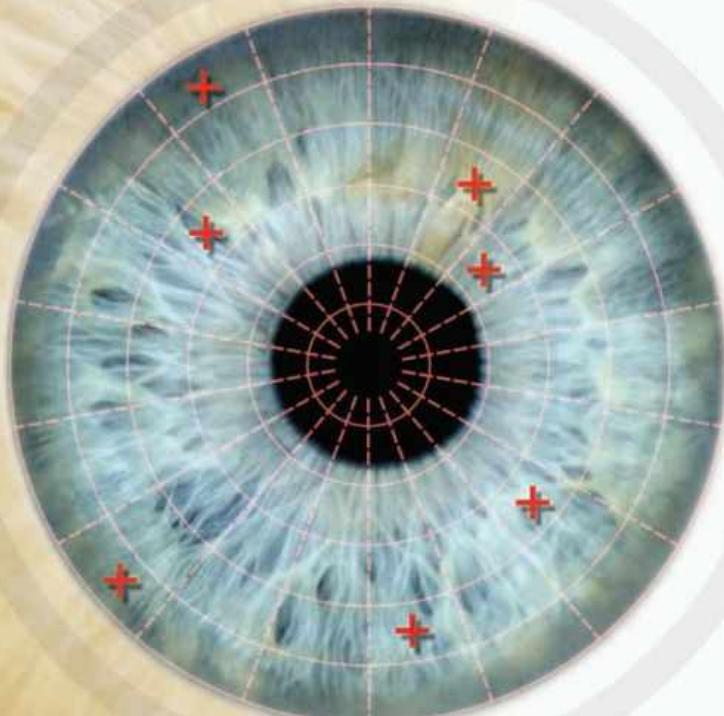


2^a Edición

Fundamentos de Seguridad en Redes

Aplicaciones y Estándares



PEARSON
Prentice
Hall

William Stallings

**FUNDAMENTOS DE SEGURIDAD
EN REDES**

APLICACIONES Y ESTÁNDARES

Segunda edición

FUNDAMENTOS DE SEGURIDAD EN REDES

APLICACIONES Y ESTÁNDARES

Segunda edición

William Stallings

Revisión técnica:

Manuel González Rodríguez

Facultad de Informática

Universidad de las Palmas de Gran Canaria

Luis Joyanes Aguilar

Universidad Pontificia de Salamanca, campus de Madrid

Traducción:

Laura Cruz García

Facultad de Informática

Universidad de Las Palmas de Gran Canaria

Manuel González Rodríguez

Facultad de Informática

Universidad de las Palmas de Gran Canaria



Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo
San Juan • San José • Santiago • São Paulo • White Plains

Datos de catalogación bibliográfica

STALLINGS, W.

FUNDAMENTOS DE SEGURIDAD EN REDES.

APLICACIONES Y ESTÁNDARES.

Segunda edición

PEARSON EDUCACIÓN, S.A., Madrid, 2004

ISBN: 84-205-4002-1

Materia: Informática 681.3

Formato: 170 × 240

Páginas: 432

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sgts. Código Penal).

DERECHOS RESERVADOS

© 2004 por PEARSON EDUCACIÓN, S.A.

Ribera del Loira, 28

28042 Madrid (España)

FUNDAMENTOS DE SEGURIDAD EN REDES. APLICACIONES Y ESTÁNDARDES. Segunda edición
STALLINGS, W.

ISBN: 84-205-4002-1

Depósito Legal: M-

PEARSON PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S. A.

Traducido de:

Network Security Essentials. Applications and Standards. Second edition, by Stallings, William.

Published by Pearson Education, Inc., publishing as Prentice Hall.

© 2003. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Equipo editorial:

Director: David Fayerman Aragón

Técnico editorial: Ana Isabel García Borro

Equipo de producción:

Director: José Antonio Clares

Técnico: José Antonio Hernán

Diseño de cubierta: Equipo de diseño de Pearson Educación, S.A.

Composición: Claroscuro Servicio Gráfico, S.L.

Impreso por:

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

Contenido

Prólogo	IX
Capítulo 1. Introducción	1
1.1. La arquitectura de seguridad OSI	4
1.2. Ataques a la seguridad	5
1.3. Servicios de seguridad	9
1.4. Mecanismos de seguridad	13
1.5. Un modelo de seguridad en redes	14
1.6. Estándares de Internet y la Sociedad Internet	17
1.7. Estructura del libro	21
1.8. Bibliografía recomendada	22
1.9. Recursos web y de Internet	22

PRIMERA PARTE Criptografía

Capítulo 2. Cifrado simétrico y confidencialidad de mensajes	27
2.1. Principios del cifrado simétrico	28
2.2. Algoritmos de cifrado simétrico	34
2.3. Modos de operación del cifrado de bloques	44
2.4. Ubicación de los dispositivos de cifrado	48
2.5. Distribución de claves	49
2.6. Bibliografía y sitios web recomendados	51
2.7. Palabras clave, preguntas de repaso y problemas	52
Capítulo 3. Criptografía de clave pública y autentificación de mensajes ..	55
3.1. Enfoques para la autentificación de mensajes	56
3.2. Funciones <i>hash</i> seguras y HMAC	61
3.3. Principios de criptografía de clave pública	71
3.4. Algoritmos de criptografía de clave pública	75
3.5. Firmas digitales	81
3.6. Gestión de claves	82
3.7. Bibliografía y sitios web recomendados	84
3.8. Términos clave, preguntas de repaso y problemas	85

SEGUNDA PARTE
Aplicaciones de seguridad en redes

Capítulo 4. Aplicaciones de autentificación	91
4.1. Kerberos	92
4.2. Servicio de autentificación de X.509	111
4.3. Bibliografía y sitios web recomendados	121
4.4. Términos clave, preguntas de repaso y problemas	121
Apéndice 4A. Técnicas de cifrado Kerberos	123
Capítulo 5. Seguridad en el correo electrónico	127
5.1. PGP (<i>Pretty Good Privacy</i>)	128
5.2. S/MIME	149
5.3. Sitios web recomendados	167
5.4. Términos clave, preguntas de repaso y problemas	167
Apéndice 5A. Compresión de datos usando Zip	168
Apéndice 5B. Conversión RADIX 64	171
Apéndice 5C. Generación de números aleatorios PGP	173
Capítulo 6. Seguridad IP	177
6.1. Introducción a la seguridad IP	178
6.2. Arquitectura de seguridad IP	181
6.3. Cabecera de autentificación	188
6.4. Encapsulamiento de la carga útil de seguridad	192
6.5. Combinación de asociaciones de seguridad	198
6.6. Gestión de claves	201
6.7. Bibliografía y sitios web recomendados	212
6.8. Términos clave, preguntas de repaso y problemas	212
Apéndice 6A. Comunicación entre redes y protocolos de Internet	214
Capítulo 7. Seguridad de la web	223
7.1. Consideraciones sobre seguridad en la web	224
7.2. SSL (<i>Secure Socket Layer</i>) y TLS (<i>Transport Layer Security</i>)	227
7.3. SET (<i>Secure Electronic Transaction</i>)	246
7.4. Bibliografía y sitios web recomendados	258
7.5. Palabras clave, preguntas de repaso y problemas	259
Capítulo 8. Seguridad en la gestión de redes	261
8.1. Conceptos básicos de SNMP	262
8.2. Comunidades SNMPv1	270
8.3. SNMPv3	272
8.4. Bibliografía y sitios web recomendados	298
8.5. Términos clave, preguntas de repaso y problemas	298

TERCERA PARTE
Seguridad de los sistemas

Capítulo 9. Intrusos	305
9.1. Intrusos	306
9.2. Detección de intrusos	310
9.3. Gestión de contraseñas	323
9.4. Bibliografía y sitios web recomendados	332
9.5. Términos clave, preguntas de repaso y problemas	334
Apéndice 9A. La falacia de la tasa base	336
Capítulo 10. Software dañino	341
10.1. Virus y otras amenazas	342
10.2. Contramedidas a los virus	353
10.3. Bibliografía y sitios web recomendados	358
10.4. Términos clave, preguntas de repaso y problemas	359
Capítulo 11. Cortafuegos	361
11.1. Principios de diseño de cortafuegos	362
11.2. Sistemas de confianza	374
11.3. Bibliografía y sitios web recomendados	380
11.4. Términos clave, preguntas de repaso y problemas	381
APÉNDICE A Estándares citados en este libro	383
A.1. Estándares ANSI	383
A.2. RFC de Internet	383
A.3. Recomendaciones ITU-T	385
A.4. Estándares de procesamiento de información de NIST	385
APÉNDICE B Algunos aspectos de la teoría de números	387
B.1. Números primos y primos relativos	388
B.2. Aritmética modular	390
Glosario	393
Referencias	399
Índice analítico	405

Prólogo

«*La corbata, si me permite la sugerencia, señor, lleva el nudo más apretado. Se persigue el efecto mariposa perfecto. ¿Me permite?*»

«*¿Qué importa, Jeeves, en un momento como éste? ¿Es que no ves cómo se tambalea la felicidad doméstica de Mr. Little?*»

«*Señor, no hay momento en el que las corbatas no importen.*»

***¡Muy bien, Jeeves!* P. G. Wodehouse**

En esta era de la conectividad electrónica universal, de virus y hackers, de escuchas y fraudes electrónicos, no hay un momento en el que no importe la seguridad. Dos tendencias han confluido para hacer de interés vital el tema de este libro. En primer lugar, el enorme crecimiento de los sistemas de computadores y sus interconexiones mediante redes ha hecho que organizaciones e individuos dependan cada vez más de la información que se almacena y se transmite a través de estos sistemas. Esto, a su vez, ha llevado a un aumento de la conciencia de la necesidad de proteger los datos y los recursos, de garantizar la autenticidad de los datos y los mensajes y de proteger los sistemas frente a ataques a la red. En segundo lugar, las disciplinas de la criptografía y la seguridad de la red han madurado, dando como resultado el desarrollo de aplicaciones prácticas, ya disponibles, para la seguridad de la red.

OBJETIVOS

El propósito de este libro es proporcionar un estudio práctico sobre las aplicaciones y los estándares relativos a la seguridad de la red. Se resaltan, por una parte, las aplicaciones que más se utilizan en Internet y en las redes corporativas y, por otra, los estándares más extendidos, especialmente los de Internet.

DESTINATARIOS

El libro está destinado a una audiencia tanto académica como profesional. Como libro de texto, está diseñado para cubrir un curso de un semestre sobre seguridad en redes para estudiantes universitarios de ciencias de la computación, ingeniería de la computación e ingeniería eléctrica. También sirve como libro básico de referencia y es adecuado para el aprendizaje autónomo.

ORGANIZACIÓN DEL LIBRO

El libro se divide en tres partes:

Primera parte. Criptografía: consiste en un estudio conciso de los algoritmos y protocolos criptográficos que subyacen a las aplicaciones de seguridad en la red, incluyendo el cifrado, las funciones *hash*, las firmas digitales y el intercambio de claves.

Segunda parte. Aplicaciones de seguridad en redes: cubre herramientas y aplicaciones importantes de seguridad en la red, incluyendo Kerberos, los certificados X.509v3, PGP, S/MIME, seguridad IP, SSL/TLS, SET y SNMPv3.

Tercera parte. Seguridad de los sistemas: observa los aspectos de seguridad en el ámbito de los sistemas, que incluyen las amenazas de intrusos y virus y sus contramedidas, y el uso de cortafuegos y sistemas confiables.

Además, el libro incluye un Glosario extenso, una lista de Acrónimos frecuentes y una Bibliografía. Cada capítulo ofrece problemas, preguntas de repaso, una lista de palabras que hacen referencia a conceptos fundamentales, así como lecturas y sitios web recomendados.

Al principio de cada una de las partes del libro se presenta un resumen más detallado de cada capítulo.

SERVICIOS DE INTERNET PARA DOCENTES Y ESTUDIANTES

Existe una página web para este libro que proporciona apoyo a los estudiantes y a los docentes. La página incluye enlaces a sitios relevantes, las transparencias en formato PDF (Adobe Acrobat) de las figuras y tablas que se muestran en el libro e información para registrarse en la lista de correo de Internet de este libro. La dirección de la página web es WilliamStallings.com/NetSec2e.html. Además, se ha creado una lista de correo para que los docentes que utilicen este libro puedan intercambiar, entre ellos y con el autor, información, sugerencias y preguntas. En WilliamStallings.com se dispone de una lista de erratas para incluir aquellos errores tipográficos o de cualquier otra índole que se detecten. También, la página *Computer Science Student Resource* (recursos para estudiantes de ciencias de la computación), en WilliamStallings.com/StudentSupport.html, proporciona documentos, información y enlaces útiles para estudiantes y profesionales.

PROYECTOS PARA LA ENSEÑANZA DE LA SEGURIDAD EN LA RED

Para muchos docentes, una parte importante de un curso sobre criptografía o seguridad lo constituye un proyecto o conjunto de proyectos mediante los cuales los estudiantes puedan realizar actividades experimentales para reforzar los conceptos del libro. Este libro proporciona un grado incomparable de apoyo para incluir un componente de proyectos en el curso. El manual del docente no sólo aporta una guía sobre cómo asignar y estructurar los proyectos, sino que también incluye una serie de propuestas de proyectos que cubren una amplia gama de temas que trata el texto:

- **Proyectos de investigación:** una serie de ejercicios de investigación que instruyen al alumno para investigar un tema en particular sobre Internet y escribir un informe.
- **Proyectos de programación:** una serie de proyectos de programación que cubren un amplio abanico de temas y que se pueden implementar en un lenguaje adecuado en una plataforma.
- **Ejercicios de lectura y redacción de informes:** una lista de artículos sobre el tema, uno para cada capítulo, que se pueden asignar para que los estudiantes los lean y después escriban un breve informe.

Véase Apéndice B.

RELACIÓN CON CRYPTOGRAPHY AND NETWORK SECURITY, TERCERA EDICIÓN

Este libro es un producto derivado de *Cryptography and Network Security*, Tercera Edición. Dicha obra proporciona un tratamiento sustancial de la criptografía, incluyendo, en 400 páginas, un análisis detallado de los algoritmos y el componente matemático significativos. *Fundamentos de Seguridad en Redes de Computadores: Aplicaciones y Estándares* ofrece una introducción concisa de estos temas en los Capítulos 2 y 3. También incluye todo el material restante del otro libro. Además, cubre la seguridad SNMP, que también se trata en *Cryptography and Network Security*. Así, este libro está diseñado para universitarios y para profesionales con un interés especial en la seguridad de la red, sin la necesidad o el deseo de ahondar en la teoría y los principios de la criptografía.

CAPÍTULO 1

Introducción

1.1. La arquitectura de seguridad OSI

1.2. Ataques a la seguridad

Ataques pasivos

Ataques activos

1.3. Servicios de seguridad

Autentificación

Control de acceso

Confidencialidad de los datos

Integridad de los datos

No repudio

Servicio de disponibilidad

1.4. Mecanismos de seguridad

1.5. Un modelo de seguridad en redes

1.6. Estándares de Internet y la Sociedad Internet

Las organizaciones de Internet y la publicación de los RFC

El proceso de estandarización

Categorías de estándares de Internet

Otros tipos de RFC

1.7. Estructura del libro

1.8. Bibliografía recomendada

1.9. Recursos web y de Internet

Sitios web para este libro

Otros sitios web

Grupos de noticias de USENET

La combinación de espacio, tiempo y fuerza, que deben considerarse como los elementos básicos de esta teoría de la defensa, hace de ésta una cuestión complicada. Por consiguiente, no es fácil encontrar un punto fijo de partida.

De la guerra, CARL VON CLAUSEWITZ

El arte de la guerra nos enseña a confiar no en la posibilidad de que el enemigo no venga, sino en nuestra propia disponibilidad para recibirla; no en la oportunidad de que no ataque, sino en el hecho de que hemos logrado que nuestra posición sea inexpugnable.

El arte de la guerra, SUN Tzu

Las necesidades de **seguridad de la información** en una organización han sufrido dos cambios fundamentales en las últimas décadas. Antes de la expansión del uso de equipamiento de procesamiento de datos, la seguridad de la información que una organización consideraba valiosa se proporcionaba, por un lado, por medios físicos, como el uso de armarios con cierre de seguridad para almacenar documentos confidenciales y, por otro, por medios administrativos, como los procedimientos de protección de datos del personal que se usan durante el proceso de contratación.

Con la introducción del computador, se hizo evidente la necesidad de disponer de herramientas automatizadas para la protección de archivos y otros tipos de información almacenada en el computador. Esto ocurre especialmente en el caso de sistemas compartidos como, por ejemplo, un sistema de tiempo compartido; y la necesidad se acentúa en sistemas a los que se puede acceder por medio de una red telefónica pública, una red de datos o Internet. El nombre genérico que se da al grupo de herramientas diseñadas para proteger los datos y evitar la intrusión de los *hackers* es el de **seguridad informática**.

El segundo cambio que afectó a la seguridad fue la introducción de sistemas distribuidos y el uso de redes y herramientas de comunicación para transportar datos entre el usuario de un terminal y el computador, y entre dos computadores. Las medidas de seguridad de la red son necesarias para proteger los datos durante la transmisión. De hecho, el término **seguridad de la red** es engañoso, en cierto modo, ya que prácticamente todas las empresas, las instituciones gubernamentales y académicas conectan sus equipos de procesamiento de datos formando un grupo de redes conectadas entre sí. Este grupo se considera con frecuencia como una internet¹, y se emplea el término **seguridad de la internet**.

No existen unas fronteras bien delimitadas entre estas dos formas de seguridad. Por ejemplo, uno de los tipos de ataque a los sistemas de información a los que más publicidad se ha dado es el virus informático. Un virus se puede introducir físicamente en un sistema por medio de un disquete o llegar por una internet. En cualquier caso, una vez que el virus reside en un sistema informático, se necesitan las herramientas internas de seguridad del computador para detectarlo, eliminarlo y restablecer el sistema.

¹ Usamos el término *Internet*, con «i» minúscula, para referirnos a cualquier grupo de redes conectadas entre sí. Una intranet corporativa es un ejemplo de internet. Internet con «I» mayúscula pudo ser una de las herramientas que usa una organización para construir su internet.

Este libro se centra en la seguridad de la internet, que consiste en las medidas para impedir, prevenir, detectar y corregir las violaciones de la seguridad que se producen durante la transmisión de la información. Esta es una afirmación muy general que abarca una gran cantidad de posibilidades. Para que el lector se haga una idea de las áreas que trata este libro, considere los siguientes ejemplos de violaciones de la seguridad:

1. El usuario A envía un archivo al usuario B. El archivo contiene información confidencial que debe protegerse (por ejemplo, registros de nóminas). El usuario C, que no está autorizado a leer el archivo, observa la transmisión y captura una copia del archivo durante dicha transmisión.
2. Un administrador de red, D, transmite un mensaje a un computador, E, que se encuentra bajo su gestión. El mensaje ordena al computador E que actualice un fichero de autorización para incluir las identidades de nuevos usuarios a los que se va a proporcionar el acceso a ese computador. El usuario F intercepta el mensaje, altera su contenido añadiendo o borrando entradas y, posteriormente, lo envía a E, que lo acepta como si procediera del administrador D y, por tanto, actualiza su archivo de autorización.
3. Más allá de interceptar un mensaje, el usuario F construye su propio mensaje con las entradas deseadas y lo transmite a E como si procediera del administrador D. Por consiguiente, el computador E acepta el mensaje y actualiza su fichero de autorización sin percatarse de la intrusión.
4. Un empleado es despedido sin previo aviso. El jefe de personal envía un mensaje a un sistema servidor para invalidar la cuenta del empleado. Cuando la invalidación se ha llevado a cabo, el servidor ha de notificar la confirmación de la acción al archivo del empleado. El empleado intercepta el mensaje y lo retrasa el tiempo suficiente para realizar un último acceso al servidor y recuperar, así, información confidencial. A continuación, se envía el mensaje, se lleva a cabo la acción y se notifica la confirmación. La acción del empleado puede pasar inadvertida durante un período de tiempo considerable.
5. Un cliente envía un mensaje a un corredor de bolsa con instrucciones para realizar diferentes transacciones. Más tarde, las inversiones pierden valor y el cliente niega haber enviado dicho mensaje.

Aunque, de ningún modo, esta lista abarca todos los tipos posibles de violaciones de la seguridad, ilustra una serie de aspectos que afectan a la seguridad de la red.

La seguridad de la red se presenta como un tema fascinante a la vez que complejo. A continuación se exponen algunas de las razones:

1. La seguridad relativa a las comunicaciones y a las redes no es tan simple como podría parecer a los principiantes en este tema. Los requisitos parecen claros; de hecho, a la mayoría de los requisitos fundamentales que deben cumplir los servicios de seguridad se les puede asignar etiquetas de una sola palabra que se explican por sí mismas: confidencialidad, autenticación, no repudio, integridad. Pero los mecanismos empleados para satisfacer estos requisitos pueden ser muy complejos, y comprenderlos puede implicar un razonamiento un tanto sutil.
2. En el desarrollo de un mecanismo particular de seguridad o algoritmo, siempre se deben tener en cuenta los posibles ataques a esas características de seguridad.

En muchos casos, los ataques con éxito están diseñados analizando el problema de una forma totalmente diferente, explotando, por lo tanto, una debilidad inadvertida del mecanismo.

- 3 Como consecuencia del punto 2, los procedimientos empleados para proporcionar servicios particulares no suelen ser intuitivos; es decir, a partir de la afirmación de un requisito particular no es obvio que sean necesarias medidas tan elaboradas. Las medidas empleadas sólo cobran sentido cuando se han considerado las diferentes contramedidas.
- 4 Después de diseñar distintos mecanismos de seguridad, es necesario decidir dónde usarlos, tanto en lo que respecta a la ubicación física (por ejemplo, en qué puntos de una red se necesitan determinados mecanismos de seguridad) como a la ubicación lógica [en qué capa o capas de una arquitectura como la TCP/IP (Transmission Control Protocol / Internet Protocol) deberían estar localizados los mecanismos].
- 5 Los mecanismos de seguridad suelen implicar más de un algoritmo o protocolo. Con frecuencia también requieren que los participantes posean alguna información secreta (que podría ser una clave de cifrado), lo que da lugar a cuestiones sobre la creación, distribución y protección de esa información secreta. También hay una dependencia de los protocolos de comunicación cuyo comportamiento puede complicar la tarea de desarrollar mecanismos de seguridad. Por ejemplo, si el funcionamiento adecuado del mecanismo de seguridad requiere establecer unos límites de tiempo para la transmisión de un mensaje desde el emisor hasta el receptor, cualquier protocolo o red que introduzca retrasos variables e impredecibles podría hacer que estos límites temporales carecieran de sentido.

Por lo tanto, son muchas las cuestiones que han de tenerse en cuenta. Este Capítulo proporciona una visión general del tema, que se irá desarrollando a lo largo del libro. Empezaremos con una discusión general sobre los servicios y mecanismos de seguridad en las redes y los tipos de ataques para los que están diseñados. A continuación, desarrollaremos un modelo general en el que se podrán observar los distintos servicios y mecanismos de seguridad.

1.1 LA ARQUITECTURA DE SEGURIDAD OSI

Para analizar de forma efectiva las necesidades de seguridad de una organización y evaluar y elegir distintos productos y políticas de seguridad, el responsable de la seguridad necesita una forma sistemática de definir los requisitos de seguridad y caracterizar los enfoques para satisfacer dichos requisitos. Esto es bastante difícil en un entorno centralizado de procesamiento de datos, y con el uso de redes de área local y de área ancha, los problemas se agravan.

La recomendación X.800 de la ITU-T², *Arquitectura de seguridad OSI*, define este enfoque sistemático. La arquitectura de seguridad OSI es útil a los administradores de

² El Sector de Estandarización de Telecomunicaciones (ITU-T) de la Unión Internacional de Telecomunicaciones (ITU) es una agencia financiada por las Naciones Unidas que desarrolla estándares, denominados Recomendaciones, relativos a las telecomunicaciones y a la interconexión de sistemas abiertos (OSI).

red para organizar la tarea de proporcionar seguridad. Además, debido a que esta arquitectura fue desarrollada como un estándar internacional, los vendedores han desarrollado características de seguridad para sus productos y servicios conforme a esta definición estructurada de servicios y mecanismos.

Para nuestros propósitos, la arquitectura de seguridad OSI proporciona una visión general útil, aunque abstracta, de muchos de los conceptos que se tratan en este libro. La arquitectura de seguridad OSI se centra en los ataques a la seguridad, los mecanismos y los servicios, que se definen brevemente a continuación:

Tabla 1.1 Amenazas y ataques (RFC 2828)

Amenaza

Una posibilidad de violación de la seguridad, que existe cuando se da una circunstancia, capacidad, acción o evento que pudiera romper la seguridad y causar perjuicio. Es decir, una amenaza es un peligro posible que podría explotar una vulnerabilidad.

Ataque

Un asalto a la seguridad del sistema derivado de una amenaza inteligente; es decir, un acto inteligente y deliberado (especialmente en el sentido de método o técnica) para eludir los servicios de seguridad y violar la política de seguridad de un sistema.

- **Ataque a la seguridad:** cualquier acción que comprometa la seguridad de la información de una organización.
- **Mecanismo de seguridad:** un mecanismo diseñado para detectar un ataque a la seguridad, prevenirlo o restablecerse de él.
- **Servicio de seguridad:** un servicio que mejora la seguridad de los sistemas de procesamiento de datos y la transferencia de información de una organización. Los servicios están diseñados para contrarrestar los ataques a la seguridad, y hacen uso de uno o más mecanismos para proporcionar el servicio.

En la literatura al respecto, los términos *amenaza* y *ataque* se usan frecuentemente para referirse más o menos a lo mismo. La Tabla 1.1 proporciona las definiciones extraídas de RFC 2828, *Internet Security Glossary*.

1.2 ATAQUES A LA SEGURIDAD

Una forma útil de clasificar los ataques a la seguridad, empleada en la recomendación X.800 y RFC 2828, es la distinción entre *ataques pasivos* y *ataques activos*. Un ataque pasivo intenta conocer o hacer uso de información del sistema, pero no afecta a los recursos del mismo. Un ataque activo, por el contrario, intenta alterar los recursos del sistema o afectar a su funcionamiento.

ATAQUES PASIVOS

Los ataques pasivos se dan en forma de escucha o de observación no autorizadas de las transmisiones. El objetivo del oponente es obtener información que se esté transmitiendo. Dos tipos de ataques pasivos son la obtención de contenidos de mensajes y el análisis del tráfico.

La **obtención de contenidos de mensajes** se entiende fácilmente. (Figura 1.1a). Una conversación telefónica, un mensaje por correo electrónico y un fichero enviado pueden contener información confidencial. Queremos evitar que un oponente conozca los contenidos de estas transmisiones.

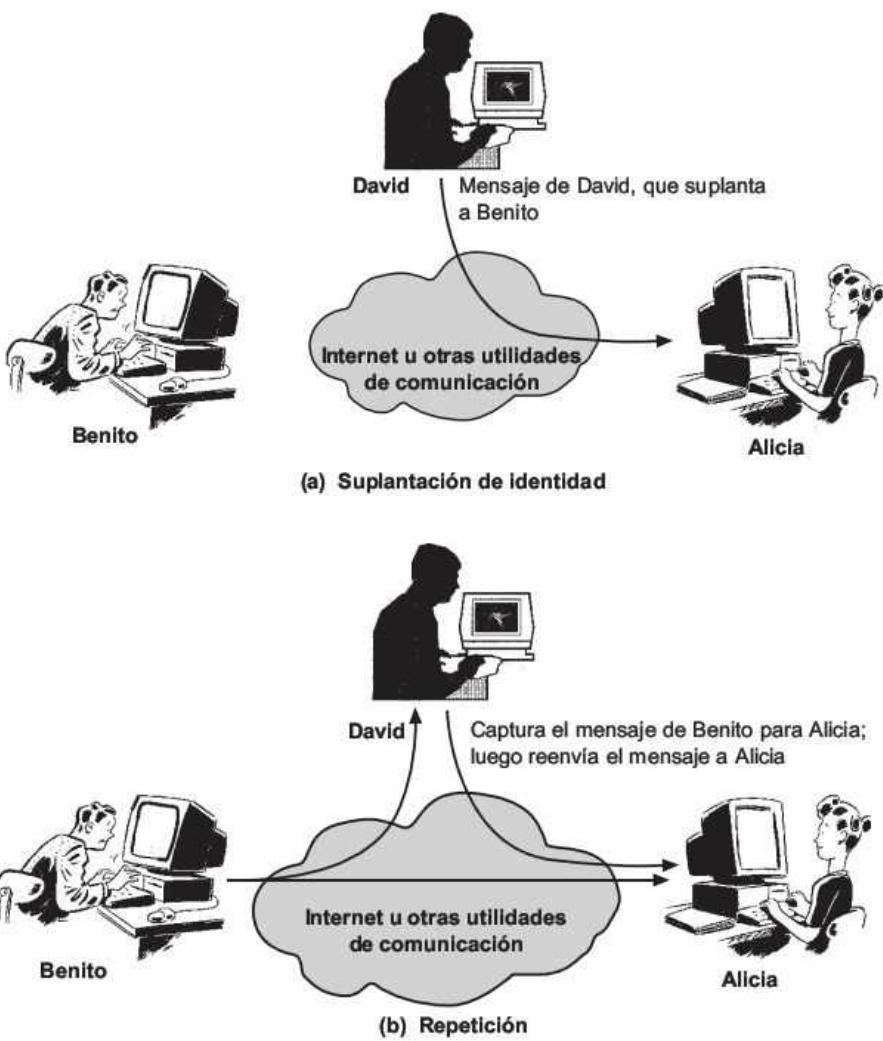


(a) Obtención del contenido del mensaje



(b) Análisis del tráfico

Figura 1.1 Ataques pasivos

**Figura 1.2** Ataques activos

Un segundo tipo de ataque pasivo, el **análisis de tráfico**, es más sutil (Figura 1.1b). Supongamos que hemos enmascarado los contenidos de los mensajes u otro tráfico de información de forma que el oponente, incluso habiendo capturado el mensaje, no pueda extraer la información que contiene. La técnica común para enmascarar los contenidos es el cifrado. Incluso si tuviésemos protección mediante cifrado, un oponente podría observar el patrón de los mensajes, determinar la localización y la identidad de los servidores que se comunican y descubrir la frecuencia y la longitud de los mensajes que se están intercambiando. Esta información puede ser útil para averiguar la naturaleza de la comunicación que está teniendo lugar.

Los ataques pasivos son muy difíciles de detectar ya que no implican alteraciones en los datos. Normalmente, el mensaje se envía y se recibe de una forma aparentemente

normal y ni el emisor ni el receptor son conscientes de que una tercera persona ha leído los mensajes o ha observado el **patrón del tráfico**. Sin embargo, es posible evitar el éxito de estos ataques, normalmente mediante el uso del cifrado. Así, al tratar con los ataques pasivos, el énfasis se pone más en la prevención que en la detección.



Figura 1.2 Ataques activos (continuación)

ATAQUES ACTIVOS

Los ataques activos implican alguna modificación del flujo de datos o la creación de un flujo falso y se pueden dividir en cuatro categorías: suplantación de identidad, repetición, modificación de mensajes e interrupción de servicio.

Una **suplantación** se produce cuando una entidad finge ser otra (Figura 1.2a). Un ataque de este tipo incluye habitualmente una de las otras formas de ataque activo. Por

ejemplo, las secuencias de autenticación pueden ser capturadas y repetidas después de que una secuencia válida de autenticación haya tenido lugar, permitiendo así, que una entidad autorizada con pocos privilegios obtenga privilegios extra haciéndose pasar por la entidad que realmente los posee.

La repetición implica la captura pasiva de una unidad de datos y su retransmisión posterior para producir un efecto no autorizado (Figura 1.2b).

La **modificación de mensajes** significa que una parte de un mensaje original es alterada, o que los mensajes se han retrasado o reordenado, para producir un efecto no autorizado (Figura 1.2c). Por ejemplo, el mensaje «Permitir a Carlos Pérez que lea las cuentas de archivos confidenciales» se modifica para convertirlo en «Permitir a Marcos Fernández que lea las cuentas de archivos confidenciales».

La **interrupción de servicio** impide el uso o la gestión normal de las utilidades de comunicación (Figura 1.2d). Este ataque podría tener un objetivo específico; por ejemplo, una entidad podría suprimir todos los mensajes dirigidos a un destino en particular (por ejemplo, el servicio de auditoría de la seguridad). Otra forma de este tipo de ataque es la interrupción de una red completa, ya sea inhabilitándola o sobrecargándola con mensajes para reducir su rendimiento.

Los ataques activos presentan las características opuestas a los pasivos. Aunque los ataques pasivos son difíciles de detectar, existen medidas para prevenir su éxito. Sin embargo, es bastante difícil prevenir por completo los ataques activos, debido a que se requeriría la protección física de todas las herramientas de comunicación y las rutas en todo momento. Por el contrario, el objetivo es el de detectarlos y recuperarse de cualquier irrupción o retraso que originen. Como la detección tiene un efecto disuasivo, también podría contribuir a la prevención.

1.3 SERVICIOS DE SEGURIDAD

La recomendación X.800 define un servicio de seguridad como un servicio proporcionado por una capa de protocolo de sistemas abiertos de comunicación, que garantiza la seguridad adecuada de los sistemas o de las transferencias de datos. Quizás es más clara la definición recogida en RFC 2828: un servicio de procesamiento o de comunicación proporcionado por un sistema para dar un tipo especial de protección a los recursos del sistema; los servicios de seguridad implementan políticas de seguridad y son implementados, a su vez, por mecanismos de seguridad.

En X.800 estos servicios quedan divididos en cinco categorías y 14 servicios específicos (Tabla 1.2). Observemos a continuación cada una de las categorías³.

³ No existe un acuerdo universal sobre la gran cantidad de términos que se emplean en la literatura sobre seguridad. Por ejemplo, el término *Integridad* se usa a veces para referirse a todos los aspectos de la seguridad de la información. El término *autenticación* se usa con frecuencia para hacer alusión tanto a la verificación de la identidad como a las diferentes funciones que aparecen referidas a integridad en este Capítulo. Nuestro uso aquí está de acuerdo con las normas X.800 y RFC 2828.

Tabla 1.2 Servicios de seguridad (X.800)

AUTENTIFICACIÓN	INTEGRIDAD DE LOS DATOS
<p>Autenticación de las entidades origen/destino La seguridad de que la entidad que se comunica es quien dice ser.</p> <p>Autenticación del origen de los datos Empleada conjuntamente con una conexión lógica para aportar confianza sobre la identidad de las entidades conectadas.</p>	<p>Integridad de la conexión con recuperación La seguridad de que los datos recibidos son exactamente como los envió una entidad autorizada (no contienen modificación, inserción, omisión, ni repetición).</p>
<p>Control de acceso En transferencias no orientadas a la conexión, garantiza que la fuente de los datos recibidos es la que dice ser.</p>	<p>Integridad de la conexión sin recuperación Proporciona la integridad de los datos de todos los usuarios en una conexión y detecta cualquier modificación, inserción, omisión o repetición de cualquier dato de una secuencia completa de datos, con intento de recuperación.</p>
<p>Confidencialidad de los datos La protección de los datos contra la revelación no autorizada.</p> <p>Confidencialidad de la conexión La protección de los datos de todos los usuarios en una conexión.</p> <p>Confidencialidad no orientada a la conexión La protección de los datos de todos los usuarios en un único bloque de datos.</p> <p>Confidencialidad de campos seleccionados La confidencialidad de campos seleccionados en los datos del usuario en una conexión o en un único bloque de datos.</p> <p>Confidencialidad del flujo de tráfico La protección de la información que podría extraerse a partir de la observación del flujo del tráfico.</p>	<p>Integridad de la conexión de campos seleccionados Igual que el anterior, pero proporciona sólo detección sin recuperación.</p> <p>Integridad no orientada a la conexión Proporciona integridad de un bloque de datos sin conexión y puede detectar la alteración de datos. Además, puede proporcionar una forma limitada de detección de repetición.</p> <p>Integridad no orientada a la conexión de campos seleccionados Proporciona la integridad de los campos seleccionados en un bloque de datos sin conexión; determina si los campos seleccionados han sido modificados, insertados, suprimidos o repetidos.</p>
	<p>No repudio Proporciona protección contra la interrupción, por parte de una de las entidades implicadas en la comunicación, de haber participado en toda o parte de la comunicación.</p> <p>No repudio, origen Prueba que el mensaje fue enviado por la parte especificada.</p> <p>No repudio, destino Prueba que el mensaje fue recibido por la parte especificada.</p>

AUTENTIFICACIÓN

El servicio de autentificación se encarga de garantizar la autenticidad de la comunicación. En el caso de un único mensaje como, por ejemplo, una señal de aviso o de alarma, la función del servicio de autentificación es asegurar al receptor que el mensaje pertenece a la fuente de la que dice proceder. En el caso de una interacción continua como la conexión de un terminal a un *host*, intervienen dos aspectos. En primer lugar, en el inicio de la conexión, el servicio asegura que las dos entidades son auténticas; es decir, cada entidad es la que dice ser. En segundo lugar, el servicio debe asegurar que la conexión no está intervenida de forma que una tercera persona pueda suplantar a una de las dos partes auténticas con la finalidad de realizar una transmisión o una recepción no autorizada.

En el estándar X.800 se definen dos tipos particulares de autentificación:

- **Autentificación de entidades origen/destino:** proporciona la confirmación de la identidad de una entidad de una asociación. Se proporciona en el establecimiento de una conexión o a veces durante la fase de transmisión de datos de dicha conexión. Intenta asegurar que una entidad no está realizando una suplantación o una repetición no autorizada de una conexión anterior.
- **Autentificación del origen de los datos:** corrobora la fuente de una unidad de datos. No aporta protección contra la repetición o la alteración de unidades de datos. Este tipo de servicio admite aplicaciones como el correo electrónico, donde no hay interacciones previas entre las entidades que se comunican.

CONTROL DE ACCESO

En el contexto de la seguridad de redes, el control de acceso es la capacidad de limitar y controlar el acceso a sistemas *host* y aplicaciones por medio de enlaces de comunicaciones. Para conseguirlo, cualquier entidad que intente acceder debe antes ser identificada o autenticada, de forma que los derechos de acceso puedan adaptarse de manera individual.

CONFIDENCIALIDAD DE LOS DATOS

La confidencialidad es la protección de los datos transmitidos por medio de ataques pasivos. En función del contenido de una transmisión de datos, existen diferentes niveles de protección. El servicio más amplio protege los datos de los usuarios que se han transmitido por conexión TCP. Se pueden distinguir formas más específicas de este servicio, incluyendo la protección de un solo mensaje o incluso de determinados campos de un mensaje. Estos refinamientos son menos útiles que el enfoque amplio y su implementación puede incluso ser más compleja y costosa.

El otro aspecto de la confidencialidad es la protección del flujo del tráfico frente al análisis del tráfico. Para ello el atacante no debería poder ver la fuente, el destino, la frecuencia, la longitud ni otras características del tráfico en una comunicación.

INTEGRIDAD DE LOS DATOS

Al igual que ocurre con la confidencialidad, la integridad se puede aplicar a una serie de mensajes, a un solo mensaje o a campos seleccionados de un mensaje. Nuevamente, el enfoque más útil y claro es la protección del flujo completo.

Un servicio de integridad orientado a la conexión que funcione sobre un flujo de mensajes garantiza que los mensajes se reciben tal y como son enviados, sin duplicación, inserción, modificación, reordenación ni repeticiones. La destrucción de datos también queda cubierta con este servicio. Así, el servicio de integridad orientado a la conexión trata tanto la modificación del flujo de mensajes como la interrupción del servicio. Por otra parte, un servicio de integridad sin conexión, que trata únicamente mensajes individuales sin tener en cuenta contextos mayores, sólo proporciona, generalmente, protección contra la modificación del mensaje.

Podemos distinguir entre el servicio con y sin recuperación. Debido a que el servicio de integridad tiene que ver con ataques activos, nos interesa más la detección que la prevención. Si se detecta una violación de la integridad, el servicio podría simplemente informar de esta violación, y será necesaria la intervención humana o de algún otro *software* para restablecerse de la violación. Por otra parte, existen mecanismos para la recuperación de la pérdida de integridad de los datos, como estudiaremos más tarde. La incorporación de mecanismos de recuperación automatizada se presenta, en general, como la alternativa más atrayente.

NO REPUDIO

El no repudio evita que el emisor o el receptor nieguen la transmisión de un mensaje. Así, cuando se envía un mensaje, el receptor puede comprobar que, efectivamente, el supuesto emisor envió el mensaje. De forma similar, cuando se recibe un mensaje, el emisor puede verificar que, de hecho, el supuesto receptor recibió el mensaje.

SERVICIO DE DISPONIBILIDAD

Tanto X.800 como RFC 2828 definen la disponibilidad como la propiedad que tiene un sistema o recurso de un sistema de estar accesible y utilizable a petición de una entidad autorizada, según las especificaciones de rendimiento para el sistema (un sistema está disponible si proporciona servicios de acuerdo con el diseño del sistema en el momento en que los usuarios lo soliciten). Una variedad de ataques puede dar como resultado la pérdida o reducción de la disponibilidad. Algunos de estos ataques son susceptibles a contramedidas automatizadas, como la autentificación o el cifrado, mientras que otras requieren algún tipo de acción física para prevenir o recuperarse de la pérdida de disponibilidad de elementos de un sistema distribuido.

La norma X.800 trata la disponibilidad como una propiedad asociada a diferentes servicios de seguridad. Sin embargo, tiene sentido solicitar un servicio concreto de disponibilidad. Un servicio de disponibilidad es aquel que protege un sistema para asegurar su disponibilidad y trata los problemas de seguridad que surgen a raíz de ataques de interrupción de servicio. Depende de la gestión y control adecuados de los recursos del sistema y, por lo tanto, del servicio de control de acceso y otros servicios de seguridad.

La Tabla 1.3 muestra la relación existente entre los servicios de seguridad y los ataques.

Tabla 1.3 Relación entre servicios de seguridad y ataques

Ataque						
Servicio	Obtención del contenido del mensaje	Ánalisis de tráfico	Suplantación	Repetición	Modificación de mensajes	Interrupción de servicio
Autentificación de las entidades origen/destino			Y			
Autentificación del origen de los datos			Y			
Control de acceso			Y			
Confidencialidad	Y					
Confidencialidad del flujo de tráfico		Y				
Integridad de los datos				Y	Y	
No repudio						
Disponibilidad						Y

1.4 MECANISMOS DE SEGURIDAD

La Tabla 1.4 presenta los mecanismos de seguridad definidos en X.800. Como se puede observar, los mecanismos se dividen en aquellos que se implementan en una capa específica de un protocolo y aquellos que no son específicos de ninguna capa de protocolo o servicio de seguridad en particular. Estos mecanismos se tratarán en las secciones correspondientes de este libro y por ello no se elaboran ahora, excepto para adelantar la definición de cifrado. X.800 distingue entre mecanismos de cifrado reversible y mecanismos de cifrado irreversible. El primero es un algoritmo de cifrado que permite cifrar los datos y, posteriormente, descifrarlos. Por otro lado, los mecanismos de cifrado irreversible incluyen algoritmos *hash* y códigos de autenticación de mensajes, que se emplean en firmas digitales y en aplicaciones de autenticación de mensajes.

La Tabla 1.5, basada en X.800, indica la relación que se da entre los servicios de seguridad y los mecanismos de seguridad.

Tabla 1.4 Mecanismos de seguridad (X.800)

MECANISMOS ESPECÍFICOS DE SEGURIDAD	
Pueden ser incorporados en la capa de protocolo adecuada para proporcionar algunos de los servicios de seguridad OSI	te los cambios de enrutamiento, especialmente cuando se sospecha de una brecha en la seguridad.
Cifrado El uso de algoritmos matemáticos para transformar datos en una forma inteligible. La transformación y la posterior recuperación de los datos depende de un algoritmo y cero o más claves de cifrado.	Notarización El uso de una tercera parte confiable para asegurar determinadas propiedades de un intercambio de datos.
Firma digital Datos añadidos a, o una transformación criptográfica de, una unidad de datos que permite al receptor verificar la fuente y la integridad de la unidad de datos y protegerla de la falsificación (por parte del receptor).	MECANISMOS GENERALES DE SEGURIDAD Mecanismos que no son específicos de ninguna capa de protocolo o sistema de seguridad OSI en particular.
Control de acceso Una serie de mecanismos que refuerzan los derechos de acceso a los recursos.	Funcionalidad fiable La que se considera correcta con respecto a algunos criterios (por ejemplo, los establecidos por una política de seguridad).
Integridad de los datos Una serie de mecanismos empleados para verificar la integridad de una unidad de datos o del flujo de unidades de datos.	Etiquetas de seguridad La marca asociada a un recurso (que podría ser una unidad de datos) que designa los atributos de seguridad de ese recurso.
Intercambio de autentificación Un mecanismo diseñado para comprobar la identidad de una entidad por medio del intercambio de información.	Detección de acciones Detección de acciones relacionadas con la seguridad.
Relleno del tráfico La inserción de bits en espacios en un flujo de datos para frustrar los intentos de análisis de tráfico.	Informe para la auditoría de seguridad Recopilación de datos para facilitar una auditoría de seguridad, que consiste en una revisión y un examen independientes de los informes y actividades del sistema.
Control de enrutamiento Permite la selección de rutas físicamente seguras para determinados datos y permiti-	Recuperación de la seguridad Maneja las peticiones de los mecanismos (como funciones de gestión de acciones) y lleva a cabo acciones de recuperación.

1.5 UN MODELO DE SEGURIDAD EN REDES

La Figura 1.3 constituye un modelo que presenta, en términos generales, gran parte de los aspectos que discutiremos a continuación. Un mensaje ha de ser transmitido de una parte a otra mediante algún tipo de internet. Las dos partes, que son los interlocutores en esta transacción, deben cooperar para que el intercambio tenga lugar. Se establece un canal de información definiendo una ruta a través de la internet que vaya de la fuente al

Tabla 1.5 Relación entre servicios y mecanismos de seguridad

Servicio	Cifrado	Firma digital	Control de acceso	Integridad de los datos	Mecanismo			
					Intercambio de autenticación	Relleno del tráfico	Control del enrutamiento	Notarización
Autentificación de entidades origen/destino	Y	Y			Y			
Autentificación del origen de los datos	Y	Y						
Control de acceso			Y				Y	
Confidencialidad	Y						Y	
Confidencialidad del flujo del tráfico	Y					Y	Y	
Integridad de los datos	Y	Y		Y				
No repudio		Y		Y				
Disponibilidad				Y	Y			Y

destino y mediante el uso cooperativo de los protocolos de comunicación (TCP/IP) por parte de los dos interlocutores.

Los aspectos de seguridad entran en juego cuando se necesita o se quiere proteger la transmisión de información de un oponente que pudiera presentar una amenaza a la confidencialidad, a la autenticidad, etc. Todas las técnicas para proporcionar seguridad tienen dos componentes:

- Una transformación relacionada con la seguridad de la información que se va a enviar. Ejemplos de ello los tenemos en el cifrado del mensaje, que lo desordena para que resulte ilegible al oponente, y la aplicación de un código basado en el contenido del mensaje, que puede usarse para verificar la identidad del emisor.
- Alguna información secreta compartida por los interlocutores y desconocida por el oponente. El ejemplo lo hallamos en una clave de cifrado usada en conjunción con la transformación para desordenar el mensaje antes de la transmisión y reordenarlo en el momento de la recepción⁴.

⁴ El Capítulo 3 trata una forma de cifrado, conocida como cifrado de clave pública, en la que sólo uno de los dos interlocutores necesita conocer la información secreta.

Para lograr una transmisión segura, puede ser necesaria una tercera parte confiable, que, por ejemplo, sea la responsable de distribuir la información secreta a los dos interlocutores y la guarde de cualquier oponente. También puede ser necesaria para arbitrar disputas entre los interlocutores en lo relativo a la autenticidad de la transmisión de un mensaje.

Este modelo general muestra que hay cuatro tareas básicas en el diseño de un servicio de seguridad particular:

1. Diseñar un algoritmo para llevar a cabo la transformación relacionada con la seguridad. El algoritmo debe estar diseñado de forma que un oponente no pueda frustrar su finalidad.
2. Generar la información secreta que deba ser usada con el algoritmo.
3. Desarrollar métodos para distribuir y compartir la información secreta.
4. Especificar un protocolo para los dos interlocutores que hagan uso del algoritmo de seguridad y la información secreta, para obtener un servicio concreto de seguridad.

La Segunda Parte de este libro se centra en los tipos de mecanismos y servicios de seguridad que encajan en el modelo que muestra la Figura 1.3. Sin embargo, existen otras situaciones relacionadas con la seguridad que son de interés y que no se corresponden claramente con este modelo, pero que se tienen en consideración en este libro. La Figura 1.4 ofrece un modelo general de estas situaciones, que refleja la preocupación por proteger un sistema de información del acceso no deseado. La mayoría de los lectores están familiarizados con los problemas ocasionados por la existencia de *hackers*, que tratan de penetrar sistemas a los que se puede acceder por una red. El *hacker* puede ser alguien que, sin la intención de hacer daño, obtiene satisfacción simplemente rompiendo y entrando en sistemas informáticos. El intruso también puede ser un empleado contrariado que quiere hacer daño, o un criminal que intenta explotar los sistemas computacionales para obtener beneficios financieros (obtención de números de tarjetas de crédito o realización de transferencias ilegales de dinero).

Otro tipo de acceso no deseado consiste en introducir en un sistema computacional *software* que explote debilidades en el sistema y que pueda afectar a programas de aplicaciones, como editores y compiladores. Los programas pueden presentar dos tipos de amenazas:

- **Amenazas al acceso a la información:** captura o alteración de datos por parte de usuarios que no deberían tener acceso a dichos datos.
- **Amenazas al servicio:** explotación de fallos del servicio en los computadores para impedir el uso por parte de los usuarios legítimos.

Los virus y gusanos son dos ejemplos de ataques mediante *software*. Tales ataques pueden introducirse en un sistema por medio de un disco que contenga el programa no deseado oculto en *software* útil. También pueden ser introducidos en un sistema a través de una red; este último mecanismo es de más interés en la seguridad de redes.

Los mecanismos de seguridad necesarios para enfrentarse a accesos no deseados se dividen en dos grandes categorías (véase la Figura 1.4). La primera categoría puede denominarse función de vigilancia. Incluye los procedimientos de conexión mediante clave, diseñados para negar acceso a usuarios no autorizados, y los *software* de ocultación, diseñados para detectar y rechazar gusanos, virus y ataques similares. Una vez que

un usuario o *software* no deseado accede, la segunda línea de la defensa consiste en una serie de controles internos que monitorizan la actividad y analizan la información almacenada con el fin de detectar la presencia de intrusos. Estos aspectos se tratan más exhaustivamente en la Tercera Parte del libro.

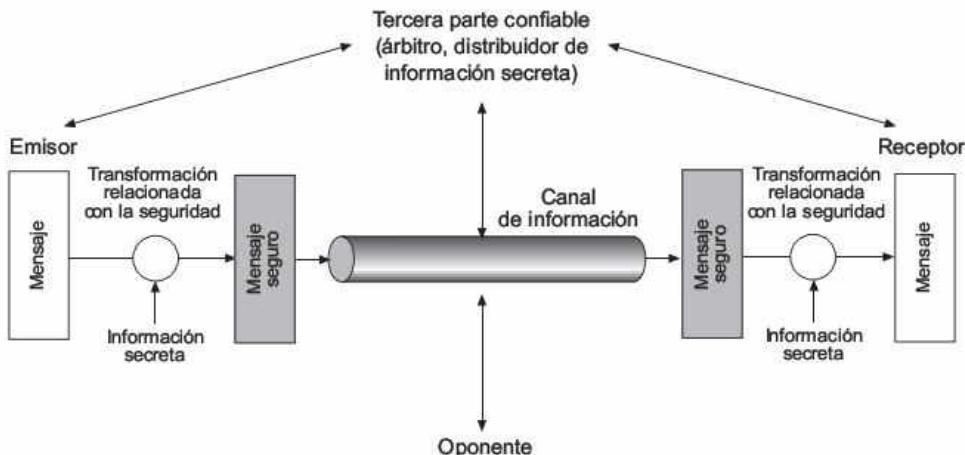


Figura 1.3 Modelo para la seguridad de redes

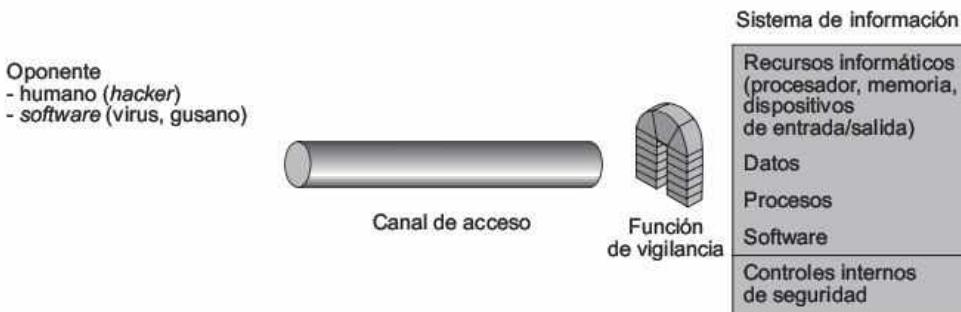


Figura 1.4 Modelo para la seguridad en el acceso a redes

1.6 ESTÁNDARES DE INTERNET Y LA SOCIEDAD INTERNET

Muchos protocolos que conforman la *suite* de protocolos TCP/IP han sido estandarizados o están en proceso de estandarización. Por acuerdo universal, una organización conocida como la Sociedad Internet es responsable del desarrollo y la publicación de los estándares. La Sociedad Internet es una organización de miembros profesionales que

controla una serie de comisiones y grupos de trabajo implicados en el desarrollo y la estandarización de Internet.

Esta sección ofrece una breve descripción de la forma en que se desarrollan los estándares para la *suite* de protocolos TCP/IP.

Tabla 1.6 Áreas de la IETF

Área IETF	Tema	Grupos de trabajo (ejemplos)
General	Proceso y procedimientos de IETF.	Políticas de trabajo. Proceso para la organización de los estándares de Internet.
Aplicaciones	Aplicaciones de Internet.	Protocolos web (HTTP). Integración EDI-Internet. LDAP.
Internet	Infraestructura de Internet.	IPv6. Extensiones PPP.
Operaciones y gestión	Estándares y definiciones para las operaciones de las redes.	SNMPv3. Monitorización remota de redes de comunicación.
Enrutamiento	Protocolos y administración para el enrutamiento de la información.	Enrutamiento Multicast. OSPF. Enrutamiento QoS.
Seguridad	Protocolos y tecnologías de seguridad.	Kerberos. IPSec. X.509. S/MIME. TLS.
Transporte	Protocolos de la capa de transporte.	Servicios diferenciados. Telefonía IP. NFS. RSVP.
Servicios de usuarios	Métodos para mejorar la calidad de la información disponible a los usuarios de Internet.	Uso responsable de Internet. Servicios de usuarios. Documentos FYI.

LAS ORGANIZACIONES DE INTERNET Y LA PUBLICACIÓN DE LOS RFC

La Sociedad Internet es el comité coordinador para el diseño, la ingeniería y la gestión de Internet. Las áreas que cubre incluyen el funcionamiento de Internet y la estandari-

zación de protocolos que se usan en sistemas terminales en Internet para operar entre sí. Existen tres organizaciones en el ámbito de la Sociedad Internet que son responsables del trabajo real del desarrollo y la publicación de los estándares:

- **Internet Architecture Board (IAB):** es responsable de definir la arquitectura general de Internet, proporcionando orientaciones a la IETF.
- **Internet Engineering Task Force (IETF):** se encarga del desarrollo y la ingeniería de protocolos en Internet.
- **Internet Engineering Steering Group (IESG):** responsable de la gestión técnica de las actividades de la IETF y del proceso de estándares de Internet.

Los grupos de trabajo de IETF llevan a cabo el desarrollo real de los nuevos estándares y protocolos para Internet. Ser miembro de un grupo de trabajo es una acción voluntaria y cualquier parte interesada puede participar. Durante el desarrollo de una especificación, un grupo de trabajo realizará un borrador del documento disponible como Borrador de Internet, que se coloca en el directorio en línea «Borradores de Internet» de la IETF. El documento puede permanecer como un Borrador de Internet durante seis meses, y las partes interesadas pueden revisarlo y hacer observaciones sobre él. Durante ese período de tiempo, el IESG puede aprobar la publicación del borrador como RFC (*Request for Comment*). Si el borrador no ha progresado hasta llegar al estado de RFC durante esos seis meses, se retira del directorio. El grupo de trabajo puede, a continuación, publicar una versión revisada del borrador.

La IETF es responsable de la publicación de los RFC con la aprobación del IESG. Los RFC son las notas de trabajo de la comunidad de investigación y desarrollo de Internet. Estos documentos pueden versar sobre cualquier asunto relacionado con las comunicaciones computacionales y puede constituir cualquier tipo de documento, desde un informe de una reunión a las especificaciones de un estándar.

El trabajo de la IETF se divide en ocho grandes áreas. La Tabla 1.6 refleja estas áreas y sus correspondientes objetivos.

EL PROCESO DE ESTANDARIZACIÓN

El IESG toma la decisión por la cual los RFC se convierten en estándares de Internet, con la recomendación de la IETF. Para llegar a ser un estándar, una especificación debe cumplir con los siguientes requisitos:

- Ser estable y comprensible.
- Ser técnicamente competente.
- Tener implementaciones múltiples, independientes e interoperativas con una experiencia operativa sustancial.
- Tener apoyo público significativo.
- Ser reconocidamente útil en algunas o en todas las partes de Internet.

La diferencia clave entre estos criterios y los que se usan para los estándares internacionales de la ITU se halla en la importancia que se otorga aquí a la experiencia operativa.

La parte izquierda de la Figura 1.5 presenta los diferentes pasos (*standards track*), que sigue una especificación hasta convertirse en estándar; este proceso está definido en la norma RFC 2026. A medida que se avanza por dichos pasos aumentan el escrutinio y las pruebas. En cada paso, la IETF debe solicitar recomendación para el avance del protocolo, y el IESG debe ratificarlo. El proceso comienza cuando el IESG aprueba la publicación de un borrador de Internet como RFC con el estatus de Estándar Propuesto.

Los recuadros blancos de la Figura 1.5 representan estados temporales, que deberían estar ocupados durante el tiempo mínimo. Sin embargo, un documento debe permanecer como Estándar Propuesto durante al menos seis meses, y un Estándar de Borrador, durante un mínimo de cuatro meses para permitir su revisión y discusión. Los recuadros grises representan los estados de larga duración que pueden estar ocupados durante años.

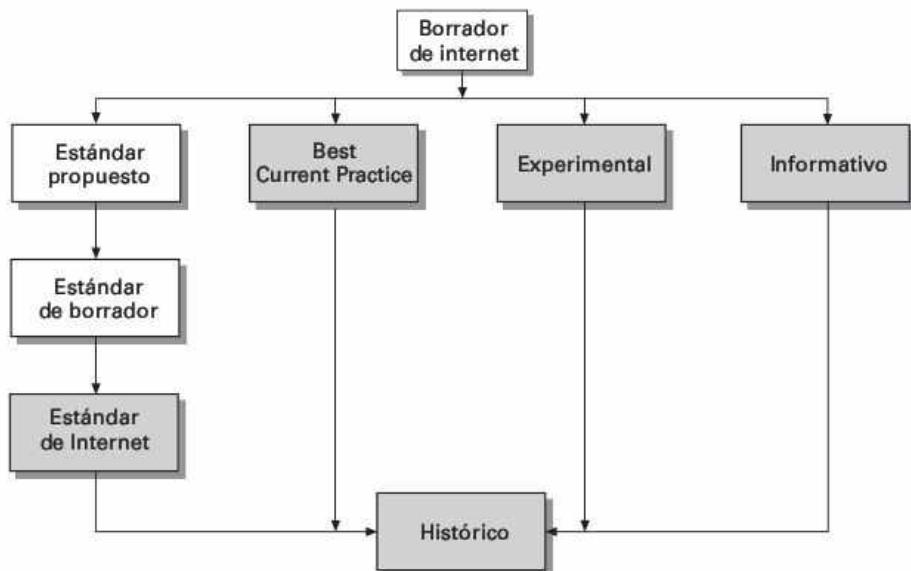


Figura 1.5 Proceso para la publicación de los RFC de Internet

Para que una especificación adquiera el estatus de Estándar de Borrador, debe haber al menos dos implementaciones independientes e interoperativas de las cuales se haya obtenido experiencia operativa adecuada.

Después de que se haya obtenido implementación significativa y experiencia operativa, la especificación puede ser elevada a Estándar de Internet. En este momento, se asigna un número STD y un número RFC a la especificación.

Finalmente, cuando un protocolo se queda obsoleto, se asigna al estado de Histórico.

CATEGORÍAS DE ESTÁNDARES DE INTERNET

Los estándares de Internet se dividen en dos categorías:

- **Especificación técnica (TS, Technical Specification):** define un protocolo, servicio, procedimiento, convención o formato. La mayoría de los estándares de Internet son especificaciones técnicas.

- **Informe de aplicabilidad (AS, *Applicability Statement*):** especifica cómo y en qué circunstancias una o más especificaciones técnicas pueden aplicarse para posibilitar una determinada capacidad de Internet. Un informe de aplicabilidad identifica a una o más especificaciones técnicas que son relevantes para la capacidad y puede especificar valores o rangos para determinados parámetros asociados con un TS o subgrupos funcionales de un TS relevantes para la capacidad.

OTROS TIPOS DE RFC

Hay numerosos RFC que no están destinados a convertirse en estándares de Internet. Algunos RFC estandarizan los resultados de las deliberaciones de la comunidad sobre los informes de principios o conclusiones sobre el mejor modo de realizar algunas operaciones o la función del proceso de la IETF. Estos RFC se denominan *Best Current Practice* (BCP). La aprobación de los BCP sigue básicamente el mismo proceso de aprobación para los Estándares Propuestos. A diferencia de los otros documentos, los BCP no siguen un proceso de tres etapas, sino que pasan del estatus de Borrador de Internet a BCP aprobado en un solo paso.

Un protocolo u otra especificación que no se considere preparada para la estandarización podría publicarse como un RFC Experimental. Más tarde, la especificación podría volverse a presentar. Si la especificación es, en general, estable, cumple los requisitos de diseño, se considera comprensible, ha recibido una revisión significativa por parte de la comunidad y parece gozar de suficiente interés en la misma, entonces el RFC se elevará al estatus de Estándar Propuesto.

Finalmente, se publica una Especificación Informativa para informar a la comunidad de Internet.

1.7 ESTRUCTURA DEL LIBRO

Este primer Capítulo, como vemos, sirve como introducción. El resto del libro se organiza en tres partes:

Primera Parte: proporciona un estudio conciso de los algoritmos criptográficos y los protocolos fundamentales para las aplicaciones de seguridad de redes, incluyendo el cifrado, las funciones *hash*, las firmas digitales y el intercambio de claves.

Segunda Parte: examina el uso de algoritmos criptográficos y protocolos de seguridad para proporcionar seguridad a las redes y a Internet. Los temas que abarca incluyen la autenticación de usuarios, el correo electrónico, la seguridad IP y web.

Tercera Parte: se centra en las herramientas de seguridad diseñadas para proteger un sistema informático de amenazas a la seguridad, como intrusos, virus y gusanos. Esta parte también menciona la tecnología de cortafuegos.

Muchos de los algoritmos criptográficos, protocolos y aplicaciones de seguridad en redes descritos en este libro han sido especificados como estándares. Los más importantes de ellos son los Estándares de Internet, definidos en los RFC de Internet (*Request for Comments*), y *Federal Information Processing Standards* (FIPS), publicados por el

NIST (*National Institute of Standards and Technology*). El apéndice A presenta un listado de los estándares citados en este libro.

1.8 BIBLIOGRAFÍA RECOMENDADA

[PFLE97] proporciona una buena introducción a la seguridad de computadores y de redes. Otro estudio excelente es [NICH99]. [SCHN00] es una lectura valiosa para cualquier iniciado en el campo de la seguridad en computadores y redes: trata las limitaciones de la tecnología y la criptografía en particular, a la hora de proporcionar seguridad, y la necesidad de tener en cuenta el *hardware*, la implementación de *software*, las redes y las personas implicadas en proporcionar seguridad y en atacarla.

NICH99 Nichols, R. Ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.

PFLE97 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley, 2000.

1.9 RECURSOS WEB Y DE INTERNET

Existe una serie de recursos web y de Internet que sirven de apoyo a este libro y que pueden ayudar a seguir el ritmo de los avances en este campo.

SITIOS WEB PARA ESTE LIBRO

Se ha creado una página web especial para este libro en:

WilliamStallings.com/NetSec2e.html

El sitio incluye lo siguiente:

- **Sitios web útiles:** enlaces a otros sitios web relevantes, organizados en capítulos, que incluyen los sitios que se mencionan en esta sección y a lo largo de este libro.
- **Fe de erratas:** se mantiene y actualiza la fe de erratas de este libro. Se agradecen las indicaciones por correo electrónico sobre cualquier error que se encuentre. La fe de erratas de mis otros libros se encuentran en WilliamStallings.com.
- **Figuras:** todas las figuras de este libro en formato PDF (Adobe Acrobat).
- **Tablas:** todas las tablas de este libro en formato PDF.
- **Diapositivas:** una serie de diapositivas en Power Point, organizadas por capítulos.
- **Lista de correo de Internet:** el sitio incluye información para participar en la lista de correo de Internet del libro.

- **Cursos de seguridad de redes:** hay enlaces a páginas de cursos basados en este libro y que pueden aportar ideas a otros profesores sobre cómo estructurar el curso.

También se mantiene el sitio de recursos para estudiantes de Ciencias de la Computación en:

WilliamStallings.com/StudentSupport.html

La finalidad de este sitio es la de proporcionar documentos, información y enlaces para estudiantes y profesionales del campo de la Computación. Los enlaces y documentos están organizados en cuatro categorías:

- **Matemáticas:** incluye un curso básico de matemáticas, un libro de texto de análisis de colas, un libro de texto de sistemas de números y enlaces a numerosos sitios de matemáticas.
- **Guía (how-to):** recomendaciones e indicaciones para resolver problemas, escribir informes técnicos y preparar presentaciones técnicas.
- **Recursos de investigación:** enlaces a artículos, informes técnicos y bibliografías.
- **Miscelánea:** otros documentos y enlaces útiles.

OTROS SITIOS WEB

Hay numerosos sitios web que proporcionan información relacionada con los temas de este libro. En la sección de *Bibliografía y sitios web recomendados* de capítulos posteriores, se pueden encontrar enlaces a sitios web específicos. Como las direcciones de los sitios web cambian con frecuencia, no están incluidas en el libro. Para todos los sitios web mencionados en este libro se puede encontrar el enlace adecuado en el sitio web del mismo, al que, además, se irán añadiendo otros enlaces que no se han mencionado.

Los siguientes sitios web de interés general están relacionados con la criptografía y la seguridad de redes:

- **COAST:** serie exhaustiva de enlaces relacionados con la criptografía y la seguridad de redes.
- **IETF Security Area:** material relacionado con la estandarización de la seguridad de Internet.
- **Computer and Network Security Reference Index:** un buen índice para productos comerciales, preguntas más frecuentes (FAQs), archivos de grupos de noticias, artículos y otros sitios web.
- **The Cryptography FAQ:** extensa y útil lista de preguntas más frecuentes que cubren todos los aspectos de la criptografía.
- **Tom Dunigan's Security Page:** una excelente lista de enlaces a sitios web sobre criptografía y seguridad de redes.
- **IEEE Technical Committee on Security and Privacy:** copias de sus newsletters, información sobre las actividades relacionadas con IEEE.
- **Computer Security Resource Center:** mantenido por el Instituto Nacional de Estándares y Tecnología (NIST); contiene una amplia información sobre amenazas a la seguridad, tecnología y estándares.

GRUPOS DE NOTICIAS DE USENET

Una serie de grupos de noticias de USENET están dedicados a algún aspecto de la seguridad de redes o la criptografía. Como con casi todos los grupos de USENET, hay un alto índice de ruido a señal, pero vale la pena comprobar si alguno cubre sus necesidades. Los más importantes son los siguientes:

- **sci.crypt.research:** es el grupo más interesante. Es un grupo moderado que trata temas de investigación; las consultas y comentarios deben guardar alguna relación con los aspectos técnicos del cifrado.
- **sci.crypt:** discusión general sobre cifrado y otros temas relacionados.
- **sci.crypt.random-numbers:** discusión sobre la aleatoriedad de la fuerza del cifrado.
- **alt.security:** discusión general sobre temas de seguridad.
- **comp.security.misc:** discusión general sobre temas de seguridad de computadores.
- **comp.security.firewalls:** discusión sobre productos y tecnología cortafuego.
- **comp.security.announce:** noticias y anuncios de CERT.
- **comp.risks:** discusión sobre riesgos ocasionados por los computadores y los usuarios.
- **comp.virus:** discusión moderada sobre virus informáticos.

P A R T E I

CRIPTOGRAFÍA

CONTENIDO DE LA PRIMERA PARTE

La herramienta automatizada más importante, con diferencia, para la seguridad de redes y comunicaciones es el cifrado. Habitualmente se usan dos formas de cifrado: cifrado convencional o simétrico, y cifrado de clave pública o asimétrico. La Primera Parte de este libro ofrece un estudio de los principios básicos de la criptografía simétrica y de clave pública, muestra los algoritmos de uso más común y discute la aplicabilidad básica de los dos enfoques.

GUÍA PARA LA PRIMERA PARTE

CAPÍTULO 2. CIFRADO SIMÉTRICO Y CONFIDENCIALIDAD DE MENSAJES

El Capítulo 2 se centra en el cifrado simétrico, resaltando la técnica de cifrado que más se usa, el DES (*Data Encryption Standard*) y sus sucesores, la versión de triple clave del DES y el AES (*Advanced Encryption Standard*). Además de las cuestiones que tratan de la construcción real de un algoritmo de cifrado simétrico, se relaciona una serie de aspectos de diseño con el uso del cifrado simétrico para proporcionar confidencialidad. El Capítulo incluye una discusión sobre el cifrado de mensajes largos, cifrado extremo a extremo frente a cifrado de enlace, y técnicas de distribución de claves.

CAPÍTULO 3. CRIPTOGRAFÍA DE CLAVE PÚBLICA Y AUTENTIFICACIÓN DE MENSAJES

De igual importancia que la confidencialidad, como medida de seguridad, es la autenticación. Como mínimo, la autenticación de mensajes garantiza que el mensaje proviene de la fuente esperada. Además, la autenticación puede incluir protección contra la modificación, el retraso, la repetición y el reordenamiento. El Capítulo 3 comienza con un análisis de requisitos para la autenticación y presenta, a continuación, una se-

rie de enfoques. Un elemento fundamental de los esquemas de autentificación es el uso de un autenticador, normalmente un código de autentificación de mensaje (MAC: *Message Authentication Code*) o una función *hash*. Se examinan consideraciones de diseño para ambos tipos de algoritmos y se analizan varios ejemplos específicos.

Después del cifrado simétrico, el otro método importante de cifrado es el de clave pública, que ha revolucionado la seguridad en las comunicaciones. El Capítulo 3 introduce el cifrado de clave pública. El algoritmo RSA se examina detalladamente y se reconsidera el problema de la gestión de claves. También se expone la técnica extendida de intercambio de claves de Diffie-Hellman. Además, se define la firma digital y se examina su aplicación.

CAPÍTULO 2

Cifrado simétrico y confidencialidad de mensajes

- 2.1. Principios del cifrado simétrico**
 - Criptografía
 - Criptoanálisis
 - Estructura de cifrado Feistel
- 2.2. Algoritmos de cifrado simétrico**
 - DES (Data Encryption Standard)
 - Triple DES
 - AES (Advanced Encryption Standard)
 - Otros cifradores simétricos de bloque
- 2.3. Modos de operación del cifrado de bloques**
 - Modo CBC (Cipher Block Chaining)
 - Modo CFB (Cipher Feedback)
- 2.4. Ubicación de los dispositivos de cifrado**
- 2.5. Distribución de claves**
- 2.6. Bibliografía y sitios web recomendados**
- 2.7. Palabras clave, preguntas de repaso y problemas**
 - Palabras clave
 - Preguntas de repaso
 - Problemas

Mungo había estado trabajando toda la tarde en el código Stern, ayudado principalmente de los últimos mensajes que había interceptado durante la caída del sistema de Nevin Square. El código Stern era muy seguro. Él debía ser consciente de que la Central de Londres sabía sobre esa caída. Se sentían tan seguros de la impenetrabilidad del código que no les importaba con qué frecuencia Mungo leía sus mensajes.

Hablar con desconocidos, RUTH RENDELL

Entre las tribus de Australia Central, cada hombre, mujer y niño tiene un nombre secreto o sagrado que el más anciano decide al poco de nacer él o ella, y que no conoce nadie sino los miembros totalmente integrados en el grupo. Este nombre secreto nunca se menciona excepto en las ocasiones más solemnes; pronunciarlo al oído de un hombre de otro grupo sería una seria violación de la costumbre tribal. Cuando se menciona uno de esos nombres, se hace con voz muy baja y tomando todas las precauciones para que nadie de otro grupo pueda oírlo. Los nativos creen que un extraño que conozca su nombre secreto tendría poder especial para causarle algún mal por medio de la magia.

La rama dorada, SIR JAMES GEORGE FRAZER

El cifrado simétrico, también conocido como cifrado convencional, de clave secreta o de clave única, era el único que se usaba antes del desarrollo del cifrado de clave pública a finales de los 70¹. Aún hoy continúa siendo el más usado de los dos tipos de cifrado.

Este Capítulo comienza presentando un modelo general del proceso de cifrado simétrico, permitiéndonos con ello entender el contexto en que se usan los algoritmos. Luego se analizan tres algoritmos de cifrado importantes: DES, triple DES y AES. También se examina la aplicación de estos algoritmos para obtener confidencialidad.

2.1 PRINCIPIOS DEL CIFRADO SIMÉTRICO

Un esquema de cifrado simétrico tiene cinco componentes (Figura 2.1):

- **Texto claro:** es el mensaje o los datos originales que se introducen en el algoritmo como entrada.
- **Algoritmo de cifrado:** el algoritmo de cifrado realiza varias sustituciones y transformaciones en el texto claro.
- **Clave secreta:** la clave es también una entrada del algoritmo. Las sustituciones y transformaciones realizadas por el algoritmo dependen de ella.
- **Texto cifrado:** el mensaje ilegible que se produce como salida. Depende del texto claro y de la clave secreta. Para un mensaje determinado, dos claves diferentes producirían dos textos cifrados diferentes.

¹ El cifrado de clave pública se describió por primera vez en 1976; la National Security Agency (NSA) afirma haberlo descubierto unos años antes.

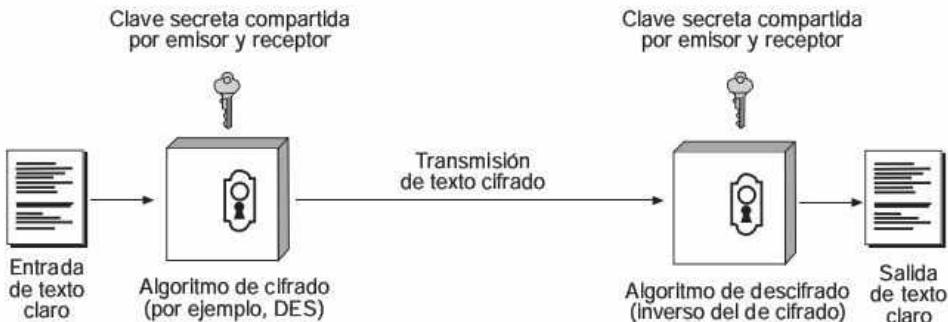


Figura 2.1 Modelo simplificado del cifrado convencional

- **Algoritmo de descifrado:** Es, básicamente, el algoritmo de cifrado ejecutado a la inversa. Toma el texto cifrado y la misma clave secreta, y genera el texto claro.

Hay dos requisitos para el uso seguro del cifrado simétrico:

1. Se necesita un algoritmo de cifrado robusto. Como mínimo nos gustaría que un oponente que conozca el algoritmo y tenga acceso a uno o más textos cifrados no pudiera descifrar el texto o averiguar la clave. Este requisito se expresa generalmente de forma más rotunda: el atacante no debería poder descifrar el texto o averiguar la clave aunque estuviera en posesión de una serie de textos cifrados junto con sus correspondientes textos originales.
2. El emisor y el receptor deben haber obtenido copias de la clave secreta de forma segura y deben guardarla de la misma manera. Si alguien descubre la clave y conoce el algoritmo, todas las comunicaciones que usen esa clave son descifrables.

Es importante observar que la seguridad del cifrado simétrico depende de la privacidad de la clave, no de la privacidad del algoritmo. Es decir, se asume que no es práctico desifrar un mensaje teniendo el texto cifrado y conociendo el algoritmo de cifrado/descifrado. En otras palabras, no es necesario que el algoritmo sea secreto; lo único que hay que mantener en secreto es la clave.

Esta característica del cifrado simétrico es la causa de su uso tan extendido. El hecho de que el algoritmo no tenga que ser secreto significa que los fabricantes pueden desarrollar y han desarrollado implementaciones con chips de bajo coste de los algoritmos de cifrado de datos. Esos chips se pueden conseguir fácilmente y se han incorporado a una serie de productos. Con el uso de cifrado simétrico, el problema principal de seguridad consiste en mantener la privacidad de la clave.

CRPTOGRAFÍA

Generalmente, los sistemas criptográficos se clasifican atendiendo a tres factores independientes:

1. **El tipo de operación utilizado para transformar el texto claro en texto cifrado.** Todos los algoritmos de cifrado se basan en dos principios generales: sustitución, donde cada elemento de texto claro (bit, letra, grupo de bits o letras) se

sustituye por otro diferente; y transposición, donde los elementos del texto claro se reordenan. Lo fundamental del proceso es que no se pierda información (es decir, que todas las operaciones sean reversibles). La mayoría de los sistemas emplean múltiples etapas de sustituciones y transposiciones.

- 2 **El número de claves usadas.** Si tanto el emisor como el receptor usan la misma clave, el sistema se denomina simétrico, de clave única, de clave secreta, o cifrado convencional. En cambio, si el emisor y el receptor usan cada uno claves diferentes, el sistema se denomina asimétrico, de dos claves, o cifrado de clave pública.
- 3 **La forma de procesar el texto claro.** Un *cifrador de bloque* procesa un bloque de elementos cada vez, produciendo un bloque de salida por cada bloque de entrada. Un cifrador de flujo procesa los elementos de entrada continuamente, produciendo la salida de un elemento cada vez.

CRPTOANÁLISIS

El *criptoanálisis* es el proceso por el que se intenta descubrir un texto claro o una clave de cifrado. La estrategia usada por el criptoanalista depende de la naturaleza del esquema de cifrado y de la información disponible.

La Tabla 2.1 resume los diferentes tipos de ataques criptoanalíticos, basados en la cantidad de información que posee el criptoanalista. El caso más difícil es aquel en que la única información disponible es el texto cifrado. En algunos casos ni siquiera se conoce el algoritmo de cifrado, pero en general se asume que el atacante lo conoce. Un posible ataque en estas circunstancias es el de fuerza bruta, intentando probar con todas las claves posibles. Si el espacio de claves es muy grande, este enfoque es inviable. En tal caso, el atacante debe contar con un análisis del texto cifrado, aplicándole, generalmente, varias pruebas estadísticas. Para utilizar este enfoque, el oponente debe tener alguna idea general del tipo de texto claro implicado (texto en inglés o francés, un fichero EXE, un listado de código fuente de Java, un fichero de cuentas, etc.).

El ataque con sólo texto cifrado es el más fácil de evitar, ya que el oponente dispone de la mínima cantidad de información con la que trabajar. En muchos casos, sin embargo, el analista dispone de más información. Éste puede ser capaz de conseguir uno o más mensajes de texto claro y sus correspondientes cifrados, o podría saber que en un mensaje aparecerán ciertos patrones de texto claro. Por ejemplo, un fichero codificado en formato Postscript siempre comienza con el mismo patrón, o podría haber un encabezado estandarizado para un mensaje de transferencia electrónica de fondos, etc. Éstos son algunos ejemplos de *texto claro conocido*. Sabiendo esa información, el analista podría deducir la clave basándose en la forma en que se transforma el texto claro que conoce.

Estrechamente relacionado con el ataque de texto claro conocido se encuentra el que se puede denominar como ataque de palabra probable. Si el atacante está trabajando con el cifrado de algún mensaje general en prosa, podría tener poco conocimiento del contenido del mensaje. Sin embargo, si va tras alguna información específica, entonces podría conocer partes del mismo. Por ejemplo, si se está transmitiendo un fichero completo de cuentas, el oponente podría conocer la situación de ciertas palabras clave en el encabezado del fichero. Otro ejemplo sería el hecho de que el código fuente de un pro-

grama desarrollado por una corporación podría incluir información de *copyright* en alguna posición estandarizada.

Si el analista, de alguna forma, puede conseguir que el sistema fuente inserte en el sistema un mensaje elegido por él, entonces se puede producir un *ataque de texto claro elegido*. En general, si el analista es capaz de elegir los mensajes que va a cifrar, podría captar deliberadamente patrones que se pueden esperar para revelar la estructura de la clave.

La Tabla 2.1 presenta otros dos tipos de ataques: texto cifrado elegido y texto elegido. Éstos se utilizan menos como técnicas criptoanalíticas pero constituyen, no obstante, vías posibles para los ataques.

Tabla 2.1 Tipos de ataque a mensajes cifrados

Tipo de ataque	Información del criptoanalista
Sólo texto cifrado	<ul style="list-style-type: none"> Algoritmo de cifrado. Texto cifrado que se va a decodificar.
Texto claro conocido	<ul style="list-style-type: none"> Algoritmo de cifrado. Texto cifrado que se va a decodificar. Uno o más pares de texto claro-texto cifrado formados con la clave secreta.
Texto claro elegido	<ul style="list-style-type: none"> Algoritmo de cifrado. Texto cifrado que se va a decodificar. Mensaje de texto en claro elegido por el criptoanalista junto con su correspondiente texto cifrado generado con la clave secreta.
Texto cifrado elegido	<ul style="list-style-type: none"> Algoritmo de cifrado. Texto cifrado que se va a decodificar. Texto cifrado intencionado elegido por el criptoanalista, junto con su correspondiente texto claro descifrado generado con la clave secreta.
Texto elegido	<ul style="list-style-type: none"> Algoritmo de cifrado. Texto cifrado que se va a decodificar. Mensaje de texto claro elegido por el criptoanalista junto con su correspondiente texto cifrado generado con la clave secreta. Texto cifrado intencionado elegido por el criptoanalista, junto con su correspondiente texto claro generado con la clave secreta.

Solamente un algoritmo relativamente débil no resistiría un ataque de sólo texto cifrado. Generalmente, un algoritmo de cifrado se diseña para resistir un ataque de texto claro conocido.

Un esquema de cifrado es **computacionalmente seguro** si el texto cifrado generado cumple uno o los dos criterios siguientes:

- El coste de romper el cifrado excede el valor de la información cifrada.

- El tiempo necesario para romper el cifrado excede el tiempo de vida útil de la información.

El problema está en que es muy difícil estimar la cantidad de esfuerzo necesario para realizar satisfactoriamente el criptoanálisis del texto cifrado. Sin embargo, si no hay debilidades matemáticas inherentes en el algoritmo, lo que procede es un enfoque de fuerza bruta, y aquí se pueden calcular algunas estimaciones razonables sobre costos y tiempo.

El enfoque de fuerza bruta implica intentar cada clave posible hasta que se obtenga una traducción legible del texto cifrado al texto claro. Como promedio, se debe intentar la mitad de todas las claves posibles para conseguir descubrirla. La Tabla 2.2 muestra el tiempo necesario para distintos tamaños de clave. El tamaño de clave de 56 bits se usa con el algoritmo DES (Data Encryption Standard). Para cada tamaño de clave, se muestran los resultados suponiendo que cada operación de descifrado simple necesita un μs , lo cual es un valor razonable para las máquinas actuales. Con el uso de arquitecturas de microprocesadores en paralelo, podría ser posible conseguir ratios de procesamiento de mayor magnitud. La última columna de la Tabla 2.2 considera los resultados para un sistema que puede procesar un millón de claves por microsegundo. Como se puede ver, con un nivel de rendimiento como éste, el DES dejaría de ser computacionalmente seguro.

Tabla 2.2 Tiempo medio para la búsqueda exhaustiva de claves

Tamaño de clave (bits)	Número de claves alternativas	Tiempo necesario a 1 cifrado/ μs	Tiempo necesario a 10^6 cífrados/ μs
32	$2^{32} = 4,3 \times 10^9$	$2^{31} \mu\text{s} = 35,8$ minutos	2,15 milisegundos
56	$2^{56} = 7,2 \times 10^{16}$	$2^{55} \mu\text{s} = 1.142$ años	10,01 horas
128	$2^{128} = 3,4 \times 10^{38}$	$2^{127} \mu\text{s} = 5,4 \times 10^{24}$ años	$5,4 \times 10^{18}$ años
168	$2^{168} = 3,7 \times 10^{50}$	$2^{167} \mu\text{s} = 5,9 \times 10^{36}$ años	$5,9 \times 10^{30}$ años
26 caracteres (permutación)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6,4 \times 10^{12}$ años	$6,4 \times 10^6$ años

ESTRUCTURA DE CIFRADO FEISTEL

La mayoría de los algoritmos de cifrado simétrico de bloque, incluido el DES, tienen una estructura descrita inicialmente por Horst Feistel de IBM en 1973 [FEIS73], y que se muestra en la Figura 2.2. Las entradas al algoritmo de cifrado son un bloque de texto claro de tamaño $2w$ bits y una clave K . El bloque de texto claro se divide en dos mitades, L_0 y R_0 . Las dos mitades de datos pasan a través de n etapas de procesamiento y luego se combinan para producir el bloque de texto cifrado. Cada etapa i tiene como entradas L_{i-1} y R_{i-1} , que se derivan de la etapa anterior, así como una subclave K_i generada a partir de K . En general, las subclaves K_i son diferentes a K y entre ellas mismas, y se generan a partir de la clave mediante un algoritmo de generación de subclaves.

Todas las etapas tienen la misma estructura. Se realiza una sustitución sobre la mitad izquierda de los datos. Esto se hace aplicando una función de etapa F a la mitad derecha

de los datos y haciendo luego un OR exclusivo (XOR) de la salida de la función y la mitad izquierda de los datos. La función de etapa tiene la misma estructura general para cada etapa pero está parametrizada por la subclave de etapa K_i . Despues de esta sustitución, se realiza una permutación que consiste en intercambiar las dos mitades de datos.

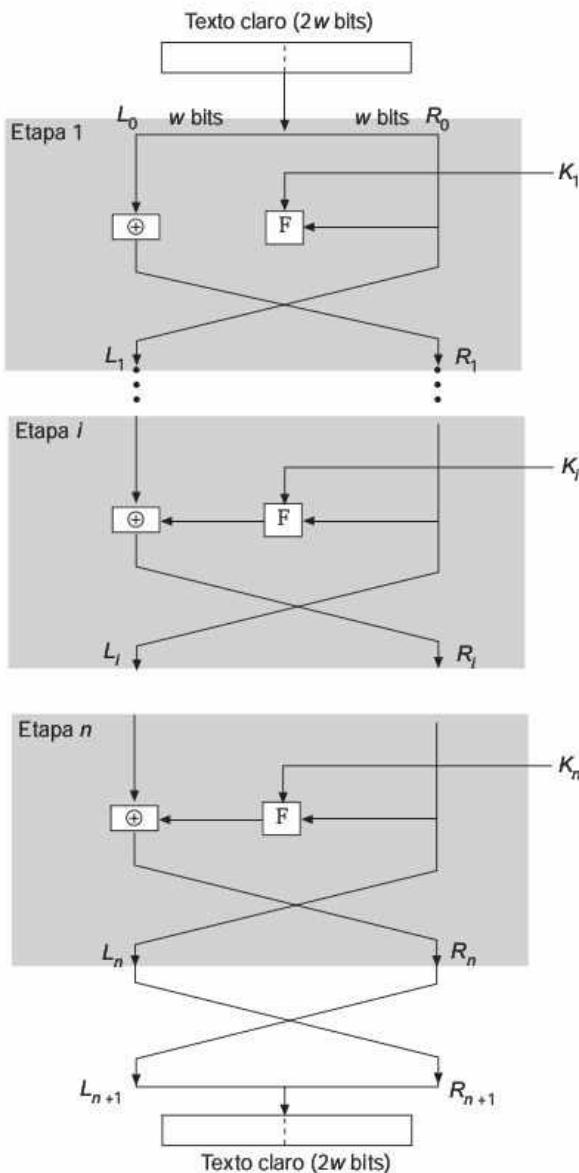


Figura 2.2 Red clásica de Feistel

La realización exacta de una red de Feistel depende de la elección de los siguientes parámetros y características de diseño:

- **Tamaño del bloque:** bloques mayores implican mayor seguridad (quedando igual todo lo demás) pero reduce la velocidad de cifrado/descifrado. Un tamaño de 64 bits es adecuado y es casi universal en el diseño del cifrador de bloques.
- **Tamaño de la clave:** claves más largas implican mayor seguridad pero puede reducir la velocidad de cifrado/descifrado. La longitud de clave más común en los algoritmos modernos es de 128 bits.
- **Número de etapas:** la esencia del cifrado de Feistel consiste en que mientras una sola etapa ofrece una seguridad inadecuada, múltiples etapas la aumentan. Un valor típico es 16 etapas.
- **Algoritmo de generación de subclaves:** cuanto más complejo sea este algoritmo más difícil resultará el criptoanálisis.
- **Función de etapa:** otra vez, generalmente, a mayor complejidad mayor resistencia al criptoanálisis.

Hay otras dos consideraciones en el diseño del cifrado de Feistel:

- **Cifrado/descifrado mediante software rápido:** en muchos casos, el cifrado es parte de aplicaciones o funciones de utilidad, lo cual imposibilita una implementación *hardware*. En esos casos, la rapidez de ejecución del algoritmo es un aspecto importante.
- **Facilidad de análisis:** aunque sería deseable hacer el algoritmo tan difícil como fuera posible para el criptoanálisis, también es ventajoso hacerlo fácil de analizar. Es decir, si el algoritmo se puede explicar de manera concisa y clara, entonces es más fácil detectar las debilidades y por tanto desarrollar un nivel mayor de garantías en función de su robustez. El DES, por ejemplo, no es funcionalmente fácil de analizar.

El descifrado es básicamente igual que el proceso de cifrado. La regla es la siguiente: usar el texto cifrado como entrada al algoritmo, pero usar las subclaves K_i en orden inverso. Es decir, usar K_n en la primera etapa, K_{n-1} en la segunda, y así sucesivamente hasta K_1 en la última. Es una ventaja ya que implica que no es necesario implementar dos algoritmos diferentes, uno para el cifrado y otro para el descifrado.

2.2 ALGORITMOS DE CIFRADO SIMÉTRICO

Los algoritmos de cifrado simétrico más comúnmente usados son los cifradores de bloques. Un cifrador de bloques procesa la entrada de texto claro en bloques de tamaño fijo y genera un bloque de texto cifrado del mismo tamaño para cada texto claro. Esta sección se centra en los tres cifradores de bloque simétrico más importantes: el DES (Data Encryption Standard) y el 3DES (triple DES), y el AES (Advanced Encryption Standard). Además, se muestran, brevemente, otros algoritmos de cifrado simétrico en uso.

DES (DATA ENCRYPTION STANDARD)

El esquema de cifrado más extendido se basa en el DES (Data Encryption Standard) adoptado en 1977 por el National Bureau of Standards, ahora el NIST (National Institute of Standards and Technology), como Federal Information Processing Standard 46.

(FIPS PUB 46). Al algoritmo se le denomina DEA (*Data Encryption Algorithm*)².

DESCRIPCIÓN DEL ALGORITMO

El texto claro tiene una longitud de 64 bits y la clave, de 56; si el texto claro es más largo se procesa en bloques de 64 bits. La estructura del DES consiste en una pequeña variación de la red de Feistel, que se muestra en la Figura 2.2. Hay 16 etapas de proceso. Se generan 16 subclaves partiendo de la clave original de 56 bits, una para cada etapa.

El proceso de descifrado con el DES es básicamente el mismo que el de cifrado. La regla es la siguiente: usar el texto cifrado como entrada al algoritmo del DES, pero las subclaves K_i se pasan en orden inverso. Es decir, en la primera etapa se usa K_{16} , K_{15} en la segunda, y así hasta K_1 en la 16^a y última.

ROBUSTEZ DEL DES

Los aspectos de robustez del DES se engloban en dos categorías: aspectos sobre el algoritmo mismo y aspectos sobre el uso de una clave de 56 bits. Los primeros se refieren a la posibilidad de que el criptoanálisis se realice explotando las características del algoritmo DES. A lo largo de los años, se han intentado encontrar debilidades que explotar en el algoritmo, lo que ha hecho del DES el algoritmo de cifrado existente más estudiado. A pesar de los numerosos enfoques, nadie ha conseguido descubrir ninguna debilidad grave en el DES³.

Un aspecto de mayor importancia es la longitud de la clave. Con una clave de 56 bits, hay 2^{56} claves posibles, que es aproximadamente $7,2 \times 10^{16}$ claves. Por este motivo, no parece práctico un ataque de fuerza bruta. Suponiendo que, en promedio, se tiene que intentar la mitad del espacio de claves, una única máquina que realice un cifrado DES por microsegundo tardaría más de mil años en romper el cifrado (véase la Tabla 2.2).

En cualquier caso, la suposición de un cifrado por microsegundo es demasiado conservadora. Finalmente y definitivamente, en julio de 1998, se probó que el DES no era seguro, cuando la Electronic Frontier Foundation (EFF) anunció que había roto un cifrado DES utilizando una máquina especializada, «DES cracker», construida por menos de 250.000 dólares. El ataque duró menos de tres días. La EFF ha publicado la descripción detallada de la máquina, haciendo posible que cualquiera construya su propio *cracker* [EFF98]. Naturalmente, los precios del *hardware* continuarán bajando mientras la velocidad irá aumentando, haciendo al DES prácticamente inútil.

Es importante tener en cuenta que para un ataque de búsqueda de clave no basta con probar todas las posibles claves. A menos que se suministre el texto claro, el analista debe ser capaz de reconocer el texto claro como tal. Si el mensaje es texto claro en inglés, entonces el resultado se obtiene fácilmente, aunque la tarea de reconocimiento del

² La terminología es un poco confusa. Hasta hace poco, los términos DES y DEA se usaban indistintamente. Sin embargo, la edición más reciente del documento del DES incluye una especificación del DEA y del triple DEA (3DES). Ambos son parte del DES (Data Encryption Standard). Es más, hasta la reciente adopción del término oficial 3DES, al algoritmo triple DEA se le denominaba *triple DES* y se escribía 3DES. Por conveniencia, nosotros usaremos 3DES.

³ Al menos, nadie ha revelado públicamente tal descubrimiento.

inglés tendría que estar automatizada. Si el mensaje de texto se ha comprimido antes del cifrado, entonces el reconocimiento es más difícil. Y si el mensaje es de un tipo más general de datos, como un fichero numérico, y ha sido comprimido, el problema es aún más difícil de automatizar. Por eso, para complementar el enfoque de fuerza bruta, se necesita algún grado de conocimiento sobre el texto claro esperado y alguna forma de distinguir automáticamente el texto claro de lo que no lo es. El enfoque de la EFF trata también este tema, e introduce algunas técnicas automatizadas que serían efectivas en muchos contextos.

Como punto final, si la única forma de ataque a un algoritmo de cifrado es la fuerza bruta, entonces la manera de contrarrestar ese ataque es obvia: usar claves más largas. Para tener una idea del tamaño de clave necesario, usaremos el *cracker* de la EFF como base de nuestras estimaciones. El *cracker* de la EFF era un prototipo, y se puede suponer que con la tecnología actual es rentable construir una máquina más rápida. Si asumimos que un dispositivo como el *cracker* puede realizar un millón de descifrados por μs , que es el ratio usado en la Tabla 2.2, entonces se tardaría alrededor de diez horas en descifrar un código DES. Esto constituye un incremento de velocidad en aproximadamente un factor de siete comparado con los resultados de la EFF. Usando este ratio, la Figura 2.3 muestra cuánto se tardaría en romper un algoritmo del estilo del DES en función del tamaño de la clave. Por ejemplo, para una clave de 128 bits, que es común entre los algoritmos actuales, se tardaría 10^{18} años en romper el código usando el *cracker* de la EFF. Incluso, aunque se aumentara la velocidad del *cracker* en un factor de un trillón (10^{12}), todavía se tardaría un millón de años en romper el código. Así que una clave de 128 bits garantiza que el algoritmo es inexpugnable por la fuerza bruta.

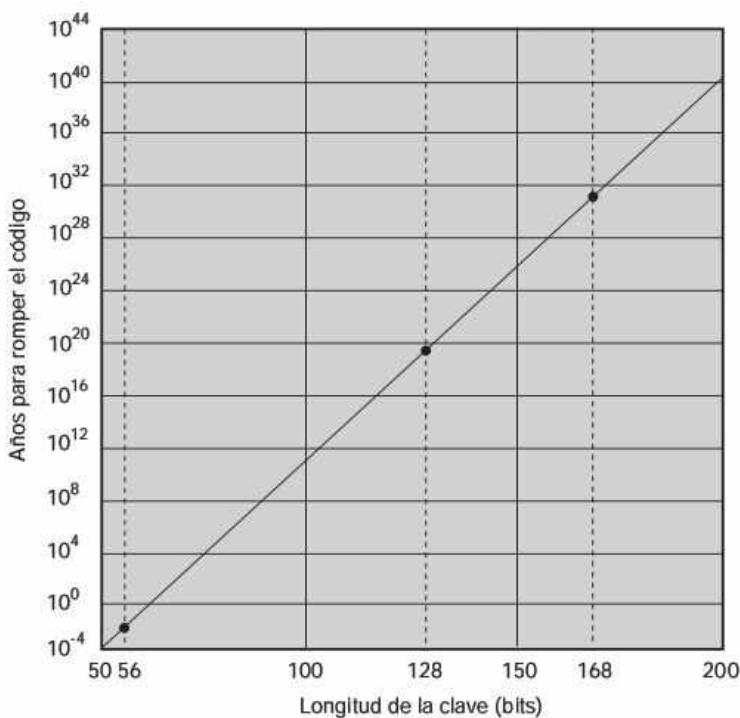


Figura 2.3 Tiempo empleado en romper un código (suponiendo 10^6 descifrados/ μs)

TRIPLE DES

El triple DES (3DES) se estandarizó inicialmente para aplicaciones financieras en el estándar ANSI X9.17 en 1985. El 3DES se incorporó como parte del DES en 1999, con la publicación de FIPS PUB 46-3.

El 3DES usa tres claves y tres ejecuciones del algoritmo DES. La función sigue la secuencia cifrar-descifrar-cifrar (EDE: encrypt-decrypt-encrypt) (Figura 2.4a):

$$C = E_{K_3} [D_{K_2} [E_{K_1} [P]]]$$

donde

C = texto cifrado

P = texto claro

$E_K[X]$ = cifrado de X usando la clave K

$D_K[Y]$ = descifrado de Y usando la clave K

El descifrado es simplemente la misma operación con las claves en orden inverso (Figura 2.4b):

$$P = D_{K_1} [E_{K_2} [D_{K_3} [C]]]$$

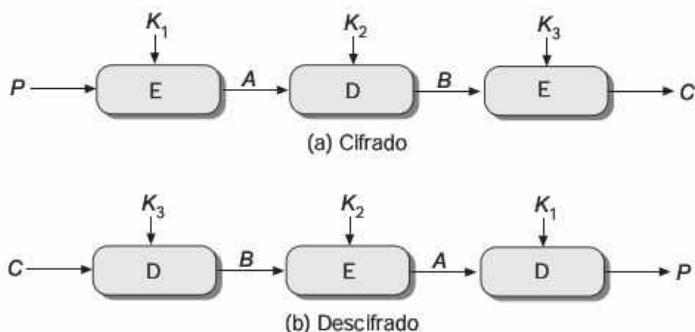


Figura 2.4 Triple DES

El descifrado del segundo paso no es significativo en términos criptográficos. Su única ventaja es que permite a los usuarios del 3DES descifrar datos cifrados por usuarios del DES:

$$C = E_{K_1} [D_{K_1} [E_{K_1} [P]]] = E_{K_1} [P]$$

Con tres claves diferentes, el 3DES tiene una longitud efectiva de clave de 168 bits. El FIPS 46-3 también permite el uso de dos claves, con $K_1 = K_3$, lo que proporciona una longitud de clave de 112 bits. El FIPS 46-3 incluye las siguientes directrices para el 3DES:

- El 3DES es el algoritmo de cifrado simétrico oficial del FIPS.
- El DES original, que usa una única clave de 56 bits, se mantiene sólo para los sistemas existentes. Las nuevas adquisiciones deberían admitir 3DES.
- Se apremia a las organizaciones gubernamentales con sistemas que usan DES a migrar a 3DES.
- Se prevé que el 3DES y el AES (Advanced Encryption Standard) coexistirán como algoritmos oficiales del FIPS, permitiendo una transición gradual hacia el AES.

Es fácil observar que el 3DES es un algoritmo robusto. Debido a que el algoritmo criptográfico que lo sustenta es el DEA, el 3DES resulta igual de resistente al criptoanálisis basado en el algoritmo que el DEA. Es más, con una clave de 168 bits de longitud, los ataques de fuerza bruta son efectivamente imposibles.

Últimamente, el AES pretende reemplazar al 3DES, pero este proceso tardará unos años. El NIST ha anunciado que el 3DES se mantendrá como algoritmo oficial (para uso del gobierno estadounidense) durante los próximos años.

AES (ADVANCED ENCRYPTION STANDARD)

El 3DES tiene dos atractivos que aseguran su uso durante los próximos años. Primero, con su longitud de clave de 168 bits evita la vulnerabilidad al ataque de fuerza bruta del DEA. Segundo, el algoritmo de cifrado que usa es el mismo que en el DEA. Este algoritmo ha estado sujeto a más escrutinios que ningún otro durante un largo período de tiempo, y no se ha encontrado ningún ataque criptoanalítico efectivo basado en el algoritmo que no sea la fuerza bruta. Por tanto, hay un alto grado de seguridad en la resistencia al criptoanálisis del 3DES. Si la seguridad fuera la única consideración, el 3DES sería una elección adecuada para un algoritmo de cifrado estándar durante las próximas décadas.

El inconveniente principal del 3DES es que el algoritmo es relativamente lento en su implementación *software*. El DEA original se diseñó para implementaciones *hardware* de mediados de los 70 y no produce código *software* eficiente. El 3DES tiene tres veces más etapas que el DEA y, por ello, es más lento. Un inconveniente secundario es que tanto el DEA como el 3DES usan un tamaño de bloque de 64 bits. Por razones tanto de eficiencia como de seguridad, es preferible un mayor tamaño de bloque.

Debido a esos inconvenientes, el 3DES no es un candidato razonable para usarlo durante mucho tiempo. Para reemplazarlo, el NIST realizó en 1997 un concurso de propuestas para el desarrollo de un nuevo estándar de cifrado avanzado (AES), que debería ser tan robusto o más que el 3DES y que mejoraría significativamente la eficiencia. Además de esos requisitos generales, el NIST especificó que el AES debía ser un cifrador simétrico de bloque con una longitud de bloque de 128 bits y permitir longitudes de clave de 128, 192 y 256 bits. Los criterios de evaluación incluyen la seguridad, la eficiencia computacional, los requisitos de memoria, la idoneidad para *hardware* y *software* y la flexibilidad.

En la primera etapa de evaluación, se aceptaron 15 de los algoritmos propuestos. La segunda etapa los redujo a cinco. El NIST completó el proceso de evaluación y publicó el estándar final (FIPS PUB 197) en noviembre de 2001. El algoritmo seleccionado por

el NIST como propuesta del AES fue el Rijndael, desarrollado y presentado por dos criptógrafos belgas: Dr. Joan Daemen y Dr. Vincent Rijmen.

DESCRIPCIÓN DEL ALGORITMO

El AES usa una longitud de bloque de 128 bits, y la longitud de la clave puede ser de 128, 192 o 256 bits. En la descripción de esta sección se asume una longitud de clave de 128 bits, que posiblemente es la más implementada.

La Figura 2.5 muestra la estructura general del AES. La entrada a los algoritmos de cifrado y descifrado es un solo bloque de 128 bits. En el FIPS PUB 197, este bloque se representa como una matriz cuadrada de bytes. Este bloque se copia en el vector **Estado**, que se modifica en cada etapa del cifrado o descifrado. Después de la última etapa, **Estado** se copia en una matriz de salida. De igual manera, la clave de 128 bits se representa como una matriz cuadrada de bytes. Esta clave luego se expande en un vector de palabras para la generación de claves; cada palabra tiene cuatro bytes, y el número total de palabras para generar claves es de 44 para la clave de 128 bits. El orden de los bytes dentro de una matriz se establece por columnas. Así, por ejemplo, los primeros cuatro bytes de una entrada de texto claro de 128 bits al cifrador ocupan la primera columna de la matriz **in**, los segundos cuatro bytes la segunda columna, y así sucesivamente. De igual forma, los primeros cuatro bytes de la clave expandida, que forman una palabra, ocupan la primera columna de la matriz **w**.

Los siguientes comentarios revelan algunos aspectos del AES:

1. Una característica notable de su estructura es que no es una estructura Feistel. En la estructura clásica de Feistel, la mitad del bloque de datos se usaba para modificar la otra mitad, y entonces se intercambiaban entre sí. El AES procesa todo el bloque de datos en paralelo durante cada etapa, realizando sustituciones y permutaciones.
2. La clave suministrada como entrada se expande en un vector de 44 palabras de 32 bits, **w[i]**. Cuatro palabras diferentes (128 bits) sirven como clave de etapa en cada ronda.
3. Se utilizan cuatro fases diferentes, una de permutación y tres de sustitución:
 - **Sustitución de bytes:** se usa una tabla, denominada *caja S*⁴, para realizar una sustitución byte a byte del bloque.
 - **Desplazamiento de filas:** una simple permutación realizada fila por fila.
 - **Mezcla de columnas:** una sustitución que altera cada byte de una columna en función de todos los bytes de la columna.
 - **Suma de la clave de etapa:** una simple operación XOR bit a bit del bloque actual con una porción de la clave expandida.
4. La estructura es muy simple. Tanto para el cifrado como para el descifrado, se comienza con una fase de suma de clave de etapa, seguido de nueve etapas de

⁴ El término *caja S* (caja de sustitución) se utiliza normalmente en la descripción de cifradores simétricos para referirse a un tipo de tabla de consulta usada por el mecanismo de sustitución.

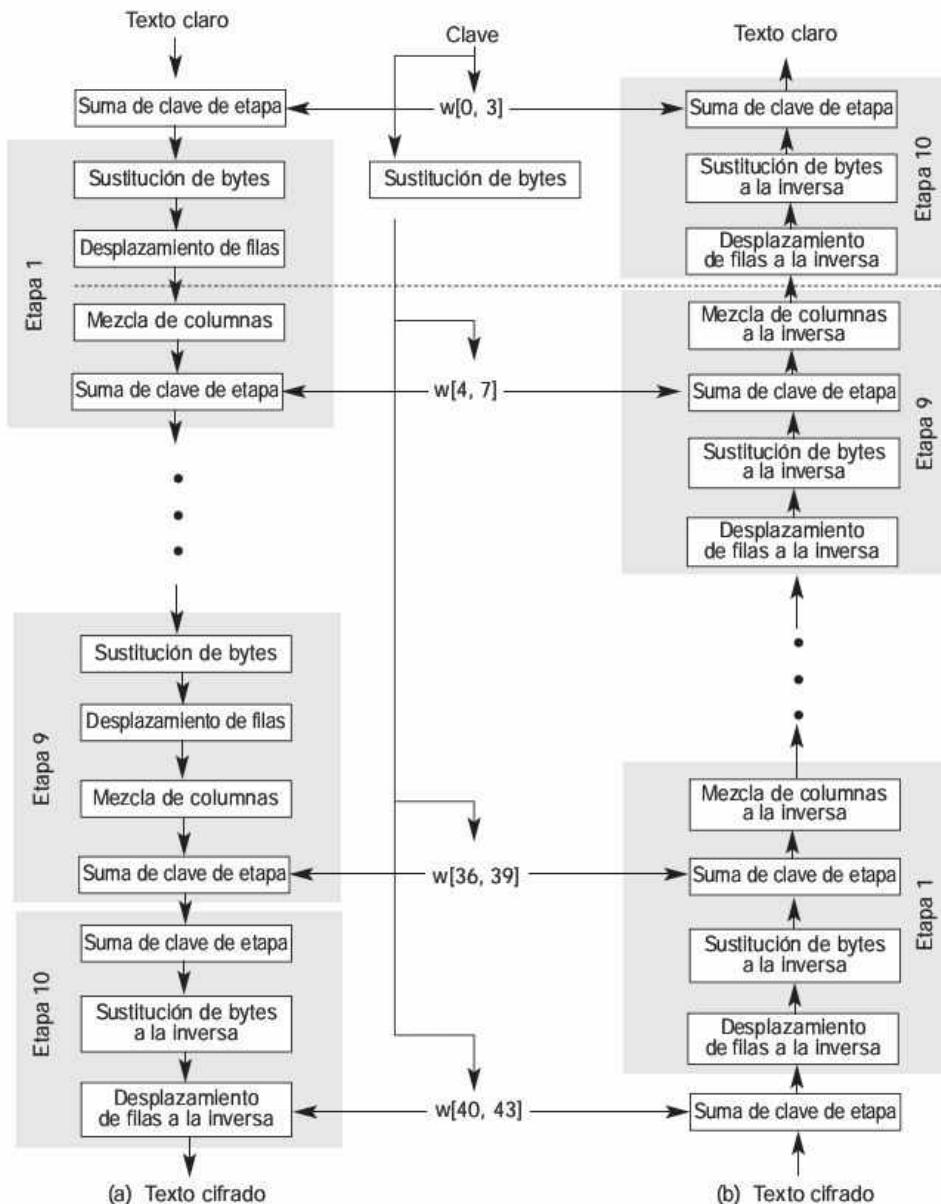


Figura 2.5 Cifrado y descifrado AES

cuatro fases cada una, y acaba con una décima etapa de tres fases. La Figura 2.6 muestra la estructura de una etapa completa de cifrado.

- 5. Solamente la fase de suma de la clave de etapa utiliza la clave. Por esta razón el cifrador comienza y termina con una suma de clave de etapa. Cualquier otra fase, aplicada al comienzo o al final, sería reversible sin conocer la clave y por tanto añadiría inseguridad.

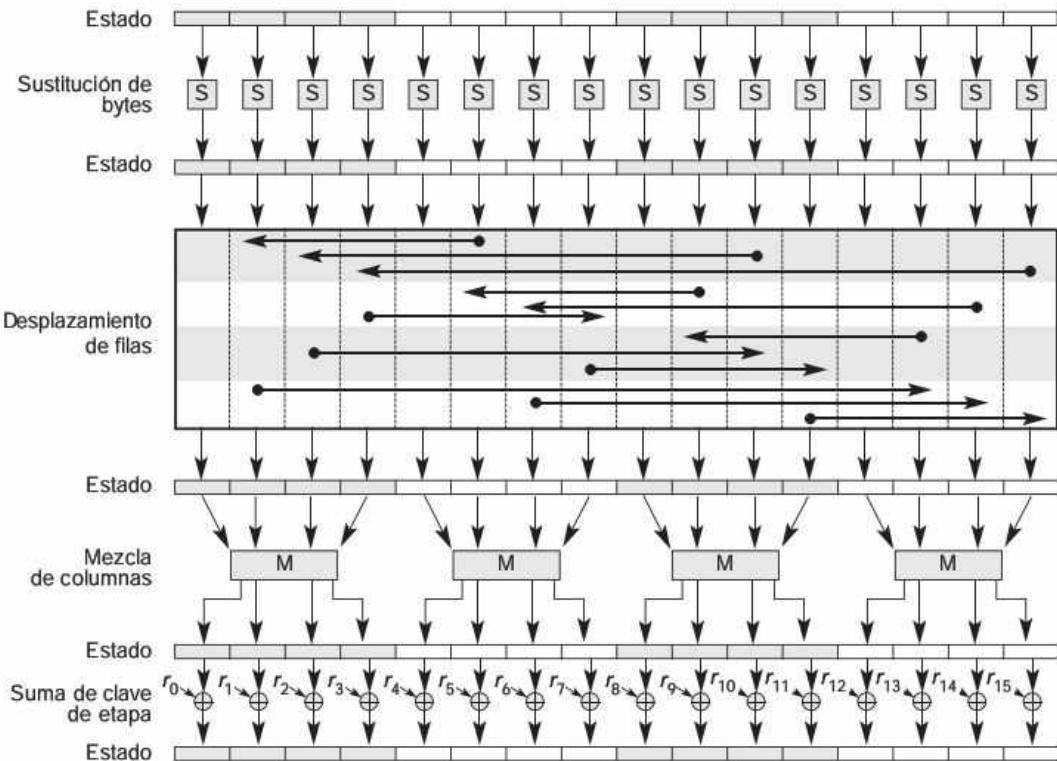


Figura 2.6 Etapa de cifrado del AES

- 6 La fase de suma de la clave de etapa no funcionaría por sí misma. Las otras tres fases juntas desordenan los bits, pero no proporcionan seguridad por sí mismas, porque no usan la clave. Se puede ver el cifrador como una secuencia alternativa de operaciones de cifrado XOR (suma de clave de etapa) de un bloque, seguida por un desordenamiento del bloque (las otras tres fases), seguida por un cifrado XOR, y así sucesivamente. Este esquema es eficiente y muy seguro.
- 7 Cada fase es fácilmente reversible. Para las fases de sustitución de byte, desplazamiento de fila y mezcla de columnas, se usa una función inversa en el algoritmo de descifrado. Para la fase de suma de clave de etapa, la inversa se consigue con un XOR entre la misma clave de etapa y el bloque, usando la propiedad de que $A \oplus A \oplus B = B$.
- 8 Como con la mayoría de los cifradores de bloque, el algoritmo de descifrado hace uso de la clave expandida en orden inverso. De todas formas, como consecuencia de la estructura particular del AES, el algoritmo de descifrado no es idéntico al de cifrado.
- 9 Una vez se ha establecido que las cuatro fases de cada etapa son reversibles, es fácil verificar que el descifrado recupera el texto claro. La Figura 2.5 muestra el cifrado y el descifrado desplazándose en direcciones verticalmente opuestas.

En cada punto horizontal (por ejemplo, la línea discontinua de la figura), **Estado** es el mismo para el cifrado y para el descifrado.

- 10.** La última etapa de cifrado y descifrado consiste sólo en tres fases. Otra vez, esto es consecuencia de la estructura particular del AES y es necesario para que el cifrador sea reversible.

OTROS CIFRADORES DE BLOQUE SIMÉTRICOS

En vez de reinventar de nuevo la rueda, casi todos los algoritmos de cifrado de bloque simétricos actuales utilizan la estructura básica de bloque de Feistel. Esto se debe a que dicha estructura se comprende muy bien, resultando más fácil determinar la robustez criptográfica de un algoritmo nuevo. Si se usara una estructura totalmente nueva, podría tener alguna debilidad sutil no detectada inmediatamente por el diseñador. En esta sección se verán otros cifradores que, al igual que el DES y el 3DES, han tenido aceptación comercial. En la Tabla 2.3 se comparan algunas de las principales características.

Tabla 2.3 Algoritmos de cifrado convencional

Algoritmo	Tamaño de clave (bits)	Tamaño de bloque (bits)	Número de etapas	Aplicaciones
DES	56	64	16	SET, Kerberos
Triple DES	112 o 168	64	48	Financial key management, PGP, S/MIME
AES	128, 192 o 256	128	10, 12 o 14	Destinado a sustituir DES y 3DES
IDEA	128	64	8	PGP
Blowfish	Variable hasta 448	64	16	Varios paquetes de software
RC5	Variable hasta 2048	64	Variable hasta 255	Varios paquetes de software

IDEA

El IDEA (International Data Encryption Algorithm) es un cifrador de bloque simétrico desarrollado por Xuejia Lai y James Massey del Instituto Federal Suizo de Tecnología en 1991 [LA91]. El IDEA usa una clave de 128 bits. Difiere notablemente del DES en la función de etapa así como en la función de generación de subclaves. Para la función de etapa el IDEA no usa cajas S, sino que cuenta con tres operaciones matemáticas diferentes: XOR, suma binaria de enteros de 16 bits, y multiplicación binaria de enteros de 16 bits. Esas funciones se combinan de tal forma que producen una transformación compleja muy difícil de analizar, y por ende muy difícil para el criptoanálisis. El algoritmo de generación de subclave se basa solamente en el uso de desplazamientos circu-

lares pero ejecutados de manera compleja para generar un total de seis subclaves para cada una de las ocho etapas del IDEA.

Debido a que el IDEA fue uno de los primeros algoritmos de 128 bits de los propuestos para remplazar al DES, ha sido sometido a considerables exámenes y por ahora parece ser muy resistente al criptoanálisis. El IDEA se usa como una alternativa en PGP (*Pretty Good Privacy*) y también en una serie de productos comerciales.

BLOWFISH

El *Blowfish* fue desarrollado en 1993 por Bruce Schneier [SCHN93, SCHN94], asesor y criptógrafo independiente, y consiguió ser rápidamente una de las alternativas más populares al DES. Se diseñó para que fuera fácil de implementar y rápido en su ejecución. También es un algoritmo muy compacto que puede ejecutarse en menos de 5K de memoria. Una característica interesante es que la longitud de la clave es variable, pudiendo alcanzar hasta los 448 bits. En la práctica se usan claves de 128 bits. El *Blowfish* usa 16 etapas.

Utiliza cajas S y la función XOR, como el DES, pero también utiliza sumas binarias. Al contrario que el DES, que utiliza cajas S estáticas, *Blowfish* usa cajas S dinámicas generadas como una función de la clave. Las subclaves y las cajas S se generan por la aplicación repetida del propio algoritmo a la clave. Se necesita un total de 521 ejecuciones del algoritmo de cifrado *Blowfish* para producir las subclaves y las cajas S. Por este motivo no es adecuado para aplicaciones en las que la clave secreta cambia frecuentemente.

Este es uno de los algoritmos de cifrado simétrico más robustos hasta la fecha, porque tanto las subclaves como las cajas S se generan por un proceso de aplicaciones repetidas del propio algoritmo, lo cual modifica totalmente los bits haciendo muy difícil el criptoanálisis. Hasta ahora, se han publicado algunos artículos sobre el *Blowfish*, sin que se hayan encontrado debilidades.

Este algoritmo se usa en varias aplicaciones comerciales.

RC5

El RC5 fue desarrollado en 1994 por Ron Rivest [RIVE94, RIVE95], uno de los inventores del algoritmo de clave pública RSA. El RC5 se define en el RFC 2040 y se diseñó para tener las siguientes características:

- **Adequado para hardware y software:** sólo usa operaciones computacionales primitivas que se encuentran comúnmente en los microprocesadores.
- **Rápido:** para conseguir esto, el RC5 es un algoritmo simple y orientado a palabras. Las operaciones básicas procesan palabras enteras de datos cada vez.
- **Adaptable a procesadores con diferentes tamaños de palabra:** el número de bits en una palabra es un parámetro del RC5; diferentes longitudes de palabra producen algoritmos diferentes.
- **Número variable de etapas:** el número de etapas es un segundo parámetro. Esto permite alcanzar un compromiso entre mayor rapidez y mayor seguridad.

- **Longitud de clave variable:** la longitud de la clave es un tercer parámetro. Otra vez, posibilita un acuerdo entre velocidad y seguridad.
- **Simple:** la estructura simple del RC5 es fácil de implementar y facilita la tarea de determinar la robustez del algoritmo.
- **Bajo consumo de memoria:** la poca necesidad de memoria hace que el RC5 sea adecuado para tarjetas inteligentes y otros dispositivos con restricciones de memoria.
- **Alta seguridad:** proporciona alta seguridad con los parámetros adecuados.
- **Rotaciones dependientes de los datos:** incorpora rotaciones (desplazamientos circulares de bits) cuya cantidad depende de los datos. Esto parece fortalecer el algoritmo contra el criptoanálisis.

El RC5 se usa en varios productos de la RSA Data Security, Inc.

2.3 MODOS DE OPERACIÓN DEL CIFRADO DE BLOQUES

Un cifrador simétrico de bloque procesa un bloque de datos cada vez. En el caso del DES y el 3DES, la longitud del bloque es de 64 bits. Para cantidades mayores de texto claro, es necesario trocearlo en bloques de 64 bits (rellenando el último bloque si fuera necesario). La forma más sencilla de proceder es la que se conoce como modo ECB (*electronic codebook*), en el cual se manejan 64 bits cada vez y cada bloque de texto claro se cifra con la misma clave. Se usa el término *codebook* (libro de códigos) porque para una clave dada hay un único texto cifrado por cada bloque de 64 bits de texto claro. Por eso, nos podemos imaginar un libro de códigos gigantesco en el que hay una entrada para cada posible patrón de texto claro de 64 bits junto con su correspondiente texto cifrado.

Con el ECB, si el mismo bloque de texto claro de 64 bits aparece más de una vez en el mensaje, éste siempre produce el mismo texto cifrado. Por este motivo, para mensajes largos, el modo ECB puede no ser seguro. Si el mensaje está muy estructurado, un criptoanalista podría explotar esas regularidades. Por ejemplo, si se sabe que el mensaje comienza siempre con ciertos campos predefinidos, entonces el criptoanalista podría conocer un buen número de pares de texto claro-texto cifrado con los que trabajar. Si el mensaje contiene elementos repetitivos, con un período de repetición múltiplo de 64 bits, entonces el criptoanalista puede identificar dichos elementos, lo cual puede ayudarle en el análisis u ofrecerle la oportunidad de sustituir o reorganizar bloques.

Para superar las deficiencias de seguridad del ECB, necesitaríamos una técnica en la que el mismo bloque de texto claro, si se repite, produzca diferentes bloques de texto cifrado. En esta sección veremos dos alternativas comunes, definidas en FIPS PUB 81.

