización de llamada de Q.931 para terminar la conexión.

Cuando se termina la llamada, la PC invocadora contacta al *gatekeeper* nuevamente con un mensaje RAS para liberar el ancho de banda que se ha asignado. De manera alterna, puede realizar otra llamada.

No hemos mencionado nada sobre la calidad del servicio, aunque ésta es esencial para que la voz sobre IP sea un éxito. La razón es que la QoS está más allá del alcance de H.323. Si la red subyacente es capaz de producir una conexión estable, libre de fluctuación de la PC invocadora (por ejemplo, utilizando las técnicas que analizamos en el capítulo 5) a la puerta de enlace, entonces la QoS de la llamada será buena; de lo contrario, no lo será. La parte del teléfono utiliza PCM y siempre está libre de fluctuación.

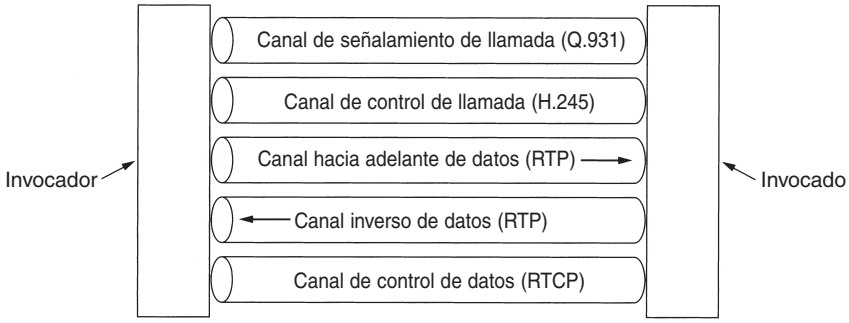


Figura 7-66. Canales lógicos entre el invocador y el invocado durante una llamada.

SIP—Protocolo de Inicio de Sesión

H.323 fue diseñado por la ITU. Muchas personas de la comunidad de Internet lo vieron como un producto típico telco: grande, complejo e inflexible. En consecuencia, la IETF estableció un comité para diseñar una forma más simple y modular para transmitir voz sobre IP. El resultado principal hasta la fecha es el **SIP (Protocolo de Inicio de Sesión),** que se describe en el RFC 3261. Este protocolo describe cómo establecer llamadas telefónicas a Internet, videoconferencias y otras conexiones multimedia. A diferencia de H.323, que es un conjunto completo de protocolos, SIP está compuesto de un solo módulo, pero se ha diseñado para que interactúe bien con las aplicaciones de Internet existentes. Por ejemplo, define los números telefónicos como URLs, a fin de que las páginas Web puedan contenerlos, lo que permite hacer clic en un vínculo para iniciar una llamada telefónica (de la misma forma en que un esquema *mailto* permite hacer clic en un vínculo que despliega un programa para enviar un mensaje de correo electrónico).

SIP puede establecer sesiones de dos partes (llamadas de teléfono ordinarias), de múltiples partes (en donde todos pueden oír y hablar) y de multidifusión (un emisor, muchos receptores). Las sesiones pueden contener audio, vídeo o datos; las de multidifusión son útiles para juegos de múltiples jugadores, por ejemplo. SIP sólo maneja establecimiento, manejo y terminación de sesiones. Para el transporte de datos, se utilizan otros protocolos, como RTP/RTCP. SIP es un protocolo de capa de aplicación y puede ejecutarse sobre UDP o TCP.

SIP soporta una variedad de servicios, como localizar al invocado (quien tal vez no esté en su máquina doméstica) y determinar las capacidades de éste, así como manejar los mecanismos del establecimiento y la terminación de la llamada. En el caso más simple, SIP establece una sesión de la computadora del invocador a la del invocado, por lo que primero analizaremos ese caso.

Los números telefónicos de SIP se representan como URLs que utilizan el esquema *sip*, por ejemplo, *sip:ilse@cs.university.edu* para un usuario de nombre Ilse en el *host* especificado por el nombre DNS *cs.university.edu*. Los URLs de SIP también pueden contener direcciones IPv4, IPv6 o números telefónicos reales.

El protocolo SIP se basa en texto y está modelado en HTTP. Una parte envía un mensaje en texto ASCII que consiste en un nombre de método en la primera línea, seguido por líneas adicionales que contienen encabezados para pasar los parámetros. Muchos de estos encabezados se toman de MIME para permitir que SIP interactúe con las aplicaciones de Internet existentes. En la figura 7-67 se listan los seis métodos definidos por la especificación central.

Método	Descripción
INVITE	Solicita el inicio de una sesión
ACK	Confirma que se ha iniciado una sesión
BYE	Solicita la terminación de una sesión
OPTIONS	Consulta a un *host* sobre sus capacidades
CANCEL	Cancela una solicitud pendiente
REGISTER	Informa a un servidor de redireccionamiento sobre la ubicación actual del usuario

Figura 7-67. Los métodos SIP definidos en la especificación central.

Para establecer una sesión, el invocador crea una conexión TCP con el invocado y envía un mensaje *INVITE* a través de ella o lo envía en un paquete UDP. En ambos casos, los encabezados de la segunda línea y de las subsiguientes describen la estructura del cuerpo del mensaje, el cual contiene las capacidades, los tipos de medios y los formatos del invocador. Si el invocado acepta la llamada, responde con un código de respuesta tipo HTTP (un número de tres dígitos que utiliza los grupos de la figura 7-42, 200 para aceptación). El invocado también puede proporcionar información sobre sus capacidades, tipos de medios y formatos, después de la línea de código de respuesta.

La conexión se realiza utilizando un acuerdo de tres vías, de modo que el invocador responde con un mensaje *ACK* para terminar el protocolo y confirmar la recepción del mensaje 200.

Cualquiera de las dos partes puede solicitar la terminación de una sesión enviando un mensaje que contiene el método *BYE*. Cuando el otro lado confirma su recepción, se termina la sesión.

El método *OPTIONS* se utiliza para consultar a una máquina sobre sus propias capacidades. Por lo general, se utiliza antes de que se inicie una sesión a fin de averiguar si esa máquina tiene la capacidad de transmitir voz sobre IP o el tipo de sesión que se ha contemplado.

El método *REGISTER* se relaciona con la capacidad de SIP de rastrear y conectarse con un usuario que esté lejos de casa. Este mensaje se envía a un servidor de ubicación SIP que mantiene la pista de quién está en qué lugar. Ese servidor se puede consultar posteriormente para encontrar la ubicación actual del usuario. En la figura 7-68 se ilustra la operación de redirección. Aquí el invocador envía el mensaje *INVITE* a un servidor *proxy* para ocultar la posible redirección. A continuación el *proxy* investiga en dónde está el usuario y envía el mensaje *INVITE* ahí. Después actúa como un regulador para los mensajes subsecuentes en el acuerdo de tres vías. Los mensajes *LOOKUP* y *REPLY* no son parte de SIP; se puede utilizar cualquier protocolo conveniente, dependiendo del tipo de servidor de ubicación que se esté utilizando.

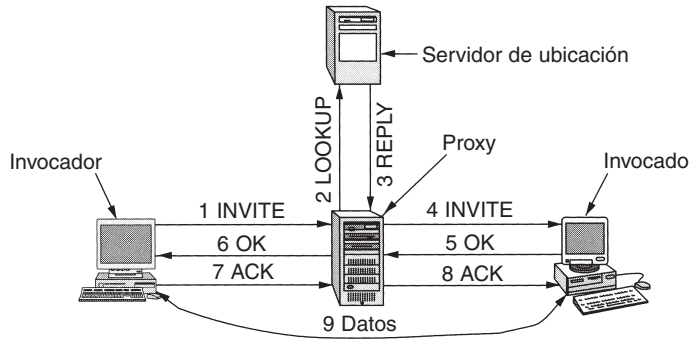


Figura 7-68. Uso de servidores proxy y de redirección con SIP.

SIP tiene una variedad de otras características que no describiremos aquí, entre ellas la espera de llamadas, bloque de llamada, codificación y autenticación. También tiene la capacidad de colocar llamadas de una computadora en un teléfono ordinario, si está disponible una puerta de enlace entre Internet y el sistema telefónico.

Comparación entre H.323 y SIP

H.323 y SIP tienen muchas similitudes pero también algunas diferencias. Ambos permiten llamadas de dos partes y múltiples partes utilizando las computadoras y los teléfonos como puntos finales. Ambos soportan negociación de parámetros, codificación y los protocolos RTP/RTCP. En la figura 7-69 se muestra un resumen de las similitudes y diferencias.

Aunque los conjuntos de características son similares, los dos protocolos difieren ampliamente en la filosofía. H.323 es un estándar típico de peso completo de la industria de la telefonía, que especifica toda la pila de protocolos y que define de manera precisa lo que está permitido y lo que está prohibido. Este enfoque produce protocolos bien definidos en cada capa, lo que facilita la tarea de interoperabilidad. El precio pagado es un estándar grande, complejo y rígido que es difícil de adaptar a aplicaciones futuras.

En contraste, SIP es un protocolo de Internet típico que funciona intercambiando líneas cortas de texto ASCII. Es un módulo de carga ligera que interactúa bien con otros protocolos de Internet pero no tan bien con los de señalización de sistemas telefónicos. Debido a que el modelo IETF de voz sobre IP es altamente modular, es flexible y se puede adaptar con facilidad a las nuevas aplicaciones. La desventaja son problemas potenciales de interoperabilidad, aunque éstos se solucionan realizando reuniones frecuentes en las que los diferentes implementadores se reúnen para probar sus sistemas.

La voz sobre IP es un tema prometedor. En consecuencia, hay varios libros sobre él. Algunos ejemplos son (Collins, 2001; Davidson y Peters, 2000; Kumar y cols., 2001, y Wright, 2001). La edición de mayo/junio de 2002 de *Internet Computing* tiene varios artículos sobre este tema.

Elemento	H.323	SIP
Diseñado por	ITU	IETF
Compatibilidad con PSTN	Sí	Ampliamente
Compatibilidad con Internet	No	Sí
Arquitectura	Monolítica	Modular
Integridad	Pila de protocolos completa	SIP sólo maneja el establecimiento
Negociación de parámetros	Sí	Sí
Señalamiento de llamadas	Q.931 sobre TCP	SIP sobre TCP o UDP
Formato de mensajes	Binario	ASCII
Transporte de medios	RTP/RTCP	RTP/RTCP
Llamadas de múltiples partes	Sí	Sí
Conferencias multimedia	Sí	No
Direccionamiento	*Host* o número telefónico	URL
Terminación de llamadas	Explícita o liberación de TCP	Explícita o terminación de temporizador
Mensajes instantáneos	No	Sí
Encriptación	Sí	Sí
Tamaño de los estándares	1400 páginas	250 páginas
Implementación	Grande y compleja	Moderada
Estado	Distribuido ampliamente	Prometedor

Figura 7-69. Comparación entre H.323 y SIP.

7.4.6 Introducción al vídeo

Ya analizamos el oído hasta ahora; es tiempo de que analicemos el ojo (no, a esta sección no le sigue una sobre la nariz). El ojo humano tiene la propiedad de que, cuando una imagen incide en la retina, se retiene durante algunos milisegundos antes de desaparecer. Si una secuencia de imágenes incide a 50 o más imágenes/seg, el ojo no nota que está viendo imágenes discretas. Todos los sistemas de vídeo (es decir, televisión) aprovechan este principio para producir imágenes en movimiento.

Sistemas analógicos

Para entender los sistemas de vídeo, es mejor comenzar por la anticuada y sencilla televisión en blanco y negro. Para representar la imagen bidimensional que está frente a ella como un voltaje unidimensional en función del tiempo, la cámara barre rápidamente un haz de electrones a lo ancho de la imagen y lentamente hacia abajo, registrando la intensidad de la luz a su paso. Al final del barrido, llamado **trama,** el haz hace un retrazado. Esta intensidad como función de tiempo se difunde, y los receptores repiten el proceso de barrido para reconstruir la imagen. En la figura 7-70 se muestra el patrón de barrido que usan tanto la cámara como el receptor. (Como nota,

las cámaras CCD integran en lugar de barrer, pero algunas cámaras y todos los monitores hacen un barrido.)

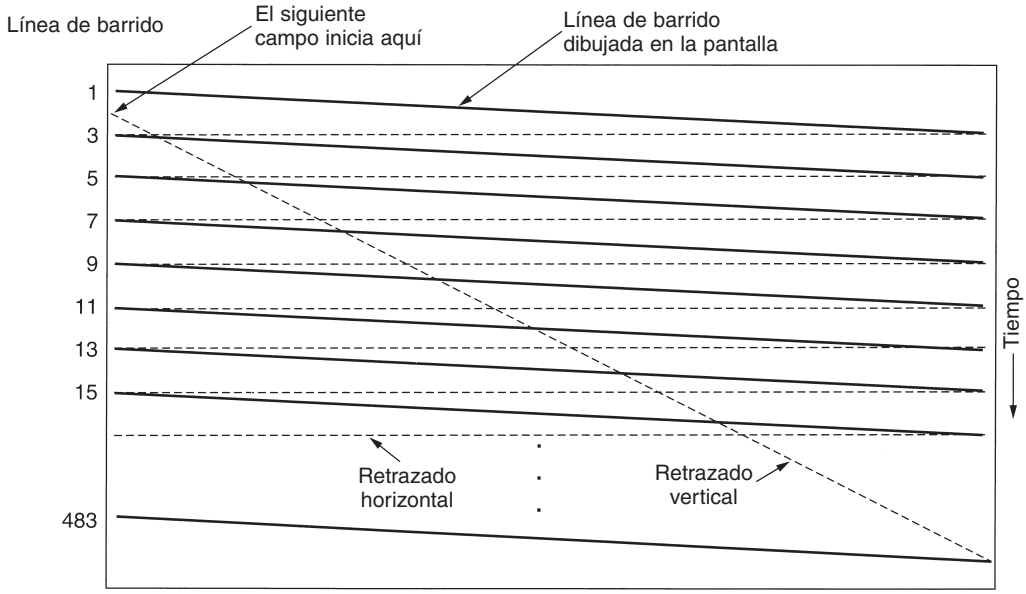


Figura 7-70. Patrón de barrido usado para el vídeo y la televisión NTSC.

Los parámetros de barrido exactos varían de país en país. El sistema usado en Norte y Sudamérica y en Japón tiene 525 líneas de barrido, una relación de aspecto horizontal a vertical de 4:3 y 30 tramas/seg. El sistema europeo tiene 625 líneas de barrido, la misma relación 4:3 y 25 tramas/seg. En ambos sistemas, no se presentan unas cuantas líneas de arriba y abajo (para aproximar una imagen rectangular a los CRTs originales, que eran redondos). Sólo se presentan 483 de las 525 líneas NTSC (y 576 de las 625 líneas de barrido PAL/SECAM). El haz se apaga durante el retrazado vertical, por lo que muchas estaciones (especialmente en Europa) usan este intervalo para difundir TeleTexto (páginas de texto que contienen noticias, informes meteorológicos, deportes, precios de acciones, etcétera).

Aunque 25 tramas/seg es suficiente para capturar una imagen continua, con esa tasa de tramas mucha gente, especialmente las personas mayores, percibe un parpadeo en la imagen (porque las imágenes viejas se desvanecen de la retina antes de la aparición de las nuevas). En lugar de aumentar la tasa de tramas, lo que requeriría usar más del escaso ancho de banda, se emplea un enfoque diferente. En lugar de presentar las líneas de barrido en orden, primero se despliegan las líneas de barrido nones, y luego las líneas de barrido pares. Cada una de estas medias tramas se llama **campo.** Hay experimentos que muestran que, aunque la gente nota el parpadeo a 25 tramas/seg, no lo nota a 50 campos/seg. Esta técnica se llama **entrelazado.** Se dice que la televisión (o vídeo) no entrelazada es **progresiva.** Observe que las películas se ejecutan a 24 fps, pero cada trama es completamente visible durante 1/24 seg.

El vídeo de color usa el mismo patrón de barrido que el monocromático (blanco y negro), excepto que, en lugar de desplegar la imagen mediante un solo haz en movimiento, se usan tres haces que se mueven al unísono. Se usa un haz para cada uno de los colores primarios aditivos: rojo, verde y azul (RGB). Esta técnica funciona porque puede construirse cualquier color a partir de la superposición lineal de rojo, verde y azul con las intensidades apropiadas. Sin embargo, para la transmisión por un solo canal, las tres señales de color deben combinarse en una sola señal **compuesta.**

Cuando se inventó la televisión a color, eran técnicamente posibles varios métodos para desplegar colores, así que varios países tomaron decisiones diferentes, lo que condujo a sistemas que aún son incompatibles. (Es importante mencionar que estas decisiones no tienen nada que ver con VHS contra Betamax contra P2000, que son métodos de grabación.) En todos los países, un requisito político fue que todos los programas que se transmitieran a color tenían que poder recibirse en los televisores existentes de blanco y negro. En consecuencia, no era aceptable el esquema más sencillo, codificar por separado las señales RGB. Sin embargo, RGB tampoco es el esquema más eficiente.

El primer sistema de color fue estandarizado en Estados Unidos por el **Comité Nacional de Estándares de Televisión**, que presentó sus siglas al estándar: **NTSC.** La televisión a color se introdujo en Europa varios años después, y para entonces la tecnología había mejorado de manera significativa, conduciendo a sistemas con mejor inmunidad contra el ruido y mejores colores. Éstos se llaman **SECAM (color secuencial con memoria),** que se usa en Francia y Europa Oriental, y **PAL (línea de fases alternas),** usado en el resto de Europa. La diferencia en la calidad del color entre NTSC y PAL/SECAM ha producido una broma de la industria que sugiere que NTSC significa en realidad Nunca Tendrás Semejanza en los Colores.

Para que las transmisiones puedan verse en receptores de blanco y negro, los tres sistemas combinan linealmente las señales RGB en una señal de **luminancia** (brillo) y dos de **crominancia** (color), aunque usan diferentes coeficientes para construir estas señales a partir de las señales RGB. Resulta interesante que el ojo es mucho más sensible a la señal de luminancia que a las de crominancia, por lo que estas últimas no necesitan transmitirse con tanta precisión. En consecuencia, la señal de luminancia puede difundirse a la misma frecuencia que la vieja señal de blanco y negro, y puede recibirse en los televisores de blanco y negro. Las dos señales de crominancia se difunden en bandas angostas a frecuencias mayores. Algunos aparatos de televisión tienen controles etiquetados brillo, matiz y saturación (o brillo, tinte y color) para controlar estas tres señales por separado. Es necesario el entendimiento de la luminancia y la crominancia para comprender el funcionamiento de la compresión de vídeo.

En los últimos años ha habido un interés considerable en la **HDTV (televisión de alta definición),** que produce imágenes más nítidas al duplicar (aproximadamente) la cantidad de líneas de barrido. Estados Unidos, Europa y Japón han desarrollado sistemas HDTV, todos diferentes y todos mutuamente incompatibles. ¿Usted esperaba otra cosa? Los principios básicos de la HDTV en términos de barrido, luminancia, crominancia, etcétera, son semejantes a los sistemas actuales. Sin embargo, los tres formatos tienen una relación de aspecto común de 16:9 en lugar 4:3 para lograr una correspondencia mejor con el formato usado para el cine (que se graba en película de 35 mm, y tiene una relación de aspecto de 3:2).

Sistemas digitales

La representación más sencilla del vídeo digital es una secuencia de tramas, cada una de las cuales consiste en una malla rectangular de elementos de imagen, o **píxeles.** Cada píxel puede ser un solo bit, para representar blanco o negro. La calidad de tal sistema es parecida a la que se obtiene al enviar una fotografía a color por fax: espantosa. (Inténtelo si puede; de lo contrario, fotocopie una fotografía a color en una máquina copiadora que no maneje tonos.)

El siguiente paso es usar ocho bits por píxel para representar 256 niveles de gris. Este esquema da vídeo en blanco y negro de alta calidad. Para vídeo a color, los sistemas buenos usan 8 bits por cada uno de los colores RGB, aunque casi todos los sistemas los mezclan en vídeo compuesto para su transmisión. Aunque el uso de 24 bits por píxel limita la cantidad de colores a unos 16 millones, el ojo humano no puede diferenciar tantos colores. Las imágenes digitales de color se producen usando tres haces de barrido, uno por color. La geometría es la misma que en el sistema analógico de la figura 7-70, excepto que las líneas continuas de barrido ahora se reemplazan por elegantes filas de píxeles discretos.

Para producir una imagen uniforme, el vídeo digital, al igual que el vídeo analógico, debe presentar cuando menos 25 tramas/seg. Sin embargo, dado que los monitores de computadora de alta calidad con frecuencia barren la pantalla a partir de las imágenes almacenadas en memoria a razón de 75 veces por segundo o más, no se requiere entrelazado y, en consecuencia, normalmente no se usa. Basta con repintar (es decir, redibujar) la misma trama tres veces consecutivas para eliminar el parpadeo.

En otras palabras, la uniformidad de movimiento es determinada por la cantidad de imágenes *diferentes* por segundo, mientras que el parpadeo es determinado por la cantidad de veces por segundo que se pinta la trama. Estos dos parámetros son diferentes. Una imagen fija pintada a 20 tramas/seg no mostrará un movimiento inestable, pero parpadeará porque una trama se desvanecerá de la retina antes de que aparezca la siguiente. Una película con 20 tramas diferentes por segundo, cada una pintada cuatro veces seguidas, no parpadeará, pero el movimiento parecerá no uniforme.

El significado de estos dos parámetros se aclara cuando consideramos el ancho de banda necesario para transmitir vídeo digital a través de una red. Todos los monitores de computadora actuales usan la relación de aspecto 4:3 para poder usar tubos de rayos catódicos económicos de producción en masa diseñados para el mercado de televisión de consumidor. Las configuraciones comunes son 1024×768, 1280×960 y 1600×1200. Incluso la más pequeña de éstas con 24 bits por píxel y 25 tramas/seg necesita alimentarse a 472 Mbps. Una portadora OC-12 de SONET tendría que manejar esta situación, y la ejecución de este tipo de portadora en la casa de todas las personas aún no es posible. La duplicación de esta tasa para evitar el parpadeo es aún menos atractiva. Una mejor solución es transmitir 25 tramas/seg y hacer que la computadora almacene cada una y la pinte dos veces. La televisión difundida no usa esta estrategia porque las televisiones no tienen memoria y, aunque la tuvieran, para almacenarse en RAM, las señales analógicas primero se tendrían que convertir a un formato digital, lo que requeriría hardware extra. Como consecuencia, se necesita el entrelazado para la televisión difundida, pero no para el vídeo digital.

7.4.7 Compresión de vídeo

A estas alturas debe ser obvio que la transmisión de vídeo sin comprimir es impensable. La única esperanza es que sea posible la compresión masiva. Por fortuna, durante las últimas décadas un gran número de investigaciones ha conducido a muchas técnicas y algoritmos de compresión que hacen factible la transmisión de vídeo. En esta sección estudiaremos cómo se consigue la compresión de vídeo.

Todos los sistemas de compresión requieren dos algoritmos: uno para la compresión de los datos en el origen y otro para la descompresión en el destino. En la literatura, estos algoritmos se conocen como algoritmos de **codificación** y **decodificación,** respectivamente. También usaremos esta terminología aquí.

Estos algoritmos tienen ciertas asimetrías y es importante comprenderlas. Primero, en muchas aplicaciones un documento multimedia, digamos una película, sólo se codificará una vez (cuando se almacene en el servidor multimedia), pero se decodificará miles de veces (cuando los clientes lo vean). Esta asimetría significa que es aceptable que el algoritmo de codificación sea lento y requiera hardware costoso, siempre y cuando el algoritmo de decodificación sea rápido y no requiera hardware costoso. A fin de cuentas, el operador de un servidor multimedia podría esta dispuesto a rentar una supercomputadora paralela durante algunas semanas para codificar su biblioteca de vídeo completa, pero requerir que los consumidores renten una supercomputadora durante dos horas para ver un vídeo seguramente no tendrá mucho éxito. Muchos sistemas de compresión prácticos llegan a extremos considerables para lograr que la decodificación sea rápida y sencilla, aun al precio de hacer lenta y complicada la codificación.

Por otra parte, para la multimedia en tiempo real, como las videoconferencias, la codificación lenta es inaceptable. La codificación debe ocurrir al momento, en tiempo real. En consecuencia, la multimedia en tiempo real usa algoritmos o parámetros diferentes que el almacenamiento de vídeos en disco, lo que con frecuencia resulta en una compresión significativamente menor.

Una segunda asimetría es que no es necesaria la capacidad de invertir el proceso de codificación/decodificación. Es decir, al comprimir, transmitir y descomprimir un archivo de datos, el usuario espera recibir el original, correcto hasta el último bit. En multimedia este requisito no existe. Por lo general, es aceptable que la señal de vídeo después de codificar y decodificar sea ligeramente diferente de la original. Cuando la salida decodificada no es exactamente igual a la entrada original, se dice que el sistema es **con pérdida** (*lossy*). Si la entrada y la salida son idénticas, el sistema es **sin pérdida** (*lossless*). Los sistemas con pérdida son importantes porque aceptar una pequeña pérdida de información puede ofrecer ventajas enormes en términos de la posible relación de compresión.

Estándar JPEG

Un vídeo es sólo una secuencia de imágenes (más sonido). Si pudiéramos encontrar un buen algoritmo para codificar una sola imagen, tal algoritmo podría aplicarse a cada imagen en sucesión para alcanzar la compresión de vídeo. Existen algoritmos buenos de compresión de imágenes fijas, por lo que comenzaremos ahí nuestro estudio de la compresión de vídeo. El estándar **JPEG**

(**Grupo Unido de Expertos en Fotografía**) para la compresión de imágenes fijas de tono continuo (por ejemplo, fotografías) fue desarrollado por expertos en fotografía que trabajaban bajo la tutela de la ITU, la ISO y el IEC, otro grupo de estándares. Es importante para la multimedia porque, a primera vista, el estándar multimedia para imágenes en movimiento, MPEG, es simplemente la codificación JPEG por separado de cada trama, más otras características extra para la compresión entre tramas y la detección de movimiento. El JPEG se define en el Estándar Internacional 10918.

JPEG tiene cuatro modos y muchas opciones; se parece más a una lista de compras que a un algoritmo. No obstante, para nuestros fines sólo es relevante el modo secuencial con pérdida, y éste se ilustra en la figura 7-71. Además, nos concentraremos en la manera en que el JPEG se usa normalmente para codificar imágenes de vídeo RGB de 24 bits y dejaremos fuera algunos de los detalles menores por cuestiones de sencillez.

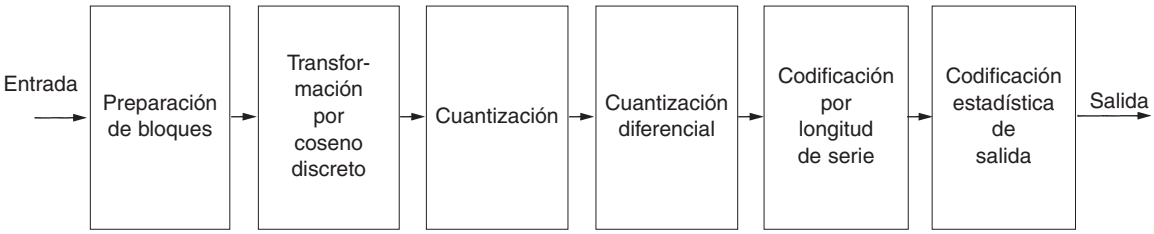


Figura 7-71. Operación de JPEG en el modo secuencial con pérdida.

El paso 1 de la codificación de una imagen con JPEG es la preparación del bloque. Para ser específicos, supongamos que la entrada JPEG es una imagen RGB de 640 × 480 con 24 bits/píxel, como se muestra en la figura 7-72(a). Puesto que el uso de luminancia y crominancia da una mejor compresión, primero calcularemos la luminancia, Y, y las dos crominancias, I y Q (para NTSC), de acuerdo con las siguientes fórmulas:

$$Y = 0.30R + 0.59G + 0.11B$$
$$I = 0.60R - 0.28G - 0.32B$$
$$Q = 0.21R - 0.52G + 0.31B$$

En PAL, las crominancias se llaman U y V, y los coeficientes son diferentes, pero la idea es la misma. SECAM es diferente tanto de NTSC como de PAL.

Se construyen matrices separadas para Y, I y Q, cada una con elementos en el intervalo de 0 a 255. A continuación se promedian tramas de cuatro píxeles en las matrices I y Q para reducirlos a 320 × 240. Esta reducción tiene pérdidas, pero el ojo apenas lo nota, ya que responde a la luminancia más que a la crominancia; no obstante, comprime los datos en un factor de dos. Ahora se resta 128 a cada elemento de las tres matrices para poner el 0 a la mitad de la gama. Por último,

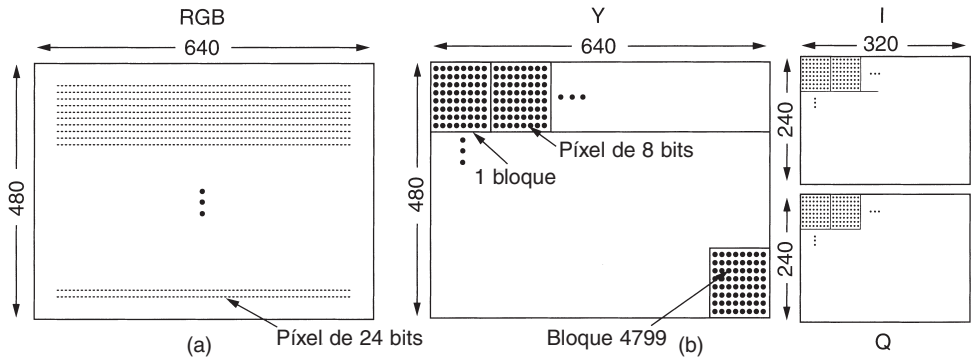


Figura 7-72. (a) Datos de entrada RGB. (b) Tras la preparación de bloques.

cada matriz se divide en bloques de 8 × 8. La matriz *Y* tiene 4800 bloques; las otras dos tienen 1200 bloques cada una, como se muestra en la figura 7-72(b).

El paso 2 de JPEG es aplicar de manera individual una **DCT** (**transformación por coseno discreto**) a cada uno de los 7200 bloques. La salida de cada DCT es una matriz de 8 × 8 de coeficientes DCT. El elemento DCT (0, 0) es el valor promedio del bloque. Los otros elementos indican la cantidad de potencia espectral que hay en cada frecuencia espacial. En teoría, una DCT no tiene pérdidas pero, en la práctica, el uso de números de punto flotante y funciones trascendentales siempre introducen algún error de redondeo que resulta en una pequeña pérdida de información. Estos elementos normalmente decaen con rapidez al alejarse del origen (0, 0), como se sugiere en la figura 7-73.

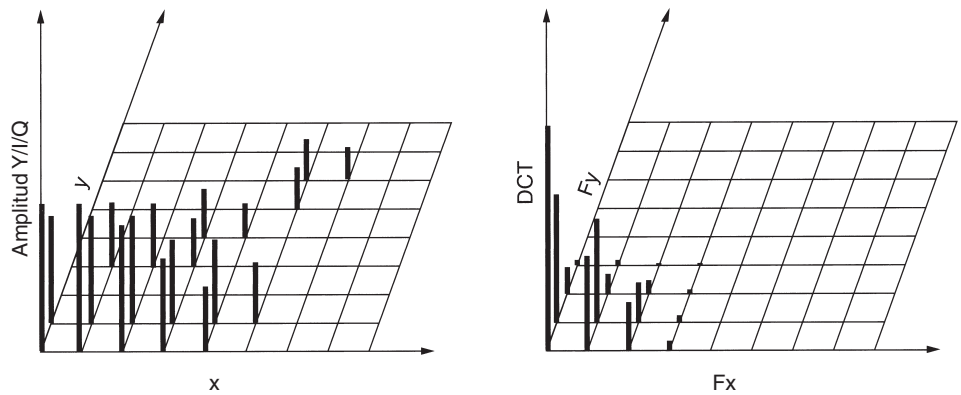


Figura 7-73. (a) Un bloque de la matriz *Y*. (b) Coeficientes DCT.

Una vez que la DCT está completa, el JPEG sigue con el paso 3, llamado **cuantización,** en el que se eliminan los coeficientes DCT menos importantes. Esta transformación (con pérdida) se hace dividiendo cada uno de los coeficientes de la matriz DCT de 8 × 8 entre un peso tomado de

una tabla. Si todos los pesos son 1, la transformación no hace nada. Sin embargo, si los pesos aumentan de manera considerable desde el origen, las frecuencias espaciales más altas se descartarán con rapidez.

En la figura 7-74 se da un ejemplo de este paso. Aquí vemos la matriz DCT inicial, la tabla de cuantización y el resultado obtenido al dividir cada elemento DCT entre el elemento correspondiente de la tabla de cuantización. Los valores de ésta no son parte del estándar JPEG. Cada aplicación debe proporcionar sus propios valores, lo que le permite controlar el equilibrio pérdida-compresión.

Coeficientes DCT

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabla de cuantización

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Coeficientes cuantizados

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 7-74. Cálculo de los coeficientes DCT cuantizados.

El paso 4 reduce el valor (0, 0) de cada bloque (el de la esquina superior izquierda) reemplazándolo por el valor de su diferencia respecto al elemento correspondiente del bloque previo. Puesto que estos elementos son las medias de sus respectivos bloques, deben cambiar lentamente, por lo que al tomarse sus valores diferenciales se debe reducir la mayoría de ellos a valores pequeños. No se calculan diferenciales de los otros valores. Los valores (0, 0) se conocen como componentes de CD; los otros valores son los componentes de CA.

El paso 5 hace lineales los 64 elementos y aplica codificación por longitud de serie a la lista. Barrer el bloque de izquierda a derecha y luego de arriba abajo no concentra los ceros, por lo que se usa un patrón de barrido de zigzag, como se muestra en la figura 7-75. En este ejemplo, el patrón de zigzag produce 38 ceros consecutivos al final de la matriz. Esta cadena puede reducirse a una sola cuenta diciendo que hay 38 ceros, una técnica conocida como **codificación por longitud de serie.**

Ahora tenemos una lista de números que representan la imagen (en espacio de transformación). El paso 6 aplica codificación de Huffman a los números para su almacenamiento o transmisión, asignando códigos más cortos de números comunes que de no comunes.

JPEG puede parecer complicado, pero eso es porque *es* complicado. Aun así se usa ampliamente, debido a que con frecuencia produce una compresión de 20:1 o mejor. La decodificación de una imagen JPEG requiere la ejecución hacia atrás del algoritmo. JEPG es más o menos simétrico: la decodificación tarda tanto como la codificación. Esta propiedad no se aplica a todos los algoritmos de compresión, como veremos a continuación.

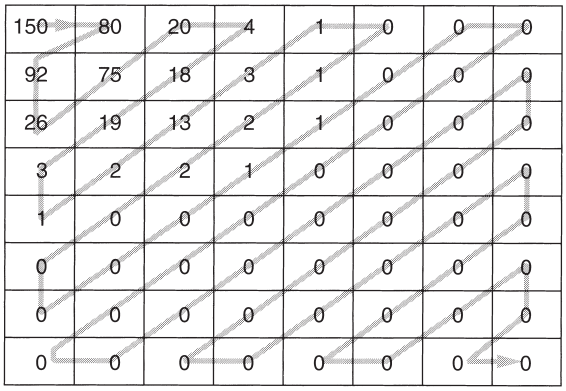


Figura 7-75. Orden en que se transmiten los valores cuantizados.

Estándar MPEG

Por último, llegamos al corazón del asunto: los estándares **MPEG (Grupo de Expertos en Imágenes en Movimiento).** Éstos son los algoritmos principales usados para comprimir vídeo y han sido estándares internacionales desde 1993. Puesto que las películas contienen imágenes y sonido, MPEG puede comprimir tanto audio como vídeo. Ya examinamos la compresión de audio y la de imágenes, por lo que ahora nos enfocaremos principalmente en la compresión de vídeo.

El primer estándar terminado fue el MPEG-1 (Estándar Internacional 11172); su meta fue producir salida con calidad de videograbadora (352×240 para NTSC) usando una tasa de bits de 1.2 Mbps. Una imagen de 352×240 con 24 bits/píxeles y 25 tramas/seg requiere 50.7 Mbps, por lo que reducirla a 1.2 Mbps no es nada fácil. Se necesita un factor de compresión 40. MPEG-1 puede transmitirse sobre líneas de transmisión de par trenzado a distancias modestas. MPEG-1 también se usa para almacenar películas en CD-ROM.

El siguiente estándar de la familia MPEG fue MPEG-2 (Estándar Internacional 13818), que originalmente se diseñó para comprimir vídeo con calidad de difusión a 4 o 6 Mbps, a fin de que se ajustara en un canal de difusión NTSC o PAL. Posteriormente, MPEG-2 se extendió para soportar resoluciones mayores, entre ellas HDTV. En la actualidad es muy común y forma la base para el DVD y la televisión por satélite digital.

Los principios básicos de MPEG-1 y MPEG-2 son parecidos, pero los detalles son diferentes. A primera vista, MPEG-2 es un supergrupo del MPEG-1, con características adicionales, formatos de trama y opciones de codificación. Estudiaremos MPEG-1 primero y MPEG-2 después.

MPEG-1 tiene tres partes: audio, vídeo y sistema, que integra los otros dos, como se muestra en la figura 7-76. Los codificadores de audio y vídeo funcionan en forma independiente, lo que hace surgir la cuestión de cómo se sincronizan los dos flujos en el receptor. Este problema se resuelve teniendo un reloj de sistema de 90 kHz que proporciona el valor de tiempo actual a ambos codificadores. Estos valores son de 33 bits, para permitir que las películas duren 24 horas sin que

den la vuelta. Estas marcas de tiempo se incluyen en la salida codificada y se propagan hasta el receptor, que puede usarlas para sincronizar los flujos de audio y vídeo.

Ahora consideremos la compresión del vídeo MPEG-1. Existen dos clases de redundancia en las películas: espacial y temporal. MPEG-1 aprovecha ambas. La redundancia espacial puede utilizarse simplemente codificando por separado cada trama mediante JPEG. Este enfoque se usa en forma ocasional, sobre todo cuando se requiere acceso aleatorio a cada trama, como en la edición de producciones de vídeo. En este modo, se puede lograr un ancho de banda comprimido en el rango de 8 a 10 Mbps.

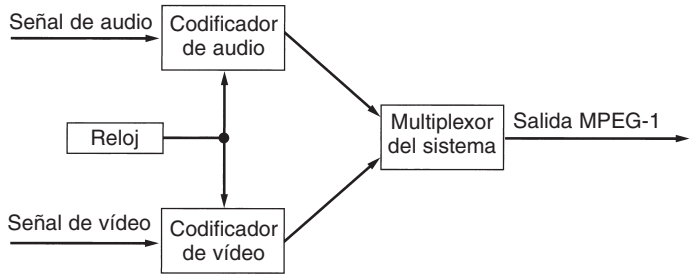


Figura 7-76. Sincronización de los flujos de audio y vídeo en MPEG-1.

Puede lograrse compresión adicional aprovechando el hecho de que las tramas consecutivas a menudo son casi idénticas. Este efecto es menor de lo que podría pensarse, puesto que muchos cineastas hacen cortes entre escenas cada 3 o 4 segundos (tome el tiempo de una película y cuente las escenas). No obstante, incluso una serie de 75 tramas muy parecidas ofrece una reducción importante que simplemente codificar cada trama por separado mediante JPEG.

En escenas en las que la cámara y el fondo son estacionarios y uno o dos actores se mueven con lentitud, casi todos los píxeles serán idénticos de una trama a otra. Aquí bastará con restar cada trama de la anterior y con aplicar JPEG a la diferencia. Sin embargo, esto no es muy bueno en las escenas en las que la cámara hace una panorámica o un acercamiento. Lo que se necesita es una forma para compensar este movimiento. Esto es precisamente lo que hace MPEG, y es la diferencia principal entre MPEG y JPEG.

La salida de MPEG-1 consiste en cuatro tipos de trama:

1. Tramas I (intracodificadas): imágenes fijas autocontenidas codificadas en JPEG.
2. Tramas P (predictivas): diferencia de bloque por bloque con la trama anterior.
3. Tramas B (bidireccionales): diferencias entre la trama anterior y la siguiente.
4. Tramas D (codificación CD): promedios de bloque usados para avance rápido.

Las tramas I son sólo imágenes fijas codificadas con una variante de JPEG, y también con luminancia de definición completa y crominancia de definición media sobre cada eje. Es necesario hacer que las tramas I aparezcan en forma periódica en el flujo de salida por tres razones. Primera, MPEG-1 puede usarse para transmisión de multidifusión, en la que los usuarios pueden realizar la sintonización a voluntad. Si todas las tramas dependieran de sus antecesores remontándose a la primera trama, cualquiera que no recibiera la primera trama nunca podría decodificar las tramas subsiguientes. Segunda, si una trama se recibiera con error, no sería posible ninguna decodificación posterior. Tercera, sin tramas I, al hacer un avance o retroceso rápido, el decodificador tendría que calcular cada trama por la que pasa para conocer el valor completo de aquella en la que se detiene. Por estas tres razones, se insertan tramas I en la salida una o dos veces por segundo.

En contraste, las tramas P codifican las diferencias entre tramas; se basan en la idea de los **macrobloques,** que cubren 16×16 píxeles de espacio de luminancia y 8×8 píxeles de espacio de crominancia. Un macrobloque se codifica buscando en la trama previa algo igual a él o ligeramente diferente de él.

En la figura 7-77 se muestra un caso en el que serían útiles las tramas P. Aquí vemos tres tramas consecutivas que tienen el mismo fondo, pero en las que cambia la posición de una persona. Los macrobloques que contienen el fondo serán exactamente iguales, pero los macrobloques que contienen la persona estarán desfasados en alguna cantidad desconocida y tendrán que rastrearse.

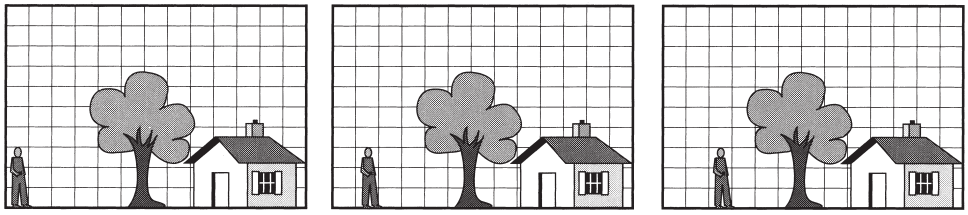


Figura 7-77. Tres tramas consecutivas.

El estándar MPEG-1 no especifica cómo efectuar la búsqueda, ni qué tan profunda o buena debe ser. Éstas son decisiones que dependen de cada implementación. Por ejemplo, una implementación podría buscar un macrobloque en la posición actual, pero de la trama anterior, y con todas las demás posiciones desplazadas $\pm\Delta x$ en la dirección x y $\pm\Delta y$ en la dirección y. Para cada posición, podría calcularse la cantidad de equivalencias en la matriz de luminancia. La posición con el puntaje más alto se declararía ganadora, siempre y cuando estuviera por encima de un umbral predefinido. En caso contrario, se diría que falta el macrobloque. Por supuesto, pueden usarse algoritmos mucho más refinados.

Si se encuentra un macrobloque, se codifica tomando la diferencia respecto a su valor en la trama previa (para la luminancia y ambas crominancias). Estas matrices de diferencias son el objeto de la transformación por coseno discreto, cuantización, codificación por longitud de serie y

codificación Huffman, al igual que en el JPEG. El valor del macrobloque en el flujo de salida es entonces el vector de movimiento (la distancia que se movió el macrobloque de su posición previa en cada sentido), seguido de la lista de codificación Huffman de los números. Si el macrobloque no se encuentra en la trama previa, se codifica el valor actual con JPEG, igual que en una trama I.

Es claro que este algoritmo es altamente asimétrico. Una implementación es libre de intentar todas las posiciones de la trama previa si lo desea, en un intento desesperado por localizar todos los macrobloques previos. Este método reducirá al mínimo el flujo MPEG-1 codificado al costo de una codificación muy lenta. Este método podría estar bien para la codificación de una sola vez de una cineteca, pero sería terrible para las videoconferencias en tiempo real.

De la misma manera, cada implementación puede decidir lo que constituye un macrobloque "encontrado". Esto permite que los implementadores compitan según la calidad y velocidad de sus algoritmos, pero siempre produce MPEG-1 que se apega. Sea cual sea el algoritmo de búsqueda usado, la salida final es la codificación JPEG del macrobloque actual, o la codificación JPEG de la diferencia entre el macrobloque actual y el de la trama previa, con un desplazamiento especificado respecto a la trama actual.

Hasta ahora, la decodificación de MPEG-1 es directa. La decodificación de tramas I es igual a la de las imágenes JPEG. La decodificación de tramas P requiere que el decodificador almacene en el búfer la trama previa, y luego construya la nueva en un segundo búfer con base en macrobloques completamente codificados y en macrobloques que contienen diferencias respecto a la trama previa. La nueva trama se ensambla macrobloque por macrobloque.

Las tramas B son parecidas a las P, excepto que permiten que el macrobloque de referencia esté en una trama previa o en una trama posterior. Esta libertad adicional permite mejorar la compensación del movimiento y también es útil cuando pasan objetos por delante o detrás de otros objetos. Para ejecutar la codificación de tramas B, el codificador necesita tener a la vez tres tramas decodificadas en la memoria: la anterior, la actual y la siguiente. Aunque las tramas B producen la mejor compresión, no todas las implementaciones las soportan.

Las tramas D sólo se usan para visualizar una imagen de baja definición cuando se realiza un rebobinado o un avance rápido. Hacer la decodificación MPEG-1 normal en tiempo real ya es bastante difícil. Esperar que el decodificador lo haga mientras trabaja a diez veces su velocidad normal es pedir demasiado. En su lugar, las tramas D se utilizan para producir imágenes de baja resolución. Cada entrada de trama D simplemente es el valor promedio de un bloque, sin mayor codificación, lo que simplifica la presentación en tiempo real. Este mecanismo es importante para permitir que la gente barra un vídeo a alta velocidad en busca de una escena en particular. Por lo general, las tramas D se colocan justo antes de las tramas I correspondientes, por lo que si se detiene el avance rápido, será posible ver nuevamente a una velocidad normal.

Una vez terminado nuestro tratamiento de MPEG-1, pasemos al de MPEG-2. La codificación MPEG-2 es parecida en lo fundamental a la codificación MPEG-1, con tramas I, tramas P y tramas B. Sin embargo, no se soportan las tramas D. Además, la transformación por coseno discreto es de 10×10 en lugar de 8×8, lo que da 50% más de coeficientes y, por lo tanto, mayor calidad. Puesto que

MPEG-2 está dirigido a la televisión difundida al igual que al DVD, reconoce imágenes tanto progresivas como entrelazadas, mientras que MPEG-1 reconoce sólo las imágenes progresivas. También son diferentes otros detalles menores entre los dos estándares.

En lugar de soportar sólo un nivel de resolución, MPEG-2 soporta cuatro: bajo (352 × 240), medio (720 × 480), alto 1440 (1440 × 1152) y alto (1920 × 1080). La resolución baja es para las VCRs y por compatibilidad hacia atrás con MPEG-1. La resolución media es la normal para la difusión NTSC. Las otras dos son para HDTV. Para una salida de alta calidad, MPEG-2 por lo general se ejecuta de 4 a 8 Mbps.

7.4.8 Vídeo bajo demanda

El vídeo bajo demanda a veces se compara con una tienda de renta de vídeos. El usuario (cliente) selecciona cualquiera de los vídeos disponibles y se lo lleva a casa para verlo. Con el vídeo bajo demanda, la selección se realiza en casa utilizando el control remoto de la televisión y el vídeo comienza de inmediato. No se necesita ningún viaje a la tienda. Sobra decir que la implementación del vídeo bajo demanda es un tanto más complicada que su descripción. En esta sección daremos un repaso general de los conceptos básicos y de su implementación.

¿El vídeo bajo demanda es en realidad como rentar un vídeo, o se parece más a seleccionar una película de un sistema de cable de 500 canales? La respuesta tiene implicaciones técnicas importantes. En particular, los usuarios que rentan vídeos están acostumbrados a la idea de poder detener el vídeo, hacer un viaje rápido a la cocina o al baño, y luego continuar a partir de donde detuvieron el vídeo. Los televidentes no esperan tener la capacidad de poner en pausa los programas.

Si el vídeo bajo demanda va a competir con éxito contra las tiendas de renta de vídeos, podría ser necesario dar a los usuarios la capacidad de detener, iniciar y rebobinar los vídeos a voluntad. Para ello, el proveedor de vídeos tendrá que transmitir una copia individual a cada quien.

Por otra parte, si el vídeo bajo demanda se considera más como una televisión avanzada, entonces puede ser suficiente hacer que el proveedor de vídeo inicie cada vídeo popular, digamos, cada 10 minutos, y luego lo exhiba sin parar. Un usuario que desee ver un vídeo popular podría tener que esperar hasta 10 minutos para que comience. Aunque no es posible la pausa/reinicio aquí, un televidente que regrese a la sala tras una interrupción corta puede pasar a otro canal que muestre el mismo vídeo, pero con 10 minutos de retraso. Una parte del material se repetirá, pero no se perderá nada. Este esquema se llama **casi vídeo bajo demanda,** y ofrece la posibilidad de un costo mucho menor, puesto que la misma alimentación del servidor de vídeo puede llegar a muchos usuarios al mismo tiempo. La diferencia entre vídeo bajo demanda y casi vídeo bajo demanda es parecida a la diferencia entre manejar su propio auto y tomar el autobús.

Ver películas (casi) bajo demanda es un servicio de una gran grama de nuevos servicios que podrían hacerse realidad una vez que estén disponibles las redes de banda amplia. En la figura 7-78 se muestra el modelo general utilizado por mucha gente. Aquí vemos una red dorsal de área

amplia (nacional o internacional) de alto ancho de banda como centro del sistema. Conectadas a ella hay miles de redes de distribución locales, como TV por cable o sistemas de distribución de compañías telefónicas. Los sistemas de distribución locales llegan hasta las casas de la gente, donde terminan en **cajas de control,** que de hecho son potentes computadoras personales especializadas.

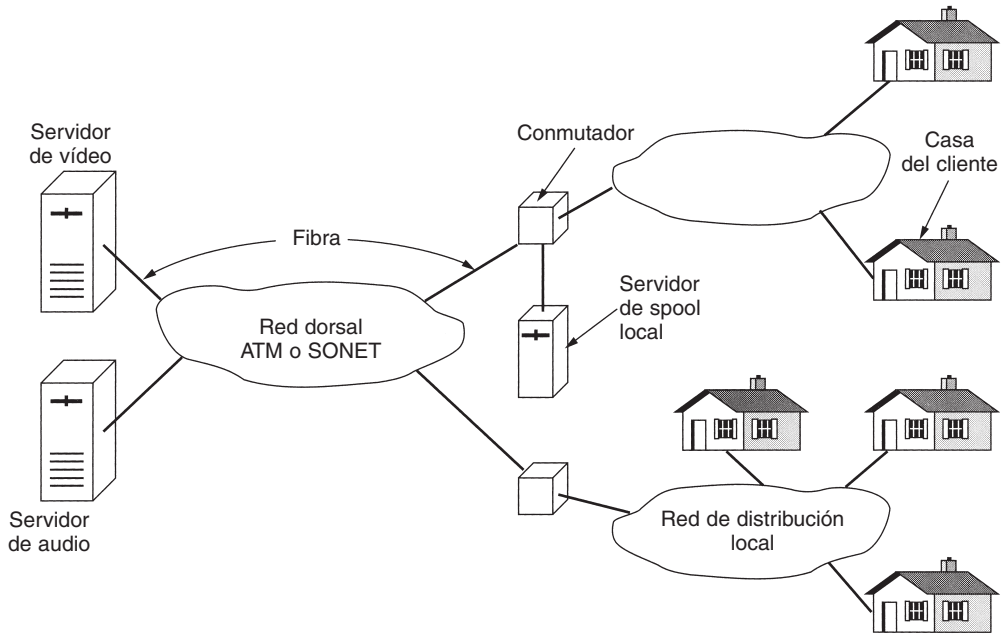


Figura 7-78. Esquema general de un sistema de vídeo bajo demanda.

Hay miles de proveedores de información conectados a la red dorsal mediante fibras ópticas de alto ancho de banda. Algunos de éstos ofrecerán vídeo o audio de pago por evento. Otros ofrecerán servicios especializados, como compras desde casa (en los que se permite que el televidente gire una lata de sopa y haga un acercamiento a la lista de ingredientes que aparece en la etiqueta de dicha lata, o que vea un fragmento de vídeo que trata sobre cómo manejar una podadora que utiliza gasolina). Sin duda, pronto habrá la disponibilidad de deportes, noticias, capítulos viejos del "Chapulín Colorado", acceso a WWW y otras innumerables posibilidades.

En el sistema también se incluyen servidores de *spool* de E/S locales que permitirán acercar los vídeos al usuario, a fin de ahorrar ancho de banda durante las horas pico. La manera en que encajarán estas partes y quién será el dueño de qué son temas de intenso debate en la industria. A continuación examinaremos el diseño de las partes principales del sistema: los servidores de vídeo y la red de distribución.

Servidores de vídeo

Para tener (casi) vídeo bajo demanda, necesitamos **servidores de vídeo** capaces de almacenar y sacar una gran cantidad de películas de manera simultánea. La cantidad total de películas que se han filmado se estima en 65,000 (Minoli, 1995). Cuando se comprime con MPEG-2, una película normal ocupa unos 4 GB, por lo que 65,000 de ellas requerirán unos 260 terabytes. Sume a todo esto todos los programas de televisión, filmaciones de deportes, noticieros, catálogos parlantes de compras, etcétera, y queda claro que tenemos en nuestras manos un problema de almacenamiento de proporciones mayúsculas.

La manera más sencilla de almacenar volúmenes grandes de información es en cinta magnética. Siempre ha sido el caso y probablemente seguirá siéndolo. Una cinta Ultrium puede almacenar 200 GB (50 películas) a un costo de aproximadamente 1-2 dólares/película. En el mercado hay servidores mecánicos grandes que almacenan miles de cintas y tienen un brazo de robot para traer cualquier cinta e introducirla en la unidad de cinta. El problema de estos sistemas es el tiempo de acceso (especialmente para la película 50 de una cinta), la tasa de transferencia y la cantidad limitada de unidades de cinta (para proporcionar n películas a la vez, se requieren n unidades).

Por fortuna, la experiencia de las tiendas de renta de vídeos, bibliotecas públicas y otras organizaciones similares demuestra que no todos los productos tienen la misma popularidad. Experimentalmente, cuando hay N películas disponibles, la fracción de todas las solicitudes que pide el elemento número k en popularidad es de aproximadamente C/k. Aquí, C se calcula para normalizar la suma a 1, es decir

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdots + 1/N)$$

Por lo tanto, la película más vista es siete veces más popular que la película número siete. Este resultado se conoce como **ley de Zipf** (Zipf, 1949).

El hecho de que algunas películas sean mucho más populares que otras sugiere una solución posible en forma de jerarquía de almacenamiento, como se muestra en la figura 7-79. Aquí, el desempeño aumenta a medida que se sube en la jerarquía.

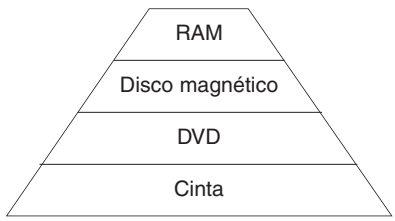


Figura 7-79. Jerarquía de almacenamiento de un servidor de vídeo.

Una alternativa a la cinta es el almacenamiento óptico. Los DVDs actuales almacenan 4.7 GB, que sirve para una película, pero la siguiente generación podrá almacenar dos películas. Aunque los tiempos de búsqueda son lentos en comparación con los discos magnéticos (50 mseg contra

5 mseg), su bajo costo y alta confiabilidad hace que los tocadiscos ópticos que contienen miles de DVDs sean una buena alternativa a la cinta para las películas de mayor uso.

A continuación están los discos magnéticos. Éstos tienen tiempos de acceso cortos (5 mseg), tasas de transferencia altas (320 MB/seg para SCSI 320) y capacidades sustanciales (> 100 GB), lo que los hace adecuados para guardar películas que en realidad se están transmitiendo (a diferencia de simplemente estar almacenadas en caso de que alguien quiera verlas). Su desventaja principal es el alto costo de almacenar películas a las que pocas veces se accede.

En la parte superior de la pirámide de la figura 7-79 está la RAM. Ésta es el medio de almacenamiento más rápido, pero también el más costoso. Cuando los precios de la RAM bajen a $50/GB, la RAM de una película de 4 GB costará $200, por lo que la RAM de 100 películas costará $20,000, debido a que éstas ocuparán 400 GB. Sin embargo, se está volviendo algo factible el que un servidor de vídeo que almacene 100 películas pueda mantenerlas a todas en la RAM. Y si dicho servidor tiene 100 clientes, pero éstos ven colectivamente sólo 20 películas diferentes, no sólo es algo factible sino un buen diseño.

Debido a que un servidor de vídeo en realidad es un dispositivo masivo de E/S en tiempo real, necesita una arquitectura de hardware y software diferente de la de una PC o una estación de trabajo UNIX. En la figura 7-80 se ilustra la arquitectura del hardware de un servidor de vídeo típico. El servidor tiene una o más CPUs de alto desempeño, cada una con un poco de memoria local, una memoria principal compartida, un caché de RAM masivo para películas populares, una variedad de dispositivos de almacenamiento para guardar las películas, y algún hardware de red, normalmente una interfaz óptica con una red dorsal ATM (o SONET) a OC-12 o mayor. Estos subsistemas se conectan mediante un bus de velocidad extremadamente alta (cuando menos 1 GB/seg).

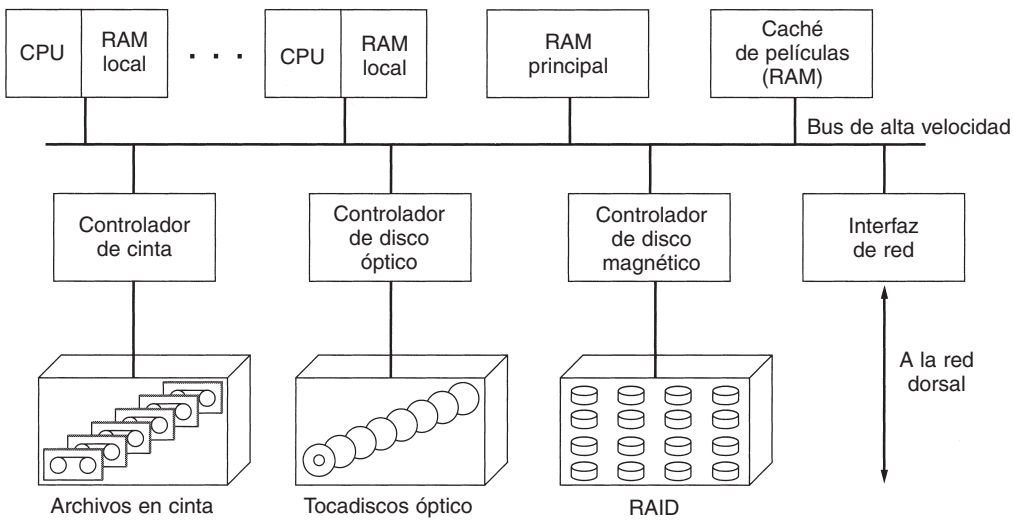


Figura 7-80. Arquitectura del hardware de un servidor de vídeo típico.

Ahora veamos brevemente el software de un servidor de vídeo. Las CPUs se usan para aceptar solicitudes de los usuarios, localizar películas, mover datos entre dispositivos, facturar al cliente

y muchas funciones más. Algunas de éstas no son de sincronización crítica, pero muchas otras sí, por lo que algunas de las CPUs, si no es que todas, tendrán que ejecutar un sistema operativo en tiempo real, como un micronúcleo en tiempo real. Por lo general, estos sistemas dividen el trabajo en tareas pequeñas, con un plazo de terminación conocido. Después el planificador puede ejecutar un algoritmo del tipo del siguiente plazo de terminación más cercano o el algoritmo de tasa monotónica (Liu y Layland, 1973).

El software de la CPU también define la naturaleza de la interfaz que el servidor presenta a los clientes (servidores de *spool* o cajas de control colocadas sobre el televisor). Son comunes dos diseños. El primero es un sistema tradicional de archivos, en el que los clientes pueden abrir, leer, escribir y cerrar archivos. Aparte de las complicaciones generadas por la jerarquía de almacenamiento y las consideraciones en tiempo real, uno de tales servidores puede tener un sistema de archivos modelado según el de UNIX.

El segundo tipo de interfaz se basa en el modelo de la videograbadora. Los comandos le piden al servidor que abra, ejecute, haga pausa, avance rápido y rebobine archivos. La diferencia con el modelo UNIX es que, una vez que se ha emitido un comando *PLAY*, el servidor simplemente sigue sacando datos a velocidad constante, sin que se requieran comandos nuevos.

El corazón del software del servidor de vídeo es el software de administración de disco, que tiene dos tareas principales: colocar películas en el disco magnético cuando tienen que traerse de almacenamiento óptico o cinta, y manejar solicitudes de disco para los muchos flujos de salida. La colocación de las películas es importante, puesto que puede afectar en gran medida el desempeño.

Dos maneras posibles de organizar el almacenamiento en disco son la granja de discos y el arreglo de discos. En la **granja de discos,** cada unidad contiene algunas películas completas. Por cuestiones de desempeño y confiabilidad, cada película debe estar presente en cuando menos dos unidades, tal vez más. La otra organización del almacenamiento es el **arreglo de discos** o **RAID** (**arreglo redundante de discos baratos**)**,** en el que cada película se divide entre varias unidades, por ejemplo, el bloque 0 en la unidad 0, el bloque 1 en la unidad 1, etcétera, con el bloque $n - 1$ en la unidad $n - 1$. Después de eso, el ciclo se repite, con el bloque n en la unidad 0, etcétera. Esta organización se llama **división** (***striping***)**.**

Un arreglo de discos con división tiene varias ventajas respecto a una granja de discos. Primero, las n unidades pueden operar en paralelo, aumentando el desempeño en un factor de n. Segundo, el arreglo puede hacerse redundante agregando una unidad extra a cada grupo de n, donde la unidad redundante contiene el OR exclusivo bloque por bloque de las otras unidades, a fin de permitir la recuperación completa de datos en caso de que una unidad falle. Por último, se resuelve el problema del equilibrio de la carga (no se requiere colocación manual para evitar que todas las películas populares se encuentren en la misma unidad). Por otra parte, la organización por arreglo de discos es más complicada que la granja de discos y es altamente sensible a diversas fallas; también es poco adecuada para operaciones de videograbadora, por ejemplo, para el rebobinado y avance rápido de una película.

La otra tarea del software del disco es dar servicio a todos los flujos de salida en tiempo real y cumplir sus restricciones de temporización. Hace apenas algunos años, esto requería algoritmos

de programación de disco complejos, pero ahora que los precios de la memoria están tan bajos, es posible un método mucho más simple. Por cada flujo servido, se almacena en RAM un búfer de, digamos, 10 seg de vídeo (5 MB). Se llena mediante un proceso de disco y se vacía mediante un proceso de red. Con 500 MB de RAM se pueden alimentar 100 flujos directamente de la RAM. Por supuesto, el subsistema de disco debe tener una tasa de datos sostenida de 50 MB/seg para mantener los búferes llenos, pero un RAID construido a partir de discos SCSI de alta calidad puede manejar fácilmente este requerimiento.

La red de distribución

La red de distribución es el grupo de conmutadores y líneas entre el origen y el destino. Como vimos en la figura 7-78, esta red consiste en una red dorsal conectada a la red local de distribución. Generalmente, la red dorsal es conmutada y la red de distribución local no lo es.

El requisito principal impuesto en la red dorsal es un ancho de banda alto. Una fluctuación baja también era un requisito, pero ahora ya no lo es, incluso ni en las PCs actuales más pequeñas, las cuales tienen la capacidad de almacenar en el búfer 10 seg de vídeo MPEG-2 de alta calidad.

La distribución local es caótica, pues las diferentes compañías usan redes distintas en diferentes regiones. Las compañías telefónicas, las de televisión por cable y los nuevos participantes, como las compañías de electricidad, están convencidos de que el que llegue primero será el gran ganador, por lo que ahora vemos una proliferación en la instalación de nuevas tecnologías. En Japón, algunas compañías de alcantarillado están en el negocio de Internet, argumentando que ellas tienen el canal más grande en la casa de todos (ejecutan una fibra óptica a través de él, pero deben ser muy cuidadosas sobre en qué lugar emerge). Los cuatro esquemas principales de distribución local de vídeo bajo demanda tienen las siglas ADSL, FTTC, FTTH y HFC. Ahora los explicaremos.

ADSL (**Línea Digital de Suscriptor Asimétrica**) fue el primer competidor de la industria telefónica por el premio de la distribución local. La idea es que en prácticamente cada casa de Estados Unidos, Europa y Japón ya tiene un par trenzado de cobre (para el servicio telefónico analógico). Si estos cables pudieran usarse para el vídeo bajo demanda, las compañías telefónicas podrían ganar.

Por supuesto, el problema es que estos cables no pueden manejar ni siquiera MPEG-1 a través de su recorrido normal de 10 km, y ni mencionar MPEG-2. El vídeo de movimiento y color completos, y de alta calidad necesita de 4 a 8 Mbps, dependiendo de la calidad deseada. ADSL realmente no es tan rápida, excepto para los ciclos locales muy cortos.

El segundo diseño de las compañías telefónicas es el **FTTC** (**Fibra Hasta la Acera**). En FTTC, la compañía telefónica tiende fibra óptica de la oficina central a cada barrio residencial, terminando en un dispositivo llamado **ONU** (**Unidad de Red Óptica**). En una ONU pueden terminar hasta 16 circuitos locales de cobre. Estos circuitos son lo bastante cortos para operar T1 o T2 de dúplex total en ellos, permitiendo películas MPEG-1 y MPEG-2, respectivamente. Además,

son posibles videoconferencias para trabajadores remotos y negocios pequeños, pues el FTTC es simétrico.

La tercera solución de las compañías telefónicas es tender fibra en todas las casas. Este sistema se llama **FTTH (Fibra Hasta la Casa).** En él, todo mundo puede tener una portadora OC-1, OC-3 o inclusive mayor, si se requiere. La FTTH es muy costosa y tardará años en hacerse realidad, pero ciertamente abrirá una vasta gama de nuevas posibilidades cuando finalmente ocurra. En la figura 7-63 vimos la forma en que cualquier persona puede operar su propia estación de radio. ¿Qué pensaría de que cada miembro de la familia opere su propia estación de televisión? ADSL, FTTC y FTTH son redes de distribución local de punto a punto, lo cual no es sorprendente debido a la forma en que está organizado el sistema telefónico actual.

Un enfoque completamente diferente es el **HFC (Híbrido Fibra/Coaxial),** la solución preferida que están instalando los proveedores de TV por cable y que se ilustra en la figura 2-47(a). La propuesta es la siguiente. Los cables coaxiales actuales de 300 a 450 MHz serán sustituidos por cables coaxiales de 750 MHz, elevando la capacidad de 50 a 75 canales de 6 MHz a 125 canales de 6 MHz. Setenta y cinco de los 125 canales se usarán para la transmisión de televisión analógica.

Los 50 canales nuevos se modularán de manera individual usando QAM-526, que proporciona unos 40 Mbps por canal, lo que da un total de 2 Gbps de ancho de banda nuevo. Los amplificadores *head-end* se moverán más hacia el interior de los vecindarios, de modo que cada cable pase por sólo 500 casas. Una división sencilla indica que es posible asignar a cada casa un canal dedicado de 4 Mbps, que puede manejar una película MPEG-2.

Aunque esto suena maravilloso, requiere que los proveedores de cable reemplacen todos los cables existentes con coaxial de 750 MHz, instalen nuevos amplificadores *head-end* y eliminen todos los amplificadores de una vía; en pocas palabras, que reemplacen todo el sistema de TV por cable. En consecuencia, la cantidad de infraestructura nueva es comparable con la que necesitan las compañías telefónicas para FTTC. En ambos casos, el proveedor local de la red debe tender fibra en los barrios residenciales. Nuevamente, en ambos casos, la fibra termina en un convertidor optoeléctrico. En FTTC, el segmento final es un circuito local punto a punto que usa cable de par trenzado. En HFC, el segmento final es cable coaxial compartido. Técnicamente, estos sistemas no son tan diferentes, como suelen asegurar sus patrocinadores.