MODO CBC (CIPHER BLOCK CHAINING)

En el modo CBC (*Cipher Block Chaining*) (Figura 2.7), la entrada al algoritmo de cifrado es el XOR entre el bloque de texto claro actual y el bloque de texto cifrado anterior; se usa la misma clave para cada bloque. Se ha encadenado el procesamiento de la secuencia de bloques de texto claro. La entrada a la función de cifrado para cada bloque

de texto claro no mantiene una relación fija con dicho bloque. Por lo tanto, no se exponen los patrones de 64 bits que se repiten.

Para descifrar cada bloque de cifrado se pasa por el algoritmo de descifrado. Con el resultado y el bloque texto cifrado precedente se hace un XOR para obtener el bloque de texto claro. Para comprobar que funciona, se puede escribir

$$C_i = E_K[C_{i-1} \oplus P_i]$$

Donde $E_K[X]$ es el cifrado del texto claro X usando la clave K , y \oplus es la operación OR exclusivo. Entonces

$$D_K[C_i] = D_K[E_K(C_{i-1} \oplus P_i)]$$

$$D_K[C_i] = (C_{i-1} \oplus P_i)$$

$$C_{i-1} \oplus D_K[C_i] = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

lo que verifica la Figura 2.7b.

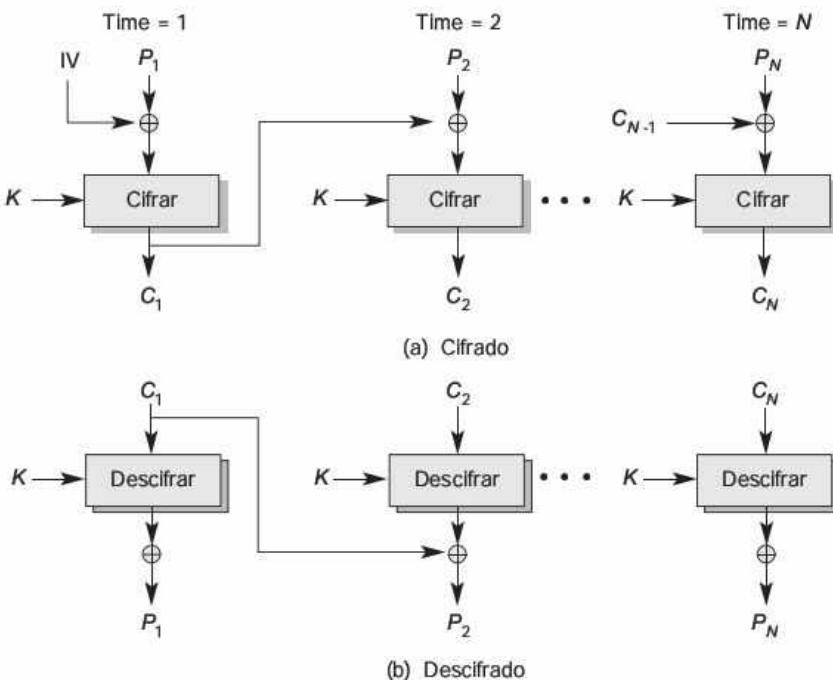


Figura 2.7 Modo CBC

Para producir el primer bloque de texto cifrado se realiza un XOR de un vector de inicialización (IV, *Initialization Vector*) con el primer bloque de texto claro. Al descifrar, se hace un XOR de IV con la salida del algoritmo de descifrado para recuperar el primer bloque de texto claro.

El IV debe ser conocido por el emisor y el receptor. Para máxima seguridad, el IV debería ser protegido, al igual que la clave. Esto podría hacerse enviando el IV utilizando cifrado ECB. Una razón para proteger el IV es la siguiente: si un atacante puede engañar al receptor para que use un valor diferente del IV, entonces el atacante tiene la posibilidad de invertir determinados bits del primer bloque de texto claro. Para verlo, considérese lo siguiente:

$$\begin{aligned} C_1 &= E_K(\text{IV} \oplus P_1) \\ P_1 &= \text{IV} \oplus D_K(C_1) \end{aligned}$$

Ahora usando la notación de que $X[j]$ representa el j -ésimo bit de los 64 bits de X . Entonces

$$P_1[j] = \text{IV}[j] \oplus D_K(C_1)[j]$$

Usando las propiedades de XOR se puede establecer

$$P_1[j]' = \text{IV}'[j] \oplus D_K(C_1)[j]$$

donde la notación prima indica el complemento de bit. Esto significa que si un atacante tiene la posibilidad de cambiar bits en IV, los bits correspondientes del valor de P_1 recibido también se pueden cambiar.

El modo CBC se utiliza en aplicaciones de seguridad, como se verá en la Segunda Parte.

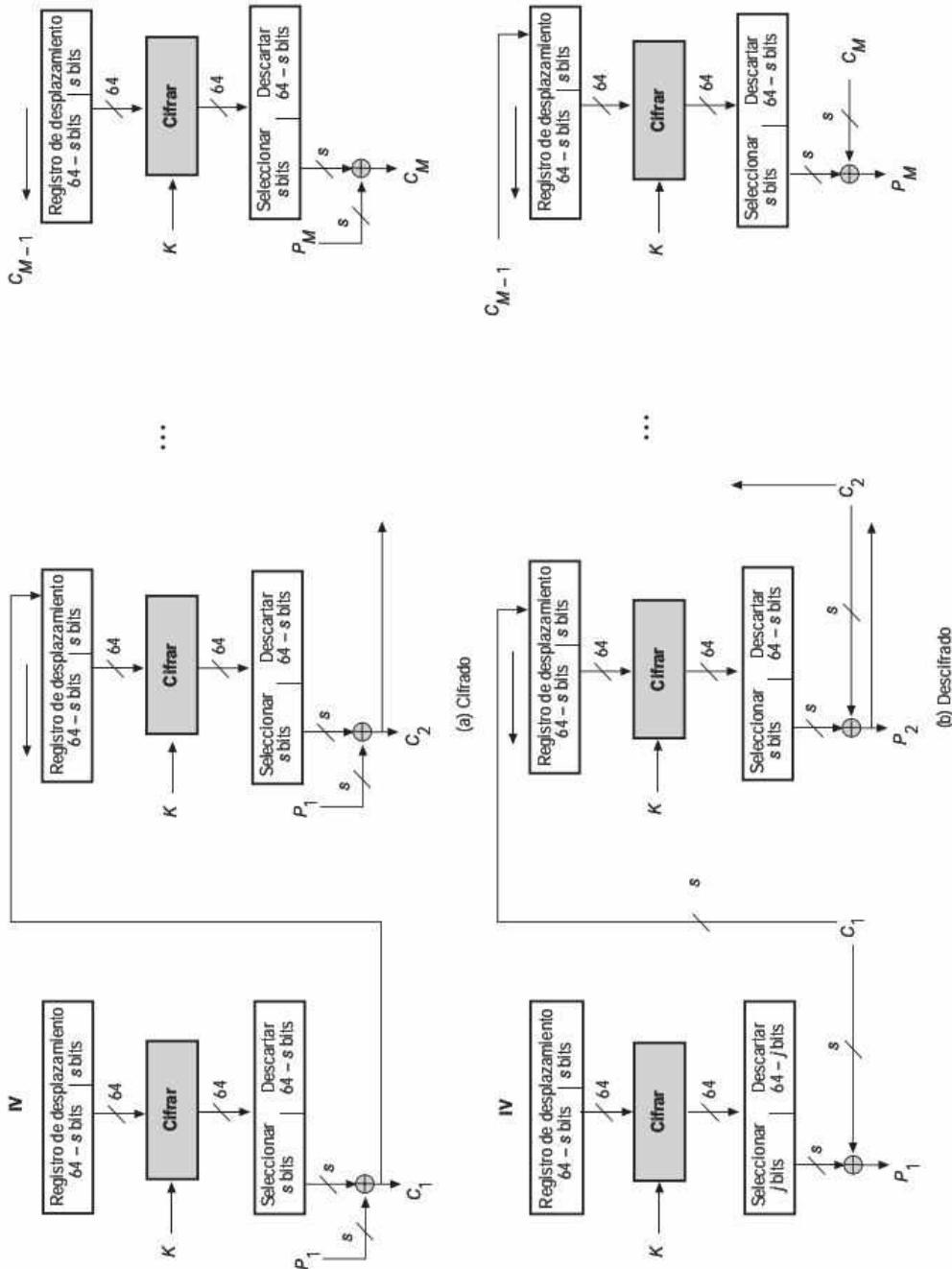
MODO CFB (CIPHER FEEDBACK)

Es posible convertir cualquier cifrador de bloque en un cifrador de flujo usando el modo de realimentación de cifrado (CFB). Un cifrador de flujo elimina la necesidad de completar el último bloque de un mensaje para que tenga la longitud requerida. También puede operar en tiempo real. Así, si se está transmitiendo una ristra de caracteres, se puede cifrar y transmitir inmediatamente cada carácter usando un cifrador de flujo orientado al mismo.

Una propiedad deseable de un cifrador de ristra es que el texto cifrado sea del mismo tamaño que el texto claro. Así, si se están transmitiendo caracteres de ocho bits, los caracteres deberían cifrarse usando ocho bits. Si se usaran más de ocho bits se estaría malgastando capacidad de transmisión.

La Figura 2.8 muestra el esquema del CFB. Se asume que la unidad de transmisión es s bits; un valor común es $s = 8$. Con el CBC las unidades de texto claro se encadenan juntas, de manera que el texto cifrado de cualquier unidad de texto claro es función de todos los textos claros anteriores.

En primer lugar, consideremos el cifrado. La entrada a la función de cifrado es un registro de desplazamiento de 64 bits que al principio se inicializa con un vector (IV). A los s bits más a la izquierda (más significativos) de la salida de la función de cifrado se les aplica un XOR con la primera unidad de texto claro P_1 para producir la primera unidad de texto cifrado C_1 , que luego se transmite. Además, los contenidos del registro de desplazamiento se mueven s bits a la izquierda y se coloca C_1 en los s bits más a la derecha (menos significativos) del registro de desplazamiento. Este proceso continúa hasta que se hayan cifrado todas las unidades del texto claro.

Figura 2.8 Modo CFB de s bits

Para descifrar se usa el mismo esquema, excepto que a la unidad de texto cifrado recibido se aplica el XOR con la salida de la función de cifrado para producir la unidad de texto claro. Nótese que se usa la función de *cifrado* no la de descifrado. Esto es fácil de explicar. Sea $S_S(X)$ los s bits más significativos de X . Entonces

$$C_1 = P_1 \oplus S_S(E(\text{IV}))$$

Por lo tanto

$$P_1 = C_1 \oplus S_S(E(\text{IV}))$$

El razonamiento es el mismo para los subsiguientes pasos del proceso.

2.4 UBICACIÓN DE LOS DISPOSITIVOS DE CIFRADO

El enfoque más potente y usual para contrarrestar las amenazas a la seguridad en la red es el cifrado. Al usar cifrado se debe decidir qué cifrar y dónde situar los engranajes de cifrado. Hay dos alternativas fundamentales: cifrado en enlace y cifrado extremo a extremo; en la Figura 2.9 se ilustra su uso sobre una red de conmutación de paquetes.

Con cifrado en enlace, cada enlace de comunicaciones vulnerable es equipado con un dispositivo de cifrado en ambos extremos. De esta manera, todo el tráfico sobre los enlaces de comunicaciones es seguro. Aunque en una red grande se necesitaría gran cantidad de estos dispositivos, proporciona un alto grado de seguridad. Una desventaja de este enfoque es que el mensaje debe descifrarse cada vez que introduce el conmutador de un paquete; esto es necesario porque el conmutador debe leer la dirección (número de circuito virtual) en la cabecera del paquete para decidir su ruta. Por eso, el mensaje es vulnerable en cada conmutador. Si la red de conmutación de paquetes es pública, el usuario no tiene control sobre la seguridad en los nodos.

Con el cifrado extremo a extremo, el proceso se realiza en los dos sistemas finales. El *host* o terminal fuente cifra los datos. Éstos se transmiten cifrados a través de la red hacia el terminal o *host* de destino sin ser alterados. El destino y la fuente comparten una clave y por tanto el primero es capaz de descifrar los datos. Este enfoque parece que aseguraría la transmisión contra ataques en los enlaces de comunicación o conmutadores. Pero todavía hay una debilidad.

Considérese la siguiente situación. Un *host* se conecta a una red X.25 de conmutación de paquetes, establece un circuito virtual con otro *host*, y está preparado para transmitir datos a ese otro *host* usando cifrado extremo a extremo. Los datos se transmiten por dicha red en forma de paquetes, que consisten en una cabecera y algunos datos de usuario. ¿Qué parte de cada paquete cifrará el *host*? Supongamos que cifra el paquete entero, incluyendo la cabecera. Esto no funcionaría porque, recuérdese, sólo el otro *host* puede realizar el descifrado. El nodo de conmutación de paquetes recibirá un paquete cifrado y será incapaz de leer la cabecera. Por tanto, no será capaz de encaminarlo. El *host* debe cifrar solamente la porción de datos de usuario y debe dejar la cabecera del paquete en claro, por lo que ésta puede ser leída por la red.

Así, con cifrado extremo a extremo, los datos de usuario están seguros. No obstante, el patrón de tráfico no lo está porque las cabeceras de los paquetes se transmiten en claro. Para conseguir mayor seguridad se necesitan ambos cifrados como se muestra en la Figura 2.9.

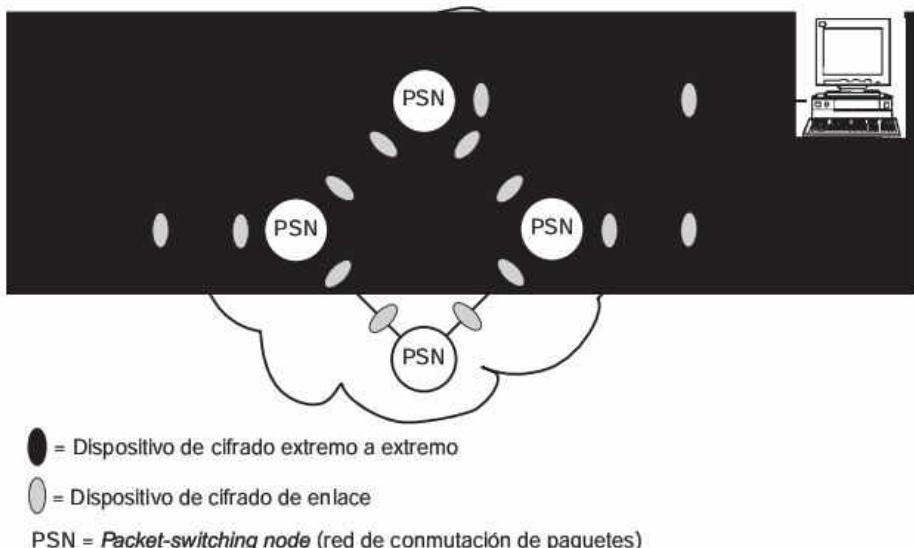


Figura 2.9 Cifrado a través de una red de comutación de paquetes

Resumiendo, cuando se emplean ambas formas, el *host* cifra la parte de datos de usuario de un paquete usando una clave de cifrado extremo a extremo. Entonces se cifra el paquete entero usando una clave de cifrado de enlace. A medida que el paquete recorre la red, cada comutador descifra el paquete usando la clave de cifrado de enlace para leer la cabecera y entonces cifra otra vez el paquete entero para enviarlo al próximo enlace. Ahora el paquete entero está seguro, excepto durante el tiempo en que está en la memoria de un comutador de paquetes, momento en el que la cabecera está en claro.

2.5 DISTRIBUCIÓN DE CLAVES

Para que el cifrado simétrico funcione, las dos partes deben tener la misma clave para realizar un intercambio seguro, y esa clave debe protegerse del acceso de otros. Más aún, es deseable cambiar frecuentemente la clave para limitar la cantidad de datos comprometidos si un atacante la descubre. Por lo tanto, la robustez de un sistema criptográfico depende de la técnica de distribución de claves, término que se refiere a la manera de entregar una clave a dos partes que desean intercambiar datos, sin permitir que otros vean dicha clave. La distribución de claves se puede conseguir de diferentes maneras. Para dos partes *A* y *B*,

1. Una clave podría ser elegida por *A* y entregada físicamente a *B*.
2. Una tercera parte podría elegir la clave y entregarla físicamente a *A* y a *B*.
3. Si con anterioridad *A* y *B* han estado usando una clave, una parte podría transmitir la nueva clave a la otra cifrada usando la antigua.
4. Si *A* y *B* disponen de una conexión cifrada a una tercera parte *C*, *C* podría distribuir mediante los enlaces cifrados una clave a *A* y a *B*.

Las opciones 1 y 2 implican la entrega manual de una clave. Para el cifrado de enlace es un requisito razonable, porque cada dispositivo de cifrado de enlace solamente intercambia datos con su interlocutor en el otro lado del enlace. Sin embargo, para cifrado extremo a extremo la entrega manual es difícil. En un sistema distribuido, cualquier *host* o terminal podría necesitar demasiado tiempo en ocuparse de los intercambios de clave con muchos otros *hosts* o terminales. Por eso, cada dispositivo necesita un número de claves, suministradas de forma dinámica. El problema es especialmente difícil en un sistema distribuido de área ancha.

La opción 3 es una posibilidad tanto para cifrado de enlace como para cifrado extremo a extremo, pero si un atacante consiguiera acceso a una clave, se revelarían todas las subsiguientes. Aunque se realizaran cambios frecuentes de las claves de cifrado de enlace, deberían hacerse manualmente. Para suministrar claves para el cifrado extremo a extremo, es preferible la opción 4.

La Figura 2.10 ilustra una implementación que satisface la opción 4 para cifrado extremo a extremo. En la figura se ignora el cifrado de enlace. Éste se puede añadir, o no, según se requiera. Para este esquema se identifican dos tipos de claves:

- **Clave de sesión:** cuando dos sistemas finales (*hosts*, terminales, etc.) desean comunicarse, establecen una conexión lógica (por ejemplo, circuito virtual). Durante el tiempo que está activa esa conexión lógica, todos los datos de usuario se cifran con una clave de sesión, válida sólo para esa conexión. Al finalizar la conexión o sesión, la clave es destruida.
- **Clave permanente:** una clave permanente es una clave usada entre entidades con el propósito de distribuir claves de sesión.

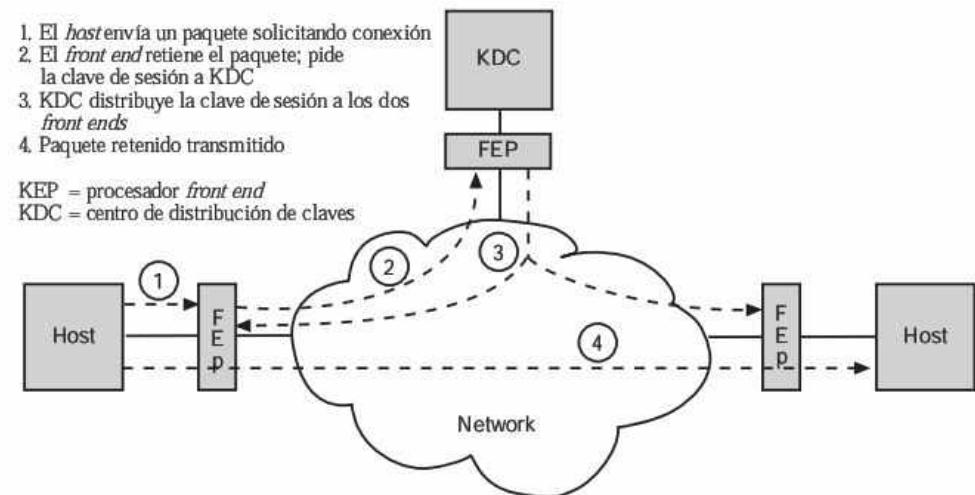


Figura 2.10 Distribución automática de clave para protocolo orientado a conexión

La configuración consiste en los siguientes elementos:

- **KDC (Key Distribution Center):** el centro de distribución de claves determina a qué sistemas se les permite comunicarse entre sí. Cuando se obtiene permiso para que dos sistemas establezcan una conexión, el centro de distribución de claves proporciona una clave de sesión válida solamente para esa conexión.
- **FEP (Front-End Processor):** el procesador *front-end* realiza cifrado extremo a extremo y obtiene claves de sesión en nombre de su *host* o terminal.

En la Figura 2.10 se muestran los pasos necesarios para el establecimiento de una conexión. Cuando un *host* desea iniciar una conexión con otro, transmite un paquete de solicitud de conexión (paso 1). El procesador *front-end* almacena el paquete y solicita al KDC permiso para establecer la conexión (paso 2). La comunicación entre el FEP y el KDC se cifra usando una clave maestra compartida solamente por ambos. Si el KDC aprueba la solicitud de conexión, entonces genera la clave de sesión y la envía a los dos procesadores *front-end* implicados, usando una única clave permanente para cada *front-end* (paso 3). El procesador *front-end* solicitante puede ahora liberar el paquete de solicitud de conexión, estableciéndose así la conexión entre los dos sistemas (paso 4). Todos los datos de usuario intercambiados entre los dos sistemas se cifran en los procesadores *front-end* respectivos usando la clave de sesión de un solo uso.

El enfoque de distribución automatizada de claves proporciona las características de flexibilidad y dinamismo necesarias para permitir que una serie de usuarios de terminal accedan a una serie de *hosts* y para que los *hosts* intercambien datos entre sí.

Otro enfoque para la distribución de clave utiliza el cifrado de clave pública, que se trata en el Capítulo 3.

2.6 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

Los temas tratados en este Capítulo se pueden ver con mayor detalle en [STAL03]. Para cubrir los algoritmos criptográficos [SCHN96] es un libro de referencia fundamental; contiene la descripción de prácticamente todos los algoritmos y protocolos criptográficos publicados en los últimos 15 años. Otro estudio valioso y detallado es [MENE97]. Un tratamiento más profundo, con discusiones matemáticas rigurosas se encuentra en [STIN02].

- MENE97** Menezes, A., Oorschot, P.; y Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
- SCHN96** Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.
- STAL03** Stallings, W. *Cryptography and Network Security: Principles and Practice, 3rd Edition*. Upper Saddle River, NJ: Prentice Hall, 2003.
- STIN02** Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

Sitios web recomendados:

- **Página web de AES:** página del NIST sobre el AES. Contiene el estándar y una serie de documentos relevantes.

- **Página de Rijndael:** mantenida por los desarrolladores del Rijndael. Contiene documentación, enlaces a implementaciones y otros enlaces de interés.

2.7 PALABRAS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

PALABRAS CLAVE

AES (<i>Advanced Encryption Standard</i>)	cifrado de Feistel	descifrado
ataque de fuerza bruta	cifrado extremo a extremo	distribución de clave
CBC (modo de encadenamiento de bloque cifrado)	cifrado simétrico	modo ECB (<i>Electronic Code Book</i>)
CFB (modo de retroalimentación de cifrado)	cifrador de bloque	subclave
texto cifrado	cifrador de flujo	texto claro
cifrado de enlace	clave de sesión	triple DES (3DES)
	criptoanálisis	
	criptografía	
	DES (<i>Data Encryption Standard</i>)	

PREGUNTAS DE REPASO

- 2.1. ¿Cuáles son los componentes esenciales de un cifrador simétrico?
- 2.2. ¿Cuáles son las dos funciones básicas usadas en los algoritmos de cifrado?
- 2.3. ¿Cuántas claves se necesitan para que dos personas se comuniquen usando un cifrador simétrico?
- 2.4. ¿Cuál es la diferencia entre un cifrador de bloque y un cifrador de flujo?
- 2.5. ¿Cuáles son los dos enfoques generales para atacar a un cifrador?
- 2.6. ¿Por qué algunos modos de operación de cifrado de bloque usan solamente cifrado mientras que otros utilizan tanto cifrado como descifrado?
- 2.7. ¿Qué es triple cifrado?
- 2.8. ¿Por qué la parte intermedia del 3DES es un descifrado en vez de un cifrado?
- 2.9. ¿Cuál es la diferencia entre cifrado de enlace y cifrado extremo a extremo?
- 2.10. Enumera las formas en que se pueden distribuir las claves secretas a dos participantes en una comunicación.
- 2.11. ¿En qué se diferencian la clave de sesión y la clave maestra?
- 2.12. ¿Qué es un centro de distribución de claves?

PROBLEMAS

- 2.1. Demuestra que el descifrado de Feistel es la inversa del cifrado de Feistel.
- 2.2. Con el modo ECB, si hay un error en un bloque del texto cifrado transmitido, solamente afecta al bloque de texto claro correspondiente. Sin embargo, en el modo CBC, este error se propaga. Por ejemplo, un error al transmitir C_1 (Figura 2.7) obviamente afecta a P_1 y P_2 .

- a) ¿Se ven afectados los bloques siguientes a P_2 ?
- b) Supongamos que hay un bit de error en la versión fuente de P_1 . ¿A través de cuántos bloques de texto cifrado se propaga este error? ¿Qué efecto produce en el receptor?
- 2.3 Si se produce un error de un bit en la transmisión de un carácter del texto cifrado en el modo CFB de ocho bits, ¿hasta dónde se propaga el error?
- 2.4 Los esquemas de distribución de clave que usan un centro de control de acceso y/o centro de distribución de clave tienen puntos centrales vulnerables a los ataques. Discute lo que dicha centralización implica para la seguridad.
- 2.5 Supongamos que alguien sugiere la siguiente manera de confirmar que dos de vosotros están en posesión de la misma clave secreta. Crea una ristra aleatoria de bits del tamaño de la clave, aplícale el XOR con la clave y envía el resultado por el canal. Tu receptor hace XOR del bloque entrante con la clave (que debería ser la misma que la tuya) y envía el resultado de vuelta. Observas lo que te devuelve y si coincide con tu ristra aleatoria, entonces has verificado que tu receptor tiene la misma clave secreta que tú, aunque ninguno haya transmitido la clave. ¿Hay algún defecto en este esquema?

CAPÍTULO 3

Criptografía de clave pública y autentificación de mensajes

- 3.1. Enfoques para la autentificación de mensajes**
 - Autentificación mediante cifrado convencional
 - Autentificación de mensajes sin cifrado
- 3.2. Funciones *hash* seguras y HMAC**
 - Requisitos de las funciones *hash*
 - Funciones *hash* simples
 - La función *hash* segura SHA-1
 - Otras funciones *hash* seguras
 - HMAC
- 3.3. Principios de criptografía de clave pública**
 - Estructura del cifrado de clave pública
 - Aplicaciones para criptosistemas de clave pública
 - Requisitos para la criptografía de clave pública
- 3.4. Algoritmos de criptografía de clave pública**
 - El algoritmo de cifrado de clave pública RSA
 - Intercambio de clave Diffie-Hellman
 - Otros algoritmos criptográficos de clave pública
- 3.5. Firmas digitales**
- 3.6. Gestión de claves**
 - Certificados de clave pública
 - Distribución de claves secretas mediante criptografía de clave pública
- 3.7. Bibliografía y sitios web recomendados**
- 3.8. Términos clave, preguntas de repaso y problemas**
 - Términos clave
 - Preguntas de repaso
 - Problemas

Cada egipcio recibía dos nombres, el verdadero y el bueno, o el grande y el pequeño, respectivamente; y mientras el nombre bueno o pequeño se hacía público, el verdadero o grande parece haber quedado cuidadosamente oculto.

La rana dorada, SIR JAMES GEORGE FRAZER

Protegerse de la influencia funesta que ejercen los extraños es, por lo tanto, un precepto elemental de la prudencia salvaje. Así, antes de permitir que los extraños entren en un distrito, o al menos antes de que se les permita mezclarse libremente con sus habitantes, a menudo los nativos del lugar celebran ciertas ceremonias para desarmar a los extraños de sus poderes mágicos, o desinfectar, de alguna forma, la atmósfera corrompida de la que se les cree rodeados.

La rana dorada, SIR JAMES GEORGE FRAZER

A demás de la confidencialidad de los mensajes, la autentificación es una función importante de la seguridad de las redes de computadores. Este Capítulo analiza tres aspectos de la autentificación de mensajes: en primer lugar, se centra en el uso de los códigos de autentificación de mensajes y en las funciones *hash* para proporcionar la autentificación; a continuación, se tratan los principios del cifrado de clave pública y dos algoritmos específicos de clave pública que son útiles en el intercambio de claves de cifrado convencional; luego, se observará el uso del cifrado público para producir firmas digitales, lo que da lugar a una forma mejorada de autentificación de mensajes. Por último, volveremos al tema de la gestión de claves.

3.1 ENFOQUES PARA LA AUTENTIFICACIÓN DE MENSAJES

El cifrado protege de los ataques pasivos (escuchas). Un requisito diferente es el de proteger de los ataques activos (falsificación de datos y transacciones). La protección contra tales ataques se conoce como autentificación de mensajes.

Se dice que un mensaje, archivo, documento o cualquier otro grupo de datos es auténtico cuando es genuino y procede de la fuente original. La autentificación de mensajes es un procedimiento que permite la comunicación entre las partes para verificar que los mensajes recibidos son auténticos. Los dos aspectos importantes son verificar que los contenidos del mensaje no han sido alterados y que la fuente es auténtica. También podríamos querer verificar si un mensaje no ha sido retrasado ni repetido de forma artificial y conocer la secuencia relativa a otros mensajes que fluyen entre dos partes.

AUTENTIFICACIÓN MEDIANTE CIFRADO CONVENCIONAL

Es posible llevar a cabo la autentificación simplemente mediante el uso de cifrado convencional. Si sólo el emisor y el receptor comparten una clave (como debería ser), sólo el auténtico emisor sería capaz de cifrar con éxito un mensaje para el otro participante. Ade-

más, si el mensaje incluye un código de detección de errores y un número de secuencia, el receptor estará seguro de que no se han producido alteraciones y de que la secuencia es adecuada. Si el mensaje también incluye un sello de tiempo, el receptor estará seguro de que el mensaje no se ha retrasado más de lo habitual para el tránsito en la red.

AUTENTIFICACIÓN DE MENSAJES SIN CIFRADO

En esta sección se examinan diferentes enfoques para la autentificación de mensajes que no se basan en el cifrado. En todos estos enfoques se genera y añade una referencia de autentificación a cada mensaje para su transmisión. El mensaje en sí mismo no está cifrado y se puede leer en el destino independiente de la función de autentificación en el destino.

Debido a que los enfoques que se tratan en esta sección no cifran el mensaje, no se proporciona la confidencialidad. Si el cifrado convencional aporta autentificación y su uso se ha extendido con productos ya disponibles, ¿por qué no usar tal enfoque, que proporciona confidencialidad y autentificación? [DAVI89] presenta tres situaciones en las que es preferible la autentificación de mensajes sin confidencialidad:

1. Hay una serie de aplicaciones mediante las cuales el mismo mensaje se emite a un número de destinos, como sería el caso de un mensaje que notifica a los usuarios que la red no está disponible o una señal de alarma en un centro de control. Es más barato y fiable tener un solo destino responsable de supervisar la autenticidad. Así, el mensaje debe emitirse en texto claro con una referencia asociada de autentificación de mensaje. El sistema responsable realiza la autentificación y si se produce una violación, los otros sistemas de destino son alertados por una alarma general.
2. Otro posible escenario es un intercambio en el que una parte tiene una gran carga y no puede permitirse el tiempo necesario para descifrar todos los mensajes que entran. La autentificación se lleva a cabo de forma selectiva, eligiéndose los mensajes de forma aleatoria para su comprobación.
3. La autentificación de un programa informático en texto claro es un servicio atractivo. El programa se puede ejecutar sin necesidad de descifrarlo cada vez, hecho que implicaría un mal uso de los recursos del procesador. Sin embargo, si se añadiera al programa una referencia de autentificación de un mensaje, se podría comprobar cuando se necesite garantía de la integridad del programa.

De este modo, tanto la autentificación como el cifrado intervienen en la tarea de cubrir las necesidades de seguridad.

CÓDIGO DE AUTENTIFICACIÓN DE MENSAJES

Una técnica de autentificación implica el uso de una clave secreta para generar un bloque de datos pequeño, conocido como código de autentificación del mensaje, que se añade al mensaje. Esta técnica implica que dos partes que se comunican, A y B, comparten una

clave secreta común, K_{AB} . Cuando A tiene un mensaje que enviar a B, calcula el código de autentificación como una función del mensaje y la clave: $\text{MAC}_M = F(K_{AB}, M)$. El mensaje y el código se transmiten al receptor deseado. El receptor realiza los mismos cálculos en el mensaje recibido, usando la misma clave secreta para generar un nuevo código de autentificación del mensaje. El código recibido se compara con el código calculado (Figura 3.1). Si sólo el receptor y el emisor conocen la identidad de la clave secreta, y si el código se corresponde con el código calculado, entonces:

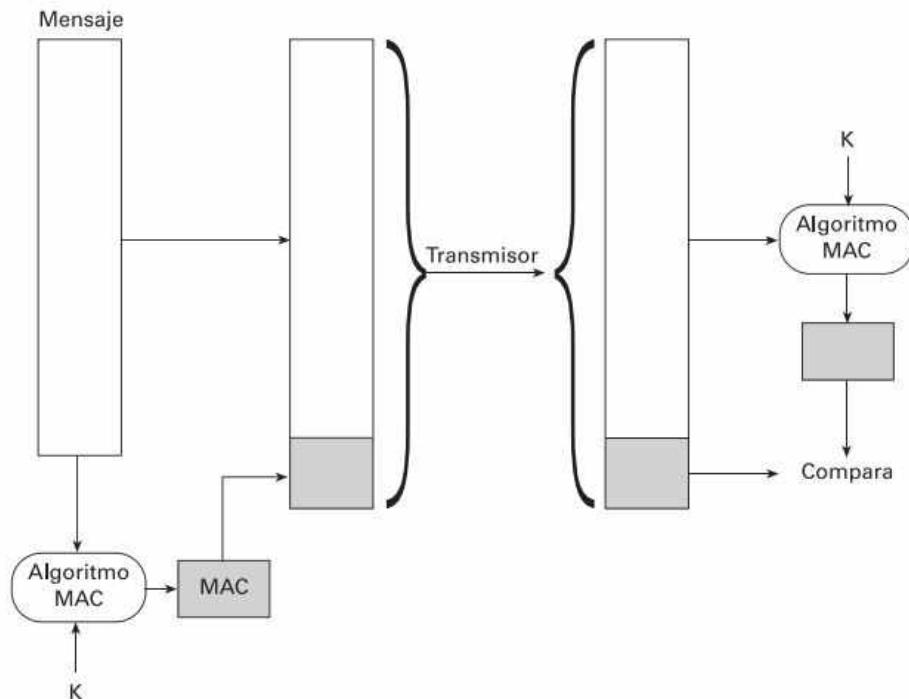


Figura 3.1 Autenticación de mensaje mediante código de autentificación de mensajes (MAC)

1. El receptor está seguro de que el mensaje no ha sido alterado. Si un oponente altera el mensaje pero no altera el código, el cálculo del código del receptor será distinto al del código recibido. Como se supone que el oponente no conoce la clave secreta, no puede alterar el código para hacerlo corresponderse con las alteraciones realizadas en el mensaje.
2. El receptor está seguro de que el mensaje es del emisor indicado. Como nadie más conoce la clave, nadie más podría crear un mensaje con un código adecuado.
3. Si el mensaje incluye un número de secuencia (como el que se usa con X.25, HDLC y TCP), el receptor puede estar seguro de la secuencia apropiada porque un oponente no puede alterar con éxito el número de secuencia.

Se podría emplear una serie de algoritmos para generar el código. La especificación NIST, FIPS PUB 113 recomienda el uso del DES. El DES se emplea para generar una

versión cifrada del mensaje, y el último número de bits de texto cifrado se usa como el código. Un código de 16 o 32 bits es bastante común.

El proceso que se acaba de describir es similar al cifrado. Una diferencia se halla en que el algoritmo de autentificación no necesita ser reversible, al contrario que en el cifrado. Por lo tanto, debido a las propiedades matemáticas de la función de autentificación, es menos vulnerable que el cifrado.

FUNCIÓN HASH UNIDIRECCIONAL

Una alternativa al código de autentificación de mensajes es la función *hash* unidireccional. Al igual que con el código de autentificación de mensajes, una función *hash* acepta un mensaje de tamaño variable, M , como entrada y produce un resumen del mensaje de tamaño fijo $H(M)$ como salida. A diferencia del MAC, una función *hash* no acepta una clave secreta como entrada. Para autenticar un mensaje, el resumen se envía con el mensaje, con lo cual se verifica la autenticidad del resumen.

La Figura 3.2 ilustra tres formas de autenticar un mensaje. El resumen del mensaje puede ser cifrado por medio de cifrado convencional (parte a); si sólo el emisor y el receptor comparten la clave de cifrado, la autentificación está garantizada. El mensaje también puede ser cifrado por medio de cifrado de clave pública (parte b); esto queda explicado en la sección 3.5. El enfoque de clave pública tiene dos ventajas: proporciona una firma digital además de autentificación del mensaje y no necesita la distribución de claves a las partes que se comunican entre sí.

La ventaja que tienen estos dos enfoques sobre los enfoques que cifran el mensaje completo es que implican un coste computacional menor. Sin embargo, ha habido interés por el desarrollo de una técnica que evite el cifrado. [TSUD92] muestra algunas de las razones por las que existe este interés:

- El *software* de cifrado es muy lento. Aunque la cantidad de datos que han de cifrarse por cada mensaje sea pequeña, podría haber un flujo fijo de mensajes entrando y saliendo del sistema.
- Los costes de *hardware* de cifrado son muy elevados. Aunque existen las implementaciones de chips de bajo coste del DES, si todos los nodos de una red deben tener esta capacidad, se añaden costes.
- El *hardware* de cifrado se optimiza con grandes cantidades de datos. Con bloques pequeños de datos, una gran proporción del tiempo se invierte en la inicialización.
- Los algoritmos de cifrado pueden estar patentados. Algunos, como el algoritmo de clave pública RSA, están patentados y deben tener licencia, lo cual añade un coste.
- Los algoritmos de cifrado pueden estar sujetos a controles de exportación, como ocurre con el DES.

La Figura 3.2c muestra una técnica que emplea una función *hash* y no usa cifrado para la autentificación de un mensaje. Esta técnica implica que dos partes que se comunican, A y B, comparten un valor secreto común S_{AB} . Cuando A tiene un mensaje que

enviar a B, calcula la función *hash* de la concatenación del valor secreto y el mensaje: $MD_M = H(S_{AB} \parallel M)$ ¹. Luego envía $[M \parallel MD_M]$ a B. Como B posee S_{AB} , puede volver a calcular $H(S_{AB} \parallel M)$ y verificar MD_M . Como la clave secreta no se ha enviado, no es posible que un oponente modifique un mensaje interceptado. Mientras el valor secreto permanezca como tal, tampoco es posible que un oponente genere un nuevo mensaje.

Una variación de la tercera técnica, llamada HMAC, es la que se ha adoptado para la seguridad IP (descrita en el Capítulo 6), que también ha sido especificada para SNMPv3 (Capítulo 8).

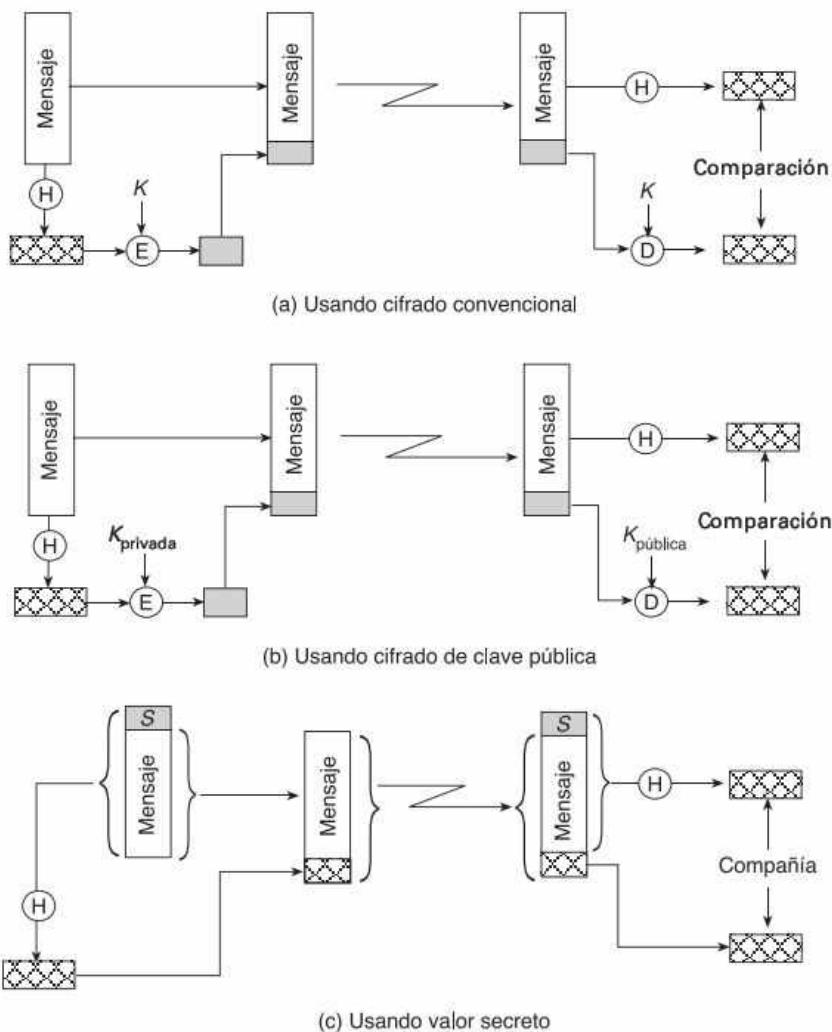


Figura 3.2 Autenticación de mensaje por medio de una función *hash* unidireccional

¹ \parallel indica concatenación.

3.2 FUNCIONES HASH SEGURAS Y HMAC

La función *hash* unidireccional o función hash segura no sólo es importante para la autentificación de mensajes sino también para las firmas digitales. En esta sección, empezamos analizando los requisitos de una función *hash* segura. A continuación se tratan las funciones *hash* más importantes, las SHA-1.

REQUISITOS DE LAS FUNCIONES HASH

La finalidad de una función *hash* es la de obtener una «huella» de un archivo, mensaje u otro bloque de datos. Para que resulte útil a la autentificación de mensajes, una función *hash* H debe poseer las siguientes propiedades:

1. H puede aplicarse a un bloque de datos de cualquier tamaño.
2. H produce una salida de tamaño fijo.
3. $H(x)$ es relativamente fácil de computar para cualquier x dado, haciendo que tanto las implementaciones de *hardware* como de *software* sean prácticas.
4. Para cualquier valor h dado, es imposible desde el punto de vista computacional encontrar x tal que $H(x) = h$, lo cual, con frecuencia, se conoce en la literatura como propiedad **unidireccional**.
5. Para cualquier bloque dado x , es imposible desde el punto de vista computacional, encontrar $y \neq x$ con $H(y) = H(x)$, lo que a veces se conoce como **resistencia débil a la colisión**.
6. Es imposible desde el punto de vista computacional encontrar un par (x, y) tal que $H(x) = H(y)$, lo que normalmente se conoce como **resistencia fuerte a la colisión**².

Las tres primeras propiedades son requisitos para la aplicación práctica de una función *hash* a la autentificación de mensajes. La cuarta propiedad es la propiedad «unidireccional»: dado un mensaje, es fácil generar un código, pero dado un código, es prácticamente imposible generar un mensaje. Esta propiedad es importante si la técnica de autentificación implica el uso de un valor secreto (Figura 3.2c). El valor secreto no se envía; sin embargo, si la función *hash* no es unidireccional, un oponente puede descubrir fácilmente el valor secreto: si el oponente puede observar o interceptar una transmisión, obtiene el mensaje M y el código *hash* $MD_M = H(S_{AB}/M)$. Entonces el oponente invierte la función *hash* para obtener $S_{AB}/M = H^{-1}(MD_M)$. Debido a que ahora el oponente tiene M y S_{AB}/M , no tiene importancia recuperar S_{AB} .

² Por desgracia, estos términos no se usan con consistencia. Los términos alternativos que se usan en la literatura al respecto incluyen *función hash unidireccional* (propiedades 4 y 5), *función hash resistente a la colisión* (propiedades 4, 5 y 6), *función hash unidireccional débil* (propiedades 4 y 5); *función hash unidireccional robusta* (propiedades 4, 5 y 6). Al leer la literatura al respecto, el lector debe tener cuidado para determinar con precisión el significado de los términos empleados.

La quinta propiedad garantiza que es imposible encontrar un mensaje alternativo con el mismo valor *hash* que un mensaje dado. Esto evita la falsificación cuando se usa un código *hash* cifrado (Figuras 3.2a y b). Si no se diera esta propiedad, un oponente conseguiría, primero, observar o interceptar un mensaje y su código *hash* cifrado; segundo, generar un código *hash* no cifrado desde el mensaje; y tercero, generar un mensaje alternativo con el mismo código *hash*.

Una función *hash* que cumpla las cinco primeras propiedades se denomina función *hash* débil. Si también posee la sexta propiedad, se denomina función *hash* robusta. La sexta propiedad protege de un tipo sofisticado de ataque conocido como ataque basado en la paradoja del cumpleaños.

Además de proporcionar autenticación, un resumen de un mensaje también proporciona integridad de los datos. Realiza la misma función que una secuencia de comprobación: si cualquier bit del mensaje se altera accidentalmente durante la transmisión, el resumen del mensaje dará error.

FUNCIONES HASH SIMPLES

Todas las funciones *hash* usan los siguientes principios generales. La entrada (mensaje, archivo, etc.) se ve como una secuencia de bloques de n bits. La entrada se procesa bloque a bloque de forma iterativa para producir una función *hash* de n bits.

Una de las funciones *hash* más simples es el OR exclusivo bit a bit (XOR) de cada bloque. Se puede expresar de la siguiente manera:

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

donde

- C_i = i -ésimo bit del código hash, $1 \leq i \leq n$
 m = número de bloques de n bits en la entrada
 b_{ij} = i -ésimo bit en el j -ésimo bloque
 \oplus = operación XOR

La Figura 3.3 ilustra esta operación; produce una paridad simple por cada posición de bit y se conoce como una comprobación de redundancia longitudinal. Es bastante efectivo para datos aleatorios como comprobación de la integridad de los datos. Cada valor *hash* de n bits es igualmente parecido. Así, la probabilidad de que un error en los datos resulte en un valor *hash* inalterable es 2^{-n} . Con datos formateados más predeciblemente, la función es menos efectiva. Por ejemplo, en la mayoría de los archivos de texto normales, el bit más significativo de cada octeto siempre es cero. Así, si se usa un valor *hash* de 128 bits en vez de una efectividad de 2^{-128} , la función *hash* en este tipo de datos tiene una efectividad de 2^{-112} .

Una forma sencilla de mejorar los hechos es realizar una rotación circular de un bit en el valor *hash* después de que se haya procesado cada bloque. El procedimiento se puede resumir de la siguiente manera:

1. Inicialmente, establecer el valor *hash* de n bits en cero.
2. Procesar cada bloque de datos sucesivo de n bits como sigue:
 - a) Rotar el valor *hash* actual a la izquierda en un bit.

b) Realizar la operación XOR del bloque con el valor *hash*.

El efecto que se produce es el de «aleatorizar» la entrada de forma más completa y salvar cualquier regularidad que se presente en la entrada.

	Bit 1	Bit 2	• • •	Bit <i>n</i>
Bloque 1	b_{11}	b_{21}		b_{n1}
Bloque 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
Bloque <i>m</i>	b_{1m}	b_{2m}		b_{nm}
Código hash	C_1	C_2		C_n

Figura 3.3 Función *hash* simple mediante XOR bit a bit

Aunque el segundo procedimiento constituye una medida aceptable para la integridad de los datos, es prácticamente inútil para su seguridad cuando un código *hash* cifrado se usa con un mensaje de texto claro, como en las Figuras 3.2a y b. Dado un mensaje, es fácil producir un nuevo mensaje que produzca ese código *hash*: sólo hay que preparar el mensaje alternativo deseado y luego añadir un bloque de *n* bits que obligue al nuevo mensaje y al bloque a producir el código *hash* deseado.

Aunque un XOR simple o un XOR rotado (RXOR) no es suficiente si sólo se cifra el código *hash*, todavía puede parecer que una función tan simple sería útil cuando se cifran tanto el mensaje como el código *hash*. Pero se debe tener cuidado. Una técnica propuesta originalmente por la Agencia Nacional de Estándares usaba el XOR simple aplicado a bloques de mensajes de 64 bits y luego un cifrado de todo el mensaje que usaba el modo de cadena de bloques de cifrado (CBC). El esquema se puede definir como sigue: dado un mensaje que consiste en una secuencia de bloques de 64 bits X_1, X_2, \dots, X_N , definir el código *hash* C como el XOR bloque a bloque o todos los bloques y añadir el código *hash* como bloque final:

$$C_i = X_{N+1} = X_1 \oplus X_2 \oplus \dots \oplus X_N$$

A continuación, cifrar el mensaje completo y el código *hash*, usando el modo CBC para producir el mensaje cifrado Y_1, Y_2, \dots, Y_{N+1} . [JUEN85] destaca varias formas de manipular el texto cifrado de este mensaje para que no sea detectado por el código *hash*. Por ejemplo, por la definición de CBC (Figura 2.7), tenemos

$$X_1 = IV \oplus D_K(Y_1)$$

$$X_i = Y_{i-1} \oplus D_K(Y_i)$$

$$X_{N+1} = Y_N \oplus D_K(Y_{N+1})$$

Pero X_{N+1} es el código *hash*:

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \dots \oplus X_N \\ &= (\text{IV} \oplus D_K(Y_1)) \oplus (Y_1 \oplus D_K(Y_2)) \oplus \dots \oplus (Y_{N-1} \oplus D_K(Y_N)) \end{aligned}$$

Debido a que se puede realizar el XOR con los términos de la ecuación anterior siguiendo cualquier orden, sucede que el código *hash* no cambiaría si los bloques de texto cifrado fueran permutados.

LA FUNCIÓN HASH SEGURA SHA-1

El algoritmo *hash* seguro (SHA) fue desarrollado por el Instituto Nacional de Estándares y Tecnología (NIST) y publicado en 1993 como un estándar federal de procesamiento de la información (FIPS PUB 180); una versión revisada, a la que normalmente se conoce como SHA-1, se publicó como FIPS PUB 180-1 en 1995.

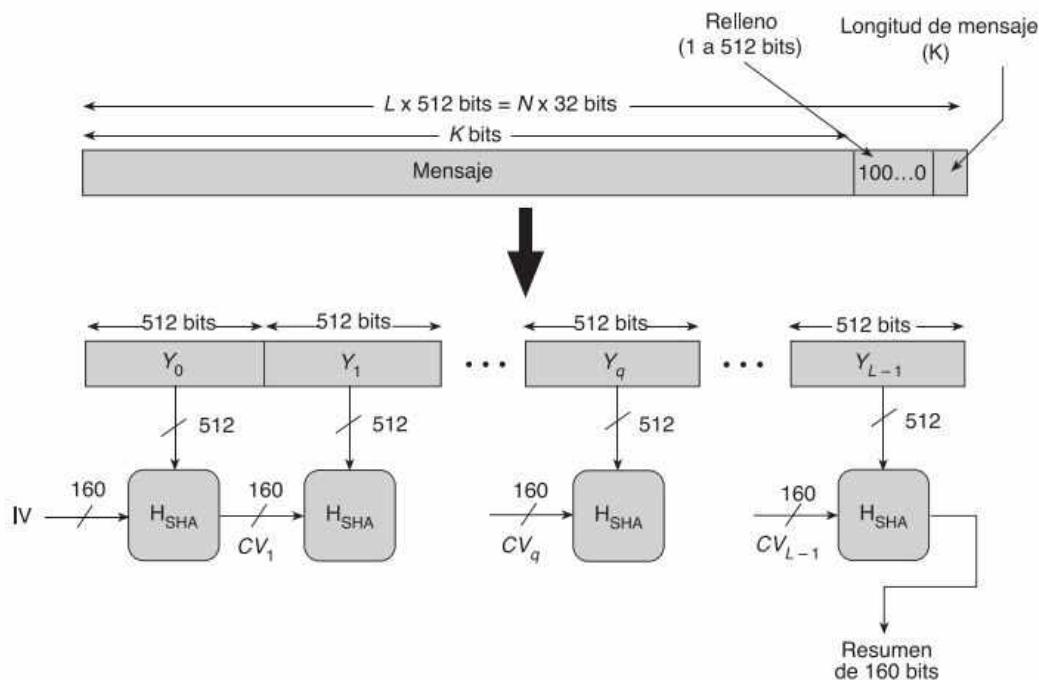


Figura 3.4 Generación del resumen de un mensaje usando SHA-1

El algoritmo toma como entrada un mensaje con una longitud máxima menor que 2^{64} bits y produce como salida un resumen de mensaje de 160 bits. La entrada se procesa en bloques de 512 bits. La Figura 3.4 muestra el procesamiento general de un mensaje para producir un resumen. El procesamiento consiste en los siguientes pasos:

Paso 1: Añadir bits de relleno. El mensaje se rellena para que su longitud sea congruente con 448 módulo 512 (longitud = 448 mod 512). Es decir, la longitud del mensaje relleno es 64 bits menor que un múltiplo de 512 bits. El relleno se añade aunque el mensaje ya tenga la longitud deseada. Así, el número de bits de relleno se encuentra entre 1 y 512. El relleno está formado por un único bit 1 seguido del número necesario de bits 0.

Paso 2: Añadir longitud. Se añade un bloque de 64 bits al mensaje. Este bloque se trata como un entero sin signo de 64 bits (el byte más significativo, primero) y contiene la longitud del mensaje original (antes del relleno). La inclusión de un valor de longitud dificulta un tipo de ataque conocido como ataque de relleno [TSUD92].

El resultado de los dos primeros pasos da lugar a un mensaje que es un entero múltiplo de 512 bits de longitud. En la Figura 3.4, el mensaje expandido se representa como la secuencia de bloques de 512 bits $Y_0 Y_1 \dots Y_{L-1}$, para que la longitud total del mensaje expandido sea $L \times 512$ bits. De la misma forma, el resultado es un múltiplo de 16 palabras de 32 bits. Sea $M[0 \dots N-1]$ las palabras del mensaje resultante, con N un entero múltiplo de 16. Entonces, $N = L \times 16$.

Paso 3: Inicializar el buffer MD. Un buffer de 160 bits se usa para tener resultados intermedios y finales de la función *hash*. El buffer puede representarse como cinco registros de 32 bits (A, B, C, D, E). Estos registros se inicializan a los siguientes enteros de 32 bits (valores hexadecimales):

$$\begin{aligned} A &= 67452301 \\ B &= EFCDAB89 \\ C &= 98BADCFE \\ D &= 10325476 \\ E &= C3D2E1F0 \end{aligned}$$

Paso 4: Procesar el mensaje en bloques de 512 bits (16 palabras). El corazón del algoritmo es un módulo, conocido como **función de compresión**, que consiste en cuatro etapas de procesamiento de 20 pasos cada una. La lógica se ilustra en la Figura 3.5. Las cuatro etapas tienen una estructura similar, pero cada una usa una función lógica primitiva diferente, a las que nos referimos como f_1, f_2, f_3 , y f_4 .

Cada etapa toma como entrada el bloque de 512 bits que se está procesando (Y_q) y el valor ABCDE del buffer de 160 bits y actualiza los contenidos del buffer. Cada etapa también hace uso de una constante adicional K_t donde $0 \leq t \leq 79$ indica uno de los 80 pasos a lo largo de cinco etapas. De hecho, sólo se usan cuatro constantes distintas. Los valores en decimal y hexadecimal son los siguientes:

Número de paso	Hexadecimal	toma parte entera de:
$0 \leq t \leq 19$	$K_t = 5A827999$	$[2^{30} \times \sqrt{2}]$
$20 \leq t \leq 39$	$K_t = 6ED9EBA1$	$[2^{30} \times \sqrt{3}]$
$40 \leq t \leq 59$	$K_t = 8F1BBCDC$	$[2^{30} \times \sqrt{5}]$
$60 \leq t \leq 79$	$K_t = CA62C1D6$	$[2^{30} \times \sqrt{10}]$

La salida de la cuarta etapa (octogésimo paso) se añade a la entrada de la primera etapa (CV_q) para producir CV_{q+1} . La suma se hace independientemente para cada una de las cinco palabras en el buffer con cada una de las palabras correspondientes en CV_q , usando suma módulo 2^{32} .

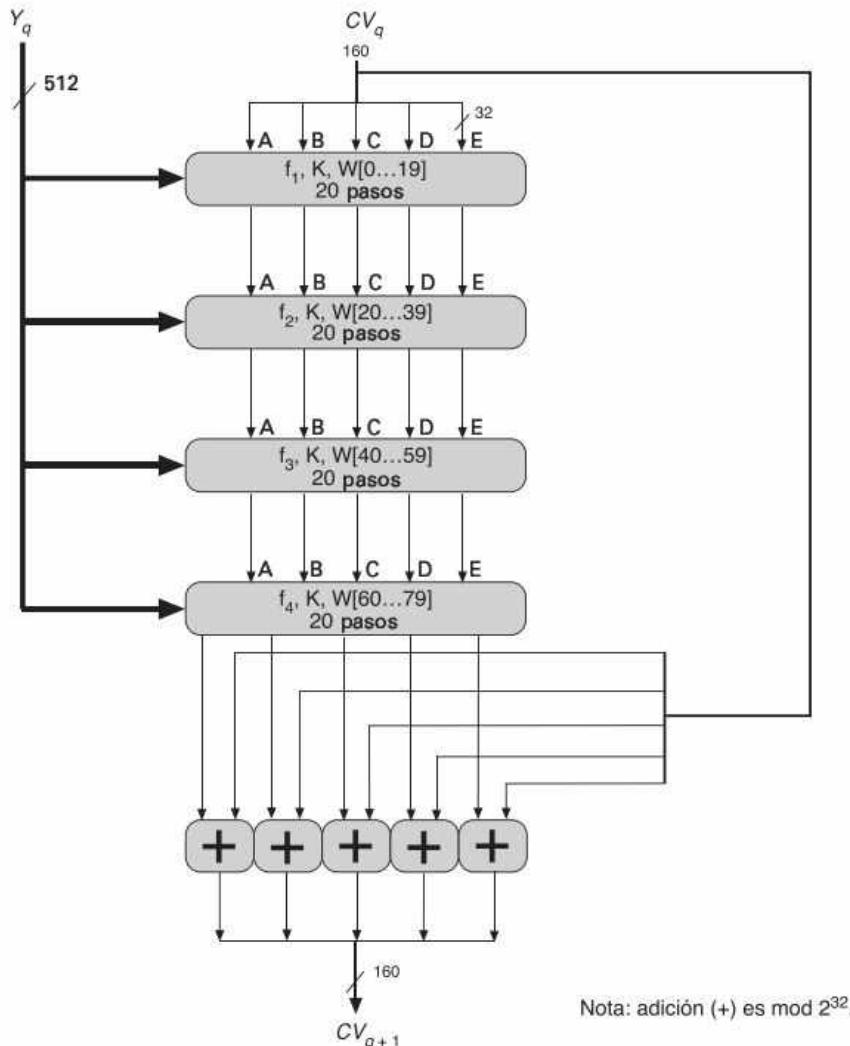


Figura 3.5 Procesamiento SHA-1 de un único bloque de 512 bits (función de compresión SHA-1)

Paso 5: Salida. Después de que todos los bloques L de 512 bits han sido procesados, la salida del L -ésimo estado es el resumen del mensaje de 160 bits.

El algoritmo SHA-1 tiene la propiedad por la cual cada bit del código *hash* es una función de cada bit de la entrada. La repetición compleja de la función básica f_t produce resultados bien mezclados; es decir, es poco probable que dos mensajes elegidos aleatoriamente, aun mostrando regularidades similares, tengan el mismo código *hash*. A

menos que haya alguna debilidad oculta en SHA-1, que hasta ahora no ha sido publicada, la dificultad de dar con dos mensajes con el mismo resumen es del orden de 2^{80} operaciones, mientras la dificultad de encontrar un mensaje con un resumen dado es del orden de 2^{160} operaciones.

OTRAS FUNCIONES HASH SEGURAS

Como ocurría en el caso de los cifrados de bloque simétricos, los diseñadores de funciones *hash* seguras se han mostrado reticentes a partir de una estructura probada. El DES se basa en el cifrado de Feistel. Prácticamente todos los cifrados de bloque siguientes siguen el diseño de Feistel porque puede ser adaptado para resistir amenazas criptoanalíticas de nuevo descubrimiento. Si, en vez de ello, se usara un diseño totalmente nuevo para un cifrado de bloque simétrico, habría peligro de que la estructura misma abriera nuevas vías de ataque aún no imaginadas. De la misma manera, la función *hash* moderna más importante sigue la estructura básica de la Figura 3.4, a la que se conoce como función *hash* repetida y fue propuesta inicialmente por Merkle [MERK79, MERK89]. La razón de esta estructura repetitiva surge de la observación de Merkle [MERK89] y Damgard [DAMG89] por la que si la función de compresión es resistente a la colisión, también lo será la función *hash* repetida resultante. Por lo tanto, la estructura puede usarse para producir una función *hash* segura para operar sobre un mensaje de cualquier longitud. El problema a la hora de diseñar una función *hash* segura se reduce al de diseñar una función de compresión resistente a la colisión que opere en entradas de tamaño fijo. Se ha demostrado que este es un enfoque muy profundo y que los nuevos diseños sólo mejoran la estructura y añaden longitud al código *hash*.

En esta sección observamos otras dos funciones *hash* seguras que, junto con la SHA-1, han obtenido gran aceptación comercial. En la Tabla 3.1 se comparan algunas de las características principales.

ALGORITMO DE RESUMEN DE MENSAJE MD5

El algoritmo de resumen de mensaje MD5 (RFC 1321) fue desarrollado por Ron Rivest. Hasta los últimos años, cuando surgió el interés tanto por la fuerza bruta como por el criptoanálisis, el MD5 fue el algoritmo *hash* seguro más usado. Este algoritmo toma como entrada un mensaje de longitud arbitraria y produce como salida un resumen de mensaje de 128 bits. La entrada se procesa en bloques de 512 bits.

Como la velocidad del procesador ha aumentado, la seguridad de un código *hash* de 128 bits se ha puesto en tela de juicio. Se puede demostrar que la dificultad de dar con dos mensajes teniendo el mismo resumen es del orden de 2^{64} operaciones, mientras la dificultad de encontrar un mensaje con un resumen dado es del orden de 2^{128} operaciones. La cifra anterior es demasiado pequeña para garantizar la seguridad. Además, se ha desarrollado una serie de ataques criptoanalíticos que sugieren la vulnerabilidad del MD5 al criptoanálisis [BERS92, BOER93, DOBB96a].

RIPEMD-160

El algoritmo de resumen de mensaje RIPEMD-160 [DOBB96b, BOSS97] se desarrolló en el proyecto RIPE (*European RACE Integrity Primitives Evaluation*), de manos de un

grupo de investigadores que lanzaron ataques con éxito parcial al MD4 y MD5. Originalmente, el grupo desarrolló una versión de RIPEMD de 128 bits. Cuando el proyecto RIPE hubo concluido, H. Dobbertin (que no formaba parte del proyecto) encontró ataques en dos etapas de RIPEMD, y más tarde de MD4 y MD5. A causa de estos ataques, algunos miembros del consorcio RIPE decidieron actualizar RIPEMD. Las tareas de diseño fueron llevadas a cabo por ellos junto con Dobbertin.

El RIPEMD-160 tiene una estructura muy similar a la del SHA-1. El algoritmo toma como entrada un mensaje de longitud arbitraria y produce como salida un resumen de mensaje de 160 bits. La entrada se procesa en bloques de 512 bits.

Tabla 3.1 Comparación de funciones *hash* seguras

	MD5	SHA-1	RIPEMD-160
Longitud del resumen	128 bits	160 bits	160 bits
Unidad básica de procesamiento	512 bits	512 bits	512 bits
Número de pasos	64 (4 etapas de 16)	80 (4 etapas de 20)	160 (5 pares de etapas de 16)
Tamaño máximo del mensaje	∞	$2^{64} - 1$ bit	∞
Funciones lógicas primitivas	4	4	5
Constantes adicionales usadas	64	4	9

HMAC

En los últimos años ha aumentado el interés por el desarrollo de un MAC derivado de un código *hash* criptográfico, como el SHA-1. Los motivos de este interés son los siguientes:

- Las funciones *hash* criptográficas, generalmente, se ejecutan más rápidamente en *software* que los algoritmos de cifrado convencional como el DES.
- Se encuentra disponible una librería para las funciones *hash* criptográficas.
- No existen restricciones de exportación desde los Estados Unidos u otros países para las funciones *hash* criptográficas, mientras que los algoritmos de cifrado convencional están restringidos, incluso cuando se usan para MAC.

Una función *hash* como, por ejemplo, SHA-1 no se diseñó para su uso como MCA y no puede usarse directamente con esa finalidad ya que no se basa en una clave secreta. Ha habido una serie de propuestas para la incorporación de una clave secreta en un algoritmo *hash* existente. El enfoque que ha recibido más apoyo ha sido el HMAC [BELL96a, BELL96b]. El HMAC se lanzó como RFC 2104, se ha elegido como el MAC de implementación obligatoria para la seguridad IP, y se emplea en otros protocolos de Internet como TLS (*Transport Layer Security*, que pronto reemplazará *Secure Sockets Layer*) y SET (*Secure Electronic Transaction*).

OBJETIVOS DEL DISEÑO DEL HMAC

El RFC 2104 presenta los siguientes objetivos del diseño del HMAC:

- Usar, sin modificaciones, las funciones *hash* disponibles, concretamente las funciones *hash* que se ejecutan bien en *software*, y para las cuales el código es gratis y fácil de conseguir.
- Permitir la sustitución fácil de la función *hash* empotrada en caso de que se encuentren o se necesiten funciones *hash* más rápidas o seguras.
- Preservar el funcionamiento original de la función *hash* sin incurrir en una degradación significativa.
- Usar y manejar claves de forma sencilla.
- Tener un análisis criptográfico comprensible de la robustez del mecanismo de autenticación basado en suposiciones razonables sobre la función *hash* empotrada.

Los dos primeros objetivos son importantes para la aceptabilidad del HMAC, que trata la función *hash* como una «caja negra». Esto tiene dos ventajas: en primer lugar, una implementación existente de una función *hash* puede usarse como un módulo en la implementación del HMAC. De esta forma, el grueso del código HMAC se empaqueta y está preparado para su uso sin modificación. Segundo, si alguna vez se deseara reemplazar una función *hash* dada en una implementación del HMAC, todo lo que se necesita es quitar el módulo de la función *hash* existente e introducir el nuevo módulo. Esto podría hacerse si se quisiera una función *hash* más rápida. Aún más importante, si la seguridad de la función *hash* empotrada se viera comprometida, la seguridad del HMAC podría conservarse sustituyendo la función *hash* empotrada por una más segura.

El último objetivo de diseño de la lista anterior es, de hecho, la principal ventaja del HMAC sobre otros esquemas propuestos basados en *hash*. El HMAC puede demostrar ser seguro si la función *hash* empotrada tiene propiedades criptográficas fuertes. Volveremos a ese punto más adelante en esta sección, pero antes examinaremos la estructura del HMAC.

ALGORITMO HMAC

La Figura 3.6 ilustra la operación general del HMAC. Define los siguientes términos:

H = función *hash* empotrada (SHA-1)

M = entrada de mensaje al HMAC (incluyendo el relleno especificado en la función *hash* empotrada)

Y_i = i -ésimo bloque de M , $0 \leq i \leq (L - 1)$

L = número de bloques en M

b = número de bits en un bloque

n = longitud del código *hash* producido por la función *hash* empotrada

K = clave secreta; si la longitud de la clave es mayor que b , la clave se introduce en la función *hash* para producir una clave de n bits; la longitud recomendada es $\geq n$

$K^+ = K$ llenado con ceros a la izquierda de forma que el resultado sea b bits de longitud

ipad = 00110110 (36 en hexadecimal) repetidos $b/8$ veces

opad = 01011100 (5C en hexadecimal) repetidos $b/8$ veces

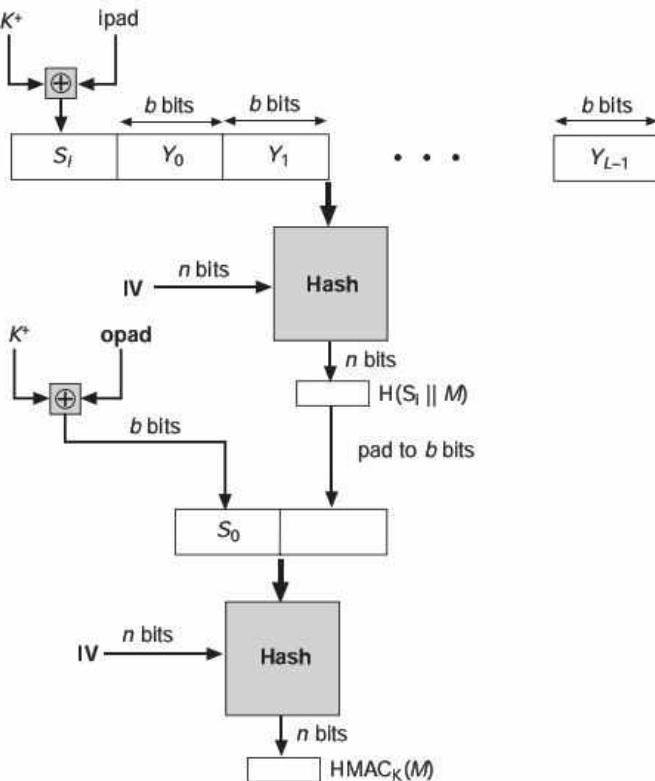


Figura 3.6 Estructura del HMAC

Entonces el HMAC puede expresarse de la siguiente manera:

$$\text{HMAC}_K(M) = H[K^+ \oplus \text{opad}] \parallel H[K^+ \oplus \text{ipad}] \parallel M]$$

Es decir,

1. Añadir ceros a la izquierda de K para crear una ristra K^+ de b bits. Por ejemplo, si K tiene una longitud de 160 bits y $b = 512$, entonces K se añadirá con 44 bytes a 0 (0x00).
2. Aplicar el XOR (OR exclusivo bit a bit) a K^+ con ipad para producir el bloque de b bits S_i .
3. Añadir M a S_i .
4. Aplicar H a la ristra generada en el paso 3.
5. Aplicar el XOR a K^+ con opad para producir el bloque de b bits S_o .

6. Añadir el resultado *hash* del paso 4 a S_o
7. Aplicar H al flujo generado en el paso 6 y extraer el resultado.

Hay que tener en cuenta que el XOR con el *ipad* da como resultado la inversión de la mitad de los bits de K . De forma parecida, el XOR con el *opad* da como resultado la inversión de la mitad de los bits de K , pero un grupo diferente de bits. En efecto, al pasar S_i y S_o por el algoritmo *hash*, hemos generado de forma pseudoaleatoria dos claves de K .

El HMAC debería ejecutarse en aproximadamente el mismo tiempo que la función *hash* empotrada para mensajes largos. El HMAC añade tres ejecuciones de la función *hash* básica (para S_i , S_o y el bloque producido por el *hash* interno).

3.3 PRINCIPIOS DE CRIPTOGRAFÍA DE CLAVE PÚBLICA

De igual importancia que el cifrado convencional es el cifrado de clave pública, que se emplea en autentificación de mensajes y en distribución de claves. Esta sección analiza, en primer lugar, el concepto básico de cifrado de clave pública e introduce aspectos preliminares de la distribución de claves. La sección 3.4 examina los dos algoritmos de clave pública más importantes, el RSA y el Diffie-Hellman, y la 3.5 introduce las firmas digitales.

ESTRUCTURA DEL CIFRADO DE CLAVE PÚBLICA

El cifrado de clave pública, propuesto por primera vez por Diffie y Hellman en 1976 [DIFF76], es el primer avance realmente revolucionario en el cifrado en miles de años. El motivo es que los algoritmos de clave pública están basados en funciones matemáticas y no en simples operaciones sobre los patrones de bits. Además, la criptografía de clave pública es asimétrica, lo que implica el uso de dos claves separadas, a diferencia del cifrado simétrico convencional, que emplea sólo una clave. El uso de dos claves tiene importantes consecuencias en el terreno de la confidencialidad, la distribución de claves y la autentificación.

Antes de continuar, deberíamos mencionar algunas confusiones comunes en lo relativo al cifrado de clave pública. La primera es la creencia de que el cifrado de clave pública es más seguro ante el criptoanálisis que el cifrado convencional. De hecho, la seguridad de cualquier esquema de cifrado depende de (1) la longitud de la clave y (2) el coste computacional necesario para romper un cifrado. No hay nada sobre el cifrado convencional ni de clave pública que haga a uno superior al otro en lo que respecta a la resistencia al criptoanálisis. Otra equivocación la hallamos en la idea de que el cifrado de clave pública es una técnica con propósitos generales que ha dejado desfasado el cifrado convencional. Por el contrario, debido al coste computacional de los esquemas actuales de cifrado de clave pública, no parece que el cifrado convencional vaya a abandonarse. Por último, se piensa que la distribución de claves no es importante cuando se usa el cifrado de clave pública, en comparación con los incómodos acuerdos previos que tienen lugar con los centros de distribución de claves para el cifrado convencional. De hecho, se necesita alguna forma de protocolo, que a menudo implica a un agente cen-

tral, y los procedimientos que tienen lugar no son más sencillos ni más eficientes que los que se requieren para el cifrado convencional.

Un esquema de cifrado de clave pública tiene seis componentes (Figura 3.7a):

- **Texto claro:** consiste en el mensaje o los datos legibles que se introducen en el algoritmo como entrada.
- **Algoritmo de cifrado:** el algoritmo de cifrado realiza diferentes transformaciones en el texto claro.
- **Clave pública y privada:** es una pareja de claves que han sido seleccionadas, de las cuales una se usa para el cifrado y la otra para el descifrado. Las transformaciones exactas llevadas a cabo por el algoritmo de cifrado dependen de la clave pública o privada que se proporciona como entrada.
- **Texto cifrado:** es el mensaje desordenado producido como salida. Depende del texto claro y de la clave. Para un mensaje dado, dos claves diferentes producirán dos textos cifrados diferentes.
- **Algoritmo de descifrado:** este algoritmo acepta el texto cifrado y la clave correspondiente y produce el texto claro original.

Como los nombres sugieren, la clave pública de dicha pareja de claves se hace pública para que otros la usen, mientras que la clave privada sólo es conocida por su propietario. Un algoritmo criptográfico de clave pública con propósito general se basa en una clave para el cifrado y otra diferente, aunque relacionada, para el descifrado.

Los pasos fundamentales son los siguientes:

1. Cada usuario genera una pareja de claves para el cifrado y el descifrado de mensajes.
2. Cada usuario localiza una de las dos claves en un registro público u otro archivo accesible. Esta es la clave pública. La otra clave no se revela. Como sugiere la Figura 3.7a, cada usuario mantiene un grupo de claves públicas que han obtenido de otros.
3. Si Benito quiere enviar un mensaje privado a Alicia, cifra el mensaje usando la clave pública de Alicia.
4. Cuando Alicia recibe el mensaje, lo descifra usando su clave privada. Ningún otro receptor puede descifrar el mensaje porque sólo Alicia conoce su clave privada.

En este enfoque, todos los participantes tienen acceso a las claves públicas, y las claves privadas las genera cada participante de forma local y, por lo tanto, nunca necesitan ser distribuidas.

Mientras el usuario proteja su clave privada, la comunicación entrante es segura. En cualquier momento un usuario puede cambiar la clave privada y publicar la clave pública que la acompaña para sustituir la clave pública antigua.

La clave empleada en el cifrado convencional se denomina comúnmente **clave secreta**. Las dos claves empleadas para el cifrado de clave pública se denominan **clave pública** y **clave privada**. Invariablemente, la clave privada se mantiene en secreto, pero en vez de llamarse clave secreta se llama clave privada para evitar confusiones con el cifrado convencional.

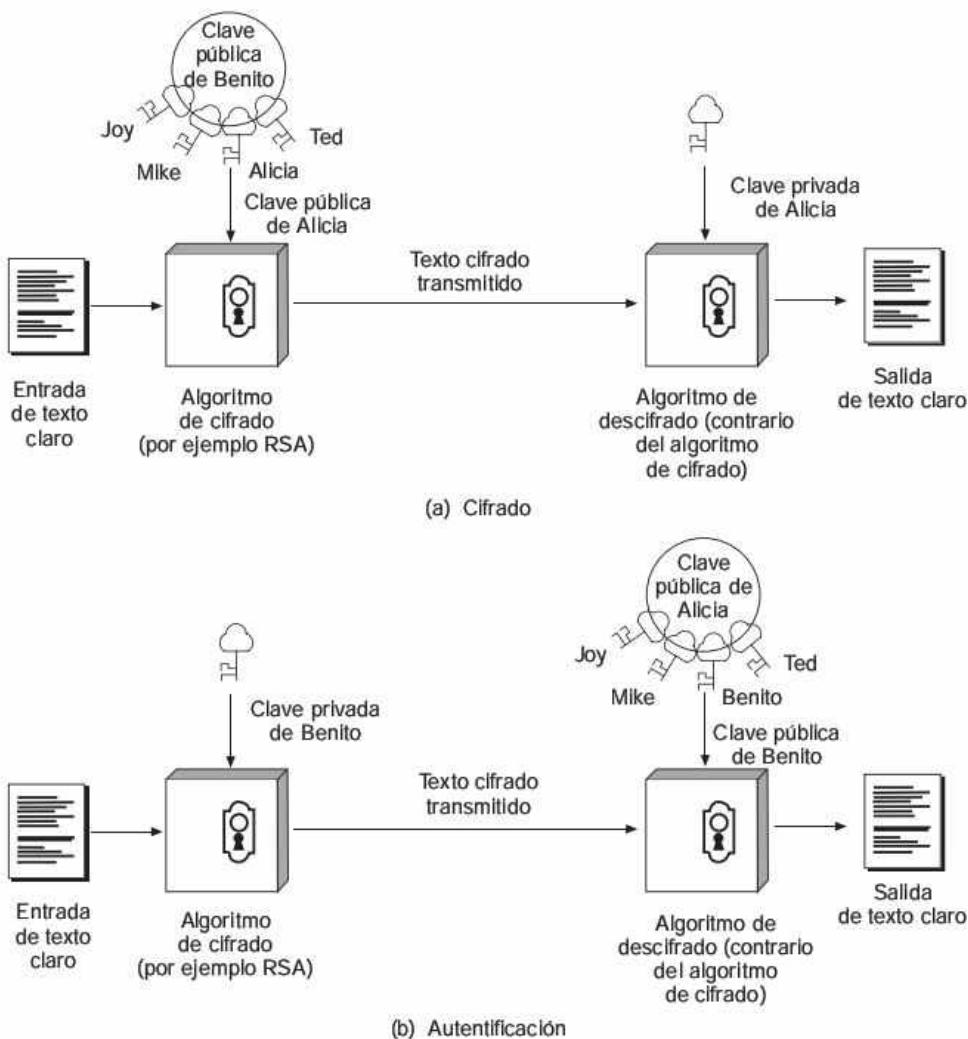


Figura 3.7 Criptografía de clave pública

APLICACIONES PARA CRIPTOSISTEMAS DE CLAVE PÚBLICA

Antes de continuar, es necesario aclarar un aspecto de los criptosistemas de clave pública que, de otro modo, podría llevar a confusión. Los sistemas de clave pública se caracterizan por el uso de un tipo de algoritmo criptográfico con dos claves, una no se revela y la otra sí. Dependiendo de la aplicación, el emisor usa su clave privada o la clave pública del receptor, o las dos, para realizar algún tipo de función criptográfica. En términos generales, podemos clasificar el uso de criptosistemas de clave pública en tres categorías:

- **Cifrado/descifrado:** el emisor cifra un mensaje con la clave pública del receptor.

- **Firma digital:** el emisor «firma» un mensaje con su clave privada. Esto se consigue mediante un algoritmo criptográfico aplicado al mensaje o a un pequeño bloque de datos que es una función del mensaje.
- **Intercambio de claves:** dos partes cooperan para intercambiar una clave de sesión. Hay distintas posibilidades que implican la clave privada de una o de las dos partes.

Algunos algoritmos son adecuados para las tres aplicaciones, mientras otros sólo se pueden usar para una o dos de ellas. La Tabla 3.2 indica las aplicaciones que soportan los algoritmos que se tratan en este Capítulo, RSA y Diffie Hellman. La tabla también incluye el Estándar de Firma Digital (DSS) y la criptografía de curva elíptica, que se trata más tarde en este Capítulo.

Tabla 3.2 Aplicaciones para criptosistemas de clave pública

Algoritmo	Cifrado/descifrado	Firma digital	Intercambio de clave
RSA	Sí	Sí	Sí
Diffie-Hellman	No	No	Sí
DSS	No	Sí	No
Curva elíptica	Sí	Sí	Sí

REQUISITOS PARA LA CRIPTOGRAFÍA DE CLAVE PÚBLICA

El criptosistema que se muestra en la Figura 3.7 depende de un algoritmo criptográfico basado en dos claves relacionadas. Diffie y Hellman postularon este sistema sin demostrar que tales algoritmos existen. Sin embargo, sí especificaron las condiciones que deben cumplir [DIFF76]:

1. Desde el punto de vista computacional, para una parte B es fácil generar una pareja de claves (clave pública KU_b , clave privada KR_b).
2. En términos computacionales, para un emisor A que conozca la clave pública y el mensaje que ha de cifrase, M , es fácil generar el texto cifrado correspondiente:

$$C = E_{KU_b}(M)$$

3. En términos computacionales, para un receptor B es fácil descifrar el texto cifrado resultante usando la clave privada para recuperar el mensaje original:

$$M = D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$$

4. Desde el punto de vista computacional, es imposible que un oponente, conociendo la clave pública, KU_b , determine la clave privada KR_b .
5. Desde el punto de vista computacional, es imposible que un oponente, conociendo la clave pública, KU_b , y un texto cifrado, C , recupere el mensaje original, M .

Podemos añadir un sexto requisito que, aunque útil, no es necesario para todas las aplicaciones de clave pública:

- 6. Cualquiera de las dos claves relacionadas puede usarse para el cifrado, y la otra para el descifrado.

$$M = D_{KR_b}[E_{KU_b}(M)] = D_{KU_b}[E_{KR_b}(M)]$$

3.4 ALGORITMOS DE CRIPTOGRAFÍA DE CLAVE PÚBLICA

Los dos algoritmos de clave pública más usados son el RSA y el Diffie-Hellman, que se analizarán en esta sección. También se introducirán brevemente otros dos algoritmos³.

EL ALGORITMO DE CIFRADO DE CLAVE PÚBLICA RSA

Uno de los primeros esquemas de clave pública fue desarrollado en 1977 por Ron Rivest, Adi Shamir y Len Adleman en el MIT y se publicó en 1978 [RIVE78]. El esquema RSA ha sido desde entonces el enfoque más aceptado e implementado para el cifrado de clave pública. El RSA es un cifrado de bloque en el que el texto claro y el texto cifrado son enteros entre 0 y $n - 1$ para algún n .

Para algún bloque de texto claro M y un bloque de texto cifrado C , el cifrado y el descifrado son de la siguiente forma:

$$\begin{aligned} C &= M^e \bmod n \\ M &= C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n \end{aligned}$$

Tanto el emisor como el receptor deben conocer los valores de n y e , y sólo el receptor conoce el valor de d . Este es un algoritmo de cifrado de clave pública con una clave pública de $KU = \{e, n\}$ y una clave privada de $KR = \{d, n\}$. Para que este algoritmo sea satisfactorio para el cifrado de clave pública, se deben cumplir los siguientes requisitos:

1. Que sea posible encontrar valores de e , d , n tal que $M^{ed} = M \bmod n$ para todo $M < n$.
2. Que sea relativamente fácil calcular M^e y C^d para todos los valores de $M < n$.
3. Que sea imposible determinar d dados e y n .

Los dos primeros requisitos se cumplen fácilmente. El tercero se puede cumplir para valores grandes de e y n .

La Figura 3.8 resume el algoritmo RSA. Se empieza seleccionando dos números primos, p y q , y calculando su producto n , que es el módulo para cifrado y descifrado. A continuación, se necesita la cantidad $\phi(n)$, conocida como *función totient de Euler* de n , que es el número de enteros positivos menor que n y primo relativo de n . Luego, se selecciona un entero e que sea primo relativo de $\phi(n)$ [el mayor común divisor de e y ϕ

³ Esta sección emplea algunos conceptos elementales de la teoría de números. Como repaso, véase el apéndice B.

Generación clave

Seleccionar p, q	$p \text{ y } q \text{ primos, } p \neq q$
Calcular $n = p \times q$	
Calcular $\phi(n) = (p - 1)(q - 1)$	
Seleccionar entero e	$\text{gcd}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcular d	$de \bmod \phi(n) = 1$
Clave pública	$KU = \{e, n\}$
Clave privada	$KR = \{d, n\}$

Cifrado

Texto claro:	$M < n$
Texto cifrado:	$C = M^e \pmod{n}$

Descifrado

Texto cifrado:	C
Texto claro:	$M = C^d \pmod{n}$

Figura 3.8 El algoritmo RSA

(n) es 1]. Finalmente, se calcula d como el inverso multiplicativo de e , módulo $\phi(n)$. Se puede demostrar que d y e tienen las propiedades deseadas.

Supongamos que el usuario A ha publicado su clave pública y que el usuario B quiere enviar el mensaje M a A. Entonces B calcula $C = M^e \pmod{n}$ y transmite C . Al recibir el texto cifrado, el usuario A descifra calculando $M = C^d \pmod{n}$.

En la Figura 3.9 se muestra un ejemplo de [SING99]. Para dicho ejemplo las claves se generaron de la siguiente manera:

1. Seleccionar dos números primos, $p = 17$ y $q = 11$.
2. Calcular $n = pq = 17 \times 11 = 187$.
3. Calcular $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Seleccionar e tal que e es primo relativo de $\phi(n) = 160$ y menor que $\phi(n)$; elegimos $e = 7$.
5. Determinar d tal que $de \bmod 160 = 1$ y $d < 160$. El valor correcto es $d = 23$, porque $23 \times 7 = 161 = 10 \times 160 + 1$.

Las claves resultantes son la clave pública $KU = \{7, 187\}$ y la clave privada $KR = \{23, 187\}$. El ejemplo muestra el uso de estas claves para una entrada de texto claro de $M = 88$. Para el cifrado necesitamos calcular $C = 88^7 \bmod 187$. Teniendo en cuenta las propiedades de la aritmética modular, podemos realizarlo como sigue:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59,969,536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894,432 \bmod 187 = 11$$

Para el descifrado calculamos $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14,641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214,358,881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79,720,245 \bmod 187 = 88$$

Hay dos enfoques posibles para romper al algoritmo RSA. El primero es el enfoque de fuerza bruta: intentar todas las claves privadas posibles. Así, cuanto mayor sea el número de bits en e y d , más seguro será el algoritmo. Sin embargo, debido a que los cálculos que tienen lugar tanto en la generación de clave como en el cifrado/descifrado son complejos, cuanto mayor sea el tamaño de la clave, más lento irá el sistema.

La mayoría de las discusiones sobre el criptoanálisis del RSA se han centrado en la tarea de factorizar n en sus dos factores primos. Un número n producto de dos números primos grandes es difícil de factorizar, aunque no tanto como solía ser. Una ilustración llamativa de ello ocurrió en 1977; los tres inventores del RSA retaron a los lectores de *Scientific American* a decodificar un cifrado que publicaron en la columna de juegos matemáticos (*Mathematical Games*) de Martin Gardner [GARD77]. Ofrecieron una recompensa de 100 dólares por la recuperación de una frase de texto claro, algo que, según predijeron, no ocurriría durante unos 40 cuatrillones de años. En abril de 1994, un grupo que trabajaba en Internet y que usaba unos 1.600 computadores consiguió el premio después de sólo ocho meses de trabajo [LEUT94]. En este reto se usó un tamaño de clave pública (longitud de n) de 129 dígitos decimales, alrededor de 428 bits. Este resultado no invalida el uso del RSA; simplemente significa que debe usarse un tamaño de clave mayor. Actualmente, un tamaño de clave de 1024 bits (300 dígitos decimales aproximadamente) se considera lo suficientemente robusto para casi todas las aplicaciones.

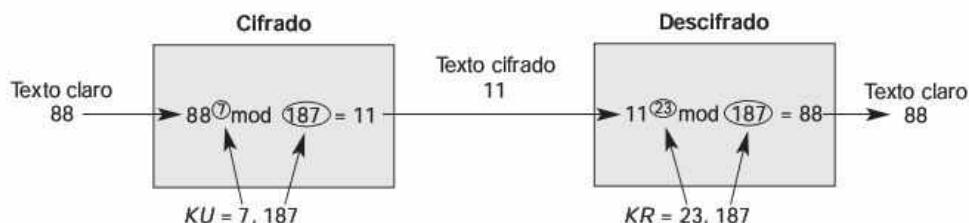


Figura 3.9 Ejemplo de algoritmo RSA

INTERCAMBIO DE CLAVE DIFFIE-HELLMAN

El primer algoritmo de clave pública apareció en el artículo de Diffie y Hellman que definía la criptografía de clave pública [DIF76] y que se conoce por intercambio de clave de Diffie-Hellman. Una serie de productos comerciales emplearon esta técnica de intercambio de claves.

La finalidad del algoritmo es hacer posible que los usuarios intercambien de forma segura una clave secreta que luego pueda ser usada para el cifrado posterior de mensajes. El algoritmo está limitado al intercambio de claves.

El algoritmo de Diffie-Hellman depende para su efectividad de la dificultad de computar logaritmos discretos. En resumen, podemos definir el logaritmo discreto de la siguiente forma: primero, definimos una raíz primitiva de un número primo p cuyas potencias generan todos los enteros desde 1 a $p - 1$. Es decir, si a es una raíz primitiva del número primo p , entonces los números

$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$

son distintos y consisten en los enteros desde 1 hasta $p - 1$ en alguna de sus permutaciones.

Para cualquier entero b menor que p y una raíz primitiva a del número primo p , se puede encontrar un único exponente i tal que

$$b = a^i \bmod p \quad \text{donde } 0 \leq i \leq (p - 1)$$

El exponente i se conoce como el logaritmo discreto o índice de b para la base a , mod p . Este valor se representa como $\text{ind}_{a,p}(b)$.

Con toda esta información se puede definir el intercambio de clave de Diffie-Hellman, que se resume en la Figura 3.10. Para este esquema, hay dos números conocidos públicamente: un número primo q y un entero α que es una raíz primitiva de q . Supongamos que los usuarios A y B quieren intercambiar una clave. El usuario A selecciona un entero aleatorio $X_A < q$ y computa $Y_A = \alpha^{X_A} \bmod q$. De igual forma, el usuario B selecciona independientemente un entero aleatorio $X_B < q$ y calcula $Y_B = \alpha^{X_B} \bmod q$. Cada parte mantiene el valor X en privado y hace público el valor Y a la otra parte. El usuario A computa la clave como $K = (Y_B)^{X_A} \bmod q$ y el usuario B computa la clave como $K = (Y_A)^{X_B} \bmod q$. Estos dos cálculos producen resultados idénticos:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= \alpha^{X_B X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

Elementos públicos globales

q	número primo
α	$\alpha < q$ y α una raíz prima de q

Generación de la clave del usuario A

Seleccionar X_A privada	$X_A < q$
Calcular Y_A pública	$Y_A = \alpha^{X_A} \bmod q$

Generación de la clave del usuario B

Seleccionar X_B privada	$X_B < q$
Calcular Y_B pública	$Y_B = \alpha^{X_B} \bmod q$

Generación de la clave secreta por el usuario A

$$K = (Y_B)^{X_A} \bmod q$$

Generación de la clave secreta por el usuario B

$$K = (Y_A)^{X_B} \bmod q$$

Figura 3.10 Algoritmo de intercambio de claves de Diffie-Hellman

Así, las dos partes han intercambiado una clave secreta. Además, como X_A y X_B son privadas, un oponente sólo tiene los siguientes componentes para trabajar: q , α , Y_A y Y_B . De esta forma, el oponente se ve obligado a tomar un logaritmo discreto para determinar la clave. Por ejemplo, atacando la clave secreta del usuario B, el oponente debe computar

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

Entonces el oponente puede calcular la clave K de la misma manera que lo hace el usuario.

La seguridad del intercambio de claves de Diffie-Hellman se basa en el hecho de que, aunque es relativamente fácil calcular exponentiales módulo un número primo, es muy difícil calcular logaritmos discretos. Para números primos grandes, la última tarea se considera imposible.

Por ejemplo, seleccionar el número primo $q = 71$ y una raíz primitiva de 71, en este caso, $\alpha = 7$. A y B seleccionan las claves privadas $X_A = 5$ y $X_B = 12$, respectivamente. Cada uno computa su clave pública:

$$Y_A = 7^5 \bmod 71 = 51$$

$$Y_B = 7^{12} \bmod 71 = 4$$

Después de intercambiar sus claves públicas, cada uno computa la clave secreta común:

$$K = (Y_B)^{X_A} \bmod 71 = 4^5 \bmod 71 = 30$$

$$K = (Y_A)^{X_B} \bmod 71 = 51^{12} \bmod 71 = 30$$

Desde $\{51, 4\}$, un atacante no puede computar 30 fácilmente.

La Figura 3.11 muestra un protocolo simple que hace uso del cálculo Diffie-Hellman. Supongamos que el usuario A quiere establecer una conexión con el usuario B y usa una clave secreta para cifrar mensajes en esa conexión. El usuario A puede generar una clave privada exclusiva X_A , calcular Y_A , y enviarlo al usuario B . El usuario B responde generando un valor privado X_B , calculando Y_B y enviando Y_B al usuario A . Ahora los dos usuarios pueden calcular la clave. Los valores públicos necesarios q y α tendrían que ser conocidos con antelación. Por otra parte, el usuario A podría tomar valores para q y α e incluirlos en el primer mensaje.

Otro ejemplo del uso del algoritmo Diffie-Hellman es el siguiente: supongamos que en un grupo de usuarios (por ejemplo, todos los usuarios de una red LAN) cada uno genera un valor privado de larga duración X_A y calcula un valor público Y_A . Estos valores públicos, junto con los valores públicos globales para q y α , se almacenan en algún directorio central. En cualquier momento, el usuario B puede acceder al valor público del usuario A , calcular una clave secreta y usarla para enviar un mensaje cifrado al usuario A . Si el directorio central es fiable, esta forma de comunicación proporciona confidencialidad y un cierto grado de autentificación. Como sólo A y B pueden determinar la clave, ningún otro usuario puede leer el mensaje (confidencialidad). El receptor A sabe que sólo el usuario B podría haber creado un mensaje usando esta clave (autentificación). Sin embargo, la técnica no protege contra ataques de repetición.

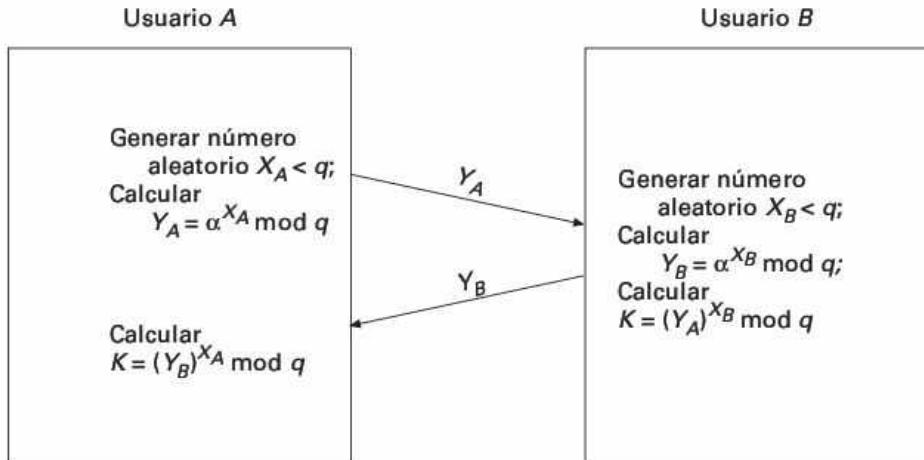


Figura 3.11 Intercambio de claves de Diffie-Hellman

OTROS ALGORITMOS CRIPTOGRÁFICOS DE CLAVE PÚBLICA

Otros dos algoritmos de clave pública han hallado aceptación comercial: el DDS y la criptografía de curva elíptica.

ESTÁNDAR DE FIRMA DIGITAL

El NIST (*National Institute of Standards and Technology*) ha publicado el *Federal Information Processing Standard*, FIPS PUB 186, conocido como DSS (*Digital Signature Standard*). El DSS hace uso del SHA-1 y presenta una nueva técnica de firma digital, el Algoritmo de Firma Digital o DSA (*Digital Signature Algorithm*). El DSS fue propuesto originalmente en 1991 y revisado en 1993 como consecuencia de la respuesta del público en lo relativo a la seguridad del esquema. Hubo otra revisión menor en 1996. El DSS usa un algoritmo diseñado para proporcionar sólo la función de firma digital. A diferencia del RSA, no puede usarse para el cifrado o el intercambio de claves.

CRIPTOGRAFÍA DE CURVA ELÍPTICA

La mayoría de los productos y estándares que usan criptografía de clave pública para el cifrado y las firmas digitales usan RSA. La longitud de bit para el uso seguro del RSA ha aumentado en los últimos años, y esto ha supuesto una mayor carga de procesamiento en las aplicaciones que usan el RSA. Esta carga tiene ramificaciones, especialmente para sitios de comercio electrónico que realizan grandes cantidades de transacciones seguras. Recientemente, un sistema competidor ha empezado a retar al RSA: la criptografía de curva elíptica (ECC), que ya está haciendo esfuerzos de estandarización, incluido el Estándar IEEE P1363 para la Criptografía de Clave Pública.

La atracción principal de la ECC en relación al RSA es que parece ofrecer igual seguridad por un tamaño de bit mucho menor, reduciendo, con ello, los costes de procesamiento. Por otra parte, aunque la teoría de la ECC ha estado presente durante algún tiempo, los productos empezaron a aparecer muy recientemente y ha habido un interés prolongado por probar sus debilidades. Así, el nivel de confianza en la ECC todavía no alcanza al del RSA.

La ECC es más difícil de explicar que el RSA o el Diffie-Hellman, y una descripción matemática exhaustiva sobrepasa el ámbito de este libro. La técnica se basa en el uso de un constructor matemático conocido como la curva elíptica.

3.5 FIRMAS DIGITALES

El cifrado de clave pública se puede usar de otra forma, como ilustra la Figura 3.7b. Supongamos que Benito quiere enviar un mensaje a Alicia y, aunque no es necesario que el mensaje se mantenga en secreto, quiere que Alicia se asegure de que el mensaje, efectivamente, proviene de él. En este caso Benito usa su propia clave privada para cifrar el mensaje. Cuando Alicia recibe el texto cifrado, se encuentra con que puede descifrarlo con la clave pública de Benito, demostrando así, que el mensaje ha debido ser cifrado por él. Nadie más tiene la clave privada de Benito y, por lo tanto, nadie más ha podido crear un texto cifrado que pueda ser descifrado con su clave pública. Por consiguiente, el mensaje cifrado sirve como **firma digital**. Además, es imposible alterar el mensaje sin acceso a la clave privada de Benito, así que el mensaje queda autenticado tanto en lo que respecta a la fuente como a la integridad de los datos.

En el esquema anterior, se ha cifrado todo el mensaje, que, aunque valide el autor y los contenidos, necesita una gran cantidad de almacenamiento. También se debería almacenar una copia en texto cifrado para que el origen y el contenido se puedan verificar en caso de desacuerdo. Una forma más efectiva de lograr los mismos resultados es cifrar un pequeño bloque de bits que sea una función de un documento. Este bloque, llamado autenticador, debe tener la propiedad por la cual es imposible cambiar el documento sin cambiar el autenticador. Si el autenticador se cifra con la clave privada del emisor, sirve como firma que verifica el origen, el contenido y la secuencia. Un código *hash* seguro como el SHA-1 puede realizar esta función, como muestra la Figura 3.2b.

Es importante resaltar que el proceso de cifrado que se acaba de describir no proporciona confidencialidad. Es decir, el mensaje que se está enviando es seguro contra alteraciones pero no contra escuchas, lo que es obvio en el caso de una firma basada en una parte del mensaje, porque el resto del mensaje se transmite en claro. Incluso en el caso del cifrado completo, no hay protección de confidencialidad ya que cualquier observador puede descifrar el mensaje usando la clave pública del emisor.

3.6 GESTIÓN DE CLAVES

Una de las funciones principales del cifrado de clave pública es la de tratar el problema de la distribución de claves. Hay dos aspectos fundamentales sobre el uso del cifrado de clave pública en este sentido:

- La distribución de claves públicas.
- El uso de cifrado de clave pública para distribuir claves secretas.

Vamos a examinar cada una de estas áreas.

CERTIFICADOS DE CLAVE PÚBLICA

A la vista de todo esto, la base del cifrado de clave pública se encuentra en el hecho de que la clave pública es pública. Así, si hay un algoritmo de clave pública aceptado, como el RSA, cualquier participante puede enviar su clave pública a cualquier otro o difundir la clave a la comunidad en general. Aunque este enfoque es conveniente, presenta una debilidad fundamental: cualquiera puede falsificar ese dato público. Es decir, un usuario podría hacerse pasar por el usuario A y enviar una clave pública a otro participante o difundirla. Hasta el momento en que A descubre la falsificación y alerta a los otros participantes, el falsificador puede leer todos los mensajes cifrados enviados a A y puede usar las claves falsificadas para la autenticación.

La solución a este problema es el certificado de clave pública. Básicamente, un certificado consiste en una clave pública y un identificador o nombre de usuario del dueño de la clave, con todo el bloque firmado por una tercera parte confiable. Comúnmente, la tercera parte es una autoridad de certificación (CA, *Certificate Authority*) en la que confía la comunidad de usuarios, que podría ser una agencia gubernamental o una institución financiera. Un usuario puede presentar su clave pública a la autoridad de forma segura, obtener un certificado y luego publicarlo. Cualquiera que necesite la clave públ-

ca de este usuario puede obtener el certificado y verificar que es válida por medio de la firma fiable adjunta. La Figura 3.12 ilustra este proceso.

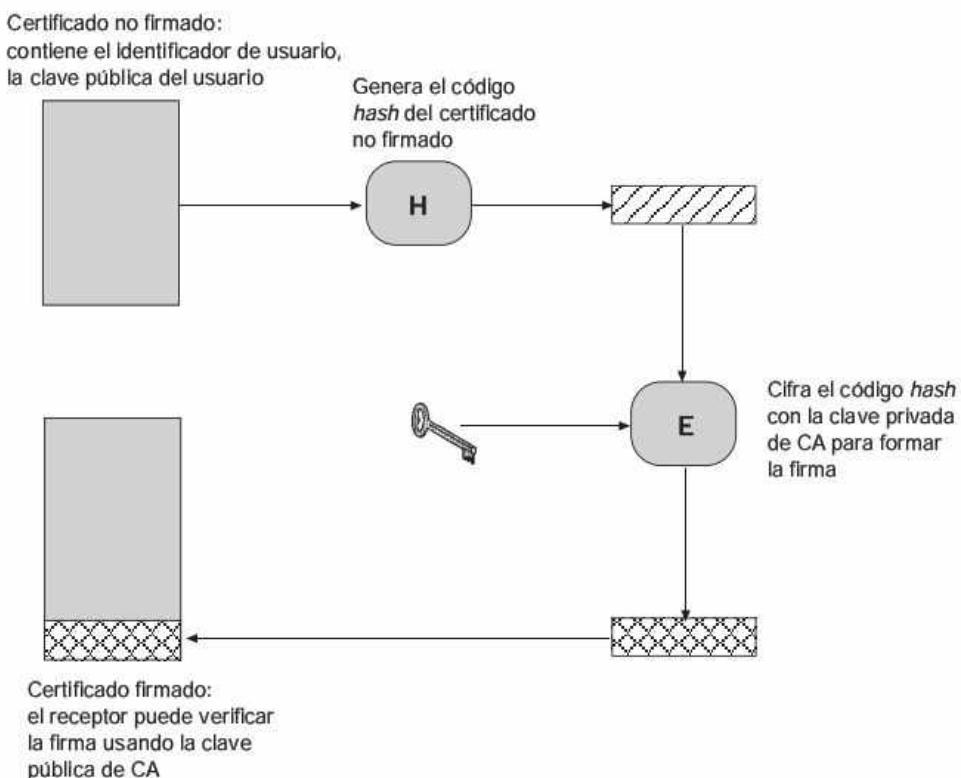


Figura 3.12 Uso del certificado de clave pública

El esquema que se ha aceptado mundialmente para el formateado de los certificados de clave pública es el estándar X.509, cuyos certificados se emplean en la mayoría de las aplicaciones de seguridad de redes, incluyendo la seguridad IP, las capas de conexión segura (SSL), las transacciones electrónicas seguras (SET) y S/MIME, que se tratarán en la Segunda Parte del libro. El X.509 se examina detalladamente en el Capítulo 4.

DISTRIBUCIÓN DE CLAVES SECRETAS MEDIANTE CRIPTOGRAFÍA DE CLAVE PÚBLICA

Con el cifrado convencional, un requisito fundamental para que las dos partes se comuniquen de forma segura es que comparten la clave secreta. Supongamos que Benito quiere crear una aplicación de mensajes que le permita intercambiar correo electrónico de forma segura con alguien que tiene acceso a Internet o a otra red que ambos comparten. Supongamos, además, que quiere hacerlo usando cifrado convencional. Con el cifrado convencional, Benito y su interlocutor, Alicia, deben acordar una forma de compartir

una clave secreta que nadie más conozca. ¿Cómo van a hacerlo? Si Alicia se encuentra en la habitación contigua a la de Benito, éste podría generar una clave y anotarla en un papel o guardarla en un disquete y entregarla a Alicia. Pero si Alicia está en el otro extremo del continente o del mundo, ¿qué puede hacer Benito? Podría cifrar la clave usando cifrado convencional y enviarla por correo electrónico a Alicia, pero esto significa que ambos deben compartir una clave secreta para cifrar esta nueva clave secreta. Además, Benito y todo aquel que use este nuevo paquete de correo electrónico se enfrenta al mismo problema con cada posible interlocutor: cada pareja de interlocutores debe compartir una clave secreta única.

Un enfoque consiste en el uso del intercambio de clave de Diffie-Hellman, que, de hecho, está muy extendido. Sin embargo, tiene la desventaja de que en su forma más simple el algoritmo de Diffie-Hellman no proporciona la autenticación de las dos partes que se comunican.

Una alternativa válida es el uso de los certificados de clave pública. Cuando Benito quiera comunicarse con Alicia, puede hacer lo siguiente:

- 1.** Preparar un mensaje.
- 2.** Cifrar el mensaje usando cifrado convencional con una clave de sesión convencional.
- 3.** Cifrar la clave de sesión utilizando el cifrado de clave pública con la clave pública de Alicia.
- 4.** Añadir la clave de sesión cifrada al mensaje y enviarlo a Alicia.

Sólo Alicia puede descifrar la clave de sesión y, por lo tanto, recuperar el mensaje original. Si Benito obtuvo la clave pública de Alicia a través del certificado de clave pública de Alicia, está seguro de que se trata de una clave válida.

3.7 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

En [STIN02] y [MENE97] se tratan en profundidad las funciones *hash* y los códigos de autenticación de mensajes.

Los tratamientos recomendados de cifrado proporcionados en el Capítulo 2 cubren el cifrado de clave pública y el cifrado convencional. [DIFF88] describe detalladamente los intentos realizados para crear algoritmos seguros de doble clave y la evolución gradual de una serie de protocolos basados en ellos. Un buen tratamiento de la criptografía de clave pública se encuentra en [SALO96]. [CORM01] proporciona un resumen conciso pero completo y claro de todos los algoritmos relevantes para la verificación, la computación y el criptoanálisis del RSA.

- CORM01** Cormen, T; Leiserson, C.; Rivest, R.; y Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
- DIFF88** Diffie, W. «The First Ten Years of Public-Key Cryptography». *Proceedings of the IEEE*, mayo 1998. Reimpreso en [SIMM92].
- MENE97** Menezes, A.; Oorschot, P.; y Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

- SAL096** Salomaa, A. *Public-Key Cryptography*. New York: Springer-Verlag, 1996.
- SIMM92** Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.
- STIN02** Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

Sitios web recomendados:

- **RSA Laboratories:** recopilación exhaustiva de material técnico sobre el RSA y otros temas sobre criptografía.
- **NIST Secure Hashing Page:** SHA FIPS y documentos relacionados.

3.8 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Autentificación de mensaje	Estándar de firma digital	MD5
Certificado de clave pública	(DSS)	Resumen de mensaje
Cifrado de clave pública	Firma digital	RIPEMD-160
Clave privada	Función <i>hash</i> segura	RSA
Clave pública	Función <i>hash</i> unidireccional	SHA-1
Clave secreta	HMAC	
Código de autentificación de mensaje (MAC)	Intercambio de clave	
Criptografía de curva elíptica (ECC)	Intercambio de clave de Diffie-Hellman	

PREGUNTAS DE REPASO

- 3.1.** Menciona tres enfoques para la autentificación de mensajes.
- 3.2.** ¿Qué es un código de autentificación de mensajes?
- 3.3.** Describe brevemente los tres esquemas que aparecen en la Figura 3.2.
- 3.4.** ¿Qué propiedades debe cumplir una función *hash* para que sea útil para la autentificación de mensajes?
- 3.5.** En el contexto de las funciones *hash*, ¿qué es una función de compresión?
- 3.6.** ¿Cuáles son los componentes principales de un criptosistema de clave pública?
- 3.7.** Menciona y define brevemente tres usos de un criptosistema de clave pública.
- 3.8.** ¿Cuál es la diferencia entre una clave privada y una clave secreta?
- 3.9.** ¿Qué es una firma digital?
- 3.10.** ¿Qué es un certificado de clave pública?
- 3.11.** ¿Cómo se puede usar el cifrado de clave pública para distribuir una clave secreta?

PROBLEMAS

- 31.** Uno de los MAC más usados, conocido como Algoritmo de Autentificación de Datos, se basa en el DES. El algoritmo es una publicación de FIPS (FIPS PUB 113) y un estándar ANSI (X9.17). El algoritmo usa el modo de operación de cadena de bloque de cifrado (CBC) del DES con un vector de inicialización con valor cero (Figura 2.7). Los datos (mensaje, registro, fichero o programa) que han de autenticarse están agrupados en bloques contiguos de 64 bits: P_1, P_2, \dots, P_N . En caso de ser necesario, el último bloque se rellena a la derecha con ceros para formar un bloque completo de 64 bits. El MAC está formado por el bloque entero de texto cifrado C_N o por los M bits más a la izquierda del bloque, con $16 \leq M \leq 64$. Demuestra que se puede obtener el mismo resultado usando el modo de retroalimentación de cifrado.
- 32.** Considérese una función *hash* de 32 bits definida como la concatenación de dos funciones de 16 bits: XOR y RXOR, definidas en la Sección 3.1 como «dos funciones *hash* simples».
- ¿Detectará esta suma de prueba todos los errores causados por un número impar de bits de error? Explícalo.
 - ¿Detectará esta suma de prueba todos los errores causados por un número par de bits de error? Si la respuesta es negativa, caracteriza los patrones de error que harán que falle la suma de prueba.
 - Comenta la efectividad de esta función para su uso como una función *hash* para la autentificación.
- 33.** Es posible emplear una función *hash* para construir un cifrado de bloque con una estructura similar al DES. Si una función *hash* es unidireccional y un cifrado de bloque debe ser reversible (para descifrar), ¿cómo es esto posible?
- 34.** Antes del descubrimiento de los esquemas específicos de clave pública, como el RSA, se desarrolló una prueba de existencia cuyo propósito era el de demostrar que, en teoría, el cifrado de clave pública es posible. Considera las funciones $f_1(x_i) = z_i$, $f_2(x_2, y_2) = z_2$, $f_3(x_3, y_3) = z_3$, donde todos los valores son enteros con $1 \leq x_i, y_i, z_i \leq N$. La función f_1 puede representarse por medio de un vector M_1 de longitud N , en el que la k -ésima entrada es el valor de $f_1(k)$. De igual forma, f_2 y f_3 pueden estar representados por matrices de $N \times N$, M_2 y M_3 . El objetivo es representar el proceso de cifrado/descifrado mediante consultas en tablas con valores muy altos de N . Dichas tablas serían innecesariamente enormes pero, en principio, podrían construirse. El esquema es como sigue: construye M_1 con una permutación aleatoria de todos los enteros entre 1 y N , es decir, cada entero aparece exactamente una vez en M_1 . Construye M_2 de forma que cada fila contenga una permutación aleatoria de los N primeros enteros. Por último, rellena M_3 para que se den las siguientes condiciones:

$$f_3(f_2(f_1(k), p), k) = p \quad \text{para todo } k, p \text{ con } 1 \leq k, p \leq N$$

Es decir:

1. M_1 toma una entrada k y produce una salida x .
2. M_2 toma las entradas x y p dando la salida z .
3. M_3 toma las entradas z y k y produce p .

Las tres tablas, una vez construidas, se hacen públicas.

- a)** Debería quedar claro que es posible construir M3 para cumplir la condición anterior. A modo de ejemplo, rellena M3 para este caso sencillo:

M1 =	5	M2 =	5 2 3 4 1 4 2 5 1 3 1 3 2 4 5 3 1 4 2 5 2 5 3 4 1	M3 =	
-------------	---	------	---	-------------	--

Convención: el i -ésimo elemento de M1 corresponde a $k = i$. La i -ésima fila de M2 corresponde a $x = i$; la j -ésima columna de M2 corresponde a $p = j$. La i -ésima fila de M3 corresponde a $z = i$; la j -ésima columna de M3 corresponde a $k = j$.

- b)** Describe el uso de este grupo de tablas para realizar cifrado y descifrado entre dos usuarios.
- c)** Razona que este es un esquema seguro.
- 3.5** Lleva a cabo el cifrado y el descifrado usando el algoritmo RSA, como en la Figura 3.9, para lo siguiente:
- a)** $p = 3; q = 11, e = 7; M = 5$
 - b)** $p = 5; q = 11, e = 3; M = 9$
 - c)** $p = 7; q = 11, e = 17; M = 8$
 - d)** $p = 11; q = 13, e = 11; M = 7$
 - e)** $p = 17; q = 31, e = 7; M = 2$. *Consejo:* El descifrado no es tan difícil como piensas; usa tu astucia.
- 3.6** En un sistema de clave pública que usa RSA, interceptas el texto cifrado $C = 10$, enviado a un usuario cuya clave pública es $e = 5, n = 35$. ¿Cuál es el texto claro M ?
- 3.7** En un sistema RSA, la clave pública de un usuario dado es $e = 31, n = 3599$. ¿Cuál es la clave privada de este usuario?
- 3.8** Supongamos que tenemos un grupo de bloques codificados con el algoritmo RSA y que no tenemos la clave privada. Supongamos que $n = pq$, e es la clave pública. ¿Nos ayuda de alguna forma saber que uno de los bloques del texto claro tiene un factor común con n ?
- 3.9** Demuestra cómo se puede representar RSA por medio de las matrices M1, M2 y M3 del Problema 3.4.
- 3.10** Considera el siguiente esquema:
- 1.** Tomar un número impar, E
 - 2.** Tomar dos números primos, P y Q , donde la división de $(P - 1)(Q - 1) - 1$ entre E es un número par.

3 Multiplicar P y Q para obtener N .

4 Calcular $D = \frac{(P-1)(Q-1)(E-1) + 1}{E}$

¿Es este esquema equivalente a RSA? Justifica tu respuesta.

- 3.11.** Considera el empleo del RSA con una clave conocida para construir una función *hash* unidireccional. Luego, procesa un mensaje formado por una secuencia de bloques como sigue: cifrar el primer bloque, aplicar el XOR al resultado con el segundo bloque y cifrar de nuevo, y así sucesivamente. Demuestra que este esquema no es seguro resolviendo el siguiente problema. Dado un mensaje de dos bloques B_1, B_2 y su *hash*

$$\text{RSAH}(B_1, B_2) = \text{RSA}(\text{RSA}(B_1) \oplus B_2)$$

Dado un bloque aleatorio C_1 , elige C_2 tal que $\text{RSAH}(C_1, C_2) = \text{RSAH}(B_1, B_2)$.

- 3.12.** Considera un esquema de Diffie-Hellman con un factor primo común $q = 11$ y una raíz primitiva $\alpha = 2$.
- a)** Si el usuario A tiene la clave pública $Y_A = 9$, ¿cuál es la clave privada X_A de A?
 - b)** Si el usuario B tiene la clave pública $Y_B = 3$, ¿cuál es la clave secreta compartida K ?

P A R T E II

APLICACIONES DE SEGURIDAD EN REDES

CONTENIDO DE LA SEGUNDA PARTE

En la Segunda Parte del libro se estudian importantes herramientas y aplicaciones de seguridad que pueden usarse en una sola red, en intranets o en Internet.

GUÍA PARA LA SEGUNDA PARTE

CAPÍTULO 4. APLICACIONES DE AUTENTIFICACIÓN

En el Capítulo 4 se estudian dos de las especificaciones de autentificación más importantes que se usan en la actualidad. Kerberos es un protocolo de autentificación basado en cifrado simétrico que ha recibido gran aceptación general y se usa en una variedad de sistemas. El estándar X.509 especifica un algoritmo de autentificación y define un certificado. Este último permite que los usuarios obtengan certificados de clave pública para que una comunidad de usuarios pueda confiar en la validez de las claves públicas. Esta herramienta se emplea como bloque de construcción en una serie de aplicaciones.

CAPÍTULO 5. SEGURIDAD EN EL CORREO ELECTRÓNICO

La aplicación distribuida que más se usa es el correo electrónico, y existe un aumento de interés en proporcionar servicios de autentificación y confidencialidad como parte de la herramienta de correo electrónico. En el Capítulo 5 se tratan los dos enfoques que podrían dominar la seguridad del correo electrónico en un futuro cercano. PGP (*Pretty Good Privacy*) es un esquema de uso extendido que no depende de ninguna organización o autoridad, por lo que es tan adecuado para el uso individual y personal como para la incorporación en configuraciones de red gestionadas por organizaciones. S/MIME

(*Secure/Multipurpose Internet Mail Extension*) se desarrolló concretamente para ser un estándar de Internet.

CAPÍTULO 6. SEGURIDAD IP

El IP (*Internet Protocol*) es el elemento central de Internet y las intranets privadas. La seguridad en el nivel IP, por lo tanto, es importante para el diseño de cualquier esquema de seguridad basada en Internet. El Capítulo 6 presenta el esquema de seguridad IP que se ha desarrollado para operar con el actual IP y con el IP de nueva generación que está surgiendo y que se conoce como IPv6.

CAPÍTULO 7. SEGURIDAD DE LA WEB

El aumento drástico del uso de la *World Wide Web* en el comercio electrónico y para difundir información ha generado la necesidad de la seguridad en la Web. El Capítulo 7 proporciona un estudio de este nuevo campo de la seguridad y se centra en dos estándares clave: el SSL (*Secure Sockets Layer*) y el SET (*Secure Electronic Transaction*).

CAPÍTULO 8. SEGURIDAD EN LA GESTIÓN DE REDES

El uso en aumento de los sistemas de gestión de redes para controlar redes que proporcionan servicios de diversa índole a un gran número de usuarios ha provocado, a su vez, un aumento en la demanda de capacidades de seguridad. El Capítulo 8 se centra en el esquema de gestión de redes más usado, el SNMP (*Simple Network Management Protocol*). Mientras la versión 1 del SNMP sólo tiene una herramienta de autentificación rudimentaria basada en contraseñas, el SNMPv2 aporta funciones adicionales, y el SNMPv3 proporciona una verdadera herramienta de seguridad para la confidencialidad y la autenticación, que se puede usar junto con SNMPv1 o SNMPv2.

CAPÍTULO 4

Aplicaciones de autentificación

4.1. Kerberos.

Motivación

Versión 4 de Kerberos

Versión 5 de Kerberos

4.2. Servicio de autentificación de X.509.

Certificados

Procedimientos de autentificación

La versión 3 de X.509

4.3. Bibliografía y sitios web recomendados.

4.4. Términos clave, preguntas de repaso y problemas.

Términos clave

Preguntas de repaso

Problemas

Apéndice 4A. Técnicas de cifrado Kerberos

Transformación de contraseña a clave

Propagación del modo CBC (*Cipher Block Chaining*)

No podemos aliarnos con los príncipes vecinos hasta que estemos familiarizados con sus diseños.

El arte de la guerra, Sun Tzu

Este Capítulo examina algunas de las funciones de autentificación que se han desarrollado para la autentificación y las firmas digitales en el nivel de aplicación.

En primer lugar, estudiaremos uno de los servicios más antiguos y de uso más extendido: Kerberos. Luego, examinaremos el servicio de autentificación de directorio del estándar X.509. Dicho estándar es importante como parte del servicio de directorio que soporta, pero también constituye un bloque básico en la construcción de otros estándares, como el S/MIME, que se trata en el Capítulo 5.

4.1 KERBEROS

Kerberos¹ es un servicio de autentificación que se desarrolló como parte del Proyecto Athena en el MIT. Fue diseñado para abordar el problema que plantea el hecho de que en un entorno abierto distribuido, los usuarios de las estaciones de trabajo quieran acceder a servicios de servidores distribuidos por toda la red. Sería conveniente que los servidores pudiesen restringir el acceso a los usuarios autorizados y autenticar las solicitudes de servicios. En este entorno no se puede confiar en que una estación de trabajo identifique a sus usuarios correctamente ante los servicios de red. Concretamente, se pueden dar las tres amenazas que se exponen a continuación:

- Un usuario podría obtener acceso a una estación de trabajo concreta y fingir ser otro usuario que opera desde esa estación de trabajo.
- Un usuario podría alterar la dirección de red de una estación de trabajo para que las solicitudes enviadas desde dicha estación parezcan proceder de la estación que ha sido suplantada.
- Un usuario podría realizar escuchas en intercambios y usar un ataque de repetición para conseguir entrar en un servidor o interrumpir las operaciones.

En cualquiera de estos casos, un usuario no autorizado podría obtener acceso a servicios y datos para los que no tenga autorización. En vez de crear protocolos elaborados de autentificación en cada servidor, Kerberos proporciona un servidor centralizado de autentificación cuya función es la de autenticar los usuarios al servidor y los servidores a los usuarios. A diferencia de la mayoría de los esquemas de autentificación descritos en este

¹ «En la mitología griega, Kerberos era un perro con varias cabezas, normalmente tres, y probablemente con cola de serpiente, que custodiaba la entrada del Hades» (*Dictionary of Subjects and Symbols In Art*, de James Hall, Harper & Row, 1979). Al igual que el Kerberos griego tenía tres cabezas, la idea inicial fue que el moderno tuviese también tres componentes para guardar la entrada a la red: (1) autentificación, (2) registro de operaciones y uso de recursos y (3) auditoría. Las dos últimas cabezas nunca llegaron a implementarse.

libro, Kerberos se basa exclusivamente en cifrado simétrico, dejando de lado el cifrado de clave pública.

Las versiones más comunes de Kerberos son dos: la versión 4 [MILL88, STEI88], muy usada todavía, y la versión 5 [KOHL94], que corrige algunas de las deficiencias de seguridad de la versión 4 y se ha propuesto como Estándar de Internet (RFC 1510)².

Esta sección comienza con una breve discusión sobre la motivación que originó el enfoque Kerberos. Luego, debido a la complejidad de Kerberos, es conveniente proceder a una descripción del protocolo de autentificación empleado en la versión 4, lo que nos permite observar la esencia de la estrategia de Kerberos sin tener en cuenta algunos de los detalles necesarios para manejar amenazas sutiles a la seguridad. Por último, examinaremos la versión 5.

MOTIVACIÓN

Si un grupo de usuarios dispone de computadores personales sin conexión a la red, los recursos y archivos de un usuario se pueden proteger asegurando físicamente cada computador. Si, por el contrario, el servidor de estos usuarios consiste en un sistema de tiempo compartido centralizado, es el sistema operativo de tiempo compartido el que debe proporcionar la seguridad. El sistema operativo puede reforzar las políticas de control de acceso basadas en la identidad del usuario y el uso de contraseñas para identificar para los usuarios.

Hoy en día, ninguno de estos casos es habitual. La arquitectura más común es la arquitectura distribuida formada por estaciones de trabajo de usuarios (clientes) y servidores distribuidos o centralizados. En este entorno, se pueden prever tres enfoques a la seguridad:

1. Confiar en que cada estación cliente individual asegure la identidad de su usuario o usuarios y en que cada servidor refuerce una política de seguridad que se apoye en la identificación del usuario (ID).
2. Exigir que los sistemas clientes se autentifiquen a los servidores, pero confiar en el sistema cliente en lo que respecta a la identidad de su usuario.
3. Exigir que el usuario demuestre su identidad para cada servicio solicitado y que los servidores también demuestren su identidad a los clientes.

En un entorno reducido y cerrado en el que todos los sistemas pertenecen a una única organización, la primera o, quizás, la segunda estrategia puede ser suficiente³. Pero en un entorno más abierto con conexiones de red a otras máquinas, se necesita el tercer enfoque para proteger la información y los recursos del usuario alojados en el servidor. Kerberos admite este tercer enfoque. Además, asume una arquitectura distribuida de cliente/servidor y emplea uno o más servidores Kerberos para proporcionar un servicio de autentificación.

El primer informe que se publicó sobre Kerberos [STEI88] presentaba los siguientes requisitos para Kerberos:

² Las versiones 1 a la 3 fueron versiones de desarrollo interno. La versión 4 es el Kerberos «original».

³ Sin embargo, incluso un sistema cerrado se enfrenta a la amenaza del ataque por parte de un empleado descontento.

- **Seguridad:** un observador de la red no debería poder obtener la información necesaria para hacerse pasar por un usuario. Es decir, Kerberos debería ser lo suficientemente robusto para que un posible oponente no lo considere un punto débil.
- **Fiabilidad:** para todos los servicios que utilizan Kerberos para el control de acceso, la falta de disponibilidad del servicio de Kerberos implica la falta de disponibilidad de los servicios que se proporcionan. Así, Kerberos debería ser muy fiable y emplear una arquitectura de servidores distribuida en la que un sistema pudiera disponer de copias de otro.
- **Transparencia:** aparte del requisito de introducir una contraseña, es preferible que el usuario no sea consciente de que está teniendo lugar la autentificación.
- **Escalabilidad:** el sistema debería poder dar cabida a un gran número de clientes y servidores, lo cual sugiere una arquitectura distribuida modular.

Para lograr estos requisitos, el esquema general de Kerberos es el de un servicio de autentificación de una tercera parte confiable. Es confiable en el sentido de que los clientes y los servidores confían en Kerberos para que medie en su autentificación mutua. Suponiendo que el protocolo de Kerberos está bien diseñado, el servicio de autentificación es seguro si el servidor Kerberos es seguro en sí mismo⁴.

VERSIÓN 4 DE KERBEROS

La versión 4 de Kerberos hace uso del DES, en un protocolo bastante elaborado, para proporcionar el servicio de autentificación. Considerando el protocolo en su conjunto, es difícil entender la necesidad de muchos de los elementos que contiene. Por lo tanto, adoptamos una estrategia empleada por Bill Bryant del Proyecto Athena [BRYA88] y desarrollamos el protocolo completo observando primero varios diálogos hipotéticos. Cada diálogo sucesivo presenta una complejidad adicional a las vulnerabilidades a la seguridad reveladas en el diálogo anterior.

Después de examinar el protocolo, trataremos otros aspectos de la versión 4.

Un diálogo de autentificación simple

En un entorno de red sin protección, cualquier cliente puede solicitar un servicio a cualquier servidor. El riesgo más evidente que corre la seguridad es el de la suplantación. Un oponente puede fingir ser otro cliente y obtener privilegios no autorizados en las máquinas del servidor. Para contrarrestar esta amenaza, los servidores deben poder confirmar las identidades de los clientes que soliciten el servicio. Se puede requerir que un

⁴ No debemos olvidar que la seguridad del servidor Kerberos no debería asumirse de forma automática, sino que debería ser vigilada cuidadosamente. Es conveniente recordar el destino del Kerberos griego, a quien Hércules capturó por orden de Eurystheus: «Hércules encontró al enorme perro encadenado y lo agarró por el cuello. Enseguida las tres cabezas intentaron atacar, y Kerberos daba latigazos con su poderosa cola. Pero Hércules aguantó con firmeza y Kerberos perdió la conciencia. Eurystheus pudo haber quedado sorprendido al ver a Hércules vivo —cuando él vio las tres cabezas babeantes y el enorme perro al que pertenecían se aterrorizó, y de un salto se refugió en su gran tinaja de bronce.» (*The Hamlyn Concise Dictionary of Greek and Roman Mythology*, de Michael Stapleton, Hamlyn, 1982).

servidor realice esta tarea para cada interacción cliente/servidor, pero en un entorno abierto, este hecho supone una carga considerable en cada servidor.

Una alternativa consiste en el uso de un servidor de autentificación (AS: *Authentication Server*) que conozca las claves de todos los usuarios y las almacene en una base de datos centralizada. Además, el AS comparte una única clave secreta con cada servidor. Estas claves han sido distribuidas físicamente o de otra forma segura. Consideremos el siguiente diálogo hipotético⁵:

- (1)** C → AS: $ID_C \parallel P_C \parallel ID_V$
- (2)** AS → C: *Ticket*
- (3)** C → V: $ID_C \parallel Ticket$
 $Ticket = E_{K_V} [ID_C \parallel AD_C \parallel ID_V]$

donde

C = cliente

AS = servidor de autentificación (*Authentication Server*)

V = servidor

ID_C = identificador de usuario en C

ID_V = identificador de V

P_C = contraseña de usuario de C

AD_C = dirección de red de C

K_V = clave secreta de cifrado compartida por AS y V

\parallel = concatenación

En este marco hipotético, el usuario se conecta a una estación de trabajo y solicita acceso al servidor V. El módulo del cliente C en la estación de trabajo del usuario solicita la contraseña del usuario y luego envía al AS un mensaje que incluye el ID del usuario, el ID del servidor y la contraseña del usuario. El AS comprueba su base de datos para verificar si el usuario ha dado la contraseña correspondiente a su ID de usuario y si este usuario tiene permiso para acceder al servidor V. Si se superan las dos comprobaciones, el AS acepta al usuario como auténtico y debe convencer al servidor de dicha autenticidad. Para ello, el AS crea un ticket que contiene el ID y la dirección de red del usuario y el ID del servidor. Este ticket se cifra usando la clave secreta que comparten el AS y este servidor. Luego, este ticket se devuelve a C. Como el ticket está cifrado, no puede ser alterado por C ni por ningún oponente.

Ahora, con este ticket C puede solicitar un servicio a V. C envía a V un mensaje que contiene el ID de C y el ticket. V desencripta el ticket y verifica que el ID del usuario en el ticket es el mismo que el ID sin cifrar en el mensaje. Si estos dos coinciden, el servidor considera al usuario autenticado y concede el servicio solicitado.

Cada uno de los componentes del mensaje (3) es significativo. El ticket se cifra para evitar modificación o falsificación. El ID del servidor (ID_V) está incluido en el ticket para que el servidor pueda verificar que ha descifrado el ticket de forma correcta. El ID_C está incluido en el ticket para indicar que este ticket ha sido emitido por parte de C. Por

⁵ La parte de la izquierda de los dos puntos se refiere al emisor y al receptor; la parte de la derecha indica los contenidos del mensaje; el símbolo \parallel indica concatenación.

último, el AD_C sirve para cotrarrestar la siguiente amenaza: un oponente podría capturar el ticket transmitido en el mensaje (2), luego usar el nombre ID_C y transmitir un mensaje de la forma (3) desde otra estación de trabajo. El servidor recibiría un ticket válido que coincide con el ID del usuario y concede acceso al usuario en esa otra estación de trabajo. Para prevenir este ataque el AS incluye en el ticket la dirección de red desde la que se envió la solicitud original y así el ticket es válido sólo si se transmite desde la misma estación de trabajo que inicialmente solicitó el ticket.

Un diálogo de autenticación más seguro

Aunque el marco anterior resuelve algunos de los problemas de autenticación en un entorno de red abierto, todavía quedan problemas. Sobresalen especialmente dos de ellos. En primer lugar, nos gustaría reducir el número de veces que un usuario tiene que introducir la contraseña. Supongamos que cada ticket se puede usar una sola vez. Si el usuario C se conecta a una estación de trabajo por la mañana y desea comprobar su correo en un servidor de correo, debe proporcionar una clave de acceso para obtener un ticket para dicho servidor. Si C desea comprobar su correo varias veces al día, cada intento exige volver a introducir la contraseña. Este hecho se puede mejorar si los tickets son reutilizables. Para una sola sesión, la estación de trabajo puede almacenar el ticket del servidor de correo después de recibirllo y usarlo en nombre del usuario para múltiples accesos al servidor de correo.

Sin embargo, en este esquema queda todavía el caso en el que un usuario necesitaría un nuevo ticket para cada uno de los distintos servicios. Si un usuario quisiera acceder a un servidor de impresión, un servidor de correo, un servidor de ficheros, etc., al principio de cada acceso se requeriría un nuevo ticket y, por consiguiente, exigiría que el usuario introdujera la clave de acceso.

El segundo problema consiste en que el contexto descrito implica la transmisión de la contraseña en texto claro [mensaje (1)]. Un escucha podría capturar la contraseña y usar cualquier servicio accesible a la víctima.

Para resolver estos problemas adicionales, introducimos un esquema para evitar las contraseñas en texto claro, y un nuevo servidor, el servidor que concede el ticket, conocido como TGS (*ticket-granting server*). El nuevo marco hipotético es el siguiente:

Una vez por sesión de usuario:

- (1) C → AS: $ID_C \parallel ID_{tgs}$
- (2) AS → C: $E_{K_c} [Ticket_{tgs}]$

Una vez por tipo de servicio:

- (3) C → TGS: $ID_C \parallel ID_V \parallel Ticket_{tgs}$
- (4) TGS → C: $Ticket_v$

Una vez por sesión de servicio:

- (5) C → V: $ID_C \parallel Ticket_v$

$$Ticket_{tgs} = E_{K_{tgs}} [ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_1 \parallel Tiempo\ de\ vida_1]$$

$$Ticket_v = E_{K_V} [ID_C \parallel AD_C \parallel ID_v \parallel TS_2 \parallel Tiempo\ de\ vida_2]$$

El nuevo servicio, TGS, emite los tickets a los usuarios que han sido autenticados al AS. Así, el usuario primero solicita al AS un TGT (*Ticket-Granting Ticket*) ticket que concede un ticket ($Ticket_{tgt}$). El módulo de cliente almacena este ticket en la estación de trabajo del usuario. Cada vez que el usuario requiera acceso a un nuevo servicio, el cliente se dirige al TGS, usando el ticket para autenticarse a sí mismo. Luego el TGS concede el ticket para el servicio concreto. El cliente guarda cada ticket de servicio y lo usa para autenticar su usuario a un servidor cada vez que se solicite un servicio en particular. Veamos los detalles de este esquema:

1. El cliente solicita un TGT en nombre del usuario enviando al AS su ID de usuario junto con el ID del TGS, indicando una solicitud para usar el servicio del TGS.
2. El AS responde con un ticket que está cifrado con una clave derivada de la contraseña del usuario. Cuando esta respuesta llega al cliente, el cliente pide al usuario su contraseña, genera la clave e intenta descifrar el mensaje que llega. Si se proporciona la clave de acceso correcta, el ticket se recupera con éxito.

Como sólo el usuario correcto podría conocer la clave de acceso, sólo el usuario correcto puede recuperar el ticket. Así, hemos usado la contraseña para obtener credenciales de Kerberos sin tener que transmitir la clave de acceso en texto claro. El ticket consiste en el ID y la dirección de red del usuario y el ID del TGS. Esto corresponde al primer contexto. La idea es que este ticket lo pueda usar el cliente para solicitar múltiples tickets que conceden servicios. Por lo tanto, el TGT ha de ser reutilizable. Sin embargo, no queremos que un oponente pueda capturar el ticket y usarlo. Consideremos el siguiente caso: un oponente captura el ticket y espera hasta que el usuario haya desconectado su estación de trabajo. Entonces, el oponente obtiene acceso a esa estación de trabajo o configura su estación con la misma dirección de red que la víctima. El oponente podría reutilizar el ticket para burlar al TGS. Para contrarrestar esta posibilidad, el ticket incluye un sello de tiempo (*timestamp*) con indicación de la fecha y la hora en que se emitió el ticket y un tiempo de vida que indica el período de validez del ticket (por ejemplo, ocho horas). De esta forma, el cliente ahora tiene un ticket reutilizable y no tiene que pedir al usuario una contraseña para cada servicio que solicite. Por último, hay que tener en cuenta que el TGT se cifra con una clave secreta que sólo conocen el AS y el TGS, para evitar alteraciones en el ticket. El ticket se vuelve a cifrar con una clave basada en la contraseña del usuario. Esto garantiza que el ticket puede ser recuperado únicamente por el usuario correcto, proporcionando autenticación.

Ahora que el cliente tiene un TGT, el acceso a cualquier servidor se puede obtener por medio de los pasos 3 y 4:

3. El cliente solicita en nombre del usuario un ticket que concede un servicio. Con este fin, el cliente transmite un mensaje al TGS que contiene el ID del usuario, el ID del servicio deseado y el TGT.
4. El TGS descifra el ticket recibido y verifica el éxito del descifrado con la presencia de su ID. Comprueba que el tiempo de vida (*lifetime*) no ha expirado. Luego compara el ID del usuario y la dirección de red con la información recibida para autenticar al usuario. Si el usuario tiene permiso para acceder a V, el TGS emite un ticket para conceder acceso al servicio solicitado.

El ticket que concede un servicio tiene la misma estructura que el TGT. De hecho, como el TGS es un servidor, sería de esperar que se necesitaran los mismos elementos para au-

tentificar un cliente al TGS y para autenticar un cliente a un servidor de aplicaciones. Nuevamente, el ticket contiene un sello de tiempo y un tiempo de vida. Si el usuario quiere acceder más tarde al mismo servicio, el cliente puede usar simplemente el ticket que concede servicio que se obtuvo previamente y no tiene que pedir al usuario una contraseña. Obsérvese que el ticket se cifra con una clave secreta (K_V) que sólo es conocida por el TGS y el servidor, para evitar alteraciones.

Por último, con un ticket concreto que conceda un servicio, el cliente puede acceder al servicio correspondiente mediante el paso 5:

5. El cliente solicita acceso a un servicio en nombre del usuario. Con este fin, el cliente transmite al servidor un mensaje que contiene el ID del usuario y el ticket que concede el servicio. El servidor autentifica usando los contenidos del ticket.

Este marco satisface los dos requisitos: el de una única contraseña por sesión de usuario y la protección de la contraseña del usuario.

El diálogo de autenticación de la versión 4

Aunque el marco previo mejora la seguridad en relación con el primer intento, quedan por resolver dos problemas adicionales. La base del primer problema es el tiempo de vida asociado al TGT. Si este tiempo de vida es muy corto (por ejemplo, unos minutos), entonces se pedirá al usuario la contraseña repetidamente. Si el tiempo de vida es largo (por ejemplo, horas), entonces un oponente tiene muchas posibilidades de atacar por repetición. Un oponente podría tener escuchas en la red, capturar una copia del TGT y esperar a que el usuario legítimo se desconecte. Entonces el oponente podría falsificar la dirección de red del usuario legítimo y enviar el mensaje del paso (3) al TGS. Esto supondría para el oponente un acceso ilimitado a los recursos y archivos que están disponibles al usuario legítimo.

De la misma manera, si un oponente capture un ticket que concede un servicio y lo usa antes de que expire, tendrá acceso al servicio correspondiente.

Así, llegamos a un requisito adicional. Un servicio de red (el TGS o un servicio de aplicaciones) debe ser capaz de comprobar que la persona que usa el ticket es la misma persona a la que se emitió ese ticket.

El segundo problema consiste en que podría existir el requisito de que los servidores se autenticaran a sí mismos a los usuarios. Sin esta autenticación, un oponente podría sabotear la configuración para que los mensajes destinados a un servidor se dirigieran a otra localización. El servidor falso estaría entonces en situación de actuar como un servidor real, capturar cualquier información del usuario y rechazar el servicio real al usuario.

Examinamos estos problemas y nos referimos a la Tabla 4.1, que muestra el protocolo real de Kerberos.

En primer lugar, consideraremos el problema de los TGT capturados y la necesidad de determinar que el que presenta el ticket es el mismo que el cliente para el que se emitió dicho ticket. La amenaza consiste en que un oponente puede robar el ticket y usarlo antes de que expire. Para solventar este problema, supongamos que el AS proporciona tanto al cliente como al TGS una parte de información secreta de forma segura. Entonces el cliente puede probar su identidad al TGS revelando la información secreta, nueva-

mente de forma segura. Una manera eficaz de llevar esto a cabo es usar una clave de cifrado como información segura, que en Kerberos se denomina clave de sesión.

La Tabla 4.1 muestra la técnica para distribuir la clave de sesión. Como anteriormente, el cliente envía un mensaje al AS solicitando acceso al TGS. El AS responde con un mensaje, cifrado con una clave derivada de la contraseña del usuario (K_c) que contiene el ticket. El mensaje cifrado también contiene una copia de la clave de sesión, $K_{c,tgs}$, donde los subíndices indican que se trata de una clave de sesión para C y TGS. Como esta clave de sesión está en el mensaje cifrado con K_c , sólo el cliente del usuario puede leerla. La misma clave de sesión se incluye en el ticket, que sólo podrá leer el TGS. Por lo tanto, la clave de sesión se ha distribuido de forma segura tanto a C como a TGS.

Antes de continuar, obsérvese que a esta primera fase del diálogo se han añadido partes adicionales de información. El mensaje (1) incluye un sello de tiempo para que el AS sepa que el mensaje es oportuno. El mensaje (2) incluye algunos elementos del ticket accesibles a C. Esto permite a C confirmar que este ticket es para el TGS, así como conocer su fecha de expiración.

Con el ticket y la clave de sesión, C está preparado para dirigirse al TGS. Como antes, C envía al TGS un mensaje que incluye el ticket y el ID del servicio solicitado [mensaje (3) en la Tabla 4.1b]. Además, C transmite un autentificador que incluye el ID

Tabla 4.1 Resumen de los intercambios de mensajes de la versión 4 de Kerberos

(a) Intercambio de servicio de autentificación: para obtener un TGT
<p>(1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$</p> <p>(2) AS → C: $E_{K_c} [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Tiempo\ de\ vida_2 \parallel Ticket_{tgs}]$ $Ticket_{tgs} = E_{K_{tgs}} [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Tiempo\ de\ vida_2]$</p>
(b) Intercambio de TGS: para obtener un ticket que concede un servicio
<p>(3) C → TGS: $ID_v \parallel Ticket_{tgs} \parallel Autentificador_c$</p> <p>(4) TGS → C: $E_{K_{c,tgs}} [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v]$ $Ticket_{tgs} = E_{K_{tgs}} [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Tiempo\ de\ vida_2]$ $Ticket_v = E_{K_v} [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Tiempo\ de\ vida_4]$ $Autentificador_c = E_{K_{c,tgs}} [ID_C \parallel AD_C \parallel TS_3]$</p>
(c) Intercambio de autentificación cliente/servidor: para obtener un servicio
<p>(5) C → V: $Ticket_v \parallel Autentificador_c$</p> <p>(6) V → C: $E_{K_{c,v}} [TS_5 + 1]$ (para autentificación mutua) $Ticket_v = E_{K_v} [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Tiempo\ de\ vida_4]$ $Autentificador_c = E_{K_{c,v}} [ID_C \parallel AD_C \parallel TS_5]$</p>

y la dirección del usuario de C y un sello de tiempo. A diferencia del ticket, que es reutilizable, el autentificador está destinado para un único uso y tiene un tiempo de vida muy corto. El TGS puede descifrar el ticket con la clave que comparte con el AS. Este ticket indica que se ha proporcionado la clave de sesión $K_{c,tgs}$ al usuario C. De hecho, el ticket dice «Cualquiera que use $K_{c,tgs}$ debe ser C». El TGS usa la clave de sesión para descifrar el autentificador. El TGS, luego, puede comprobar el nombre y la dirección del autentificador con el del ticket y con la dirección de red del mensaje recibido. Si todo coincide, entonces el TGS está seguro de que el emisor del ticket es el auténtico propietario del ticket. En efecto, el autentificador dice «En la fecha TS_3 , uso $K_{c,tgs}$ ». Obsérvese que el ticket no prueba la identidad de nadie, sino que se trata de una forma segura de distribuir las claves. Es el autentificador el que demuestra la identidad del cliente. Como el autentificador se puede usar una sola vez y tiene un tiempo de vida corto, se contrarresta la amenaza de un oponente que robe tanto el ticket como el autentificador para su posterior presentación.

La respuesta desde el TGS, en el mensaje (4), sigue la forma del mensaje (2). El mensaje se cifra con la clave de sesión compartida por el TGS y C e incluye una clave de sesión que ha de ser compartida por C y el servidor V, el ID de V, y el sello de tiempo del ticket. El ticket incluye la misma clave de sesión.

Ahora C tiene un ticket que concede un servicio reutilizable para V. Cuando C presenta este ticket, como se muestra en el mensaje (5), también envía un autentificador. El servidor puede descifrar el ticket, recuperar la clave de sesión y descifrar el autentificador.

Si se requiere autenticación mutua, el servidor puede responder como se muestra en el mensaje (6) de la Tabla 4.1. El servidor devuelve el valor del sello de tiempo del autentificador, incrementado en 1 y cifrado en la clave de sesión. C puede descifrar este mensaje para recuperar el sello de tiempo incrementado. Debido a que el mensaje fue cifrado por la clave de sesión, C está seguro de que sólo pudo haber sido creado por V. Los contenidos del mensaje garantizan a C que no se trata de la repetición de una respuesta antigua.

Por último, en la conclusión de este proceso, el cliente y el servidor comparten una clave secreta. Esta clave puede usarse para cifrar futuros mensajes entre los dos o para intercambiar una nueva clave de sesión aleatoria con ese propósito.

La Tabla 4.2 resume la justificación para cada uno de los elementos del protocolo Kerberos, y la Figura 4.1 proporciona una visión simplificada de la acción.

Tabla 4.2 Base lógica para los elementos del protocolo de la versión 4 de Kerberos

(a) Intercambio del servicio de autenticación	
Mensaje (1)	El cliente solicita el TGT
ID_C :	Dice al AS la identidad del usuario desde este cliente
ID_{tgs} :	Dice a AS que el usuario solicita acceso al TGS
TS_1 :	Permite que AS verifique que el reloj del cliente está sincronizado con el del AS

(continúa)

Tabla 4.2 Base lógica para los elementos del protocolo de la versión 4 de Kerberos (*continuación*)

Mensaje (2)	AS devuelve el TGT
E_{Kc} :	El cifrado se basa en la contraseña del usuario, permitiendo que AS y el cliente verifiquen la contraseña, y protegiendo los contenidos del mensaje (2)
$K_{c,tgs}$:	Copia de la clave de sesión accesible al cliente; creada por el AS para permitir un intercambio seguro entre el cliente y el TGS sin exigirles que comparten una clave permanente
ID_{tgs} :	Confirma que este ticket es para el TGS
TS_2 :	Informa al cliente del momento en que este ticket fue emitido
<i>Tiempo de vida</i> ₂ :	Informa al cliente sobre el tiempo de vida de este ticket
$Ticket_{tgs}$:	Ticket que va a usar el cliente para acceder al TGS
(b) Intercambio de TGS	
Mensaje (3)	El cliente solicita un ticket que concede un servicio
ID_V :	Dice al TGS que el usuario solicita acceso al servidor V
$Ticket_{tgs}$:	Garantiza al TGS que este usuario ha sido autenticado por el AS
$Autentificador_c$:	Generado por el cliente para validar el ticket
Mensaje (4)	El TGS devuelve el ticket que concede un servicio
$E_{Kc,tgs}$:	Clave compartida sólo por C y TGS; protege los contenidos del mensaje (4)
$K_{c,tgs}$:	Copia de la clave de sesión accesible al cliente; creada por TGS para permitir un intercambio seguro entre el cliente y el servidor sin exigirles que comparten una clave permanente
ID_V :	Confirma que este ticket es para el servidor V
TS_4 :	Informa al cliente del momento en que este ticket fue emitido
$Ticket_V$:	Ticket que ha de usar el cliente para acceder al servidor V
$Ticket_{tgs}$:	Reutilizable para que el usuario no tenga que volver a introducir una clave de acceso
E_{Ktgs} :	El ticket se cifra con una clave que es conocida sólo por AS y TGS, para evitar falsificación
$K_{c,tgs}$:	Copia de la clave de sesión accesible a TGS; usada para descifrar el autentificador, autenticando por lo tanto el ticket
ID_C :	Indica el propietario correcto de este ticket

(continúa)

Tabla 4.2 Base lógica para los elementos del protocolo de la versión 4 de Kerberos (*continuación*)

AD_C :	Evita que el ticket se use desde otra estación de trabajo que no sea la que inicialmente lo solicitó
ID_{tgs} :	Garantiza al servidor que ha descifrado el ticket correctamente
TS_2 :	Informa al TGS del momento en que este ticket fue emitido
<i>Tiempo de vida</i> ₂ :	Evita la repetición una vez que el ticket ha expirado
<i>Autentificador</i> _C :	Garantiza al TGS que el que presenta el ticket es el mismo que el cliente para el cual se emitió el ticket; tiene un tiempo de vida muy corto para evitar repeticiones
(b) Intercambio de TGS (<i>continuación</i>)	
$E_{Kc,tgs}$:	El autentificador se cifra con una clave conocida sólo por el cliente y el TGS, para evitar falsificaciones
ID_C :	Debe coincidir con el ID que aparece en el ticket para autenticar el ticket
AD_C :	Debe coincidir con la dirección que aparece en el ticket para autenticar el ticket
TS_2 :	Informa al TGS del momento en que se generó este autentificador
(c) Intercambio de autenticación cliente/servidor	
Mensaje (5) El cliente solicita un servicio	
<i>Ticket_V</i>	Garantiza al servidor que este usuario ha sido autenticado por AS
<i>Autentificador</i> _C :	Generado por el cliente para validar el ticket
Mensaje (6) Autenticación opcional del servidor al cliente	
E_{Kcv} :	Garantiza a C que este mensaje proviene de V
TS_{5+1} :	Garantiza a C que no se trata de una repetición de una respuesta antigua
<i>Ticket_V</i> :	Reutilizable para que el cliente no tenga que solicitar un nuevo ticket al TGS para cada acceso al mismo servidor
E_{Kv} :	El ticket se cifra con una clave conocida sólo por el TGS y el servidor, para evitar falsificación
$K_{c,v}$:	Copia de la clave de sesión accesible al cliente; usada para descifrar el autentificador, autenticando por lo tanto el ticket
ID_C :	Indica el propietario correcto de este ticket

(continúa)

Tabla 4.2 Base lógica para los elementos del protocolo de la versión 4 de Kerberos (*continuación*)

AD_C :	Evita el uso del ticket desde una estación de trabajo que no sea la que inicialmente lo solicitó
ID_V :	Garantiza al servidor que ha descifrado el ticket correctamente
TS_4	Informa al servidor del momento en que se emitió el ticket
<i>Tiempo de vida₄</i> :	Evita repeticiones una vez que el ticket ha expirado
<i>Autentificador_C</i> :	Garantiza al servidor que el que presenta el ticket es el mismo que el cliente para el cual se emitió; tiene un tiempo de vida muy corto para evitar repeticiones
$E_{K_{C,V}}$:	El autentificador se cifra con una clave conocida sólo por el cliente y el servidor, para evitar falsificaciones
ID_C :	Debe coincidir con el ID que aparece en el ticket para autenticar el ticket
AD_C :	Debe coincidir con la dirección que aparece en el ticket para autenticar el ticket
TS_5 :	Informa al servidor del momento en que se generó este autentificador

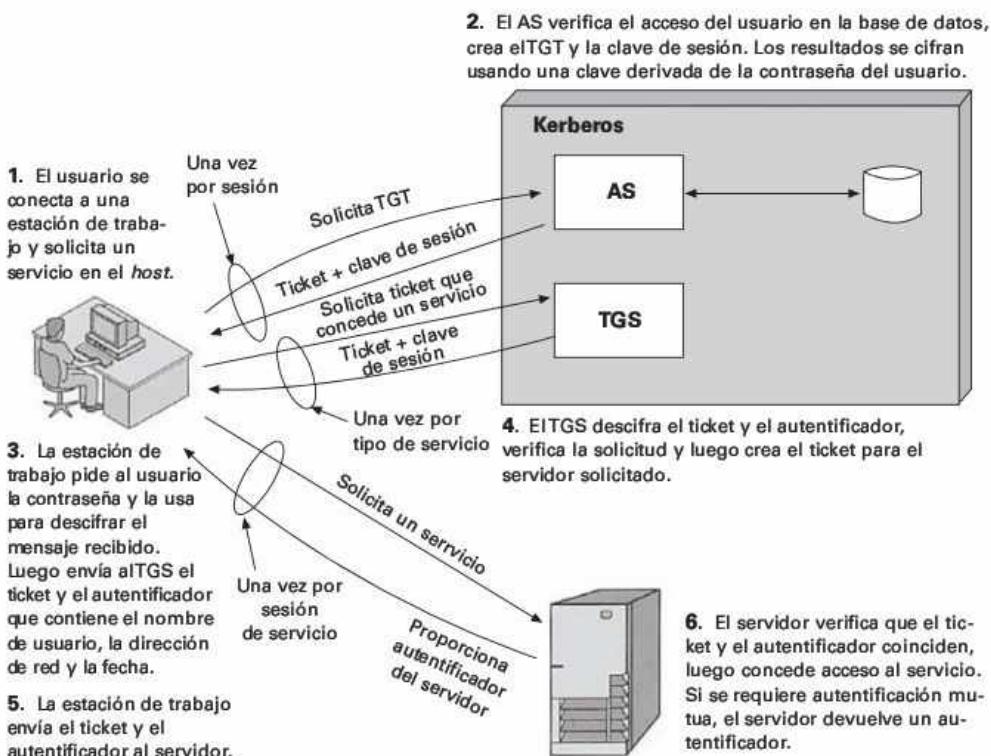


Figura 4.1 Esquema general de Kerberos

DOMINIOS DE KERBEROS Y KERBEROS MÚLTIPLES

Un entorno de Kerberos con todo tipo de servicios formado por un servidor Kerberos, una serie de clientes y un grupo de servidores de aplicaciones necesitan lo siguiente:

1. El servidor Kerberos debe tener el ID de usuario y la contraseña de todos los usuarios en esta base de datos. Todos los usuarios están registrados con el servidor Kerberos.
2. El servidor Kerberos debe compartir una clave secreta con cada servidor. Todos los servidores están registrados con el servidor Kerberos.

Un entorno como este se conoce como **dominio**. Las redes de clientes y servidores pertenecientes a diferentes organizaciones administrativas constituyen comúnmente distintos dominios. Es decir, en general no es práctico, o no se ajusta a la política administrativa, tener usuarios y servidores en un dominio administrativo registrados con un servidor Kerberos de otro dominio. Sin embargo, los usuarios de un dominio pueden necesitar acceso a servidores en otros dominios, y algunos servidores pueden estar dispuestos a proporcionar el servicio a usuarios de otros dominios si dichos usuarios se autentifican.

Kerberos proporciona un mecanismo para permitir la autentificación entre dominios. Para que dos dominios permitan la autentificación entre ellos, se añade un tercer requisito:

3. El servidor Kerberos en cada uno de los dominios que interoperan comparte una clave secreta con el servidor del otro dominio. Los dos servidores Kerberos se registran entre sí.

El esquema requiere que el servidor Kerberos de un dominio confíe en el servidor Kerberos del otro dominio para autenticar sus usuarios. Además, los servidores en el segundo dominio también deben estar dispuestos a confiar en el servidor Kerberos del primer dominio.

Teniendo en cuenta estas normas, podemos describir el mecanismo de la siguiente manera (Figura 4.2): un usuario que quiera un servicio de un servidor en otro dominio necesita un ticket para ese servidor. El cliente del usuario sigue los procedimientos habituales para acceder al TGS local y luego solicita un TGT para un TGS remoto (un TGS en otro dominio). Luego, el cliente puede solicitar al TGS remoto un ticket que concede un servicio para el servidor deseado en el dominio del TGS remoto.

Los detalles de los intercambios ilustrados en la Figura 4.2 son los siguientes (comparar con la Tabla 4.1):

- (1) C → AS: $ID_C \parallel ID_{tgs} \parallel TS_1$
- (2) AS → C: $E_{Kc} [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Tiempo\ de\ vida_2 \parallel Ticket_{tgs}]$
- (3) C → TGS: $ID_{tgsrem} \parallel Ticket_{tgs} \parallel Autentificador_c$
- (4) TGS → C: $E_{Kc,tgs} [K_{c,tgsrem} \parallel ID_{tgsrem} \parallel TS_4 \parallel Ticket_{tgsrem}]$
- (5) C → TGS_{rem}: $ID_{vrem} \parallel Ticket_{tgsrem} \parallel Autentificador_c$
- (6) TGS_{rem} → C: $E_{Kc,tgsrem} [K_{c,vrem} \parallel ID_{vrem} \parallel TS_6 \parallel Ticket_{vrem}]$
- (7) C → V_{rem}: $Ticket_{vrem} \parallel Autentificador_c$

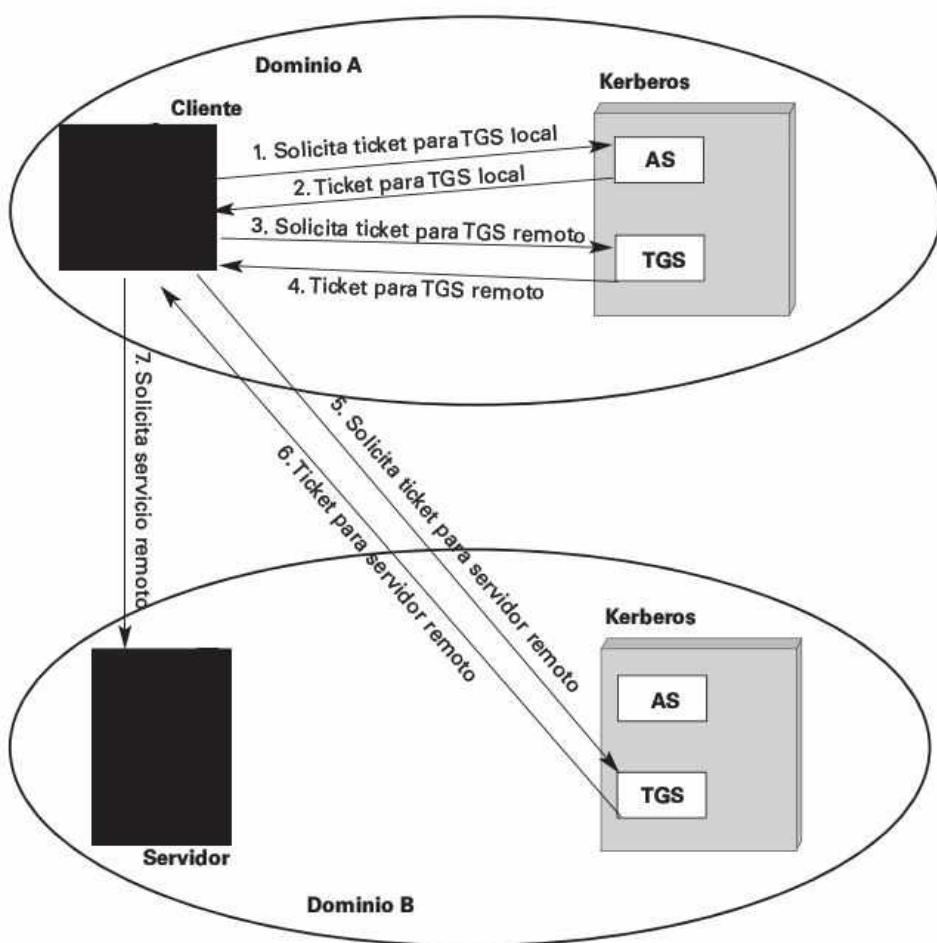


Figura 4.2 Solicitud de servicio en otro dominio

El ticket presentado al servidor remoto (V_{rem}) indica el dominio en el que el usuario fue autenticado originalmente. El servidor decide si acepta la solicitud remota.

Un problema que presenta el enfoque anterior es que no puede abarcar muchos dominios. Si hay N dominios, entonces debe haber $N(N - 1)/2$ intercambios de clave seguros para que cada dominio de Kerberos pueda interoperar con todos los demás dominios.

VERSIÓN 5 DE KERBEROS

La versión 5 de Kerberos se especifica en el RFC 1510 y proporciona una serie de mejoras con respecto a la versión 4 [KOHL94]. En primer lugar, consideraremos una visión general de los cambios producidos desde la versión 4 a la 5 y posteriormente nos centraremos en el protocolo de la versión 5.

Diferencias entre las versiones 4 y 5

La versión 5 está diseñada para superar las limitaciones de la versión 4 en dos áreas determinadas: deficiencias de entorno y deficiencias técnicas. Vamos a resumir brevemente las mejoras realizadas en cada área⁶.

La versión 4 de Kerberos se desarrolló para su uso en el entorno del proyecto Athena y, por consiguiente, no cubría del todo la necesidad de ser de propósito general. Esto originó los siguientes **fallo de entorno**:

- 1. Dependencia del sistema de cifrado:** la versión 4 requiere el uso del DES. La restricción de exportación del DES y las dudas sobre su robustez son, por ello, aspectos de interés. En la versión 5, al texto cifrado se le añade un identificador del tipo de cifrado para que pueda usarse cualquier técnica de cifrado. A las claves de cifrado se les añade un tipo y una longitud, permitiendo que la misma clave se emplee en diferentes algoritmos y haciendo posible la especificación de diferentes variaciones en un algoritmo dado.
- 2. Dependencia del protocolo de Internet:** la versión 4 requiere el uso de direcciones IP (*Internet Protocol*). Otros tipos de direcciones, como la dirección de red ISO, no tienen cabida. En cambio, las direcciones de red de la versión 5 están marcadas con tipo y longitud, permitiendo así el uso de cualquier tipo de dirección.
- 3. Ordenación de los bytes del mensaje:** en la versión 4, el emisor de un mensaje emplea un orden de bytes de su elección y añade al mensaje un indicador del byte menos significativo en la dirección más baja o el byte más significativo en la dirección más alta. Esta técnica funciona pero no sigue convenciones establecidas. En la versión 5, todas las estructuras de mensaje se definen usando ASN.1 (*Abstract Syntax Notation One*) y BER (*Basic Encoding Rules*), que proporcionan una ordenación de bytes sin ambigüedad.
- 4. Tiempo de vida del ticket:** los valores del tiempo de vida en la versión 4 se codifican en cantidades de ocho bits en unidades de cinco minutos. Así, el tiempo de vida máximo que puede expresarse es $2^8 \times 5 = 1280$ minutos, o algo más de 21 horas. Esto puede ser inadecuado para algunas aplicaciones (por ejemplo, una simulación de larga ejecución que requiera credenciales válidas de Kerberos a lo largo de la ejecución). En la versión 5, los tickets incluyen una fecha explícita de comienzo y de fin, permitiendo que los tickets tengan tiempos de vida arbitrarios.
- 5. Envío de autenticación:** la versión 4 no permite que las credenciales emitidas a un cliente sean enviadas a otro *host* y usadas por otro cliente. En cambio, la versión 5 sí permite que un cliente tenga acceso a un servidor y que ese servidor tenga acceso a otro servidor en nombre del cliente. Por ejemplo, un cliente envía una solicitud a un servidor de impresión que luego accede al fichero del cliente desde un servidor de ficheros, usando las credenciales del cliente para acceder.
- 6. Autenticación entre dominios:** en la versión 4, la interoperabilidad entre N dominios necesita del orden de N^2 relaciones Kerberos a Kerberos, como se describió anteriormente. En la versión 5 se requieren menos relaciones.

Además de estas limitaciones de entorno, hay **deficiencias técnicas** en el protocolo de la versión 4. La mayoría de estas deficiencias se documentaron en [BELL90], y la versión 5 intenta superarlas. Dichas deficiencias son las siguientes:

⁶ La siguiente discusión sigue la presentación en [KOHL94].

- 1. Cifrado doble:** obsérvese en la Tabla 4.1 [mensajes (2) y (4)] que los tickets proporcionados a los clientes se cifran dos veces, una vez con la clave secreta del servidor meta y luego otra vez con la clave secreta conocida por el cliente. El segundo cifrado no es necesario y es inútil computacionalmente.
- 2. Cifrado PCBC:** el cifrado en la versión 4 hace uso de un modo no estándar del DES conocido como PCBC (*Propagating Cipher Block Chaining*)⁷. Se ha demostrado que este modo es vulnerable a un ataque que implica el intercambio de bloques de texto cifrado [KOHL89]. El PCBC fue diseñado para la comprobación de la integridad como parte de la operación de cifrado. La versión 5 proporciona mecanismos explícitos de integridad, permitiendo el uso del modo estándar CBC para el cifrado.
- 3. Claves de sesión:** cada ticket incluye una clave de sesión que usa un cliente para cifrar el autenticador enviado al servicio asociado con ese ticket. Además, la clave de sesión pueden usarla posteriormente el cliente y el servidor para proteger los mensajes que pasen durante esa sesión. Sin embargo, debido a que el mismo ticket podría usarse repetidamente para obtener servicio de un servidor particular, existe el riesgo de que un oponente reenvíe mensajes de una sesión antigua al cliente o al servidor. En la versión 5 es posible que un cliente y un servidor negocien una clave de subsesión, que ha de usarse sólo para esa conexión. Un nuevo acceso por parte del cliente daría como resultado el uso de una nueva clave de subsesión.
- 4. Ataques de contraseña:** las dos versiones son vulnerables al ataque de contraseña. El mensaje que envía el AS al cliente incluye material cifrado con una clave basada en la contraseña del cliente⁸. Un oponente puede capturar este mensaje e intentar descifrarlo probando varias contraseñas. Si el resultado de un descifrado de prueba es adecuado, entonces el oponente ha descubierto la contraseña del cliente y puede usarla posteriormente para obtener credenciales de autentificación de Kerberos. Este es el mismo tipo de ataque de contraseña descrito en el Capítulo 9, al que se aplica el mismo tipo de contramedidas. La versión 5 ofrece un mecanismo conocido como *preatentificación*, que debería dificultar los ataques de contraseña, aunque no los evita.

EL DIÁLOGO DE AUTENTIFICACIÓN DE LA VERSIÓN 5

La Tabla 4.3 resume el diálogo básico de la versión 5. Se explica mejor por comparación con la versión 4 (Tabla 4.1).

En primer lugar, consideremos el **intercambio de servicio de autentificación**. El mensaje (1) es la solicitud de un TGT realizada por un cliente. Como antes, incluye el ID del usuario y el TGS. Se añaden los siguientes elementos:

- **Dominio:** indica el dominio del usuario.
- **Opciones:** se usan para solicitar que se fijen determinados indicadores en el ticket de regreso.

⁷ Descrito en el Apéndice 4A.

⁸ El Apéndice 4A describe la correspondencia entre las contraseñas y las claves de cifrado.

Tabla 4.3 Resumen de los intercambios de mensajes de la versión 5 de Kerberos

(a) Intercambio de servicio de autenticación: para obtener el TGT	
(1) C → AS:	$Opciones \parallel ID_C \parallel Dominio_c \parallel ID_{tgs} \parallel Tiempos \parallel Nonce_1$
(2) AS → C:	$Dominio_c \parallel ID_C \parallel Ticket_{tgs} \parallel E_{Kc}[K_{c,tgs}] \parallel Tiempos \parallel Nonce_1 \parallel Dominio_{tgs}$ $\parallel ID_{tgs}$
$Ticket_{tgs} = E_{K_{tgs}}[Indicadores \parallel K_{c,tgs} \parallel Dominio_c \parallel ID_C \parallel AD_C \parallel Tiempos]$	
(b) Intercambio de TGS: para obtener un ticket que concede un servicio	
(3) C → TGS:	$Opciones \parallel ID_V \parallel Tiempos \parallel Nonce_2 \parallel Ticket_{tgs} \parallel Autentificador_c$
(4) TGS → C:	$Dominio_c \parallel ID_C \parallel Ticket_v \parallel E_{K_{c,tgs}}[K_{c,v}] \parallel Tiempos \parallel Nonce_2 \parallel Dominio_v$ $\parallel ID_V$
$Ticket_{tgs} = E_{K_{tgs}}[Indicadores \parallel K_{c,tgs} \parallel Dominio_c \parallel ID_C \parallel AD_C \parallel Tiempos]$	
$Ticket_v = E_{Kv}[Indicadores \parallel K_{c,v} \parallel Dominio_c \parallel ID_C \parallel AD_C \parallel Tiempos]$	
$Autentificador_c = E_{K_{c,tgs}}[ID_C \parallel Dominio_c \parallel TS_1]$	
(c) Intercambio de autenticación cliente/servidor: para obtener servicio	
(5) C → V:	$Opciones \parallel Ticket_v \parallel Autentificador_c$
(6) V → C:	$E_{K_{c,v}}[TS_2 \parallel Subclave \parallel Seq\#]$
$Ticket_v = E_{Kv}[Indicadores \parallel K_{c,v} \parallel Dominio_c \parallel ID_C \parallel AD_C \parallel Tiempos]$	
$Autentificador_c = E_{K_{c,v}}[ID_C \parallel Dominio_c \parallel TS_2 \parallel Subclave \parallel Seq\#]$	

- **Tiempos:** los usa el cliente para solicitar los siguientes valores de tiempo para el ticket:
 - *from*: la fecha de comienzo deseada para el ticket solicitado.
 - *till*: la fecha de expiración deseada para el ticket solicitado.
 - *rtime*: la nueva fecha de expiración solicitada.
- **Nonce** un valor aleatorio que se debe repetir en el mensaje (2) para garantizar que la respuesta es reciente y no ha sido capturada y reenviada por ningún oponente

El mensaje (2) devuelve un TGT, información de identificación del cliente y un bloque cifrado por medio de la clave de cifrado basada en la contraseña del usuario. Este bloque incluye la clave de sesión que ha de usarse entre el cliente y el TGS, los tiempos especificados en el mensaje (1), el *nonce* del mensaje (1) e información de identificación del TGS. El ticket incluye la clave de sesión, información de identificación del cliente, los valores de tiempo requeridos y los indicadores que reflejan el estado de este ticket y las opciones solicitadas. Aunque estos indicadores añaden mayor funcionalidad a la versión 5, por el momento postergamos la discusión sobre estos indicadores y nos concentraremos en la estructura general del protocolo de la versión 5.

Comparemos ahora el **intercambio de TGS** para las versiones 4 y 5. Se observa que el mensaje (3) para las dos versiones incluye un autentificador, un ticket y el nombre del servicio requerido. Además, la versión 5 incluye los tiempos y las opciones solicitados para el ticket y un *nonce*, todos con funciones similares a las del mensaje (1). El autentificador es el mismo que el que se usa en la versión 4.

El mensaje (4) tiene la misma estructura que el mensaje (2), devolviendo un ticket e información que necesita el cliente. Dicha información se ha cifrado con la clave de sesión que ahora comparten el cliente y el TGS.

Por último, en la versión 5 aparecen nuevas características para el **intercambio de autentificación cliente/servidor**. En el mensaje (5), el cliente puede solicitar de forma opcional que se requiera autentificación mutua. El autentificador incluye los nuevos campos que se exponen a continuación:

- **Subclave:** una clave de cifrado elegida por el cliente que se usará para proteger esta sesión concreta de la aplicación. Si se omite este campo, se usa la clave de sesión ($K_{c,v}$) suministrada en el ticket del mensaje (4).
- **Número de secuencia:** un campo opcional que especifica el número inicial de secuencia que va a usar el servidor para los mensajes enviados al cliente durante esta sesión. Los mensajes se pueden numerar secuencialmente para detectar ataques de repetición.

Si se requiere autentificación mutua, el servidor responde con el mensaje (6). El mensaje incluye el sello de tiempo que contenía el autentificador. Obsérvese que en la versión 4, el sello de tiempo se incrementaba en uno. Esto no es necesario en la versión 5 porque la naturaleza del formato de los mensajes no permite que un oponente cree el mensaje (6) sin conocer las claves de cifrado apropiadas. En caso de estar presente, el campo de subclave invalida, si estuviera presente, al campo de subclave del mensaje (5). El campo opcional de número de secuencia especifica el número inicial de secuencia que va a usar el cliente.

Indicadores del ticket

El campo de indicadores incluido en los tickets en la versión 5 ofrece mayor funcionalidad, en comparación con la versión 4. La Tabla 4.4 resume los indicadores que pueden incluirse en un ticket.

El indicador INITIAL indica que este ticket fue emitido por el AS, no por el TGS. Cuando un cliente solicita al TGS un ticket que concede un servicio, presenta un TGT obtenido del AS. En la versión 4, éste era el único modo de obtener un ticket que concede un servicio. La versión 5 proporciona la capacidad adicional de que el cliente puede obtener dicho ticket directamente del AS. La utilidad de este hecho reside en que un servidor como, por ejemplo, un servidor de cambio de contraseña, puede querer saber que la contraseña del cliente se ha comprobado recientemente.

El indicador PRE-AUTHENT, si se ha utilizado, indica que cuando el AS recibió la solicitud inicial [mensaje (1)], autenticó al cliente antes de emitir un ticket. La forma exacta de esta preautentificación queda sin especificar. Un ejemplo de ello lo tenemos en que la implementación MIT de la versión 5 dispone de preautentificación cifrada de sello de tiempo, habilitada por defecto. Cuando un usuario quiere obtener un ticket, tiene

Tabla 4.4 Indicadores de la versión 5 de Kerberos

INITIAL	Este ticket se emitió usando el protocolo del AS, y no basado en un TGT.
PRE-AUTHENT	Durante la autentificación inicial, el cliente fue autenticado por el KDC antes de que se emitiera un ticket.
HW-AUTHENT	El protocolo empleado para la autentificación inicial requirió el uso del <i>hardware</i> que se esperaba que poseyera sólo el cliente en cuestión.
RENEWABLE	Dice al TGS que este ticket se puede usar para obtener un ticket nuevo que expira en una fecha posterior.
MAY-POSTDATE	Dice al TGS que se puede emitir un ticket con fecha posterior basado en este TGT.
POSTDATED	Indica que la fecha de este ticket ha sido pospuesta; el servidor final puede comprobar el campo <i>authtime</i> para ver cuándo se produjo la autentificación original.
INVALID	Este ticket no es válido y debe ser validado por el KDC antes de ser usado.
PROXiable	Le dice al TGS que se puede emitir un nuevo ticket que otorga servicios con una dirección de red diferente, basado en el ticket presentado.
PROXY	Indica que este ticket es un <i>proxy</i> .
FORWARDABLE	Le dice al TGS que se puede emitir un nuevo TGT con una dirección de red diferente, basado en este TGT.
FORWARDED	Indica que este ticket se ha reenviado o que se ha emitido basándose en la autentificación que implica un TGT reenviado.

que enviar al AS un bloque de preautentificación que contenga un valor aleatorio, un número de versión y un sello de tiempo, cifrado en la clave basada en la contraseña del usuario. El AS descifra el bloque y no entregará el TGT solicitado a menos que el sello de tiempo del bloque de preautentificación se encuentre dentro de los márgenes de tiempo permitidos (intervalo de tiempo para dar cuenta de fallos de reloj y retardos de red). Otra posibilidad es el uso de una tarjeta inteligente que genera contraseñas que cambian continuamente y que se incluyen en los mensajes preautentificados. Las contraseñas generadas por la tarjeta pueden estar basadas en la contraseña de un usuario pero pueden ser transformadas por la tarjeta para que, en efecto, se usen contraseñas arbitrarias, lo cual evita los ataques basados en contraseñas fáciles de adivinar. El uso de una tarjeta inteligente o de un dispositivo similar quedaría reflejado mediante el indicador HW-AUTHENT.

Cuando un ticket tiene un tiempo de vida largo, existe la posibilidad de que un oponente lo robe y lo use durante un período de tiempo considerable. Sin embargo, fijar un tiempo de vida corto para reducir la amenaza implica un alto coste en la adquisición de nuevos tickets. En el caso de un TGT, el cliente tendría que almacenar la clave secreta del usuario, lo cual es arriesgado, o pedirle repetidamente una contraseña. Un esquema intermedio consiste en el uso de tickets renovables. Un ticket con el indicador RENEWABLE incluye dos tiempos de caducidad: uno para este ticket específico y otro que es

el último valor permitido para un tiempo de expiración. Un cliente puede renovar el ticket presentándolo al TGS con un nuevo tiempo de expiración solicitado. Si el nuevo tiempo se encuentra dentro del límite del último valor permitido, el TGS puede emitir un nuevo ticket con un nuevo tiempo de sesión y un tiempo de expiración específico posterior. La ventaja de este mecanismo es que el TGS podría denegar la renovación de un ticket si tiene constancia de que ha sido robado.

Un cliente puede solicitar que el AS proporcione un TGT con el indicador MAY-POSTDATE. Luego el cliente puede usar este ticket para solicitar al TGS un ticket con el indicador POSTDATED e INVALID. A continuación, el cliente puede someter a validación el ticket cuya fecha se ha pospuesto. Este esquema puede ser útil para ejecutar una tarea batch larga en un servidor que requiere un ticket de forma periódica. El cliente puede obtener de una sola vez un número de tickets para esta sesión, con valores de tiempo diferentes. Inicialmente son inválidos todos menos el primer ticket. Cuando la ejecución alcanza el momento en que se necesita un ticket nuevo, el cliente puede obtener la validación del ticket apropiado. Con este enfoque, el cliente no tiene que usar repetidamente su TGT para obtener un ticket que otorga servicios.

En la versión 5 un servidor puede actuar como *proxy* en nombre de un cliente, adoptando, efectivamente, las credenciales y los privilegios del cliente para solicitar un servicio de otro servidor. Si un cliente desea usar este mecanismo, solicita un TGT con el indicador PROXiable. Cuando este ticket se presenta al TGS, el TGS tiene permiso para emitir un ticket que otorga servicio con una dirección de red diferente; este último ticket tendrá su indicador PROXY. Una aplicación que reciba tal ticket podría aceptarla o exigir autentificación adicional para proporcionar un informe de auditoría.

El concepto *proxy* es un caso limitado del procedimiento más potente de reenvío. Si a un ticket se le añade el indicador FORWARDABLE, un TGS puede emitir al solicitante un TGT con una dirección de red diferente y con el indicador FORWARDED. Este ticket, luego, puede presentarse a un TGS remoto. Esta capacidad permite a un cliente acceder a un servidor de otro dominio sin que sea necesario que cada Kerberos mantenga una clave secreta con los servidores Kerberos en cada uno de los demás dominios. Por ejemplo, los dominios podrían estructurarse jerárquicamente. Entonces un cliente podría ascender por el diagrama de árbol hasta llegar a un nodo común y luego volver atrás para llegar a un dominio meta. Cada paso del recorrido implicaría el reenvío de un TGT al siguiente TGS en la ruta.

4.2 SERVICIO DE AUTENTIFICACIÓN DE X.509

El estándar X.509 de la recomendación ITU-T es parte de la serie de recomendaciones X.500 que definen un servicio de directorio. El directorio es, en efecto, un servidor o un grupo distribuido de servidores que mantiene una base de datos con información sobre los usuarios. La información incluye la correspondencia entre nombre de usuario y dirección de red, así como otros atributos e información acerca de los usuarios.

X.509 define un marco para la provisión de servicios de autentificación por parte del directorio X.500 a sus usuarios. El directorio puede servir como depósito de certificados de clave pública. Cada certificado contiene la clave pública de un usuario y está firmado con la clave privada de una autoridad de certificación confiable. Además, X.509 de-

fine protocolos alternativos de autenticación basados en el uso de certificados de clave pública.

El X.509 es un estándar importante debido a que la estructura del certificado y los protocolos de autenticación definidos en él se usan en una variedad de contextos. Por ejemplo, el formato del certificado X.509 se usa en S/MIME (Capítulo 5), seguridad IP (Capítulo 6) y SSL/TLS y SET (Capítulo 7).

El X.509 se publicó por primera vez en 1988. El estándar se revisó posteriormente para cubrir algunos de los aspectos de seguridad documentados en [IANS90] y [MITC90]; en 1993 se emitió una recomendación revisada. Una tercera versión se lanzó en 1995 y se revisó en 2000.

El X.509 se basa en el uso de criptografía de clave pública y firmas digitales. El estándar no indica el uso de un algoritmo específico pero recomienda el RSA. El esquema de firma digital requiere el uso de una función *hash*. Nuevamente, el estándar no indica un algoritmo *hash* específico. La recomendación de 1988 incluía la descripción de un algoritmo *hash* recomendado, que desde entonces ha demostrado ser inseguro y fue retirado de la recomendación de 1993.

CERTIFICADOS

La base del esquema de X.509 es el certificado de clave pública asociado a cada usuario. Estos certificados de usuario los crea alguna autoridad de certificación confiable (AC), y dicha autoridad o el usuario los colocan en el directorio. El servidor de directorio no es el responsable de la creación de claves públicas ni de la función de certificación; simplemente proporciona una ubicación de fácil acceso para que los usuarios obtengan certificados.

La Figura 4.3a muestra el formato general de un certificado, que incluye los siguientes elementos:

- **Versión:** distingue entre versiones sucesivas del formato de certificado; la versión por defecto es la 1. Si el identificador único del emisor del certificado o el identificador único del sujeto están presentes, el valor debe ser la versión 2. Si están presentes una o más extensiones, la versión debe ser la versión 3.
- **Número de serie:** un valor entero, único en la CA que emite el certificado, que está asociado sin ambigüedad a este certificado.
- **Identificador del algoritmo de la firma:** el algoritmo que se emplea para firmar el certificado junto con parámetros asociados. Debido a que esta información se repite en el campo Firma al final del certificado, este campo tiene muy poca o ninguna utilidad.
- **Nombre del emisor del certificado:** el nombre X.509 de la CA que creó y firmó este certificado.
- **Período de validez:** consiste en dos fechas: la fecha inicial y la fecha final de validez del certificado.
- **Nombre del sujeto:** el nombre del usuario a quien se remite este certificado. Es decir, este certificado asegura la clave pública del sujeto que tiene la clave privada correspondiente.

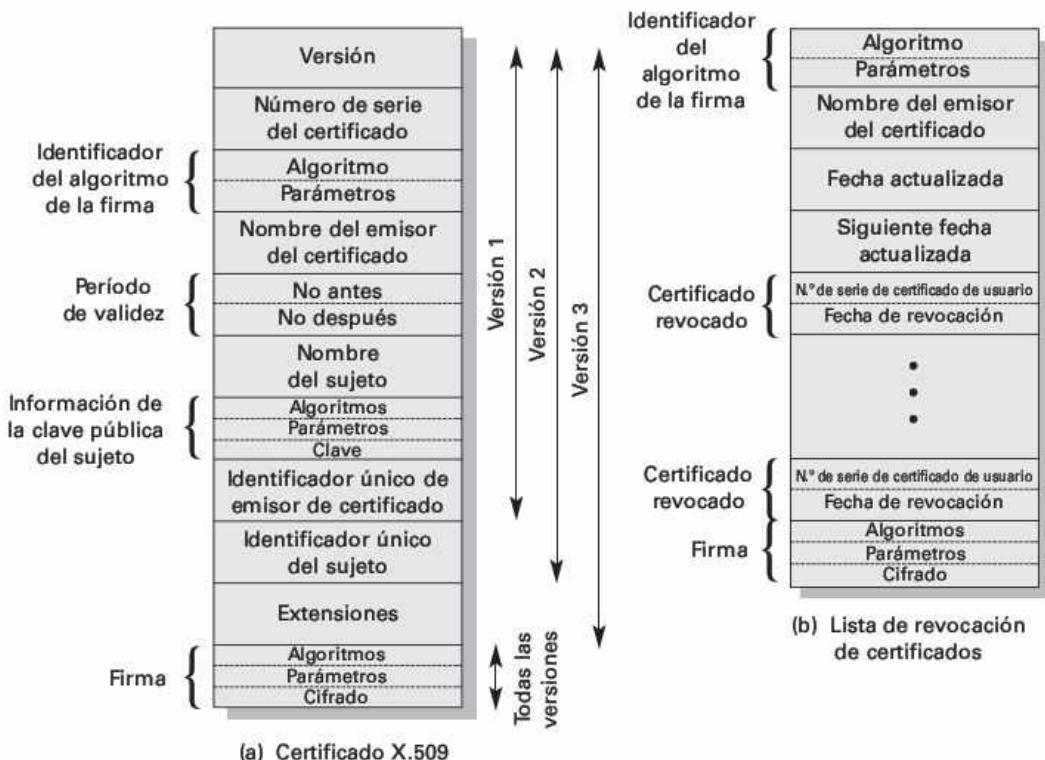


Figura 4.3 Formatos X.509

- **Información de la clave pública del sujeto:** la clave pública del sujeto y un identificador del algoritmo para el cual se usa esta clave, junto con cualquier parámetro asociado.
- **Identificador único del emisor del certificado:** un campo opcional de ristras de bits que se usa para identificar únicamente a la CA que emite el certificado en el caso de que el nombre del X.500 haya sido reutilizado para distintas entidades.
- **Identificador único de sujeto:** un campo opcional de ristras de bits que se usa para identificar únicamente al sujeto en el caso de que el nombre del X.500 haya sido reutilizado para distintas entidades.
- **Extensiones:** un grupo de uno o más campos de extensión. Las extensiones se añadieron en la versión 3 y se tratan más adelante en esta sección.
- **Firma:** cubre todos los demás campos del certificado; contiene el código *hash* de los otros campos, cifrado con la clave privada de la CA. Este campo incluye el identificador del algoritmo de la firma.

Los campos de identificador único se añadieron en la versión 2 para tratar la posible reutilización de los nombres de sujeto y/o emisor de certificado durante más tiempo, pero rara vez se usan.

El estándar emplea la siguiente notación para definir un certificado:

$$\text{CA} <> \text{A} = \text{CA} \{ V, \text{SN}, \text{AI}, \text{CA}, T_A, A, \text{Ap} \}$$

Donde

$Y<<X>>$ = el certificado del usuario X emitido por la autoridad de certificación Y

$Y\{I\}$ = la firma de I por parte de Y. Está formado por I con un código *hash* cifrado añadido

La CA firma el certificado con su clave privada. Si la clave pública correspondiente es conocida por un usuario, entonces ese usuario puede verificar que un certificado firmado por la CA es válido. Este es el enfoque más común de firma digital que se muestra en la Figura 3.2c.

Obtención de un certificado de usuario

Los certificados de usuario generados por una CA tienen las siguientes características:

- Cualquier usuario con acceso a la clave pública de la CA puede verificar la clave pública del usuario que fue certificada.
- Sólo la CA puede modificar el certificado sin que esto se detecte.

Debido a que los certificados no son falsificables, pueden colocarse en un directorio sin que sea necesario tomar medidas especiales de protección.

Si todos los usuarios se suscriben a la misma autoridad, entonces hay una confianza común en esa CA. Todos los certificados de usuario se pueden situar en el directorio para el acceso de todos los usuarios. Además, un usuario puede transmitir su certificado directamente a otros usuarios. En cualquier caso, una vez que B está en posesión del certificado de A, B confía en que los mensajes que cifra con la clave pública de A están a salvo de escuchas y que los mensajes firmados con la clave privada de A no son falsificables.

Si hay una comunidad amplia de usuarios, puede no ser práctico que todos los usuarios se suscriban a la misma CA. Como es la autoridad la que firma los certificados, cada usuario participante debe tener una copia de la clave pública de la autoridad para verificar las firmas. Esta clave pública debe suministrarse a cada usuario de una forma totalmente segura (con respecto a la integridad y la autenticidad) para que el usuario confie en los certificados asociados. Así, con muchos usuarios, sería más práctico que hubiese un número de autoridades de certificación, y que cada una de ellas proporcionara de manera segura su clave pública a una parte de los usuarios.

Ahora supongamos que A ha obtenido un certificado de la autoridad de certificación X_1 y B ha obtenido un certificado de la autoridad X_2 . Si A no conoce de forma segura la clave pública de X_2 , entonces el certificado de B, emitido por X_2 , es inútil para A. A puede leer el certificado de B, pero A no puede verificar la firma. Sin embargo, si las dos autoridades han intercambiado de forma segura sus propias claves públicas, el siguiente procedimiento permitirá que A obtenga la clave pública de B:

1. A obtiene del directorio el certificado de X_2 firmado por X_1 . Como A conoce de forma segura la clave pública de X_1 , A puede obtener la clave pública de X_2 a partir de su certificado y verificarlo por medio de la firma de X_1 en el certificado.

2. Luego, A vuelve al directorio y obtiene el certificado de B firmado por X_2 . Como A tiene ahora una copia confiable de la clave pública de X_2 , A puede verificar la firma y obtener de forma segura la clave pública de B.

A ha empleado una cadena de certificados para obtener la clave pública de B. En la notación de X.509, esta cadena se expresa de la siguiente manera:

$$X_1 << X_2 >> X_2 << B >>$$

De la misma forma, B puede obtener la clave pública de A con la cadena inversa:

$$X_2 << X_1 >> X_1 << A >>$$

Este esquema no tiene que limitarse a una cadena de dos certificados. Se puede seguir una ruta de longitud arbitraria de las autoridades de certificación para producir una cadena. Una cadena con N elementos se expresaría así:

$$X_1 << X_2 >> X_2 << X_3 >> \dots X_N << B >>$$

En este caso, cada pareja de CA en la cadena (X_i, X_{i+1}) debe haber creado certificados para intercambiarlos entre sí.

Todos estos certificados de las autoridades de certificación por parte de las mismas necesitan aparecer en el directorio, y el usuario necesita conocer cómo están enlazadas para seguir una ruta hacia el certificado de clave pública de otro usuario. El estándar X.509 sugiere que las CA se organicen de forma jerárquica para facilitar la navegación.

La Figura 4, extraída de X.509, es un ejemplo de jerarquía. Los círculos conectados indican la relación jerárquica entre las CA; las cajas asociadas indican los certificados mantenidos en el directorio para cada entrada de CA. La entrada del directorio para cada CA incluye dos tipos de certificados:

- **Certificados forward** certificados de X generados por otras CA.
- **Certificados reverse** certificados generados por X que son los certificados de otras CA.

En este ejemplo, el usuario A puede adquirir los siguientes certificados del directorio para establecer una ruta de certificación hacia B:

$$X << W >> W << V >> V << Y >> Y << Z >> Z << B >>$$

Cuando A ha obtenido estos certificados se puede retroceder en la ruta de certificación secuencialmente para recuperar una copia confiable de la clave pública de B. Usando esta clave pública, A puede enviar mensajes cifrados a B. Si A desea recibir mensajes cifrados de B, o firmar mensajes enviados a B, entonces B necesitará la clave pública de A, que se puede obtener de la siguiente ruta de certificación:

$$Z << Y >> Y << V >> V << W >> W << X >> X << A >>$$

B puede obtener este grupo de certificados del directorio, o A puede proporcionarlos como parte de su mensaje inicial a B.

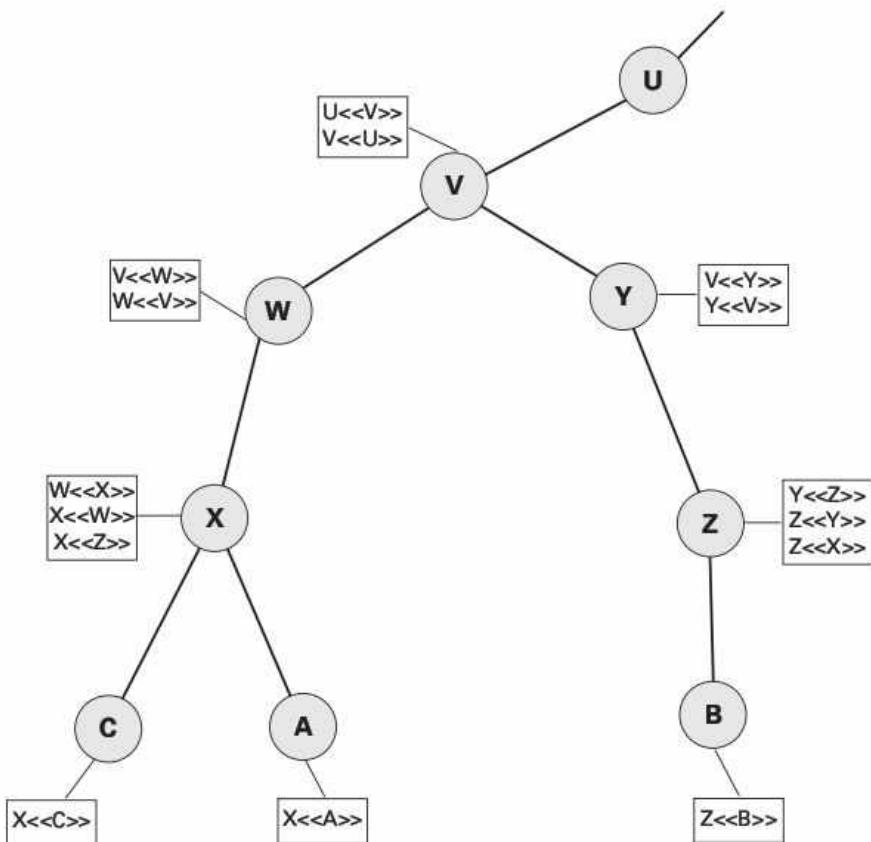


Figura 4.4 Jerarquía de X.509: un ejemplo hipotético

Revocación de certificados

Como reflejaba la Figura 4.3, cada certificado incluye un período de validez, parecido al de una tarjeta de crédito. Normalmente, un certificado nuevo se emite justo antes de la expiración del anterior. Además, a veces puede ser preferible revocar un certificado antes de que expire, por una de las siguientes razones:

1. Se sospecha que la clave privada del usuario está comprometida.
2. El usuario ya no está certificado por esa CA.
3. Se sospecha que el certificado de la CA está comprometido.

Cada CA debe mantener una lista de todos los certificados emitidos por ella que han sido revocados pero que no han expirado, incluidos tanto los emitidos a los usuarios como a otras autoridades. Estas listas también deberían añadirse al directorio.

Cada lista de revocación de certificados agregada al directorio es firmada por el emisor de certificado e incluye (Figura 4.3b) el nombre del emisor, la fecha de creación de la lista, la fecha en que se va a emitir la nueva lista y una entrada para cada certificado revocado. Cada entrada está formada por el número de serie de un certificado y la fecha

de revocación de ese certificado. Como los números de serie son únicos en una CA, el número de serie es suficiente para identificar el certificado.

Cuando un usuario recibe un certificado en un mensaje, el usuario debe determinar si el certificado ha sido revocado. El usuario podría comprobar el directorio cada vez que se reciba un certificado. Para evitar los retrasos (y posibles costes) asociados a las búsquedas en directorios, el usuario puede mantener una caché local de certificados y listas de certificados revocados.

PROCEDIMIENTOS DE AUTENTIFICACIÓN

El X.509 también incluye tres procedimientos alternativos de autentificación diseñados para su uso en una gran variedad de aplicaciones. Todos estos procedimientos hacen uso de firmas de clave pública. Se supone que cada una de las dos partes conoce la clave pública de la otra, obteniendo del directorio el certificado de la otra parte o porque el certificado está incluido en el mensaje inicial de cada parte.

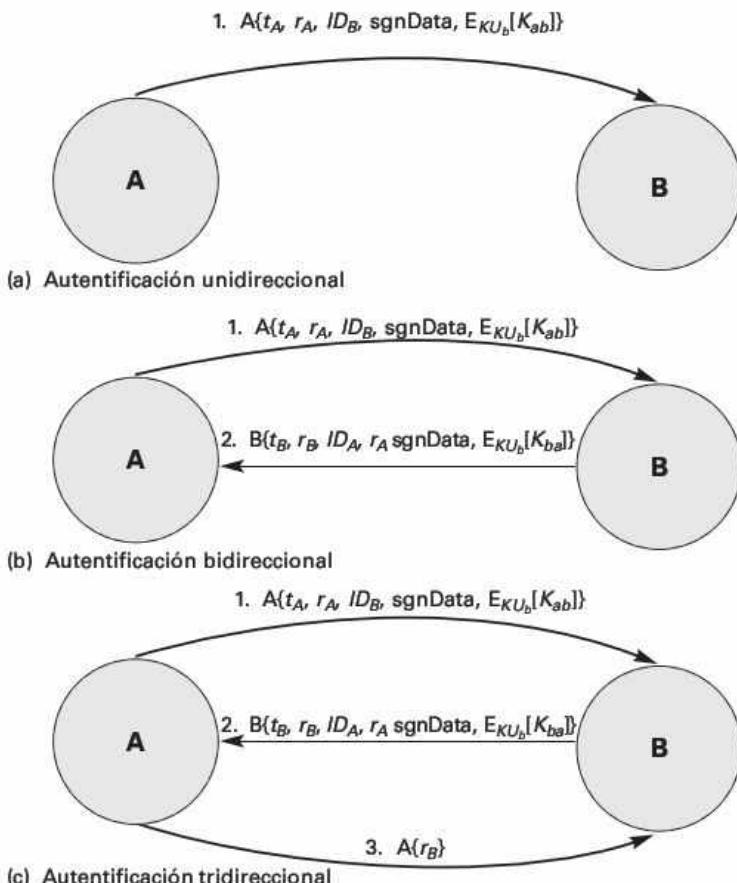


Figura 4.5 Procedimientos robustos de autentificación de X.509

La Figura 4.5 ilustra los tres procedimientos.

Autentificación unidireccional

La autentificación unidireccional implica una sola transferencia de información desde un usuario (A) a otro (B), y establece:

1. La identidad de A y que el mensaje fue generado por A.
2. Que el mensaje iba dirigido a B.
3. La integridad y la originalidad del mensaje (no se ha enviado varias veces).

Obsérvese que en este proceso sólo se verifica la identidad de la entidad iniciadora, no la de la entidad que responde.

Como mínimo, el mensaje incluye un sello de tiempo t_A , un *nonce* r_A , y la identidad de B, y está firmado con la clave privada de A. El sello de tiempo consiste en una fecha opcional de generación y una fecha de expiración. Esto evita retrasos en el envío de mensajes. El *nonce* puede usarse para detectar ataques de repetición. El valor del *nonce* debe ser único en el tiempo de expiración del mensaje. Así, B puede almacenar el *nonce* hasta que expire y rechazar cualquier mensaje nuevo con el mismo *nonce*.

Para la autentificación, el mensaje se usa simplemente para presentar credenciales a B. El mensaje también puede incluir información que haya que transmitir. Esta información, *sgnData*, se incluye en el ámbito de esta firma, garantizando su autenticidad e integridad. El mensaje puede usarse también para transportar una clave de sesión a B, cifrada con la clave pública de B.

Autentificación bidireccional

Además de los tres elementos mencionados, la autentificación bidireccional establece los siguientes elementos:

4. La identidad de B y que el mensaje de respuesta fue generado por B.
5. Que el mensaje se creó para A.
6. La integridad y la originalidad de la respuesta.

Así, la autentificación bidireccional permite que cada una de las dos partes involucradas en la comunicación verifiquen la identidad de la otra.

El mensaje de respuesta incluye un *nonce* de A para validar la respuesta. También incluye un sello de tiempo y un *nonce* generado por B. Como antes, el mensaje puede incluir información firmada adicional y una clave de sesión cifrada con la clave pública de A.

Autentificación tridireccional

En la autentificación tridireccional se incluye un mensaje final de A a B, que contiene una copia firmada del *nonce* r_B . El propósito de este diseño es que los sellos de tiempo no tengan que ser comprobados: como ambos *nonces* son devueltos por el otro lado, cada lado puede comprobar el *nonce* devuelto para detectar ataques de repetición. Este enfoque es necesario cuando no se dispone de relojes sincronizados.

LA VERSIÓN 3 DE X.509

Según la experiencia actual en diseño e implementación, el formato de la versión 2 de X.509 no transmite toda la información necesaria. En [FORD95] se presentan los siguientes requisitos que no satisface la versión 2:

1. El campo Sujeto no es adecuado para transmitir la identidad del propietario de una clave a un usuario de clave pública. Los nombres de X.509 podrían ser relativamente cortos y carecer de datos obvios de identificación que pueda necesitar el usuario.
2. El campo Sujeto también es inadecuado para muchas aplicaciones, ya que normalmente reconocen entidades mediante una dirección de correo electrónico, una URL u otro tipo de identificación relacionada con Internet.
3. Existe la necesidad de indicar información sobre la política de seguridad. Esto permite que una aplicación o función de seguridad, como IPSec, relacione un certificado X.509 con una política determinada.
4. Es necesario limitar el daño que pueda resultar de una CA defectuosa o malintencionada, imponiendo restricciones a la aplicabilidad de un certificado particular.
5. Es importante poder identificar por separado diferentes claves usadas por el mismo propietario en momentos diferentes. Esta característica beneficia la gestión del ciclo de vida de la clave, particularmente la habilidad de actualizar parejas de claves para usuarios y autoridades de certificación de forma regular y en circunstancias excepcionales.

En vez de continuar añadiendo campos a un formato fijo, los desarrolladores de estándares pensaron que era necesario un enfoque más flexible. Así, la versión 3 incluye una serie de extensiones opcionales que se pueden añadir al formato de la versión 2. Cada extensión está formada por un identificador de extensión, un indicador de riesgo y un valor de extensión. El indicador de riesgo señala si una extensión se puede ignorar sin peligro. Si el indicador tiene un valor de TRUE y una implementación no reconoce la extensión, debe considerar el certificado como no válido.

Las extensiones del certificado se dividen en tres categorías: información sobre claves y política, atributos de sujeto y emisor de certificado y limitaciones en la ruta de certificación.

INFORMACIÓN SOBRE CLAVES Y POLÍTICA

Estas extensiones aportan información adicional sobre las claves del sujeto y del emisor, así como los indicadores de política de certificados. Una política de certificados es una serie de normas que indican la aplicabilidad de un certificado a una comunidad particular y/o clase de aplicación con requisitos comunes de seguridad. Por ejemplo, una política podría ser aplicable a la autenticación de transacciones de intercambio de datos electrónicos (EDI: *Electronic Data Interchange*) para el comercio de productos en una escala dada de precios.

Incluye lo siguiente:

- **Identificador de clave de autoridad:** identifica la clave pública que ha de usarse para verificar la firma de ese certificado o la lista de revocación de certificados. Permite diferenciar distintas claves de la misma CA. Uno de los usos de este campo se halla en la actualización de parejas de claves de CA.

- **Identificador de clave de sujeto:** identifica la clave pública que se está certificando. Es útil para la actualización de parejas de clave de sujeto. También un sujeto puede tener varias parejas de claves y, por lo tanto, diferentes certificados para diferentes propósitos (por ejemplo, firma digital y negociación de clave de cifrado).
- **Uso de clave:** indica la imposición de una restricción a los propósitos para los que se pueda usar la clave pública certificada y a la política que la regula. Puede indicar uno o más de los siguientes: firma digital, no repudio, cifrado de clave, cifrado de datos, acuerdo de clave, verificación de firma de CA en los certificados, verificación de firma de CA en la lista de revocación de certificados.
- **Período de uso de la clave privada:** indica el período de uso de la clave privada correspondiente a la clave pública. Normalmente, la clave privada se usa durante un período diferente de la validez de la clave pública. Por ejemplo, con claves de firma digital, el período de uso para la clave privada de firma es generalmente menor que para la clave pública de verificación.
- **Políticas de certificados:** los certificados pueden usarse en entornos donde se apliquen diversas políticas. Esta extensión presenta las políticas que el certificado admite, junto con información opcional del calificador.
- **Correspondencias de políticas:** se usa sólo en certificados para CA emitidos por otras CA. Permiten que una CA que emite indique que una o más políticas de ese emisor de certificado se puedan considerar equivalentes a otra política usada en el dominio de CA del sujeto.

Sujeto de certificado y atributos de emisor de certificado

Estas extensiones permiten nombres alternativos en formatos alternativos para un sujeto de certificado o emisor de certificado, y pueden aportar información adicional sobre éste, para convencer al usuario de un certificado de que el sujeto del certificado es una persona o entidad particular. Por ejemplo, se puede necesitar información como la dirección postal, el puesto en una empresa o una foto.

Los campos de extensión en esta área incluyen:

- **Nombre alternativo de sujeto:** contiene uno o más nombres alternativos, usando cualquiera de una variedad de formas. Este campo es importante para ciertas aplicaciones, como correo electrónico, EDI e IPSec, que pueden emplear sus propias formas de nombre.
- **Nombre alternativo de emisor:** contiene uno o más nombres alternativos, usando cualquiera de una variedad de formas.
- **Atributos de directorio de sujeto:** aporta cualquier valor deseado de atributo del directorio X.500 para el sujeto de este certificado.

Limitaciones en la ruta de certificación

Estas extensiones permiten incluir especificaciones sobre limitaciones en los certificados emitidos por autoridades de certificación a otras. Las limitaciones pueden restringir los tipos de certificados que puede emitir la CA del sujeto o que puedan producirse posteriormente en una cadena de certificación.

Los campos de extensiones que se incluyen son:

- **Limitaciones básicas:** indica si el sujeto puede actuar como una CA. De ser así, se podría especificar una limitación en la longitud de la ruta de certificación.
- **Limitaciones de nombre:** indica un espacio para nombre en el que deben colocarse todos los nombres del sujeto en certificados siguientes.
- **Limitaciones de política:** especifica las limitaciones que pueden requerir identificación explícita de política de certificado o impedir la correspondencia de políticas para el resto de la ruta de certificación.

4.3 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

[BRYA88] ayuda a entender los conceptos de Kerberos de forma sencilla. Uno de los mejores tratamientos de Kerberos se encuentra en [KOHL94]. [TUNG99] describe Kerberos desde el punto de vista del usuario.

- BRYA88** Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Project Athena document, February 1988. Disponible en <http://web.mit.edu/kerberos/www/dialogue.html>.
- KOHL94** Kohl, J.; Neuman, B.; and Ts'o, T. «The Evolution of the Kerberos Authentication Service.» In Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Disponible en <http://web.mit.edu/kerberos/www/papers.html>
- TUNG99** Tung, B. *Kerberos: A Network Authentication System*. Reading, MA: Addison-Wesley, 1999.

Sitios web recomendados:

- **MIT Kerberos Site:** información sobre Kerberos que incluye FAQ, artículos, documentos y enlaces a sitios de productos comerciales.
- **USC/ISI Kerberos Page:** otra buena fuente con material sobre Kerberos.
- **Public-Key Infrastructure Working Group:** grupo de la IETF que desarrolla estándares basados en X.509v3.
- **Verisign:** vendedor líder de productos relacionados con X.509; en este sitio se encuentran documentos y material valioso.

4.4 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Autentificación	Kerberos	subclave
certificado de clave pública	modo PCBC	TGS (<i>Ticket-Granting Server</i>)
certificado X.509	<i>nonce</i>	ticket
dominio	número de secuencia	tiempo de vida
Dominio de Kerberos	Servidor de autentificación	

PREGUNTAS DE REPASO

- 41.** Kerberos se diseñó para resolver un problema ¿cuál?
- 42.** ¿Cuáles son las tres amenazas asociadas a la autentificación de usuario en una red o en Internet?
- 43.** Enumera los tres enfoques que aseguran la autentificación de usuario en un entorno distribuido.
- 44.** ¿Qué cuatro requisitos se definieron para Kerberos?
- 45.** ¿Qué entidades constituyen un entorno de servicio completo de Kerberos?
- 46.** En el contexto de Kerberos, ¿qué es un dominio?
- 47.** ¿Cuáles son las diferencias principales entre la versión 4 y la versión 5 de Kerberos?
- 48.** ¿Cuál es la finalidad del estándar X.509?
- 49.** ¿Qué es una cadena de certificados?
- 50.** ¿Cómo se revoca un certificado X.509?

PROBLEMAS

- 41.** Demuestra que un error aleatorio en un bloque de texto cifrado se propaga a todos los bloques siguientes en el modo PCBC (Figura 4.7).
- 42.** Imagina que, en el modo PCBC, los bloques C_i y C_{i+1} se intercambian durante la transmisión. Demuestra que esto afecta sólo a los bloques descifrados P_i y P_{i+1} pero no a los bloques siguientes.
- 43.** El procedimiento original de autentificación tridireccional para X.509 que se ilustra en la Figura 4.5c tiene un fallo en la seguridad. La base del protocolo es la siguiente:

$$\begin{aligned} A \rightarrow B: & A \{t_A, r_A, ID_B\} \\ B \rightarrow A: & B \{t_B, r_B, ID_A, r_A\} \\ A \rightarrow B: & A \{r_B\} \end{aligned}$$

El texto de X.509 afirma que comprobar los sellos de tiempo t_A y t_B es opcional para la autentificación tridireccional. Pero considera el siguiente ejemplo: imagina que A y B han utilizado en ocasiones anteriores el protocolo mencionado, y un oponente C ha interceptado los tres mensajes anteriores. Además, imagina que los sellos de tiempo no se usan y están fijados a 0. Por último, imagina también que C quiere suplantar a A ante B . Inicialmente, C envía el primer mensaje capturado a B :

$$C \rightarrow B: A\{0, r_A, ID_B\}$$

B responde a C creyendo que se comunica con A :

$$B \rightarrow C: B\{0, r'_B, ID_A, r_A\}$$

Mientras tanto C hace que A inicie la autentificación con C . Como resultado, A envía a C lo siguiente:

$$A \rightarrow B: A\{0, r'_A, ID_C\}$$

C responde a *A* usando el mismo *nonce* que *B* le proporcionó.

$$C \rightarrow A: C\{0, r'_B, A, r'_A\}$$

A responde con

$$A \rightarrow C: A\{r'_B\}$$

Esto es exactamente lo que *C* necesita para convencer a *B* de que está hablando con *A*, por lo que *C* ahora reenvía el mensaje entrante a *B*.

$$C \rightarrow B: A\{r'_B\}$$

Así, *B* creerá que está hablando con *A* cuando realmente lo hace con *C*. Presenta una solución simple a este problema que no implique el uso de sellos de tiempo.

- 44** La versión del X.509 de 1988 enumera las propiedades que las claves RSA deben poseer para ser seguras, conociendo la dificultad que plantea factorizar números grandes. La discusión concluye con una limitación en el exponente público y el módulo *n*:

Debe asegurarse que $e > \log_2(n)$ para prevenir un ataque tomando la *e*-ésima raíz mod *n* para revelar un texto claro.

Aunque la limitación es correcta, la razón por la que se pide es incorrecta. ¿Cuál es el error en la razón dada y cuál es la razón correcta?

APÉNDICE 4A TÉCNICAS DE CIFRADO KERBEROS

Kerberos incluye una librería de cifrado que permite diferentes operaciones relacionadas con el cifrado.

TRANSFORMACIÓN DE CONTRASEÑA A CLAVE

En Kerberos, las contraseñas se limitan al uso de caracteres que pueden representarse en un formato ASCII de 7 bits. Esta contraseña, de longitud arbitraria, se convierte en una clave de cifrado que se almacena en la base de datos de Kerberos. La Figura 4.6 ilustra el procedimiento.

En primer lugar, la ristra de caracteres, *s*, se empaqueta en una ristra de bits, *b*, de forma que el primer carácter se almacena en los primeros siete bits, el segundo carácter en los segundos siete bits, y así sucesivamente. Se puede expresar como sigue:

$$b[0] = \text{bit } 0 \text{ de } s[0]$$

$$\begin{aligned} b[6] &= \text{bit } 6 \text{ de } s[0] \\ b[7] &= \text{bit } 0 \text{ de } s[1] \end{aligned}$$

$$b[7i + m] = \text{bit } m \text{ de } s[i] \quad 0 \leq m \leq 6$$

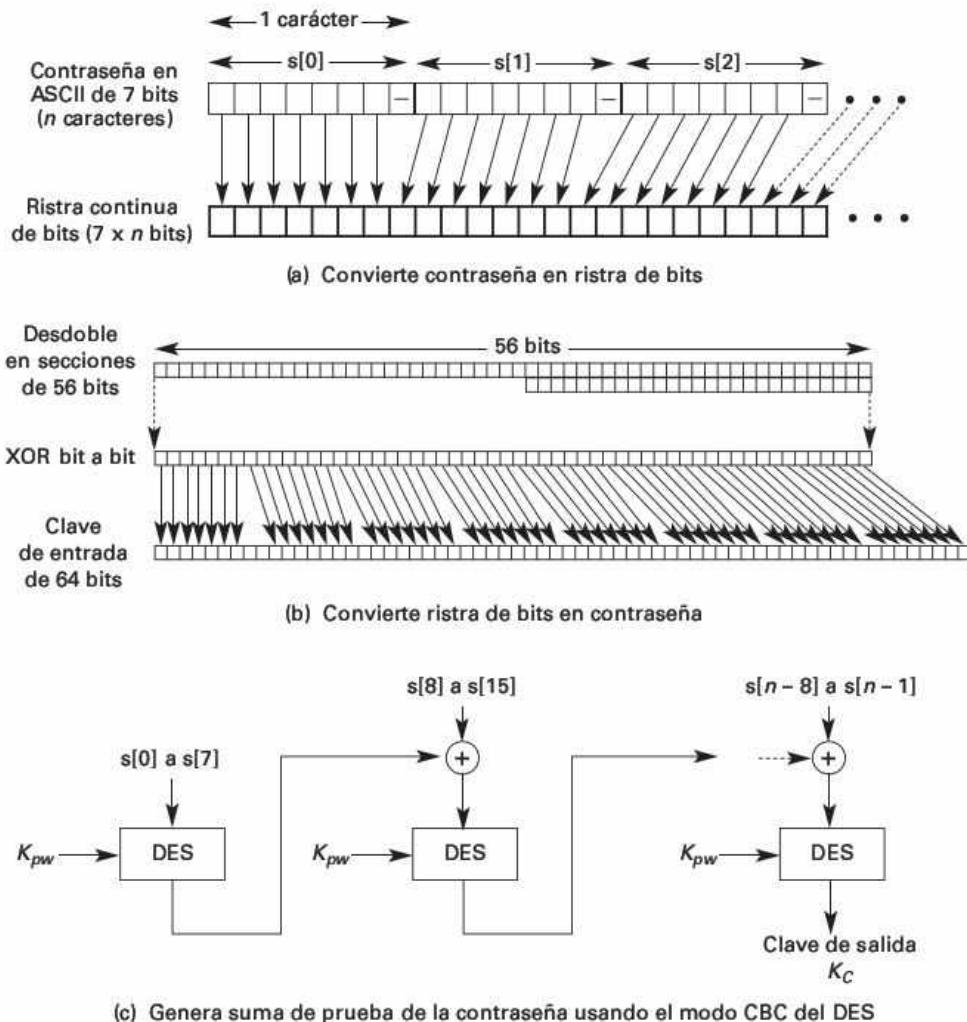


Figura 4.6 Generación de clave de cifrado a partir de contraseña

Luego, la ristra de bits se compacta a 56 bits alineando los bits desdoblando en secciones y realizando un XOR bit a bit. Por ejemplo, si la ristra de bits es de longitud 59, entonces

$$\begin{aligned} b[55] &= b[55] \oplus b[56] \\ b[54] &= b[54] \oplus b[57] \\ b[53] &= b[53] \oplus b[58] \end{aligned}$$

Esto crea una clave DES de 56 bits. Para ajustarse al formato de clave esperado de 64 bits, la ristra se trata como una secuencia de ocho bloques de siete bits y se mapea en ocho bloques de ocho bits para formar una clave de entrada K_{pw} .

Por último, la contraseña original se cifra usando el modo CBC del DES con la clave K_{pw} . El último bloque de 64 bits que se origina de este proceso, conocido como la suma de prueba del CBC, es la clave de salida asociada a esta contraseña.

El algoritmo completo se puede ver como una función *hash* que mapea una contraseña arbitraria con un código *hash* de 64 bits.

PROPAGACIÓN DEL MODO PCBC (CIPHER BLOCK CHAINING)

Recordemos del Capítulo 2 que, en el modo CBC del DES, la entrada al algoritmo DES en cada fase consiste en el XOR del bloque actual de texto claro y el bloque de texto cifrado anterior, usando la misma clave para cada bloque (Figura 3.12). La ventaja de este modo con respecto al modo ECB, en el que cada bloque de texto claro se cifra independientemente, es la siguiente: con el CBC, el mismo bloque de texto claro, si se repite, produce diferentes bloques de texto cifrado.

El CBC tiene la propiedad de que si se produce un error en la transmisión del bloque de texto cifrado C_1 , entonces este error se propaga a los bloques de texto claro recuperados P_1 y P_{I+1} .

La versión 4 de Kerberos usa una extensión del CBC, denominada modo PCBC [MEYE82]. Este modo tiene la propiedad de que un error en un bloque de texto cifrado se propaga a todos los bloques descifrados siguientes del mensaje, inutilizando cada bloque. Así, el cifrado y la integridad de los datos se combinan en una sola operación.

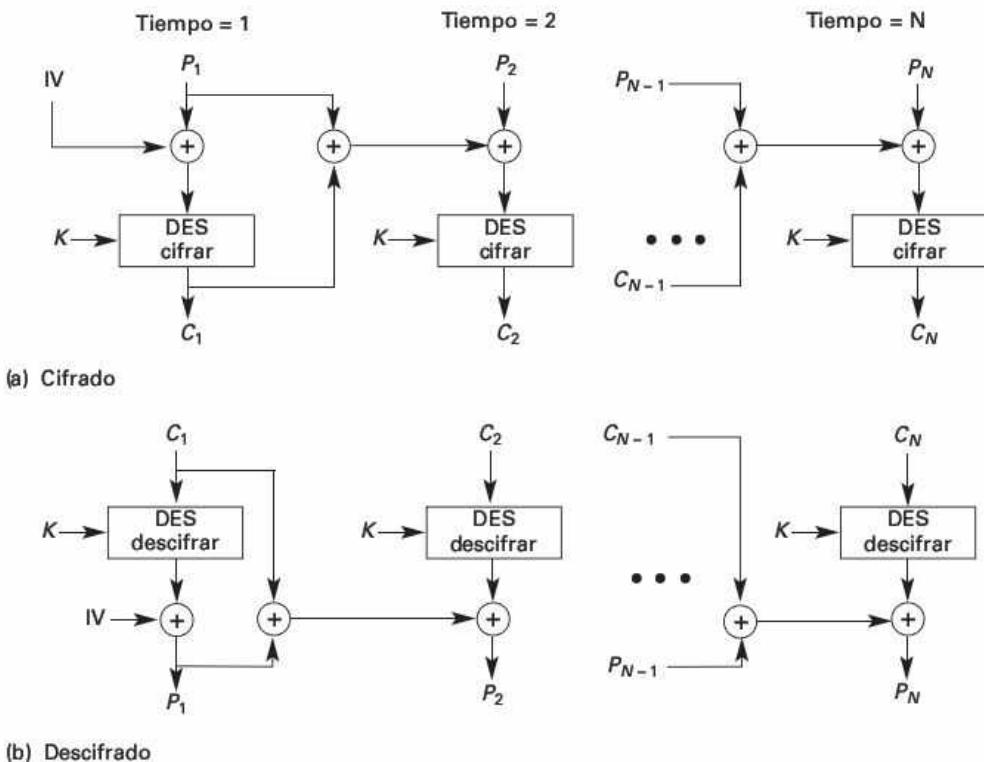


Figura 4.7 Modo PCBC

El PCBC se ilustra en la Figura 4.7. En este esquema, la entrada al algoritmo de cifrado es el XOR del bloque actual de texto claro, el bloque anterior de texto cifrado y el bloque anterior de texto claro:

$$C_n = E_K[C_{n-1} \oplus P_{n-1} \oplus P_n]$$

En el descifrado, cada bloque de texto cifrado pasa por el algoritmo de descifrado. Entonces a la salida se le aplica el XOR con el bloque anterior de texto cifrado y el bloque anterior de texto claro. Se puede demostrar de la siguiente forma que este esquema funciona:

$$\begin{aligned} D_K[C_n] &= D_K[E_K[C_{n-1} \oplus P_{n-1} \oplus P_n]] \\ &= C_{n-1} \oplus P_{n-1} \oplus P_n \\ C_{n-1} \oplus P_{n-1} \oplus D_K[C_n] &= P_n \end{aligned}$$

CAPÍTULO 5

Seguridad en el correo electrónico

5.1 Pretty Good Privacy

Notación
Descripción operativa
Claves criptográficas y ficheros de claves
Gestión de clave pública

5.2 S/MIME

RFC 822
MIME (*Multipurpose Internet Mail Extensions*)
Funcionalidad S/MIME
Mensajes S/MIME
Procesamiento de certificados S/MIME
Servicios de seguridad avanzada

5.3 Sitios web recomendados

5.4 Términos clave, preguntas de repaso y problemas

Términos clave
Preguntas de repaso
Problemas

Apéndice 5A Compresión de datos usando Zip

Algoritmo de compresión
Algoritmo de descompresión

Apéndice 5B Conversión RADIX 64

Apéndice 5C Generación de números aleatorios de PGP

Números aleatorios
Números pseudoaleatorios

A pesar de la negativa de VADM Poindexter y LtCol North a estar presentes, el acceso de la Cámara a otras fuentes de información cubrió gran parte de este vacío. El FBI aportó documentos extraídos de los archivos del Asesor de Seguridad Nacional y miembros importantes del personal de NSC, incluyendo los mensajes del sistema PROF entre VADM Poindexter y LtCol North. Los mensajes de PROF eran conversaciones por computador, escritos en el momento en que ocurrían los acontecimientos y, como creían sus autores, protegidos de ser revelados. En este sentido, proporcionaron un informe de los hechos actual y de primera mano.

**El informe de la Comisión Tower
al presidente Reagan sobre el caso «Irán-contra», 1987**

*Bless the man who made it,
And pray that he ain't dead
He could've made a million
If he'd sold it to the feds,
But he was hot for freedom;
He gave it out for free.
Now every common citizen's got PGP.¹*

De la canción «P.G.P.» de LESLIE FISH

En casi todos los entornos distribuidos, el correo electrónico es la aplicación de red más usada. También es la única aplicación distribuida que se utiliza en todas las arquitecturas y plataformas. Los usuarios esperan poder enviar correo, y de hecho lo hacen, a otros que se encuentren conectados directa o indirectamente a Internet, independientemente del sistema operativo del *host* y de la *suite* de comunicaciones.

A medida que aumenta nuestra dependencia del correo electrónico con cualquier finalidad imaginable, aumenta también la demanda de servicios de autenticación y confidencialidad. Los dos esquemas que se analizan en este Capítulo son PGP (*Pretty Good Privacy*) y S/MIME, que constituyen los dos enfoques de uso más extendido.

5.1 PRETTY GOOD PRIVACY

PGP es un fenómeno singular y el resultado del esfuerzo, en gran parte, de un sola persona, Phil Zimmermann. PGP proporciona un servicio de confidencialidad y de autenticación que se puede usar para correo electrónico y aplicaciones de almacenamiento de ficheros. Básicamente, Zimmermann ha hecho lo siguiente:

¹ Bendice a quien lo creó / y reza porque aún viva /un millón pudo hacer/y a los federales vender/pero tenía ansias de libertad/y lo dio por nada a cambio/ahora cualquier hombre de a pie tiene PGP.

1. Seleccionar, como base, los mejores algoritmos criptográficos existentes.
2. Integrar estos algoritmos en una aplicación de propósito general independiente del sistema operativo y del procesador, y que se basa en un grupo reducido de comandos fáciles de usar.
3. Ofrecer gratuitamente el paquete y su documentación, incluido el código fuente, por medio de Internet, tablones de anuncios y redes comerciales como la AOL (*America on Line*).
4. Llegar a un acuerdo con una compañía (Viacrypt, ahora Network Associates) para proporcionar una versión comercial de PGP totalmente compatible y de bajo coste.

PGP ha crecido rápidamente y está muy extendido, lo cual se debe a las siguientes razones:

1. Está disponible de forma gratuita en versiones que se ejecutan en una gran variedad de plataformas, incluidas Windows, UNIX, Macintosh y muchas más. Además, la versión comercial satisface a los usuarios que quieren un producto con asistencia del fabricante.
2. Se basa en algoritmos que han sobrevivido a revisiones exhaustivas y se consideran sumamente seguros. Concretamente, el paquete incluye RSA, DSS y Diffie-Hellman para cifrado de clave pública; CAST-128, IDEA y 3DES para cifrado simétrico; y SHA-1 para codificación *hash*.
3. Tiene un amplio ámbito de aplicabilidad, desde corporaciones que desean seleccionar y reforzar un esquema normalizado para cifrar archivos y mensajes, hasta particulares que desean comunicarse de forma segura con usuarios de todo el mundo por medio de Internet y otras redes de computadores.
4. No fue desarrollado por ninguna organización gubernamental o de estándares, ni lo controlan en la actualidad. Esto hace que PGP sea atractivo para aquellas personas que desconfían instintivamente del «sistema».
5. En la actualidad, PGP está propuesto como estándar de Internet (RFC 3156). Sin embargo, todavía lo rodea un aura de resistencia a lo establecido.

Empezaremos ofreciendo una visión general del funcionamiento de PGP. Luego, examinaremos la forma en que se crean y se almacenan las claves criptográficas, para pasar a continuación al aspecto fundamental relativo a la gestión de claves públicas.

NOTACIÓN

La mayor parte de la notación que se emplea en este Capítulo se ha usado anteriormente, pero se introducen algunos términos nuevos. Lo mejor será resumirlos al principio. Se usan los siguientes símbolos:

K_S = clave de sesión usada en el esquema de cifrado simétrico

KR_a = clave privada del usuario A, utilizada en el esquema de cifrado de clave pública

KU_a = clave pública del usuario A, empleada en el esquema de cifrado de clave pública

EP = cifrado de clave pública

DP = descifrado de clave pública

EC = cifrado simétrico

DC = descifrado simétrico

H = función *hash*

|| = concatenación

Z = compresión usando el algoritmo ZIP

R64 = conversión al formato ASCII radix 64

La documentación de PGP usa con frecuencia el término *clave secreta* para referirse a una clave emparejada con una clave pública en un esquema de cifrado de esta clave. Como se apuntó con anterioridad, este hecho puede llevar a confusión con la clave secreta que se utiliza en el cifrado simétrico. Por este motivo, en este libro se emplea el término *clave privada*.

DESCRIPCIÓN OPERATIVA

La operación real de PGP, al contrario que la gestión de claves, consiste en cinco servicios: autentificación, confidencialidad, compresión, compatibilidad con correo electrónico y segmentación (Tabla 5.1). Examinemos cada uno de ellos.

Autentificación

La Figura 5.1a ilustra el servicio de firma digital suministrado por PGP. Este es el esquema de firma digital que se estudió en el Capítulo 3 y se mostró en la Figura 3.2c. La secuencia es la siguiente:

- 1.** El emisor crea un mensaje.
- 2.** Se usa SHA-1 para generar un código *hash* del mensaje de 160 bits.
- 3.** El código *hash* se cifra con RSA usando la clave privada del emisor y el resultado se añade antepuesto al mensaje.
- 4.** El receptor usa RSA con la clave pública del emisor para descifrar y recuperar el código *hash*.
- 5.** El receptor genera un nuevo código *hash* para el mensaje y lo compara con el código *hash* descifrado. Si los dos coinciden, el mensaje se considera auténtico y se acepta.

La combinación de SHA-1 y RSA ofrece un esquema eficaz de firma digital. Por una parte, debido a la robustez del RSA, el receptor está seguro de que sólo el poseedor de la clave privada correspondiente puede generar la firma; por otra, debido a la robustez de SHA-1, el receptor está seguro de que nadie más puede generar un nuevo mensaje que coincida con el código *hash* y, por lo tanto, con la firma del mensaje original.

Una alternativa para la generación de firmas podría ser el uso de DSS/SHA-1.

Tabla 5.1 Resumen de los servicios de PGP

Función	Algoritmos usados	Descripción
Firma digital	DSS/SHA o RSA/SHA	Se crea un código <i>hash</i> de un mensaje usando SHA-1. El resumen del mensaje se cifra usando DSS o RSA con la clave privada del emisor, y se incluye en el mensaje.
Cifrado de mensaje	CAST o IDEA o TripleDES de tres claves con Diffie-Hellman o RSA	Se cifra un mensaje usando CAST-128, IDEA o 3DES con una clave de sesión de un solo uso generada por el emisor. La clave de sesión se cifra usando Diffie-Hellman o RSA con la clave pública del receptor, y se incluye en el mensaje.
Compresión	ZIP	Un mensaje se puede comprimir usando ZIP para su almacenamiento o transmisión.
Compatibilidad con correo electrónico	Conversión radix 64	Para proporcionar transparencia para las aplicaciones de correo electrónico, un mensaje cifrado se puede convertir en una ristra ASCII usando conversión radix 64.
Segmentación	—	Para ajustarse a las limitaciones de tamaño de mensaje, PGP realiza la segmentación y el reensamblado.

Aunque las firmas se encuentran normalmente adjuntas al mensaje o el fichero que firman, no siempre es así: también pueden encontrarse separadas. Una firma puede almacenarse y transmitirse separada del mensaje que firma, lo cual es útil en varios contextos. Un usuario podría desear mantener un fichero de seguimiento (*log*) con las firmas separadas de todos los mensajes enviados y recibidos. Además, una firma separada de un programa ejecutable puede detectar un virus. Por último, las firmas separadas se pueden usar cuando más de una parte debe firmar un documento, como ocurre, por ejemplo, en un contrato legal. La firma de cada persona es independiente y, por lo tanto, se aplica sólo al documento. De no ser así, las firmas tendrían que anidarse, con lo cual el segundo firmante firma el documento y la primera firma, y así sucesivamente.

CONFIDENCIALIDAD

Otro servicio básico que proporciona PGP es la confidencialidad, lo cual se consigue cifrando los mensajes que van a transmitirse o a almacenarse localmente como ficheros. En ambos casos se puede usar el algoritmo de cifrado simétrico CAST-128 y, como alternativa, IDEA o 3DES, utilizando siempre el modo CFB de 64 bits.

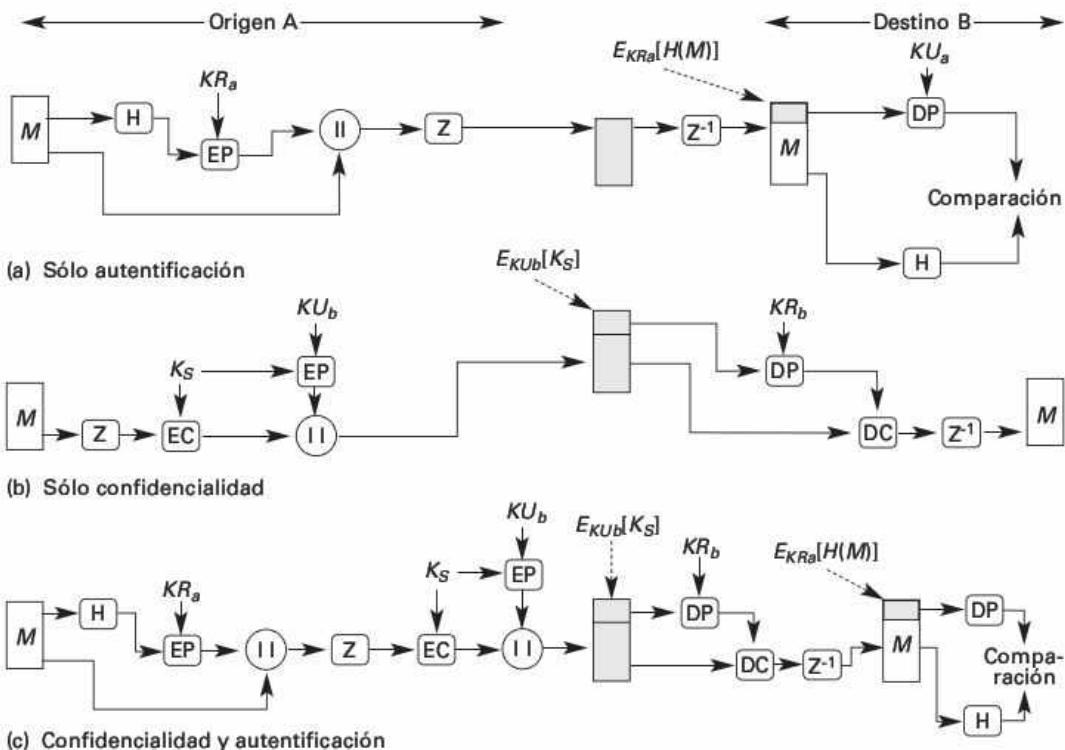


Figura 5.1 Funciones criptográficas de PGP

Como es habitual, se debe afrontar el problema de la distribución de claves. En PGP, cada clave simétrica se usa una sola vez. Es decir, una nueva clave se genera como un número aleatorio de 128 bits para cada mensaje. Así, aunque a ésta se le conoce en la documentación como clave de sesión, en realidad se trata de una clave de un solo uso. Como ha de usarse una sola vez, la clave de sesión se añade al mensaje y se transmite con él. Para proteger la clave, se cifra con la clave pública del receptor. La Figura 5.1b muestra la secuencia, que se puede describir de la siguiente manera:

1. El emisor genera un mensaje y un número aleatorio de 128 bits para usarlo como clave de sesión sólo para este mensaje.
2. El mensaje se cifra, usando CAST-128 (o IDEA o 3DES) con la clave de sesión.
3. La clave de sesión se cifra con RSA, usando la clave pública del receptor, y se añade antepuesta al mensaje.
4. El receptor usa RSA con su clave privada para descifrar y recuperar la clave de sesión.
5. La clave de sesión se usa para descifrar el mensaje.

Como alternativa al uso de RSA para el cifrado de la clave, PGP proporciona la opción conocida como *Diffie-Hellman*. Tal y como se explicó en el Capítulo 3, Diffie-Hellman es un algoritmo de intercambio de claves. De hecho, PGP usa una variante de Diffie-Hellman que proporciona cifrado/descifrado, conocido como ElGamal.

Se pueden hacer algunas observaciones al respecto. En primer lugar, para reducir el tiempo de cifrado es preferible usar la combinación de cifrado simétrico y de clave pública que usar simplemente RSA o ElGamal para cifrar el mensaje directamente: CAST-128 y los demás algoritmos simétricos son sensiblemente más rápidos que RSA o ElGamal. Por otra parte, el uso del algoritmo de clave pública resuelve el problema de la distribución de las claves de sesión, porque sólo el receptor puede recuperar la clave de sesión que va unida al mensaje. Cada mensaje consiste en un elemento único e independiente con su propia clave. Además, dado que el correo electrónico se almacena y se reenvía, no son prácticas las negociaciones para garantizar que las dos partes que se comunican tienen la misma clave de sesión. Por último, el empleo de claves simétricas de un solo uso refuerza lo que ya es un enfoque robusto de cifrado simétrico. Sólo se cifra una pequeña cantidad de texto claro con cada clave y no existe relación entre las claves. Así, hasta donde el algoritmo de clave pública es seguro, el esquema total es seguro. Con este fin, PGP proporciona al usuario una gama de opciones de tamaño de clave, desde 768 hasta 3072 bits (la clave DSS para firmas se limita a 1024 bits).

CONFIDENCIALIDAD Y AUTENTIFICACIÓN

Como muestra la Figura 5.1c, ambos servicios se pueden usar para el mismo mensaje. Primero, se genera una firma para el mensaje en texto claro y se adjunta antepuesta a dicho mensaje. Luego, se cifra el mensaje en texto claro y la firma usando CAST-128 (o IDEA o 3DES), y la clave de sesión se cifra usando RSA (o ElGamal). Esta secuencia es preferible a la secuencia inversa, que consistiría en cifrar el mensaje y luego generar una firma para el mensaje cifrado. Generalmente, es más conveniente almacenar una firma con una versión en texto claro del mensaje. Además, para la verificación de la tercera parte, si la firma se lleva a cabo en primer lugar, la tercera parte no necesita ocuparse de la clave simétrica al verificar la firma.

En resumen, cuando se usan los dos servicios, primero el emisor firma el mensaje con su propia clave privada, luego cifra el mensaje con una clave de sesión y a continuación cifra la clave de sesión con la clave pública del receptor.

COMPRESIÓN

De forma predeterminada, PGP comprime el mensaje después de aplicar la firma, pero antes del cifrado. Esto tiene la ventaja de ahorrar espacio tanto para la transmisión de correo electrónico como para el almacenamiento de ficheros.

La ubicación del algoritmo de compresión, indicado en la Figura 5.1 con Z para compresión y Z^{-1} para descompresión, es crítica:

1. La firma se genera antes de la compresión debido a dos razones fundamentales:
 - a) Es preferible firmar un mensaje descomprimido para poder almacenar solamente el mensaje descomprimido junto con la firma para su verificación posterior. Si se firma un documento comprimido, sería necesario almacenar una versión comprimida del mensaje para su posterior verificación o volver a comprimir el mensaje cuando se requiera verificación.

- b)** Incluso si se estuviese dispuesto a generar de forma dinámica un mensaje que se ha vuelto a comprimir para su verificación, el algoritmo de compresión de PGP presenta una dificultad. El algoritmo no es determinista; distintas implementaciones del algoritmo permiten diferentes compromisos entre la velocidad de ejecución y el ratio de compresión y, como resultado, producen distintas formas comprimidas. Sin embargo, los distintos algoritmos de compresión pueden operar entre sí, ya que cualquier versión del algoritmo puede descomprimir correctamente la salida de cualquier otra versión. Aplicar la función *hash* y la firma después de la compresión restringiría todas las implementaciones de PGP a la misma versión del algoritmo de compresión.
- 2** El cifrado del mensaje se aplica después de la compresión para reforzar la seguridad criptográfica. Como el mensaje comprimido tiene menos redundancia que el texto claro original, el criptoanálisis presenta más dificultades. El algoritmo de compresión que se utiliza es ZIP, que se describe en el Apéndice 5a.

COMPATIBILIDAD CON CORREO ELECTRÓNICO

Cuando se usa PGP, se cifra al menos una parte del bloque que se va a transmitir. Si sólo se usa el servicio de firma, se cifra el resumen del mensaje (con la clave privada del emisor). Si se usa el servicio de confidencialidad, se cifran (con una clave simétrica de un solo uso) el mensaje y la firma (si estuviese presente). Por consiguiente, parte del bloque resultante consiste en una ristra de octetos arbitrarios de ocho bits. Sin embargo, muchos sistemas de correo electrónico sólo permiten el uso de bloques de texto ASCII. Para ajustarse a esta restricción, PGP proporciona el servicio de convertir la ristra binaria de ocho bits en una ristra de caracteres ASCII imprimibles.

El esquema que se usa para ello es la conversión radix 64. A partir de cada grupo de tres octetos de datos binarios se obtienen cuatro caracteres ASCII. Este formato también añade un CRC para detectar errores de transmisión. El Apéndice 5B ofrece la descripción.

El uso de radix 64 expande un mensaje un 33%. Afortunadamente, las partes del mensaje correspondientes a la clave de sesión y a la firma son relativamente compactas, y el mensaje en texto claro ha sido comprimido. De hecho, la compresión debería ser más que suficiente para compensar la expansión de radix 64. Por ejemplo, en [HELD96] se presenta un promedio del ratio de compresión de 2.0 usando ZIP. Si ignoramos los componentes relativamente pequeños de la firma y la clave, el efecto global habitual de la compresión y la expansión de un archivo de longitud X sería $1.33 \times 0.5 \times X = 0.665 \times X$. Así, todavía hay una compresión general de un tercio aproximadamente.

Un aspecto destacable del algoritmo radix 64 es que convierte la ristra de entrada a formato radix 64 independientemente del contenido, incluso si la entrada es texto ASCII. Por consiguiente, si un mensaje está firmado, pero no cifrado, y la conversión se aplica al bloque completo, la salida será ilegible al observador casual, lo cual proporciona un cierto grado de confidencialidad. De manera opcional, PGP puede configurarse para convertir a formato radix 64 sólo la parte de la firma de los mensajes firmados en texto claro. Esto permite que el receptor humano lea el mensaje sin usar PGP. Sin embargo, aún tendría que usarse PGP para verificar la firma.

La Figura 5.2 muestra la relación entre los cuatro servicios tratados hasta ahora. En la transmisión, si es necesario, se genera una firma usando un código *hash* del texto claro descomprimido. Luego se comprime el texto claro y, si está presente, también la firma. A continuación, si se necesita confidencialidad, se cifra el bloque (texto claro comprimido o firma comprimida más texto claro) y se añade antepuesto con la clave de cifrado simétrico cifrada con clave pública. Por último, el bloque completo se convierte a formato radix 64.

En la recepción, el bloque que se recibe se convierte de formato radix 64 nuevamente a binario. Luego, si el mensaje está cifrado, el receptor recupera la clave de sesión y descifra el mensaje. El bloque resultante, luego, se descomprime. Si el mensaje está firmado, el receptor recupera el código *hash* transmitido y lo compara con su propio cálculo del código *hash*.

SEGMENTACIÓN Y REENSAMBLADO

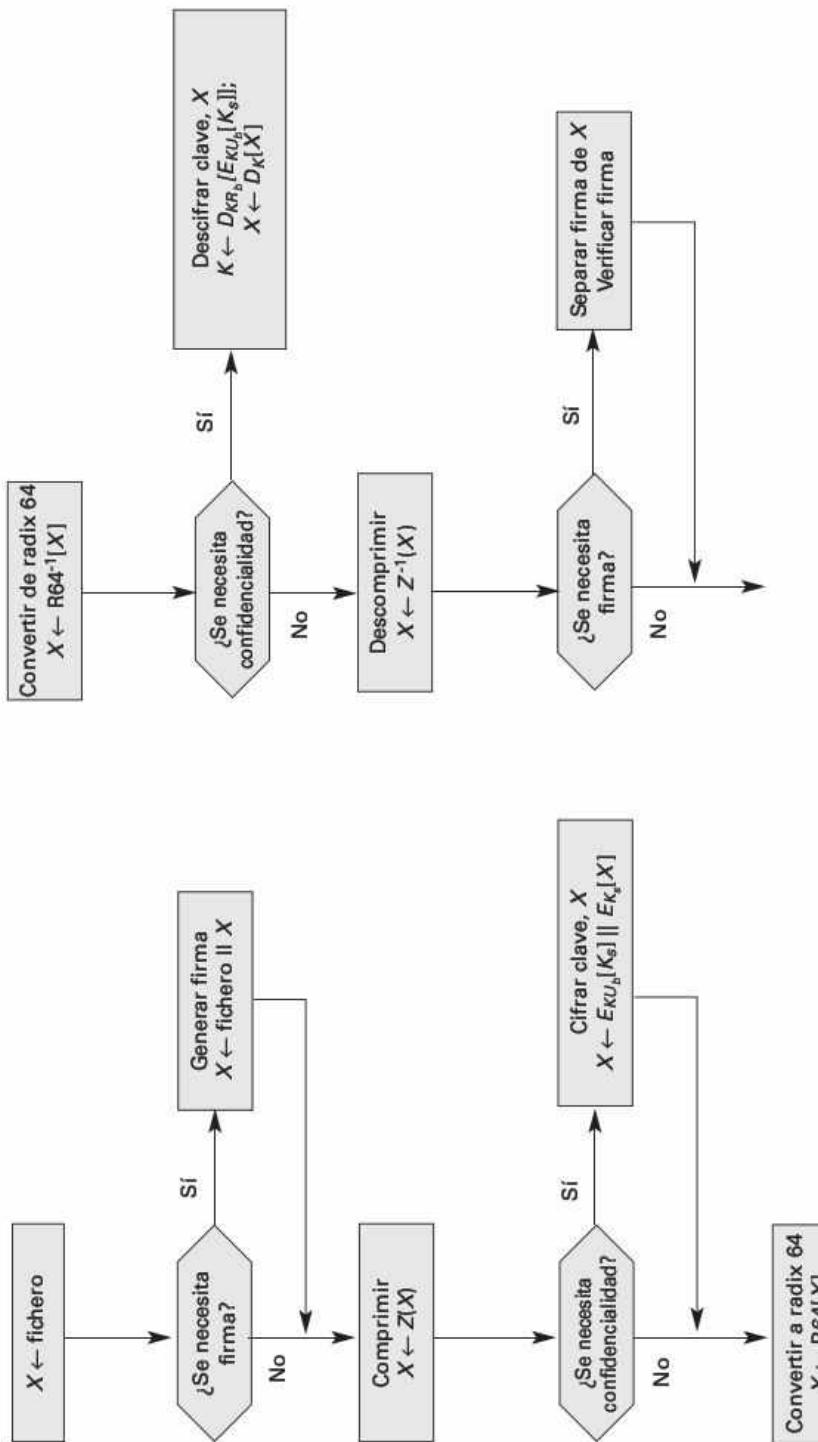
Las herramientas de correo electrónico se limitan con frecuencia a una longitud máxima de mensaje. Por ejemplo, muchas de las herramientas accesibles a través de Internet imponen una longitud máxima de 50.000 octetos. Cualquier mensaje mayor debe subdividirse en segmentos más reducidos, cada uno de los cuales se envía por separado.