No obstante, hay una diferencia real que es importante indicar. HFC usa un medio compartido sin conmutación ni enrutamiento. Cualquier información que se ponga en el cable puede ser retirada por cualquier suscriptor sin mayores trámites. FTTC, que es completamente conmutado, no tiene esta propiedad. Como resultado, los operadores HFC quieren que los servidores de vídeo transmitan flujos cifrados, de modo que los clientes que no hayan pagado una película no puedan verla. A los operadores FTTC no les interesa la encriptación, puesto que agrega complejidad, disminuye el desempeño y no proporciona seguridad adicional a su sistema. Desde el punto de vista de la compañía que opera un servidor de vídeo, ¿es buena idea encriptar? Un servidor operado por una compañía telefónica o una de sus subsidiarias o socios podría decidir de manera intencional no encriptar sus vídeos, e indicar como razón la eficiencia, pero en realidad lo hace para provocar pérdidas a sus competidores HFC.

Con todas estas redes locales de distribución es probable que se equipe a cada vecindario con uno o más servidores de *spool*. Éstos son, de hecho, simplemente versiones más pequeñas de los servidores de vídeo que estudiamos antes. La gran ventaja de estos servidores locales es que eliminan algo de la carga de la red dorsal.

Dichos servidores locales pueden recargarse con películas por reservación. Si las personas indican con suficiente anticipación al proveedor qué películas desean ver, éstas pueden bajarse al servidor local en horas no pico, con lo que se obtienen aún más ahorros. Probablemente esta observación llevará a los operadores de red a despedir a los ejecutivos de aviación que diseñan sus tarifas. De esta manera puede planearse un descuento en las tarifas de las películas ordenadas con 24 a 72 horas de anticipación que vayan a verse un martes o jueves por la tarde, antes de las 6 p.m. o después de las 11 p.m. Las películas ordenadas el primer domingo del mes antes de las 8 a.m. para verse un miércoles por la tarde de un día cuya fecha sea un número primo tendrán 43% de descuento, y así sucesivamente.

7.4.9 Mbone—Red dorsal de multidifusión

Mientras todas estas industrias hacen grandes (y muy publicitados) planes para el vídeo digital (inter)nacional bajo demanda, la comunidad de Internet ha estado implementando calladamente su propio sistema digital de multimedia, **MBone** (**Red Dorsal de Multidifusión**)**.** En esta sección daremos un panorama de lo que es y cómo funciona.

Puede pensar en MBone como una radio y televisión de Internet. A diferencia del vídeo bajo demanda, donde lo más importante es solicitar y ver películas precomprimidas almacenadas en un servidor, MBone se usa para difundir audio y vídeo en vivo de forma digital por todo el mundo a través de Internet. MBone ha estado en operación desde comienzos de 1992. Se han difundido muchas conferencias científicas, especialmente las reuniones del IETF, así como eventos científicos de importancia noticiosa, como varios lanzamientos del trasbordador espacial. Alguna vez se difundió un concierto de los Rolling Stones por MBone, al igual que algunas partes del Festival de Cine de Canes. El hecho de que este suceso se califique como un hecho científico notable es discutible.

Desde el aspecto técnico, MBone es una red virtual sobrepuesta a Internet. Mbone consiste en islas capaces de multidifundir, conectadas mediante túneles, como se muestra en la figura 7-81. En esta figura, la MBone consiste en seis islas, *A* a *F*, conectadas mediante siete túneles. Cada isla (por lo común, una LAN o grupo de LANs interconectadas) soporta la multidifusión por *hardware* a sus *hosts*. Los túneles propagan paquetes MBone entre las islas. En el futuro, cuando todos los enrutadores sean capaces de manejar en forma directa tráfico de multidifusión, dejará de necesitarse esta superestructura, pero por el momento lleva a cabo el trabajo.

Cada isla contiene uno o más enrutadores especiales, llamados **enrutadores m** o *mrouters* (**enrutadores de multidifusión**)**.** Algunos de éstos son en realidad enrutadores normales, pero otros sencillamente son estaciones de trabajo UNIX que ejecutan software especial de nivel de usuario (pero como la raíz). Los enrutadores m se conectan de manera lógica mediante túneles.

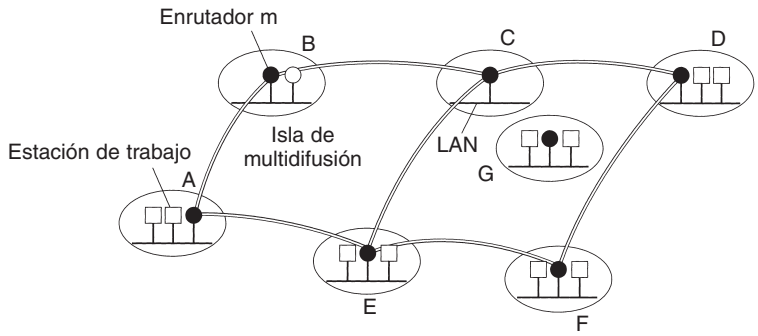


Figura 7-81. MBone consiste en islas de multidifusión conectadas mediante túneles.

Los paquetes MBone se encapsulan en paquetes IP y se envían como paquetes de unidifusión normales a la dirección IP del enrutador m de destino.

Los túneles se configuran de manera manual. Por lo general, un túnel corre por una ruta para la que existe una conexión física, pero esto no es un requisito. Si por accidente se cae la ruta física subyacente de un túnel, los enrutadores m que usan el túnel ni siquiera lo notarán, puesto que la Internet reenrutará automáticamente todo el tráfico IP entre ellos por otras líneas.

Cuando aparece una isla nueva que desea unirse a la MBone, como en el caso de *G* de la figura 7-81, su administrador envía un mensaje que indica su existencia a la lista de correo de MBone. Los administradores de los sitios cercanos se ponen en contacto con ella para acordar el establecimiento de túneles. A veces los túneles existentes se reordenan para aprovechar la nueva isla y optimizar la topología. A fin de cuentas, los túneles no existen físicamente, se definen mediante tablas en los enrutadores m y pueden agregarse, eliminarse o moverse con sólo cambiar estas tablas. Por lo general, cada país en la MBone tiene una red dorsal con islas regionales conectadas a ella. Normalmente, la MBone se configura con uno o dos túneles que cruzan el Océano Atlántico y el Pacífico, lo que hace que la MBone sea de escala global.

Por lo tanto, en cualquier momento la MBone consiste en una topología específica de túneles e islas, independiente de la cantidad de direcciones de multidifusión en uso en el momento y de quiénes las estén escuchando u observando. Esta situación es muy parecida a una subred normal (física), por lo que se aplican los mismos algoritmos de enrutamiento. En consecuencia, la MBone inicialmente usaba un algoritmo de enrutamiento, el **DVMRP** (**Protocolo de Enrutamiento Multidifusión de Vector de Distancia**)**,** que se basa en el algoritmo de vector de distancia Bellman-Ford. Por ejemplo, en la figura 7-81, la isla *C* puede enrutar a *A* vía *B* o *E* (o tal vez vía *D*). *C* lo decide tomando los valores que le dan los nodos sobre sus respectivas distancias a *A*, y sumando después su distancia a ellos. De esta manera, cada isla determina la mejor ruta a todas las demás islas. Sin embargo, las rutas en realidad no se usan de esta manera, como veremos en breve.

Consideremos ahora la manera en que ocurre la multidifusión. Para multidifundir un programa de audio o vídeo, una fuente primero debe adquirir una dirección de multidifusión clase D, que actúa como la frecuencia o el número de canal de una estación. Las direcciones clase D se reservan usando un programa que busca direcciones de multidifusión libres en una base de datos. Pueden ocurrir simultáneamente muchas multidifusiones, y un *host* puede "sintonizarse" con aquellas en la que está interesado si escucha en la dirección de multidifusión adecuada.

Periódicamente, cada enrutador m envía un paquete de difusión IGMP limitando a su isla, preguntando quién está interesado en qué canal. Los *hosts* que desean (continuar) recibiendo uno o más canales envían como respuesta otro paquete IGMP. Estas respuestas se reparten en el tiempo, para evitar sobrecargar la LAN local. Cada enrutador m mantiene una tabla de los canales que debe poner en sus LANs, para evitar el desperdicio de ancho de banda al multidifundir canales que nadie quiere.

Las multidifusiones se propagan por la MBone como se explica a continuación. Cuando una fuente de audio o vídeo genera un paquete nuevo, lo multidifunde a su isla local usando el recurso de multidifusión del hardware. El enrutador m local recoge este paquete, y después lo copia en todos los túneles a los que está conectado.

Cada enrutador m que recibe uno de tales paquetes por medio de un túnel lo revisa para ver si llegó por la mejor ruta, es decir, la ruta que, según su tabla, debe usarse para llegar al origen (como si fuera el destino). Si el paquete llegó por la mejor ruta, el enrutador m copia el paquete en todos sus otros túneles. Si el paquete llegó por una ruta subóptima, se descarta. Por ejemplo, en la figura 7-81, si las tablas de *C* le indican que use *B* para llegar a *A*, entonces cuando un paquete de multidifusión de *A* a *C* llega vía *B*, se copia de ahí a *D* y a *E*. Sin embargo, cuando un paquete de multidifusión de *A* a *C* llega vía *E* (que no es la mejor ruta), simplemente se descarta. Este algoritmo sólo es el reenvío por ruta invertida que vimos en el capítulo 5; aunque no es perfecto, es bastante bueno y muy sencillo de implementar.

Además de usar reenvío por ruta invertida para evitar inundar la Internet, también se usa el campo *Tiempo de vida* del IP para limitar el alcance de la multidifusión. Cada paquete comienza con algún valor (determinado por el origen). A cada túnel se le asigna un peso. Un paquete sólo pasa a través de un túnel si tiene suficiente peso; de otra manera, se descarta. Por ejemplo, los túneles transoceánicos se configuran normalmente con un peso de 128, por lo que los paquetes pueden limitarse al continente de origen dándoles un *Tiempo de vida* inicial de 127 o menos. Tras pasar por el túnel, al campo de *Tiempo de vida* se le resta el peso del túnel.

Aunque el algoritmo de enrutamiento de la MBone funciona, se han realizado muchas investigaciones para mejorarlo. Una propuesta mantiene la idea del enrutamiento por vector de distancia, pero hace jerárquico el algoritmo, agrupando en regiones las instalaciones de la MBone y enrutando primero hacia ellas (Thyagarajan y Deering, 1995).

Otra propuesta es usar una forma modificada del enrutamiento por estado de enlace en lugar del enrutamiento por vector de distancia. En particular, un grupo de trabajo del IETF está modificando el OSPF para adecuarlo a la multidifusión en un solo sistema autónomo. El OSPF de multidifusión resultante se llama **MOSPF** (Moy, 1994). Lo que hacen las modificaciones es que el mapa completo construido por el MOSPF lleva el registro de las islas y túneles de multidifusión, además de la información normal de enrutamiento. Si se conoce la topología completa, es directo

el cálculo de la mejor ruta desde cualquier isla a cualquier otra usando los túneles. Por ejemplo, puede usarse el algoritmo de Dijkstra.

Una segunda área de investigación es el enrutamiento entre los AS. Aquí, otro grupo de trabajo del IETF está desarrollando un algoritmo llamado **PIM** (**Multidifusión Independiente del Protocolo**). El PIM viene en dos versiones, dependiendo de si las islas son densas (casi todos quieren ver) o dispersas (casi nadie quiere ver). Ambas versiones usan las tablas de enrutamiento de unidifusión estándar, en lugar de crear una topología superpuesta como lo hacen DVMRP y MOSPF.

En el PIM-DM (modo denso), la idea es recortar las rutas inútiles. El recortado funciona como sigue. Cuando llega un paquete de multidifusión por el túnel "equivocado", se devuelve un paquete de recorte por dicho túnel, indicando al transmisor que cese el envío de paquetes desde la fuente en cuestión. Cuando un paquete llega por el túnel "correcto", se copia en todos los demás túneles que no han sido previamente recortados. Si todos los demás túneles se han recortado y no hay interés por el canal en la isla local, el enrutador m devuelve un mensaje de recorte por el canal "correcto". De esta manera, la multidifusión se adapta en forma automática y sólo va a donde se necesita.

El PIM-SM (modo disperso), que se describe en el RFC 2362, funciona de manera diferente. Aquí la idea es evitar la saturación de Internet porque tres personas de Berkeley quieren realizar una llamada en conferencia por una dirección clase D. El PIM-SM opera estableciendo puntos de reunión. Cada una de las fuentes de un grupo de multidifusión PIM disperso envía sus paquetes a los puntos de reunión. Cualquier sitio interesado en unirse solicita a uno de los puntos de reunión que establezca un túnel a él. De esta manera, todo el tráfico PIM-SM transporta por unidifusión en lugar de multidifusión. PIM-SM se está volviendo más popular, y MBone está migrando a su uso. A medida que el uso de PIM-SM se está volviendo más común, el de MOSPF está desapareciendo gradualmente. Por otro lado, la MBone misma parece algo estática y probablemente nunca se vuelva muy popular.

Sin embargo, la multimedia en red aún es un campo emocionante y en rápido movimiento, aunque MBone no sea un gran éxito. Las nuevas tecnologías y aplicaciones se anuncian a diario. La multidifusión y la calidad del servicio cada vez están más unidas, como se discute en (Striegel y Manimaran, 2002). Otro tema interesante es la multidifusión inalámbrica (Gossain y cols., 2002). Es probable que en los próximos años toda el área de la multidifusión y todo lo relacionado con ella siga siendo importante.

7.5 RESUMEN

Los nombres de Internet utilizan un esquema jerárquico llamado DNS. En el nivel superior se encuentran los dominios genéricos bien conocidos, entre ellos *com* y *edu*, así como cerca de 200 dominios de países. DNS está implementado como un sistema de base de datos distribuido con servidores en todo el mundo. DNS mantiene registros con direcciones IP, registros de correo y otra información. Al consultar un servidor DNS, un proceso puede asignar un nombre de dominio de Internet a la dirección IP utilizada para comunicarse con el dominio.

El correo electrónico es una de las dos aplicaciones principales para Internet. Todo mundo, desde niños pequeños hasta los abuelos, lo utiliza. La mayoría de los sistemas de correo electrónico en el mundo utiliza el sistema de correo definido en los RFCs 2821 y 2822. Los mensajes enviados en este sistema utilizan encabezados ASCII de sistema para definir propiedades de mensajes. MIME se puede utilizar para enviar mucho contenido de diversos tipos. Los mensajes se envían utilizando SMTP, que funciona al establecer una conexión TCP del *host* de origen al de destino y entregando de manera directa el correo electrónico a través de la conexión TCP.

La otra aplicación principal de Internet es World Wide Web. Web es un sistema para vincular documentos de hipertexto. Originalmente, cada documento era una página escrita en HTML con hipervínculos a otros documentos. Hoy en día, XML está comenzando gradualmente a tomar ventaja sobre HTML. Además, una gran cantidad de contenido se genera en forma dinámica utilizando secuencias de comandos en el servidor (PHP, JSP y ASP), así como secuencias de comandos en el cliente (principalmente JavaScript). Un navegador puede desplegar un documento estableciendo una conexión TCP con su servidor, solicitando el documento y después cerrando la conexión. Estos mensajes de solicitud contienen una variedad de encabezados para proporcionar información adicional. El almacenamiento en caché, la replicación y las redes de entrega de contenido se utilizan ampliamente para mejorar el desempeño de Web.

La Web inalámbrica apenas está comenzando. Los primeros sistemas son WAP e i-mode, cada uno con pantallas pequeñas y ancho de banda limitado, pero la siguiente generación será más poderosa.

La multimedia también es una estrella en ascenso en el firmamento de las redes. Permite que el audio y el vídeo sean digitalizados y transportados de manera electrónica para su despliegue. El audio requiere menos ancho de banda. El audio de flujo continuo, la radio en Internet y la voz sobre IP ahora son una realidad, y continuamente surgen nuevas aplicaciones. El vídeo bajo demanda es un área prometedora en la que hay un gran interés. Por último, la MBone es un servicio experimental de televisión digital mundial que se envía a través de Internet.

PROBLEMAS

1. Muchas computadoras de negocios tienen tres identificadores únicos en todo el mundo. ¿Cuáles son?

2. De acuerdo con la información que se dio en la figura 7-3, ¿*little-sister.cs.vu.nl* corresponde a una red A, B o C?

3. En la figura 7-3, ¿hay un punto después de *rowboat*? ¿Por qué no?

4. Adivine qué significa :-X (algunas veces se escribe como :-#).

5. DNS utiliza UDP en lugar de TCP. Si se pierde un paquete DNS, no hay recuperación automática. ¿Esto causa un problema, y si es así, cómo se resuelve?
6. Además de ser propensos a perderse, los paquetes UDP tienen una longitud máxima, potencialmente tan baja como 576 bytes. ¿Qué pasa cuando un nombre DNS que se va a buscar excede esta longitud? ¿Se puede enviar en dos paquetes?
7. ¿Una máquina con un solo nombre DNS puede tener múltiples direcciones IP? ¿Cómo puede ocurrir esto?
8. ¿Una computadora puede tener dos nombres DNS que pertenecen a dominios de nivel superior diferentes? De ser así, dé un ejemplo razonable. De lo contrario, explique por qué no.
9. El número de compañías con un sitio Web ha crecido de manera explosiva en los años recientes. Como resultado, miles de compañías están registradas con el dominio *com*, lo que causa una carga pesada en el servidor de nivel superior de este dominio. Sugiera una manera de aliviar este problema sin cambiar el esquema de nombres (es decir, sin introducir nuevos nombres de dominio de nivel superior). Es válido que su solución requiera cambios al código cliente.
10. Algunos sistemas de correo electrónico soportan un campo de encabezado *Content Return:*. Especifique si el cuerpo de un mensaje se va a regresar en caso de que no se entregue. ¿Este campo pertenece al sobre o al encabezado?
11. Los sistemas de correo electrónico necesitan directorios a fin de que se puedan buscar las direcciones de correo electrónico de las personas. Para construir tales directorios y para que la búsqueda sea posible, los nombres deben dividirse en componentes estándar (por ejemplo, nombre, apellido). Mencione algunos problemas que deben resolverse a fin de que un estándar mundial sea aceptable.
12. La dirección de correo electrónico de una persona es su nombre de inicio de sesión + @ + el nombre de un dominio DNS con un registro *MX*. Los nombres de inicio de sesión pueden ser nombres de pila, apellidos, iniciales y todos los tipos de nombres. Suponga que una compañía grande decidió que se estaba perdiendo mucho correo debido a que las personas no sabían el nombre de inicio de sesión del receptor. ¿Hay alguna forma de que dicha compañía arregle este problema sin cambiar el DNS? De ser así, dé una propuesta y explique cómo funciona. De lo contrario, explique por qué no es posible.
13. Un archivo binario tiene una longitud de 3072 bytes. ¿Qué longitud tendrá si se codifica mediante base64, y se inserta un par CR+LF después de cada 80 bytes enviados y al final?
14. Considere el esquema de codificación MIME entrecomillado-imprimible. Mencione un problema que no se analiza en el texto y proponga una solución.
15. Nombre cinco tipos MIME que no se listan en este libro. Puede verificar su navegador o Internet para obtener información.
16. Suponga que desea enviar un archivo MP3 a un amigo, pero el ISP de éste limita a 1 MB la cantidad de correo entrante y el archivo MP3 es de 4 MB. ¿Hay alguna forma para manejar esta situación utilizando el RFC 822 y MIME?
17. Suponga que alguien establece un demonio de vacaciones y que después envía un mensaje justo antes de terminar su sesión. Desgraciadamente, el receptor ha estado de vacaciones una semana y también tiene un demonio de vacaciones en su lugar. ¿Qué sucede a continuación? Las respuestas grabadas se enviarán y regresarán hasta que alguien regrese?

18. En cualquier estándar, como el RFC 822, se necesita una gramática precisa de lo que está permitido de manera que diferentes implementaciones puedan interactuar. Incluso los elementos simples se tienen que definir con cuidado. Los encabezados SMTP permiten espacios en blanco entre los *tokens*. Dé *dos* definiciones de alternativas razonables de espacios en blanco entre los *tokens*.

19. ¿El demonio de vacaciones es parte del agente de usuario o del agente de transferencia de mensajes? Por supuesto, se establece con el agente de usuario, pero ¿éste envía realmente las respuestas? Explique.

20. POP3 permite que los usuarios obtengan y bajen correo electrónico de un buzón remoto. ¿Esto significa que el formato interno de los buzones debe estandarizarse para que cualquier programa POP3 en el cliente pueda leer el buzón en cualquier servidor de correo? Explique su respuesta.

21. Desde el punto de vista de un ISP, POP3 e IMAP tienen diferencias importantes. Por lo general, los usuarios de POP3 vacían sus buzones todos los días. Los usuarios de IMAP mantienen su correo electrónico en el servidor de manera indefinida. Imagine que se le pide a usted que aconseje a un ISP sobre cuáles protocolos debe soportar. ¿Qué aspectos tomaría en cuenta?

22. ¿Webmail utiliza POP3, IMAP o ninguno? Si utiliza alguno de éstos, ¿por qué se eligió? Si no se utiliza ninguno, ¿cuál está más cerca de ser usado?

23. Cuando se envían las páginas Web, se les anteponen encabezados MIME. ¿Por qué?

24. ¿Cuándo son necesarios los visores externos? ¿Cómo sabe un navegador cuál utilizar?

25. ¿Es posible que cuando un usuario haga clic en un vínculo con Netscape se inicie una aplicación auxiliar en particular, y que cuando haga clic en el mismo vínculo en Internet Explorer se inicie una aplicación auxiliar completamente diferente, aunque el tipo MIME regresado en ambos casos sea idéntico? Explique su respuesta.

26. Un servidor Web de múltiples subprocesos está organizado como se muestra en la figura 7-21. Tarda 500 μseg en aceptar una solicitud y verificar el caché. La mitad del tiempo es para encontrar el archivo en el caché y para regresarlo de inmediato. En la otra mitad del tiempo, el módulo tiene que bloquearse por 9 mseg mientras su solicitud de disco se coloca en la cola y se procesa. ¿Cuántos módulos debe tener el servidor para mantener ocupada todo el tiempo a la CPU (suponiendo que el disco no es un cuello de botella)?

27. El URL *http* estándar da por hecho que el servidor Web está escuchando en el puerto 80. Sin embargo, es posible que un servidor Web escuche en otro puerto. Diseñe una sintaxis razonable para que un URL acceda a un archivo en un puerto no estándar.

28. Aunque no se mencionó en el texto, una forma alternativa de un URL es utilizar una dirección IP en lugar de su nombre DNS. Un ejemplo del uso de una dirección IP es *http://192.31.231.66/index.html*. ¿Cómo sabe el navegador si el nombre que sigue al esquema es un nombre DNS o una dirección IP?

29. Imagine que alguien del Departamento de Computación de Stanford acaba de escribir un nuevo programa que desea distribuir mediante FTP. Esa persona coloca el programa en el directorio *ftp/pub/freebies/newprog.c* de FTP. ¿Cuál será el URL más probable de este programa?

30. En la figura 7-25, *www.aportal.com* mantiene un registro de las preferencias del usuario en una *cookie*. Una desventaja de este esquema es que las *cookies* están limitadas a sólo 4 KB, de manera que si las preferencias son grandes —por ejemplo, muchas acciones, equipos de deportes, tipos de noticias, el clima

de varias ciudades, ediciones especiales de varias categorías de productos, etcétera— puede alcanzarse el límite de 4 KB. Diseñe una forma alternativa para mantener el registro de las preferencias que no tengan este problema.

31. El Banco Sloth desea que sus clientes flojos puedan utilizar con facilidad su banca en línea, por lo que después de que un cliente firma y se autentica mediante una contraseña, el banco regresa una *cookie* que contiene un número de ID del cliente. De esta forma, el cliente no tiene que identificarse a sí mismo o escribir una contraseña en visitas futuras a la banca en línea. ¿Qué opina de esta idea? ¿Funcionará? ¿Es una buena idea?

32. En la figura 7-26, el parámetro *ALT* se establece en la etiqueta <img>. ¿Bajo qué condiciones lo utiliza el navegador, y cómo?

33. ¿Cómo utiliza HTML a fin de que se pueda hacer clic en una imagen? Dé un ejemplo.

34. Muestre la etiqueta <a> que se necesita para hacer que la cadena "ACM" sea un hipervínculo a *http://www.acm.org*.

35. Diseñe un formulario para que la nueva compañía Interburger permita ordenar hamburguesas a través de Internet. Dicho formulario debe incluir el nombre, la dirección y la ciudad del cliente, así como opciones que permitan seleccionar el tamaño (gigante o inmensa) y si llevará queso. Las hamburguesas se pagarán en efectivo a la entrega por lo que no se necesita información de tarjeta de crédito.

36. Diseñe un formulario que pida al usuario que teclee dos números. Cuando el usuario haga clic en el botón de envío, el servidor regresará la suma de dichos números. Escriba el servidor como una secuencia de comandos PHP.

37. Para cada una de las siguientes aplicaciones, indique si sería (1) posible y (2) mejor utilizar una secuencia de comandos PHP o una JavaScript y por qué.
(a) Desplegar un calendario por cada mes solicitado desde septiembre de 1752.
(b) Desplegar la agenda de vuelos de Ámsterdam a Nueva York.
(c) Graficar un polinomio a partir de coeficientes proporcionados por el usuario.

38. Escriba un programa en JavaScript que acepte un entero mayor que 2 e indique si es un número primo. Observe que JavaScript tiene instrucciones if y while con la misma sintaxis que C y Java. El operador de módulo es %. Si necesita la raíz cuadrada de *x*, utilice *Math.sqrt* (*x*).

39. Una página HTML es como sigue:

```
<html> <body>
<a href="www.info-source.com/welcome.html"> Haga clic aquí para obtener información </a>
</body> </html>
```

Si el usuario hace clic en el hipervínculo, se abre una conexión TCP y se envía una serie de líneas al servidor. Liste todas las líneas enviadas.

40. Es posible utilizar el encabezado *If-Modified-Since* para verificar si una página almacenada en caché aún es válida. Las solicitudes pueden realizarse para obtener páginas que contengan imágenes, sonido, vídeo, HTML, etcétera. ¿Cree que la efectividad de esta técnica es mejor o peor para imágenes JPEG en comparación con HTML? Piense con cuidado sobre lo que significa "efectividad" y explique su respuesta.

41. El día en que se celebra un evento deportivo importante, como el juego de campeonato de la NFL, muchas personas visitan el sitio Web oficial. ¿Es ésta una aglomeración instantánea en el mismo sentido que con las elecciones en Florida del 2000? ¿Por qué sí o por qué no?

42. ¿Tiene sentido que un solo ISP funcione como una CDN? De ser así, ¿cómo funcionaría? De lo contrario, ¿qué está mal con esa idea?

43. ¿Bajo qué condiciones es una mala idea utilizar una CDN?

44. Las terminales de los sistemas inalámbricos Web tienen un ancho de banda bajo, lo cual provoca que la codificación eficiente sea importante. Diseñe un esquema para transmitir texto en inglés de manera eficiente a través de un enlace inalámbrico a un dispositivo WAP. Puede asumir que la terminal tiene algunos megabytes de ROM y una CPU con potencia moderada. *Sugerencia:* piense en transmitir japonés, en el que cada símbolo es una palabra.

45. Un disco compacto contiene 650 MB de datos. ¿La compresión se utiliza para CDs de audio? Explique su respuesta.

46. En la figura 7-57(c) el ruido de cuantización ocurre debido al uso de muestras de 4 bits para representar nueve valores de señal. La primera muestra, en 0, es exacta, pero las siguientes no lo son. ¿Cuál es el porcentaje de error para las muestras en 1/32, 2/32 y 3/32 del periodo?

47. ¿Es posible utilizar un modelo psicoacústico para reducir el ancho de banda necesario para la telefonía de Internet? De ser así, ¿cuáles condiciones, si las hay, se tendrían que cumplir para que funcionara? De lo contrario, ¿por qué no es posible?

48. Un servidor de flujo continuo de audio tiene una distancia de una sola vía de 50 mseg con un reproductor de medios. Tiene una salida de 1 Mbps. Si el reproductor de medios tiene un búfer de 1 MB, ¿qué puede decir acerca de la posición de la marca de agua baja y de la alta?

49. El algoritmo de entrelazado de la figura 7-60 tiene la ventaja de que puede resolver la pérdida de un paquete ocasional sin introducir un hueco en la reproducción. Sin embargo, cuando se utiliza para la telefonía de Internet, también tiene una pequeña desventaja. ¿Cuál es?

50. ¿La voz sobre IP tiene los mismos problemas con los servidores de seguridad que el audio de flujo continuo? Explique su respuesta.

51. ¿Cuál es la tasa de bits para transmitir tramas de color sin comprimir de 800×600 píxeles con 8 bits/píxel a 40 tramas/seg?

52. ¿Un error de 1 bit en una trama MPEG puede dañar más que la trama en la que ocurrió el error? Explique su respuesta.

53. Considere un servidor de vídeo de 100,000 clientes, en donde cada cliente ve dos películas por mes. La mitad de las películas se proporcionan a las 8 p.m. ¿Cuántas películas tiene que transmitir a la vez el servidor durante este periodo? Si cada película requiere 4 Mbps, ¿cuántas conexiones OC-12 necesita el servidor para la red?

54. Suponga que la ley de Zipf se cumple para acceder un servidor de vídeo con 10,000 películas. Si el servidor mantiene las 1000 películas más populares en un disco magnético y las restantes 9000 en un disco óptico, dé una expresión para la fracción de todas las referencias que estarán en un disco magnético. Escriba un pequeño programa para evaluar esta expresión de manera numérica.

55. Algunos cibernautas han registrado nombres de dominio que son muy parecidos a algunos sitios corporativos, por ejemplo, *www.microsfot.com,* con algunas diferencias en la ortografía. Haga una lista de por lo menos cinco dominios de este tipo.

56. Muchas personas han registrado nombres DNS que consisten en *www.palabra.com* donde *palabra* es una palabra común. Para cada una de las siguientes categorías, liste cinco sitios Web y resuma brevemente lo que es (por ejemplo, *www.estomago.com* es un gastroenterólogo de Long Island). Las categorías son: animales, comidas, objetos domésticos y partes del cuerpo. Para la última categoría, por favor apéguese a las partes del cuerpo que se encuentran arriba de la cintura.

57. Diseñe sus propios emoji utilizando un mapa de bits de 12×12. Incluya novio, novia, profesor y político.

58. Escriba un servidor POP3 que acepte los siguientes comandos: *USER*, *PASS*, *LIST*, *RETR*, *DELE* y *QUIT*.

59. Rescriba el servidor de la figura 6-6 como un servidor real Web utilizando el comando *GET* para HTTP 1.1. También debe aceptar el mensaje *Host*. El servidor debe mantener un caché de archivos recientemente obtenidos del disco y atender solicitudes del caché cuando sea posible.

8

SEGURIDAD EN REDES

Durante las primeras décadas de su existencia, las redes de computadoras fueron usadas principalmente por investigadores universitarios para el envío de correo electrónico, y por empleados corporativos para compartir impresoras. En estas condiciones, la seguridad no recibió mucha atención. Pero ahora, cuando millones de ciudadanos comunes usan redes para sus transacciones bancarias, compras y declaraciones de impuestos, la seguridad de las redes aparece en el horizonte como un problema potencial de grandes proporciones. En las siguientes secciones estudiaremos la seguridad de las redes desde varios ángulos, señalaremos muchos peligros y estudiaremos varios algoritmos y protocolos para hacer más seguras las redes.

La seguridad es un tema amplio que cubre una multitud de pecados. En su forma más sencilla, la seguridad se ocupa de garantizar que los curiosos no puedan leer, o peor aún, modificar mensajes dirigidos a otros destinatarios. Tiene que ver con la gente que intenta acceder a servicios remotos no autorizados. También se ocupa de mecanismos para verificar que el mensaje supuestamente enviado por la autoridad fiscal que indica: "Pague el viernes o aténgase a las consecuencias" realmente venga de ella y no de la mafia. La seguridad también se ocupa del problema de la captura y reproducción de mensajes legítimos, y de la gente que intenta negar el envío de mensajes.

La mayoría de los problemas de seguridad son causados intencionalmente por gente maliciosa que intenta ganar algo o hacerle daño a alguien. En la figura 8-1 se muestran algunos de los tipos de transgresores más comunes. Debe quedar claro por esta lista que hacer segura una red comprende mucho más que simplemente mantener los programas libres de errores de programación. Implica ser más listo que adversarios a menudo inteligentes, dedicados y a veces bien financiados. Debe quedar claro también que las medidas para detener a los adversarios casuales tendrán poca eficacia

contra los adversarios serios. Los registros policiales muestran que la mayoría de los ataques no son cometidos por intrusos que interfieren una línea telefónica sino por miembros internos con resentimientos. En consecuencia, los sistemas de seguridad deben diseñarse tomando en cuenta este hecho.

Adversario	Objetivo
Estudiante	Divertirse husmeando el correo de la gente
Cracker	Probar el sistema de seguridad de alguien; robar datos
Representante de ventas	Indicar que representa a toda Europa, no sólo a Andorra
Hombre de negocios	Descubrir el plan estratégico de marketing de un competidor
Ex empleado	Vengarse por haber sido despedido
Contador	Estafar dinero a una compañía
Corredor de bolsa	Negar una promesa hecha a un cliente por correo electrónico
Timador	Robar números de tarjeta de crédito
Espía	Conocer la fuerza militar o los secretos industriales de un enemigo
Terrorista	Robar secretos de guerra bacteriológica

Figura 8-1. Algunos tipos de personas que causan problemas de seguridad, y por qué.

Los problemas de seguridad de las redes pueden dividirse en términos generales en cuatro áreas interrelacionadas: confidencialidad, autenticación, no repudio y control de integridad. La confidencialidad consiste en mantener la información fuera de las manos de usuarios no autorizados. Esto es lo que normalmente viene a la mente cuando la gente piensa en la seguridad de las redes. La autenticación se encarga de determinar con quién se está hablando antes de revelar información delicada o hacer un trato de negocios. El no repudio se encarga de las firmas: ¿cómo comprobar que su cliente realmente hizo un pedido electrónico por 10 millones de utensilios para zurdos a 89 centavos cada uno cuando él luego aduce que el precio era de 69 centavos? O tal vez argumente que él nunca realizó ningún pedido. Por último, ¿cómo puede asegurarse de que un mensaje recibido realmente fue enviado, y no algo que un adversario malicioso modificó en el camino o cocinó por su propia cuenta?

Todos estos temas (confidencialidad, autenticación, no repudio y control de integridad) son pertinentes también en los sistemas tradicionales, pero con algunas diferencias importantes. La confidencialidad y la integridad se logran usando correo certificado y poniendo bajo llave los documentos. El robo del tren del correo es más difícil de lo que era en los tiempos de Jesse James.

También, la gente puede por lo general distinguir entre un documento original en papel y una fotocopia, y con frecuencia esto es importante. Como prueba, haga una fotocopia de un cheque válido. Trate de cobrar el cheque original en su banco el lunes. Ahora trate de cobrar la fotocopia del cheque el martes. Observe la diferencia de comportamiento del banco. Con los cheques electrónicos, el original y la copia son idénticos. Es posible que los bancos tarden un poco en acostumbrarse a esto.

La gente autentica la identidad de otras personas al reconocer sus caras, voces y letra. Las pruebas de firmas se manejan mediante firmas en papel, sellos, etc. Generalmente puede detectarse la alteración de documentos con el auxilio de expertos en escritura, papel y tinta. Ninguna de estas opciones está disponible electrónicamente. Es obvio que se requieren otras soluciones.

Antes de adentrarnos en las soluciones, vale la pena dedicar un instante a considerar el lugar que corresponde a la seguridad de las redes en la pila de protocolos. Probablemente no es un solo lugar. Cada capa tiene algo que contribuir. En la capa física podemos protegernos contra la intervención de las líneas de transmisión encerrando éstas en tubos sellados que contengan gas a alta presión. Cualquier intento de hacer un agujero en el tubo liberará un poco de gas, con lo cual la presión disminuirá y se disparará una alarma. Algunos sistemas militares usan esta técnica.

En la capa de enlace de datos, los paquetes de una línea punto a punto pueden encriptarse cuando se envíen desde una máquina y desencriptarse cuando lleguen a otra. Los detalles pueden manejarse en la capa de enlace de datos, sin necesidad de que las capas superiores se enteren de ello. Sin embargo, esta solución se viene abajo cuando los paquetes tienen que atravesar varios enrutadores, puesto que los paquetes tienen que desencriptarse en cada enrutador, dentro del cual son vulnerables a posibles ataques. Además, no se contempla que algunas sesiones estén protegidas (por ejemplo, aquellas que comprenden compras en línea mediante tarjeta de crédito) y otras no. No obstante, la **encriptación de enlace** (*link encryption*), como se llama a este método, puede agregarse fácilmente a cualquier red y con frecuencia es útil.

En la capa de red pueden instalarse *firewalls* para mantener adentro (o afuera) a los paquetes. La seguridad de IP también funciona en esta capa.

En la capa de transporte pueden encriptarse conexiones enteras, de extremo a extremo, es decir, proceso a proceso. Para lograr una máxima seguridad, se requiere seguridad de extremo a extremo.

Por último, los asuntos como la autenticación de usuario y el no repudio sólo pueden manejarse en la capa de aplicación.

Puesto que la seguridad no encaja por completo en ninguna capa, no podía integrarse en ningún capítulo de este libro. Por lo tanto, merece su propio capítulo.

Si bien este capítulo es largo, técnico y esencial, por el momento también es casi irrelevante. Por ejemplo, está bien documentado que la mayoría de las fallas de seguridad en los bancos se debe a empleados incompetentes, procedimientos de seguridad deficientes o a fraudes internos, más que a delincuentes inteligentes que intervienen líneas telefónicas y decodifican los mensajes encriptados. Si una persona entrara a un banco con una tarjeta de cajero automático que hubiera encontrado en la calle y solicitara un nuevo PIN argumentando haber olvidado el suyo, y éste se le proporcionara en el acto (en aras de las buenas relaciones con el cliente), toda la criptografía del mundo no podría evitar el abuso. A este respecto, el libro de Ross Anderson es una verdadera revelación, debido a que documenta cientos de ejemplos de fallas de seguridad en numerosas industrias, casi todas ellas debidas a lo que cortésmente podría llamarse prácticas de negocios descuidadas o falta de atención a la seguridad (Anderson, 2001). Sin embargo, pensamos con optimismo que a medida que el comercio electrónico se difunda con mayor amplitud, en algún momento las compañías depurarán sus procedimientos operativos, eliminando estos vacíos y dándole la importancia debida a los aspectos técnicos de seguridad.

Casi toda la seguridad se basa en principios de criptografía, a excepción de la seguridad en la capa física. Por esta razón, comenzaremos nuestro estudio de la seguridad examinando con detalle la criptografía. En la sección 8.1 veremos algunos de los principios básicos. En las secciones 8-2 a 8-5 examinaremos algunos de los algoritmos y estructuras de datos fundamentales que se utilizan en criptografía. A continuación examinaremos con detalle la forma en que pueden utilizarse estos conceptos para lograr la seguridad en la red. Finalizaremos con algunas ideas acerca de la tecnología y la sociedad.

Antes de comenzar, una última aclaración: qué es lo que no se cubre. Hemos tratado de enfocarnos en los aspectos de conectividad más que en los de sistema operativo o de aplicación, aunque la línea es muy sutil de distinguir. Por ejemplo, no hay nada aquí sobre la autenticación de usuario que utilice biométrica, seguridad mediante contraseña, ataques de sobreflujo de búfer, caballos de Troya, falsificación de inicio de sesión, bombas lógicas, virus, gusanos y cosas por el estilo. Todos estos temas se cubren con detalle en el capítulo 9 del libro *Sistemas Operativos Modernos* (Tanenbaum, 2003). El lector interesado en los aspectos de seguridad debe leer ese libro. Ahora comencemos nuestro viaje.

8.1 CRIPTOGRAFÍA

Criptografía viene del griego y significa "escritura secreta". Tiene una larga y colorida historia que se remonta a miles de años. En esta sección sólo delinearemos algunos de los puntos de interés, como antecedentes de lo que sigue. Si desea la historia completa de la criptografía, le recomiendo que lea el libro de Kahn (1995). Para un tratamiento más completo de los avances en lo que se refiere a los algoritmos de seguridad y criptografía, a los protocolos y a las aplicaciones, vea (Kaufman y cols., 2002). Para un enfoque más matemático, vea (Stinson, 2002). Para un enfoque menos matemático, vea (Burnett y Paine, 2001).

Los profesionales hacen una distinción entre cifrados y códigos. Un **cifrado** es una transformación carácter por carácter o bit por bit, sin importar la estructura lingüística del mensaje. En contraste, un **código** reemplaza una palabra con otra palabra o símbolo. Los códigos ya no se utilizan, aunque tienen una historia gloriosa. El código con mayor éxito fue el de las fuerzas armadas de Estados Unidos durante la Segunda Guerra Mundial en el Pacífico. Tenían indios navajos hablando entre sí, utilizando palabras específicas en navajo correspondientes a términos militares; por ejemplo, *chay-da-gahi-nail-tsaidi* (literalmente: asesino de tortugas) quiere decir arma antitanque. El lenguaje navajo es altamente tonal, extremadamente complejo y no tiene forma escrita. Ninguna persona en Japón sabía algo sobre él.

En septiembre de 1945, el *San Diego Union* describió el código diciendo "Durante tres años, en cualquier lugar donde los soldados de la Marina desembarcaban, los japoneses escuchaban una combinación de extraños gorjeos entremezclados con otros ruidos que se asemejaban al canto de un monje tibetano y al sonido de una botella de agua caliente al vaciarse". Los japoneses nunca descifraron el código y muchos de los soldados que utilizaban las claves navajo fueron premiados

con altos honores militares por su extraordinario servicio y valor. El hecho de que Estados Unidos descifrara el código japonés y que Japón no haya podido descifrar el navajo tuvo un papel importante en las victorias estadounidenses en el Pacífico.

8.1.1 Introducción a la criptografía

Históricamente, cuatros grupos de personas han utilizado y contribuido al arte de la criptografía: los militares, el cuerpo diplomático, los redactores de los periódicos y los amantes. De éstos, la milicia ha tenido el papel más importante y ha moldeado el campo a través de los siglos. Dentro de las organizaciones militares, los mensajes por encriptar se han entregado tradicionalmente a empleados mal pagados para su codificación y transmisión. El inmenso volumen de los mensajes ha impedido asignar esta labor a unos cuantos especialistas.

Hasta la llegada de las computadoras, una de las principales restricciones de la criptografía había sido la capacidad del empleado encargado de la codificación para realizar las transformaciones necesarias, con frecuencia en un campo de batalla con poco equipo. Una restricción adicional ha sido la dificultad de cambiar rápidamente de un método de criptografía a otro, debido a que esto implica volver a capacitar a una gran cantidad de personas. Sin embargo, el peligro de que un empleado fuera capturado por el enemigo ha hecho indispensable la capacidad de cambiar el método de criptografía de manera instantánea, de ser necesario. Estos requerimientos en conflicto han dado lugar al modelo de la figura 8-2.

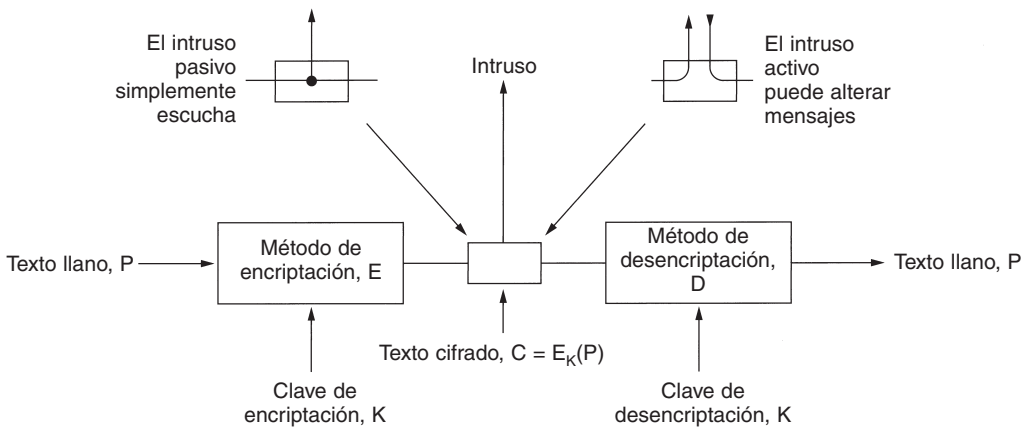


Figura 8-2. El modelo de encriptación (para un cifrado de clave simétrica).

Los mensajes por encriptar, conocidos como **texto llano,** son transformados por una función parametrizada por una **clave.** El resultado del proceso de encriptación, conocido como **texto cifrado,** se transmite a continuación, muchas veces por mensajero o radio. Suponemos que el enemigo, o **intruso,** escucha y copia con exactitud todo el texto cifrado. Sin embargo, a diferencia del destinatario original, el intruso no conoce la clave de desencriptación y no puede desencriptar con

facilidad el texto cifrado. En ocasiones el intruso no sólo escucha el canal de comunicación (intruso pasivo) sino también puede registrar mensajes y reproducirlos posteriormente, inyectar sus propios mensajes y modificar los mensajes legítimos antes de que lleguen al destinatario (intruso activo). El arte de descifrar los mensajes (**criptoanálisis**) y el arte de crear los cifrados (criptografía), se conocen en conjunto como **criptología.**

A menudo resulta útil tener una notación para relacionar el texto llano, el texto cifrado y las claves. Utilizaremos $C = E_K(P)$ para indicar que la encriptación del texto llano P usando la clave K produce el texto cifrado C. Del mismo modo, $P = D_K(C)$ representa la desencriptación de C para obtener el texto llano nuevamente. Por lo tanto,

$$D_K(E_K(P)) = P$$

Esta notación sugiere que E y D son sólo funciones matemáticas, lo cual es cierto. El único truco es que ambas son funciones de dos parámetros, y hemos escrito uno de los parámetros (la clave) como subíndice, en lugar de como argumento, para distinguirlo del mensaje.

Una regla fundamental de la criptografía es que se debe suponer que el criptoanalista conoce el método general de encriptación y desencriptación usado. En otras palabras, el criptoanalista sabe con detalle cómo funciona el método de encriptación, E, y desencriptación, D, de la figura 8-2. La cantidad de esfuerzo necesario para inventar, probar e instalar un nuevo algoritmo cada vez que el método antiguo está en peligro, o se piensa que lo está, siempre ha hecho impráctico mantener en secreto el algoritmo de encriptación. Además, pensar que dicho algoritmo es secreto cuando no lo es, hace más daño que bien.

Aquí es donde entra la clave. Ésta consiste en una cadena corta (relativamente) que selecciona una de muchas encriptaciones potenciales. En contraste con el método general, que tal vez se cambie cada cierto número de años, la clave puede cambiarse con la frecuencia requerida. Por lo tanto, nuestro modelo básico es un método general estable y conocido públicamente pero parametrizado por una clave secreta y que puede cambiarse con facilidad. La idea de que el criptoanalista conozca los algoritmos y que la naturaleza secreta se base principalmente en las claves se conoce como **principio de Kerckhoff,** que debe su nombre al criptógrafo militar holandés Auguste Kerckhoff, que lo estableció en 1883 (Kerckhoff, 1883). Por lo tanto, tenemos:

Principio de Kerckhoff: Todos los algoritmos deben ser públicos; sólo las claves deben ser secretas

La naturaleza no secreta del algoritmo no puede remarcarse lo suficiente. Tratar de mantener secreto el algoritmo, lo que se conoce como **seguridad por desconocimiento,** nunca funciona. Además, al hacer público el algoritmo, el criptógrafo recibe asesoría gratuita de una gran cantidad de criptólogos académicos ansiosos por descifrar el sistema con el propósito de publicar trabajos que demuestren su inteligencia. Si muchos expertos han tratado de descifrar el algoritmo durante cinco años después de su publicación y nadie lo ha logrado, probablemente es bastante sólido.

Puesto que la parte secreta debe ser la clave, la longitud de ésta es un aspecto importante del diseño. Considere una cerradura de combinación. El principio general es que se introducen dígitos

en secuencia. Todo el mundo lo sabe, pero la clave es secreta. Una longitud de clave de dos dígitos significa que hay 100 posibilidades. Una clave de tres dígitos significa que hay 1000 posibilidades y una clave de seis dígitos de longitud significa un millón. Cuanto más grande sea la clave, mayor será el **factor de trabajo** que tendrá que enfrentar el criptoanalista. El factor de trabajo para descifrar el sistema mediante una búsqueda exhaustiva del espacio de clave crece exponencialmente con la longitud de la clave. El secreto radica en tener un algoritmo robusto (pero público) y una clave larga. Para evitar que su hermano menor lea su correo electrónico, las claves de 64 bits son suficientes. Para el uso comercial común, se requieren por lo menos 128 bits. Para mantener a raya a gobiernos poderosos se requieren claves de al menos 256 bits, y de preferencia mayores.

Desde el punto de vista del criptoanalista, el problema del criptoanálisis presenta tres variaciones principales. Cuando el criptoanalista cuenta con cierta cantidad de texto cifrado pero no tiene texto llano, enfrenta el problema de **sólo texto cifrado.** Los criptogramas que aparecen en la sección de acertijos de los diarios presentan este tipo de problema. Cuando el criptoanalista cuenta con texto cifrado y el texto llano correspondiente, se enfrenta al problema del **texto llano conocido.** Por último, cuando el criptoanalista tiene la capacidad de encriptar textos normales que él escoge, enfrenta el problema del **texto llano seleccionado.** Los criptogramas de los periódicos podrían descifrarse con facilidad si el criptoanalista pudiera hacer preguntas como: ¿cuál es la encriptación de ABCDEFGHIJKL?

Los principiantes en el campo de la criptografía con frecuencia suponen que, si un cifrado puede resistir el ataque de sólo texto cifrado, entonces es seguro. Esta suposición es muy ingenua. En muchos casos, el criptoanalista puede hacer una buena estimación de partes de texto llano. Por ejemplo, lo primero que muchas computadoras dicen cuando se les llama es "Indique su clave:". Equipado con algunos pares concordantes de texto llano-texto cifrado, el trabajo del criptoanalista se vuelve mucho más fácil. Para lograr seguridad, el criptógrafo debe ser conservador y asegurarse de que el sistema sea inviolable aun si su oponente puede encriptar cantidades arbitrarias de texto llano seleccionado.

Los métodos de encriptación han sido divididos históricamente en dos categorías: cifrados por sustitución y cifrados por transposición. A continuación reseñaremos cada uno de éstos como antecedente para la criptografía moderna.

8.1.2 Cifrados por sustitución

En un **cifrado por sustitución,** cada letra o grupo de letras se reemplazan por otra letra o grupo de letras para disfrazarla. Uno de los cifrados más viejos conocidos es el **cifrado de César,** atribuido a Julio César. En este método, *a* se vuelve *D*, *b* se vuelve *E*, *c* se vuelve *F*, ... , y *z* se vuelve *C*. Por ejemplo, *ataque* se vuelve *DWDTXH*. En los ejemplos, el texto llano se presentará en minúsculas y el texto cifrado en mayúsculas.

Una pequeña generalización del cifrado de César permite que el alfabeto de texto cifrado se desplace *k* letras, en lugar de siempre 3. En este caso, *k* se convierte en una clave del método general de alfabetos desplazados circularmente. El cifrado de César posiblemente engañó a Pompeyo, pero no ha engañado a nadie desde entonces.

La siguiente mejora es hacer que cada uno de los símbolos del texto llano, digamos las 26 letras del abecedario (no incluimos la *ñ*) inglés, tengan una correspondencia con alguna otra letra. Por ejemplo,

texto llano: a b c d e f g h i j k l m n o p q r s t u v w x y z
texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

El sistema general de sustitución de símbolo por símbolo se llama **sustitución monoalfabética,** siendo la clave la cadena de 26 letras correspondiente al alfabeto completo. Para la clave anterior, el texto llano *ataque* se transformaría en el texto cifrado *QZQJXT.*

A primera vista, esto podría parecer un sistema seguro, porque, aunque el criptoanalista conoce el sistema general (sustitución letra por letra), no sabe cuál de las 26! $\approx 4 \times 10^{26}$ claves posibles se está usando. En contraste con el cifrado de César, intentarlas todas no es una estrategia muy alentadora. Aun a 1 nseg por solución, una computadora tardaría 10^{10} años en probar todas las claves.

No obstante, si se cuenta con una cantidad aun pequeña de texto cifrado, puede descifrarse fácilmente. El ataque básico aprovecha las propiedades estadísticas de los lenguajes naturales. En inglés, por ejemplo, la *e* es la letra más común, seguida de *t, o, a, n, i,* etc. Las combinaciones de dos letras más comunes, o **digramas,** son *th, in, er, re* y *an.* Las combinaciones de tres letras más comunes, o **trigramas,** son *the, ing, and* e *ion.*

Un criptoanalista que intenta descifrar un cifrado monoalfabético comenzaría por contar la frecuencia relativa de todas las letras del texto cifrado. Entonces podría asignar tentativamente la más común a la letra *e* y la siguiente más común a la letra *t*. Vería entonces los trigramas para encontrar uno común de la forma *tXe,* lo que sugerirá fuertemente que *X* es *h*. De la misma manera, si el patrón *thYt* ocurre con frecuencia, *Y* probablemente representa *a*. Con esta información se puede buscar un trigrama frecuente de la forma *aZW*, que con mucha probabilidad es *and*. Adivinando las palabras comunes, digramas y trigramas, y conociendo los patrones probables de las vocales y consonantes, el criptoanalista construye un texto llano tentativo, letra por letra.

Otra estrategia es adivinar una palabra o frase probable. Por ejemplo, considere el siguiente texto cifrado de una compañía contable (en bloques de cinco caracteres):

```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW
```

Una palabra muy probable en un mensaje de una compañía contable de un país de habla inglesa es *financial*. Usando nuestro conocimiento de que *financial* tiene una letra repetida (*i*), con cuatro letras intermedias entre su aparición, buscamos letras repetidas en el texto cifrado con este espaciado. Encontramos 12 casos, en las posiciones 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 y 82. Sin embargo, sólo dos de éstos, el 31 y 42, tienen la siguiente letra (correspondiente a *n* en el texto llano) repetida en el lugar adecuado. De estos dos, sólo el 31 tiene también la *a* en la posición

correcta, por lo que sabemos que *financial* comienza en la posición 30. A partir de aquí, la deducción de la clave es fácil usando las estadísticas de frecuencia del texto en inglés.

8.1.3 Cifrados por transposición

Los cifrados por sustitución conservan el orden de los símbolos de texto llano, pero los disfrazan. Los **cifrados por transposición,** en contraste, reordenan las letras pero no las disfrazan. En la figura 8-3 se presenta un cifrado de transposición común, la transposición columnar. La clave del cifrado es una palabra o frase que no contiene letras repetidas. En este ejemplo, la clave es MEGABUCK. El propósito de la clave es numerar las columnas, estando la columna 1 bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente. El texto llano se escribe horizontalmente, en filas, las cuales se rellenan para completar la matriz si es necesario. El texto cifrado se lee por columnas, comenzando por la columna cuya letra clave es la más baja.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Texto llano

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Texto cifrado

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Figura 8-3. Cifrado por transposición.

Para descifrar un cifrado por transposición, el criptoanalista debe primero estar consciente de que está tratando con un cifrado de este tipo. Observando la frecuencia de *E, T, A, O, I, N,* etc., es fácil ver si se ajustan al patrón usual del texto llano. De ser así, es evidente que se trata de un cifrado por transposición, pues en tal cifrado cada letra se representa a sí misma y la distribución de frecuencia permanece intacta.

El siguiente paso es adivinar la cantidad de columnas. En muchos casos, puede adivinarse una palabra o frase probable por el contexto del mensaje. Por ejemplo, supóngase que nuestro criptoanalista sospecha que la frase de texto llano *milliondollars* aparece en algún lugar del mensaje. Observe que los digramas *MO, IL, LL, LA, IR* y *OS* aparecen en el texto cifrado como resultado de la envoltura de la frase. La letra de texto cifrado *O* sigue a la letra de texto cifrado *M* (es decir, son

adyacentes verticalmente en la columna 4) puesto que están separados en la frase probable por una distancia igual a la longitud de la clave. Si se hubiera usado una clave de longitud siete, habrían aparecido los digramas *MD, IO, LL, LL, IA, OR* y *NS*. De hecho, para cada longitud de clave, se produce un grupo diferente de digramas en el texto cifrado. Buscando las diferentes posibilidades, el criptoanalista con frecuencia puede determinar fácilmente la longitud de la clave.

El paso restante es ordenar las columnas. Cuando la cantidad de columnas, k, es pequeña, puede examinarse cada uno de los pares de columnas $k(k-1)$ para ver si la frecuencia de sus digramas es igual a la del texto llano. El par con la mejor concordancia se supone correctamente ubicado. Ahora cada columna restante se prueba tentativamente como sucesora de este par. La columna cuyas frecuencias de digramas y trigramas produce la mejor concordancia se toma tentativamente como correcta. La columna antecesora se encuentra de la misma manera. El proceso completo se repite hasta encontrar un orden potencial. Es probable que en este punto se pueda reconocer texto llano (por ejemplo, si aparece *milloin*, quedará claro en dónde está el error).

Algunos cifrados por transposición aceptan un bloque de longitud fija como entrada y producen un bloque de longitud fija como salida. Estos cifrados pueden describirse por completo con sólo dar una lista que indique el orden en el que deben salir los caracteres. Por ejemplo, el cifrado de la figura 8-3 puede verse como un cifrado de bloque de 64 caracteres. Su salida es 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. En otras palabras, el cuarto carácter de entrada, *a,* es el primero en salir, seguido del decimosegundo, *f*, etcétera.

8.1.4 Rellenos de una sola vez

La construcción de un cifrado inviolable en realidad es bastante sencilla; la técnica se conoce desde hace décadas. Primero se escoge una cadena de bits al azar como clave. Luego se convierte el texto llano en una cadena de bits, por ejemplo usando su representación ASCII. Por último, se calcula el OR EXCLUSIVO de estas dos cadenas, bit por bit. El texto cifrado resultante no puede descifrarse, porque en una muestra suficientemente grande de texto cifrado cada letra aparecerá con la misma frecuencia, lo mismo que cada digrama y trigrama, y así sucesivamente. Este método, conocido como **relleno de una sola vez,** es inmune a todos los ataques actuales y futuros sin importar cuánta potencia computacional tenga el intruso. La razón se deriva de una teoría de la información: simplemente no hay información en el mensaje debido a que todos los textos llanos posibles de una longitud dada son parecidos.

En la figura 8-4 se muestra un ejemplo de cómo se utilizan los rellenos de una sola vez. Primero, el mensaje 1, "I love you." se convierte a ASCII de 7 bits. A continuación se elige el relleno de una sola vez, relleno 1, y se efectúa un XOR con el mensaje para obtener el texto cifrado. Un criptoanalista podría probar todos los rellenos de una sola vez posibles para ver qué texto llano proviene de cada uno. Por ejemplo, podría probarse el relleno 2 de la figura, lo que resultaría en el texto llano 2, "Elvis lives", lo cual podría ser o no verosímil (un tema que está fuera del alcance de este libro). De hecho, para cada texto llano ASCII de 11 caracteres hay un relleno de una sola vez que lo genera. Eso es lo que queremos dar a entender cuando decimos que no hay

información en el texto cifrado: es posible obtener cualquier mensaje con la longitud correcta a partir de él.

Mensaje 1:	1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
Relleno 1:	1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
Texto cifrado:	0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101
Relleno 2:	1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
Texto llano 2:	1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

Figura 8-4. El uso de un relleno de una sola vez para cifrado y la posibilidad de obtener cualquier texto llano posible a partir del texto cifrado mediante el uso de un relleno diferente.

En teoría, los rellenos de una sola vez son excelentes, pero en la práctica tienen varias desventajas. Para comenzar, la clave no se puede memorizar, por lo que tanto el emisor como el receptor deben cargar una copia escrita con ellos. Si cualquiera de ellos corre el peligro de ser capturado, es obvio que las claves escritas no son una buena idea. Además, la cantidad total de datos que se pueden transmitir está limitada por la cantidad de claves disponibles. Si el espía tiene suerte y descubre una gran cantidad de datos, quizá no los pueda transmitir al cuartel general debido a que la clave se ha agotado. Otro problema es la sensibilidad del método a los caracteres perdidos o insertados. Si el emisor y el receptor pierden la sincronización, de ahí en adelante todos los datos aparecerán distorsionados.

Con la aparición de las computadoras, el relleno de una sola vez podría volverse práctico para algunas aplicaciones. El origen de la clave podría ser un DVD especial que contenga varios gigabytes de información y si se transporta en una caja de película para DVD y se antecede con algunos minutos de vídeo, no sería sospechoso. Por supuesto, a velocidades de red de gigabit, tener que insertar un nuevo DVD cada 30 seg podría volverse tedioso. Y los DVDs deben transportarse personalmente desde el emisor hasta el receptor antes de que cualquier mensaje pueda ser enviado, lo que reduce en gran medida su utilidad práctica

Criptografía cuántica

Podría haber una solución al problema de la transmisión del relleno de una sola vez a través de la red, y proviene de un origen muy inverosímil: la mecánica cuántica. Esta área aún es experimental, pero las pruebas iniciales son alentadoras. Si pudiera perfeccionarse y fuera eficiente, con el tiempo casi toda la criptografía se realizará utilizando rellenos de una sola vez debido a su seguridad. A continuación explicaremos brevemente cómo funciona el método de la **criptografía cuántica.** En particular, describiremos un protocolo llamado **BB84** que debe su nombre a sus autores y a su año de publicación (Bennet y Brassard, 1984).

Un usuario, Alice, desea establecer un relleno de una sola vez con un segundo usuario, Bob. Alice y Bob son los **personajes principales.** Por ejemplo, Bob es un banquero con quien Alice quiere hacer negocios. En la década pasada, los nombres “Alice” y “Bob” se utilizaron como

personajes principales en casi todo lo escrito sobre criptografía. Los criptógrafos aman la tradición. Si utilizáramos "Andy" y "Barbara" como personajes principales, nadie creería nada de lo que se dice en este capítulo. Por lo tanto, dejémoslo así.

Si Alice y Bob pudieran establecer un relleno de una sola vez, podrían utilizarlo para comunicarse de manera segura. La pregunta es: ¿Cómo pueden establecerlo sin intercambiar antes DVDs? Podemos asumir que Alice y Bob se encuentran en extremos opuestos de un cable de fibra óptica a través del cual pueden enviar y recibir pulsos de luz. Sin embargo, un intruso intrépido, Trudy, puede cortar la fibra para establecer una derivación activa. Trudy puede leer todos los bits en ambas direcciones. También puede enviar mensajes falsos en ambas direcciones. Para Alice y Bob la situación podría parecer irremediable, pero la criptografía cuántica puede arrojarle un poco de luz.

La criptografía cuántica se basa en el hecho de que la luz viene en pequeños paquetes llamados **fotones,** los cuales tienen algunas propiedades peculiares. Además, la luz puede polarizarse al pasarla a través de un filtro de polarización, un hecho bien conocido por quienes utilizan anteojos oscuros y por los fotógrafos. Si se pasa un haz de luz (es decir, un flujo de fotones) a través de un filtro polarizador, todos los fotones que emerjan de él se polarizarán en la dirección del eje del filtro (por ejemplo, el vertical). Si a continuación el haz se pasa a través de un segundo filtro polarizador, la intensidad de la luz que emerja del segundo filtro es proporcional al cuadrado del coseno del ángulo entre los ejes. Si los dos ejes son perpendiculares, no pasa ningún fotón. La orientación absoluta de los dos filtros no importa; sólo cuenta el ángulo entre sus ejes.

Para generar un relleno de una sola vez, Alice necesita dos conjuntos de filtros polarizados. El primer conjunto consiste en un filtro vertical y en uno horizontal. Esta opción se conoce como **base rectilínea.** Una base es simplemente un sistema de coordenadas. El segundo conjunto de filtros consiste en lo mismo, excepto que se gira 45 grados, de forma que un filtro va del extremo inferior izquierdo al extremo superior derecho y el otro va del extremo superior izquierdo al extremo inferior derecho. Esta opción se conoce como **base diagonal.** Por lo tanto, Alice tiene dos bases, las cuales puede insertar a voluntad y rápidamente en su haz. En la realidad, Alice no tiene cuatro filtros separados, sino un cristal cuya polarización puede cambiarse de manera eléctrica y a gran velocidad a cualquiera de las cuatro direcciones permitidas. Bob tiene el mismo equipo que Alice. El hecho de que Alice y Bob dispongan de dos bases cada uno es esencial para la criptografía cuántica.

Ahora Alice asigna una dirección como 0 y la otra como 1 para cada base. En el ejemplo siguiente suponemos que elige 0 para vertical y 1 para horizontal. De manera independiente, también elige 0 para la dirección del extremo inferior izquierdo al superior derecho y 1 para la dirección del extremo superior izquierdo al inferior derecho. Alice envía estas opciones a Bob como texto llano.

Ahora Alice elige un relleno de una sola vez, por ejemplo, con base en un generador de números aleatorios (un tema complejo por sí mismo). Lo transmite bit por bit a Bob, eligiendo de manera aleatoria una de sus dos bases para cada bit. Para enviar un bit, su pistola de fotones emite un fotón polarizado apropiado para la base que está utilizando para ese bit. Por ejemplo, podría elegir bases diagonal, rectilínea, rectilínea, diagonal, rectilínea, etcétera. Para enviarle un relleno de una sola vez de 1001110010100110 con estas bases, Alice debería enviar los fotones que se muestran en la figura 8-5(a). Dados el relleno de una sola vez y la secuencia de bases, la polarización

por utilizar para cada bit se determina de manera única. Los bits enviados un fotón a la vez se conocen como **qubits.**

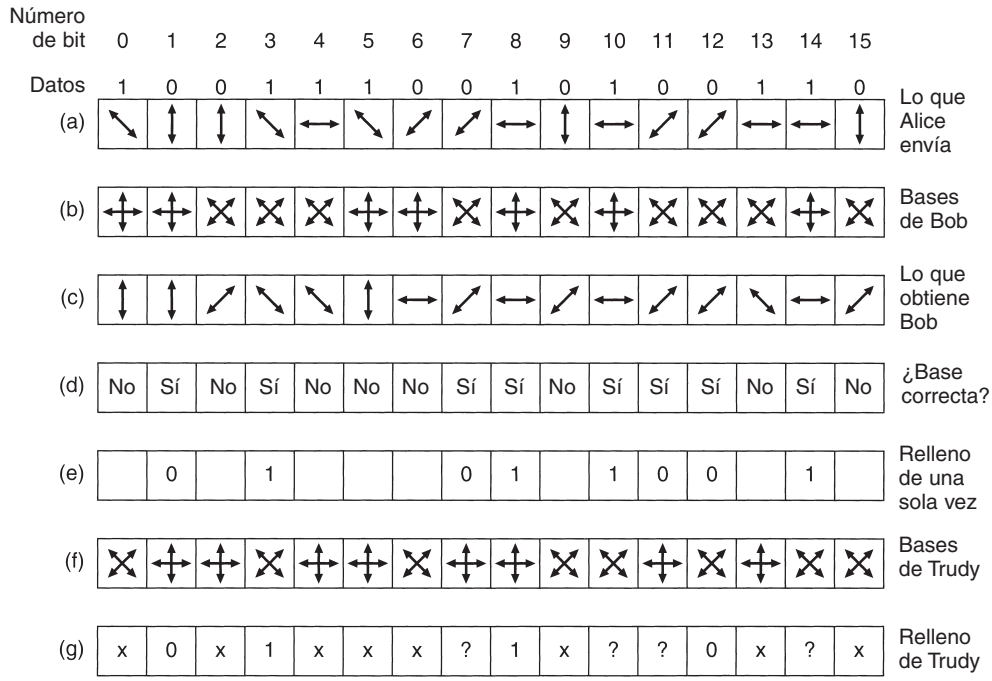


Figura 8-5. Ejemplo de criptografía cuántica.

Bob no sabe cuáles bases utilizar, por lo que elige una al azar para cada fotón entrante y simplemente la usa, como se muestra en la figura 8-5(b). Si elige la base correcta, obtiene el bit correcto. Si elige la base incorrecta, obtiene un bit aleatorio debido a que si un fotón choca con un filtro que está polarizado a 45 grados de su propia polarización, salta de manera aleatoria a la polarización del filtro o a una polarización perpendicular al filtro con igual probabilidad. Esta propiedad de los fotones es fundamental para la mecánica cuántica. Por lo tanto, algunos de los bits son correctos y algunos son aleatorios, pero Bob no sabe cuál es cuál. Los resultados de Bob se muestran en la figura 8-5(c).

¿Cómo sabe Bob cuáles de las bases que obtuvo son correctas y cuáles erróneas? Simplemente indica a Alice, mediante texto llano, cuál base utilizó para cada bit y ella le indica a él, también en texto llano, cuáles son correctas y cuáles erróneas, como se muestra en la figura 8-5(d). A partir de esta información los dos pueden construir una cadena de bits tomando en cuenta los aciertos, como se muestra en la figura 8-5(e). En promedio, esta cadena de bits tendrá la mitad del tamaño de la cadena de bits original, pero debido a que las dos partes lo saben, ambas pueden utilizarla como un relleno de una sola vez. Todo lo que Alice tiene que hacer es transmitir una cadena de bits un poco mayor que el doble de la longitud deseada y ella y Bob tendrán un relleno de una sola vez con la longitud deseada. Problema resuelto.

Pero espere un momento. Nos olvidamos de Trudy. Suponga que ella tiene curiosidad por saber lo que dirá Alice y corta la fibra, insertando sus propios detector y transmisor. Desgraciadamente para ella, tampoco sabe cuál base utilizar para cada fotón. Lo mejor que puede hacer es elegir una al azar para cada fotón, tal como lo hace Bob. En la figura 8-5(f) se muestra un ejemplo de sus elecciones. Cuando más tarde Bob informa (en texto llano) cuáles bases utilizó y Alice le indica (en texto llano) cuáles son correctas, Trudy sabe cuándo acertó y cuándo no. En la figura 8-5 acertó en los bits 0, 1, 2, 3, 4, 6, 8, 12 y 13. Pero sabe a partir de la respuesta de Alice en la figura 8-5(d) que sólo los bits 1, 3, 7, 8, 10, 11, 12 y 14 son parte del relleno de una sola vez. Acertó en cuatro de estos bits (1, 3, 8 y 12). En los otros cuatro (7, 10, 11 y 14) no coincidió y no conoce el bit transmitido. Por lo tanto, a partir de la figura 8-5(e), Bob sabe que el relleno de una sola vez inicia con 01011001, pero todo lo que Trudy tiene es 01?1??0?, tomando en cuenta la figura 8-5(g).

Por supuesto, Alice y Bob están conscientes de que tal vez Trudy haya capturado parte de su relleno de una sola vez, por lo que les gustaría reducir la información que Trudy tiene. Esto lo pueden conseguir realizando una transformación sobre el relleno. Por ejemplo, pueden dividir el relleno de una sola vez en bloques de 1024 bits y elevar al cuadrado cada uno de ellos para formar un número de 2048 bits y utilizar la concatenación de estos números de 2048 bits como el relleno de una sola vez. Con su conocimiento parcial de la cadena de bits transmitida, Trudy no tiene forma de elevar al cuadrado y, por lo tanto, no tiene nada. La transformación del relleno original de una sola vez a uno diferente que reduce el conocimiento de Trudy se conoce como **amplificación de privacidad.** En la práctica, en lugar de las elevaciones al cuadrado se utilizan transformaciones complejas en las que cada bit de salida depende de cada bit de entrada.

Pobre Trudy. No sólo no tiene idea de cuál es el relleno de una sola vez, sino que su presencia tampoco es un secreto. Después de todo, debe transmitir cada bit recibido a Bob para hacerle creer que está hablando con Alice. El problema es que lo mejor que puede hacer es transmitir el qubit que recibió, utilizando la polarización que utilizó para recibirlo, y casi la mitad del tiempo estará mal, causando muchos errores en el relleno de una sola vez de Bob.

Cuando Alice finalmente comienza a enviar datos, los codifica mediante un pesado código de corrección de errores hacia adelante. Desde el punto de vista de Bob, un error de un 1 bit en el relleno de una sola vez es lo mismo que un error de transmisión de 1 bit. De cualquier forma, obtiene el bit incorrecto. Si hay suficiente corrección de errores hacia adelante, puede recuperar el mensaje original a pesar de todos los errores, y puede contar fácilmente cuántos errores fueron corregidos. Si este número es mayor que la tasa de errores esperada del equipo, él sabe que Trudy ha intervenido la línea y puede actuar en consecuencia (por ejemplo, decirle a Alice que cambie a un canal de radio, que llame a la policía, etcétera). Si Trudy tuviera alguna forma de clonar un fotón a fin de contar con un fotón para inspeccionar y uno idéntico para enviar a Bob, podría evitar la detección, pero en la actualidad no se conoce ninguna forma de clonar perfectamente un fotón. Sin embargo, aunque Trudy pudiera clonar fotones, no se podría reducir el valor de la criptografía cuántica para establecer rellenos de una sola vez.

Aunque se ha mostrado que la criptografía cuántica puede operar a través de distancias de 60 km de fibra, el equipo es complejo y costoso. Aun así, la idea es prometedora. Para obtener mayor información sobre la criptografía cuántica, vea (Mullins, 2002).

8.1.5 Dos principios criptográficos fundamentales

Aunque estudiaremos muchos sistemas criptográficos diferentes en las siguientes páginas, hay dos principios que los sostienen a todos y que es importante entender.

Redundancia

El primer principio es que todos los mensajes encriptados deben contener redundancia, es decir, información no necesaria para entender el mensaje. Un ejemplo puede dejar en claro la necesidad de esto. Considere una compañía de compras por correo, El Perezoso (EP), que tiene 60,000 productos. Pensando que son muy eficientes, los programadores de EP deciden que los mensajes de pedidos deben consistir en un nombre de cliente de 16 bytes seguido de un campo de datos de 3 bytes (1 byte para la cantidad y 2 para el número de producto). Los últimos 3 bytes deben encriptarse usando una clave muy grande conocida sólo por el cliente y EP.

En un principio, esto podría parecer seguro, y en cierto sentido lo es, puesto que los intrusos pasivos no pueden descifrar los mensajes. Desgraciadamente, el sistema también tiene una falla mortal que lo vuelve inútil. Supongamos que un empleado recientemente despedido quiere vengarse de EP por haberlo cesado. Antes de irse, se lleva con él (parte de) la lista de clientes; entonces trabaja toda la noche escribiendo un programa para generar pedidos ficticios usando nombres reales de los clientes. Dado que no tiene la lista de claves, simplemente pone números aleatorios en los últimos 3 bytes y envía cientos de pedidos a EP.

Al llegar estos mensajes, la computadora de EP usa el nombre del cliente para localizar la clave y descifrar el mensaje. Para mala suerte de EP, casi cada mensaje de 3 bytes es válido, por lo que la computadora comienza a imprimir instrucciones de embarque. Aunque podría parecer extraño que un cliente ordene 837 juegos de columpios para niños, o 540 cajas de arena, en lo que a la computadora concierne el cliente bien podría estar planeando abrir una cadena de franquicias con juegos infantiles. De esta manera, un intruso activo (el ex empleado) puede causar muchísimos problemas, aun cuando no pueda entender los mensajes que genera su computadora.

Este problema puede resolverse agregando redundancia a todos los mensajes. Por ejemplo, si se extienden los mensajes de pedido a 12 bytes, de los cuales los primeros 9 deben ser ceros, entonces este ataque ya no funciona, porque el ex empleado ya no puede generar una cadena grande de mensajes válidos. La moraleja de esta historia es que todos los mensajes deben contener una cantidad considerable de redundancia para que los intrusos activos no puedan enviar basura al azar y lograr que se interprete como mensajes válidos.

Sin embargo, la adición de redundancia también simplifica a los criptoanalistas el descifrado de los mensajes. Supongamos que el negocio de pedidos por correo es altamente competido, y que la competencia principal de El Perezoso, El Bolsón, estaría encantado de conocer la cantidad de cajas de arena que EP vende. Para ello, ha intervenido la línea telefónica de EP. En el esquema original con mensajes de 3 bytes, el criptoanálisis era prácticamente imposible puesto que, tras adivinar una clave, el criptoanalista no tenía manera de saber si había adivinado correctamente. A fin de cuentas, casi cualquier mensaje era técnicamente legal. Con el nuevo esquema de 12 bytes,

es fácil para el criptoanalista distinguir un mensaje válido de uno no válido. Por lo tanto, tenemos:

Principio criptográfico 1: Los mensajes deben contener alguna redundancia

En otras palabras, al desencriptar un mensaje, el destinatario debe tener la capacidad de saber si es válido con sólo inspeccionarlo y tal vez realizando un cálculo simple. Esta redundancia es necesaria para evitar que intrusos activos envíen basura y engañen al receptor para que descifre la basura y realice algo con el "texto llano". Sin embargo, esta misma redundancia simplifica en gran medida la violación del sistema por parte de los intrusos pasivos, por lo que aquí hay un poco de preocupación. Más aún, la redundancia nunca debe estar en la forma de *n* ceros al principio o al final del mensaje, debido a que al ejecutar tales mensajes a través de algunos algoritmos criptográficos los resultados son más predecibles, lo que facilita el trabajo del criptoanalista. Un polinomio CRC es mejor que una serie de 0s puesto que el receptor puede verificarlo fácilmente, pero implica más trabajo para el criptoanalista. Un método aún mejor es utilizar un *hash* criptográfico, concepto que exploraremos más adelante.

Regresando a la criptografía cuántica por un momento, también podemos ver que la redundancia juega un papel ahí. Debido a que Trudy interceptó los fotones, algunos bits en el relleno de una sola vez de Bob serán incorrectos. Bob necesita algo de redundancia en los mensajes entrantes para determinar que se presentaron errores. Una forma muy burda de redundancia es repetir dos veces el mensaje. Si ambas copias no son idénticas, Bob sabe que o la fibra tiene mucho ruido o que alguien está interviniendo la transmisión. Por supuesto, enviar todo dos veces es excesivo; los códigos de Hamming o de Reed-Solomon constituyen formas más eficientes para realizar la detección y corrección de errores. Pero debe quedar claro que para distinguir un mensaje válido de uno inválido, especialmente en el caso de que haya un intruso activo, es necesaria cierta cantidad de redundancia.

Actualización

El segundo principio criptográfico es que se deben tomar medidas para asegurar que cada mensaje recibido se verifique a fin de saber si está actualizado, es decir, que se haya enviado muy recientemente. Esta medida es necesaria para evitar que intrusos activos reproduzcan mensajes antiguos. Si no se tomaran tales medidas, el ex empleado podría intervenir la línea telefónica de EP y simplemente continuar repitiendo los mensajes válidos enviados previamente. En otras palabras, tenemos:

Principio criptográfico 2: Es necesario algún método para frustrar los ataques de repetición

Una de tales medidas es incluir en cada mensaje una marca de tiempo válida durante, digamos, 10 segundos. El receptor puede entonces guardar los mensajes unos 10 segundos, para compararlos con los mensajes nuevos que lleguen y filtrar los duplicados. Los mensajes con una antigüedad mayor a 10 segundos pueden descartarse, dado que todas las repeticiones enviadas más de 10 segundos después también se rechazarán como demasiado viejas. Más adelante estudiaremos algunas medidas diferentes a las marcas de tiempo.

8.2 ALGORITMOS DE CLAVE SIMÉTRICA

La criptografía moderna usa las mismas ideas básicas que la criptografía tradicional (la transposición y la sustitución), pero su orientación es distinta. Tradicionalmente, los criptógrafos han usado algoritmos. Hoy día se hace lo opuesto: el objetivo es hacer el algoritmo de encriptación tan complicado y rebuscado que incluso si el criptoanalista obtiene cantidades enormes de texto cifrado a su gusto, no será capaz de entender nada sin la clave.

La primera clase de algoritmos de encriptación que analizaremos en este capítulo se conocen como **algoritmos de clave simétrica** porque utilizan la misma clave para encriptar y desencriptar. La figura 8-2 ilustra el uso de un algoritmo de clave simétrica. En particular, nos enfocaremos en los **cifrados en bloques,** que toman un bloque de *n* bits de texto llano como entrada y lo transforman utilizando la clave en un bloque de *n* bits de texto cifrado.

Los algoritmos criptográficos pueden implementarse ya sea en hardware (para velocidad) o en software (para flexibilidad). Aunque la mayoría de nuestro tratamiento tiene que ver con algoritmos y protocolos, que son independientes de la implementación real, no están de más unas cuantas palabras acerca de la construcción de hardware criptográfico. Las transposiciones y las sustituciones pueden implementarse mediante circuitos eléctricos sencillos. En la figura 8-6(a) se muestra un dispositivo, conocido como **caja P** (la P significa permutación), que se utiliza para efectuar una transposición de una entrada de 8 bits. Si los 8 bits se designan de arriba hacia abajo como 01234567, la salida de esta caja P en particular es 36071245. Mediante el cableado interno adecuado, puede hacerse que una caja P efectúe cualquier transposición prácticamente a la velocidad de la luz, pues no se requiere ningún cálculo, sólo propagación de la señal. Este diseño sigue el principio de Kerckhoff: el atacante sabe que el método general está permutando los bits. Lo que no sabe es qué bit va en qué lugar, y esto es la clave.

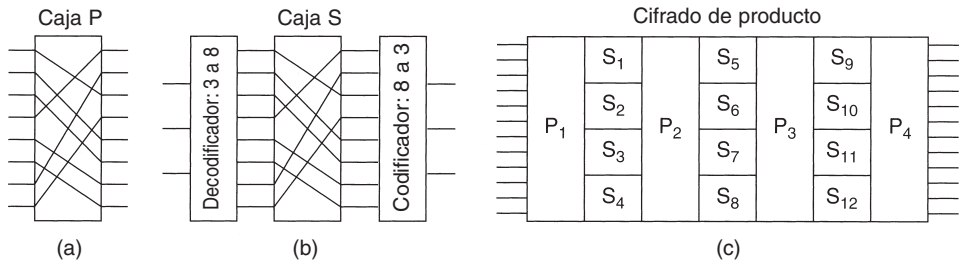


Figura 8-6. Elementos básicos del cifrado de producto. (a) Caja P. (b) Caja S. (c) Producto.

La sustitución se lleva a cabo mediante **cajas S,** como se muestra en la figura 8-6(b). En este ejemplo, se ingresa un texto llano de 3 bits y sale un texto cifrado de 3 bits. La entrada de 3 bits selecciona una de las ocho líneas de salida de la primera etapa y la establece en 1; las demás líneas son 0. La segunda etapa es una caja P. La tercera etapa codifica en binario nuevamente la línea de entrada seleccionada. Con el alambrado que se muestra, si los ocho números octales

01234567 entraran uno tras otro, la secuencia de salida sería de 24506713. En otras palabras, se ha reemplazado el 0 por el 2, el 1 por el 4, etcétera. Nuevamente, puede lograrse cualquier sustitución mediante el alambrado adecuado de la caja P dentro de la caja S. Además, tal dispositivo puede integrarse en hardware y puede alcanzarse una gran velocidad debido a que los codificadores y decodificadores sólo tienen uno o dos retardos de puerta lógica (subnanosegundos) y el tiempo de propagación a través de la caja P tal vez podría muy bien ser menor a 1 picosegundo.

La potencia real de estos elementos básicos sólo se hace aparente cuando ponemos en cascada una serie completa de cajas para formar un **cifrado de producto,** como se muestra en la figura 8-6(c). En este ejemplo, se trasponen (es decir, se permutan) 12 líneas de entrada en la primera etapa (P_1). En teoría, sería posible hacer que la segunda etapa fuera una caja S que relacionara un número de 12 bits con otro número de 12 bits. En vez de ello, tal dispositivo requeriría $2^{12} = 4096$ alambres cruzados en su etapa media. En cambio, la entrada se divide en cuatro grupos de 3 bits, cada uno de los cuales se sustituye independientemente de los demás. Aunque este método es menos general, aún es poderoso. Mediante la inclusión de una cantidad suficiente de etapas en el cifrado de producto, puede hacerse de la salida una función excesivamente complicada de la entrada.

Los cifrados de producto que operan en entradas de k bits para generar salidas de k bits son muy comunes. Por lo general, k es 64 a 256. Una implementación de hardware por lo general tiene por lo menos 18 etapas físicas, en lugar de sólo siete como en la figura 8-6(c). Una implementación de software se programa como un ciclo con por lo menos 8 iteraciones, cada una de las cuales realiza sustituciones de tipo caja S en subbloques del bloque de datos de 64 a 256 bits, seguido por una permutación que mezcla las salidas de las cajas S. Con frecuencia hay una permutación inicial especial y también una al final. En la literatura, las iteraciones se conocen como **rondas.**

8.2.1 DES—El Estándar de Encriptación de Datos

En enero de 1977, el gobierno de Estado Unidos adoptó un cifrado de producto desarrollado por IBM como su estándar oficial para información no secreta. Este cifrado, el **DES (Estándar de Encriptación de Datos),** se adoptó ampliamente en la industria para usarse con productos de seguridad. Ya no es seguro en su forma original, pero aún es útil en una forma modificada. Ahora explicaremos el funcionamiento del DES.

En la figura 8-7(a) se muestra un esbozo del DES. El texto llano se encripta en bloques de 64 bits, produciendo 64 bits de texto cifrado. El algoritmo, que se parametriza mediante una clave de 56 bits, tiene 19 etapas diferentes. La primera etapa es una transposición, independiente de la clave, del texto llano de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas restantes son funcionalmente idénticas, pero se parametrizan mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que la desencriptación se haga con la misma clave que la encriptación. Los pasos simplemente se ejecutan en el orden inverso.

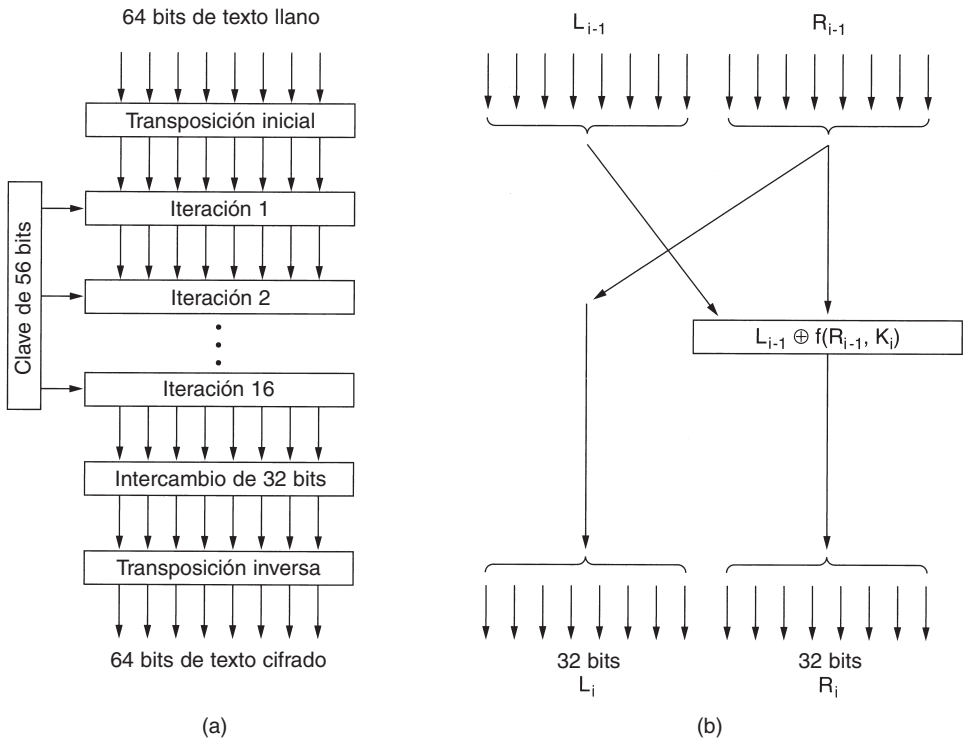


Figura 8-7. El estándar de encriptación de datos. (a) Esbozo general. (b) Detalle de una iteración. El signo de suma del círculo significa OR exclusivo.

La operación de una de estas etapas intermedias se ilustra en la figura 8-7(b). Cada etapa toma dos entradas de 32 bits y produce dos salidas de 32 bits. La salida de la izquierda simplemente es una copia de la entrada de la derecha. La salida de la derecha es el OR exclusivo a nivel de bits de la entrada izquierda y una función de la entrada derecha y la clave de esta etapa, K_i. Toda la complejidad reside en esta función.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, expandiendo el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Después, se aplica un OR exclusivo a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una caja S distinta. Cada una de las 64 entradas posibles a la caja S se mapea en una salida de 4 bits. Por último, estos 8×4 bits se pasan a través de una caja P.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de 28 bits, cada una de las cuales se gira hacia la izquierda una cantidad de bits dependiente del número de iteración. K_i se deriva de esta clave girada aplicándole otra transposición de 56 bits. En cada vuelta se extrae y permuta de los 56 bits un subgrupo de 48 bits diferente.

Una técnica que algunas veces se utiliza para hacer a DES más fuerte se conoce como **blanqueamiento** (*whitening*). Consiste en aplicar un OR exclusivo a una clave aleatoria de 64 bits con cada bloque de texto llano antes de alimentarla al DES y después aplicar un OR exclusivo a una segunda clave de 64 bits con el texto cifrado resultante antes de transmitirla. El blanqueamiento puede eliminarse fácilmente mediante la ejecución de las operaciones inversas (si el receptor tiene las dos claves de blanqueamiento). Puesto que esta técnica agrega más bits a la longitud de la clave, provoca que una búsqueda completa del espacio de claves consuma mucho más tiempo. Observe que se utiliza la misma clave de blanqueamiento para cada bloque (es decir, sólo hay una clave de blanqueamiento).

El DES se ha visto envuelto en controversias desde el día en que se lanzó. Se basó en un cifrado desarrollado y patentado por IBM, llamado Lucifer, excepto que dicho cifrado usaba una clave de 128 bits en lugar de una de 56 bits. Cuado el gobierno federal de Estados Unidos quiso estandarizar un método de cifrado para uso no confidencial, "invitó" a IBM a "debatir" el asunto con la NSA, la dependencia de descifrado de códigos del gobierno, que es el empleador de matemáticos y criptólogos más grande del mundo. La NSA es tan secreta que una broma de la industria reza:

Pregunta: ¿Qué significa NSA?
Respuesta: Ninguna Supuesta Agencia.

En realidad, NSA significa Agencia Nacional de Seguridad de Estados Unidos.

Tras estos debates, IBM redujo la clave de 128 a 56 bits y decidió mantener en secreto el proceso de diseño del DES. Mucha gente sospechó que la longitud de la clave se redujo para asegurar que la NSA pudiera descifrar el código, pero no ninguna organización de menor presupuesto. El objetivo del diseño secreto supuestamente era esconder una puerta trasera que pudiera facilitar aún más a la NSA el descifrado del DES. Cuando un empleado de la NSA pidió discretamente al IEEE que cancelara una conferencia planeada sobre criptografía, eso incomodó aún más a la gente. La NSA negó todo.

En 1977, dos investigadores de criptografía de Stanford, Diffie y Hellman (1977) diseñaron una máquina para violar el DES y estimaron que el costo de la construcción podría ascender a unos 20 millones de dólares. Dado un trozo pequeño de texto llano y el texto cifrado correspondiente, esta máquina podría encontrar la clave mediante una búsqueda exhaustiva del espacio de claves de 2^{56} entradas en menos de un día. Hoy día tal máquina costaría tal vez menos de un millón de dólares.

Triple DES

Ya en 1979, IBM se dio cuenta de que la longitud de la clave DES era muy corta y diseñó una forma de incrementarla de manera efectiva, utilizando cifrado triple (Tuchman, 1979). El método elegido, que desde entonces se ha incorporado en el Estándar Internacional 8732, se ilustra en la figura 8-8. Aquí se utilizan dos claves y tres etapas. En la primera etapa, el texto llano se encripta mediante DES de la forma usual con K_1. En la segunda etapa, DES se ejecuta en modo de desencriptación, utilizando K_2 como la clave. Por último, se realiza otra encriptación DES con K_1.

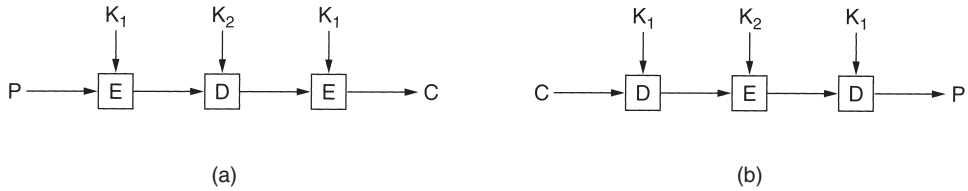


Figura 8-8. (a) Triple encriptación con DES. (b) Desencriptación.

Este diseño da lugar inmediatamente a dos preguntas. Primera, ¿por qué sólo se usan dos claves en lugar de tres? Segunda, ¿por qué se utiliza **EDE** (**Encriptar Desencriptar Encriptar**) en lugar de **EEE** (**Encriptar Encriptar Encriptar**)? La razón de que se usen dos claves es que incluso los criptógrafos más paranoicos coinciden en que por ahora 112 bits son suficientes para las aplicaciones comerciales. (Y entre los criptógrafos la paranoia se considera una ventaja, no un error.) Subir a 168 bits simplemente agregaría la sobrecarga innecesaria de administrar y transportar otra clave.

La razón para encriptar, desencriptar y luego encriptar de nuevo es la compatibilidad hacia atrás con los sistemas DES de una sola clave. Tanto las funciones de encriptación como de desencriptación son correspondencias entre grupos de números de 64 bits. Desde el punto de vista criptográfico, las dos correspondencias son igualmente robustas. Sin embargo, mediante el uso de EDE en lugar de EEE, una computadora que emplea triple encriptación puede comunicarse con otra que usa encriptación sencilla simplemente estableciendo $K_1 = K_2$. Esta propiedad permite la introducción gradual de la triple encriptación, algo que no interesa a los criptógrafos académicos, pero que es de importancia considerable para IBM y sus clientes.

8.2.2 AES—El Estándar de Encriptación Avanzada

Conforme el DES comenzó a acercarse al final de su vida útil, aun con el DES triple, el **NIST** (**Instituto Nacional de Estándares y Tecnología**), la agencia del Departamento de Comercio de Estados Unidos encargada de aprobar estándares del Gobierno Federal de Estados Unidos, decidió que el gobierno necesitaba un nuevo estándar criptográfico para uso no confidencial. El NIST estaba completamente consciente de toda la controversia alrededor del DES y sabía que si sólo anunciaba un nuevo estándar, todos los que tuvieran conocimiento sobre criptografía supondrían de manera automática que la NSA había construido una puerta trasera con el fin de leer todo lo que se encriptara con ella. Bajo estas condiciones, probablemente nadie utilizaría el estándar y con seguridad tendría una muerte silenciosa.

Por esto, el NIST adoptó una estrategia sorprendentemente diferente para una burocracia gubernamental: promovió un concurso. En enero de 1997, los investigadores de todo el mundo fueron invitados a emitir propuestas para un nuevo estándar, que se llamaría **AES** (**Estándar de Encriptación Avanzada**). Las reglas fueron:

1. El algoritmo debe ser un cifrado de bloques simétricos.
2. Todo el diseño debe ser público.
3. Deben soportarse las longitudes de claves de 128, 192 y 256 bits.
4. Deben ser posibles las implementaciones tanto de software como de hardware.
5. El algoritmo debe ser público o con licencia en términos no discriminatorios.

Se realizaron quince propuestas serias y se organizaron conferencias para presentarlas, en las cuales se alentó activamente a los asistentes a que encontraran errores en todas ellas. En agosto de 1998, el NIST seleccionó cinco finalistas con base en su seguridad, eficiencia, simplicidad, flexibilidad y requerimientos de memoria (importantes para los sistemas integrados). Se llevaron a cabo más conferencias y se tomaron más críticas. En la última conferencia se realizó una votación no obligatoria. Los finalistas y sus puntajes fueron los siguientes:

1. Rijndael (de Joan Daemen y Vincent Rijmen, 86 votos).
2. Serpent (de Ross Anderson, Eli Biham y Lars Knudsen, 59 votos).
3. Twofish (de un equipo encabezado por Bruce Schneier, 31 votos).
4. RC6 (de los Laboratorios RSA, 23 votos).
5. MARS (de IBM, 13 votos).

En octubre de 2000, el NIST anunció que él también votó por Rijndael, y en noviembre de 2001 Rijndael se volvió un estándar del gobierno de Estados Unidos publicado como FIPS 197 (Estándar Federal para el Procesamiento de Información). Debido a la extraordinaria apertura de la competencia, las propiedades técnicas de Rijndael y al hecho de que el equipo ganador estuvo compuesto por dos jóvenes criptógrafos belgas (quienes no es probable que hayan construido una puerta trasera sólo para complacer a la NSA), se espera que Rijndael se vuelva el estándar criptográfico dominante en el mundo por lo menos por una década. El nombre Rijndael se deriva de los apellidos de los autores: Rijmen + Daemen.

Rijndael soporta longitudes de clave y tamaños de bloque de 128 a 256 bits en pasos de 32 bits. Las longitudes de clave y de bloque pueden elegirse de manera independiente. Sin embargo, el AES especifica que el tamaño de bloque debe ser de 128 bits y la longitud de clave debe ser de 128, 192 o 256 bits. Es indudable que nadie utilizará claves de 192 bits, por lo que, de hecho, el AES tiene dos variantes: un bloque de 128 bits con clave de 128 bits y un bloque de 128 bits con clave de 256 bits.

En el tratamiento de nuestro siguiente algoritmo examinaremos sólo el caso 128/128 porque es probable que éste se vuelva la norma comercial. Una clave de 128 bits da un espacio de claves de $2^{128} \approx 3 \times 10^{38}$ claves. Incluso si la NSA se las arregla para construir una máquina con mil millones de procesadores paralelos, cada uno de los cuales es capaz de evaluar una clave por picosegundo, a tal máquina le llevaría 10^{10} años buscar en el espacio de claves. Para ese entonces el Sol

ya se habrá apagado, por lo que las personas existentes tendrán que leer los resultados a la luz de las velas.

Rijndael

Desde una perspectiva matemática, Rijndael se basa en la teoría de campos de Galois, la cual da algunas propiedades verificables de seguridad. Sin embargo, también puede verse como código C, sin meterse a las matemáticas.

Al igual que el DES, Rijndael utiliza sustitución y permutaciones, así como múltiples rondas. El número de rondas depende del tamaño de clave y del tamaño de bloque, y es de 10 para las claves de 128 bits con bloques de 128 bits y aumenta hasta 14 para la clave o el bloque más grande. Sin embargo, a diferencia del DES, todas las operaciones involucran bytes completos, para permitir implementaciones eficientes tanto en hardware como en software. En la figura 8-9 se muestra un esquema del código.

```
#define LENGTH 16                       /* # de bytes en el bloque de datos o en
                                           la clave */
#define NROWS 4                         /* número de filas en estado */
#define NCOLS 4                         /* número de columnas en estado */
#define ROUNDS 10                       /* número de iteraciones */
typedef unsigned char byte;             /* entero sin signo de 8 bits */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
 int r;                                 /* índice de ciclo */
 byte state[NROWS][NCOLS];              /* estado actual */
 struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];  /* claves de ronda */

 expand_key(key, rk);                   /* construye las claves de ronda */
 copy_plaintext_to_state(state, plaintext); /* inicia el estado actual */
 xor_roundkey_into_state(state, rk[0]); /* aplica OR exclusivo a la clave dentro
                                           del estado */

 for (r = 1; r <= ROUNDS; r++) {
     substitute(state);                 /* aplica la caja S a cada byte */
     rotate_rows(state);                /* gira la fila i en i bytes */
     if (r < ROUNDS) mix_columns(state); /* función de mezcla */
     xor_roundkey_into_state(state, rk[r]);  /* aplica OR exclusivo a la clave
                                                dentro del estado */
 }
 copy_state_to_ciphertext(ciphertext, state);    /* devuelve el resultado */
}
```

Figura 8-9. Un esquema de Rijndael.

La función *rijndael* tiene tres parámetros: *plaintext*, un arreglo de 16 bytes que contiene los datos de entrada; *ciphertext,* un arreglo de 16 bytes a donde se regresará la salida cifrada, y *key*, la clave de 16 bytes. Durante el cálculo, el estado actual de los datos se mantiene en un arreglo de

bytes, *state*, cuyo tamaño es *NROWS* × *NCOLS*. Para bloques de 128 bits, este arreglo es de 4 × 4 bytes. Con 16 bytes, se puede almacenar el bloque de datos completo de 128 bits.

El arreglo *state* se inicializa al texto llano y se modifica en cada paso en el cálculo. En algunos pasos, se realiza la sustitución de byte por byte. En otros, los bytes se permutan dentro del arreglo. También se utilizan otras transformaciones. Al final, el contenido de *state* se regresa como texto cifrado.

El código inicia expandiendo la clave en 11 arreglos del mismo tamaño que el estado. Éstos se almacenan en *rk*, que es un arreglo de estructuras, cada una de las cuales contiene un arreglo de estado. Uno de los arreglos se utilizará al inicio del cálculo y los otros 10 se utilizarán en 10 rondas, uno por ronda. El cálculo de las claves de ronda de la clave de encriptación es muy complicado para tratarlo aquí. Basta decir que las claves de ronda se producen mediante una rotación repetida y aplicando OR exclusivo a varios grupos de bits de clave. Para todos los detalles, vea (Daemen y Rijmen, 2002).

El siguiente paso es copiar el texto llano en el arreglo *state* a fin de poder procesarlo durante las rondas. Se copia en orden de columna, con los primeros cuatro bytes colocados en la columna 0, los siguientes cuatro bytes en la columna 1, y así sucesivamente. Las columnas y las filas se numeran comenzando en 0, aunque las rondas se numeren comenzando en 1. En la figura 8-10 se muestra la configuración inicial de los 12 arreglos de bytes de tamaño 4 × 4.

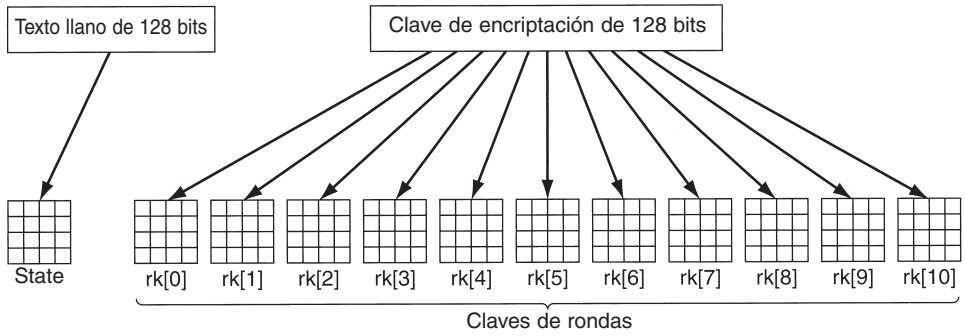


Figura 8-10. Creación de los arreglos *state* y *rk*.

Hay un paso adicional antes de que comience el cálculo principal: se aplica OR exclusivo a *rk*[0] dentro de *state*, byte por byte. En otras palabras, cada uno de los 16 bytes de *state* se reemplaza con el resultado del OR exclusivo de sí mismo y el byte correspondiente de *rk*[0].

Es el momento de presentar la atracción principal. El ciclo ejecuta 10 iteraciones, una por ronda, y transforma a *state* en cada iteración. El contenido de cada ronda consiste de cuatro pasos. El paso 1 realiza una sustitución de byte por byte sobre *state*. Cada byte se utiliza a la vez como un índice en una caja S para reemplazar su valor por el contenido de entrada de la caja S. Este paso es un cifrado de sustitución monoalfabética directo. A diferencia del DES, que tiene múltiples cajas S, Rijndael sólo tiene una caja S.

El paso 2 gira a la izquierda cada una de las cuatro filas. La fila 0 se gira 0 bytes (es decir, no cambia), la fila 1 se gira 1 byte, la fila 2 se gira 2 bytes y la fila 3 se gira 3 bytes. Este paso disemina por el bloque el contenido de los datos actuales, de forma análoga a las permutaciones de la figura 8-6.

El paso 3 mezcla cada una de las columnas independientemente de las demás. La mezcla se realiza utilizando multiplicación de matriz en la que cada nueva columna es el producto de la columna vieja (antes de la multiplicación) y una matriz constante, y la multiplicación se realiza mediante el campo finito de Galois, $GF(2^8)$. Aunque esto podría sonar complicado, existe un algoritmo que permite calcular cada elemento de la nueva columna mediante dos búsquedas de tabla y tres OR exclusivos (Daemen y Rijmen, 2002, apéndice E).

Por último, el paso 4 aplica OR exclusivo a la clave de esta ronda dentro del arreglo *state.*

Puesto que cada paso es reversible, la desencriptación se puede realizar con sólo ejecutar el algoritmo en sentido inverso. Sin embargo, también hay un truco mediante el cual la desencriptación se puede realizar ejecutando el algoritmo de encriptación y utilizando tablas diferentes.

El algoritmo se ha diseñado no sólo para una gran seguridad, sino también para una gran velocidad. Una buena implementación de software en una máquina de 2 GHz debe tener la capacidad de alcanzar una tasa de encriptación de 700 Mbps, que es lo suficientemente rápida para encriptar cerca de 100 vídeos MPEG-2 en tiempo real. Las implementaciones de hardware son aún más rápidas.

8.2.3 Modos de cifrado

A pesar de toda esta complejidad, el AES (o el DES o, de hecho, cualquier cifrado de bloques) es básicamente un cifrado de sustitución monoalfabética que utiliza caracteres grandes (caracteres de 128 bits para el AES y caracteres de 64 bits para el DES). Siempre que el mismo bloque de texto llano entra en la etapa inicial, el mismo bloque de texto cifrado sale de la etapa final. Si encripta 100 veces el texto llano *abcdefgh* con la misma clave DES, obtiene 100 veces el mismo texto cifrado. Un intruso puede aprovechar esta propiedad para violar el cifrado.

Modo de libro de código electrónico

Para ver cómo puede utilizarse esta propiedad de cifrado de sustitución monoalfabética para vencer parcialmente el cifrado, utilizaremos el (triple) DES porque es más fácil diseñar bloques de 64 bits que de 128 bits, pero el AES tiene exactamente el mismo problema. La forma directa de utilizar el DES para cifrar una pieza grande de texto llano es dividirla en bloques consecutivos de 8 bytes (64 bits) y encriptarlos después uno tras otro con la misma clave. La última pieza de texto llano se rellena a 64 bits, en caso de ser necesario. Esta técnica se conoce como **modo ECB** (**modo de Libro de Código Electrónico**) en analogía con los libros de código pasados de moda en los que se listaba cada palabra de texto llano, seguida por su texto cifrado (por lo general, un número decimal de cinco dígitos).

En la figura 8-11 se muestra el inicio de un archivo de computadora que lista los bonos anuales que una compañía ha decidido otorgar a sus empleados. Este archivo consiste en registros consecutivos de 32 bytes, uno por empleado, en el formato que se muestra: 16 bytes para el nombre,

8 bytes para el puesto y 8 bytes para el bono. Cada uno de los 16 bloques de 8 bytes (numerados del 0 al 15) se encripta mediante (triple) DES.

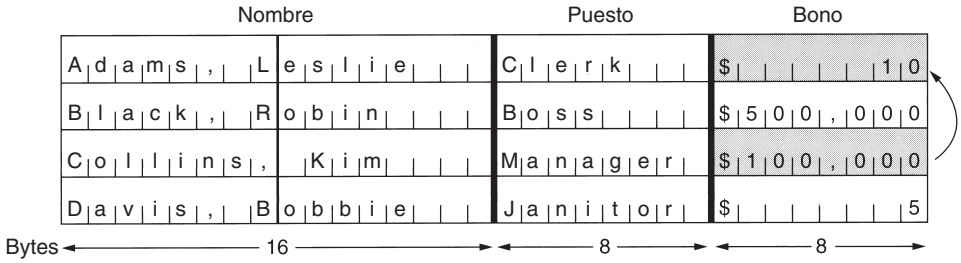


Figura 8-11. El texto llano de un archivo encriptado como 16 bloques DES.

Leslie tiene una pelea con el jefe y no espera un bono significativo. Kim, en contraste, es la favorita del jefe, y todos saben esto. Leslie puede obtener acceso al archivo después de que se encripta, pero antes de que se envíe al banco. ¿Leslie puede enmendar esta situación injusta con sólo contar con el archivo encriptado?

No hay problema. Todo lo que Leslie tiene que hacer es realizar una copia del doceavo bloque de texto cifrado (que contiene el bono de Kim) y utilizarlo para reemplazar el cuarto bloque de texto cifrado (que contiene el bono de Leslie). Incluso sin saber lo que dice el doceavo bloque, Leslie puede esperar contar con una Navidad mucho más feliz este año. (Copiar el octavo bloque de texto cifrado también es una posibilidad, pero es más probable de detectar; además, Leslie no es una persona avara.)

Modo de encadenamiento de bloques de cifrado

Para frustrar este tipo de ataque, todos los cifrados en bloques pueden encadenarse de varias formas a fin de que el reemplazo de un bloque de la forma en que lo hizo Leslie cause que el texto llano se desencripte comenzando en el bloque reemplazado que se desechará. Una forma de encadenar es el **encadenamiento de bloques de cifrado.** En este método, que se muestra en la figura 8-12, a cada bloque de texto llano se le aplica un OR exclusivo con el bloque anterior de texto cifrado antes de ser encriptado. En consecuencia, el mismo bloque de texto llano ya no corresponde con el mismo bloque de texto cifrado, y la encriptación deja de ser un enorme cifrado de sustitución monoalfabética. Al primer bloque se le aplica un OR exclusivo con un **IV** (**Vector de Inicialización**) elegido de manera aleatoria, que se transmite (en texto llano) junto con el texto cifrado.

En el ejemplo de la figura 8-12 podemos ver la forma en que funciona el modo de encadenamiento de bloques de cifrado. Iniciamos calculando $C_0 = E(P_0 \text{ XOR } IV)$. Después calculamos $C_1 = E(P_1 \text{ XOR } C_0)$, y así sucesivamente. La desencriptación también utiliza OR exclusivo para invertir el proceso, con $P_0 = IV \text{ XOR } D(C_0)$, etcétera. Observe que la encriptación del bloque i es una función de todo el texto llano de los bloques 0 a $i - 1$, por lo que el mismo texto llano genera texto cifrado diferente dependiendo de dónde ocurre. Una transformación del tipo que realizó Leslie no tendrá sentido para dos bloques que inician en el campo de bono de Leslie. Para un oficial de seguridad astuto, esta peculiaridad podría sugerir en dónde comenzar la investigación resultante.

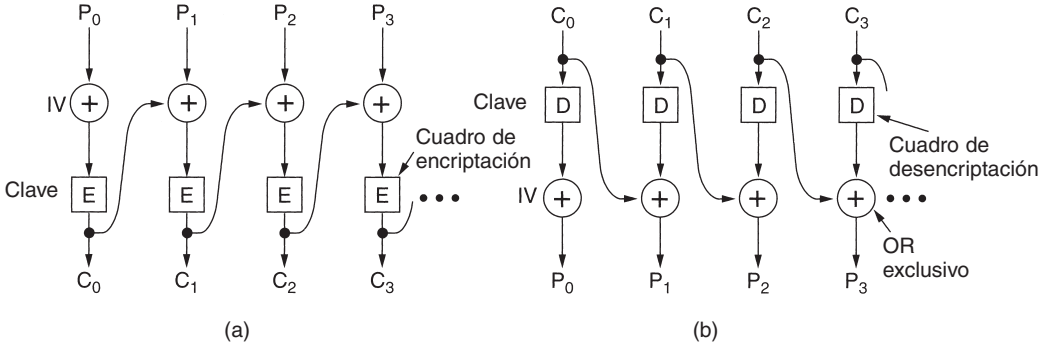


Figura 8-12. Encadenamiento de bloques de cifrado. (a) Encriptación. (b) Desencriptación.

El encadenamiento de bloques cifrado también tiene la ventaja de que el mismo bloque de texto llano no resultará en el mismo bloque de texto cifrado, lo que dificulta el criptoanálisis. De hecho, ésta es la razón principal por la que se utiliza.

Modo de retroalimentación de cifrado

Sin embargo, el encadenamiento de bloques tiene la desventaja de que requiere la llegada de un bloque completo de 64 bits antes de que pueda comenzar la desencriptación. Este modo no es adecuado para terminales interactivas, en las que se pueden escribir líneas máximo de ocho caracteres y es necesario detenerse a esperar una respuesta. Para la encriptación byte por byte, **modo de retroalimentación de cifrado,** se utiliza (triple) DES, como se muestra en la figura 8-13. Para el AES la idea es exactamente la misma; sólo se utiliza un registro de desplazamiento de 128 bits. En esta figura se muestra el estado de la máquina de encriptación después de que se han encriptado y enviado los bytes 0 a 9. Cuando llega el byte 10 de texto llano, como se ilustra en la figura 8-13(a), el algoritmo DES opera en el registro de desplazamiento de 64 bits para generar texto cifrado de 64 bits. El byte más a la izquierda de ese texto cifrado se extrae y se le aplica un OR exclusivo con P_{10}. Ese byte se transmite en la línea de transmisión. Además, el registro de desplazamiento se mueve a la izquierda 8 bits, causando que C_2 se caiga del extremo izquierdo y que C_{10} se inserte en la posición que C_9 acaba de dejar libre en el extremo derecho. Observe que el contenido del registro de desplazamiento depende de la historia completa anterior del texto llano, por lo que un patrón que se repite múltiples veces en el texto llano se encriptará de manera diferente cada vez en el texto cifrado. Al igual que con el encadenamiento de bloques de cifrado, se necesita un vector de inicialización para comenzar.

La desencriptación con el modo de retroalimentación de cifrado hace lo mismo que la encriptación. En particular, el contenido del registro de desplazamiento se *encripta*, no se *desencripta*, a fin de que el byte seleccionado al cual se le aplica el OR exclusivo con C_{10} para obtener P_{10} sea el mismo al que se le aplicó el OR exclusivo con P_{10} para generar C_{10} en primera instancia. Siempre y cuando los dos registros de desplazamiento permanezcan idénticos, la desencriptación funcionará de manera correcta. Esto se ilustra en la figura 8-13(b).

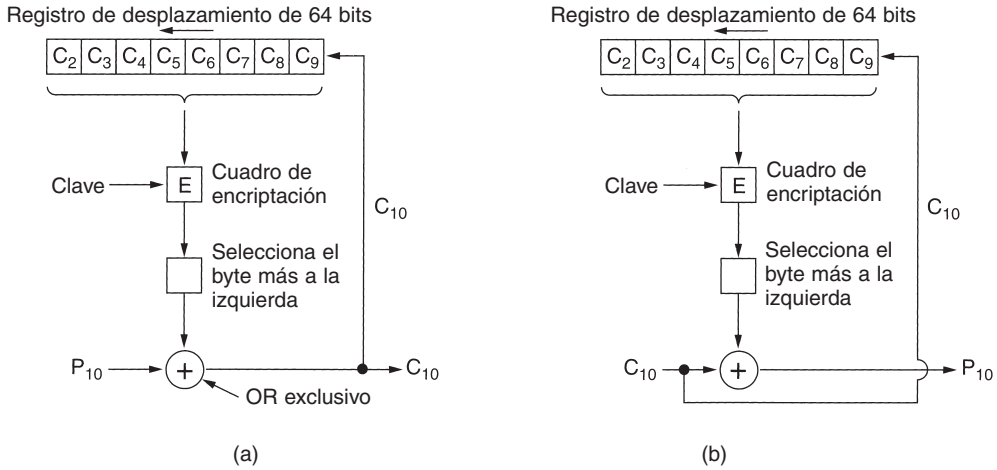


Figura 8-13. Modo de retroalimentación de cifrado. (a) Encriptación. (b) Desencriptación.

Un problema con el modo de retroalimentación de cifrado es que si un bit del texto cifrado se invierte de manera accidental durante la transmisión, se dañarán los 8 bytes que se desencriptan mientras el byte incorrecto se encuentra en el registro de desplazamiento. Una vez que el byte incorrecto se elimine del registro de desplazamiento, se generará nuevamente texto llano correcto. Por lo tanto, los efectos de un solo bit invertido se limitan relativamente a un sitio y no se daña el resto del mensaje, pero sí se dañan los bits que haya en el registro de desplazamiento.

Modo de cifrado de flujo

Sin embargo, existen aplicaciones en las que un error de transmisión de 1 bit que arruina 64 bits de texto llano es demasiado. Para estas aplicaciones, existe una cuarta opción, el **modo de cifrado de flujo.** Funciona encriptando un vector de inicialización y usando una clave para obtener un bloque de salida. A continuación se encripta este bloque usando la clave para obtener un segundo bloque de salida. A continuación este bloque se encripta para obtener un tercer bloque, y así sucesivamente. La secuencia (arbitrariamente grande) de bloques de salida, llamada **flujo de claves,** se trata como un relleno de una sola vez y se le aplica OR exclusivo con el texto llano para obtener el texto cifrado, como se muestra en la figura 8-14(a). Observe que el IV se utiliza sólo en el primer paso. Después de eso, se encripta la salida. Observe también que el flujo de claves es independiente de los datos, por lo que puede calcularse por adelantado, si es necesario, y es completamente insensible a los errores de transmisión. En la figura 8-14(b) se muestra la desencriptación.

La desencriptación se realiza generando el mismo flujo de claves en el lado receptor. Puesto que el flujo de claves depende sólo del IV y de la clave, no le afectan los errores de transmisión en el texto cifrado. Por lo tanto, un error de 1 bit en el texto cifrado transmitido genera sólo un error de 1 bit en el texto llano desencriptado.

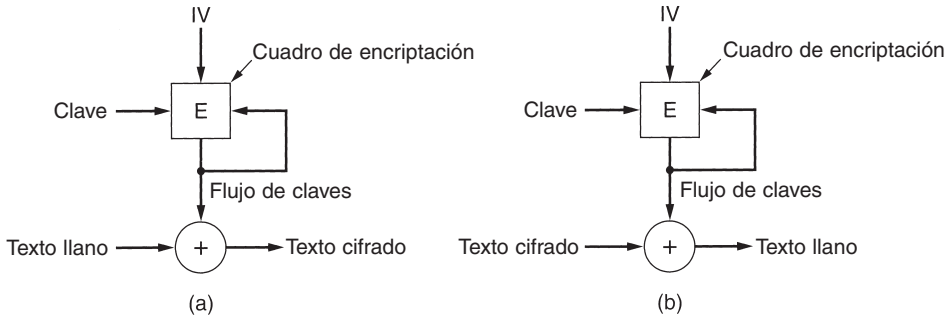


Figura 8-14. Un cifrado de flujo. (a) Encriptación. (b) Desencriptación.

Es esencial nunca utilizar dos veces el mismo par (clave, IV) con un cifrado de flujo, pues de hacerlo generará cada vez el mismo flujo de claves. Utilizar dos veces el mismo flujo de claves expone al texto cifrado a un **ataque de reutilización de flujo de claves.** Imagine que el bloque de texto llano, P_0, se encripta con el flujo de claves para obtener P_0 XOR K_0. Posteriormente, un segundo bloque de texto llano, Q_0, se encripta con el mismo flujo de claves para obtener Q_0 XOR K_0. Un intruso que capture estos dos bloques de texto cifrado puede sencillamente aplicarles un OR exclusivo en conjunto para obtener P_0 XOR Q_0, lo cual elimina la clave. Ahora el intruso tiene el OR exclusivo de los dos bloques de texto llano. Si se conoce alguno de ellos o es posible adivinarlo, el otro también se puede descubrir. En cualquier caso, el OR exclusivo de dos flujos de texto llano puede ser atacado utilizando propiedades estadísticas del mensaje. Por ejemplo, para texto en inglés, tal vez el carácter más común en el flujo será el OR exclusivo de dos espacios, seguido por el OR exclusivo de un espacio y la letra "e", etc. En resumen, al poseer el OR exclusivo de dos bloques de texto llano, el criptoanalista tiene una excelente oportunidad de adivinarlos.

Modo de contador

Un problema que todos los modos tienen, excepto el modo de libro de código electrónico, es que el acceso aleatorio a datos encriptados es imposible. Por ejemplo, suponga que un archivo se transmite a través de una red y después se almacena en disco de forma encriptada. Ésta sería una forma razonable de operar si la computadora receptora fuera una computadora portátil con riesgo de ser robada. Almacenar todos los archivos importantes de forma encriptada reduce en gran medida el potencial daño que se podría sufrir por la información confidencial que se fugaría en caso de que la computadora cayera en las manos equivocadas.

Sin embargo, los archivos de disco con frecuencia se acceden en orden no secuencial, especialmente los archivos de bases de datos. Con un archivo encriptado mediante encadenamiento de bloques de cifrado, el acceso a un bloque aleatorio requiere que primero se desencripten todos los bloques que estén delante de él, lo cual es muy costoso. Por esta razón, se ha inventado otro modo, el **modo de contador,** como se ilustra en la figura 8-15. Aquí el texto llano no se encripta en forma

directa. En su lugar, se encripta el vector de inicialización más una constante, y al texto cifrado resultante se le aplica un OR exclusivo con el texto llano. Al incrementar en 1 el vector de inicialización por cada nuevo bloque, es más fácil desencriptar un bloque en cualquier parte del archivo sin tener que desencriptar primero todos sus predecesores.

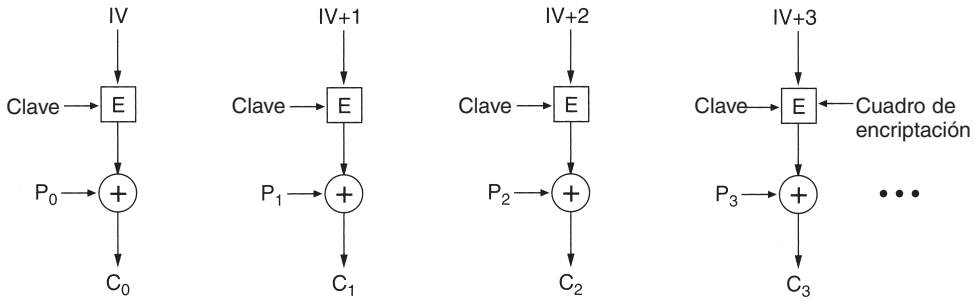


Figura 8-15. Encriptación con el modo de contador.

Aunque el modo de contador es útil, tiene una debilidad que vale la pena mencionar. Suponga que en el futuro se utiliza nuevamente la misma clave, *K* (con texto llano diferente pero el mismo IV), y un atacante adquiere todo el texto cifrado de ambas corridas. Los flujos de claves son los mismos en los dos casos, y exponen el cifrado a un ataque de reutilización de flujo de claves del mismo tipo que vimos en los cifrados de flujo. Todo lo que el criptoanalista tiene que hacer es aplicar un OR exclusivo a los dos textos cifrados en conjunto para eliminar toda la protección criptográfica y sólo obtener el OR exclusivo de los textos llanos. Esta debilidad no significa que el modo de contador sea una mala idea. Simplemente significa que las claves y los vectores de inicialización deben elegirse de manera independiente y al azar. Aun cuando la misma clave se utilice dos veces de manera accidental, si el IV es diferente cada vez, el texto llano estará a salvo.

8.2.4 Otros cifrados

DES y Rijndael son los algoritmos criptográficos de clave simétrica más conocidos. Sin embargo, vale la pena mencionar que se han diseñado otros cifrados de clave simétrica. Algunos de ellos están incluidos en varios productos. En la figura 8-16 se listan algunos de los más comunes.

8.2.5 Criptoanálisis

Antes de dejar el tema de la criptografía de clave simétrica, vale la pena cuando menos mencionar cuatro avances recientes del criptoanálisis. El primero es el **criptoanálisis diferencial** (Biham y Shamir, 1993). Esta técnica puede utilizarse para atacar cualquier cifrado en bloques; empieza con un par de bloques de texto llano que difieren sólo en una cantidad pequeña de bits y observando cuidadosamente lo que ocurre en cada iteración interna a medida que avanza la

Cifrado	Autor	Longitud de clave	Comentarios
Blowfish	Bruce Schneier	1–448 bits	Antiguo y lento
DES	IBM	56 bits	Muy débil para utilizarlo en la actualidad
IDEA	Massey y Xuejia	128 bits	Bueno, pero patentado
RC4	Ronald Rivest	1–2048 bits	Precaución: algunas claves son débiles
RC5	Ronald Rivest	128–256 bits	Bueno, pero patentado
Rijndael	Daemen y Rijmen	128–256 bits	La mejor opción
Serpent	Anderson, Biham, Knudsen	128–256 bits	Muy robusto
Triple DES	IBM	168 bits	Segunda mejor opción
Twofish	Bruce Schneier	128–256 bits	Muy robusto; ampliamente utilizado

Figura 8-16. Algunos algoritmos criptográficos comunes de clave simétrica.

encriptación. En muchos casos, algunos patrones son mucho más comunes que otros, y esta observación conduce a un ataque probabilístico.

El segundo avance que vale la pena mencionar es el **criptoanálisis lineal** (Matsui, 1994). Éste puede descifrar el DES con sólo 2^{43} textos llanos conocidos. Funciona aplicando un OR exclusivo a ciertos bits del texto llano y el texto cifrado en conjunto y buscando patrones en el resultado. Al hacerse repetidamente, la mitad de los bits deben ser 0s y la otra mitad 1s. Sin embargo, con frecuencia los cifrados introducen una desviación en una dirección o en la otra, y esta desviación, por pequeña que sea, puede explotarse para reducir el factor de trabajo. Para los detalles, vea el trabajo de Matsui.

El tercer avance es el análisis del consumo de energía eléctrica para averiguar las claves secretas. Las computadoras por lo general utilizan 3 voltios para representar un bit 1 y 0 voltios para representar un bit 0. Por lo tanto, procesar un 1 gasta más energía eléctrica que procesar un 0. Si un algoritmo criptográfico consiste en un ciclo en el que los bits clave se procesan en orden, un atacante que reemplace el reloj principal de *n* GHz con uno lento (por ejemplo, 100 Hz) y coloque pinzas de caimán en los pines de energía y tierra de la CPU, puede monitorear con precisión la energía consumida por cada instrucción de la máquina. A partir de estos datos, deducir la clave es sorprendentemente fácil. Este tipo de criptoanálisis puede vencerse sólo al codificar con sumo cuidado el algoritmo en lenguaje ensamblador para asegurarse de que el consumo de energía sea independiente de la clave, así como de todas las claves de rondas individuales.

El cuarto avance es el análisis de temporización. Los algoritmos criptográficos están llenos de instrucciones `if` que prueban bits en las claves de ronda. Si las partes `then` y `else` toman diferentes cantidades de tiempo, reduciendo la velocidad del reloj y viendo el tiempo que tardan en ejecutarse varios pasos, también podría ser posible deducir las claves de ronda. Una vez que se conocen todas las claves de ronda, por lo general puede calcularse la clave original. Los análisis de energía y temporización también pueden utilizarse de manera simultánea para facilitar el trabajo. Si bien los análisis de energía y temporización pueden parecer exóticos, en realidad son técnicas poderosas que pueden romper cualquier cifrado que no esté específicamente diseñado para resistirlas.

8.3 ALGORITMOS DE CLAVE PÚBLICA

Históricamente el problema de distribución de claves siempre ha sido la parte débil de la mayoría de los criptosistemas. Sin importar lo robusto que sea un criptosistema, si un intruso puede robar la clave, el sistema no vale nada. Los criptólogos siempre daban por hecho que las claves de encriptación y desencriptación eran la misma (o que se podía derivar fácilmente una de la otra). Pero la clave tenía que distribuirse a todos los usuarios del sistema. Por lo tanto, parecía haber un problema inherente: las claves se tenían que proteger contra robo, pero también se tenían que distribuir, por lo que no podían simplemente guardarse en una caja fuerte.

En 1976, dos investigadores de la Universidad de Stanford, Diffie y Hellman (1976), propusieron una clase nueva de criptosistema, en el que las claves de encriptación y desencriptación eran diferentes y la clave de desencriptación no podía derivarse de la clave de encriptación. En su propuesta, el algoritmo de encriptación (con clave), E, y el algoritmo de desencriptación (con clave), D, tenían que cumplir con los tres requisitos siguientes. Estos requisitos pueden expresarse sencillamente como sigue:

1. $D(E(P)) = P$.
2. Es excesivamente difícil deducir D de E.
3. E no puede descifrarse mediante un ataque de texto llano seleccionado.

El primer requisito dice que, si aplicamos D a un mensaje cifrado, $E(P)$, obtenemos nuevamente el mensaje de texto llano original, P. Sin esta propiedad, el receptor legítimo no podría desencriptar el texto cifrado. El segundo requisito no requiere explicación. El tercer requisito es necesario porque, como veremos en un momento, los intrusos pueden experimentar a placer con el algoritmo. En estas condiciones, no hay razón para que una clave de encriptación no pueda hacerse pública.

El método funciona como sigue. Una persona, llamémosla Alice, que quiera recibir mensajes secretos, primero diseña dos algoritmos, E_A y D_A, que cumplan los requisitos anteriores. El algoritmo de encriptación y la clave de Alice se hacen públicos, de ahí el nombre de **criptografía de clave pública.** Por ejemplo, Alice podría poner su clave pública en su página de inicio en Web. Utilizaremos la notación E_A para denotar el algoritmo de encriptación parametrizado por la clave pública de Alice. De manera similar, el algoritmo de desencriptación (secreto) parametrizado por la clave privada de Alice es D_A. Bob hace lo mismo, haciendo pública E_B pero manteniendo secreta D_B.

Ahora veamos si podemos resolver el problema de establecer un canal seguro entre Alice y Bob, que nunca han tenido contacto previo. Se supone que tanto la clave de encriptación de Alice, E_A, como la clave de encriptación de Bob, E_B, están en un archivo de lectura pública. Ahora, Alice toma su primer mensaje, P, calcula $E_B(P)$ y lo envía a Bob. Bob entonces lo desencripta aplicando su clave secreta D_B [es decir, calcula $D_B(E_B(P)) = P$]. Nadie más puede leer el mensaje

encriptado, $E_B(P)$, porque se supone que el sistema de encriptación es robusto y porque es demasiado difícil derivar D_B de la E_B públicamente conocida. Para enviar una respuesta, R, Bob transmite $E_A(R)$. Ahora, Alice y Bob se pueden comunicar con seguridad.

Es útil una nota sobre terminología. La criptografía de clave pública requiere que cada usuario tenga dos claves: una clave pública, usada por todo el mundo para encriptar mensajes a enviar a ese usuario, y una clave privada, que necesita el usuario para desencriptar los mensajes. Consistentemente nos referimos a estas claves como claves *públicas* y *privadas*, respectivamente, y las distinguiremos de las claves *secretas* usadas en la criptografía convencional de clave simétrica.

8.3.1 El algoritmo RSA

La única dificultad estriba en que necesitamos encontrar algoritmos que realmente satisfagan estos tres requisitos. Debido a las ventajas potenciales de la criptografía de clave pública, muchos investigadores están trabajando a todo vapor, y ya se han publicado algunos algoritmos. Un buen método fue descubierto por un grupo del M.I.T. (Rivest y cols., 1978). Es conocido por las iniciales de sus tres descubridores (Rivest, Shamir, Adleman): **RSA.** Ha sobrevivido a todos los intentos para romperlo por más de un cuarto de siglo y se le considera muy robusto. Mucha de la seguridad práctica se basa en él. Su mayor desventaja es que requiere claves de por lo menos 1024 bits para una buena seguridad (en comparación con los 128 bits de los algoritmos de clave simétrica), por lo cual es muy lento.

Su método se basa en ciertos principios de la teoría de los números. Ahora resumiremos el uso del método; para los detalles, consulte el trabajo original.

1. Seleccionar dos números primos grandes, p y q (generalmente de 1024 bits).
2. Calcular $n = p \times q$ y $z = (p - 1) \times (q - 1)$.
3. Seleccionar un número primo con respecto a z, llamándolo d.
4. Encontrar e tal que $e \times d = 1 \; mod \; z$.

Con estos parámetros calculados por adelantado, estamos listos para comenzar la encriptación. Dividimos el texto llano (considerado como una cadena de bits) en bloques, para que cada mensaje de texto llano, P, caiga en el intervalo $0 \leq P < n$. Esto puede hacerse agrupando el texto llano en bloques de k bits, donde k es el entero más grande para el que $2^k < n$ es verdad.

Para encriptar un mensaje, P, calculamos $C = P^e \pmod{n}$. Para desencriptar C, calculamos $P = C^d \pmod{n}$. Puede demostrarse que, para todos los P del intervalo especificado, las funciones de encriptación y desencriptación son inversas. Para ejecutar la encriptación, se necesitan e y n. Para llevar a cabo la desencriptación, se requieren d y n. Por tanto, la clave pública consiste en el par (e, n), y la clave privada consiste en (d, n).

La seguridad del método se basa en la dificultad para factorizar números grandes. Si el criptoanalista pudiera factorizar n (conocido públicamene), podría encontrar p y q y, a partir de éstos, z. Equipado con el conocimiento de z y de e, puede encontrar d usando el algoritmo

de Euclides. Afortunadamente, los matemáticos han estado tratando de factorizar números grandes durante los últimos 300 años, y las pruebas acumuladas sugieren que se trata de un problema excesivamente difícil.

De acuerdo con Rivest y sus colegas, la factorización de un número de 500 dígitos requiere 10^{25} años de tiempo de cómputo utilizando el mejor algoritmo conocido y una computadora con un tiempo de instrucción de 1 μseg. Aun si las computadoras continúan aumentando su velocidad en un orden de magnitud cada década, pasarán siglos antes de que sea factible la factorización de un número de 500 dígitos, y para entonces nuestros descendientes simplemente pueden escoger un p y un q todavía más grandes.

Un ejemplo pedagógico trivial del algoritmo RSA se muestra en la figura 8-17. Para este ejemplo hemos seleccionado $p = 3$ y $q = 11$, dando $n = 33$ y $z = 20$. Un valor adecuado de d es $d = 7$, puesto que 7 y 20 no tienen factores comunes. Con estas selecciones, e puede encontrarse resolviendo la ecuación $7e = 1 \pmod{20}$, que produce $e = 3$. El texto cifrado, C, de un mensaje de texto llano, P, se da por la regla $C = P^3 \pmod{33}$. El receptor desencripta el texto cifrado de acuerdo con la regla $P = C^7 \pmod{33}$. En la figura se muestra como ejemplo la encriptación del texto llano "SUZANNE".

Texto llano (P)			Texto cifrado (C)		Después de la desencriptación	
Simbólico	Numérico	P^3	P^3 (mod 33)	C^7	C^7 (mod33)	Simbólico
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
		Cálculo del emisor			Cálculo del receptor	

Figura 8-17. Ejemplo del algoritmo RSA.

Dado que los números primos escogidos para este ejemplo son tan pequeños, P debe ser menor que 33, por lo que cada bloque de texto llano puede contener sólo un carácter. El resultado es un cifrado por sustitución monoalfabética, no muy impresionante. En cambio, si hubiéramos seleccionado p y $q \approx 2^{512}$, podríamos tener $n \approx 2^{1024}$, de tal manera que cada bloque podría ser de hasta 1024 bits o 128 caracteres de 8 bits, en comparación con 8 caracteres para el DES y 16 para el AES.

Cabe señalar que el uso del RSA como lo hemos descrito es semejante a usar un algoritmo simétrico en modo ECB: el mismo bloque de entrada da el mismo bloque de salida. Por tanto, se requiere alguna forma de encadenamiento para la encriptación de datos. Sin embargo, en la práctica la mayoría de los sistemas basados en RSA usan criptografía de clave pública principalmente para distribuir claves de sesión de una sola vez para su uso con algún algoritmo de clave simétrica

como el AES o el triple DES. El RSA es demasiado lento para poder encriptar grandes volúmenes de datos, pero se utiliza con amplitud para la distribución de claves.