Para ajustarse a esta restricción, PGP subdivide automáticamente los mensajes demasiado largos en segmentos lo suficientemente cortos para ser enviados por correo electrónico. La segmentación se lleva a cabo después de todo el procesamiento, incluida la conversión de radix 64. Por lo tanto, el componente clave de sesión y el componente firma aparecen una sola vez, al principio del primer segmento. En el extremo receptor, PGP debe retirar todas las cabeceras del correo electrónico y reensamblar el bloque original completo antes de realizar los pasos que se muestran en la Figura 5.2b.

CLAVES CRIPTOGRÁFICAS Y FICHEROS DE CLAVES

PGP hace uso de cuatro tipos de claves: claves simétricas de sesión de un solo uso, claves públicas, claves privadas y claves simétricas basadas en frases clave (que se explicarán más adelante). Con respecto a estas claves, se pueden identificar tres requisitos:

1. Se necesita un medio para la generación imprevisible de claves de sesión.
2. Nos gustaría permitir que un usuario tenga múltiples parejas de clave pública/clave privada. El motivo de esto es que el usuario podría querer cambiar su pareja de claves de vez en cuando. Cuando esto ocurre, cualquier mensaje en proceso se creará con una clave obsoleta. Además, los receptores sólo conocerán la clave pública antigua hasta recibir una actualización. Aparte de la necesidad de cambiar las claves de vez en cuando, un usuario podría querer tener múltiples parejas de claves en un momento dado para interactuar con diferentes grupos de interlocutores o simplemente para mejorar la seguridad limitando la cantidad de material cifrado con una de las claves. El resultado de todo esto es que no hay una correspondencia de uno a uno entre los usuarios y sus claves públicas. Por lo tanto, se necesita algún medio para identificar claves particulares.



(a) Diagrama genérico de transmisión

(b) Diagrama genérico de recepción (desde A)

Figura 5.2 Transmisión y recepción de mensajes PGP

3. Cada entidad PGP debe mantener un archivo con sus propias parejas de claves y otro con las claves públicas de los interlocutores.

Examinemos cada uno de estos requisitos.

GENERACIÓN DE CLAVES DE SESIÓN

Cada clave de sesión está asociada a un solo mensaje y se usa sólo con el fin de cifrar y descifrar ese mensaje. Recordemos que el cifrado/descifrado de mensajes se realiza con un algoritmo de cifrado simétrico. CAST-128 e IDEA usan claves de 128 bits; 3DES usa una de 168 bits. Para lo que se expone a continuación, adoptamos CAST-128.

Se generan números aleatorios de 128 bits usando CAST-128. La entrada al generador de números aleatorios consiste en una clave de 128 bits y dos bloques de 64 bits que se tratan como texto claro que se va a cifrar. Usando el modo de realimentación de cifrado (CFB), el cifrador CAST-128 produce dos bloques de texto cifrado de 64 bits, que se concatenan para formar la clave de sesión de 128 bits. El algoritmo que se usa se basa en el que se especifica en ANSI X12.17.

La entrada «texto claro» al generador de números aleatorios, que consiste en dos bloques de 64 bits, procede de una ristra de números generados de forma aleatoria de 128 bits. Estos números se basan en entradas de pulsaciones de teclas por parte del usuario. El tiempo de pulsación y las teclas pulsadas se usan para generar la ristra aleatoria. Por lo tanto, si el usuario pulsa teclas arbitrarias a su ritmo normal, se generará una entrada razonablemente «aleatoria». Esta entrada aleatoria también se combina con la salida de la clave de sesión anterior de CAST-128 para formar la entrada de la clave al generador. El resultado de la alteración de CAST-128 es producir una secuencia de claves de sesión efectivamente impredecible.

El Apéndice 5C trata más detalladamente las técnicas de generación de números aleatorios de PGP.

IDENTIFICADORES DE CLAVE

Como hemos dicho, un mensaje cifrado está acompañado de una forma cifrada de la clave de sesión que se empleó. La clave de sesión se cifra con la clave pública del receptor. Así, sólo el receptor podrá recuperar la clave de sesión y, por consiguiente, el mensaje. Si cada usuario utilizó una única pareja de claves pública/privada, el receptor debería saber automáticamente qué clave usar para descifrar la clave de sesión: la clave privada única del receptor. Sin embargo, no olvidemos el requisito de que cualquier usuario dado podría tener múltiples parejas de claves.

Entonces, ¿cómo sabe el receptor cuál de sus claves públicas se usó para cifrar la clave de sesión? Una solución simple sería transmitir la clave pública con el mensaje. Entonces, el receptor podría verificar que efectivamente se trata de una de sus claves públicas, y continuar. Este esquema funcionaría, pero constituye un gasto innecesario de espacio. Una clave pública RSA podría tener una longitud de cientos de dígitos decimales. Otra solución sería asociar un identificador a cada clave pública que sea única al menos en un usuario. Es decir, la combinación del identificador de usuario (*user ID*) y

el identificador de clave (*key ID*) sería suficiente para identificar una clave en especial. Entonces, sólo sería necesario transmitir el identificador de clave más corto. Sin embargo, esta solución trae consigo un problema de gestión y de costes adicionales: los identificadores de clave deben ser asignados y almacenados para que tanto el emisor como el receptor puedan establecer la relación entre identificador de clave y clave pública. Esto parece una carga innecesaria.

La solución adoptada por PGP es la de asignar un identificador de clave a cada clave pública que, con un gran índice de probabilidad, es única en el identificador de un usuario. El identificador de clave asociado con cada clave pública consiste en sus 64 bits menos significativos. Es decir, el identificador de clave de clave pública KU_a es $(KU_a \bmod 2^{64})$. Esta es una longitud suficiente para que la probabilidad de duplicación de identificadores de clave sea muy pequeña.

También se necesita un identificador de clave para la firma digital PGP. Como un emisor puede usar una clave privada, de una serie de claves privadas, para cifrar el resumen del mensaje, el receptor debe saber qué clave pública se debe usar. Por lo tanto, el componente firma digital de un mensaje incluye el identificador de clave de 64 bits de la clave pública que se requiere. Cuando se recibe el mensaje, el receptor verifica que el identificador de clave es el de una clave pública para ese emisor y entonces procede a verificar la firma.

Ahora que se ha introducido el concepto de identificador de clave, podemos observar en profundidad el formato de un mensaje transmitido, que se muestra en la Figura 5.3. Un mensaje está formado por tres componentes: el componente de mensaje, el componente de firma (opcional) y el componente de clave de sesión (opcional).

El **componente mensaje** incluye los datos reales que se van a almacenar o transmitir, así como un nombre de archivo y un sello de tiempo que especifica el momento de creación.

El **componente firma** incluye lo siguiente:

- **Sello de tiempo:** el momento en que se creó la firma.
- **Resumen de mensaje:** el resumen SHA-1 de 160 bits, cifrado con la clave de firma privada del emisor. El resumen se calcula con el sello de tiempo de la firma concatenado con la parte de datos del componente de mensaje. La inclusión del sello de tiempo de la firma en el resumen evita los ataques de repetición. La exclusión de las partes del nombre del archivo y sello de tiempo del componente de mensaje garantiza que las firmas adjuntas son exactamente las mismas que las firmas adjuntas antepuestas al mensaje. Las firmas separadas se calculan en un fichero separado que no tiene ninguno de los campos de cabecera del componente de mensaje.
- **Dos octetos iniciales del resumen del mensaje:** para permitir que el receptor determine si se usó la clave pública correcta para descifrar el resumen del mensaje para la autenticación, se compara la copia en texto claro de los dos primeros octetos con los dos primeros octetos del resumen descifrado. Estos octetos también sirven de comprobación del marco de 16 bits para el mensaje.
- **Identificador de clave de la clave pública del emisor:** identifica la clave pública que debería usarse para descifrar el resumen del mensaje y, por lo tanto, identifica la clave privada que se utilizó para cifrar el resumen del mensaje.

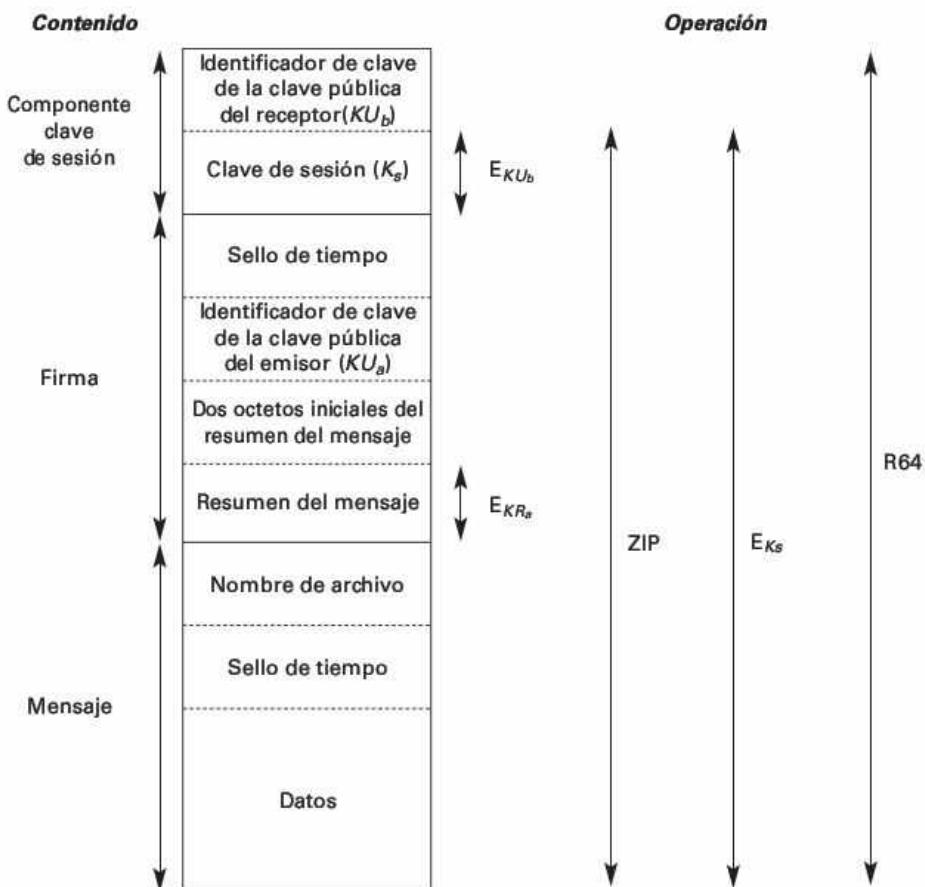


Figura 5.3 Formato general del mensaje PGP (de A a B)

El componente de mensaje y el componente opcional de firma pueden comprimirse usando ZIP y pueden cifrarse usando una clave de sesión.

El **componente clave de sesión** incluye la clave de sesión y el identificador de la clave pública del receptor que utilizó el emisor para cifrar la clave de sesión.

El bloque entero se codifica normalmente con radix 64.

FICHEROS DE CLAVES

Hemos comprobado que los identificadores de clave son críticos para la operación de PGP y que en cualquier mensaje PGP que proporcione confidencialidad y autenticación se incluyen dos identificadores de clave. Es necesario almacenar y organizar estas claves de forma sistemática para que todas las partes las usen de forma eficaz y efectiva.

va. El esquema usado en PGP es el de proporcionar un par de estructuras de datos en cada nodo, una para almacenar las parejas de claves pública/privada pertenecientes a ese nodo, y otra para almacenar las claves públicas de otros usuarios conocidos en ese nodo. Estas estructuras de datos se conocen, respectivamente, como fichero de claves privadas y fichero de claves públicas.

La Figura 5.4 muestra la estructura general de un **fichero de claves privadas**. Podemos ver el fichero como una tabla en la que cada fila representa una de las parejas de claves pública/privada que posee ese usuario. Cada fila contiene las siguientes entradas:

- **Sello de tiempo:** fecha y hora en que se generó la pareja de claves.
- **Identificador de clave:** los 64 bits menos significativos de la clave pública para esa entrada.
- **Clave pública:** la parte de clave pública de la pareja en cuestión.
- **Clave privada:** la parte de clave privada de la pareja en cuestión; este campo está cifrado.
- **Identificador de usuario:** normalmente, será la dirección de correo electrónico del usuario (por ejemplo, stallings@acm.org). Sin embargo, el usuario puede elegir asociar un nombre diferente a cada pareja (por ejemplo, Stalllings, WStallings, William-Stallings, etc.) o reutilizar el mismo identificador de usuario más de una vez.

Fichero de claves privadas

<i>Sello de tiempo</i>	<i>Identificador de clave</i>	<i>Clave pública</i>	<i>Clave privada cifrada</i>	<i>Identificador de usuario*</i>
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$KU_i \ mod \ 2^{64}$	KU_i	$E_{H(P_i)}(KR_i)$	<i>Usuario i</i>
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Fichero de claves públicas

<i>Sello de tiempo</i>	<i>Identificador de clave*</i>	<i>Clave pública</i>	<i>Confianza en el dueño</i>	<i>Identificador de usuario*</i>	<i>Legitimidad de clave</i>	<i>Firma(s)</i>	<i>Confianza en la firma</i>
•	•	•	•	•	•	•	•
T_i	$KU_i \ mod \ 2^{64}$	KU_i	$trus_flag_i$	<i>Usuario i</i>	$trus_flag_i$		
•	•	•	•	•	•	•	•

* campo utilizado para indexar la tabla

Figura 5.4 Estructura general de los ficheros de claves públicas y privadas

El fichero de claves privadas se puede indexar por el identificador de usuario o el identificador de clave; más adelante veremos la necesidad de los dos medios de indexación.

Aunque se intenta que el fichero de claves privadas se almacene sólo en la máquina del usuario que creó y posee la pareja de claves, y que sólo ese usuario pueda acceder a él, tiene sentido hacer que el valor de la clave privada sea lo más seguro posible. Por lo tanto, la clave privada no se almacena en el fichero de claves. En vez de eso, esta clave se cifra usando CAST-128 (o IDEA o 3DES). El procedimiento es el siguiente:

1. El usuario elige una frase clave para el cifrado de claves privadas.
2. Cuando el sistema genera una nueva pareja de claves pública/privada usando RSA, pide la frase clave al usuario. Usando SHA-1, se genera un código *hash* de 160 bits a partir de la frase clave, y la frase clave se descarta.
3. El sistema cifra la clave privada usando CAST-128 con los 128 bits del código *hash* como clave. Luego el código *hash* se descarta, y la clave privada cifrada se almacena en el fichero de claves privadas.

Posteriormente, cuando un usuario accede al fichero de claves privadas para recuperar una clave privada, debe suministrar la frase clave. PGP recuperará la clave privada cifrada, generará el código *hash* de la frase clave y descifrará la clave privada cifrada usando CAST-128 con el código *hash*.

Este es un esquema muy compacto y eficaz. Como en cualquier sistema que se base en contraseñas, la seguridad de este sistema depende de la seguridad de la contraseña. Para evitar la tentación de escribirla, el usuario debería utilizar una frase clave que no sea fácil de adivinar, pero sí fácil de recordar.

La Figura 5.4 también presenta la estructura general de un archivo de claves públicas. Esta estructura de datos se usa para almacenar claves públicas de otros usuarios conocidos por este usuario. Por el momento, dejamos pendientes algunos de los campos que se muestran en la tabla y describiremos los siguientes:

- **Sello de tiempo:** fecha y hora en que se generó la entrada.
- **Identificador de clave:** los 64 bits menos significativos de la clave pública para esta entrada.
- **Clave pública:** clave pública para esta entrada.
- **Identificador de usuario:** identifica al propietario de la clave. Varios identificadores de usuario pueden estar asociados a una sola clave pública.

El fichero de claves públicas se puede indexar por identificador de usuario o identificador de clave; veremos posteriormente que los dos medios de indexación son necesarios.

Ahora podemos observar cómo se usan estos ficheros de claves en la transmisión y la recepción de mensajes. Para simplificar la explicación, dejaremos de lado la compresión y la conversión de radix 64. En primer lugar, consideraremos la transmisión del mensaje (Figura 5.5) y asumamos que el mensaje debe firmarse y cifrarse. La entidad PGP emisora realiza los siguientes pasos:

1. Firmar el mensaje
 - a) PGP recupera la clave privada del emisor del fichero de claves privadas usando tu_identificador de usuario (*your_userid*) como índice. Si tu-identificador

de usuario no se proporcionó en el comando, se recupera la primera clave del fichero.

- b)** PGP solicita al usuario la frase clave para recuperar la clave privada que no está cifrada.
 - c)** Se construye el componente de firma del mensaje.
- 2** Cifrar el mensaje
- a)** PGP genera una clave de sesión y cifra el mensaje.
 - b)** PGP recupera la clave pública del receptor del fichero de claves públicas usando su identificador de usuario (*her-userid*) como índice.
 - c)** Se construye el componente de clave de sesión del mensaje.

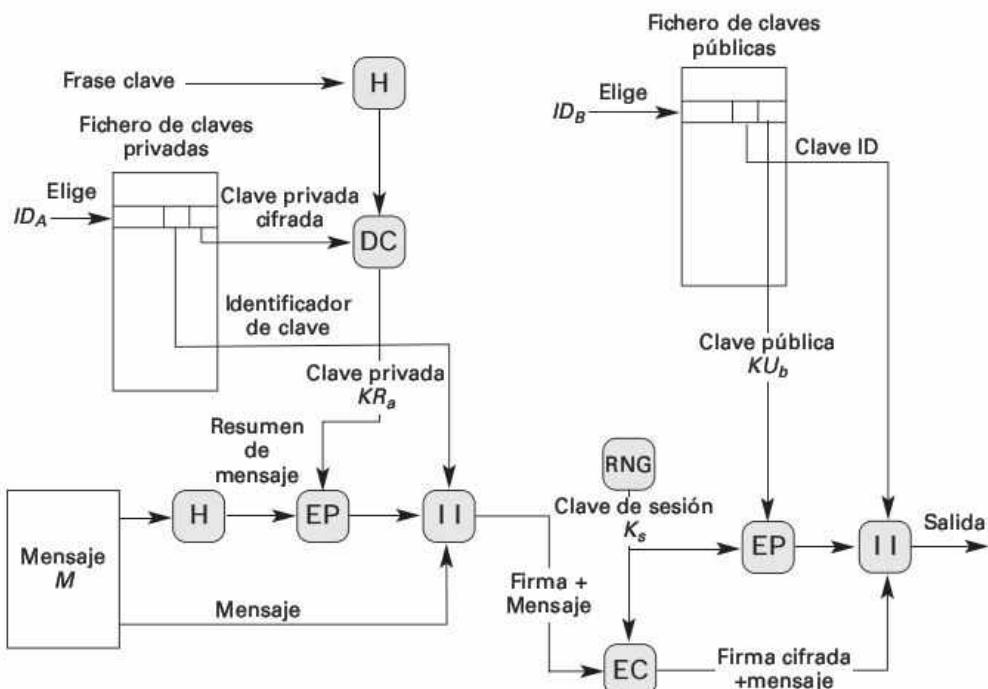


Figura 5.5 Generación de mensaje PGP (del usuario A al usuario B; sin compresión ni conversión de radix 64)

La entidad PGP receptora realiza los siguientes pasos (Figura 5.6):

- 1.** Descifrar el mensaje
 - a)** PGP recupera la clave privada del receptor seleccionada del fichero de claves privadas, usando como índice el campo identificador de clave del componente clave de sesión del mensaje.
 - b)** PGP pide al usuario la frase clave para recuperar la clave privada sin cifrar.
 - c)** Luego, PGP recupera la clave de sesión y descifra el mensaje.

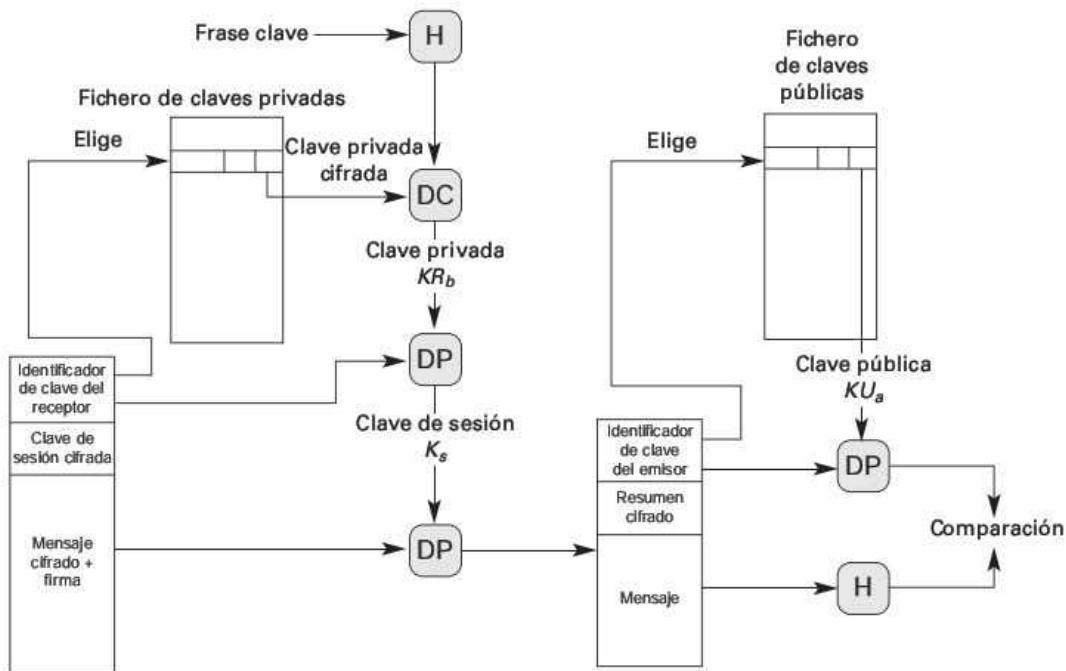


Figura 5.6 Recepción PGP (del usuario A al usuario B; sin compresión ni conversión de radix 64)

2. Autentificar el mensaje

- PGP recupera la clave pública del emisor del fichero de claves públicas, usando como índice el campo identificador de clave en el componente clave de firma del mensaje.
- PGP recupera el resumen del mensaje transmitido.
- PGP calcula el resumen de mensaje para el mensaje recibido y lo compara con el resumen de mensaje transmitido para autentificar.

GESTIÓN DE CLAVE PÚBLICA

Como se ha podido observar, PGP contiene una serie de funciones y formatos inteligentes y eficientes entrelazados para proporcionar un servicio efectivo de confidencialidad y autenticación. Para completar el sistema, es necesario tratar un aspecto final, el de la gestión de claves públicas. La documentación PGP presenta la importancia de este aspecto:

Toda la cuestión sobre la protección de claves públicas ante la posible falsificación o irrupción es el problema más difícil en las aplicaciones prácticas de clave pública. Es el talón de Aquiles de la criptografía de clave pública, y el software se hace cada vez más complejo para resolver este problema.

PGP ofrece una estructura para resolver este problema, para lo cual se pueden tener en cuenta algunas opciones. Como PGP está diseñado para su uso en diferentes entornos formales e informales, no se establece un esquema rígido de gestión de claves públicas, como veremos que ocurre en S/MIME en este Capítulo.

ENFOQUES PARA LA GESTIÓN DE CLAVES PÚBLICAS

La esencia del problema es la siguiente: el usuario A debe crear un fichero de claves públicas que contenga las claves públicas de otros usuarios para interoperar con ellos usando PGP. Supongamos que el fichero de claves de A contiene una clave pública atribuida a B pero que, de hecho, pertenece a C. Esto podría ocurrir si, por ejemplo, A obtuvo la clave de un sistema de tablón de anuncios que fue usado por B para publicar la clave pública y que, a su vez, se vio comprometido por C. Esto da como resultado dos amenazas. En primer lugar, C puede enviar mensajes a A y falsificar la firma de B, de forma que A aceptará el mensaje como si proviniese de B. En segundo lugar, cualquier mensaje cifrado de A a B puede ser leído por C.

Existen algunos enfoques para reducir el riesgo de que el fichero de claves públicas de un usuario contenga claves públicas falsas. Supongamos que A quiere obtener una clave pública fiable para B. A continuación se exponen algunos de los enfoques que pueden usarse:

1. Obtener la clave de B físicamente. B podría almacenar su clave pública (KU_B) en un disquete y entregarla a A. Luego A podría cargar la clave en su sistema desde el disquete. Este es un método seguro pero tiene limitaciones prácticas obvias.
2. Verificar una clave por teléfono. Si A puede reconocer a B al teléfono, A podría llamar a B y pedirle que le dicte la clave, en formato radix 64. Como alternativa más práctica, B podría transmitir su clave por correo electrónico a A. El usuario A podría hacer que PGP genere un resumen SHA-1 de la clave de 160 bits y mostrarlo en formato hexadecimal; esto se conoce como la «huella» de la clave. A podría, luego, llamar a B y pedirle que le dicte la huella por teléfono. Si las dos huellas coinciden, se verifica la clave.
3. Obtener la clave pública de B a partir de un usuario D de confianza mutua, o presentador. Con este fin, D crea un certificado firmado. El certificado incluye la clave pública de B, la fecha de creación de la clave y el período de validez de la misma. D genera un resumen SHA-1 de este certificado, lo cifra con su clave privada y añade la firma al certificado. Como sólo D podía haber creado la firma, nadie más puede crear una clave pública falsa y hacer creer que fue firmada por D. El certificado firmado podría ser enviado directamente a A por B o D, o podría publicarse en un tablón de anuncios.
4. Obtener la clave pública de B a partir de una autoridad de certificación confiable. De nuevo, un certificado de clave pública lo crea y firma la autoridad. Entonces, A podría acceder a la autoridad, proporcionando un nombre de usuario y recibiendo un certificado firmado.

Para los casos 3 y 4, A ya tendría que tener una copia de la clave pública de D y confiar en que dicha clave sea válida. Por último, confiar en quien actúe como presentador es decisión de A.

EL USO DE LA CONFIANZA

Aunque PGP no incluye ninguna especificación para establecer autoridades de certificación o para establecer la confianza, sí que proporciona un medio conveniente de usarla, asociándola a las claves públicas y explotando la información de confianza.

La estructura básica es la siguiente: cada entrada en el fichero de claves públicas es un certificado de clave pública, tal y como se describió en el apartado anterior. Asociado con cada entrada hay un **campo de legitimidad de la clave**, que indica hasta qué punto PGP confiará en que se trata de una clave pública válida para este usuario; cuanto mayor sea este nivel de confianza, mayor será la relación del identificador de usuario con esta clave. Este campo es calculado por PGP. También asociadas con la entrada hay varias firmas, o ninguna, recopiladas por el propietario del fichero de claves y que firman este certificado. A su vez, un **campo de confianza en la firma** se asocia con cada firma. Este campo indica hasta qué punto este usuario de PGP confía en el firmante para certificar claves públicas. El campo de legitimidad de la clave se deriva del grupo de campos de confianza en la firma en la entrada. Por último, cada entrada define una clave pública asociada con un propietario particular, y se incluye un **campo de confianza en el dueño** que indica hasta qué punto se confía en esta clave pública para firmar otros certificados de clave pública; este nivel de confianza lo asigna el usuario. Podemos considerar los campos de confianza en la firma como copias ocultas del campo de confianza en el dueño de otra entrada.

Los tres campos mencionados están incluidos en una estructura conocida como *byte indicador de confianza*. En la Tabla 5.2 se muestra el contenido del indicador de confianza para cada uno de estos tres usos. Supongamos que se trata del fichero de claves públicas del usuario A. Podemos describir la operación del procesamiento de confianza como sigue:

1. Cuando A inserta una nueva clave pública en el fichero de claves públicas, PGP debe asignar un valor al indicador de confianza que está asociado con el dueño de esta clave pública. Si el propietario es A y, por lo tanto, esta clave pública aparece también en el fichero de claves privadas, entonces se asigna un valor de *confianza definitiva (ultimate trust)* al campo de confianza. Si no, PGP pide a A su valoración de la confianza que ha de asignarse al propietario de esta clave, y A debe introducir el nivel deseado. El usuario puede especificar que este usuario es desconocido, no fiable, ligeramente fiable o absolutamente fiable.
2. Cuando se introduce la nueva clave pública, se pueden unir a ella una o varias firmas. Posteriormente se pueden añadir más firmas. Cuando se inserta una firma en la entrada, PGP busca el fichero de claves públicas para comprobar si el autor de esa firma se encuentra entre los propietarios conocidos. Si es así, se asigna el valor OWNERTRUST (confianza en el propietario) para ese propietario al campo SIGTRUST (confianza en la firma) para esa firma. De no ser así, se asigna el valor *usuario desconocido (unknown user)*.

Tabla 5.2 Contenidos del byte indicador de confianza

(a) Confianza asignada al dueño de una clave pública (aparece después del paquete de claves; definido por el usuario)	(b) Confianza asignada a la pareja de clave pública/ID de usuario (aparece después del paquete ID de usuario; calculado por PGP)	(c) Confianza asignada a la firma (aparece después del paquete de la firma; copia oculta del OWNERTRUST para este firmante)
<p>Campo OWNERTRUST</p> <ul style="list-style-type: none"> – confianza indefinida – usuario desconocido – normalmente no fiable para firmar otras claves – normalmente fiable para firmar otras claves – siempre fiable para firmar otras claves – esta clave está en el fichero de claves secretas (confianza definitiva) <p>Bit BUCKSTOP</p> <ul style="list-style-type: none"> – se fija si la clave aparece en el fichero de claves secretas 	<p>Campo KEYLEGIT</p> <ul style="list-style-type: none"> – desconocido o confianza indefinida – propiedad de la clave no fiable – confianza incierta en la propiedad de la clave – confianza absoluta en la propiedad de la clave <p>Bit WARNONLY</p> <ul style="list-style-type: none"> – se fija si el usuario quiere ser avisado sólo cuando una clave que no está totalmente validada se usa para cifrar 	<p>Campo SIGTRUST</p> <ul style="list-style-type: none"> – confianza indefinida – usuario desconocido – normalmente no fiable para firmar otras claves – normalmente fiable para firmar otras claves – siempre fiable para firmar otras claves <p>Bit CONTIG</p> <ul style="list-style-type: none"> – se fija si la firma lleva a una ruta contigua de certificación fiable hasta el propietario último del fichero de claves confiable

3. El valor del campo de legitimidad de la clave se calcula partiendo de los campos de confianza en la firma presentes en esa entrada. Si al menos una firma tiene un valor de confianza en la firma de *definitivo*, entonces el valor de legitimidad de la clave se fija en absoluto. Si no, PGP calcula una suma ponderada de los valores de confianza. A las firmas que son siempre fiables se les da un peso de $1/X$, y a las que lo son con frecuencia, $1/Y$, donde X e Y son parámetros configurables por el usuario. Cuando el total de los pesos de los que introducen una combinación de clave/identificador de usuario alcanza 1, la relación se considera digna de confianza, y el valor de legitimidad de la clave se fija en absoluto. Así, en ausencia de confianza absoluta, se necesitan al menos X firmas siempre fiables, o Y firmas frecuentemente fiables o una combinación de ellas.

Periódicamente, PGP procesa el fichero de claves públicas para alcanzar consistencia. Básicamente, este es un proceso de descenso. Para cada campo OWNERTRUST, PGP busca en el fichero todas las firmas creadas por un propietario y actualiza el campo SIG-TRUST para igualarlo al campo OWNERTRUST. Este proceso empieza con claves para las cuales hay confianza definitiva. Entonces, todos los campos KEYLEGIT se calculan basándose en las firmas adjuntas.

La Figura 5.7 proporciona un ejemplo de la forma en que se relacionan la confianza en la firma y la legitimidad de la clave². La figura muestra la estructura de un fichero de claves públicas. El usuario ha adquirido una serie de claves públicas, algunas directamente de sus propietarios y otras de una tercera parte como, por ejemplo, un servidor de claves.

El nodo con la etiqueta «You» se refiere a la entrada en el fichero de claves públicas correspondiente a ese usuario. Esta clave es legítima y el valor OWNERTRUST es confianza definitiva. Cada uno de los demás nodos del fichero de claves tiene un valor OWNERTRUST de indefinido a menos que el usuario asigne otro valor. En este ejemplo, este usuario ha especificado que siempre confía en los siguientes usuarios para firmar otras claves: D, E, F, L, y que confía parcialmente en los usuarios A y B para firmar otras claves.

Por lo tanto, el sombreado, o la ausencia de sombreado, de los nodos de la Figura 5.7 indican el nivel de confianza asignado por este usuario. La estructura de árbol indica qué claves han sido firmadas y qué usuarios las han firmado. Si una clave está firmada por un usuario cuya clave también se encuentra en este fichero de claves, la flecha va de la clave firmada al firmante. Si la clave está firmada por un usuario cuya clave no se encuentra en el fichero de claves, la flecha une la clave firmada a un signo de interrogación, indicando que el firmante es desconocido para el usuario.

En la Figura 5.7 se ilustran varios aspectos:

1. Obsérvese que todas las claves cuyos propietarios son total o parcialmente fiables para el usuario han sido firmadas por ese usuario, a excepción del nodo L. La firma del usuario no siempre es necesaria, como indica la presencia del nodo L, pero en la práctica, la mayoría de los usuarios tienden a firmar las claves para la mayoría de los propietarios en los que confían. Así, por ejemplo, aunque la clave de E ya está firmada por un presentador confiable F, el usuario decide firmar la clave de E directamente.
2. Asumimos que dos firmas parcialmente confiables son suficientes para certificar una clave. Por consiguiente, PGP considera legítima la clave para el usuario H porque está firmada por A y B, que son parcialmente fiables.
3. Se puede determinar que una clave es legítima porque está firmada por un firmante absolutamente fiable o por dos parcialmente fiables, pero su usuario puede no ser fiable para firmar otras claves. Por ejemplo, la clave de N es legítima porque está firmada por E, en quien confía este usuario, pero N no es fiable para firmar otras claves porque este usuario no ha asignado a N ese valor de confianza. Por lo tanto, aunque la clave de R está firmada por N, PGP no considera legítima la clave de R. Esta situación tiene sentido. Si se desea enviar un mensaje privado a un individuo, no es necesario confiar en él en ningún sentido. Sólo es necesario estar seguro de tener la clave pública correcta para ese individuo.

² Figura ofrecida al autor por Phil Zimmermann.

- 4 La Figura 5.7 también muestra un ejemplo de un nodo S separado «huérfano», con dos firmas desconocidas. Esta clave puede haber sido adquirida de un servidor de claves. PGP no puede asumir que esa clave es legítima simplemente porque proviene de un servidor de confianza. El usuario debe declarar que la clave es legítima firmándola o diciendo a PGP que está dispuesto a confiar por completo en uno de los firmantes de la clave.

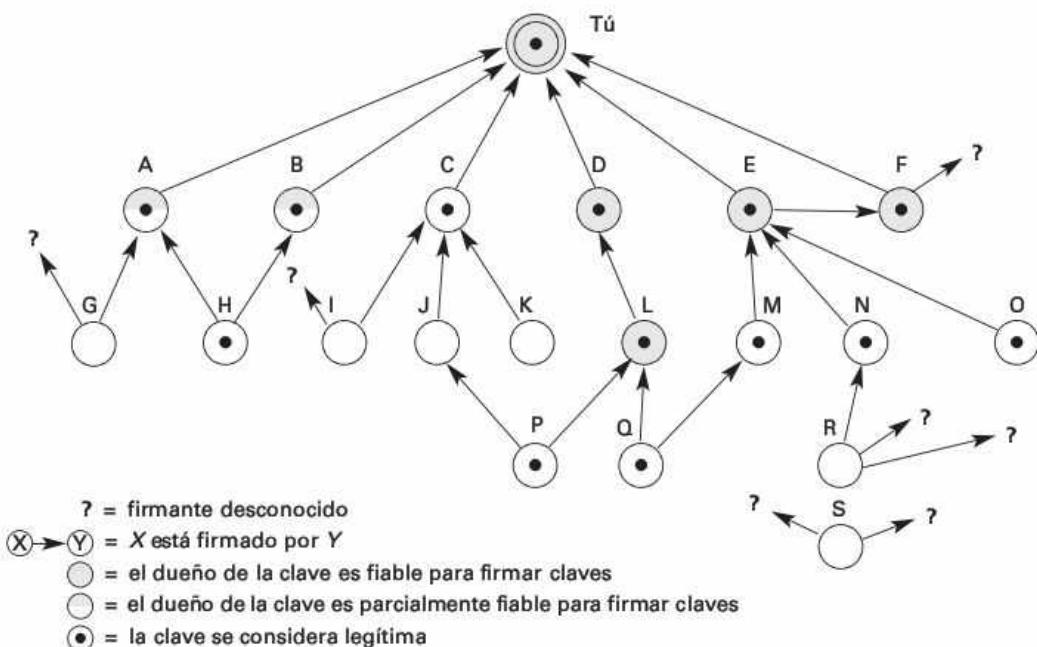


Figura 5.7 Ejemplo del modelo de confianza de PGP

Por último, recordemos que anteriormente se mencionó que varios identificadores de usuario podían asociarse con una única clave pública del fichero de claves públicas. Esto podría ser porque una persona ha cambiado los nombres o ha sido introducida mediante firma con múltiples nombres, indicando diferentes direcciones de correo electrónico para la misma persona, por ejemplo. Por lo tanto podemos imaginar una clave pública como la raíz de un árbol. Una clave pública tiene asociada una serie de identificadores de usuarios, con una serie de firmas para cada identificador de usuario. La vinculación de un identificador de usuario particular a una clave depende de las firmas asociadas con él y esa clave, mientras que el nivel de confianza en esta clave (para firmar otras claves) es una función de todas las firmas dependientes.

REVOCACIÓN DE CLAVES PÚBLICAS

Un usuario podría querer revocar su clave pública actual porque sospecha que existe un riesgo o simplemente para evitar el uso de la misma clave durante un período largo de

tiempo. Obsérvese que un compromiso requeriría que un oponente, de alguna forma, haya obtenido una copia de la clave privada de alguien sin cifrar o que el oponente haya obtenido la clave privada de su fichero de claves privadas y su frase clave.

La convención para revocar una clave pública es que el propietario emita un certificado de revocación de clave, firmado por él. Este certificado tiene la misma forma que un certificado de firma normal pero incluye un indicador que señala que el propósito de este certificado es el de revocar el uso de esta clave pública. Obsérvese que la clave privada correspondiente debe usarse para firmar un certificado que revoca una clave pública. El dueño debería intentar difundir este certificado lo más amplia y rápidamente posible para permitir que los interlocutores potenciales actualicen sus ficheros de claves públicas.

Hay que tener en cuenta que un oponente que ha comprometido la clave privada de un propietario puede también emitir ese certificado. Sin embargo, esto denegaría al oponente y al dueño legítimo el uso de la clave pública, y por lo tanto parece una amenaza menos probable que el uso fraudulento de una clave privada robada.

5.2 S/MIME

S/MIME (*Secure/Multipurpose Internet Mail Extension*) supone una mejora a la seguridad del formato estándar de correo electrónico de Internet de MIME, basado en tecnología de RSA Data Security. Aunque PGP y S/MIME son especificaciones de la IETF, parece probable que S/MIME se convierta en el estándar industrial para uso comercial y empresarial, mientras PGP se mantendrá como una opción para la seguridad personal del correo electrónico para muchos usuarios. S/MIME se define en una serie de documentos, especialmente en los RFC 2630, 2632 y 2633.

Para entender S/MIME, es necesario primero tener un conocimiento general de MIME, que es el formato de correo electrónico subyacente que utiliza. Pero para comprender la importancia de MIME, debemos retroceder al estándar tradicional del formato de correo electrónico, RFC 822, que aún es de uso común. Por ello, esta sección ofrece una introducción a estos estándares iniciales y analiza el S/MIME.

RFC 822

El RFC 822 define un formato para los mensajes de texto que se envían por medio de correo electrónico. Ha sido el estándar para los mensajes de texto de correo electrónico basado en Internet y sigue siendo de uso común. En el contexto del RFC 822, se considera que los mensajes se componen de un sobre (*envelope*) y unos contenidos (*content*). El sobre contiene la información necesaria para llevar a cabo la transmisión y la distribución. Los contenidos constituyen el objeto que se va a enviar al receptor. El estándar RFC 822 se aplica únicamente a los contenidos. Sin embargo, el contenido estándar incluye un conjunto de campos de cabecera que el sistema de correo electrónico puede usar para crear el sobre, y el estándar está diseñado para facilitar la adquisición de dicha información por parte de los programas.

La estructura general de un mensaje que se ajuste al RFC 822 es muy sencilla. Un mensaje está formado por un número determinado de líneas de cabecera (*the header*)

seguido de un texto arbitrario (*the body*). La cabecera se separa del cuerpo del texto mediante una línea en blanco. Es decir, un mensaje es texto ASCII, y todas las líneas hasta llegar a la primera línea en blanco forman la cabecera que usa el agente de usuario del sistema de correo electrónico.

Una línea de cabecera consta normalmente de una palabra clave, seguida de dos puntos, seguidos, a su vez, del argumento de la palabra clave; el formato permite dividir una línea muy larga en varias líneas. Las palabras clave más empleadas son *De* (*From*), *A* (*To*), *Asunto* (*Subject*) y *Fecha* (*Date*). A continuación se presenta un ejemplo de mensaje:

```
Date: Tue, 16 Jan 1998      10:37:17 (EST)
From: <William stallings> <ws@shore.net>
Subject: La sintaxis del RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com
```

Hola. Esta sección es el comienzo del cuerpo de texto, que está separado de la cabecera del mensaje por una línea en blanco.

Otro campo que se encuentra habitualmente en las cabeceras RFC 822 es el *Identificador de mensaje* (*Message-ID*). Este campo contiene un identificador único asociado con este mensaje.

MIME (MULTIPURPOSE INTERNET MAIL EXTENSIONS)

MIME es una extensión del marco de trabajo del RFC 822 diseñado para afrontar algunos de los problemas y limitaciones del uso del SMTP (*Simple Mail Transfer Protocol*) u otros protocolos de transferencia de correo y RFC 822 para correo electrónico. [MURH98] presenta las siguientes limitaciones del esquema del SMTP/822:

1. El SMTP no puede transmitir ficheros ejecutables ni otros objetos binarios. Se utilizan una serie de esquemas para convertir ficheros binarios a formato texto que puedan usar los sistemas de correo de SMTP, incluido el conocido esquema de UNIX UUencode/UUdecode. Sin embargo, ninguno de ellos es un estándar.
2. El SMTP no puede transmitir datos de texto que incluyan caracteres de un idioma nacional porque éstos se representan por códigos de ocho bits con valores de 128 decimal o superiores, y SMTP se limita a caracteres ASCII de siete bits.
3. Los servidores SMTP pueden rechazar mensajes que superen una determinada extensión.
4. Las pasarelas SMTP que traducen de ASCII a código EBCDIC no usan correspondencias consistentes, lo que da lugar a problemas de traducción.
5. Las pasarelas SMTP a redes de correo electrónico X.400 no pueden manejar datos que no sean de texto incluidos en mensajes X.400.
6. Algunas implementaciones SMTP no se adhieren por completo a los estándares SMTP definidos en el RFC 821. Los problemas habituales incluyen:
 - Eliminación, incorporación o reordenamiento de caracteres de retorno de carro y de salto de línea.

- Truncamiento o división de líneas de más de 76 caracteres.
- Eliminación de espacios finales en blanco (caracteres tabuladores y de espacio).
- Relleno de líneas de un mensaje para que tengan la misma longitud.
- Conversión de caracteres tabuladores a múltiples caracteres de espacio.

MIME tiene la finalidad de resolver estos problemas de forma que resulte compatible con las implementaciones existentes del RFC 822. La especificación se encuentra en los RFC 2045 al 2049.

DESCRIPCIÓN GENERAL

La especificación MIME incluye los siguientes elementos:

1. Se definen cinco nuevos campos de cabecera del mensaje, que se pueden incluir en una cabecera RFC 822. Estos campos aportan información sobre el cuerpo del mensaje.
2. Se define una serie de formatos de contenido, estandarizando así las representaciones que dan soporte al correo electrónico multimedia.
3. Se definen esquemas de codificación de transferencia que permiten la conversión de cualquier formato de contenido a una forma protegida contra alteraciones por el sistema de correo.

En este subapartado se introducen cinco campos de cabecera de mensaje. En los siguientes dos subapartados se analizan los formatos de contenido y los esquemas de codificación de transferencia.

Los cinco campos de cabecera definidos en MIME son los siguientes:

- **Versión MIME:** debe tener el valor de parámetro 1.0. Este campo indica que el mensaje se ajusta a los RFC 2045 y 2046.
- **Tipo de contenido:** describe los datos que contiene el cuerpo con suficiente detalle para que el usuario receptor pueda elegir un agente o mecanismo apropiado para representar los datos al usuario o, si no, manejar los datos de forma apropiada.
- **Codificación de transferencia de contenido:** indica el tipo de transformación que se ha utilizado para representar el cuerpo del mensaje de forma aceptable para el transporte de correo.
- **Identificación de contenido:** utilizada para identificar entidades MIME de forma única, en contextos múltiples.
- **Descripción del contenido:** una descripción en texto del objeto del cuerpo; es útil cuando el objeto no es legible (por ejemplo, datos de audio).

Cualquiera de estos campos puede aparecer en una cabecera normal de RFC 822. Una correcta implementación debe permitir los campos Versión MIME, Tipo de contenido y Codificación de transferencia de contenido; los campos Identificación de contenido y Descripción de contenido son opcionales y pueden ser ignorados por la implementación receptora.

TIPOS DE CONTENIDO MIME

El grueso de la especificación MIME trata de la definición de una serie de tipos de contenido. Esto refleja la necesidad de proporcionar formas estandarizadas de tratar una gran variedad de representaciones de la información en un entorno multimedia.

La Tabla 5.3 presenta un listado de los tipos de contenido especificados en el RFC 2046. Hay siete tipos principales y un total de 15 subtipos. En general, un tipo de contenido declara el tipo general de datos, y el subtipo especifica un formato particular para ese tipo de datos.

Tabla 5.3 Tipos de contenido de MIME

Tipo	Subtipo	Descripción
Texto	Claro	Texto sin formatear; puede ser ASCII o ISO 8859.
	Enriquecido	Proporciona una mayor flexibilidad de formato.
Multiparte <i>(multipart)</i>	Mezclado <i>(mixed)</i>	Las diferentes partes son independientes pero se van a transmitir juntas. Deberían presentarse al receptor en el orden en que aparecen en el mensaje de correo.
	Paralelo <i>(parallel)</i>	Se diferencia del Mezclado únicamente en que no se define ningún orden para el envío de las partes al receptor.
	Alternativo <i>(alternative)</i>	Las distintas partes son versiones alternativas de la misma información. Están ordenadas en exactitud creciente al original, y el sistema de correo receptor debería mostrar la «mejor» versión al usuario.
	Resumen <i>(digest)</i>	Similar a Mezclado, pero el tipo/subtipo de cada parte por defecto es message/rfc822.
Mensaje	rfc822	El cuerpo es por sí mismo un mensaje encapsulado que se ajusta al RFC 822.
	Parcial	Se usa para permitir la fragmentación de elementos de correo muy largos, de forma que sea transparente al receptor.
	Cuerpo externo	Contiene un enlace a un objeto que existe en alguna otra parte.
Imagen	jpeg	La imagen está en formato JPEG, codificación JFIF.
	gif	La imagen está en formato GIF.
Vídeo	mpeg	Formato MPEG.
Audio	Básico	Codificación en ley mu de un canal único ISDN de 8 bits a 8 kHz.
Aplicación	PostScript	Adobe Postscript.
	Flujo de octetos	Datos binarios generales formados por bytes de 8 bits.

Para el **tipo texto** del cuerpo, no se requiere *software* especial para lograr el significado completo del texto, aparte de permitir el grupo de caracteres indicado. El subtipo fundamental es *texto claro*, que consiste simplemente en una ristra de caracteres ASCII o ISO 8859. El subtipo *enriquecido* permite una mayor flexibilidad en el formato.

El **tipo multipart** indica que el cuerpo contiene varias partes independientes. El campo de cabecera Tipo de Contenido incluye un parámetro, llamado *Límite (boundary)*, que define el delimitador entre las partes del cuerpo. Este límite no debería aparecer en ninguna de las partes del mensaje. Cada límite empieza en una nueva línea y está formado por dos guiones seguidos del valor del límite. El límite final, que indica el final de la última parte, también tiene un sufijo de dos guiones. En cada parte puede haber una cabecera común MIME opcional.

A continuación se muestra un ejemplo sencillo de un mensaje multipart, que contiene dos partes compuestas por texto simple (extraído de RFC 2046):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: mensaje de muestra
MIME-Version: 1.0
Content-type: multipart/mixed; boundary= «simple boundary»
```

Esto es el preámbulo. Ha de ser ignorado, aunque es un lugar útil para que los que escriben correo incluyan una nota explicativa para los lectores que no siguen MIME. --simple boundar y

Esto es implícitamente un texto escrito ASCII. No acaba con una ruptura de línea.--simple boundary

```
Content-type: text/plain; charset = us-ascii
```

Esto es explícitamente un texto escrito ASCII. Acaba con una ruptura de línea.

--simple boundary--

Este es el epílogo. También ha de ser ignorado.

Hay cuatro subtipos del tipo multipart, todos con la misma sintaxis general. El **subtipo multipart/mezclado** se usa cuando hay varias partes del cuerpo independientes que necesitan disponerse en un orden particular. Para el **subtipo multipart/paralelo**, el orden de las partes no es significativo. Si el sistema receptor es adecuado, las distintas partes pueden presentarse en paralelo. Por ejemplo, una parte de imagen o de texto podría ir acompañada de un comentario verbal que sonara cuando se mostraran el texto o la imagen.

Para el **subtipo multipart/alternativo**, las distintas partes son diferentes representaciones de la misma información. A continuación se ofrece un ejemplo:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: correo de texto formateado
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary = boundary42
--boundary42

Content-Type: text/plain; charset=us-ascii
```

... aquí va la versión de texto del mensaje ...

--boundary42

Content-Type: text/enriched

... aquí va la versión de texto enriquecido RFC 1896 del mismo mensaje...

--boundary42 --

En este subtipo, las partes del cuerpo se ordenan en términos de preferencia creciente. Para este ejemplo, si el sistema receptor es capaz de mostrar el mensaje en el formato texto enriquecido, lo hace; si no, se usa el formato de texto.

El **subtipo multipart/resumen** se usa cuando cada una de las partes del cuerpo se interpreta como un mensaje RFC 822 con cabeceras. Este subtipo permite la construcción de un mensaje cuyas partes son mensajes individuales. Por ejemplo, el moderador de un grupo podría recoger mensajes de correo electrónico de los participantes, agruparlos y enviarlos en un mensaje MIME encapsulado.

El **tipo mensaje** proporciona una serie de capacidades importantes en MIME. El **subtipo mensaje/rfc822** indica que el cuerpo es un mensaje completo, que incluye cabecera y cuerpo. A pesar del nombre de este subtipo, el mensaje encapsulado puede ser cualquier mensaje MIME, y no sólo un mensaje RFC 822.

El **subtipo mensaje/parcial** permite la fragmentación de un mensaje largo en una serie de partes, que deben volver a ensamblarse en el destino. Para este subtipo se especifican tres parámetros en el campo Tipo de Contenido: mensaje/parcial: un *identificador* común a todos los fragmentos del mismo mensaje, un *número de secuencia* único para cada fragmento y el número *total* de fragmentos.

El **subtipo mensaje/cuerpo externo** indica que los datos reales que se van a transportar en este mensaje no están contenidos en el cuerpo. En vez de ello, el cuerpo contiene la información necesaria para acceder a los datos. Como con otros tipos de mensaje, el subtipo mensaje/cuerpo externo tiene una cabecera externa y un mensaje encapsulado con su propia cabecera. El único campo necesario en la cabecera externa es el de Tipo de Contenido, que lo identifica como un subtipo mensaje/cuerpo externo. La cabecera interna es la cabecera del mensaje para el mensaje encapsulado. El campo Tipo de Contenido de la cabecera externa debe incluir un parámetro de tipo de acceso, que indica el método de acceso, como por ejemplo FTP (*File Transfer Protocol*).

El **tipo de aplicación** se refiere a otros tipos de datos, generalmente datos binarios sin interpretación, o información que va a procesarse mediante una aplicación basada en correo.

CODIFICACIÓN DE TRANSFERENCIA DE MIME

El otro componente principal de la especificación MIME, además de la especificación del tipo de contenido, es una definición de la codificación de transferencia para cuerpos de mensaje. El objetivo es proporcionar un envío seguro a través de la gama más amplia de entornos.

El estándar MIME define dos métodos de codificación de datos. El campo Codificación de Transferencia de Contenido en realidad puede tomar seis valores, como se muestra en la Tabla 5.4. Sin embargo, tres de estos valores (siete bits, ocho bits y binario)

indican que no se ha realizado la codificación pero proporcionan información sobre la naturaleza de los datos. Para la transferencia SMTP, es seguro usar la forma de siete bits. Las formas de ocho bits y binarias se pueden utilizar en otros contextos de transporte de correo. Otro valor de Codificación de Transferencia de Contenido es *x-token*, que indica que se usa algún otro esquema de codificación, para el cual se proporciona un nombre. Éste podría ser un esquema específico del vendedor o de la aplicación. Los dos esquemas de codificación reales definidos son *imprimible textualmente (quoted-printable)* y *base64*. Se definen dos esquemas para proporcionar una elección entre una técnica de transferencia esencialmente legible para las personas y otra que es segura para todos los tipos de datos en una forma razonablemente compacta.

Tabla 5.4 Codificación de transferencia de MIME

7 bit	Todos los datos se representan mediante líneas cortas de caracteres ASCII.
8 bit	Las líneas son cortas, pero podría haber caracteres que no fueran ASCII (octetos con el bit de orden más alto).
binario	Además de que pueden estar presentes caracteres que no sean ASCII, las líneas no son suficientemente cortas para el transporte SMTP.
Imprimible textualmente	Codifica los datos de forma que si los datos que se están codificando son en su mayoría texto ASCII, la forma codificada de los datos es fácilmente reconocible por los usuarios humanos.
base64	Codifica los datos convirtiendo los bloques de entrada de 6 bits en bloques de salida de 8 bits, todos ellos caracteres ASCII imprimibles.
<i>x-token</i>	Una codificación no estándar.

La codificación de transferencia **imprimible textualmente** es útil cuando los datos son en su mayoría octetos que corresponden a caracteres ASCII imprimibles. Básicamente, representa caracteres inseguros por medio de la representación hexadecimal de su código e introduce rupturas de líneas reversibles (suaves) para limitar las líneas del mensaje a 76 caracteres.

La codificación de transferencia **base64**, también conocida como codificación radix 64, es común para codificar datos binarios arbitrarios para que sean invulnerables al procesamiento que llevan a cabo programas de transporte de correo. También se usa en PGP y se describe en el Apéndice 5B.

UN EJEMPLO MULTIPARTE

La Figura 5.8, extraída del RFC 2045, es el esquema de un mensaje multiparte complejo. El mensaje tiene cinco partes que se muestran en serie: dos partes introductorias de texto claro, un mensaje multipart insertado, una parte de texto enriquecido y un mensaje de cierre de texto encapsulado en caracteres no ASCII. El mensaje multiparte insertado tiene dos partes que se muestran en paralelo, una imagen y un fragmento de audio.

FORMA CANÓNICA

Un concepto importante en MIME y S/MIME es el de forma canónica. Se trata de un formato, apropiado al tipo de contenido, que está normalizado para usarse entre distintos sistemas. Es la opuesta a la forma nativa, que es un formato concreto para un sistema particular. La Tabla 5.5, del RFC 2049, ayuda a clarificar esta cuestión.

MIME-Version: 1.0

From: Nathaniel Borenstein <nsb@bellcore.com>

To: Ned Freed <ned@innosoft.com>

Subject: un ejemplo multiparte

Content-Type: multipart/mixed;

boundary = unique-boundary-1

Esto es el preámbulo de un mensaje multiparte. Los lectores de correo que entienden el formato multiparte deberían ignorar este preámbulo. Si está leyendo este texto, podría querer cambiar a un lector de correo que entienda cómo mostrar adecuadamente mensajes multiparte.

-unique boundary-1

..aquí aparece texto...

[Obsérvese que la línea en blanco anterior significa que no se dieron campos de cabecera y que esto es texto, con caracteres ASCII. Podría haberse hecho escribiendo explícitamente, como en la siguiente parte.]

-unique boundary-1

Content-type: text/plain; charset=US-ASCII

Esto podría haber sido parte de la parte anterior, pero muestra la escritura explícita de las partes del cuerpo (en contraposición a la implícita).

-unique-boundary-1

Content-Type: multipart/parallel; boundary=unique-boundary-2

-unique-boundary-2

Content-Type: audio/basic

Content-Transfer-Encoding:base64

..aquí van los datos de audio en ley mu de un canal único codificado en base64 a 8000 Hz

-unique-boundary-2

Content-Type: image/jpeg

Content-Transfer-Encoding:base64

... aquí van los datos de imagen codificados en base64

-unique-boundary-2

-unique-boundary-1

Content-type: text/enriched

Esto es <bold><italic>texto enriquecido. </italic></bold><smaller> definido en RFC 1896 </smaller>

¿No es <bigger><bigger>fantástico? </bigger></bigger>

-unique-boundary-1

Content-Type: message/rfc822

From: (buzón en US-ASCII)

To: (dirección en US-ASCII)

Subject: (asunto en US-ASCII)

Content-Type: Text/plain; charset=ISO-8859-1

Content-Transfer-Encoding: imprimible textualmente

..aquí va el texto adicional en ISO-8859-1...

-unique-boundary-1--

Figura 5.8 Ejemplo de estructura de mensaje MIME

Tabla 5.5 Forma nativa y forma canónica

Forma nativa	El cuerpo que se va a transmitir se crea en el formato nativo del sistema. Se usa el conjunto de caracteres nativos y, donde sea adecuado, se usan también convenciones locales de final de línea. El cuerpo puede ser un archivo de texto del estilo de UNIX, o una imagen en <i>Sun raster</i> , o un archivo indexado VMS, o datos de audio en un formato dependiente del sistema almacenados sólo en memoria, o cualquier otra cosa que corresponda al modelo local para la representación de alguna forma de información. Básicamente, los datos se crean en la forma «nativa» que corresponde al tipo especificado por el tipo de medio.
Forma canónica	El cuerpo entero, incluida la información «fuera de banda», como por ejemplo la longitud de los registros y la posible información sobre atributos de ficheros, se convierte a forma canónica universal. El tipo de medio específico del cuerpo y sus atributos asociados dictan la naturaleza de la forma canónica que se usa. La conversión a la forma canónica adecuada puede implicar conversión del conjunto de caracteres, transformación de datos de audio, compresión u otras operaciones específicas a los distintos tipos de medios. Si tiene lugar la conversión del grupo de caracteres, sin embargo, se debe tener cuidado para entender la semántica del tipo de medio, que puede tener serias implicaciones para la conversión de cualquier grupo de caracteres (por ejemplo, con respecto a los caracteres sintácticamente significativos en un subtipo de texto que no sea «sólo texto»).

FUNCIONALIDAD S/MIME

En términos de funcionalidad general, S/MIME es muy similar a PGP. Los dos ofrecen la posibilidad de firmar y/o cifrar mensajes. En este subapartado, se resume brevemente la capacidad de S/MIME. Luego se analizará más detalladamente esta capacidad examinando los formatos de mensaje y la preparación de mensajes.

FUNCIONES

S/MIME proporciona las siguientes funciones:

- **Datos empaquetados (*enveloped data*):** consiste en contenido cifrado de cualquier tipo y claves de cifrado de contenido cifrado para uno o más receptores.
- **Datos firmados (*signed data*):** una firma digital se forma tomando el resumen de mensaje del contenido que se va a firmar y cifrándolo con la clave privada del firmante. El contenido más la firma se codifican luego usando codificación base64. Un mensaje de datos firmado sólo puede verlo un receptor con capacidad S/MIME.
- **Datos firmados en claro (*clear-signed data*):** al igual que con los datos firmados, se crea una firma digital del contenido. Sin embargo, en este caso, sólo se codifica la firma digital usando base64. Como resultado, los receptores sin capacidad S/MIME pueden ver el contenido del mensaje, aunque no pueden verificar la firma.

- **Datos firmados y empaquetados (*signed and enveloped data*):** se pueden anidar entidades sólo firmadas y sólo cifradas, para que los datos cifrados puedan ser firmados y que los datos firmados o en formato *clear-signed* puedan ser cifrados.

ALGORITMOS CRIPTOGRÁFICOS

La Tabla 5.6 resume los algoritmos criptográficos que se usan en S/MIME. S/MIME usa la siguiente terminología, extraída de RFC 2119 para especificar el nivel de requisitos:

Tabla 5.6 Algoritmos criptográficos usados en S/MIME

Función	Requisito
Crea un resumen de mensaje para crear una firma digital.	DEBE permitir SHA-1. El receptor DEBERÍA soportar MD5 para tener compatibilidad con versiones anteriores.
Cifra un resumen de mensaje para crear una firma digital.	Enviar y recibir agentes DEBE permitir DDS. Enviar agentes DEBERÍA permitir cifrado RSA. Recibir agentes DEBERÍA permitir verificación de firmas RSA con claves de 512 bits a 1024 bits.
Cifra la clave de sesión para su transmisión con el mensaje.	Enviar y recibir agentes DEBE soportar Diffie-Hellman. Enviar agente DEBERÍA permitir cifrado RSA con claves de 512 bits a 1024 bits. Recibir agente DEBERÍA permitir cifrado RSA.
Cifra el mensaje para su transmisión con la clave de sesión de un solo uso.	Enviar agentes DEBERÍA permitir cifrado con triple DES y RC2/40. Recibir agentes DEBE permitir descifrado usando triple DES y DEBERÍA permitir descifrado con RC2/40.

- **DEBE:** la definición es un requisito absoluto de la especificación. Una implementación debe incluir esta característica o función para ajustarse a la especificación.
- **DEBERÍA:** pueden existir razones válidas en circunstancias particulares para ignorar esta característica o función, pero se recomienda que una implementación incluya la característica o función.

S/MIME incorpora tres algoritmos de clave pública. El DSS (*Digital Signature Standard*), mencionado en el Capítulo 3, es el algoritmo preferido para firmas digitales.

S/MIME propone Diffie-Hellman como el algoritmo preferido para cifrar claves de sesión; de hecho, S/MIME usa una variante de Diffie-Hellman que proporciona cifrado/descifrado, conocida como ElGamal. Como alternativa, RSA, descrito en el Capítulo 3, se puede usar tanto para cifrar firmas como claves de sesión. Son los mismos algoritmos que se usan en PGP y aportan una mayor seguridad. Para la función *hash* utilizada para crear la firma digital, la especificación requiere SHA-1 de 160 bits, pero recomienda que el receptor admita MD5 de 128 bits para lograr compatibilidad con versiones anteriores de S/MIME. Como se exponía en el Capítulo 3, hay una preocupación justificada por la seguridad de MD5, por lo que SHA-1 es la alternativa preferida.

Para el cifrado de mensajes, se recomienda el tripleDES de tres claves (tripleDES), pero las implementaciones adecuadas deben dar soporte a RC2 de 40 bits. El último es un algoritmo débil de cifrado pero se ajusta a los controles de exportación en Estados Unidos.

La especificación S/MIME incluye una discusión del procedimiento para decidir qué algoritmo de cifrado de contenido usar. Básicamente, un agente emisor puede tomar dos decisiones. En primer lugar, el emisor debe determinar si el receptor es capaz de descifrar usando un algoritmo de cifrado determinado. Segundo, si el receptor sólo es capaz de aceptar contenido que se ha cifrado mediante un algoritmo débil, el emisor debe decidir si es aceptable enviar usando un cifrado débil. Para apoyar este proceso de decisión, un emisor puede anunciar sus capacidades de descifrado en orden de preferencia para cualquier mensaje que envíe. Un receptor puede almacenar esa información para usarla en el futuro.

El agente emisor debería seguir las siguientes reglas, en el siguiente orden:

- 1.** Si el emisor tiene una lista de capacidades de descifrado preferentes de un receptor determinado, DEBERÍA elegir la primera capacidad (más alto grado de preferencia) de la lista que pueda utilizar.
- 2.** Si el emisor no dispone de dicha lista pero ha recibido uno o más mensajes del receptor, entonces el mensaje saliente DEBERÍA usar el mismo algoritmo de cifrado que se usó en el último mensaje cifrado y firmado que recibió de dicho receptor.
- 3.** Si el emisor no tiene conocimiento de las capacidades de descifrado del receptor y está dispuesto a arriesgarse a que éste no pueda descifrar el mensaje, entonces el emisor DEBERÍA usar tripleDES.
- 4.** Si el emisor no tiene conocimiento de las capacidades de descifrado del receptor y no está dispuesto a arriesgarse a que éste no pueda descifrar el mensaje, entonces el agente emisor DEBE usar RC2/40.

Si un mensaje debe enviarse a múltiples receptores y no se puede seleccionar un algoritmo de cifrado común para todos, entonces el emisor necesitará enviar dos mensajes. Sin embargo, en ese caso, es importante observar que la seguridad del mensaje se hace vulnerable por la transmisión de una copia con menor seguridad.

MENSAJES S/MIME

S/MIME usa una serie de nuevos tipos de contenido MIME, que se muestran en la Tabla 5.7. Todos los nuevos tipos de aplicaciones usan la designación PKCS, que hace

referencia a un grupo de especificaciones criptográficas de clave pública emitidas por RSA Laboratories y que están disponibles para la mejora de S/MIME.

Tabla 5.7 Tipos de contenido S/MIME

Tipo	Subtipo	Parámetro smime	Descripción
Multiparte	Firmado		Un mensaje firmado en claro en dos partes: una es el mensaje y la otra la firma.
Aplicación	Pkcs7-mime	SignedData	Una entidad S/MIME firmada.
	Pkcs7-mime	EnvelopedData	Una entidad S/MIME cifrada.
	Pkcs7-mime	degenerate signed Data	Una entidad que contiene sólo certificados de clave pública.
	Pkcs7-firma	—	El tipo de contenido de la subparte de firma de un mensaje multiparte firmado.
	Pkcs10-mime	—	Un mensaje de solicitud de registro de certificado.

Después de examinar los procedimientos generales para la preparación de mensajes S/MIME, se analizará cada uno de ellos.

Asegurar una entidad MIME

S/MIME asegura una entidad MIME con una firma, con cifrado o con ambos. Una entidad MIME puede ser un mensaje completo (excepto las cabeceras RFC 822), o si el tipo de contenido MIME es multiparte, entonces una entidad MIME es una o más supartes del mensaje. La entidad MIME se prepara de acuerdo con las reglas normales para la preparación de mensajes MIME. Entonces la entidad MIME más algunos datos relacionados con la seguridad, como identificadores de algoritmos y certificados, son procesados por MIME para producir lo que se conoce como objeto PKCS. Luego, un objeto PKCS se trata como contenido de mensaje y se incluye en MIME (está provisto de las cabeceras MIME apropiadas). Este proceso debería aclararse a medida que observemos objetos específicos y algunos ejemplos.

En todos los casos, el mensaje que se va a enviar se convierte a forma canónica. En particular, para un tipo y un subtipo dados, se usa la forma canónica adecuada para el contenido del mensaje. En un mensaje multiparte, para cada subparte se usa la forma canónica adecuada.

El uso de codificación de transferencia requiere especial atención. Para la mayoría de los casos, el resultado de aplicar el algoritmo de seguridad será producir un objeto representado parcial o totalmente en datos binarios arbitrarios. Éste luego se agrupará en un mensaje MIME externo y en ese punto se puede aplicar la codificación de transferencia, normalmente base64. Sin embargo, en el caso de un mensaje firmado multiparte, que se describirá más detalladamente luego, el contenido de mensaje en una de las subpartes no es modificado por el proceso de seguridad. A menos que el contenido sea

de siete bits, la transferencia debería codificarse usando base64 o imprimible textualmente, para que no haya peligro de alteración del contenido al que se aplicó la firma.

A continuación observemos cada uno de los tipos de contenido S/MIME.

Datos empaquetados (*EnvelopedData*)

Un subtipo application/pkcs7-mime se usa para una de las cuatro categorías del procesamiento S/MIME, cada una con un parámetro único de tipo smime. En todos los casos, la entidad resultante, a la que se conoce como *objeto*, se representa en una forma conocida como BER (*Basic Encoding Rules*), que se define en la Recomendación X.209 de la ITU-T. El formato BER está formado por ristras de octetos arbitrarios y consiste, por lo tanto, en datos binarios. La transferencia de dicho objeto debería ser codificada con base64 en el mensaje MIME externo. Empecemos con *envelopedData*.

Los pasos para la preparación de una entidad MIME envelopedData son los siguientes:

- 1.** Generar una clave de sesión pseudoaleatoria para un algoritmo de cifrado simétrico particular (RC2/40 o triple DES).
- 2.** Para cada receptor, cifrar la clave de sesión con la clave pública RSA del receptor.
- 3.** Para cada receptor, preparar un bloque conocido como *RecipientInfo* (información sobre el receptor) que contiene un identificador del certificado de clave pública del receptor³, un identificador del algoritmo utilizado para cifrar la clave de sesión y la clave de sesión cifrada.
- 4.** Cifrar el contenido de mensaje con la clave de sesión.

Los bloques RecipientInfo seguidos del contenido cifrado constituyen los envelopedData. Esta información luego se codifica en base64. El siguiente es un ejemplo de mensaje (excluyendo las cabeceras RFC 822):

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGGhQbnj756YT64V
```

Para recuperar el mensaje cifrado, el receptor, en primer lugar, elimina la codificación base64. Luego, se usa la clave privada del receptor para recuperar la clave de sesión. Por último, el contenido de mensaje se cifra con la clave de sesión.

Datos firmados (*SignedData*)

El tipo smime signedData puede usarse con uno o más firmantes. Para aportar mayor claridad, aplicamos nuestra descripción al caso de una única firma digital. Los pasos para preparar una entidad MIME signedData son los siguientes:

³ Este es un certificado X.509, que se tratará más adelante en esta sección.

1. Seleccionar un algoritmo de resumen de mensaje (SHA o MD5).
2. Calcular el resumen del mensaje, o función *hash*, del contenido que se va a firmar.
3. Cifrar el resumen de mensaje con la clave privada del firmante.
4. Preparar un bloque conocido como SignerInfo (información sobre el firmante) que contenga el certificado de clave pública del firmante, un identificador del algoritmo de resumen del mensaje, un identificador del algoritmo usado para cifrar el resumen el mensaje y el resumen de mensaje cifrado.

La entidad signedData se compone de una serie de bloques, incluyendo un identificador del algoritmo de resumen de mensaje, el mensaje que se está firmando y SignerInfo. La entidad signedData también puede incluir un conjunto de certificados de clave pública suficiente para constituir una cadena que va desde una autoridad de certificación raíz o de alto nivel, reconocida, hasta el firmante. Esta información luego se codifica en base64. El siguiente es un ejemplo de mensaje (excluyendo las cabeceras RFC 822):

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHdYT6ghyHhUUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBfhYHhUUujhJhjH
HUUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jh7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

Para recuperar el mensaje firmado y verificar la firma, el receptor elimina la codificación base64. Luego, la clave pública del firmante se usa para descifrar el resumen del mensaje. El receptor computa independientemente el resumen de mensaje y lo compara con el resumen del mensaje descifrado para verificar la firma.

Firma en claro (*clear signing*)

La firma en claro (*clear signing*) se consigue usando el tipo de contenido multiparte con un subtipo firmado. Como ya se ha mencionado, este proceso de firma no implica transformar el mensaje que se va a firmar, para que el mensaje se envíe «en claro». Así, los receptores con capacidad MIME pero sin S/MIME pueden leer el mensaje entrante.

Un mensaje multiparte/firmado tiene dos partes. La primera parte puede ser cualquier tipo MIME pero debe prepararse para que no sea modificado durante la transferencia desde la fuente al destino. Esto significa que si la primera parte no es de siete bits, entonces necesita ser codificada usando base64 o imprimible textualmente. Luego, esta parte se procesa de la misma forma que signedData, pero en este caso se crea un objeto con signedData que tenga un campo de contenido de mensaje vacío. Este objeto es una firma separada. Luego la transferencia se codifica usando base64 para convertirse en la segunda parte del mensaje multiparte firmado. Esta segunda parte tiene un tipo de contenido MIME de aplicación y un subtipo de firma pkcs7. Este es un ejemplo de mensaje:

```

Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

Este es un mensaje firmado en claro.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfHfYT6
4VQpfyF467GhIGfHfYT6jh77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--boundary42--

```

El parámetro de protocolo indica que se trata de una entidad de dos partes *clear-signed*. El parámetro micalg indica el tipo de resumen de mensaje usado. El receptor puede verificar la firma tomando el resumen de mensaje de la primera parte y comparándolo con el resumen de mensaje recuperado de la firma en la segunda parte.

Solicitud de registro

Normalmente, una aplicación o usuario solicitará un certificado de clave pública a una autoridad de certificación. La entidad S/MIME aplicación/pkcs10 se usa para transferir una solicitud de certificación. Dicha solicitud incluye un bloque de información de solicitud de certificación (*certificationRequestInfo*), seguido de un identificador del algoritmo de cifrado de clave pública, seguido de la firma del bloque de información de solicitud de certificación, que se hace usando la clave privada del emisor. Dicho bloque incluye un nombre del sujeto del certificado (la entidad cuya clave pública se va a certificar) y una representación de ristra de bits de la clave pública del usuario.

Mensaje que contiene sólo certificados

Un mensaje que contenga sólo certificados o una lista de revocación de certificados (CRL: *Certificate Revocation List*) se puede enviar en respuesta a una solicitud de registro. El mensaje es un tipo/subtipo aplicación/pkcs7-mime con un parámetro del tipo smime de *degenerate*. Los pasos necesarios son los mismos que los que se usan para crear un mensaje signedData, a excepción de que no hay contenido de mensaje y que el campo signerInfo está vacío.

PROCESAMIENTO DE CERTIFICADOS S/MIME

S/MIME usa certificados de clave pública que se ajustan a la versión 3 de X.509 (ver Capítulo 4). El esquema de gestión de claves utilizado por S/MIME es un híbrido entre

una jerarquía estricta de certificación X.509 y la red de confianza de PGP. Como con el modelo PGP, los administradores y/o usuarios de S/MIME deben configurar cada cliente con una lista de claves fiables y con listas de revocación de certificados. Es decir, la responsabilidad es local para mantener los certificados necesarios para la verificación de las firmas entrantes y para cifrar los mensajes salientes. Por otra parte, los certificados los firman las autoridades de certificación.

El papel del agente usuario

Un usuario S/MIME tiene varias funciones de gestión de claves que realizar:

- **Generación de claves:** el usuario de alguna herramienta administrativa relacionada (por ejemplo, una asociada con la gestión de LAN) DEBE poder generar parejas separadas de claves Diffie-Hellman y DSS y DEBERÍA poder generar parejas de claves RSA. Cada pareja de claves DEBE ser generada a partir de una buena fuente de entrada aleatoria no determinística y ser protegida de forma segura. Un agente usuario DEBERÍA generar parejas de clave RSA con una longitud de entre 768 y 1024 bits y NO DEBE generar una longitud inferior a 512 bits.
- **Registro:** una clave pública de usuario debe registrarse con una autoridad de certificación para recibir un certificado de clave pública X.509.
- **Almacenamiento y recuperación de certificado:** un usuario requiere acceso a una lista local de certificados para verificar las firmas entrantes y para cifrar los mensajes salientes. Esta lista podría ser mantenida por el usuario o por alguna entidad administrativa local en nombre de una serie de usuarios.

Certificados VeriSign

Existen varias compañías que proporcionan servicios de autoridad de certificación (CA). Por ejemplo, Nortel ha diseñado una solución de empresa CA y puede proporcionar soporte S/MIME en una organización. Hay una serie de CA basadas en Internet, entre las que se incluyen VeriSign, GTE y el servicio postal de Estados Unidos. De estos, el más usado es el servicio CA de VeriSign, del que se hará una breve descripción.

VeriSign ofrece un servicio de CA diseñado para ser compatible con S/MIME y otras aplicaciones. Emite certificados X.509 con el nombre *VeriSign Digital ID*. A principios de 1998, más de 35.000 sitios web comerciales usaban los identificadores digitales del servidor VeriSign, y más de un millón de identificadores digitales habían sido emitidos a usuarios de Netscape y navegadores de Microsoft.

La información que contiene un identificador digital depende del tipo de identificador digital y su uso. Como mínimo, cada identificador contiene:

- Clave pública del dueño.
- Nombre o alias del dueño.
- Fecha de caducidad del identificador digital.
- Número de serie del identificador digital.
- Nombre de la autoridad de certificación que emitió el identificador digital.
- Firma digital de la autoridad de certificación que emitió el identificador digital.

Los identificadores digitales pueden contener también otra información suministrada por el usuario, como

- Dirección.
- Dirección de correo electrónico.
- Información básica de registro (país, código postal, edad y género).

Tabla 5.8 Clases de certificados de clave pública VeriSign

	Resumen de confirmación de identidad	Protección de clave privada de la autoridad emisora	Solicitante de certificado y protección de clave privada del suscriptor	Aplicaciones implementadas o contempladas por usuarios
Clase 1	Nombre automatizado sin ambigüedad y búsqueda de dirección de correo electrónico.	PCA: <i>hardware fiable</i> ; CA: <i>software fiable</i> o <i>hardware fiable</i> .	<i>Software de cifrado (protegido con PIN)</i> recomendado pero no necesario.	Navegador y cierto uso de correo electrónico.
Clase 2	Igual que la clase 1, más comprobación automatizada de la información de registro más comprobación automatizada de dirección.	PCA y CA: <i>hardware fiable</i> .	<i>Software de cifrado (protegido con PIN)</i> necesario.	Correo electrónico individual y entre empresas, suscripciones en línea, sustitución de contraseñas y validación de <i>software</i> .
Clase 3	Igual que la clase 1, más presencia personal y documentos de ID más la comprobación automatizada de ID para individuos de la clase 2; registros de empresas (o archivos) para organizaciones.	PCA y CA: <i>hardware fiable</i> .	<i>Software de cifrado (protegido con PIN)</i> necesario; dispositivo <i>hardware</i> de identificación recomendado pero no necesario.	Banca electrónica, acceso a bases de datos, banca personal, servicios en línea basados en miembros, servicios de integridad de contenidos, servidor de comercio electrónico, validación de <i>software</i> , autenticación de las LRAA; y cifrado robusto para ciertos servidores.

PCA: Autoridad de certificación pública primaria de VeriSign.

PIN: número de identificación personal.

LRAA: administrador local de autoridad de registro.

VeriSign proporciona tres niveles, o clases, de seguridad para certificados de clave pública, como se resume en la Tabla 5.8. Un usuario solicita un certificado en línea en el sitio web de VeriSign u otros sitios web participantes. Las solicitudes de clase 1 y clase 2 se procesan en línea, y en la mayoría de los casos tardan sólo unos segundos en aprobarse. Se usan los siguientes procedimientos:

- Para los identificadores digitales de la clase 1, VeriSign confirma la dirección de correo electrónico del usuario enviando un PIN e información del identificador digital a la dirección de correo electrónico suministrada en la aplicación.
- Para los identificadores digitales de la clase 2, VeriSign verifica la información en la aplicación mediante una comparación automática con una base de datos de consumidores, además de realizar toda la comprobación asociada a un identificador digital de clase 1. Por último, se envía la confirmación a la dirección postal específica avisando al usuario de que se ha emitido un identificador digital en su nombre.
- Para los identificadores digitales de la clase 3, VeriSign requiere un nivel más alto de seguridad de la entidad. Un individuo puede probar su identidad proporcionando credenciales notarizadas o solicitándolo en persona.

SERVICIOS DE SEGURIDAD AVANZADA

Se han propuesto tres servicios de seguridad como borrador de Internet. Pueden cambiar sus características y añadirse servicios adicionales. Los servicios son los siguientes:

- **Recibos firmados:** se puede solicitar un recibo firmado en un objeto SignedData. Devolver un recibo firmado proporciona una prueba de la entrega al creador de un mensaje y le permite demostrar a una tercera parte que el receptor recibió el mensaje. Básicamente, el receptor firma el mensaje original completo más la firma original (del emisor) y añade la nueva firma para formar un nuevo mensaje S/MIME.
- **Etiquetas de seguridad:** una etiqueta de seguridad se puede incluir en los atributos autenticados de un objeto SignedData. Una etiqueta de seguridad es información sobre seguridad con respecto a la confidencialidad del contenido protegido por el encapsulado de S/MIME. Las etiquetas pueden usarse para el control del acceso, indicando qué usuarios tienen permiso para acceder a un objeto. También se utilizan para indicar prioridad (secreta, confidencial, restringida, etc.) o para describir qué tipo de personas pueden ver la información (por ejemplo, equipo médico de un paciente, etc.).
- **Listas de correo seguras:** cuando un usuario envía un mensaje a varios receptores, es necesaria una cierta cantidad de procesamiento por cada receptor, incluyendo el uso de la clave pública de cada uno de ellos. El usuario puede liberarse de este trabajo empleando los servicios de un MLA (*Mail List Agent*) de S/MIME. Un MLA puede tomar un solo mensaje entrante, realizar el cifrado específico para cada receptor y reenviar el mensaje. El creador de un mensaje sólo necesita enviar el mensaje a la MLA, habiendo realizado el cifrado con la clave pública de la MLA.

5.3 SITIOS WEB RECOMENDADOS

Sitios web recomendados:

- **PGP Home Page:** sitio web PGP de Network Associates, el vendedor líder de PGP.
- **MIT Distribution Site for PGP:** distribuidor líder de PGP libre. Contiene FAQ, otra información y enlaces a otros sitios PGP.
- **S/MIME Charter:** los últimos borradores de RFC e Internet para S/MIME.
- **S/MIME Central:** el sitio web de RSA Inc. para S/MIME. Incluye FAQ y otra información útil.

5.4 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Clave de sesión	MIME (<i>Multipurpose Internet Mail Extensions</i>)	S/MIME
Confianza		ZIP
Correo electrónico	PGP (<i>Pretty Good Privacy</i>)	
Firma separada	Radix 64	

PREGUNTAS DE REPASO

- 5.1.** ¿Cuáles son los cinco servicios principales que ofrece PGP?
- 5.2.** ¿Cuál es la utilidad de una firma separada?
- 5.3.** ¿Por qué PGP genera una firma antes de aplicar la compresión?
- 5.4.** ¿Qué es la conversión R64?
- 5.5.** ¿Por qué es la conversión R64 útil para una aplicación de correo electrónico?
- 5.6.** ¿Por qué es necesaria la función de segmentación y reensamblado en PGP?
- 5.7.** ¿Cómo usa PGP el concepto de confianza?
- 5.8.** ¿Qué es RFC 822?
- 5.9.** ¿Qué es MIME?
- 5.10.** ¿Qué es S/MIME?

PROBLEMAS

- 5.1.** PGP usa el modo CFB (*Cipher FeedBack*) de CAST-128, mientras que la mayoría de las aplicaciones de cifrado (diferentes del cifrado de claves) usan el modo CBC (*cipher block chaining*). Tenemos

$$\text{CBC: } C_i = E_K[C_{i-1} \oplus P_i]; \quad P_i = C_{i-1} \oplus D_K[C_i]$$

$$\text{CFB: } C_i = P_i \oplus E_K[C_{i-1}]; \quad P_i = C_i \oplus E_K[C_{i-1}]$$

Los dos parecen proporcionar la misma seguridad. Razona por qué PGP utiliza el modo CFB.

- 5.2** Los primeros 16 bits del resumen del mensaje en una firma PGP se traducen en claro.
- ¿Hasta qué punto compromete este hecho la seguridad del algoritmo *hash*?
 - ¿Hasta qué punto realiza realmente la función de ayudar a determinar si se usó la clave RSA correcta para descifrar el resumen?
- 5.3** En la Figura 5.4, cada entrada en el fichero de claves públicas contiene un campo de confianza en el propietario que indica el grado de confianza asociada a ese dueño de clave pública. ¿Por qué no es suficiente con eso? Es decir, si este propietario es fiable y se supone que esa es la clave pública del dueño, ¿por qué no es eso suficiente para permitir que PGP use esta clave pública?
- 5.4** Consideremos la conversión radix 64 como una forma de cifrado. En este caso no hay clave. Pero supongamos que un oponente supiera sólo que se estaba usando algún algoritmo de sustitución para cifrar el texto en inglés. ¿En qué grado sería efectivo este algoritmo contra el criptoanálisis?
- 5.5** Phil Zimmermann eligió los algoritmos de cifrado simétrico IDEA, triple DES de tres claves y CAST-128 para PGP. Razona por qué cada uno de los siguientes algoritmos de cifrado simétrico descritos en este libro es adecuado o no para PGP: DES, triple DES de dos claves, Blowfish y RC5.

APÉNDICE 5A COMPRESIÓN DE DATOS USANDO ZIP

PGP hace uso de un paquete de compresión llamado ZIP, escrito por Jean-lup Gailly, Mark Adler y Richard Wales. ZIP es un paquete *freeware* escrito en C que se ejecuta como una utilidad en UNIX y otros sistemas. Funcionalmente es equivalente a PKZIP, un paquete *shareware* muy utilizado para sistemas Windows desarrollados por PKWARE, Inc. El algoritmo Zip es quizás la técnica de compresión más usada por distintas plataformas; las versiones *freeware* y *shareware* están disponibles para Macintosh y otros sistemas, así como para Windows y UNIX.

Zip y algoritmos similares son el producto de la investigación de Jacob Ziv y Abraham Lempel. En 1977, describieron un técnica basada en un *buffer* de ventana con desplazamiento que contiene los textos procesados más recientemente [ZIV77]. Este algoritmo se conoce generalmente como LZ77. Se usa una versión de este algoritmo en el esquema de compresión zip (PKZIP, gzip, zipit, etc.).

LZ77 y sus variantes explotan el hecho de que las palabras y las frases de un flujo de texto (patrones de imagen en el caso de GIF) podrían repetirse. Cuando se produce una repetición, la secuencia repetida puede reemplazarse por un código corto. El programa de compresión busca las repeticiones y desarrolla códigos sobre la marcha para reemplazar la secuencia repetida. De vez en cuando, los códigos se reutilizan para capturar nuevas secuencias. El algoritmo debe definirse de forma que el programa de descompresión pueda deducir la correspondencia real entre códigos y secuencias de datos fuente.

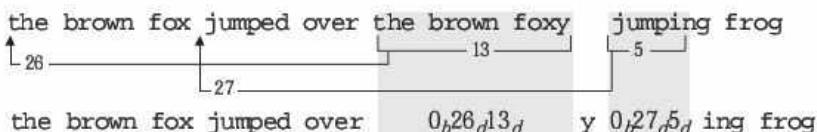


Figura 5.9 Ejemplo de esquema LZ77

Antes de fijarnos en los detalles de LZ77, observemos un ejemplo sencillo⁴. Consideremos la frase sin sentido

The brown fox jumped over the brown foxy jumping frog

con una extensión de 53 octetos = 424 bits. El algoritmo procesa este texto de izquierda a derecha. Inicialmente, cada carácter se corresponde con un patrón de nueve bits que consiste en un 1 binario seguido de la representación ASCII de ocho bits del carácter. A medida que avanza el procesamiento, el algoritmo busca secuencias repetidas. Cuando se encuentra una repetición, el algoritmo continúa buscando hasta que la repetición termina. En otras palabras, cada vez que se produce una repetición, el algoritmo incluye todos los caracteres posibles. La primera secuencia encontrada es **the brown fox**. Esta secuencia se sustituye por un puntero a la secuencia anterior y la longitud de la secuencia. En este caso la secuencia anterior a **the brown fox** ocurre 26 posiciones de caracteres antes y la longitud de la secuencia es de 13 caracteres. Para este ejemplo, asumimos dos opciones para la codificación; un puntero de ocho bits y una longitud de cuatro bits, o un puntero de doce bits y una longitud de seis bits; una cabecera de dos bits indica qué opción se elige, con 00 indicando la primera opción y 01 la segunda. Así, la segunda aparición de **the brown fox** se codifica como <00_b><26_d><13_d> o 00 00011010 1101.

Las partes que quedan del mensaje comprimido son la letra **y**, la secuencia <00_b><27_d><5_d>, que sustituye la secuencia formada por el carácter de espacio seguido de **jump**, y la secuencia de caracteres **ing frog**.

La Figura 5.9 ilustra las correspondencias de compresión. El mensaje comprimido se compone de 35 caracteres de nueve bits y dos códigos, para un total de $35 \times 9 + 2 \times 14 = 343$ bits. Esto se compara con los 424 bits del mensaje descomprimido para un ratio de compresión de 1.24.

ALGORITMO DE COMPRESIÓN

El algoritmo de compresión para LZ77 y sus variantes hacen uso de dos *buffers*. Un **buffer histórico deslizante** contiene los últimos N caracteres de los datos de entrada que se han procesado, y un **buffer de entrada** contiene los siguientes L caracteres que se van a procesar (Figura 5.10a). El algoritmo intenta hacer coincidir dos o más caracteres del principio del *buffer* de entrada con una ristra del *buffer* histórico de desplazamiento. Si no se encuentran coincidencias, el primer carácter del *buffer* de entrada se devuelve como un carácter de nueve bits y también se desplaza en la ventana deslizante, con lo cual se elimina el carácter más antiguo de la misma. Si se encuentra una coincidencia, el algoritmo continúa buscando la coincidencia más larga. Entonces la ristra coinciden-

⁴ Basado en un ejemplo de [WEIS93].

te se devuelve en forma de terceto (indicador, puntero, longitud). Para una ristra de K caracteres, los K caracteres más antiguos en la ventana deslizante se retiran, y los K caracteres de la ristra codificada se introducen en la ventana.

La Figura 5.10b muestra la operación de este esquema en nuestra secuencia de ejemplo. La ilustración asume una ventana deslizante de 39 caracteres y un *buffer* de entrada de 13 caracteres. En la parte superior del ejemplo, los primeros 40 caracteres se han procesado y la versión descomprimida de los 39 caracteres más recientes está en la ventana deslizante. Lo que queda se encuentra en la ventana de entrada. El algoritmo de compresión determina la siguiente coincidencia, traspasa 5 caracteres del *buffer* de entrada a la ventana con desplazamiento, y extrae el código para esta ristra. El estado del *buffer* después de estas operaciones se muestra en la parte inferior del ejemplo.

Mientras LZ77 es efectivo y se adapta a la naturaleza de la entrada actual, tiene algunas desventajas. El algoritmo usa una ventana finita para buscar coincidencias en el texto anterior. Para un bloque muy largo de texto, comparado con el tamaño de la ventana, se eliminan muchas posibles coincidencias. El tamaño de la ventana se puede aumentar, pero esto trae consigo dos consecuencias negativas: (1) el tiempo de procesamiento del algoritmo aumenta porque debe realizar una comparación de ristas con el *buffer* de entrada para cada posición de la ventana deslizante, y (2) el campo <puntero> debe ser más largo para ajustarse a los saltos más largos.

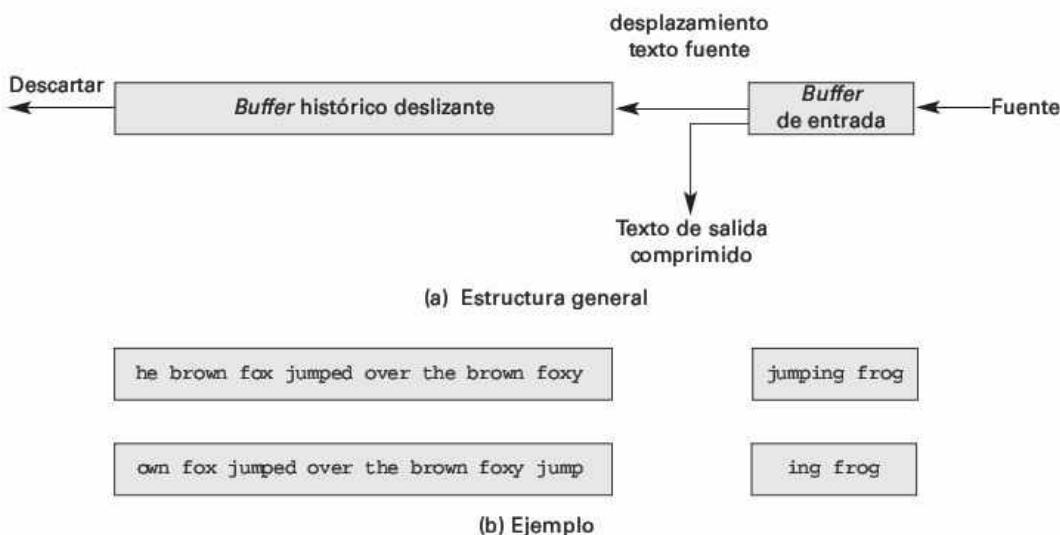


Figura 5.10 Esquema LZ77

ALGORITMO DE DESCOMPRESIÓN

La descompresión de texto comprimido LZ77 es sencilla. El algoritmo de descompresión debe guardar los últimos N caracteres de la salida descomprimida. Cuando se encuentra una ristra codificada, el algoritmo de descompresión usa los campos <puntero> y <longitud> para reemplazar el código con la ristra real de texto.

APÉNDICE 5B CONVERSIÓN RADIX 64

Tanto PGP como S/MIME hacen uso de una técnica de codificación conocida como conversión radix 64. Esta técnica convierte entradas binarias arbitrarias en salidas de caracteres imprimibles. Su forma de codificación tiene las siguientes características:

1. El rango de la función es un grupo de caracteres representable universalmente en todos los sitios, no una codificación binaria específica de ese grupo de caracteres. Así, los propios caracteres se pueden codificar en cualquier forma que sea necesaria para un sistema específico. Por ejemplo, el carácter «E» se representa en un sistema basado en ASCII como 45 hexadecimal y en un sistema basado en EBCDIC como C5 hexadecimal.
2. El grupo de caracteres consiste en 65 caracteres imprimibles, uno de los cuales se usa para relleno. Con $2^6 = 64$ caracteres disponibles, cada carácter se puede usar para representar seis bits de entrada.
3. No se incluyen caracteres de control en el grupo. Así, un mensaje codificado en radix 64 puede atravesar sistemas de correo que comprueben el flujo de datos en busca de caracteres de control.
4. No se usa el carácter guión («-»). Debido a que este carácter es significativo en el formato RFC 822 debería evitarse aquí.

La Tabla 5.9 muestra las correspondencias de los valores de entrada de seis bits con los caracteres. El grupo de caracteres consiste en los caracteres alfanuméricos más «+» y «/». El carácter «=» se usa como carácter de relleno.

Tabla 5.9 Codificación radix 64

Valor de 6 bits	Codificación del carácter	Valor de 6 bits	Codificación del carácter	Valor de 6 bits	Codificación del carácter	Valor de 6 bits	Codificación del carácter
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						Relle-no	=

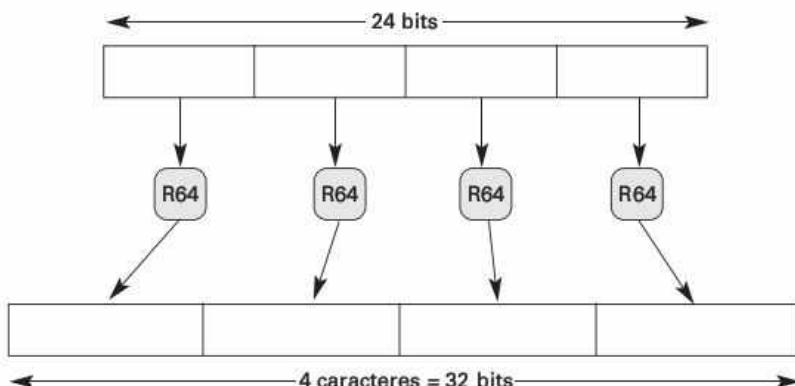


Figura 5.11 Codificación imprimible de datos binarios en formato radix 64

La Figura 5.11 ilustra el esquema simple de correspondencias. La entrada binaria se procesa en bloques de tres octetos, o 24 bits. Cada grupo de seis bits en el bloque de 24 bits se convierte en un carácter. En la figura, los caracteres se muestran codificados como cantidades de ocho bits. En este caso típico, cada entrada de 24 bits se expande a una salida de 32 bits.

Por ejemplo, consideremos la secuencia de texto de 24 bits 00100011 01011100 10010001, 235C91 en hexadecimal. Si se organiza esta entrada en bloques de seis bits, se obtiene:

001000 110101 110010 010001

Los valores decimales de seis bits extraídos son 8, 53, 50, 17. Buscarlos en la Tabla 5.9 da como resultado los siguientes caracteres con codificación radix 64: I1yR. Si estos caracteres se almacenan en formato ASCII de ocho bits con el bit de paridad fijado a cero, tenemos

01001001 00110001 01111001 01010010

En hexadecimal corresponde a 49317952. Para resumir,

Datos de entrada	
Representación binaria	00100011 01011100 10010001
Representación hexadecimal	235C91
Codificación radix 64 de datos de entrada	
Representación de caracteres	I1yR
Código ASCII (8 bits, paridad cero)	01001001 00110001 01111001 01010010
Representación hexadecimal	49317952

APÉNDICE 5C GENERACIÓN DE NÚMEROS ALEATORIOS DE PGP

PGP usa un esquema complejo y potente para generar números aleatorios y números pseudoaleatorios para una gran variedad de propósitos. PGP genera números aleatorios del contenido y el tiempo de las pulsaciones de tecla del usuario, y números pseudoaleatorios usando un algoritmo basado en el de ANSI X9.17. PGP usa estos números con la siguiente finalidad:

- Números aleatorios:
 - usados para generar parejas de claves RSA
 - proporcionan la semilla inicial para el generador de números pseudoaleatorios
 - proporcionan entrada adicional durante la generación de números pseudoaleatorios
- Números pseudoaleatorios:
 - usados para generar claves de sesión
 - usados para generar vectores de inicialización para ser usados con la clave de sesión en el cifrado en modo CFB

NÚMEROS ALEATORIOS

PGP tiene un *buffer* de 256 bytes de bits aleatorios. Cada vez que PGP espera una pulsación en el teclado, graba la hora, en formato de 32 bits, a la que empieza a esperar. Cuando recibe la pulsación, registra la hora a la que se pulsó la tecla y el valor de ocho bits de la pulsación. La información de tiempo y la pulsación se usan para generar una clave que, a su vez, se usa para cifrar el valor actual del *buffer* de bits aleatorios.

NÚMEROS PSEUDOALEATORIOS

La generación de números pseudoaleatorios hace uso de una semilla de 24 octetos y produce una clave de sesión de 16 octetos, un vector de inicialización de ocho octetos y una nueva semilla para la generación de los próximos números pseudoaleatorios. El algoritmo usa las siguientes estructuras de datos:

1. Entrada
 - randseed.bin (24 octetos): si el fichero está vacío, se llena con 24 octetos aleatorios.
 - mensaje: la clave de sesión y el vector de inicialización que se van a usar para cifrar un mensaje son por sí mismos una función de ese mensaje. Esto contribuye a la aleatoriedad de la clave y el vector de inicialización, y si un oponente ya conoce el contenido en texto claro del mensaje, no hay necesidad aparente de capturar la clave de sesión de un solo uso.
2. Salida
 - K (24 octetos): los primeros 16 octetos, K[0..15], contienen una clave de sesión, y los últimos ocho octetos, K[16..23], contienen un vector de inicialización.
 - randseed.bin (24 octetos): se coloca una nueva semilla en este fichero.

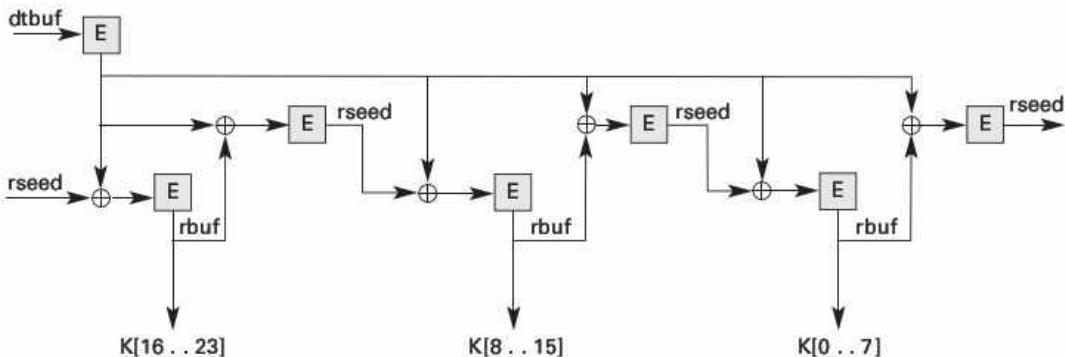


Figura 5.12 Generación de clave de sesión y vector de inicialización PGP
(pasos desde G2 hasta G8)

3 Estructuras internas de datos

- dtbuf (ocho octetos): los primeros cuatro octetos, dtbuf[0..3], se inicializan con el valor actual de fecha y hora. Este *buffer* equivale a la variable DT en el algoritmo X12.17.
- rkey (16 octetos): se usa la clave de cifrado CAST-128 en todas las fases del algoritmo.
- rseed (ocho octetos): equivalente a la variable X12.17 V_i.
- rbuf (ocho octetos): un número pseudoaleatorio generado por el algoritmo. Este *buffer* equivale a la variable X12.17 R_i.
- K' (24 octetos); *buffer* temporal para el nuevo valor de randseed.bin.

El algoritmo se compone de nueve pasos, desde G1 hasta G9. El primero y el último pasos son pasos de confusión, que tienen la finalidad de reducir el valor de un fichero randseed.bin capturado a un oponente. Los pasos restantes equivalen básicamente a tres iteraciones del algoritmo X12.17 y se ilustran en la Figura 5.12. Para resumir,

G1. [Pre-eliminar (prewash) la semilla anterior]

- a) Copiar randseed.bin en K[0..23].
- b) Tomar el *hash* del mensaje (éste ya se ha generado si el mensaje se está firmando; si no, se usan los octetos de los primeros 4K del mensaje). Usar el resultado como clave, usar un vector de inicialización nulo y cifrar K en modo CFB; almacenar el resultado en K.

G2. [Establecer la semilla inicial]

- a) Fijar dtbuf[0..3] a la hora local de 32 bits. Fijar dtbuf[4..7] todo a cero. Copiar rkey \leftarrow K[0..15]. Copiar rseed \leftarrow K[16..23].
- b) Cifrar el dtbuf de 64 bits usando el rkey de 128 bits en modo ECB; almacenar el resultado en dtbuf.

G3 [Preparar para generar octetos aleatorios] Fijar rcount $\leftarrow 0$ y k $\leftarrow 23$. El bucle de pasos G4-G7 se ejecutará 24 veces ($k = 23\dots 0$), una vez por cada octeto aleatorio generado y situado en K. La variable rcount es el número de octetos aleatorios no utilizados en rbuf. Contará hacia atrás de 8 a 0 tres veces para generar los 24 octetos.

G4 [¿Bytes disponibles?] Si rcount = 0 ir a G5, si no, ir a G7. Los pasos G5 y G6 realizan una instancia del algoritmo X12.17 para generar un nuevo *batch* de ocho octetos aleatorios.

G5 [Generar nuevos octetos aleatorios]

- rseed \leftarrow rseed \oplus dtbuf.
- rbuf $\leftarrow E_{rkey}[rseed]$ en modo ECB.

G6 [Generar la siguiente semilla]

- rseed \leftarrow rbuf \oplus dtbuf.
- rseed $\leftarrow E_{rkey}[rseed]$ en modo ECB.
- Fijar rcount $\leftarrow 8$.

G7. [Transferir un bit cada vez desde rbuf a K]

- Fijar rcount \leftarrow rcount - 1.
- Generar un byte aleatorio b, y fijar $K[k] \leftarrow rbuf[rcount] \oplus b$.

G8 [¿Hecho?] Si $k=0$ ir a G9 si no fijar $k \leftarrow k-1$ e ir a G4

G9 [Post-eliminar (postwash) la semilla y devolver resultado]

- Generar 24 bytes más con el método de los pasos G4-G7, pero no aplicar el XOR a un byte aleatorio en G7. Colocar el resultado en el *buffer K'*.
- Cifrar K' con la clave $K[0..15]$ y el vector de inicialización $K[16..23]$ en modo CFB; almacenar el resultado en randseed.bin.
- Devolver K.

No debería ser posible determinar la clave de sesión a partir de los 24 nuevos octetos generados en el paso G9.a. Sin embargo, para garantizar que el fichero randseed.bin almacenado no proporciona información sobre la clave de sesión más reciente, los 24 nuevos octetos se cifran y el resultado se almacena como la nueva semilla.

Este algoritmo elaborado debería proporcionar números pseudoaleatorios criptográficamente robustos.

CAPÍTULO 6

Seguridad IP

6.1 Introducción a la seguridad IP

Aplicaciones de IPSec
Beneficios de IPSec
Aplicaciones de enrutamiento

6.2 Arquitectura de seguridad IP

Documentos de IPSec
Servicios IPSec
Asociaciones de seguridad
Modo transporte y modo túnel

6.3 Cabecera de autentificación

Servicio contra repeticiones
Valor de comprobación de integridad
Modos transporte y túnel

6.4 Encapsulamiento de la carga útil de seguridad

El formato ESP
Algoritmos de cifrado y autentificación
Relleno
Modo transporte y modo túnel

6.5 Combinación de asociaciones de seguridad

Autentificación más confidencialidad
Combinaciones básicas de asociaciones de seguridad

6.6 Gestión de claves

Protocolo de determinación de claves Oakley
ISAKMP

6.7 Bibliografía y sitios web recomendados

6.8 Términos clave, preguntas de repaso y problemas

Términos clave
Preguntas de repaso
Problemas

Apéndice 6A Comunicación entre redes y protocolos de internet

El papel de un protocolo de Internet
IPv4
IPv6

Si un espía divulga una noticia secreta antes de que sea el momento oportuno, debe ser ejecutado, junto con el hombre a quien reveló el secreto.

El arte de la guerra, SUN Tzu

La comunidad de Internet ha desarrollado mecanismos de seguridad para aplicaciones específicas en una serie de áreas como, por ejemplo, correo electrónico (S/MIME, PGP), cliente/servidor (Kerberos), acceso a la Web (SSL, *Secure Sockets Layer*), entre otras. Sin embargo, los usuarios tienen preocupaciones relativas a la seguridad que rebasan las capas de protocolos. Por ejemplo, una empresa puede hacer que su red privada TCP/IP sea segura desautorizando conexiones a sitios no fiables, cifrando paquetes que salen de sus instalaciones y autenticando los paquetes que se reciben. Implementando la seguridad en el nivel de IP, una organización puede conseguir una red segura no sólo para las aplicaciones que poseen mecanismos de seguridad, sino también para las aplicaciones que no los tienen.

La seguridad IP abarca tres áreas funcionales: autenticación, confidencialidad y gestión de claves. El mecanismo de autenticación garantiza que un paquete recibido, de hecho, fue transmitido por la parte identificada como fuente u origen en la cabecera del paquete. Además, este mecanismo asegura que el paquete no ha sido alterado durante la transmisión. La herramienta de confidencialidad permite que los nodos que se comunican cifren los mensajes para prevenir escuchas de terceras partes. La herramienta de gestión de claves se ocupa del intercambio seguro de claves.

Este Capítulo comienza con una descripción general de la seguridad IP (IPSec) y una introducción a la arquitectura de IPSec. Luego, se estudia en profundidad cada una de las tres áreas funcionales. A su vez, el apéndice de este Capítulo repasa los protocolos de Internet.

6.1 INTRODUCCIÓN A LA SEGURIDAD IP

En 1994, el Comité de Arquitectura de Internet (IAB: *Internet Architecture Board*) publicó un informe titulado *Seguridad en la arquitectura de Internet* (RFC 1636). El informe manifestaba el consenso general sobre la necesidad de una mayor y mejor seguridad en Internet, e identificaba las áreas fundamentales para los mecanismos de seguridad. Entre éstas se encontraba la necesidad de proteger la infraestructura de la red de la observación y el control no autorizados del tráfico de red, así como la necesidad de asegurar el tráfico entre usuarios finales utilizando mecanismos de autenticación y cifrado.

Estas preocupaciones están del todo justificadas. Como prueba, el informe anual de 2001 del Equipo de Respuesta a Emergencias en Computadores o CERT (*Computer Emergency Response Team*) informaba sobre más de 52.000 incidentes de seguridad. Los tipos de ataque más graves incluían los falsos IP, en los que los intrusos crean paquetes con direcciones IP falsas y explotan las aplicaciones que usan autenticación basada en IP; y distintas formas de escucha y captura de paquetes, donde los atacantes leen la información transmitida, incluida la de conexión al sistema y la de los contenidos de bases de datos.

En respuesta a estas cuestiones, la IAB incluyó la autentificación y el cifrado como características necesarias de seguridad en el IP de nueva generación, que se conoce como IPv6. Afortunadamente, estas capacidades de seguridad se diseñaron para que fueran empleadas con el actual IPv4 y el futuro IPv6. Esto significa que los fabricantes ya pueden empezar a ofrecer estas características, y ya muchos de ellos han incorporado algunas capacidades de IPSec a sus productos.

APLICACIONES DE IPSEC

IPSec proporciona la capacidad de asegurar las comunicaciones a través de una LAN, de una WAN privada y pública y de Internet. Los siguientes son algunos ejemplos de su uso:

- **Conexión segura entre oficinas sucursales a través de Internet:** una compañía puede construir una red virtual privada y segura a través de Internet o de una WAN pública. Esto permite que una empresa se apoye fuertemente en Internet y reduzca su necesidad de redes privadas, disminuyendo gastos y costes en la gestión de red.
- **Acceso remoto seguro a través de Internet:** un usuario final cuyo sistema esté dotado de protocolos de seguridad IP puede hacer una llamada local a su proveedor de servicios de Internet (ISP, *Internet Service Provider*) y acceder de forma segura a la red de una compañía. Esto reduce los gastos de los empleados que se tienen que desplazar y de los empleados a distancia.
- **Establecimiento de conexión extranet e intranet con socios:** IPSec se puede usar para hacer que las comunicaciones con otras organizaciones sean seguras, garantizando la autentificación y la confidencialidad y proporcionando un mecanismo de intercambio de claves.
- **Mejora de la seguridad en el comercio electrónico:** aunque algunas aplicaciones web y de comercio electrónico tienen incorporados protocolos de seguridad, el uso de IPSec mejora la seguridad.

La característica principal de IPSec que permite dar soporte a esta variedad de aplicaciones es que puede cifrar y/o autenticar *todo* el tráfico en el nivel IP. Por lo tanto, pueden asegurarse todas las aplicaciones distribuidas, incluyendo conexión remota, cliente/servidor, correo electrónico, transferencia de ficheros, acceso a la web, etc.

La Figura 6.1 representa un ejemplo común del uso de IPSec. Una organización tiene algunas LAN en lugares dispersos. En cada LAN hay tráfico IP que no es seguro. Los protocolos IPSec se utilizan para el tráfico exterior, a través de una WAN privada o pública. Estos protocolos operan en dispositivos de red, como por ejemplo un *router* o un cortafuegos, que conecta cada LAN al mundo exterior. El dispositivo de red IPSec cifrará y comprimirá todo el tráfico que entre en la WAN, y descifrará y descomprimirá todo el tráfico que provenga de ella; estas operaciones son transparentes a las estaciones de trabajo y a los servidores en la LAN. También es posible la transmisión segura con usuarios individuales que se conectan a la WAN. Dichas estaciones de trabajo de usuarios deben implementar los protocolos IPSec para proporcionar seguridad.

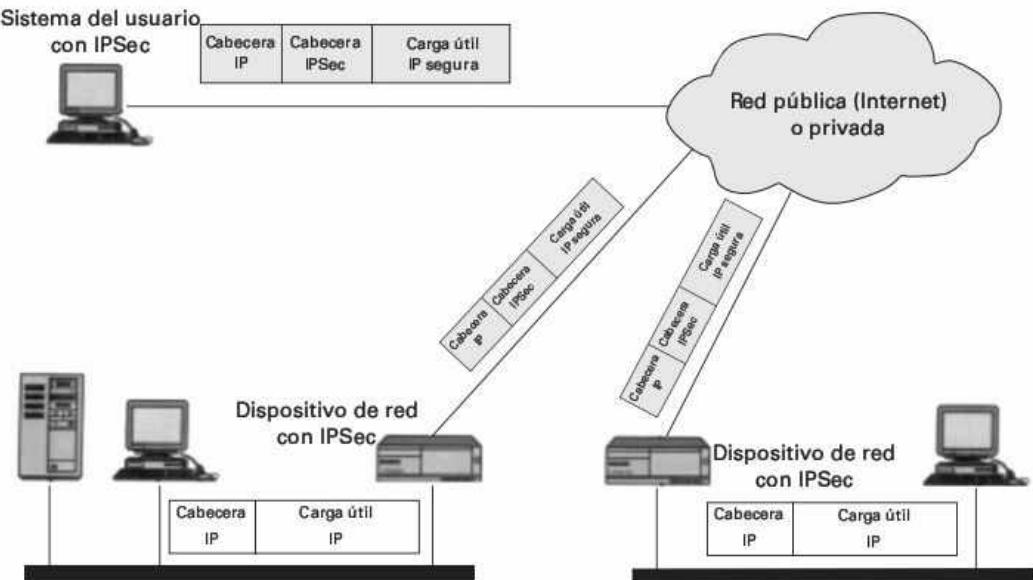


Figura 6.1 Entorno de seguridad IP

BENEFICIOS DE IPSEC

En [MARK97] se presentan los siguientes beneficios de IPSec:

- Cuando IPSec se implementa en un cortafuegos o un *router*, proporciona una gran seguridad que se puede aplicar a todo el tráfico que lo cruza. El tráfico en una compañía o grupo de trabajo no provoca costes adicionales de procesamiento relativo a la seguridad.
- IPSec es seguro en un cortafuegos si se obliga a que todo el tráfico que proviene del exterior use IP, y el cortafuegos es el único medio de entrada desde Internet a la organización.
- IPSec está por debajo de la capa de transporte (TCP, UDP) y, por ello, es transparente a las aplicaciones. No es necesario cambiar el *software* en el sistema de un usuario o de un servidor cuando IPSec se implementa en el cortafuegos o el *router*. Incluso si IPSec se implementa en sistemas finales, el *software* de nivel superior, incluyendo aplicaciones, no se ve afectado.
- IPSec puede ser transparente a usuarios finales. No es necesario entrenar a los usuarios para la utilización de mecanismos de seguridad, ni suministrar material relativo al uso de claves por cada usuario, ni inhabilitar dicho material cuando los usuarios abandonan la organización.
- IPSec puede proporcionar seguridad a usuarios individuales si es necesario, lo cual es útil para trabajadores externos y para establecer una subred virtual segura en una organización para las aplicaciones confidenciales.

APLICACIONES DE ENRUTAMIENTO

Además de dar soporte a usuarios finales y proteger los sistemas y las redes de las instalaciones de la organización, IPSec puede desempeñar un papel fundamental en la arquitectura de enrutamiento necesaria para la comunicación entre redes. [HUIT98] ofrece una serie de ejemplos del uso de IPSec. IPSec puede garantizar que:

- Un anuncio de *router* (un nuevo *router* anuncia su presencia) procede de un *router* autorizado.
- Un anuncio de un *router* vecino (un *router* intenta establecer o mantener una relación con un *router* en otro dominio de enrutamiento) viene de un *router* autorizado.
- Un mensaje redirigido proviene del *router* al que se envió el paquete inicial.
- Una actualización de enrutamiento no se falsifica.

Sin estas medidas de seguridad, un oponente puede interrumpir las comunicaciones o desviar el tráfico. Los protocolos de enrutamiento como OSPF deberían ejecutarse por encima de las asociaciones de seguridad entre *routers* que están definidos por IPSec.

6.2 ARQUITECTURA DE SEGURIDAD IP

La especificación IPSec se ha hecho muy compleja. Para entender la arquitectura general, en primer lugar, conviene observar los documentos que definen IPSec. Luego se tratan los servicios de IPSec y se introduce el concepto de asociación de seguridad.

DOCUMENTOS DE IPSEC

La especificación IPSec se compone de numerosos documentos. Los más importantes de ellos, publicados en noviembre de 1998, son los RFC 2401, 2402, 2406 y 2408:

- RFC 2401: descripción general de una arquitectura de seguridad
- RFC 2402: descripción de la extensión de autentificación de un paquete a IPv4 e IPv6
- RFC 2406: descripción de la extensión de cifrado de un paquete a IPv4 e IPv6
- RFC 2408: especificación de las capacidades de gestión de claves

Permitir estas características es obligatorio para IPv6 y opcional para IPv4. En ambos casos, las características de seguridad se implementan como cabeceras de extensión que siguen a la cabecera IP principal. La cabecera de extensión para la autentificación se conoce como *cabecera de autentificación (AH, Authentication Header)*; y para el cifrado se conoce como *cabecera de encapsulado de carga útil de seguridad (ESP, Encapsulating Security Payload header)*.

Además de estos cuatro RFC, el grupo de trabajo (*Security Protocol Working Group*), creado por la IETF, ha publicado una serie de borradores adicionales. Los documentos se dividen en siete grupos, como se muestra en la Figura 6.2 (RFC 2401):

- **Aquitectura:** cubre los conceptos generales, los requisitos de seguridad, las definiciones y los mecanismos que definen la tecnología IPSec.

- **Encapsulado de carga útil de seguridad (ESP):** cubre el formato del paquete y los aspectos generales relacionados con el uso de ESP para el cifrado de paquetes y, de manera opcional, para la autentificación.

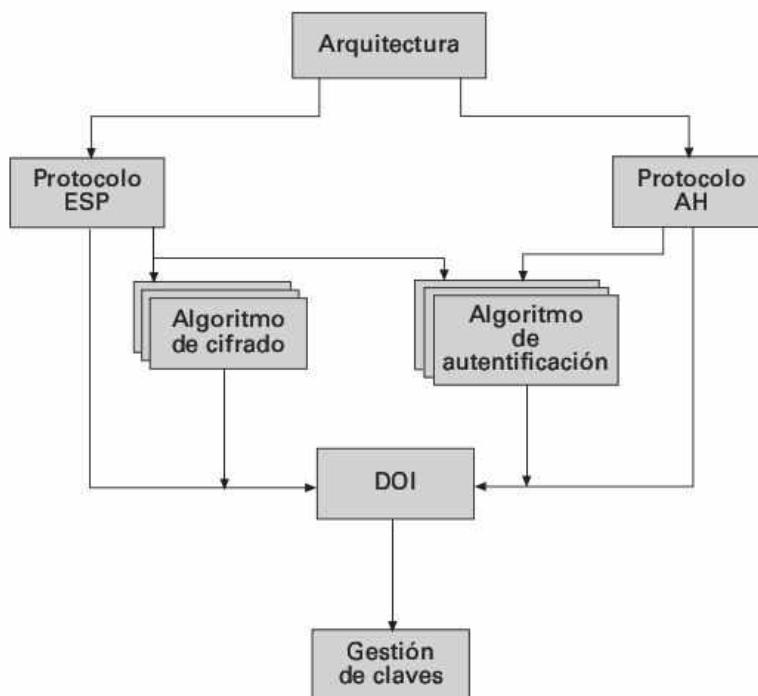


Figura 6.2 Esquema general de documento IPsec

- **Cabecera de autentificación (AH):** cubre el formato del paquete y los aspectos generales relacionados con el uso de AH para la autentificación de paquetes.
- **Algoritmo de cifrado:** un conjunto de documentos que describen cómo se utilizan distintos algoritmos de cifrado para ESP.
- **Algoritmo de autentificación:** un conjunto de documentos que describen cómo se utilizan distintos algoritmos de autentificación para AH y para la opción de autentificación de ESP.
- **Gestión de claves:** documentos que describen los esquemas de gestión de claves.
- **Dominio de interpretación (DOI: Domain of Interpretation):** contiene los valores necesarios para que los demás documentos se relacionen entre sí. Estos incluyen identificadores para algoritmos aprobados de cifrado y de autentificación, así como parámetros operativos como el tiempo de vida de las claves.

SERVICIOS IPSEC

IPSec proporciona servicios de seguridad en la capa IP permitiendo que un sistema elija los protocolos de seguridad necesarios, determine los algoritmos que va a usar para el

servicio o servicios y ubique las claves criptográficas necesarias para proporcionar los servicios solicitados. Se usan dos protocolos para proporcionar seguridad: un protocolo de autenticación designado por la cabecera del protocolo, AH, y un protocolo combinado de cifrado/autenticación designado por el formato del paquete para ese protocolo, ESP. Los servicios son los siguientes:

- Control de acceso.
- Integridad sin conexión.
- Autenticación del origen de datos.
- Rechazo de paquetes reenviados.
- Confidencialidad (cifrado).
- Confidencialidad limitada del flujo de tráfico.

Tabla 6.1 Servicios de IPSec

	AH	ESP (sólo cifrado)	ESP (cifrado y autenticación)
Control de acceso	✓	✓	✓
Integridad sin conexión	✓		✓
Autenticación del origen de datos			
	✓		✓
Rechazo de paquetes reenviados	✓	✓	✓
Confidencialidad		✓	✓
Confidencialidad limitada del flujo de tráfico		✓	✓

La Tabla 6.1 muestra qué servicios proporcionan los protocolos AH y ESP. Para ESP, existen dos casos: con y sin opción de autenticación. Tanto AH como ESP son vehículos para el control de acceso, basados en la distribución de claves criptográficas y la gestión de flujos de tráfico referente a estos protocolos de seguridad.

ASOCIACIONES DE SEGURIDAD

Un concepto fundamental que aparece en los mecanismos de autenticación y confidencialidad en IP es el de asociación de seguridad (SA, *Security Association*). Una asociación es una relación unidireccional entre un emisor y un receptor que ofrece servicios de seguridad al tráfico que se transporta. Si se necesita una relación que haga posible un intercambio bidireccional seguro, entonces se requieren dos asociaciones de seguridad. Los servicios de seguridad se suministran a una SA para que use AH o ESP, pero no los dos.

Una asociación de seguridad se identifica únicamente por tres parámetros:

- **Índice de parámetros de seguridad (SPI, Security Parameters Index):** una ristra de bits asignada a esta SA y que tiene sólo significado local. El SPI se transporta en cabeceras AH y ESP para permitir que el sistema receptor elija la SA con la cual se procesará un paquete recibido.
- **Dirección IP de destino:** actualmente sólo se permiten direcciones de un único destino (*unicast*); ésta es la dirección del destino final de la SA, que puede ser un sistema de un usuario final o un sistema de red, como por ejemplo un cortafuegos o un *router*.
- **Identificador del protocolo de seguridad:** indica si la asociación es una asociación de seguridad AH o ESP.

Por consiguiente, en cualquier paquete IP¹, la asociación de seguridad se identifica únicamente por la dirección de destino en la cabecera IPv4 o IPv6 y el SPI en la cabecera de extensión adjunta (AH o ESP).

Parámetros de SA

En cada implementación de IPSec hay una base de datos nominal² de asociaciones de seguridad que define los parámetros asociados con cada SA. Una asociación de seguridad se define, normalmente, por los siguientes parámetros:

- **Contador de número de secuencia:** un valor de 32 bits que se utiliza para generar el campo *número de secuencia* en las cabeceras AH o ESP, que se describen en el apartado 6.3 (se requiere para todas las implementaciones).
- **Desbordamiento del contador de secuencia:** un indicador que señala si el desbordamiento del contador de números de secuencia debería generar una acción de auditoría y evitar la transmisión de más paquetes en esta SA (se requiere para todas las implementaciones).
- **Ventana contra repeticiones:** se usa para determinar si un paquete AH o ESP que llega es una repetición, como se describe en el apartado 6.3 (se requiere para todas las implementaciones).
- **Información AH:** algoritmo de autenticación, claves, tiempos de vida de las claves y parámetros relacionados que se usan con AH (se requiere para las implementaciones AH).
- **Información ESP:** algoritmo de cifrado y autenticación, claves, valores de inicialización, tiempos de vida de las claves y parámetros relacionados que se usan con ESP (se requiere para las implementaciones ESP).
- **Tiempo de vida de la asociación de seguridad:** un intervalo de tiempo o contador de bytes después del cual una SA se debe reemplazar con una nueva SA (y un nuevo SPI) o se debe finalizar, junto con una indicación de cuáles de estas acciones deberían ocurrir (se requiere para todas las implementaciones).
- **Modo de protocolo IPSec:** túnel, transporte o modo comodín (se requiere para todas las implementaciones). Estos modos se tratarán en esta sección.

¹ En este Capítulo, el término *paquete IP* se refiere tanto a un datagrama IPv4 como a un paquete IPv6.

² Nominal en el sentido de que la funcionalidad que proporciona una base de datos de asociaciones de seguridad debe estar presente en cada implementación de IPSec, pero la forma en que se proporciona dicha funcionalidad es decisión del implementador.

- **MTU (maximum transmission unit) del camino:** cualquier unidad de transferencia máxima que se observe en el camino (tamaño máximo de un paquete que se puede transmitir sin fragmentación) y variables de caducidad (se requiere en todas las implementaciones).

El mecanismo de gestión de claves que se emplea para que éstas se distribuyan está ligado a los mecanismos de autenticación y privacidad sólo mediante el índice de parámetros de seguridad. Por ello, la autenticación y la privacidad se han especificado independientemente de cualquier mecanismo específico de gestión de claves.

Selectores de SA

IPSec proporciona al usuario una gran flexibilidad en la forma en que los servicios de IPSec se aplican al tráfico IP. Como se verá más adelante, las SA se pueden combinar de distintas maneras para producir la configuración de usuario deseada. Además, IPSec proporciona un alto grado de segmentación al discriminar entre tráfico al que se aplica protección de IPSec y tráfico que tiene permiso para evitar o pasar por alto a IPSec, refiriéndose en el primer caso al tráfico de IP a SA específicas.

El medio por el que el tráfico IP se relaciona con SA específicas (o a ninguna SA en el caso del tráfico al que no se aplica IPSec) es la base de datos de políticas de seguridad (SPD, *Security Policy Database*). En su forma más simple, una SPD contiene entradas, cada una de las cuales define un subconjunto de tráfico IP y señala una SA para ese tráfico. En entornos más complejos, puede haber múltiples entradas que se relacionan potencialmente con una sola SA o con varias SA asociadas con una única entrada a la SPD. Con el fin de tratar este aspecto más detalladamente, se hace referencia a la documentación relevante sobre IPSec.

Cada entrada de la SPD se define por un conjunto de valores de campos del protocolo IP y de protocolos de capas superiores, llamados *selectores*. En efecto, estos selectores se usan para filtrar tráfico saliente y establecer la correspondencia con una SA en particular. El procesamiento de tráfico saliente obedece a la siguiente secuencia general para cada paquete IP:

1. Comparar los valores de los campos adecuados del paquete (los campos del selector) con la SPD para encontrar una entrada coincidente de la SPD, que señalará cero o más SA.
2. Determinar la SA, si la hubiese, para este paquete y su SPI asociado.
3. Llevar a cabo el procesamiento IPSec necesario (por ejemplo, procesamiento AH o ESP).

Los siguientes selectores determinan una entrada de la SPD:

- **Dirección IP de destino:** puede ser una única dirección IP, una lista o rango de direcciones o una dirección comodín (máscara). Las dos últimas son necesarias para permitir que más de un sistema de destino comparta la misma SA (por ejemplo, detrás de un cortafuegos).
- **Dirección IP fuente:** puede ser una única dirección IP, una lista o rango de direcciones o una dirección comodín (máscara). Las dos últimas son necesarias para permitir que más de un sistema fuente comparta la misma SA (por ejemplo, detrás de un cortafuegos).

- **ID de usuario:** un identificador de usuario obtenido del sistema operativo. No es un campo de las cabeceras IP ni de capas superiores, sino que está disponible si IPSec se ejecuta en el mismo sistema operativo que el usuario.
- **Nivel de confidencialidad de los datos:** se usa para los sistemas que proporcionan seguridad en el flujo de información (por ejemplo, secreta o no clasificada).
- **Protocolo de la capa de transporte:** se obtiene del protocolo IPv4 o del campo *siguiente cabecera* de IPv6. Puede ser un número de protocolo individual, una lista de números de protocolo o un rango de números de protocolo.
- **Protocolo IPSec (AH o ESP o AH/ESP):** si está presente, éste se obtiene del protocolo IPv4 o del campo *siguiente cabecera* de IPv6.
- **Puertos fuente y destino:** pueden ser valores individuales de puertos TCP o UDP, una lista enumerada de puertos o un puerto comodín.
- **Clase IPv6:** se obtiene de la cabecera de IPv6. Puede ser un valor específico de clase IPv6 o un valor comodín.
- **Etiqueta de flujo IPv6:** se obtiene de la cabecera IPv6. Puede ser un valor específico de la etiqueta de flujo IPv6 o un valor comodín.
- **Tipo de servicio IPv4 (TOS, Type of Service):** se obtiene de la cabecera de IPv4. Puede ser un valor específico del tipo de servicio de IPv4 o un valor comodín.

MODO TRANSPORTE Y MODO TÚNEL

Tanto AH como ESP permiten dos modos de uso: modo transporte y modo túnel. La operación de estos dos modos se entiende más fácilmente a partir de la descripción de AH y ESP, que se tratan en las secciones 6.3 y 6.4, respectivamente. Aquí se ofrece una breve introducción.

Modo transporte

El modo transporte proporciona protección principalmente a los protocolos de capas superiores. Es decir, la protección del modo transporte se extiende a la carga útil de un paquete IP. Algunos ejemplos incluyen un segmento TCP o UDP o un paquete ICMP, que operan directamente encima de IP en la pila de protocolos de un *host*. Normalmente, el modo transporte se usa para la comunicación extremo a extremo entre dos *hosts* (por ejemplo, un cliente y un servidor o dos estaciones de trabajo). Cuando un *host* ejecuta AH o ESP sobre IPv4, la carga útil consiste en los datos que habitualmente siguen a la cabecera IP. Para IPv6, la carga útil consiste en los datos que normalmente siguen a la cabecera IP y a cualquier cabecera de extensión de IPv6 que esté presente, con la posible excepción de la cabecera de opciones de destino, que se puede incluir en la protección.

ESP en modo transporte cifra y, de manera opcional, autentifica la carga útil de IP, pero no la cabecera IP. AH en modo transporte autentifica la carga útil de IP y partes seleccionadas de la cabecera IP.

Modo túnel

El modo túnel proporciona protección al paquete IP completo. Para conseguirlo, después de que se han añadido los campos AH y ESP al paquete IP, el paquete completo más los campos de seguridad se tratan como carga útil de un paquete IP «exterior» nuevo con

una nueva cabecera IP exterior. El paquete original entero, o interior, viaja a través de un «túnel» desde un punto de la red IP a otro; ningún *router* a lo largo del camino puede examinar la cabecera IP interior. Como el paquete original está encapsulado, el nuevo paquete, que es mayor, puede tener direcciones de origen y destino totalmente diferentes, lo cual añade seguridad. El modo túnel se usa cuando uno o los dos extremos de una SA es una pasarela de seguridad, como podría ser un cortafuegos o un *router* que implementa IPSec. Con el modo túnel, una serie de *hosts* en redes, detrás de cortafuegos pueden estar implicados en comunicaciones seguras sin implementar IPSec. Los paquetes no protegidos generados por dichos *hosts* se transmiten por un túnel a través de redes externas por medio de asociaciones de seguridad en modo túnel, establecidas por el *software* IPSec en el cortafuegos o el *router* seguro en los límites de la red local.

A continuación se ofrece un ejemplo del funcionamiento de IPSec en modo túnel. El *host* A de una red genera un paquete IP con la dirección de destino del *host* B en otra red. Este paquete se encamina desde el *host* origen a un cortafuegos o *router* seguro en los límites de la red de A. El cortafuegos filtra todos los paquetes que salen para determinar si se necesita procesamiento IPSec. Si este paquete de A a B requiere IPSec, el cortafuegos realiza el procesamiento IPSec y encapsula el paquete con una cabecera IP exterior. La dirección IP de origen de este paquete IP exterior es el cortafuegos, y la dirección de destino puede ser un cortafuegos que conforma la frontera de la red local de B. Ahora, este paquete se encamina al cortafuegos de B, y los *routers* intermedios sólo examinan la cabecera IP exterior. En el cortafuegos de B, se elimina la cabecera IP exterior y el paquete exterior se envía a B.

Tabla 6.2 Funcionalidad del modo túnel y del modo transporte

	SA en modo transporte	SA en modo túnel
AH	Autentifica la carga útil de IP y las partes seleccionadas de la cabecera IP y de las cabeceras de extensión de IPv6.	Autentifica todo el paquete IP interno (cabecera interior más carga útil de IP) más las partes seleccionadas de la cabecera IP exterior y las cabeceras externas de extensión de IPv6.
ESP	Cifra la carga útil de IP y cualquier cabecera de extensión de IPv6 que siga a la cabecera ESP.	Cifra el paquete IP interior.
ESP con autenticación	Cifra la carga útil de IP y cualquier cabecera de extensión de IPv6 que siga a la cabecera ESP. Autentifica la carga útil de IP, pero no la cabecera IP.	Cifra el paquete IP interior. Autentifica el paquete IP interior.

ESP en modo túnel cifra y, de manera opcional, autentifica el paquete IP interior completo, incluyendo la cabecera IP interior. AH en modo túnel autentifica el paquete IP interior completo y las partes seleccionadas de la cabecera IP exterior.

La Tabla 6.2 resume la funcionalidad del modo de transporte y del modo túnel.

6.3 CABECERA DE AUTENTIFICACIÓN

La cabecera de autentificación proporciona soporte para la integridad de los datos y la autentificación de paquetes IP. La característica de integridad de los datos garantiza que no es posible que se produzca modificación no detectada en el contenido de un paquete durante la transmisión. La característica de autentificación permite que un sistema final o dispositivo de red autentifique al usuario o aplicación y filtre el tráfico adecuadamente; también evita los ataques de suplantación de dirección que se observan hoy en día en Internet. La AH también protege contra los ataques de repetición que se describen más adelante en esta sección.

La autentificación se basa en el uso de un código de autentificación de mensajes (MAC, *Message Authentication Code*), tal y como se explicó en el Capítulo 3; de ahí que las dos partes deban compartir una clave secreta.

La cabecera de autentificación se compone de los siguientes campos (Figura 6.3):

- **Cabecera siguiente (ocho bits)**: identifica el tipo de cabecera que está inmediatamente después de ésta.
- **Longitud de carga útil (ocho bits)**: longitud de la cabecera de autentificación en palabras de 32 bits, menos dos. Por ejemplo, la longitud predeterminada del campo de datos de autentificación es de 96 bits, o tres palabras de 32 bits. Con una cabecera fija de tres palabras, hay un total de seis palabras en la cabecera, y el campo *longitud de carga útil* tiene un valor de cuatro.
- **Reservado (16 bits)**: para usos posteriores.
- **Índice de parámetros de seguridad (32 bits)**: identifica una asociación de seguridad.
- **Número de secuencia (32 bits)**: un valor de un contador que se incrementa monotónicamente, que se trata más tarde.

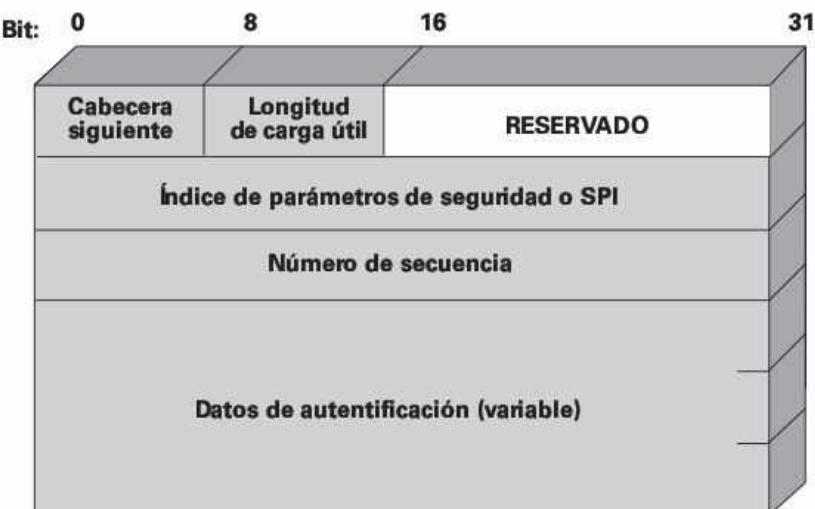


Figura 6.3 Cabecera de autentificación IPSec

- **Datos de autenticación (variable):** un campo de longitud variable (debe ser un número entero de palabras de 32 bits) que contiene el valor de comprobación de integridad (ICV, *Integrity Check Value*), o el MAC para este paquete, que se descubrirá más tarde.

SERVICIO CONTRA REPETICIONES

Un ataque de repetición es aquel en que un atacante obtiene una copia de un paquete autenticado y lo transmite más tarde al destino previsto inicialmente. El recibo de paquetes IP autenticados duplicados puede interrumpir de algún modo un servicio o tener otras consecuencias negativas. El campo *número de secuencia* sirve para impedir tales ataques. En primer lugar, analizaremos la generación de números de secuencia por parte del emisor, y luego observaremos cómo se procesa por el receptor.

Cuando se establece una nueva SA, el **emisor** inicializa un contador de números de secuencia a 0. Cada vez que se envía un paquete con esta SA, el emisor incrementa el contador y coloca el valor en el campo *número de secuencia*. Así, el primer valor que se ha de usar es 1. Si el servicio contra repeticiones está habilitado (predeterminado), el emisor no debe permitir que el número de secuencia sobrepase $2^{32} - 1$ para volver a reiniciarse en 0, ya que en ese caso habría múltiples paquetes válidos con el mismo número de secuencia. Si se alcanza el límite de $2^{32} - 1$, el emisor debería terminar esta SA y negociar una nueva SA con una clave nueva.

Debido a que IP es un servicio sin conexión y no fiable, el protocolo no garantiza que los paquetes se enviarán en orden, ni tampoco que todos los paquetes serán enviados. Por consiguiente, el documento de autenticación IPSec dicta que el **receptor** debería implementar una ventana de tamaño W , con un valor predeterminado de $W = 64$. El extremo derecho de la ventana representa el número de secuencia más alto, N , recibido hasta el momento para un paquete válido. Para cualquier paquete con un número de secuencia entre $N - W + 1$ y N que haya sido recibido correctamente (por ejemplo, adecuadamente autenticado), se marca la posición correspondiente en la ventana (Figura 6.4). Cuando se recibe un paquete, el procesamiento procede como se explica a continuación:

1. Si el paquete recibido cae dentro de la ventana y es nuevo, se comprueba el MAC. Si el paquete está autenticado, se marca la ranura correspondiente en la ventana.
2. Si el paquete recibido está a la derecha de la ventana y es nuevo, se comprueba el MAC. Si el paquete está autenticado, la ventana avanza de forma que el número de secuencia sea el extremo derecho de la ventana, y se marque en la ventana la ranura correspondiente.
3. Si el paquete recibido está a la izquierda de la ventana, o si falla la autenticación, se descarta el paquete; este hecho se puede registrar.

VALOR DE COMPROBACIÓN DE INTEGRIDAD

El campo *datos de autenticación* contiene un valor conocido como *valor de comprobación de la integridad* o, como ya se ha indicado, ICV. El ICV es un código de autenticación de mensajes o una versión truncada de un código producido por un algoritmo MAC. La especificación actual dicta que una implementación adecuada debe permitir

- HMAC-MD5-96
- HMAC-SHA-1-96

Los dos usan el algoritmo HMAC, el primero de ellos con el código *hash* MD5 y el segundo con el código *hash* SHA-1 (estos algoritmos se describieron en el Capítulo 3). En ambos casos, el valor HMAC total se calcula, pero luego se trunca usando los primeros 96 bits, que es la longitud predeterminada para el campo *datos de autenticación*.

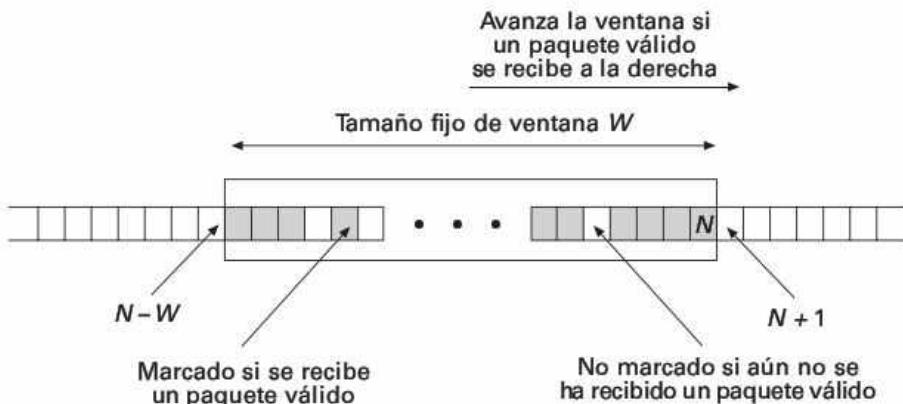


Figura 6.4 Mecanismo contra repeticiones

El MAC se calcula sobre

- Los campos de cabecera IP que no cambian durante la transmisión (invariables) o que son predecibles en valor en la llegada en el punto final para la SA de AH. Los campos que pueden cambiar durante la transmisión y cuyo valor a la llegada es impredecible se fijan a cero para el cálculo tanto en el origen como en el destino.
- La cabecera AH, a excepción del campo *datos de autenticación*. Este campo se fija a cero para el cálculo tanto en el origen como en el destino.
- Los datos completos de los protocolos de nivel superior, que se suponen invariables durante la transmisión (por ejemplo, un segmento TCP o un paquete IP interior en modo túnel).

Para IPv4, los ejemplos de campos variables son la *longitud de cabecera de Internet* y la *dirección del origen*. Un ejemplo de un campo variable pero predecible es la *dirección de destino* (con enrutado de origen flexible o estricto). Ejemplos de campos variables que se fijan a cero antes del cálculo del ICV son los campos *tiempo de vida* y *checksum de cabecera*. Obsérvese que los campos de dirección de origen y destino están protegidos para prevenir la suplantación de dirección.

Para IPv6, los ejemplos en la cabecera base son *versión* (invariable), *dirección de destino* (variable pero predecible) y *etiqueta de flujo* (variable y fijada a cero para el cálculo).

MODOS TRANSPORTE Y TÚNEL

La Figura 6.5 muestra las dos formas en que se puede usar el servicio de autenticación IPSec. En un caso, se proporciona autenticación directamente entre el servidor y las

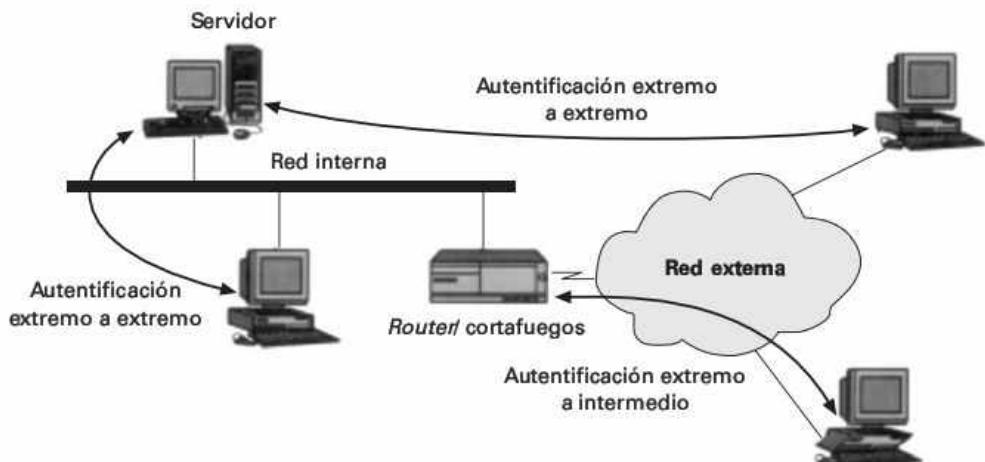


Figura 6.5 Autenticación extremo a extremo frente a extremo a intermedio

estaciones de trabajo clientes; la estación de trabajo puede estar en la misma red que el servidor o en una red exterior. Mientras la estación de trabajo y el servidor comparten una clave secreta protegida, el proceso de autenticación es seguro. Este caso utiliza una SA en modo transporte. En el otro caso, una estación de trabajo remota se autentifica a sí misma ante el cortafuegos colectivo, ya sea para el acceso a toda la red interna o porque el servidor solicitado no permite la característica de autenticación. Este caso utiliza una SA en modo túnel.

En este subapartado, nos fijaremos en el ámbito de autenticación que proporciona AH y la localización de la cabecera de autenticación para los dos modos. Las consideraciones son, en cierto modo, diferentes para IPv4 e IPv6. La Figura 6.6a muestra paquetes típicos IPv4 e IPv6. En este caso, la carga útil IP es un segmento TCP; también podría ser una unidad de datos para cualquier otro protocolo que use IP, como UDP o ICMP.

Para **AH en modo transporte** usando IPv4, la AH se inserta después de la cabecera IP original y antes de la carga útil de IP (por ejemplo, un segmento TCP); esto se ilustra en la parte superior de la Figura 6.6b. La autenticación cubre el paquete completo, excluyendo campos variables de la cabecera IPv4 que estén fijados a cero para el cálculo del MAC.

En el contexto de IPv6, AH se considera una carga útil de extremo a extremo; es decir, no la examinan ni la procesan routers intermedios. Por lo tanto, la AH aparece después de la cabecera base IPv6 y las cabeceras de extensión *salto en salto (hop by hop), enrutamiento y fragmento*. Las cabeceras de extensión de opciones de destino podrían aparecer antes o después de la cabecera AH, dependiendo de la semántica deseada. Nuevamente, la autenticación cubre el paquete completo, excluyendo campos variables que se fijan a cero para el cálculo del MAC.

Para **AH en modo túnel**, se autentica el paquete IP original completo y la AH se inserta entre la cabecera IP original y la nueva cabecera IP externa (Figura 6.6c). La cabecera IP interna contiene las direcciones finales de origen y destino, mientras que una cabecera IP externa puede contener diferentes direcciones IP (por ejemplo, direcciones de cortafuegos u otras pasarelas de seguridad).

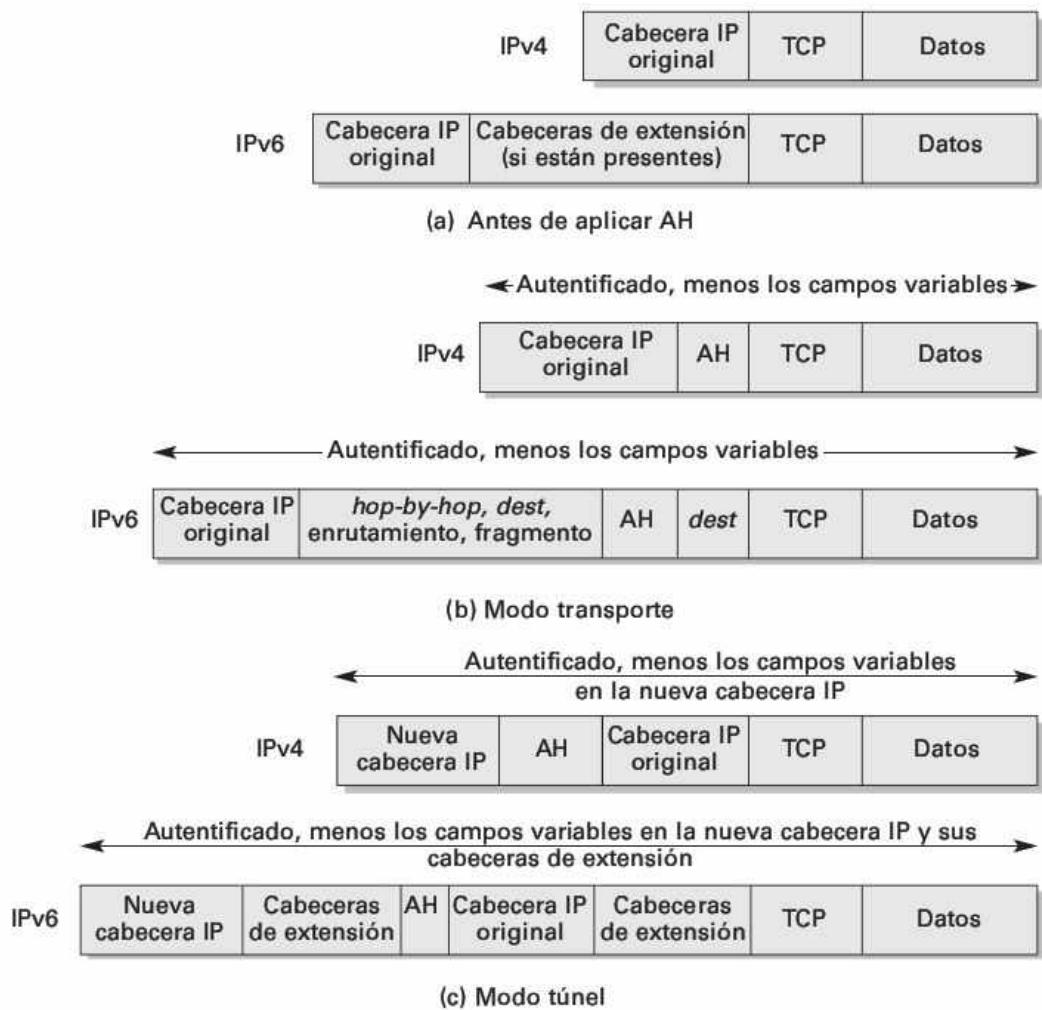


Figura 6.6 Ámbito de autentificación de AH

Con el modo túnel, AH protege el paquete IP interior completo, incluyendo la cabecera IP interior completa. La cabecera IP exterior (y en el caso de IPv6, las cabeceras de extensión IP externas) se protege, a excepción de los campos variables e impredecibles.

6.4 ENCAPSULAMIENTO DE LA CARGA ÚTIL DE SEGURIDAD

El encapsulamiento de la carga útil de seguridad proporciona servicios de confidencialidad, incluyendo confidencialidad del contenido de los mensajes y confidencialidad limitada del flujo de tráfico. Como característica opcional, ESP también puede ofrecer los mismos servicios de autentificación que AH.

EL FORMATO ESP

La Figura 6.7 muestra el formato de un paquete ESP. Contiene los siguientes campos:

- **Índice de parámetros de seguridad (32 bits):** identifica una asociación de seguridad.
- **Número de secuencia (32 bits):** el valor de un contador que se incrementa monótonamente; proporciona la función antirrepetición, como se explicó para AH.
- **Datos de carga útil (variable):** es un segmento de la capa de transporte (modo de transporte) o un paquete IP (modo túnel) protegido mediante cifrado.
- **Relleno (0-255 bytes):** la finalidad de este campo se trata más tarde.

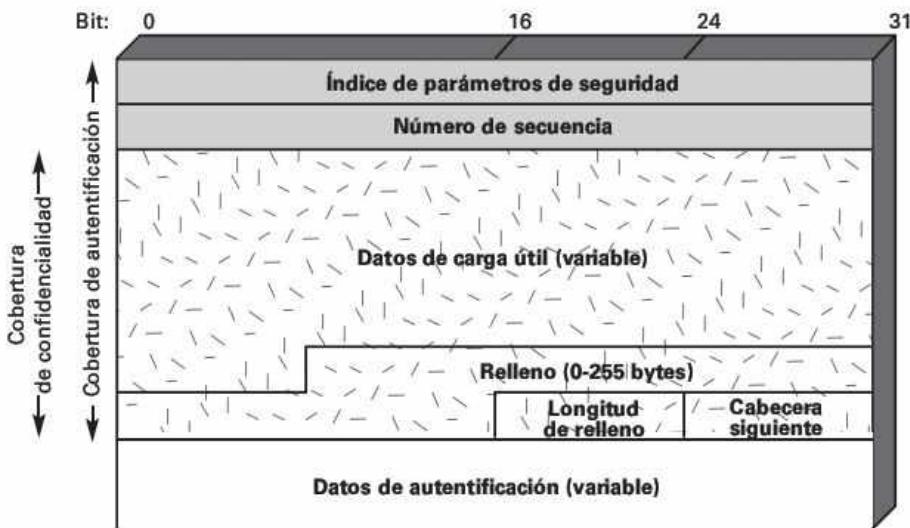


Figura 6.7 Formato ESP de IPSec

- **Longitud de relleno (ocho bits):** indica el número de bytes de relleno en el campo inmediatamente anterior.
- **Cabecera siguiente (ocho bits):** identifica el tipo de datos que contiene el campo de datos de carga útil identificando la primera cabecera en esa carga útil (por ejemplo, una cabecera de extensión en IPv6 o un protocolo de la capa superior como TCP).
- **Datos de autentificación (variable):** un campo de longitud variable (debe ser un número entero de palabras de 32 bits) que contiene el *valor de comprobación de integridad* calculado sobre el paquete ESP menos el campo de datos de autenticación.

ALGORITMOS DE CIFRADO Y AUTENTIFICACIÓN

Los campos *datos de carga útil*, *relleno*, *longitud de relleno* y *cabecera siguiente* se cifran mediante el servicio ESP. Si el algoritmo utilizado para cifrar la carga útil requie-

re datos de sincronización criptográfica, como puede ser un vector de inicialización (IV, *Initialization Vector*), entonces estos datos pueden aparecer explícitamente al principio del campo *datos de carga útil*. Si se incluye, un IV no se cifra normalmente, aunque con frecuencia se considera que es parte del texto cifrado.

La especificación actual dicta que una implementación adecuada debe permitir DES en modo CBC (descrito en el Capítulo 2). A otros algoritmos se les ha asignado identificadores en el documento DOI y podrían, por lo tanto, usarse fácilmente para el cifrado; estos incluyen

- Triple DES de tres claves
- RC5
- IDEA
- Triple IDEA de tres claves
- CAST
- *Blowfish*

Muchos de estos algoritmos se describen en el Capítulo 2.

Como con AH, ESP permite el uso de un MAC con una longitud predeterminada de 96 bits. También como con AH, la especificación actual dicta que una implementación adecuada debe permitir HMAC-MD5-96 y HMAC-SHA-1-96.

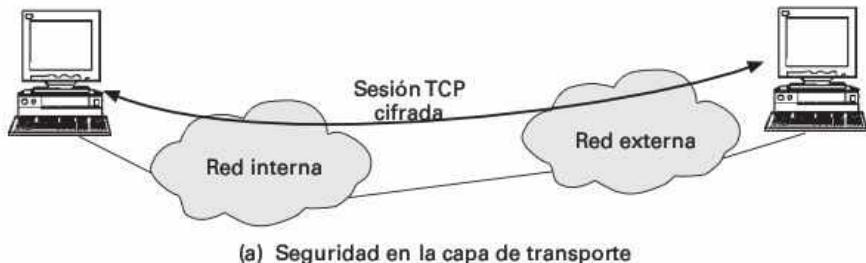
RELENO

El campo de *relleno* sirve para los siguientes propósitos:

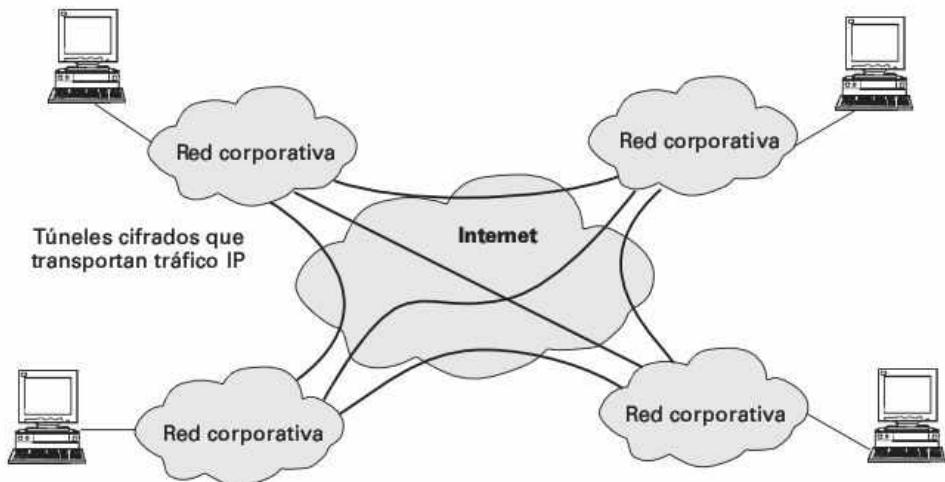
- Si un algoritmo de cifrado requiere que el texto claro sea un múltiplo de un número de bytes (por ejemplo, el múltiplo de un solo bloque para un cifrador de bloque), el campo de *relleno* se usa para expandir el texto claro (que consiste en los campos *datos de carga útil*, *relleno*, *longitud de relleno* y *cabecera siguiente*) para alcanzar la longitud necesaria.
- El formato ESP requiere que los campos *longitud de relleno* y *cabecera siguiente* estén correctamente alineados en una palabra de 32 bits. El campo *relleno* se usa para garantizar esta alineación.
- Se puede añadir relleno adicional para proporcionar confidencialidad parcial del flujo de tráfico, disimulando la longitud real de la carga útil.

MODO TRANSPORTE Y MODO TÚNEL

La Figura 6.8 muestra dos formas de utilización del servicio ESP de IPSec. En la parte superior de la figura, el cifrado (y de manera opcional, la autenticación) se proporciona directamente entre dos *hosts*. La Figura 6.8b ilustra cómo se puede usar la operación del modo túnel para establecer una *red virtual privada*. En este ejemplo, una organización tiene cuatro redes privadas conectadas entre sí en Internet. Los *hosts* de las redes internas usan Internet para el transporte de datos pero no interactúan con otros *hosts* basados en Internet. Finalizados los túneles en la pasarela de seguridad para cada red interna, la configuración permite que los *hosts* eviten la implementación de la capacidad de seguridad. La primera técnica usa una SA en modo transporte, mientras que la segunda se vale de una SA en modo túnel.



(a) Seguridad en la capa de transporte



(b) Red virtual privada en modo túnel

Figura 6.8 Cifrado en modo transporte frente a cifrado en modo túnel

En esta sección, nos centramos en el ámbito de ESP para los dos modos. Las consideraciones varían de IPv4 a IPv6. Al igual que con el ámbito de AH, usaremos los formatos de paquetes de la Figura 6.6a como punto de inicio.

ESP en modo transporte

ESP en modo transporte se usa para cifrar y, de manera opcional, para autenticar los datos transportados por IP (por ejemplo, un segmento TCP), como se puede observar en la Figura 6.9a. Para este modo, usando IPv4, la cabecera ESP se inserta en el paquete IP inmediatamente antes de la cabecera de la capa de transporte (por ejemplo, TCP, UDP, ICMP) y después del paquete IP se coloca una terminación ESP (los campos *relleno*, *longitud de relleno* y *cabecera siguiente*); si se elige autenticación, se añade el campo *datos de autenticación* ESP después de la terminación ESP. Se cifran el segmento entero de la capa de transporte más la terminación ESP. La autenticación cubre todo el texto cifrado más la cabecera ESP.

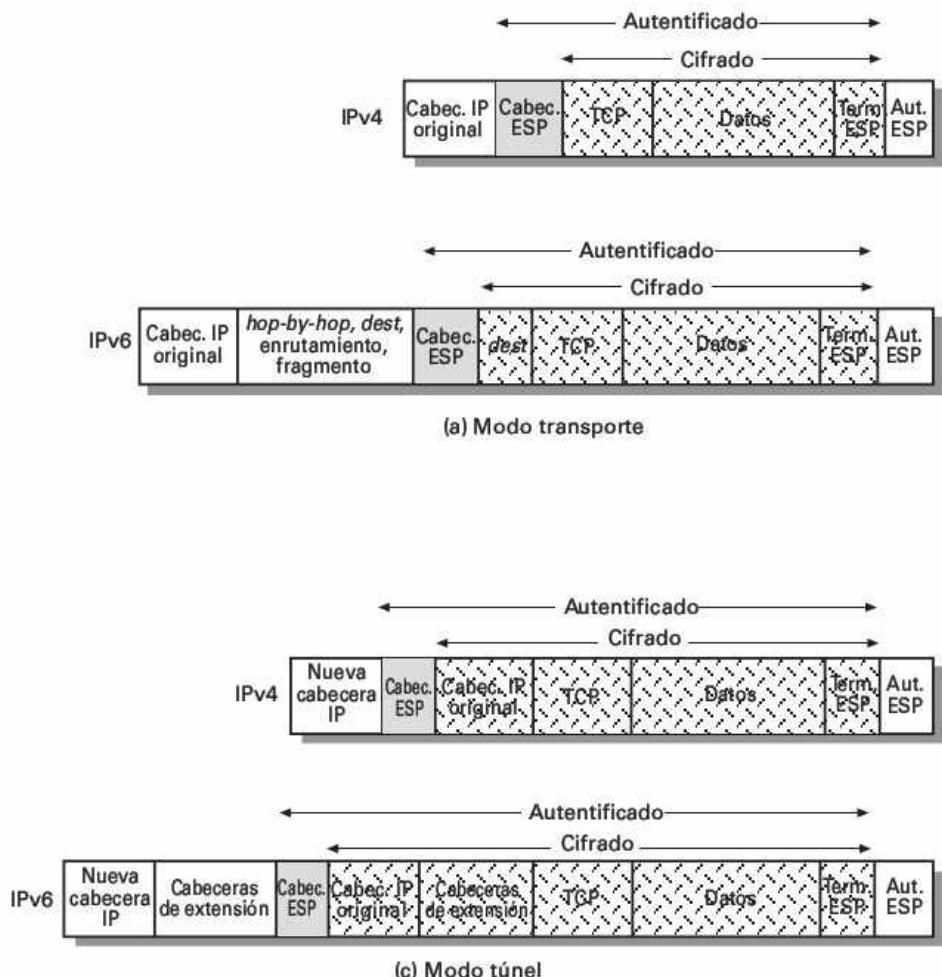


Figura 6.9 Ámbito de cifrado y autentificación de ESP

En el contexto de IPv6, ESP se considera una carga útil de extremo a extremo; es decir, no se examina ni se procesa por *routers* intermedios. Por lo tanto, la cabecera ESP aparece después de la cabecera base IPv6 y las cabeceras de extensión *hop-by-hop, enrutamiento y fragmento*. La cabecera de *extensión de opciones de destino* podría aparecer antes o después de la cabecera ESP, dependiendo de la semántica deseada. Para IPv6, el cifrado cubre el segmento completo del nivel de transporte más la terminación ESP más la cabecera de extensión de opciones de destino si ocurre después de la cabecera ESP. Nuevamente, la autentificación cubre el texto cifrado y la cabecera ESP.

La operación del modo transporte se puede resumir de la siguiente manera:

1. En el origen, el bloque de datos formado por la terminación ESP y el segmento completo de la capa de transporte se cifra y el texto claro de este bloque se sustituye por su texto cifrado para formar el paquete IP para la transmisión. La autentificación se añade si se elige esta opción.

2. Luego, el paquete se encamina al destino. Cada *router* intermedio necesita examinar y procesar la cabecera IP y cualquier cabecera de extensión IP de texto claro, pero no necesita examinar el texto cifrado.
3. El nodo de destino examina y procesa la cabecera IP más cualquier cabecera de extensión IP de texto claro. Luego, basándose en el SPI en la cabecera ESP, el nodo de destino descifra el resto del paquete para recuperar el segmento en texto claro de la capa de transporte.

La operación del modo de transporte proporciona confidencialidad para cualquier aplicación que lo use, evitando, de esta forma, la necesidad de implementar confidencialidad en cada aplicación individual. Este modo de operación también es razonablemente eficaz, añadiendo una parte pequeña a la longitud total del paquete IP. Una desventaja de este modo es que es posible analizar el tráfico en los paquetes transmitidos.

ESP en modo túnel

ESP en modo túnel se utiliza para cifrar un paquete IP entero (Figura 6.9b). Para este modo, la cabecera ESP se antepone al paquete y luego se cifran el paquete y la terminación ESP. Este método se puede usar para contrarrestar el análisis del tráfico.

Como la cabecera IP contiene la dirección de destino y, posiblemente, información sobre las directivas de enrutamiento de origen y la opción *salto en salto*, no es posible simplemente transmitir el paquete IP cifrado precedido de la cabecera ESP. Los *routers* intermedios no podrían procesar un paquete así. Por consiguiente, es necesario encapsular el bloque completo (cabecera SP más texto cifrado más datos de autenticación, en caso de estar presentes) con una nueva cabecera IP que contenga suficiente información para el enrutamiento pero no para el análisis de tráfico.

Mientras el modo transporte es adecuado para proteger conexiones entre *hosts* que permiten la característica ESP, el modo túnel es útil en una configuración que incluye un cortafuegos u otro tipo de pasarela de seguridad que protege una red fiable frente a redes externas. En este último caso, el cifrado se produce sólo entre un *host* externo y la pasarela de seguridad, o entre dos pasarelas de seguridad. Esto libera a los *hosts* de la red interna de la carga de procesamiento del cifrado y simplifica la distribución de claves reduciendo el número de claves necesarias. Además, dificulta el análisis del tráfico basado en el destino final.

Consideremos el caso en el que un *host* externo desea comunicarse con un *host* en una red interna protegida por un cortafuegos, y en el que ESP se implementa en el *host* externo y los cortafuegos. Para la transferencia de un segmento de la capa de transporte del *host* externo al interno se llevan a cabo los siguientes pasos:

1. El origen prepara un paquete IP interno con una dirección de destino del *host* interno de destino. Este paquete está precedido de una cabecera ESP; luego, el paquete y la terminación ESP se cifran y se pueden añadir los datos de autenticación. El bloque resultante se encapsula con una nueva cabecera IP (cabecera base más extensiones opcionales como las opciones de enrutamiento y salto en salto para IPv6) cuya dirección de destino es el cortafuegos; esto conforma el paquete externo IP.

- 2 El paquete externo se encamina al cortafuegos destino. Cada *router* intermedio necesita examinar y procesar la cabecera IP externa más cualquier cabecera externa de extensión IP, pero no necesita examinar el texto cifrado.
- 3 El cortafuegos destino examina y procesa la cabecera IP externa y cualquier cabecera de extensión IP externa. Luego, basándose en el SPI de la cabecera ESP, el nodo de destino descifra el resto del paquete para recuperar el paquete IP interior en texto claro. Este paquete, luego, se transmite a la red interna.
- 4 El paquete interno se encamina a través de cero o más *routers* en la red interna hacia el *host* de destino.

6.5 COMBINACIÓN DE ASOCIACIONES DE SEGURIDAD

Una SA individual puede implementar el protocolo AH o el ESP pero no los dos. A veces, un flujo de tráfico particular pedirá los servicios proporcionados por AH y ESP. Además, un flujo de tráfico particular puede requerir servicios IPSec entre *hosts* y, para ese mismo flujo, servicios separados entre pasarelas de seguridad, como, por ejemplo, cortafuegos. En todos estos casos, se deben emplear varias SA para el mismo flujo de tráfico con el objetivo de conseguir los servicios IPSec deseados. El término *grupo de asociaciones de seguridad* se refiere a una secuencia de asociaciones de seguridad a través de las cuales se debe procesar el tráfico para proporcionar un conjunto de servicios IPSec. Las SA en un grupo pueden finalizar en distintos extremos o los mismos.

Las asociaciones de seguridad se pueden combinar de dos formas:

- **Transporte adyacente:** se refiere a la aplicación de más de un protocolo de seguridad al mismo paquete IP, sin invocar el modo túnel. Este enfoque para la combinación de AH y ESP permite un solo nivel de combinación; un mayor grado de anidamiento no produce beneficios adicionales ya que el procesamiento se realiza en una instancia IPSec: el destino (final).
- **Anidamiento de túneles:** se refiere a la aplicación de varias capas de protocolos de seguridad mediante modo túnel IP. Este enfoque permite múltiples niveles de anidamiento, ya que cada túnel puede originarse o terminar en un sitio IPSec diferente a lo largo del recorrido.

Los dos enfoques se pueden combinar, por ejemplo, haciendo que una SA de transporte entre *hosts* viaje parte del camino a través de una SA túnel entre pasarelas de seguridad.

Un aspecto interesante que surge al considerar los grupos de SA es el orden en el que se puede aplicar la autenticación y el cifrado entre un par dado de extremos finales y las formas de hacerlo. A continuación examinaremos este aspecto y luego se observarán combinaciones de SA que implican al menos un túnel.

AUTENTIFICACIÓN MÁS CONFIDENCIALIDAD

El cifrado y la autenticación se pueden combinar para transmitir un paquete IP con confidencialidad y autenticación entre *hosts*. Analizaremos varios enfoques.

Opción ESP con autentificación

Este enfoque se ilustra en la Figura 6.9. En él, el usuario aplica ESP a los datos que se han de proteger y luego añade el campo de datos de autentificación. Existen dos casos:

- **ESP en modo transporte:** la autentificación y el cifrado se aplican a la carga útil de IP enviada al *host*, pero la cabecera IP no está protegida.
- **ESP en modo túnel:** la autentificación se aplica al paquete IP completo enviado a la dirección IP externa de destino (por ejemplo, un cortafuegos), y la autentificación se realiza en ese destino. El paquete IP interno completo está protegido por un mecanismo de privacidad, para su envío al destino IP interno.

Para los dos casos, la autentificación se aplica al texto cifrado, en vez de al texto claro.

Transporte adyacente

Otra forma de aplicar autentificación después del cifrado es usar dos SA de transporte agrupadas, donde la interna es una SA de ESP y la externa una SA de AH. En este caso, ESP se usa sin su opción de autentificación. Debido a que la SA interna es una SA de transporte, el cifrado se aplica a la carga útil IP. El paquete resultante está formado por una cabecera IP (y posiblemente extensiones de cabecera IPv6) seguida de un ESP. Luego, se aplica AH en modo transporte, para que la autentificación cubra el ESP más la cabecera IP original (y extensiones) excepto los campos variables. La ventaja de este enfoque con respecto a usar simplemente una sola SA de ESP con la opción de autentificación ESP reside en que la autentificación abarca más campos, incluyendo las direcciones IP de origen y de destino. La desventaja se halla en el uso de dos SA en vez de una.

Grupo túnel/transporte

El uso de autentificación antes del cifrado podría ser preferible por varios motivos. En primer lugar, como los datos de autentificación están protegidos mediante cifrado, es imposible que nadie intercepte el mensaje y altere los datos de autentificación sin ser detectado. Segundo, se puede desear almacenar información de autentificación con el mensaje en el destino para una referencia posterior. Es más conveniente hacer esto si la información de autentificación se aplica al mensaje sin cifrar; de otro modo, el mensaje tendría que volver a cifrarse para verificar la información de autentificación.

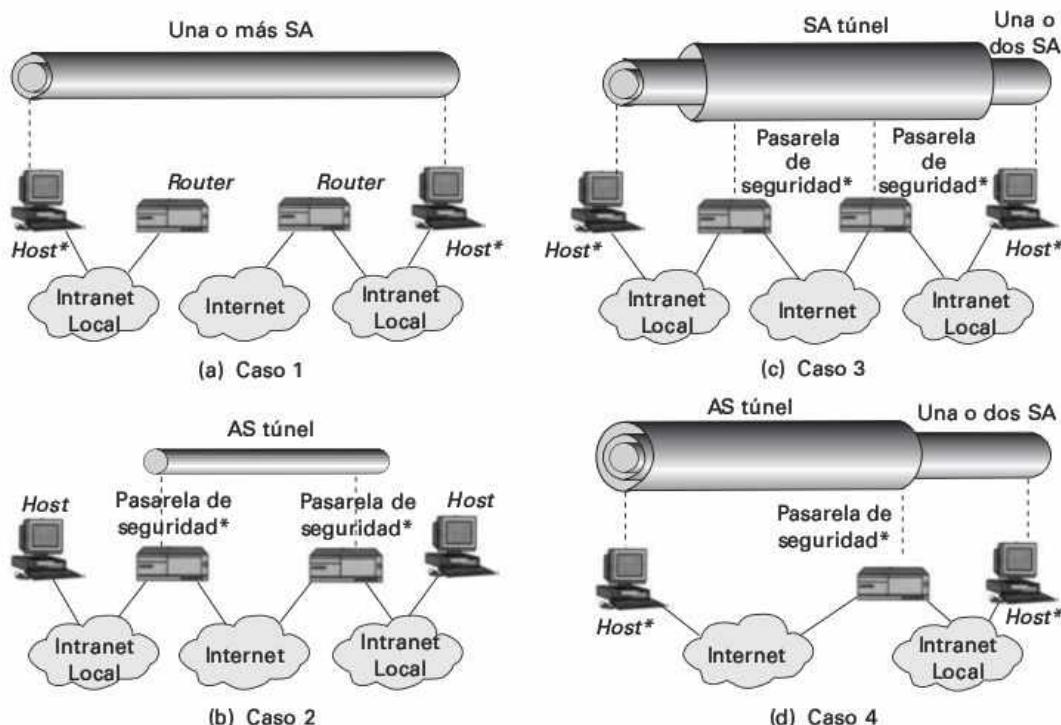
Un enfoque para la aplicación de autentificación antes del cifrado entre dos *hosts* es usar un grupo formado por una SA de transporte AH interna y una SA túnel ESP externa. En este caso, la autentificación se aplica a la carga útil IP más la cabecera IP (y extensiones) a excepción de los campos variables. El paquete IP resultante luego se procesa en modo túnel por ESP; el resultado es que todo el paquete interno autenticado se cifra y se añade una nueva cabecera IP externa (y extensiones).

COMBINACIONES BÁSICAS DE ASOCIACIONES DE SEGURIDAD

El documento de arquitectura IPSec presenta cuatro ejemplos de combinaciones de SA que deben ser soportadas por *hosts* IPSec adecuados (por ejemplo, estación de trabajo,

servidor) o pasarelas de seguridad (por ejemplo, cortafuegos, *router*), que se ilustran en la Figura 6.10. La parte inferior de cada caso en la figura representa la conexión física de los elementos; la parte superior representa la conexión lógica mediante una o más SA anidadas. Cada SA puede ser tanto AH o ESP. Para asociaciones de seguridad *host a host*, el modo puede ser transporte o túnel; si no, debe ser modo túnel.

En el **caso 1**, toda la seguridad se proporciona entre sistemas finales que implementan IPSec. Para que dos sistemas finales se comuniquen mediante una SA, deben compartir las claves secretas apropiadas. Las siguientes son algunas de las posibles combinaciones:



* = implementa IPSec

Figura 6.10 Combinación básica de asociaciones de seguridad

- a AH en modo transporte
- b ESP en modo túnel
- c AH seguida de ESP en modo transporte (una SA ESP dentro de una SA AH)
- d Cualquiera de a, b o c dentro de una AH o ESP en modo túnel.

Ya hemos discutido cómo se pueden usar estas combinaciones para permitir autenticación, cifrado, autenticación antes del cifrado y autenticación después del cifrado.

Para el **caso 2**, la seguridad se proporciona sólo entre pasarelas (*routers*, cortafuegos, etc.) y ningún *host* implementa IPSec. Este caso ilustra el modo túnel en una red virtual

privada. El documento de la arquitectura de seguridad especifica que sólo se necesita una SA túnel para este caso. El túnel podría permitir AH, ESP o E^SP con la opción de autenticación. No se requieren túneles anidados porque los servicios IPSec se aplican al paquete interno completo.

El **caso 3** se construye sobre el caso 2 añadiendo seguridad de extremo a extremo. Se permiten las mismas combinaciones discutidas para los casos 1 y 2. El túnel de pasarela a pasarela proporciona autenticación o confidencialidad o ambos para todo el tráfico entre sistemas finales. Cuando el túnel de pasarela a pasarela es ESP, también proporciona una forma limitada de confidencialidad del tráfico. Los *hosts* individuales pueden implementar cualquier servicio IPSec adicional requerido para ciertas aplicaciones o para ciertos usuarios por medio de asociaciones de seguridad de extremo a extremo.

El **caso 4** proporciona soporte para un *host* remoto que usa Internet para llegar al cortafuegos de una organización y luego acceder a algún servidor o estación de trabajo detrás del cortafuegos. Sólo se requiere el modo túnel entre el *host* remoto y el cortafuegos. Como en el caso 1, se pueden usar una o dos SA entre el *host* remoto y el *host* local.

6.6 GESTIÓN DE CLAVES

La parte de gestión de claves de IPSec implica la determinación y distribución de claves secretas. Un requisito habitual es el de cuatro claves para la comunicación entre dos aplicaciones; parejas de transmisión y recepción tanto para AH como para ESP. El documento de la arquitectura de IPSec asigna soporte para dos tipos de gestión de claves:

- **Manual:** un administrador de sistema configura manualmente cada sistema con sus propias claves y con las claves de otros sistemas que se comunican. Esto es práctico para entornos pequeños relativamente estáticos.
- **Automática:** un sistema automático permite la creación bajo demanda de claves para asociaciones de seguridad y facilita el uso de claves en un sistema distribuido grande con una configuración cambiante.

El protocolo de gestión de claves automático predeterminado para IPSec se conoce como ISAKMP/Oakley y se compone de los siguientes elementos:

- **Protocolo de determinación de claves Oakley:** Oakley es un protocolo de intercambio de claves que se basa en el algoritmo Diffie-Hellman pero que proporciona seguridad adicional. Oakley es genérico en el sentido de que no dicta formatos específicos.
- **Asociación de seguridad y Protocolo de gestión de claves (ISAKMP, Internet Security Association and Key Management Protocol):** ISAKMP proporciona un marco de trabajo para la gestión de claves en Internet y el soporte del protocolo específico, incluyendo formatos, para la negociación de los atributos de seguridad.

ISAKMP por sí mismo no impone un algoritmo específico de intercambio de claves; consiste en un conjunto de tipos de mensaje que permiten el uso de una variedad de algoritmos de intercambio de claves. Oakley es el algoritmo específico de intercambio de claves de uso obligado con la versión inicial de ISAKMP.

Empecemos con una descripción general de Oakley para pasar posteriormente al ISAKMP.

PROTOCOLO DE DETERMINACIÓN DE CLAVES OAKLEY

Oakley es una mejora del algoritmo de intercambio de claves Diffie-Hellman. Recorremos que Diffie-Hellman implica la siguiente interacción entre los usuarios A y B. Hay un acuerdo previo sobre dos parámetros globales: q , un número primo grande; y α , una raíz primitiva de q . A elige un entero aleatorio X_A como su clave privada, y transmite a B su clave pública $Y_A = \alpha^{X_A} \text{ mod } q$. De la misma forma, B elige un entero aleatorio X_B como su clave privada, y transmite a A su clave pública $Y_B = \alpha^{X_B} \text{ mod } q$. Ahora, cada parte puede calcular la clave secreta de sesión:

$$K = (Y_B)^{X_A} \text{ mod } q = (Y_A)^{X_B} \text{ mod } q = \alpha^{X_A X_B} \text{ mod } q$$

El algoritmo Diffie-Hellman tiene dos características interesantes:

- Las claves secretas se crean sólo cuando es necesario. No hay que almacenar claves secretas durante un período largo de tiempo exponiéndolas a vulnerabilidad.
- El intercambio no requiere una infraestructura preexistente, sino un acuerdo sobre los parámetros globales.

Sin embargo, hay una serie de debilidades en Diffie-Hellman, tal y como se señala en [HUIT98]:

- No proporciona información sobre las identidades de las partes.
- Está expuesto al ataque por interceptación (*man-in-the-middle*), en el que una tercera parte C suplanta a B mientras se comunica con A y suplanta a A mientras se comunica con B. Tanto A como B acaban en la negociación de una clave con C, que puede observar el tráfico y conducirlo entre A y B. El ataque por interceptación se produce como sigue:
 1. B envía su clave pública Y_B en un mensaje dirigido a A (ver Figura 3.11).
 2. El enemigo (E) intercepta este mensaje, guarda la clave pública de B y envía un mensaje a A que tiene el identificador de usuario de B, pero la clave pública de E es Y_E . Este mensaje se envía de forma que parezca enviado por el sistema de B. A recibe el mensaje de E y almacena la clave pública de E con el identificador de usuario de B. De la misma forma, E envía un mensaje a B con la clave pública de E, fingiendo que proviene de A.
 3. B calcula una clave secreta K_1 basada en la clave privada de B y en Y_E . A calcula una clave secreta K_2 basada en la clave privada de A y en Y_E . E calcula K_1 usando la clave secreta de E X_E e Y_B y calcula K_2 usando X_E e Y_A .
 4. De ahora en adelante, E puede retransmitir mensajes de A a B y de B a A, cambiando de forma adecuada sus cifrados en ruta de forma que ni A ni B sabrán que están compartiendo su comunicación con E.
- Requiere un elevado número de cálculos. Como resultado, es vulnerable al ataque de obstrucción (*clogging*), en el que un oponente solicita un gran número de claves. La víctima gasta recursos considerables de computación haciendo exponentiación modular inútil, en vez de trabajo real.

Oakley está diseñado para tener las ventajas de Diffie-Hellman y contrarrestar sus puntos débiles.

Características de Oakley

El algoritmo Oakley se caracteriza por cinco aspectos importantes:

1. Emplea un mecanismo conocido como *cookies* para impedir los ataques de obstrucción.
2. Permite que las dos partes negocien un *grupo*: éste, básicamente, especifica los parámetros globales del intercambio de claves de Diffie-Hellman.
3. Usa valores aleatorios (*nonce*) para proteger de ataques de repetición.
4. Permite el intercambio de valores de clave pública de Diffie-Hellman.
5. Autentifica el intercambio Diffie-Hellman para evitar ataques de interceptación.

Ya se ha tratado Diffie-Hellman. Pasemos ahora a cada uno de los demás elementos. En primer lugar, consideremos el problema de los ataques de obstrucción. En este ataque, un oponente falsifica la dirección origen de un usuario legítimo y envía una clave pública Diffie-Hellman a la víctima. Luego, la víctima realiza una exponenciación modular para calcular la clave secreta. Mensajes repetidos de este tipo pueden *obstaculizar* el sistema de la víctima con trabajo inútil. El **intercambio de cookies** requiere que cada parte envíe un número pseudoaleatorio, la *cookie*, en el mensaje inicial, que la otra parte reconoce. Este reconocimiento debe repetirse en el primer mensaje del intercambio de claves Diffie-Hellman. Si la dirección fuente fue falsificada, el oponente no obtiene respuesta. Así, un oponente sólo puede forzar a un usuario a generar reconocimientos y no a realizar el cálculo Diffie-Hellman.

ISAKMP obliga a que la generación de *cookies* satisfaga tres requisitos básicos:

1. La *cookie* debe depender de las partes específicas. Esto evita que un atacante obtenga una *cookie* usando una dirección IP y un puerto UDP reales y usándola luego para inundar a la víctima con solicitudes de direcciones IP o puertos elegidos de forma aleatoria.
2. No debe ser posible que alguien que no sea la entidad emisora genere *cookies* que sean aceptadas por esa entidad. Esto implica que la entidad emisora usará información secreta local en la generación y posterior verificación de una *cookie*. No debe ser posible deducir esta información secreta a partir de una *cookie* particular. La importancia de este requisito se halla en que la entidad emisora no necesita guardar copias de sus *cookies*, en cuyo caso serían más vulnerables, pero puede verificar el reconocimiento de una *cookie* entrante cuando sea necesario.
3. Los métodos de generación y verificación de *cookies* deben ser rápidos para evitar ataques que intenten sabotear los recursos del procesador.

El método recomendado para crear *cookies* es realizar un *hash* rápido (por ejemplo, MD5) sobre las direcciones IP de origen y de destino, los puertos UDP fuente y destino y un valor secreto generado localmente.

Oakley permite el uso de diferentes **grupos** para el intercambio de claves Diffie-Hellman. Cada grupo incluye la definición de los dos parámetros globales y la identidad del algoritmo. La actual especificación incluye los siguientes grupos:

- Exponenciación modular con módulo de 768 bits

$$q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149686)$$

$$\alpha = 2$$

- Exponenciación modular con módulo de 1024 bits

$$q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129093)$$

$$\alpha = 2$$

- Exponenciación modular con módulo de 1536 bits

- Parámetros que deben determinarse

- Grupo de curvas elípticas sobre 2^{155}

- Generador (hexadecimal): X = 7B, Y = 1C8

- Parámetros de curva elíptica (hexadecimal): A = 0, Y = 7338F

- Grupo de curvas elípticas sobre 2^{185}

- Generador (hexadecimal): X = 18, Y = D

- Parámetros de curva elíptica (hexadecimal): A = 0, Y = 1EE9

Los primeros tres grupos son el algoritmo clásico de Diffie-Hellman usando exponenciación modular. Los últimos dos grupos usan la curva elíptica análoga a la de Diffie-Hellman.

Oakley usa **valores aleatorios (nonces)** para proteger de los ataques de repetición. Cada *nonce* es un número pseudoaleatorio generado localmente. Los *nonces* aparecen en respuestas y se cifran durante ciertas partes del intercambio para asegurar su uso.

Se pueden usar tres métodos de **autentificación** con Oakley:

- **Firmas digitales:** el intercambio se autentifica firmando un *hash* obtenible mutuamente; cada parte cifra el *hash* con su clave privada. El *hash* se genera usando parámetros importantes como identificadores de usuario y *nonces*.
- **Cifrado de clave pública:** el intercambio se autentifica cifrando parámetros como identificadores y *nonces* con la clave privada del emisor.
- **Cifrado de clave simétrica:** se puede usar una clave procedente de algún mecanismo fuera de banda para autenticar el intercambio mediante el cifrado simétrico de los parámetros de intercambio.

Ejemplo de intercambio Oakley

La especificación Oakley incluye una serie de ejemplos de intercambios que están permitidos por el protocolo. Para dar una idea de Oakley, presentamos un ejemplo, llamado *intercambio agresivo de clave* en la especificación, porque sólo se intercambian tres mensajes.

La Figura 6.11 muestra el protocolo de intercambio agresivo de claves. En el primer paso, el iniciador (I) transmite una *cooky*, el grupo que se va a usar, y la clave pública

I → R: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, S_{KI}[ID_I || ID_R || N_I || GRP || g^x || EHAO]
 R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, S_{KR}[ID_R || ID_I || N_R || N_I || GRP || g^y || g^x || EHAS]
 I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, S_{KI}[ID_I || ID_R || N_I || N_R || GRP || g^x || g^y || EHAS]

Notación:

I = Iniciador

R = Replicante

CKY_I, CKY_R = Cookies de iniciador y replicante

OK_KEYX = Tipo de mensaje de intercambio de claves

GRP = Nombre del grupo Diffie-Hellman para este intercambio

g^x, g^y = clave pública del iniciador y del replicante; g^{xy} = clave de sesión de este intercambio

EHAO, EHAS = Funciones de cifrado, hash y de autentificación ofrecidas y seleccionadas

NIDP = indica que no se usa cifrado para el resto del mensaje

ID_I, ID_R = Identificador del iniciador y del replicante

N_I, N_R = nonce aleatorio suministrado por el iniciador o el replicante para este intercambio

S_{KI}[X], S_{KR}[X] = Indica la firma sobre X usando la clave privada (clave de firma) del iniciador, o del replicante

Figura 6.11 Ejemplo de intercambio agresivo de claves Oakley

Diffie-Hellman de I para este intercambio. También indica los algoritmos ofrecidos de cifrado de clave pública, hash y de autentificación que se van a usar en este intercambio. Además, se incluyen en el mensaje los identificadores de I y del replicante (R) y el nonce de I para este intercambio. Por último, I añade una firma usando la clave privada de I que firma los dos identificadores, el nonce, el grupo, la clave pública Diffie-Hellman y los algoritmos ofrecidos.

Cuando R recibe el mensaje, verifica la firma usando la clave pública de firma de I. R reconoce el mensaje devolviendo la cookie de I, el identificador y el nonce, así como el grupo. R también incluye una cookie en el mensaje, la clave pública Diffie-Hellman de R, los algoritmos seleccionados (que deben estar entre los algoritmos ofrecidos), el identificador de R y el nonce de R para este intercambio. Por último, R añade una firma usando la clave privada de R que firma los dos identificadores, los dos nonces, el grupo, las dos claves públicas Diffie-Hellman y los algoritmos seleccionados.

Cuando I recibe el segundo mensaje, verifica la firma usando la clave pública de R. Los valores nonce en el mensaje aseguran que no es la repetición de un mensaje antiguo. Para completar el intercambio, I debe enviar un mensaje de vuelta a R para verificar que I ha recibido la clave pública de R.

ISAKMP

ISAKMP define los procedimientos y los formatos de los paquetes para establecer, negociar, modificar y eliminar asociaciones de seguridad. Como parte del establecimiento de la SA, ISAKMP define las cargas útiles para intercambiar la generación de claves y los datos de autentificación. Los formatos de las cargas útiles proporcionan un marco de trabajo consistente independiente del protocolo específico de intercambio de claves, del algoritmo de cifrado y del mecanismo de autentificación.

Formato de la cabecera ISAKMP

Un mensaje ISAKMP se compone de una cabecera ISAKMP seguida de una o más cargas útiles. Todo esto se transfiere en un protocolo de transporte. La especificación obliga a que las implementaciones permitan el uso de UDP para el protocolo de transporte.

La Figura 6.12a muestra el formato de cabecera para un mensaje ISAKMP. Consta de los siguientes campos:

- ***Cookie del iniciador (64 bits)***: *cookie* de la entidad que inició el establecimiento de la SA, notificación de SA o eliminación de la misma.
- ***Cookie del replicante (64 bits)***: *cookie* de la entidad que responde; nula en el primer mensaje del iniciador.
- ***Siguiente carga útil (ocho bits)***: indica el tipo de la primera carga útil del mensaje; las cargas útiles se discuten en el siguiente subapartado.
- ***Versión mayor (cuatro bits)***: indica la versión mayor que se usa del ISAKMP.
- ***Versión menor (cuatro bits)***: indica la versión menor en uso.
- ***Tipo de intercambio (ocho bits)***: indica el tipo de intercambio; se discute más adelante en esta sección.
- ***Indicadores (ocho bits)***: indica las opciones específicas fijadas para este intercambio ISAKMP. Dos bits definidos hasta ahora: el *bit de cifrado (encryption bit)* se fija si todas las cargas útiles que siguen a la cabecera se cifran usando el algoritmo de cifrado para esa SA. El *bit de garantía (commit bit)* se usa para asegurar que el material cifrado no se recibe antes de terminar el establecimiento de la SA.

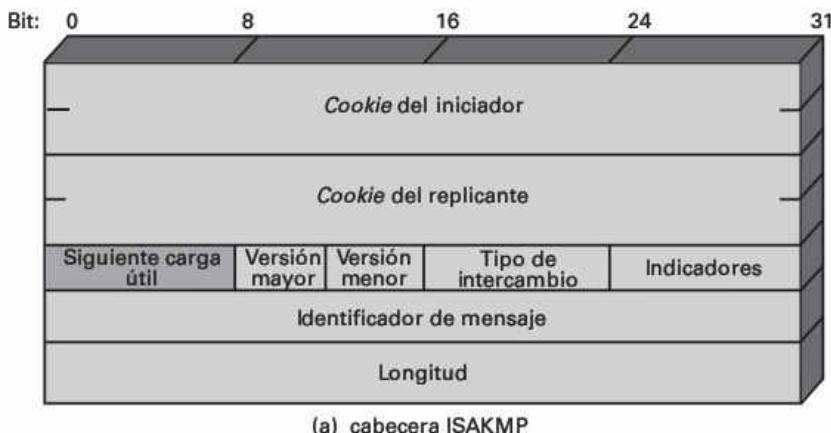


Figura 6.12 Formatos ISAKMP

- **Identificador de mensaje (32 bits):** identificador único para este mensaje.
- **Longitud (32 bits):** longitud del mensaje total (cabecera y todas las cargas útiles) en octetos.

Tipos de carga útil ISAKMP

Todas las cargas útiles ISAKMP empiezan con la misma cabecera genérica de carga útil que muestra la Figura 6.12b. El campo *siguiente carga útil* tiene un valor de 0 si es la última carga útil del mensaje; si no, su valor es el tipo de la carga útil siguiente. El campo *longitud de carga útil* indica la longitud en octetos de esa carga útil, incluyendo la cabecera genérica de carga útil.

La Tabla 6.3 resume los tipos de cargas útiles definidos para ISAKMP, y presenta los campos, o parámetros, que son parte de cada carga útil. La **carga útil de SA** se usa para empezar el establecimiento de una SA. En esta carga útil, el parámetro *dominio de interpretación* identifica el DOI en el que se está llevando a cabo la negociación. El DOI IPSec es un ejemplo, pero ISAKMP puede usarse en otros contextos. El parámetro *situación* define la política de seguridad para esta negociación; básicamente, se especifican los niveles de seguridad requeridos para el cifrado y la confidencialidad (por ejemplo, nivel de confidencialidad, comportamiento de seguridad).

La **carga útil de propuesta** contiene información usada durante la negociación de la SA. La carga útil indica el protocolo para esta SA (ESP o AH) para la cual se están negociando servicios y mecanismos. La carga útil también incluye el SPI de la entidad emisora y el número de transformaciones. Cada transformación está contenida en una carga útil de transformaciones. El uso de varias cargas útiles de transformaciones permite que el iniciador ofrezca varias posibilidades, de las cuales el replicante debe elegir una o rechazar la oferta.

La **carga útil de la transformación** define una transformación de seguridad que se debe usar para asegurar el canal de comunicaciones para el protocolo designado. El parámetro *número de transformaciones* sirve para identificar esta carga útil concreta con el objetivo de que el replicante pueda usarlo para indicar la aceptación de esta transformación. Los campos *identificación de transformación* y *atributos* identifican una transformación específica (por ejemplo, 3DES para ESP, HMAC-SHA-1-96 para AH) con sus atributos asociados (por ejemplo, longitud de *hash*).

La **carga útil del intercambio de claves** se puede usar para una variedad de técnicas de intercambio de claves, incluyendo Oakley, Diffie-Hellman y el intercambio de claves basado en RSA que usa PGP. El campo de datos de *Intercambio de claves* contiene los datos necesarios para generar una clave de sesión y depende del algoritmo de intercambio de claves que se ha utilizado.

La **carga útil de identificación** se usa para determinar la identidad de las entidades que se comunican y se puede usar para determinar la autenticidad de la información. Normalmente, el campo *datos de identificación* contiene una dirección IPv4 o IPv6.

Tabla 6.3 Tipos de carga útil ISAKMP

Tipo	Parámetros	Descripción
Asociación de seguridad (SA).	Dominio de interpretación, situación.	Se usa para negociar atributos de seguridad e indicar el DOI y la Situación en la que tiene lugar la negociación.
Propuesta (P)	Número de propuestas, identificador de protocolo, número de SPI, nº de transformaciones, SPI.	Se usa durante la negociación de SA; indica el protocolo que ha de usarse y un número de transformaciones.
Transformación (T)	Número de transformaciones, identificación de transformaciones, atributos de la SA.	Se usa durante la negociación de la SA; indica los atributos de transformaciones y los relacionados con las SA.
Intercambio de clave (KE)	Datos de intercambio de claves.	Permite una variedad de técnicas de intercambio de claves.
Identificación (ID)	Tipo de identificador, datos de identificadores.	Se usa para intercambiar información de identificación.
Certificado (CERT)	Codificación de certificados, datos de certificados.	Se usa para transportar certificados y otra información relacionada.
Solicitud de certificado (CR)	Número de tipos de certificado, tipos de certificados, número de autenticaciones de certificado, autoridades de certificación.	Se usa para solicitar certificados; indica los tipos de certificado solicitados y las autoridades de certificación aceptables.
Hash (HASH)	Datos del hash.	Contiene datos generados por una función <i>hash</i> .
Firma (SIG)	Datos de la firma.	Contiene datos generados por una función de firma digital.
Nonce (NONCE)	Datos del nonce.	Contiene un nonce.
Notificación (N)	DOI, identificación de protocolo, tamaño de SPI, notificar tipo de mensaje, SPI, datos de notificación.	Se usa para transmitir datos de notificación, como condición de error.
Eliminar (D)	DOI, identificación de protocolo, tamaño de SPI, nº de SPI, SPI (uno o más).	Indica una SA que ya no es válida.

La **carga útil del certificado** transfiere un certificado de clave pública. El campo *codificación del certificado* indica el tipo de certificado o la información referente al éste, que puede incluir la siguiente:

- Certificado X.509-PKCS#7
- Certificado PGP
- Clave firmada DNS
- Certificado X.509 - firma
- Certificado X.509 - intercambio de claves
- *Tokens* de Kerberos
- Lista de revocación de certificados (CRL)
- Lista de revocación de autoridades (ARL)
- Certificado SPKI

En cualquier momento del intercambio ISAKMP, el emisor puede incluir una carga útil de **solicitud de certificado** para solicitar el certificado de las otras entidades comunicantes. La carga útil puede presentar más de un tipo de certificado aceptable y más de una autoridad de certificación aceptable.

La **carga útil hash** contiene datos generados por una función *hash* sobre alguna parte del mensaje y/o estado ISAKMP. Esta carga útil puede usarse para verificar la integridad de los datos en un mensaje y para autenticar a las entidades negociadoras.

La **carga útil de firma** contiene datos generados por una función de firma digital sobre alguna parte del mensaje y/o estado ISAKMP. Esta carga útil se usa para verificar la integridad de los datos en un mensaje y puede usarse para servicios de no repudio.

La **carga útil nonce** contiene datos aleatorios que se usan para garantizar la validez temporal durante un intercambio y proteger de ataques de repetición.

La **carga útil de notificación** contiene la información de error o la información de estado asociada con esta SA o esta negociación de SA. Se han definido los siguientes mensajes de error ISAKMP:

Tipo de carga útil inválido	Identificador de protocolo inválido	Codificación de certificado inválida
DOI no permitido	SPI inválido	Certificado inválido
Situación no permitida	Identificador de transformación inválido	Mala sintaxis de solicitud de certificado
<i>Cookie</i> inválida	Atributos no permitidos	Autoridad de certificación inválida
Versión mayor inválida	Ninguna propuesta elegida	Información <i>hash</i> inválida
Versión menor inválida	Mala sintaxis de propuesta	Falló la autenticación
Tipo de intercambio inválido	Carga útil mal formada	Firma inválida
Indicadores inválidos	Información de clave inválida	Notificación de dirección
Identificación de mensaje inválida		

El único mensaje de estado ISAKMP definido hasta ahora es *Conectado*. Además de estas notificaciones ISAKMP, se usan las notificaciones específicas DOI. Para IPSec, se definen los siguientes mensajes adicionales de estado:

- **Tiempo de vida del replicante:** comunica el tiempo de vida de la SA elegido por el replicante.
- **Estado de repetición:** se usa para la confirmación positiva de la elección del replicante en cuanto a si el replicante realizará detección antirrepeticiones o no.
- **Contacto inicial:** informa a la otra parte que ésta es la primera SA establecida con el sistema remoto. Entonces el receptor de esta notificación podría eliminar cualquier SA existente que tenga para el sistema emisor basándose en que el sistema emisor ha reiniciado y ya no tiene acceso a esas SA.

La **carga útil de eliminación** indica una o más SA que el emisor ha eliminado de su base de datos y que, por lo tanto, ya no son válidas.

Intercambios ISAKMP

ISAKMP proporciona un marco de trabajo para el intercambio de mensajes, donde los tipos de carga útil sirven como pilares de construcción. La especificación identifica cinco tipos predeterminados de intercambio que deberían permitirse; éstos se resumen en la Tabla 6.4. En la Tabla, SA se refiere a la carga útil de una SA con cargas útiles asociadas de Protocolo y Autentificación.

El **intercambio base** permite que el material de intercambio de claves y autenticación se transmita junto. Esto minimiza el número de intercambios a costa de no proporcionar protección de identidad. Los dos primeros mensajes proporcionan *cookies* y establecen una SA con el protocolo y las transformaciones acordadas; ambas partes usan un *nonce* para evitar ataques de repetición. Los dos últimos mensajes intercambian el material de clave y los identificadores de usuario, con la carga útil AUTH que se usa para autenticar claves, identidades y los *nonces* de los dos primeros mensajes.

El **intercambio de protección de identidad** expande el *intercambio base* para proteger las identidades de los usuarios. Los dos primeros mensajes establecen la SA. Los dos siguientes mensajes realizan el intercambio de claves, con *nonces* para la protección contra repeticiones. Una vez que la clave de sesión ha sido calculada, las dos partes intercambian mensajes cifrados que contienen información de autenticación, como firmas digitales y, opcionalmente, certificados que validan las claves públicas.

El **intercambio de sólo autenticación** se usa para realizar autenticación mutua, sin intercambio de claves. Los dos primeros mensajes establecen la SA. Además, el replicante usa el segundo mensaje para transportar su identificador y utiliza la autenticación para proteger el mensaje. El iniciador envía el tercer mensaje para transmitir su identificador autenticado.

El **intercambio agresivo** reduce el número de intercambios a costa de no proporcionar protección de identidad. En el primer mensaje, el iniciador propone una SA con opciones ofrecidas de protocolo y transformación asociadas. El iniciador también empieza el intercambio de claves y proporciona su identificador. En el segundo mensaje, el replicante indica su aceptación de la SA con un protocolo y una transformación particulares, completa el intercambio de claves y autentifica la información transmitida. En el tercer mensaje, el iniciador transmite un resultado de autenticación que cubre la información previa, cifrada usando la clave de sesión secreta compartida.

El **intercambio informativo** se usa para la transmisión unidireccional de información para la gestión de SA.

Tabla 6.4 Tipos de intercambio ISAKMP

Intercambio	Nota
(a) Intercambio base	
(1) I → R: SA; NONCE	Empieza la negociación SA ISAKMP
(2) R → I: SA; NONCE	Acuerdo sobre la SA básica
(3) I → R: KE; ID _I ; AUTH	Clave generada; identidad del iniciador verificada por el replicante
(4) R → I: KE; ID _R ; AUTH	Identidad del replicante verificada por el iniciador; clave generada; SA establecida
(b) Intercambio de protección de identidad	
(1) I → R: SA	Empieza la negociación SA-ISAKMP
(2) R → I: SA	Acuerdo sobre la SA básica
(3) I → R: KE; NONCE	Clave generada
(4) R → I: KE; NONCE	Clave generada
(5)* I → R: ID _I ; AUTH	Identidad del iniciador verificada por el replicante
(6)* R → I: ID _R ; AUTH	Identidad del replicante verificada por el iniciador; SA establecida
(c) Intercambio sólo de autentificación	
(1) I → R: SA; NONCE	Empieza la negociación SA-ISAKMP
(2) R → I: SA; NONCE; ID _R ; AUTH	Acuerdo sobre la SA básica; identidad del replicante verificada por el iniciador; SA establecida
(3) I → R: ID _I ; AUTH	Identidad del iniciador verificada por el replicante; SA establecida
(d) Intercambio agresivo	
(1) I → R: SA; KE; NONCE; ID _I	Empieza la negociación SA-ISAKMP y el intercambio de claves
(2) R → I: SA; KE; NONCE; ID _R ; AUTH	Identidad del iniciador verificada por el replicante; clave generada; acuerdo sobre la SA básica
(3)* I → R: AUTH	Identidad del replicante verificada por el iniciador; SA establecida
(e) Intercambio informativo	
(1) * I → R: N/D	Notificación o eliminación de error o estado

Notación:

I = iniciador

R = replicante

* = significa cifrado de carga útil después de la cabecera ISAKMP

6.7 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

IPv6 e IPv4 se tratan más detalladamente en [STAL00]. En [COME00] y [STEV94] se encuentra una buena explicación de la comunicación entre redes o *Internetworking* e IPv4. [HUIT98] proporciona una descripción técnica sencilla de los distintos RFC que forman la especificación IPv6; ofrece una discusión sobre la finalidad de las distintas características y la operación del protocolo. [MILL98] trata también IPv6, resaltando los aspectos prácticos y de implementación. [CHEN98] presenta una buena discusión sobre el diseño de IPSec. [FRAN01] y [DORA99] son análisis más exhaustivos de IPSec.

- CHEN98** Cheng, P., et al. «A Security Architecture for the Internet Protocol." *IBM Systems Journal*, número 1, 1998.
- COME00** Comer, D. *Internetworking with TCP/IP, Volume 1: Principles, Protocols and Architecture*. Upper Saddle River, NJ: Prentice Hall, 2000.
- DORA99** Doraswamy, N., y Harkins, D. *IPSec*. Upper Saddle River, NJ: Prentice Hall, 1999.
- FRAN01** Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.
- HUIT98** Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.
- MILL98** Miller, S. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.
- STAL00** Stallings, W. *Data and Computer Communications*, 6.^a edición. Upper Saddle River, NJ: Prentice Hall, 2000.
- STEV94** Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994.

Sitios web recomendados:

- **Carta del protocolo de seguridad IP (ipsec):** últimos RFC y borradores de Internet para IPSec.
- **Noticias sobre el grupo de trabajo de seguridad IP:** documentos del grupo de trabajo, archivos de correo, artículos técnicos relacionados y otro material útil.
- **Recursos de seguridad IP (IPSEC):** lista de empresas que implementan IPSec, estudio de implementación y otro material útil.

6.7 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Asociación de seguridad (AS)	Encapsulamiento de la carga útil de seguridad (ESP)	Seguridad IP Servicio antirrepeticIÓN
Asociación de seguridad de Internet y protocolo de gestión de claves (ISAKMP)	IPv4 IPv6 Modo transporte Modo túnel	
Ataque de repetición	Protocolo de determinación de claves Oakley	
Cabecera de autentificación (AH)		

PREGUNTAS DE REPASO

- 6.1.** Proporciona algunos ejemplos de aplicaciones de IPSec.
- 6.2.** ¿Qué servicios proporciona IPSec?
- 6.3.** ¿Qué parámetros identifican una SA y cuáles caracterizan la naturaleza de una SA concreta?
- 6.4.** ¿Cuál es la diferencia entre modo transporte y modo túnel?
- 6.5.** ¿Qué es un ataque de repetición?
- 6.6.** ¿Por qué ESP incluye un campo de relleno?
- 6.7.** ¿Cuáles son los enfoque básicos sobre la agrupación de distintas SA?
- 6.8.** ¿Qué papel desempeña el protocolo de determinación de claves Oakley e ISAKMP en IPSec?

PROBLEMAS

- 6.1.** Al tratar el procesamiento de AH, se mencionó que no todos los campos de una cabecera IP se incluyen en el cálculo MAC.
 - a)** Para cada campo de la cabecera IPv4, indica si el campo es invariable, variable pero predecible, o variable (fijado a cero antes del cálculo del ICV).
 - b)** Haz lo mismo para la cabecera IPv6.
 - c)** Haz lo mismo para las cabeceras de extensión IPv6.

En cada caso, justifica tu decisión para cada campo.
- 6.2.** Cuando se usa el modo túnel, se construye una nueva cabecera IP externa. Indica, para IPv4 e IPv6, la relación de cada campo de la cabecera IP externa y cada cabecera de extensión del paquete externo con el campo o cabecera de extensión correspondiente del paquete IP interno. Es decir, indica qué valores externos se derivan de valores internos y cuáles se construyen independientemente de los valores internos.
- 6.3.** La autenticación y el cifrado extremo a extremo se prefieren entre dos *hosts*. Crea figuras similares a las Figuras 6.6 y 6.7 que muestren
 - a)** Contigüidad de transporte, con cifrado aplicado antes de la autenticación.
 - b)** Una SA de transporte agrupada en una SA en modo túnel, con cifrado aplicado antes de la autenticación.
 - c)** Una SA de transporte agrupada en una SA en modo túnel, con autenticación aplicada antes del cifrado.
- 6.4.** El documento de la arquitectura IPSec expresa que cuando dos SA en modo transporte se agrupan para permitir protocolos AH y ESP en el mismo flujo extremo a extremo, sólo una ordenación de los protocolos de seguridad parece apropiada: realizando el protocolo ESP antes de realizar el protocolo AH. ¿Por qué se recomienda esto en vez de la autenticación antes del cifrado?

- 6.5 a)** ¿Cuál de los tipos de intercambio ISAKMP (Tabla 6.4) corresponde al intercambio agresivo de claves Oakley (Figura 6.11)?
- b)** Indica, para el intercambio agresivo de claves Oakley, qué parámetros de cada mensaje corresponden a qué tipos de carga útil ISAKMP.

APÉNDICE 6A COMUNICACIÓN ENTRE REDES Y PROTOCOLOS DE INTERNET

Este apéndice proporciona una visión general de los protocolos de Internet. Empezamos con un resumen del papel que desempeña un protocolo de Internet en las comunicaciones entre redes. Luego, se introducen los dos protocolos de Internet principales, IPv4 e IPv6.

EL PAPEL DE UN PROTOCOLO DE INTERNET

Un protocolo de Internet (IP) proporciona la funcionalidad para conectar entre sí sistemas finales a través de varias redes. Con este fin, IP se implementa en cada sistema final y en *routers*, que son dispositivos que proporcionan conexión entre redes. Los datos de capas superiores en un sistema final origen se encapsulan en una unidad de datos del protocolo IP (PDU) para ser transmitidos. Esta PDU luego se transmite a través de una o más redes y *routers* para llegar al sistema final de destino.

El *router* debe poder afrontar una variedad de diferencias entre redes, incluyendo las siguientes:

- **Esquemas de dirección:** las redes pueden usar diferentes esquemas para asignar direcciones a los dispositivos. Por ejemplo, una LAN 802 IEEE usa direcciones binarias de 26 bits o de 48 bits para cada dispositivo en la red; una red pública de conmutación de paquetes X.25 usa direcciones decimales de 12 dígitos (codificadas como cuatro bits por dígito para una dirección de 48 bits). Se debe proporcionar alguna forma de dirección de red global, así como un servicio de directorio.
- **Tamaños máximos de paquete:** los paquetes de una red pueden haber sido divididos en partes pequeñas para transmitirse a otra red, un proceso conocido como **fragmentación**. Por ejemplo, Ethernet impone un tamaño máximo de paquete de 1500 bytes; en redes X.25 es común el tamaño máximo de 1000 bytes. Un paquete que se transmite en un sistema Ethernet y recogido por un *router* para la retransmisión en una red X.25 puede tener que fragmentar en dos partes el paquete que recibe.
- **Interfaces:** las interfaces *hardware* y *software* a las distintas redes son diferentes. El concepto de *router* debe ser independiente de estas diferencias.
- **Fiabilidad:** distintos servicios de red pueden proporcionar cualquier cosa, desde un circuito virtual fiable extremo a extremo hasta un servicio no fiable. La operación de los *routers* no debería depender de la suposición de fiabilidad de la red.

La operación del *router*, como indica la Figura 6.13, depende de un protocolo de Internet. En este ejemplo, el protocolo de Internet (IP) del *suite* de protocolos TCP/IP reali-

za esa función. IP debe implementarse en todos los sistemas finales de todas las redes así como en los *routers*. Además, cada sistema final debe tener protocolos compatibles por encima de IP para comunicarse con éxito. Los *routers* intermedios sólo necesitan tener protocolos hasta IP.

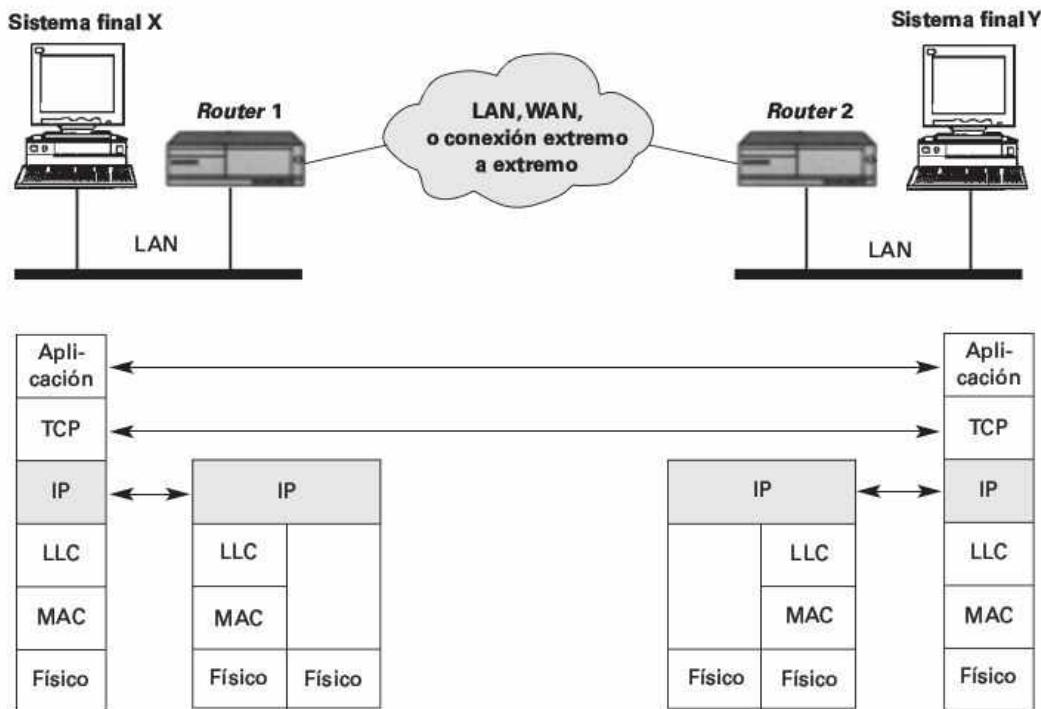


Figura 6.13 Configuración para un ejemplo de TCP/IP

Consideremos la transferencia de un bloque de datos desde el sistema final X al sistema final Y en la Figura 6.13. La capa IP en X recibe bloques de datos que han de enviarse a Y desde TCP en X. La capa IP adjunta una cabecera que especifica la dirección global de la internet de Y. Esta dirección tiene dos partes: identificador de red e identificador de sistema final. Llámese a este bloque paquete IP. Luego, IP reconoce que el destino (Y) está en otra subred. Por lo tanto, el primer paso consiste en enviar el paquete a un *router*, en este caso el *router* 1. Para realizarlo, IP pasa su unidad de datos a LLC con la información apropiada sobre la dirección. LLC crea una PDU de LLC, que se pasa a la capa MAC. Esta capa construye un paquete MAC cuya cabecera contiene la dirección del *router* 1.

Luego, el paquete viaja a través de la LAN al *router* 1. El *router* extrae las cabeceras y las terminaciones del paquete y analiza la cabecera IP para determinar el destino último de los datos, en este caso Y. El *router* ahora debe tomar una decisión de enrutamiento. Hay dos posibilidades:

1. El sistema final de destino Y está conectado directamente a una de las subredes a la que está unida el *router*.

- 2.** Para alcanzar el destino, se deben cruzar uno o más *routers* adicionales.

En este ejemplo, el paquete debe ser encaminado a través del *router* 2 antes de llegar al destino. Así, el *router* 1 pasa el paquete IP al *router* 2 por medio de la red intermedia. Con este fin, se usan los protocolos de esta red. Por ejemplo, si la red intermedia es una red X.25, la unidad de datos IP se adapta en un paquete X.25 con información adecuada de direccionamiento para llegar al *router* 2. Cuando este paquete llega al *router* 2, se elimina la cabecera del paquete. El *router* determina que este paquete IP está destinado a Y, que está conectado directamente a una subred a la que el *router* está unido. Por lo tanto, el *router* crea un paquete con una dirección de destino de Y y lo envía a la LAN. Finalmente, los datos llegan a Y, donde el paquete, LLC, y las cabeceras internet y las terminaciones se pueden eliminar.

Este servicio que ofrece IP no es fiable. Es decir, IP no garantiza que todos los datos serán entregados o que todos los datos que se envíen llegarán en el orden correcto. Es responsabilidad de la capa inmediatamente superior, en este caso TCP, resolver cualquier error que se produzca. Este enfoque proporciona mucha flexibilidad. Como la entrega no está garantizada, no hay requisito particular de fiabilidad en ninguna de las subredes. Así, el protocolo funcionará con cualquier combinación de tipos de subred. Como la secuencia del envío no está garantizada, los paquetes sucesivos pueden seguir diferentes caminos a través de la internet. Esto permite al protocolo reaccionar ante la congestión o el fallo en la internet cambiando las rutas.

IPV4

Durante décadas, la clave de la arquitectura del protocolo TCP/IP ha sido la versión 4 del protocolo de Internet (IP). La Figura 6.14a muestra el formato de la cabecera IP, que consta de un mínimo de 20 octetos o 160 bits. Los campos son los siguientes:

- **Versión (cuatro bits):** indica el número de versión, para permitir la evolución del protocolo; el valor es cuatro.
- **Longitud de cabecera de Internet (IHL) (cuatro bits):** longitud de la cabecera en palabras de 32 bits. El valor mínimo es cinco, para una longitud mínima de cabecera de 20 octetos.
- **Tipo de servicio (ocho bits):** proporciona una guía para los módulos IP de sistemas finales y para los *router* a lo largo del recorrido del paquete, en términos de prioridad relativa del paquete.
- **Longitud total (16 bits):** longitud total del paquete IP, en octetos.
- **Identificación (16 bits):** un número de secuencia que, junto con la dirección de origen, la de destino y el protocolo del usuario, trata de identificar un paquete de forma única. Así, el identificador debería ser único para la dirección fuente del paquete, la dirección de destino y el protocolo de usuario durante el tiempo en que el paquete permanezca en la internet.
- **Indicadores (tres bits):** sólo se definen dos de los bits. Cuando un paquete se fragmenta, el bit *Más (More)* indica si éste es el último fragmento del paquete original. El bit de *No Fragmentar (Don't Fragment)* prohíbe la fragmentación una vez que se ha fijado. Este bit puede ser útil si se sabe que el destino no tiene la capacidad

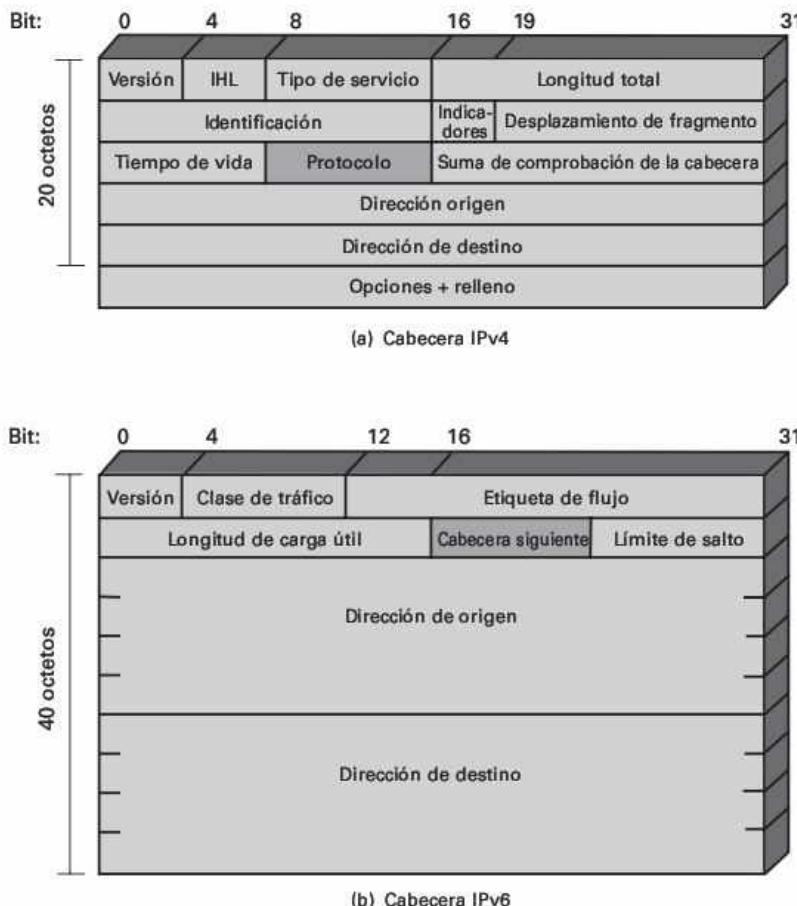


Figura 6.14 Cabeceras IP

de reensamblar fragmentos. Sin embargo, si se fija este bit, el paquete se ignorará si excede el tamaño máximo de una subred en ruta. Por lo tanto, si se fija este bit, puede ser recomendable usar el enrutamiento de la fuente para evitar subredes con un tamaño máximo de paquete pequeño.

- **Desplazamiento de fragmento (13 bits):** indica el lugar en el paquete original al que pertenece este fragmento, medido en unidades de 64 bits. Esto implica que los fragmentos, menos el último, deben contener un campo de datos que sea múltiplo de 64 bits de longitud.
- **Tiempo de vida (TTL, time to live) (ocho bits):** especifica cuánto tiempo, en segundos, está permitido que un paquete esté en la internet. Cada *router* que procesa un paquete debe disminuir el TTL al menos en uno. Por lo tanto, el TTL es similar al contador de saltos.
- **Protocolo (ocho bits):** indica el protocolo del nivel inmediatamente superior, que va a recibir el campo de datos en el destino; así, este campo identifica el tipo de la cabecera siguiente en el paquete después de la cabecera IP.

- **Suma de comprobación de la cabecera (16 bits):** un código de detección de errores aplicado sólo a la cabecera. Como algunos campos de la cabecera pueden cambiar durante la transmisión (por ejemplo, tiempo de vida, campos relacionados con la segmentación), éste se vuelve a verificar y a calcular en cada *router*. El campo de suma de comprobación es la suma de 16 bits en complemento a uno de todas las palabras de 16 bits de la cabecera. Para los cálculos, el campo de suma de comprobación se inicializa a un valor de cero.
- **Dirección de origen (32 bits):** codificada para permitir una ubicación variable de bits para especificar la red y el sistema final conectado a la red especificada (siete y 24 bits, 14 y 16 bits, o 21 y ocho bits).
- **Dirección de destino (32 bits):** iguales características que la dirección de origen.
- **Opciones (variable):** codifica las opciones solicitadas por el usuario emisor; pueden incluir etiqueta de seguridad, enrutamiento de origen, registro de enrutamiento y sellado de tiempo.
- **Relleno (variable):** se usa para asegurar que la cabecera del paquete es un múltiplo de 32 bits de longitud.

IPv6

En 1995, la IETF (*Internet Engineering Task Force*), que desarrolla estándares de protocolos para Internet, publicó una especificación para el IP de la siguiente generación, conocido como IPng. Esta especificación se convirtió en 1996 en un estándar conocido como IPv6. IPv6 proporciona una serie de mejoras funcionales al IP existente (conocido como IPv4), diseñado para ajustarse a las altas velocidades de las redes actuales y a la mezcla de flujo de datos, que incluyen vídeo y audio, que se están haciendo más comunes. Pero detrás del desarrollo del nuevo protocolo se halla la necesidad imperante de nuevas direcciones. IPv4 usa una dirección de 32 bits para especificar un origen o un destino. Con el crecimiento explosivo de Internet y las redes privadas conectadas a Internet, esta longitud de dirección se hizo insuficiente para ajustarse a todos los sistemas que necesitan direcciones. Como muestra la Figura 6.14b, IPv6 incluye campos de dirección fuente y destino de 128 bits. Por último, se espera que todas las instalaciones que usan TCP/IP migren del actual IP a IPv6, pero este proceso durará muchos años, si no décadas.

Cabecera IPv6

La cabecera IPv6 tiene una longitud fija de 40 octetos y consta de los siguientes campos (Figura 6.14b):

- **Versión (cuatro bits):** número de versión del protocolo de Internet; el valor es seis.
- **Clase de tráfico (ocho bits):** disponible originando nodos y/o reenviando *routers* para identificar y distinguir entre diferentes clases o prioridades de paquetes IPv6. El uso de este campo todavía se está estudiando.
- **Etiqueta de flujo (20 bits):** puede usarlo un *host* para etiquetar aquellos paquetes para los que esté solicitando un trato especial por parte de los *routers* en una red.

El etiquetado del flujo puede ayudar en la reserva de recursos y en el procesamiento de tráfico en tiempo real.

- **Longitud de carga útil (16 bits):** longitud del resto del paquete que sigue a la cabecera, en octetos. Es decir, es la longitud total de todas las cabeceras de extensión más la PDU del nivel de transporte.
- **Cabecera siguiente (ocho bits):** identifica el tipo de cabecera que sigue a la cabecera IPv6; será una cabecera de extensión IPv6 o una cabecera de capa superior, como TCP o UDP.
- **Límite de salto (ocho bits):** el número restante de saltos permitidos para este paquete. El límite de saltos lo fija el origen en un valor máximo deseado y disminuye en uno por cada nodo que reenvía el paquete. El paquete se ignora si el límite de saltos llega a cero.
- **Dirección de origen (128 bits):** la dirección del originador del paquete.
- **Dirección de destino (128 bits):** la dirección del receptor del paquete. Puede, de hecho, no ser el destino último deseado si una cabecera de extensión de enruteamiento está presente, como se explica más tarde.

Aunque la cabecera IPv6 es mayor que la porción obligatoria de la cabecera IPv4 (40 octetos frente a 20 octetos), contiene menos campos (ocho frente a 12). Por lo tanto, los *routers* tienen que hacer menos procesamiento por cabecera, lo cual debería aumentar la velocidad de enruteamiento.

Cabeceras de extensión IPv6

Un paquete IPv6 incluye la cabecera IPv6 que se acaba de explicar, y cero o más cabeceras de extensión. Fuera de IPSec, se han definido las siguientes cabeceras de extensión:

- **Cabecera de opciones salto en salto:** define opciones especiales que requieren procesamiento salto en salto
- **Cabecera de enruteamiento:** proporciona enruteamiento extendido, similar al enruteamiento fuente IPv4
- **Cabecera de fragmento:** contiene información sobre fragmentación y reensamblado
- **Cabecera de autenticación:** proporciona integridad y autenticación de paquetes
- **Cabecera de encapsulamiento de la carga útil de seguridad:** proporciona privacidad
- **Cabecera de opciones de destino:** contiene información opcional que ha de examinar el nodo de destino

El estándar IPv6 recomienda que, cuando se usen varias cabeceras de extensión, las cabeceras IPv6 aparezcan en el siguiente orden:

1. Cabecera IPv6: obligatoria, siempre debe aparecer en primer lugar
2. Cabecera de opciones salto en salto

- 3** Cabecera de opciones de destino: para las opciones que va a procesar el primer destino que aparece en el campo *dirección de destino* IPv6 más los siguientes destinos presentados en la cabecera de enrutamiento
- 4** Cabecera de enrutamiento
- 5** Cabecera de fragmento
- 6** Cabecera de autentificación
- 7**. Cabecera de encapsulamiento de carga útil de seguridad
- 8** Cabecera de opciones de destino: para las opciones que va a procesar sólo el destino último del paquete

La Figura 6.15 muestra un ejemplo de un paquete IPv6 que incluye una instancia de cada cabecera que no es de seguridad. Obsérvese que la cabecera IPv6 y cada cabecera de extensión incluye un campo de *cabecera siguiente*. Este campo identifica el tipo de cabecera siguiente. Si la siguiente cabecera es una cabecera de extensión, entonces este campo contiene el identificador del tipo de esa cabecera. Si no, este campo contiene el identificador de protocolo del protocolo de la capa superior usando IPv6 (normalmente un protocolo de la capa de transporte), usando los mismos valores que el campo *protocolo* IPv4. En la figura, el protocolo de la capa superior es TCP, con lo que los datos de la capa superior transportados por el paquete IPv6 constan de una cabecera TCP seguida de un bloque de datos de aplicación.

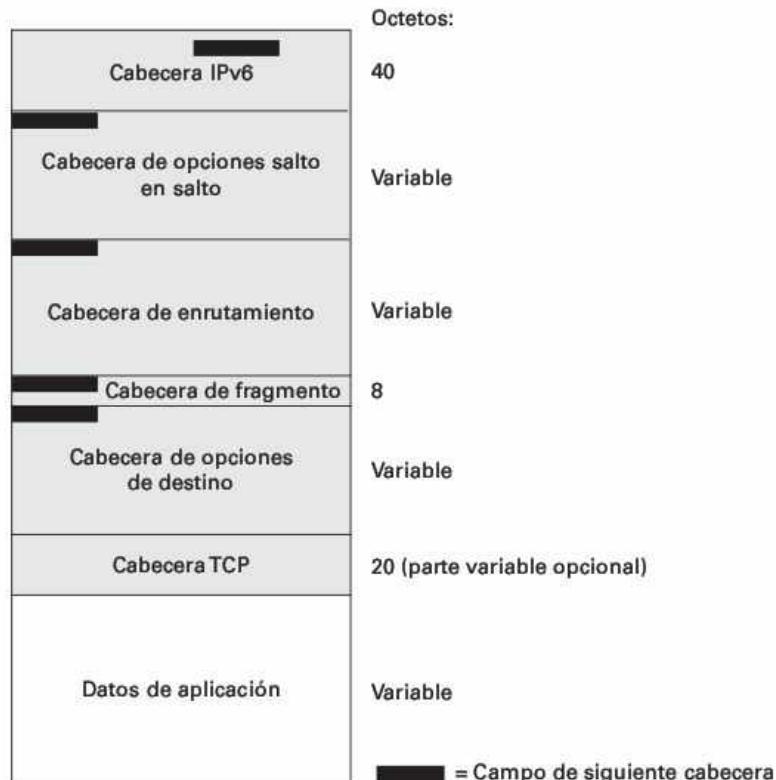


Figura 6.15 Paquete IPv6 con cabeceras de extensión (contiene un segmento TCP)

La **cabecera de opciones salto en salto** lleva información opcional que, de estar presente, debe ser examinada por cada *router* a lo largo del camino. La cabecera consta de los siguientes campos:

- **Siguiente cabecera (ocho bits):** identifica el tipo de cabecera que sigue a esta cabecera.
- **Longitud de la extensión de la cabecera (ocho bits):** longitud de esta cabecera en unidades de 64 bits, sin incluir los primeros 64 bits.
- **Opciones:** contiene una o más opciones. Cada opción está formada por tres subcampos: un indicador, mostrando el tipo de opción; una longitud; y un valor.

Hasta el momento se ha definido únicamente una opción: la opción *carga útil Jumbo*, que se usa para enviar paquetes IPv6 con cargas útiles mayores que $2^{16} - 1 = 65.535$ octetos. El campo *datos de opciones* de esta opción tiene una longitud de 32 bits y da la longitud del paquete en octetos, excluyendo la cabecera IPv6. Para estos paquetes, el campo de *longitud de carga útil* en la cabecera IPv6 debe estar a cero, y no debe haber cabecera de fragmento. Con esta opción, IPv6 permite tamaños de paquete de hasta más de 4.000 millones de octetos. Esto facilita la transmisión de paquetes grandes de vídeo y permite que IPv6 haga el mejor uso de la capacidad disponible en cualquier medio de transmisión.

La **cabecera de enrutamiento** contiene una lista de uno o más nodos intermedios que se han de visitar en el camino de destino del paquete. Todas las cabeceras de enrutamiento empiezan con un bloque de 32 bits formados por cuatro campos de ocho bits, seguidos de datos de enrutamiento específicos de un tipo dado de enrutamiento. Los cuatro campos de ocho bits son *siguiente cabecera*, *longitud de extensión de cabecera* y

- **Tipo de enrutamiento:** identifica una variante particular de cabecera de enrutamiento. Si un *router* no reconoce el valor del tipo de enrutamiento, debe ignorar el paquete.
- **Segmentos restantes:** número de nodos intermedios explícitos que todavía hay que visitar antes de llegar al destino final.

Además de esta definición general de cabecera, la especificación IPv6 define la cabecera de enrutamiento del tipo 0. Cuando se usa esta cabecera, el nodo de origen no coloca la dirección última de destino en la cabecera IPv6. En vez de eso, esa dirección es la última dirección de la cabecera de enrutamiento, y la cabecera IPv6 contiene la dirección de destino del primer *router* deseado en el camino. La cabecera de enrutamiento no será examinada hasta que el paquete llegue al nodo identificado en la cabecera IPv6. En ese punto, los contenidos de IPv6 y de la cabecera de enrutamiento se actualizan y se reenvía el paquete. La actualización consiste en colocar la siguiente dirección que ha de ser visitada en la cabecera IPv6 y disminuir el campo de *segmentos restantes* de la cabecera de enrutamiento.

IPv6 requiere un nodo IPv6 para invertir las rutas en un paquete que recibe que contenga una cabecera de enrutamiento, para devolver un paquete al emisor.

La **cabecera de fragmento** la usa una fuente cuando se necesita fragmentación. En IPv6, sólo pueden llevarla a cabo los nodos de origen, no los *routers* a lo largo del camino de envío del paquete. Para sacar el máximo partido del entorno de comunicación entre redes, un nodo debe realizar un algoritmo de descubrimiento del camino que le

permite conocer la unidad de transmisión máxima (MTU) más pequeña que permite cualquier subred en el camino.

En otras palabras, el algoritmo de descubrimiento del camino permite que un nodo conozca el MTU de la subred que produce «cuello de botella» en el camino. Con este conocimiento, el nodo fuente fragmentará, como se requiere, para cada dirección de destino dada. Si no, la fuente debe limitar todos los paquetes a 1280 octetos, que es el MTU mínimo que debe permitir cada subred.

Además del campo *cabecera siguiente*, la cabecera de fragmento incluye los siguientes campos:

- **Desplazamiento de fragmento (13 bits):** indica a qué lugar del paquete original pertenece la carga útil de este fragmento. Se mide en unidades de 64 bits. Esto implica que los fragmentos (que no sean el último fragmento) deben contener un campo de datos que sea múltiplo de 64 bits de largo.
- **Res (dos bits):** reservado para su uso en el futuro
- **Indicador M (un bit):** 1 = más fragmentos; 0 = último fragmento.
- **Identificación (32 bits):** diseñado para identificar únicamente al paquete original. El identificador debe ser único para la dirección fuente del paquete y la dirección de destino durante el tiempo en el que el paquete esté en la internet. Todos los fragmentos con el mismo identificador, dirección de origen y dirección de destino se vuelven a ensamblar para formar el paquete original.

La **cabecera de opciones de destino** lleva información opcional que, de estar presente, es examinada sólo por el nodo de destino del paquete. El formato de esta cabecera es el mismo que el de la cabecera de *opciones salto en salto*.

CAPÍTULO 7

Seguridad de la web

- 7.1 Consideraciones sobre seguridad en la web**
 - Amenazas a la seguridad de la web
 - Enfoques para la seguridad del tráfico en la web
- 7.2 SSL (*Secure Socket Layer*) y TLS (*Transport Layer Security*)**
 - Arquitectura SSL
 - Protocolo *Record* de SSL
 - Protocolo *Change Cipher Spec*
 - Protocolo *Alert*
 - Protocolo *Handshake*
 - Cálculos criptográficos
 - TLS
- 7.3 SET (*Secure Electronic Transaction*)**
 - Introducción a SET
 - Firma dual
 - Procesamiento del pago
- 7.4 Bibliografía y sitios web recomendados**
- 7.5 Palabras clave, preguntas de repaso y problemas**
 - Palabras clave
 - Preguntas de repaso
 - Problemas

*Usa tu mentalidad
Despierta a la realidad*

De la canción «I've got you under my skin» de COLE PORTER

Prácticamente todas las empresas, la mayoría de las agencias gubernamentales y muchos particulares disponen hoy en día de sitios web. El número de personas y compañías con acceso a Internet está creciendo rápidamente, y todos disponen de navegadores web gráficos. Como consecuencia de esto, las empresas muestran gran entusiasmo con la implantación de herramientas web para el comercio electrónico. Pero la realidad es que Internet y la web son sumamente vulnerables ante peligros de diversos tipos. A medida que las empresas adquieran conciencia de esta realidad, la demanda de servicios web seguros aumenta.

El tema de la seguridad en la web es extenso y puede, fácilmente, abarcar un libro entero (se recomiendan algunos al final del capítulo). El capítulo comienza con una discusión sobre los requisitos generales para la seguridad en la web para luego centrarse en dos esquemas que son cada vez más importantes como parte del comercio web: SSL/TLS y SET.

7.1 CONSIDERACIONES SOBRE SEGURIDAD EN LA WEB

La *World Wide Web* es, básicamente, una aplicación cliente/servidor que se ejecuta en Internet y en las intranets TCP/IP. Como tal, las herramientas de seguridad y los enfoques discutidos anteriormente en este libro son importantes para la seguridad de la web. Pero, como se destaca en [GARF97], la web presenta nuevos retos que generalmente no se aprecian en el contexto de la seguridad de los computadores ni de la red:

- Internet es bidireccional. Al contrario que los entornos de publicación tradicionales, incluso los sistemas de publicación electrónica que hacen uso de teletexto, respuesta de voz o respuesta de fax, la web es vulnerable a los ataques a los servidores web desde Internet.
- La web se emplea cada vez más para presentar información de empresas y de productos y como plataforma para transacciones de negocios. Se puede perjudicar la imagen y ocasionar pérdidas económicas si se manipulan los servidores web.
- Aunque los navegadores web son muy fáciles de usar, los servidores relativamente sencillos de configurar y gestionar y los contenidos web cada vez más fáciles de desarrollar, el *software* subyacente es extraordinariamente complejo. Éste puede ocultar muchos posibles fallos de seguridad. La corta historia de la web está llena de ejemplos de sistemas nuevos y actualizados, instalados de manera adecuada, que son vulnerables a una serie de ataques a la seguridad.
- Un servidor web puede utilizarse como plataforma de acceso a todo el complejo de computadores de una agencia o corporación. Una vez comprometida la seguridad del servidor web, un atacante podría tener acceso a datos y sistemas fuera del propio servidor pero que están conectados a éste en el sitio local.

- Habitualmente, los clientes de servicios basados en web son usuarios ocasionales y poco preparados (en lo que a seguridad se refiere). Estos usuarios no tienen por qué ser conscientes de los riesgos que existen y no tienen las herramientas ni los conocimientos necesarios para tomar medidas efectivas.

Tabla 7.1 Comparación de amenazas en la web [RUBI97]

	Amenazas	Consecuencias	Contramedidas
Integridad	<ul style="list-style-type: none"> • Modificación de datos de usuario • Navegador caballo de Troya • Modificación de memoria • Modificación del tráfico del mensaje en tránsito 	<ul style="list-style-type: none"> • Pérdida de información • Máquina en peligro • Vulnerabilidad al resto de amenazas 	<ul style="list-style-type: none"> • Suma de comprobación (<i>checksum</i>) criptográfica
Confidencialidad	<ul style="list-style-type: none"> • Escuchas ocultas en la red • Robo de información del servidor • Robo de datos del cliente • Información sobre la configuración de la red • Información sobre qué cliente se comunica con el servidor 	<ul style="list-style-type: none"> • Pérdida de información • Pérdida de privacidad 	<ul style="list-style-type: none"> • Cifrado, proxy web
Denegación de servicio	<ul style="list-style-type: none"> • Interrupción de procesos del usuario • Inundar la máquina con amenazas fraudulentas • Llenar el espacio de disco o la memoria • Aislar la máquina mediante ataques de DNS 	<ul style="list-style-type: none"> • Destructivo • Molesto • Impide que los usuarios finalicen su trabajo 	<ul style="list-style-type: none"> • Difícil de prevenir
Autentificación	<ul style="list-style-type: none"> • Suplantación de usuarios legítimos • Falsificación de datos 	<ul style="list-style-type: none"> • Falsificación de usuario • Creer que la información falsa es válida 	<ul style="list-style-type: none"> • Técnicas criptográficas

AMENAZAS A LA SEGURIDAD DE LA WEB

La Tabla 7.1 muestra un resumen de los tipos de amenazas a la seguridad que se afrontan al usar la web. Esas amenazas se pueden agrupar en ataques pasivos y ataques activos. Los pasivos incluyen escuchas del tráfico de la red entre el navegador y el servidor y la obtención de acceso a información de un sitio web que se supone restringida. Los ataques activos incluyen la suplantación de otro usuario, la alteración de mensajes en tránsito entre cliente y servidor y la modificación de información de un sitio web.

Otra manera de clasificar las amenazas a la seguridad de la web es en función de la ubicación de la amenaza: servidor web, navegador web y tráfico de red entre navegador y servidor. Los aspectos referentes a la seguridad del servidor y del navegador se enmarcan dentro de la categoría de la seguridad de los sistemas de computadores; la Parte IV de este libro trata el aspecto de la seguridad de los sistemas en general, pero también es aplicable a la seguridad de los sistemas web. Los temas de la seguridad del tráfico encajan en la categoría de la seguridad de la red y se tratan en este Capítulo.

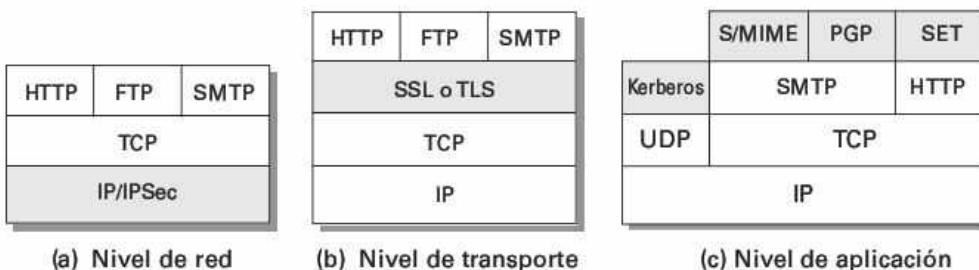


Figura 7.1 Ubicación relativa de las herramientas de seguridad en la pila de protocolos TCP/IP

ENFOQUES PARA LA SEGURIDAD DEL TRÁFICO EN LA WEB

Hay varios enfoques para proporcionar seguridad en la web. Dichos enfoques son similares en los servicios que proporcionan y, hasta cierto punto, en los mecanismos que usan, pero difieren en lo referente a su ámbito de aplicabilidad y en cuanto a su ubicación relativa dentro de la pila de protocolos TCP/IP.

La Figura 7.1 ilustra estas diferencias. Una forma de proporcionar seguridad en la web es usar Seguridad IP (*IP Security*, Figura 7.1a). La ventaja de usar IPsec es que es transparente al usuario final y a las aplicaciones, proporcionando una solución de propósito general. Además, IPsec incluye una capacidad de filtrado, de manera que solamente el tráfico seleccionado afecta a la carga de procesamiento de IPsec.

Otra solución de propósito relativamente general es implementar la seguridad justo encima de TCP (Figura 7.1b). El principal ejemplo de este enfoque es *Secure Sockets Layer* (SSL), y su sucesor, el estándar de Internet *Transport Layer Security* (TLS). En este nivel, hay dos opciones de implementación. SSL (o TLS) se podría proporcionar como parte de la *suite* de protocolos y, de esta manera, ser transparente a las aplicaciones.

Como alternativa, SSL se podría incluir en paquetes específicos. Por ejemplo, los navegadores Netscape y Microsoft Explorer vienen equipados con SSL, y la mayoría de los servidores web tienen el protocolo implementado.

El último enfoque consiste en la inclusión de servicios de seguridad específicos de las aplicaciones. La Figura 7.1c muestra ejemplos de esta arquitectura. La ventaja de este enfoque es que el servicio se puede adecuar a las necesidades específicas de una aplicación. En el contexto de la seguridad en la web, un ejemplo importante de este enfoque es *Secure Electronic Transaction* (SET)¹.

El resto de este capítulo está dedicado a SSL/TLS y SET.

7.2 SSL (SECURE SOCKET LAYER) Y TLS (TRANSPORT LAYER SECURITY)

Netscape creó SSL. La versión 3 del protocolo se diseñó con la participación pública y con el apoyo de la industria y se publicó como borrador de Internet. Posteriormente, cuando se alcanzó el consenso de presentar el protocolo para su estandarización en Internet, se formó el grupo de trabajo TLS dentro de la IETF con el objetivo de desarrollar un estándar común. La primera versión publicada de TLS puede verse, básicamente, como SSLv3.1 y es muy similar y compatible con SSLv3.

La mayor parte de esta sección está dedicada a SSLv3. Al final de la misma se describen las principales diferencias entre SSLv3 y TLS.

ARQUITECTURA SSL

SSL está diseñado de forma que utilice TCP para proporcionar un servicio fiable y seguro extremo a extremo. SSL no es un protocolo simple sino que tiene dos niveles de protocolos, como se observa en la Figura 7.2.

El protocolo *Record* de SSL proporciona servicios de seguridad básica a varios protocolos de nivel más alto. En particular, el *Hypertext Transfer Protocol* (HTTP), que suministra el servicio de transferencia para la interacción entre cliente y servidor web, puede operar encima de SSL. Se definen tres protocolos de nivel más alto como parte de SSL: *Handshake Protocol*, *Change Cipher Spec Protocol* y *Alert Protocol*. Esos protocolos específicos de SSL se utilizan en la gestión de intercambios SSL y se examinarán más tarde en esta sección.

Dos conceptos importantes de SSL son la sesión SSL y la conexión SSL, definidos en las especificaciones de la siguiente manera:

- **Conexión:** una conexión es un transporte (según la definición del modelo de niveles OSI) que proporciona un tipo de servicio idóneo. Para SSL, tales conexiones son relaciones de igual a igual. Las conexiones son transitorias. Cada conexión está asociada con una sesión.
- **Sesión:** una sesión SSL es una asociación entre un cliente y un servidor. Las sesiones las crea el protocolo *Handshake* (o de negociación). Las sesiones definen un

¹ La Figura 7.1c muestra SET encima de HTTP; ésta es una implementación habitual. En algunas implementaciones, SET utiliza TCP directamente.

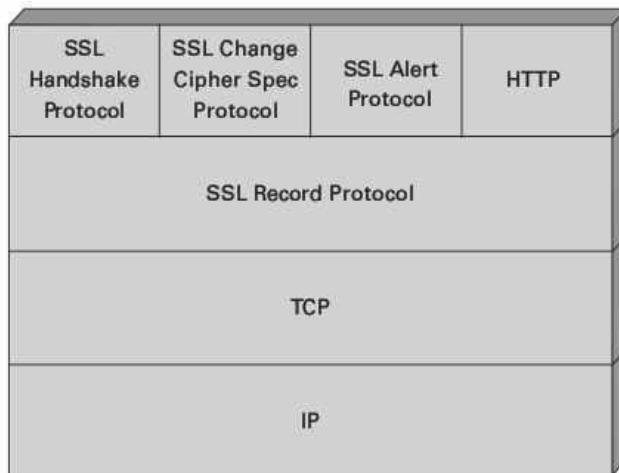


Figura 7.2 Pila de protocolos SSL

conjunto de parámetros criptográficos de seguridad, que se pueden compartir entre múltiples conexiones. Las sesiones se usan para evitar el costoso proceso de negociación de nuevos parámetros de seguridad para cada conexión.

Entre cualquier par de interlocutores (aplicaciones como el HTTP sobre cliente y servidor), podría haber múltiples conexiones seguras. En teoría, también podría haber múltiples sesiones simultáneas entre las partes, pero en la práctica no se usa esta característica.

En realidad hay un número de estados asociados con cada sesión. Una vez se establece una sesión, hay un estado operativo para leer y escribir (por ejemplo, recibir y enviar). Además, durante el protocolo *Handshake*, se crean estados de lectura y escritura pendientes. Al finalizar satisfactoriamente el protocolo *Handshake*, los estados pendientes se convierten en estados operativos.

Una fase de sesión se define por los siguientes parámetros (estas definiciones se han extraído de la especificación de SSL):

- **Identificador de sesión (*session identifier*)**: secuencia arbitraria de bytes elegida por el servidor para identificar un estado de sesión activo o reanudable.
- **Certificado de la entidad par (*peer certificate*)**: certificado X509.v3 del par. Este elemento puede ser nulo.
- **Método de compresión (*compression method*)**: el algoritmo usado para comprimir datos antes del cifrado.
- **Especificación de cifrado (*cipher spec*)**: especifica el grueso del algoritmo de cifrado (tal como nulo, DES, etc.) y un algoritmo de *hash* (tal como MD5 o SHA-1) usado para el cálculo del MAC. También define atributos criptográficos como, por ejemplo, el *hash_size*.
- **Clave maestra (*master secret*)**: contraseña de 48 bytes compartida entre cliente y servidor.
- **Es reanudable (*is resumable*)**: indicador que refleja si la sesión se puede utilizar para iniciar nuevas conexiones.

Un estado de conexión se define por los siguientes parámetros:

- **Valores aleatorios del servidor y del cliente (Server and client random):** secuencias de bytes elegidas por el servidor y el cliente para cada conexión.
- **Clave secreta para MAC de escritura del servidor (Server write MAC secret):** la clave secreta usada en operaciones MAC sobre datos enviados por el servidor.
- **Clave secreta para MAC de escritura del cliente (Client write MAC secret):** la clave secreta usada en operaciones MAC sobre datos enviados por el cliente.
- **Clave de escritura del servidor (Server write key):** la clave de cifrado convencional para los datos cifrados por el servidor y descifrados por el cliente.
- **Clave de escritura del cliente (Client write key):** la clave de cifrado convencional para los datos cifrados por el cliente y descifrados por el servidor.
- **Vector de inicialización (IV, Initialization vector):** cuando se usa un cifrador de bloque en modo CBC, se necesita un vector de inicialización para cada clave. Este campo lo inicializa primero el protocolo Handshake. A partir de ahí el bloque final de texto cifrado de cada registro se usa como vector de inicialización (IV) del siguiente registro.
- **Números de secuencia (Sequence numbers):** cada parte mantiene números de secuencia separados para los mensajes transmitidos y recibidos en cada conexión. Cuando una de las partes envía o recibe un mensaje *change cipher spec*, se pone a cero el número de secuencia correspondiente. Los números de secuencia no pueden exceder de $2^{64}-1$.

PROTOCOLO RECORD DE SSL

El protocolo *Record* proporciona dos servicios a las conexiones SSL:

- **Confidencialidad:** el protocolo *Handshake* define una clave secreta compartida que se usa para cifrado convencional de la carga útil de SSL.
- **Integridad de mensajes:** el protocolo *Handshake* también define una clave secreta compartida que se usa para formar un código de autentificación de mensajes (MAC, *message authentication code*).

La Figura 7.3 indica las diferentes operaciones del protocolo *Record*. El protocolo toma un mensaje de la aplicación que se va a transmitir, fragmenta los datos en bloques manejables, opcionalmente los comprime, le añade un MAC, realiza el cifrado, le añade una cabecera y transmite la unidad resultante en un segmento TCP. Los datos recibidos se descifran, se verifican, se descomprimen y se ensamblan. Luego, se envían a la aplicación receptora en algún nivel superior.

El primer paso es la **fragmentación**. Cada mensaje de nivel superior se fragmenta en bloques de 2^{14} bytes (16384 bytes) o más pequeños. Despues, opcionalmente, se puede aplicar la **compresión**. La compresión debe realizarse sin pérdidas y no debe incrementar la longitud de los contenidos en más de 1024 bytes². En SLLv3 (así como en la ver-

² Naturalmente, se espera que la compresión reduzca los datos, en vez de aumentarlos. De todas formas, debido a las convenciones del formato, es posible que para bloques muy pequeños el algoritmo de compresión produzca una salida mayor que la entrada.

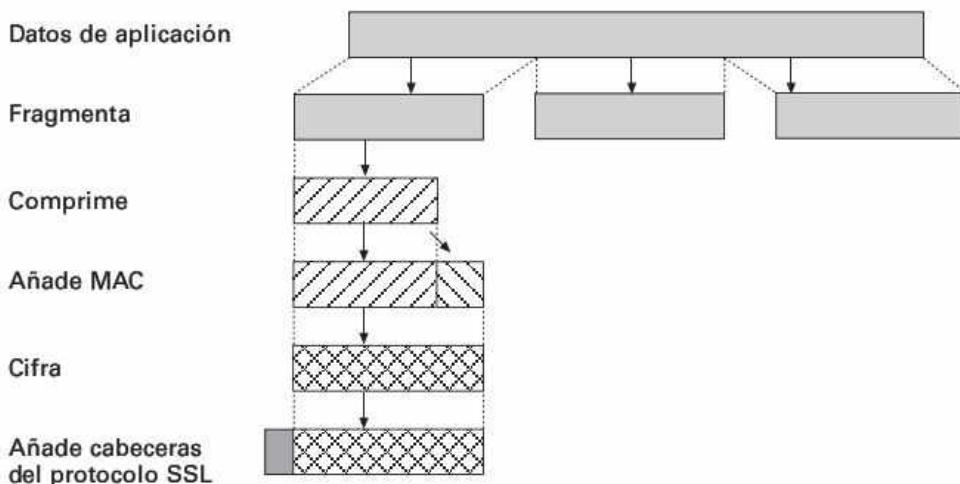


Figura 7.3 Operación del protocolo *Record* de SSL

sión actual de TLS), no se especifica ningún algoritmo de compresión, por lo que el algoritmo de compresión por defecto es nulo.

El siguiente paso en el proceso es calcular un **código de autentificación de mensaje** (MAC) sobre los datos comprimidos. Para este propósito, se usa una clave secreta compartida. El cálculo se define como:

```
hash(MAC_write_secret || pad_2 ||  
     hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type ||  
           SSLCompressed.length || SSLCompressed.fragment))
```

donde

	= concatenación
MAC_write_secret	= clave secreta compartida
hash	= algoritmo <i>hash</i> criptográfico; MD5 o SHA-1
pad_1	= el byte 0x36 (0011 0110) repetido 48 veces (384 bits) para MD5 y 40 veces (320 bits) para SHA-1
pad_2	= el byte 0x5C (0101 1100) repetido 48 veces para MD5 y 40 veces para SHA-1
seq_num	= el número de secuencia para este mensaje
SSLCompressed.type	= el protocolo de nivel superior usado para procesar este fragmento
SSLCompressed.length	= la longitud del fragmento comprimido
SSLCompressed.fragment	= el fragmento comprimido (si no se usa compresión, el fragmento de texto claro)

Obsérvese que esto es muy similar al algoritmo HMAC definido en el Capítulo 3. La diferencia se halla en que los dos valores de relleno (`pad_1` y `pad_2`) se concatenan en SSLv3, mientras que en HMAC se suman con un OR exclusivo. El algoritmo MAC de SSLv3 se basa en el borrador de Internet original para HMAC, que usa la concatenación. La versión final de HMAC, definida en el RFC 2104, utiliza XOR.

Lo siguiente es cifrar el mensaje comprimido más el MAC usando cifrado simétrico. El cifrado no puede incrementar la longitud del contenido en más de 1024 bytes para que la longitud total no exceda de $2^{14} + 2048$. Se permiten los siguientes algoritmos de cifrado:

Cifradores de bloque		Cifradores de flujo	
Algoritmo	Tamaño de clave	Algoritmo	Tamaño de clave
IDEA	128	RC4-40	40
RC2-40	40	RC4-128	128
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

El algoritmo Fortezza se puede usar en un esquema de cifrado mediante tarjeta inteligente.

Para el cifrado de flujo, se cifra el mensaje comprimido más el MAC. Obsérvese que el MAC se calcula antes de realizar el cifrado y que el MAC se cifra junto con el texto en claro o el texto en claro comprimido.

Para el cifrado de bloque, el relleno se puede añadir después del MAC y antes de realizar cifrado. El relleno se hace añadiendo un número de bytes de relleno seguidos de un byte que indica la longitud del relleno. La cantidad total de relleno es la cantidad más pequeña tal que el tamaño total de los datos que se van a cifrar (texto en claro más el MAC más el relleno) es un múltiplo de la longitud del bloque de cifrado. Un ejemplo es un texto en claro (o texto comprimido si se usa compresión) de 58 bytes, con un MAC de 20 bytes (usando SHA-1), que se cifra usando una longitud de bloque de ocho bytes (por ejemplo, con el DES). Con el byte que indica la longitud del relleno (`padding.length`), se alcanza una longitud total de 79 bytes. Para hacer que el total sea un múltiplo de ocho se añade un byte de relleno.

El paso final del proceso del protocolo *Record* consiste en añadir una cabecera, compuesta por los siguientes campos:

- **Tipo de contenido (ocho bits):** el protocolo de nivel superior usado para procesar el fragmento interno.
- **Versión mayor (ocho bits):** indica la versión mayor de SSL en uso. Para SSLv3, el valor es tres.
- **Versión menor (ocho bits):** indica la versión menor en uso. Para SSLv3, el valor es cero.
- **Longitud de compresión (16 bits):** la longitud en bytes del fragmento de texto en claro (o comprimido, si es el caso). El valor máximo es $2^{14}+2048$.

Los tipos de contenido definidos son *change_cipher_spec*, *alert*, *handshake* y *application_data*. Los tres primeros son protocolos específicos de SSL que se discutirán a continuación. Obsérvese que no se hace distinción entre las diferentes aplicaciones (por ejemplo, HTTP) que podrían usar SSL; el contenido de los datos creados por dichas aplicaciones es opaco para SSL.

La Figura 7.4 muestra el formato del protocolo *Record* de SSL.

PROTOCOLO CHANGE CIPHER SPEC

El protocolo *Change Cipher Spec* es uno de los tres protocolos específicos de SSL que utilizan el protocolo *Record* y, además, es el más simple. Este protocolo se compone de un único mensaje (Figura 7.5a), que contiene un único byte con el valor 1. El único propósito de este mensaje es hacer que un estado pendiente se copie en el estado operativo, lo cual actualiza la *suite* de cifrado que se usará en esta conexión.

PROTOCOLO ALERT

El protocolo *Alert* se usa para transmitir las alertas relacionadas con SSL a la entidad par. Como con otras aplicaciones que usan SSL, los mensajes de alerta se comprimen y se cifran, según se especifica en el estado en operativo.

Cada mensaje de este protocolo está formado por dos bytes (Figura 7.5b). El primer byte toma el valor de aviso(1) o fatal(2) para hacer saber la gravedad del mensaje. Si el nivel es fatal, SSL termina la conexión inmediatamente. Otras conexiones de la misma sesión pueden continuar, pero ya no se pueden establecer nuevas conexiones para esta sesión. El segundo byte contiene un código que indica la alerta específica. Primero vamos a enumerar las alertas que siempre son fatales (definiciones según la especificación de SSL):

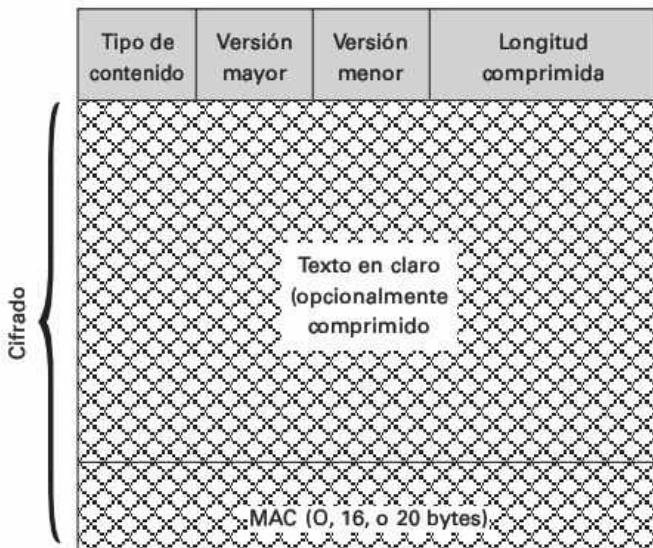


Figura 7.4 Formato del mensaje del protocolo *Record* de SSL

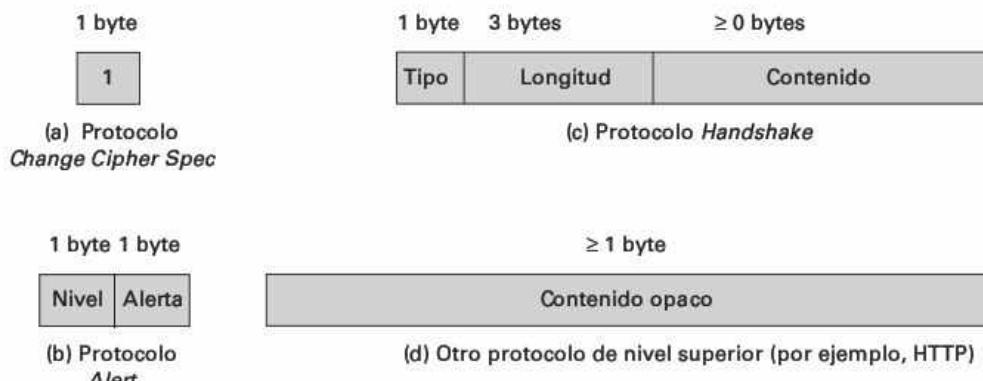


Figura 7.5 Mensajes para el procesamiento del protocolo *Record* de SSL

- **mensaje inesperado (*unexpected message*):** se recibió un mensaje inapropiado.
- **mac de registro erróneo (*bad_record_mac*):** se recibió un MAC incorrecto.
- **fallo de descompresión (*decompression_failure*):** la función de descompresión recibió una entrada inadecuada (por ejemplo, no la puede descomprimir o la ha descomprimido dando como resultado una longitud mayor de la permitida)
- **fallo de negociación (*handshake_failure*):** el emisor, dadas las opciones disponibles, fue incapaz de negociar un conjunto de parámetros de seguridad aceptables.
- **parámetro ilegal (*illegal_parameter*):** un campo en un mensaje de negociación estaba fuera de rango o era inconsistente con otros campos.

El resto de alertas son las siguientes:

- **notificación de cierre (*close_notify*):** notifica al receptor que el emisor no enviará más mensajes en esta conexión. Se requiere que cada parte envíe una alerta *close_notify* antes de cerrar el lado que escribe de una conexión.
- **no certificado (*no_certificate*):** puede enviarse como respuesta a una solicitud de certificado si no hay disponible un certificado apropiado.
- **certificado erróneo (*bad_certificate*):** un certificado recibido no es correcto (por ejemplo, contiene una firma que no se pudo verificar).
- **certificado no permitido (*unsupported_certificate*):** el tipo de certificado recibido no está permitido.
- **certificado revocado (*certificate_revoked*):** un certificado ha sido revocado por su firmante.
- **certificado caducado (*certificate_expired*):** un certificado ha caducado.

- **certificado desconocido (*certificate_unknown*):** algún otro asunto no especificado se detectó durante el tratamiento del certificado, considerándose éste inaceptable.

PROTOCOLO HANDSHAKE

La parte más compleja de SSL es el protocolo *Handshake*. Este protocolo permite la autenticación mutua de servidor y cliente y negociar un algoritmo de cifrado y de cálculo del MAC y las claves criptográficas que se utilizarán para proteger los datos enviados en un registro SSL. Este protocolo *Handshake* se utiliza antes de que se transmita cualquier dato de aplicación.

El protocolo *Handshake* consiste en una serie de mensajes intercambiados entre servidor y cliente. Todos los mensajes tienen el formato que se muestra en la Figura 7.5c. Cada mensaje tiene tres campos:

- **Tipo: (un byte):** indica uno de los 10 mensajes. La Tabla 7.2 enumera los tipos de mensajes definidos.
- **Longitud (tres bytes):** la longitud del mensaje en bytes.
- **Contenido (\geq un byte):** los parámetros asociados con el mensaje; también aparecen en la Tabla 7.2.

La Figura 7.6 muestra el intercambio inicial necesario para establecer una conexión lógica entre el cliente y el servidor. El intercambio puede verse dividido en cuatro fases.

Tabla 7.2 Tipos de mensajes del protocolo Handshake de SSL

Tipo de mensaje	Parámetros
<i>hello_request</i>	Nulo
<i>client_hello</i>	Versión, valor aleatorio, id de sesión, <i>suite de cifrado</i> , método de compresión
<i>server_hello</i>	Versión, valor aleatorio, id de sesión, <i>suite de cifrado</i> , método de compresión
<i>certificate</i>	Cadena de certificados X.509v3
<i>server_key_exchange</i>	Parámetros, firma
<i>certificate_request</i>	Tipo, autoridades
<i>server_done</i>	Nulo
<i>certificate_verify</i>	Firma
<i>client_key_exchange</i>	Parámetros, firma
<i>finished</i>	Valor <i>hash</i>

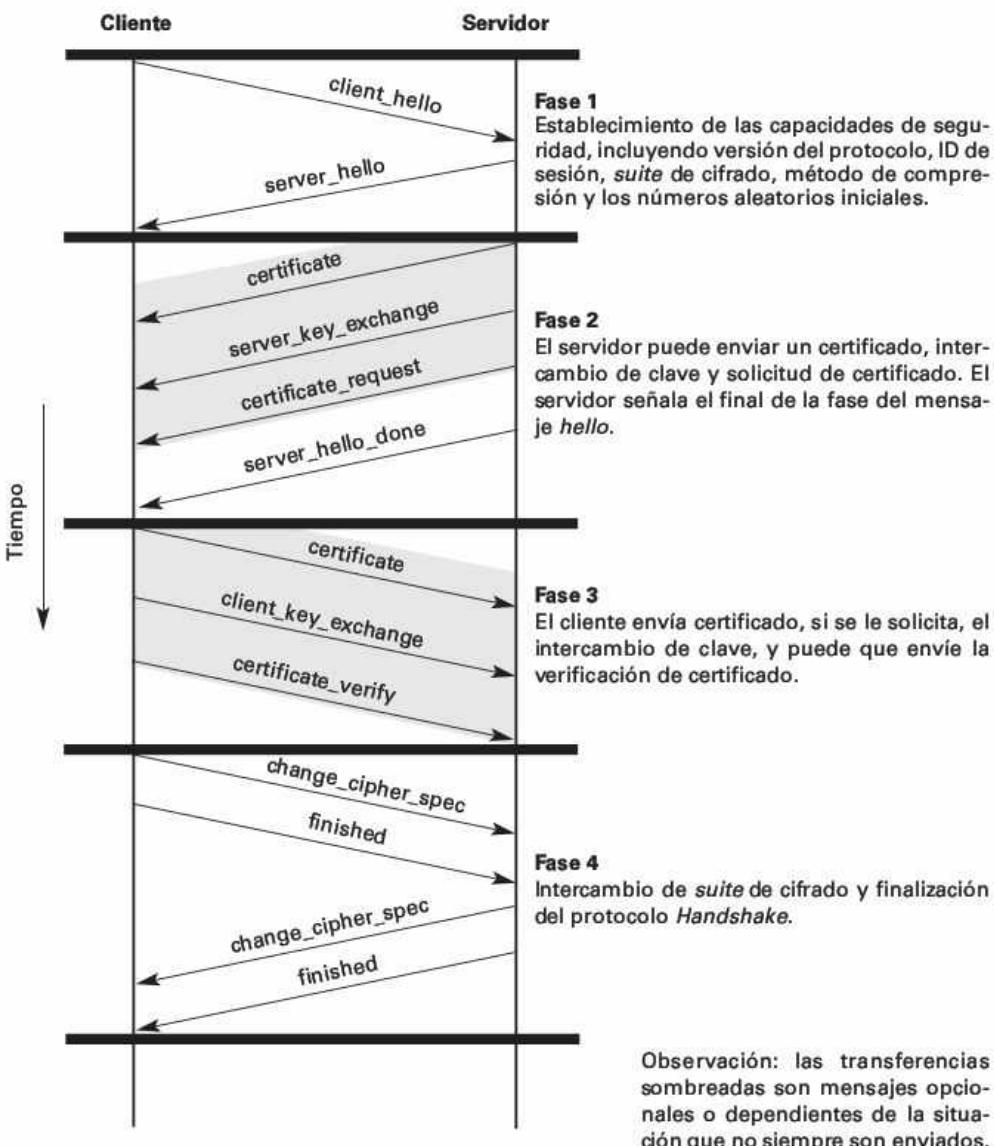


Figura 7.6 Acción del protocolo *Handshake*

Fase 1. Establecimiento de las capacidades de seguridad

En esta Fase se inicia una conexión lógica y se establecen las capacidades de seguridad asociadas con ésta. El intercambio lo empieza el cliente, quien envía un **mensaje `client_hello`** con los siguientes parámetros:

- **Versión:** la versión más alta de SSL con la que se puede entender el cliente.
- **Valor aleatorio (`random`):** estructura aleatoria generada por el cliente, que consiste en un sello de tiempo de 32 bits y 28 bytes producidos por un generador de

números aleatorios seguro. Esos valores sirven como *nonces* y se usan durante el intercambio de clave para prevenir ataques de repetición.

- **Identificador de sesión (*Session ID*):** un identificador de sesión de longitud variable. Un valor distinto de cero indica que el cliente desea actualizar los parámetros de una conexión existente o crear una nueva conexión en esta sesión. Un valor de cero indica que el cliente desea establecer una nueva conexión en una nueva sesión.
- **Suite de cifrado (*CipherSuite*):** es una lista que contiene las combinaciones de algoritmos criptográficos admitidos por el cliente, en orden decreciente de preferencia. Cada elemento de la lista (cada *suite* de cifrado) define un algoritmo de intercambio de clave y una especificación de cifrado (*CipherSpec*); esto se discutirá a continuación.
- **Método de compresión (*Compression Method*):** es una lista de métodos de compresión admitidos por el cliente.

Después de enviar el mensaje *client_hello*, el cliente espera el **mensaje *server_hello***, que contiene los mismos parámetros que el mensaje *client_hello*. Para el mensaje *server_hello* se aplican las siguientes convenciones. El campo Versión contiene una versión más baja que la sugerida por el cliente y la más alta permitida por el servidor. El campo con la estructura aleatoria (*random*) es generado por el servidor y es independiente del campo Valor Aleatorio del cliente. Si el campo Identificación de Sesión del cliente no es cero, el servidor utiliza el mismo valor; en caso contrario, el campo Identificación de Sesión del servidor contiene el valor para una nueva sesión. El campo *Suite de Cifrado* contiene una única *suite* de cifrado seleccionada por el servidor de entre las propuestas por el cliente. El campo Método de Compresión contiene el método de compresión elegido por el servidor de entre los propuestos por el cliente.

El primer elemento del parámetro *Suite de Cifrado* es el método de intercambio de clave (por ejemplo, la forma en que se intercambian las claves criptográficas para el cifrado convencional y el cálculo de MAC). Se permiten los siguientes métodos de intercambio de clave:

- **RSA:** la clave secreta se cifra con la clave pública RSA del receptor. Debe disponerse de un certificado de clave pública para la clave del receptor.
- **Diffie-Hellman fijo:** éste es un intercambio de clave Diffie-Hellman en el cual el certificado del servidor contiene los parámetros públicos de Diffie-Hellman firmados por una autoridad de certificación. Es decir, el certificado de clave pública contiene los parámetros de clave pública de Diffie-Hellman. El cliente proporciona sus parámetros de clave pública de Diffie-Hellman, en un certificado, si se solicita autenticación del cliente, o en un mensaje de intercambio de clave. Este método determina una clave secreta fija entre las dos partes, basada en los cálculos de Diffie-Hellman utilizando claves públicas fijas.
- **Diffie-Hellman efímero:** esta técnica se usa para crear claves secretas efímeras (temporales, de un solo uso). En este caso, las claves públicas de Diffie-Hellman se intercambian firmadas con la clave privada RSA o DSS del emisor. El receptor puede usar la correspondiente clave pública para verificar la firma. Los certificados se usan para autenticar las claves públicas. Ésta parece ser la más segura de las tres opciones de Diffie-Hellman ya que proporciona una clave temporal y autenticada.

- **Diffie-Hellman anónimo:** se usa el algoritmo básico de Diffie-Hellman sin autenticación. Es decir, cada lado envía sus parámetros Diffie-Hellman públicos al otro, sin autenticación. Este enfoque es vulnerable al ataque de interceptación, en el cual el atacante, utilizando Diffie-Hellman anónimo, se conecta con ambas partes dirigiendo la comunicación, mientras las partes creen que ésta es directa.

- **Fortezza:** la técnica definida para el esquema de Fortezza.

Después de la definición de un método de intercambio de clave se encuentra la especificación del cifrado (*CipherSpec*), la cual incluye los siguientes campos:

- **Algoritmo de cifrado:** cualquiera de los algoritmos mencionados anteriormente: RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza
- **Algoritmo MAC:** MD5 o SHA-1
- **Tipo de cifrador:** en flujo o de bloque
- **Es exportable:** verdadero o falso
- **Tamaño del hash:** 0, 16 bytes (para MD5) o 20 bytes (para SHA-1)
- **Material de clave:** una secuencia de bytes que contiene datos utilizados para generar las claves de escritura
- **Tamaño del vector de inicialización:** el tamaño del vector de inicialización para el cifrado en modo CBC

Fase 2. Autentificación del servidor e intercambio de clave

El servidor comienza esta Fase enviando su certificado, si necesita ser autenticado; el mensaje contiene uno o una cadena de certificados X.509. El **mensaje de certificado** es necesario para cualquier acuerdo sobre el método de intercambio de clave excepto para el método Diffie-Hellman anónimo. Obsérvese que si se usa Diffie-Hellman fijo, este mensaje de certificado funciona como un mensaje de intercambio de clave del servidor porque contiene los parámetros Diffie-Hellman públicos del servidor.

Luego, se puede enviar un **mensaje server key exchange**, si es necesario. No es necesario en los siguientes casos: (1) el servidor ha enviado un certificado con los parámetros Diffie-Hellman fijos, o (2) se va a usar intercambio de clave RSA. El mensaje de intercambio de clave del servidor es necesario para lo siguiente:

- **Diffie-Hellman anónimo:** el contenido del mensaje se compone de dos valores globales de Diffie-Hellman (un número primo y una raíz primitiva de ese número) más la clave Diffie-Hellman pública del servidor (ver la Figura 3.10).
- **Diffie-Hellman efímero:** el contenido del mensaje incluye los tres parámetros Diffie-Hellman proporcionados por Diffie-Hellman anónimo, más una firma de esos parámetros.
- **Intercambio de clave RSA, en el cual el servidor está usando RSA pero tiene una clave RSA sólo para firmar:** por lo tanto, el cliente no puede simplemente enviar una clave secreta cifrada con la clave pública del servidor. En vez de esto, el servidor debe crear un par de claves pública/privada RSA temporales y usar el

mensaje de intercambio de clave del servidor para enviar la clave pública. El contenido del mensaje incluye los dos parámetros de la clave pública RSA temporal (exponente y módulo; ver la Figura 3.8) más una firma de esos parámetros.

- **Forteza**

En cuanto a las firmas, se garantizan algunos detalles adicionales. Como es habitual, una firma se crea tomando el *hash* de un mensaje y cifrándolo con la clave privada del emisor. En este caso el *hash* se define como:

```
hash(ClientHello.random || ServerHello.random || ServerParams)
```

Así, el *hash* no sólo cubre los parámetros Diffie-Hellman o RSA, sino también los dos *nonces* extraídos de los mensajes *hello* iniciales. Esto previene ataques de repetición y falsificación. En el caso de una firma DSS, el *hash* se realiza usando el algoritmo SHA-1. En el caso de una firma RSA, se calcula un *hash* con MD5 y otro con SHA-1, y la concatenación de los dos *hash* (36 bytes) se cifra con la clave privada del servidor.

Después, un servidor no anónimo (servidor que no usa Diffie-Hellman anónimo) puede solicitar un certificado al cliente. El **mensaje certificate request** incluye dos parámetros: tipo de certificado (*certificate_type*) y autoridades de certificación (*certificateAuthorities*). El tipo de certificado indica el algoritmo de clave pública y su uso:

- RSA, sólo firma
- DSS, sólo firma
- RSA para Diffie-Hellman fijo; en este caso la firma se usa solamente para autenticación, enviando un certificado firmado con RSA
- DSS para Diffie-Hellman fijo; también se usa solamente para autenticación
- RSA para Diffie-Hellman efímero
- DSS para Diffie-Hellman efímero
- Fortezza

El segundo parámetro en el mensaje de solicitud de certificado es una lista de los diferentes nombres de autoridades de certificación aceptables.

El mensaje final de la Fase 2, y el único que es obligatorio, es el **mensaje server done**, enviado por el servidor para indicar el final del mensaje *hello* del servidor y demás mensajes relacionados. Después de enviar este mensaje, el servidor esperará para recibir una respuesta del cliente. Este mensaje no tiene parámetros.

Fase 3. Autentificación del cliente e intercambio de clave

Tras recibir el mensaje *server done*, el cliente debería verificar, si fuese necesario, que el servidor proporcionó un certificado válido y comprobar que los parámetros del *hello* del servidor son aceptables. Si todo es satisfactorio, el cliente envía uno o más mensajes respondiendo al servidor.

Si el servidor ha solicitado un certificado, el cliente comienza esta Fase enviando un **mensaje de certificado**. Si no hay disponible ningún certificado adecuado, el cliente envía la alerta *no_certificate*.

Luego está el **mensaje client key exchange**, que debe enviarse en esta Fase. El contenido del mensaje depende del tipo de intercambio de clave, como se explica a continuación:

- **RSA:** el cliente genera un valor de 48 bytes previo de la clave maestra (*pre-master secret*) y cifra con la clave pública obtenida del certificado del servidor o con la clave RSA temporal obtenida de un mensaje de intercambio de clave del servidor. Su uso en el cálculo de una clave maestra (*master secret*) se explica más tarde.
- **Diffie-Hellman efímero o anónimo:** se envían los parámetros Diffie-Hellman públicos del cliente.
- **Diffie-Hellman fijo:** los parámetros Diffie-Hellman públicos del cliente se enviaron en un mensaje de certificado, por lo que el contenido de este mensaje es nulo.
- **Forteza:** se envían los parámetros Fortezza del cliente.

Por último, en esta fase, el cliente puede enviar un **mensaje certificate verify** para proporcionar verificación explícita de un certificado del cliente. Este mensaje sólo se envía después de algún certificado del cliente que tenga capacidad de firma (por ejemplo, todos los certificados excepto los que contengan parámetros de Diffie-Hellman fijos). Este mensaje firma un código *hash* basado en los mensajes precedentes y se define de la siguiente manera:

```
CertificateVerify.signature.md5_hash  
MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1));  
Certificate.signature.sha_hash  
SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1));
```

donde *pad_1* y *pad_2* son los valores definidos anteriormente para el MAC, *handshake_messages* se refiere a todos los mensajes del protocolo de negociación (*Handshake*) enviados o recibidos comenzando en el saludo del cliente (*client_hello*) pero sin incluirlo, y clave maestra (*master_secret*) es la clave calculada, cuya generación se explicará más tarde en esta sección. Si la clave privada del usuario es DSS, entonces se usa ésta para cifrar el *hash* SHA-1. Si la clave privada del usuario es RSA, se usa ésta para cifrar la concatenación de los *hash* MD5 y SHA-1. En ambos casos, el fin es verificar que el cliente es el propietario de la clave privada en concordancia con el certificado del cliente. Incluso, si alguien está usando de forma fraudulenta el certificado del cliente, sería incapaz de enviar este mensaje.

Fase 4. Finalización

Esta Fase completa el establecimiento de una conexión segura. El cliente envía un **mensaje change cipher spec** y copia la especificación de cifrado pendiente en la especificación de cifrado operativa. Obsérvese que este mensaje no se considera parte del protocolo de negociación sino que se envía utilizando el protocolo Change Cipher Spec. El cliente entonces envía inmediatamente un **mensaje finished** usando los nuevos algoritmos, claves y secretos. El mensaje de finalización verifica que el intercambio de clave y los procesos de autentificación fueron satisfactorios. El contenido del mensaje de finalización es la concatenación de dos valores *hash*.

$$\text{MD5}(\text{master_secret} \parallel \text{pad2} \parallel \text{MD5}(\text{handshake_messages} \parallel \text{Sender} \parallel \text{master_secret} \parallel \text{pad1}))$$

$$\text{SHA}(\text{master_secret} \parallel \text{pad2} \parallel \text{SHA}(\text{handshake_messages} \parallel \text{Sender} \parallel \text{master_secret} \parallel \text{pad1}))$$

donde *Sender* es un código que identifica que el emisor es el cliente y *handshake_messages* son todos los datos de los mensajes del protocolo de negociación hasta este mensaje, pero sin incluir a éste último.

En respuesta a esos dos mensajes, el servidor envía su propio mensaje *change_cipher_spec*, transfiriendo la especificación de cifrado pendiente a la especificación de cifrado operativa, y envía su mensaje *finished*. En este punto la negociación se ha completado, y el servidor y el cliente pueden comenzar a intercambiar datos del nivel de aplicación.

CÁLCULOS CRIPTOGRÁFICOS

Otros dos aspectos de interés son la creación de la clave maestra compartida (*master secret*) mediante del intercambio de clave y la generación de parámetros criptográficos a partir de la clave maestra.

Creación de la clave maestra

La clave maestra compartida es un valor de un solo uso de 48 bytes (384 bits) generado para esta sesión mediante el intercambio seguro de claves. La creación se realiza en dos pasos. Primero, se intercambia un valor previo de la clave maestra (*pre_master_secret*). Segundo, ambas partes calculan la clave maestra. Para el intercambio del valor previo de la clave maestra existen dos posibilidades:

- **RSA:** el cliente genera un valor de 48 bytes previo a la clave maestra, lo cifra con la clave pública RSA del servidor y se lo envía a éste. El servidor descifra el texto cifrado con su clave privada para recuperar el valor previo de la clave maestra.
- **Diffie-Hellman:** el cliente y el servidor generan una clave pública de Diffie-Hellman. Después se las intercambian y cada lado realiza los cálculos Diffie-Hellman para crear el valor previo de la clave maestra.

Ahora ambos lados calculan la clave maestra de la siguiente forma:

```
master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret ||
                                              ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA('BB' || pre_master_secret ||
                                              ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA('CCC' || pre_master_secret ||
                                              ClientHello.random || ServerHello.random))
```

donde *ClientHello.random* y *ServerHello.random* son los dos valores *nonces* intercambiados en el mensaje de saludo inicial (*hello*).

Generación de parámetros criptográficos

La especificación de cifrado (*CipherSpecs*) necesita una clave para MAC de escritura del cliente (*client write MAC secret*), una clave para MAC de escritura del servidor (*server write MAC secret*), una clave de escritura del cliente (*client write key*), una clave de escritura del servidor (*server write key*), un vector de inicialización o IV (*initialization vector*) del cliente y un IV del servidor, lo cual se genera a partir de la clave maestra en ese orden. Estos parámetros se crean aplicando una operación *hash* a la clave maestra para producir una secuencia de bytes seguros con suficiente longitud como para disponer de todos los parámetros necesarios.

La generación de los parámetros a partir de la clave maestra usa el mismo formato que la generación de la clave maestra a partir del valor previo de ésta:

```
key_block = MD5(master_secret || SHA('A' || master_secret ||
                                         ServerHello.random || ClientHello.random)) ||
MD5(master_secret || SHA('BB' || master_secret ||
                                         ServerHello.random || ClientHello.random)) ||
MD5(master_secret || SHA('CCC' || master_secret ||
                                         ServerHello.random || ClientHello.random)) ||
...
```

hasta que se haya generado salida suficiente. El resultado de esta estructura algorítmica es una función pseudocaleatoria. La clave maestra puede verse como el valor semilla pseudocaleatorio de la función. Los números aleatorios del cliente y del servidor pueden verse como valores *salt* para complicar el criptoanálisis (véase el Capítulo 9 para el tratamiento del uso de valores *salt*).

TLS

TLS (*Transport Layer Security*) es una iniciativa de estandarización de la IETF cuyo objetivo es producir una versión estándar de Internet de SSL. TLS se define como un Es-tándar Propuesto de Internet en el RFC 2246. El RFC 2246 es muy similar a SSLv3. En esta sección se destacarán las diferencias.

Número de versión

El formato del registro de TLS es el mismo que el de SSL (Figura 7.4), y los campos de cabecera tienen el mismo significado. La única diferencia radica en los valores de versión. Para la versión actual de TLS, la versión mayor es tres y la menor, uno.

Código de Autentificación de Mensaje (MAC, *Message Authentication Code*)

Hay dos diferencias entre los esquemas MAC de SSLv3 y TLS: el algoritmo presente y el alcance de los cálculos del MAC. TLS utiliza el algoritmo HMAC definido en el RFC 2104. Recuérdese la definición del algoritmo HMAC que se ofrece en el Capítulo 3:

$$\text{HMAC}_K(M) = \text{H}[(K^+ \oplus \text{opad}) \parallel \text{H}[(K^+ \oplus \text{ipad}) \parallel M]]$$

donde

H = función *hash* anidada (para TLS, MD5 o SHA-1)

M = mensaje de entrada a HMAC

K^+ = clave secreta rellena con ceros a la izquierda de forma que su longitud sea igual a la longitud del bloque del código *hash* (para MD5 y SHA-1, la longitud de bloque = 512 bits)

$\text{ipad} = 00110110$ (36 en hexadecimal) repetido 64 veces (512 bits)

$\text{opad} = 01011100$ (5C en hexadecimal) repetido 64 veces (512 bits)

SSLv3 usa el mismo algoritmo, excepto que los bytes de relleno se concatenan con la clave secreta en vez de aplicárseles el XOR con la clave secreta rellena hasta la longitud del bloque. El nivel de seguridad debería ser el mismo en ambos casos.

Para TLS, el cálculo del MAC abarca los campos indicados en la siguiente expresión:

$$\text{HMAC_hash}(\text{MAC_write_secret}, \text{seq_num} \parallel \text{TLSCompressed.type} \parallel$$

$$\text{TLSCompressed.version} \parallel \text{TLSCompressed.length} \parallel \text{TLSCompressed.fragment}))$$

El cálculo del MAC cubre todos los campos cubiertos por el cálculo en SSLv3, más el campo `TLSCompressed.version`, que es la versión del protocolo que se está empleando.

Función pseudoaleatoria

TLS usa una función pseudoaleatoria conocida como PRF (*PseudoRandom Function*) para expandir valores secretos en bloques de datos útiles para generación de claves o validación. El objetivo es utilizar un valor secreto compartido relativamente pequeño, pero generar bloques de datos más grandes de forma segura contra los ataques a las funciones *hash* y a los MAC. La PRF se basa en la siguiente función de expansión de datos (Figura 7.7).

La función de expansión de datos utiliza el algoritmo HMAC, con MD5 o con SHA-1 como función *hash* de base. Como se puede ver, `P_hash` puede repetirse tantas veces como sea necesario para producir la cantidad de datos que se necesita. Por ejemplo, si `P_SHA-1` se usó para generar 64 bytes de datos, tendría que haberse iterado cuatro veces para obtener 80 bytes de datos, de los cuales los últimos 16 serían descartados. En este caso, `P_MD5` también tendría que haberse iterado cuatro veces, produciendo exactamente 64 bytes de datos. Obsérvese que cada repetición implica dos ejecuciones de HMAC, cada una de las cuales a su vez implica dos ejecuciones del algoritmo de *hash* base.

Para hacer que la PRF sea lo más segura posible, utiliza dos algoritmos *hash*, de manera que su seguridad está garantizada si uno de los algoritmos se mantiene seguro. PRF se define como

$$\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P_MD5}(\text{S1}, \text{label} \parallel \text{seed}) \oplus \text{P_SHA-1}(\text{S2}, \text{label} \parallel \text{seed})$$

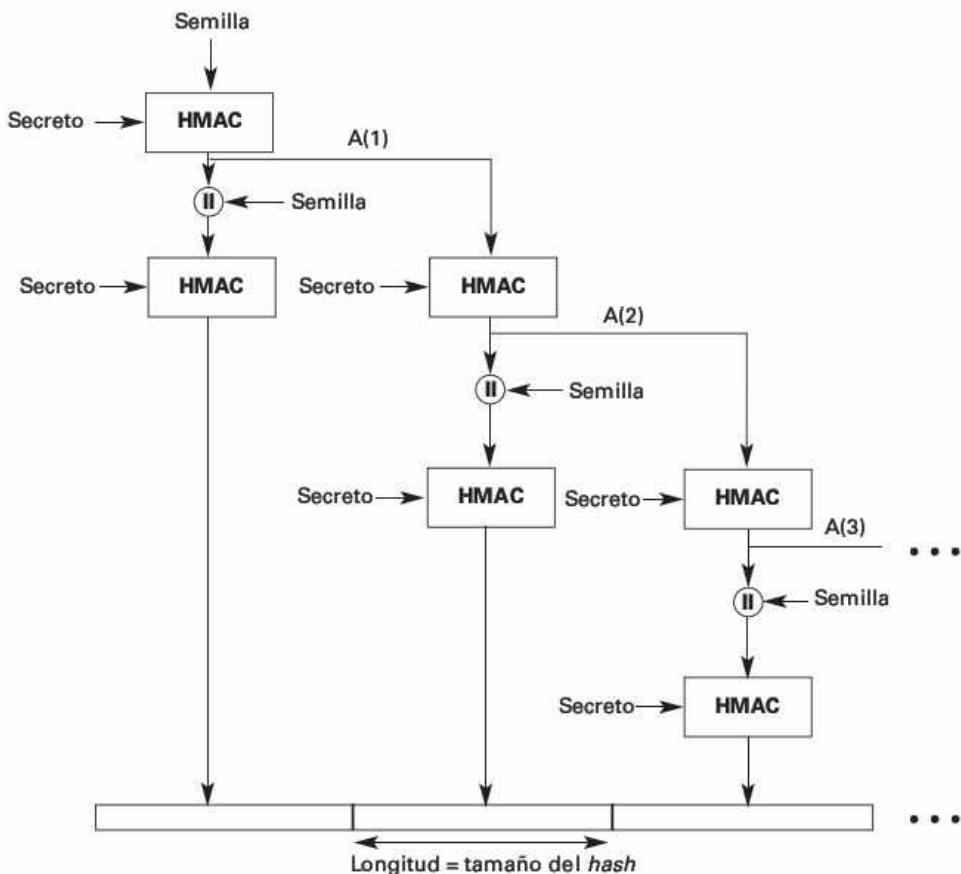


Figura 7.7 Función P_hash de TLS (secreto, semilla)

```
P_hash = HMAC_hash(secret, A(1) || seed) ||
        HMAC_hash(secret, A(2) || seed) ||
        HMAC_hash(secret, A(3) || seed) || ...
```

donde A() se define como

$$\begin{aligned}A(0) &= \text{seed} \\A(i) &= \text{HMAC_hash}(\text{secret}, A(i-1))\end{aligned}$$

PRF toma como entrada un valor secreto, una etiqueta de identificación y un valor «semilla» y produce una salida de longitud arbitraria. La salida se crea dividiendo el valor secreto en dos mitades (S1 y S2) y realizando P_hash sobre cada mitad, usando MD5 en una mitad y SHA-1 en la otra. Los dos resultados se suman con OR exclusivo para producir la salida; con este fin, P_MD5 generalmente tiene que ser repetido más veces que P_SHA-1 para producir igual cantidad de datos para la entrada de la función OR exclusivo.

Códigos de alerta

TLS admite todos los códigos de alerta definidos en SSLv3 con la excepción de la alerta no certificado (*no_certificate*). En TLS se ha definido un número de códigos adicionales, de los cuales, los siguientes son siempre fatales:

- **fallo de descifrado (*decryption_failed*):** un texto cifrado se descifró de manera no válida; bien porque no resultó un múltiplo par de la longitud de bloque o bien porque los valores de relleno, cuando se comprobaron, eran incorrectos.
- **sobrecarga de registro (*record_overflow*):** un registro TLS se recibió con una carga útil (texto cifrado) cuya longitud excede de $2^{14} + 2048$ bytes, o el texto descifrado resultó con una longitud mayor que $2^{14} + 1024$ bytes.
- **autoridad de certificación desconocida (*unknown_ca*):** se recibió una cadena o cadena parcial válida de certificados, pero el certificado no se aceptó porque el certificado de la CA no se pudo localizar o no se pudo comprobar con una CA conocida o confiable.
- **acceso denegado (*access_denied*):** se recibió un certificado válido, pero al aplicar el control de acceso, el emisor decidió no continuar con la negociación.
- **error de decodificación (*decode_error*):** un mensaje no se pudo decodificar porque algún campo estaba fuera de su rango especificado o la longitud del mensaje era incorrecta.
- **restricción de exportación (*export_restriction*):** se detectó una negociación en desacuerdo con las restricciones de exportación de la longitud de la clave.
- **versión del protocolo (*protocol_version*):** se reconoció la versión del protocolo que el cliente intentó negociar, pero el servidor no la admite.
- **seguridad insuficiente (*insufficient_security*):** devuelta en vez de fallo de la negociación cuando la negociación ha errado específicamente porque el servidor necesita cifradores más seguros que los que tiene el cliente.
- **error interno (*internal_error*):** un error interno no relacionado con el interlocutor ni con el funcionamiento del protocolo hace imposible continuar.

El resto de nuevas alertas son las siguientes:

- **error de descifrado (*decrypt_error*):** fallo de alguna operación criptográfica durante la negociación, entre las que se incluyen no poder verificar una firma, descifrar un intercambio de clave o validar un mensaje de finalización.
- **cancelado por usuario (*user_canceled*):** la negociación se ha cancelado por alguna razón no relacionada con fallos del protocolo.
- **no renegociación (*no_renegotiation*):** enviado por un cliente en respuesta a una solicitud de saludo (*hello_request*) o por el servidor en respuesta a un saludo del cliente (*client_hello*) después de la negociación inicial. Normalmente, cualquiera de esos mensajes activaría la renegociación, pero esta alerta indica que el emisor no es capaz de renegociar. Este mensaje siempre es un aviso de precaución.

Suites de cifrado

Hay pequeñas diferencias entre las *suites* de cifrado disponibles en SSLv3 y en TLS:

- **Intercambio de clave:** TLS permite todas las técnicas de intercambio de claves permitidas en SSLv3 con la excepción de Fortezza.
- **Algoritmos de cifrado simétrico:** TLS incluye todos los algoritmos de cifrado simétricos encontrados en SSLv3, con la excepción de Fortezza.

Tipos de certificado de cliente

TLS define los siguientes tipos de certificados requeridos en un mensaje de solicitud de certificado: `rsa_sign`, `dss_sign`, `rsa_fixed_dh` y `dss_fixed_dh`. Todos están definidos en SSLv3. Además, SSLv3 incluye `rsa_ephemeral_dh`, `dss_ephemeral_dh` y `fortezza_kea`. Diffie-Hellman efímero implica firmar los parámetros de Diffie-Hellman con RSA o DSS; para TLS, los tipos `rsa_sign` y `dss_sign` se usan para esa función; no es necesario un tipo separado de firma para firmar los parámetros de Diffie-Hellman. TLS no incluye el esquema Fortezza.

Mensajes *certificate_verify* y *finished*

En el mensaje *certificate_verify* de TLS, los *hash* MD5 y SHA-1 se calculan solamente sobre los mensajes de negociación. Recuérdese que para SSLv3, el cálculo del *hash* también incluía la clave maestra y los valores de relleno (*pads*). Estos campos extra no parecen suponer mayor seguridad.

Como con el mensaje de finalización en SSLv3, el de TLS es un *hash* basado en la clave maestra compartida, los mensajes previos de la negociación y una etiqueta que identifica al servidor o al cliente. Los cálculos son ligeramente diferentes. Para TLS tenemos:

$\text{PRF}(\text{master_secret}, \text{finished_label}, \text{MD5}(\text{handshake_messages}) \parallel \text{SHA-1}(\text{handshake_messages}))$

donde *finished_label* es la ristra *client finished* para el cliente y *server finished* para el servidor.

Cálculos criptográficos

El valor previo de la clave maestra (*pre_master_secret*) para TLS se calcula de la misma manera que en SSLv3. Como en SSLv3, en TLS se calcula la clave maestra como una función *hash* del valor previo de clave maestra y los dos números aleatorios (*random*) de los mensajes *hello*. El cálculo en TLS difiere del de SSLv3 y se define de la siguiente manera:

`master_secret =`

$\text{PRF}(\text{pre_master_secret}, \text{"master secret"}, \text{Client Hello.random} \parallel \text{Server_Hello.random})$

El algoritmo se aplica hasta producir 48 bytes pseudoaleatorios de salida. La generación del material del bloque de claves (claves secretas para MAC, claves de cifrado de sesión y vectores de inicialización) se define de la siguiente forma:

```
key_block =
    PRF(master_secret, "key expansion",
        SecurityParameters.server_random || SecurityParameters.client_random)
```

hasta que se haya generado la cantidad suficiente de salida. Como en SSLv3, el bloque de claves (*key_block*) es una función de la clave maestra y de los números aleatorios del cliente y del servidor, pero para TLS, el algoritmo es diferente.

Relleno

En SSL, la cantidad de relleno añadido antes del cifrado de los datos de usuario es la mínima necesaria, de manera que el tamaño total de los datos que se van a cifrar es un múltiplo de la longitud del bloque de cifrado. En TLS, la cantidad de relleno puede ser cualquier cantidad, como máximo hasta 255 bytes, para que el total sea múltiplo de la longitud del bloque de cifrado. Por ejemplo, si el texto en claro (o comprimido si se usa compresión) más el MAC más el byte de longitud del relleno (*padding.length*) es de 79 bytes, entonces la longitud de relleno, en bytes, puede ser uno, nueve, 17, etc. hasta 249. Puede usarse una longitud variable del relleno para frustrar ataques basados en el análisis de las longitudes de los mensajes intercambiados.

7.3 SET (SECURE ELECTRONIC TRANSACTION)

SET es una especificación abierta de cifrado y seguridad diseñada para proteger transacciones con tarjetas de crédito en Internet. La versión actual, SETv1, surgió a partir de la petición de MasterCard y Visa para crear un estándar de seguridad en febrero de 1996. Un gran número de compañías se implicaron en el desarrollo de las especificaciones iniciales, entre las que se incluyen IBM, Microsoft, Netscape, RSA, Terisa y Verisign. En 1996 se comenzó con numerosas pruebas del concepto, y en 1998 estaba disponible la primera serie de productos orientados a SET.

SET no es un sistema de pago en sí mismo, sino que más bien es un conjunto de protocolos de seguridad y formatos que permite a los usuarios emplear las infraestructuras existentes de pago por tarjeta de crédito, de forma segura, en una red abierta como Internet. Básicamente, SET proporciona tres servicios:

- Proporciona un canal de comunicación seguro entre todas las partes implicadas en la transacción
- Proporciona confianza mediante el uso de certificados digitales X.509v3
- Asegura la privacidad porque la información solamente está disponible cuando y donde es necesario a las partes de una transacción.

SET es una especificación compleja definida en tres libros publicados en mayo de 1997:

- **Libro 1:** descripción de negocios (*Business Description*) (80 págs.)
- **Libro 2:** guía del programador (*Programmer's Guide*) (629 págs.)
- **Libro 3:** definición formal del protocolo (*Formal Protocol Definition*) (262 págs.)

Esto hace un total de 971 páginas de especificaciones. Por el contrario, el tamaño de las especificaciones de SSLv3 es de 63 páginas y las de TLS de 80. Por tanto, en esta sección solamente se proporcionará un resumen de las muchas facetas de las especificaciones.

INTRODUCCIÓN A SET

Una buena manera de comenzar la discusión sobre SET es observando los requisitos que exigen los negocios a SET, sus características fundamentales y los participantes en las transacciones SET.

Requisitos

El Libro 1 de las especificaciones de SET enumera los siguientes requisitos para el procesamiento de pago con tarjetas de crédito en Internet y en otras redes:

- **Proporcionar confidencialidad de la información de pago y de pedido:** es necesario asegurar a los titulares de tarjetas que esta información está segura y accesible solamente a los receptores pertinentes. La confidencialidad también reduce el riesgo de fraude en la transacción, ya sea por una de las partes o por terceras partes dañinas. SET utiliza cifrado para proporcionar confidencialidad.
- **Asegurar la integridad de todos los datos transmitidos:** esto es, asegurar que no se producen cambios en el contenido de los mensajes SET durante la transmisión. Para proporcionar integridad se utilizan firmas digitales.
- **Proporcionar autenticación de que un titular de tarjeta es el usuario legítimo de una cuenta de tarjeta de crédito:** un mecanismo que enlace al titular de tarjeta con un número específico de cuenta reduce la incidencia de fraudes y los costes globales del procesamiento de pago. Para verificar que un titular de tarjeta es un usuario legítimo de una cuenta válida se utilizan firmas digitales y certificados.
- **Proporcionar autenticación de que el comerciante puede aceptar transacciones con tarjetas de crédito a través de su relación con una entidad financiera:** este requisito es complementario del anterior. Los titulares de tarjetas necesitan poder identificar a los comerciantes con los que realizan transacciones seguras. Otra vez, se utilizan firmas digitales y certificados.
- **Asegurar el uso de las mejores prácticas de seguridad y técnicas de diseño de sistemas para proteger a todas las partes legítimas en una transacción de comercio electrónico:** SET es una especificación probada basada en protocolos y algoritmos criptográficos muy seguros.
- **Crear un protocolo que no dependa de los mecanismos de seguridad de transporte ni que evite su uso:** SET puede operar con seguridad sobre una pila TCP/IP

«pura». Sin embargo, SET no interfiere en el uso de otros mecanismos de seguridad, tales como IPSec y SSL/TLS.

- **Facilitar y fomentar la interoperabilidad entre proveedores de software y redes:** los protocolos y formatos de SET son independientes de la plataforma *hardware*, del sistema operativo y del *software* Web.

Características fundamentales de SET

Para cumplir con los requisitos recién expuestos, SET incorpora las siguientes características:

- **Confidencialidad de información:** la información de la cuenta del titular y del pago está segura mientras viaja por la red. Una característica interesante e importante de SET es que evita que el comerciante conozca los números de las tarjetas de crédito; éstos se proporcionan solamente al banco emisor. Para proporcionar confidencialidad se utiliza cifrado convencional con DES.
- **Integridad de los datos:** la información enviada del pago efectuado con una tarjeta desde el usuario hacia el comerciante incluye información del pedido, datos personales e instrucciones de pago. SET garantiza que los contenidos del mensaje no son alterados durante el tránsito. Las firmas digitales RSA, usando códigos *hash* SHA-1, proporcionan integridad a los mensajes. Ciertos mensajes se protegen también con HMAC usando SHA-1.
- **Autenticación de cuenta del titular:** SET permite a los comerciantes verificar que un titular de tarjeta es un usuario legítimo de un número de cuenta de tarjeta válido. SET utiliza certificados digitales X.509v3 con firmas RSA para este propósito.
- **Autenticación del comerciante:** SET permite que los titulares verifiquen que un comerciante tiene una relación con una institución financiera que le permite aceptar tarjetas de pago. Con este fin, SET usa certificados digitales X.509v3 con firmas RSA.

Obsérvese que, al contrario que IPSec y SSL/TLS, SET proporciona sólo una opción para cada algoritmo criptográfico. Esto tiene sentido porque SET es una única aplicación con un único conjunto de requisitos, mientras que IPSec y SSL/TLS están destinados a una variedad de aplicaciones.

Participantes de SET

La Figura 7.8 indica los participantes en el sistema SET, el cual incluye los siguientes:

- **Titular de tarjeta o comprador:** en el entorno electrónico, los consumidores y entidades de compra interactúan con los comerciantes desde computadores personales en Internet. Un titular de tarjeta es un poseedor autorizado de una tarjeta de pago (por ejemplo, MasterCard, Visa) que ha sido emitida por un emisor.
- **Vendedor:** un vendedor es una persona u organización que tiene bienes y servicios para vender a los titulares de tarjetas. Normalmente, esos bienes y servicios se ofrecen en sitios web o por correo electrónico. Un vendedor que acepta tarjetas de pago debe tener una relación con un banco adquisidor.

- **Emisor o banco del comprador:** es una institución financiera como, por ejemplo, un banco, que proporciona la tarjeta de pago al titular. Generalmente, las cuentas se solicitan y se abren por correo electrónico o en persona. En última instancia, el emisor es el responsable del pago de los débitos del titular.
- **Adquisidor o banco del vendedor:** es una institución financiera que establece una cuenta con un vendedor y procesa las autorizaciones de tarjeta de pago y los pagos. Los vendedores, normalmente, aceptan más de una marca de tarjeta de crédito pero no quieren tratar con múltiples asociaciones financieras o con múltiples emisores individuales. El adquisidor proporciona autorización al vendedor indicándole que la cuenta de una tarjeta dada está activa y que la compra propuesta no excede el límite de crédito. El adquisidor también proporciona transferencia electrónica de pagos a la cuenta del vendedor. Posteriormente, el emisor realiza el reembolso al adquisidor a través de algún tipo de red de pago para transferencias electrónicas de fondos.
- **Pasarela de pago:** esta función la realiza el adquisidor o una tercera parte designada a tal efecto que procesa los mensajes de pago del vendedor. La pasarela de pago interacciona entre SET y las redes de pago existentes de los bancos para realizar funciones de autorización y de pago. El vendedor intercambia mensajes SET con la pasarela de pago en Internet, mientras que la pasarela de pagos tiene algunas conexiones directas o en red con los sistemas de procesamiento financiero del adquisidor.
- **Autoridad de certificación (CA):** ésta es una entidad de confianza para emitir certificados de clave pública X.509v3 para titulares, vendedores y pasarelas de pago. El éxito de SET dependerá de una infraestructura de CA disponible para este propósito. Como se indicó en capítulos anteriores, se usa una jerarquía de autoridades de certificación, de forma que los participantes no necesiten ser certificados directamente por una autoridad raíz.

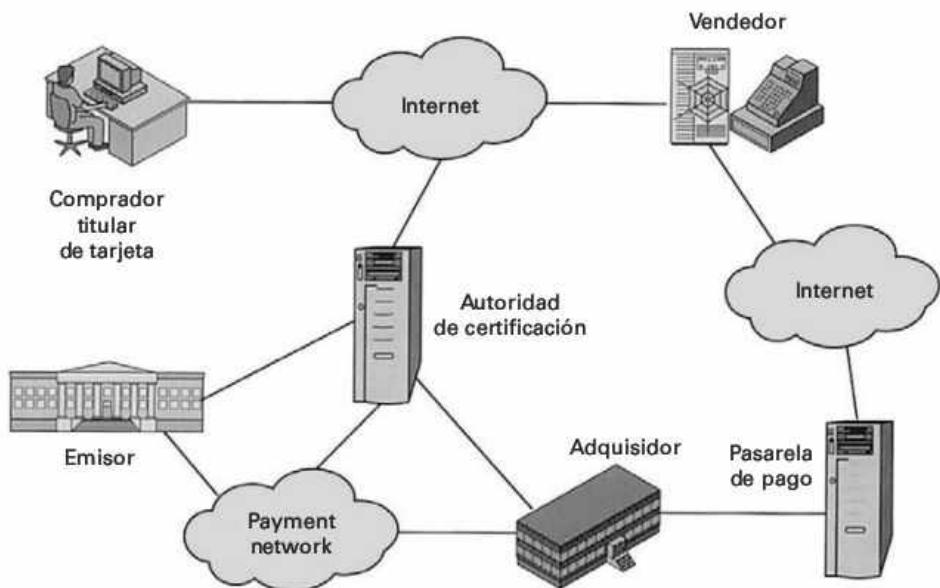


Figura 7.8 Componentes del comercio electrónico seguro

Ahora se describe brevemente la secuencia de acciones necesarias para realizar una transacción. Luego se verán algunos de los detalles criptográficos.

- 1. El comprador abre una cuenta.** El comprador obtiene una cuenta de tarjeta de crédito como, por ejemplo, MasterCard o Visa, con un banco capacitado para trabajar con pagos electrónicos y con SET.
- 2. El comprador recibe un certificado.** Despues de una verificación satisfactoria de identidad, el comprador recibe un certificado digital X.509v3 firmado por el banco. El certificado verifica la clave pública RSA del comprador y su fecha de caducidad. También establece, garantizada por el banco, la relación entre el par de claves del comprador y su tarjeta de crédito.
- 3. Los vendedores tienen sus propios certificados.** Un vendedor que acepta cierta marca de tarjeta debe poseer dos certificados para dos claves públicas propiedad del vendedor: uno para firmar mensajes y otro para el intercambio de claves. El vendedor también necesita una copia del certificado de clave pública de la pasarela de pago.
- 4. El comprador hace un pedido.** Éste es un proceso que podría implicar que el comprador navegue en el sitio web del vendedor para seleccionar productos y determinar su precio. Luego envía una lista de productos que desea comprar al vendedor, quien devuelve un formulario de pedido con la lista de productos, sus precios, el precio total y un número de pedido.
- 5. Se verifica al vendedor.** Además del formulario de pedidos, el vendedor envía una copia de su certificado, de manera que el comprador pueda verificar que está tratando con un comercio válido.
- 6. Se envían la orden de compra y la de pago.** El comprador envía la orden de compra y la información de pago al vendedor, junto con su certificado. La primera confirma la compra de los productos del formulario de pedido. La segunda contiene los detalles de la tarjeta de crédito. La información de pago se cifra para que el vendedor no pueda leerla. El certificado del comprador permite que éste sea verificado por el vendedor.
- 7. El vendedor solicita la autorización del pago.** El vendedor envía la información de pago a la pasarela de pago, solicitando autorización que confirme que el crédito disponible del comprador es suficiente para realizar la compra.
- 8. El vendedor confirma la orden de compra.** El vendedor envía confirmación de la orden de compra al comprador.
- 9. El vendedor suministra los productos o servicios.** El vendedor envía los productos o proporciona los servicios al comprador.
- 10. El vendedor solicita el pago.** Esta solicitud se envía a la pasarela de pago, la cual maneja todo el procesamiento de pagos.

FIRMA DUAL

Antes de ver en detalle el protocolo SET, vamos a tratar una innovación importante introducida en SET: la firma dual. La finalidad de la firma dual es asociar dos mensajes

destinados a dos receptores diferentes. En este caso, el comprador quiere enviar la información de orden de compra (OI, *order information*) al vendedor y la información de pago (PI, *payment information*) al banco. El vendedor no necesita conocer el número de la tarjeta de crédito del comprador, y el banco no necesita saber los detalles del pedido del comprador. Éste obtiene una protección extra en términos de privacidad manteniendo esos dos elementos separados. Sin embargo, los dos elementos deben estar asociados de forma que se puedan utilizar para resolver disputas si fuera necesario. La asociación es necesaria para que el comprador pueda probar que este pago está destinado a esta orden de compra y no para otro producto o servicio.

Para entender la necesidad de esta asociación, supongamos que los compradores envían al vendedor dos mensajes –uno de OI firmado y uno de PI firmado– y el vendedor pasa el de PI al banco. Si el vendedor puede capturar otra OI de este comprador, el vendedor podría reclamar que este OI va con el PI en vez de la OI original. La asociación de los mensajes evita esta situación.

La Figura 7.9 muestra el uso de la firma dual para satisfacer los requisitos reseñados en el párrafo anterior. El comprador calcula el *hash* (utilizando SHA-1) de la PI y el *hash* de la OI. Concatena los dos *hash* y calcula el *hash* del resultado. Por último, el comprador cifra el *hash* final con su clave privada de firma, creando la firma dual. La operación se puede resumir como:

$$DS = E_{KR_C}[H(H(PI) \parallel H(OI))]$$

donde KR_C es la clave de firma privada del comprador. Ahora supongamos que el vendedor está en posesión de la firma dual (DS, *dual signature*), la OI y el resumen del mensaje de PI (PIMD). El vendedor también tiene la clave pública del comprador, tomada del certificado de éste. Entonces el vendedor puede calcular las siguientes dos cantidades:

$$H(PIMD) \parallel H(OI) \text{ y } D_{KU_C}[DS]$$

donde KU_C es la clave pública del comprador. Si esas dos cantidades son iguales, entonces el vendedor ha verificado la firma. De manera similar, si el banco está en posesión de la DS, la PI y un resumen del mensaje de OI (OIMD) y la clave pública del comprador, entonces el banco puede calcular lo siguiente:

$$H(H(PI) \parallel OIMD) \text{ y } D_{KU_C}[DS]$$

Otra vez, si esas dos cantidades son iguales, entonces el banco ha verificado la firma. Resumiendo,

1. El vendedor ha recibido el mensaje de OI y ha verificado la firma.
2. El banco ha recibido el mensaje de PI y ha verificado la firma.
3. El comprador ha asociado los mensajes de OI y de PI y puede probar tal asociación.

Por ejemplo, supongamos que el vendedor desea sustituir una OI por otra durante esta transacción para obtener provecho. Entonces tendría que encontrar otra OI cuyo *hash* coincidiera con el OIMD existente. Con SHA-1, esto no se considera factible. Por eso, el vendedor no puede asociar una OI diferente a la que se corresponde con el PI dado.

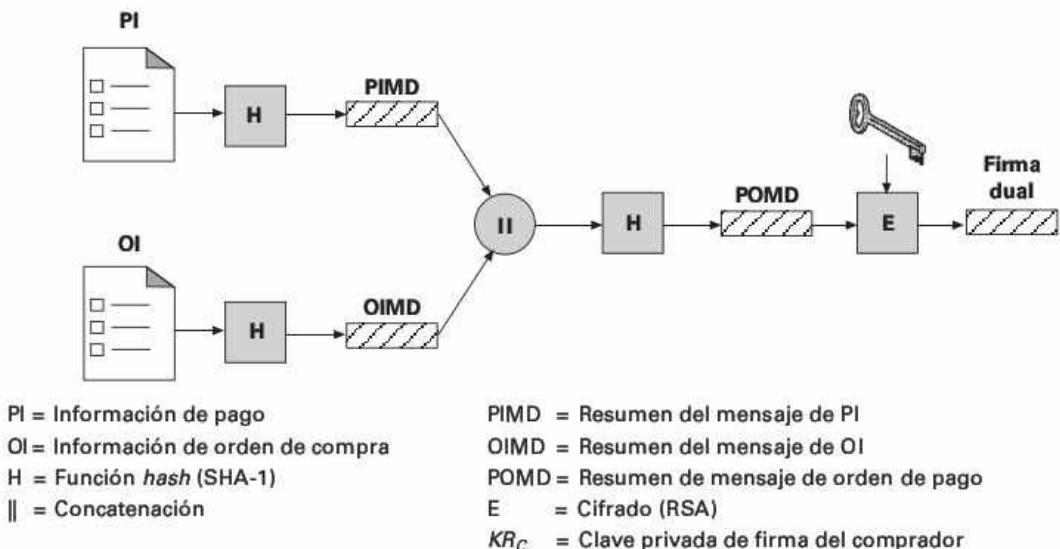


Figura 7.9 Construcción de la firma dual

PROCESAMIENTO DEL PAGO

La Tabla 7.3 enumera los tipos de transacciones permitidos por SET. A continuación veremos algunos detalles de las transacciones siguientes:

- Solicitud de compra
- Autorización de pago
- Captura de pago

Solicitud de compra

Antes de que comience el intercambio de solicitud de compra, el comprador ha completado la navegación en el sitio web, la selección de productos o servicios y la solicitud de compra. El final de esta fase preliminar ocurre cuando el vendedor envía un formulario de compra completo al comprador. Todos los pasos precedentes se realizan sin la intervención de SET.

El intercambio de solicitud de compra se compone de cuatro mensajes: solicitud de inicio, respuesta de inicio, solicitud de compra y respuesta de compra.

Para poder enviar mensajes SET al vendedor, el comprador debe tener una copia de los certificados del vendedor y de la pasarela de pago. El comprador solicita los certificados en el mensaje de **solicitud de inicio**, enviado al vendedor. Este mensaje incluye la clase de tarjeta de crédito que usa el comprador. El mensaje también incluye un identificador asignado por el comprador para el par solicitud/respuesta y un valor *nonce* utilizado para garantizar la validez temporal.

Tabla 7.3 Tipos de transacciones SET

Registro de titular	Los titulares deben registrarse con una CA antes de que puedan enviar mensajes SET a los vendedores.
Registro de vendedor	Los vendedores deben registrarse con una CA antes de que puedan intercambiar mensajes SET con compradores y pasarelas de pago.
Solicitud de compra	Mensaje del comprador al vendedor que contiene la OI para el vendedor y la PI para el banco.
Autorización de pago	Intercambio entre vendedor y pasarela de pago que autoriza una cantidad determinada para el pago de una compra con una cuenta de tarjeta de crédito dada.
Captura de pago	Permite al vendedor solicitar el pago a la pasarela de pago.
Informe y estado de certificado	Si la CA es incapaz de completar el procesamiento de una solicitud de certificado rápidamente, enviará una respuesta al comprador o al vendedor indicando que el solicitante deberá volver a realizar la comprobación más tarde. El titular o el vendedor envía el mensaje de informe de certificado para determinar el estado de la solicitud de éste y recibirla si la solicitud ha sido aceptada.
Informe de compra	Permite al comprador comprobar el estado de procesamiento de una orden de compra después de recibir la respuesta de compra. Obsérvese que este mensaje no incluye información como el estado de los productos solicitados sino que indica el estado de autorización, captura y procesamiento del crédito.
Deshacer autorización	Permite a los vendedores corregir solicitudes de autorización previas. Si la orden de compra no se puede completar, el vendedor deshace la autorización completa. Si parte de la orden de pedido no se puede completar (como cuando se ordena la devolución de alguno o algunos de los productos solicitados), el vendedor «deshace» o descuenta la cantidad correspondiente de la autorización.
Deshacer captura	Permite a los vendedores corregir errores en las solicitudes de captura tales como cantidades de las transacciones introducidas erróneamente por un empleado.
Crédito	Permite a un vendedor realizar un abono en la cuenta del comprador ya sea por la devolución de los productos o porque se dañaron durante la entrega. Obsérvese que el mensaje de crédito de SET lo inicia siempre el vendedor, nunca el comprador. Todas las comunicaciones entre comprador y vendedor que hayan dado como resultado el procesamiento de un crédito (abono) ocurren fuera de SET.
Deshacer crédito	Permite a un vendedor corregir una solicitud de crédito previa.
Solicitud de certificado de pasarela de pago	Permite a un vendedor consultar a la pasarela de pago y recibir una copia de los certificados actuales de firma y de intercambio de clave de la pasarela.
Administración por lotes	Permite a un vendedor comunicar información a la pasarela de pago referente a lotes del vendedor
Mensaje de error	Indica que el que contesta rechaza un mensaje porque fallan las comprobaciones de verificación del contenido o del formato.

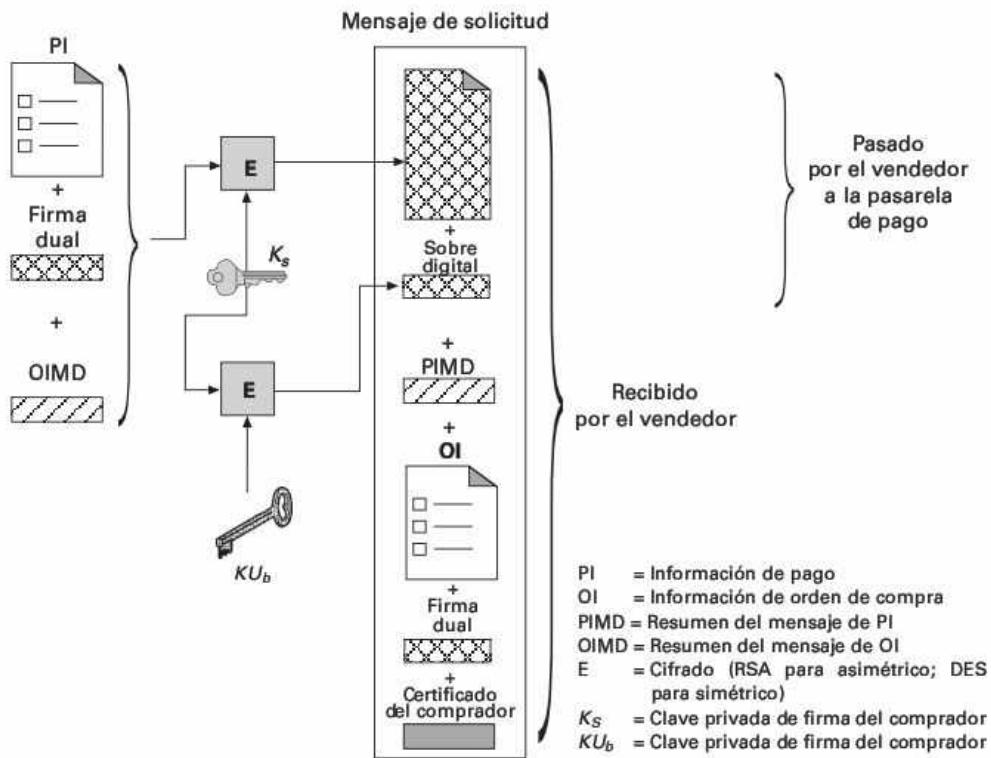


Figura 7.10 El comprador envía la solicitud de compra

El vendedor genera una respuesta y la firma con su clave de firma privada. La respuesta incluye el *nonce* recibido del comprador, otro *nonce* para que el comprador lo devuelva en el próximo mensaje y un identificador de transacción para esta transacción de compra. Además de la respuesta firmada, el mensaje de **Respuesta de Inicio** incluye el certificado de firma del vendedor y el certificado de intercambio de clave de la pasarela de pago.

El comprador verifica los certificados del vendedor y de la pasarela por medio de sus respectivas firmas de la autoridad de certificación y entonces crea la OI y la PI. El identificador de la transacción asignado por el vendedor se coloca en la OI y en la PI. La OI no contiene datos explícitos del pedido como son el número y el precio de los productos. Lo que contiene es una referencia de la orden de compra generada durante el intercambio entre comprador y vendedor en la fase anterior al primer mensaje de SET. Después, el comprador prepara el mensaje de **Solicitud de Compra** (Figura 7.10). Para este propósito, el comprador genera una clave de cifrado simétrico de un solo uso, K_s . El mensaje incluye lo siguiente:

1. **Información relativa a la adquisición.** Esta información será reenviada a la pasarela de pago por el vendedor y consiste en

- La PI

- La firma dual, calculada sobre PI y OI, firmada con la clave privada de firma del comprador
- El resumen del mensaje de la OI (OIMD)

El OIMD es necesario para que la pasarela de pago verifique la firma dual, como se explicó anteriormente. Todos esos elementos se cifran con K_S . El elemento final es

- El sobre digital. Éste se forma cifrando K_S con la clave pública de intercambio de clave de la pasarela de pago. Se le llama sobre digital porque debe abrirse (descifrarse) antes de que se puedan leer los elementos mencionados previamente.

El valor de K_S no está disponible para el vendedor. De esta manera, éste no puede leer la información relativa al pago.

2. Información relativa a la orden de compra. Esta información es necesaria para el vendedor y consiste en

- La OI
- La firma dual, calculada sobre la PI y la OI, firmada con la clave privada de firma del comprador
- El resumen del mensaje de PI (PIMD)

El PIMD lo necesita el vendedor para verificar la firma dual. Obsérvese que la OI se envía sin cifrar (en claro).

3. Certificado del comprador. Contiene la clave pública de firma del comprador. Lo necesitan el vendedor y la pasarela de pago.

Cuando el vendedor recibe el mensaje de solicitud de compra, realiza las siguientes acciones (Figura 7.11):

1. Verifica los certificados del comprador mediante sus firmas de la CA.
2. Verifica la firma dual utilizando la clave de firma pública del comprador. Esto garantiza que la orden de pedido no ha sido modificada durante el tránsito y que ha sido firmada con la clave privada del comprador.
3. Procesa el pedido y reenvía la información de pago a la pasarela de pago para la autorización (que se describe más tarde).
4. Envía una respuesta de compra al comprador.

El mensaje de **Respuesta de Compra** incluye un bloque de respuesta que reconoce el pedido y referencia el número de transacción correspondiente. Este bloque lo ha firmado el vendedor con su clave de firma privada. El bloque y su firma se envían al comprador, junto con el certificado de firma del vendedor.

Cuando el *software* del comprador recibe el mensaje de respuesta de compra, verifica el certificado del vendedor y luego verifica la firma del bloque de respuesta. Finalmente, realiza algunas acciones basadas en la respuesta, tales como mostrar un mensaje en la pantalla del usuario o actualizar una base de datos con el estado del pedido.

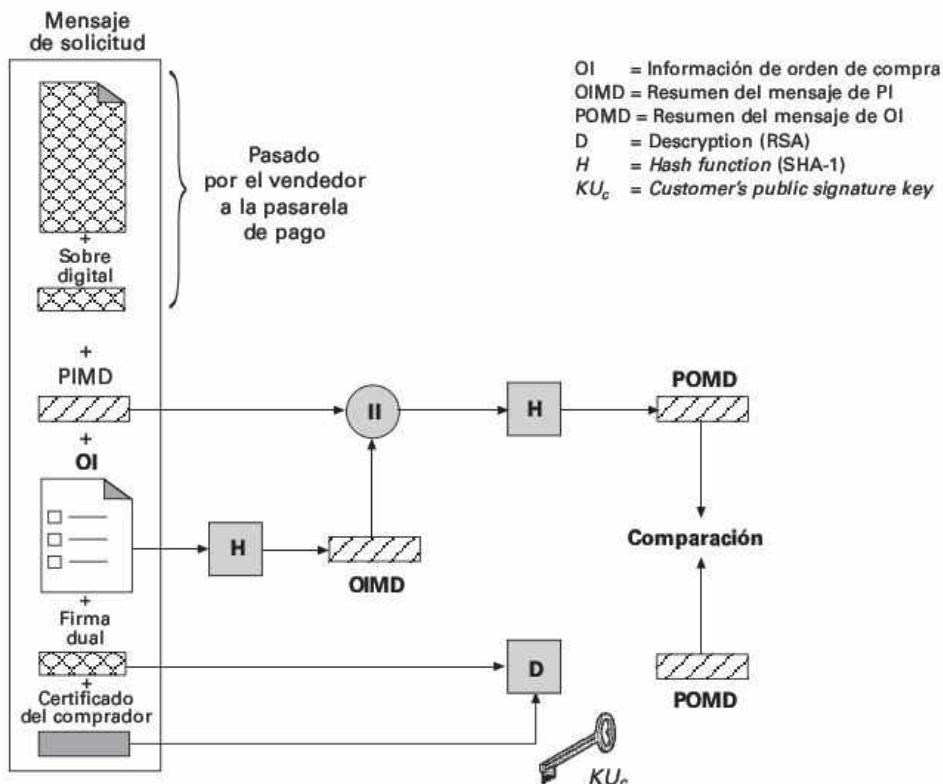


Figura 7.11 El vendedor verifica la solicitud de compra del comprador

Autorización de pago

Durante el procesamiento de una orden de pedido de un comprador, el vendedor autoriza la transacción con la pasarela de pago. La autorización de pago asegura que la transacción fue aprobada por el emisor de la tarjeta. La autorización de pago garantiza que el vendedor recibirá el dinero; por lo tanto, el vendedor puede proporcionar los servicios o productos al comprador. El intercambio de autorización de pago consta de dos mensajes: solicitud de Autorización y Respuesta de Autorización.

El vendedor envía a la pasarela de pago un mensaje de **Solicitud de Autorización** que contiene lo siguiente:

1. **Información relativa a la adquisición.** Esta información se obtuvo del comprador y consta de lo siguiente:
 - La PI
 - La firma dual, calculada sobre la PI y la OI, firmada con la clave privada de firma del comprador
 - El resumen del mensaje de OI (OIMD)
 - El sobre digital

- 2. Información relativa a la autorización.** Esta información es generada por el vendedor y consta de
 - Un bloque de autorización que incluye el identificador de la transacción, firmado con la clave de firma privada del vendedor y cifrado con una clave simétrica de un solo uso generada por el vendedor
 - Un sobre digital. Éste se forma cifrando la clave de un solo uso con la clave de intercambio de clave pública de la pasarela de pago.
- 3. Certificados.** El vendedor incluye el certificado de clave de firma del comprador (utilizado para verificar la firma dual), el certificado de clave de firma del vendedor (utilizado para verificar la firma del vendedor) y el certificado de intercambio de clave del vendedor (necesario en la respuesta de la pasarela de pago).

La pasarela de pago realiza las siguientes tareas:

- 1.** Verifica todos los certificados
- 2.** Descifra el sobre digital del bloque de autorización para obtener y descifrar la clave simétrica de éste
- 3.** Verifica la firma del vendedor contenida en el bloque de autorización
- 4.** Descifra el sobre digital del bloque de pago para obtener y descifrar la clave simétrica de éste
- 5.** Verifica la firma dual contenida en el bloque de pago
- 6.** Verifica que el identificador de la transacción recibido del vendedor coincide con el recibido en el mensaje de PI (indirectamente) del comprador
- 7.** Solicita y recibe una autorización del emisor de la tarjeta

Una vez obtenida la autorización del emisor de la tarjeta, la pasarela de pago devuelve un mensaje de **Respuesta de Autorización** al vendedor que incluye los siguientes elementos:

- 1. Información relativa a la autorización.** Incluye un bloque de autorización, firmado con la clave de firma privada de la pasarela y cifrado con una clave simétrica de un solo uso generada por la pasarela. También incluye un sobre digital que contiene la clave de un solo uso cifrada con la clave de intercambio de clave pública del vendedor.
- 2. Información del bono de captura.** Esta información se utilizará para efectuar el pago más tarde. Este bloque tiene la misma forma que el (1), es decir, un bono de captura firmado y cifrado junto con un sobre digital. Este bono no es procesado por el vendedor, sino que debe ser devuelto, como está, con una solicitud de pago.
- 3. Certificado.** El certificado de clave de firma de la pasarela.

Con la autorización de la pasarela, el vendedor puede proporcionar los productos o servicios al comprador.

Captura de pago

Para cobrar el dinero, el vendedor realiza una transacción de captura de pago con la pasarela de pago, que consta de un mensaje de solicitud de captura y otro de respuesta de captura.

Para el mensaje de **Solicitud de Captura**, el vendedor genera, firma y cifra un bloque de solicitud de captura, que incluye la cantidad que se va a cobrar y el identificador de la transacción. El mensaje también incluye el bono de captura cifrado recibido recientemente (en la respuesta de autorización) para esta transacción, así como los certificados de clave de firma y clave de intercambio de clave del vendedor.

Cuando la pasarela de pago recibe el mensaje de solicitud de captura, descifra y verifica el bloque de solicitud de captura y descifra y verifica el bloque del bono de captura. Entonces comprueba la consistencia entre la solicitud de captura y el bono de captura. Luego crea una solicitud de liquidación que se envía al emisor de la tarjeta a través de una red privada de servicios de pago. Esta solicitud hace que se transfieran los fondos a la cuenta del vendedor.

La pasarela, entonces, notifica al vendedor la realización del pago en un mensaje de **Respuesta de Captura**. El mensaje incluye un bloque de respuesta de captura firmado y cifrado por la pasarela. También incluye el certificado de clave de firma de la pasarela. El *software* del vendedor almacena la respuesta de captura que se utiliza para su comprobación con los pagos recibidos del adquisidor.

7.4 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

[RESC01] trata en detalle SSL y TLS.

La mejor descripción sobre SET se encuentra en el Libro 1 de las especificaciones, disponible en el sitio web SET de MasterCard. Otra descripción general muy útil se encuentra en [MACG97]. [DREW99] es también una buena fuente.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; y Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG244978-00, 1997. Disponible en www.redbooks.ibm.com.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

Sitios web recomendados:

- **Página web SSL de Netscape:** contiene las especificaciones de SSL.
- **Transport Layer Security Charter:** últimos RFCs y borradores de Internet sobre TLS.
- **Página web SET de MasterCard:** últimos documentos sobre SET, glosario de términos, información de aplicación.
- **SETCor:** últimos documentos SET y glosario de términos, entre otros.

7.5 PALABRAS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

PALABRAS CLAVE

Adquisidor o banco del vendedor	Firma dual	SSL (<i>Secure Socket Layer</i>)
Autoridad de certificación (CA)	Pasarela de pago	TLS (<i>Transport Layer Security</i>)
Comprador/titular	SET (<i>Secure Electronic Transaction</i>)	Vendedor
Emisor de tarjeta		

PREGUNTAS DE REPASO

- 7.1.** ¿Cuáles son las ventajas de cada uno de los tres enfoques de la Figura 7.1?
- 7.2.** ¿De qué protocolos se compone SSL?
- 7.3.** ¿Cuál es la diferencia entre una conexión SSL y una sesión SSL?
- 7.4.** Cita y describe brevemente los parámetros que definen un estado de sesión de SSL.
- 7.5.** Cita y describe brevemente los parámetros que definen una conexión de sesión de SSL.
- 7.6.** ¿Qué servicios proporciona el protocolo *Record* de SSL?
- 7.7.** ¿Qué pasos implica la transmisión del protocolo *Record* de SSL?
- 7.8.** Enumera y define brevemente las principales categorías de participantes de SET.
- 7.9.** ¿Qué es una firma dual y cuál es su propósito?

PROBLEMAS

- 7.1.** ¿Por qué hay un protocolo *Change Cipher Spec* separado en SSL y TLS, en vez de incluir un mensaje *change_cipher_spec* en el protocolo *Handshake*?
- 7.2.** Considera las siguientes amenazas a la seguridad Web y describe cómo se contrarresta cada una mediante una característica particular de SSL.
 - a)** Ataque criptoanalítico de fuerza bruta: una búsqueda exhaustiva del espacio de claves para un algoritmo de cifrado convencional.
 - b)** Ataque por diccionario de texto claro conocido: muchos mensajes contienen texto claro predecible, como el comando GET de HTTP. Un atacante construye un diccionario que contiene todos los posibles cifrados del mensaje de texto claro conocido. Cuando un mensaje cifrado es interceptado, el atacante extrae la porción que contiene el texto claro conocido cifrado y busca el texto cifrado en el diccionario. El texto cifrado debería coincidir con una entrada del diccionario cifrada con la misma clave secreta que el mensaje. Si

hubiera varias coincidencias, se puede probar aplicando cada una a todo el texto cifrado para determinar cuál es la correcta. Este ataque es especialmente eficaz con claves de pequeño tamaño (por ejemplo, claves de 40 bits).

- c) Ataque de repetición: los mensajes de negociación de SSL anteriores son repetidos.
 - d) Ataque de interceptación: un atacante se interpone durante el intercambio de clave, actuando como el cliente para el servidor y como el servidor para el cliente.
 - e) Escucha de contraseñas: se están observando las contraseñas de HTTP o el tráfico de otras aplicaciones.
 - f) Suplantación IP: uso de direcciones IP «robadas» para conseguir que un computador acepte datos fraudulentos.
 - g) Secuestro de IP: el atacante interrumpe una conexión activa y autenticada entre dos computadores haciéndose pasar por uno de ellos.
 - h) Inundación SYN: un atacante envía mensajes SYN de TCP solicitando una conexión, pero no responde al mensaje final para establecer ésta por completo. El módulo TCP atacado normalmente deja la conexión abierta durante unos pocos minutos. La repetición continua de mensajes SYN puede atascar el módulo TCP.
- 7.3** ¿Es posible en SSL, basándote en lo aprendido en este capítulo, que un receptor reordene bloques de registro de SSL que llegan desordenados? Si es así, explica cómo se puede hacer. Si no, ¿por qué no?

CAPÍTULO 8

Seguridad en la gestión de redes

8.1 Conceptos básicos de SNMP

Arquitectura de gestión de red
Arquitectura del protocolo de gestión de red
Agentes *proxy*
SNMPv2

8.2 Comunidades SNMPv1

Comunidades y nombres de comunidades
Servicio de autentificación
Política de acceso
Servicio *proxy*

8.3 SNMPv3

Arquitectura SNMP
Procesamiento de mensajes y modelo de seguridad de usuario
Control de acceso basado en vistas

8.4 Bibliografía y sitios web recomendados

8.5 Términos clave, preguntas de repaso y problemas

Términos clave
Preguntas de repaso
Problemas

El control de una gran fuerza es, en principio, el mismo que el de unos pocos hombres; sólo es cuestión de dividirlos.

El arte de la guerra, SUN Tzu

La importancia de las redes y los sistemas de procesamiento distribuido ha aumentado en el ámbito de los negocios, el gobierno y otras organizaciones. En una institución dada, se tiende a establecer redes mayores y más complejas para dar cabida a más aplicaciones y más usuarios. A medida que se produce el crecimiento de estas redes, se hacen evidentes dos hechos fundamentales:

- La red, sus recursos asociados y las aplicaciones distribuidas se convierten en elementos indispensables para la organización.
- Pueden fallar más cosas, inhabilitando la red o una parte de ella o reduciendo el rendimiento a niveles inaceptables.

Una red extensa no se puede instalar y gestionar sólo con el esfuerzo humano. La complejidad de un sistema de este tipo exige el uso de herramientas automatizadas de gestión de red. Si, además, la red incluye equipamiento de múltiples distribuidores, aumenta la necesidad de disponer de dichas herramientas, así como la dificultad en suministrarlas. En respuesta a esta necesidad, se han desarrollado estándares que tratan sobre la gestión de red, que abarcan servicios, protocolos y bases de datos de información de gestión. Hasta ahora, el estándar más utilizado es el SNMP (*Simple Network Management Protocol*).

Desde su publicación en 1988, SNMP se ha usado en un número de redes cada vez mayor y en entornos cada vez más complejos. A medida que se extendía el uso de SNMP, se necesitaban nuevas funcionalidades para ajustarse a las nuevas demandas. También, la importancia de proporcionar seguridad como parte de la gestión de redes se hizo cada vez más evidente. Para mejorar la funcionalidad, se definió la versión 2 de SNMP¹. Las mejoras de la seguridad fueron promulgadas en SNMPv3.

Este capítulo describe la herramienta rudimentaria de seguridad de SNMPv1 y el conjunto de características de seguridad mucho más extenso que ofrece SNMPv3.

8.1 CONCEPTOS BÁSICOS DE SNMP

Esta sección ofrece una descripción de los conceptos básicos de SNMP.

ARQUITECTURA DE GESTIÓN DE RED

Un sistema de gestión de red es un conjunto de herramientas para supervisar y controlar la red y que se considera integrado en los siguientes sentidos:

¹ La versión 2 se conoce como SNMPv2. Para distinguir la versión original de la nueva, la original se conoce en la actualidad como SNMPv1.

- Una sola interfaz de operador con un grupo de comandos potentes, pero sencillos, para realizar la mayoría de las tareas de gestión de red o, incluso, todas.
- Una cantidad mínima de equipamiento separado. Es decir, la mayor parte del *hardware* y el *software* necesarios para la gestión de red está incorporada en el equipo del usuario.

Un sistema de gestión de red se compone de *hardware* y *software* adicionales implementados entre los componentes de red existentes. El *software* que se emplea en las tareas de gestión de red reside en los *host* y en los procesadores de comunicaciones (por ejemplo, procesadores frontales (*front-end*), controladores de grupos de terminales). Un sistema de gestión de red está diseñado para ver toda la red como una arquitectura unificada, con direcciones y etiquetas asignadas a cada punto y los atributos específicos de cada elemento y enlace conocidos por el sistema. Los elementos activos de la red proporcionan una retroalimentación regular de la información de estado al centro de control de red.

El modelo de gestión de red que se usa para SNMP incluye los siguientes elementos fundamentales:

- Estación de gestión
- Agente de gestión
- Base de información de gestión
- Protocolo de gestión de red

La **estación de gestión** es, normalmente, un dispositivo autónomo, pero puede ser una capacidad implementada en un sistema compartido. En cualquier caso, la estación de gestión sirve como interfaz para que el administrador de red humano acceda al sistema de gestión de red. La estación de gestión, como mínimo, tendrá:

- Un conjunto de aplicaciones de gestión para el análisis de los datos, recuperación de fallos, etc.
- Una interfaz mediante la cual el administrador puede supervisar y controlar la red
- La capacidad de trasladar los requerimientos del administrador de red a la supervisión y el control real de elementos remotos en la red
- Una base de datos de información extraída de las MIB o bases de datos de información de gestión de todas las entidades gestionadas en la red

Sólo los dos últimos elementos están sujetos a la normalización de SNMP.

El otro elemento activo en el sistema de gestión de red es el **agente de gestión**. Las plataformas principales, como *hosts*, puentes, *routers* y concentradores (*hubs*) se pueden equipar con SNMP para que puedan ser gestionadas desde una estación de gestión. El agente responde a las solicitudes de información y acción provenientes de una estación de gestión y puede, de forma asíncrona, proporcionar a la estación de gestión información importante aunque no haya sido solicitada.

Para gestionar los recursos de la red, cada recurso se representa como un objeto. Básicamente, un objeto es una variable de datos que representa un aspecto del agente gestionado. El conjunto de objetos se conoce como **base de información de gestión** (MIB, *Management Information Base*). La MIB funciona como un conjunto de puntos de ac-

ceso en el agente para la estación de gestión. Estos objetos están estandarizados en los distintos tipos de sistemas (por ejemplo, todos los puentes tienen los mismos objetos de gestión). Una estación de gestión realiza la función de supervisión mediante la recuperación del valor de los objetos de la MIB. También puede hacer que una acción tenga lugar en un agente o puede cambiar la configuración de un agente modificando el valor de objetos específicos.

La estación de gestión y los agentes se comunican mediante un **protocolo de gestión de red**. El protocolo que se emplea para la gestión de redes TCP/IP es el protocolo sencillo de gestión de red (SNMP, *Simple Network Management Protocol*). Este protocolo incluye las siguientes capacidades fundamentales:

- **Get** permite a la estación de gestión **obtener** el valor de los objetos del agente
- **Set** permite a la estación de gestión **establecer** los valores de los objetos del agente
- **Notify**: permite al agente **notificar** los hechos significativos a la estación de gestión

ARQUITECTURA DEL PROTOCOLO DE GESTIÓN DE RED

En 1988 se publicó la especificación para SNMP y rápidamente se convirtió en el estándar predominante de gestión de red. Una serie de distribuidores ofrecen estaciones de gestión de red autónomas basadas en SNMP, y la mayoría de los vendedores de puertos, *routers*, estaciones de trabajo y computadores personales ofrecen paquetes de agente SNMP que permiten que sus productos sean gestionados por una estación de gestión SNMP.

Como el nombre sugiere, SNMP es una herramienta sencilla para la gestión de red. Define una base de información de gestión (MIB), limitada y de fácil implementación, de variables escalares y tablas de dos dimensiones, y define un protocolo más eficiente para permitir que un administrador obtenga y establezca variables de la MIB y que un agente emita notificaciones no solicitadas, llamadas *interceptaciones (traps)*. La robustez de SNMP se basa en esta simplicidad. SNMP se implementa fácilmente y consume una cantidad moderada de recursos de procesador y de red. Además, las estructuras del protocolo y de la MIB son lo suficientemente sencillas para facilitar la interacción entre las estaciones de gestión y *software* de agente de diversos vendedores.

Las tres especificaciones fundamentales son:

- **Estructura e identificación de la información de gestión para redes basadas en TCP/IP (RFC 1155):** describe cómo se definen los objetos gestionados contenidos en la MIB
- **MIB para la gestión de red de redes basadas en TCP/IP: MIB-II (RFC 1213):** describe los objetos gestionados contenidos en la MIB
- **Protocolo simple de gestión de red (RFC 1157):** define el protocolo empleado para gestionar estos objetos

SNMP se diseñó como protocolo del nivel de aplicación para formar parte de la *suite* de protocolos TCP/IP. Fue pensado para operar sobre UDP (*User Datagram Protocol*), definido en la RFC 768. Para una estación de gestión independiente, un proceso de administrador controla el acceso a la MIB central en la estación de gestión y proporciona una

interfaz al administrador de red. El proceso de administrador consigue la gestión de la red usando SNMP, que se implementa sobre UDP, IP y los protocolos dependientes de la red de que se trate (por ejemplo, Ethernet, FDDI, X.25).

Cada agente también debe implementar SNMP, UDP e IP. Además, hay un proceso agente que interpreta los mensajes SNMP y controla la MIB del agente. Para un dispositivo agente que dé soporte a otras aplicaciones, como FTP, se requiere TCP y UDP.

La Figura 8.1 ilustra el contexto del protocolo de SNMP. Desde una estación de gestión se emiten tres tipos de mensajes SNMP en nombre de una aplicación de gestión: GetRequest, GetNextRequest y SetRequest. Los dos primeros son variaciones de la función Get. Los tres mensajes son reconocidos por el agente mediante un mensaje GetResponse, que se entrega a la aplicación de gestión. Además, un agente puede emitir un mensaje *trap* (de interceptación) en respuesta a un hecho que afecta a la MIB y a los recursos gestionados implicados.

Debido a que SNMP se basa en UDP, que es un protocolo no orientado a conexión, SNMP es, en sí mismo, un protocolo no orientado a conexión. No se mantienen conexiones entre una estación de gestión y sus agentes. En vez de ello, cada intercambio es una transacción separada entre una estación de gestión y un agente.

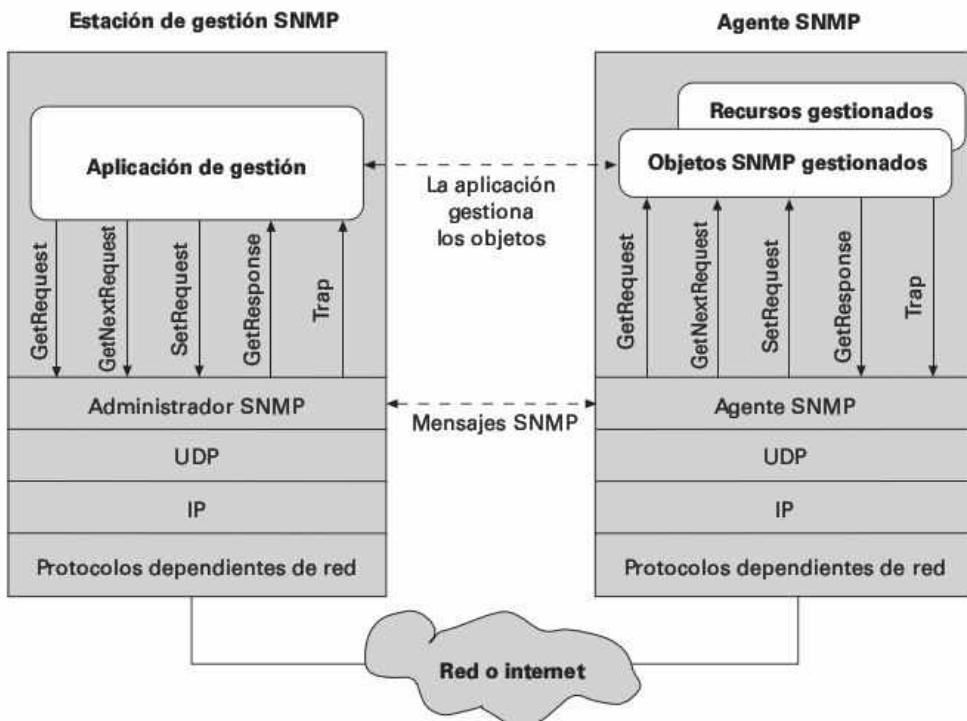


Figura 8.1 El papel de SNMP

AGENTES PROXY

En SNMPv1 todos los agentes y las estaciones de gestión deben disponer de UDP e IP. Esto limita la gestión directa a ciertos dispositivos y excluye otros, como algunos puentes y módems, que no admiten ninguna parte de la *suite* de protocolos TCP/IP. Además, puede haber numerosos sistemas pequeños (computadores personales, estaciones de trabajo, controladores programables) que implementen TCP/IP para dar cabida a sus aplicaciones, pero para los cuales no es deseable añadir la carga adicional que suponen SNMP, la lógica de agente y el mantenimiento de la MIB.

Para acomodar los dispositivos que no implementan SNMP, se desarrolló el concepto de *proxy*. En este esquema, un agente SNMP actúa como *proxy* para uno o más dispositivos; es decir, el agente SNMP actúa en nombre de esos dispositivos.

La Figura 8.2 indica el tipo de arquitectura de protocolo más habitual. La estación de gestión envía a su agente *proxy* peticiones relativas a un dispositivo. El agente *proxy* convierte cada petición al protocolo de gestión que usa el dispositivo. Cuando el agente recibe una respuesta a una petición, la pasa, a su vez, a la estación de gestión. De igual forma, si se transmite desde el dispositivo la notificación de un evento al *proxy*, éste la envía a la estación de gestión en forma de mensaje *trap*.

SNMPv2 no sólo permite el uso de la *suite* de protocolos TCP/IP, sino también de otros. En particular, SNMPv2 está diseñado para funcionar en la *suite* de protocolos OSI. De esta forma, SNMPv2 se puede usar para gestionar una mayor variedad de configuraciones de red. Con respecto a los agentes *proxy*, cualquier dispositivo que no implemente SNMPv2 sólo puede ser gestionado mediante un *proxy*. Esto además incluye dispositivos SNMPv1. Es decir, si un dispositivo implementa el *software* agente para SNMPv1, se puede alcanzar desde un administrador SNMPv2 sólo por un dispositivo *proxy* que implemente el agente SNMPv2 y el *software* de administrador de SNMPv1.

Los casos descritos en el párrafo anterior se conocen en SNMPv2 como relaciones *proxy* extranjeras. Además, SNMPv2 permite una relación *proxy* nativa en la que el dispositivo representado admite SNMPv2. En este caso, un administrador SNMPv2 se comunica con un nodo SNMPv2 que actúa como agente. Este nodo luego actúa como administrador para alcanzar el dispositivo representado, que actúa como agente SNMPv2. La razón de este tipo de comunicación indirecta es la de permitir a los usuarios configurar sistemas de gestión de red jerárquicos y descentralizados, como se explicará más tarde.

SNMPv2

La potencia de SNMP se halla en su sencillez. SNMP proporciona un conjunto básico de herramientas de gestión de red en un paquete fácil de implementar y de configurar. Sin embargo, a medida que los usuarios se han ido basando cada vez más en SNMP para gestionar redes en continua expansión con cargas de trabajo cada vez mayores, sus deficiencias se han hecho demasiado evidentes. Estas deficiencias se dividen en tres categorías:

- Falta de soporte para la gestión de red distribuida
- Deficiencias funcionales
- Deficiencias de seguridad

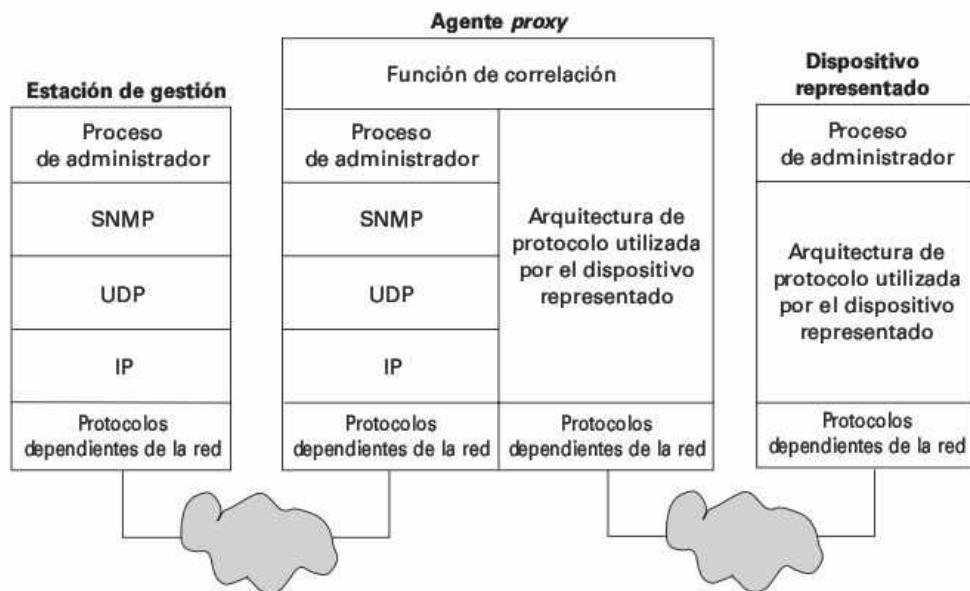


Figura 8.2 Configuración de agentes proxy

Las dos primeras categorías de deficiencias se tratan en SNMPv2, que fue emitido en 1993, con una versión revisada publicada en 1996 (actualmente los RFC 1901, de 1904 a 1908, 2578 y 2579). Muy pronto SNMPv2 obtuvo apoyo y una serie de vendedores anunciaron sus productos en los meses posteriores a la publicación del estándar. Las deficiencias de seguridad han sido tratadas en SNMPv3.

En el resto de este subapartado, se resumen brevemente las nuevas características que se proporcionan con SNMPv2. Las características de seguridad de SNMPv1 y SNMPv3 se examinan detalladamente en las secciones 8.2 y 8.3 respectivamente.

Gestión de red distribuida

En un esquema tradicional de gestión de red centralizada, un computador de la configuración desempeña la función de una estación de gestión de red; puede haber una o dos estaciones de gestión más con la función de apoyo. Los demás dispositivos de la red contienen *software agente* y una MIB, para permitir la supervisión y el control desde la estación de gestión. Como las redes aumentan su tamaño y la carga de tráfico, un sistema centralizado es inviable. Hay demasiada carga en la estación de gestión y demasiado tráfico, con informes de cada uno de los agentes que tienen que guiarlos por toda la red hacia su destino. En tales circunstancias, funciona mejor un enfoque distribuido descentralizado (Figura 8.3). En un esquema descentralizado de gestión de red, puede haber múltiples estaciones de gestión de alto nivel, que se podrían denominar servidores de gestión. Cada uno de estos servidores podría gestionar directamente una parte del total de agentes. Sin embargo, para muchos de los agentes, el servidor de gestión delega responsabilidad a un administrador intermedio. El administrador intermedio desempeña la función de administrador para supervisar y controlar los agentes bajo su responsabili-

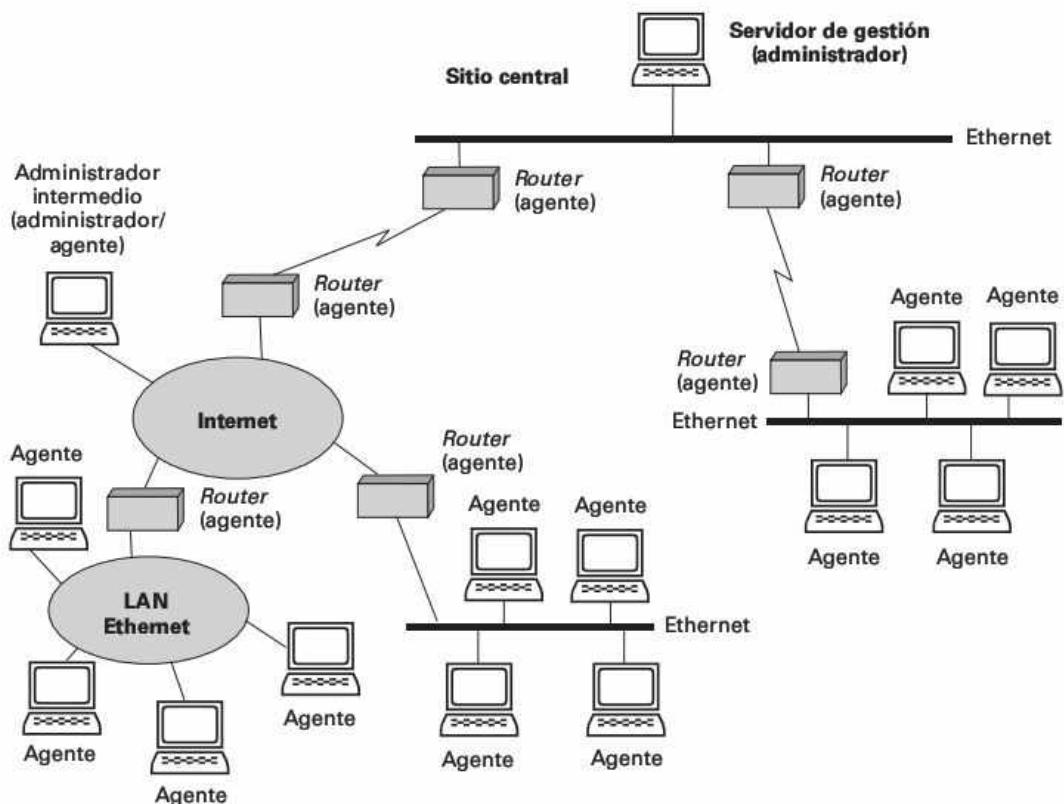


Figura 8.3 Ejemplo de configuración de gestión de red distribuida

dad. También funciona como agente para proporcionar información y aceptar el control por parte de un servidor de gestión de nivel superior. Este tipo de arquitectura dispersa la carga de procesamiento y reduce el tráfico en la red.

SNMPv2 permite una estrategia de gestión de red muy centralizada así como una distribuida. En el último caso, algunos sistemas funcionan tanto con el papel de administrador como de agente. En su papel de agente, tal sistema aceptará órdenes de un sistema de gestión superior. Algunas de esas órdenes se relacionan con la MIB local en el agente. Otras órdenes requieren que el agente actúe como *proxy* para dispositivos remotos. En este caso, el agente *proxy* asume el papel de administrador para acceder a información en un agente remoto y luego asume el papel de un agente para pasar esa información a un administrador superior.

Mejoras funcionales

La Tabla 8.1 presenta las mejoras funcionales que se han realizado en SNMPv2. Los dos protocolos se definen en términos de un conjunto de comandos que se transmiten como unidades de datos del protocolo (PDU, *Protocol Data Units*). En el caso de SNMPv1,

hay cinco comandos. Un administrador emite el comando Get a un agente para obtener valores de objetos en la MIB. El comando GetNext se beneficia del hecho de que los objetos de una MIB se organizan en forma de árbol. Cuando se nombra un objeto en un comando GetNext, el agente encuentra el siguiente objeto en el árbol y devuelve su valor. GetNext es útil porque permite que un administrador «recorra» un árbol en un agente cuando no conoce el conjunto exacto de objetos admitidos por ese agente. El comando Set permite que un administrador actualice los valores en un agente; también se usa para crear y borrar filas en tablas. Un agente usa el comando GetResponse para responder a un comando del administrador. Por último, el comando Trap permite que un agente envíe información a un administrador sin esperar solicitud de gestión. Por ejemplo, un agente podría estar configurado para enviar un Trap cuando falle una conexión o cuando el tráfico exceda los límites.

SNMPv2 incluye todos los comandos que se encuentran en SNMPv1 más dos nuevos. El más importante de ellos es el comando Inform. Este comando se envía de una estación de gestión a otra y, al igual que el Trap, incluye información relativa a las condiciones y los eventos en el emisor. La ventaja de dicho comando es que se puede usar para construir una configuración en la que múltiples administradores cooperan para compartir las responsabilidades de la gestión de una red grande.

Tabla 8.1 Comparación de las PDU de SNMPv1 y SNMPv2

SNMPv1 PDU	SNMPv2 PDU	Dirección	Descripción
GetRequest	GetRequest	De administrador a agente	Solicita valor para cada objeto enumerado
GetNextRequest	GetNextRequest	De administrador a agente	Solicita siguiente valor para cada objeto enumerado
—	GetBulkRequest	De administrador a agente	Solicita valores múltiples
SetRequest	SetRequest	De administrador a agente	Establece valor para cada objeto enumerado
—	InformRequest	De administrador a administrador	Transmite información no solicitada
GetResponse	Response	De agente a administrador o de administrador a administrador (SNMPv2)	Responde a la solicitud del administrador
Trap	SNMPv2-Trap	De agente a administrador	Transmite información no solicitada

El otro nuevo comando, GetBulk, permite que un administrador obtenga de una vez un gran bloque de datos. Concretamente, este comando está diseñado para la transmisión de tablas enteras con un comando.

Una última diferencia se halla en el hecho de que el comando Get es atómico en el caso de SNMPv1 pero no en el caso de SNMPv2. Si un comando Get de SNMPv1 contiene una lista de objetos para los cuales se solicitan valores, y al menos uno de los objetos no existe en el agente, entonces se rechaza el comando entero. Para SNMPv2, se pueden devolver resultados parciales. El comando Get no atómico permite un uso más eficaz de la capacidad de la red por parte del administrador.

8.2 COMUNIDADES SNMPv1

SNMPv1, tal y como se define en el RFC 1157, proporciona sólo una herramienta rudimentaria de seguridad basada en el concepto de comunidad. Esta herramienta aporta un cierto nivel de seguridad pero está abierta a distintos ataques [CERT02, JIAN02].

COMUNIDADES Y NOMBRES DE COMUNIDADES

Al igual que otras aplicaciones distribuidas, la gestión de red implica la interacción de una serie de entidades de aplicación admitidas por un protocolo de aplicación. En el caso de la gestión de red SNMP, las entidades de aplicación son las aplicaciones de administrador y las aplicaciones agente que usan SNMP.

La gestión de red SNMP tiene varias características que no son comunes a todas las aplicaciones distribuidas. La aplicación implica una relación de uno a muchos entre un administrador y un grupo de agentes: el administrador puede obtener y establecer objetos en los agentes y puede recibir *traps* de los agentes. Así, desde un punto de vista operativo o de control, el administrador gestiona un número de agentes. Puede haber una serie de administradores, cada uno de los cuales gestiona todos los agentes, o un subgrupo, de la configuración. Estos subgrupos se pueden solapar.

También necesitamos poder ver la gestión de red SNMP como una relación de uno a muchos entre un agente y un conjunto de administradores. Cada agente controla su propia MIB local y debe ser capaz de controlar el uso de esa MIB por parte de una serie de administradores. Existen tres aspectos de este control:

- **Servicio de autentificación:** el agente puede querer limitar el acceso a la MIB a los administradores autorizados.
- **Política de acceso:** el agente puede querer conceder diferentes privilegios de acceso a diferentes administradores.
- **Servicio proxy:** un agente puede actuar como *proxy* para otros agentes. Esto puede implicar la implementación del servicio de autentificación y/o política de acceso para los otros agentes en el sistema *proxy*.

Todos estos aspectos se relacionan con la seguridad. En un entorno en el que la responsabilidad de los componentes de la red está dividida, por ejemplo, entre un número de entidades administrativas, los agentes necesitan protegerse a sí mismos y a sus MIB frente al acceso no deseado o no autorizado. SNMP, como se define en el RFC 1157, proporciona sólo una capacidad primitiva y limitada para la seguridad, concretamente el concepto de comunidad.

Una **comunidad SNMP** consiste en una relación entre un agente SNMP y un conjunto de administradores SNMP que define la autenticación, el control de acceso y las características del *proxy*. El concepto de comunidad es local, definido en el agente. El agente establece una comunidad para cada combinación deseada de autenticación, control de acceso y características del *proxy*. A cada comunidad se le asigna un nombre de comunidad único (en ese agente), y a los administradores en esa comunidad se les proporciona y deben emplear el nombre de comunidad en todas las operaciones Get y Set. El agente puede establecer un número de comunidades, donde los administradores miembros se pueden solapar.

Como las comunidades se definen localmente en el agente, diferentes agentes pueden utilizar el mismo nombre. Esta identidad de nombres es irrelevante y no indica ninguna similitud entre las comunidades definidas. Así, un administrador debe mantenerse al tanto del nombre o nombres de comunidad que estén asociados con cada uno de los agentes a los que desea acceder.

SERVICIO DE AUTENTIFICACIÓN

La finalidad del servicio de autenticación SNMPv1 es garantizar al receptor que un mensaje SNMPv1 proviene de la fuente de la que dice proceder. SNMPv1 sólo proporciona un esquema trivial para la autenticación. Cada mensaje (*get* o *put request*) de un administrador a un agente incluye un nombre de comunidad. Este nombre funciona como una contraseña, y se asume que el mensaje es auténtico si el emisor conoce la contraseña.

Con esta forma limitada de autenticación, muchos administradores de red son reacios a permitir otra cosa que no sea la supervisión de la red; es decir, operaciones *get* y *trap*. El control de la red mediante una operación *set* es claramente un área más sensible. El nombre de comunidad se podría usar para activar un procedimiento de autenticación, con el nombre funcionando simplemente como un dispositivo inicial de ocultación de contraseña. El procedimiento de autenticación podría implicar el uso de cifrado/descifrado para funciones de autenticación más seguras. Sin embargo, esto escapa del ámbito del RFC 1157.

POLÍTICA DE ACCESO

Al definir una comunidad, un agente limita el acceso a su MIB a un conjunto seleccionado de administradores. Con el uso de más de una comunidad, el agente puede proporcionar diferentes categorías de acceso a la MIB a diferentes administradores. Hay dos aspectos relativos a este control de acceso:

- **Vista MIB de SNMP:** subconjunto de los objetos de una MIB. Se pueden definir diferentes vistas MIB para cada comunidad. El conjunto de objetos en una vista no necesita pertenecer a un solo subárbol de la MIB.
- **Modo de acceso SNMP:** elemento del conjunto {*READ-ONLY*, *READ-WRITE*} (sólo lectura, lectura-escritura). Se define un modo de acceso para cada comunidad.

La combinación de una vista MIB y un modo de acceso se conoce como **perfil de comunidad SNMP**. Así, un perfil de comunidad se compone de un subconjunto definido

de la MIB en el agente, más un modo de acceso para esos objetos. El modo de acceso SNMP se aplica de manera uniforme a todos los objetos en la vista MIB. Por lo tanto, si se selecciona el modo de acceso *READ-ONLY*, se aplica a todos los objetos en la vista y limita el acceso de los administradores de esta vista a operaciones de sólo lectura.

Un perfil de comunidad se asocia con cada comunidad definida por un agente; la combinación de una comunidad SNMP y un perfil de comunidad SNMP se conoce como una **política de acceso SNMP**. La Figura 8.4 ilustra los distintos conceptos que se acaban de introducir.

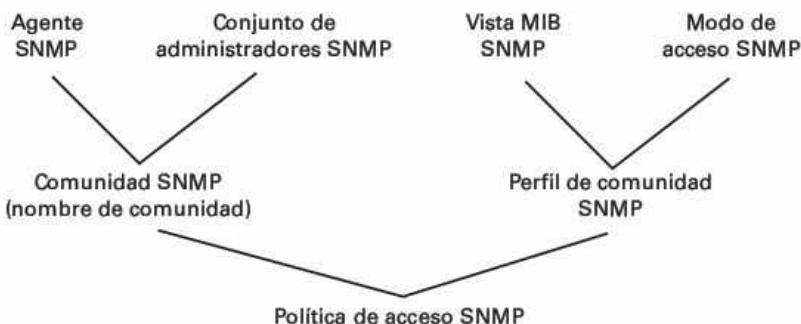


Figura 8.4 Conceptos administrativos de SNMPv1

SERVICIO PROXY

El concepto de comunidad también es útil para implementar el servicio *proxy*. Recorremos que un *proxy* es un agente SNMP que actúa en nombre de otros dispositivos. Normalmente, los otros dispositivos son extranjeros, en el sentido de que no admiten TCP/IP ni, por lo tanto, SNMP. En algunos casos, el sistema que utiliza *proxy* puede admitir SNMP, pero el *proxy* se utiliza para reducir la interacción entre los dispositivos representados por él y los sistemas de gestión de red.

Para cada dispositivo que el sistema *proxy* representa, mantiene una política de acceso SNMP. Así, el *proxy* sabe qué objetos MIB se pueden usar para gestionar el sistema representado por el *proxy* (la vista MIB) y su modo de acceso.

8.3 SNMPv3

En 1998, el grupo de trabajo SNMPv3 de la IETF produjo una serie de estándares propuestos de Internet, actualmente los RFC desde 2570 hasta 2576. Este conjunto de documentos define un marco de trabajo para la incorporación de características de seguridad en una capacidad general que incluye funcionalidad SNMPv1 o SNMPv2. Además, los documentos definen un grupo específico de capacidades para la seguridad en la red y el control de acceso.

Es importante observar que SNMPv3 no es una sustitución independiente de SNMPv1 y/o SNMPv2. SNMPv3 define una capacidad de seguridad para ser usada con-

juntamente con SNMPv2 (preferentemente) o SNMPv1. Además, el RFC 2571 describe una arquitectura en la cual encajan todas las versiones actuales y futuras de SNMP. El RFC 2575 describe una herramienta de control de acceso, diseñada para operar independientemente de la capacidad central de SNMPv3. En esta sección, se ofrece una visión general y un estudio de las capacidades definidas en los RFC del 2570 al 2576.

La Figura 8.5 indica la relación entre las diferentes versiones de SNMP por medio de los formatos involucrados. La información se intercambia entre una estación de gestión y un agente en forma de mensaje SNMP. El procesamiento relativo a la seguridad ocurre en el nivel de mensaje; por ejemplo, SNMPv3 especifica un Modelo de Seguridad de Usuario (USM, *User Security Model*) que utiliza campos en la cabecera del mensaje. La carga útil de un mensaje SNMP es una unidad de datos del protocolo (PDU, *Protocol Data Unit*) SNMPv1 o SNMPv2. Una PDU indica un tipo de acción de gestión [por ejemplo, obtener (*get*) o establecer (*set*) un objeto gestionado] y una lista de nombres de variables relacionados con esa acción.

Los RFC 2570 hasta 2576 describen una arquitectura general y estructuras de mensajes y características de seguridad específicas, pero no definen un nuevo formato de PDU de SNMP. Por lo tanto, en la nueva arquitectura se debe usar el formato existente de PDU de SNMPv1 o de SNMPv2. Una implementación conocida como SNMPv3 consiste en las características de seguridad y de arquitectura definidas en los RFC 2570 hasta 2576 más el formato de PDU y la funcionalidad definida en los documentos SNMPv2. En RFC 2570 se expresa como sigue: «SNMPv3 se puede ver como SNMPv2 con capacidades adicionales de seguridad y gestión.»

Lo que resta de esta sección se organiza de la siguiente manera. En primer lugar, se ofrece una breve introducción a la arquitectura básica de SNMP definida en RFC 2571. Luego, se describen las herramientas de confidencialidad y autenticación proporcionadas por el USM de SNMPv3. Por último, se trata el control de acceso y el modelo de control de acceso basado en vistas (VACM, *View-based Access Control Model*).

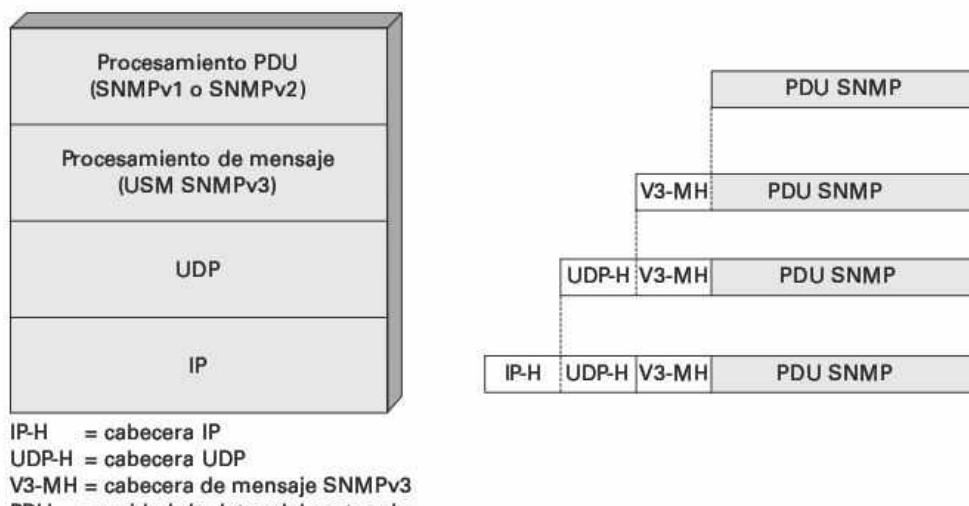


Figura 8.5 Arquitectura del protocolo SNMP

ARQUITECTURA SNMP

La arquitectura SNMP, como la presenta el RFC 2571, consiste en un conjunto distribuido de entidades SNMP que actúan entre sí. Cada entidad implementa una parte de la capacidad SNMP y puede actuar como un nodo agente, un nodo administrador o una combinación de ellos. Cada entidad SNMP se compone de un conjunto de módulos que interactúan entre sí para suministrar servicios. Estas interacciones se pueden modelar como un conjunto de primitivas y parámetros abstractos.

La arquitectura del RFC 2571 refleja un requisito de diseño fundamental para SNMPv3: diseñar una arquitectura modular que (1) permita la implementación en una gran variedad de entornos, de los cuales, algunos necesiten funcionalidad mínima y de bajo coste, y otros puedan admitir características adicionales para la gestión de redes grandes; (2) hacer posible que partes de la arquitectura avancen en el proceso de estandarización incluso aunque no se haya alcanzado consenso en todas las partes; y (3) dar cabida a modelos alternativos de seguridad.

Entidad SNMP

Cada entidad SNMP incluye un único motor SNMP. Un motor SNMP implementa funciones para enviar y recibir mensajes, autenticar y cifrar/descifrar mensajes y controlar el acceso a los objetos administrados. Estas funciones se proporcionan como servicios a una o más aplicaciones que se configuran con el motor SNMP para formar una entidad SNMP.

Tanto el motor SNMP como las aplicaciones que soporta se definen como una colección de módulos discretos. Esta arquitectura ofrece varias ventajas. Primero, el papel de una entidad SNMP se determina en función de qué módulos se implementan en esa entidad. Se necesita un determinado conjunto de módulos para un agente SNMP, mientras que para un gestor SNMP se necesita un conjunto diferente de módulos (aunque se solapen). Segundo, la estructura modular de la especificación se presta a definir diferentes versiones de cada módulo. Esto, a su vez, hace posible (1) definir capacidades alternativas o mejoradas para ciertos aspectos de SNMP sin necesidad de remitirse a una nueva versión del estándar completo (por ejemplo, SNMPv4), y (2) especificar con claridad las estrategias de coexistencia y transición (RFC 2576).

Para comprender mejor el papel de cada módulo y su relación con otros módulos, es conveniente observar su uso en gestores y agentes SNMP tradicionales. El término *tradicional*, equivalente a *puro*, se usa para resaltar el hecho de que una implementación dada no necesita ser un gestor o agente puro, sino que puede tener módulos que permitan a la entidad realizar tareas de gestión y de agente.

La Figura 8.6, basada en una figura del RFC 2571, es un diagrama de bloque de un **gestor SNMP tradicional**. Un gestor SNMP tradicional interactúa con agentes SNMP emitiendo comandos (Get, Set) y recibiendo mensajes *trap* o de interceptación; el gestor también puede interactuar con otros gestores emitiendo PDU de solicitud de información (*Inform Request*), que proporciona alertas, y recibiendo PDU de respuesta de información (*Inform Response*), que reconocen solicitudes de información. En la terminología de SNMPv3, un gestor SNMP tradicional incluye tres categorías de aplicaciones. Las Aplicaciones de Generación de Comandos (*Command Generator Applications*) supervisan y

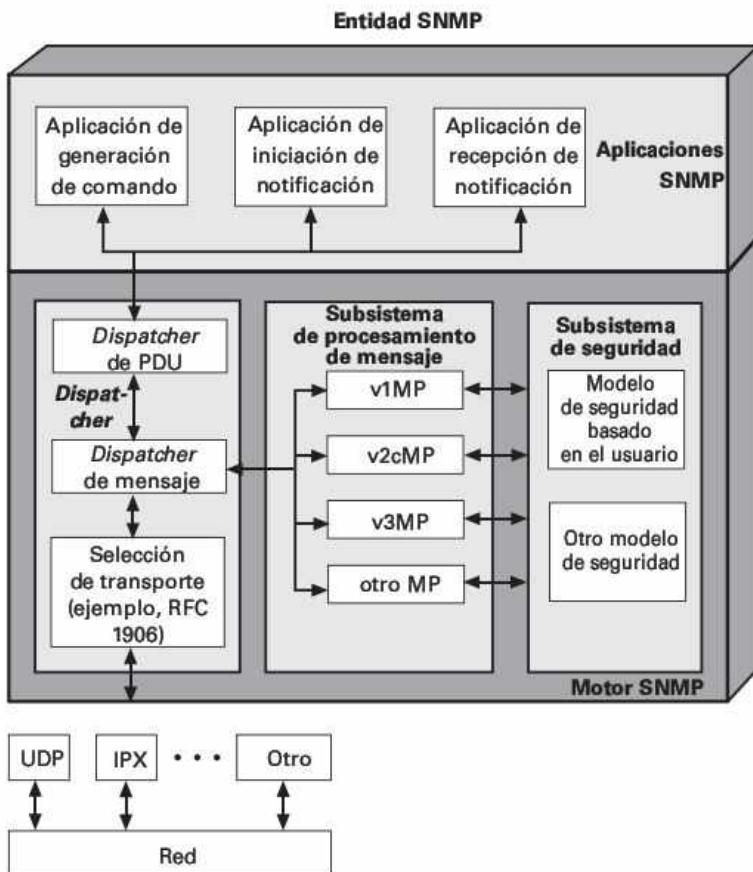


Figura 8.6 Gestor SNMP tradicional

manipulan los datos de gestión en agentes remotos; usan PDU de SNMPv1 y/o SNMPv2, que incluyen Get, GetNext, GetBulk y Set. Una Aplicación de Iniciación de Notificación (*Notification Originator Application*) inicia mensajes asíncronos; en el caso de un gestor tradicional, se usa la PDU de solicitud de información (InformRequest) para esta aplicación. Una Aplicación de Recepción de Notificación (*Notification Receiver Application*) procesa los mensajes asíncronos entrantes; éstos incluyen PDU InformRequest, SNMPv2-Trap y SNMPv1 Trap. En el caso de una PDU entrante de solicitud de información, la Aplicación de Recepción de Notificación responderá con una PDU de respuesta.

Todas las aplicaciones descritas hacen uso de los servicios proporcionados por el motor SNMP para esta entidad. El motor SNMP realiza dos funciones generales:

- Acepta PDU salientes de aplicaciones SNMP; realiza el procesamiento necesario, incluyendo la inserción de códigos de autentificación y el cifrado; y luego encapsula las PDU en mensajes para su transmisión.
- Acepta mensajes SNMP entrantes de la capa de transporte; realiza el procesamiento necesario, incluyendo la autentificación y el cifrado; y luego extrae las PDU de los mensajes y los pasa a la aplicación SNMP correspondiente.

En un gestor tradicional, el motor SNMP contiene un *dispatcher*, un subsistema de procesamiento de mensajes y un subsistema de seguridad. El *dispatcher* es un gestor simple de tráfico. Para las PDU salientes, el *dispatcher* acepta las PDU de las aplicaciones y realiza las siguientes funciones: para cada PDU, determina el tipo de procesamiento de mensaje requerido (por ejemplo, para SNMPv1, SNMPv2 o SNMPv3) y pasa la PDU al módulo correspondiente de procesamiento de mensajes del subsistema de procesamiento. Posteriormente, el subsistema de procesamiento de mensajes devuelve un mensaje que contiene esa PDU y que incluye las cabeceras de mensaje adecuadas. Luego, el *dispatcher* correlaciona este mensaje con una capa de transporte para la transmisión.

Para los mensajes entrantes, el *dispatcher* acepta mensajes de la capa de transporte y lleva a cabo las siguientes funciones: encamina cada mensaje al módulo adecuado de procesamiento de mensajes. A continuación, el subsistema de procesamiento de mensajes devuelve la PDU contenida en el mensaje. Luego el *dispatcher* pasa esta PDU a la aplicación correspondiente.

El subsistema de procesamiento de mensajes acepta las PDU salientes del *dispatcher* y las prepara para la transmisión encapsulándolas en la cabecera de mensaje adecuada y devolviéndolas al *dispatcher*. El subsistema de procesamiento de mensajes también acepta mensajes entrantes del *dispatcher*, procesa cada cabecera de mensaje y devuelve al *dispatcher* la PDU adjunta. Una implementación del subsistema de procesamiento de mensajes puede admitir un solo formato de mensaje correspondiente a una sola versión de SNMP (SNMPv1, SNMPv2c, SNMPv3), o puede contener una serie de módulos, cada uno de los cuales admite una versión diferente de SNMP.

El subsistema de seguridad desempeña funciones de autentificación y cifrado. Cada mensaje saliente se pasa al subsistema de seguridad desde el subsistema de procesamiento de mensajes. Dependiendo de los servicios requeridos, el subsistema de seguridad puede cifrar la PDU adjunta y posiblemente algunos campos de la cabecera del mensaje, y puede generar un código de autentificación e insertarlo en la cabecera del mensaje. El mensaje procesado luego se devuelve al subsistema de procesamiento de mensajes. De la misma forma, cada mensaje entrante se pasa al subsistema de seguridad desde el subsistema de procesamiento de mensajes. Si es necesario, el subsistema de seguridad comprueba el código de autentificación y lleva a cabo el descifrado. A continuación devuelve el mensaje procesado al subsistema de procesamiento de mensajes. Una implementación del subsistema de seguridad puede permitir uno o más modelos de seguridad diferenciados. Hasta ahora, el único modelo de seguridad definido es el modelo de seguridad basado en el usuario (*USM, User-Based Security Model*) para SNMPv3, especificado en el RFC 2574.

La Figura 8.7, basada en una figura del RFC 2571, es un diagrama de bloques de un **agente SNMP tradicional**. El agente tradicional puede contener tres tipos de aplicaciones. Las Aplicaciones de Respuesta a Comandos (*Command Responder Applications*) proporcionan acceso a datos de gestión. Estas aplicaciones responden a las solicitudes entrantes obteniendo y/o estableciendo objetos gestionados y emitiendo después una PDU de respuesta. Una aplicación de iniciación de notificación inicia mensajes asíncronos; en el caso de un agente tradicional se usa la PDU Trap de SNMPv1 o SNMPv2-Trap para esta aplicación. Una aplicación de reenvío mediante proxy (*Proxy Forwarder Applications*) reenvía mensajes entre entidades.

El motor SNMP para un agente tradicional tiene todos los componentes encontrados en el motor SNMP para un gestor tradicional, más un subsistema de control de acceso.

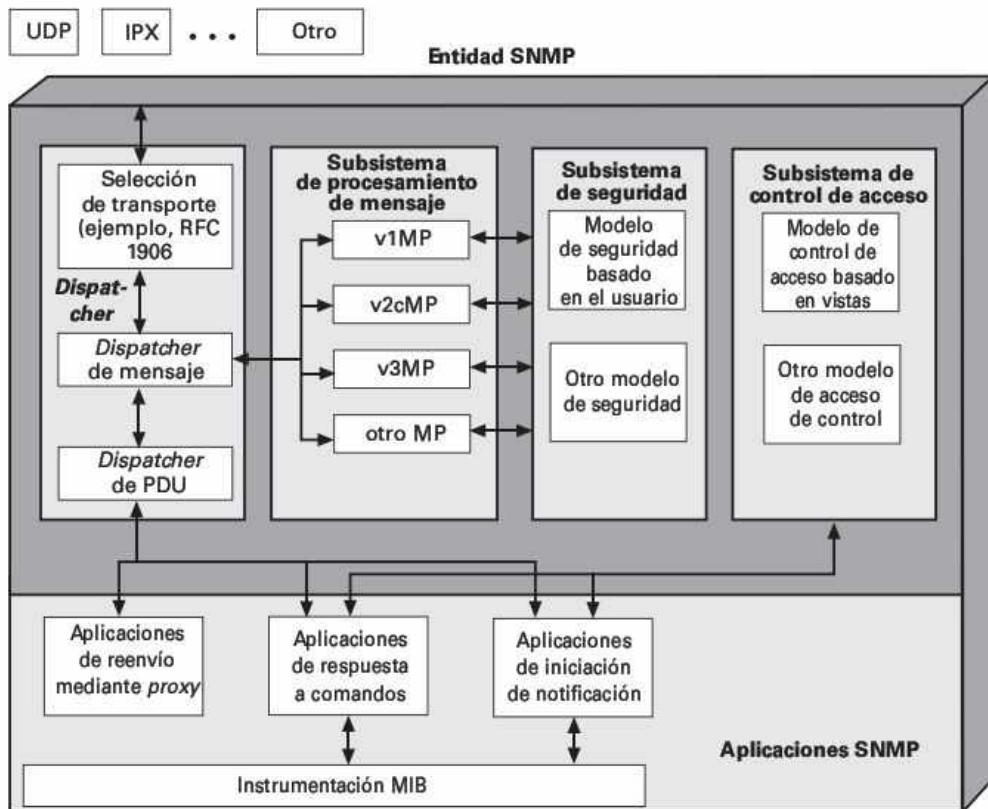


Figura 8.7 Agente SNMP tradicional

Este subsistema proporciona servicios de autorización para controlar el acceso a las MIB para los objetos de gestión de lectura (*reading*) y de selección (*setting*). Estos servicios se realizan en función de los contenidos de las PDU. Una implementación del subsistema de seguridad puede admitir uno o más modelos de control de acceso diferenciados. Hasta el momento, el único modelo de seguridad definido es el modelo de control de acceso basado en vistas (VACM), especificado en el RFC 2575.

Obsérvese que las funciones relacionadas con la seguridad se organizan en dos subsistemas separados: seguridad y control de acceso. Este es un ejemplo excelente de un buen diseño modular, porque los dos subsistemas realizan funciones muy distintas y, por lo tanto, tiene sentido permitir la estandarización de estas dos áreas para proceder de manera independiente. El subsistema de seguridad tiene que ver con la privacidad y la autenticación y opera en mensajes SNMP. El subsistema de control de acceso se ocupa del acceso autorizado a información de gestión y opera en las PDU SNMP.

Terminología

La Tabla 8.2 define brevemente algunos de los términos introducidos en el RFC 2571. Asociado con cada entidad SNMP hay un *snmpEngineID* único. Para los propósitos del

control de acceso, se considera que cada entidad SNMP gestiona una serie de contextos de información gestionada, cada uno de los cuales tiene un nombre de contexto (contextName) que es único en esa entidad. Para resaltar que hay un solo gestor de contextos en una entidad, cada una de ellas tiene asociado un contextEngineID único; debido a que hay una correspondencia de uno a uno entre el motor de contextos y el de SNMP en esta entidad, el contextEngineID tiene un valor idéntico al snmpEngineID. El control de acceso está regido por el contexto específico para el cual se intenta el acceso y la identidad del usuario que solicita acceso; esta última identidad se expresa como entidad principal, que puede ser un individuo, una aplicación o un grupo de individuos o de aplicaciones.

Tabla 8.2 Terminología de SNMPv3

snmpEngineID	Identificador único y carente de ambigüedad de un motor SNMP, así como la entidad SNMP que corresponde a ese motor. Se define por convención textual, con una sintaxis de ristra de octetos.
contextEngineID	Identifica únicamente una entidad SNMP que puede realizar una instancia de un contexto con un contextName particular.
contextName	Identifica un contexto particular en un motor SNMP. Se pasa como un parámetro de <i>dispatcher</i> y al subsistema de control de acceso.
scopedPDU	Un bloque de datos formado por un contextEngineID, un contextName y una PDU SNMP. Se pasa como un parámetro al/desde el sistema de seguridad.
snmpMessageProcessingModel	Identificador único de un modelo de procesamiento de mensajes del subsistema de procesamiento de mensajes. Algunos valores posibles incluyen SNMPv1, SNMPv2c y SNMPv3. Se define por convención textual, con una sintaxis de entero.
snmpSecurityModel	Identificador único de un modelo de seguridad del subsistema de seguridad. Algunos valores incluyen SNMPv1, SNMPv2c y USM. Se define por convención textual, con una sintaxis de entero.
snmpSecurityLevel	Un nivel de seguridad en el que se pueden enviar mensajes SNMP o con el cual se están procesando operaciones, expresado en función de si se proporciona autenticación y/o privacidad o no. Los valores alternativos son noAuthnoPriv, authNoPriv y authPriv. Se define por convención textual, con una sintaxis de entero.
principal	La entidad en cuyo nombre se proporcionan servicios o tiene lugar el procesamiento. Puede ser un individuo actuando con un papel particular; un conjunto de individuos, que actúan con una función particular, una aplicación o conjunto de aplicaciones, y la combinación de ellos.
securityName	Una ristra legible por humanos representando al principal. Se pasa como un parámetro en todas las primitivas SNMP (<i>dispatcher</i> , procesamiento de mensajes, seguridad, control de acceso).

Otros términos importantes están relacionados con el procesamiento de mensajes. El snmpMessageProcessingModel determina el formato de mensaje y la versión SNMP para el procesamiento de mensajes. El snmpSecurityModel determina qué modelo de seguridad se va a usar. El snmpSecurityLevel determina qué servicios de seguridad se solicitan para esta operación específica. El usuario puede solicitar sólo autentificación, o autentificación más privacidad (cifrado) o ninguno de ellos.

Aplicaciones SNMPv3

Los servicios entre módulos en una entidad SNMP se definen en los RFC en términos de primitivas y parámetros. Una primitiva especifica la función que se ha de realizar, y los parámetros se usan para pasar datos e información de control. Podemos considerar estas primitivas y parámetros como una manera formalizada de definir servicios SNMP. La forma real de una primitiva es dependiente de la implementación; un ejemplo de ello es una llamada a procedimiento. En lo que se expone a continuación, puede ser útil remitirse a la Figura 8.8, basada en una figura del RFC 2571, para ver cómo encajan todas estas primitivas. La Figura 8.8a muestra la secuencia de hechos en que una aplicación de generación de comandos o de iniciación de notificación solicita que se envíe una PDU y, posteriormente, cómo se devuelve a esa aplicación la respuesta correspondiente; estas acciones se producen en un gestor. La Figura 8.8b muestra los hechos correspondientes en un agente. La figura muestra cómo un mensaje entrante da como resultado el envío de la PDU adjunta a una aplicación, y cómo la respuesta de esa aplicación da como resultado un mensaje saliente. Obsérvese que algunas de las flechas del diagrama están etiquetadas con un nombre de primitiva, que representa una llamada. Las flechas sin etiqueta representan la respuesta de una llamada, y las partes sombreadas indican la correspondencia entre la llamada y la respuesta.

El RFC 2573 define, en términos generales, los procedimientos seguidos por cada tipo de aplicación al generar las PDU para la transmisión o al procesar las PDU entrantes. En todos los casos, los procedimientos se definen en términos de la interacción con el *dispatcher* por medio de primitivas del *dispatcher*.

Una **aplicación de generación de comandos** emplea las primitivas del *dispatcher* sendPdu y processResponsePdu. La sendPdu proporciona al *dispatcher* información sobre el destino elegido, parámetros de seguridad y la PDU que se ha de enviar. El *dispatcher* invoca, luego, al modelo de procesamiento de mensajes, que a su vez llama al modelo de seguridad, para preparar el mensaje. El *dispatcher* pasa el mensaje preparado a la capa de transporte (por ejemplo, UDP) para la transmisión. Si la preparación del mensaje falla, el valor devuelto de sendPdu, fijado por el *dispatcher*, es una indicación de error. Si la preparación del mensaje es satisfactoria, el *dispatcher* asigna un identificador sendPduHandle a esta PDU y devuelve ese valor al generador de comandos. El generador de comandos almacena el sendPduHandle para poder relacionar la PDU de respuesta posterior con la solicitud original.

El *dispatcher* entrega cada PDU de respuesta entrante a la aplicación correcta de generación de comandos, usando la primitiva processResponsePdu.

Una **aplicación de respuesta a comandos** utiliza cuatro primitivas de *dispatcher* (registerContextEngineID, unregisterContextEngineID, processPdu, returnResponsePdu) y una primitiva del subsistema de control de acceso (isAccessAllowed).

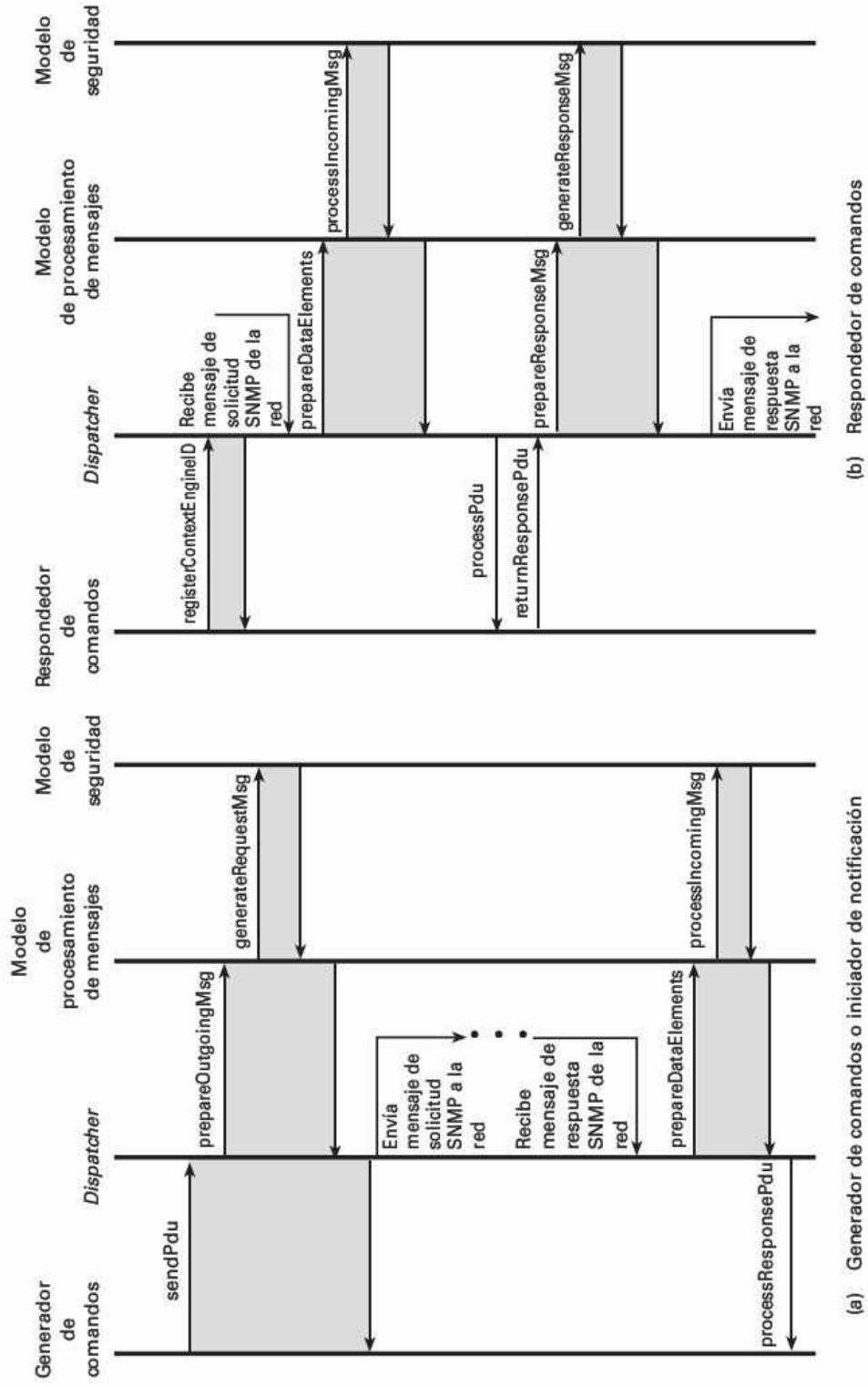


Figura 8.8 Flujo SNMPv3

La primitiva registerContextEngineID permite que una aplicación de respuesta a comando se asocie con un motor SNMP con el fin de procesar ciertos tipos de PDU para un motor de contexto. Una vez que el respondedor de comandos se ha registrado, todos los mensajes recibidos de manera asíncrona que contienen la combinación registrada de contextEngineID y pduType admitida son enviados al respondedor de comandos que se registró para admitir esa combinación. Un respondedor de comandos puede desasociarse de un motor SNMP usando la primitiva unregisterContextEngineID.

El *dispatcher* entrega cada PDU de solicitud entrante a la aplicación correcta de respuesta a comando, usando la primitiva processPdu. Luego, el respondedor de comandos realiza los siguientes pasos:

- Examina los contenidos de la PDU de solicitud. El tipo de operación debe corresponderse con uno de los tipos registrados previamente por esta aplicación.
- Determina si se permite el acceso para realizar la operación de gestión solicitada en esta PDU. Con este fin, se llama a la primitiva isAccessAllowed.
- El parámetro securityModel indica qué modelo de seguridad va a usar el subsistema de control de acceso en respuesta a esta llamada. El subsistema de control de acceso determina si la entidad principal (securityName) que hace la petición en este nivel de seguridad (securityLevel) tiene permiso para solicitar la operación de gestión (viewType) en el objeto de gestión (variableName) en este contexto (contextName).
- Si se permite el acceso, el respondedor de comandos realiza la operación de gestión y prepara una PDU de respuesta. Si el acceso falla, prepara la PDU de respuesta adecuada para indicar dicho error.
- El respondedor de comandos llama al *dispatcher* con una primitiva returnResponsePdu para enviar la PDU de respuesta.

Una **aplicación de generación de notificación** sigue los mismos procedimientos generales usados para una aplicación de generación de comandos. Si se va a enviar una PDU de solicitud de información, se usan las primitivas sendPdu y processResponsePdu, de la misma forma que para las aplicaciones de generación de comandos. Si se va a enviar una PDU de interceptación (Trap), sólo se usa la primitiva sendPdu.

Una **aplicación de recepción de notificación** sigue un subgrupo de los procedimientos generales que se usan para la aplicación de respuesta a comandos. El receptor de notificación debe primero registrarse para recibir las PDU de información y/o de interceptación. Los dos tipos de PDU se reciben por medio de una primitiva processPdu. Para una PDU de información, se usa una primitiva returnResponsePdu para responder.

Una **aplicación de reenvío mediante proxy** usa primitivas de *dispatcher* para reenviar mensajes SNMP. Dicha aplicación trata cuatro tipos básicos de mensajes:

- Mensajes que contienen tipos de PDU de una aplicación de generación de comandos. La aplicación de reenvío mediante proxy determina el motor SNMP objetivo o un motor SNMP que esté más próximo al objetivo, o por debajo de él, y envía la PDU de respuesta adecuada.
- Mensajes que contienen tipos de PDU de una aplicación de generación de notificación. La aplicación de reenvío mediante proxy determina qué motores SNMP deberían recibir la notificación y envía la PDU o las PDU de notificación adecuadas.

- Mensajes que contienen un tipo de PDU de respuesta. La aplicación de reenvío mediante *proxy* determina qué solicitud o notificación reenviada anteriormente, de haberla, se corresponde con esta respuesta, y envía la PDU de respuesta adecuada.
- Mensajes que contienen una indicación de informe. Las PDU de informe son comunicaciones entre motores SNMPv3. La aplicación de reenvío mediante *proxy* determina qué solicitud o notificación reenviada previamente, de haberla, se corresponde con esta indicación de informe y reenvía de vuelta la indicación de informe al iniciador de la solicitud o notificación.

PROCESAMIENTO DE MENSAJES Y MODELO DE SEGURIDAD DE USUARIO

El procesamiento de mensajes implica un modelo de procesamiento de mensajes de propósito general y un modelo de seguridad específico; esta relación se muestra en la Figura 8.8.

Modelo de procesamiento de mensajes

El RFC 2572 define un modelo de procesamiento de mensajes de propósito general. Este modelo es responsable de aceptar las PDU del *dispatcher*, encapsularlas en mensajes e invocar al USM para insertar parámetros relacionados con la seguridad en la cabecera del mensaje. El modelo de procesamiento de mensajes también acepta mensajes entrantes, llama al USM para procesar los parámetros relativos a la seguridad en la cabecera del mensaje y entrega la PDU encapsulada al *dispatcher*.

La Figura 8.9 ilustra la estructura del mensaje. Los primeros cinco campos son generados por el modelo de procesamiento de mensajes en los mensajes salientes y procesados por el mismo en los mensajes entrantes. Los siguientes seis campos muestran los parámetros de seguridad usados por USM. Por último, la PDU, junto con el contextEngineID y contextName, constituye una *PDU extendida (scoped)*, que se usa para el procesamiento de la PDU.

Los primeros cinco campos son los siguientes:

- **msgVersion:** fijado a snmpv3(3).
- **msgID:** identificador exclusivo usado entre entidades SNMP para coordinar mensajes de solicitud y de respuesta, y usado por el procesador de mensajes para coordinar el procesamiento del mensaje por medio de diferentes modelos de subsistema en la arquitectura. El rango de este ID es de 0 a $2^{31} - 1$.
- **msgMaxSize:** expresa en octetos el tamaño máximo de un mensaje permitido por el emisor del mensaje, con un rango de 484 hasta $2^{31} - 1$. Este es el tamaño máximo de segmento que el emisor puede aceptar de otro motor SNMP (ya sea una respuesta u otro tipo de mensaje).
- **msgFlags:** ristra de octetos que contiene tres indicadores en los tres bits menos significativos: reportableFlag, privFlag, authFlag. Si reportableFlag = 1, entonces se debe devolver una PDU de informe al emisor bajo esas condiciones que pueden ocasionar la generación de una PDU de informe; cuando el indicador es cero, puede no enviarse una PDU de informe (*Report PDU*). El reportableFlag se fija a

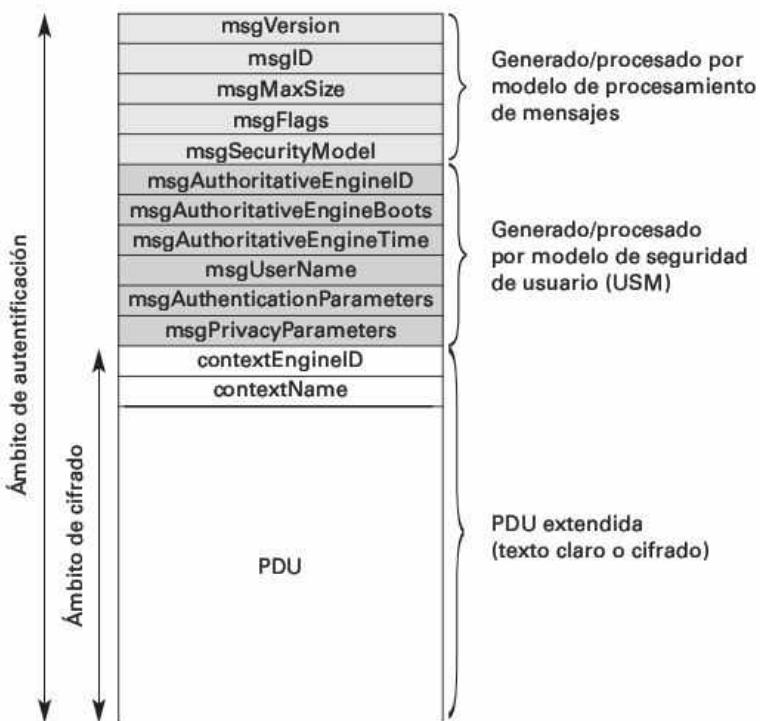


Figura 8.9 Formato de mensaje SNMP con USM

uno por el emisor en todos los mensajes que contienen una solicitud (Get, Set) o un Inform, y se fija a cero para los mensajes que contienen una respuesta, un Trap o una PDU de informe. El reportableFlag es una ayuda secundaria para determinar cuándo enviar un informe. Éste sólo se usa en casos en los que la parte de PDU del mensaje no puede ser decodificada (por ejemplo, cuando falla el descifrado debido a una clave incorrecta). El privFlag y el authFlag son establecidos por el emisor para indicar el nivel de seguridad que se aplicó al mensaje. Para privFlag = 1, se aplicó cifrado y para privFlag = 0, se aplicó autentificación. Se permiten todas las combinaciones excepto (privFlag = 1 Y authFlag = 0); es decir, no se permite cifrado sin autentificación.

- **msgSecurityModel:** un identificador en el rango de 0 a $2^{31}-1$ que indica qué modelo de seguridad usó el emisor para preparar este mensaje y, por lo tanto, qué modelo de seguridad debe usar el receptor para procesarlo. Los valores reservados incluyen uno para SNMPv1, dos para SNMPv2c (SNMPv2 con la herramienta de comunidad SNMPv1) y tres para SNMPv3.

Modelo de seguridad de usuario

El RFC 2574 define el modelo de seguridad de usuario (USM). USM proporciona servicios de autentificación y privacidad para SNMP. Concretamente, USM está diseñado para proteger de las siguientes amenazas:

- **Modificación de información:** una entidad podría alterar un mensaje en tránsito generado por una entidad autorizada de tal forma que cause operaciones de gestión no autorizadas, incluyendo fijar valores de objetos. La base de esta amenaza es que una entidad no autorizada podría cambiar cualquier parámetro de gestión, incluyendo los relacionados con la configuración, las operaciones y el registro de actividades (*accounting*).
- **Suplantación:** una entidad puede emprender operaciones de gestión para las que no tiene autorización, asumiendo la identidad de una entidad autorizada.
- **Modificación del flujo de mensajes:** SNMP está diseñado para operar sobre un protocolo de transporte no orientado a conexión. Hay una amenaza por la que los mensajes SNMP podrían ser reordenados, retrasados o repetidos (duplicados) para producir operaciones de gestión no autorizadas. Por ejemplo, se podría copiar un mensaje para reiniciar un dispositivo y repetirlo más tarde.
- **Revelación:** una entidad podría observar los intercambios entre un gestor y un agente y, por lo tanto, descubrir los valores de objetos gestionados y los eventos notificados. Por ejemplo, la observación de un comando Set que cambia las contraseñas podría permitir a un atacante descubrir las nuevas contraseñas.

USM no está diseñado contra las siguientes amenazas:

- **Denegación de servicio:** un atacante puede imposibilitar los intercambios entre un gestor y un agente.
- **Ánalisis del tráfico:** un atacante puede observar el patrón general del tráfico entre gestores y agentes.

La falta de medidas contra la amenaza de denegación de servicio se puede justificar de dos formas: en primer lugar, estos ataques se distinguen en muchos casos del tipo de fallos en la red con los que de hecho se las tiene que arreglar cualquier aplicación viable de gestión de red; y segundo, es probable que un ataque de denegación de servicio interrumpa todos los tipos de intercambio, que es una cuestión de una herramienta general de seguridad y no de una insertada en un protocolo de gestión de red. En lo que respecta al análisis del tráfico, muchos patrones de tráfico de gestión de redes son predecibles (por ejemplo, las entidades pueden gestionarse mediante comandos SNMP emitidos de forma regular por una o varias estaciones de gestión) y, por lo tanto, la protección contra la observación de estos patrones de tráfico no implica ventajas significativas.

Funciones criptográficas

Dos funciones criptográficas se definen para USM: autentificación y cifrado. Para permitir estas funciones, un motor SNMP requiere dos valores: una clave privada (*privKey*) y una clave de autentificación (*authKey*). Los valores separados de estas dos claves se mantienen para los siguientes usuarios:

- **Usuarios locales:** cualquier agente principal en este motor SNMP para el cual se autorizan operaciones de gestión.
- **Usuarios remotos:** cualquier agente principal en un motor SNMP remoto para el cual se desea la comunicación.

Estos valores son atributos de usuario almacenados para cada usuario relevante. Los valores de privKey y authKey no son accesibles mediante SNMP.

USM permite el uso de uno de los dos protocolos de autentificación alternativos: HMAC-MD5-96 y HMAC-SHA-96. HMAC, descrito en el Capítulo 3, usa una función *hash* segura y una clave secreta para producir un código de autentificación de mensajes. Para HMAC-MD5-96, HMAC se usa con MD5 como función *hash* subyacente. Como entrada al algoritmo HMAC se usa una clave de autentificación de 16 octetos (128 bits). El algoritmo produce una salida de 128 bits, que se trunca en 12 octetos (96 bits). Para HMAC-SHA-96, la función *hash* subyacente es SHA-1. La clave de autentificación tiene una longitud de 20 octetos. El algoritmo produce una salida de 20 octetos, que nuevamente se trunca en 12 octetos.

USM usa el modo CBC del DES para el cifrado. Como entrada al protocolo de cifrado, se proporciona una clave privada de 16 octetos. Los primeros ocho octetos (64 bits) de esta clave privada se usan como una clave DES. Como DES sólo requiere una clave de 56 bits, se ignora el bit menos significativo de cada octeto. Para el modo CBC, se necesita un vector de inicialización de 64 bits. Los últimos ocho octetos de la clave privada contienen un valor que se usa para generar dicho vector de inicialización.

Motores autoritativos y no autoritativos

En cualquier transmisión de mensajes, una de las dos entidades, emisor o receptor, está designada como el motor SNMP autoritativo, según las siguientes reglas:

- Cuando un mensaje SNMP contiene una carga útil que espera una respuesta (por ejemplo, una Get, GetNext, GetBulk, Set o Inform PDU), entonces el receptor de dichos mensajes es autoritativo.
- Cuando un mensaje SNMP contiene una carga útil que no espera una respuesta (por ejemplo, una SNMPv2-Trap, Response o Report PDU), entonces el emisor de dicho mensaje es autoritativo.

Así, para los mensajes enviados en nombre de un generador de comandos y para mensajes Inform de un iniciador de notificación, el receptor es autoritativo. Para mensajes enviados en nombre de un respondedor de comandos o para mensajes Trap de un iniciador de notificación, el emisor es autoritativo. Esta designación sirve con dos finalidades:

- La validez temporal de un mensaje se determina con respecto a un reloj que mantiene el motor autoritativo. Cuando un motor autoritativo envía un mensaje (Trap, Response, Report), contiene el valor actual de su reloj, de forma que el receptor no autoritativo puede sincronizarse con ese reloj. Cuando un motor no autoritativo envía un mensaje (Get, GetNext, GetBulk, Set, Inform), incluye su estimación actual del valor temporal en el destino, permitiendo que el destino acceda a la validez temporal del mensaje.
- Un proceso de localización de clave, que se describe más tarde, permite que una sola entidad principal tenga claves almacenadas en múltiples motores; estas claves están localizadas para el motor autoritativo de tal forma que la entidad principal sea responsable de una sola clave y se evite, así, el riesgo que corre la seguridad si se almacenan copias múltiples de la misma clave en una red distribuida.

Tiene sentido designar al receptor del generador de comandos y las PDU Inform como el motor autoritativo y, por lo tanto, responsable de comprobar la validez temporal del mensaje. Si una respuesta o *trap* se retrasa o se repite, no se producirían muchos daños. Sin embargo, el generador de comandos y, hasta cierto punto, las PDU Inform dan como resultado operaciones de gestión como, por ejemplo, la lectura o el establecimiento de objetos MIB. Así, es importante garantizar que dichas PDU no se retrasan ni se repiten, lo cual podría ocasionar efectos no deseados.

Parámetros de mensajes USM

Cuando el procesador de mensajes pasa un mensaje saliente al USM, éste rellena los parámetros relacionados con la seguridad en la cabecera del mensaje. Cuando el procesador de mensajes pasa un mensaje entrante al USM, el USM procesa los valores contenidos en esos campos. Los parámetros relacionados con la seguridad son los siguientes:

- **msgAuthoritativeEngineID:** el snmpEngineID del motor SNMP autoritativo implicado en el intercambio de este mensaje. Así, este valor se refiere a la fuente para un Trap, Response o Report, y al destino para un Get, GetNext, GetBulk, Set o Inform.
- **msgAuthoritativeEngineBoots:** el valor snmpEngineBoots del motor SNMP autoritativo implicado en el intercambio de este mensaje. El objeto snmpEngineBoots es un entero del rango entre 0 y $2^{31}-1$ que representa el número de veces que este motor SNMP se ha inicializado o reinicializado desde su configuración inicial.
- **msgAuthoritativeEngineTime:** el valor snmpEngineTime del motor SNMP autoritativo implicado en el intercambio de este mensaje. El objeto snmpEngineTime es un entero de entre 0 y $2^{31}-1$ que representa el número de segundos desde que este motor SNMP autoritativo incrementó por última vez el objeto snmpEngineBoots. Cada motor SNMP autoritativo es responsable de incrementar su propio valor snmpEngineTime una vez por segundo. Un motor no autoritativo es responsable de incrementar su noción de snmpEngineTime para cada motor autoritativo remoto con el que se comunique.
- **msgUserName:** el agente principal en cuyo nombre se está intercambiando el mensaje.
- **msgAuthenticationParameters:** es nulo si no se está usando autentificación para este intercambio. De lo contrario, es un parámetro de autentificación. Para la definición actual de USM, el parámetro de autentificación es un código de autentificación de mensajes HMAC.
- **msgPrivacyParameters:** es nulo si no se usa privacidad para este intercambio. De lo contrario, es un parámetro de privacidad. Para la definición actual de USM, el parámetro de privacidad es un valor que se usa para formar el valor inicial en el algoritmo CBC del DES.

La Figura 8.10 resume la operación del USM. Para la transmisión de mensajes, se realiza el cifrado en primer lugar, si fuera necesario. La PDU extendida se cifra y se sitúa en la carga útil del mensaje, y el valor msgPrivacyParameters se fija al valor necesario para generar el valor inicial. Luego, se realiza la autentificación, si es necesario. El mensaje completo, incluyendo la PDU extendida se introduce en el HMAC, y el código de autentificación resultante se coloca en msgAuthenticacionParameters. Para los mensajes

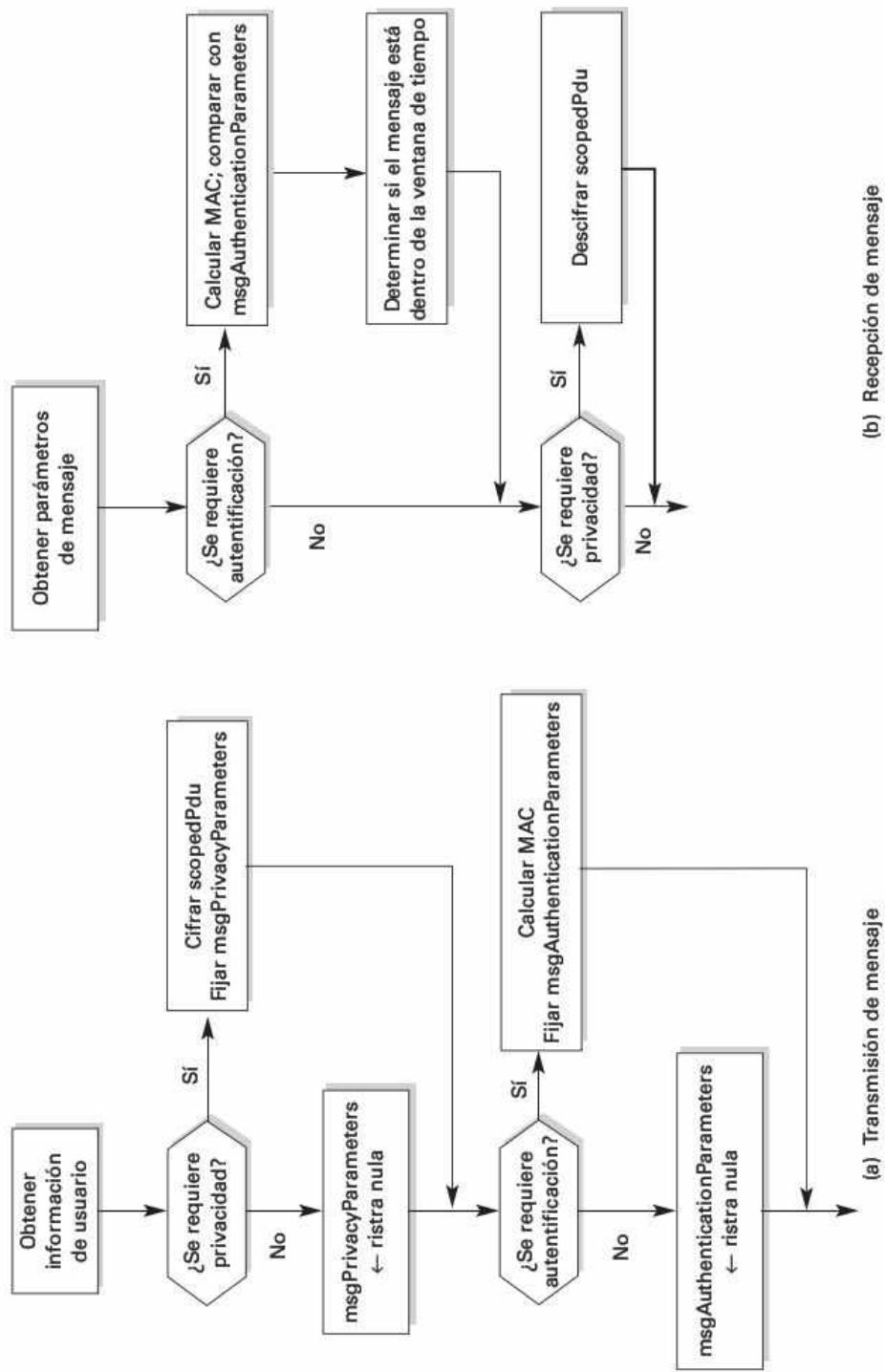


Figura 8.10 Procesamiento de mensajes USM

entrantes, la autenticación se realiza en primer lugar, si es necesario. El USM, primero, contrasta el MAC entrante con un MAC que calculó; si los dos valores coinciden, entonces el mensaje se considera auténtico (proviene de la fuente supuesta y no ha sido alterado durante la transmisión). Luego el USM comprueba si el mensaje se encuentra en una ventana de tiempo válida, como se explica más tarde. Si el mensaje no es válido en tiempo, se descarta como no auténtico. Por último, si se ha cifrado la PDU extendida, el USM lleva a cabo un descifrado y devuelve el texto claro.

Mecanismos de validez temporal de USM

USM incluye un conjunto de mecanismos de validez temporal para evitar retrasos o repeticiones en los mensajes. Cada motor SNMP que pueda actuar en capacidad de un motor autoritativo debe mantener dos objetos, `snmpEngineBoots` y `snmpEngineTime`, que se refieren a su valor de tiempo local. Cuando se instala un motor SNMP por primera vez, estos dos valores se fijan a cero. A partir de ahí, `snmpEngineTime` se incrementa una vez por segundo. Si alguna vez `snmpEngineTime` alcanza su valor máximo ($2^{31}-1$), `snmpEngineBoots` se incrementa como si se hubiese reiniciado el sistema, y `snmpEngineTime` se fija a cero y empieza a incrementarse otra vez. Usando un mecanismo de sincronización, un motor no autoritativo mantiene una estimación de los valores temporales de cada motor autoritativo con el que se comunica. Estos valores estimados se ponen en cada mensaje saliente y permiten al motor autoritativo receptor determinar si el valor temporal del mensaje entrante es válido o no.

El mecanismo de sincronización funciona de la siguiente forma. Un motor no autoritativo guarda una copia local de tres variables para cada motor SNMP autoritativo conocido por ese motor:

- **`snmpEngineBoots`**: el valor más reciente de `snmpEngineBoots` para el motor autoritativo remoto.
- **`snmpEngineTime`**: esta noción del motor de `snmpEngineTime` para el motor autoritativo remoto. Este valor se sincroniza con el motor autoritativo remoto mediante el proceso de sincronización que se describe más tarde. Entre los sucesos de sincronización, este valor se incrementa una vez por segundo para mantener una sincronización aproximada con el motor autoritativo remoto.
- **`latestReceivedEngineTime`**: el valor más alto de `msgAuthoritativeEngineTime` que este motor ha recibido del motor autoritativo remoto; este valor se actualiza cada vez que se recibe un valor mayor de `msgAuthoritativeEngineTime`. La finalidad de esta variable es proteger contra ataques de repetición que podrían hacer que la noción de `snmpEngineTime` del motor SNMP no autoritativo no avanzara.

Para cada motor autoritativo remoto conocido por este motor se mantiene un conjunto formado por estas tres variables. Lógicamente, los valores se mantienen en algún tipo de caché, indexado por el `snmpEngineID` exclusivo de cada motor autoritativo remoto.

Para hacer posible que los motores no autoritativos mantengan la sincronización de tiempo, cada motor autoritativo inserta sus valores de inicio y de tiempo, así como su valor de `snmpEngineID`, en cada mensaje saliente de respuesta, de informe y de interceptación en los campos `msgAuthoritativeEngineBoots`, `msgAuthoritativeEngineTime` y

`msgAuthoritativeEngineID`. Si el mensaje es auténtico y está dentro de la ventana de tiempo, entonces el motor no autoritativo receptor actualiza sus variables locales (`snmpEngineBoots`, `snmpEngineTime` y `latestReceivedEngineTime`) para ese motor remoto de acuerdo con las siguientes reglas:

1. Se produce una actualización si al menos es cierta una de las dos condiciones siguientes:
 - (`msgAuthoritativeEngineBoots > snmpEngineBoots`) O
 - [$(\text{msgAuthoritativeEngineBoots} = \text{snmpEngineBoots})$ Y
 - (`msgAuthoritativeEngineTime > latestReceivedEngineTime`)]

La primera condición dice que debería producirse una actualización si el valor de inicio del motor autoritativo ha aumentado desde la última actualización. La segunda condición dice que si el valor de inicio no ha aumentado, entonces debería producirse una actualización si el valor temporal del motor entrante es mayor que la última recibida. El valor de tiempo del motor entrante será menor que el último recibido si dos mensajes entrantes llegan desordenados, lo que puede ocurrir, o si se está produciendo un ataque de repetición; en cualquier caso, el motor receptor no realizará una actualización.

2. Si se hace una llamada de actualización, se llevan a cabo los siguientes cambios:
 - fijar `snmpEngineBoots` al valor de `msgAuthoritativeEngineBoots`
 - fijar `snmpEngineTime` al valor de `msgAuthoritativeEngineTime`
 - fijar `latestReceivedEngineTime` al valor de `msgAuthoritativeEngineTime`

Si damos la vuelta a la lógica se observa que si `msgAuthoritativeEngineBoots < snmpEngineBoots`, no se produce actualización. Un mensaje como este no se considera auténtico y debe ser ignorado. Si `msgAuthoritativeEngineBoots = snmpEngineBoots`, pero `msgAuthoritativeEngineTime < latestReceivedEngineTime`, tampoco se produce actualización. En este caso, el mensaje puede ser auténtico, pero puede estar desordenado, en cuyo caso no se garantiza una actualización de `snmpEngineTime`.

Obsérvese que la sincronización sólo se utiliza si el servicio de autentificación está en uso para este mensaje y se ha determinado que el mensaje es auténtico mediante HMAC. Esta restricción es esencial porque el ámbito de autentificación incluye `msgAuthoritativeEngineID`, `msgAuthoritativeEngineBoots` y `msgAuthoritativeEngineTime`, asegurando así que estos valores son válidos.

SNMPv3 dicta que un mensaje debe recibirse dentro de una ventana de tiempo razonable, para evitar ataques de retraso y de repetición. La ventana de tiempo debería elegirse de forma que fuera lo más pequeña posible dada la exactitud de los relojes implicados, los retrasos de las comunicaciones y la frecuencia con que se sincronizan estos relojes. Si la ventana de tiempo se fija demasiado pequeña, los mensajes auténticos serán rechazados como no auténticos. Por otra parte, una ventana de tiempo grande aumenta la vulnerabilidad a retrasos dañinos de mensajes.

Consideramos el caso más importante de un receptor autoritativo; la comprobación de la validez temporal por parte de un receptor no autoritativo difiere ligeramente. Con cada mensaje entrante que se ha autenticado y cuyo `msgAuthoritativeEngineID` es el mismo que el valor de `snmpEngineID` para este motor, el motor compara los valores de `msgAuthoritativeEngineBoots` y `msgAuthoritativeEngineTime` del mensaje entrante con

los valores de `snmpEngineBoots` y `snmpEngineTime` que mantiene este motor para sí mismo. El mensaje entrante se considera fuera de la ventana de tiempo si alguna de estas condiciones es verdadera:

- $\text{snmpEngineBoots} = 2^{31} - 1$ O
- $\text{msgAuthoritativeEngineBoots} \neq \text{snmpEngineBoots}$ O
- el valor de `msgAuthoritativeEngineTime` difiere del valor de `snmpEngineTime` en más de ± 150 segundos.

La primera condición dice que si `snmpEngineBoots` se queda fijo en su valor máximo, ningún mensaje entrante se puede considerar auténtico. La segunda condición dice que un mensaje debe tener un tiempo de inicio igual al del motor local; por ejemplo, si el motor local se ha reiniciado y el motor remoto no se ha sincronizado con el motor local desde el reinicio, los mensajes de ese motor remoto no se consideran auténticos. La condición final dice que el valor de tiempo del mensaje entrante debe ser mayor que el valor de tiempo local menos 150 segundos y menor que el valor de tiempo local más 150 segundos.

Si un mensaje se considera fuera de la ventana de tiempo, el mensaje no se considera auténtico y se devuelve una indicación de error (`notInTimeWindow`) al módulo que hizo la llamada.

Nuevamente, como con la sincronización, la comprobación de la validez de tiempo sólo se lleva a cabo si el servicio de autenticación está en uso y el mensaje es auténtico, asegurando la validez de los campos de cabecera del mensaje.

Localización de claves

Un requisito para el uso de los servicios de autenticación y de privacidad de SNMPv3 es que, para cualquier comunicación entre una entidad principal en un motor no autoritativo y un motor autoritativo remoto, se deben compartir una clave de autenticación secreta y una de privacidad secreta. Estas claves permiten a un usuario en un motor no autoritativo (generalmente, un sistema de gestión) emplear autenticación y privacidad con sistemas autoritativos remotos que gestionan el usuario (normalmente, sistemas agentes). El RFC 2574 proporciona directrices para la creación, actualización y gestión de estas claves.

Para simplificar la carga de la gestión de claves a las entidades principales, cada una sólo tiene que mantener una única clave de autenticación y una única clave de cifrado. Estas claves no se almacenan en una MIB y no se puede acceder a ellas a través de SNMP. En este subapartado, se observan primero las técnicas para la generación de estas claves a partir de una contraseña. A continuación, nos fijaremos en el concepto de localización de claves, que permite a una entidad principal compartir una sola clave de autenticación y cifrado con cada motor remoto, mientras mantiene localmente una sola clave de autenticación y cifrado. Estas dos técnicas fueron propuestas por primera vez en [BLUM97a].

Un usuario necesita una clave de privacidad de 16 octetos y una clave de autenticación de 16 o 20 octetos. Para las claves pertenecientes a usuarios humanos, es conveniente que el usuario pueda emplear una contraseña legible para ellos, en vez de una

clave de ristras de bits. Por lo tanto, el RFC 2574 define un algoritmo para establecer la correspondencia entre la contraseña del usuario y una clave de 16 o de 20 octetos. USM no impone restricciones a la contraseña en sí misma, pero las políticas locales de gestión obligan a los usuarios a emplear contraseñas que no sean fáciles de adivinar.

La generación de claves a partir de contraseñas se realiza de la siguiente manera:

- Usar la contraseña del usuario como entrada y producir una ristra de 2^{20} octetos (1.048.576 octetos) repitiendo el valor de la contraseña tantas veces como sea necesario, truncando el último valor si fuese necesario, para formar la ristra digest0. Por ejemplo, una contraseña de ocho caracteres (2^3 octetos) se concatenaría consigo misma 2^{17} veces para formar digest0.
- Si se desea una clave de 16 octetos, se toma el *hash* MD5 de digest0 para formar digest1. Si se quiere una clave de 20 octetos, se toma el *hash* SHA-1 de digest0 para formar digest1. La salida es la clave del usuario.

Una ventaja de esta técnica es que reduce en gran medida el ataque de diccionario por fuerza bruta, en el que un adversario prueba con muchas posibles contraseñas, generando la clave a partir de cada una, y luego comprueba si la clave resultante funciona con los datos de autentificación o cifrado de que dispone. Por ejemplo, si un atacante intercepta un mensaje autenticado, podría intentar generar el valor HMAC con distintas claves de usuario posibles. Si se produce una coincidencia, el atacante puede asumir que la contraseña ha sido descubierta. Este proceso de dos pasos aumenta de forma significativa la cantidad de tiempo que necesitará un ataque de este tipo.

Otra ventaja de esta técnica es que desasocia las claves de usuario de cualquier sistema particular de gestión de red (NMS, *Network Management System*). Ningún NMS necesita almacenar valores de claves de usuario. En vez de eso, cuando sea necesario, una clave de usuario se genera a partir de la contraseña de ese usuario. [BLUM97b] presenta las siguientes consideraciones que promueven el uso de un enfoque de contraseña independiente de NMS:

- Si hay que almacenar una clave, en vez de generarla a partir de una contraseña, una alternativa es mantener un depósito centralizado de claves secretas. Pero, por otra parte, esto afecta la fiabilidad general y puede hacer que la resolución de problemas sea imposible si no se puede acceder a dicho depósito cuando sea necesario.
- Por otra parte, si se mantienen depósitos duplicados, esto pone en peligro la seguridad general proporcionando a los posibles atacantes más objetivos que atacar.
- Si se usan depósitos centralizados o depósitos múltiples duplicados, deben guardarse en ubicaciones seguras. Esto puede reducir la oportunidad de establecer un «campamento de reenvío» durante la extinción del peligro (por ejemplo, resolución de problemas cuando segmentos impredecibles de la red están inoperativos y/o inaccesibles durante un período no previsible de tiempo).

Se podría usar una sola contraseña para generar una única clave tanto para autentificación como para cifrado. Un esquema más seguro consiste en usar dos contraseñas, una para generar una clave de autentificación y otra para generar una clave de cifrado diferenciada.

Una clave localizada se define en el RFC 2574 como una clave secreta compartida entre un usuario y un motor SNMP autoritativo. El objetivo es que el usuario sólo tenga que mantener una clave (o dos claves si se requiere autentificación y privacidad) y, por lo tanto, sólo necesite recordar una contraseña (o dos). Los secretos reales compartidos entre un usuario particular y cada motor SNMP autoritativo son diferentes. El proceso mediante el cual una clave única de usuario se convierte en múltiples claves únicas, una para cada motor SNMP remoto, se conoce como localización de claves. [BLUM97a] presenta las razones para el uso de esta estrategia, que se resume a continuación.

Se pueden definir los siguientes objetivos para la gestión de claves:

- Cada sistema agente SNMP en una red distribuida tiene su propia clave única para que cada usuario autorizado la gestione. Si hay múltiples usuarios autorizados como administradores, el agente tiene una clave única de autentificación y de cifrado para cada usuario. Así, si la clave para un usuario está comprometida, no se ven afectadas las claves de otros usuarios.
- Las claves para un usuario en diferentes agentes son distintas. Así, si un agente está comprometido, sólo las claves de usuario para ese agente están comprometidas y no las claves del usuario en vigor en otros agentes.
- La gestión de red se puede llevar a cabo desde cualquier punto de la red, independientemente de la disponibilidad de un sistema de gestión de red preconfigurado (NMS). Esto permite que un usuario realice funciones de gestión desde cualquier estación de gestión. Esta capacidad la proporciona el algoritmo de contraseña a clave descrito anteriormente.

También se pueden definir los siguientes aspectos que deben ser evitados:

- Un usuario tiene que recordar (o de otra forma gestionar) un gran número de claves, número que aumenta con la incorporación de nuevos agentes gestionados.
- Un adversario que descubra una clave para un agente es capaz de suplantar a cualquier otro agente ante cualquier usuario, o a cualquier usuario ante otro agente.

Para alcanzar los objetivos y las consideraciones que se han mencionado, se correlaciona una sola clave de usuario, por medio de una función unidireccional no reversible (por ejemplo, una función *hash* segura) con distintas claves localizadas para diferentes motores autenticados (diferentes agentes). El procedimiento es el siguiente:

- Formar la ristra digest2 concatenando digest1 (que se describió anteriormente), el valor snmpEngineID del motor autoritativo y digest1.
- Si se desea una clave de 16 octetos, se toma el *hash* MD5 de digest2. Si se desea una clave de 20 octetos, se toma el *hash* SHA-1 de digest2. La salida es la clave localizada del usuario.

La clave localizada resultante se puede, luego, configurar en el sistema del agente de alguna forma segura. Debido a la naturaleza unidireccional de MD5 y SHA-1, es imposible que un adversario descubra una clave de usuario, incluso si consigue descubrir una clave localizada.

La Figura 8.11 resume el proceso de localización de claves.

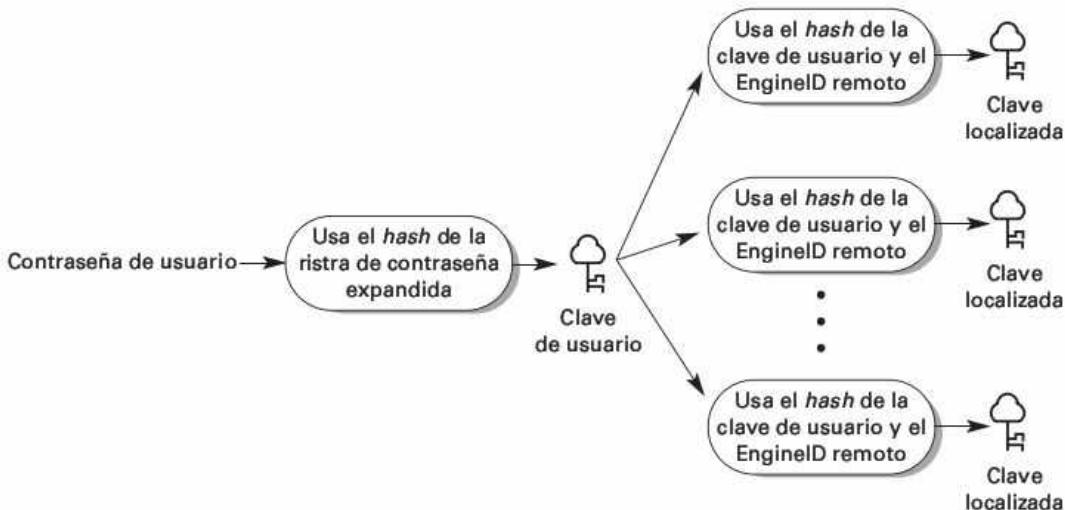


Figura 8.11 Localización de clave

CONTROL DE ACCESO BASADO EN VISTAS

El control de acceso es una función que se realiza en el nivel de PDU. Un documento sobre el control de acceso define los mecanismos para determinar si debería permitirse el acceso a un objeto gestionado en una MIB local por una entidad principal remota. Se podrían definir muchos mecanismos de control de acceso imaginables. Los documentos SNMPv3 definen el modelo de control de acceso basado en vistas (VACM, *View-Based Access Control Model*). Dicho modelo hace uso de una MIB que define la política de control de acceso para este agente y hace posible usar configuración remota.

El modelo tiene dos características fundamentales:

- VACM determina si debería permitirse el acceso a un objeto gestionado en una MIB local por un usuario remoto.
- VACM hace uso de una MIB que
 - define la política de control para este agente
 - hace posible usar configuración remota

Elementos del modelo VACM

El RFC 2575 define cinco elementos que conforman el VACM: grupos, nivel de seguridad, contextos, vistas MIB y política de acceso.

Un **grupo** se define como un conjunto de cero o más secuencias <securityModel, securityName> en cuyo nombre se puede acceder a objetos de gestión SNMP. Un securityName se refiere a una entidad principal, y los derechos de acceso para todas las entidades principales de un grupo determinado son idénticos. Un groupName único se asocia con cada grupo. El concepto de grupo es una herramienta útil para la categoriza-

ción de los gestores con respecto a los derechos de acceso. Por ejemplo, todos los administradores del nivel superior pueden tener una serie de derechos de acceso, mientras que los administradores de niveles intermedios pueden tener una serie de derechos de acceso diferente.

Cualquier combinación dada de securityModel y securityName puede pertenecer al menos a un grupo. Es decir, para este agente, una entidad principal dada cuyas comunicaciones estén protegidas por un modelo de seguridad determinado sólo puede incluirse en un grupo.

Los derechos de acceso para un grupo pueden diferir dependiendo del **nivel de seguridad** del mensaje que contiene la solicitud. Por ejemplo, un agente puede permitir acceso de sólo lectura (*read-only*) para una solicitud transmitida en un mensaje autenticado, pero puede requerir autenticación para el acceso de escritura (*write*). Además, para determinados objetos sensibles, el agente puede requerir que la solicitud y su respuesta se transmitan usando el servicio de privacidad.

Un **contexto MIB** es un subconjunto, al que se asigna un nombre, de las instancias de objeto en la MIB local. Los contextos proporcionan una forma útil de agregar objetos a colecciones con diferentes políticas de acceso.

El contexto es un concepto que se relaciona con el control de acceso. Cuando una estación de gestión interactúa con un agente para acceder a la información de gestión en el agente, entonces la interacción se produce entre una entidad principal de gestión y el motor SNMP del agente, y los privilegios de control de acceso se expresan en una vista MIB que se aplica a esta entidad principal y a este contexto. Los contextos tienen las siguientes características fundamentales:

- Una entidad SNMP, identificada únicamente por un contextEngineID, puede tener más de un contexto.
- Un objeto o una instancia de objeto puede aparecer en más de un contexto.
- Cuando existen múltiples contextos, para identificar una instancia de un objeto individual, se deben identificar su contextName y contextEngineID, además de su tipo de objeto e instancia.

Con frecuencia se desea restringir el acceso de un grupo particular a una parte de los objetos gestionados en un agente. Para lograr este objetivo, el acceso a un contexto se realiza por medio de una **vista MIB**, que define un conjunto específico de objetos gestionados (y, opcionalmente, instancias específicas de objetos). VACM hace uso de una técnica potente y flexible para definir las vistas MIB, basada en los conceptos de subárboles de vistas y familias de vistas. La vista MIB se define como una colección, o familia, de subárboles, cada uno de los cuales se incluye en la vista o se excluye de ella.

Los objetos gestionados en una base de datos local se organizan de forma jerárquica o de árbol, basándose en sus identificadores de objeto. Esta base de datos local contiene un subconjunto de todos los tipos de objeto definidos según el estándar de Internet Estructura de la Información de Gestión (SMI, *Structure of Management Information*) e incluye instancias de objeto cuyos identificadores se ajustan a las convenciones de SMI.

SNMPv3 incluye el concepto de subárbol. Un subárbol es simplemente un nodo de la jerarquía de denominación de la MIB más todos sus elementos subordinados. Más

formalmente, un subárbol puede definirse como un conjunto de todos los objetos e instancias de objetos que añaden a sus nombres un prefijo común ASN.1 OBJECT IDENTIFIER (identificador de objeto). El prefijo común más largo de todas las instancias del subárbol es el identificador de objeto del nodo padre de ese subárbol.

Hay tres vistas MIB asociadas con cada entrada en vacmAccessTable, una para leer, otra para escribir y otra para notificar acceso. Cada vista MIB se compone de una serie de subárboles de vistas. Cada subárbol de vistas en la vista MIB se especifica como incluido o excluido. Es decir, la vista MIB incluye o excluye todas las instancias de objetos contenidas en ese subárbol. Además, se define una máscara de vista para reducir la cantidad de información de configuración necesaria cuando se requiere control de acceso exhaustivo (por ejemplo, control de acceso en el nivel de instancia de objeto).

VACM permite que un motor SNMP se configure para imponer una serie concreta de derechos de acceso, lo cual constituye una **política de acceso**. La determinación de acceso depende de los siguientes factores:

- La **entidad principal** que realiza la solicitud de acceso. El VACM hace posible que un agente permita diferentes privilegios de acceso a distintos usuarios. Por ejemplo, un sistema gestor responsable de la configuración de una red amplia puede tener absoluta autoridad para alterar elementos en la MIB local, mientras que un administrador de nivel intermedio con responsabilidad de supervisión puede tener acceso sólo de lectura y puede, además, estar limitado a acceder sólo a un subconjunto de la MIB local. Como se ha explicado, las entidades principales están asignadas a grupos y la política de acceso se especifica con respecto a grupos.
- El **nivel de seguridad** por el cual se transmitió una solicitud en un mensaje SNMP. Normalmente, un agente requerirá el uso de autenticación para los mensajes que contengan una solicitud de establecimiento (operación de escritura).
- El **modelo de seguridad** usado para procesar el mensaje de solicitud. Si se implementan en un agente múltiples modelos de seguridad, el agente puede configurarse para proporcionar diferentes niveles de acceso a las solicitudes comunicadas por mensajes procesados mediante diferentes modelos de seguridad. Por ejemplo, determinados elementos pueden estar accesibles si el mensaje de solicitud viene a través de USM, y no lo están si el modelo de seguridad es SNMPv1.
- El **contexto MIB** para la solicitud.
- La **instancia de objeto** específica para la cual se requiere el acceso. Algunos objetos guardan información más crítica o confidencial que otros y, por lo tanto, la política de acceso debe depender de la instancia específica de objeto que se ha solicitado.
- El **tipo de acceso** solicitado (de lectura, escritura o notificación). Leer, escribir y notificar son operaciones de gestión bien diferenciadas, y las distintas políticas de control de acceso pueden aplicarse para cada una de estas operaciones.

Procesamiento de control de acceso

Una aplicación SNMP invoca al VACM mediante la primitiva isAccessAllowed, con los parámetros de entrada securityModel, securityName, securityLevel, viewType, context-

Name y variableName. Todos estos parámetros son necesarios para tomar la decisión de control de acceso. En otras palabras, el subsistema de control de acceso se define de tal forma que proporcione una herramienta muy flexible para configurar el control de acceso en el agente, dividiendo los componentes de la decisión de control de acceso en seis variables separadas.

La Figura 8.12, adaptada de una figura de RFC 2575, proporciona una forma útil de observar las variables de entrada y muestra cómo entran en juego las distintas tablas en la MIB de VACM tomando la decisión de control de acceso.

- **Quién:** la combinación de securityModel y securityName definen el *quién* de esta operación; identifica una entidad principal dada cuyas comunicaciones están protegidas por un securityModel dado. Esta combinación pertenece al menos a un grupo en este motor SNMP. El vacmSecurityToGroupTable proporciona el groupName, dados el securityModel y el securityName.
- **Dónde:** el contextName especifica *dónde* se va a encontrar el objeto gestionado deseado. El vacmContextTable contiene una lista de los contextNames reconocidos.
- **Cómo:** la combinación de securityModel y securityLevel define *cómo* se protegió la solicitud entrante o la PDU Inform. La combinación de quién, dónde y cómo identifica cero o una entrada en vacmAccessTable.
- **Por qué:** el viewType especifica *por qué* se solicita el acceso: para una operación de lectura, escritura o notificación. La entrada seleccionada en vacmAccessTable contiene un viewName de la MIB para cada uno de estos tres tipos de operación, y viewType se usa para seleccionar un viewName específico. Este viewName selecciona la vista MIB adecuada de vacmViewTreeFamilyTable.
- **Qué:** el variableName es un identificador de objeto cuyo prefijo identifica un tipo de objeto específico y cuyo sufijo identifica una instancia específica de objeto. El tipo de objeto indica *qué* tipo de información de gestión se solicita.
- **Cuál:** la instancia de objeto indica *cuál* de los elementos de información se solicita específicamente.

Por último, el variableName se compara con la vista MIB obtenida. Si coincide con un elemento de la vista MIB, entonces se concede el acceso.

Motivación

Los conceptos que conforman el VACM parecen dar como resultado una definición compleja del control de acceso. Las razones para la introducción de estos conceptos son clarificar las relaciones implicadas en el acceso a la información de gestión y minimizar los requisitos de almacenamiento y procesamiento en el agente. Para entender estos motivos, considérese lo siguiente. En SNMPv1, el concepto de comunidad se usa para representar la siguiente información relacionada con la seguridad:

- La identidad de la entidad solicitante (estación de gestión)
- La identidad de la entidad que actúa (agente actuando para sí mismo o para una entidad representada mediante proxy)

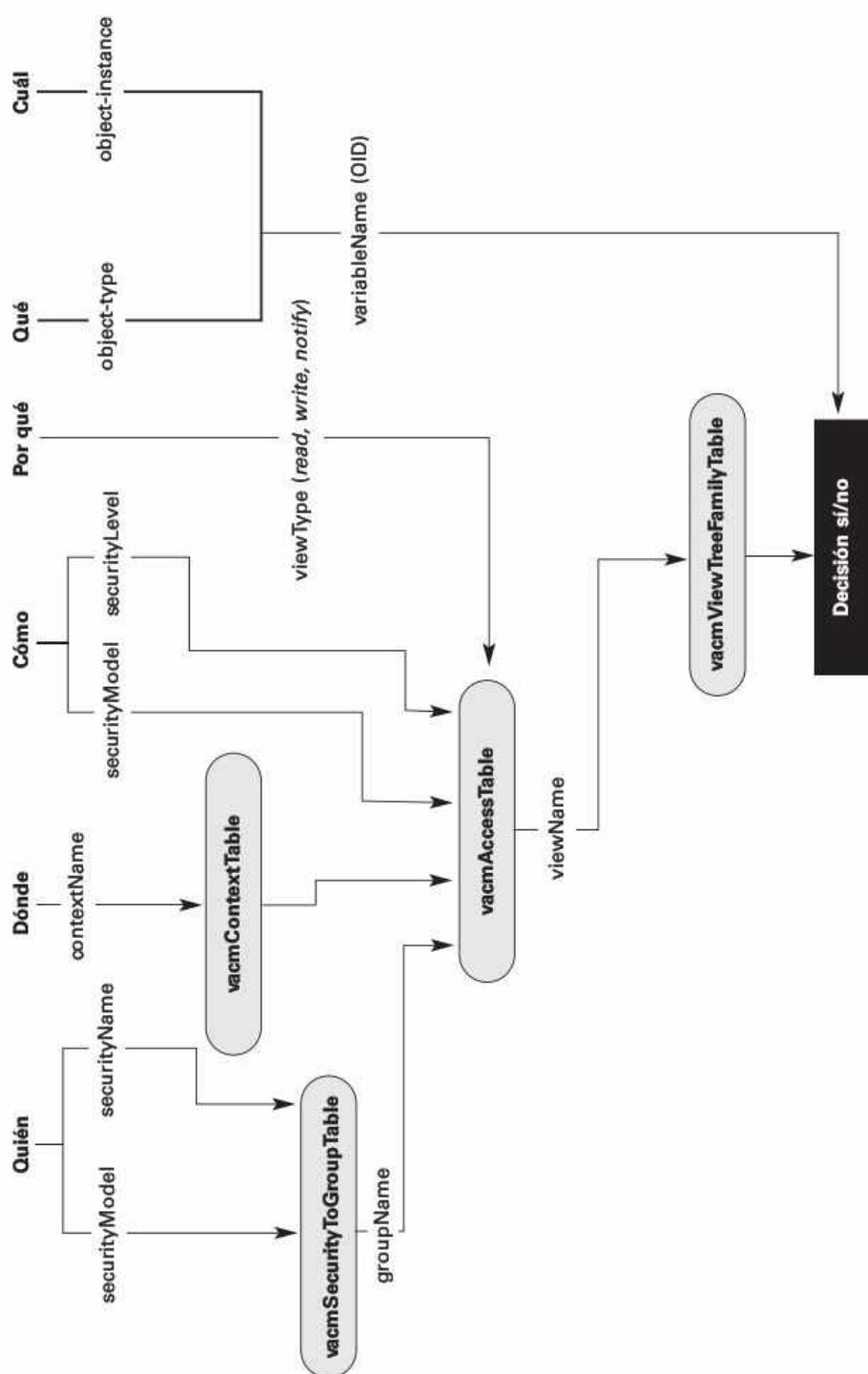


Figura 8.12 Lógica de VACM

- La identidad de la ubicación de la información de gestión a la que se va a acceder (agente o entidad representada mediante *proxy*)
- Información de autenticación
- Información de control de acceso (autorización para realizar la operación requerida)
- Información de la vista MIB

Agrupando todos estos conceptos en una única variable, se pierden la flexibilidad y la funcionalidad. VACM proporciona el mismo conjunto de información relativa a la seguridad usando variables diferenciadas para cada elemento. Ésta es una mejora sustancial con respecto a SNMPv1. Desasocia distintos conceptos para que los valores se puedan asignar a cada uno de forma separada.

8.4 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

[STAL99] ofrece un análisis exhaustivo y detallado de SNMP, SNMPv2 y SNMPv3; también proporciona una introducción general a la tecnología de gestión de red.

STAL99 Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.

Sitios web recomendados:

- **Grupo de trabajo de SNMPv3 de la IETF:** contiene las copias más recientes de los RFC y los borradores de Internet relacionados con SNMPv3, además de una planificación del trabajo pasado y futuro.
- **Sitio web SNMPv3:** mantenido por la Universidad Técnica de Braunschweig. Proporciona enlaces a los RFC y los borradores de Internet, copias de aclaraciones y cambios propuestos enviados por el grupo de trabajo, y enlaces a vendedores con implementaciones SNMPv3.
- **The Simple Web Site:** mantenido por la Universidad de Twente. Constituye una buena fuente de información sobre SNMP e incluye enlaces a muchas implementaciones de dominio público así como a listas de libros y artículos.

8.5 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

agente	localización de claves	Modelo de seguridad de
base de información de gestión (MIB)	nombre de comunidad	usuario (USM)
comunidad	modelo de control de acceso	política de acceso
estación de gestión	basado en vistas (VACM)	<i>proxy</i>
gestión de red	modelo de procesamiento	SNMP
	de mensajes	

PREGUNTAS DE REPASO

- 81.** ¿En qué sentido se considera integrada una arquitectura de gestión de red?
- 82.** ¿Cuáles son los elementos fundamentales del modelo SNMP?
- 83.** ¿Qué es una MIB?
- 84.** ¿Qué capacidades o comandos básicos se proporcionan en SNMPv1?
- 85.** ¿Cuál es la función de un *proxy* SNMP?
- 86.** Explica brevemente el concepto de comunidad de SNMPv1.
- 87.** ¿Cuál es la relación entre SNMPv1, SNMPv2 y SNMPv3?
- 88.** ¿Qué amenazas contrarresta USM?
- 89.** ¿Cuál es la diferencia entre un motor autoritativo y uno no autoritativo?
- 90.** ¿Qué es la localización de claves?
- 91.** Enumera y define brevemente los elementos que componen el VACM.

PROBLEMAS

- 81.** SNMPv1 define un tipo de datos conocidos como «calibrador» (*gauge*) e incluye la siguiente explicación de la semántica de este tipo:

Este tipo representa un entero no negativo, que puede aumentar o disminuir, pero que se queda fijo al alcanzar un valor máximo. Este estándar especifica un valor máximo de $2^{32} - 1$ (4294967295 decimal) para calibradores.

Desgraciadamente, la expresión *quedarse fijo* no se define, y esto ha dado como resultado dos interpretaciones. El estándar SNMPv2 aclaraba la ambigüedad con la siguiente definición:

El valor de un calibrador llega al máximo cuando la información que se está modelando sea mayor o igual que ese valor máximo; si la información que se está modelando posteriormente disminuye por debajo de dicho valor máximo, también disminuye el calibrador.

a) ¿Cuál es la interpretación alternativa?

b) Comenta los pros y los contra de las dos interpretaciones.

- 82.** En SNMPv1, cualquier objeto en una MIB tiene una categoría de acceso MIB, que puede asignarse a uno de los siguientes valores: sólo lectura, sólo escritura y no accesible. Con una operación Get o Trap se lleva a cabo una lectura, y con una operación Set se lleva a cabo escritura. Para sólo escritura, el objeto puede estar disponible para las operaciones Get y Trap, pero es dependiente de la implementación. La categoría de acceso MIB especifica el acceso máximo permitido para un objeto, pero en la herramienta de comunidad de SNMPv1, el modo de acceso puede restringir más este acceso para un perfil de comunidad dado. En la siguiente tabla, rellena cada entrada para mostrar el acceso permitido.

Categoría de Acceso MIB	Modo de Acceso SNMP	
	SÓLO LECTURA	SÓLO ESCRITURA
Sólo lectura		
Lectura/escritura		
Sólo escritura		
No accesible		

- 8.3** a) El RFC 2574 afirma que para un motor no autoritativo, los valores de msgAuthoritativeEngineBoots y msgAuthoritativeEngineTime en la cabecera de un mensaje saliente se establecen sólo si el mensaje ha de ser autenticado por el receptor autoritativo. ¿A qué se debe esta restricción?
- b) Sin embargo, para un mensaje de respuesta de un motor autoritativo, los valores de msgAuthoritativeEngineBoots y msgAuthoritativeEngineTime en la cabecera de un mensaje saliente siempre se establecen. ¿Por qué?
- 8.4** El RFC 2574 especifica que la sincronización del reloj (actualización del reloj local basada en valores entrantes) se produce antes de la verificación de la ventana de tiempo (comprueba la validez temporal del mensaje entrante). Esto significa que un motor no autoritativo puede actualizar su noción del reloj del motor autoritativo si el mensaje recibido es auténtico, incluso si el mensaje no está dentro de la ventana de tiempo. Desde la publicación del RFC, ha habido una discusión continua al respecto en la lista de correo SNMPv3, pero por el momento parece que lo estipulado en el estándar no va a cambiar. Es útil observar las implicaciones de este hecho. Dadas las siguientes definiciones:

MAEB = msgAuthoritativeEngineBoots

MAET = msgAuthoritativeEngineTime

SEB = noción local de snmpEngineBoots para el motor autoritativo remoto

SET = noción local de snmpEngineTime para el motor autoritativo remoto

LRET = latestReceivedEngineTime

Ahora, imagina que un motor autoritativo recibe un mensaje

$$(MAEB = SEB) \text{ Y } [LRET < MAET < (SET - 150)]$$

Luego las condiciones son correctas para una actualización de reloj, por lo que se lleva a cabo:

$$SET := MAET; LRET := MAET$$

Ahora, en la comprobación de la ventana de tiempo, tenemos

$$(MAEB = SEB) \text{ Y } (MAET = SET)$$

por lo que afirmamos que el mensaje es válido en términos de tiempo. Supón, sin embargo, que hemos hecho primero la comprobación de la ventana de tiempo. En términos de tiempo, ¿habríamos considerado el mensaje válido o no válido?

- 85.** En la versión original publicada de la especificación USM (RFC 2274), que trata los procedimientos de sincronización de reloj y de verificación de la ventana de tiempo, se hace el siguiente comentario: «Obsérvese que este procedimiento no permite la sincronización automática de tiempo si el motor SNMP no autoritativo tiene una situación real de fuera de sincronización, donde el motor SNMP autoritativo va a más de 150 segundos por detrás del motor SNMP no autoritativo.» Esta afirmación se retiró de la versión revisada (RFC 2574) después de que su autor indicara al grupo de trabajo que la afirmación no es siempre cierta. Usando el ejemplo del Problema 8.4, demuestra que dicha afirmación no es siempre verdadera.
- 86.** SNMPv3 asume que hay algún medio seguro de distribuir claves localizadas a sistemas autenticados (agentes). Esta distribución segura va más allá del ámbito de SNMPv3; puede ser manual o mediante otro protocolo seguro. Una vez que se ha entregado a un agente una clave inicial (o pareja de claves para autenticación y privacidad), SNMPv3 proporciona un mecanismo para actualizar de forma segura las claves. Es conveniente que las claves se cambien de vez en cuando para aumentar la seguridad. Un usuario, por sí mismo, podría iniciar el proceso de cambio de clave solicitando y proporcionando una nueva contraseña. De otro modo, el sistema de gestión de red o NMS podría iniciar el proceso solicitando una nueva contraseña. En cualquier caso, la clave de usuario en el NMS se actualiza. El NMS puede, después, calcular una clave localizada para cada agente de la comunicación. Luego, el NMS debe comunicarse de forma segura con cada agente para hacer que actualicen sus claves localizadas. Es obvio que el NMS no puede simplemente enviar la clave en texto claro por la red. Se presentan dos opciones:

- Cifrar la nueva clave usando la clave antigua como clave de cifrado.
- Usar algún tipo de función unidireccional para producir un valor a partir de la clave antigua. Aplicar un OR exclusivo a este valor y la nueva clave y enviar el resultado al agente. El agente puede luego aplicar un XOR al resultado entrante con la misma función aplicada a la clave antigua para producir la clave nueva.

SNMPv3 utiliza una variante del segundo método. ¿Cuál es la ventaja de este enfoque con respecto al primero?

- 87.** El enfoque de SNMPv3 implica el uso de un objeto KeyChange en la MIB del sistema meta. Una entidad principal remota o NMS fija este objeto, que luego usa automáticamente el agente para actualizar la clave correspondiente. El algoritmo ahora tiene dos fases, una tiene lugar en el motor solicitante y otra en el motor agente remoto.

El proceso empieza cuando un solicitante desea actualizar una clave existente keyOld a un nuevo valor keyNew. El solicitante realiza los siguientes pasos:

1. Generar un valor aleatorio (*random*) a partir de un generador de números pseudoaleatorios o un generador de números aleatorios.
2. Calcular

$$\text{digest} = \text{Hash}(\text{keyOld} \parallel \text{random})$$

donde *Hash* es MD5 o SHA-1, dependiendo de si se desea una clave de 16 octetos o de 20, y el símbolo \parallel representa concatenación.

3. Calcular

$$\text{delta} = \text{digest} \oplus \text{keyNew}$$

$$\text{protocolKeyChange} = (\text{random} \parallel \text{delta})$$

donde \oplus es la operación OR exclusivo.

4. El valor protocolKeyChange luego se envía al agente en un comando Set para actualizar una instancia de objeto KeyChange en la MIB del agente.

¿Qué debe hacer el agente con el valor entrante para actualizar la clave?

8.8 Un procedimiento más sencillo que el explicado en el problema anterior sería aplicar un OR exclusivo a keyOld y keyNew y transmitir ese valor. El receptor entonces toma el XOR del valor recibido y keyOld para producir keyNew. Como el atacante no conoce keyOld, no puede deducir el valor de keyNew. ¿Cuáles son las ventajas de usar el número aleatorio y la función hash unidireccional segura del Problema 8.7 en comparación con este enfoque?

P A R T E III

SEGURIDAD DE LOS SISTEMAS

CONTENIDO DE LA TERCERA PARTE

La tercera parte de este libro trata aspectos sobre la seguridad de los sistemas, que incluyen las amenazas de intrusos y virus y sus correspondientes contramedidas, así como el uso de cortafuegos y sistemas confiables.

GUÍA PARA LA TERCERA PARTE

CAPÍTULO 9. INTRUSOS

El Capítulo 9 abarca una variedad de amenazas de acceso a la información y a los servicios por parte de los *hackers*, que se benefician de las vulnerabilidades que presentan los sistemas de computadores conectados en red. El capítulo empieza con una descripción de los tipos de ataques que los usuarios no autorizados, o intrusos, pueden llevar a cabo, y analiza distintos enfoques para la prevención y la detección. También cubre el aspecto relacionado con la gestión de claves.

CAPÍTULO 10. SOFTWARE DAÑINO

En el Capítulo 10 se observan las amenazas de software a los sistemas, haciendo especial hincapié en los virus y los gusanos. El capítulo empieza con un estudio de distintos tipos de *software* dañino, tratando con más detalle la naturaleza de los virus y los gusanos. El resto del capítulo se dedica a las contramedidas correspondientes.

CAPÍTULO 11. CORTAFUEGOS

Un enfoque estándar para la protección de equipos locales frente a amenazas externas lo constituye el uso de cortafuegos. En el Capítulo 11 se presentan los principios del diseño de los cortafuegos y se observan las técnicas específicas. Este capítulo también cubre el aspecto relacionado de los sistemas confiables.

CAPÍTULO 9

Intrusos

9.1. Intrusos

Técnicas de intrusión

9.2. La detección de intrusos

Registros de auditoría

Detección estadística de anomalías

Detección de la intrusión basada en reglas

La falacia de la tasa base

Detección distribuida de la intrusión

Honeypots

Formato de intercambio de detección de intrusos

9.3. Gestión de contraseñas

Protección de contraseñas

Estrategias de selección de contraseñas

9.4. Bibliografía y sitios web recomendados

9.5. Términos clave, preguntas de repaso y problemas

Términos clave

Preguntas de repaso

Problemas

Apéndice 9A La falacia de la tasa base

Probabilidad condicional e independencia

El teorema de Bayes

La falacia de la tasa base demostrada

Estaban de acuerdo en que Graham debería hacer la prueba para Charles Mabledene. Consistía ni más ni menos en que Dragon debería conseguir el código de Stern. Esto sería posible si él tenía el «enchufe» en Utting que decía tener. Sólo la lealtad al Centro de Moscú lo evitaría. Si él conseguía la clave del código demostraría su lealtad a la Central londinense más allá de cualquier duda.

Hablar con extraños, RUTH RENDELL

Un problema de seguridad significativo para los sistemas en red es el acceso hostil, o al menos no autorizado, por parte de usuarios o *software*. La entrada ilegal de usuarios puede adoptar la forma de identificación no autorizada a una máquina o, en el caso de un usuario no autorizado, la adquisición de privilegios o la realización de acciones más allá de las que han sido autorizadas. La entrada ilegal de *software* puede tener la forma de un virus, un gusano o un caballo de Troya.

Todos estos ataques tienen relación con la seguridad en la red porque, efectivamente, la entrada al sistema puede lograrse por medio de una red. Sin embargo, estos ataques no están restringidos a los ataques realizados exclusivamente en la red. Un usuario con acceso a una terminal local puede intentar acceder sin necesidad de usar una red intermedia. Un virus o un caballo de Troya pueden introducirse en un sistema por medio de un disquete. Sólo el gusano es un fenómeno exclusivo de la red. Por lo tanto, la violación del sistema es un área en la que se solapa lo concerniente a la seguridad en la red y a la seguridad de los computadores.

Debido a que el objetivo de este libro es la seguridad en la red, no es nuestra intención hacer un análisis exhaustivo de los ataques o las contramedidas en la seguridad relacionadas con la entrada ilegal en el sistema, sino presentar una panorámica general de estos aspectos.

Este capítulo abarca el tema de los intrusos. Primero, examinaremos la naturaleza del ataque y luego nos fijaremos en las estrategias destinadas a la prevención y, en el caso en que éstas fracasen, a la detección. Seguidamente examinaremos el aspecto relacionado con la gestión de contraseñas.

9.1 INTRUSOS

Una de las dos amenazas a la seguridad más extendidas es el intruso (la otra es el virus), generalmente conocido como *hacker* o *cracker*. En un estudio importante sobre la intrusión, Anderson [ANDE80] identificó tres clases de intrusos:

- **Suplantador:** un individuo que no está autorizado a usar el computador y que penetra hasta los controles de acceso del sistema para obtener provecho de la cuenta de un usuario legítimo.
- **Usuario fraudulento:** un usuario legítimo que accede a datos, programas o recursos para los que el acceso no está autorizado, o que, estando autorizado para tal acceso, hace un uso fraudulento de sus privilegios.
- **Usuario clandestino:** un individuo que toma el control de supervisión del sistema y lo usa para evadir los controles de auditoría y de acceso o para suprimir información de auditoría.

Es probable que el suplantador sea un usuario externo; el usuario fraudulento normalmente es un usuario interno; y el usuario clandestino puede ser alguien de fuera o de dentro.

Los ataques de intrusos pueden ser desde benignos a graves. En el extremo benigno de la escala, hay mucha gente que simplemente desea explorar en las redes y ver lo que ocurre en ellas. En el extremo grave hay individuos que intentan leer información privilegiada, llevar a cabo modificaciones no autorizadas a los datos o interrumpir el sistema.

La amenaza del intruso es muy conocida, en concreto a raíz del famoso incidente «Wily Hacker» de 1986-1987, documentado por Cliff Stoll [STOL88,89]. En 1990 hubo medidas energéticas a escala nacional sobre los *hackers* informáticos ilícitos, con arrestos, cargos criminales, un dramático juicio público, varias declaraciones de culpabilidad y confiscaciones de inmensas cantidades de datos y de equipos informáticos [STER92]. Mucha gente creyó que el problema ya estaba bajo control.

Pero, en la realidad, el problema no se había conseguido controlar. Para citar un ejemplo, un grupo de Bell Labs [BELL92, BELL93] informó sobre frecuentes y persistentes ataques en su complejo informático a través de Internet durante un gran período de tiempo y desde una gran variedad de fuentes. En aquel momento, el grupo Bell estaba experimentando lo siguiente:

- Intentos de copiar el archivo de contraseñas (que se discutirá más tarde) una vez en días alternos.
- Solicitudes sospechosas, una vez por semana, de llamadas a procedimiento remoto (RPC, *Remote Procedure Call*).
- Intentos de conectarse a máquinas «anzuelo» inexistentes al menos cada dos semanas.

Los intrusos benignos pueden ser tolerables, aunque consumen recursos y pueden ralentizar la ejecución a los usuarios legítimos. Sin embargo, no hay forma de saber con antelación si un intruso será benigno o maligno. Como consecuencia, incluso para los sistemas que no tengan recursos especialmente confidenciales, hay motivos que apuntan a la necesidad de controlar este problema.

Un ejemplo que ilustra dramáticamente la amenaza ocurrió en la A&M University de Texas [SAFF93]. En agosto de 1992, se notificó al centro informático que una de sus máquinas estaba siendo utilizada para atacar computadores de otro lugar por medio de Internet. Mediante la supervisión de la actividad, el personal de la central informática descubrió que había varios intrusos externos implicados, que estaban ejecutando rutinas para la violación de contraseñas en varios computadores (el sitio contiene un total de 12.000 máquinas conectadas entre sí). El centro desconectó las máquinas afectadas, conectó los agujeros de seguridad conocidos y recuperó el funcionamiento normal. Unos días más tarde, uno de los administradores del sistema local detectó que el ataque del intruso había vuelto a producirse. Resultó que el ataque era mucho más sofisticado de lo que se había creído en un principio. Se encontraron archivos que contenían clientes de contraseñas capturadas, incluidas algunas de los servidores principales que, supuestamente, eran más seguros. Además, una máquina local había sido utilizada como tablón de anuncios que los *hackers* usaban para contactar entre ellos y discutir las técnicas y los progresos.

Un análisis de este ataque reveló que, en realidad, había dos niveles de *hackers*. El nivel superior lo componían usuarios sofisticados con un amplio conocimiento de la tec-

nología; el nivel inferior estaba constituido por los «soldados de a pie», que simplemente usaban los programas fraudulentos suministrados con poca comprensión sobre su funcionamiento. Este equipo de trabajo combinaba las dos armas más importantes del arsenal del intruso: primero, un conocimiento sofisticado de cómo introducirse y, segundo, su predisposición para emplear incontables horas «abriendo puertas» con el fin de descubrir debilidades.

Uno de los resultados de la creciente concienciación del problema del intrusismo ha sido el establecimiento de equipos de respuesta de emergencias informáticas (CERT, *Computer Emergency Response Teams*). Estas empresas recogen información sobre las vulnerabilidades de los sistemas y las transmiten a los administradores de sistemas. Desgraciadamente, los *hackers* también pueden tener acceso a los informes de los CERT. En el incidente de la A&M de Tejas, los análisis posteriores mostraron que los *hackers* habían desarrollado programas para comprobar cada una de las vulnerabilidades que habían sido anunciadas por los CERT en las máquinas atacadas. Incluso si una máquina no respondía rápidamente al consejo de CERT, estaba gravemente expuesta a tales ataques.

Además de llevar a cabo programas para descubrir las contraseñas, los intrusos intentaron modificar el *software* de inicio que les permitía capturar las contraseñas de los usuarios conectados a los sistemas. Esto les facilitaba la elaboración de una impresionante relación de contraseñas comprometidas, que se encontraba disponible en el tablón de anuncios ubicado en el propio computador de una de las víctimas.

En esta sección nos centraremos en las técnicas usadas para la intrusión. Luego examinaremos las formas de detectar la intrusión y, por último, nos fijaremos en los enfoques basados en contraseñas para la prevención de la intrusión.

TÉCNICAS DE INTRUSIÓN

El objetivo del intruso es obtener acceso a un sistema o aumentar el rango de privilegios accesibles en el sistema. Generalmente, esto requiere que el intruso adquiera información que debería estar protegida. En la mayoría de los casos, esta información se encuentra en forma de contraseña de usuario. Conociendo la contraseña de algún otro usuario, un intruso puede entrar en un sistema y ejercitar todos los privilegios conferidos al usuario legítimo.

Normalmente, un sistema debe mantener un archivo que asocie una contraseña con cada usuario autorizado. Si este archivo se almacena sin protección, es tarea fácil acceder a él y descubrir las contraseñas. El archivo de contraseñas puede ser protegido de una de estas dos formas:

- **Cifrado unidireccional:** el sistema almacena sólo una forma cifrada de la contraseña del usuario. Cuando el usuario presenta una contraseña, el sistema la cifra y la compara con el valor almacenado. En la práctica, el sistema suele realizar una transformación en una sola dirección (no reversible) en la que la contraseña se usa para generar una clave para la función de cifrado y en la que se produce una salida de longitud fija.
- **Control de acceso:** el acceso al archivo de contraseñas está limitado a una o muy pocas cuentas.

Si se adoptan una o las dos contramedidas mencionadas, se requiere cierto esfuerzo por parte del intruso potencial para averiguar las contraseñas. Basándose en un estudio de la

literatura al respecto y entrevistas a un número de *crackers* de contraseñas, [ALVA90] presenta las siguientes técnicas para descubrir contraseñas:

1. Probar las contraseñas predeterminadas que se usan con las cuentas estándar suministradas con el sistema. Muchos administradores no se molestan en realizar estos cambios.
2. Probar de forma exhaustiva todas las contraseñas cortas (las de uno a tres caracteres).
3. Probar las palabras del diccionario en línea del sistema o una lista de posibles contraseñas, que se pueden encontrar en tablones de anuncios de *hackers*.
4. Recoger información sobre los usuarios como, por ejemplo, los nombres completos, el nombre de su cónyuge e hijos, cuadros en su oficina y libros en la misma relacionados con aficiones.
5. Probar los números de teléfono de los usuarios, los números de la Seguridad Social y los números de los despachos.
6. Probar con todas las matrículas de los coches del Estado.
7. Usar un caballo de Troya (descrito en la sección 9.2) para evitar las restricciones de acceso.
8. Interceptar la línea entre un usuario remoto y el sistema *host*.

Los primeros seis métodos constituyen distintas formas de averiguar una contraseña. Si un intruso tiene que verificar el descubrimiento intentando entrar, es un medio de ataque tedioso y fácilmente contrarrestado. Por ejemplo, un sistema puede simplemente rechazar cualquier conexión después de tres intentos de introducción de contraseña, obligando al intruso a conectarse de nuevo al *host*. En estas circunstancias, no es práctico intentar más de unas pocas contraseñas. Sin embargo, es poco probable que el intruso intente métodos tan burdos. Por ejemplo, si un intruso puede acceder con un bajo nivel de privilegios a un archivo de contraseñas cifradas, la estrategia consistiría en capturar ese archivo y luego usar, cuando quiera, el mecanismo de cifrado de ese sistema concreto hasta encontrar una contraseña válida que proporcione mayores privilegios.

Los ataques de descubrimiento de contraseñas son factibles y, de hecho, muy efectivos cuando se puede intentar automáticamente un gran número de averiguaciones y cada descubrimiento se puede comprobar, sin que el proceso de verificación sea detectado. Más adelante en este capítulo, se tratarán las formas de frustrar estos ataques.

El séptimo método de ataque enumerado arriba, el caballo de Troya, puede ser especialmente difícil de contrarrestar. En [ALVA90] se cita un ejemplo de un programa que evita los controles de acceso. Un usuario de pocos privilegios produjo un programa de juegos e invitó al operador del sistema a usarlo en su tiempo libre. El programa era ciertamente un juego, pero también contenía código para copiar en el fichero del usuario el archivo de contraseñas, que no estaba cifrado, aunque sí protegido de accesos. Como el juego estaba funcionando en el modo de alto privilegio del operador, podía acceder al archivo de contraseñas.

El octavo ataque enumerado, interceptación de línea, es un asunto de seguridad física. Puede contrarrestarse con técnicas de cifrado de enlace, estudiadas en el Capítulo 2.

Volvemos ahora a la discusión sobre las dos contramedidas principales: detección y prevención. La detección tiene que ver con el descubrimiento de un ataque, ya sea antes o después de su realización. La prevención es un reto en la seguridad y una batalla ardua en todo momento. La dificultad se deriva del hecho de que el defensor debe intentar evitar todos los ataques posibles, mientras que el atacante es libre de intentar encontrar el eslabón más débil en la cadena de defensa y atacar en ese punto.

9.2 DETECCIÓN DE INTRUSOS

Inevitablemente, el mejor sistema de prevención contra la intrusión fallará. Una segunda línea de defensa del sistema es la detección de la intrusión, y ésta ha sido el centro de gran parte de la investigación de los últimos años. Este interés está motivado por una serie de consideraciones, que incluyen las siguientes:

1. Si se detecta una intrusión con bastante rapidez, el intruso puede ser identificado y expulsado del sistema antes de producir daños o comprometer datos. Incluso si la detección no se realiza con suficiente brevedad para evitar la intrusión, cuanto antes se detecte, menor será la gravedad de los daños y más rápidamente se podrá lograr la recuperación.
2. Un sistema efectivo de detección de la intrusión puede servir como elemento disuasivo, actuando para prevenir intrusiones.
3. La detección de intrusos facilita la recopilación de información sobre técnicas de intrusión, que se puede usar para reforzar la prevención de la intrusión.

La detección de la intrusión se basa en la idea de que el comportamiento del intruso difiere del comportamiento del usuario legítimo de manera cuantificable. Es evidente que no se puede esperar que haya una distinción exacta entre un ataque realizado por un intruso y el uso normal de las fuentes por parte de un usuario autorizado. Más bien, debemos esperar que haya un solape.

La Figura 9.1 sugiere, en términos muy abstractos, la naturaleza de la misión a la que se enfrenta el diseñador de un sistema de detección de intrusos. Aunque el comportamiento típico de un intruso difiere del de un usuario autorizado, hay un solape en estos comportamientos. Así, una interpretación vaga del comportamiento del intruso, que detectará a más intrusos, también llevará a un número de «falsos positivos», o usuarios autorizados identificados como intrusos. Por otro lado, un intento de limitar los falsos positivos mediante una interpretación muy estricta del comportamiento del intruso conducirá a un incremento de los falsos negativos, o intrusos no identificados como tales. Por lo tanto, hay un elemento de compromiso y arte en la práctica de la detección de la intrusión.

En el estudio de Anderson [ANDE80], se postulaba que se podía, con un grado de seguridad razonable, distinguir entre un suplantador y un usuario legítimo. Los patrones de comportamiento del usuario legítimo pueden establecerse observando la historia pasada, y se puede detectar una significativa desviación de tales modelos. Anderson sugiere que la tarea de detectar a un usuario fraudulento (usuario legítimo que actúa de forma no autorizada) es más difícil, ya que la diferencia entre un comportamiento normal y uno anormal puede ser pequeña. Anderson concluyó que tales irrupciones no serían

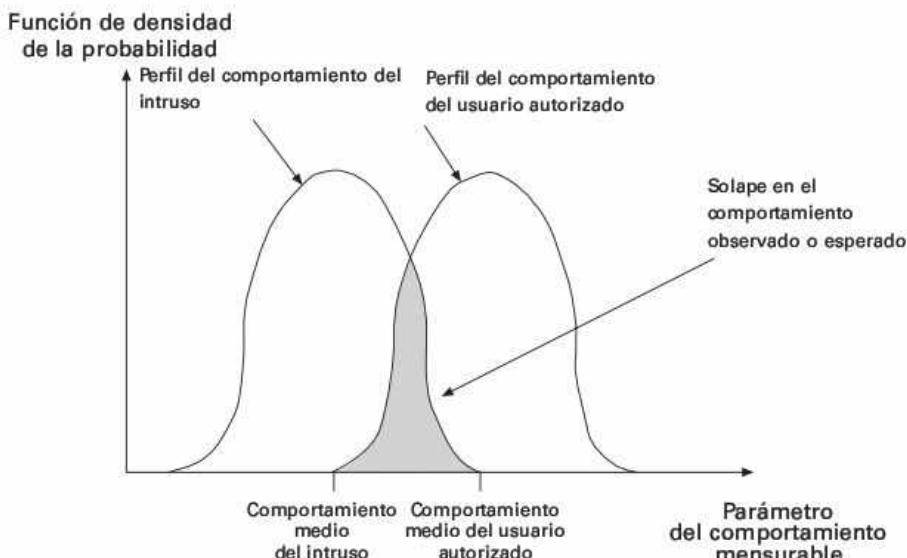


Figura 9.1 Perfiles de comportamiento de intrusos y usuarios autorizados

detectables solamente a través de la búsqueda de un comportamiento anómalo. Por el contrario, el comportamiento de un usuario fraudulento podría ser detectable mediante una definición inteligente de la clase de condiciones que sugieren el uso no autorizado. Finalmente, la detección del usuario clandestino se percibía más allá del ámbito de las técnicas puramente automatizadas. Estas observaciones, que fueron hechas en 1980, permanecen válidas hoy.

[PORR92] identifica los siguientes enfoques para la detección de la intrusión:

1. **Detección estadística de anomalías:** implica la recopilación de datos relacionados con el comportamiento de los usuarios legítimos en un período de tiempo. Las pruebas estadísticas se aplican a comportamientos observados para determinar con un alto grado de fiabilidad si ese comportamiento no es el de un usuario legítimo.
 - a) Detección de umbrales: este enfoque trae consigo definir los umbrales, independientes del usuario, para la frecuencia en que se producen distintos acontecimientos.
 - b) Basado en perfiles: se desarrolla un perfil de la actividad de cada usuario y se utiliza para detectar cambios en el comportamiento de cuentas individuales.
2. **Detección basada en reglas:** implica un intento de definir un conjunto de reglas que se pueden usar para determinar si un comportamiento dado es el de un intruso.
 - a) Detección de anomalías: las reglas se desarrollan para detectar una desviación de los modelos de uso previos.
 - b) Identificación de la penetración: un enfoque basado en sistemas expertos que busca comportamientos sospechosos.

Es decir, los enfoques estadísticos intentan definir el comportamiento normal, o esperado, mientras las propuestas basadas en reglas intentan definir el comportamiento correcto.

Atendiendo a los tipos de atacantes enumerados anteriormente, la detección estadística de anomalías es efectiva contra suplantadores, que probablemente no imitarán los modelos de comportamiento de las cuentas de las que se apropián. Por otro lado, es posible que dicha técnica no pueda tratar con usuarios fraudulentos. Para tales ataques, los enfoques basados en reglas podrían reconocer los acontecimientos y las secuencias que, en el contexto, revelan penetración. En la práctica, un sistema puede presentar una combinación de ambos enfoques para ser efectivo ante una amplia gama de ataques.

REGISTROS DE AUDITORÍA

Una herramienta fundamental para la detección de la intrusión es el registro de auditoría. Para un sistema de detección de intrusos es imprescindible que se registren las actividades que realizan los usuarios. Básicamente, se usan dos planes:

- **Registros nativos de auditoría:** prácticamente todos los sistemas operativos multiusuario incluyen *software* que recoge información sobre la actividad de los usuarios (*accounting*). La ventaja de usar esta información es que no se necesita un *software* de recopilación adicional. La desventaja es que los informes de auditoría nativos pueden no contener la información necesaria o no contenerla de una forma conveniente.
- **Registros de auditoría específicos para la detección:** se puede implementar una herramienta de recopilación que genere informes de auditoría que contengan sólo la información requerida por el sistema de detección de intrusión. Una ventaja de tal enfoque es que se podría hacer independiente del vendedor y trasladable a una variedad de sistemas. La desventaja son los costes adicionales que trae consigo tener, en efecto, dos paquetes ejecutándose en una máquina.

Un buen ejemplo de registros de auditoría específicos para la detección es el desarrollado por Dorothy Denning [DENN87]. Cada registro de auditoría contiene los siguientes campos:

- **Sujeto:** iniciador de acciones. Normalmente, un sujeto es un usuario de una terminal pero podría ser también un proceso que actúa en nombre de los usuarios o grupos de usuarios. Toda actividad surge a través de los comandos emitidos por los sujetos. Éstos se pueden agrupar en diferentes clases de acceso, y estas clases pueden solaparse.
- **Acción:** operación llevada a cabo por el sujeto sobre o con un objeto; por ejemplo, acceso al sistema (*login*), leer, realizar I/O, ejecutar.
- **Objeto:** receptor de acciones. Los ejemplos incluyen archivos, programas, mensajes, registros, terminales, impresoras y estructuras creadas por usuarios o programas. Cuando un sujeto es el receptor de una acción, como el correo electrónico, entonces ese sujeto se considera un objeto. Los objetos pueden agruparse por tipos. La granularidad del objeto puede variar en función del tipo de objeto y del entorno. Por ejemplo, las acciones de una base de datos pueden auditarse para la base de datos en su globalidad o a nivel de registro.

- **Condición de excepción:** denota, si la hubiera, la condición de excepción.
- **Uso de recursos:** una lista de elementos cuantitativos en la que cada elemento da la cantidad utilizada de algún recurso (por ejemplo, número de líneas impresas o mostradas, número de registros leídos o escritos, tiempo del procesador, unidades I/O usadas, tiempo de sesión transcurrido).
- **Sello de tiempo:** sello de tiempo y fecha única que identifica cuándo tuvo lugar la acción.

La mayor parte de las operaciones de usuarios se compone de un número de acciones elementales. Por ejemplo, la copia de un archivo implica la ejecución del comando de usuario, que incluye realizar la validación de acceso y la copia, más la lectura desde un archivo, más la escritura a otro archivo. Considérese el comando

COPY GAME.EXE TO <Library>GAME.EXE

emitido por Smith para copiar un archivo ejecutable GAME desde el directorio actual al directorio <Library>. Se pueden generar los siguientes registros de auditoría:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith>GAME.EXE	0	RECORDS=0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS=0	11058721680

En este caso, la copia se suspende porque Smith no tiene permiso de escritura en <Library>.

La descomposición de una operación de usuario en acciones elementales tiene tres ventajas:

1. Como los objetos son las entidades que se pueden proteger en el sistema, el uso de las acciones elementales permite una auditoría de todo el comportamiento que afecta a un objeto. Así, el sistema puede detectar intentos de subversión de los controles de acceso (percibiendo una anomalía en el número de condiciones de excepción devuelto) y puede detectar subversiones realizadas percibiendo una anomalía en el conjunto de objetos accesibles al sujeto.
2. Los registros de auditoría de un solo objeto y de una sola acción simplifican el modelo y la implementación.
3. Debido a la estructura simple y uniforme de los registros de auditoría específicos de detección, puede ser relativamente fácil obtener esta información, o al menos parte de ella, por medio de la correspondencia exhaustiva de los registros de auditoría nativos existentes con los registros de auditoría específicos para la detección.

DETECCIÓN ESTADÍSTICA DE ANOMALÍAS

Como ya se mencionó, las técnicas de detección estadística de anomalías forman parte de dos grandes categorías: la detección de umbrales y los sistemas basados en perfiles.

La detección de umbrales implica contar el número de incidencias de un tipo de evento específico en un intervalo de tiempo. La intrusión se asume si el cálculo sobrepasa lo que se considera un número razonable que se podría esperar que ocurriera.

El análisis de los umbrales, por sí mismo, es un detector ordinario y poco efectivo de ataques incluso moderadamente sofisticados. Se debe determinar tanto el límite como el intervalo de tiempo. Por la diversidad de los usuarios, es probable que dichos umbrales generen un gran número de falsos positivos o de falsos negativos. Sin embargo, los detectores simples de los umbrales pueden ser útiles en conjunción con técnicas más sofisticadas.

La detección de anomalías basada en perfiles se centra en caracterizar el comportamiento pasado de usuarios individuales o grupos relacionados de usuarios y luego detectar desviaciones significativas. Un perfil puede consistir en un conjunto de parámetros, para que la desviación de un solo parámetro no pueda ser suficiente en sí misma para indicar alerta.

El fundamento de este enfoque es un análisis de los registros de auditoría. Éstos constituyen una aportación a la función de detección de la intrusión de dos formas. Primero, el diseñador debe decidir sobre un número de métricas cuantitativas que puedan usarse para medir el comportamiento del usuario. Se puede emplear un análisis de los registros de auditoría en un período de tiempo para determinar el perfil de actividad del usuario medio. Así, los registros de auditoría sirven para definir el comportamiento típico. En segundo lugar, los registros actuales de auditoría son la entrada que se usa para detectar la intrusión. Es decir, el modelo de detección de intrusión analiza los registros de auditoría entrantes para determinar la desviación del comportamiento medio.

Los siguientes son ejemplos de métricas útiles para la detección de la intrusión basada en perfiles:

- **Contador:** número entero no negativo que puede ser incrementado pero no disminuido hasta que sea inicializado por una acción de la administración. Normalmente, se guarda una cuenta de ciertos tipos de acontecimientos durante un período de tiempo concreto. Algunos ejemplos incluyen el número de accesos realizados por un solo usuario durante una hora, el número de veces que se ejecuta un comando dado durante una sola sesión de usuario y el número de fallos en la contraseña durante un minuto.
- **Calibre:** número entero no negativo que puede ser incrementado o disminuido. Normalmente, un calibre se usa para medir el valor actual de alguna entidad. Algunos ejemplos incluyen el número de conexiones lógicas asignadas a una aplicación de usuario y el número de mensajes salientes en espera para un proceso de usuario.
- **Intervalo de tiempo:** período de tiempo entre dos acontecimientos relacionados. Un ejemplo es el espacio de tiempo entre entradas sucesivas a una cuenta.
- **Utilización de recursos:** la cantidad de recursos consumidos durante un período específico. Algunos ejemplos incluyen el número de páginas impresas durante una sesión de usuario y el tiempo total consumido por la ejecución de un programa.

Dadas estas métricas generales, se pueden llevar a cabo distintas pruebas para determinar si la actividad actual encaja dentro de límites aceptables. [DENN87] presenta los siguientes enfoques que pueden adoptarse:

- Media y desviación estándar

- Multivariable
- Procesos de Markov
- Serie temporal
- Operativo

La prueba estadística más sencilla es medir **la media y la desviación estándar** de un parámetro en un período histórico. Esto da una idea del comportamiento medio y su variabilidad. El uso de la media y la desviación estándar es aplicable a una gran variedad de contadores, temporizadores y medidas de recursos. Pero estas medidas, por sí mismas, son en general demasiado toscas para las intenciones de detección de la intrusión.

Un modelo **multivariable** se basa en las correlaciones entre dos o más variables. El comportamiento del intruso puede caracterizarse con mayor exactitud teniendo en cuenta dichas correlaciones (por ejemplo, el tiempo del procesador y el uso de los recursos, o la frecuencia de entrada y el tiempo transcurrido en la sesión).

Un **modelo de procesos de Markov** se usa para establecer las probabilidades de transición entre varios estados. Como ejemplo, este modelo podría usarse para observar las transiciones entre ciertos comandos.

Un modelo **de series temporales** se centra en los intervalos temporales, buscando secuencias de acontecimientos que suceden muy rápidamente o muy lentamente. Se puede aplicar una variedad de pruebas estadísticas para caracterizar tiempos anormales.

Por último, un modelo **operativo** se basa en un juicio de lo que se considera anormal, más que en un análisis automatizado de los registros de auditoría pasados. Generalmente, se definen límites fijos y la intrusión se sospecha para una observación que se halla fuera de los límites. Este tipo de enfoque funciona mejor donde el comportamiento del intruso se pueda deducir de ciertos tipos de actividades. Por ejemplo, un gran número de intentos de acceso a un sistema en un período corto sugiere un intento de intrusión.

Como ejemplo del uso de las distintas métricas y modelos, la Tabla 9.1 muestra las diferentes medidas consideradas o comprobadas para el sistema de detección de intrusión (IDES, *Intrusion Detection System*) del Stanford Research Institute (SRI) [DENN87, JAVI91, LUNT88].

La principal ventaja del uso de perfiles estadísticos es que no requiere un conocimiento previo de los fallos de seguridad. El programa detector aprende lo que es comportamiento «normal» y luego busca las desviaciones. El enfoque no se basa en características y vulnerabilidades dependientes del sistema. Por este motivo, debería ser fácilmente transportable entre una variedad de sistemas.

DETECCIÓN DE LA INTRUSIÓN BASADA EN REGLAS

Las técnicas basadas en reglas detectan la intrusión observando los acontecimientos que tienen lugar en el sistema y aplicando un conjunto de reglas que conduzcan a una decisión en lo que respecta a si un patrón dado de actividad es o no sospechoso. En términos muy generales, se puede afirmar que todos los enfoques se centran o bien en la detección de anomalías, o bien en la identificación de penetraciones, aunque hay un cierto solape entre ellos.

Tabla 9.1 Medidas para la detección de la intrusión

Medida	Modelo	Tipo de intrusión detectada
Actividad de acceso y de sesión		
Frecuencia de entrada por día y hora	Media y desviación estándar	Es probable que los intrusos entren fuera de las horas punta
Frecuencia de entrada en diferentes lugares	Media y desviación estándar	Los intrusos pueden acceder desde un lugar que un usuario concreto apenas o nunca use
Tiempo desde la última entrada	Operativo	Irrupción en una cuenta «muerta»
Tiempo transcurrido por sesión	Media y desviación estándar	Desviaciones significativas podrían indicar un suplantador
Cantidad de salida a la localización	Media y desviación estándar	Excesivas cantidades de datos transmitidos a localizaciones remotas podrían significar fuga de datos confidenciales
Utilización de recursos de sesión	Media y desviación estándar	Procesador inusual o niveles I/O podrían indicar intrusión
Fallos en las contraseñas al entrar	Operativo	Intento de irrupción por averiguación de contraseña
Fallos en el acceso desde terminales especificadas	Operativo	Intento de irrupción
Actividad de ejecución de comandos o programas		
Frecuencia de ejecución	Media y desviación estándar	Puede detectar intrusos, que probablemente usen diferentes comandos, o una introducción segura por parte de un usuario legítimo, que ha obtenido acceso a comandos privilegiados
Utilización de recursos de programa	Media y desviación estándar	Un valor anormal podría sugerir la inyección de un virus o un troyano, lo cual representa efectos indirectos que aumentan la utilización de I/O o del procesador

(continúa)

(continuación)

Medida	Modelo	Tipo de intrusión detectada
Actividad de ejecución de comandos o programas		
Negación de ejecución	Modelo operativo	Puede detectar intento de penetración por parte de un usuario individual que busque privilegios mayores
Actividad de acceso a archivos		
Frecuencia de lectura, escritura, de crear, de borrar	Media y desviación estándar	Anomalías para el acceso de creación, eliminación a leer y escribir para usuarios individuales puede significar suplantación u observación
Registros leídos, escritos	Media y desviación estándar	Una anomalía podría significar un intento de obtener datos confidenciales por inferencia o agregación
Cuenta de fallos para leer, escribir, crear, borrar	Operativo	Puede detectar usuarios que permanentemente intentan acceder a archivos no autorizados
Contador de agotamiento de recursos de archivo	Operativo	

La **detección de anomalías basada en reglas** es parecida a la detección estadística de anomalías en lo que respecta al enfoque y sus posibilidades. Con el enfoque basado en reglas, se analizan los registros de auditoría históricos para identificar los patrones de uso y generar automáticamente reglas que describan aquellos modelos. Las reglas pueden representar los modelos de comportamiento de usuarios en el pasado, programas, privilegios, fracciones de tiempo, terminales, etc. Entonces, se observa el comportamiento actual y se relaciona cada transacción con el conjunto de reglas para determinar si se ajusta a algún patrón de comportamiento observado anteriormente.

Al igual que con la detección estadística de anomalías, la detección de anomalías basada en reglas no requiere un conocimiento de las vulnerabilidades en la seguridad en el sistema. Más bien, el esquema se basa en la observación del comportamiento pasado y, en efecto, asumir que el futuro será como el pasado. Para que este enfoque sea efectivo, se necesitará una base de datos bastante extensa que contenga las reglas. Por ejemplo, un esquema descrito en [VACC89] contiene entre 10^4 y 10^8 reglas.

La **identificación de la penetración basada en reglas** adopta un enfoque distinto a la detección de la intrusión, basada en la tecnología de sistemas expertos. La característica clave de dichos sistemas se halla en el uso de reglas para identificar penetraciones conocidas o penetraciones que explotarían debilidades conocidas. También se pueden

definir reglas que identifiquen comportamientos sospechosos, incluso cuando el comportamiento está dentro de los límites de patrones de uso establecidos. Normalmente, las reglas que se usan en estos sistemas son específicas de la máquina y del sistema operativo. Además, estas reglas son generadas por «expertos» más que por medio de un análisis automatizado de registros de auditoría. El procedimiento normal es entrevistar a administradores de sistemas y a analistas de seguridad para recoger una serie de escenarios de penetración conocidos y acontecimientos fundamentales que amenacen la seguridad del sistema objetivo¹. Así, la fuerza de un enfoque depende de la destreza de los encargados de establecer las reglas.

Es posible encontrar un ejemplo sencillo del tipo de reglas que se pueden usar en NIDX, un sistema antiguo que usaba reglas heurísticas que pueden usarse para asignar grados de sospecha a actividades [BAUE88]. A continuación se exponen algunos ejemplos heurísticos:

- 1.** Los usuarios no deberían leer archivos en los directorios personales de otros usuarios.
- 2.** Los usuarios no deben escribir archivos de otros usuarios.
- 3.** Los usuarios que entran con frecuencia después de hora acceden a los archivos que utilizaron con anterioridad.
- 4.** Los usuarios normalmente no abren los dispositivos del disco directamente sino que se basan en las utilidades de más alto nivel del sistema operativo.
- 5.** Los usuarios no deberían permanecer conectados más de una vez en el mismo sistema.
- 6.** Los usuarios no hacen copias de los programas del sistema.

El esquema de identificación de penetración usado en IDES es representativo de la estrategia seguida. Los registros de auditoría se examinan a medida que se generan y se contrastan con la base de reglas. Si se encuentra una coincidencia, entonces se incrementa la *tasa de sospecha* del usuario. Si coinciden suficientes reglas, entonces el ratio pasará un umbral que da como resultado el registro de una anomalía.

El enfoque IDES se basa en un estudio de los registros de auditoría. Un punto débil de este plan está en su falta de flexibilidad. Un escenario de penetración dado puede dar como resultado un número de posibles secuencias alternativas de registros de auditoría que varían entre sí ligeramente o de forma sutil. Puede ser difícil precisar todas estas variaciones en reglas explícitas. Otro método consiste en desarrollar un modelo de alto nivel independiente de registros de auditoría específicos. Un ejemplo de ello es un modelo de transición de estado conocido como USTAT [ILGU93]. USTAT abarca acciones generales en vez de las acciones específicas detalladas registradas por el mecanismo de auditoría de UNIX. USTAT se implementa en el sistema SunOS que proporciona registros de auditoría en 239 acontecimientos. De estos, un preprocesador solamente usa 28, y las correlaciona con 10 acciones generales (Tabla 9.2). Usando sólo estas acciones y los parámetros que se invocan con cada acción, se desarrolla un diagrama de transición de estado que caracteriza la actividad sospechosa.

¹ Estas entrevistas pueden incluso extenderse a *crackers* reformados o no reformados que compartirán sus conocimientos a cambio de un pago [FREE93].

Tabla 9.2 Acciones USTAT frente a tipos de acontecimientos SunOS

Acción USTAT	Tipo de acontecimiento SunOS
Read	open_r, open_rc, open_RTC, open_rwc, open_rwTC, open_rt, open_rw, open_rwt
Write	truncate, ftruncate, creat, open_RTC, open_rwc, open_rwTC, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct
Create	mkdir, creat, open_rc, open_RTC, open_rwc, open_rwTC, open_wc, open_wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify_Owner	chown, fchown
Modify_Perm	chmod, fchmod
Rename	rename
Hardlink	link

Debido a que una serie de acontecimientos auditables distintos encajan en un número menor de acciones, el proceso de creación de reglas es más sencillo. Además, el modelo de diagrama de transición de estado se puede modificar fácilmente para ajustarlo nuevamente a los comportamientos de intrusión aprendidos.

LA FALACIA DE LA TASA BASE

Para ser práctico, un sistema de detección de intrusión debería detectar un porcentaje sustancial de intrusiones mientras mantiene la tasa de alarma falsa a un nivel aceptable. Si sólo se detecta un porcentaje modesto de las intrusiones reales, el sistema ofrece un falso sentido de seguridad. Por otra parte, si el sistema dispara con frecuencia una alerta cuando no hay intrusión (falsa alarma), entonces o bien los administradores del sistema comienzan a ignorar las alarmas, o se perderá mucho tiempo analizándolas.

Por desgracia, por la naturaleza de las probabilidades implicadas, es muy difícil satisfacer el estándar de tasa alta de detecciones con una tasa baja de falsas alarmas. En general, si el número real de intrusiones es bajo comparado con el número de usos legítimos de un sistema, entonces la tasa de falsas alarmas será alta a menos que la prueba sea extremadamente discriminatoria. Un estudio de los sistemas de detección de intrusión existentes, publicado en [AXEL00], indicaba que los sistemas actuales no han superado el problema de la falacia de la tasa base. El Apéndice 9A ofrece una breve base matemática para este problema.

DETECCIÓN DISTRIBUIDA DE LA INTRUSIÓN

Hasta hace poco, el trabajo en los sistemas de detección de intrusos se centraba en sistemas individuales independientes. La organización más común, sin embargo, necesita defender un conjunto distribuido de *hosts* en una LAN o en redes conectadas entre sí. Aunque es posible aumentar la defensa usando sistemas de detección de intrusos aislados en cada *host*, se puede lograr una defensa más efectiva mediante la coordinación y la cooperación entre los sistemas de detección de intrusión de toda la red.

Porras señala importantes resultados en el diseño de un sistema distribuido de detección de intrusos [PORR92]:

- Un sistema distribuido de detección de intrusión puede necesitar tratar con diferentes formatos de registro de auditorías. En un entorno heterogéneo, diferentes sistemas emplearán distintos sistemas nativos de recopilación de información de auditoría y, si usa la detección de intrusión, puede emplear diferentes formatos para los registros de auditoría relacionados con la seguridad.
- Uno o más nodos de la red servirán como puntos de recopilación y análisis de los datos de los sistemas en la red. Así, se deben transmitir por la red datos simples de auditoría o datos de resumen. Por lo tanto, hay un requisito para asegurar la integridad y la confidencialidad de estos datos. La integridad se requiere para evitar que un intruso enmascare sus actividades alterando la información de auditoría transmitida. La confidencialidad se requiere porque la información de auditoría transmitida podría ser valiosa.
- Se podría usar tanto una arquitectura centralizada como descentralizada. Con una arquitectura centralizada hay un solo punto central de recopilación y análisis de todos los datos de auditoría. Esto facilita la tarea de correlacionar los registros entrantes pero crea un embottellamiento potencial y un solo punto de fisura. Con una arquitectura descentralizada, hay varios centros de análisis, pero éstos deben coordinar sus actividades e intercambiar información.

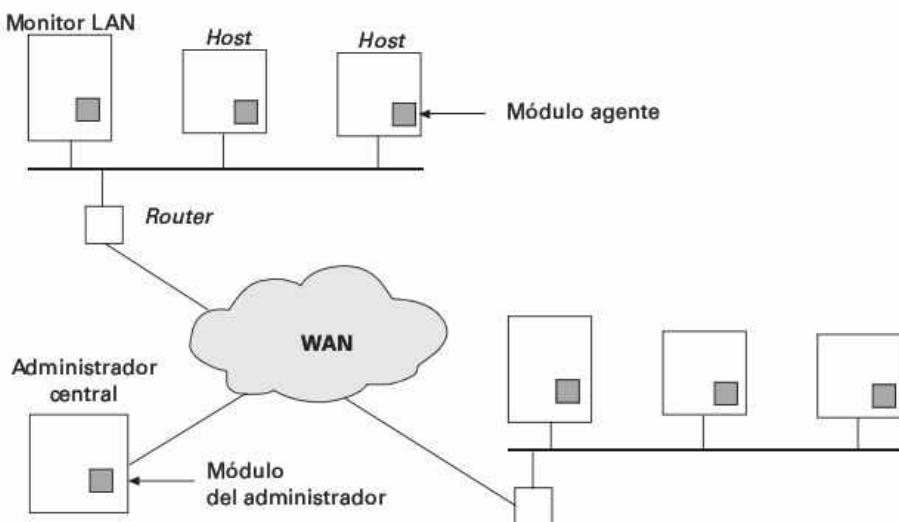


Figura 9.2 Arquitectura para la detección distribuida de intrusos

Un buen ejemplo de un sistema distribuido de detección de intrusos es el desarrollado en la Universidad de California en Davis [HEBE92, SNAP91]. La Figura 9.2 muestra la arquitectura general, que consiste en tres componentes principales:

- **Módulo agente del host**: módulo de recopilación de información de auditoría que opera como un proceso subordinado en un sistema monitorizado. Su intención es recoger datos sobre acontecimientos relacionados con la seguridad en el *host* y transmitirlos al administrador central.
- **Módulo agente monitor de LAN**: opera de la misma forma que un módulo agente de *host* excepto en que analiza el tráfico en la LAN e informa de los resultados al administrador central.
- **Módulo administrador central**: recibe los informes del monitor de la LAN y de los agentes de *host* y procesa y relaciona estos informes para detectar la intrusión.

El esquema está diseñado para ser independiente de cualquier sistema operativo o implementación de auditoría del sistema. La Figura 9.3 [SNAP91] muestra el enfoque general adoptado. El agente captura cada registro de auditoría producido por el sistema nativo de recopilación de auditoría. Se aplica un filtro que retiene sólo aquellos registros de interés para la seguridad. Luego, estos informes se convierten a un formato estandarizado conocido como registro de auditoría del *host* (HAR, *Host Audit Record*). A continuación, un módulo lógico basado en plantillas analiza los registros en busca de actividades sospechosas. En el nivel más bajo, el agente explora en busca de acontecimientos importantes de interés independientes de cualquier acontecimiento pasado. Algunos ejemplos incluyen accesos frustrados a ficheros, accesos a ficheros del sistema y cambios del control de acceso de un fichero. En el siguiente nivel superior, el agente busca secuencias de acontecimientos, tales como patrones conocidos de ataques (firmas). Por último, el agente busca comportamientos anómalos de un usuario individual basado en un perfil histórico de ese usuario como, por ejemplo, el número de programas ejecutados y el número de archivos a los que ha accedido, entre otros.

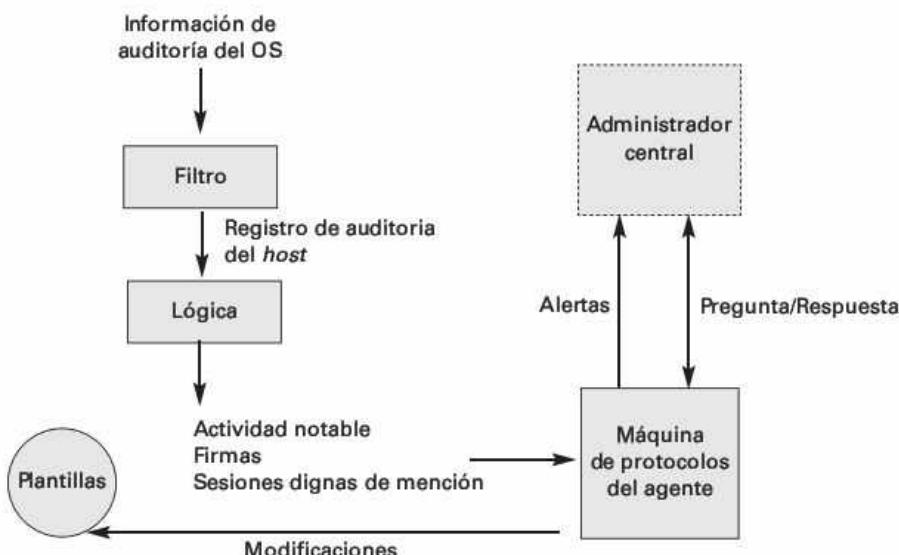


Figura 9.3 Arquitectura del agente

Cuando se detecta una actividad sospechosa, se envía una alerta al administrador central. El administrador central incluye un sistema experto que puede perfilar inferencias de los datos recibidos. El administrador puede también buscar copias de HAR en sistemas individuales para correlacionarlas con las de otros agentes.

El agente monitor de la LAN también suministra información al administrador central; audita las conexiones de *host a host*, los servicios usados y el volumen de tráfico. Busca acontecimientos significativos, tales como cambios repentinos en la carga de la red, el uso de servicios relacionados con la seguridad y actividades de la red como *rlogin*.

La arquitectura descrita en las Figuras 9.2 y 9.3 es bastante general y flexible. Ofrece una fundamentación para un enfoque independiente de la máquina que se puede ampliar desde la detección de la intrusión en computadores independientes a un sistema capaz de correlacionar la actividad de un número de sitios y redes para detectar actividades sospechosas que, de otra forma, pasarían desapercibidas.

HONEYPOTS

Una innovación relativamente reciente en la tecnología de la detección de intrusos es el *honeypot* («tarro de miel», literalmente en español). Los *honeypots* son sistemas de reclamo diseñados para alejar a un atacante potencial de los sistemas críticos. Están diseñados para

- desviar a un atacante del acceso a sistemas críticos
- recoger información sobre la actividad del atacante
- favorecer que el atacante permanezca en el sistema el tiempo suficiente para que los administradores puedan responder

Estos sistemas están llenos de información inventada, valiosa en apariencia, pero a la que un usuario legítimo del sistema no accedería. Así, cualquier acceso al *honeypot* es sospechoso. El sistema está equipado con monitores sensibles y registradores de acontecimientos que detectan estos accesos y recogen información sobre las actividades del atacante. Debido a que se hace que cualquier ataque contra el *honeypot* parezca realizado, los administradores tienen tiempo de movilizar, registrar y seguir la pista del atacante sin haber expuesto en ningún momento los sistemas productivos.

Los esfuerzos iniciales involucraban a un solo computador *honeypot* con direcciones IP diseñadas para atraer *hackers*. Las investigaciones más recientes se han centrado en la construcción de redes completas de *honeypots* que emulen una empresa, posiblemente con tráfico y datos reales o simulados. Una vez que los *hackers* están dentro de la red, los administradores pueden observar su comportamiento con detalle y perfilar las defensas.

FORMATO DE INTERCAMBIO DE DETECCIÓN DE INTRUSOS

Para facilitar el desarrollo de los sistemas distribuidos de detección de intrusos que pueden funcionar a través de un amplio abanico de plataformas y entornos, se necesitan estándares para permitir la interoperabilidad. Tales estándares son el eje del Grupo de Trabajo de Detección de Intrusos de la IETF. El propósito del grupo de trabajo es definir los

formatos de los datos e intercambiar los procedimientos para compartir información de interés para la detección de la intrusión y sistemas de respuesta, y para los sistemas de gestión que puedan necesitar interactuar con ellos. Los resultados de este grupo de trabajo incluyen los siguientes:

1. Un documento de requisitos que describe los requisitos funcionales de alto nivel para la comunicación entre sistemas de detección de intrusos, y los requisitos para la comunicación entre los sistemas de detección de intrusos y los sistemas de gestión, incluyendo los motivos que llevaron a esos requisitos.
2. Una especificación de lenguaje de intrusión común, que describe formatos de datos que satisfagan los requisitos.
3. Un documento marco, que identifica los protocolos existentes mejor usados para la comunicación entre los sistemas de detección de intrusos, y describe cómo los formatos de datos ideados se relacionan con ellos.

Todos estos documentos se encuentran en fase de borrador de Internet.

9.3 GESTIÓN DE CONTRASEÑAS

PROTECCIÓN DE CONTRASEÑAS

La línea frontal de defensa contra los intrusos es el sistema de contraseñas. Prácticamente todos los sistemas multiusuario requieren que un usuario proporcione no sólo un nombre o identificador (ID) sino también una contraseña. La contraseña sirve para autenticar el identificador del individuo que entra en el sistema. A su vez, el identificador provee seguridad de la siguiente manera:

- Determina si el usuario está autorizado para acceder al sistema. En algunos sistemas, sólo aquellos que ya tienen un identificador archivado en el sistema tienen permiso de acceso.
- Determina los privilegios otorgados al usuario. Unos pocos usuarios pueden tener estatus de supervisión o «superusuario», que les permite leer archivos y llevar a cabo funciones que están especialmente protegidas por el sistema operativo. Algunos sistemas tienen cuentas de invitados o anónimas, y algunos usuarios de estas cuentas tienen privilegios más limitados que otros.
- Se usa en lo que se conoce como control de acceso discrecional. Por ejemplo, mediante un listado de los identificadores de los otros usuarios, un usuario puede concederles permiso para leer archivos de su propiedad.

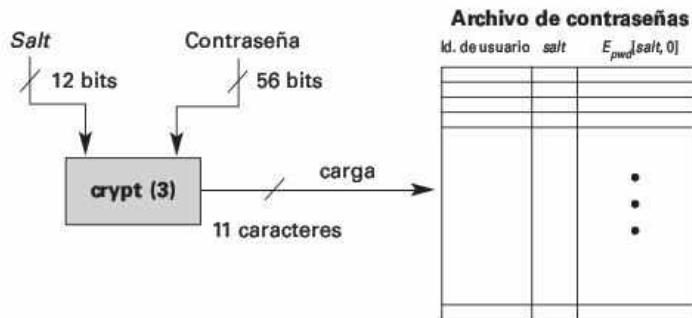
La vulnerabilidad de las contraseñas

Para comprender la naturaleza de las amenazas a los sistemas basados en contraseñas, vamos a considerar un esquema muy usado en UNIX, en el cual las contraseñas nunca se almacenan en claro. Se emplea el siguiente procedimiento (Figura 9.4a). Cada usuario selecciona una contraseña de hasta ocho caracteres imprimibles. Esto se convierte en

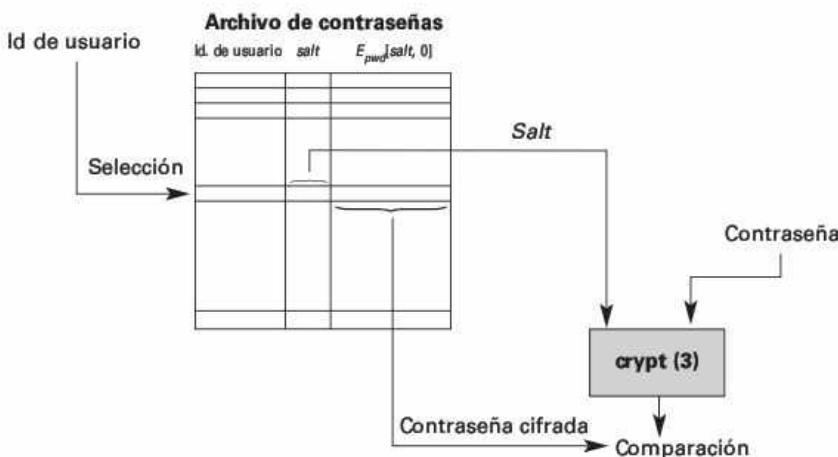
un valor de 56 bits (usando ASCII de siete bits) que sirve como la entrada de clave a una rutina de cifrado. La rutina de cifrado, conocida como `crypt(3)`, se basa en DES. El algoritmo DES se modifica usando un valor *salt* de 12 bits. Normalmente, este valor está relacionado con el tiempo en el cual una contraseña está asignada al usuario. El algoritmo DES modificado se hace con una entrada de datos formada por un bloque de ceros de 64 bits. La salida del algoritmo sirve entonces como entrada para un segundo cifrado. Este proceso se repite para un total de 25 cifrados. Luego, la salida resultante de 64 bits se traduce a una secuencia de 11 caracteres. A continuación, la contraseña de texto cifrado se almacena, junto con una copia en texto claro del valor *salt*, en el archivo de contraseñas para el identificador de usuario correspondiente.

El valor *salt* sirve con tres propósitos:

- Evita que las contraseñas duplicadas estén visibles en el archivo de contraseñas. Incluso si dos usuarios eligen la misma contraseña, esas contraseñas se asignarán en distintos momentos. Así, las contraseñas «extendidas» de los dos usuarios diferirán.



(a) Introducción de nueva contraseña en el fichero



(b) Verificación de contraseña

Figura 9.4 Esquema de contraseñas UNIX

- Incrementa de manera efectiva la longitud de la contraseña sin requerir al usuario que recuerde dos caracteres adicionales. De aquí, el número de posibles contraseñas se multiplica por un factor de 4096, aumentando la dificultad de averiguar una contraseña.
- Evita el uso de una implementación *hardware* de DES, que reduciría la dificultad de un ataque de contraseña por fuerza bruta.

Cuando un usuario intenta entrar en un sistema UNIX proporciona un identificador y una contraseña. El sistema operativo usa el identificador para indexar en el archivo de contraseñas y recuperar el valor *salt* en texto claro y la contraseña cifrada. El valor *salt* y la contraseña suministrada se usan como entrada a la rutina de cifrado. Si el resultado coincide con el valor almacenado, se acepta la contraseña.

La rutina de cifrado está diseñada para disuadir de realizar ataques de averiguación de contraseñas. Las implementaciones de *software* del DES son lentas comparadas con las versiones de *hardware*, y el uso de 25 iteraciones multiplica el tiempo requerido por 25. Sin embargo, desde el diseño original de este algoritmo, se han producido dos cambios. En primer lugar, las implementaciones más recientes del algoritmo han dado como resultado un aumento de la velocidad. Por ejemplo, el gusano de Internet descrito en el Capítulo 10 podía averiguar varios cientos de contraseñas en la red en un tiempo bastante corto usando un algoritmo de cifrado más eficiente que el estándar almacenado en los sistemas UNIX que atacaba. Segundo, el rendimiento del *hardware* continúa aumentando, de tal forma que cualquier algoritmo de *software* se ejecuta más rápidamente.

Así, hay dos amenazas al esquema de contraseñas UNIX. Por una parte, un usuario puede acceder a una máquina utilizando una cuenta de huéspedes, o por algún otro medio, y luego ejecutar en esa máquina un programa de averiguación de contraseñas, conocido como pirata de contraseñas. El atacante debería poder comprobar cientos y quizás miles de posibles contraseñas consumiendo pocos recursos. Además, si un oponente puede obtener una copia del archivo de contraseñas, entonces un programa pirata puede ejecutarse en otra máquina en cualquier momento. Esto hace posible que el oponente pase por miles de posibles contraseñas en un período razonable.

Tabla 9.3 Longitudes de contraseñas observadas [SPAF92a]

Longitud	Número	Fracción del total
1	55	.004
2	87	.006
3	212	.02
4	449	.03
5	1260	.09
6	3035	.22
7	2917	.21
8	5772	.42
Total	13787	1.0

En agosto de 1993, por ejemplo, se dio aviso de un pirata de contraseñas en Internet [MADS93]. Usando un computador paralelo de Thinking Machines Corporation, se al-

canzó un rendimiento de 1560 cifrados por segundo por unidad de vector. Con cuatro unidades de vector por nodo de procesamiento (una configuración estándar), esto funciona a 800.000 cifrados por segundo en una máquina de 128 nodos (que es un tamaño modesto) y 6,4 millones de cifrados por segundo en una máquina de 1024 nodos.

Incluso estos magníficos promedios de averiguación no hacen todavía posible que un atacante use la técnica burda de fuerza bruta de intentar todas las posibles combinaciones de caracteres para descubrir una contraseña. En su lugar, los piratas de contraseñas se basan en el hecho de que algunas personas eligen contraseñas fáciles de adivinar.

Algunos usuarios, cuando se les permite elegir sus contraseñas, escogen una absurdamente corta. En la Tabla 9.3 se muestran los resultados de un estudio de la Universidad de Purdue. El estudio observaba las elecciones de cambio de contraseña en 54 máquinas, que representaban aproximadamente 7000 cuentas de usuarios. Casi el 3 por ciento de las contraseñas tenían tres caracteres o menos. Un atacante podía comenzar un ataque probando exhaustivamente todas las contraseñas posibles de tres o menos caracteres. Una solución sencilla para el sistema es rechazar cualquier elección de contraseña de, por ejemplo, menos de seis caracteres o incluso requerir que todas las contraseñas tengan exactamente ocho caracteres. La mayoría de los usuarios no se quejarían de tal restricción.

La longitud de las contraseñas es sólo una parte del problema. Mucha gente, cuando tiene la posibilidad de elegir su propia contraseña, escoge una fácil de adivinar, como su propio nombre, el nombre de su calle, una palabra común del diccionario, etc. Esto hace que el trabajo de un pirata de contraseñas sea sencillo. El pirata simplemente tiene que contrastar el archivo de contraseñas con la lista de posibles contraseñas. Ya que mucha gente utiliza contraseñas adivinables, tal estrategia debería triunfar prácticamente en todos los sistemas.

En [KLEI90] se ofrece una demostración de la efectividad de la averiguación de contraseñas. Desde una variedad de fuentes, el autor recopiló archivos de contraseñas de UNIX, que contenían casi 14.000 contraseñas cifradas. El resultado, que el autor correctamente caracteriza como asombroso, se muestra en la Tabla 9.4. En todos, se averiguó casi el 25 por ciento de las contraseñas. Se usó la siguiente estrategia:

- 1.** Probar el nombre del usuario, las iniciales, el nombre de cuenta y otra información personal relevante. En todos, se intentaron 130 permutaciones diferentes para cada usuario.
- 2.** Intentar palabras de distintos diccionarios. El autor compiló un diccionario de más de 60.000 palabras, incluyendo el diccionario en línea del sistema y otras listas diferentes, como se puede observar.
- 3.** Realizar distintas permutaciones en las palabras del paso 2. Esto incluía hacer la primera letra mayúscula o un carácter de control, hacer la palabra completa mayúscula, invertir la palabra, cambiar la letra «o» por el dígito «cero», etc. Estas permutaciones sumaron otro millón de palabras a la lista.
- 4.** Probar distintas permutaciones de mayúsculas en las palabras del paso 2 que no se consideraron en el paso 3. Esto añadió casi dos millones de palabras a la lista.

De este modo, la prueba abarcó tres millones de palabras. Usando la implementación más rápida de Thinking Machines presentada anteriormente, el tiempo para cifrar todas estas palabras para todos los posibles valores *salt* está por debajo de una hora. Es conveniente recordar que una búsqueda tan exhaustiva podría producir un promedio de éxito de alrededor del 25 por ciento, mientras que un solo ataque puede ser suficiente para obtener una amplia gama de privilegios en el sistema.

Tabla 9.4 Contraseñas averiguadas de una muestra de 13.797 cuentas [KLE90]

Tipo de contraseña	Tamaño de la búsqueda	Número de coincidencias	Porcentaje de contraseñas coincidentes (%)	Ratio coste/beneficio*
Nombre de la cuenta o de usuario	130	368	2,7	2,830
Secuencias de caracteres	866	22	0,2	0,025
Números	427	9	0,1	0,021
Chino	392	56	0,4	0,143
Nombres de lugares	628	82	0,6	0,131
Nombres comunes	2239	548	4,0	0,245
Nombres femeninos	4280	161	1,2	0,038
Nombres masculinos	2866	140	1,0	0,049
Nombres poco comunes	4955	130	0,9	0,026
Mitos y leyendas	1246	66	0,5	0,053
Sobre Shakespeare	473	11	0,1	0,023
Deportes	238	32	0,2	0,134
Ciencia ficción	691	59	0,4	0,085
Películas y actores	99	12	0,1	0,121
Dibujos animados	92	9	0,1	0,098
Gente famosa	290	55	0,4	0,190
Frases y patrones	933	253	1,8	0,271
Apellidos	33	9	0,1	0,273
Biología	58	1	0,0	0,017
Diccionario	19683	1027	7,4	0,052
Nombres de máquinas	9018	132	1,0	0,015
Mnemotécnica	14	2	0,0	0,143
Biblia King James	7525	83	0,6	0,011
Palabras diversas	3212	54	0,4	0,017
Palabras Yiddish	56	0	0,0	0,000
Asteroides	2407	19	0,1	0,007
TOTAL	62727	3340	24,2	0,053

* Calculado como el número de coincidencias dividido por el tamaño de la búsqueda. Cuanto mayor sea el número de palabras que necesiten ser comprobadas para una correspondencia, menor será el ratio de coste/beneficio.

Control de acceso

Una forma de frustrar un ataque de contraseña es denegar al oponente el acceso al archivo de contraseñas. Si la parte de contraseña cifrada del archivo es accesible sólo para un usuario privilegiado, entonces el oponente no puede leerlo sin conocer previamente la contraseña de dicho usuario. [SPAF92a] señala varios defectos en esta estrategia:

- Muchos sistemas, incluyendo la mayoría de los sistemas UNIX, son susceptibles a entradas no previstas. Una vez que el atacante ha obtenido acceso por algún medio, puede desear obtener una relación de contraseñas para usar diferentes cuentas en diferentes sesiones de entrada y, así, disminuir el riesgo de detección. O un usuario con una cuenta puede desear la cuenta de otro usuario para acceder a datos privilegiados o para sabotear el sistema.

- Un accidente en la protección podría presentar el archivo de contraseñas legible, comprometiendo así todas las cuentas.
- Algunos de los usuarios tienen cuentas en otras máquinas en otros dominios de protección, y usan la misma contraseña. Así, si las contraseñas pudieran ser leídas por alguien en una máquina, otra máquina en otro lugar podría estar comprometida.

Por lo tanto, una estrategia más efectiva sería obligar a los usuarios a seleccionar contraseñas difíciles de adivinar.

ESTRATEGIAS DE SELECCIÓN DE CONTRASEÑAS

Lo que se extrae de los dos experimentos descritos (Tablas 9.3 y 9.4) es que, impulsados por sus propios mecanismos, muchos usuarios eligen contraseñas demasiado cortas o demasiado fáciles de averiguar. En el otro extremo, si a los usuarios se les asignan contraseñas de ocho caracteres imprimibles aleatorios, el descubrimiento de las contraseñas es efectivamente imposible. Pero sería casi igual de imposible que la mayoría de los usuarios recordaran sus contraseñas. Afortunadamente, incluso si limitamos el universo de las contraseñas a ristras de caracteres más o menos fáciles de memorizar, el tamaño del universo es aún demasiado extenso para permitir una violación práctica. Nuestro objetivo, pues, es eliminar las contraseñas adivinables al tiempo que se permite al usuario seleccionar una contraseña fácil de memorizar. Hay cuatro técnicas básicas:

- educación del usuario
- contraseñas generadas por computador
- comprobación reactiva de contraseñas
- comprobación proactiva de contraseñas

A los usuarios se les puede explicar la importancia de usar contraseñas difíciles de adivinar y se les puede proporcionar recomendaciones para la selección de contraseñas fuertes. Es poco probable que prospere la estrategia de la **educación del usuario** en la mayoría de las instalaciones, sobre todo donde haya una gran población de usuarios o mucha rotación de empleados. Muchos usuarios simplemente ignorarán las recomendaciones. Otros puede que no sean buenos jueces de lo que es una contraseña fuerte. Por ejemplo, muchos usuarios (erróneamente) creen que dar la vuelta a una palabra o poner la última letra en mayúscula hace que una contraseña sea difícil de adivinar.

Las **contraseñas generadas por computador** también plantean problemas. Si las contraseñas son bastante aleatorias, los usuarios no podrán recordarlas. Incluso si la contraseña es pronunciable, el usuario puede tener dificultad en recordarla y así estar tentado a escribirla en papel. En general, los esquemas de contraseñas generadas por computador han tenido poca aceptación por parte de los usuarios. FIPS PUB 181 define uno de los generadores de contraseñas automatizados mejor diseñados. El estándar incluye no sólo una descripción del enfoque sino también un listado completo del código C fuente del algoritmo. Éste genera palabras formando sílabas pronunciables y las encadena para formar una palabra. Un generador de números aleatorios produce un flujo aleatorio de caracteres que se usa para construir las sílabas y las palabras.

Una estrategia de **comprobación de contraseña reactiva** es aquella en la que el sistema ejecuta periódicamente su propio pirata de contraseñas para encontrar contraseñas adivinables. El sistema cancela cualquier contraseña averiguada y lo notifica al usuario. Esta táctica tiene una serie de inconvenientes. Primero, es un recurso intensivo si el trabajo se hace bien. Puesto que un determinado oponente que robe un archivo de contraseñas puede dedicar todo el tiempo de la CPU a la tarea durante horas o incluso días, un comprobador de contraseñas reactivo efectivo está en clara desventaja. Además, cualquier contraseña existente permanece vulnerable hasta que el comprobador de contraseñas reactivo la encuentra.

El avance más prometedor para la mejora de la seguridad de las contraseñas es el **comprobador de contraseñas proactivo**. En este esquema, a un usuario se le permite seleccionar su propia contraseña. Sin embargo, en el momento de la selección, el sistema comprueba si la contraseña está permitida y, si no, la rechaza. Tales comprobadores están basados en la filosofía de que, con suficiente guía desde el sistema, los usuarios pueden seleccionar contraseñas fáciles de memorizar, a partir de una gama de contraseñas bastante grande, y que son difíciles de adivinar en un ataque de diccionario.

La clave con un comprobador de contraseñas proactivo está en mantener un balance entre la aceptabilidad del usuario y la robustez. Si el sistema rechaza demasiadas contraseñas, los usuarios se quejarán de que es demasiado complicado seleccionar una contraseña. Si el sistema usa algún algoritmo sencillo para definir lo que es aceptable, se ofrece una guía a los piratas de contraseñas para perfeccionar sus técnicas de ataque. En el resto de este subapartado, se observarán posibles enfoques para la comprobación de contraseñas proactiva.

El primer enfoque consiste en un sistema sencillo para el endurecimiento de las reglas. Por ejemplo, se podrían reforzar las siguientes reglas:

- Todas las contraseñas deben tener al menos ocho caracteres.
- En los primeros ocho caracteres, las contraseñas deben incluir al menos una mayúscula, una minúscula, un dígito numérico y un signo de puntuación.

Estas reglas podrían combinarse con recomendaciones al usuario. Aunque este enfoque es superior al simple hecho de educar a los usuarios, puede no ser suficiente para evitar a los piratas de contraseñas. Este esquema alerta a los piratas sobre qué contraseñas *no* intentar, pero aún puede hacer posible la violación de contraseñas.

Otro posible procedimiento es simplemente compilar un gran diccionario de probables «malas» contraseñas. Cuando un usuario selecciona una contraseña, el sistema se asegura de que no está en la lista desaprobada. Sin embargo, con esta opción surgen dos problemas:

- **Espacio:** el diccionario debe ser muy amplio para ser efectivo. Por ejemplo, el diccionario usado en el estudio de Purdue [SPAF92a] ocupa más de 30 megabytes de capacidad.
- **Tiempo:** realizar búsquedas en un diccionario muy extenso puede requerir mucho tiempo. Además, para buscar posibles permutaciones de las palabras de los diccionarios, o bien esas palabras deben incluirse en el diccionario, haciéndolo verdaderamente extenso, o bien cada búsqueda debe también implicar una cantidad considerable de procesamiento.

Existen dos técnicas prometedoras para desarrollar un comprobador de contraseñas proactivo efectivo basado en el rechazo de las palabras de una lista. Una de ellas desarrolla un modelo de Markov para la generación de contraseñas adivinables [DAVI93]. La Figura 9.5 muestra una versión simplificada de dicho modelo. Este modelo muestra un lenguaje formado por un alfabeto de tres caracteres. El estado del sistema en cualquier momento es la identidad de la letra más reciente. El valor de la transición de un estado a otro representa la probabilidad de que una letra siga a otra. Así, la probabilidad de que la siguiente letra sea b , dado que la letra actual es a , es de 0.5.

En general, un modelo Markov es un cuádruplo $[m, A, \mathbf{T}, k]$, donde m es el número de estados en el modelo, A es el espacio del estado, \mathbf{T} es la matriz de probabilidades de transición, y k es el orden del modelo. Para un modelo de orden k , la probabilidad de hacer una transición a una letra particular depende de las k letras previas que han sido generadas. La Figura 9.5 muestra un modelo sencillo de primer orden.

Los autores describen el desarrollo y el uso de un modelo de segundo orden. Para comenzar, se construye un diccionario de contraseñas adivinables. Luego se calcula la matriz de transición como sigue:

1. Determinar la matriz de frecuencia \mathbf{f} , donde $\mathbf{f}(i, j, k)$ es el número de ocurrencias de la cadena compuesta por los i -ésimo, j -ésimo y k -ésimo caracteres. Por ejemplo, la contraseña *parsnips* produce los trigramas par, ars, rsn, sin, nip e ips.
2. Para cada cadena ij , calcular $\mathbf{f}(i, j, \infty)$ como el número total de trigramas que comienzan por ij . Por ejemplo, $\mathbf{f}(a, b, \infty)$ sería el número total de trigramas de la forma aba, abb, abc, etc.
3. Calcular las entradas de \mathbf{T} como sigue:

$$\mathbf{T}(i, j, k) = \frac{\mathbf{f}(i, j, k)}{\mathbf{f}(i, j, \infty)}$$

El resultado es un modelo que refleja la estructura de las palabras en el diccionario. Con este modelo, la pregunta «¿Es ésta una mala contraseña?» se transforma en «¿Fue esta ristra (contraseña) generada por este modelo de Markov?». Para una contraseña dada, se pueden buscar las probabilidades de transición de todos sus trigramas. Luego se pueden usar algunas pruebas estadísticas estándar para determinar si la contraseña es probable o improbable para ese modelo. Las contraseñas probables de ser generadas por ese modelo son rechazadas. Los autores ofrecen buenos resultados para un modelo de segundo orden. Su sistema atrapa casi todas las contraseñas de su diccionario y no excluye tantas contraseñas potencialmente buenas como para presentar dificultades a los usuarios.

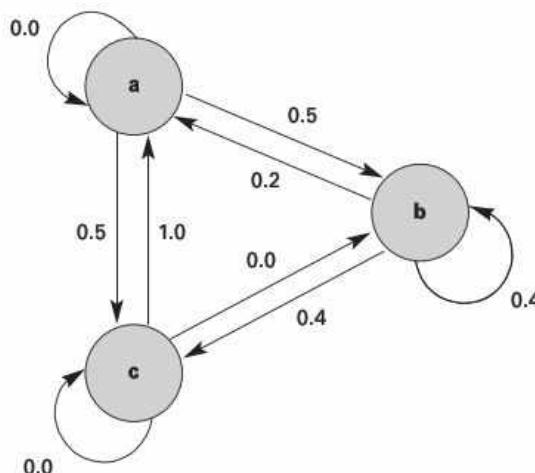
Un enfoque bastante diferente es el de Spafford [SPAF92a, SPAF92b]. Se basa en el uso de un filtro de Bloom [BLOO70]. Para empezar, explicamos la operación del filtro de Bloom. Un filtro Bloom de orden k consiste en un conjunto de k funciones *hash* independientes $H_1(x), H_2(x), \dots, H_k(x)$, donde cada función relaciona una contraseña con un valor *hash* en el rango de 0 a $N - 1$. Es decir,

$$H_i(X_j) = y \quad 1 \leq i \leq k; \quad 1 \leq j \leq D; \quad 0 \leq y \leq N - 1$$

donde

$X_j = j$ -ésima palabra en el diccionario de contraseñas

$D =$ número de palabras en el diccionario de contraseñas



$M = \{3, \{a, b, c\}, T, 1\}$ donde

$$T = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

ejemplo, ristra probablemente de este lenguaje: abbcacaba

ejemplo, ristra probablemente no de este lenguaje: aacccbaaa

Figura 9.5 Ejemplo de modelo de Markov

El siguiente procedimiento se aplica al diccionario:

1. Se define una tabla *hash* de N bits, con todos los bits inicialmente fijados a 0.
2. Para cada contraseña, se calculan sus k valores *hash*, y los correspondientes bits en la tabla *hash* se fijan a 1. Así, si $H_i(X) = 67$ para algunos (i, j) , entonces el bit en el orden 67 de la tabla *hash* se fija a 1; si el bit ya tiene el valor 1, permanece en 1.

Cuando una nueva contraseña se presenta al comprobador, se calculan sus k valores *hash*. Si todos los bits correspondientes de la tabla *hash* son iguales a 1, entonces la contraseña se rechaza. Todas las contraseñas en el diccionario serán rechazadas, pero habrá también algunos «falsos positivos» (es decir, contraseñas que no están en el diccionario pero que producen una coincidencia en la tabla *hash*). Para entender esto, imaginemos un esquema con dos funciones *hash*. Supongamos que las contraseñas *undertaker* y *hulkhogan* están en el diccionario, pero *xG%# jj98* no está. Además supongamos que

$$\begin{array}{lll} H_1(\text{undertaker}) = 25 & H_1(\text{hulkhogan}) = 83 & H_1(\text{xG%# jj98}) = 665 \\ H_2(\text{undertaker}) = 998 & H_2(\text{hulkhogan}) = 665 & H_2(\text{xG%# jj98}) = 998 \end{array}$$

Si la contraseña *xG%# jj98* se presenta al sistema, será rechazada incluso aunque no esté en el diccionario. Si hay demasiados falsos positivos, será difícil para los usuarios seleccionar contraseñas. Por lo tanto, nos gustaría diseñar el esquema *hash* para mini-

mizar los falsos positivos. Se puede mostrar que la probabilidad de un falso positivo puede ser aproximada por

$$P \approx (1 - e^{kD/N})^k = (1 - e^{kR})^k$$

o, de forma equivalente,

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

donde

k = número de funciones *hash*

N = número de bits en la tabla *hash*

D = número de palabras en el diccionario

$R = N/D$, proporción del tamaño de la tabla *hash* (bits) con respecto al tamaño del diccionario (palabras)

La Figura 9.6 determina P como función de R para distintos valores de k . Supongamos que tenemos un diccionario de un millón de palabras y deseamos tener un 0,01 de probabilidad de rechazar una contraseña que no esté en el diccionario. Si elegimos seis funciones *hash*, la proporción requerida es $R = 9,6$. Por lo tanto, necesitamos una tabla *hash* de $9,6 \times 10^6$ bits o, aproximadamente, 1,2 MBytes de capacidad. En contraste, la capacidad del diccionario completo requeriría del orden de ocho MBytes. Así, alcanzamos una compresión de casi un factor de siete. Además, la comprobación de las contraseñas trae consigo el cálculo de seis funciones *hash* y es independiente del tamaño del diccionario, mientras que con el uso del diccionario completo, hay una búsqueda sustancial².

9.4 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

Dos tratamientos minuciosos de la detección de la intrusión se encuentran en [BACE00] y [PROC01]. Un tratamiento más conciso pero útil se ofrece en [BACE01]. Dos artículos estadísticos cortos aunque prácticos sobre la materia son [KENT00] y [MCHU00]. En [HONE01] se presenta el informe definitivo sobre los *honeypots* y proporciona un análisis detallado de las herramientas y métodos de los *hackers*.

- BACE00** Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

² Tanto el modelo de Markov como el filtro de Bloom implican el uso de técnicas probabilísticas. En el caso del modelo de Markov, hay una pequeña probabilidad de que no se tomen algunas contraseñas que estén en el diccionario y una pequeña probabilidad de que algunas contraseñas que no estén en el diccionario sean rechazadas. En el caso del filtro de Bloom, hay una pequeña probabilidad de que algunas contraseñas que no están en el diccionario sean rechazadas.

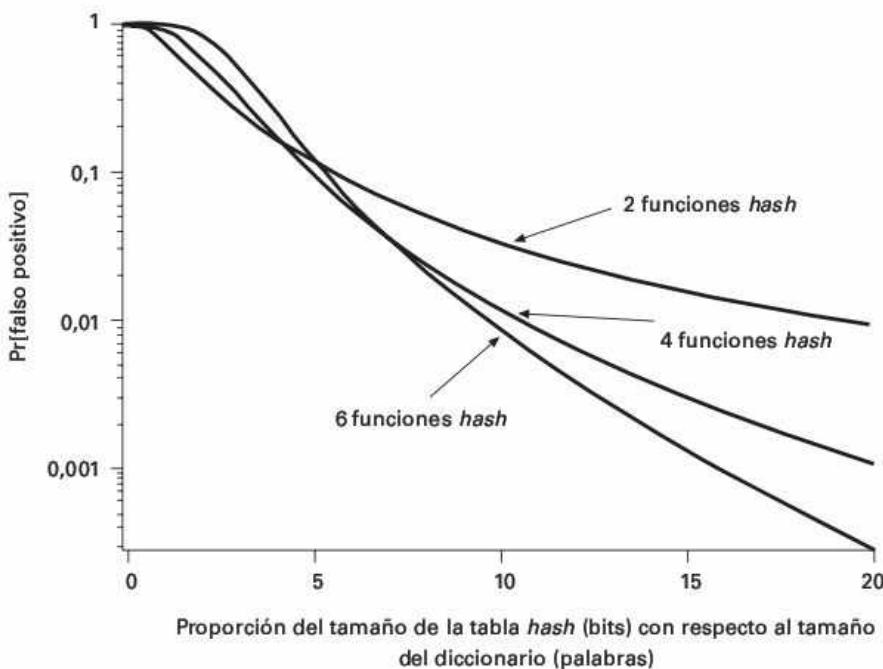


Figura 9.6 Funcionamiento del filtro de Bloom

BACE01 Bace, R., y Mell, P. *Intrusion Detection Systems*. NIST Special Publication SP 800-31, noviembre de 2000.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

KENT00 Kent, S. «On the Trail of Intrusions into Information Systems». *IEEE Spectrum*, diciembre de 2000.

MCHU00 McHugh, J.; Christie, A.; y Allen, J. «The role of Intrusion Detection Systems». *IEEE Software*, septiembre/octubre de 2000.

PROC01 Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

Sitios web recomendados:

- **CERT Coordination Center:** la organización que surgió a partir del equipo de respuesta ante urgencias en computadores formado por la Defense Advanced Research Projects Agency. El sitio proporciona información adecuada sobre las amenazas a la seguridad en Internet, vulnerabilidades y estadísticas de ataques.
- **Honeypot Project:** un proyecto de investigación que estudia las técnicas de los hackers predadores y que desarrolla productos *honeypot*.
- **Intrusion Detection Working Group:** incluye todos los documentos generados por este grupo de trabajo para la detección de intrusos.

9.5 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Contraseña	Falacia de la tasa base	Registro de auditoría
Detección de intrusos	Formato de intercambio de detección de intrusos	<i>Salt</i>
Detección de intrusos basada en reglas	<i>Honeypot</i>	Teorema de Bayes
Detección estadística de anomalías	Intruso	

PREGUNTAS DE REPASO

- 9.1. Enumera y define brevemente tres clases de intrusos.
- 9.2. ¿Cuáles son las dos técnicas comunes que se usan para proteger un archivo de contraseñas?
- 9.3. ¿Cuáles son los tres beneficios que puede proporcionar un sistema de detección de intrusos?
- 9.4. ¿Cuál es la diferencia entre la detección estadística de anomalías y la detección de intrusos basada en reglas?
- 9.5. ¿Qué métricas son útiles para la detección de intrusos basada en perfiles?
- 9.6. ¿Cuál es la diferencia entre la detección de la anomalía basada en reglas y la identificación de la penetración basada en reglas?
- 9.7. ¿Qué es un *honeypot*?
- 9.8. ¿Qué es *salt* en el contexto de la gestión de contraseñas UNIX?
- 9.9. Enumera y define brevemente cuatro técnicas que se usan para evitar las contraseñas fáciles de adivinar.

PROBLEMAS

- 9.1. Un taxi se vio una noche involucrado en un fatal accidente con fuga. Dos compañías de taxis, la Verde y la Azul, operan en la ciudad. Se te ha informado de que
 - El 85 por ciento de los taxis de la ciudad son Verde y el 15 por ciento Azul
 - Un testigo identificó el taxi como Azul
 El juez comprobó la fiabilidad del testigo en las mismas circunstancias que se dieron en la noche del accidente y concluyó que el testigo tenía razón al identificar el color del taxi en un 80 por ciento de las veces. ¿Cuál es la probabilidad de que el taxi involucrado en el accidente fuera Azul y no Verde?
- 9.2. Imagina que las contraseñas se seleccionan a partir de combinaciones de cuatro caracteres de los 26 caracteres del alfabeto. Imagina que un adversario puede probar contraseñas a una velocidad de una por segundo.

- a)** Si el adversario no obtiene retroalimentación hasta que cada intento haya sido completado, ¿cuál es el tiempo previsto para descubrir la contraseña correcta?
- b)** Si el adversario obtiene retroalimentación con señal de error cada vez que se introduce una contraseña equivocada, ¿cuál es el tiempo esperado para descubrir la contraseña correcta?
- 9.3** Supón que los elementos origen de longitud k se relacionan de manera uniforme con unos elementos destino de longitud p . Si cada dígito puede tomar uno de entre r valores, entonces el número de elementos origen es r^k y el número de elementos destino es el número menor r^p . Un elemento origen concreto x_j se corresponde con un elemento destino concreto y_j
- a)** ¿Cuál es la probabilidad de que el elemento origen correcto pueda ser seleccionado por un adversario en un intento?
- b)** ¿Cuál es la probabilidad de que un elemento origen diferente x_i ($x_i \neq x_j$) que resulta en el mismo elemento destino, y_j , pudiera ser producido por un adversario?
- c)** ¿Cuál es la probabilidad de que el elemento destino correcto pueda ser producido por un adversario en un intento?
- 9.4** Un generador fonético de contraseñas escoge dos segmentos al azar para cada contraseña de seis letras. La forma de cada segmento es CVC (consonante, vocal, consonante), donde V = <a,e,i,o,u> y C = V.
- a)** ¿Cuál es la población total de contraseñas?
- b)** ¿Cuál es la probabilidad de que un adversario advine una contraseña correctamente?
- 9.5** Imagina que las contraseñas están limitadas al uso de los 95 caracteres imprimibles ASCII y que todas las contraseñas tienen 10 caracteres. Imagina un pirata de contraseñas con una tasa de cifrado de 6,4 millones de cifrados por segundo. ¿Cuánto tiempo le llevará comprobar exhaustivamente todas las posibles contraseñas en un sistema UNIX?
- 9.6** Debido a los riesgos conocidos del sistema de contraseñas UNIX, la documentación SunOS-4.0 recomienda que el archivo de contraseñas se omita y se sustituya por un archivo legible públicamente llamado /etc/publickey. Una entrada en el archivo por el usuario A consiste en un identificador de usuario ID_A , la clave pública del usuario, KU_A , y la correspondiente clave privada KR_A . Esta clave privada se cifra usando DES con una clave derivada de la contraseña de acceso del usuario P_A . Cuando A entra en el sistema descifra $E_{P_A}[KR_A]$ para obtener KR_A
- a)** El sistema entonces verifica que P_A se proporcionó correctamente. ¿Cómo?
- b)** ¿Cómo puede un oponente atacar este sistema?
- 9.7** El esquema de cifrado usado por las contraseñas UNIX es de una sola dirección; no es posible invertirlas. Por lo tanto, ¿sería exacto decir que se trata, en realidad, de un código hash en vez de un cifrado de la contraseña?

- 9.8.** Se constató que la inclusión del valor *salt* en el esquema de contraseñas de UNIX multiplica la dificultad de averiguar por un factor de 4096. Pero este valor se almacena en texto claro en la misma entrada que la correspondiente contraseña de texto cifrado. Por lo tanto, esos dos caracteres son conocidos por el atacante y no necesitan ser averiguados. ¿Por qué se dice que el valor *salt* aumenta la seguridad?
- 9.9.** Asumiendo que has contestado correctamente el problema anterior y entiendes el significado de *salt*, ¿sería posible eliminar completamente todos los piratas de contraseñas aumentando drásticamente el tamaño de *salt*, por ejemplo, a 24 o 48 bits?
- 9.10.** Considera el filtro de Bloom discutido en la sección 9.3. Define k = número de funciones *hash*; N = número de bits en una tabla *hash*; y D = número de palabras en un diccionario.
- Demuestra que el número esperado de bits en la tabla *hash* que son iguales a cero se expresa como

$$\phi = \left(1 - \frac{k}{N}\right)^D$$

- Demuestra que la probabilidad de que una palabra introducida que no esté en el diccionario sea aceptada erróneamente como si estuviera en el diccionario es

$$P = (1 - \phi)^k$$

- Demuestra que la expresión precedente puede ser aproximada como

$$P \approx (1 - e^{-kD/N})^k$$

APÉNDICE 9A LA FALACIA DE LA TASA BASE

Comenzaremos con un repaso de los resultados importantes de la teoría de la probabilidad, para luego demostrar la falacia de la tasa base.

PROBABILIDAD CONDICIONAL E INDEPENDENCIA

A menudo queremos conocer una probabilidad que es condicional en algún acontecimiento. El efecto de la condición es eliminar algunos de los resultados del espacio del muestreo. Por ejemplo, ¿cuál es la probabilidad de conseguir una suma de ocho en la tirada de dos dados, si sabemos que la cara de al menos un dado es un número par? Lo podemos razonar de la siguiente manera: como un dado es par y la suma es par, el segundo dado debe mostrar un número par. Así, hay tres posibles resultados igualmente probables: (2, 6), (4, 4) y (6, 2), de un conjunto total de posibilidades de $[36 - (\text{número de opciones con las dos caras impares})] = 36 - 3 \times 3 = 27$. La probabilidad resultante es $3/27 = 1/9$.

Formalmente, la **probabilidad condicional** de un acontecimiento A asumiendo que el acontecimiento B ha ocurrido, denotado por $\Pr[A|B]$, se define como el ratio

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]}$$

donde asumimos que $\Pr[B]$ no es cero.

En nuestro ejemplo, $A = \{\text{suma de ocho}\}$ y $B = \{\text{al menos un dado par}\}$. La cantidad $\Pr[AB]$ abarca todos aquellos resultados en los que la suma es ocho y al menos un dado es par. Como hemos visto, hay tres resultados posibles. Así, $\Pr[AB] = 3/36 = 1/12$. Un momento de reflexión convencería de que $\Pr[B] = 3/4$. Ahora podemos calcular

$$\Pr[A|B] = \frac{1/12}{3/4} = \frac{1}{9}$$

Esto confirma nuestro razonamiento anterior.

Dos acontecimientos A y B se llaman **independientes** si $\Pr[AB] = \Pr[A]\Pr[B]$. Se puede observar fácilmente que si A y B son independientes, $\Pr[A|B] = \Pr[A]$ y $\Pr[B|A] = \Pr[B]$.

TEOREMA DE BAYES

Uno de los resultados más importantes de la teoría de la probabilidad se conoce como el teorema de Bayes. Primero necesitamos comentar la fórmula de la probabilidad total. Dado un conjunto de acontecimientos mutuamente exclusivos E_1, E_2, \dots, E_n , tales que la unión de estos acontecimientos cubra todos los posibles resultados y, dado un acontecimiento arbitrario A , se puede demostrar que

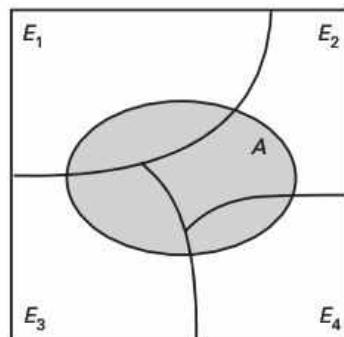
$$\Pr[A] = \sum_{i=1}^n \Pr[A|E_i]\Pr[E_i] \quad (9.1)$$

El teorema de Bayes puede exponerse como sigue:

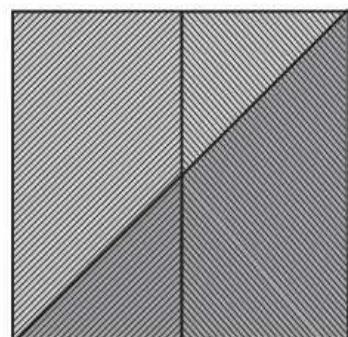
$$\Pr[E_i|A] = \frac{\Pr[A|E_i]\Pr[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i]\Pr[E_i]}{\sum_{j=1}^n \Pr[A|E_j]\Pr[E_j]} \quad (9.2)$$

La Figura 9.7a ilustra los conceptos de probabilidad total y el teorema de Bayes.

El teorema de Bayes se usa para calcular «probabilidades a posteriori,» es decir, la probabilidad de que algo realmente ocurra, dada una evidencia. Por ejemplo, supongamos que estamos transmitiendo una secuencia de ceros y unos por una línea de transmisión ruidosa. Sean S_0 y S_1 los hechos de que se envía un cero en un momento dado y que se envía un uno, respectivamente, y R_0 y R_1 los acontecimientos de que se recibe un cero y que se recibe un uno. Supongamos que conocemos las probabilidades del origen, a saber, $\Pr[S_1] = p$ y $\Pr[S_0] = 1 - p$. Ahora se observa la línea para determinar con qué frecuencia sucede un error cuando se envía un uno y cuándo se envía un cero,



(a) Diagrama para ilustrar los conceptos



<input checked="" type="checkbox"/> SO; 0 enviado	<input type="checkbox"/> RO; 0 recibido
<input checked="" type="checkbox"/> SO; 0 enviado	<input type="checkbox"/> R1; 1 recibido

(b) Ejemplo

Figura 9.7 Ilustración de la probabilidad total y el teorema de Bayes

y así, se calculan las siguientes probabilidades: $\Pr[R0|S1] = p_a$ y $\Pr[R1|S0] = p_b$. Si se recibe un cero, podemos entonces calcular la probabilidad condicional de un error, es decir, la probabilidad condicional de que se envió un uno dado que recibió un cero, usando el teorema de Bayes:

$$\Pr[S1|R0] = \frac{\Pr[R0|S1]\Pr[S1]}{\Pr[R0|S1]\Pr[S1] + \Pr[R0|S0]\Pr[S0]} = \frac{p_a p}{p_a p + (1-p_b)(1-p)}$$

La Figura 9.7b ilustra la ecuación anterior. En la figura, el espacio de muestra se representa por un cuadrado de área uno. La mitad del cuadrado corresponde a S0 y la otra mitad a S1, de modo que $\Pr[S0] = \Pr[S1] = 0,5$. Del mismo modo, la mitad del cuadrado corresponde a RO y la otra mitad a R1, así que $\Pr[R0] = \Pr[R1] = 0,5$. Dentro del área que representa S0, 1/4 de esa área corresponde a R1, así que $\Pr[R1|S0] = 0,25$. Otras probabilidades condicionales son igualmente evidentes.

LA FALACIA DE LA TASA BASE DEMOSTRADA

Consideremos la siguiente situación. Un paciente tiene una prueba para detectar una enfermedad que da como resultado positivo (indicando que tiene la enfermedad). Se nos informa de que

- la fiabilidad de la prueba es del 87 por ciento (por ejemplo, si un paciente tiene la enfermedad, el 87 por ciento de las veces, el test da el resultado correcto, y si el paciente no tiene la enfermedad, 87 por ciento de las veces, el test da el resultado correcto)
- la incidencia de la enfermedad en la población es del uno por ciento.

Dado que el test es positivo, ¿qué probabilidad existe de que el paciente no tenga la enfermedad? Es decir, ¿cuál es la probabilidad de que sea una falsa alarma? Necesitamos el teorema de Bayes para obtener la respuesta correcta:

$$\begin{aligned} \Pr[\text{bien/positivo}] &= \frac{\Pr[\text{positivo/bien}] \Pr[\text{bien}]}{\Pr[\text{positivo/enfermedad}] \Pr[\text{enfermedad}] + \Pr[\text{positivo/bien}] \Pr[\text{bien}]} \\ &= \frac{(0,13)(0,99)}{(0,87)(0,01) + (0,13)(0,99)} = 0,937 \end{aligned}$$

De tal modo, en la inmensa mayoría de los casos, cuando se detecta una condición de enfermedad, es una falsa alarma.

Este problema, usado en un estudio [PIAT91], se presentó a un número de personas. La mayoría de los sujetos dieron la respuesta de 13 por ciento. La inmensa mayoría, incluyendo muchos físicos, dieron un número menor a 50 por ciento. Muchos físicos que lo contestaron erróneamente se lamentaban, «si tú tienes razón, ¡no tiene sentido hacer pruebas clínicas!». La razón por la que la mayoría de las personas dan la respuesta equivocada es que no tienen en cuenta la tasa base de incidencia (la tasa base) cuando solucionan intuitivamente un problema. Este error es conocido como la *falacia de la tasa base*.

¿Cómo se podría solucionar este problema? Supongamos que pudiéramos dirigir ambas tasas de resultado correctas a 99 por ciento. Es decir, supongamos que tenemos $\Pr[\text{positivo/enfermedad}] = 0,99$ y $\Pr[\text{negativo/bien}] = 0,99$. Insertando estos números en la ecuación (9.2), obtenemos $\Pr[\text{bien/positivo}] = 0,01$. Así, si podemos detectar la enfermedad con precisión y detectar la falta de enfermedad con precisión a un nivel de 99 por ciento, entonces la tasa de falsas alarmas será sólo de un uno por ciento. Sin embargo, supongamos ahora que la incidencia de la enfermedad en la población es sólo de un $1/10000 = 0,0001$. Entonces acabamos con una tasa de falsas alarmas de 99 por ciento. En situaciones reales, [AXEL00] encontró que las probabilidades asociadas a los sistemas de detección de la intrusión eran tales que la tasa de falsa alarma era insatisfactoria.

CAPÍTULO 10

Software dañino

10.1 Virus y otras amenazas

Programas dañinos
La naturaleza de los virus
Tipos de virus
Virus de macro
Virus de correo electrónico
Gusanos

10.2 Contramedidas a los virus

Enfoques de antivirus
Técnicas avanzadas de antivirus
Software de bloqueo de acciones

10.3 Bibliografía y sitios web recomendados

10.4 Términos clave, preguntas de repaso y problemas

Términos clave
Preguntas de repaso
Problemas

El concepto de defensa: la detención de un golpe. Su rasgo característico: la espera de ese golpe.

De la guerra, CARL VON CLAUSEWITZ

Este capítulo examina el *software* dañino, concretamente los virus y los gusanos.

10.1 VIRUS Y OTRAS AMENAZAS

Quizá el tipo de amenaza más sofisticada para un computador sean los programas que explotan las vulnerabilidades de los sistemas computacionales. En este contexto, nos referimos tanto a programas de aplicaciones como a programas de utilidades, tales como editores y compiladores.

Comenzamos este apartado con una perspectiva del espectro de este tipo de amenazas constituidas por *software*. El resto de esta sección está dedicado a los virus y los gusanos, señalando en primer lugar su naturaleza y, a continuación, las contramedidas.

PROGRAMAS DAÑINOS

La Figura 10.1 ofrece una taxonomía general de las amenazas mediante *software*, o programas dañinos. Estas amenazas se pueden dividir en dos categorías: las que necesitan un programa donde alojarse y las que son independientes. Las primeras son, básicamente, fragmentos de programas que no pueden existir independientemente de algún programa de aplicación, utilidad o programa del sistema. Las segundas son programas por sí mismos que pueden ser ejecutados por el sistema operativo.

Se puede, además, diferenciar entre aquellas amenazas de *software* que no se replican y aquellas que sí lo hacen. La primera consiste en fragmentos de programas que van a ser activados cuando se invoca el programa anfitrión para llevar a cabo una función específica. La segunda consiste tanto en un fragmento de programa (virus) como en un programa independiente (gusano, bacteria) que, cuando se ejecuta, puede producir una o más copias de sí mismo que se activan luego en el mismo sistema o en algún otro sistema.

Aunque la taxonomía de la Figura 10.1 es útil para organizar la información que aquí se trata, no abarca todas las posibilidades. En concreto, las bombas lógicas o los caballos de Troya pueden ser parte de un virus o de un gusano. En el resto de este subapartado, se examinará brevemente cada uno de estos ejemplos de *software* dañino, con la excepción de los virus y los gusanos, que se tratarán con más detalle con posterioridad en esta sección.

Trampas

Una trampa es una entrada secreta a un programa que permite que alguien que es consciente de la trampa acceda sin pasar por los procedimientos de acceso de seguridad ha-

bituales. Las trampas han sido utilizadas legítimamente durante muchos años por los programadores para depurar y probar programas. Normalmente, esto se hace cuando el programador está desarrollando una aplicación que tiene un procedimiento de autenticación o un período largo de tiempo de inicio, lo que requiere que el usuario introduzca muchos valores diferentes para ejecutar la aplicación. Para depurar el programa, el desarrollador puede querer obtener privilegios especiales o evitar los procedimientos de inicialización y de autenticación necesarios. El programador puede también querer garantizar que hay un método para activar el programa en caso de que algo fallara con el procedimiento de autenticación que se está desarrollando en la aplicación. La trampa consiste en un código que reconoce secuencias especiales de entrada o que se activa cuando se ejecuta desde un identificador de usuario en particular o a partir de una secuencia improbable de acontecimientos.

Las trampas se vuelven amenazas cuando programadores sin escrúpulos las utilizan para obtener acceso no autorizado. La trampa era la idea básica de vulnerabilidad retratada en la película *Juegos de guerra* (*War Games*). Otro ejemplo lo hallamos en el desarrollo de Multics, en el que el «equipo tigre» del Ejército del Aire llevó a cabo pruebas de penetración (simulando enemigos). Una de las tácticas empleadas consistía en enviar una actualización de un sistema operativo falso a un sitio que ejecutara Multics. La actualización contenía un caballo de Troya (descrito más adelante) que podía ser activado por una trampa y que permitía al equipo tigre acceder. La amenaza estaba tan bien implementada que los desarrolladores de Multics no pudieron encontrarla, incluso después de ser informados de su presencia [ENGE80].

Es difícil implementar controles de sistema operativo para trampas. Las medidas de seguridad deben centrarse en el desarrollo del programa y en las actividades de actualización del *software*.

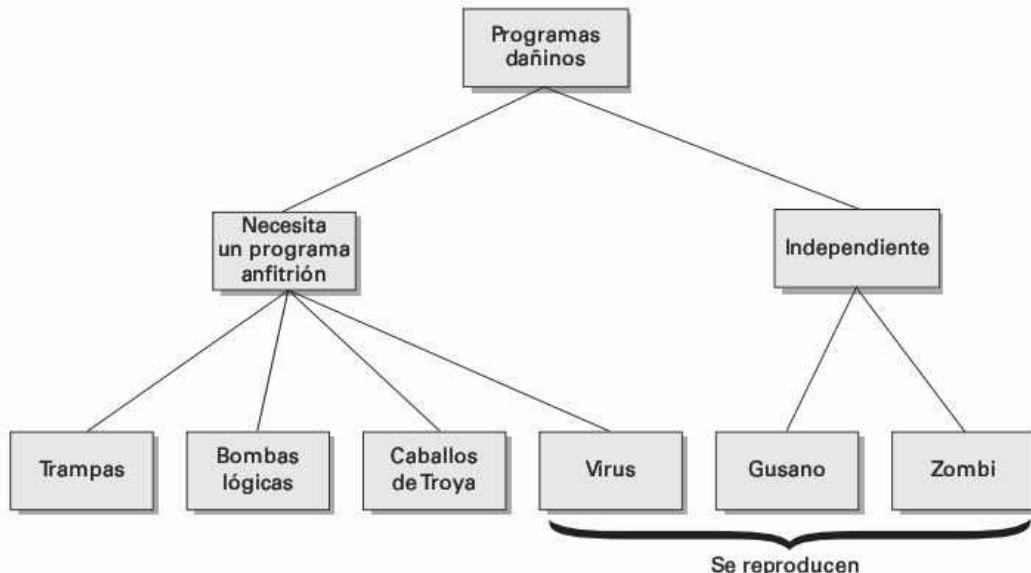


Figura 10.1 Taxonomía de los programas dañinos

Bomba lógica

Uno de los tipos más antiguos de amenaza, antecesor de los virus y los gusanos, es la bomba lógica. La bomba lógica es un código introducido en algún programa legítimo que está preparado para «explotar» cuando convergen ciertas condiciones. Entre las condiciones que pueden representar detonantes para una bomba lógica se encuentran la presencia o ausencia de ciertos archivos, un día concreto de la semana o fecha, o un usuario concreto ejecutando la aplicación. Una vez activada, una bomba puede alterar o borrar datos o archivos completos, causar una interrupción de la máquina, o provocar otros daños. Un ejemplo impactante de cómo se pueden usar las bombas lógicas fue el caso de Tim Lloyd, acusado de introducir una bomba lógica que costó a la empresa en la que trabajaba, Omega Engineering, más de diez millones de dólares, descarriló su estrategia de crecimiento corporativo y provocó el despido de 80 trabajadores [GAUD00]. Finalmente, Lloyd fue sentenciado a 41 meses en prisión y obligado a pagar dos millones de dólares en concepto de indemnización.

Caballos de Troya

Un caballo de Troya es un programa o procedimiento de comandos útil, o aparentemente útil, que contiene código oculto que al invocarse lleva a cabo funciones no deseadas o perjudiciales.

Estos programas se pueden usar para llevar a cabo indirectamente funciones que un usuario no autorizado no podría realizar directamente. Por ejemplo, para acceder a los archivos de otro usuario en un sistema compartido, un usuario podría crear un caballo de Troya que, cuando fuera ejecutado, cambiara los permisos de los ficheros del usuario que ejecute el programa, para que así los archivos sean legibles para cualquier otro usuario. El autor podría, entonces, inducir a los usuarios a ejecutar el programa, colocándolo en un directorio común y llamándolo de tal forma que parezca ser una utilidad práctica. Un ejemplo es un programa que aparentemente produzca un listado de los archivos del usuario en un formato conveniente. Después de que otro usuario haya ejecutado el programa, el autor puede acceder a la información en los archivos del usuario. Un caballo de Troya difícil de detectar sería un compilador modificado para insertar código adicional en ciertos programas mientras son compilados, como por ejemplo un programa de acceso al sistema (*login*) [THOM84]. El código crea una trampa en el programa de acceso que permite al autor entrar en el sistema usando una contraseña especial. Este troyano no se puede descubrir al leer el código fuente del programa de acceso.

Otra función común del caballo de Troya es la destrucción de datos. El programa parece estar realizando una función útil (por ejemplo, de calculadora), pero puede también estar borrando los archivos del usuario sin ser percibido. Por ejemplo, un ejecutivo de CBS fue víctima de un caballo de Troya que destruyó toda la información contenida en la memoria de su computador [TIME90]. El caballo de Troya fue implantado en una rutina de gráficos ofrecida en un tablón de anuncios electrónico.

Zombi

Un zombi es un programa que, sin ser percibido, toma el control de otro computador conectado a Internet y, luego, utiliza ese computador para lanzar ataques que hacen difícil

seguir la pista del creador del zombi. Los zombis se usan en ataques de denegación de servicio, normalmente contra sitios web muy concretos. El zombi se instala en cientos de computadores pertenecientes a terceras partes que no sospechan de nada, y luego se usan para inundar el sitio web elegido produciendo un elevado tráfico de Internet.

LA NATURALEZA DE LOS VIRUS

Un virus es un programa que puede «infectar» otros programas modificándolos; la modificación incluye una copia del programa del virus, que puede luego continuar infectando otros programas.

Los virus biológicos son pequeños extractos de código genético –ADN o ARN– que pueden tomar el control de la maquinaria de una célula viva y transformarla en miles de réplicas impecables del virus original. Como su homólogo biológico, un virus informático contiene en su código de instrucciones la receta para hacer copias perfectas de sí mismo. Alojado en un computador, el virus típico toma temporalmente el control del sistema operativo de dicho compadador. Luego, cuando éste, infectado, entra en contacto con una parte de *software* no infectada, una nueva copia del virus pasa al nuevo programa. Así, la infección puede propagarse de computador en computador por medio de usuarios desprevenidos que o bien intercambian disquetes o envían programas entre sí a través de la red. En un entorno de red, la habilidad para acceder a las aplicaciones y a los servicios del sistema de otros computadores proporciona un marco perfecto para la propagación de los virus.

Un virus puede hacer cualquier cosa que realicen otros programas. La única diferencia es que se adjunta a otro programa y se ejecuta secretamente a la vez que lo hace dicho programa. Una vez que el virus se ha ejecutado, puede llevar a cabo cualquier función, como borrar archivos y programas.

Durante el tiempo de vida, un virus común pasa por las siguientes cuatro fases:

- **Fase inactiva:** el virus está inactivo, pero acabará siendo activado por algún acontecimiento, como una fecha, la presencia de otro programa o archivo, o el exceso de algún límite por parte del disco. No todos los virus tienen esta fase.
- **Fase de propagación:** el virus coloca una copia idéntica de sí mismo en otros programas o en determinadas áreas del sistema en el disco. Cada programa infectado contendrá ahora un clon del virus, que entrará a su vez en una fase de propagación.
- **Fase de activación:** el virus se activa para llevar a cabo la función para la cual se creó. Al igual que en la fase inactiva, la fase de activación puede ser producida por una variedad de acontecimientos en el sistema, incluyendo un cálculo del número de veces que esta copia del virus ha hecho copias de sí mismo.
- **Fase de ejecución:** la función está ejecutada, y puede ser inofensiva, como un mensaje en la pantalla, o perjudicial, como la destrucción de programas y archivos.

La mayoría de los virus desempeñan su labor de forma específica para un sistema operativo en particular y, en algunos casos, específica para una plataforma de *hardware* concreta. Por lo tanto, se diseñan para beneficiarse de los detalles y puntos débiles de sistemas concretos.

La estructura de los virus

Un virus puede ir antepuesto o pospuesto a un programa ejecutable, o puede estar introducido de alguna otra forma. La clave de su funcionamiento está en que el programa infectado, cuando sea invocado, primero ejecutará el código del virus y luego el código original del programa.