8.3.2 Otros algoritmos de clave pública

Aunque el RSA se usa ampliamente, de ninguna manera es el único algoritmo de clave pública conocido. El primer algoritmo de clave pública fue el de la mochila (Merkle y Hellman, 1978). La idea aquí es que alguien es dueño de una gran cantidad de objetos, todos con pesos diferentes. El dueño cifra el mensaje seleccionando secretamente un subgrupo de los objetos y colocándolos en la mochila. El peso total de los objetos contenidos en la mochila se hace público, así como la lista de todos los objetos posibles. La lista de los objetos que se metieron en la mochila se mantiene en secreto. Con ciertas restricciones adicionales, el problema de determinar una lista posible de los objetos a partir del peso dado se consideró no computable, y formó la base del algoritmo de clave pública.

El inventor del algoritmo, Ralph Merkle, estaba bastante seguro de que este algoritmo no podía violarse, por lo que ofreció una recompensa de 100 dólares a cualquiera que pudiera descifrarlo. Adi Shamir (la "S" de RSA) pronto lo violó y cobró la recompensa. Sin desmotivarse, Merkle reforzó el algoritmo y ofreció una recompensa de 1000 dólares a quien pudiera violar el nuevo. Ronald Rivest (la "R" de RSA) pronto lo descifró y cobró la recompensa. Merkle no se atrevió a ofrecer 10,000 dólares para la siguiente versión, por lo que "A" (Leonard Adleman) no tuvo suerte. Aunque se ha modificado nuevamente, el algoritmo de la mochila no se considera seguro y pocas veces se usa.

Otros esquemas de clave pública se basan en la dificultad para calcular logaritmos discretos. El Gamal (1985) y Schnorr (1991) han inventado algoritmos basados en este principio.

Existen algunos otros esquemas, como los basados en curvas elípticas (Menezes y Vanstone, 1993), pero las dos categorías principales son las basadas en la dificultad para factorizar números grandes y calcular logaritmos discretos módulo un número primo grande. Estos problemas se consideran verdaderamente difíciles de resolver porque los matemáticos han estado trabajando en ellos durante años sin adelantos importantes.

8.4 FIRMAS DIGITALES

La autenticidad de muchos documentos legales, financieros y de otros tipos se determina por la presencia o ausencia de una firma manuscrita autorizada. Las fotocopias no cuentan. Para que los sistemas de mensajes computarizados reemplacen el transporte físico de papel y tinta, debe encontrarse un método para que la firma de los documentos sea infalsificable.

El problema de inventar un reemplazo para las firmas manuscritas es difícil. Básicamente, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje "firmado" a otra parte de modo que:

1. El receptor pueda verificar la identidad del transmisor.
2. El transmisor no pueda repudiar (negar) después el contenido del mensaje.
3. El receptor no haya podido elaborar el mensaje él mismo.

El primer requisito es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, la computadora del banco necesita asegurarse de que la computadora que da la orden realmente pertenece a la compañía a la que se le aplicará el débito. En otras palabras, el banco tiene que autenticar la identidad del cliente (y éste a su vez tiene que hacer lo propio).

El segundo requisito es necesario para proteger al banco contra fraude. Supongamos que el banco compra la tonelada de oro, e inmediatamente después cae marcadamente el precio del oro. Un cliente deshonesto podría demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presenta un mensaje en la corte, el cliente niega haberlo enviado. La propiedad de que ninguna parte de un contrato pueda negar haber firmado se conoce como **no repudio.** Los esquemas de firma digital que analizaremos ayudan a proporcionar dicha propiedad.

El tercer requisito es necesario para proteger al cliente en el caso de que el precio del oro suba mucho y que el banco trate de falsificar un mensaje firmado en el que el cliente solicitó un lingote de oro en lugar de una tonelada. En este escenario fraudulento, el banco simplemente mantiene el resto del oro para sí mismo.

8.4.1 Firmas de clave simétrica

Un enfoque de las firmas digitales sería tener una autoridad central que sepa todo y en quien todos confíen, digamos el *Big Brother* (*BB*). Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas del *BB*. Por tanto, sólo Alice y el *BB* conocen la clave secreta de Alice, K_A, etcétera.

Cuando Alice quiere enviar un mensaje de texto llano firmado, P, a su banquero, Bob, genera $K_A(B, R_A, t, P)$ donde B es la identidad de Bob, R_A es un número aleatorio elegido por Alice, t es una marca de tiempo para asegurar que el mensaje sea reciente, y $K_A(B, R_A, t, P)$ es el mensaje encriptado con su clave, K_A. A continuación lo envía como se muestra en la figura 8-18. El *BB* ve que el mensaje es de Alice, lo desencripta, y envía un mensaje a Bob como se muestra. El mensaje a Bob contiene el texto llano del mensaje de Alice y también el mensaje firmado $K_{BB}(A, t, P)$. Ahora, Bob atiende la solicitud de Alice.

¿Qué ocurre si Alice niega posteriormente haber enviado el mensaje? El paso 1 es que todos demandan a todos (cuando menos en Estados Unidos). Por último, cuando el caso llega a la corte y Alice niega haber enviado a Bob el mensaje en disputa, el juez pregunta a Bob por qué está tan seguro de que el mensaje en disputa vino de Alice y no de Trudy. Bob primero indica que el *BB* no aceptaría un mensaje de Alice a menos que estuviera encriptado con K_A, por lo que no hay posibilidad de que Trudy enviara al *BB* un mensaje falso de Alice sin que el *BB* lo detectara de inmediato.

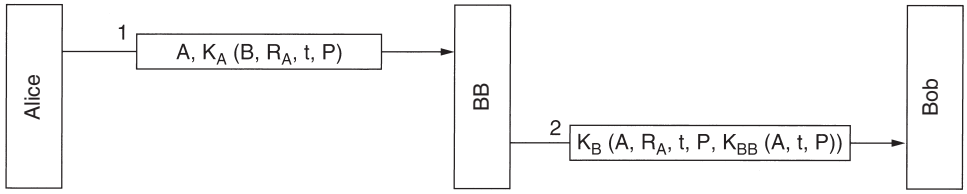


Figura 8-18. Firmas digitales con el Big Brother.

Bob entonces presenta la prueba A, K_{BB} (*A, t, P*). Bob dice que éste es un mensaje firmado por el *BB* que comprueba que Alice envió *P* a Bob. El juez entonces pide al *BB* (en quien todo el mundo confía) que descifre la prueba A. Cuando el *BB* testifica que Bob dice la verdad, el juez se pronuncia a favor de Bob. Caso cerrado.

Un problema potencial del protocolo de firma de la figura 8-18 es que Trudy repita cualquiera de los dos mensajes. Para minimizar este problema, en todos los intercambios se usan marcas de tiempo. Es más, Bob puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Observe que Bob rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, Bob simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de Alice durante la hora pasada. Si no, Bob puede suponer con seguridad que éste es una solicitud nueva.

8.4.2 Firmas de clave pública

Un problema estructural del uso de la criptografía de clave simétrica para las firmas digitales es que todos tienen que confiar en el *Big Brother*. Es más, el *Big Brother* lee todos los mensajes firmados. Los candidatos más lógicos para operar el servidor del *Big Brother* son el gobierno, los bancos, los contadores y los abogados. Por desgracia, ninguna de estas organizaciones inspira confianza completa a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriese una autoridad confiable.

Afortunadamente, la criptografía de clave pública puede hacer una contribución importante aquí. Supongamos que los algoritmos públicos de encriptación y desencriptación tienen la propiedad de que $E(D(P)) = P$ además de la propiedad normal de $D(E(P)) = P$. (El RSA tiene esta propiedad, por lo que el supuesto es razonable.) Suponiendo que éste es el caso, Alice puede enviar un mensaje de texto llano firmado, *P*, a Bob, transmitiendo $E_B(D_A(P))$. Observe que Alice conoce su propia clave (privada), D_A, así como la clave pública de Bob, E_B, por lo que Alice puede elaborar este mensaje.

Cuando Bob recibe el mensaje, lo transforma usando su clave privada, como es normal, produciendo $D_A(P)$, como se muestra en la figura 8-19. Bob almacena este texto en un lugar seguro y lo descifra usando E_A para obtener el texto llano original.

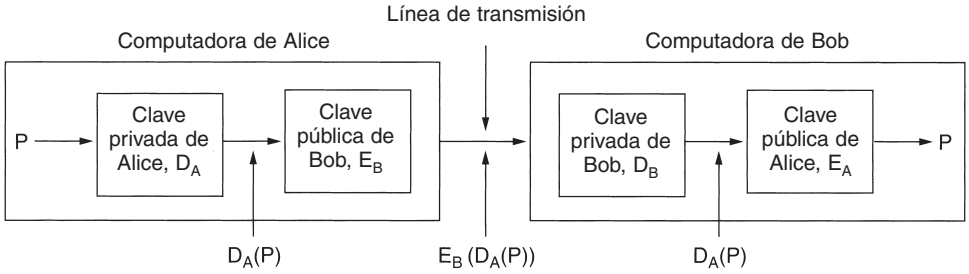


Figura 8-19. Firmas digitales usando criptografía de clave pública.

Para ver cómo funciona la propiedad de firma, supongamos que Alice posteriormente niega haber enviado el mensaje *P* a Bob. Cuando el caso llega a la corte, Bob puede presentar tanto *P* como $D_A(P)$. El juez puede comprobar fácilmente que Bob tiene un mensaje válido encriptado con D_A con sólo aplicarle E_A. Puesto que Bob no conoce la clave privada de Alice, la única forma en que Bob pudo haber adquirido un mensaje encriptado con esa clave sería que Alice en efecto lo hubiera enviado. Mientras está en la cárcel por perjurio y fraude, Alice tendrá tiempo suficiente para diseñar algoritmos de clave pública nuevos e interesantes.

Aunque el uso de la criptografía de clave pública para las firmas digitales es un esquema elegante, hay problemas relacionados con el entorno en el que opera más que con el algoritmo básico. Por una parte, Bob puede demostrar que un mensaje fue enviado por Alice siempre y cuando D_A permanezca en secreto. Si Alice divulga su clave secreta, el argumento ya no es válido, puesto que cualquiera pudo haber enviado el mensaje, incluido el mismo Bob.

El problema podría surgir, por ejemplo, si Bob es el corredor de bolsa de Alice. Alice le indica a Bob que compre ciertas acciones o bonos. Inmediatamente después, el precio cae en picada. Para negar haberle enviado el mensaje a Bob, Alice corre a la policía para informarles que robaron su casa y que también sustrajeron su clave. Dependiendo de las leyes de su estado o país, puede o no ser responsable legalmente, en especial si indica no haber descubierto el robo sino hasta después de llegar del trabajo, varias horas después.

Otro problema con el esquema de firmas es qué ocurre si Alice decide cambiar su clave. Hacerlo ciertamente es legal, y probablemente es una buena idea cambiar la clave periódicamente. Si luego surge un caos en la corte, como se describió antes, el juez aplicará la E_A *actual* a $D_A(P)$ y descubrirá que no produce *P.* Bob quedará en ridículo en ese momento.

En principio, cualquier algoritmo de clave pública puede usarse para firmas digitales. El estándar *de facto* de la industria es el algoritmo RSA, y muchos productos de seguridad lo usan. Sin embargo, en 1991, el NIST (Instituto Nacional de Estándares y Tecnología) propuso el uso de una variación del algoritmo de clave pública de El Gamal para su nuevo **Estándar de Firmas Digitales (DSS).** La seguridad de El Gamal radica en la dificultad para calcular logaritmos discretos, en lugar de la dificultad para factorizar números grandes.

Como es normal cuando el gobierno intenta dictar estándares criptográficos, hubo una protesta general. El DSS tuvo críticas por ser

1. Demasiado secreto (el NSA diseñó el protocolo para usar El Gamal).
2. Demasiado lento (10 a 40 veces más lento que el RSA para comprobar firmas).
3. Demasiado nuevo (El Gamal no ha sido analizado exhaustivamente).
4. Demasiado inseguro (clave fija de 512 bits).

En una modificación posterior, el cuarto punto se prestó a discusión pública cuando se permitieron claves de hasta 1024 bits. Sin embargo, los primeros dos puntos permanecen válidos.

8.4.3 Compendios de mensaje

Una crítica a los métodos de firma es que con frecuencia combinan dos funciones dispares: autenticación y confidencialidad. En muchos casos se requiere la autenticación, pero no confidencialidad. Asimismo, con frecuencia la obtención de una licencia de exportación se facilita si el sistema en cuestión sólo proporciona autenticación pero no confidencialidad. A continuación describiremos un esquema de autenticación que no requiere la encriptación del mensaje completo.

Este esquema se base en la idea de una función de *hash* unidireccional que toma una parte arbitrariamente grande de texto llano y a partir de ella calcula una cadena de bits de longitud fija. Esta función de *hash*, *MD*, llamada **compendio de mensaje** (*message digest*), tiene cuatro propiedades importantes:

1. Dado P, es fácil calcular $MD(P)$.
2. Dado $MD(P)$, es imposible encontrar P.
3. Dado P nadie puede encontrar P' de tal manera que $MD(P') = MD(P)$.
4. Un cambio a la entrada de incluso 1 bit produce una salida muy diferente.

Para cumplir el criterio 3, la función de *hash* debe ser de cuando menos 128 bits de longitud, y de preferencia mayor. Para cumplir el criterio 4, la función de *hash* debe truncar los bits minuciosamente, de manera semejante a como lo hacen los algoritmos de encriptación de clave simétrica que hemos visto.

El cálculo de un compendio de mensaje a partir de un trozo de texto llano es mucho más rápido que la encriptación de ese texto llano con un algoritmo de clave pública, por lo que los compendios de mensaje pueden usarse para acelerar los algoritmos de firma digital. Para ver su funcionamiento, considere nuevamente el protocolo de firma de la figura 8-18. En lugar de firmar P con $K_{BB}(A, t, P)$, el *BB* ahora calcula el compendio del mensaje aplicando *MD* a P para producir $MD(P)$. El *BB* entonces incluye $K_{BB}(A, t, MD(P))$ como quinto elemento de la lista encriptada con K_B que se envía a Bob, en lugar de $K_{BB}(A, t, P)$.

Si surge una disputa, Bob puede presentar tanto P como $K_{BB}(A, t, MD(P))$. Una vez que el *Big Brother* lo desencripta para el juez, Bob tiene $MD(P)$, que está garantizado que es genuino, y el P

supuesto. Dado que es prácticamente imposible que Bob encuentre otro mensaje que dé este resultado, el juez se convencerá fácilmente de que Bob dice la verdad. Este uso de compendios de mensaje ahorra tanto tiempo de encriptación como costos de transporte de mensajes.

Los compendios de mensaje funcionan también en los criptosistemas de clave pública, como se muestra en la figura 8-20. Aquí, Alice primero calcula el compendio de mensaje de su texto llano: luego firma el compendio del mensaje y envía a Bob tanto el compendio firmado como el texto llano. Si Trudy reemplaza *P* en el camino, Bob verá esto cuando calcule *MD*(*P*) él mismo.

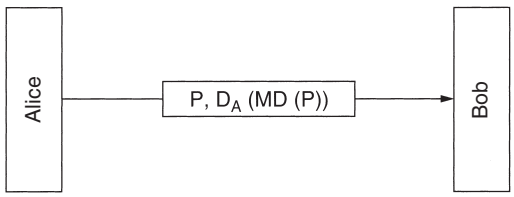


Figura 8-20. Firmas digitales usando compendios de mensaje.

MD5

Se ha propuesto una variedad de funciones para el compendio de mensajes. Las de mayor uso son MD5 (Rivest, 1992) y SHA-1 (NIST, 1993). **MD5** es la quinta de una serie de compendios de mensaje diseñados por Ronald Rivest. Opera truncando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Muy brevemente, comienza por rellenar el mensaje a una longitud de 448 bits (módulo 512). Después la longitud original del mensaje se agrega como entero de 64 bits para dar una entrada total cuya longitud es un múltiplo de 512 bits. El último paso del cálculo previo es la inicialización de un búfer de 128 bits a un valor fijo.

Ahora comienza el cálculo. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el búfer de 128 bits. Por si fuera poco, se introduce también una tabla construida a partir de la función seno. El objetivo de usar una función conocida como el seno no es porque sea más aleatoria que un generador de números aleatorios, sino para evitar cualquier sospecha de que el diseñador construyó una puerta trasera ingeniosa por la que sólo él puede entrar. La negativa de IBM a hacer públicos los principios en que se basó el diseño de las cajas S del DES dio pie a una gran cantidad de especulación sobre las puertas traseras. Se hacen cuatro rondas por cada bloque de entrada. Este proceso continúa hasta que todos los bloques de entrada se han consumido. El contenido del búfer de 128 bits forma el compendio del mensaje.

MD5 ha existido aproximadamente por una década, y muchas personas lo han atacado. Se han encontrado algunas vulnerabilidades, pero ciertos pasos internos evitan que sea violado. Sin embargo, si cayeran las barreras restantes dentro de MD5, éste podría fallar con el tiempo. Sin embargo, al momento de escribir esto, aún están de pie.

SHA-1

La otra función principal para el compendio de mensajes es **SHA-1** (**Algoritmo Seguro de *Hash* 1**), desarrollado por NSA y aprobado por el NIST en FIPS 180-1. Al igual que MD5, SHA-1 procesa datos de entrada en bloques de 512 bits, sólo a diferencia de MD5, genera un compendio de mensaje de 160 bits. En la figura 8-21 se ilustra una forma típica para que Alice envíe a Bob un mensaje no secreto, pero firmado. Aquí su mensaje de texto llano se alimenta en el algoritmo SHA-1 para obtener un *hash* SHA-1 de 160 bits. A continuación Alice firma el *hash* con su clave privada RSA y envía a Bob tanto el mensaje de texto llano como el *hash* firmado.

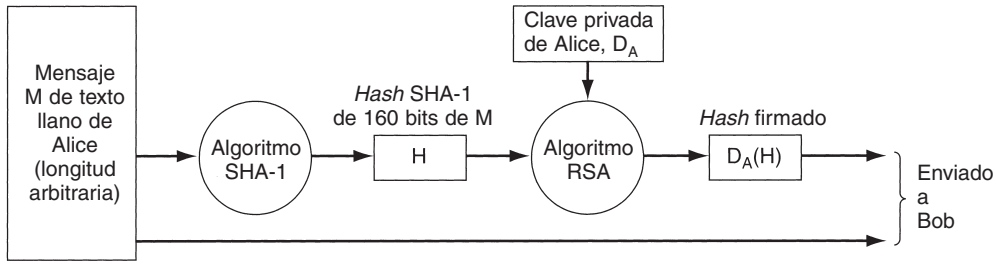


Figura 8-21. Uso de SHA-1 y RSA para firmar mensajes no secretos.

Después de recibir el mensaje, Bob calcula el *hash* SHA-1 él mismo y también aplica la clave pública de Alice al *hash* firmado para obtener el *hash* original, *H*. Si los dos concuerdan, el mensaje se considera válido. Puesto que no hay forma de que Trudy modifique el mensaje (de texto llano) mientras está en tránsito y producir uno nuevo que haga *hash* a *H*, Bob puede detectar con facilidad cualquier cambio que Trudy haya hecho al mensaje. Para los mensajes cuya integridad es importante pero cuyo contenido no es secreto, se utiliza ampliamente el esquema de la figura 8-21. Por un costo de cómputo relativamente bajo, garantiza que cualquier modificación hecha al mensaje de texto llano en tránsito pueda detectarse con una probabilidad muy alta.

Ahora veamos brevemente cómo funciona SHA-1. Comienza rellenando el mensaje con 1 bit al final, seguido de tantos bits 0 como sean necesarios para que la longitud sea un múltiplo de 512 bits. A continuación, al número de 64 bits que contiene la longitud del mensaje antes del relleno se le aplica un OR dentro de los 64 bits de menor orden. En la figura 8-22, el mensaje se muestra con un relleno a la derecha debido a que el texto y las figuras van de izquierda a derecha (es decir, el extremo inferior derecho por lo general se percibe como el final de la figura). En las computadoras esta orientación corresponde a máquinas *big-endian* como la SPARC, pero SHA-1 siempre rellena el final del mensaje, sin importar cuál máquina *endian* se utilice.

Durante el cálculo, SHA-1 mantiene variables de 32 bits, H_0 a H_4, donde se acumula el *hash*. Dichas variables se muestran en la figura 8-22(b). Se inicializan a constantes especificadas en el estándar.

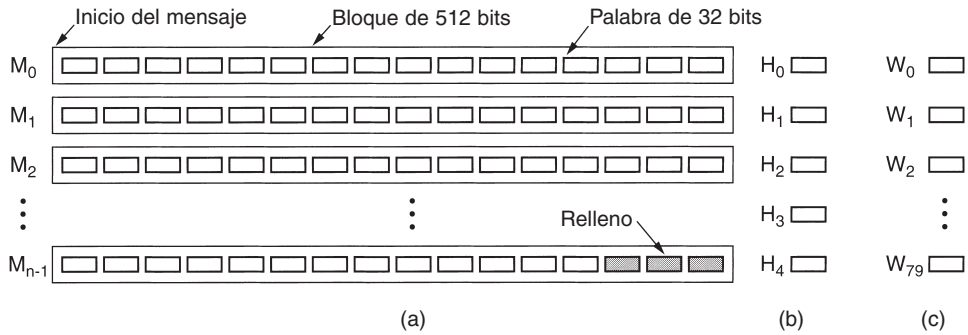


Figura 8-22. (a) Un mensaje rellenado a un múltiplo de 512 bits. (b) Las variables de salida. (c) El arreglo de palabras.

Ahora se procesa cada uno de los bloques M_0 a M_{n-1}. Para el bloque actual, las 16 palabras primero se copian al inicio de un arreglo auxiliar de 80 palabras, W, como se muestra en la figura 8-22(c). Después las otras 64 palabras de W se rellenan utilizando la fórmula

$$W_i = S^1 (W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

donde $S^b(W)$ representa la rotación circular izquierda de la palabra de 32 bits, W, por b bits. Ahora cinco variables de trabajo, A a E se inicializan de H_0 a H_4, respectivamente.

El cálculo real puede expresarse en pseudo-C como

```
for (i = 0; i < 80; i++) {
    temp = S^5 (A) + f_i (B, C, D) + E + W_i + K_i;
    E = D; D = C; C = S^30(B); B = A; A = temp;
}
```

donde las constantes K_i se definen en el estándar. Las funciones de mezcla f_i se definen como

$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D)$ $(0 \le i \le 19)$
$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D$ $(20 \le i \le 39)$
$f_i\,(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D)$ $(40 \le i \le 59)$
$f_i\,(B,C,D) = B \text{ XOR } C \text{ XOR } D$ $(60 \le i \le 79)$

Cuando las 80 iteraciones del ciclo estén completas, A a E se agregan a H_0 a H_4, respectivamente.

Ahora que se han procesado los primeros bloques de 512 bits, se inicia el siguiente. El arreglo W se reinicia desde el nuevo bloque, pero H se deja como estaba. Cuando se termina este bloque, se inicia el siguiente, y así sucesivamente, hasta que los bloques de mensajes de 512 bits se hayan procesado por completo. Cuando se termina el último bloque, las cinco palabras de 32 bits en el arreglo H se envían a la salida como el *hash* criptográfico de 160 bits. El código C completo para SHA-1 se da en el RFC 3174.

Nuevas versiones de SHA-1 están en desarrollo para *hashes* de 256, 384 y 512 bits, respectivamente.

8.4.4 El ataque de cumpleaños

En el mundo de la criptografía, nada es lo que parece. Podríamos pensar que se requieren del orden de 2^m operaciones para subvertir un compendio de mensajes de m bits. De hecho, con frecuencia $2^{m/2}$ operaciones son suficientes si se usa el **ataque de cumpleaños,** un enfoque publicado por Yuval (1979) en su ahora clásico trabajo "*How to Swindle Rabin*" (Cómo estafar a Rabin).

La idea de este ataque proviene de una técnica que con frecuencia usan los profesores de matemáticas en sus cursos de probabilidad. La pregunta es: ¿Cuántos estudiantes se necesitan en un grupo antes de que la probabilidad de tener dos personas con el mismo cumpleaños exceda 1/2? Muchos estudiantes suponen que la respuesta debe ser mucho mayor que 100. De hecho, la teoría de la probabilidad indica que es de apenas 23. Sin hacer un análisis riguroso, intuitivamente, con 23 personas, podemos formar $(23 \times 22)/2 = 253$ pares diferentes, cada uno de los cuales tiene una probabilidad de 1/365 de cumplir el requisito. Bajo esta luz, la respuesta ya no es en realidad tan sorprendente.

En términos más generales, si hay alguna correspondencia entre las entradas y las salidas con n entradas (gente, mensajes, etc.) y k salidas posibles (cumpleaños, compendios de mensajes, etc.) hay $n(n-1)/2$ pares de entradas. Si $n(n-1)/2 > k$, la posibilidad de que cuando menos una coincida es bastante buena. Por lo tanto, en términos aproximados, es probable una igualación para $n > \sqrt{k}$. Este resultado significa que un compendio de mensajes de 64 bits probablemente puede violarse generando unos 2^{32} mensajes y buscando dos con el mismo compendio de mensaje.

Veamos ahora un ejemplo práctico. El Departamento de Informática de la Universidad Estatal tiene una cátedra para un miembro facultativo y tiene dos candidatos, Tom y Dick. Tom fue contratado dos años antes que Dick, por lo que su candidatura será considerada antes. Si Tom obtiene el puesto, mala suerte para Dick. Tom sabe que la jefa del departamento, Marilyn, tiene buen concepto de su trabajo, por lo que le pide que escriba una carta de recomendación para el rector, quien decidirá el caso de Tom. Una vez enviadas, todas las cartas se vuelven confidenciales.

Marilyn le dice a su secretaria, Ellen, que escriba una carta al rector, delineando lo que quiere en ella. Cuando está lista, Marilyn la revisará, calculará y firmará el compendio de 64 bits y lo enviará al rector. Ellen puede mandar la carta después por correo electrónico.

Desgraciadamente para Tom, Ellen tiene una relación sentimental con Dick y le gustaría dejar fuera a Tom, por lo que escribe la carta siguiente con las 32 opciones entre corchetes.

Estimado rector Smith,

Esta [*carta* | *mensaje*] es para dar mi opinión [*franca* | *honesta*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] una cátedra. He [*conocido a* | *trabajado con*] el profesor Wilson durante [*unos* | *casi*] seis años. Es un investigador [*sobresaliente* | *excelente*] de gran [*talento* | *habilidad*] conocido [*mundialmente* | *internacionalmente*] por sus [*brillantes* | *creativas*] investigaciones sobre [*muchos* | *una gran variedad de*] problemas [*difíciles* | *desafiantes*].

Él es también un [*profesor* | *educador*] [*altamente* | *grandemente*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*sobresalientes* | *espectaculares*] de sus [*clases* | *cursos*]; es el [*profesor* | *instructor*] [*más admirado* | *más querido*] [*del departamento* | *por nosotros*].

[*Además* | *Por otra parte*], el profesor Wilson es [*hábil* | *diligente*] para obtener financiamiento. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*importante* | *sustancial*] de dinero [*al* | *a nuestro*] departamento. [*Este dinero ha* | *Estos fondos han*] [*posibilitado* | *permitido*] que [*emprendamos* | *pongamos en práctica*] muchos programas [*especiales* | *importantes*], [*como* | *por ejemplo*] su programa Estado 2000. Sin estos fondos [*seríamos incapaces de* | *no podríamos*] continuar este programa, que es tan [*importante* | *esencial*] para ambos. Recomiendo encarecidamente que se le otorgue la cátedra.

Desgraciadamente para Tom, tan pronto como Ellen termina de redactar y mecanografiar esta carta, también escribe una segunda:

Estimado rector Smith:

Esta [*carta* | *mensaje*] es para dar mi opinión [*franca* | *honesta*] sobre el profesor Tom Wilson, que [*ahora* | *este año*] [*es candidato a* | *está en espera de*] una cátedra. He [*conocido a* | *trabajado con*] el profesor Wilson durante [*unos* | *casi*] seis años. Es un investigador [*malo* | *mediocre*] poco conocido en su [*campo* | *área*]. Sus investigaciones [*casi nunca* | *pocas veces*] muestran [*entendimiento* | *comprensión del*] los problemas [*clave* | *principales*] [*del día* | *de nuestros tiempos*].

Es más, no es un [*profesor* | *educador*] [*respetado* | *admirado*]. Sus estudiantes han hecho evaluaciones [*pobres* | *malas*] de sus [*clases* | *cursos*]; es el [*maestro* | *instructor*] menos querido [*del departamento* | *por nosotros*], conocido [*principalmente* | *más*] en [*el* | *nuestro*] departamento por su [*tendencia* | *propensión*] a [*ridiculizar* | *avergonzar*] a los estudiantes lo bastante [*tontos* | *imprudentes*] como para hacer preguntas durante su clase.

[*Además* | *Por otra parte*], el profesor Wilson no es [*hábil* | *diligente*] para obtener financiamiento. Sus [*subvenciones* | *contratos*] han aportado una cantidad [*insustancial* | *insignificante*] de dinero [*al* | *a nuestro*] departamento. A menos que [*se recolecte dinero nuevo* | *se consigan fondos nuevos*] pronto tendremos que cancelar algunos programas esenciales, como su programa Estado 2000. Desgraciadamente, en estas [*condiciones* | *circunstancias*] no puedo recomendarlo de buena [*conciencia* | *fe*] a usted para [*la cátedra* | *un puesto permanente*].

Ahora Ellen prepara su computadora para calcular los 2^{32} compendios de mensaje para cada carta durante la noche. Con suerte, un compendio de la primera carta será igual a un compendio de la segunda. De no serlo puede agregar algunas opciones más e intentar de nuevo durante el fin de semana. Supongamos que encuentra una correspondencia. Llamemos a la carta "buena" *A* y a la "mala" *B*.

Ahora Ellen envía la carta *A* a Marilyn para su aprobación y mantiene en completo secreto la carta *B*, sin mostrársela a nadie. Marilyn, por supuesto, la aprueba, calcula su compendio de mensaje de 64 bits, firma el compendio y manda por correo electrónico el compendio firmado al rector Smith. Independientemente, Ellen envía la carta *B* al rector (no la carta *A*, que se suponía debía haber enviado).

Al recibir la carta y el compendio de mensaje firmado, el rector ejecuta el algoritmo de compendio de mensaje con la carta *B*, constata que coincide con lo que Marilyn le envió y despide a Tom. El rector no se da cuenta de que Ellen se las ingenió para generar dos cartas con el mismo compendio de mensaje y le envió una diferente de la que Marilyn vio y aprobó. (Desenlace opcio-

nal: Ellen le dice a Dick lo que hizo. Dick se horroriza y rompe con ella. Ellen se pone furiosa y confiesa su falta a Marilyn. Ésta llama al rector. Tom consigue la cátedra a fin de cuentas.) Con el MD5, el ataque de cumpleaños es difícil porque aun a una velocidad de mil millones de compendios por segundo, se requerirían más de 500 años para calcular los 2^{64} compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantiza una equivalencia. Por supuesto, con 5000 computadoras operando en paralelo, 500 años se convierten en 5 semanas. SHA-1 es mejor (porque es más largo).

8.5 ADMINISTRACIÓN DE CLAVES PÚBLICAS

La criptografía de clave pública hace posible que las personas que no comparten una clave común se comuniquen con seguridad. También posibilita firmar mensajes sin la presencia de un tercero confiable. Por último, los compendios de mensajes firmados hacen que verificar fácilmente la integridad de mensajes recibidos sea una realidad.

Sin embargo, hay un problema que hemos pasado por alto: si Alice y Bob no se conocen entre sí, ¿cómo obtiene cada uno la clave pública del otro para iniciar el proceso de comunicación? La solución más obvia —colocar su clave pública en su sitio Web— no funciona por la siguiente razón. Suponga que Alice quiere buscar la clave pública de Bob en el sitio Web de él. ¿Cómo lo hace? Comienza tecleando el URL de Bob. A continuación su navegador busca la dirección DNS de la página de inicio de Bob y le envía una solicitud *GET*, como se muestra en la figura 8-23. Desgraciadamente, Trudy intercepta la solicitud y responde con una página de inicio falsa, probablemente una copia de la de Bob, excepto por el reemplazo de la clave pública de Bob con la de Trudy. Cuando Alice encripta su primer mensaje con E_T, Trudy lo desencripta, lo lee, lo vuelve a encriptar con la clave pública de Bob y lo envía a éste, quien no tiene la menor idea de que Trudy está leyendo los mensajes que le llegan. Peor aún, Trudy puede modificar los mensajes antes de volverlos a encriptar para Bob. Claramente, se necesita un mecanismo para asegurar que las claves públicas puedan intercambiarse de manera segura.

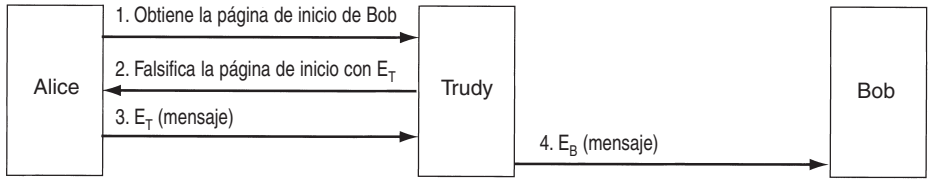


Figura 8-23. Una forma mediante la que Trudy puede subvertir la encriptación de clave pública.

8.5.1 Certificados

Como un primer intento para distribuir claves públicas de manera segura, podemos imaginar un centro de distribución de claves disponible en línea las 24 horas del día que proporciona claves públicas a petición. Uno de los muchos problemas con esta solución es que no es escalable, y el

centro de distribución de claves podría volverse rápidamente un cuello de botella. Además, si alguna vez fallara, la seguridad en Internet podría reducirse a nada.

Por estas razones, se ha desarrollado una solución diferente, una que no requiere que el centro de distribución esté en línea todo el tiempo. De hecho, ni siquiera tiene que estar en línea. En su lugar, lo que hace es certificar las claves públicas que pertenecen a las personas, empresas y otras organizaciones. Una organización que certifica claves públicas se conoce como **CA (autoridad de certificación).**

Como un ejemplo, suponga que Bob desea permitir que Alice y otras personas se comuniquen con él de manera segura. Él puede ir con la CA con su clave pública junto con su pasaporte o licencia de conducir para pedir su certificación. A continuación, la CA emite un certificado similar al que se muestra en la figura 8-24 y firma su *hash* SHA-1 con la clave privada de la CA. Bob paga la cuota de la CA y obtiene un disco flexible que contiene el certificado y su *hash* firmado.

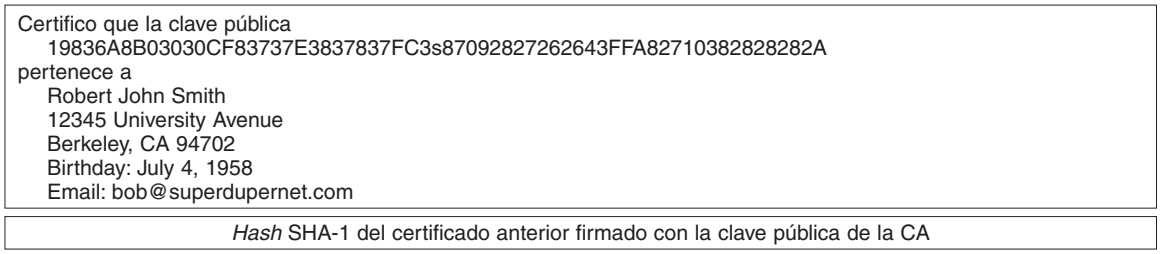


Figura 8-24. Un posible certificado y su *hash* firmado.

El trabajo fundamental de un certificado es enlazar una clave pública con el nombre de un personaje principal (individual, empresa, etcétera). Los certificados mismos no son secretos ni protegidos. Por ejemplo, Bob podría decidir colocar su nuevo certificado en su sitio Web, con un vínculo en la página de inicio que diga: Haga clic aquí para obtener mi certificado de clave pública. El clic resultante podría regresar el certificado y el bloque de la firma (el *hash* SHA-1 firmado del certificado).

Ahora vayamos nuevamente al escenario que se muestra en la figura 8-23. Cuando Trudy intercepta la solicitud que Alice realiza para obtener la página de inicio de Bob, ¿qué puede hacer Trudy? Puede poner su propio certificado y bloque de firma en la página falsificada, pero cuando Alice lea el certificado, verá inmediatamente que no está hablando con Bob porque el nombre de éste no se encuentra en dicho certificado. Trudy puede modificar sobre la marcha la página de inicio de Bob, reemplazando la clave privada de Bob con la suya. Sin embargo, cuando Alice ejecute el algoritmo SHA-1 en el certificado, obtendrá un *hash* que no corresponde con el que obtuvo cuando aplicó la clave privada bien conocida de la CA al bloque de firma. Puesto que Trudy no tiene la clave privada de la CA, no tiene forma de generar un bloque de firma que contenga el *hash* de la página Web modificada con su clave pública en él. De esta manera, Alice puede estar segura de que tiene la clave pública de Bob y no la de Trudy o la de alguien más. Y, como prometimos, este esquema no requiere que la CA esté en línea para la verificación, por lo tanto se elimina un cuello de botella potencial.

Mientras que la función estándar de un certificado es enlazar una clave pública a un personaje principal, un certificado también se puede utilizar para enlazar una clave pública a un **atributo.** Por ejemplo, un certificado podría decir: Esta clave pública pertenece a alguien mayor de 18 años. Podría utilizarse para probar que el dueño de la clave privada no es una persona menor de edad y, por lo tanto, se le permitió acceder material no apto para niños, entre otras cosas, pero sin revelar la identidad del dueño. Por lo general, la persona que tiene el certificado podría enviarlo al sitio Web, al personaje principal o al proceso que se preocupa por la edad. El sitio, el personaje principal o el proceso podrían generar a continuación un número aleatorio y encriptarlo con la clave pública del certificado. Si el dueño pudiera desencriptarlo y regresarlo, ésa sería una prueba de que el dueño tenía el atributo establecido en el certificado. De manera alternativa, el número aleatorio podría utilizarse para generar una clave de sesión para la conversación resultante.

Otro ejemplo en el que un certificado podría contener un atributo es un sistema distribuido orientado a objetos. Cada objeto normalmente tiene múltiples métodos. El dueño del objeto podría proporcionar a cada cliente un certificado que dé un mapa de bits de cuáles métodos puede invocar y que enlace dicho mapa de bits a una clave pública mediante un certificado firmado. Nuevamente, si el dueño del certificado puede probar la posesión de la clave privada correspondiente, se le permitirá realizar los métodos en el mapa de bits. Tiene la propiedad de que la identidad del dueño no necesita conocerse, lo cual es útil en las situaciones en las que la privacidad es importante.

8.5.2 X.509

Si todas las personas que desean algo firmado fueran a la CA con un tipo diferente de certificado, administrar todos los formatos diferentes pronto se volvería un problema. Para resolverlo se ha diseñado un estándar para certificados, el cual ha sido aprobado por la ITU. Dicho estándar se conoce como **X.509** y se utiliza ampliamente en Internet. Ha tenido tres versiones desde su estandarización inicial en 1988. Aquí analizaremos la versión V3.

El X.509 ha recibido una enorme influencia del mundo de OSI, y ha tomado prestadas algunas de sus peores características (por ejemplo, la asignación de nombres y la codificación). Sorprendentemente, la IETF estaba de acuerdo con el X.509, aunque en casi todas las demás áreas, desde direcciones de máquinas, protocolos de transporte hasta formatos de correo electrónico, la IETF por lo general ignoró a la OSI y trató de hacerlo bien. La versión IETF del X.509 se describe en el RFC 3280.

En esencia, el X.509 es una forma de describir certificados. Los campos principales en un certificado se listan en la figura 8-25. Las descripciones dadas ahí deben proporcionar una idea general de lo que hacen los campos. Para información adicional, por favor consulte el estándar mismo o el RFC 2459.

Por ejemplo, si Bob trabaja en el departamento de préstamos del Banco Monetario, su dirección X.500 podría ser:

```
/C=MX/O=BancoMonetario/OU=Prestamo/CN=Bob/
```

donde *C* corresponde al país, *O* a la organización, *OU* a la unidad organizacional y *CN* a un nombre común. Las CAs y otras entidades se nombran de forma similar. Un problema considerable con

Campo	Significado
Versión	Cuál versión del X.509
Número de serie	Este número junto con el nombre de la CA identifican de manera única el certificado
Algoritmo de firma	El algoritmo que se utilizó para firmar el certificado
Emisor	El nombre X.500 de la CA
Validez	Las fechas de inicio y final del periodo de validez
Nombre del sujeto	La entidad cuya clave se está certificando
Clave pública	La clave pública del sujeto y el ID del algoritmo usado para generarla
ID del emisor	Un ID opcional que identifica de manera única al emisor del certificado
ID del sujeto	Un ID opcional que identifica de manera única al sujeto del certificado
Extensiones	Se han definido muchas extensiones
Firma	La firma del certificado (firmada por la clave privada de la CA)

Figura 8-25. Los campos básicos de un certificado X.509.

los nombres X.500 es que si Alice está tratando de contactar a *bob@bancomonetario.com* y se le da un certificado con un nombre X.500, tal vez no sea obvio para ella que el certificado se refiera al Bob que ella busca. Por fortuna, a partir de la versión 3 se permiten los nombres DNS en lugar de los de X.500, por lo que este problema terminará por resolverse en algún momento.

Los certificados están codificados mediante la **ASN.1** (**Notación de Sintaxis Abstracta 1**) de la OSI, que puede considerarse como si fuera una estructura de C, pero con una notación peculiar y poco concisa. Es posible encontrar información sobre X.509 en (Ford y Baum, 2000).

8.5.3 Infraestructuras de clave pública

El hecho de que una sola CA emita todos los certificados del mundo obviamente no funciona. Podría derrumbarse por la carga y también podría ser un punto central de fallas. Una posible solución sería tener múltiples CAs que fueran ejecutadas por la misma organización y que utilizaran la misma clave privada para firmar los certificados. Si bien esto podría solucionar los problemas de carga y de fallas, introduciría un nuevo problema: la fuga de claves. Si hubiera docenas de servidores esparcidos por todo el mundo, todos con la misma clave privada de la CA, la probabilidad de que la clave privada fuera robada o filtrada se incrementaría de manera considerable. Puesto que la situación comprometida de esta clave arruinaría la infraestructura de la seguridad electrónica mundial, tener una sola CA central es muy peligroso.

Además, ¿qué organización podría operar la CA? Es difícil imaginar cualquier autoridad que podría ser aceptada mundialmente como legítima y digna de confianza. En algunos países las personas insistirían en que fuera el gobierno, mientras que en otros lo rechazarían.

Por estas razones, se ha desarrollado una forma diferente para certificar claves públicas. Tiene el nombre general **PKI** (**Infraestructura de Clave Pública**). En esta sección resumiremos cómo funciona, aunque ha habido diversas propuestas, por lo que los detalles podrían cambiar con el tiempo.

Una PKI tiene múltiples componentes, entre ellos usuarios, CAs, certificados y directorios. Lo que una PKI hace es proporcionar una forma para estructurar estos componentes y definir estándares para los diversos documentos y protocolos. Una forma particularmente simple de PKI es una jerarquía de CAs, como se muestra en la figura 8-26. En este ejemplo mostramos tres niveles, pero en la práctica podrían ser menos o más. La CA de nivel superior, la raíz, certifica a CAs de segundo nivel, a las que llamaremos **RAs** (**Autoridades Regionales**) debido a que podrían cubrir alguna región geográfica, como un país o un continente. Sin embargo, este término no es estándar; de hecho, ningún término es realmente estándar para los diversos niveles del árbol. Estas RAs, a su vez, certifican a los CAs reales, las cuales emiten los certificados X.509 a organizaciones e individuos. Cuando la raíz autoriza una nueva RA, genera un certificado X.509 donde indica que ha aprobado la RA, e incluye en él la nueva clave pública de la RA, la firma y se la proporciona a la RA. De manera similar, cuando una RA aprueba una CA, produce y firma un certificado que indica su aprobación y que contiene la clave pública de la CA.

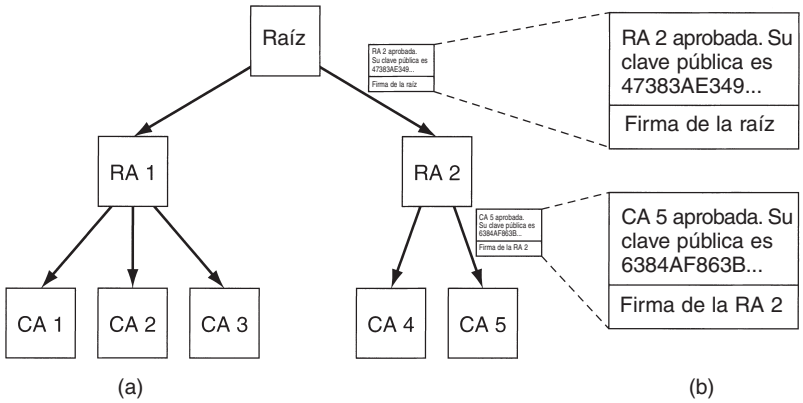


Figura 8-26. (a) Una PKI jerárquica. (b) Una cadena de certificados.

Nuestra PKI funciona como se muestra a continuación. Suponga que Alice necesita la clave pública de Bob para comunicarse con él, por lo que busca y encuentra un certificado que la contiene, firmado por la CA 5. Sin embargo, Alice nunca ha escuchado sobre la CA 5. Por lo que ella sabe, CA 5 podría ser la hija de 10 años de Bob. Puede ir con la CA 5 y decirle: Prueba tu autenticidad. La CA 5 responde con el certificado que obtuvo de la RA 2, el cual contiene la clave pública de la CA 5. Una vez que tiene la clave pública de la CA 5, Alice puede verificar que el certificado de Bob realmente fue firmado por la CA 5 y que, por lo tanto, es legal.

Para asegurarse de que la RA 2 no sea el hijo de 12 años de Bob, el siguiente paso es que Alice pida a la RA 2 que pruebe su autenticidad. La respuesta a su consulta es un certificado firmado por la raíz que contiene la clave pública de la RA 2. Ahora Alice está segura de que tiene la clave pública de Bob.

Pero, ¿cómo averigua Alice la clave pública de la raíz? Magia. Se supone que todos conocen la clave pública de la raíz. Por ejemplo, su navegador podría haberse enviado con la clave pública de la raíz integrada en él.

Bob es una persona amigable y no desea causar a Alice tanto trabajo. Él sabe que ella va a tener que verificar la CA 5 y la RA 2, por lo que para ahorrarle algunos problemas, recolecta los dos certificados necesarios y se los proporciona junto con el de él. Ahora puede utilizar el conocimiento que tiene de la clave pública de la raíz para verificar el certificado de nivel superior y la clave pública contenida ahí para verificar la segunda. De esta manera, Alice no necesita contactar a nadie para realizar la verificación. Debido a que todos los certificados están firmados, Alice puede detectar con facilidad cualquier intento de alterar el contenido. Una cadena de certificados que va de esta forma a la raíz algunas veces se conoce como **cadena de confianza** o **ruta de certificación.** La técnica se utiliza ampliamente en la práctica.

Por supuesto, aún tenemos el problema de quién va a ejecutar la raíz. La solución no es tener una sola raíz, sino tener muchas, cada una con sus propias RAs y CAs. De hecho, los navegadores modernos vienen precargados con claves públicas para aproximadamente 100 raíces, algunas veces llamadas **anclas de confianza.** De esta forma, es posible evitar tener una sola autoridad mundial confiable.

Pero ahora queda el problema de en qué forma el fabricante del navegador decide cuáles anclas de confianza propuestas son confiables y cuáles no. Queda a criterio del usuario confiar en el fabricante del navegador para elegir las mejores opciones y no simplemente aprobar todas las anclas de confianza deseando pagar sus cuotas de inclusión. La mayoría de los navegadores permiten que los usuarios inspeccionen las claves de la raíz (por lo general en la forma de certificados firmados por la raíz) y eliminen las que parezcan sospechosas.

Directorios

Otro problema de cualquier PKI es en dónde están almacenados los certificados (y sus cadenas hacia un ancla de confianza conocida). Una posibilidad es hacer que cada usuario almacene sus propios certificados. Si bien esto es seguro (es decir, no hay forma de que los usuarios falsifiquen certificados firmados sin que esto se detecte), también es inconveniente. Una alternativa que se ha propuesto es utilizar DNS como un directorio de certificados. Antes de contactar a Bob, Alice probablemente tiene que buscar la dirección IP de Bob mediante DNS, entonces, ¿por qué no hacer que DNS devuelva toda la cadena de certificados de Bob junto con su dirección IP?

Algunas personas piensan que ésta es una forma de proceder, pero tal vez otras prefieren dedicar servidores de directorio cuyo único trabajo sea manejar los certificados X.509. Tales directorios podrían proporcionar servicios de búsqueda utilizando propiedades de los nombres X.500. Por ejemplo, en teoría un servicio de directorio como éste podría contestar una consulta como: “Dame una lista de todas las personas que tengan el nombre Alice y que trabajen en los departamentos de ventas en cualquier lugar de Estados Unidos o Canadá”. LDAP podría ser un candidato para almacenar esta información.

Revocación

El mundo real también está lleno de certificados, como los pasaportes y las licencias de conducir. Algunas veces estos certificados pueden revocarse, por ejemplo, las licencias de conducir pueden revocarse por conducir en estado de ebriedad y por otros delitos de manejo. En el mundo digital ocurre el mismo problema: el otorgante de un certificado podría decidir revocarlo porque la persona u organización que lo posee ha abusado de alguna manera. También puede revocarse si la clave privada del sujeto se ha expuesto o, peor aún, si la clave privada de la CA está en peligro. Por lo tanto, una PKI necesita tratar el problema de la revocación.

Un primer paso en esta dirección es hacer que cada CA emita periódicamente una **CRL** (**lista de revocación de certificados**) que proporcione los números seriales de todos los certificados que ha revocado. Puesto que los certificados contienen fechas de vencimiento, la CRL sólo necesita contener los números seriales de los certificados que no han expirado. Una vez que pasa la fecha de vencimiento de un certificado, éste se invalida de manera automática, por lo que no hay necesidad de hacer una distinción entre los certificados que han expirado y los que fueron revocados. Ninguno de esos tipos de certificados puede utilizarse.

Desgraciadamente, introducir CRLs significa que un usuario que está próximo a utilizar un certificado debe adquirir la CRL para ver si su certificado ha sido revocado. Si es así, dicho certificado no debe utilizarse. Sin embargo, si el certificado no está en la lista, pudo haber sido revocado justo después de que se publicó la lista. Por lo tanto, la única manera de estar seguro realmente es preguntar a la CA. Y la siguiente vez que se utilice ese mismo certificado, se le tiene que preguntar nuevamente a la CA, puesto que dicho certificado pudo haber sido revocado segundos antes.

Otra complicación es que un certificado revocado puede reinstalarse nuevamente, por ejemplo, si fue revocado por falta de pago, pero ahora se ha puesto al corriente. Tener que tratar con la revocación (y, posiblemente, con la reinstalación) elimina una de la mejores propiedades de los certificados, principalmente, que pueden utilizarse sin tener que contactar a una CA.

¿Dónde deben almacenarse las CRLs? Un buen lugar sería el mismo en el que se almacenan los certificados. Una estrategia es que una CA quite de manera activa y periódica CRLs y hacer que los directorios las procesen con sólo eliminar los certificados revocados. Si no se utilizan directorios para almacenar certificados, las CRLs pueden almacenarse en caché en varios lugares convenientes alrededor de la red. Puesto que una CRL es por sí misma un documento firmado, si se altera, esa alteración puede detectarse con facilidad.

Si los certificados tienen tiempos de vida largos, las CRLs también los tendrán. Por ejemplo, si las tarjetas de crédito son válidas durante cinco años, el número de revocaciones pendientes será mucho más grande que si se emitieran nuevas tarjetas cada tres meses. Una forma estándar para tratar con CRLs grandes es emitir una lista maestra ocasionalmente, pero emitir actualizaciones con más frecuencia. Hacer esto reduce el ancho de banda necesario para distribuir las CRLs.

8.6 SEGURIDAD EN LA COMUNICACIÓN

Hemos terminado nuestro estudio de las herramientas en cuestión. Ya cubrimos la mayoría de las técnicas y protocolos importantes. El resto del capítulo trata de cómo se utilizan estas técnicas en la práctica para proporcionar seguridad de red, más algunas reflexiones sobre los aspectos sociales de la seguridad, al final del capítulo.

En las siguientes cuatro secciones veremos la seguridad en la comunicación, es decir, cómo obtener los bits de manera secreta y sin modificación desde el origen hasta el destino y cómo mantener fuera a los bits no deseados. Éstos no son de ningún modo los únicos aspectos de seguridad en las redes, pero ciertamente están entre los más importantes, lo que hace de éste un buen lugar para comenzar.

8.6.1 IPsec

La IETF ha sabido por años que hay una falta de seguridad en Internet. Agregarla no era fácil pues surgió una controversia sobre dónde colocarla. La mayoría de los expertos en seguridad creían que para estar realmente seguro, el cifrado y las verificaciones de integridad tenían que llevarse a cabo de extremo a extremo (es decir, en la capa de aplicación). De tal manera, el proceso de origen encripta y/o protege la integridad de los datos y los envía al proceso de destino en donde se desencriptan y/o verifican. Por lo tanto, cualquier alteración hecha en medio de estos dos procesos, o en cualquier sistema operativo, puede detectarse. El problema con este enfoque es que requiere cambiar todas las aplicaciones para que estén conscientes de la seguridad. Desde esta perspectiva, el siguiente mejor enfoque es colocar el cifrado en la capa de transporte o en una nueva capa entre la capa de aplicación y la de transporte, con lo que se conserva el enfoque de extremo a extremo pero no requiere que se cambien las aplicaciones.

La perspectiva opuesta es que los usuarios no entiendan la seguridad y no sean capaces de utilizarla correctamente, así como que nadie desee modificar los programas existentes de ninguna forma, por lo que la capa de red debe autentificar y/o cifrar paquetes sin que los usuarios estén involucrados. Después de años de batallas encarnizadas, esta perspectiva ganó soporte suficiente para que se definiera un estándar de seguridad de capa de red. El argumento fue en parte que tener cifrado de la capa de red no evitaba que los usuarios conscientes de la seguridad la aplicaran correctamente y que ayudaba hasta cierto punto a los usuarios no conscientes de ella.

El resultado de esta guerra fue un diseño llamado **IPsec** (**Seguridad IP**), que se describe en los RFCs 2401, 2402 y 2406, entre otros. No todos los usuarios desean cifrado (pues éste es costoso computacionalmente). En lugar de hacerlo opcional, se decidió requerir cifrado todo el tiempo pero permitir el uso de un algoritmo nulo. Éste se describe y alaba por su simplicidad, facilidad de implementación y gran velocidad en el RFC 2410.

El diseño IPsec completo es una estructura para servicios, algoritmos y granularidades múltiples. La razón para los servicios múltiples es que no todas las personas quieren pagar el precio por tener todos los servicios todo el tiempo, por lo que los servicios están disponibles a la carta. Los servicios principales son confidencialidad, integridad de datos y protección contra ataques de

repetición (un intruso repite una conversación). Todos estos se basan en criptografía simétrica debido a que el alto rendimiento es crucial.

La razón de tener múltiples algoritmos es que un algoritmo que ahora se piensa es seguro puede ser violado en el futuro. Al hacer independiente al algoritmo IPsec, la estructura puede sobrevivir incluso si algún algoritmo particular es violado posteriormente.

La razón de tener múltiples granularidades es para hacer posible la protección de una sola conexión TCP, todo el tráfico entre un par de *hosts* o todo el tráfico entre un par de enrutadores seguros, entre otras posibilidades.

Un aspecto ligeramente sorprendente de IPsec es que aunque se encuentra en la capa IP, es orientado a la conexión. En la actualidad, eso no es tan sorprendente porque para tener seguridad, se debe establecer y utilizar una clave por algún periodo —en esencia, un tipo de conexión. Además, las conexiones amortizan los costos de configuración sobre muchos paquetes. Una "conexión" en el contexto de IPsec se conoce como **SA** (**asociación de seguridad).** Una SA es una conexión símplex entre dos puntos finales y tiene un identificador de seguridad asociado con ella. Si se necesita tráfico seguro en ambas direcciones, se requieren dos asociaciones de seguridad. Los identificadores de seguridad se transportan en paquetes que viajan en estas conexiones seguras y se utilizan para buscar claves y otra información relevante cuando llega un paquete seguro.

Técnicamente, IPsec tiene dos partes principales. La primera describe dos encabezados nuevos que pueden agregarse a paquetes para transportar el identificador de seguridad, datos de control de integridad, entre otra información. La otra parte, **ISAKMP** (**Asociación para Seguridad en Internet y Protocolo de Administración de Claves),** tiene que ver con el establecimiento de claves. No trataremos con mayor detalle a ISAKMP porque (1) es extremadamente complejo y (2) su protocolo principal, **IKE** (**Intercambio de Claves de Internet),** está plagado de fallas y necesita ser reemplazado (Perlman y Kaufman, 2000).

IPsec puede utilizarse en cualquiera de dos modos. En el **modo de transporte,** el encabezado IPsec se inserta justo después del encabezado IP. El campo *Protocolo* del encabezado IP se cambia para indicar que un encabezado IPsec sigue al encabezado IP normal (antes del encabezado TCP). El encabezado IPsec contiene información de seguridad, principalmente el identificador SA, un nuevo número de secuencia y tal vez una verificación de integridad del campo de carga.

En el **modo de túnel,** todo el paquete IP, encabezado y demás, se encapsula en el cuerpo de un paquete IP nuevo con un encabezado IP completamente nuevo. El modo de túnel es útil cuando un túnel termina en una ubicación que no sea el destino final. En algunos casos, el final del túnel es una máquina de puerta de enlace de seguridad, por ejemplo, un *firewall* de una empresa. En este modo, el *firewall* encapsula y desencapsula paquetes conforme pasan a través del *firewall*. Al terminar el túnel en esta máquina segura, las máquinas en la LAN de la empresa no tienen que estar consciente de IPsec. Sólo el *firewall* tiene que saber sobre él.

El modo de túnel también es útil cuando se agrega un conjunto de conexiones TCP y se maneja como un solo flujo cifrado porque así se evita que in intruso vea quién está enviando cuántos paquetes a quién. Algunas veces el simple hecho de saber cuánto tráfico está pasando y hacia dónde se dirige es información valiosa. Por ejemplo, si durante una crisis militar, la cantidad de tráfico que fluye entre el Pentágono y la Casa Blanca se reduce de manera significativa, pero la cantidad de tráfico entre el Pentágono y alguna instalación militar oculta entre las Montañas Rocosas de

Colorado se incrementa en la misma cantidad, un intruso podría ser capaz de deducir alguna información útil a partir de estos datos. El estudio de los patrones de flujo de paquetes, aunque estén cifrados, se conoce como **análisis de tráfico.** El modo de túnel proporciona una forma de frustrarlo hasta cierto punto. La desventaja del modo de túnel es que agrega un encabezado IP extra, por lo que se incrementa el tamaño del paquete en forma sustancial. En contraste, el modo de transporte no afecta tanto el tamaño del paquete.

El primer nuevo encabezado es **AH** (**encabezado de autenticación**)**.** Proporciona verificación de integridad y seguridad antirrepetición, pero no la confidencialidad (es decir, no cifrado de datos). El uso de AH en el modo de transporte se ilustra en la figura 8-27. En el IPv4 se coloca entre el encabezado IP (incluyendo cualquier opción) y el TCP. En IPv6 es sólo otro encabezado de extensión y se trata como tal. De hecho, el formato está cerca del de un encabezado de extensión IPv6 estándar. La carga útil tal vez tenga que rellenarse a alguna longitud en particular para el algoritmo de autenticación, como se muestra.

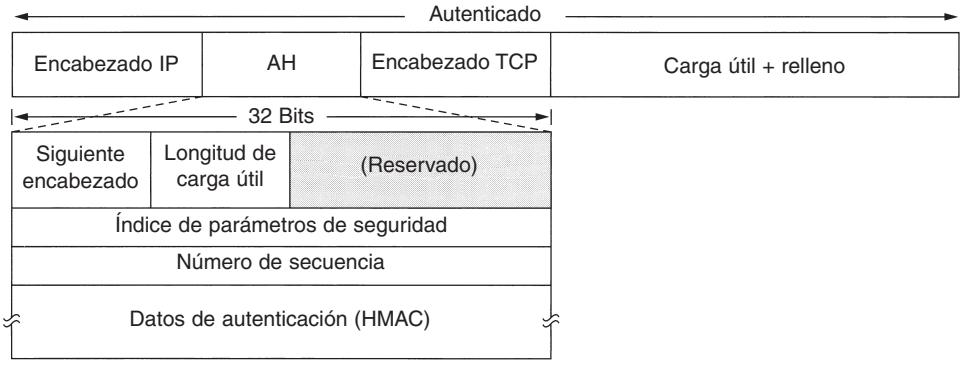


Figura 8-27. El encabezado de autenticación IPsec en el modo de transporte para IPv4.

Examinemos el encabezado AH. El campo *Siguiente encabezado* se utiliza para almacenar el valor anterior que tenía el campo *Protocolo* IP antes de que se reemplazara con 51 para indicar que seguía un encabezado AH. En muchos casos, el código para TCP (6) irá aquí. La *Longitud de carga útil* es el número de palabras de 32 bits en el encabezado AH menos 2.

El *Índice de parámetros de seguridad* es el indicador de conexión. Es insertado por el emisor para indicar un registro en particular en la base de datos del receptor. Este registro contiene la clave compartida utilizada en esta conexión y otra información sobre dicha conexión. Si este protocolo hubiera sido inventado por la ITU en lugar del IETF, este campo se hubiera llamado *Número de circuitos virtuales*.

El campo *Número de secuencia* se utiliza para numerar todos los paquetes enviados en una SA. Cada paquete obtiene un número único, incluso las retransmisiones. En otras palabras, la retransmisión de un paquete obtiene un número diferente aquí que el original (aunque su número de secuencia TCP sea el mismo). El propósito de este campo es detectar ataques de repetición. Tal vez estos números de secuencia no se ajusten. Si todos los 2^{32} se agotan, debe establecerse una nueva SA para continuar la comunicación.

Por último, veamos el campo *Datos de autenticación*, que es de longitud variable y contiene la firma digital de la carga útil. Cuando se establece la SA, los dos lados negocian cuál algoritmo de firmas van a utilizar. Por lo general, aquí no se utiliza la criptografía de clave pública porque los paquetes se deben procesar extremadamente rápido y todos los algoritmos de clave pública conocidos son muy lentos. Puesto que IPsec se basa en la criptografía de clave simétrica y el emisor y el receptor negocian una clave compartida antes de establecer una SA, dicha clave compartida se utiliza en el cálculo de la firma. Una forma simple es calcular el *hash* sobre el paquete más la clave compartida. Por supuesto, ésta no se transmite. Un esquema como éste se conoce como **HMAC (Código de Autenticación de Mensajes basado en *Hash*).** Es más rápido realizar el cálculo que primero ejecutar el SHA-1 y luego el RSA sobre el resultado.

El encabezado AH no permite la encriptación de los datos, por lo que es útil principalmente cuando la verificación de la integridad es necesaria pero la confidencialidad no lo es. Una característica de AH que vale la pena es que la verificación de integridad abarca algunos de los campos en el encabezado IP, principalmente aquellos que no cambian conforme el paquete se mueve de enrutador a enrutador. El campo *Tiempo de vida* cambia en cada salto, por ejemplo, de manera que no se puede incluir en la verificación de integridad. Sin embargo, la dirección IP de origen se incluye en la verificación, lo que hace imposible que un intruso falsifique el origen de un paquete.

El encabezado IPsec alternativo es **ESP (Carga Útil de Encapsulamiento de Seguridad).** Su uso para modo de transporte y para modo de túnel se muestra en la figura 8-28.

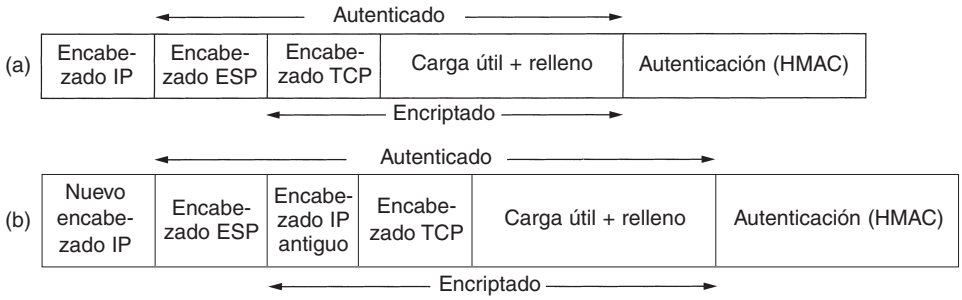


Figura 8-28. (a) ESP en modo de transporte. (b) ESP en modo de túnel.

El encabezado ESP consiste en dos palabras de 32 bits. Éstas son los campos *Índice de parámetros de seguridad* y *Número de secuencia* que vimos en AH. Una tercera palabra que generalmente va después de ellos (pero que técnicamente no es parte del encabezado) es el *Vector de inicialización* utilizado para la encriptación de datos, a menos que se utilice la encriptación nula, en cuyo caso se omite.

ESP también incluye verificaciones de integridad HMAC, al igual que AH, pero en lugar de incluirse en el encabezado, van después de la carga útil, como se muestra en la figura 8-28. Colocar el HMAC al final tiene una ventaja en una implementación de hardware. El HMAC se puede calcular conforme los bits se transmiten por la interfaz de red y agregarse al final. Ésta es la razón por la cual Ethernet y otras LANs tienen sus CRCs en un terminador, en lugar de en un encabezado. Con AH, el paquete tiene que almacenarse en el búfer y la firma tiene que calcularse antes

de que se envíe el paquete, con lo que se reduce de manera potencial el número de paquetes/seg que pueden enviarse.

Puesto que ESP puede hacer lo mismo que AH y más, y debido a que es más eficiente para iniciar, surge la pregunta: ¿Por qué molestarse en tener a AH? La respuesta es principalmente histórica. En un principio, AH manejaba sólo integridad y ESP manejaba sólo confidencialidad. Más tarde, la integridad se agregó a ESP, pero las personas que diseñaron AH no querían dejarlo morir después de todo el trabajo que habían realizado. Sin embargo, su único argumento es que AH verifica parte del encabezado IP, lo cual ESP no hace, pero es un argumento débil. Otro argumento débil es que un producto que soporta AH y no ESP podría tener menos problemas para obtener una licencia de exportación porque no puede realizar encriptación. Es probable que AH sea desplazado en el futuro.

8.6.2 *Firewalls*

La capacidad de conectar una computadora, en cualquier lugar, con cualquier computadora, de cualquier lugar, es una ventaja a medias. Para los usuarios domésticos, navegar en Internet significa mucha diversión. Para los gerentes de seguridad empresarial, es una pesadilla. Muchas empresas tienen en línea grandes cantidades de información confidencial —secretos de comercio, planes de desarrollo de productos, estrategias de marketing, análisis financieros, etcétera. Si esta información cae en manos de un competidor podría tener graves consecuencias.

Además del peligro de la fuga de información, también existe el peligro de la infiltración de información. En particular, virus, gusanos y otras pestes digitales pueden abrir brechas de seguridad, destruir datos valiosos y hacer que los administradores pierdan mucho tiempo tratando de arreglar el daño que hayan hecho. Por lo general, son traídos por empleados descuidados que desean ejecutar algún nuevo juego ingenioso.

En consecuencia, se necesitan mecanismos para mantener adentro a los bits "buenos" y afuera a los bits "malos". Un método es utilizar IPsec. Este método protege a los datos en tránsito entre los sitios seguros. Sin embargo, IPsec no hace nada para mantener afuera de la LAN de la compañía a las pestes digitales y a los intrusos. Para saber cómo alcanzar ese objetivo, necesitamos ver los *firewalls*.

Los ***firewalls*** (servidores de seguridad) son simplemente una adaptación moderna de la vieja estrategia medieval de seguridad: excavar un foso defensivo profundo alrededor de su castillo. Este diseño obligaba a que todos los que entraran o salieran del castillo pasaran a través de un puente levadizo, en donde los encargados de la E/S los podían inspeccionar. En las redes es posible el mismo truco: una compañía puede tener muchas LANs conectadas de formas arbitrarias, pero se obliga a que todo el tráfico desde o hacia la compañía pase a través de un puente levadizo electrónico (*firewall*), como se muestra en la figura 8-29.

En esta configuración el *firewall* tiene dos componentes: dos enrutadores que realizan filtrado de paquetes y una puerta de enlace de aplicación. También existen configuraciones más simples, pero la ventaja de este diseño es que cada paquete debe transitar a través de dos filtros y una puerta de enlace de aplicación para entrar o salir. No existe otra ruta. Es evidente que quienes piensen que un punto de verificación de seguridad es suficiente, no han hecho vuelos internacionales recientemente en alguna aerolínea.

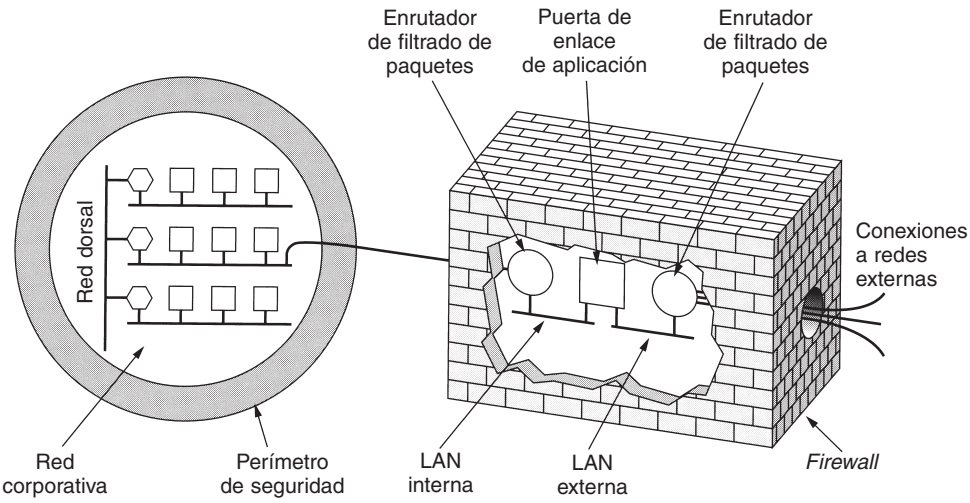


Figura 8-29. Un firewall que consiste en dos filtros de paquetes y en una puerta de enlace de aplicación.

Cada **filtro de paquete** es un enrutador estándar equipado con alguna funcionalidad extra. Ésta permite inspeccionar cada paquete entrante o saliente. Los paquetes que cumplan con algún criterio se reenvían de manera normal. Los que fallen la prueba se descartan.

En la figura 8-29 probablemente el filtro de paquete de la LAN interna verifica los paquetes salientes y el que está en la LAN externa verifica los paquetes entrantes. Los paquetes que cruzan la primera barrera pasan a la puerta de enlace de aplicación, donde se vuelven a examinar. El objetivo de colocar los dos filtros de paquetes en LANs diferentes es asegurar que ningún paquete entre o salga sin pasar a través de la puerta de enlace de aplicación: no hay otra ruta.

Por lo general, los filtros de paquetes son manejados por tablas configuradas por el administrador del sistema. Dichas tablas listan orígenes y destinos aceptables, orígenes y destinos bloqueados, y reglas predeterminadas sobre lo que se debe hacer con los paquetes que van o vienen de otras máquinas.

En el caso común de una configuración TCP/IP, un origen o un destino consiste en una dirección IP y un puerto. Los puertos indican qué servicio se desea. Por ejemplo, el puerto TCP 23 es para telnet, el puerto TCP 79 es para directorio y el puerto TCP 119 es para las noticias USENET. Una compañía podría bloquear los paquetes entrantes de todas las direcciones IP combinadas con uno de estos puertos. De esta forma, nadie afuera de la compañía puede iniciar una sesión a través de telnet o buscar personas mediante el demonio de directorio. Además, la compañía podría estar en contra de que sus empleados desperdicien todo el día leyendo noticias USENET.

Bloquear paquetes salientes es un poco difícil pues aunque la mayoría de los sitios se apegan a las convenciones estándar de numeración de puertos, no están obligados a hacerlo. Además, para algunos servicios importantes, como FTP (Protocolo de Transferencia de Archivos), los números de puerto se asignan de manera dinámica. Asimismo, aunque el bloqueo de conexiones TCP es difícil, el de paquetes UDP lo es aún más debido a que se sabe muy poco con anticipación sobre lo

que harán. Muchos filtros de paquetes están configurados para simplemente prohibir el tráfico de paquetes UDP.

La segunda mitad del *firewall* es la **puerta de enlace de aplicación.** En lugar de sólo buscar los paquetes simples, la puerta de enlace opera a nivel de aplicación. Por ejemplo, es posible configurar una puerta de enlace de correo para examinar cada mensaje que sale o entra. Para cada uno, la puerta de enlace decide si transmitir o descartar el mensaje con base en los campos de encabezado, el tamaño del mensaje o incluso en el contenido (por ejemplo, en una instalación militar, la presencia de palabras como "nuclear" o "bomba" podría causar que se tomara alguna acción especial).

Las instalaciones son libres de configurar una o más puertas de enlace de aplicación para aplicaciones específicas, aunque no es poco común que organizaciones desconfiadas permitan correo entrante y saliente, e incluso el acceso a World Wide Web, pero que consideren todo lo demás muy peligroso. Combinado con la encriptación y el filtrado de paquetes, este arreglo ofrece una cantidad limitada de seguridad con el costo de algunas inconveniencias.

Incluso si el *firewall* está configurado perfectamente, aún existirá una gran cantidad de problemas. Por ejemplo, si un *firewall* está configurado para permitir sólo paquetes entrantes de redes específicas (por ejemplo, de otras instalaciones de la compañía), un intruso afuera del *firewall* puede introducir direcciones de origen falsas para evadir esta verificación. Si un miembro interno desea enviar documentos secretos, puede encriptarlos o incluso fotografiarlos y enviar las fotos como archivos JPEG, los cuales pueden evadir cualquier filtro de palabras. Y no hemos analizado el hecho de que el 70% de todos los ataques provienen del lado interno del *firewall*, por ejemplo, de empleados descontentos (Schneier, 2000).

Además, hay otra clase de ataques que los *firewalls* no pueden manejar. La idea básica de un *firewall* es evitar que entren intrusos y que salga información secreta. Desgraciadamente, hay personas que no tienen nada mejor que hacer que tratar de perjudicar ciertos sitios. Esto lo hacen enviando grandes cantidades de paquetes legítimos al destino, hasta que el sitio se caiga debido a la carga. Por ejemplo, para derribar un sitio Web, un intruso puede enviar un paquete TCP *SYN* para establecer una conexión. A continuación el sitio asignará una ranura de tabla para la conexión y enviará como respuesta un paquete *SYN + ACK*. Si el intruso no responde, la ranura de tabla se conservará durante algunos segundos hasta que expire. Si el intruso envía miles de solicitudes de conexión, todas las ranuras de tabla se llenarán y no podrá establecerse ninguna conexión legítima. Los ataques en los que el objetivo del intruso es bloquear el destino en lugar de robar datos se conocen como ataques **DoS (negación de servicio).** Por lo general, los paquetes de solicitud tienen direcciones falsas de origen por lo que el intruso no puede ser rastreado con facilidad.

Una variante aún peor es aquella en la que el intruso ha entrado en cientos de computadoras en cualquier parte del mundo, y después ordena a todas ellas que ataquen al mismo objetivo al mismo tiempo. Este método no sólo incrementa el poder del intruso, también reduce la probabilidad de su detección, debido a que los paquetes provienen de una gran cantidad de máquinas que pertenecen a usuarios ingenuos. Un ataque de este tipo se conoce como ataque **DDoS (negación de servicio distribuida).** Es difícil defenderse de un ataque como éste. Incluso si la máquina atacada puede reconocer rápidamente una solicitud falsa, le toma algún tiempo procesar y descartar la

solicitud, y si llegan suficientes solicitudes por segundo, la CPU pasará todo su tiempo ocupándose de ellas.

8.6.3 Redes privadas virtuales

Muchas compañías tienen oficinas e instalaciones esparcidas en muchas ciudades, algunas veces en múltiples países. En el pasado, antes de que existieran las redes de datos públicas, era común que algunas compañías alquilaran líneas a las compañías telefónicas entre todas o entre sólo algunas ubicaciones. Algunas compañías aún hacen esto. Una red constituida por computadoras de compañías y líneas telefónicas alquiladas se conoce como **red privada.** En la figura 8-30(a) se muestra una red privada de ejemplo que conecta tres ubicaciones.

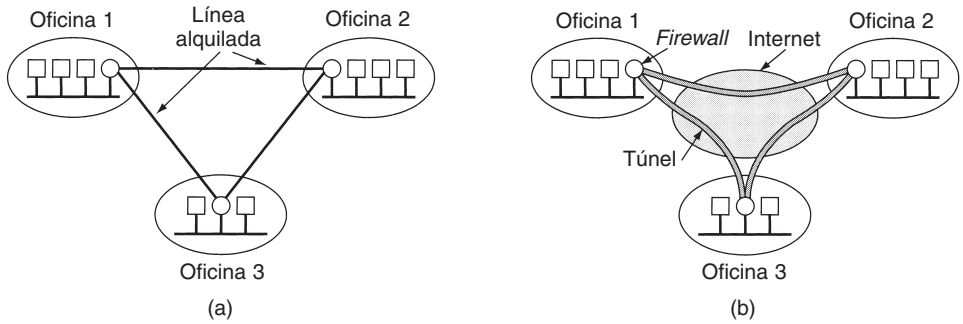


Figura 8-30. (a) Una red privada con línea alquilada. (b) Una red privada virtual.

Las redes privadas funcionan bien y son muy seguras. Si las únicas líneas disponibles son las alquiladas, el tráfico no puede fugarse de las ubicaciones de la compañía y los intrusos tienen que intervenir físicamente las líneas para infiltrarse, lo cual no es fácil de hacer. El problema con las redes privadas es que alquilar una sola línea T1 cuesta miles de dólares mensuales y las líneas T3 son muchas veces más costosas. Cuando aparecieron las redes de datos públicas y, más tarde, Internet, muchas compañías quisieron trasladar su tráfico de datos (y, posiblemente, de voz) a la red pública, aunque sin renunciar a la seguridad de la red privada.

Esta demanda pronto llevó a la invención de las **VPNs (redes privadas virtuales),** que son redes superpuestas sobre redes públicas pero con muchas propiedades de las redes privadas. Se llaman "virtuales" porque son sólo una ilusión, al igual que los circuitos virtuales no son circuitos reales ni la memoria virtual es memoria real.

Aunque las VPNs pueden implementarse encima de ATM (o de Frame Relay), un método cada vez más popular es construir VPNs directamente sobre Internet. Un diseño común es equipar cada oficina con un *firewall* y crear túneles a través de Internet entre todos los pares de oficinas,

como se ilustra en la figura 8-30(b). Si IPsec se utilizara para el proceso de entunelamiento, entonces sería posible agregar todo el tráfico entre cualquiera de los dos pares de oficinas en una sola SA encriptada y autenticada, con lo que se proporcionaría control de integridad, confidencialidad e incluso inmunidad considerable al análisis de tráfico.

Cuando se inicia el sistema, cada par de *firewalls* tiene que negociar los parámetros de su SA, incluyendo los servicios, modos, algoritmos y claves. Muchos *firewalls* tienen capacidades VPN integradas, aunque algunos enrutadores ordinarios también pueden hacer esto. Pero debido a que los *firewalls* están principalmente en el negocio de la seguridad, es natural que los túneles empiecen y terminen en los *firewalls*, estableciendo una clara separación entre la compañía e Internet. Por lo tanto, los *firewalls*, las VPNs e IPsec con ESP en modo de túnel son una combinación natural y se utilizan ampliamente en la práctica.

Una vez que se han establecido las SAs, el tráfico puede comenzar a fluir. Para un enrutador en Internet, un paquete que viaja a través de un túnel VPN es sólo un paquete ordinario. Lo único extraño es la presencia del encabezado IPsec después del encabezado IP, pero debido a que estos encabezados adicionales no tienen efecto en el proceso de reenvío, los enrutadores no se preocupan por ellos.

Una ventaja principal de organizar de esta forma una VPN es que es completamente transparente para todo el software de usuario. Los *firewalls* configuran y manejan las SAs. La única persona que está consciente de esta configuración es el administrador del sistema, quien tiene que configurar y manejar los *firewalls*. Para todos los demás, es como tener nuevamente una red privada mediante una línea alquilada. Para mayor información sobre las VPNs, vea (Brown, 1999, e Izzo, 2000).

8.6.4 Seguridad inalámbrica

Diseñar un sistema que sea lógica y completamente seguro mediante VPNs y *firewalls* es muy fácil, pero eso, en la práctica, puede fallar. Esta situación puede darse si algunas de las máquinas son inalámbricas y utilizan comunicación de radio, que pase justo encima del *firewall* en ambas direcciones. El rango de las redes 802.11 con frecuencia es de algunos cientos de metros, por lo que cualquiera que desee espiar una compañía puede simplemente introducirse en el estacionamiento para empleados por la mañana, dejar una computadora portátil habilitada para 802.11 en el automóvil para que grabe todo lo que oiga. Al anochecer, el disco duro estará repleto de datos valiosos. En teoría, se supone que esta fuga no debería suceder. Pero, en teoría, las personas tampoco deberían robar bancos.

La mayor parte del problema de seguridad puede remontarse a los fabricantes de las estaciones base inalámbricas (puntos de acceso), quienes tratan de hacer que sus productos sean amigables para el usuario. Por lo general, si el usuario saca el dispositivo de la caja y lo conecta en el enchufe de la energía eléctrica, comienza a operar inmediatamente —casi siempre sin seguridad, revelando secretos a quienes estén dentro del rango de radio. Si a continuación se conecta a una Ethernet, de pronto todo el tráfico de ésta aparecerá también en el estacionamiento. La tecnología

inalámbrica es el sueño de todo espía: datos gratuitos sin tener que hacer nada. Por lo tanto, sobra decir que la seguridad es mucho más importante para los sistemas inalámbricos que para los cableados. En esta sección veremos algunas formas en que las redes inalámbricas manejan la seguridad. Es posible encontrar información adicional en (Nichols y Lekkas, 2002).

Seguridad del 802.11

El estándar 802.11 establece un protocolo de seguridad en el nivel de capa de enlace de datos llamado **WEP (Privacidad Inalámbrica Equivalente),** diseñado para que la seguridad de una LAN inalámbrica sea tan buena como la de una LAN cableada. Puesto que lo predeterminado para las LANs alámbricas no es la seguridad, este objetivo es fácil de alcanzar, y WEP lo consigue, como veremos más adelante.

Cuando se habilita la seguridad para el estándar 802.11, cada estación tiene una clave secreta que comparte con la estación base. La forma en que se distribuyen las claves no se especifica en el estándar. Éstas sólo pueden ser precargadas por el fabricante. Pueden intercambiarse por adelantado a través de la red alámbrica. Por último, la estación base o la máquina del usuario pueden tomar una clave aleatoria y enviársela al otro por aire encriptada con la clave pública del otro. Una vez establecidas, por lo general las claves permanecen estables por meses o años.

La encriptación WEP utiliza un cifrado de flujo con base en el algoritmo RC4. Éste fue diseñado por Ronald Rivest y se mantuvo en secreto hasta que fue filtrado y se publicó en Internet en 1994. Como señalamos anteriormente, es casi imposible mantener en secreto los algoritmos, incluso cuando el objetivo es proteger la propiedad intelectual (como fue en este caso) en lugar de la seguridad gracias al anonimato (que no era el objetivo de RC4). En WEP, RC4 genera un flujo de claves al cual se le aplica un OR exclusivo con el texto llano para dar lugar al texto cifrado.

La carga útil de cada paquete se encripta a través del método de la figura 8-31. Primero se realiza una suma de verificación de la carga útil utilizando el CRC-32 polinomial y la suma de verificación se agrega a la carga útil para formar el texto llano para el algoritmo de encriptación. A continuación, a este texto llano se le aplica un OR exclusivo con un fragmento de flujo de claves de su propio tamaño. El resultado es el texto cifrado. El IV utilizado para iniciar el RC4 se envía junto con el texto cifrado. Cuando el receptor obtiene el paquete, extrae la carga útil de él, genera el flujo de claves a partir de la clave secreta compartida y el IV que acaba de recibir, y aplica un OR exclusivo al flujo de claves con la carga útil para recuperar el texto llano. A continuación puede comprobar la suma de verificación para saber si el paquete fue modificado.

Aunque esta estrategia parece buena a primera vista, se ha publicado un método para violarla (Borisov y cols., 2001). A continuación resumiremos sus resultados. Primero que nada, sorprendentemente muchas instalaciones utilizan la misma clave compartida para todos los usuarios, en cuyo caso cada usuario puede leer todo el tráfico de los demás usuarios. Esto ciertamente es equivalente a Ethernet, pero no es muy seguro.

Pero incluso si cada usuario tiene una clave distinta, WEP aún puede ser atacado. Puesto que por lo general las claves son estables por largos periodos, el estándar WEP recomienda (no obliga a) que el IV se cambie en cada paquete para evitar los ataques de reutilización de flujo de claves

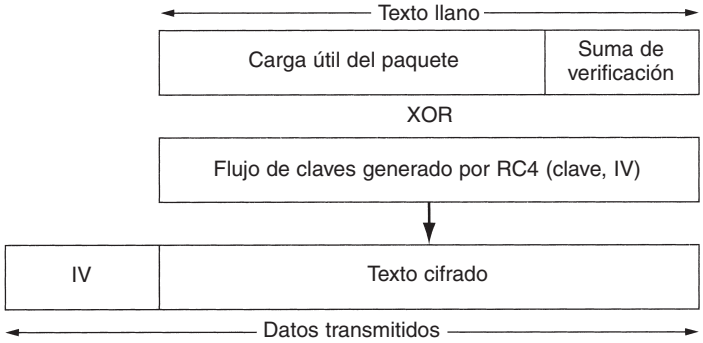


Figura 8-31. Encriptación de paquetes mediante WEP.

que analizamos en la sección 8.2.3. Desgraciadamente, muchas tarjetas 802.11 para computadoras portátiles restablecen el IV a 0 cuando la tarjeta se introduce en la computadora, y lo incrementan en uno por cada paquete enviado. Puesto que las personas remueven e insertan con frecuencia estas tarjetas, son comunes los paquetes con valores bajos de IV. Si Trudy puede coleccionar varios paquetes enviados por el mismo usuario con el mismo valor de IV (que también es enviado en texto llano junto con cada paquete), puede calcular el OR exclusivo de dos valores de texto llano y probablemente romper el cifrado de esa forma.

Pero incluso si la tarjeta 802.11 elige un IV aleatorio para cada paquete, los IVs son de sólo 24 bits, por lo que después de que se hayan enviado 2^{24} paquetes, tienen que reutilizarse los IVs. Peor aún, con los IVs elegidos de manera aleatoria, la cantidad esperada de paquetes que tiene que enviarse antes de que se utilice dos veces el mismo IV es de aproximadamente 5000, debido al ataque de cumpleaños descrito en la sección 8.4.4. Por lo tanto, si Trudy escucha por algunos minutos, es casi seguro que atrape dos paquetes con el mismo IV y la misma clave. Al aplicar un OR exclusivo a los textos cifrados, puede obtener el OR exclusivo de los textos llanos. Esta secuencia de bits puede ser atacada de varias formas para recuperar los textos llanos. Con algo más de trabajo, también puede obtenerse el flujo de claves para ese IV. Trudy puede continuar trabajando de esta manera por un tiempo y compilar un diccionario de flujo de claves para varios IVs. Una vez que se ha descifrado un IV, se pueden desencriptar por completo todos los paquetes que se envíen con él en el futuro (aunque también en el pasado).

Además, puesto que los IVs se utilizan de manera aleatoria, una vez que Trudy ha determinado un par válido (IV, flujo de claves), puede emplearlo para generar todos los paquetes que desee que lo utilicen y, por lo tanto, interferir activamente la comunicación. En teoría, un receptor podría notar que de repente grandes cantidades de paquetes tienen el mismo IV, pero (1) WEP permite esto, y (2) nadie lo verifica.

Por último, el CRC no es de mucha ayuda, puesto que es posible que Trudy cambie la carga útil y haga el cambio correspondiente al CRC, incluso sin tener que eliminar la encriptación. En resumen, es muy sencillo violar la seguridad del 802.11, y no hemos listado todos los ataques que encontraron Borisov y cols.

En agosto de 2001, un mes después de que se presentó el trabajo de Borisov y cols., se publicó otro ataque devastador contra WEP (Fluhrer y cols., 2001). Éste encontró debilidades criptográficas en el RC4 mismo. Fluhrer y cols., descubrieron que muchas de las claves tienen la propiedad de que es posible derivar algunos bits de las claves a partir del flujo de claves. Si este ataque se realiza de manera repetida, es posible deducir toda la clave con un mínimo de esfuerzo. Como el enfoque de su investigación era teórico, Fluhrer y cols., no trataron de romper ninguna LAN 802.11 en la práctica.

En contraste, cuando un estudiante y dos investigadores de los Laboratorios AT&T supieron sobre el ataque de Fluhrer y cols., decidieron llevarlo a la práctica (Stubblefield y cols., 2002). En una semana habían descifrado su primera clave de 128 bits de una LAN 802.11 en funcionamiento, y la mayor parte de esa semana la dedicaron realmente a buscar la tarjeta 802.11 más barata, obtener el permiso para comprarla, instalarla y probarla. La programación tardó sólo dos horas.

Cuando anunciaron sus resultados, la CNN publicó un reportaje en el que algunos gurús de la industria trataron de menospreciar sus resultados afirmando que lo que habían hecho era trivial, pues habían tomado como base el trabajo de Fluhrer y cols. Si bien ese comentario es técnicamente cierto, lo relevante es que los esfuerzos combinados de estos dos equipos demostraron un defecto fatal en WEP y el 802.11.

El 7 de septiembre de 2001, el IEEE respondió al hecho de que WEP se había roto por completo emitiendo una corta declaración en la que señalaba seis puntos que pueden resumirse de la siguiente manera:

1. Les dijimos que la seguridad de WEP no era mejor que la de Ethernet.
2. Es mucho peor olvidarse de establecer alguna clase de seguridad.
3. Traten de utilizar otro tipo de seguridad (por ejemplo, seguridad en la capa de transporte).
4. La siguiente versión, 802.11i, tendrá mejor seguridad.
5. La certificación futura requerirá el uso del 802.11i.
6. Trataremos de determinar qué hacer en tanto llega el 802.11i.

Hemos analizado detenidamente esta historia para resaltar el hecho de que no es sencillo conseguir la seguridad correcta, incluso para los expertos.

Seguridad de Bluetooth

Bluetooth tiene un rango considerablemente más corto que el 802.11, por lo que no puede atacarse desde un estacionamiento, pero la seguridad sigue siendo un problema aquí. Por ejemplo, imagine que la computadora de Alice está equipada con un teclado inalámbrico Bluetooth. En un escenario donde no hubiera seguridad, si Trudy se encontrara en la oficina adyacente, podría leer todo lo que Alice escribiera, incluyendo todo su correo electrónico saliente. También podría capturar todo lo que la computadora de Alice enviara a la impresora Bluetooth instalada junto a ella (por ejemplo, correo electrónico entrante e informes confidenciales). Por fortuna, Bluetooth tiene

un complejo esquema de seguridad para tratar de que todas las Trudies del mundo fracasen. A continuación resumiremos sus características principales.

Bluetooth tiene tres modos de seguridad, que van desde ninguna seguridad en absoluto hasta encriptación completa de datos y control de integridad. Al igual que con el 802.11, si se deshabilita la seguridad (lo predeterminado), no hay seguridad. La mayoría de los usuarios mantiene deshabilitada la seguridad hasta que ocurre una brecha de seguridad; entonces es cuando la habilitan. Este enfoque se parece al dicho "después del niño ahogado, pozo tapado".

Bluetooth proporciona seguridad en múltiples capas. En la capa física, los saltos de frecuencia proporcionan un poco de seguridad, pero debido a que es necesario indicar a cualquier dispositivo Bluetooth de una *piconet* la secuencia de saltos de frecuencia, esta secuencia obviamente no es un secreto. La seguridad real inicia cuando el esclavo recién llegado pide un canal al maestro. Se da por hecho que los dos dispositivos comparten una clave secreta establecida con anticipación. En algunos casos, el fabricante es quien las incluye (por ejemplo, para un teléfono móvil con auriculares vendidos como una sola unidad). En otros casos, un dispositivo (por ejemplo, los auriculares) tiene una clave integrada y el usuario tiene que introducir esa clave en el otro dispositivo (por ejemplo, el teléfono móvil) como un número decimal. Estas claves compartidas se conocen como **claves maestras.**

Para establecer un canal, tanto el esclavo como el maestro verifican si el otro conoce la clave maestra. De ser así, negocian si ese canal será encriptado, si se va a controlar su integridad, o ambas cosas. Después pueden seleccionar una clave de sesión aleatoria de 128 bits, de los cuales algunos pueden ser públicos. El objetivo de permitir esta debilidad de clave es respetar las restricciones gubernamentales de varios países diseñadas para evitar la exportación o el uso de claves más grandes de lo que el gobierno puede romper.

La encriptación utiliza un cifrado de flujo llamado $\mathbf{E_0}$**;** el control de integridad utiliza **SAFER+.** Ambos son cifrados en bloque de clave simétrica tradicionales. SAFER+ fue emitido para el AES *bake-off*, pero se eliminó en la primera ronda porque era más lento que los otros candidatos. Bluetooth se terminó antes de que se eligiera el cifrado AES; de lo contrario éste hubiera utilizado probablemente Rijndael.

En la figura 8-14 se muestra la encriptación real que utiliza el cifrado de flujo, con el texto llano al cual se le aplica OR exclusivo con el flujo de claves para generar el texto cifrado. Desafortunadamente, E_0 mismo (al igual que RC4) podría tener debilidades fatales (Jakobsson y Wetzel, 2001). Si bien no ha sido roto al tiempo de escribir esto, sus similitudes con el cifrado A5/1, cuya falla espectacular puso en peligro el tráfico telefónico de GSM, son causa de preocupación (Biryukov y cols., 2000). Algunas veces sorprende a toda la gente (incluyendo al autor), que en el eterno juego del gato y el ratón entre los criptógrafos y los criptoanalistas, estos últimos salen ganando con mucha frecuencia.

Otro problema de seguridad es que Bluetooth sólo autentica dispositivos, no usuarios, por lo que el robo de un dispositivo Bluetooth podría conceder acceso al ladrón a la cuenta financiera del usuario. Sin embargo, Bluetooth también implementa seguridad en las capas superiores, por lo que incluso en el caso de una brecha de seguridad en el nivel de enlace, podría permanecer algo de seguridad, en especial en las aplicaciones que requieren que se introduzca manualmente un código PIN desde algún tipo de teclado para completar la transacción.

Seguridad de WAP 2.0

En su mayor parte, el foro WAP aprendió su lección al tener una pila de protocolos no estándar en WAP 1.0. En su mayor parte, WAP 2.0 utiliza protocolos estándares en todas las capas. La seguridad no es una excepción. Puesto que está basado en el IP, soporta el uso completo de IPsec en la capa de red. En la capa de transporte, las conexiones TCP pueden protegerse mediante TLS, un estándar IETF que analizaremos más adelante en este capítulo. Más arriba todavía, utiliza autenticación de cliente HTTP, como se define en el RFC 2617. Las crypto bibliotecas a nivel de aplicación proporcionan control de integridad y de no repudio. Después de todo, puesto que WAP 2.0 se basa en estándares bien conocidos, hay una posibilidad de que sus servicios de seguridad —en particular, privacidad, autenticación, control de integridad y no repudio— sean mejores que la seguridad del 802.11 y Bluetooth.

8.7 PROTOCOLOS DE AUTENTICACIÓN

La **autenticación** es la técnica mediante la cual un proceso verifica que su compañero de comunicación sea quien se supone que debe ser y no un impostor. Verificar la identidad de un proceso remoto en la presencia de un intruso activo y malicioso es sorprendentemente difícil y requiere protocolos complejos con base en la criptografía. En esta sección estudiaremos algunos de los muchos protocolos de autenticación que se utilizan en redes de computadoras no seguras.

Además, algunas personas confunden la autorización con la autenticación. Esta última tiene que ver con la interrogante de si usted se está comunicando con un proceso específico. La autorización tiene que ver con lo que ese proceso tiene permitido hacer. Por ejemplo, un proceso cliente contacta un servidor de archivos y dice: Soy el proceso de Scott y deseo eliminar el archivo *cookbook.old*. Desde el punto de vista del servidor, deben contestarse dos preguntas:

1. ¿Éste es el proceso real de Scott (autenticación)?
2. ¿Scott tiene permitido eliminar *cookbook.old* (autorización)?

Sólo después de que estas preguntas se contestan afirmativamente y sin ambigüedad, se puede realizar la acción solicitada. La primera pregunta es la clave. Una vez que el servidor de archivos sabe con quién está hablando, verificar la autorización es sólo cuestión de buscar entradas en las tablas locales o en las bases de datos. Por esta razón, en esta sección nos concentraremos en la autenticación.

Este modelo general es el que utilizan todos los protocolos de autenticación. Alice inicia enviando un mensaje ya sea a Bob o a un **KDC** (**Centro de Distribución de Claves**) confiable, el cual se espera sea honesto. A continuación siguen otros intercambios de mensajes en varias direcciones. Conforme se envían estos mensajes, Trudy podría interceptarlos, modificarlos o repetirlos para engañar a Alice y a Bob o simplemente dañar el trabajo.

Sin embargo, cuando el protocolo se haya completado, Alice está segura de que está hablando con Bob y Bob está seguro de que está hablando con Alice. Asimismo, en la mayoría de los protocolos, dos de ellos también habrán establecido una **clave de sesión** secreta para utilizarla en la próxima conversación. En la práctica, por razones de rendimiento, todo el tráfico de datos se encripta utilizando criptografía de clave simétrica (por lo general, AES o triple DES), aunque la criptografía de clave pública se utiliza ampliamente para los protocolos de autenticación mismos y para establecer la clave de sesión.

El objetivo de utilizar una nueva clave de sesión elegida al azar para cada nueva conexión es minimizar la cantidad de tráfico que se envía con las claves secretas o públicas del usuario, para reducir la cantidad de texto cifrado que un intruso puede obtener, y para minimizar el daño hecho si un proceso falla y su vaciado del núcleo cae en manos equivocadas. Por fortuna, la única clave presente será la de sesión. Todas las claves permanentes deben ponerse en cero con cuidado después de que se haya establecido la sesión.

8.7.1 Autenticación basada en una clave secreta compartida

Para nuestro primer protocolo de autenticación daremos por hecho que Alice y Bob ya comparten una clave secreta, K_{AB}. Esta clave compartida podría haberse acordado por teléfono o en persona, pero no en la red (insegura).

Este protocolo se basa en un principio encontrado en muchos protocolos de autenticación: una parte envía un número aleatorio a la otra, quien a continuación lo transforma en una forma especial y después regresa el resultado. Tales protocolos se conocen como de **desafío-respuesta.** En éste y en los protocolos de autenticación subsiguientes se utilizará la notación que se muestra a continuación:

A, B son las identidades de Alice y Bob.
R_i son los desafíos, donde el subíndice es el retador.
K_i son claves, donde i es el dueño.
K_S es la clave de sesión.

La secuencia de mensajes de nuestro primer protocolo de autenticación de clave compartida se ilustra en la figura 8-32. En el mensaje 1, Alice envía su identidad, A, a Bob en una forma que él entiende. Por supuesto, Bob no tiene forma de saber si este mensaje proviene de Alice o de Trudy, por lo que elige un desafío, un número aleatorio grande, R_B, y lo envía a "Alice" como mensaje 2, en texto llano. Los números aleatorios utilizados una sola vez en los protocolos de desafío-respuesta como éste se conocen como **marcas aleatorias** (***nonces***)**.** A continuación Alice encripta el mensaje con la clave que comparte con Bob y envía el texto cifrado, $K_{AB}(R_B)$, como mensaje 3. Cuando Bob ve este mensaje, inmediatamente sabe que proviene de Alice porque Trudy no conoce K_{AB} y, por lo tanto, no pudo haberlo generado. Además, puesto que R_B fue elegido de manera aleatoria a partir de un espacio grande (digamos, números aleatorios de 128 bits), no es probable que Trudy haya visto R_B y su respuesta en una sesión anterior. Tampoco es probable que pueda adivinar la respuesta correcta de cualquier desafío.

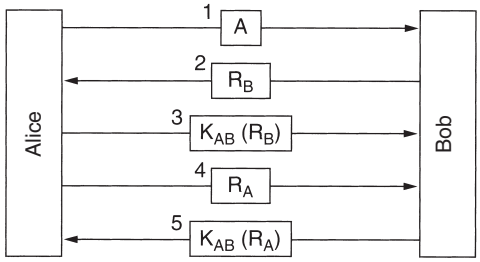


Figura 8-32. Autenticación de dos vías que utiliza un protocolo de desafío-respuesta.

En este punto, Bob está seguro de que está hablando con Alice, pero ella no está segura de nada. Por lo que Alice sabe, Trudy podría haber interceptado el mensaje 1 y regresar R_B como respuesta. Tal vez Bob murió la noche anterior. Para averiguar con quién está hablando, Alice elige un número al azar, R_A y lo envía a Bob como texto llano, en el mensaje 4. Cuando Bob responde con $K_{AB}(R_A)$, Alice sabe que está hablando con Bob. Si desean establecer una clave de sesión ahora, Alice puede elegir una, K_S, encriptarla con K_{AB} y enviarla a Bob.

El protocolo de la figura 8-32 contiene cinco mensajes. Veamos si podemos ser ingeniosos para eliminar algunos de ellos. En la figura 8-33 se muestra un método. Aquí Alice inicia el protocolo de desafío-respuesta en lugar de esperar a que Bob lo haga. De manera similar, mientras Bob responde al desafío de Alice, envía el suyo. El protocolo puede reducirse a tres mensajes en lugar de a cinco.

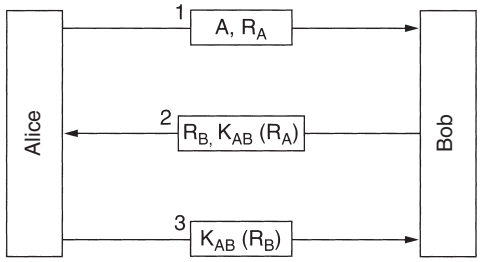


Figura 8-33. Un protocolo de autenticación de dos vías acortado.

¿Este nuevo protocolo es una mejora del original? En un sentido sí lo es: es más corto. Desgraciadamente, también es incorrecto. Bajo algunas circunstancias, Trudy puede vencer este protocolo utilizando lo que se conoce como un **ataque de reflexión.** En particular, Trudy puede romperlo si es posible para abrir a la vez múltiples sesiones con Bob. Por ejemplo, esta situación podría ocurrir si Bob es un banco y está preparado para aceptar muchas conexiones simultáneas de cajeros automáticos.

En la figura 8-34 se muestra el ataque de reflexión de Trudy. Inicia cuando Trudy afirma que es Alice y envía R_T. Bob responde, como es usual, con su propio desafío, R_B. Ahora Trudy está atorada. ¿Qué puede hacer? No conoce K_{AB} (R_B).

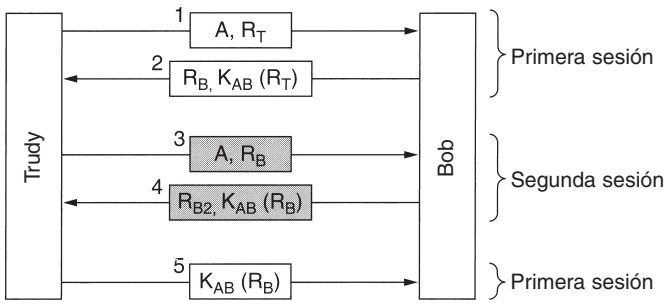


Figura 8-34. El ataque de reflexión.

Trudy puede abrir una segunda sesión con el mensaje 3, y proporcionar un R_B, tomado del mensaje 2, como su desafío. Bob lo encripta con calma y regresa K_{AB} (R_B) en el mensaje 4. Sombreamos los mensajes de la segunda sesión para que resalten. Ahora Trudy tiene la información que le faltaba, por lo que puede completar la primera sesión y abortar la segunda. Bob ahora está convencido de que Trudy es Alice, por lo que cuando ella le pide su estado de cuenta bancaria, Bob se lo proporciona sin más. Después, cuando ella le pide que transfiera todo su dinero a una cuenta secreta en un banco de Suiza, lo hace sin titubeo alguno.

La moraleja de esta historia es:

Diseñar un protocolo de autenticación correcto es más difícil de lo que parece.

Las siguientes cuatro reglas generales con frecuencia ayudan:

1. Obligue al iniciador a que pruebe que es quien dice ser antes de que el contestador tenga que hacerlo. En este caso, Bob reveló información valiosa antes de que Trudy proporcionara cualquier evidencia de que era quien decía ser.
2. Obligue a que tanto el iniciador como el contestador utilicen claves diferentes para probar, aunque esto signifique tener dos claves compartidas, K_{AB} y K'_{AB}.
3. Obligue a que el iniciador y el contestador utilicen conjuntos diferentes para elaborar sus desafíos. Por ejemplo, el iniciador debe utilizar números pares y el contestador números impares.
4. Haga que el protocolo resista a ataques que involucren una segunda sesión paralela en la que la información obtenida en una sesión se utilice en una diferente.

Si se viola alguna de estas reglas, el protocolo puede romperse con frecuencia. Aquí, se violaron las cuatro reglas, con consecuencias desastrosas.

Ahora regresemos y demos un vistazo más de cerca a la figura 8-32. ¿Existe la seguridad de que este protocolo no está sujeto a un ataque de reflexión? Bueno, depende. Es muy sutil. Trudy fue capaz de derrotar nuestro protocolo utilizando un ataque de reflexión porque fue posible abrir una segunda sesión con Bob y engañarlo para que contestara sus propias preguntas. ¿Qué habría pasado si Alice fuera una computadora de propósito general que también aceptara sesiones múltiples, en lugar de una persona en una computadora? Veamos lo que puede hacer Trudy.

Para saber cómo funciona el ataque de Trudy, vea la figura 8-35. Alice inicia anunciando su identidad en el mensaje 1. Trudy intercepta ese mensaje y comienza su propia sesión con el mensaje 2, afirmando que es Bob. Nuevamente sombreamos los mensajes de la sesión 2. Alice responde al mensaje 2 diciendo "¿Dices ser Bob? Pruébalo." en el mensaje 3. En este punto Trudy está atorada porque no puede probar que es Bob.

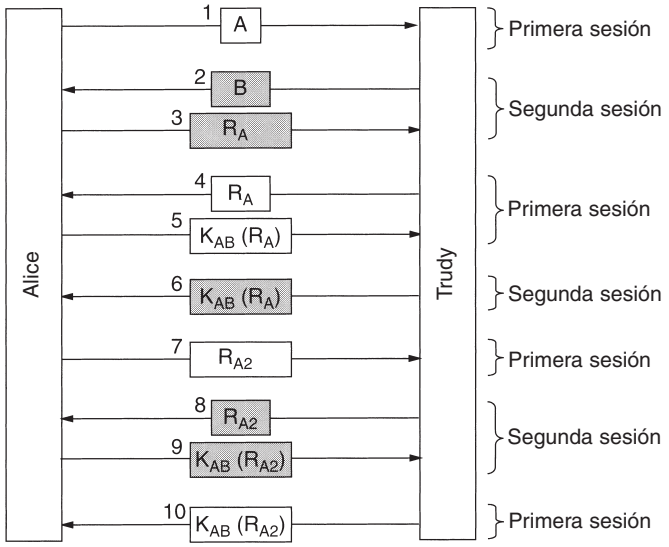


Figura 8-35. Un ataque de reflexión al protocolo de la figura 8-32.

¿Qué hace Trudy ahora? Regresa a la primera sesión, en donde le toca enviar un desafío, y envía el R_A que obtuvo en el mensaje 3. Alice responde amablemente a él en el mensaje 5, y de esta forma proporciona a Trudy la información que necesita para enviar el mensaje 6 en la sesión 2. En este punto, Trudy está prácticamente del otro lado porque ha respondido exitosamente al desafío de Alice en la sesión 2. Ahora puede cancelar la sesión 1, enviar cualquier número antiguo por el resto de la sesión 2 y tendrá una sesión autenticada con Alice en la sesión 2.

Pero Trudy es vil y realmente desea insistir. En lugar de enviar cualquier número antiguo en toda la sesión 2, espera hasta que Alice envía en el mensaje 7, el desafío de Alice para la sesión 1. Por supuesto, Trudy no sabe cómo responder, por lo que utiliza nuevamente el ataque de reflexión, regresando R_{A2} como el mensaje 8. Alice encripta de manera apropiada R_{A2} en el mensaje 9. Trudy

ahora cambia a la sesión 1 y envía a Alice el número que desea en el mensaje 10, copiado de manera conveniente a partir de lo que Alice envió en el mensaje 9. En este punto Trudy tiene dos sesiones completamente autenticadas con Alice.

Este ataque tiene un resultado ligeramente diferente que el ataque del protocolo de tres mensajes que se muestra en la figura 8-34. Esta vez, Trudy tiene dos conexiones autenticadas con Alice. En el ejemplo anterior, tenía una conexión autenticada con Bob. Aquí, si hubiéramos aplicado todas las reglas de protocolos de autenticación analizadas previamente, este ataque podría haberse detenido. Un análisis detallado de este tipo de ataques y cómo frustrarlos se da en (Bird y cols., 1993). También muestran cómo es posible construir de manera sistemática protocolos que tengan altas probabilidades de ser correctos. No obstante, el protocolo más simple de este tipo es un poco complicado, por lo que a continuación mostraremos una clase diferente de protocolo que también funciona.

El nuevo protocolo de autenticación se muestra en la figura 8-36 (Bird y cols., 1993). Utiliza un HMAC del tipo que vimos cuando estudiamos IPsec. Alice inicia enviando a Bob una marca aleatoria, R_A como mensaje 1. Bob responde seleccionando su propia marca aleatoria, R_B, y enviándola junto con un HMAC. El HMAC se forma para construir una estructura de datos que consiste en la marca aleatoria de Alice, la marca aleatoria de Bob, sus identidades y la clave secreta compartida, K_{AB}. A continuación a estos datos estructurados se les aplica un *hash* en el HMAC, por ejemplo utilizando SHA-1. Cuando Alice recibe el mensaje 2, tiene una R_A (que ella misma eligió), R_B, que llega como texto llano, las dos identidades y la clave secreta, K_{AB}, que ha sabido todo el tiempo, por lo que ella misma puede calcular el HMAC. Si corresponde con el HMAC del mensaje, Alice sabe que está hablando con Bob porque Trudy no conoce K_{AB} y, por lo tanto, no sabe cuál HMAC enviar. Alice responde a Bob con un HMAC que contiene sólo las dos marcas aleatorias.

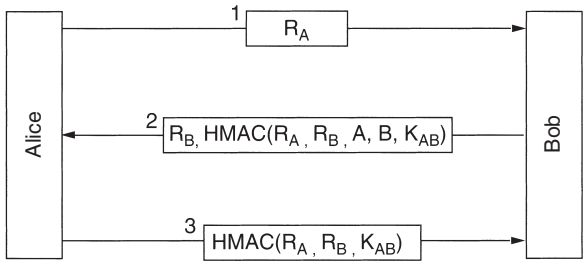


Figura 8-36. Autenticación mediante HMACs.

¿Trudy puede romper de alguna forma este protocolo? No, porque ella no puede obligar a ninguna de las dos partes a encriptar o aplicar un *hash* a un valor de su elección, como sucede en las figuras 8-34 y 8-35. Ambos HMACs incluyen valores elegidos por la parte emisora, algo que Trudy no puede controlar.

El uso de HMACs no es la única forma de utilizar esta idea. Un esquema alternativo que se utiliza con mucha frecuencia en lugar de calcular el HMAC por sobre una serie de elementos es encriptar dichos elementos de manera secuencial utilizando encadenamiento de bloques de cifrado.

8.7.2 Establecimiento de una clave compartida: el intercambio de claves de Diffie-Hellman

Hasta ahora hemos supuesto que Alice y Bob comparten una clave secreta. Suponga que no lo hacen (porque hasta el momento no hay una PKI aceptada universalmente para firmar y distribuir certificados). ¿Cómo pueden establecer una? Una forma sería que Alice llamara a Bob y le diera su clave por teléfono, pero probablemente él iniciaría diciendo: ¿Cómo sé que eres Alice y no Trudy? Podrían tratar de establecer una reunión, en la que cada uno trajera un pasaporte, una licencia de conducir y tres tarjetas de crédito, pero al ser gente ocupada, tal vez no encuentren una fecha aceptable para los dos en meses. Por fortuna, tan increíble como pueda parecer, hay una forma para que personas totalmente extrañas establezcan una clave secreta compartida a la vista de todos, incluso si Trudy está grabando con cuidado cada mensaje.

El protocolo que permite que extraños establezcan una clave secreta compartida se conoce como **intercambio de claves de Diffie-Hellman** (Diffie y Hellman, 1976) y funciona como sigue. Alice y Bob tienen que estar de acuerdo en dos números grandes, n y g, donde n es un número primo, $(n - 1)/2$ también es un número primo y ciertas condiciones se aplican a g. Estos números podrían ser públicos, por lo que cualquiera de ellos simplemente pueden elegir n y g y decirle al otro de manera abierta. Ahora Alice elige un número grande (digamos, de 512 bits), x, y lo mantiene en secreto. De manera similar, Bob elige un número secreto grande, y.

Alice inicia el protocolo de intercambio de claves enviando a Bob un mensaje que contiene (n, g, g^x mod n), como se muestra en la figura 8-37. Bob responde enviando a Alice un mensaje que contiene g^y mod n. Ahora Alice eleva a la x potencia módulo n el número que Bob le envió para obtener $(g^y \bmod n)^x \bmod n$. Bob realiza una operación similar para obtener $(g^x \bmod n)^y \bmod n$. Por las leyes de la aritmética modular, ambos cálculos dan como resultado $g^{xy} \bmod n$. Como por arte de magia, Alice y Bob comparten una clave secreta, g^{xy} mod n.

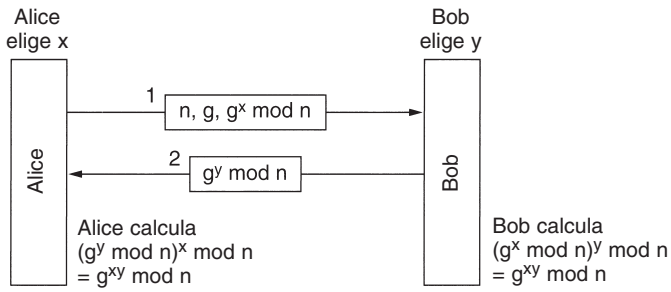


Figura 8-37. El intercambio de claves de Diffie-Hellman.

Por supuesto, Trudy ha visto ambos mensajes. Ella conoce g y n a partir del mensaje 1. Si pudiera calcular x y y, podría adivinar la clave secreta. El problema es que, con sólo g^x mod n, no puede encontrar x. No se conoce ningún algoritmo práctico para calcular logaritmos discretos módulo un número primo muy grande.

Para que el ejemplo sea más concreto, utilizaremos los valores (que no son reales en absoluto) de $n = 47$ y $g = 3$. Alice elige $x = 8$ y Bob elige $y = 10$. Esto se mantiene en secreto. El mensaje de Alice para Bob es (47, 3, 28) porque 3^8 mod 47 es 28. El mensaje de Bob para Alice es (17). Alice calcula 17^8 mod 47, lo cual es 4. Bob calcula 28^{10} mod 47, lo cual es 4. Alice y Bob han determinado de manera independiente que la clave secreta ahora es 4. Trudy tiene que resolver la ecuación 3^x mod 47 = 28, lo cual puede hacerse mediante una búsqueda minuciosa de números pequeños como éstos, pero no cuando todos los números tienen una longitud de cientos de bits. Todos los algoritmos conocidos en la actualidad simplemente tardan mucho, incluso en supercomputadoras paralelas masivas.

A pesar de la elegancia del algoritmo de Diffie-Hellman, hay un problema: cuando Bob obtiene el triple (47, 3, 28), ¿cómo sabe que proviene de Alice y no de Trudy? No hay forma de que pueda saberlo. Desgraciadamente, Trudy puede explotar este hecho para engañar tanto a Alice como a Bob, como se ilustra en la figura 8-38. Aquí, mientras Alice y Bob están eligiendo x y y, respectivamente, Trudy elige su propio número aleatorio, z. Alice envía el mensaje 1 dirigido a Bob. Trudy lo intercepta y envía el mensaje 2 a Bob, utilizando los valores g y n correctos (que de todas formas son públicos) pero con su propio valor z en lugar de x. También regresa el mensaje 3 a Alice. Más tarde Bob envía el mensaje 4 a Alice, el cual es interceptado y conservado nuevamente por Trudy.

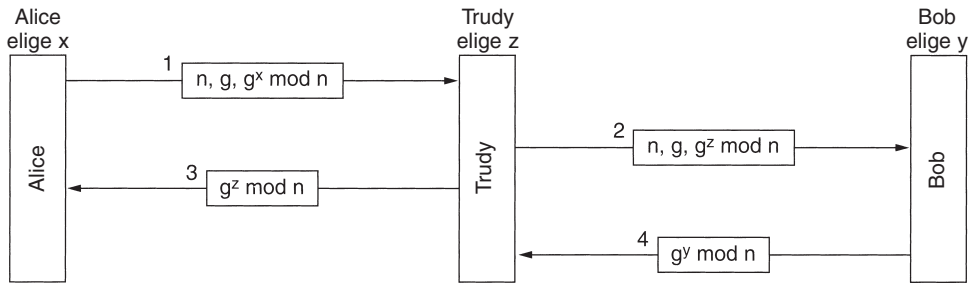


Figura 8-38. El ataque de la brigada de bomberos o de hombre en medio.

Ahora todos realizan la aritmética modular. Alice calcula la clave secreta como g^{xz} mod n, al igual que Trudy (para mensajes destinados a Alice). Bob calcula g^{yz} mod n al igual que Trudy (para mensajes destinados a Bob). Alice piensa que está hablando con Bob por lo que establece una clave de sesión (con Trudy). Bob también lo hace. Cada mensaje que Alice envía en la sesión encriptada, Trudy lo captura, almacena, modifica si lo desea, y lo pasa (lo cual es opcional) a Bob. De manera similar, en la otra dirección. Trudy ve todo y puede modificar todos los mensajes a voluntad, mientras que Alice y Bob tienen la creencia de que tienen un canal seguro. Este ataque se conoce como **ataque de la brigada de bomberos,** porque se parece vagamente a un antiguo departamento de bomberos voluntarios en el que éstos pasan cubetas en fila desde el camión de bomberos hacia el fuego. También se conoce como **ataque de hombre en medio.**

8.7.3 Autenticación que utiliza un centro de distribución de claves

El secreto compartido con un extraño casi funcionó, pero no del todo. Por otro lado, tal vez no valga la pena hacerlo. Para hablar de esta manera con *n* personas, podría necesitar *n* claves. Para las personas populares, la administración de claves podría volverse una verdadera carga, especialmente si cada clave tiene que almacenarse aparte en una tarjeta de chips de plástico.

Un método diferente es introducir un KDC (centro de distribución de claves confiable.) En este modelo, cada usuario tiene una clave compartida con el KDC. Ahora el KDC realiza la administración de claves de sesión y autenticación.

En la figura 8-39 se muestran el protocolo de autenticación KDC más simple conocido que involucra dos partes y un KDC confiable.

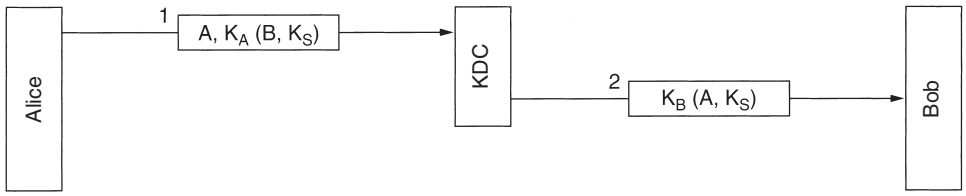


Figura 8-39. Un primer intento en un protocolo de autenticación que utiliza un KDC.

La idea detrás de este protocolo es simple: Alice elige una clave de sesión, K_S, e indica al KDC que desea hablar con Bob utilizando K_S. Este mensaje se encripta con la clave secreta que Alice (sólo) comparte con el KDC, K_A. El KDC desencripta este mensaje, extrayendo la identidad de Bob y la clave de sesión. Después construye un nuevo mensaje que contiene la identidad de Alice y la clave de sesión y envía este mensaje a Bob. Esta encriptación se realiza con K_B, la clave secreta que Bob comparte con el KDC. Cuando Bob desencripta el mensaje, sabe que Alice desea hablar con él y cuál clave desea utilizar.

La autenticación sucede aquí sin costo alguno. El KDC sabe que el mensaje 1 debe provenir de Alice, puesto que nadie más podría encriptarlo con la clave secreta de Alice. De manera similar, Bob sabe que el mensaje 2 debe provenir del KDC, en quien él confía, puesto que nadie más sabe su clave secreta.

Desgraciadamente, este protocolo tiene un defecto importante. Trudy necesita algo de dinero, por lo que idea algún servicio legítimo que puede realizar por Alice, realiza una oferta atractiva y obtiene el trabajo. Después de realizar el trabajo, Trudy educadamente solicita a Alice que pague por transferencia bancaria. A continuación Alice establece una clave de sesión con su banquero, Bob. Después envía a Bob un mensaje en el que le pide que transfiera dinero a la cuenta de Trudy.

Mientras tanto, Trudy regresa a sus antiguos hábitos, espiar la red. Copia los dos mensajes de la figura 8-39 y la solicitud de transferencia de dinero que le sigue. Más tarde, lo repite para Bob. Éste los obtiene y piensa: Alice debe haber contratado nuevamente a Trudy. Ésta seguramente

hace un magnífico trabajo. Después Bob transfiere una cantidad igual de dinero de la cuenta de Alice a la de Trudy. Algún tiempo después del quincuagésimo par de mensajes, Bob sale corriendo de la oficina a fin de encontrar a Trudy para ofrecerle un gran préstamo de manera que pueda ampliar su obviamente exitoso negocio. Este problema se conoce como el **ataque de repetición.**

Son posibles algunas soluciones al ataque de repetición. La primera es incluir una marca de tiempo en cada mensaje. Después, si cada quien recibe un mensaje obsoleto, puede descartarse. El problema con este método es que los relojes en una red nunca están sincronizados, por lo que debe haber un intervalo durante el cual sea válida la marca de tiempo. Trudy puede repetir el mensaje durante este intervalo y marcharse con él.

La segunda solución es colocar una marca aleatoria en cada mensaje. A continuación cada parte tiene que recordar todas las marcas aleatorias previas y rechazar cualquier mensaje que contenga una marca aleatoria utilizada anteriormente. Sin embargo, las marcas aleatorias deben ser recordadas por siempre, a fin de que Trudy no trate de repetir un mensaje de 5 años de antigüedad. Además, si una máquina falla y pierde su lista de marcas aleatorias, nuevamente es vulnerable a un ataque de repetición. Las marcas de tiempo y las marcas aleatorias pueden combinarse para limitar cuánto tiempo deben recordarse las marcas aleatorias, pero claramente el protocolo se va a poner mucho más complicado.

Un método mucho más refinado para la autenticación mutua es utiliza un protocolo de desafío-respuesta de múltiples vías. Un ejemplo bien conocido de tal protocolo es el protocolo de **autenticación de Needham-Schroeder** (Needham y Schroeder, 1978), una variante de lo que se muestra en la figura 8-40.

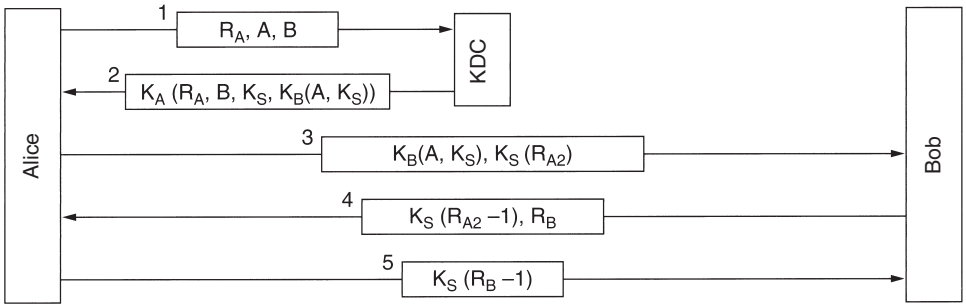


Figura 8-40. El protocolo de autenticación Needham-Schroeder.

El protocolo comienza con Alice diciéndole al KDC que desea hablar con Bob. Este mensaje contiene como marca aleatoria un número aleatorio grande, R_A. El KDC regresa el mensaje 2 que contiene el número aleatorio de Alice, una clave de sesión y un boleto que ella puede regresar a Bob. El objetivo del número aleatorio, R_A, es asegurar a Alice que el mensaje 2 está actualizado y que no es una repetición. La identidad de Bob también está encerrada en caso de que Trudy tenga algunas ideas graciosas sobre reemplazar *B* en el mensaje 1 con su propia identidad a fin de que el

KDC encripte el boleto al final del mensaje 2 con K_T en lugar de K_B. El boleto encriptado con K_B se incluye dentro del mensaje encriptado para evitar que Trudy lo reemplace con algo más cuando el mensaje vaya hacia Alice.

Alice ahora envía el boleto a Bob, junto con un nuevo número aleatorio, R_{A2}, encriptado con la clave de sesión, K_S. En el mensaje 4, Bob regresa K_S ($R_{A2} - 1$) para probar a Alice que ella está hablando con el Bob verdadero. Regresar K_S (R_{A2}) podría no haber funcionado, puesto que Trudy podría haberlo robado del mensaje 3.

Después de recibir el mensaje 4, Alice está convencida de que está hablando con Bob y de que hasta el momento no es posible que se hayan utilizado repeticiones. Después de todo, simplemente generó R_{A2} algunos milisegundos antes. El propósito del mensaje 5 es convencer a Bob de que sí es Alice con la que está hablando, y de que aquí tampoco se han utilizado repeticiones. Al hacer que las dos partes generen un desafío y una respuesta, se elimina cualquier posibilidad de algún tipo de ataque de repetición.

Aunque este protocolo parece muy sólido, tiene una ligera debilidad. Si Trudy se las arregla para conseguir una sesión de clave antigua en texto llano, puede iniciar una nueva sesión con Bob al repetir el mensaje 3 que corresponde a la clave comprometida y al convencerlo de que ella es Alice (Denning y Sacco, 1981). Esta vez puede saquear la cuenta bancaria de Alice sin tener que realizar ni siquiera una vez el servicio legítimo.

Posteriormente Needham y Schroeder publicaron un protocolo que corrige este problema (Needham y Schroeder, 1987). En el mismo número de la misma publicación, Otway y Rees (1987) también dieron a conocer un protocolo que resuelve el problema en una forma más corta. La figura 8-41 muestra un protocolo Otway-Rees con algunas modificaciones ligeras.

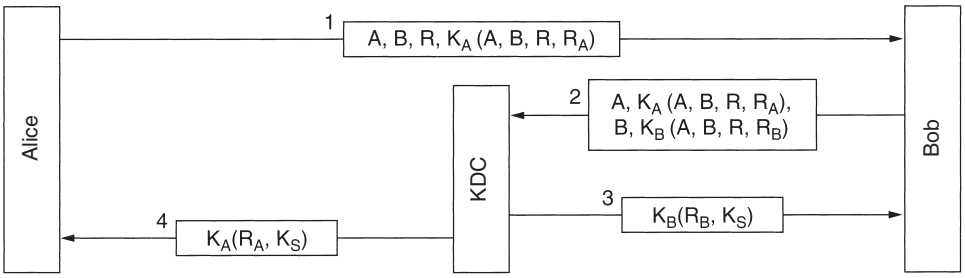


Figura 8-41. El protocolo de autenticación de Otway-Rees (ligeramente simplificado).

En el protocolo de Otway-Rees, Alice inicia generando un par de números aleatorios, *R*, que se utilizará como identificador común, y R_A, que Alice utilizará para retar a Bob. Cuando Bob obtenga este mensaje, construirá un mensaje nuevo a partir de la parte encriptada del mensaje de Alice y del suyo propio. Ambas partes encriptadas con K_A y K_B identifican a Alice y a Bob, contienen el identificador común y un desafío.

El KDC verifica si en ambas partes *R* es el mismo. Podría no ser porque Trudy falsificó *R* en el mensaje 1 o reemplazó parte del mensaje 2. Si los dos *R*s coinciden, el KDC cree que el mensaje

de solicitud de Bob es válido. Después genera una clave de sesión y la encripta dos veces, una para Alice y la otra para Bob. Cada mensaje contiene el número aleatorio del receptor, como prueba de que el KDC, y no Trudy, generó el mensaje. En este punto tanto Alice como Bob poseen la misma clave de sesión y pueden comenzar a comunicarse. La primera vez que intercambian mensajes de datos, cada uno puede ver que el otro tiene una copia idéntica de K_S, por lo que la autenticación está completa.

8.7.4 Autenticación utilizando Kerberos

Un protocolo de autenticación utilizado en muchos sistemas reales (incluyendo Windows 2000) es **Kerberos,** que está basado en una variante de Needham-Schroeder. Su nombre proviene del perro de múltiples cabezas de la mitología griega que custodiaba la entrada a Hades (para mantener fuera a las personas indeseables). Kerberos se diseñó en el M.I.T. para permitir que los usuarios de estaciones de trabajo accedieran recursos de red en una forma segura. Su mayor diferencia con respecto de Needham-Schroeder es su suposición de que todos los relojes están bien sincronizados. El protocolo ha pasado por varias iteraciones. V4 es la versión más ampliamente utilizada en la industria, por lo que la describiremos. Más tarde, daremos algunas palabras sobre su sucesor, V5. Para mayor información, vea (Steiner y cols., 1988).

Kerberos involucra a tres servidores además de Alice (una estación de trabajo cliente):

Authentication Server (AS): verifica usuarios durante el inicio de sesión
Ticket-Granting Server (TGS): emite "prueba de boletos de identidad"
Bob, el servidor: hace el trabajo que Alice desea

AS es similar a KDC en que comparte una contraseña secreta con cada usuario. El trabajo de TGS es emitir boletos que puedan convencer a los servidores reales de que el portador de un boleto TGS es realmente quien afirma ser.

Para iniciar una sesión, Alice se sienta frente a una estación de trabajo pública arbitraria y teclea su nombre. La estación de trabajo envía su nombre al AS en texto llano, como se muestra en la figura 8-42. Lo que regresa es una clave de sesión y un boleto, $K_{TGS}(A, K_S)$, destinado para TGS. Estos elementos se empaquetan y encriptan mediante la clave secreta de Alice, de forma que sólo Alice pueda desencriptarlos. Únicamente cuando el mensaje 2 llega, la estación de trabajo pide la contraseña de Alice. A continuación, dicha contraseña se utiliza para generar K_A a fin de desencriptar el mensaje 2 y obtener la clave de sesión y el boleto TGS dentro de él. En este punto, la estación de trabajo sobrescribe la contraseña de Alice para asegurarse de que sólo esté dentro de la estación de trabajo por lo mucho durante algunos milisegundos. Si Trudy intenta iniciar una sesión como Alice, la contraseña que teclee será errónea y la estación de trabajo detectará esto porque la parte estándar del mensaje 2 será incorrecta.

Después de que inicia la sesión, Alice podría decirle a la estación de trabajo que desea contactar a Bob, el servidor de archivos. A continuación la estación de trabajo envía el mensaje 3 al TGS preguntándole por un boleto para utilizar con Bob. El elemento clave en esta petición es $K_{TGS}(A, K_S)$, que se encripta con la clave secreta del TGS y que se utiliza como prueba de que el

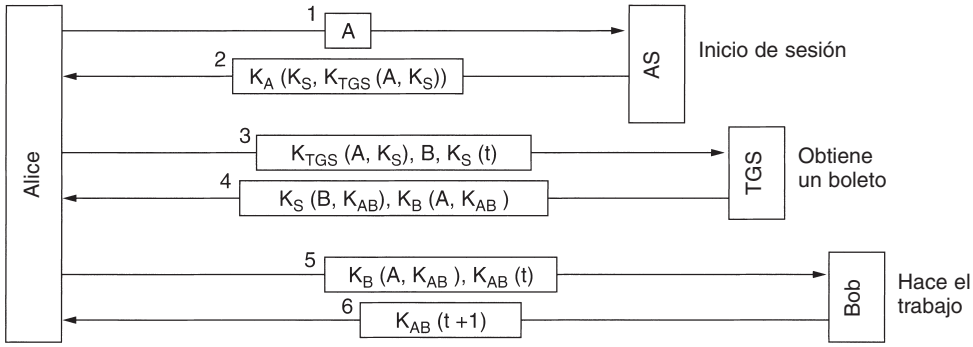


Figura 8-42. El funcionamiento de Kerberos V4.

emisor es realmente Alice. El TGS responde con la creación de una clave de sesión, K_{AB}, para que Alice la utilice con Bob. Se regresan dos versiones de dicha clave. La primera se encripta sólo con K_S, de manera que Alice pueda leerla. La segunda se encripta con la clave de Bob, K_B, de manera que Bob pueda leerla.

Trudy puede copiar el mensaje 3 y tratar de utilizarlo nuevamente, pero no podrá hacerlo porque se lo impedirá la marca de tiempo encriptada, *t*, la cual se envía junto con él. Trudy no puede reemplazar la marca de tiempo con una más reciente, porque ella no conoce K_S, la clave de sesión que Alice utiliza para hablar con el TGS. Incluso si Trudy repite con rapidez el mensaje 3, todo lo que obtendrá será otra copia del mensaje 4, el cual no pudo desencriptar la primera vez y que tampoco podrá desencriptar esta segunda vez.

Ahora Alice puede enviar K_{AB} a Bob para establecer una sesión con él. Este intercambio también tiene establecidas marcas de tiempo. La respuesta es probar a Alice que realmente está hablando con Bob, no con Trudy.

Después de esta serie de intercambios, Alice puede comunicarse con Bob bajo la cobertura de K_{AB}. Si más adelante Alice necesita hablar con otro servidor, Carol, simplemente repite el mensaje 3 al TGS, sólo que ahora especificando *C* en lugar de *B*. El TGS responderá rápidamente con un boleto encriptado con K_C que Alice puede enviar a Carol, y que Carol aceptará como prueba de que proviene de Alice.

El objetivo de todo este trabajo es que ahora Alice puede acceder servidores a través de toda la red de forma segura y que su contraseña no tiene que pasar a través de la red. De hecho, sólo tuvo que estar en su propia estación de trabajo durante algunos milisegundos. Sin embargo, observe que cada servidor realiza su propia autorización. Cuando Alice presenta su boleto a Bob, esto simplemente prueba a Bob quién lo envió. Bob es quien determina las acciones que Alice tiene permitido realizar.

Puesto que los diseñadores de Kerberos no esperaron que todo el mundo confiara en un solo servidor de autenticación, establecieron reglas para tener múltiples **dominios,** cada uno con su propio AS y TGS. Para obtener un boleto para un servidor de un dominio distante, Alice podría solicitar a su propio TGS un boleto que sea aceptado por el TGS en el dominio distante. Si el TGS distante se ha registrado con el TGS local (de la misma manera en que lo hacen los servidores

locales), este último le dará a Alice un boleto válido en el TGS distante. A continuación ella puede hacer negocios, como obtener boletos para servidores de ese dominio. Sin embargo, observe que para que las partes de dos dominios hagan negocios, cada una debe confiar en el TGS de la otra.

Kerberos V5 es más elegante que V4 y tiene más sobrecarga. También utiliza OSI ASN.1 (Notación de Sintaxis Abstracta 1) para describir tipos de datos y tiene pequeños cambios en los protocolos. Además, sus boletos tienen duraciones más largas, permite que los boletos sean renovados y emitirá boletos postfechados. Además, al menos en teoría, no depende de DES, como lo hace V4, y soporta múltiples dominios pues delega la generación de boletos a múltiples servidores de boletos.

8.7.5 Autenticación utilizando criptografía de clave pública

La autenticación mutua también puede realizarse utilizando criptografía de clave pública. Para empezar, Alice necesita obtener la clave pública de Bob. Si existe una PKI con un servidor de directorios que proporciona certificados de clave pública, Alice puede pedir la de Bob, que en la figura 8-43 se muestra como mensaje 1. La repetición, en el mensaje 2, es un certificado X.509 que contiene la clave pública de Bob. Cuando Alice verifica que la firma sea correcta, envía a Bob un mensaje que contiene su identidad y una marca aleatoria.

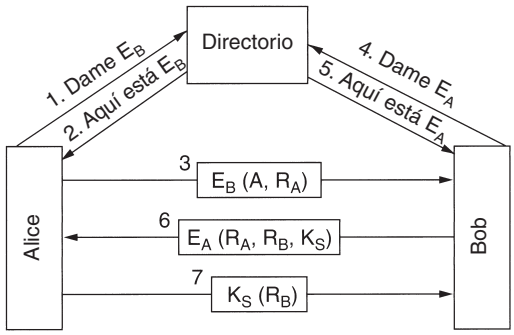


Figura 8-43. Autenticación mutua utilizando la criptografía de clave pública.

Cuando Bob recibe este mensaje, no tiene idea de si proviene de Alice o de Trudy, pero sigue adelante y pide al servidor de directorio la clave pública de Alice (mensaje 4) la cual obtiene de inmediato (mensaje 5). A continuación envía a Alice como mensaje 6 uno que contiene el R_A de Alice, su propia marca aleatoria, R_B, y una clave de sesión propuesta, K_S.

Cuando Alice obtiene el mensaje 6, lo desencripta mediante su clave privada. Ve el R_A en dicho mensaje, lo cual le trae alivio. El mensaje debe provenir de Bob, puesto que Trudy no tiene forma de determinar R_A. Además, debe ser una actualización y no una repetición, puesto que acaba de enviarle a Bob el R_A. Alice accede a la sesión enviando el mensaje 7. Cuando Bob ve R_B encriptado con la clave de sesión que acaba de generar, sabe que Alice obtuvo el mensaje 6 y verificó R_A.

¿Qué puede hacer Trudy para tratar de arruinar este protocolo? Puede fabricar el mensaje 3 y engañar a Bob para que investigue a Alice, pero ésta verá un R_A que ella no envió y ya no proseguirá. Trudy no podrá falsificar el mensaje 7 para Bob porque no conoce R_B o K_S y no puede determinarlos sin la clave privada de Alice. No es el día de suerte de Trudy.

8.8 SEGURIDAD DE CORREO ELECTRÓNICO

Cuando se envía un mensaje de correo electrónico entre dos sitios distantes, por lo general pasará por docenas de máquinas. Cualquiera de ellas puede leer y registrar el mensaje para uso futuro. En la práctica, no existe la privacidad, a pesar de lo que mucha gente piensa. Sin embargo, muchas personas desearían tener la capacidad de enviar correo electrónico que sólo pueda ser leído por el destinatario y por nadie más: ni siquiera su jefe ni su gobierno. Este deseo ha estimulado a varias personas y grupos a que apliquen al correo electrónico los principios criptográficos que estudiamos anteriormente para producir correo electrónico seguro. En las siguientes secciones analizaremos un sistema seguro ampliamente utilizado, PGP, y después mencionaremos brevemente dos más, PEM y S/MIME. Para información adicional sobre el correo electrónico seguro, vea (Kaufman y cols., 2002, y Schneier, 1995).

8.8.1 PGP—Privacidad Bastante Buena

Nuestro primer ejemplo, **PGP** (**Privacidad Bastante Buena**), es esencialmente la idea original de una persona, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann es un defensor de la privacidad cuyo lema es: Si la privacidad se elimina, solamente los delincuentes tendrán privacidad. Liberado en 1991, PGP es un paquete de seguridad de correo electrónico completo que proporciona privacidad, autenticación, firmas digitales y compresión, todo en una forma fácil de utilizar. Además, todo el paquete, incluyendo todo el código fuente, se distribuye sin costo alguno a través de Internet. Debido a su cualidad, precio (ninguno) y fácil disponibilidad en las plataformas UNIX, Linux, Windows y Mac OS, se utiliza ampliamente hoy en día.

PGP encripta los datos utilizando un cifrado de bloque llamado **IDEA** (**Algoritmo Internacional de Encriptación de Datos**), que utiliza claves de 128 bits. Se diseñó en Suecia en la época en que el DES fue visto como defectuoso y el AES todavía no se inventaba. De manera conceptual, IDEA es similar a DES y AES: mezcla los bits en una serie de rondas, pero los detalles de las funciones de mezcla son diferentes de los de DES y AES. La administración de claves utiliza RSA y la integridad de datos utiliza MD5, temas que ya hemos analizado.

PGP también se ha visto envuelto en controversias desde su primer día (Levy, 1993). Debido a que Zimmermann no hizo nada para que otras personas no colocaran a PGP en Internet, en donde personas de todo el mundo podían accederlo, el gobierno de los Estados Unidos declaró que Zimmermann había violado las leyes de los Estados Unidos que prohibían la exportación de equipo de guerra. La investigación del gobierno de los Estados Unidos en contra de Zimmermann duró cinco años, pero con el tiempo se abandonó, probablemente por dos razones. Primera, Zimmermann

mismo no colocó a PGP en Internet, por lo que su abogado alegó que *Zimmermann* nunca exportó nada (y ahí entra la controversia de si la creación de un sitio Web significa exportación). Segunda, con el tiempo el gobierno se dio cuenta de que ganar el juicio significaba convencer al jurado de que el hecho de crear un sitio Web que contenga un programa de privacidad descargable estaba cubierto por la ley de tráfico de armas que prohíbe la exportación de material de guerra, como tanques, submarinos, aviones militares y armas nucleares. Los años de mala publicidad tampoco fueron de mucha ayuda.

Además, las leyes de exportación son extrañas, por decir lo menos. El gobierno consideró la colocación de código en un sitio Web como una exportación ilegal y persiguió a Zimmermann durante cinco años por eso. Por otro lado, cuando alguien publicó todo el código fuente PGP, en C, en un libro (en una fuente grande con una suma de verificación en cada página para hacer que su digitalización fuera fácil) y después exportó dicho libro, no hubo problema con el gobierno debido a que los libros no se clasifican como equipo de guerra. La espada es más poderosa que la pluma, por lo menos para el Tío Sam.

Otro problema en el que se vio involucrado PGP tuvo que ver con la infracción de las patentes. La compañía que poseía la patente RSA, RSA Security, Inc., alegó que el uso que PGP hacía del algoritmo RSA infringió su patente, pero ese problema quedó resuelto a partir de la versión 2.6. Además, PGP utiliza otro algoritmo de encriptación patentado, IDEA, cuyo uso causó algunos problemas al principio.

Puesto que el código de PGP es abierto, diversas personas y grupos lo han modificado y producido varias versiones. Algunas de ellas se diseñaron para resolver los problemas con las leyes de equipo de guerra, otras se enfocaron en evitar el uso de algoritmos patentados, y otros más desearon convertirlo en un producto comercial de código cerrado. Aunque en la actualidad las leyes de equipo de guerra se han liberado ligeramente (de lo contrario los productos que utilizan AES no podrían exportarse de los Estados Unidos), y la patente de RSA caducó en septiembre de 2000, el legado de todos estos problemas es que hay en circulación varias versiones incompatibles de PGP, bajo diversos nombres. La discusión anterior se enfoca en el PGP clásico, que es la versión más antigua y simple. En el RFC 2440 se describe otra versión popular, Open PGP. Otra más es GNU Privacy Guard.

PGP utiliza de manera intencional algoritmos criptográficos existentes en lugar de inventar nuevos. Se basa principalmente en algoritmos que han aguantado revisiones extensas de iguales y no fueron diseñados o influenciados por ninguna agencia gubernamental para debilitarlos. Para las personas que intentan desconfiar del gobierno, esta propiedad es una gran ganancia.

PGP soporta la compresión de texto, confidencialidad y firmas digitales, además de que proporciona características de administración extensa de claves, pero por extraño que parezca, no proporciona características de correo electrónico. Es más que un preprocesador que toma como entrada texto llano y produce como salida texto cifrado firmado en base64. Por supuesto, después esta salida puede enviarse por correo electrónico. Algunas implementaciones de PGP llaman a un agente de usuario como paso final para enviar realmente el mensaje.

Para ver cómo funciona PGP, consideremos el ejemplo que se muestra en la figura 8-44. Aquí, Alice desea enviar de forma segura un mensaje en texto llano firmado, *P*, a Bob. Tanto Alice

como Bob tienen claves públicas (E_X) y privadas (D_X) RSA. Supongamos que cada uno conoce la clave pública del otro; analizaremos la administración de claves PGP dentro de poco.

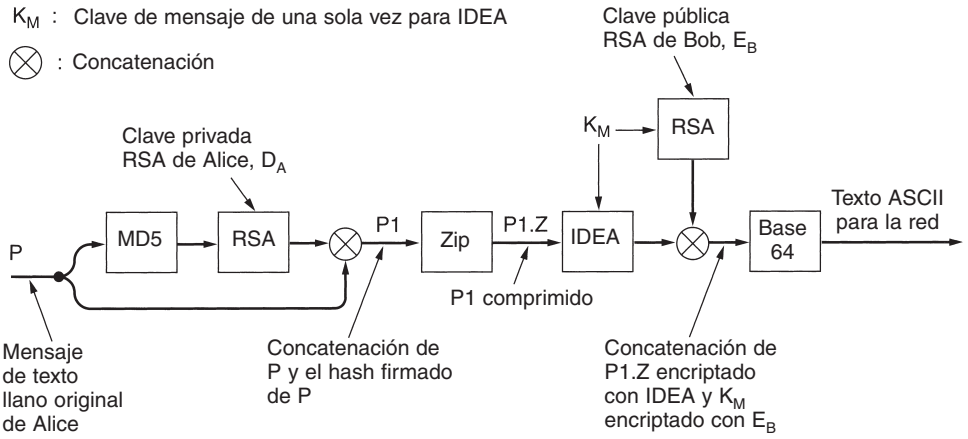


Figura 8-44. PGP en funcionamiento para enviar un mensaje.

Alice empieza por abrir el programa PGP en su computadora. Lo primero que hace PGP es aplicar un *hash* al mensaje, *P*, mediante MD5, y después encripta el *hash* resultante mediante su clave privada RSA, D_A. Cuando Bob obtiene el mensaje, puede desencriptar el *hash* con la clave pública de Alice y verificar que dicho *hash* sea correcto. Incluso si alguien más (por ejemplo, Trudy) pudiera adquirir el *hash* en esta etapa y desencriptarlo con la clave pública conocida de Alice, la fuerza de MD5 garantiza que desde un punto de vista computacional sea imposible producir otro mensaje con el mismo *hash* MD5.

El *hash* encriptado y el mensaje original ahora están concatenados en un solo mensaje, *P1*, y se comprimen mediante un programa ZIP, que utiliza el algoritmo Ziv-Lempel (Ziv y Lempel, 1977). Llame *P1.Z* a la salida de este paso.

A continuación, PGP pide a Alice alguna entrada aleatoria. Tanto el contenido como la velocidad de tecleo se utilizan para generar una clave de mensaje IDEA de 128 bits, K_M (llamada clave de sesión en la literatura de PGP, pero éste realmente es un nombre inapropiado puesto que no es una sesión). K_M ahora se utiliza para encriptar *P*1.*Z* con IDEA en modo de retroalimentación de cifrado. Además, K_M se encripta con la clave pública de Bob, E_B. Estos dos componentes después se concatenan y convierten a base64, como analizamos en la sección sobre MIME en el capítulo 7. El mensaje resultante contiene sólo letras, dígitos y los símbolos +, / e =, lo que significa que puede colocarse en el cuerpo del RFC 822 y se puede esperar que llegue sin modificaciones.

Cuando Bob obtiene el mensaje, invierte la codificación base64 y desencripta la clave IDEA mediante su clave privada RSA. Bob puede utilizar esta clave para desencriptar el mensaje a fin

de obtener *P1.Z*. Después de descomprimirlo, Bob separa el texto llano del *hash* encripta y desencripta este último meditante la clave pública de Alice. Si el *hash* de texto llano coincide con su propio cálculo MD5, Bob sabe que *P* es el mensaje correcto y que proviene de Alice.

Vale la pena señalar que aquí RSA sólo se utiliza en dos lugares: para encriptar el *hash* MD5 de 128 bits y para desencriptar la clave IDEA de 128 bits. Aunque RSA es lento, sólo tiene que encriptar 256 bits, no un gran volumen de datos. Además, los 256 bits de texto llano son en extremo aleatorios, por lo que se necesitará que Trudy realice una cantidad considerable de trabajo para determinar si una clave adivinada es correcta. IDEA realiza la encriptación pesada, que es varios órdenes de magnitud más rápida que RSA. Por lo tanto, PGP proporciona seguridad, compresión y una firma digital y lo hace en una forma más eficiente que el esquema ilustrado en la figura 8-19.

PGP soporta cuatro longitudes de clave RSA. Es responsabilidad del usuario seleccionar la más adecuada. Las longitudes son:

1. Casual (384 bits): En la actualidad puede romperse con facilidad.
2. Commercial (512 bits): Organizaciones de tres letras pueden romperla.
3. Military (1024 bits): Nadie de este mundo puede romperla.
4. Alien (2048 bits): Ni siquiera alguien de otro planeta puede romperla.

Puesto que RSA sólo se utiliza para dos pequeños cálculos, todo mundo debería utilizar las claves de fuerza alien todo el tiempo.

En la figura 8-45 se muestra el formato de un mensaje PGP clásico. También se utilizan otros formatos. El mensaje tiene tres partes, las cuales contienen la clave IDEA, la firma y el mensaje, respectivamente. La parte de la clave no sólo contiene ésta, sino también un identificador de clave, puesto que se permite que los usuarios tengan múltiples claves públicas.

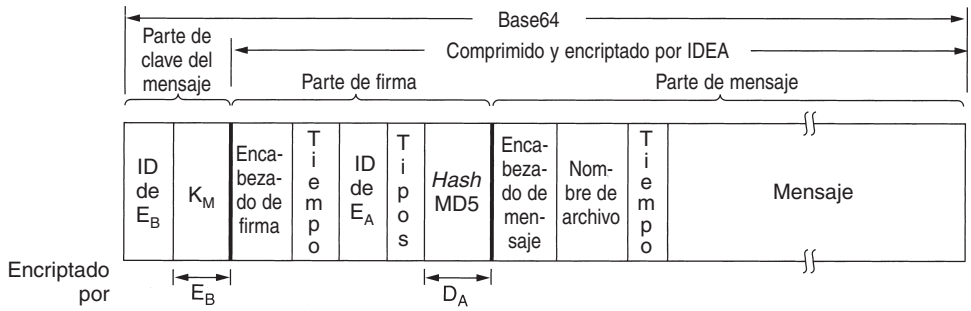


Figura 8-45. Un mensaje PGP.

La parte de firma contiene un encabezado, el cual no trataremos aquí. El encabezado está seguido por una marca de tiempo, el identificador para la clave pública del emisor que puede utili-

zarse para desencriptar el *hash* de firma, alguna información de tipos que identifica los algoritmos utilizados (para permitir que MD6 y RSA2 se puedan utilizar cuando se inventen) y el *hash* encriptado mismo.

La parte del mensaje también contiene un encabezado, el nombre predeterminado del archivo que se va a utilizar si el receptor escribe el archivo en disco, una marca de creación de mensajes y, finalmente, el mensaje mismo.

La administración de claves ha recibido una gran cantidad de atención en PGP puesto que es el talón de Aquiles de los sistemas de seguridad. La administración de claves funciona como sigue. Cada usuario mantiene localmente dos estructuras de datos: un anillo de clave privada y un anillo de clave pública. El **anillo de clave privada** contiene uno o más pares de claves pública-privada personales. La razón para soportar múltiples pares por usuario es permitir que los usuarios cambien sus claves públicas periódicamente o cuando piensen que han sido comprometidas, sin invalidar los mensajes que están actualmente en preparación o en tránsito. Cada par tiene asociado un identificador asociado, por lo que un emisor de mensajes puede indicarle al destinatario cuál clave pública se utilizó para encriptarlo. Los identificadores de mensajes consisten en los 64 bits de orden menor de la clave pública. Los usuarios son responsables de evitar conflictos en sus identificadores de clave pública. Las claves privadas en disco se encriptan mediante una contraseña especial (de tamaño arbitrario) para protegerlas contra ataques de robo.

El **anillo de clave pública** contiene claves públicas de los contactos del usuario. Éstas son necesarias para encriptar las claves de mensaje asociadas con cada mensaje. Cada entrada del anillo de clave pública no sólo contiene la clave pública, sino también su identificador de 64 bits y una indicación de qué tanto confía el usuario en la clave.

El problema que se trata de solucionar aquí es el siguiente. Suponga que las claves públicas se mantienen en boletines electrónicos. Un forma para que Trudy pueda leer el correo electrónico secreto de Bob es atacar el boletín electrónico y reemplazar la clave pública de Bob con una de su elección. Cuando más tarde Alice obtiene la clave que supuestamente pertenece a Bob, Trudy puede atacar a Bob mediante un ataque de brigada de bomberos.

Para evitar tales ataques, o por lo menos minimizar las consecuencias de ellos, Alice necesita saber cuánto debe confiar en el elemento llamado "clave de Bob" de su anillo de clave pública. Si Alice sabe que Bob le proporcionó personalmente un disco flexible que contiene la clave, puede establecer el valor confiable al valor más alto. Este enfoque controlado por el usuario y descentralizado hacia la administración de claves públicas es el que mantiene apartado a PGP de los esquemas centralizados PKI.

Sin embargo, las personas algunas veces obtienen claves públicas cuando consultan al servidor de claves confiable. Por esta razón, después de que X.509 se estandarizó, PGP soportó estos certificados, así como el mecanismo de anillo de clave pública de PGP. Todas las versiones actuales de PGP soportan X.509.

8.8.2 PEM—Correo con Privacidad Mejorada

En contraste con PGP, que inicialmente fue idea de un solo hombre, nuestro segundo ejemplo, **PEM (Correo con Privacidad Mejorada),** desarrollado a finales de la década de 1980, es un

estándar oficial de Internet que se describe en cuatro RFCs: del RFC 1421 al 1424. PEM cubre más o menos el mismo territorio que PGP: privacidad y autenticación para los sistemas de correo electrónico basados en el RFC 822. Sin embargo, también tiene algunas diferencias con respecto a PGP en lo que se refiere a metodología y tecnología.

Los mensajes enviados mediante PEM primero se convierten en una forma canónica a fin de que tengan las mismas convenciones sobre los espacios en blanco (por ejemplo, tabuladores, espacios en blanco). Después se calcula un *hash* de mensaje mediante MD2 o MD5, y la concatenación del *hash* y del mensaje se encripta mediante DES. Puesto que se sabe que una clave de 56 bits es débil, esta opción no es muy conveniente. El mensaje encriptado puede codificarse más tarde con codificación base64 y transmitirse al destinatario.

Al igual que en PGP, cada mensaje se encripta con una clave de una sola vez que se encierra junto con el mensaje. La clave puede protegerse con el RSA o con el DES triple mediante EDE.

La administración de claves está más estructurada que en PGP. Las claves están certificadas por certificados X.509 emitidos por CAs, que están ordenados en una jerarquía rígida que comienza en una sola raíz. La ventaja de este esquema es que es posible la revocación de certificados al hacer que la raíz emita CRLs de manera periódica.

El único problema con PEM es que nadie lo ha utilizado y hace mucho tiempo que se fue al cielo de los bits. El problema es principalmente político: ¿quién podría operar la raíz y bajo qué condiciones? No faltaban candidatos, pero las personas estaban temerosas de confiar a alguna compañía la seguridad de todo el sistema. El candidato más serio, RSA Security, Inc., quería cobrar por certificado emitido. Sin embargo, algunas organizaciones se opusieron a esta idea. En particular, el gobierno de los Estados Unidos tiene permitido utilizar sin costo alguno todas las patentes, y las compañías fuera de Estados Unidos se habían acostumbrado a utilizar de manera gratuita el algoritmo RSA (la compañía olvidó patentarla fuera de Estados Unidos). Nadie estaba feliz por tener que pagar de repente a RSA Security, Inc., por algo que siempre se había realizado de manera gratuita. Al final, no se encontró ninguna raíz y PEM se colapsó.

8.8.3 S/MIME

La siguiente aventura del IETF en lo que se refiere a la seguridad de correo electrónico, llamada **S/MIME** (**MIME Seguro**), se describe en los RFCs 2632 al 2643. Al igual que PEM, proporciona autenticación, integridad de datos, confidencialidad y no repudio. También es muy flexible, pues soporta una variedad de algoritmos criptográficos. No es sorprendente, dado el nombre, que S/MIME se integre bien con MIME, lo cual permite la protección de todos los mensajes. Se define una gran variedad de encabezados MIME, por ejemplo, para contener firmas digitales.

IETF definitivamente aprendió algo de la experiencia de PEM. S/MIME no tiene una jerarquía de certificados rígida que comienza en una sola raíz. En su lugar, los usuarios pueden tener múltiples anclas confiables. Un certificado se considera válido siempre y cuando pueda ser rastreado hacia alguna ancla en la que el usuario confíe. S/MIME utiliza los algoritmos y protocolos estándar que hemos examinado hasta ahora, por lo que no los trataremos aquí. Para más detalles, por favor consulte los RFCs.

8.9 SEGURIDAD EN WEB

Hemos estudiado dos áreas importantes en donde es necesaria la seguridad: las comunicaciones y el correo electrónico. Puede considerarlos como la sopa y el entremés. Ahora es tiempo del platillo principal: la seguridad en Web. Web es el lugar donde la mayoría de las Trudies vagan en la actualidad y tratan de realizar trabajo sucio. En las siguientes secciones analizaremos algunos de los problemas y cuestiones que se relacionan con la seguridad en Web.

La seguridad en Web se puede dividir en tres partes. La primera, ¿cómo se nombran de manera segura los objetos y los recursos? Segunda, ¿cómo pueden establecerse las conexiones seguras y autenticadas? Tercera, ¿qué sucede cuando un sitio Web envía a un cliente una pieza del código ejecutable? Después de ver algunas amenazas, examinaremos todas estas cuestiones.

8.9.1 Amenazas

Casi semanalmente, uno lee en los periódicos los problemas de seguridad en los sitios Web. La situación es muy sombría. Veamos algunos ejemplos de lo que ha sucedido en realidad. Primero, la página de inicio de numerosas organizaciones ha sido atacada y reemplazada por una nueva página de inicio de la elección de los *crackers*. (La prensa popular llama "hackers" a las personas que irrumpen en las computadoras, pero muchos programadores reservan ese término para los grandes programadores. Nosotros preferimos llamar "crackers" a las personas que irrumpen sin permiso a la computadora de otra persona.) Entre los sitios que han sido atacados por los *crackers* se incluyen Yahoo, el Ejército de Estados Unidos, la CIA, la NASA y el *New York Times*. En muchos casos, los *crackers* simplemente colocaron algún texto gracioso y los sitios fueron reparados en algunas horas.

Ahora veamos algunos casos más serios. Muchos sitios han sido derribados por ataques de negación de servicio, en los que el *cracker* inunda con tráfico el sitio, dejándolo incapaz de responder a consultas legítimas. Con frecuencia, el ataque se realiza desde varias máquinas en las que el *cracker* ya ha irrumpido (ataques DDoS). Estos ataques son tan comunes que ya no son noticia, pero pueden costar al sitio atacado miles de dólares en negocios perdidos.

En 1999, un *cracker* sueco irrumpió en el sitio Web Hotmail de Microsoft y creó un sitio espejo que permitió que cualquiera tecleara el nombre de un usuario de Hotmail y leyera todo el correo electrónico de la persona correspondiente.

En otro caso, un *cracker* ruso de 19 años llamado Maxim irrumpió en un sitio Web de comercio y robó 300,000 números de tarjetas de crédito. Después contactó a los dueños del sitio y les dijo que si no le pagaban $100,000, publicaría en Internet todos los números de las tarjetas. Los dueños no cedieron a su chantaje y Maxim publicó dichos números, dañando a muchas víctimas inocentes.

Otro caso fue el de un estudiante de California de 23 años que envió por correo electrónico un comunicado de prensa a una agencia de noticias en el que declaraba falsamente que Emulex Corporation iba a publicar una gran pérdida trimestral y que el director general iba a renunciar

inmediatamente. En pocas horas, las acciones de la compañía cayeron en 60%, causando que los inversionistas perdieran más de $2 mil millones. El autor de esta noticia ganó un cuarto de millón de dólares al vender las acciones poco antes de enviar el anuncio. Si bien este evento no tuvo que ver con que alguien irrumpiera en un sitio Web, es claro que colocar un anuncio de tal magnitud en una página de inicio de cualquier corporación importante podría tener un efecto similar.

(Desgraciadamente) podríamos seguir contando muchos casos como éstos, pero es tiempo de que examinemos algunos de los aspectos técnicos relacionados con la seguridad en Web. Para mayor información sobre los problemas de seguridad de todo tipo, vea (Anderson, 2001; Garfinkel con Spafford, 2002, y Schneier, 2000). En Internet podría encontrar una gran cantidad de casos específicos.

8.9.2 Asignación segura de nombres

Comencemos con algo muy básico: Alice desea visitar el sitio Web de Bob. Ella teclea el URL de Bob en su navegador y segundos más tarde, aparece una página Web. Pero, ¿es la de Bob? Tal vez. Trudy podría estar haciendo sus viejos trucos nuevamente. Por ejemplo, tal vez esté interceptando y examinando todos los paquetes salientes de Alice. Cuando captura una solicitud *GET* de HTTP dirigida al sitio Web de Bob, puede ir ella misma a dicho sitio para obtener la página, modificarla como lo desea y regresar la página falsa a Alice, la cual no sería extraña para Alice. Peor aún, Trudy podría recortar los precios de la tienda electrónica de Bob para hacer que los precios de los productos parezcan muy atractivos, con lo que engañaría a Alice para que enviara su número de tarjeta de crédito a "Bob" para comprar algo de mercancía.

Una desventaja de este clásico ataque de hombre en medio es que Trudy tiene que estar en una posición para interceptar el tráfico saliente de Alice y falsificar su tráfico entrante. En la práctica, tiene que intervenir la línea telefónica de Alice o la de Bob, puesto que intervenir la red dorsal de fibra es mucho más difícil. Aunque intervenir la línea es ciertamente posible, también significa mucho trabajo, y aunque Trudy es inteligente, también es floja. Además, hay formas más fáciles de engañar a Alice.

Falsificación de DNS

Por ejemplo, suponga que Trudy es capaz de violar el sistema DNS, tal vez sólo el caché DNS del ISP de Alice, y reemplazar la dirección IP de Bob (digamos, 36.1.2.3) con la suya propia (digamos, 42.9.9.9). Eso lleva al siguiente ataque. En la figura 8-46(a) se ilustra la forma en que se supone debe trabajar. Aquí (1) Alice pide al DNS la dirección IP de Bob, (2) la obtiene, (3) le pide a Bob su página de inicio, y (4) también la obtiene. Después de que Trudy ha modificado el registro DNS de Bob para contener su propia dirección IP en lugar de la de Bob, obtenemos la situación de la figura 8-46(b). Aquí, cuando Alice busca la dirección IP de Bob, obtiene la de Trudy, por lo que todo su tráfico dirigido a Bob va a parar a las manos de Trudy. Ahora, ella puede iniciar un ataque de hombre en medio sin tener que intervenir ninguna línea telefónica. En su lugar, tiene que irrumpir en un servidor DNS y cambiar un registro, lo cual es más fácil.

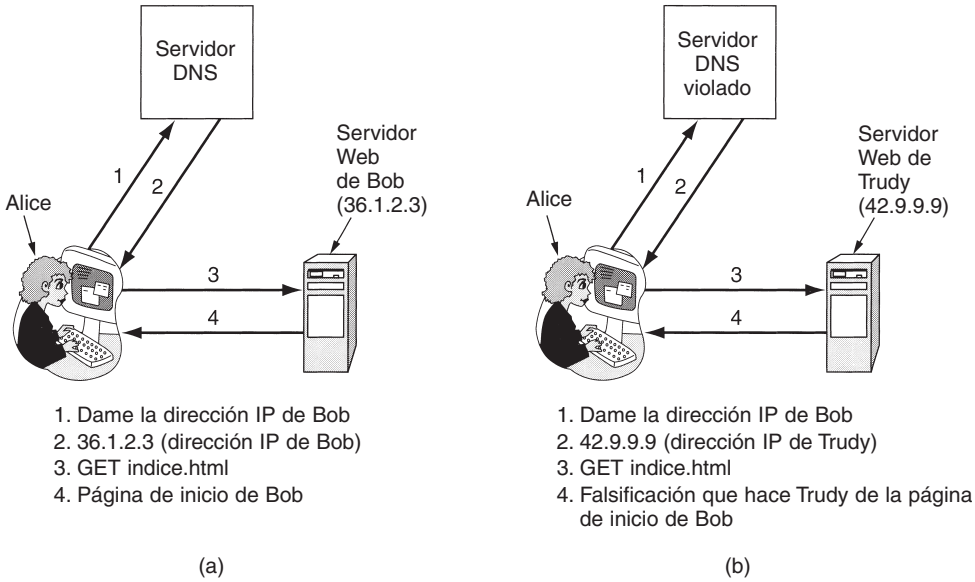


Figura 8-46. (a) Situación normal. (b) Un ataque basado en la violación de un DNS para modificar el registro de Bob.

¿Cómo engaña Trudy al DNS? Parece bastante fácil. Brevemente, Trudy puede engañar al servidor DNS en el ISP de Alice para que envíe una consulta a fin de encontrar la dirección de Bob. Desgraciadamente, puesto que el DNS utiliza UDP, el servidor DNS no puede verificar quién proporcionó la respuesta. Trudy puede explotar esta propiedad falsificando la respuesta esperada e introduciendo una dirección IP falsa en el caché del servidor DNS. Por simplicidad, supondremos que el ISP de Alice inicialmente no tiene una entrada para el sitio Web de Bob, *bob.com*. Si la tiene, Trudy puede esperar hasta que expire y probar otra vez (o utilizar otros trucos).

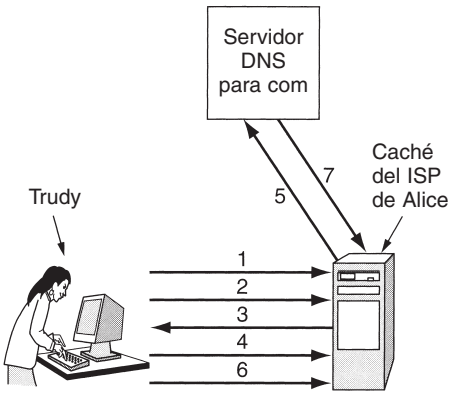
Trudy inicia el ataque enviando una solicitud de respuesta al ISP de Alice preguntando por la dirección IP de *bob.com*. Puesto que no hay entrada para este nombre DNS, el servidor de caché pide al servidor de nivel superior el dominio *com* para obtener uno. Sin embargo, Trudy evade al servidor *com* y regresa una respuesta falsa diciendo: "*bob.com* es 42.9.9.9"; sin embargo, esta dirección IP es la de ella. Si la respuesta falsa de Trudy llega primero al ISP de Alice, se almacenará en caché y la respuesta real se rechazará como respuesta no solicitada de una consulta que ya no está pendiente. Engañar a un servidor DNS para que instale una dirección IP falsa se conoce como **falsificación de DNS.** Un caché que contiene una dirección IP intencionalmente falsa como ésta se conoce como **caché envenenado.**

Por lo general, las cosas no son tan sencillas. Primero, el ISP de Alice verifica si la respuesta lleva la dirección de origen IP correcta del servidor de nivel superior. Pero debido a que Trudy puede colocar lo que quiera en ese campo IP, puede vencer esa prueba con facilidad puesto que las direcciones IP de los servidores de nivel superior tienen que ser públicas.

Segundo, para permitir que los servidores DNS digan cuál respuesta corresponde a cual solicitud, todas las solicitudes llevan un número de secuencia. Para falsificar el ISP de Alice, Trudy tiene que conocer su número de secuencia actual. La forma más fácil para que Trudy conozca el número de secuencia actual es que ella misma registre un dominio, digamos, *trudy-la-intrusa.com*. Supongamos que su dirección IP también es 42.9.9.9. Trudy también crea un servidor DNS para su nuevo dominio, *dns.trudy-la-intrusa.com*. El servidor DNS utiliza la dirección IP 42.9.9.9 de Trudy, puesto que ésta sólo tiene una computadora. Ahora Trudy tiene que informarle al ISP de Alice sobre su servidor DNS. Esto es fácil. Todo lo que tiene que hacer es pedir *foobar.trudy-la-intrusa.com* al IPS de Alice, lo que causará que dicho ISP pregunte al servidor *com* de nivel superior quién proporciona el nuevo dominio de Trudy.

Una vez que tiene seguro el *dns.trudy-la-intrusa.com* en el caché del ISP de Alice, el ataque real puede comenzar. Trudy ahora solicita *www.trudy-la-intrusa.com* al ISP de Alice. Como es natural, el ISP envía al servidor DNS de Trudy una consulta en la que se lo pide. Dicha consulta lleva el número de secuencia que Trudy está buscando. Rápidamente, Trudy pide al ISP de Alice que busque a Bob. Trudy responde de inmediato su propia pregunta enviando al ISP una respuesta falsificada, supuestamente del servidor *com* de nivel superior que dice: "*bob.com* es 42.9.9.9". Esta respuesta falsificada lleva un número de secuencia, un número más que el que acaba de recibir. Mientras esté allí, puede enviar una segunda falsificación con un número de secuencia incrementado en dos, y tal vez una docena más con números de secuencia crecientes. Uno de ellos está obligado a coincidir. El resto simplemente se elimina. Cuando llega la respuesta falsificada de Alice, se almacena en caché; cuando la respuesta real viene más tarde, se rechaza puesto que ninguna consulta está pendiente.

Cuando Alice busca a *bob.com*, se le indica que utilice 42.9.9.9, la dirección de Trudy. Ésta ha realizado un ataque hombre en medio desde la comodidad de su casa. En la figura 8-47 se ilustran los diversos pasos para este ataque. Para empeorar las cosas, ésta no es la única forma de falsificar un DNS. Hay muchas otras formas.



1. Busca foobar.trudy-la-intrusa.com (para obligarlo a entrar al caché del ISP)
2. Busca www.trudy-la-intrusa.com (para obtener el siguiente número de secuencia del ISP)
3. Solicita www.trudy-la-intrusa.com (lleva el siguiente número de secuencia del ISP, n)
4. Rápidamente, busca bob.com (para obligar a que el ISP consulte al servidor com en el paso 5)
5. Consulta legítima para bob.com con seq = n+1
6. Respuesta falsificada de Trudy: Bob es 42.9.9.9, seq = n+1
7. Respuesta real (rechazada, demasiado tarde)

Figura 8-47. La forma en que Trudy falsifica el ISP de Alice.

DNS seguro

Este ataque específico puede frustrarse al hacer que los servidores DNS utilicen IDs aleatorios en cualquiera de las consultas en lugar de sólo contar, pero parece que cada vez que se tapa un hoyo, otro se destapa. El problema real es que el DNS se diseñó en la época en la que Internet era una herramienta de investigación para algunos cientos de universidades y ni Alice, ni Bob, ni mucho menos Trudy fueron invitados a la fiesta. En ese entonces la seguridad no era un problema; hacer que Internet funcionara era el problema. Con los años, el entorno ha cambiado radicalmente, por lo que en 1994 el IETF estableció un grupo funcional para hacer que DNS fuera seguro. Este proyecto se conoce como **DNSsec (seguridad DNS);** su salida se presenta en el RFC 2535. Desgraciadamente, DNSsec aún no se ha distribuido por completo, por lo que numerosos servidores DNS aún son vulnerables a ataques de falsificación.

DNSsec es en concepto muy sencillo. Se basa en la criptografía de clave pública. Cada zona DNS (en el sentido de la figura 7-4) tiene un par de claves pública/privada. Toda la información enviada por un servidor DNS se firma con la clave privada de la zona originaria, por lo que el receptor puede verificar su autenticidad.

DNSsec ofrece tres servicios fundamentales:

1. Prueba de en dónde se originaron los datos.
2. Distribución de la clave pública.
3. Autenticación de transacciones y solicitudes.

El servicio principal es el que se lista primero, el cual verifica que los datos que se están regresando ha sido probado por el dueño de la zona. El segundo es útil para almacenar y recuperar de manera segura claves públicas. El tercero es necesario para proteger contra ataques de repetición y falsificación. Observe que la confidencialidad no es un servicio ofrecido puesto que toda la información en el DNS se considera como pública. Puesto que la planificación en etapas del DNSsec se lleva algunos años, es esencial la capacidad que tienen los servidores concientes de la seguridad para interactuar con los servidores que no están conscientes de ella, lo cual implica que el protocolo no puede cambiarse. Veamos ahora algunos de los detalles.

Los registros DNS están agrupados en conjuntos llamados **RRSets (Conjuntos de Registro de Recursos),** en los que todos los registros tienen el mismo nombre, clase y tipo en un conjunto. Un RRSet puede contener múltiples registros *A*; por ejemplo, si un nombre DNS se resuelve en una dirección IP primaria y una secundaria. Los RRSets se extienden con varios nuevos tipos de registro (que se analizan a continuación). A cada RRSet se le aplica un *hash* de manera criptográfica (por ejemplo, utilizando MD5 o SHA-1). La clave privada de la zona firma (por ejemplo, utilizando RSA) el *hash*. La unidad de transmisión a clientes es el RRSet firmado. Al momento de la recepción de un RRSet firmado, el cliente puede verificar si fue firmado por la clave privada de la zona originaria. Si la firma concuerda, se aceptan los datos. Puesto que cada RRSet contiene su propia firma, los RRSets pueden cambiarse en cualquier lugar, incluso en servidores no confiables, sin poner en peligro la seguridad.

DNSsec introduce varios tipos nuevos de registros. El primero de éstos es el registro *KEY*. Estos registros mantienen la clave pública de una zona, usuario, *host* u otro personaje principal, el algoritmo criptográfico utilizado para firmar, el protocolo utilizado para transmisión y otros bits. La clave pública se almacena tal como está. Los certificados X.509 no se utilizan debido a su volumen. El campo de algoritmo contiene un 1 para las firmas MD5/RSA (la opción preferida) y otros valores para otras combinaciones. El campo de protocolo puede indicar el uso de IPsec y otros protocolos de seguridad, si es que hay.

SIG es el segundo nuevo tipo de registro. Contiene el *hash* firmado de acuerdo con el algoritmo especificado en el registro *KEY.* La firma se aplica a todos los registros del RRSet, incluyendo a cualquiera de los registros *KEY* presentes, pero excluyéndose a sí mismo. También contiene la fecha en la que la firma comienza su periodo de validación y cuando ésta expira, así como el nombre de quien firma y algunos otros elementos.

El diseño de DNSsec tiene la característica de que es posible mantener sin conexión una clave privada de una zona. Una o dos veces al día, el contenido de una base de datos de una zona puede transportarse de manera manual (por ejemplo, en CD-ROM) a una máquina desconectada en la que se localiza una clave privada. Todos los RRSets pueden firmarse ahí y los registros *SIG* producidos de esa manera pueden transportarse al servidor primario de la zona en CD-ROM. De esta manera, la clave privada puede almacenarse en un CD-ROM guardado en forma segura excepto cuando se inserta en la máquina desconectada para firmar los nuevos RRSets del día. Después de que se termina el proceso de firma, todas las copias de la clave se eliminan de la memoria y el disco y el CD-ROM son guardados. Este procedimiento reduce la seguridad electrónica a seguridad física, algo que las personas saben cómo tratar.

Este método de prefirmar los RRSets aumenta la velocidad de manera considerable el proceso de responder consultas puesto que no es necesario realizar criptografía sobre la marcha. La desventaja consiste en que se necesita una gran cantidad de espacio en disco para almacenar todas las claves y firmas en las bases de datos DNS. Algunos registros incrementarán diez veces el tamaño debido a la firma.

Cuando un proceso de cliente obtenga un RRSet firmado, debe aplicar la clave pública de la zona originaria para desencriptar el *hash*, calcular el *hash* mismo y comparar los dos valores. Si concuerdan, los datos se consideran válidos. Sin embargo, este procedimiento asume la manera de cómo obtiene el cliente la clave pública de la zona. Una forma es adquirirla de un servidor confiable, utilizando una conexión segura (por ejemplo, utilizando IPsec).

Sin embargo, en la práctica, se espera que los clientes serán preconfigurados con las claves públicas de todos los dominios de nivel superior. Si Alice ahora desea visitar el sitio Web de Bob, puede pedir al DNS el RRSet de *bob.com*, que contendrá su dirección IP y un registro *KEY* que contiene la clave pública de Bob. Este RRSet será firmado por el dominio *com* de nivel superior, por lo que Alice puede verificar fácilmente su validez. Un ejemplo de lo que podría contener este RRSet se muestra en la figura 8-48.

Una vez que Alice tiene una copia verificada de la clave pública de Bob, puede pedir al servidor DNS de Bob la dirección IP de *www.bob.com*. La clave privada de Bob firmará este RRSet, a fin de que Alice pueda verificar la firma del RRSet que Bob regresa. Si Trudy hace algo para inyectar un RRSet falso en cualquiera de los cachés, Alice puede detectar fácilmente su falta de autenticidad porque el registro *SIG* contenido será incorrecto.

Nombre del dominio	Tiempo de vida	Clase	Tipo	Valor
bob.com.	86400	IN	A	36.1.2.3
bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D...
bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C...

Figura 8-48. Un RRSet de ejemplo para *bob.com*. El registro *KEY* es la clave pública de Bob. El registro *SIG* es el *hash* firmado del servidor *com* de nivel superior de los registros *A* y *KEY* para verificar su autenticidad.

Sin embargo, DNSsec también proporciona un mecanismo criptográfico para enlazar una respuesta a una consulta específica, a fin de evitar el tipo de falsificación utilizado por Trudy en la figura 8-47. Esta medida (opcional) de antifalsificación agrega a la respuesta un *hash* del mensaje de consulta firmado con la clave privada del receptor. Puesto que Trudy no conoce la clave privada del servidor *com* de nivel superior, no puede falsificar una respuesta a una consulta que el ISP de Alice haya enviado a dicho servidor. Ciertamente Trudy puede conseguir que su respuesta regrese primero, pero será rechazada debido a la firma inválida sobre la consulta con *hash*.

DNSsec también soporta algunos otros tipos de registros. Por ejemplo, el registro *CERT* puede utilizarse para almacenar certificados (por ejemplo, X.509). Este registro ha sido proporcionado porque algunas personas desean cambiar un DNS en una PKI. Si esto realmente sucede, está por verse. Detendremos nuestro análisis de DNSsec aquí. Para mayores detalles, por favor consulte el RFC 2535.

Nombres autocertificables

El DNS seguro no es la única posibilidad para asegurar los nombres. Un método completamente diferente se utiliza en el **Sistema de Archivos Seguro** (Mazières y cols., 1999). En este proyecto, los autores diseñaron un sistema de archivos seguro y escalable a nivel mundial, sin modificar el DNS (estándar) y sin utilizar certificados o suponer la existencia de una PKI. En esta sección mostraremos la forma en que estas ideas pueden aplicarse a Web. De manera acorde, en la descripción que daremos a continuación, utilizaremos la terminología de Web en lugar de la del sistema de archivos. Pero para evitar cualquier posible confusión, mientras que este esquema *podría* aplicarse a Web a fin de lograr una máxima seguridad, actualmente no está en uso y podría requerir cambios sustanciales de software para introducirla.

Comenzaremos suponiendo que cada servidor Web tiene un par de claves pública/privada. La esencia de la idea es que cada URL contiene un *hash* criptográfico del nombre del servidor y una clave pública como parte del URL. Por ejemplo, en la figura 8-49 podemos ver el URL para la foto de Bob. Comienza con el esquema HTTP usual seguido por el nombre DNS del servidor (*www.bob.com*). A continuación se encuentra un punto y coma y un *hash* de 32 caracteres. Al final está el nombre del archivo, nuevamente como es usual. Excepto por el *hash*, éste es un URL estándar. Con el *hash*, es un **URL autocertificable.**

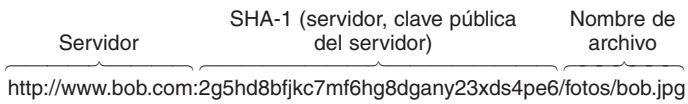


Figura 8-49. Un URL autocertificable que contiene un *hash* del nombre y la clave pública del servidor.

La pregunta obvia es: ¿Para qué es el *hash*? Éste se calcula concatenando el nombre DNS del servidor con la clave pública del servidor y ejecutando el resultado mediante la función SHA-1 para obtener un *hash* de 160 bits. En este esquema, el *hash* se representa como una secuencia de 32 dígitos y letras en minúsculas, a excepción de las letras "l" y "o" y los dígitos "1" y "0", para evitar la confusión. Esto deja 32 dígitos y letras. Con 32 caracteres disponibles, cada uno puede codificar una cadena de 5 bits. Una cadena de 32 caracteres puede contener el *hash* SHA-1 de 160 bits. Realmente, no es necesario utilizar un *hash*; puede utilizarse la clave misma. La ventaja del *hash* es que se reduce la longitud del nombre.

La forma más sencilla (pero menos conveniente) para ver la foto de Bob, es que Alice simplemente teclee la cadena de la figura 8-49 en su navegador. Éste envía un mensaje al sitio Web de Bob pidiéndole la clave pública. Cuando llega la clave pública de Bob, el navegador concatena el nombre del servidor y la clave pública y ejecuta el algoritmo de *hash*. Si el resultado concuerda con el *hash* de 32 caracteres en el URL seguro, el navegador está seguro de que tiene la clave pública de Bob. Después de todo, debido a las propiedades de SHA-1, incluso aunque Trudy intercepte la solicitud y falsifique la respuesta, no tendrá manera de encontrar una clave pública que dé el *hash* esperado. Por lo tanto, se detectará cualquier interferencia por parte de Trudy. La clave pública de Bob puede almacenarse en caché para uso futuro.

Ahora Alice tiene que verificar que Bob tiene la clave privada correspondiente. Ella construye un mensaje que contiene una clave de sesión AES propuesta, una marca aleatoria y una marca de tiempo. A continuación encripta el mensaje con la clave pública de Bob y se la envía. Puesto que sólo Bob tiene la clave privada correspondiente, únicamente él puede desencriptar el mensaje y regresar la marca aleatoria encriptada con la clave AES. Una vez que recibe la marca aleatoria AES encriptada correcta, Alice sabe que está hablando con Bob. Además, Alice y Bob ahora tienen una clave de sesión AES para las solicitudes y respuestas *GET* subsecuentes.

Una vez que Alice tiene la foto de Bob (o cualquier página Web), puede marcarla, por lo que ya no tiene que teclear nuevamente todo el URL. Además, los URLs incrustados en páginas Web también pueden ser autocertificables, por lo que pueden utilizarse con sólo hacer clic en ellos, pero con la seguridad adicional de que sabe que la página regresada es la correcta. Otras formas de evitar los URLs autocertificables son obtenerlos a través de una conexión segura a un servidor confiable o tenerlos presentes en certificados X.509 firmados por CAs.

Otra forma de obtener URLs autocertificables podría ser conectarse con una máquina de búsqueda confiable tecleando su URL autocertificable (la primera vez) e ir a través del mismo protocolo como se describió anteriormente, lo que produciría una conexión autenticada segura con la máquina de búsqueda confiable. A continuación podría consultarse dicha máquina de búsqueda, y los resultados aparecerían en una página firmada llena de URLs autocertificables en los que se podría hacer clic sin tener que teclear cadenas largas.

Ahora veamos qué efectos tiene este método en la falsificación de DNS de Trudy. Si Trudy se las arregla para envenenar el caché del ISP de Alice, la solicitud de ésta podría ser falsamente entregada a Trudy en lugar de a Bob. Pero el protocolo ahora requiere que el destinatario de un mensaje original (es decir, Trudy) regrese una clave pública que produce el *hash* correcto. Si Trudy regresa su propia clave pública, Alice lo detectará inmediatamente porque el *hash* SHA-1 no coincidirá con el URL autocertificable. Si Trudy regresa la clave pública de Bob, Alice no detectará el ataque, pero encriptará su siguiente mensaje, utilizando la clave de Bob. Trudy obtendrá el mensaje, pero no tendrá forma de desencriptarlo para extraer la clave AES y la marca aleatoria. De cualquier forma, todo lo que la falsificación DNS puede hacer es proporcionar un ataque de negación de servicio.

8.9.3 SSL—La Capa de Sockets Seguros

La asignación de nombres segura es un buen inicio, pero hay mucho más sobre la seguridad en Web. El siguiente paso son las conexiones seguras. A continuación veremos cómo pueden lograrse las conexiones seguras.

Cuando Web irrumpió a la vista pública, inicialmente sólo se utilizó para distribuir páginas estáticas. Sin embargo, pronto algunas compañías tuvieron la idea de utilizarla para transacciones financieras, como para comprar mercancía con tarjeta de crédito, operaciones bancarias en línea y comercio electrónico de acciones. Estas aplicaciones crearon una demanda de conexiones seguras. En 1995, Netscape Communications Corp, el entonces fabricante líder de navegadores, respondió a esta demanda con un paquete de seguridad llamado **SSL (Capa de Sockets Seguros).** En la actualidad, este software y su protocolo se utilizan ampliamente, incluso por Internet Explorer, por lo que vale la pena examinarlo en detalle.

SSL construye una conexión segura entre los dos sockets, incluyendo

1. Negociación de parámetros entre el cliente y el servidor.
2. Autenticación tanto del cliente como del servidor.
3. Comunicación secreta.
4. Protección de la integridad de los datos.

Ya hemos visto antes estos elementos, por lo que no hay necesidad de volver a tratarlos.

En la figura 8-50 se ilustra la posición de SSL en la pila de protocolos usuales. Efectivamente, es una nueva capa colocada entre la capa de aplicación y la de transporte, que acepta solicitudes del navegador y enviándolas al TCP para transmitir al servidor. Una vez que se ha establecido la conexión segura, el trabajo principal de SSL es manejar la compresión y encriptación. Cuando HTTP se utiliza encima de SSL, se conoce como **HTTPS (HTTP Seguro),** aunque es el protocolo HTTP estándar. Sin embargo, algunas veces está disponible en un nuevo puerto (443) en lugar de en uno estándar (80). Además, SSL no está restringido a utilizarse sólo con navegadores Web, pero ésa es la aplicación más común.

El protocolo SSL ha pasado por varias versiones. A continuación sólo trataremos la versión 3, que es la versión más ampliamente utilizada. SSL soporta una variedad de algoritmos y opciones

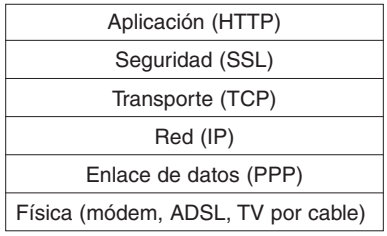


Figura 8-50. Capas (y protocolos) para una navegación de usuario doméstico con SSL.

diferentes. Entre dichas opciones se incluyen la presencia o ausencia de compresión, los algoritmos criptográficos a utilizar y algunos asuntos relacionados con la exportación de restricciones en la criptografía. La última es la destinada principalmente para asegurarse de que se utilice criptografía seria sólo cuando ambos lados de la conexión estén en los Estados Unidos. En otros casos, las claves se limitan a 40 bits, lo que los criptógrafos consideran como una broma. El gobierno de los Estados Unidos obligó a Netscape a incluir esta restricción para obtener una licencia de exportación.

SSL consiste en dos subprotocolos, uno para establecer una conexión segura y otro para utilizarla. Comencemos viendo cómo se establecen las conexiones seguras. En la figura 8-51 se muestra el subprotocolo de establecimiento de conexión. Comienza con el mensaje 1, cuando Alice envía una solicitud a Bob para que establezca una conexión. La solicitud especifica la versión SSL que tiene Alice y sus preferencias con respecto a los algoritmos criptográficos y de compresión. También contiene una marca aleatoria, R_A, para utilizarse más tarde.

Ahora es el turno de Bob. En el mensaje 2, Bob realiza una elección de entre los diversos algoritmos que Alice puede soportar y envía su propia marca aleatoria, R_B. Después, en el mensaje 3, envía un certificado que contiene su clave pública. Si este certificado no está firmado por alguna autoridad bien conocida, también envía una cadena de certificados que pueden seguirse hasta encontrar uno. Todos los navegadores, incluyendo el de Alice, vienen precargados con aproximadamente 100 claves públicas, por lo que si Bob puede establecer una cadena anclada a una de ellas, Alice podrá verificar la clave pública de Bob. En este punto Bob podría enviar algunos otros mensajes (por ejemplo, podría solicitar el certificado de la clave pública de Alice). Cuando Bob termina, envía el mensaje 4 para indicar a Alice que es su turno.

Alice responde eligiendo una **clave premaestra** aleatoria de 384 bits y enviándola a Bob encriptada con la clave pública de él (mensaje 5). La clave de sesión real utilizada para encriptar datos se deriva de la clave premaestra combinada con las dos marcas aleatorias en una forma compleja. Después de que se ha recibido el mensaje 5, tanto Alice como Bob pueden calcular la clave de sesión. Por esta razón, Alice indica a Bob que cambie al nuevo cifrado (mensaje 6) y también que ha terminado con el establecimiento del subprotocolo (mensaje 7). Después Bob confirma que ha recibido la indicación (mensajes 8 y 9).

Sin embargo, aunque Alice sabe quién es Bob, éste no sabe quién es Alice (a menos que Alice tenga una clave pública y un certificado correspondiente para ella, una situación no probable para un individuo). Por lo tanto, el primer mensaje de Bob puede ser una solicitud para que Alice inicie una sesión utilizando un nombre de inicio de sesión y una contraseña previamente establecidos.

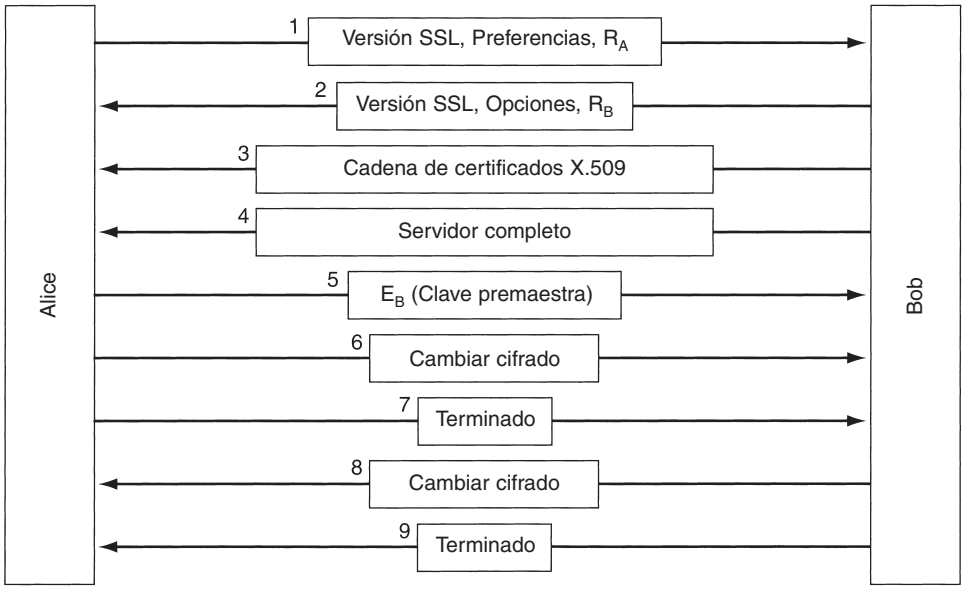


Figura 8-51. Una versión simplificada del subprotocolo de establecimiento de conexión SSL.

Sin embargo, el protocolo de inicio de sesión está fuera del alcance de SSL. Una vez que se ha consumado, por cualquier medio, puede comenzar el transporte de los datos.

Como se mencionó anteriormente, SSL soporta múltiples algoritmos criptográficos. El más robusto utiliza triple DES con tres claves separadas para encriptación y SHA-1 para la integridad de mensajes. Esta combinación es relativamente lenta, por lo que se utiliza principalmente para operaciones bancarias y otras aplicaciones en las que se requiere la seguridad de mayor nivel. Para las aplicaciones comunes de comercio electrónico, se utiliza RC4 con una clave de 128 bits para encriptación y MD5 se utiliza para la autenticación de mensajes. RC4 toma la clave de 128 bits como semilla y la expande a un número mucho más grande para uso interno. Después utiliza este número interno para generar un flujo de claves. A éste se le aplica un OR exclusivo con el texto llano para proporcionar un cifrado clásico de flujo, como vimos en la figura 8-14. Las versiones de exportación también utilizan RC4 con claves de 128 bits, 88 de los cuales se hacen públicos para que el cifrado sea fácil de romper.

Para un transporte real se utiliza un segundo subprotocolo, como se muestra en la figura 8-52. Los mensajes que provengan del navegador primero se dividen en unidades de hasta 16 KB. Si se activa la compresión, cada unidad se comprime por separado. Después de eso, se deriva una clave secreta a partir de las dos marcas aleatorias y la clave premaestra se concatena con el texto comprimido y al resultado se le aplica un *hash* con el algoritmo de *hash* acordado (por lo general MD5). Este *hash* se agrega a cada fragmento como el MAC. Después, el fragmento comprimido y el MAC se encriptan con el algoritmo de encriptación simétrico acordado (por lo general, aplicándole un OR exclusivo con el flujo de claves RC4). Por último, se agrega un encabezado de fragmento y el fragmento se transmite a través de la conexión TCP.

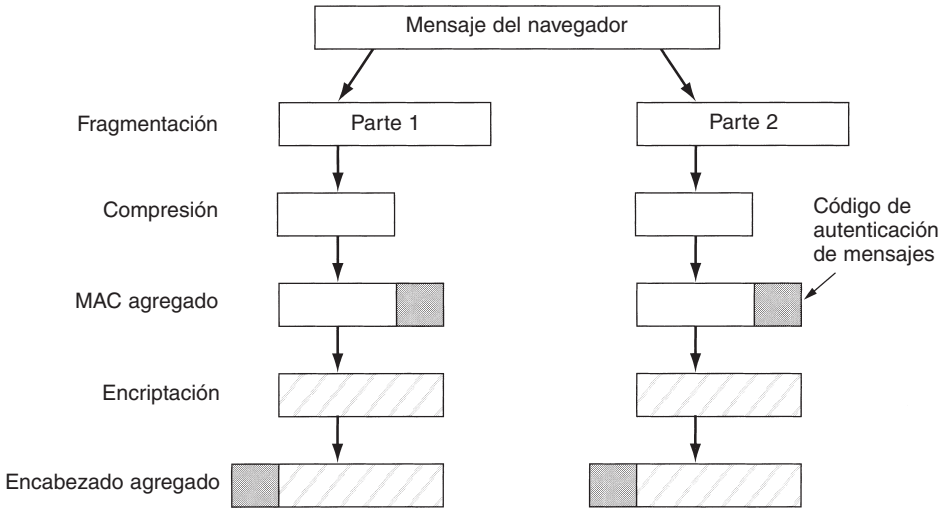


Figura 8-52. Transmisión de datos mediante SSL.

Sin embargo, es necesaria una advertencia. Puesto que se ha mostrado que el RC4 tiene claves débiles que pueden criptoanalizarse con facilidad, la seguridad de SSL mediante RC4 no es muy confiable (Fluhrer y cols., 2001). Los navegadores que permiten que el usuario elija el conjunto de cifrado deben configurarse para utilizar todo el tiempo triple DES con claves de 168 bits y SHA-1, aunque esta combinación es más lenta que RC4 y MD5.

Otro problema con SSL es que tal vez los personajes principales no tienen certificados e incluso si los tienen, no siempre verifican que coincidan las claves que se utilizan.

En 1996, Netscape Communications Corp. mandó el SSL a la IETF para su estandarización. El resultado fue **TLS** (**Seguridad de Capa de Transporte**)**.** Se describe en el RFC 2246.

Los cambios hechos a SSL fueron relativamente pequeños, pero sólo lo suficiente para que SSL versión 3 y TLS no puedan interoperar. Por ejemplo, la forma en que se deriva una clave de sesión a partir de la clave premaestra y las marcas aleatorias se cambió para hacer que la clave fuera más fuerte (es decir, fuera difícil de criptoanalizar). La versión TLS también se conoce como SSL versión 3.1. Las primeras implementaciones aparecieron en 1999, pero aún no es claro si TLS reemplazará a SSL en la práctica, aunque es ligeramente más fuerte. Sin embargo, permanece el problema con las claves débiles RC4.

8.9.4 Seguridad de código móvil

La asignación de nombres y las conexiones son dos áreas de preocupación que se relacionan con la seguridad en Web. Pero hay más. En los primeros días, cuando las páginas Web eran simplemente archivos HTML estáticos, no contenían código ejecutable. Ahora con frecuencia contienen pequeños programas, incluyendo subprogramas de Java, controles ActiveX y JavaScripts. Descargar y ejecutar tal **código móvil** obviamente es un riesgo de seguridad masivo, por lo que se han

diseñado varios métodos para minimizarlo. A continuación daremos un vistazo a algunos de los problemas surgidos debido al código móvil y a algunos métodos para tratarlos.

Seguridad de subprogramas de Java

Los subprogramas de Java son pequeños programas de Java compilados a un lenguaje de máquina orientado a pilas llamado **JVM** (**Máquina Virtual de Java**). Pueden colocarse en una página Web para descargarlos junto con dicha página. Después de que la página se carga, los subprogramas se insertan en un intérprete JVM dentro del navegador, como se ilustra en la figura 8-53.

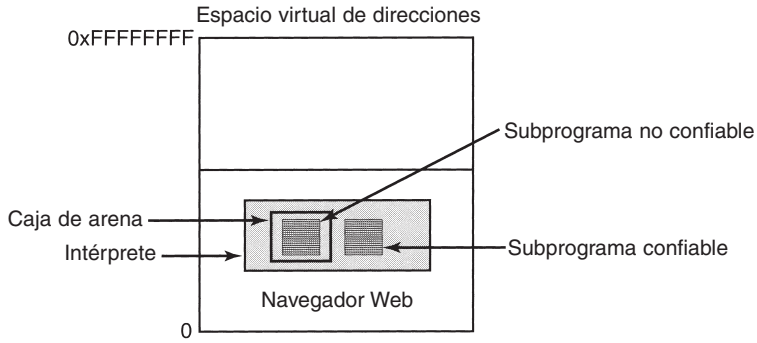


Figura 8-53. Los subprogramas pueden ser interpretados por un navegador Web.

La ventaja de ejecutar código interpretado en lugar de compilado es que cada instrucción es examinada por el intérprete antes de ser ejecutada. Esto da al intérprete la oportunidad de verificar si la dirección de la instrucción es válida. Además, las llamadas de sistema también son atrapadas e interpretadas. La forma en que se manejan estas llamadas depende de la política de seguridad. Por ejemplo, si un subprograma es confiable (por ejemplo, proviene del disco local), sus llamadas de sistema podrían transportarse sin pregunta alguna. Sin embargo, si un subprograma no es confiable (por ejemplo, proviene de Internet), podría encapsularse en lo que se conoce como **caja de arena** para restringir su comportamiento y atrapar sus intentos por utilizar los recursos del sistema.

Cuando un subprograma intenta utilizar un recurso del sistema, su llamada se pasa al monitor de seguridad para que la apruebe. El monitor examina la llamada a la luz de la política de seguridad local y después toma una decisión para permitirla o rechazarla. De esta manera, es posible proporcionar acceso a los subprogramas a algunos recursos pero no a todos. Desgraciadamente, la realidad es que el modelo de seguridad no funciona del todo bien y con frecuencia surgen errores.

ActiveX

Los controles ActiveX son programas binarios Pentium que pueden incrustarse en páginas Web. Cuando se encuentra alguno de ellos, se realiza una verificación para ver si se debe ejecutar, y si pasa la prueba, se ejecuta. No está interpretado o puesto en una caja de arena de ninguna

manera, por lo que tiene tanto poder que cualquier otro programa de usuario, y puede hacer mucho daño. Por lo tanto, toda la seguridad recae en la decisión de si se ejecuta o no el control ActiveX.

El método que Microsoft eligió para tomar esta decisión se basa en la idea de la **firma de código.** Cada control ActiveX está acompañado por una firma digital —un *hash* del código que está firmado por su creador utilizando la encriptación de clave pública. Cuando aparece un control ActiveX, el navegador primero verifica la firma para asegurarse de que no ha sido alterada en el camino. Si la firma es correcta, el navegador verifica a continuación sus tablas internas para ver si el creador del programa es confiable o hay una cadena de confianza hacia un creador confiable. Si el creador es confiable, el programa se ejecuta; de lo contrario, no se ejecuta. El sistema de Microsoft para verificar controles ActiveX se conoce como **Authenticode.**

Es útil comparar los métodos de Java y ActiveX. Con el método de Java, no se realiza ningún intento por determinar quién escribió el subprograma. En su lugar, un intérprete en tiempo de ejecución se asegura de que el subprograma no haga lo que el dueño de la máquina haya prohibido. En contraste, con la firma de código, no hay intento por supervisar el comportamiento en tiempo de ejecución del código móvil. Si proviene de un origen confiable y no ha sido modificado en el camino, simplemente se ejecuta. No se realiza ningún intento por ver si el código es maligno o no. Si el programador original *destinó* el código para formatear el disco duro y después borra la *flash* ROM a fin de que la computadora nunca vuelva a encenderse, y si el programador ha sido certificado como confiable, el código se ejecutará y destruirá la computadora (a menos que los controles ActiveX hayan sido desactivados en el navegador).

Muchas personas piensan que confiar en software de una compañía desconocida es temible. Para demostrar el problema, un programador de Seattle formó una compañía de software y la certificó como confiable, lo cual es fácil de hacer. Después escribió un control ActiveX que apagaba limpiamente la máquina y lo distribuyó ampliamente. Apagó muchas máquinas, pero podían encenderse nuevamente, por lo que no causó ningún daño. Simplemente trataba de exponer el problema al mundo. La respuesta oficial fue revocar el certificado a este control ActiveX específico, lo cual acabó con un corto episodio vergonzoso, pero el problema subyacente aún está allí para que un programador con no buenas intenciones se aproveche de él (Garfinkel y Spafford, 2002). Puesto que no hay forma de vigilar miles de compañías de software que podrían escribir código móvil, la técnica de firma de código es un desastre al acecho.

JavaScript

JavaScript no tiene ningún modelo de seguridad formal, pero tiene una larga historia de implementaciones defectuosas. Cada fabricante maneja la seguridad de forma diferente. Por ejemplo, la versión 2 de Netscape Navigator utilizó algo semejante al modelo de Java, pero en la versión 4 se abandonó por un modelo de firma de código.

El problema fundamental es que permitir la ejecución de código extraño en su máquina es abrir la puerta a los problemas. Desde el punto de vista de la seguridad, es como invitar a un ladrón a su casa y después tratar de vigilarlo cuidadosamente de manera que no pueda ir de la cocina

a la sala. Si algo inesperado sucede y está distraído por un momento, pueden suceder cosas extrañas. El suspenso aquí es que el código móvil permite gráficos vistosos e interacción rápida, y muchos diseñadores de sitios Web piensan que esto es más importante que la seguridad, especialmente cuando es la máquina de alguien más la que está en riesgo.

Virus

Los virus son otra forma de código móvil. Sólo que a diferencia de los ejemplos anteriores, los virus no se invitan de ninguna manera. La diferencia entre un virus y el código móvil ordinario es que los virus se escriben para que se reproduzcan a sí mismos. Cuando llegan los virus, ya sea por medio de una página Web, un archivo adjunto de correo electrónico, o algún otro medio, por lo general inician infectando los programas ejecutables del disco. Cuando se ejecuta alguno de estos programas, el control se transfiere al virus, que por lo general trata de esparcirse a sí mismo a otras máquinas, por ejemplo, enviando por correo electrónico copias de sí mismo a cualquier persona que se encuentre en la libreta de direcciones de la víctima. Algunos virus infectan el sector de arranque del disco duro, por lo que cuando la máquina se inicia, el virus se ejecuta. Los virus se han convertido en un gran problema en Internet y han causado que se pierdan miles de millones de dólares por los daños. No hay solución a este problema. Tal vez podría ayudar una nueva generación de sistemas operativos basados en *microkernels* seguros y el compartimiento de usuarios, procesos y recursos.

8.10 ASPECTOS SOCIALES

Internet y su tecnología de seguridad es un área donde confluyen los aspectos sociales, las políticas de seguridad y la tecnología, frecuentemente con consecuencias importantes. A continuación sólo examinaremos con brevedad tres áreas: privacidad, libertad de expresión y derechos de autor. No es necesario decir que sólo echaremos un vistazo. Para mayor información, vea (Anderson, 2001; Garfinkel con Spafford, 2002, y Schneier, 2000). En Internet también puede encontrar mucho material. Simplemente introduzca en cualquier máquina de búsqueda palabras como “privacidad”, “censura” y “derechos de autor”. Además, vea el sitio Web de este libro para obtener algunos vínculos.

8.10.1 Privacidad

¿Las personas tienen derecho a la privacidad? Buena pregunta. La Cuarta Enmienda de los Estados Unidos prohíbe que el gobierno busque en las casas, documentos y bienes de las personas sin una razón válida y restringe las circunstancias en las que se deben emitir las órdenes de cateo. Por lo tanto, la privacidad ha estado en la agenda pública aproximadamente durante 200 años, por lo menos en Estados Unidos.

En la década pasada cambió la facilidad con la que el gobierno puede espiar a sus ciudadanos y la facilidad con la que éstos pueden evitar tal espionaje. En el siglo XVIII, para que el gobierno pudiera buscar en los documentos de los ciudadanos, tenía que enviar un policía a caballo a la granja de dicho ciudadano exigiendo ver ciertos documentos. Era un procedimiento engorroso. Hoy en día, las compañías telefónicas y los proveedores de Internet proporcionan con facilidad intervenciones telefónicas cuando se les presenta una orden de cateo. Esto facilita la vida del policía y ya no hay peligro de que se caiga del caballo.

La criptografía cambia todo esto. Cualquiera que se tome la molestia de bajar e instalar PGP y quien utilice una clave bien custodiada de fuerza alien puede estar seguro de que nadie en el universo conocido puede leer su correo electrónico, haya o no orden de cateo. Los gobiernos entienden bien esto y no les gusta. La privacidad real para ellos significa que les será mucho más difícil espiar a los criminales de todo tipo, y todavía les será más difícil espiar a los reporteros y oponentes políticos. En consecuencia, algunos gobiernos restringen o prohíben el uso o exportación de la criptografía. Por ejemplo, en Francia, antes de 1999, la criptografía estaba prohibida a menos que se le proporcionaran las claves al gobierno.

Esto no sólo sucedía en Francia. En abril de 1993, el gobierno de Estados Unidos anunció su intención de hacer que un criptoprocesador de hardware, el **procesador clipper,** fuera el estándar en toda la comunicación en red. De esta manera, se dijo, la privacidad de los ciudadanos estaría garantizada. También se mencionó que el procesador proporcionaría al gobierno la capacidad de desencriptar todo el tráfico a través de un esquema llamado **depósito de claves,** el cual permitía que el gobierno accediera a todas las claves. Sin embargo, se prometió que sólo se espiaría cuando se tuviera una orden de cateo válida. No es necesario decir que se generó una gran controversia por esta situación, en la que los activistas de la privacidad condenaban todo el plan y los oficiales del cumplimiento de la ley la aclamaban. En algún momento, el gobierno se retractó y abandonó la idea.

En el sitio Web de la Electronic Frontier Foundation, *www.eff.org*, está disponible una gran cantidad de información acerca de la privacidad electrónica.

Retransmisores de correo anónimos

PGP, SSL y otras tecnologías hacen posible que dos partes establezcan comunicación segura y autenticada, libre de vigilancia e interferencia de terceros. Sin embargo, algunas veces la privacidad se aplica mejor cuando *no* hay autenticación, es decir, haciendo que la comunicación sea anónima. El anonimato podría quererse para los mensajes punto a punto, grupos de noticias o ambos.

Veamos algunos ejemplos. Primero, los disidentes políticos que viven bajo regímenes autoritarios con frecuencia desean comunicarse de manera anónima para evitar ser encarcelados o asesinados. Segundo, por lo general las acciones ilegales en muchas organizaciones gubernamentales, educacionales, corporativas, entre otras, son denunciadas por personas que con frecuencia prefieren permanecer en el anonimato para evitar represalias. Tercero, las personas con creencias religiosas, políticas y sociales impopulares podrían querer comunicarse entre ellas a través de correo electrónico o grupos de noticias sin exponerse a sí mismos. Cuarto, las personas podrían desear discutir el alcoholismo, las enfermedades mentales, el acoso sexual, el abuso a infantes o ser

miembros de una minoría perseguida de un grupo de noticias sin revelar su identidad. Por supuesto, existen muchos otros ejemplos.

Consideremos un ejemplo específico. En la década de 1990 algunos críticos publicaron sus puntos de vista sobre un grupo religioso no tradicional, en un grupo de noticias de USENET a través de un **retransmisor de correo anónimo.** Este servidor permitía que los usuarios crearan pseudónimos y le enviaran correo electrónico, y después dicho servidor volvía a enviar o publicar tal correo utilizando el pseudónimo creado, de manera que nadie podía saber de dónde provenía realmente el mensaje. Algunas publicaciones revelaron información que el grupo religioso afirmaba eran secretos comerciales y documentos con derechos de autor. Por lo tanto, dicho grupo fue con las autoridades locales y les dijo que sus secretos comerciales habían sido revelados y que sus derechos de autor habían sido violados, lo cual eran delitos en el lugar donde se localizaba el servidor. En consecuencia, se produjo un juicio y el operador del servidor fue obligado a entregar la información de correspondencia, la cual reveló las verdaderas identidades de las personas quienes realizaron las publicaciones. (Incidentalmente, ésta no fue la primera vez que una religión no estaba de acuerdo con que alguien revelara sus secretos: William Tyndale fue quemado en la hoguera en 1536 por traducir al inglés la Biblia.)

Un segmento considerable de la comunidad de Internet se indignó por esta brecha de confidencialidad. La conclusión a la que todos llegaron es que no sirve de nada un retransmisor anónimo que almacena una correspondencia entre las direcciones reales de correo electrónico y los pseudónimos (llamado retransmisor de correo de tipo 1). Este caso estimuló a varias personas a diseñar retransmisores de correo anónimos que pudieran resistir ataques de citaciones legales.

Estos nuevos retransmisores, con frecuencia llamados **retransmisores de correo cypherpunks,** funcionan como se describe a continuación. El usuario produce un mensaje de correo electrónico, lleno de encabezados RFC 822 (excepto *From*:, por supuesto), lo encripta con la clave pública del retransmisor y lo envía a éste. Ahí se eliminan los encabezados externos RFC 822, el contenido se desencripta y el mensaje es retransmitido. El retransmisor no tiene cuentas ni mantiene registros, por lo que aunque el servidor se confisque posteriormente, no contiene ni una huella de los mensajes que han pasado a través de él.

Muchos usuarios que desean el anonimato encadenan sus solicitudes a través de múltiples retransmisores anónimos, como se muestra en la figura 8-54. Aquí, Alice desea enviar a Bob una tarjeta del Día de San Valentín en verdad anónima, por lo que utiliza tres retransmisores. Redacta el mensaje, *M*, y coloca en él un encabezado que contiene la dirección de correo electrónico de Bob. Después encripta todo el mensaje con la clave pública del retransmisor 3, E_3 (indicado por el sombreado horizontal). Para esto anexa al principio un encabezado con la dirección de correo electrónico en texto llano del retransmisor 3. En la figura, este mensaje es el que se muestra entre los retransmisores 2 y 3.

A continuación encripta este mensaje con la clave pública del retransmisor 2, *E2* (indicado por el sombreado vertical) y anexa al principio un encabezado en texto llano que contiene la dirección de correo electrónico del retransmisor 2. Este mensaje se muestra en la figura 8-54 entre los retransmisores 1 y 2. Por último, Alice encripta todo el mensaje con la clave pública del retransmisor 1, E_1, y anexa al inicio un encabezado en texto llano con la dirección de correo electrónico del retransmisor 1. En la figura, este mensaje es el que está a la derecha de Alice y también es el que ella transmite en realidad.

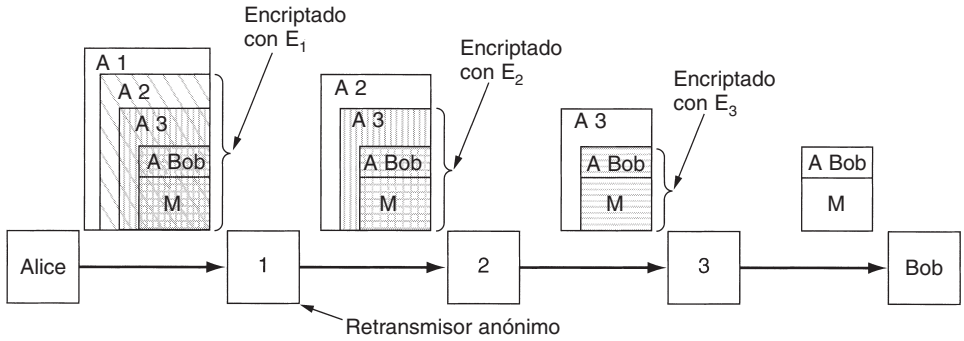


Figura 8-54. Cómo utiliza Alice tres retransmisores de correo para enviar un mensaje a Bob.

Cuando el mensaje llega al retransmisor de correo 1, el encabezado exterior se elimina. El cuerpo se desencripta y después se envía al retransmisor de correo 2. En los otros dos retransmisores se realizan pasos similares.

Aunque para cualquiera es muy difícil rastrear el mensaje final de regreso a Alice, muchos retransmisores de correo toman precauciones de seguridad adicionales. Por ejemplo, tal vez mantengan los mensajes por un tiempo aleatorio, agreguen o eliminen basura al final de un mensaje, y reordenen los mensajes, todo esto para dificultar que alguien pueda indicar cuál mensaje de un retransmisor de correo corresponde a qué entrada, a fin de frustrar el análisis de tráfico. Para una descripción de un sistema que representa lo último en correo electrónico anónimo, vea (Mazières y Kaashoek, 1998).

El anonimato no se limita al correo electrónico. También existen servicios que permiten la navegación anónima en Web. El usuario configura su navegador para utilizar el *anonymizer* como un *proxy*. De ahí en adelante, todas las solicitudes HTTP se dirigirán al anonymizer, el cual solicitará y regresará la página. El sitio Web ve al *anonymizer*, y no al usuario, como el origen de la solicitud. Siempre y cuando el *anonymizer* se abstenga de mantener una bitácora, nadie podrá determinar quién solicitó qué página.

8.10.2 Libertad de expresión

La privacidad se refiere a los individuos que desean restringir lo que otras personas ven en ellos. Un segundo problema social clave es la libertad de expresión, y su aspecto opuesto, la censura, que tiene que ver con el hecho de que los gobiernos desean restringir lo que los individuos pueden leer y publicar. Debido a que la Web contiene millones y millones de páginas, se ha vuelto un paraíso de censura. Dependiendo de la naturaleza e ideología del régimen, entre el material prohibido podrían encontrarse los sitios Web que contengan cualquiera de lo siguiente:

1. Material inapropiado para niños o adolescentes.
2. Odio dirigido a varios grupos religiosos, étnicos o sexuales, entre otros.
3. Información sobre democracia y valores democráticos.

4. Relatos de eventos históricos que contradigan la versión del gobierno.
5. Manuales para abrir candados, construir armas, encriptar mensajes, etcétera.

La respuesta común es prohibir los sitios malos.

Algunas veces los resultados son inesperados. Por ejemplo, algunas bibliotecas públicas han instalado filtros Web en sus computadoras para que sean aptas para los niños y bloqueados los sitios pornográficos. Los filtros vetan los sitios que se encuentran en sus listas negras y también verifican las páginas antes de desplegarlas para ver si contienen palabras obscenas. En Loudoun County, Virginia, sucedió que el filtro bloqueó la búsqueda que un cliente realizó para encontrar información sobre el cáncer de mama porque el filtro vio la palabra "mama". Dicho usuario de la biblioteca demandó al condado Loudoun. Sin embargo, en Livermore, California, después de que se sorprendió a un niño de 12 años de edad viendo pornografía, su padre demandó a la biblioteca por *no* instalar un filtro. ¿Qué tenía que hacer la biblioteca?

Mucha gente ha obviado el hecho de que World Wide Web es una red mundial. Cubre a todo el mundo. No todos los países están de acuerdo en lo que debe permitirse en Web. Por ejemplo, en noviembre de 2000, una corte de Francia ordenó a Yahoo, una corporación de California, que bloqueara a sus usuarios franceses para que no pudieran ver las subastas de objetos de recuerdo nazis, porque poseer tal material viola las leyes francesas. Yahoo apeló en una corte de Estados Unidos, la cual le dio la razón, pero aún está lejos de resolverse el problema de dónde aplicar las leyes de quién.

Simplemente imagínese. ¿Qué pasaría si alguna corte de Utah ordenara a Francia que bloqueara los sitios Web relacionados con el vino porque no cumplen con las muy estrictas leyes de Utah sobre el alcohol? Suponga que China demandara que todos los sitios Web que tienen que ver con la democracia fueran prohibidos porque no son del interés del Estado. ¿Las leyes iraníes sobre la religión se aplican a la Suecia más liberal? ¿Puede Arabia Saudita bloquear los sitios Web que tienen que ver con los derechos de la mujer? Todo el problema es un verdadera caja de Pandora.

Un comentario relevante de John Gilmore es: "la red interpreta la censura como una avería y encuentra una ruta alterna". Para una implementación concreta, considere el **servicio eternidad** (Anderson, 1996). Su objetivo es asegurarse de que la información publicada no pueda ser eliminada o reescrita, como era común en la Unión Soviética durante el reinado de Josef Stalin. Para utilizar el servicio eternidad, el usuario especifica cuánto tiempo se mantendrá el material, paga una cuota proporcional a su duración y tamaño, y lo carga. Después de eso, nadie puede eliminarlo o modificarlo, ni siquiera quien lo cargó.

¿Cómo se puede implementar tal servicio? El modelo más sencillo es utilizar un sistema de igual a igual en el que los documentos almacenados se coloquen en docenas de servidores participantes, cada uno de los cuales obtenga una parte de la cuota y, por lo tanto, un incentivo para unirse al sistema. Los servidores deben esparcirse a través de muchas jurisdicciones legales para obtener una máxima elasticidad. Las listas de los 10 servidores seleccionados al azar podrían almacenarse en forma segura en varios lugares, por lo que si algunos estuvieran en peligro, otros aún existirían. Una autoridad dispuesta a destruir el documento nunca estará segura de que ha encontrado todas las copias. El sistema también podría repararse a sí mismo; por ejemplo, si se sabe que se han destruido algunas copias, los sitios restantes podrían intentar encontrar nuevos depósitos para reemplazarlas.

El servicio eternidad fue la primera propuesta en lo que se refiere a sistemas anticensura. Desde entonces se han propuesto otros sistemas y, en algunos casos, se han implementado. Asimismo, se han agregado algunas nuevas características, como encriptación, anonimato y tolerancia a fallas. Con frecuencia los archivos se dividen en múltiples fragmentos, los cuales se almacenan en muchos servidores. Algunos de estos sistemas son Freenet (Clarke y cols., 2002), PASIS (Wylie y cols., 2000) y Publius (Waldman y cols., 2000). En (Serjantov, 2002), se reporta más trabajo.

En la actualidad, cada vez más países tratan de regular la exportación de valores intangibles, entre los que se encuentran sitios Web, software, documentos científicos, correo electrónico, servicios de ayuda telefónica, entre otros. Incluso en el Reino Unido, que tiene una tradición de siglos de libertad de expresión, ahora está considerando seriamente las leyes muy restrictivas, las cuales podrían, por ejemplo, definir las discusiones técnicas entre un profesor británico y su estudiante extranjero de la Universidad de Cambridge como exportación regulada que necesita una licencia del gobierno (Anderson, 2002). No es necesario decir que tales políticas son controversiales.

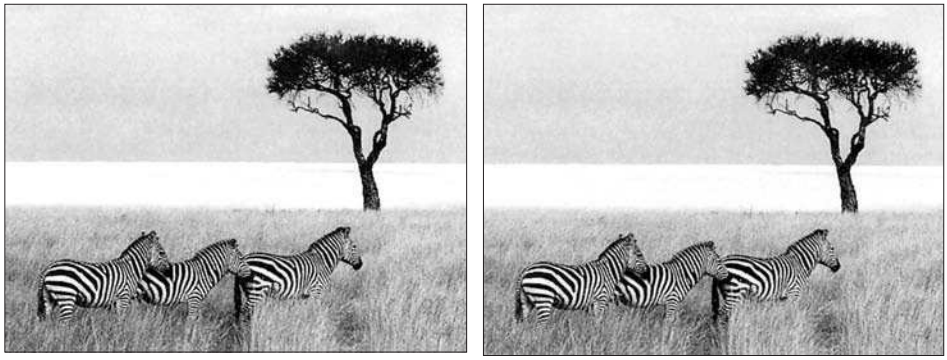
Esteganografía

En los países en donde abunda la censura, los disidentes con frecuencia tratan de utilizar la tecnología para evadirla. La criptografía permite el envío de mensajes secretos (aunque tal vez no legalmente), pero si el gobierno piensa que Alice es una Mala Persona, el simple hecho de que ella se esté comunicando con Bob podría ponerlo a él también en esta categoría, pues los gobiernos represivos entienden el concepto de clausura transitiva, aunque no entiendan bien las matemáticas. Los retransmisores de correo anónimos pueden ayudar, pero si están prohibidos domésticamente, y los mensajes dirigidos a extranjeros requieren una licencia de exportación por parte del gobierno, no serían de mucha ayuda. Pero Web sí puede.

Las personas que desean comunicarse de manera secreta con frecuencia tratan de ocultar el hecho de que se está realizando la comunicación. La ciencia de ocultar mensajes se conoce como **esteganografía,** cuyo origen proviene de las palabras griegas correspondientes a "escritura encubierta". De hecho, los antiguos griegos la utilizaron. Herodoto escribió sobre un general que rapó a un mensajero, tatuó un mensaje en el cuero cabelludo de éste y dejó que le creciera el cabello antes de enviarlo a realizar la entrega. Las técnicas modernas son conceptualmente las mismas, sólo que tienen un mayor ancho de banda y una latencia menor.

Como ejemplo, considere la figura 8-55(a). Esta fotografía, tomada en Kenia, contiene tres cebras que están contemplando un árbol de acacia. La figura 8-55(b) parece ser la misma foto de las tres cebras y el árbol de acacia, pero tiene una atracción extra. Contiene todo el texto de cinco obras de Shakespeare: *Hamlet*, *El Rey Lear*, *Macbeth*, *El Mercader de Venecia* y *Julio César*. Juntas, estas obras tienen un tamaño de 700 KB.

¿Cómo funciona este canal esteganográfico? La imagen a color original es de 1024×768 píxeles. Cada píxel consiste en tres números de 8 bits, cada uno para la intensidad de rojo, verde y azul de ese píxel. El color del píxel se forma por la superposición lineal de los tres colores. El método de codificación esteganográfica utiliza como canal secreto el bit de orden menor de cada valor de color RGB. De esta manera, cada píxel tiene espacio para 3 bits de información secreta,



(a) (b)

Figura 8-55. (a) Tres cebras y un árbol. (b) Tres cebras, un árbol y todo el texto de cinco obras de William Shakespeare.

uno en el valor de rojo, otro en el valor de verde y otro más en el de azul. En una imagen de este tamaño se pueden almacenar hasta $1024 \times 768 \times 3$ bits o 294,912 bytes de información secreta.

Todo el texto de las cinco obras y una noticia corta suman 734,891 bytes. Este texto primero se comprimió a casi 274 KB mediante un algoritmo de compresión estándar. Después la salida comprimida se encriptó utilizando IDEA y se insertó en los bits de orden menor de cada valor de color. Como puede verse (o más bien, no puede verse), la información es completamente invisible. También es invisible en la versión a todo color de la foto. El ojo no puede distinguir con facilidad los colores de 21 bits de los de 24 bits.

Al ver las dos imágenes en blanco y negro con una resolución baja no se distingue con facilidad el poder de esta técnica. Para tener una mejor idea de cómo funciona la esteganografía, el autor ha preparado una demostración, que incluye la imagen de alta resolución a todo color de la figura 8-55(b) con las cinco obras incrustadas en ella. En el sitio Web del libro se incluye la demostración y las herramientas para insertar y extraer texto en imágenes.

Para utilizar la esteganografía para la comunicación no detectada, los disidentes podrían crear un sitio Web que contenga imágenes políticamente correctas, como fotografías de un gran líder, de deportes locales, de estrellas de películas y de televisión, etcétera. Por supuesto, las imágenes contendrán mensajes esteganográficos. Si los mensajes primero se comprimieran y después se desencriptaran, incluso alguien que sospechara su presencia tendría mucha dificultad para distinguir dichos mensajes del ruido blanco. Por supuesto, las imágenes deben ser digitalizaciones recientes; copiar una imagen de Internet y cambiar algunos de los bits produce una revelación involuntaria.

Las imágenes no son el único medio para los mensajes esteganográficos. Los archivos de audio también funcionan bien. Este tipo de archivos tiene un ancho de banda esteganográfico alto. Incluso el orden de las etiquetas de un archivo HTML puede llevar información.

Aunque hemos examinado la esteganografía en el contexto de la libertad de expresión, tiene varios usos. Uno de los más comunes es para que los dueños de imágenes codifiquen mensajes secretos en ellos que indiquen sus derechos de propiedad. Si una imagen de éstas es robada y se coloca en un sitio Web, el propietario legal puede revelar el mensaje esteganográfico en la corte para probar a quién pertenece esa imagen. Esta técnica se conoce como **watermarking** (**marca de agua**) y se describe en (Piva y cols., 2002).

Para obtener mayor información sobre la esteganografía, vea (Artz, 2001; Johnson y Jajoda, 1998; Katzenbeisser y Petitcolas, 2000, y Wayner, 2002).

8.10.3 Derechos de autor

La privacidad y la censura son sólo dos áreas en las que la tecnología se encuentra con la política pública. Una tercera son los **derechos de autor.** Éstos son el otorgamiento a los creadores de la **IP** (**propiedad intelectual**)**,** incluyendo a los escritores, artistas, compositores, músicos, fotógrafos, cinematógrafos, coreógrafos, entre otros, del derecho exclusivo para explotar su IP por algún tiempo, generalmente durante la vida del autor más 50 o 75 años en el caso de la propiedad corporativa. Después de que expiran los derechos de autor de algún trabajo, pasa a ser del dominio público y cualquiera puede utilizarlo o venderlo como lo desee. Por ejemplo, el Gutenberg Project (*www.promo.net/pg*) ha colocado en Web miles de trabajos de dominio público (de Shakespeare, Twain, Dickens). En 1998, el Congreso de los Estados Unidos extendió por 20 años más los derechos de autor en ese país por solicitud de Hollywood, que afirmó que si no se otorgaba una extensión, nadie crearía nada más. En contraste, las patentes sólo duran 20 años y las personas aún siguen inventando cosas.

Los derechos de autor dieron de qué hablar cuando Napster, un servicio de intercambio de música, tenía 50 millones de miembros. Aunque Napster realmente no copiaba la música, las cortes aseveraron que el hecho de que mantuviera una base de datos de quien tenía las canciones era infracción contributoria, es decir, Napster ayudaba a otras personas a infringir la ley. Si bien nadie alega que los derechos de autor son una mala idea (aunque muchas personas alegan que el término es demasiado tiempo, lo que favorece a las grandes corporaciones y no a las públicas), la siguiente generación de compartición de música ya está provocando problemas mayores de ética.

Por ejemplo, considere una red de igual a igual en la que las personas comparten archivos legales (música del dominio público, vídeos domésticos, pistas religiosas que no son pistas comerciales, etcétera) algunos de los cuales tal vez tengan derechos de autor. Suponga que todos están en línea todo el día mediante ADSL o cable. Cada máquina tiene un índice de lo que hay en el disco duro, más una lista de otros miembros. Alguien que esté buscando un elemento específico puede elegir un miembro al azar y ver si éste tiene tal elemento. Si no lo tiene, puede verificar a todos los miembros que se encuentran en la lista de esa persona y, si no, a todos los miembros que se encuentren en las listas de dichos miembros, y así sucesivamente. Las computadoras son muy buenas para este tipo de trabajo. Cuando encuentra el elemento, el solicitante simplemente lo copia.

Si el trabajo tiene derechos de autor, es probable que el solicitante esté infringiendo la ley (aunque para las transferencias internacionales, la pregunta de la ley de quién se aplica no es clara). Pero, ¿qué sucede con el proveedor? ¿Es un crimen mantener la música por la que se ha pagado y

que se ha descargado legalmente en su disco duro donde otros pueden encontrarla? Si usted tiene una cabina abierta en el país y un ladrón de IP entra con una computadora portátil y un digitalizador, copia un libro que tiene derechos de autor y se va de manera furtiva, ¿*usted* es culpable por no poder proteger los derechos de autor de alguien más?

Pero hay más problemas relacionados con los derechos de autor. En la actualidad hay una gran batalla entre Hollywood y la industria de la computación. El primero quiere protección estricta de toda la propiedad intelectual y la segunda no desea ser el policía de Hollywood. En octubre de 1998, el Congreso aprobó la **DMCA** (**Ley de Propiedad Intelectual para el Milenio Digital**) que considera un crimen evadir cualquier mecanismo de protección presente en un trabajo con derechos de autor o indicar a otros cómo evadirlo. En la Unión Europea se está estableciendo una legislación similar. Si bien nadie piensa que debería permitirse que los piratas del Lejano Oriente duplicaran los trabajos con derechos de autor, muchas personas piensan que la DMCA desplaza por completo el balance entre los intereses de los propietarios con derechos de autor y los públicos.

Por ejemplo, en septiembre de 2000, un consorcio de la industria de la música encargado de construir un sistema irrompible para vender música en línea patrocinó un concurso en el que invitaba a las personas a que trataran de romper el sistema (que es precisamente lo que debería hacerse con cualquier sistema de seguridad). Un equipo de investigadores de seguridad de diversas universidades, dirigidas por el profesor Edward Felten de Princeton, aceptó el desafío y rompió el sistema. Después, este equipo escribió un documento sobre sus descubrimientos y lo emitió a la conferencia de seguridad USENIX, en donde se le sometió a evaluación por expertos y fue aceptado. Antes de que el documento se presentara, Felten recibió una carta de la Recording Industry Association of America, la cual amenazaba con demandarlos apoyándose en la DMCA si publicaban el documento.

Su respuesta fue llevar a juicio pidiendo a la corte federal que reglamentara sobre si el publicar documentos científicos sobre investigación de seguridad era todavía legal. Temiendo que la corte fallara en contra de ellos, la industria retiró su amenaza y la corte desechó la demanda de Felten. Sin duda la industria estaba motivada por la debilidad de su caso: habían invitado a las personas a que trataran de romper su sistema y después trataron de demandar a algunas de ellas por aceptar el desafío. Una vez retirada la amenaza, el documento se publicó (Craver y cols., 2001). Una nueva confrontación es virtualmente cierta.

Un asunto relacionado es la extensión de la **doctrina de uso razonable,** que ha sido establecida por fallos de la corte en varios países. Esta doctrina predica que los compradores de un trabajo con derechos de autor tienen ciertos derechos limitados para copiar el trabajo, incluyendo el derecho de citar partes de él para propósitos científicos, utilizarlo como material de enseñanza en escuelas o colegios y, en algunos casos, realizar copias de respaldo para uso personal en caso de que el medio original falle. Las pruebas de lo que constituye uso razonable incluyen (1) si el uso es comercial, (2) qué porcentaje se copia, y (3) el efecto del copiado en las ventas del trabajo. Puesto que el DMCA y las leyes similares dentro de la Unión Europea prohíben la evasión de los esquemas de protección contra copia, estas leyes también prohíben el uso razonable legal. En efecto, la DMCA elimina los derechos históricos de los usuarios para dar más poder a los vendedores de contenido. Es inevitable un enfrentamiento mayor.

Otro desarrollo en los trabajos que eclipsa incluso a la DMCA en su desplazamiento del equilibrio entre los propietarios de los derechos de autor y los usuarios es la **TCPA** (**Alianza para una Plataforma Informática Confiable**), dirigida por Intel y Microsoft. La idea es que el procesador de la CPU y el sistema operativo supervisen de diversas maneras y con cuidado el comportamiento del usuario (por ejemplo, ejecutando música pirateada) y prohíban el comportamiento no deseable. El sistema incluso permite que los propietarios de contenido manipulen en forma remota las PCs de los usuarios para cambiar las reglas cuando se estime necesario. No es necesario decir que las consecuencias sociales de este esquema son inmensas. Es bueno que la industria finalmente esté poniendo atención a la seguridad, pero es lamentable que esté enteramente dirigido a reforzar las leyes de derechos de autor en lugar de tratar con los virus, *crackers*, intrusos y otros problemas de seguridad por lo que la mayoría de la gente está preocupada.

En resumen, en los siguientes años los legisladores y abogados estarán ocupados equilibrando los intereses económicos de los propietarios de derechos de autor con los intereses públicos. El ciberespacio no es muy diferente de la realidad: constantemente está confrontando a un grupo con otro, lo que resulta en batallas por el poder, litigaciones y (con suerte) algún tipo de resolución, por lo menos hasta que aparezca alguna tecnología inquietante.

8.11 RESUMEN

La criptografía es una herramienta que puede utilizarse para mantener confidencial la información y para asegurar su integridad y autenticidad. Todos los sistemas criptográficos modernos se basan en el principio de Kerckhoff de tener un algoritmo conocido públicamente y una clave secreta. Muchos algoritmos criptográficos utilizan transformaciones complejas que incluyen sustituciones y permutaciones para transformar el texto llano en texto cifrado. Sin embargo, si la criptografía cuántica puede hacerse práctica, el uso de rellenos de una sola vez podría proporcionar criptosistemas realmente irrompibles.

Los algoritmos criptográficos pueden dividirse en algoritmos de clave simétrica y algoritmos de clave pública. Los primeros alteran los bits en una serie de rondas parametrizadas por la clave para convertir al texto llano en texto cifrado. En la actualidad los algoritmos de clave simétrica más populares son el triple DES y Rijndael (AES). Éstos pueden utilizarse en modo de libro de código electrónico, modo de encadenamiento de bloque de cifrado, modo de cifrado de flujo, modo de contador, entre otros.

Los algoritmos de clave pública tienen la propiedad de que se utilizan diferentes claves para la encriptación y desencriptación y de que la clave de desencriptación no puede derivarse de la clave de encriptación. Estas propiedades hacen posible publicar la clave pública. El algoritmo principal de clave pública es RSA, el cual deriva su fuerza del hecho de que es muy difícil factorizar números grandes.

Los documentos legales, comerciales y de otro tipo necesitan firmarse. De manera acorde, se han diseñado varios esquemas para las firmas digitales, las cuales utilizan tanto algoritmos de clave simétrica como de clave pública. Por lo general, a los mensajes que se van a firmar se les aplica un

hash mediante algoritmos como MD5 o SHA-1, y después se firman los *hashes* en lugar de los mensajes originales.

El manejo de claves públicas puede realizarse utilizando certificados, los cuales son documentos que enlazan a un personaje principal con una clave pública. Los certificados son firmados por una autoridad de confianza o por alguien aprobado (recursivamente) por una autoridad confiable. La raíz de la cadena se tiene que obtener por adelantado, pero los navegadores por lo general tienen integrados muchos certificados raíz.

Estas herramientas criptográficas se pueden utilizar para asegurar el tráfico de red seguro. IPsec opera en la capa de red, encriptando flujos de paquetes de *host* a *host*. Los *firewalls* pueden filtrar tráfico entrante y saliente de una organización, frecuentemente con base en el protocolo y puerto utilizado. Las redes privadas virtuales pueden simular una red de línea rentada antigua para proporcionar ciertas propiedades de seguridad deseables. Por último, las redes inalámbricas necesitan buena seguridad y WEP de 802.11 no la proporciona, aunque 802.11i debería mejorar las cosas de manera considerable.

Cuando dos partes establecen una sesión, deben autenticarse entre sí y, si es necesario, establecer una clave de sesión compartida. Existen varios protocolos de autenticación, entre ellos algunos que utilizan un tercero confiable, Diffie-Hellman, Kerberos y la criptografía de clave pública.

La seguridad de correo electrónico puede alcanzarse mediante una combinación de las técnicas que hemos estudiado en este capítulo. Por ejemplo, PGP comprime mensajes, después los encripta utilizando IDEA. Envía la clave IDEA encriptada con la clave pública del receptor. Además, también aplica un *hash* al mensaje y envía el *hash* firmado para verificar la integridad del mensaje.

La seguridad en Web también es un tema importante, iniciando con la asignación de nombres segura. DNSsec proporciona una forma de evitar la falsificación de DNS, al igual que los nombres autocertificables. La mayoría de los sitios Web de comercio electrónico utilizan SSL para establecer sesiones autenticadas seguras entre el cliente y el servidor. Se utilizan varias técnicas para tratar con el código móvil, especialmente las cajas de arena y la firma de código.

Internet provoca muchos problemas en los que la tecnología interactúa de manera considerable con la política pública. Algunas de estas áreas incluyen la privacidad, libertad de expresión y derechos reservados.

PROBLEMAS

1. Rompa el siguiente cifrado monoalfabético. El texto llano, que consiste sólo en letras, es un fragmento bien conocido de un poema de Lewis Carroll.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mftnm zhx mfudm zhx mdzythc pzq ur ezsszcdm zhx gthcm
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthcm

sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd ur om zid ouk ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Rompa el siguiente cifrado de transposición columnar. El texto llano proviene de un libro de texto de computadoras muy popular, por lo que "computadora" es una palabra probable. El texto llano consiste por completo en letras (sin espacios). Para mayor claridad, el texto cifrado está dividido en bloques de cinco caracteres.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lbolr otoex

3. Encuentre un relleno de una sola vez de 77 bits que genere el texto "Donald Duck" a partir del texto cifrado de la figura 8-4.

4. La criptografía cuántica requiere tener un arma de fotones que pueda disparar a solicitud un solo fotón que transporte 1 bit. En este problema, calcule cuántos fotones transporta un bit en un enlace de fibra de 100 Gbps. Suponga que la longitud de un fotón es igual a su longitud de onda, que por propósitos de este problema, es 1 micra. La velocidad de la luz en la fibra es de 20 cm/nseg.

5. Si Trudy captura y regenera fotones cuando está en uso la criptografía cuántica, obtendrá algunos fotones erróneos y causará errores en el relleno de una sola vez de Bob. ¿En promedio qué fracción de los bits de relleno de una sola vez de Bob serán erróneos?

6. Un principio fundamental de criptografía indica que todos los mensajes deben tener redundancia. Pero también sabemos que la redundancia ayuda a que un intruso sepa si una clave adivinada es correcta. Considere dos formas de redundancia. Primero, los *n* bits iniciales de texto llano contienen un patrón conocido. Segundo, los bits *n* finales del mensaje contienen un *hash* en el mensaje. Desde un punto de vista de seguridad, ¿estos dos son equivalentes? Analice su respuesta.

7. En la figura 8-6, se alternan las cajas P y S. Aunque este arreglo es estético, ¿es más seguro que primero tener todas las cajas P y después todas las S?

8. Diseñe un ataque a DES con base en el conocimiento de que el texto llano consiste exclusivamente en letras ASCII mayúsculas, más espacio, coma, punto, punto y coma, retorno de carro y avance de línea. No se sabe nada sobre los bits de paridad de texto llano.

9. En el texto calculamos que una máquina para romper cifrado con mil millones de procesadores que pueden analizar una clave en 1 picosegundo podría tardar sólo 10^{10} años para romper la versión de 128 bits de AES. Sin embargo, las máquinas actuales podrían tener 1024 procesadores y tardar 1 mseg en analizar una clave, por lo que necesitamos un factor de mejora en rendimiento de 10^{15} sólo para obtener una máquina de rompimiento AES. Si la ley de Moore (el poder de cómputo se duplica cada 18 meses) sigue vigente, ¿cuántos años se necesitarían tan sólo para construir la máquina?

10. AES soporta una clave de 256 bits. ¿Cuántas claves tiene AES-256? Vea si puede obtener algún número en física, química o astronomía de aproximadamente el mismo tamaño. Utilice Internet para buscar números grandes. Elabore una conclusión a partir de su investigación.

11. Suponga que un mensaje se ha encriptado utilizando DES en modo de encadenamiento de bloque de texto cifrado. Un bit de texto cifrado en el bloque C_i se transforma accidentalmente de 0 a un 1 durante la transmisión. ¿Cuánto texto llano se obtendrá como resultado?

12. Ahora considere nuevamente el encadenamiento de bloque de texto cifrado. En lugar de que un solo bit 0 sea transformado en un bit 1, un bit 0 extra se inserta en el flujo de texto cifrado después del bloque C_i. ¿Cuánto texto llano se distorsionará como resultado?

13. Compare el encadenamiento de bloque cifrado con el modo de retroalimentación de cifrado en términos del número de operaciones de encriptación necesarias para transmitir un archivo grande. ¿Cuál es más eficiente y por cuánto?

14. Utilizando el criptosistema de clave pública RSA, con $a = 1$, $b = 2$, etcétera,
(a) Si $p = 7$ y $q = 11$, liste cinco valores legales para d.
(b) Si $p = 13$, $q = 31$ y $d = 7$, encuentre e.
(c) Utilizando $p = 5$, $q = 11$ y $d = 27$, encuentre e y encripte "abcdefghij".

15. Suponga que un usuario, María, descubre que su clave privada RSA ($d1$, $n1$) es la misma que la clave pública RSA ($e2$, $n2$) de otro usuario, Frances. En otras palabras, $d1 = e2$ y $n1 = n2$. ¿María debe considerar cambiar sus claves pública y privada? Explique su respuesta.

16. Considere el uso del modo de contador, como se muestra en la figura 8-15, pero con $IV = 0$. ¿El uso de 0 amenaza la seguridad del cifrado en general?

17. El protocolo de firma de la figura 8-18 tiene la siguiente debilidad. Si Bob falla, podría perder el contenido de su RAM. ¿Qué problemas causa esto y qué puede hacer él para evitarlos?

18. En la figura 8-20 vimos la forma en que Alice puede enviar a Bob un mensaje firmado. Si Trudy reemplaza *P*, Bob puede detectarlo. Pero, ¿qué sucede si Trudy reemplaza tanto *P* como la firma?

19. Las firmas digitales tienen una debilidad potencial debido a los usuarios flojos. En las transacciones de comercio electrónico podría suscribirse un contrato y el usuario podría solicitar la firma de su *hash* SHA-1. Si el usuario no verifica que el contrato y el *hash* correspondan, firmará de manera inadvertida un contrato diferente. Suponga que la mafia trata de explotar esta debilidad para ganar algo de dinero. Los mafiosos establecen un sitio Web no gratuito (por ejemplo, de pornografía, apuestas, etcétera) y piden a los nuevos clientes un número de tarjeta de crédito. Después envían al cliente un contrato en el que estipulan que éste desea utilizar su servicio y pagar con tarjeta de crédito y le pide que lo firme, a sabiendas de que la mayoría de los clientes simplemente firmarán sin verificar que el contrato y el *hash* correspondan. Muestre la forma en que la mafia puede comprar diamantes de un joyero legítimo de Internet y pueda cargarlos a clientes inocentes.

20. Una clase de matemáticas tiene 20 estudiantes. ¿Cuál es la probabilidad de que por lo menos dos estudiantes tengan la misma fecha de nacimiento? Suponga que nadie nació en año bisiesto, por lo que hay posibles 365 fechas de nacimiento.

21. Después de que Ellen confiesa a Marilyn que la engañó en el asunto de Tom, Marilyn decide evitar este problema dictando el contenido de los mensajes futuros en una máquina de dictado y dárselos a su nueva secretaria para que los teclee. A continuación, Marilyn planea examinar los mensajes en su terminal después de que la secretaria los haya tecleado para asegurarse de que contengan sus palabras exactas. ¿La nueva secretaria aún puede utilizar el ataque de cumpleaños para falsificar un mensaje, y de ser así, cómo? *Sugerencia*: Ella puede hacerlo.

22. Considere el intento fallido de Alice para obtener la clave pública de Bob en la figura 8-23. Suponga que Bob y Alice ya comparten una clave secreta, pero Alice aún quiere la clave pública de Bob. ¿Hay ahora una forma para obtenerla de manera segura? De ser así, ¿cómo?

23. Alice se quiere comunicar con Bob, utilizando la criptografía de clave pública. Ella establece una conexión con alguien que espera sea Bob. Le pide su clave pública y él se la envía en texto llano junto con un certificado X.509 firmado por la raíz CA. Alice ya tiene la clave pública de la raíz CA. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob? Suponga que a Bob no le importa con quién está hablando (por ejemplo, Bob es algún tipo de servicio público).

24. Suponga que un sistema utiliza PKI con base en una jerarquía con estructura de árbol de CAs. Alice desea comunicarse con Bob y recibe un certificado de Bob firmado por un CA *X* después de establecer el canal de comunicación con Bob. Suponga que Alice nunca ha escuchado sobre *X*. ¿Qué pasos debe realizar Alice para verificar que está hablando con Bob?

25. ¿Es posible utilizar en modo de transporte IPsec con AH si alguna de las máquinas está detrás de una caja NAT? Explique su respuesta.

26. Dé una ventaja de HMACs con respecto del uso de RSA para firmar *hashes* SHA-1.

27. Dé una razón por la cual se configuraría un *firewall* para inspeccionar el tráfico entrante. Dé una razón por la cual se configuraría para inspeccionar tráfico saliente. ¿Cree que las inspecciones sean exitosas?

28. En la figura 8-31 se muestra el formato de paquetes WEP. Suponga que la suma de verificación es de 32 bits, calculada mediante la aplicación de OR exclusivos a todas las palabras de 32 bits en la carga útil. También suponga que los problemas con RC4 se corrigen reemplazándolo con un cifrado de flujo que no tiene debilidad y que los de IV se extienden a 128 bits. ¿Hay alguna forma para que un intruso espíe o interfiera con el tráfico sin ser detectado?

29. Suponga que una organización utiliza VPN para conectar de manera segura sus sitios a Internet. ¿Hay alguna necesidad de que un usuario, Jim, de esta organización utilice la encriptación o cualquier otro mecanismo de seguridad para comunicarse con otro usuario, Mary, de la organización?

30. Cambie ligeramente un mensaje del protocolo de la figura 8-34 para hacerlo resistente al ataque de reflexión. Explique por qué funcionaría su modificación.

31. El intercambio de claves de Diffie-Hellman se utiliza para establecer una clave secreta entre Alice y Bob. Alice envía a Bob (719, 3, 191). Bob responde con (543). El número secreto de Alice, *x*, es 16. ¿Cuál es la clave secreta?

32. Si Alice y Bob no se conocen, no comparten secretos ni tienen certificados, de cualquier manera pueden establecer una clave secreta compartida utilizando el algoritmo de Diffie-Hellman. Explique por qué es muy difícil protegerse contra un ataque de hombre en medio.

33. En el protocolo de la figura 8-39, ¿por qué *A* se envía en texto llano junto con la clave de sesión encriptada?

34. En el protocolo de la figura 8-39, señalamos que iniciar cada mensaje de texto llano con 32 bits 0 es un riesgo de seguridad. Suponga que cada mensaje comienza con un número aleatorio por usuario, efectivamente una segunda clave secreta conocida sólo por su usuario y el KDC. ¿Esto elimina el ataque de texto llano conocido? ¿Por qué?

35. En el protocolo Needham-Schroeder, Alice genera dos desafíos, R_A y R_{A2}. Esto parece excesivo. ¿Uno solo no podría haber sido suficiente?

36. Suponga que una organización utiliza Kerberos para la autenticación. En términos de seguridad y disponibilidad de servicio, ¿cuál es el efecto si AS o TGS se desactivan?

37. En el protocolo de autenticación de clave pública de la figura 8-43, en el mensaje 7, R_B se encripta con K_S. ¿Esta encriptación es necesaria, o podría haber sido adecuado regresarla en texto llano? Explique su respuesta.

38. Las terminales de punto de ventas que utilizan tarjetas con banda magnética y códigos PIN tienen una falla fatal: un comerciante malicioso puede modificar su lector de tarjetas para capturar y almacenar toda la información de la tarjeta, así como el código PIN a fin de realizar posteriormente transacciones adicionales (falsas). La siguiente generación de terminales de punto de ventas utilizará tarjetas con una CPU completa, teclado y una pequeña pantalla en la tarjeta. Diseñe un protocolo para este sistema que los comerciantes maliciosos no puedan romper.

39. Dé *dos* razones por las cuales PGP comprime mensajes.

40. Suponiendo que todos en Internet utilizaron PGP, ¿un mensaje PGP puede enviarse a una dirección arbitraria y decodificarse de manera correcta por todos los interesados? Explique su respuesta.

41. El ataque mostrado en la figura 8-47 omite un paso. Éste no es necesario para que el falsificador trabaje, pero incluyéndolo podría reducir la sospecha potencial después del hecho. ¿Cuál es el paso faltante?

42. Se ha propuesto hacer que la falsificación DNS fracase utilizando la predicción ID haciendo que el servidor coloque un ID aleatorio en lugar de utilizar un contador. Discuta los aspectos de seguridad de este método.

43. El protocolo de transporte de datos SSL involucra dos marcas aleatorias, así como una clave premaestra. ¿Cuál valor, en caso de que haya, tiene el uso de las marcas aleatorias?

44. La imagen de la figura 8-55(b) contiene el texto ASCII de cinco obras de Shakespeare. ¿Sería posible ocultar música entre las cebras en lugar de ocultar texto? De ser así, ¿cómo funcionaría y cuánto podría ocultar en esta imagen? Si no es posible, explique por qué.

45. Alice era usuario de un retransmisor de correo anónimo de tipo 1. Ella podía publicar muchos mensajes en su grupo de noticias favorito, *alt.fanclub.alice*, y todos sabían que tales mensajes provenían de Alice porque todos llevaban el mismo pseudónimo. Suponiendo que el retransmisor de correo funcionaba de manera correcta, Trudy no podía suplantar a Alice. Después de que los retransmisores de correo de tipo 1 desaparecieran, Alice cambió a un retransmisor *cypherpunk* e inició un nuevo subproceso en su grupo de noticias. Diseñe una manera para que Alice evite que Trudy la suplante y publique nuevos mensajes en el grupo de noticias.

46. Busque en Internet un caso interesante que involucre la privacidad y escriba un informe de una página sobre él.

47. Busque en Internet algún caso de una corte que involucre los derechos reservados contra el uso razonable y escriba un informe de una página en el que resuma lo que haya encontrado.

48. Escriba un programa que encripte su entrada al aplicar a ésta un OR exclusivo con un flujo de claves. Busque o escriba lo mejor que pueda un generador de números aleatorios para generar el flujo de claves. El programa debe actuar como un filtro, tomando texto llano como entrada estándar y produciendo texto cifrado como salida estándar (y viceversa). El programa debe tomar un parámetro, la clave que alimenta al generador de números aleatorios.

49. Escriba un procedimiento que calcule el *hash* SHA-1 de un bloque de datos. El procedimiento debe tener dos parámetros: un apuntador al búfer de entrada y otro a un búfer de salida de 20 bytes. Para ver la especificación exacta de SHA-1, busque en Internet FIPS 180-1, que es la especificación completa.

9

LISTA DE LECTURAS Y BIBLIOGRAFÍA

Hemos terminado nuestro estudio de redes de computadoras, pero éste es sólo el comienzo. Por falta de espacio, no tratamos muchos temas interesantes con el detalle que se merecen y omitimos otros. Para beneficio de los lectores que se interesan en continuar sus estudios de redes de computadoras, en este capítulo ofrecemos algunas sugerencias de lecturas adicionales y una bibliografía.

9.1. SUGERENCIAS DE LECTURAS ADICIONALES

Hay una extensa literatura sobre todos los aspectos de redes de computadoras. Tres revistas que con frecuencia publican artículos sobre esta área son *IEEE Transactions on Communications, IEEE Journal on Selected Areas in Communications* y *Computer Communication Review.* Otras muchas revistas también contienen artículos ocasionales sobre el tema.

El IEEE también publica tres revistas —*IEEE Internet Computing, IEEE Network Magazine* e *IEEE Communications Magazine*— que contienen encuestas, tutoriales y casos de estudios sobre conexión de redes. Las dos primeras se enfocan principalmente en la arquitectura, los estándares y el software, y la última tiende a la tecnología de comunicaciones (fibra óptica, satélites, etcétera).

Además, hay varias conferencias anuales o semestrales que provocan la creación de ensayos sobre redes y sistemas distribuidos, en particular, *SIGCOMM Annual Conference, The International Conference on Distributed Computer Systems* y *The Symposium on Operating Systems Principles.*

A continuación presentamos una lista de sugerencias de lectura adicional, relacionadas con los capítulos de este libro. La mayoría son tutoriales o encuestas sobre el tema; otras son capítulos de libros de texto.

9.1.1 Introducción y obras generales

Bi y cols., "Wireless Mobile Communications at the Start of the 21st Century".

Siglo nuevo, tecnología nueva. Suena bien. Después de algunas historias sobre los sistemas inalámbricos, aquí se cubren los temas principales, como los estándares, las aplicaciones, Internet y las tecnologías.

Comer, *The Internet Book.*

Cualquiera que desee una introducción fácil a Internet debe buscarla aquí. Comer describe la historia, el crecimiento, la tecnología, los protocolos y servicios de Internet en términos que los principiantes pueden entender, pero abarca tanto material que el libro también es de interés para lectores más técnicos.

Garber, "Will 3G Really Be the Next Big Wireless Technology?"

Se supone que la tercera generación de teléfonos móviles combinará voz y datos y proporcionará tasas de datos de hasta 2 Mbps. Su lanzamiento ha sido un poco lento. En este artículo fácil de leer se cubren las promesas, los problemas, la tecnología, las políticas y los aspectos económicos del uso de la comunicación inalámbrica de banda ancha.

IEEE Internet Computing, enero-febrero de 2000.

La primera edición de *IEEE Internet Computing* en el nuevo milenio hizo exactamente lo que usted esperaría: preguntar a la gente que ayudó a crear Internet en el milenio anterior hacia dónde piensa que irá en el actual. Los expertos son Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy, entre otros. Para mejores resultados, espere 500 años, y *entonces* verifique si sus predicciones fueron acertadas.

Kipnis, "Beating the System: Abuses of the Standars Adoption Process".

Los comités de estándares tratan de ser justos y neutrales en cuanto a proveedores pero, por desgracia, hay compañías que tratan de abusar del sistema. Por ejemplo, con frecuencia sucede que una compañía ayuda al desarrollo de un estándar y, una vez que éste se aprueba, anuncia que dicho estándar se basa en una patente que es de su propiedad, y que sólo dará licencias a compañías

de su agrado, a precios que sólo ella determina. Si desea echar un vistazo al lado oscuro de la estandarización, este artículo es un excelente inicio.

Kwok, "A Vision for Residential Broadband Service".

Si desea saber qué pensaba Microsoft en 1995 acerca de la entrega de vídeo a solicitud, este artículo es para usted. Cinco años más tarde la visión era irremediablemente obsoleta. El valor del artículo está en demostrar que incluso personas muy enteradas y bien intencionadas no pueden ver siquiera cinco años a futuro con certeza alguna. Ésta debe ser una lección para todos.

Kyas y Crawford, *ATM Networks.*

Alguna vez ATM fue aclamado como el protocolo de conexión de redes del futuro, y sigue siendo importante en el sistema de la telefonía. Este libro es una guía actualizada acerca del estado actual de ATM, con información detallada sobre los protocolos ATM y cómo se pueden integrar con las redes basadas en IP.

Naughton, *A Brief History of the Future.*

Sin embargo, ¿quién inventó Internet? Muchas personas han reclamado el crédito. Y con todo derecho, puesto que muchas personas tienen que ver en ello de diversas maneras. Esta historia de Internet narra cómo sucedió todo, de una manera amena y encantadora, rebosante de anécdotas, como la de AT&T que dejaba repetidamente bien sentada su creencia de que las comunicaciones digitales no tenían futuro.

Perkins, "Mobile Networking in the Internet".

Si desea un panorama de redes móviles capa por capa, éste es el lugar donde buscar. Se tratan las capas desde la de transporte hasta la física, así como middleware, seguridad y conexión de redes *ad hoc*.

Tegger y Waks, "End-User Perspectives on Home Networking".

Las redes domésticas no son como las corporativas. Las aplicaciones son diferentes (más intensivas en multimedia), el equipo proviene de un amplio rango de proveedores y los usuarios reciben un breve entrenamiento técnico y no toleran fallas. Para enterarse de más, busque aquí.

Varshney y Vetter, "Emerging Mobile and Wireless Networks".

Otra introducción a la comunicación inalámbrica. Comprende las LANs inalámbricas, los ciclos locales inalámbricos y los satélites, así como algo sobre el software y las aplicaciones.

Wetteroth, *OSI Reference Model for Telecommunications.*

Aunque los protocolos OSI en sí ya no se utilizan, el modelo de siete capas ha llegado a ser muy bien conocido. Además de dar una explicación sobre OSI, este libro aplica el modelo a las redes de telecomunicación (a diferencia de las de computadora), y muestra dónde encajan la telefonía común y otros protocolos de voz en la pila de conexión de redes.

9.1.2 La capa física

Abramson, "Internet Access Using VSATs".

Las estaciones pequeñas en tierra se están volviendo más populares tanto para la telefonía rural como para el acceso corporativo a Internet en países desarrollados. Sin embargo, la naturaleza del tráfico en estos dos casos difiere de manera dramática, por lo que se requieren protocolos diferentes para manejar los dos casos. En este artículo, el inventor del sistema ALOHA expone varios métodos de asignación de canales que se pueden utilizar para sistemas VSAT.

Alkhatib y cols., "Wireless Data Networks: Reaching the Extra Mile".

Si desea una introducción rápida a los términos y tecnologías de la conexión de redes inalámbricas, incluyendo el espectro disperso, este documento tutorial es un buen punto de partida.

Azzam y Ransom, *Broadband Access Technologies*.

Aquí el sistema telefónico, las fibras, ADSL, las redes de cable, los satélites, incluso las líneas de energía, se cubren como tecnologías de acceso de red. Entre otros temas se encuentran las redes domésticas, los servicios, el desempeño de redes y los estándares. El libro concluye con la biografía de las principales compañías en el negocio de redes y telecomunicación, pero con la velocidad de cambios que hay en la industria, este capítulo puede tener una vida de estantería más corta que los capítulos de tecnología.

Bellamy, *Digital Telephony.*

En este libro autorizado se encuentra todo lo que siempre quiso saber sobre el sistema telefónico, entre muchas cosas más. Los capítulos sobre transmisión y multiplexión, conmutación digital, fibras ópticas, telefonía móvil y ADSL son particularmente interesantes.

Berezdivin y cols., "Next-Generation Wireless Communications Concepts and Technologies".

Estas personas van un paso adelante de todas las demás. La parte "next generation (siguiente generación)" se refiere a la cuarta generación de redes inalámbricas. Se espera que estas redes proporcionen servicio IP en todas partes con conectividad a Internet transparente, con alto ancho de banda y una excelente calidad de servicio. Estos objetivos se deben alcanzar mediante el uso inteligente del espectro, administración dinámica de recursos y un servicio adaptable. Todo esto ahora suena visionario, pero aproximadamente en 1995 los teléfonos celulares sonaban bastante visionarios.

Dutta Roy, "An Overview of Cable Modem Technology and Market Perspectives".

La televisión por cable ha dejado de ser un sistema sencillo para convertirse en un sistema complejo de distribución de televisión, Internet y telefonía. Estos cambios han afectado en forma considerable la infraestructura de cable. Vale la pena leer este artículo para tener una explicación sobre las plantas, estándares y marketing de cable, con un énfasis en DOCSIS.

Farserotu y Prasad, "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends".

Varios satélites de comunicaciones están en el cielo o en la mesa de dibujo, como Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic e iSky. Éstos utilizan varias técnicas, como tubo doblado y conmutación satelital. Si desea un panorama de los diferentes sistemas y técnicas de satélite, este documento es un buen punto de partida.

Hu y Li, "Satellite-Based Internet: A Tutorial".

El acceso a Internet vía satélite es distinto del acceso por medio de líneas terrestres. No sólo está el problema del retardo, sino que además el enrutamiento y la conmutación son diferentes. En este documento los autores examinan algunos de los problemas relacionados con el uso de satélites para el acceso a Internet.

Joel, "Telecommunications and the IEEE Communications Society".

Para una historia condensada, pero sorprendentemente amplia, de las telecomunicaciones, empezando con el telégrafo y terminando con el estándar 802.11, este artículo es lo que hay que ver. También cubre radio, teléfonos, conmutación análoga y digital, cables submarinos, transmisión digital, ATM, difusión de televisión, satélites, televisión por cable, comunicaciones ópticas, teléfonos móviles, conmutación de paquetes, ARPANET e Internet.

Metcalfe, "Computer/Network Interface Design: Lessons from Arpanet & Ethernet".

Aunque los ingenieros han estado construyendo interfaces de red durante décadas, con frecuencia nos preguntamos si han aprendido algo de toda esta experiencia. En este artículo, el diseñador de Ethernet describe la manera de construir una interfaz de red, y qué se debe hacer una vez construida. No hace concesiones, indicando lo que hizo mal y lo que hizo bien.

Palais, *Fiber Optic Communication,* 3a. ed.

Los libros sobre la tecnología de la fibra óptica tienden a estar orientados al especialista, pero éste es más accesible que la mayoría; cubre las guías de onda, fuentes de luz, detectores de luz, acopladores, modulación, ruido y muchos otros temas.

Pandya, "Emerging Mobile and Personal Communications Systems".

Para obtener una introducción corta y agradable a los sistemas de comunicación personales portátiles, vea este artículo. Una de las nueve páginas contiene una lista de 70 siglas que se usan en las otras ocho páginas.

Sarikaya, "Packet Mode in Wireless Networks: Overview of Transition to Third Generation".

Toda la idea de las redes celulares de la tercera generación está en los datos de transmisión inalámbrica. Si desea ver un panorama de cómo manejan los datos las redes de la segunda generación y cuál será la evolución a la tercera generación, éste es un buen lugar. Entre los temas que se cubren se encuentran GPRS, IS-95B, WCDMA y CDMA2000.

9.1.3 La capa de enlace de datos

Carlson, *PPP Design, Implementation and Debugging,* 2a. ed.

Si a usted le interesa una información detallada sobre todos los protocolos que integran la suite PPP, incluyendo CCP (compresión) y ECP (encriptación), este libro es una buena referencia. Hay un enfoque particular en ANU PPP-2.3, una implementación popular de PPP.

Gravano, *Introduction to Error Control Codes*.

Los errores se infiltran en casi todas las comunicaciones digitales, y se han desarrollado muchos tipos de código para detectarlos y corregirlos. Este libro explica algunos de los más importantes, desde los sencillos códigos lineales de Hamming hasta los campos y los códigos de Galois más complejos. Aun cuando se intenta utilizar el mínimo de álgebra, contiene una gran cantidad de ésta.

Holzmann, *Design and Validation of Computer Protocols.*

Los lectores interesados en los aspectos más formales de los protocolos de enlace de datos (y similares) deben leer este libro. En él se cubren la especificación, el modelado, la exactitud y las pruebas de tales protocolos.

Peterson y Davie, *Computer Networks: A Systems Approach.*

El capítulo 2 contiene material acerca de muchos problemas de enlace de datos, como tramado, detección de errores, protocolos de parada y espera, protocolos de deslizamiento de ventana y las LAN IEEE 802.

Stallings, *Data and Computer Communications.*

El capítulo 7 trata la capa de enlace de datos y cubre el control de flujo, la detección de errores y los protocolos básicos de enlace de datos, como los protocolos de paro y espera y de regreso n. También se cubren los protocolos de tipo HDLC.

9.1.4 La subcapa de control de acceso al medio

Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps".

Si desea una introducción directa al sistema Bluetooth, éste es un buen lugar de inicio. Se explican los protocolos y perfiles medulares, radio, *piconets* y enlaces, seguidos de una introducción a los diversos protocolos.

Bisdikian, "An Overview of the Bluetooth Wireless Technology".

Al igual que el documento de Bhagwat (el anterior), éste también es un buen punto de partida para aprender más acerca del sistema Bluetooth. Explica las *piconets*, la pila de protocolos y los perfiles, entre otros temas.

Crow y cols., "IEEE 802.11 Wireless Local Area Networks".

Éste es un buen lugar para empezar una introducción sencilla a la tecnología y a los protocolos del estándar 802.11. Se destaca la subcapa MAC y se cubren tanto el control distribuido como

el control centralizado. El documento concluye con algunos estudios de simulación en diversas circunstancias del desempeño del estándar 802.11.

Eklund y cols., "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access".

El ciclo local inalámbrico estandarizado por el IEEE en 2002, como 802.16, puede revolucionar el servicio telefónico, ofreciendo banda ancha para el hogar. En este panorama, los autores explican los principales problemas de tecnología relacionados con este estándar.

Kapp, "802.11: Leaving the Wire Behind".

Esta breve introducción al estándar 802.11 cubre los fundamentos, los protocolos y los estándares relevantes.

Kleinrock, "On Some Principles of Nomadic Computing and Multi-Access Communications".

El acceso inalámbrico sobre un canal compartido es más complejo que tener estaciones cableadas que comparten un canal. Entre otros problemas se encuentran las topologías dinámicas, el enrutamiento y la administración de energía. En este artículo se cubren estos y otros problemas relacionados con el acceso a canales a través de dispositivos móviles inalámbricos.

Miller y Cummins, *LAN Technologies Explained.*

¿Necesita conocer más acerca de las tecnologías que se pueden emplear en una LAN? Este libro cubre la mayoría de ellas, incluyendo FDDI y token ring, así como la siempre popular Ethernet. Aunque en la actualidad son muy raras las nuevas instalaciones de las dos primeras, muchas redes existentes aún las usan, además de que todavía son comunes las redes de anillo (por ejemplo, SONET).

Perlman, *Interconnections,* 2a. ed.

El libro de Perlman es un buen lugar para buscar un tratamiento entretenido y bien documentado sobre los puentes, enrutadores y enrutamiento en general. El autor diseñó los algoritmos que se utilizan en el puente de árbol de expansión IEEE 802, y evidentemente es una de las autoridades mundiales en varios aspectos de la conectividad.

Webb, "Broadband Fixed Wireless Access".

En este documento se analizan el "porqué" y el "cómo" del ancho de banda fijo inalámbrico. La sección del "porqué" argumenta que las personas no quieren una dirección de correo electrónico doméstica, una dirección de correo electrónica de trabajo, números telefónicos separados para el hogar, trabajo y celular, una cuenta de mensajes instantánea y quizás uno o dos números de fax. Más bien desean un sistema integrado único que funcione en todas partes. En la sección de tecnología se destaca la capa física, incluyendo temas como TDD en comparación con FDD, modulación adaptable en comparación con fija y número de portadoras.

9.1.5 La capa de red

Bhatti y Crowcroft, "QoS Sensitive Flows: Issues in IP Packet Handling".

Una de las maneras de conseguir la mejor calidad de servicio de una red es programar con cuidado las salidas de paquetes de cada enrutador. En este documento se explican en detalle varios algoritmos de programación de paquetes, así como problemas relacionados.

Chakrabarti, "QoS Issues in Ad Hoc Wireless Networks".

El enrutamiento de redes *ad hoc* de computadoras portátiles que están cerca unas de otras es bastante difícil sin tener que preocuparse por la calidad del servicio. No obstante, la gente se preocupa por la calidad del servicio, por lo que hay que prestar atención a este tema. En este artículo se exponen la naturaleza de las redes *ad hoc* y algunos de los problemas relacionados con el enrutamiento y la calidad del servicio.

Comer, *Internetworking with TCP/IP,* Vol. 1, 4a. ed.

Comer ha escrito el libro definitivo sobre el conjunto de protocolos TCP/IP. Los capítulos 4 a 11 tratan el IP y los protocolos relacionados de la capa de red. Los otros capítulos se encargan principalmente de las capas superiores, y también vale la pena leerlos.

Huitema, *Routing in the Internet.*

Si quiere saber todo lo que hay que saber sobre el enrutamiento en Internet, éste es el libro para usted. Se tratan con lujo de detalle tanto los algoritmos pronunciables (por ejemplo, RIP, CIDR y MBONE), como los impronunciables (por ejemplo, OSPF, IGRP, EGP y BGP). También están aquí las características nuevas, como multidifusión, IP móvil y reservación de recursos.

Malhotra, *IP Routing.*

Si desea una guía detallada al enrutamiento IP, este libro contiene mucho material. Entre los protocolos cubiertos están RIP, RIP-2, IGRP, EIGRP, OSPF y BGP-4.

Metz, "Differentiated Services".

Las garantías de calidad de servicio son importantes para muchas aplicaciones multimedia. Los servicios integrados y los diferenciados son dos métodos posibles para alcanzarlas. Los dos se explican aquí, pero se resaltan los servicios diferenciados.

Metz, "IP Routers: New Tool for Gigabit Networking".

La mayoría de las referencias al capítulo 5 son acerca de algoritmos de enrutamiento. Ésta es diferente: se refiere a cómo funcionan en la actualidad los enrutadores. Han pasado por un proceso evolutivo desde ser estaciones de trabajo de propósito general hasta ser máquinas de enrutamiento de propósitos altamente especiales. Si desea saber más, este artículo es un buen lugar para empezar.

Nemeth y cols., *UNIX System Administration Handbook.*

Para un cambio de ritmo, el capítulo 13 de este libro aborda la conexión de redes de una manera más práctica que la mayoría de nuestras demás referencias. En lugar de tratar sólo los conceptos abstractos, aquí se proporcionan muchos consejos sobre qué hacer si usted administra una red real actualmente.

Perkins, "Mobile Networking through Mobile IP".

Conforme los dispositivos de computación portátil se vuelven cada vez más comunes, Mobile IP crece en importancia. Este tutorial da una buena introducción a este y otros temas relacionados.

Perlman, *Interconnections: Bridges and Routers,* 2a. ed.

En los capítulos 12 a 15, Perlman describe muchos de los asuntos implicados en el diseño de algoritmos de enrutamiento unidifusión y multidifusión, tanto en las WANs como en las redes de varias LANs, así como su implementación en varios protocolos. Pero la mejor parte del libro es el capítulo 18, en el que el autor destila sus años de experiencia en protocolos de red en un capítulo divertido e informativo.

Puzmanova, *Routing and Switching: Time of Convergence?*

A finales de 1990, algunos proveedores de equipos de conexión de redes empezaron a llamar conmutador a todo, mientras que muchos administradores de redes grandes decían que estaban cambiando de enrutadores a conmutadores. Como implica el título, este libro predice el futuro de los enrutadores y conmutadores, y cuestiona si en realidad están convergiendo.

Royer y Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks".

El algoritmo de enrutamiento *ad hoc* AODV que analizamos en el capítulo 5 no es el único conocido. Hay otros más, como DSDV, CGSR, WRP, DSR, TORA, ABR, DRP y SRP, que se explican aquí y se comparan entre sí. Desde luego, si planea inventar un nuevo protocolo de enrutamiento, el paso 1 es idear una sigla de tres o cuatro letras.

Stevens, *TCP/IP Illustrated,* Vol. 1.

Los capítulos 3 a 10 proporcionan un estudio detallado de IP y de los protocolos relacionados (ARP, RARP e ICMP), ilustrado con ejemplos.

Striegel y Manimaram, "A Survey of QoS Multicasting Issues".

La multidifusión y la calidad de servicio son dos temas que cada vez toman más importancia conforme empiezan a despegar servicios como radio y televisión por Internet. En esta encuesta documentada, los autores explican cómo los algoritmos de enrutamiento pueden tener en cuenta estos dos problemas.

Yang y Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks".

Los autores han diseñado una clasificación para los algoritmos de control de congestionamiento. Las categorías principales son ciclo abierto con control de origen, ciclo abierto con control de destino, ciclo cerrado con retroalimentación explícita y ciclo cerrado con realimentación implícita. Los autores usan esta clasificación para describir y clasificar 23 algoritmos existentes.

9.1.6 La capa de transporte

Comer, *Internetworking with TCP/IP,* Vol. 1, 4a. ed.

Como dijimos antes, Comer ha escrito la obra definitiva sobre el conjunto de protocolos TCP/IP. El capítulo 12 se refiere al UDP; el capítulo 13 a TCP.

Hall y Cerf, *Internet Core Protocols: The Definitive Guide.*

Si desea obtener su información directamente del origen, éste es el lugar para aprender acerca de TCP. Después de todo, Cerf lo coinventó. El capítulo 7 es una buena referencia de TCP, pues muestra cómo interpretar la información proporcionada por el análisis de protocolos y las herramientas de administración de redes. Otros capítulos tratan UDP, IGMP, ICMP y ARP.

Kurose y Ross, *Computer Networking: A Top-Down Approach Featuring the Internet.*

El capítulo 3 se refiere a la capa de transporte y contiene una buena cantidad de material sobre UDP y TCP. También presenta los protocolos de parada y espera y de regreso n que analizamos en el capítulo 3.

Mogul, "IP Network Performance".

A pesar del título de este artículo, en su mayoría trata el desempeño de TCP y las redes en general, más que el de IP en particular; está lleno de lineamientos y reglas empíricas útiles.

Peterson y Davie, *Computer Networks: A Systems Approach.*

El capítulo 5 trata sobre UDP, TCP y algunos protocolos relacionados. Aunque brevemente, también se toca el comportamiento de redes.

Stevens, *TCP/IP Illustrated,* Vol. 1.

Los capítulos 17-24 dan un tratamiento detallado de TCP, ilustrado con ejemplos.

9.1.7 La capa de aplicación

Bergholz, "Extending Your Markup: An XML Tutorial".

Una introducción directa y breve a XML para principiantes.

Berners-Lee y cols., "The World Wide Web".

Una perspectiva de Web y hacia dónde va, de parte de su inventor y de algunos de los colegas de éste en CERN. El artículo se enfoca en la arquitectura de Web, los URLs, HTPP y HTML, así como en las direcciones futuras, y la compara con los otros sistemas de información distribuidos.

Cardellini y cols., *"The State-of-the-Art in Locally Distributed Web-Server Systems"*.

A medida que Web se hace más popular, algunos sitios Web necesitan tener grandes granjas de servidores para manejar el tráfico. La parte ardua de construir una granja de servidores es distribuir la carga entre las máquinas. Este documento tutorial explica ampliamente ese tema.

Choudbury y cols., "Copyright Protection for Electronic Publishing on Computer Networks".

Aunque numerosos libros y artículos describen los algoritmos criptográficos, pocos describen la manera en que podrían usarse para evitar que los usuarios distribuyan más documentos de los que tienen permitido desencriptar. Este artículo describe varios mecanismos que pueden ayudar a proteger los derechos de los autores en la era electrónica.

Collins, "Carrier Grade Voice over IP".

Si usted ha leído el trabajo de Varshney y cols., y ahora quiere conocer todos los detalles acerca de la voz a través de IP utilizando H.323, éste es un buen lugar para ver. Aunque el libro es largo y detallado, es tutorial por naturaleza y no requiere conocimientos previos de ingeniería telefónica.

Davison, "A Web Caching Primer".

Conforme Web crece, el uso de caché se hace indispensable para un buen desempeño. Si desea una breve introducción al uso de caché de Web, éste es un buen lugar para ver.

Krishnamurthy y Rexford, *Web Protocols and Practice*.

Sería difícil encontrar un libro más extenso sobre todos los aspectos de Web que éste. Abarca clientes, servidores, *proxies* y uso de caché, como usted podía esperar. Sin embargo, también hay capítulos sobre tráfico y medidas de Web, así como otros sobre investigaciones y mejoras actuales a Web.

Rabinovich y Spatscheck, *Web Caching and Replication.*

Éste contiene un tratamiento extenso del uso de caché y la duplicación de Web. Se cubren en gran detalle *proxies*, cachés, extracción previa, redes de entrega de contenido, selección de servidor y mucho más.

Shahabi y cols., "Yima: A Second-Generation Continuous Media Server".

Los servidores multimedia son sistemas complejos que tienen que administrar programación de CPU, colocación de archivos de disco, sincronización de flujos y más. Con el tiempo, la gente

ha aprendido a diseñarlos mejor. En este documento se presenta un panorama de la arquitectura de uno de los sistemas más recientes.

Tittel y cols., *Mastering XHTML.*

Dos libros en un volumen grande, que cubren el nuevo lenguaje de marca estándar de Web. Primero, hay texto que describe HTML, enfocándose sobre todo en la manera en que se diferencia del HTML regular. Después viene una amplia guía de referencias a etiquetas, códigos y caracteres especiales que se utilizan en XHTML 1.0.

Varshney y cols., "Voice over IP".

¿Cómo funciona la voz a través de IP? ¿Va a reemplazar a la red telefónica conmutada pública? Lea y descúbralo.

9.1.8 Seguridad en redes

Anderson, "Why Cryptosystems Fail".

Según Anderson, la seguridad de los sistemas bancarios es mala, pero no debido a que intrusos astutos violen el DES desde sus PCs. Los verdaderos problemas van desde empleados deshonestos (un empleado bancario que cambia la dirección de correo de un cliente a la suya para interceptar el número de tarjeta bancaria y el número de PIN), a errores de programación (dar a todos los clientes el mismo código PIN). Lo que es especialmente interesante es la altanera respuesta que dan los bancos cuando se les confronta ante un problema evidente: "Nuestros sistemas son perfectos y, por lo tanto, todos los errores deben ser causados por errores del cliente o por fraude".

Anderson, *Security Engineering.*

Algo extenso, este libro es una versión de 600 páginas de "Why Cryptosystems Fail". Es más técnico que *Secrets and Lies,* pero menos que *Network Security* (vea más adelante). Después de una introducción a las técnicas básicas de seguridad, capítulos enteros están dedicados a diversas aplicaciones, como seguridad bancaria, comandos y control nucleares, seguridad en impresión, biometría, seguridad física, guerra electrónica, seguridad en telecomunicaciones, comercio electrónico y protección a derechos de autor. La tercera parte del libro se refiere a políticas, administración y evaluación de sistemas.

Artz, "Digital Steganography".

La esteganografía se remonta a la antigua Grecia, donde se fundía cera en tablas limpias de manera que los mensajes secretos se pudieran aplicar a la madera que estaba debajo antes de aplicar la cera nuevamente. En la actualidad se usan técnicas diferentes, pero el objetivo es el mismo. Aquí se explican varias técnicas modernas para ocultar información en imágenes, audio y otros medios de transporte.

Brands, *Rethinking Public Key Infrastructures and Digital Certificates.*

Más que una introducción amplia a los certificados digitales, éste también es un poderoso trabajo de defensa. El autor cree que los sistemas basados en documentos de verificación de identidad

son obsoletos e inútiles, y argumenta que los certificados digitales se pueden usar para aplicaciones como votaciones electrónicas, administración de derechos digitales e incluso como sustitutos de dinero en efectivo. También advierte que sin PKI ni encriptación, Internet podría llegar a ser una herramienta de supervisión a gran escala.

Kaufman y cols. *Network Security,* 2a. ed.

Para obtener más información técnica sobre algoritmos y protocolos de seguridad en redes, este libro autorizado e ingenioso es lo primero que hay que ver. Algoritmos y protocolos de clave pública y secreta, hashes de mensajes, autenticación, Kerberos, PKI, IPsec, SSL/TLS y seguridad en el correo electrónico, se explican amplia y cuidadosamente, con muchos ejemplos. El capítulo 26 sobre el folklore de la seguridad es una verdadera joya. En seguridad, el demonio está en los detalles. Cualquiera que planee diseñar un sistema de seguridad que realmente se usará, aprenderá en este capítulo mucho del consejo del mundo real.

Pohlmann, *Firewall Systems.*

Los firewalls son la primera línea (y la última) de defensa contra salteadores de redes. Este libro explica cómo funcionan y qué hacen, desde el *firewall* más sencillo con base en software diseñado para proteger una PC, hasta las avanzadas aplicaciones de *firewall* que se sitúan entre una red privada y su conexión a Internet.

Schneier, *Applied Cryptography,* 2a. ed.

Este compendio monumental es la peor pesadilla de NSA: un libro único que describe cada algoritmo criptográfico conocido. Para empeorar las cosas (o para mejorarlas, dependiendo de su punto de vista), el libro contiene la mayoría de los algoritmos como programas ejecutables (en C). Y todavía mas, se proporcionan más de 600 referencias de literatura criptográfica. Este libro no es para principiantes, pero si quiere guardar *realmente* en secreto sus archivos, léalo.

Schneier, *Secrets and Lies.*

Si leyó *Applied Cryptography* de pasta a pasta, lo sabrá todo acerca de algoritmos criptográficos. Si luego lee *Secrets and Lies* de pasta a pasta (lo que se puede hacer en mucho menos tiempo), aprenderá que los algoritmos criptográficos no son toda la historia. La mayoría de las debilidades de seguridad no se debe a fallas en los algoritmos o incluso a claves demasiado cortas, sino a fallas en el entorno de la seguridad. Se presentan innumerables ejemplos sobre amenazas, ataques, defensas, contraataques y mucho más. Para una explicación sencilla y fascinante sobre la seguridad de las computadoras en el más amplio sentido, éste es el libro que hay que leer.

Skoudis, *Counter Hack.*

La mejor manera de detener a un *hacker* es pensar como *hacker*. Este libro muestra cómo ven los *hackers* a una red, y argumenta que la seguridad debe ser una función de todo el diseño de la red, no un pensamiento posterior con base en una tecnología específica. Cubre casi todos los ataques comunes, incluyendo los tipos de "ingeniería social" que se aprovechan de los usuarios que no están familiarizados con las medidas de seguridad de computadora.

9.2 BIBLIOGRAFÍA EN ORDEN ALFABÉTICO

ABRAMSON, N., "Development of the ALOHANET", *IEEE Trans. on Information Theory,* vol. IT-31, págs. 119-123, marzo de 1985.

ABRAMSON, N., "Internet Access Using VSATs", *IEEE Commun. Magazine,* vol. 38, págs. 60-68, julio de 2000.

ADAMS, M., y DULCHINOS, D., "OpenCable", *IEEE Commun. Magazine,* vol. 39, págs. 98-105, junio de 2001.

ALKHATIB, H.S.; BAILEY, C.; GERLA, M., y McRAE, J., "Wireless Data Networks: Reaching the Extra Mile", *Computer,* vol. 30, págs. 59-62, diciembre de 1997.

ANDERSON, R.J., "Free Speech Online and Office", *Computer,* vol. 25, págs. 28-30, junio de 2002.

ANDERSON, R.J., *Security Engineering*, Nueva York: Wiley, 2001.

ANDERSON, R.J., "The Eternity Service", *Proc. First Int'l Conf. on Theory and Appl. Of Cryptology,* CTU Publishing House, 1996.

ANDERSON, R.J., "Why Cryptosystems Fail", *Commun. of the ACM,* vol. 37, págs. 32-40, noviembre de 1994.

ARTZ, D., "Digital Steganography", *IEEE Internet Computing,* vol. 5, págs. 75-80, 2001.

AZZAM, A.A., y RANSOM, N., *Broadband Access Technologies,* Nueva York: McGraw-Hill, 1999.

BAKNE, A., y BADRINATH, B.R., "I-TCP: Indirect TCP for Mobile Hosts", *Proc. 15th Int'l Conf. on Distr. Computer Systems,* IEEE, págs. 136-143, 1995.

BALAKRISHNAN, H.; SESHAN, S., y KATZ, R.H., "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", *Proc. ACM Mobile Computing and Networking Conf.,* ACM, págs. 2-11, 1995.

BALLARDIE, T.; FRANCIS, P., y CROWCROFT, J., "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.,* ACM, págs. 85-95, 1993.

BARAKAT, C.; ALTMAN, E., y DABBOUS, W., "On TCP Performance in a Heterogeneous Network: A Survey", *IEEE Commun. Magazine,* vol. 38, págs. 40-46, enero de 2000.

BELLAMY, J., *Digital Telephony*, 3a. ed., Nueva York: Wiley, 2000.

BELLMAN, R.E., *Dynamic Programming,* Princeton, Nueva Jersey: Princeton University Press, 1957.

BELSNES, D., "Flow Control in the Packet Switching Networks", *Communications Networks,* Uxbridge, Inglaterra: Online, págs. 349-361, 1975.

BENNET, C.H., y BRASSARD, G., "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing,* págs. 175-179, 1984.

BEREZDIVIN, R.; BREINIG, R., y TOPP, R., "Next-Generation Wireless Communication Concepts and Technologies", *IEEE Commun. Magazine,* vol. 40, págs. 108-116, marzo de 2002.

BERGHEL, H.L., "Cyber Privacy in the New Millennium", *Computer,* vol. 34, págs. 132-134, enero de 2001.

BERGHOLZ, A., "Extending Your Markup: An XML Tutorial", *IEEE Internet Computing,* vol. 4, págs. 74-79, julio-agosto de 2000.

BERNERS-LEE, T.; CAILLIAU, A.; LOUTONEN, A.; NIELSEN, H.F., y SECRET, A., "The World Wide Web", *Commun. of the ACM,* vol. 37, págs. 76-82, agosto de 1994.

BERTSEKAS, D., y GALLAGER, R., *Data Networks,* 2a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 1992.

BHAGWAT, P., "Bluetooth: Technology for Short-Range Wireless Apps", *IEEE Internet Computing,* vol. 5, págs. 96-103, mayo-junio de 2001.

BHARGHAVAN, V.; DEMERS, A.; SHENKER, S., y ZHANG, L., "MACAW: A Media Access Protocol for Wireless LANs", *Proc. SIGCOMM '94 Conf.,* ACM, págs. 212-225, 1994.

BHATTI, S.N., y CROWCROFT, J., "QoS Sensitive Flows: Issues in IP Packet Handling", *IEEE Internet Computing,* vol. 4, págs. 48-57, julio-agosto de 2000.

BI, Q.; ZYSMAN, G.I., y MENKES, H., "Wireless Mobile Communications at the Start of the 21st Century", *IEEE Commun. Magazine,* vol. 39, págs. 110-116, enero de 2001.

BIHAM, E., y SHAMIR, A., "Differential Cryptanalysis of the Data Encryption Standard", *Proc. 17th Ann. Int'l Cryptology Conf.,* Berlín: Springer-Verlag LNCS 1294, págs. 513-525, 1997.

BIRD, R.; GOPAL, I.; HERZBERG, A.; JANSON, P.A.; KUTTEN, S.: MOLVA, R., y YUNG, M., "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas in Commun.,* vol. 11, págs. 679-693, junio de 1993.

BIRRELL, A.D., y NELSON, B.J., "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems,* vol. 2, págs. 39-59, febrero de 1984.

BIRYUKOV, A.; SHAMIR, A., y WAGNER, D., "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption,* Berlín: Springer-Verlag LNCS 1978, p. 1, 2000.

BISDIKIAN, C., "An Overview of the Bluetooth Wireless Technology", *IEEE Commun. Magazine,* vol. 39, págs. 86-94, diciembre de 2001.

BLAZE, M., "Protocol Failure in the Escrowed Encryption Standard", *Proc. Second ACM Conf. on Computer and Commun. Security,* ACM, págs. 59-67, 1994.

BLAZE, M., y BELLOVIN, S., "Tapping on My Network Door", *Commun. of the ACM,* vol. 43, p. 136, octubre de 2000.

BOGINENI, K.; SIVALINGAM, K.M., y DOWD, P.W., "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks", *IEEE Journal on Selected Areas in Commun.,* vol. 11, págs. 590-604, mayo de 1993.

BOLCSKEI, H.; PAULRAJ, A.J.; HARI, K.V.S., y NABAR, R.U., "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Commun. Magazine,* vol. 39, págs. 100-108, enero de 2001.

BORISOV, N.; GOLDBERG, I., y WAGNER, D., "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 180-188, 2001.

BRANDS, S., *Rethinking Public Key Infrastructures and Digital Certificates,* Cambridge, Massachusetts: M.I.T. Press, 2000.

BRAY, J., y STURMAN, C.F., *Bluetooth 1.1: Connect without Cables,* 2a. ed., Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

BREYER, R., y RILEY, S., *Switched, Fast, and Gigabit Ethernet,* Indianápolis, Indiana: New Riders, 1999.

BROWN, S., *Implementing Virtual Private Networks,* Nueva York: McGraw-Hill, 1999.

BROWN, L.; KWAN, M.; PIEPRZYK, J., y SEBERRY, J., "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI", *ASIACRYPT '91 Abstracts,* págs. 25-30, 1991.

BURNETT, S., y PAINE, S., *RSA Security's Official Guide to Cryptography,* Berkeley, California: Osborne/McGraw-Hill, 2001.

CAPETANAKIS, J.I., "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory,* vol. IT-25, págs. 505-515, septiembre de 1979.

CARDELLINI, V.; CASALICCHIO, E.; COLAJANNI, M., y YU, P.S., "*The State-of-the-Art in Locally Distributed Web-Server Systems*", *ACM Computing Surveys,* vol. 34, págs. 263 311, junio de 2002.

CARLSON, J., *PPP Design, Implementation and Debugging,* 2a. ed., Boston: Addison-Wesley, 2001.

CERF, V., y KAHN, R., "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.,* vol. COM-22, págs. 637-648, mayo de 1974.

CHAKRABARTI, S., "QoS Issues in Ad Hoc Wireless Networks", *IEEE Commun. Magazine,* vol. 39, págs. 142-148, febrero de 2001.

CHASE, J.S.; GALLATIN, A.J., y YOCUM, K.G., "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine,* vol. 39, págs. 68-75, abril de 2001.

CHEN, B.; JAMIESON, K.; BALAKRISHNAN, H., y MORRIS, R., "Span: An Energy Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *ACM Wireless Networks,* vol. 8, septiembre de 2002.

CHEN, K.-C., "Medium Access Control of Wireless LANs for Mobile Computing", *IEEE Network Magazine,* vol. 8, págs. 50-63, septiembre-octubre de 1994.

CHOUDBURY, A.K.; MAXEMCHUK, N.F.; PAUL, S., y SCHULZRINNE, H.G., "Copyright Protection for Electronic Publishing on Computer Networks", *IEEE Network Magazine,* vol. 9, págs. 12-20, mayo-junio de 1995.

CHU, Y.; RAO, S.G., y ZHANG, H., "A Case for End System Multicast", *Proc. Int'l Conf. on Measurements and Modeling of Computer Syst.,* ACM, págs. 1-12, 2000.

CLARK, D.D., "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.,* ACM, págs. 106-114, 1988.

CLARK, D.D., "Window and Acknowledgement Strategy in TCP", RFC 813, julio de 1982.

CLARK, D.D.; DAVIE, B.S.; FARBER, D.J.; GOPAL, I.S.; KADABA, B.K.; SINCOSKIE, W.D.; SMITH, J.M., y TENNENHOUSE, D.L., "The Aurora Gigabit Testbed", *Computer Networks and ISDN Systems,* vol. 25, págs. 599-621, enero de 1993.

CLARK, D.D.; JACOBSON, V.; ROMKEY, J., y SALWEN, H., "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine,* vol. 27, págs. 23-29, junio de 1989.

CLARK, D.D.; LAMBERT, M., y ZHANG, L., "NETBLT: A High Throughput Transport Protocol", *Proc. SIGCOMM '87 Conf.,* ACM, págs. 353-359, 1987.

CLARKE, A.C., "Extra-Terrestrial Relays", *Wireless World,* 1945.

CLARKE, I.; MILLER, S.G.; HONG, T.W.; SANDBERG, O., y WILEY, B., "Protecting Free Expression Online with Freenet", *IEEE Internet Computing,* vol. 6, págs. 40-49, enero-febrero de 2002.

COLLINS, D., *Carrier Grade Voice over IP,* Nueva York: McGraw-Hill, 2001.

COLLINS, D., y SMITH, C., *3G Wireless Networks,* Nueva York: McGraw-Hill, 2001.

COMER, D.E., *The Internet Book*, Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

COMER, D.E., *Internetworking with TCP/IP,* vol. 1, 4a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2000.

COSTA, L.H.M.K.; FDIDA, S., y DUARTE, O.C.M.B., "Hop by Hop Multicast Routing Protocol", *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun.,* ACM, págs. 249-259, 2001.

CRAVER, S.A.; WU, M.; LIU, B.; STUBBLEFIELD, A., SWARTZLANDER, B.; WALLACH, D.W.; DEAN, D., y FELTEN, E.W., "Reading Between the Lines: Lessons from the SDMI Challenge", *Proc. 10th USENIX Security Symp.,* USENIX, 2001.

CRESPO, P.M.; HONIG, M.L., y SALEHI, J.A., "Spread-Time Code-Division Multiple Access", *IEEE Trans. on Commun.,* vol. 43, págs. 2139-2148, junio de 1995.

CROW, B.P.; WIDJAJA, I; KIM, J.G., y SAKAI, P.T., "IEEE 802.11 Wireless Local Area Networks", *IEEE Commun. Magazine,* vol. 35, págs. 116-126, septiembre de 1997.

CROWCROFT, J.; WANG, Z.; SMITH, A., y ADAMS, J., "A Rough Comparison of the IETF and ATM Service Models", *IEEE Network Magazine,* vol. 9, págs. 12-16, noviembre-diciembre de 1995.

DABEK, F.; BRUNSKILL, E.; KAASHOEK, M.F.; KARGER, D.; MORRIS, R.; STOICA, R., y BALAKRISHNAN, H., "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service", *Proc. 8th Workshop on Hot Topics in Operating Systems,* IEEE, págs. 71-76, 2001a.

DABEK, F.; KAASHOEK, M.F.; KARGER, D.; MORRIS, R., y STOICA, I., "Wide-Area Cooperative Storage with CFS", *Proc. 18th Symp. on Operating Systems Prin.,* ACM, págs. 202-215, 2001b.

DAEMEN, J., y RIJMEN, V., *The Design of Rijndael,* Berlín: Springer-Verlag, 2002.

DANTHINE, A.A.S., "Protocol Representation with Finite-State Models", *IEEE Trans. on Commun.,* vol. COM-28, págs. 632-643, abril de 1980.

DAVIDSON, J., y PETERS, J. *Voice over IP Fundamentals,* Indianápolis, Indiana: Cisco Press, 2000.

DAVIE, B., y REKHTER, Y., *MPLS Technology and Applications,* San Francisco: Morgan Kaufmann, 2000.

DAVIS, P.T., y McGUFFIN, C.R., *Wireless Local Area Networks,* Nueva York: McGraw-Hill, 1995.

DAVISON, B.D., "A Web Caching Primer", *IEEE Internet Computing,* vol. 5, págs. 38-45, julio-agosto de 2001.

DAY, J.D., "The (Un)Revised OSI Reference Model", *Computer Commun. Rev.,* vol. 25, págs. 39-55, octubre de 1995.

DAY, J.D., y ZIMMERMANN, H., "The OSI Reference Model", *Proc. of the IEEE,* vol. 71, págs. 1334-1340, diciembre de 1983.

DE VRIENDT, J.; LAINE, P.; LEROUGE, C., y XU, X., "Mobile Network Evolution: A Revolution on the Move", *IEEE Commun. Magazine,* vol. 40, págs. 104-111, abril de 2002.

DEERING, S.E., "SIP: Simple Internet Protocol", *IEEE Network Magazine,* vol. 7, pp. 16-28, mayo-junio de 1993.

DEMERS, A.; KESHAV, S., y SHENKER, S., "Analysis and Simulation of a Fair Queueing Algorithm", *Internetwork: Research and Experience,* vol. 1, págs. 3-26, septiembre de 1990.

DENNING, D.E., y SACCO, G.M., "Timestamps in Key Distribution Protocols", *Commun. of the ACM,* vol. 24, págs. 533-536, agosto de 1981.

DIFFIE, W., y HELLMAN, M.E., "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer,* vol. 10, págs. 74-84, junio de 1977.

DIFFIE, W., y HELLMAN, M.E., "New Directions in Cryptography", *IEEE Trans. on Information Theory,* vol. IT-22, págs. 644-654, noviembre de 1976.

DIJKSTRA, E.W., "A Note on Two Problems in Connexion with Graphs", *Numer. Math.,* vol. 1, págs. 269-271, octubre de 1959.

DOBROWSKI, G., y GRISE, D., *ATM and SONET Basics,* Fuquay-Varina, Carolina del Norte: APDG Telecom Books, 2001.

DONALDSON, G., y JONES, D., "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine,* vol. 39, págs. 122-126, junio de 2001.

DORFMAN, R., "Detection of Defective Members of a Large Population", *Annals Math. Statistics,* vol. 14, págs. 436-440, 1943.

DOUFEXI, A.; ARMOUR, S.; BUTLER, M.; NIX, A.; BULL, D.; McGEEHAN, J., y KARLSSON, P., "A Comparison of the HIPERLAN/2 and IEEE 802.11A Wireless LAN Standards", *IEEE Commun. Magazine,* vol. 40, págs. 172-180, mayo de 2002.

DURAND, A., "Deploying IPv6", *IEEE Internet Computing,* vol. 5, págs. 79-81, enero-febrero de 2001.

DUTCHER, B., *The NAT Handbook*, Nueva York: Wiley, 2001.

DUTTA-ROY, A., "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine,* vol. 39, págs. 81-88, junio de 2001.

EASTTOM, C., *Learn JavaScript,* Ashburton, Reino Unido: Wordware Publishing, 2001.

EL GAMAL, T., "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory,* vol. IT-31, págs. 469-472, julio de 1985.

ELHANANY, I.; KAHANE, M., y SADOT, D., "Packet Scheduling in Next-Generation Multiterabit Networks", *Computer,* vol. 34, págs. 104-106, abril de 2001.

ELMIRGHANI, J.M.H., y MOUFTAH, H.T., "Technologies and Architectures for Scalable Dynamic Dense WDM Networks", *IEEE Commun. Magazine,* vol. 38, págs. 58-66, febrero de 2000.

FARSEROTU, J., y PRASAD, R., "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends", *IEEE Commun. Magazine,* vol. 38, págs. 128-133, junio de 2000.

FIORINI, D.; CHIANI, M.; TRALLI, V., y SALATI, C., "Can we Trust HDLC?", *Computer Commun. Rev.,* vol. 24, págs. 61-80, octubre de 1994.

FLOYD, S., y JACOBSON, V., "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking,* vol. 1, págs. 397-413, agosto de 1993.

FLUHRER, S.; MANTIN, I., y SHAMIR, A., "Weakness in the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas in Cryptography,* 2001.

FORD, L.R., Jr., y FULKERSON, D.R., *Flows in Networks*, Princeton, Nueva Jersey: Princeton, University Press, 1962.

FORD, W., y BAUM, M.S., *Secure Electronic Commerce,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2000.

FORMAN, G.H., y ZAHORJAN, J., "The Challenges of Mobile Computing", *Computer,* vol. 27, págs. 38-47, abril de 1994.

FRANCIS, P., "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine,* vol. 7, págs. 30-37, mayo-junio de 1993.

FRASER, A.G., "Towards a Universal Data Transport System", en *Advances in Local Area Networks,* Kummerle, K.; Tobagi, F., y Limb, J.O. (Eds.), Nueva York: IEEE Press, 1987.

FRENGLE, N., *I-Mode: A Primer,* Nueva York: Hungry Minds, 2002.

GADECKI, C., y HECKERT, C., *ATM for Dummies,* Nueva York: Hungry Minds, 1997.

GARBER, L., "Will 3G Really Be the Next Big Wireless Technology?", *Computer,* vol. 35, págs. 26-32, enero de 2002.

GARFINKEL, S., con SPAFFORD, G., *Web Security, Privacy, and Commerce,* Sebastopol, California: O'Reilly, 2002.

GEIER, J., *Wireless LANs,* 2a. ed., Indianápolis, Indiana: Sams, 2002.

GEVROS, P.; CROWCROFT, J.; KIRSTEIN, P., y BHATTI, S., "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network Magazine,* vol. 15, págs. 16-25, mayo-junio de 2001.

GHANI, N., y DIXIT, S., "TCP/IP Enhancements for Satellite Networks", *IEEE Commun. Magazine,* vol. 37, págs. 64-72, 1999.

GINSBURG, D., *ATM: Solutions for Enterprise Networking,* Boston: Addison-Wesley, 1996.

GOODMAN, D.J., "The Wireless Internet: Promises and Challenges", *Computer,* vol. 33, págs. 36-41, julio de 2000.

GORALSKI, W.J., *Introduction to ATM Networking,* Nueva York: McGraw-Hill, 1995.

GORALSKI, W.J., *Optical Networking and WDM,* Nueva York: McGraw-Hill, 2001.

GORALSKI, W.J., *SONET,* 2a. ed., Nueva York: McGraw-Hill, 2000.

GOSSAIN, H., DE MORAIS CORDEIRO y AGRAWAL, D.P., "Multicast: Wired to Wireless", *IEEE Commun. Mag.,* vol. 40, págs. 116-123, junio de 2002.

GRAVANO, S., *Introduction to Error Control Codes*, Oxford, Reino Unido: Oxford University Press, 2001.

GUO, Y., y CHASKAR, H., "Class-Based Quality of Service over Air Interfaces in 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 132-137, marzo de 2002.

HAARTSEN, J., "The Bluetooth Radio System", *IEEE Personal Commun. Magazine,* vol. 7, págs. 28-36, febrero de 2000.

HAC, A., "Wireless and Cellular Architecture and Services", *IEEE Commun. Magazine,* vol. 33, págs. 98-104, noviembre de 1995.

HAC, A., y GUO, L., "A Scalable Mobile Host Protocol for the Internet", *Int'l J. of Network Mgmt,* vol. 10, págs. 115-134, mayo-junio de 2000.

HALL, E., y CERF, V., *Internet Core Protocols: The Definitive Guide,* Sebastopol, California: O'Reilly, 2000.

HAMMING, R.W., "Error Detecting and Error Correcting Codes", *Bell System Tech. J.,* vol. 29, págs. 147-160, abril de 1950.

HANEGAN, K., *Custom CGI Scripting with Perl,* Nueva York: Wiley, 2001.

HARRIS, A., *JavaScript Programming for the Absolute Beginner,* Premier Press, 2001.

HARTE, L.; KELLOGG, S.; DREHER, R., y SCHAFFNIT, T., *The Comprehensive Guide to Wireless Technology,* Fuquay-Varina, Carolina del Norte: APDG Publishing, 2000.

HARTE, L.; LEVINE, R., y KIKTA, R., *3G Wireless Demystified,* Nueva York: McGraw-Hill, 2002.

HAWLEY, G.T., "Historical Perspectives on the U.S. Telephone System", *IEEE Commun. Magazine,* vol. 29, págs. 24-28, marzo de 1991.

HECHT, J., "Understanding Fiber Optics", Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

HEEGARD, C.; COFFEY, J.T.; GUMMADI, S.; MURPHY, P.A.; PROVENCIO, R.; ROSSIN, E.J.; SCHRUM, S., y SHOEMAKER, M.B., "High-Performance Wireless Ethernet", *IEEE Commun. Magazine,* vol. 39, págs. 64-73, noviembre de 2001.

HELD, G., *The Complete Modem Reference,* 2a. ed., Nueva York: Wiley, 1994.

HELLMAN, M.E., "A Cryptanalytic Time-Memory Tradeoff", *IEEE Trans. on Information Theory,* vol. IT-26, págs. 401-406, julio de 1980.

HILLS, A., "Large-Scale Wireless LAN Design", *IEEE Commun. Magazine,* vol. 39, págs. 98-104, noviembre de 2001.

HOLZMANN, G.J., *Design and Validation of Computer Protocols,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1991.

HU, Y., y LI, V.O.K., "Satellite-Based Internet Access", *IEEE Commun. Magazine,* vol. 39, págs. 155-162, marzo de 2001.

HU, Y.-C., y JOHNSON, D.B., "Implicit Source Routes for On-Demand Ad Hoc Network Routing", *Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking & Computing,* ACM, págs. 1-10, 2001.

HUANG, V., y ZHUANG, W., "QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications", *IEEE Commun. Magazine,* vol. 40, págs. 118-125, marzo de 2002.

HUBER, J.F.; WEILER, D., y BRAND, H., "UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization", *IEEE Commun. Magazine,* vol. 38, págs. 129-136, septiembre de 2000.

HUI, J., "A Broadband Packet Switch for Multi-rate Services", *Proc. Int'l Conf. on Commun.,* IEEE, págs. 782-788, 1987.

HUITEMA, C., *Routing in the Internet*, Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

HULL, S., *Content Delivery Networks,* Berkeley, California: Osborne/McGraw-Hill, 2002.

HUMBLET, P.A.; RAMASWAMI, R., y SIVARAJAN, K.N., "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 2-13, 1992.

HUNTER, D.K., y ANDONOVIC, I., "Approaches to Optical Internet Packet Switching", *IEEE Commun. Magazine,* vol. 38, págs. 116-122, septiembre de 2000.

HUSTON, G., "TCP in a Wireless World", *IEEE Internet Computing,* vol. 5, págs. 82-84, marzo-abril de 2001.

IBE, O.C., *Essentials of ATM Networks and Services,* Boston: Addison-Wesley, 1997.

IRMER, T., "Shaping Future Telecommunications: The Challenge of Global Standardization", *IEEE Commun. Magazine,* vol. 32, págs. 20-28, enero de 1994.

IZZO, P., *Gigabit Networks,* Nueva York: Wiley, 2000.

JACOBSON, V., "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.,* ACM, págs. 314-329, 1988.

JAIN, R., "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey", *Computer Networks and ISDN Systems,* vol. 27, noviembre de 1995.

JAIN, R., "Congestion Control in Computer Networks: Issues and Trends", *IEEE Network Magazine,* vol. 4, págs. 24-30, mayo-junio de 1990.

JAIN, R., *FDDI Handbook—High-Speed Networking Using Fiber and Other Media,* Boston: Addison-Wesley, 1994.

JAKOBSSON, M., y WETZEL, S., "Security Weaknesses in Bluetooth", *Topics in Cryptology: CT-RSA 2001,* Berlín: Springer-Verlag LNCS 2020, págs. 176-191, 2001.

JOEL, A., "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine,* edición del 50 aniversario, págs. 6-14 y 162-167, mayo 2002.

JOHANSSON, P.; KAZANTZIDIS, M.; KAPOOR, R., y GERLA, M., "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network Magazine,* vol. 15, págs. 28-37, septiembre-octubre de 2001.

JOHNSON, D.B., "Scalable Support for Transparent Mobile Host Internetworking", *Wireless Networks,* vol. 1, págs. 311-321, octubre de 1995.

JOHNSON, H.W., *Fast Ethernet—Dawn of a New Network,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1996.

JOHNSON, N.F., y JAJODA, S., "Exploring Steganography: Seeing the Unseen", *Computer,* vol. 31, págs. 26-34, febrero de 1998.

KAHN, D., "Cryptology Goes Public", *IEEE Commun. Magazine,* vol. 18, págs. 19-28, marzo de 1980.

KAHN, D., *The Codebreakers,* 2a. ed., Nueva York: Macmillan, 1995.

KAMOUN, F., y KLEINROCK, L., "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks,* vol. 3, págs. 337-353, noviembre de 1979.

KAPP, S., "802.11: Leaving the Wire Behind", *IEEE Internet Computing,* vol. 6, págs. 82-85, enero-febrero de 2002.

KARN, P., "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.,* págs. 134-140, 1990.

KARTALOPOULOS, S., *Introduction to DWDM Technology: Data in a Rainbow,* Nueva York, Nueva York: IEEE Communications Society, 1999.

KASERA, S.K.; HJALMTYSSON, G.; TOWLSEY, D.F., y KUROSE, J.F., "Scalable Reliable Multicast Using Multiple Multicast Channels", *IEEE/ACM Trans. on Networking,* vol. 8, págs. 294-310, 2000.

KATZ, D., y FORD, P.S., "TUBA: Replacing IP with CLNP", *IEEE Network Magazine,* vol. 7, págs. 38-47, mayo-junio de 1993.

KATZENBEISSER, S., y PETITCOLAS, F.A.P., *Information Hiding Techniques for Steganography and Digital Watermarking,* Londres, Artech House, 2000.

KAUFMAN, C.; PERLMAN, R., y SPECINER, M., *Network Security,* 2a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2002.

KELLERER, W.; VOGEL, H.-J., y STEINBERG, K.-E., "A Communication Gateway for Infrastructure-Independent 4G Wireless Access", *IEEE Commun. Magazine,* vol. 40, págs. 126-131, marzo de 2002.

KERCKHOFF, A., "La Cryptographie Militaire", *J. des Sciences Militaires,* vol. 9, págs. 5-38, enero de 1883 y págs. 161-191, febrero de 1883.

KIM, J.B.; SUDA, T., y YOSHIMURA, M., "International Standardization of B-ISDN", *Computer Networks and ISDN Systems,* vol. 27, págs. 5-27, octubre de 1994.

KIPNIS, J., "Beating the System: Abuses of the Standards Adoptions Process", *IEEE Commun. Magazine,* vol. 38, págs. 102-105, julio de 2000.

KLEINROCK, L., "On Some Principles of Nomadic Computing and Multi-Access Communications", *IEEE Commun. Magazine,* vol. 38, págs. 46-50, julio de 2000.

KLEINROCK, L., y TOBAGI, F., "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels", *Proc. Nat. Computer Conf.,* págs. 187-201, 1975.

KRISHNAMURTHY, B., y REXFORD, J., *Web Protocols and Practice,* Boston: Addison-Wesley, 2001.

KUMAR, V.; KORPI, M., y SENGODAN, S., *IP Telephony with H.323,* Nueva York: Wiley, 2001.

KUROSE, J.F., y ROSS, K.W., *Computer Networking: A Top-Down Approach Featuring the Internet,* Boston: Addison-Wesley, 2001.

KWOK, T., "A Vision for Residential Broadband Service: ATM to the Home", *IEEE Network Magazine,* vol. 9, págs. 14-28, septiembre-octubre de 1995.

KYAS, O., y CRAWFORD, G., *ATM Networks,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

LAM, C.K.M., y TAN, B.C.Y., "The Internet Is Changing the Music Industry", *Commun. of the ACM,* vol. 44, págs. 62-66, agosto de 2001.

LANSFORD, J.; STEPHENS, A, y NEVO, R., "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence", *IEEE Network Magazine,* vol. 15, págs. 20-27, septiembre-octubre de 2001.

LASH, D.A., *The Web Wizard's Guide to Perl and CGI,* Boston: Addison-Wesley, 2002.

LAUBACH, M.E.; FARBER, D.J., y DUKES, S.D., *Delivering Internet Connections over Cable,* Nueva York: Wiley, 2001.

LEE, J.S., y MILLER, L.E., *CDMA Systems Engineering Handbook,* Londres: Artech House, 1998.

LEEPER, D.G., "A Long-Term View of Short-Range Wireless", *Computer,* vol. 34, págs. 39-44, junio de 2001.

LEINER, B.M.; COLE, R.; POSTEL, J., y MILLS, D., "The DARPA Internet Protocol Suite", *IEEE Commun. Magazine,* vol. 23, págs. 29-34, marzo de 1985.

LEVINE, D.A., y AKYILDIZ, I.A., "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology", *IEEE/ACM Trans. on Networking,* vol. 3, págs. 158-168, abril de 1995.

LEVY, S., "Crypto Rebels", *Wired,* págs. 54-61, mayo-junio de 1993.

LI, J.; BLAKE, C.; DE COUTO, D.S.J.; LEE, H.I., y MORRIS, R., "Capacity of Ad Hoc Wireless Networks", *Proc. 7th Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 61-69, 2001.

LIN, F.; CHU, P., y LIU, M., "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies", *Proc. SIGCOMM '87 Conf.,* ACM, págs. 126-135, 1987.

LIN, Y.-D.; HSU, N.-B., y HWANG, R.-H., "QoS Routing Granularity in MPLS Networks", *IEEE Commun. Magazine,* vol. 40, págs. 58-65, junio de 2002.

LISTANI, M.; ERAMO, V., y SABELLA, R., "Architectural and Technological Issues for Future Optical Internet Networks", *IEEE Commun. Magazine,* vol. 38, págs. 82-92, septiembre de 2000.

LIU, C.L., y LAYLAND, J.W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of the ACM,* vol. 20, págs. 46-61, enero de 1973.

LIVINGSTON, D., *Essential XML for Web Professionals,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

LOSHIN, P., *IPv6 Clearly Explained,* San Francisco: Morgan Kaufmann, 1999.

LOUIS, P.J., *Broadband Crash Course,* Nueva York: McGraw-Hill, 2002.

LU, W., *Broadband Wireless Mobile: 3G and Beyond,* Nueva York: Wiley, 2002.

MACEDONIA, M.R., "Distributed File Sharing", *Computer,* vol. 33, págs. 99-101, 2000.

MADRUGA, E.L., y GARCIA-LUNA-ACEVES, J.J., "Scalable Multicasting: the Core Assisted Mesh Protocol", *Mobile Networks and Applications,* vol. 6, págs. 151-165, abril de 2001.

MALHOTRA, R., *IP Routing,* Sebastopol, California: O'Reilly, 2002.

MATSUI, M., "Linear Cryptanalysis Method for DES Cipher", *Advances in Cryptology—Eurocrypt '93 Proceedings,* Berlín: Springer-Verlag LNCS 765, págs. 386-397, 1994.

MAUFER, T.A., *IP Fundamentals,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1999.

MAZIERES, D., y KAASHOEK, M.F., "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf. on Computer and Commun. Security,* ACM, págs. 27-36, 1998.

MAZIERES, D.; KAMINSKY, M.; KAASHOEK, M.F., y WITCHEL, E., "Separating Key Management from File System Security", *Proc. 17th Symp. on Operating Systems Prin.,* ACM, págs. 124-139, diciembre de 1999.

McFEDRIES, P., *Using JavaScript,* Indianápolis, Indiana: Que, 2001.

McKENNEY, P.E., y DOVE, K.F., "Efficient Demultiplexing of Incoming TCP Packets", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 269-279, 1992.

MELTZER, K., y MICHALSKI, B., *Writing CGI Applications with Perl,* Boston: Addison-Wesley, 2001.

MENEZES, A.J., y VANSTONE, S.A., "Elliptic Curve Cryptosystems and Their Implementation", *Journal of Cryptology,* vol. 6, págs. 209-224, 1993.

MERKLE, R.C., "Fast Software Encryption Functions", *Advances in Cryptology—CRYPTO '90 Proceedings,* Berlín: Springer-Verlag LNCS 473, págs. 476-501, 1991.

MERKLE, R.C., y HELLMAN, M., "Hiding and Signatures in Trapdoor Knapsacks", *IEEE Trans. on Information Theory,* vol. IT-24, págs. 525-530, septiembre de 1978.

MERKLE, R.C., y HELLMAN, M., "On the Security of Multiple Encryption", *Commun. of the ACM,* vol. 24, págs. 465-467, julio de 1981.

METCALFE, R.M., "Computer/Network Interface Design: Lessons from Arpanet and Ethernet", *IEEE Journal on Selected Areas in Commun.,* vol. 11, págs. 173-179, febrero de 1993.

METCALFE, R.M., "On Mobile Computing", *Byte,* vol. 20, pág. 110, septiembre de 1995.

METCALFE, R.M., y BOGGS; D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM*, vol. 19, págs. 395-404, julio de 1976.

METZ, C., "Differentiated Services", *IEEE Multimedia Magazine,* vol. 7, págs. 84-90, julio-septiembre de 2000.

METZ, C., "Interconnecting ISP Networks", *IEEE Internet Computing,* vol. 5, págs. 74-80, marzo-abril de 2001.

METZ, C., "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing,* vol. 2, págs. 14-18, noviembre-diciembre de 1998.

MILLER, B.A., y BISDIKIAN, C., *Bluetooth Revealed,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

MILLER, P., y CUMMINS, M., *LAN Technologies Explained,* Woburn, Massachusetts: Butterworth-Heinemann, 2000.

MINOLI, D., *Video Dialtone Technology,* Nueva York: McGraw-Hill, 1995.

MINOLI, D., y VITELLA, M., *ATM & Cell Relay for Corporate Environments,* Nueva York: McGraw-Hill, 1994.

MISHRA, P.P., y KANAKIA, H., "A Hop by Hop Rate-Based Congestion Control Scheme", *Proc. SIGCOMM '92 Conf.,* ACM, págs. 112-123, 1992.

MISRA, A.; DAS, S.; DUTTA, A.; McAULEY, A., y DAS, S., "IDMP-Based Fast Hand-offs and Paging in IP-Based 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 138-145, marzo de 2002.

MOGUL, J.C., "IP Network Performance", en *Internet System Handbook*, Lynch, D.C. y Rose, M.T. (eds.), Boston: Addison-Wesley, págs. 575-675, 1993.

MOK, A.K., y WARD, S.A., "Distributed Broadcast Channel Access", *Computer Networks*, vol. 3, págs. 327-335, noviembre de 1979.

MOY, J., "Multicast Routing Extensions", *Commun. of the ACM,* vol. 37, págs. 61-66, agosto de 1994.

MULLINS, J., "Making Unbreakable Code", *IEEE Spectrum*, págs. 40-45, mayo de 2002.

NAGLE, J., "Congestion Control in TCP/IP Internetworks", *Computer Commun. Rev.,* vol. 14, págs. 11-17, octubre de 1984.

NAGLE, J., "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.,* vol. COM-35, págs. 435-438, abril de 1987.

NARAYANASWAMI, C.; KAMIJOH, N.; RAGHUNATH, M.; INOUE, T.; CIPOLLA, T.; SANFORD, J.; SCHLIG, E.; VENTKITESWARAN, S.; GUNIGUNTALA, D.; KULKARNI, V., y YAMAZAKI, K., "IBM's Linux Watch: The Challenge of Miniaturization", *Computer,* vol. 35, págs. 33-41, enero de 2002.

NAUGHTON, J., "A Brief History of the Future", Woodstock, Nueva York: Overlook Press, 2000.

NEEDHAM, R.M., y SCHROEDER, M.D., "Authentication Revisited", *Operating Systems Rev.,* vol. 21, pág. 7, enero de 1987.

NEEDHAM, R.M., y SCHROEDER, M.D., "Using Encryption for Authentication in Large Networks of Computers", *Commun. of the ACM,* vol. 21, págs. 993-999, diciembre de 1978.

NELAKUDITI, S., y ZHANG, Z.-L., "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine,* vol. 40, págs. 66-71, junio de 2002.

NEMETH, E.; SNYDER, G.; SEEBASS, S., y HEIN, T.R., *UNIX System Administration Handbook,* 3a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 2000.

NICHOLS, R.K., y LEKKAS, P.C., *Wireless Security,* Nueva York: McGraw-Hill, 2002.

NIST, "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

O'HARA, B., y PETRICK, A., *802.11 Handbook: A Designer's Companion,* Nueva York: IEEE Press, 1999.

OTWAY, D., y REES, O., "Efficient and Timely Mutual Authentication", *Operating Systems Rev.,* págs. 8-10, enero de 1987.

OVADIA, S., *Broadband Cable TV Access Networks: from Technologies to Applications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

PALAIS, J.C., *Fiber Optic Commun.,* 3a. ed., Englewood Cliffs, Nueva Jersey: Prentice Hall, 1992.

PAN, D., "A Tutorial on MPEG/Audio Compression", *IEEE Multimedia Magazine,* vol. 2, págs. 60-74, verano de 1995.

PANDYA, R., "Emerging Mobile and Personal Communication Systems", *IEEE Commun. Magazine,* vol. 33, págs. 44-52, junio de 1995.

PARAMESWARAN, M.; SUSARLA, A., y WHINSTON, A.B., "P2P Networking: An Information-Sharing Alternative", *Computer,* vol. 34, págs. 31-38, julio de 2001.

PARK, J.S., y SANDHU, R., "Secure Cookies on the Web", *IEEE Internet Computing,* vol. 4, págs. 36-44, julio-agosto de 2000.

PARTRIDGE, C.; HUGHES, J., y STONE, J., "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 68-76, 1995.

PAXSON, V., "Growth Trends in Wide-Area TCP Connections", *IEEE Network Magazine,* vol. 8, págs. 8-17, julio-agosto de 1994.

PAXSON, V., y FLOYD, S., "Wide-Area Traffic: The Failure of Poisson Modeling", *Proc. SIGCOMM '94 Conf.,* ACM, págs. 257-268, 1995.

PEPELNJAK, I., y GUICHARD, J., *MPLS and VPN Architectures,* Indianápolis, Indiana: Cisco Press, 2001.

PERKINS, C.E., *Mobile IP Design Principles and Practices,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1998a.

PERKINS, C.E., "Mobile Networking in the Internet", *Mobile Networks and Applications,* vol. 3, págs. 319-334, 1998b.

PERKINS, C.E., "Mobile Networking through Mobile IP", *IEEE Internet Computing,* vol. 2, págs. 58-69, enero-febrero de 1998c.

PERKINS, C.E. (ed.), *Ad Hoc Networking,* Boston: Addison-Wesley, 2001.

PERKINS, C.E., *RTP: Audio and Video for the Internet,* Boston: Addison-Wesley, 2002.

PERKINS, C.E., y ROYER, E., "Ad-hoc On-Demand Distance Vector Routing", *Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications,* IEEE, págs. 90-100, 1999.

PERKINS, C.E., y ROYER, E., "The Ad Hoc On-Demand Distance-Vector Protocol", en *Ad Hoc Networking,* editado por C. Perkins, Boston: Addison-Wesley, 2001.

PERLMAN, R., *Network Layer Protocols with Byzantine Robustness,* Ph.D. thesis, M.I.T., 1988.

PERLMAN, R., *Interconnections,* 2a. ed., Boston: Addison-Wesley, 2000.

PERLMAN, R., y KAUFMAN, C., "Key Exchange in IPsec", *IEEE Internet Computing,* vol. 4, págs. 50-56, noviembre-diciembre de 2000.

PETERSON, L.L., y DAVIE, B.S., *Computer Networks: A Systems Approach,* San Francisco: Morgan Kaufmann, 2000.

PETERSON, W.W., y BROWN, D.T., "Cyclic Codes for Error Detection", *Proc. IRE,* vol. 49, págs. 228-235, enero de 1961.

PICKHOLTZ, R.L.; SCHILLING, D.L., y MILSTEIN, L.B., "Theory of Spread Spectrum Communication—A Tutorial", *IEEE Trans. on Commun.,* vol. COM-30, págs. 855-884, mayo de 1982.

PIERRE, G.; KUZ, I.; VAN STEEN, M., y TANENBAUM, A.S., "Differentiated Strategies for Replicating Web Documents", *Computer Commun.,* vol. 24, págs. 232-240, febrero de 2001.

PIERRE, G.; VAN STEEN, M., y TANENBAUM, A.S., "Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Trans. on Computers,* vol. 51, junio de 2002.

PISCITELLO, D.M., y CHAPIN, A.L., *Open Systems Networking: TCP/IP and OSI,* Boston: Addison-Wesley, 1993.

PITT, D.A., "Bridging—The Double Standard", *IEEE Network Magazine*, vol. 2, págs. 94-95, enero de 1988.

PIVA, A., BARTOLINI, F., y BARNI, M., "Managing Copyrights in Open Networks, *IEEE Internet Computing,* vol. 6, págs. 18-26, mayo-junio de 2002.

POHLMANN, N., *Firewall Systems,* Bonn, Alemania: MITP-Verlag, 2001.

PUZMANOVA, R., *Routing and Switching: Time of Convergence?,* Londres: Addison-Wesley, 2002.

RABINOVICH, M., y SPATSCHECK, O,. *Web Caching and Replication,* Boston: Addison-Wesley, 2002.

RAJU, J., y GARCIA-LUNA-ACEVES, J.J., "Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks", *ACM Computer Communications Review,* vol. 31, octubre de 2001.

RAMANATHAN, R., y REDI, J., "A Brief Overview of Ad Hoc Networks: Challenges and Directions", *IEEE Commun. Magazine,* edición de 50 aniversario, págs. 20-22, mayo de 2002.

RATNASAMY, S.; FRANCIS, P.; HANDLEY, M.; KARP, R., y SHENKER, S., "A Scalable Content-Addressable Network", *Proc. SIGCOMM '01 Conf.,* ACM, págs. 1161-1172, 2001.

RIVEST, R.L., "The MD5 Message-Digest Algorithm", RFC 1320, abril de 1992.

RIVEST, R.L., y SHAMIR, A., "How to Expose an Eavesdropper", *Commun. of the ACM,* vol. 27, págs. 393-395, abril de 1984.

RIVEST, R.L.; SHAMIR, A., y ADLEMAN, L., "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM,* vol. 21, págs. 120-126, febrero de 1978.

ROBERTS, L.G., "Dynamic Allocation of Satellite Capacity through Packet Reservation", *Proc. NCC,* AFIPS*,* págs. 711-716, 1973.

ROBERTS, L.G., "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", *Proc. Spring Joint Computer Conference,* AFIPS, págs. 295-298, 1972.

ROBERTS, L.G., "Multiple Computer Networks and Intercomputer Communication", *Proc. First Symp. on Operating Systems Prin.,* ACM, 1967.

ROSE, M.T., *The Internet Message,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1993.

ROSE, M.T., *The Simple Book,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1994.

ROSE, M.T., y McCLOGHRIE, K., *How to Manage Your Network Using SNMP,* Englewood Cliffs, Nueva Jersey: Prentice Hall, 1995.

ROWSTRON, A., y DRUSCHEL, P., "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility", *Proc. 18th Symp. on Operating Systems Prin.,* ACM, págs. 188-201, 2001a.

ROWSTRON, A., y DRUSCHEL, P., "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18th Int'l Conf. on Distributed Systems Platforms,* ACM/IFIP, 2001b.

ROYER, E.M., y TOH, C.-K., "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Commun. Magazine,* vol. 6, págs. 46-55, abril de 1999.

RUIZ-SANCHEZ, M.A.; BIERSACK, E.W., y DABBOUS, W., "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine,* vol. 15, págs. 8-23, marzo-abril de 2001.

SAIRAM, K.V.S.S.S.S.; GUNASEKARAN, N., y REDDY, S.R., "Bluetooth in Wireless Communication", *IEEE Commun. Mag.,* vol. 40, págs. 90-96, junio de 2002.

SALTZER, J.H.; REED, D.P., y CLARK, D.D., "End-to-End Arguments in System Design", *ACM Trans. on Computer Systems,* vol. 2, págs. 277-288, noviembre de 1984.

SANDERSON, D.W., y DOUGHERTY, D., *Smileys,* Sebastopol, California: O'Reilly, 1993.

SARI, H.; VANHAVERBEKE, F., y MOENECLAEY, M., "Extending the Capacity of Multiple Access Channels", *IEEE Commun. Magazine,* vol. 38, págs. 74-82, enero de 2000.

SARIKAYA, B., "Packet Mode in Wireless Networks: Overview of Transition to Third Generation", *IEEE Commun. Magazine,* vol. 38, págs. 164-172, septiembre de 2000.

SCHNEIER, B., *Applied Cryptography,* 2a. ed., Nueva York: Wiley, 1996.

SCHNEIER, B., "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]", *Proc. of the Cambridge Security Workshop,* Berlín: Springer-Verlag LNCS 809, págs. 191-204, 1994.

SCHNEIER, B., *E-Mail Security,* Nueva York: Wiley, 1995.

SCHNEIER, B., *Secrets and Lies,* Nueva York: Wiley, 2000.

SCHNORR, C.P., "Efficient Signature Generation for Smart Cards", *Journal of Cryptology,* vol. 4, págs. 161-174, 1991.

SCHOLTZ, R.A., "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.,* vol. COM-30, págs. 822-854, mayo de 1982.

SCOTT, R., "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia,* vol. 9, págs. 75-90, enero de 1985.

SEIFERT, R., *Gigabit Ethernet,* Boston: Addison-Wesley, 1998.

SEIFERT, R., *The Switch Book,* Boston: Addison-Wesley, 2000.

SENN, J.A., "The Emergence of M-Commerce", *Computer,* vol. 33, págs. 148-150, diciembre de 2000.

SERJANTOV, A., "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to-Peer Systems,* Berlín: Springer-Verlag LNCS, 2002.

SEVERANCE, C., "IEEE 802.11: Wireless Is Coming Home", *Computer,* vol. 32, págs. 126-127, noviembre de 1999.

SHAHABI, C.; ZIMMERMANN, R.; FU, K., y YAO, S.-Y.D., "YIMA: A Second-Generation Continuous Media Server", *Computer,* vol. 35, págs. 56-64, junio de 2002.

SHANNON, C., "A Mathematical Theory of Communication", *Bell System Tech. J.,* vol. 27, págs. 379-423, julio de 1948; y págs. 623-656, octubre de 1948.

SHEPARD, S., *SONET/SDH Demystified,* Nueva York: McGraw-Hill, 2001.

SHREEDHAR, M., y VARGHESE, G., "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 231-243, 1995.

SKOUDIS, E., *Counter Hack,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

SMITH, D.K., y ALEXANDER, R.C., *Fumbling the Future,* Nueva York: William Morrow, 1988.

SMITH, R.W., *Broadband Internet Connections,* Boston: Addison Wesley, 2002.

SNOEREN, A.C., y BALAKRISHNAN, H., "An End-to-End Approach to Host Mobility", *Int'l Conf. on Mobile Computing and Networking,* ACM, págs. 155-166, 2000.

SOBEL, D.L., "Will Carnivore Devour Online Privacy", *Computer,* vol. 34, págs. 87-88, mayo de 2001.

SOLOMON, J.D., *Mobile IP: The Internet Unplugged,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1998.

SPOHN, M., y GARCIA-LUNA-ACEVES, J.J., "Neighborhood Aware Source Routing", *Proc. ACM MobiHoc 2001,* ACM, págs. 2001.

SPURGEON, C.E., *Ethernet: The Definitive Guide,* Sebastopol, California: O'Reilly, 2000.

STALLINGS, W., *Data and Computer Communications,* 6a. ed., Upper Saddle River, Nueva Jersey: Prentice Hall, 2000.

STEINER, J.G.; NEUMAN, B.C., y SCHILLER, J.I., "Kerberos: An Authentication Service for Open Network Systems", *Proc. Winter USENIX Conf.,* USENIX, págs. 191-201, 1988.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 1: Media Coding and Content Processing,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 2: Media Proessing and Communications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2003a.

STEINMETZ, R., y NAHRSTEDT, K., *Multimedia Fundamentals. Vol. 3: Documents, Security, and Applications,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2003b.

STEVENS, W.R., *TCP/IP Illustrated,* Vol. 1, Boston: Addison-Wesley, 1994.

STEVENS, W.R., *UNIX Network Programming, Volume 1: Networking APIs - Sockets and XTI,* Upper Saddle River, Nueva Jersey: Prentice Hall, 1997.

STEWART, R., y METZ, C., "SCTP: New Transport Protocol for TCP/IP", *IEEE Internet Computing,* vol. 5, págs. 64-69, noviembre-diciembre de 2001.

STINSON, D.R., *Cryptography Theory and Practice,* 2a. ed., Boca Ratón, Florida: CRC Press, 2002.

STOICA, I.; MORRIS, R.; KARGER, D.; KAASHOEK, M.F., y BALAKRISHNAN, H., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. SIGCOMM '01 Conf.,* ACM, págs. 149-160, 2001.

STRIEGEL, A., y MANIMARAN, G., "A Survey of QoS Multicasting Issues", *IEEE Commun. Mag.,* vol. 40, págs. 82-87, junio de 2002.

STUBBLEFIELD, A.; IOANNIDIS, J., y RUBIN, A.D., "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proc Network and Distributed Systems Security Symp.,* ISOC, págs. 1-11, 2002.

SUMMERS, C.K., *ADSL: Standards, Implementation, and Architecture,* Boca Ratón, Florida: CRC Press, 1999.

SUNSHINE, C.A., y DALAL, Y.K., "Connection Management in Transport Protocols", *Computer Networks,* vol. 2, págs. 454-473, 1978.

TANENBAUM, A.S., *Modern Operating Systems,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2001.

TANENBAUM, A.S., y VAN STEEN, M., *Distributed Systems: Principles and Paradigms,* Upper Saddle River, Nueva Jersey: Prentice Hall, 2002.

TEGER, S., y WAKS, D.J., "End-User Perspectives on Home Networking", *IEEE Commun. Magazine,* vol. 40, págs. 114-119, abril de 2002.

THYAGARAJAN, A.S., y DEERING, S.E., "Hierarchical Distance-Vector Multicast Routing for the MBone", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 60-66, 1995.

TITTEL, E.; VALENTINE, C.; BURMEISTER, M., y DYKES, L., *Mastering XHTML,* Alameda, California: Sybex, 2001.

TOKORO, M., y TAMARU, K., "Acknowledging Ethernet", *Compcon,* IEEE, págs. 320-325, otoño de 1977.

TOMLINSON, R.S., "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop,* ACM, págs. 11-23, 1975.

TSENG, Y.-C.; WU, S.-L.; LIAO, W.-H., y CHAO, C.-M., "Location Awareness in Ad Hoc Wireless Mobile Networks", *Computer,* vol. 34, págs. 46-51, 2001.

TUCHMAN, W., "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum,* vol. 16, págs. 40-41, julio de 1979.

TURNER, J.S., "New Directions in Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine,* vol. 24, págs. 8-15, octubre de 1986.

VACCA, J.R., *I-Mode Crash Course,* Nueva York: McGraw-Hill, 2002.

VALADE, J., *PHP & MySQL for Dummies,* Nueva York: Hungry Minds, 2002.

VARGHESE, G, y LAUCK, T., "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.,* ACM, págs. 25-38, 1987.

VARSHNEY, U.; SNOW, A.; McGIVERN, M., y HOWARD, C., *"Voice over IP", Commun. of the ACM,* vol. 45, págs. 89-96, 2002.

VARSHNEY, U., y VETTER, R., "Emerging Mobile and Wireless Networks", *Commun. of the ACM,* vol. 43, págs. 73-81, junio de 2000.

VETTER, P.; GODERIS, D.; VERPOOTEN, L., y GRANGER, A., "Systems Aspects of APON/VDSL Deployment", *IEEE Commun. Magazine,* vol. 38, págs. 66-72, mayo de 2000.

WADDINGTON, D.G., y CHANG, F., "Realizing the Transition to IPv6", *IEEE Commun. Mag.,* vol. 40, págs. 138-148, junio de 2002.

WALDMAN, M.; RUBIN, A.D., y CRANOR, L.F., "Publius: A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System", *Proc. Ninth USENIX Security Symp.,* USENIX, págs. 59-72, 2000.

WANG, Y., y CHEN, W., "Supporting IP Multicast for Mobile Hosts", *Mobile Networks and Applications,* vol. 6, págs. 57-66, enero-febrero de 2001.

WANG, Z., *Internet QoS,* San Francisco: Morgan Kaufmann, 2001.

WARNEKE, B.; LAST, M.; LIEBOWITZ, B., y PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *Computer,* vol. 34, págs. 44-51, enero de 2001.

WAYNER, P., *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking,* 2a. ed., San Francisco: Morgan Kaufmann, 2002.

WEBB, W., "Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment", *IEEE Commun. Magazine,* vol. 39, págs. 115-121, septiembre de 2001.

WEISER, M., "Whatever Happened to the Next Generation Internet?", *Commun. of the ACM,* vol. 44, págs. 61-68, septiembre de 2001.

WELTMAN, R., y DAHBURA, T., *LDAP Programming with Java,* Boston: Addison-Wesley, 2000.

WESSELS, D., *Web Caching,* Sebastopol, California: O'Reilly, 2001.

WETTEROTH, D., *OSI Reference Model for Telecommunications,* Nueva York: McGraw-Hill, 2001.

WILJAKKA, J., "Transition to IPv6 in GPRS and WCDMA Mobile Networks", *IEEE Commun. Magazine,* vol. 40, págs. 134-140, abril de 2002.

WILLIAMSON, H., *XML: The Complete Reference,* Nueva York: McGraw-Hill, 2001.

WILLINGER, W.; TAQQU, M.S.; SHERMAN, R., y WILSON, D.V., "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proc. SIGCOMM '95 Conf.,* ACM, págs. 100-113, 1995.

WRIGHT, D.J., *Voice over Packet Networks,* Nueva York: Wiley, 2001.

WYLIE, J.; BIGRIGG, M.W.; STRUNK, J.D.; GANGER, G.R.; KILICCOTE, H., y KHOSLA, P.K., "Survivable Information Storage Systems", *Computer,* vol. 33, págs. 61-68, agosto de 2000.

XYLOMENOS, G.; POLYZOS, G.C.; MAHONEN, P., y SAARANEN, M., "TCP Performance Issues over Wireless Links", *IEEE Commun. Magazine,* vol. 39, págs. 52-58, abril de 2001.

YANG, C.-Q., y REDDY, A.V.S., "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks", *IEEE Network Magazine,* vol. 9, págs. 34-45, julio-agosto de 1995.

YUVAL, G., "How to Swindle Rabin", *Cryptologia,* vol. 3, págs. 187-190, julio de 1979.

ZACKS, M., "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing,* vol. 5, págs. 8-9, noviembre-diciembre de 2001.

ZADEH, A.N.; JABBARI, B.; PICKHOLTZ, R., y VOJCIC, B., "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag.,* vol. 40, págs. 149-157, junio de 2002.

ZHANG, L., "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine,* vol. 2, págs. 44-48, enero-febrero de 1988.

ZHANG, L., "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine,* vol. 7, págs. 8-18, septiembre-octubre de 1993.

ZHANG, Y., y RYU, B., "Mobile and Multicast IP Services in PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications,* vol. 6, págs. 81-94, enero-febrero de 2001.

ZIMMERMANN, P.R., *The Official PGP User's Guide,* Cambridge, Massachusetts: M.I.T. Press, 1995a.

ZIMMERMANN, P.R., *PGP: Source Code and Internals,* Cambridge, Massachusetts: M.I.T. Press, 1995b.

ZIPF, G.K., *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology,* Boston: Addison-Wesley, 1949.

ZIV, J., y LEMPEL, Z., "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory,* vol. IT-23, págs. 337-343, mayo de 1977.

ÍNDICE

Números

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

ACERCA DEL AUTOR

Andrew S. Tanenbaum tiene una licenciatura en ciencias del M.I.T. y un doctorado en filosofía por la University of California en Berkeley. Actualmente es profesor de ciencias de la computación en la Vrije Universiteit de Ámsterdam, Países Bajos, donde encabeza el Grupo de Sistemas de Computadoras. También es decano de la Escuela Avanzada de Computación y Procesamiento de Imágenes, una escuela interuniversitaria de posgrado en la que se llevan a cabo investigaciones sobre sistemas paralelos avanzados, distribuidos y de procesamiento de imágenes. No obstante, hace un gran esfuerzo para no convertirse en burócrata.

También ha efectuado investigaciones sobre compiladores, sistemas operativos, conectividad de redes y sistemas distribuidos de área local. Sus investigaciones actuales se enfocan principalmente en el diseño e implementación de sistemas distribuidos de área amplia en la escala de millones de usuarios. Estas investigaciones, realizadas junto con el profesor Maarten van Steen, se describen en *www.cs.vu.nl/globe*. Estos proyectos de investigación han conducido a más de 100 artículos evaluados por expertos en publicaciones periódicas y actas de congresos, así como a cinco libros.

Por otra parte, el profesor Tanenbaum ha producido una cantidad considerable de software; fue el arquitecto principal del *Amsterdam Compiler Kit*, un conjunto de herramientas de amplio uso para escribir compiladores portátiles, y de MINIX, un pequeño sistema operativo tipo UNIX para cursos de sistemas operativos. Este sistema fue la inspiración y base en la que se desarrolló Linux. Junto con sus estudiantes de doctorado y programadores, ayudó a diseñar Amoeba, un sistema operativo distribuido de alto desempeño basado en micronúcleo. MINIX y Amoeba ahora están disponibles gratuitamente en Internet.

Los estudiantes de doctorado del profesor Tanenbaum han cosechado grandes triunfos tras recibir sus títulos. Él está muy orgulloso de ellos.

El profesor Tanenbaum es socio del ACM, miembro senior del IEEE, miembro de la Academia Real Holandesa de Ciencias y Artes, ganador en 1994 del premio Karl V. Karlstrom del ACM para educadores sobresalientes, ganador en 1997 del Premio ACM/SIGCSE por contribuciones sobresalientes en la educación sobre ciencias de la computación y ganador en 2002 del premio Texty por su excelencia en libros de texto. También está listado en Web en el *Who's Who in the World*. Su página principal de World Wide Web se encuentra en *http://www.cs.vu.nl/~ast/*.