```

program V :=
{goto main;
 1234567;

 subroutine infect-executable :=
 {loop:
   file := get-random-executable-file;
   if (first-line-of-file = 1234567)
     then goto loop
     else prepend V to file;}
 subroutine do-damage :=
 {whatever damage is to be done}
 subroutine trigger-pulled :=
 {return true if some condition holds}

main: main-program :=
 {infect-executable;
  if trigger-pulled then do-damage;
  goto next;}
next:
}
```

Figura 10.2 Un virus sencillo

En la Figura 10.2 se muestra una descripción muy general de la estructura de un virus (basada en [COHE94]). En este caso, el código del virus, V, está antepuesto a los programas infectados, y se asume que el punto de entrada al programa, cuando sea invocado, es la primera línea de éste.

Un programa infectado comienza con el código del virus y funciona como sigue: la primera línea de código es un salto al programa principal del virus. La segunda línea es un indicador especial que usa el virus para determinar si un posible programa víctima ya ha sido infectado con este virus o no. Cuando el programa es invocado, el control se transfiere inmediatamente al programa principal del virus. El programa del virus primero localiza archivos ejecutables que no estén infectados y los infecta. Luego, el virus puede llevar a cabo alguna acción, normalmente perjudicial para el sistema. Esta acción podría realizarse cada vez que se llama al programa, o podría ser una bomba lógica que se acciona sólo bajo ciertas circunstancias. Por último, el virus transfiere el control al programa original. Si la fase de infección del programa es lo suficientemente rápida, es poco probable que el usuario note diferencia entre la ejecución de un programa infectado y uno no infectado.

Un virus como el descrito es fácil de detectar porque una versión infectada de un programa es mayor que la correspondiente no infectada. Una forma de impedir este medio tan sencillo de localización de un virus es comprimir el archivo ejecutable para que tanto las

versiones infectadas como las no infectadas tengan idéntica longitud. La Figura 10.3 [COHE94] muestra en términos generales la lógica requerida. Las líneas clave de este virus están numeradas y la Figura 10.4 [COHE94] ilustra la operación. Asumimos que

```

program CV :=
{Goto main;
 01234567;

subroutine infect-executable :=
{loop:
  file := get-random-executable-file;
  if (first-line-of-file = 01234567) then goto loop;
  (1) compress file;
  (2) prepend CV to file;
}

main: main-program :=
{if ask-permission then infect-executable;
(3) uncompress rest-of-file;
(4) run uncompressed file;}

```

Figura 10.3 Lógica para un virus de compresión

el programa P_1 está infectado con el virus CV. Cuando se invoca este programa, el control pasa a su virus, que realiza los siguientes pasos:

1. Para cada archivo no infectado P_2 que se encuentre, el virus, primero, comprime ese archivo para producir P'_2 , que se acorta con respecto al programa original en una longitud igual al tamaño del virus.
2. Se antepone una copia del virus al programa comprimido.
3. Se descomprime la versión comprimida del programa original infectado, P'_1 .
4. Se ejecuta el programa original descomprimido.

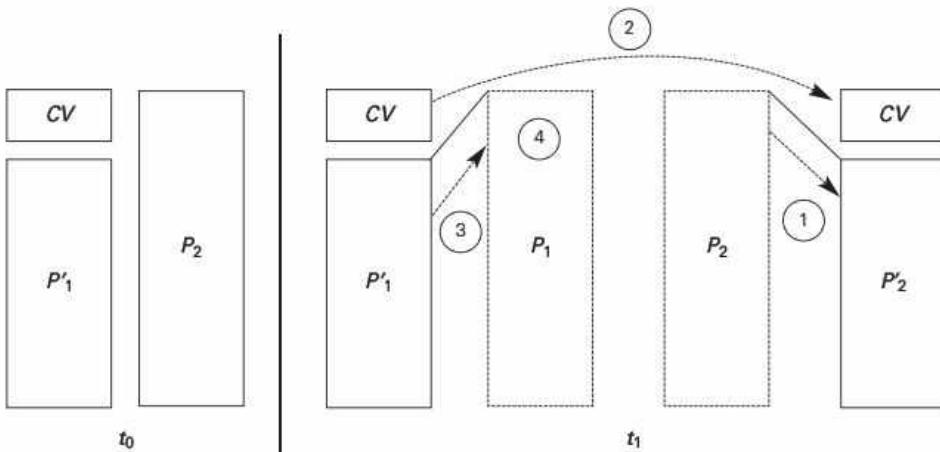


Figura 10.4 Un virus de compresión

En este ejemplo, el virus no hace otra cosa que propagarse. Como en el ejemplo anterior, el virus puede incluir una bomba lógica.

Infeción inicial

Una vez que el virus ha obtenido acceso al sistema infectando un solo programa, está en posición de infectar algunos o todos los demás archivos ejecutables de ese sistema cuando el programa infectado se ejecute. Así, la infección del virus puede prevenirse por completo si se evita que el virus acceda en primer lugar. Desgraciadamente, la preventión es extraordinariamente difícil porque un virus puede ser parte de cualquier programa fuera del sistema. Así, a menos que uno se conforme con una caja de latón completamente vacía y escriba sus propios programas y aplicaciones, cualquiera es vulnerable.

La mayoría de las infecciones de virus se inician con un disco del cual se han copiado programas en un computador. Muchos de ellos son discos que tienen juegos o utilidades sencillas pero útiles que los empleados obtienen para los computadores de sus casas y luego los llevan a la oficina y los instalan en una máquina. Algunos, sorprendentemente, están presentes en los discos que vienen con el embalaje del fabricante de una aplicación. Sólo una pequeña fracción de las infecciones comienza a través de una conexión a red. De nuevo, lo más habitual es que un empleado descargue un juego o una utilidad aparentemente útil, simplemente para descubrir que contiene un virus.

TIPOS DE VIRUS

Desde que los virus aparecieron por primera vez, ha habido una lucha continua entre los creadores de virus y los de antivirus. A medida que se han desarrollado medidas que contrarresten los tipos de virus existentes, se han desarrollado nuevos tipos. [STEP93] presenta las siguientes categorías, entre las que están los tipos de virus más significativos:

- **Virus parásito:** es el virus tradicional y todavía el más común. Un virus parásito se adjunta por sí mismo a los archivos ejecutables y se replica, cuando el programa infectado se ejecuta, encontrando otros archivos ejecutables que infectar.
- **Virus residente en la memoria:** se aloja en la memoria principal como parte de un programa residente del sistema. Desde ahí, el virus infecta cada programa que se ejecuta.
- **Virus del sector de arranque:** infecta un registro de arranque principal o cualquier registro de arranque y se extiende cuando un sistema se arranca desde el disco que contiene el virus.
- **Virus furtivo:** un virus diseñado explícitamente para evitar la detección por parte de un antivirus.
- **Virus polimórfico:** un virus que se modifica con cada una de las infecciones, haciendo imposible la detección por la «firma» del virus.

Anteriormente se expuso un ejemplo de **virus furtivo**: un virus que usa la compresión para que el programa infectado tenga exactamente la misma longitud que la versión no infectada. Pero son posibles técnicas aún más sofisticadas. Por ejemplo, un virus puede interceptar la lógica de las rutinas I/O del disco, para que cuando haya un intento de leer

las partes sospechosas del disco usando estas rutinas, el virus presente el programa original, sin infección. Esta forma sigilosa de actuar es una técnica empleada por un virus para evadir la detección.

Un **virus polimórfico** crea copias durante la reproducción que son funcionalmente equivalentes, pero que tienen, claramente, distintos patrones de bits. Al igual que con el virus furtivo, la intención es esquivar a los programas que buscan virus. En este caso, la «firma» del virus variará con cada copia. Para alcanzar esta variación, el virus puede, de manera aleatoria, insertar instrucciones superfluas o intercambiar el orden de las instrucciones independientes. Un enfoque más efectivo lo constituye el cifrado. Una parte del virus, generalmente llamada (*mutation engine*) *generadora de mutaciones*, crea una clave de cifrado aleatoria para cifrar el resto del virus. La clave se almacena con el virus, y el propio generador de mutaciones se ve alterado. Cuando se activa un programa infectado, el virus utiliza la clave aleatoria almacenada para descifrarlo. Cuando el virus se replica, se selecciona una clave aleatoria distinta.

Otra arma en el arsenal del creador de virus es la caja de herramientas de creación de virus. Estas herramientas permiten a un principiante con algunos conocimientos crear rápidamente una cantidad de virus distintos. Aunque los virus creados con la caja de herramientas tienden a ser menos sofisticados que los virus diseñados partiendo de la improvisación, el número total de nuevos virus que pueden generarse crea un problema para los programas antivirus.

VIRUS DE MACRO

En los últimos años, el número de virus registrados en sitios corporativos ha aumentado dramáticamente. Prácticamente todo este aumento se debe a la proliferación de uno de los últimos tipos de virus: el virus de macro. Según la National Computer Security Agency (www.ncsa.com), los virus de macro representan hoy en día las dos terceras partes del total de los virus informáticos.

Los virus de macro son especialmente amenazadores por una serie de razones:

1. Un virus de macro es una plataforma independiente. Prácticamente todos los virus de macro infectan los documentos de Microsoft Word. Cualquier plataforma de *hardware* y sistema operativo que contenga Word puede ser infectado.
2. Los virus de macro infectan documentos, no partes de código ejecutables. La mayor parte de la información introducida en un sistema computacional tiene forma de documento más que de programa.
3. Los virus de macro se propagan fácilmente. Un método muy común es a través del correo electrónico.

Los virus de macro se aprovechan de una característica encontrada en Word y otras aplicaciones de oficina como Microsoft Excel, concretamente la macro. En esencia, una macro es un programa ejecutable insertado en un documento de procesamiento de texto u otro tipo de archivo. Normalmente, los usuarios utilizan las macros para automatizar tareas repetitivas y, de este modo, ahorrar pulsaciones en el teclado. El lenguaje de macros normalmente es alguna variante del lenguaje de programación Basic. Un usuario podría definir una secuencia de pulsaciones en una macro y fijarla para que la macro se active cuando se introduzca una tecla de función o una combinación corta de teclas concreta.

Lo que hace posible crear un virus de macro es la macro autoejecutable. Se trata de una macro que se activa automáticamente, sin una entrada explícita de un usuario. Algunas de las acciones comunes que producen la autoejecución consisten en abrir un archivo, cerrar un archivo e iniciar una aplicación. Una vez que la macro está funcionando, puede copiarse a sí misma en otros documentos, borrar archivos y causar otros tipos de daño al sistema del usuario. En Microsoft Word, hay tres tipos de macros autoejecutables:

- **Autoexecute:** si una macro llamada AutoExec está en la plantilla «normal.dot» o en una plantilla global almacenada en un directorio de inicio de Word, se ejecuta cada vez que Word se inicia.
- **Automacro:** una automacro se ejecuta cuando se produce un acontecimiento concreto, como abrir o cerrar un documento, crear un nuevo documento o salir de Word.
- **Command macro:** si una macro en un archivo de macro global o una macro adjunta a un documento tiene el nombre de un comando de Word existente, se ejecuta cada vez que el usuario invoca ese comando (por ejemplo, Guardar Archivo).

A continuación se presenta una técnica común para extender un virus de macro. Una automacro o una macro de comandos se adjunta a un documento de Word que se ha introducido en el sistema por correo electrónico o mediante transferencia desde un disco. En algún momento, después de que el documento se ha abierto, la macro se ejecuta; luego, se copia a sí misma en el archivo de macro global. Cuando se abre la siguiente sesión de Word, la macro global infectada está activa. Cuando esta macro se ejecuta, puede replicarse y causar daños.

Las sucesivas versiones de Word proporcionan una protección en aumento contra los virus de macros. Por ejemplo, Microsoft ofrece una herramienta opcional de protección contra virus de macros (*Macro Virus Protection*) que detecta archivos sospechosos de Word y alerta al cliente del riesgo potencial que implica abrir un archivo con macros. Distintos vendedores de productos antivirus también han desarrollado herramientas para detectar y corregir virus de macro. Como en otros tipos de virus, la carrera armamentística continúa en el campo de los virus de macro.

VIRUS DE CORREO ELECTRÓNICO

Un desarrollo más reciente del *software* dañino es el virus por correo electrónico. Los primeros virus de correo electrónico de expansión rápida, como el Melissa, hicieron uso de una macro de Microsoft Word insertada en un archivo adjunto. Si el destinatario abre el archivo adjunto del correo, la macro de Word se activa. Entonces

1. El virus del correo electrónico se envía a todos los de la lista de correo en el paquete de correo del usuario.
2. El virus produce daños locales.

A finales de 1999 apareció una versión más potente del virus de correo electrónico. Esta nueva versión puede activarse simplemente al abrir un correo electrónico que contenga el virus, sin necesidad de abrir el archivo adjunto. El virus usa el lenguaje de *script* de Visual Basic a través del paquete de correo.

Así, nos encontramos con una nueva generación de *malware*, o *software* dañino, que llega vía e-mail y usa las características del *software* de correo electrónico para replicarse a través de Internet. El virus se propaga inmediatamente al ser activado (o bien al abrir un archivo adjunto del correo o al abrir el correo) a todas las direcciones de correo electrónico conocidas por el computador infectado. Como resultado, mientras los virus solían necesitar meses o años para propagarse, ahora lo hacen en horas. Esto hace que al *software* antivirus le resulte muy difícil responder antes de que se hayan producido muchos daños. Por todo ello, se hace necesario alcanzar un mayor grado de seguridad en la utilidad de Internet y en las aplicaciones de los computadores para contraatacar la creciente amenaza.

GUSANOS

Un virus de correo electrónico tiene algunas de las características de un gusano, porque se propaga de sistema en sistema. Sin embargo, aún podemos clasificarlo como virus porque necesita a un ser humano para avanzar. Un gusano busca activamente más máquinas que infectar y cada máquina que sea infectada sirve de lanzadera automática para atacar a otras máquinas.

Los programas de gusanos en la red usan conexiones de red para extenderse de sistema en sistema. Una vez activo dentro del sistema, un gusano de la red puede comportarse como un virus de computador o una bacteria, o podría implantar caballos de Troya o llevar a cabo cualquier número de acciones perjudiciales o destructivas.

Para replicarse a sí mismo, un gusano usa algún tipo de vehículo de la red. Los ejemplos incluyen

- **Correo electrónico:** un gusano envía por correo una copia de sí mismo a otros sistemas.
- **Capacidad de ejecución remota:** un gusano ejecuta una copia de sí mismo en otro sistema.
- **Capacidad de acceso remoto:** un gusano entra en la sesión de un sistema remoto como usuario y luego usa los comandos para copiarse de un sistema a otro.

La nueva copia del gusano se ejecuta, luego, en el sistema remoto donde, además de cualquier función que lleve a cabo en ese sistema, continúa su propagación de la misma manera.

Un gusano de la red presenta las mismas características que un virus informático: una fase inactiva, una fase de propagación, una fase de activación y una fase de ejecución. La fase de propagación normalmente desempeña las siguientes funciones:

1. Buscar otros sistemas para infectarlos, examinando tablas de computadores o listas similares que contengan direcciones de sistemas remotos.
2. Establecer una conexión con un sistema remoto.
3. Copiarse en el sistema remoto y hacer que la copia se active.

El gusano, antes de copiarse en el sistema, puede también intentar determinar si un sistema ha sido infectado previamente. En un sistema de multiprogramación, puede tam-

bien disfrazar su presencia denominándose a sí mismo como un proceso del sistema o usando algún otro nombre que no pueda ser percibido por el operador del sistema.

Como con los virus, los gusanos son difíciles de controlar. Sin embargo, si tanto la seguridad de la red como las medidas de seguridad de sistemas individuales están diseñadas e implementadas adecuadamente, la amenaza de los gusanos se reduce.

El gusano Morris

Hasta la generación actual de gusanos, el más conocido era el gusano lanzado en Internet por Robert Morris en 1988. El gusano Morris fue diseñado para propagarse en los sistemas UNIX y usó diferentes técnicas de propagación. Cuando una copia comenzaba la ejecución, su primera tarea era descubrir otros *host* conocidos que permitirían la entrada desde este *host*. El gusano llevaba a cabo esta tarea examinando una variedad de listas y tablas, incluyendo las del sistema, que informaban sobre qué otras máquinas eran de confianza para este *host*, archivos de redirecciónamiento de correo de los usuarios, tablas por las que los usuarios daban permiso para acceder a cuentas remotas, y desde un programa que informaba del estado de las conexiones en red. Por cada *host* descubierto, el gusano probaba una serie de métodos para acceder:

1. Intentaba acceder a un *host* remoto como usuario legítimo. Mediante este método, el gusano primero intentaba forzar el archivo local de contraseñas, y luego usaba las contraseñas descubiertas y los correspondientes identificadores de usuario. El supuesto era que muchos usuarios utilizarían la misma contraseña en distintos sistemas. Para obtener las contraseñas, el gusano ejecutaba un programa para descubrir claves que probaban lo siguiente:
 - a) Cada nombre de cuenta de usuario y permutaciones sencillas de él.
 - b) Una lista de 432 contraseñas incorporadas que Morris consideró como posibles candidatas.
 - c) Todas las palabras en el directorio local del sistema.
2. Explotaba un fallo en el protocolo *finger*, que informa del paradero de un usuario remoto.
3. Explotaba una puerta trampa en la opción de depurar del proceso remoto que recibe y envía correo.

Si alguno de estos ataques prosperaba, el gusano lograba la comunicación con el intérprete de comandos del sistema operativo. Luego enviaba a este intérprete un pequeño programa de arranque y un comando para ejecutar ese programa, y luego se desconectaba. A continuación, el programa de arranque llamaba al programa padre y descargaba el resto del gusano. Después, el nuevo gusano se ejecutaba.

Ataques de gusanos actuales

La era contemporánea de amenazas de gusanos comenzó con el lanzamiento del gusano Código Rojo (*Code Red*) en julio de 2001. Este gusano explota un agujero de seguridad en el Servidor de Información de Internet de Microsoft (IIS, *Internet Information Ser-*

ver) para penetrar y expandirse. También inutiliza el comprobador de archivos de sistema en Windows. El gusano prueba direcciones IP aleatorias para extenderse por otros computadores. Durante un cierto período de tiempo, lo único que hace es expandirse. Luego inicia un ataque de denegación de servicio contra un sitio web gubernamental inundándolo con paquetes de numerosos *hosts*. Después, el gusano suspende las actividades y se reactiva periódicamente. En la segunda oleada de ataque, el Código Rojo infectó casi 360.000 servidores en 14 horas. Además de los estragos que causa al servidor elegido, el Código Rojo puede consumir enormes cantidades de capacidad en Internet, interrumpiendo el servicio.

El Código Rojo II es una variante dirigida a los IIS de Microsoft. Además, este gusano instala una puerta trasera que permite a un pirata informático controlar las actividades de los computadores víctimas.

A finales de 2001 apareció un gusano más versátil, conocido como Nimda. Éste se extiende mediante diversos mecanismos:

- de cliente a cliente vía e-mail
- de cliente a cliente a través de recursos compartidos abiertos de la red
- de servidor web a cliente mediante búsqueda de sitios web comprometidos
- de cliente a servidor web por medio de la exploración activa y la explotación de distintas vulnerabilidades transversales del directorio del IIS 4.0/5.0 de Microsoft
- de cliente a servidor web mediante la exploración de las puertas traseras dejadas atrás por los gusanos «Código Rojo II»

El gusano modifica los documentos web (por ejemplo, archivos .htm, .html y .asp) y ciertos archivos ejecutables encontrados en los sistemas que infecta; y crea numerosas copias de sí mismo con diferentes nombres de archivo.

10.2 CONTRAMEDIDAS A LOS VIRUS

ENFOQUES DE ANTIVIRUS

La solución ideal para la amenaza de los virus es la prevención, es decir, no permitir que un virus entre en el sistema. Este objetivo es, en general, imposible de alcanzar, aunque la prevención puede reducir el número de ataques con éxito. El segundo mejor enfoque consiste en hacer lo siguiente:

- **Detección:** una vez que se ha producido la infección, determinar qué ha ocurrido y localizar el virus.
- **Identificación:** una vez que la detección se ha realizado, identificar el virus específico que ha infectado un programa.
- **Eliminación:** una vez que el virus específico ha sido identificado, eliminar cualquier rastro del virus del programa infectado y restaurarlo a su estado original. Eliminar el virus de todos los sistemas infectados para que no pueda continuar extendiéndose.

Si se logra la detección, pero no es posible ni la identificación ni la eliminación, entonces la alternativa es ignorar el programa infectado y volver a cargar la versión limpia de la copia de seguridad.

Los avances en la tecnología de virus y antivirus van de la mano. Los primeros virus eran fragmentos de código relativamente simples y podían ser identificados y purgados con paquetes de *software* antivirus relativamente sencillos. A medida que los virus han evolucionado, tanto los virus como, necesariamente, los antivirus se han hecho más complejos y sofisticados.

[STEP93] identifica cuatro generaciones de *software* antivirus:

- Primera generación: exploradores simples
- Segunda generación: exploradores heurísticos
- Tercera generación: detección de actividad
- Cuarta generación: protección integral

Un explorador de **primera generación** requiere una firma del virus para poder identificarlo. El virus puede contener patrones de búsqueda (*wildcards*) pero tiene básicamente la misma estructura y patrón de bits en todas las copias. Estos exploradores de firmas están limitados a la detección de virus conocidos. Otro tipo de exploradores de primera generación tiene un registro con la longitud de los programas y busca cambios en la extensión.

Un explorador de **segunda generación** no se basa de una firma concreta. En lugar de ello, usa reglas heurísticas para buscar posibles infecciones de virus. Algunos de estos exploradores buscan fragmentos de código que, con frecuencia, se asocian con los virus. Por ejemplo, un explorador puede buscar el comienzo de un bucle de cifrado usado en un virus polimórfico y descubrir la clave de cifrado. Una vez que la clave ha sido descubierta, el explorador puede descifrar el virus para identificarlo, luego eliminar la infección y volver a poner en servicio el programa.

Otro enfoque de la segunda generación es la comprobación de la integridad. Se puede añadir una suma de comprobación a cada programa. Si un virus infecta el programa sin cambiar la suma de comprobación, entonces una comprobación de la integridad detectará el cambio. Para contrarrestar un virus lo suficientemente sofisticado como para cambiar la suma de comprobación cuando infecta un programa, se puede usar una función *hash* cifrada. La clave de cifrado se almacena separada del programa para que el virus no pueda generar un nuevo código *hash* y cifrar el programa. Usando una función *hash* en vez de una suma de comprobación más sencilla, se evita que el virus ajuste el programa para producir el mismo código *hash*.

Los programas de **tercera generación** son programas residentes en la memoria que identifican un virus por su acción más que por su estructura en un programa infectado. Tales programas tienen la ventaja de que no es necesario desarrollar firmas ni heurística para una amplia gama de virus. Al contrario, sólo es necesario identificar el pequeño grupo de acciones que indican que se está intentando producir una infección y, luego, intervenir.

Los productos de **cuarta generación** son paquetes compuestos por una variedad de técnicas antivirus usadas conjuntamente. Éstas incluyen componentes de exploración y de detección de acciones. Además, este paquete incluye capacidad de control de acceso,

que limita la habilidad de los virus para penetrar en un sistema y luego limita la habilidad de un virus para actualizar archivos con el fin de dejar pasar la infección.

La carrera armamentística continúa. Con los paquetes de cuarta generación, se emplea una estrategia defensiva más exhaustiva, ampliando el ámbito de la defensa a medidas de seguridad computacional de propósito más general.

TÉCNICAS AVANZADAS DE ANTIVIRUS

Continúan apareciendo enfoques y productos antivirus más sofisticados. En este subapartado, se destacan dos de los más importantes.

Descifrado genérico

El descifrado genérico (GD, *Generic Decryption*) permite al programa antivirus detectar fácilmente incluso los virus polimórficos más complejos, a la vez que mantiene velocidades de rastreo importantes [NACH97]. Recordemos que cuando se ejecuta un archivo que contiene un virus polimórfico, el virus puede descifrarse para activarse. Para detectar dicha estructura, se ejecutan archivos a través de un explorador GD, que contiene los siguientes elementos:

- **Emulador de CPU:** un computador virtual basado en *software*. Las instrucciones de un archivo ejecutable, en vez de ser ejecutadas en el procesador subyacente, son interpretadas por un emulador. El emulador incluye versiones de *software* de todos los registros y otros elementos *hardware* del procesador, para que el procesador subyacente no se vea afectado por los programas interpretados en el emulador.
- **Explorador de firma de virus:** un módulo que explora el código objetivo buscando firmas de virus conocidas.
- **Módulo de control de emulación:** controla la ejecución del código objetivo.

Al comienzo de cada simulación, el emulador comienza a interpretar las instrucciones en el código del objetivo, de una en una. Así, si el código incluye una rutina de descifrado que descifre y, por lo tanto, exponga el virus, ese código es interpretado. En efecto, el virus hace el trabajo del programa antivirus exponiendo el virus. Periódicamente, el módulo de control interrumpe la interpretación para examinar el código objetivo en busca de firmas de virus.

Durante la interpretación, el código meta no puede causar daño al entorno real del computador personal, porque está siendo interpretado en un entorno completamente controlado.

El aspecto de diseño más difícil con un explorador GD es determinar cuánto tiempo dedicar a cada interpretación. Normalmente, los elementos de virus se activan poco después de que un programa comienza su ejecución, pero no tiene que ser el caso. Cuanto más tiempo emule el explorador un programa concreto, mayor probabilidad de que se atrape cualquier virus escondido. Sin embargo, el programa antivirus puede utilizar solamente una limitada cantidad de tiempo y de recursos antes de que los usuarios se quejen.

Sistema de inmunidad digital

El sistema de inmunidad digital es un enfoque integral a la protección contra virus desarrollado por IBM [KEPH97a, KEPH97b]. La razón de este desarrollo ha sido la creciente amenaza de la propagación de virus a través de Internet. En primer lugar, se describirá brevemente esta amenaza y luego se resumirá el enfoque de IBM.

Tradicionalmente, la amenaza de los virus se caracterizaba por la relativa lentitud en la expansión de los nuevos virus y las nuevas mutaciones. El *software* antivirus se solía actualizar mensualmente, y esto era suficiente para controlar el problema. También, Internet jugaba un papel comparativamente menor en la propagación de los virus. Pero como señala [CHES97], las dos tendencias principales en la tecnología de Internet han tenido un impacto creciente en el ritmo de propagación de los virus de estos últimos años:

- **Sistemas de correo integrado:** sistemas como Lotus Notes y Microsoft Outlook hacen muy sencillo enviar cualquier cosa a cualquier persona y trabajar con los objetos que se reciben.
- **Sistemas de programas móviles:** capacidades como Java y ActiveX permiten que los programas se muevan por sí mismos de un sistema a otro.

En respuesta a la amenaza planteada por estas capacidades de Internet, IBM ha desarrollado el prototipo de un sistema de inmunidad digital. Este sistema se extiende en el uso de la emulación de programas, aspecto tratado en el subapartado anterior, y proporciona una emulación de propósito general y un sistema de detección de virus. El objetivo de este sistema es suministrar un tiempo de respuesta rápido para que los virus se puedan erradicar casi tan pronto como sean introducidos. Cuando un nuevo virus entra en una organización, el sistema de inmunidad lo captura automáticamente, lo analiza, añade los procedimientos de detección y de protección, lo elimina, y pasa la información sobre ese virus a sistemas que ejecutan antivirus de IBM para que pueda ser detectado antes de que pueda ejecutarse en algún otro lugar.

La Figura 10.5 ilustra los pasos más comunes en el funcionamiento del sistema de inmunidad digital:

1. Un programa de supervisión en cada PC usa heurística basada en el comportamiento del sistema, en cambios sospechosos en los programas o en la firma específica para inferir que un virus puede estar presente. El programa de supervisión reenvía una copia de cualquier programa que se suponga infectado a una máquina administrativa dentro de la organización.
2. La máquina administrativa cifra la muestra y la envía a la máquina central de análisis de virus.
3. Esta máquina crea un entorno en el que el programa infectado puede ser ejecutado sin peligro para su análisis. Las técnicas empleadas con este propósito incluyen la emulación o la creación de un entorno protegido dentro del cual el programa sospechoso pueda ser ejecutado y supervisado. La máquina de análisis de virus produce entonces una prescripción para identificar y eliminar el virus.
4. La prescripción resultante se envía de vuelta a la máquina administrativa.
5. La máquina administrativa envía la prescripción al cliente infectado.

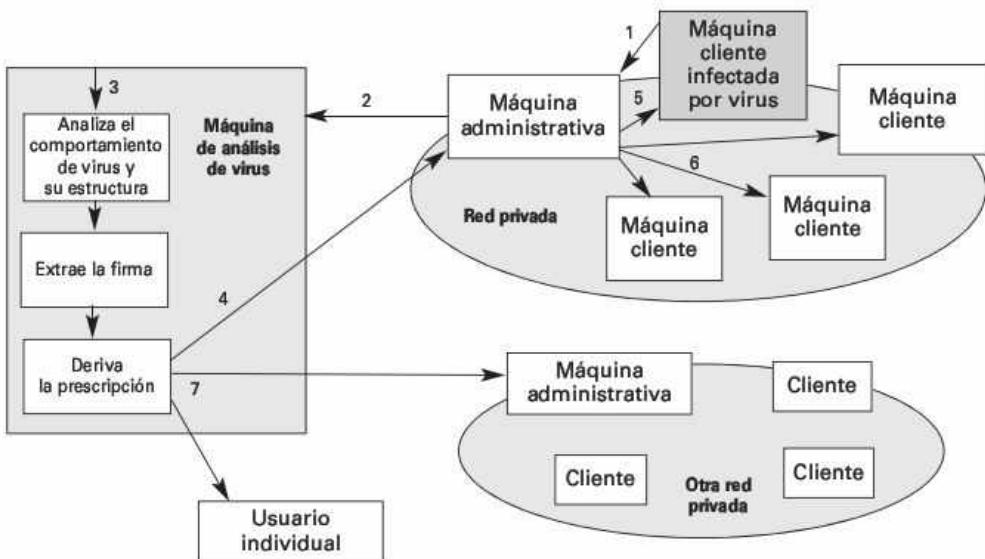


Figura 10.5 Sistema de inmunidad digital

6. La prescripción también se reenvía a otros clientes de la organización.
7. Los suscriptores de todo el mundo reciben actualizaciones del antivirus de forma regular que les protegen de los nuevos virus.

El éxito del sistema de inmunidad digital depende de la habilidad de la máquina de análisis de virus para detectar nuevas tendencias de virus. Analizando y supervisando constantemente los virus catalogados como más extendidos, debería ser posible actualizar continuamente el *software* de inmunidad digital para afrontar las amenazas.

SOFTWARE DE BLOQUEO DE ACCIONES

A diferencia de los exploradores heurísticos o basados en huellas digitales, el *software* de bloqueo de acciones se integra con el sistema operativo de un computador y controla el funcionamiento del programa en tiempo real en busca de acciones dañinas. Así, el *software* bloquea las posibles acciones perjudiciales antes de que tengan la oportunidad de afectar al sistema. Los comportamientos que se controlan pueden incluir:

- Intentos de abrir, visualizar, borrar, y/o modificar archivos
- Intentos de formatear unidades de disco y otras operaciones del disco no recuperables
- Modificaciones en la lógica de archivos ejecutables, *scripts* de macros
- Modificaciones de configuraciones críticas del sistema, como configuraciones de arranque
- Creación de *scripts* de correo electrónico y clientes de mensajería instantánea para enviar contenido ejecutable

- Iniciación de las comunicaciones en la red

Si el bloqueador de acciones detecta que al ejecutarse un programa podría estar iniciando comportamientos perjudiciales, puede bloquear estas iniciativas en tiempo real y/o finalizar la ejecución del *software*. Esto ofrece una ventaja fundamental a las técnicas de detección antivirus ya establecidas como, por ejemplo, las basadas en huellas o las heurísticas. Mientras hay literalmente billones de diferentes maneras de confundir y adaptar las instrucciones de un virus o de un gusano, muchos de los cuales evadirán la detección de un explorador de huella digital o heurístico, normalmente el código dañino debe realizar una solicitud bien definida al sistema operativo. Dado que el bloqueador puede interceptar todas estas solicitudes, puede identificar y bloquear acciones perjudiciales sin tener en cuenta el grado de confusión que parezca tener la lógica del programa.

La habilidad para vigilar el software mientras se está ejecutando en tiempo real, confiere claramente un gran beneficio al bloqueador de acciones; sin embargo, también tiene inconvenientes. Debido a que el código dañino tiene ciertamente que ejecutarse en la máquina elegida antes de que todos sus comportamientos puedan ser identificados, puede causar un gran daño al sistema antes de que haya sido detectado y paralizado por el sistema de bloqueo. Por ejemplo, un nuevo virus podría desordenar un número de archivos aparentemente poco importantes del disco duro antes de infectar un solo archivo y ser bloqueado. Incluso aunque la infección real fuera bloqueada, sería posible que el usuario no pudiera localizar sus archivos, causando una pérdida en la productividad o incluso algo peor.

10.3 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

Para una mayor comprensión de los virus, resulta útil leer el libro [HARL01]. En [CASS01], [FORR97], [KEPH97], y [NACH97] se ofrecen artículos con interesantes puntos de vista sobre el tema de los virus y los gusanos. Un buen tratamiento del gusano Código Rojo se encuentra en [MEIN01].

- CASS01** Cass, S. «Anatomy of Malice». *IEEE Spectrum*, noviembre de 2001.
- FORR97** Forrest, S.; Hofmeyr, S.; y Somayaji, A. «Computer Immunology». *Communications of the ACM*, octubre de 1997.
- HARL01** Harley, D.; Slade, R.; y Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.
- KEPH97** Kephart, J.; Sorkin, G.; Chess, D.; y White, S. «Fighting Computer Viruses». *Scientific American*, noviembre de 1997.
- MEIN01** Meinel, C. «Code Red for the Web». *Scientific American*, octubre de 2001.
- NACH97** Nachenberg, C. «Computer Virus-Antivirus Coevolution». *Communications of the ACM*, enero de 1997.

Sitio Web recomendado:

- **AntiVirus On-line:** sitio de IBM con información sobre virus; es uno de los mejores.

10.4 TÉRMINOS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

TÉRMINOS CLAVE

Bomba lógica	<i>Software dañino (malware)</i>	Virus de macro
caballo de Troya	Trampa	Virus furtivo
gusano	virus	Virus polimórfico
Sistema de inmunidad digital	Virus de correo electrónico	zombi

PREGUNTAS DE REPASO

- 10.1.** Define brevemente cada tipo de *malware* que se muestra en la Figura 10.1.
- 10.2.** ¿Cuál es el papel que desempeña la compresión en el funcionamiento de un virus?
- 10.3.** ¿Cuál es el papel que desempeña el cifrado en el funcionamiento de un virus?
- 10.4.** ¿Cuáles son las fases habituales de la operación de un virus o un gusano?
- 10.5.** En términos generales, ¿cómo se propaga un gusano?
- 10.6.** ¿Qué es un sistema de inmunidad digital?
- 10.7.** ¿Cómo funciona el *software* de bloqueo de acciones?

PROBLEMAS

- 10.1.** Hay un fallo en el programa del virus de la Figura 10.2. ¿Cuál es?
- 10.2.** Cabe preguntarse si es posible desarrollar un programa que pueda analizar una parte de *software* para determinar si es un virus. Consideremos que tenemos un programa D que se supone que puede hacerlo. Es decir, para cualquier programa P, si ejecutamos D(P), el resultado obtenido es VERDADERO (P es un virus) o (FALSO) (P no es un virus). Ahora consideremos el siguiente programa:

```

Program CV :=
  ...
main-program :=
  {if D(CV) then goto next;
   else infect-executable;
  }
next:
}

```

En el programa expuesto, «infect-executable» es un módulo que explora la memoria en busca de programas ejecutables y, luego, se replica en ellos. Determina si D puede decidir correctamente si CV es un virus.

CAPÍTULO 11

Cortafuegos

11.1 Principios de diseño de cortafuegos

Características de los cortafuegos

Tipos de cortafuegos

Configuraciones de cortafuegos

11.2 Sistemas de confianza

Control de acceso a los datos

El concepto de sistema de confianza

Defensa frente a caballos de Troya

11.3 Bibliografía y sitios web recomendados

11.4 Palabras clave, preguntas de repaso y problemas

Palabras clave

Preguntas de repaso

Problemas

La función de una posición fuerte es hacer que las fuerzas que la sostienen sean inexpugnables.

De la guerra, CARL VON CLAUSEWITZ

El día en que tomes el mando, bloquea las fronteras, destruye aquello que te comprometa y para el paso de todos los emisarios.

El arte de la guerra, SUN TZU

Los cortafuegos pueden ser un medio eficaz de protección de un sistema o red local frente a las amenazas de seguridad provenientes de la red, mientras que al mismo tiempo proporcionan acceso al exterior mediante redes de área ancha e Internet.

Este capítulo comienza con una descripción general de los principios de funcionalidad y diseño de los cortafuegos. Después, se tratará el aspecto de la seguridad del propio cortafuegos y, en particular, el concepto de sistema de confianza, o sistema operativo seguro.

11.1 PRINCIPIOS DE DISEÑO DE CORTAFUEGOS

Los sistemas de información de corporaciones, agencias gubernamentales y otras instituciones han experimentado una evolución constante:

- Sistemas de procesamiento de datos centralizados, con un gran computador central (*mainframe*) al que se conectan directamente una serie de terminales.
- Redes de área local (LAN) que conectan computadores personales y terminales entre ellos y con el computador central.
- Red del edificio de una empresa, compuesta de varias LAN, que conecta entre sí a computadores personales, servidores y quizás uno o dos computadores centrales.
- Red de toda la empresa, formada por múltiples redes corporativas distribuidas geográficamente y que se conectan entre sí mediante una red privada de área ancha (WAN).
- Conectividad con Internet, por la cual todas las redes corporativas se enganchan a Internet y también podrían o no estar conectadas por una WAN privada.

Para la mayoría de las organizaciones, la conexión a Internet ha dejado de ser una opción para convertirse en una necesidad, debido a que la información y los servicios disponibles les son esenciales. Incluso, usuarios individuales dentro de la organización quieren y necesitan acceso a Internet, y si esto no se les suministra mediante su LAN, utilizarán la capacidad de conexión telefónica de su PC con un proveedor de servicio de Internet (ISP, *Internet Service Provider*). Sin embargo, mientras el acceso a Internet beneficia a la organización, también constituye una amenaza, ya que permite al mundo exterior llegar hasta la red local e interactuar con ella. A pesar de que es posible equipar a cada estación de trabajo y a cada servidor de la red corporativa con fuertes característi-

cas de seguridad, como por ejemplo protección de intrusos, este enfoque no es práctico. Consideremos una red con cientos o miles de sistemas, funcionando con una mezcla de distintas versiones de *UNIX*, más Windows. Cuando se descubre un fallo de seguridad, cada sistema que pueda estar afectado debe ser actualizado para corregir el fallo. La alternativa, cada vez más aceptada, es el cortafuegos. El cortafuegos se inserta entre la red corporativa e Internet para establecer un enlace controlado y levantar un muro de seguridad exterior o perímetro. El objetivo de este perímetro es proteger la red corporativa de ataques procedentes de Internet y proporcionar un único punto de resistencia donde implantar la seguridad y la auditoría. El cortafuegos puede ser un único sistema o un conjunto de dos o más sistemas que cooperan para realizar la función de cortafuegos.

En esta sección, se observan primero las características generales de los cortafuegos. Despues se tratarán los tipos de cortafuegos de uso más común en la actualidad. Finalmente, se examinarán algunas de las configuraciones más habituales de cortafuegos.

CARACTERÍSTICAS DE LOS CORTAFUEGOS

[BELL94] enumera los siguientes objetivos de diseño para un cortafuegos:

1. Todo el tráfico desde el interior hacia el exterior, y viceversa, debe pasar a través del cortafuegos. Esto se consigue bloqueando físicamente todos los accesos a la red local excepto a través del cortafuegos. Hay diferentes configuraciones, como se explicará más tarde en esta sección.
2. Se permitirá pasar solamente el tráfico autorizado, definido por la política de seguridad local. Se utilizan diferentes tipos de cortafuegos que implementan diferentes tipos de políticas de seguridad, como se explicará más tarde en esta sección.
3. El propio cortafuegos es inmune a la penetración. Esto implica que utiliza un sistema de confianza con un sistema operativo seguro. Este tema se discute en la sección 11.2.

[SMIT97] presenta cuatro técnicas generales que utilizan los cortafuegos para controlar el acceso e imponer la política de seguridad del sitio. Originalmente, los cortafuegos se centraban principalmente en el control de servicios, pero han ido evolucionando para proporcionar las cuatro técnicas:

- **Control de servicio:** determina los tipos de servicios de Internet a los que se puede acceder, interna o externamente. El cortafuegos puede filtrar el tráfico basándose en las direcciones IP y el número de puertos TCP; puede proporcionar *software de proxy* que reciba e interprete cada solicitud de servicio antes de permitir su paso; o puede alojar *software* del servidor como, por ejemplo, un servicio de correo o de web.
- **Control de dirección:** determina en qué dirección se pueden iniciar las solicitudes de servicios particulares y en qué dirección se les permite el paso a través del cortafuegos.
- **Control de usuario:** controla el acceso a un servicio en función de qué usuario esté intentando acceder a él. Esta característica se aplica normalmente a usuarios que se hallan dentro del perímetro del cortafuegos (usuarios locales). También puede aplicarse al tráfico entrante desde usuarios externos; pero esto último re-

quiere algún tipo de tecnología de autentificación segura como, por ejemplo, la que proporciona IPSec (Capítulo 6).

- **Control de comportamiento:** controla cómo se utilizan los servicios particulares. Por ejemplo, el cortafuegos puede filtrar el correo electrónico para eliminar el «correo basura» (*spam*), o puede permitir el acceso externo sólo a una parte de la información de un servidor web local.

Antes de proceder a los detalles de los tipos y configuraciones de cortafuegos, es mejor resumir qué es lo que podemos esperar de un cortafuegos. Las siguientes capacidades están dentro de sus competencias:

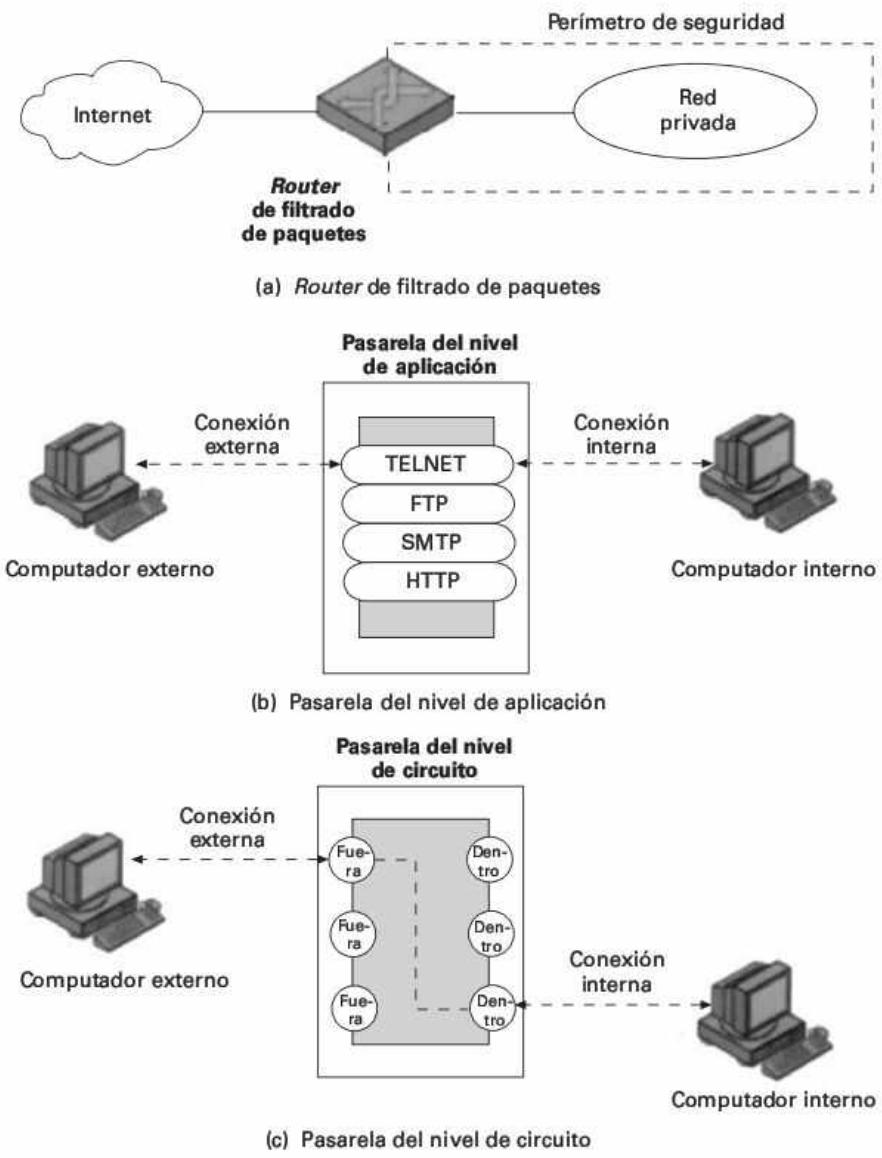
1. Un cortafuegos define un punto único de resistencia que mantiene a los usuarios no autorizados fuera de la red protegida, prohíbe la entrada o salida de la red de servicios potencialmente vulnerables y proporciona protección frente a distintos tipos de ataques de suplantación de IP (*spoofing*) y de enrutamiento. El uso de un solo punto de resistencia simplifica la gestión de la seguridad porque las capacidades de seguridad se consolidan en un único sistema o conjunto de sistemas.
2. Un cortafuegos proporciona una ubicación para supervisar sucesos relacionados con la seguridad. En los sistemas cortafuegos se pueden implementar auditorías y alarmas.
3. Un cortafuegos es una plataforma adecuada para distintas funciones de Internet que no están relacionadas con la seguridad. Entre éstas se incluye un traductor de direcciones de red, que hace corresponder direcciones locales con direcciones de Internet, y una función de gestión de red que audita o registra el uso de Internet.
4. Un cortafuegos puede servir como plataforma para IPSec. Utilizando la capacidad del modo túnel descrita en el Capítulo 6, el cortafuegos puede utilizarse para implementar redes privadas virtuales.

Sin embargo, los cortafuegos tienen sus limitaciones:

1. El cortafuegos no puede proteger contra ataques que no se produzcan a través del cortafuegos. De hecho, los sistemas internos pueden tener capacidad de conexión telefónica para conectarse a un ISP, y una LAN interna puede disponer de una batería de módems que ofrezca capacidad de conexión telefónica con dicha LAN para empleados que viajen y trabajadores a distancia.
2. El cortafuegos no protege contra amenazas internas (un empleado disgustado o un empleado que inconscientemente coopera con un atacante externo).
3. El cortafuegos no puede proteger contra la transferencia de programas o ficheros infectados por virus. Debido a la variedad de sistemas operativos y aplicaciones existentes dentro del perímetro, sería poco práctico y quizás imposible que el cortafuegos explorase todos los ficheros, correo electrónico y mensajes entrantes en busca de virus.

TIPOS DE CORTAFUEGOS

La Figura 11.1 muestra los tres tipos de cortafuegos más comunes: filtros de paquetes, pasarelas del nivel de aplicación y pasarelas del nivel de circuito. A continuación examinaremos cada uno de ellos.

**Figura 11.1** Tipos de cortafuegos

Router de filtrado de paquetes

Un *router* de filtrado de paquetes aplica un conjunto de reglas a cada paquete IP y entonces retransmite o descarta dicho paquete. El *router*, normalmente, se configura para filtrar paquetes que van en ambas direcciones (desde y hacia la red interna). Las reglas de filtrado se basan en la información contenida en un paquete de red.

- **Dirección IP de origen:** la dirección IP del sistema que originó el paquete IP (por ejemplo, 192.168.1.1).

- **Dirección IP de destino:** la dirección IP del sistema al que intenta llegar el paquete IP (por ejemplo, 192.168.1.2).
- **Direcciones del nivel de transporte de origen y destino:** el número de puerto del nivel de transporte (por ejemplo, TCP o UDP), que define aplicaciones como SNMP o TELNET.
- **Campo de protocolo IP:** define el protocolo de transporte.
- **Interfaz:** para un *router* con tres o más interfaces, indica desde qué interfaz viene el paquete o hacia cuál va destinado.

El filtrador de paquetes normalmente se configura como una lista de reglas basadas en correspondencias con los campos en la cabecera IP o TCP. Si hay una correspondencia en una de las reglas, se invoca esa regla para determinar si se retransmite o se descarta el paquete. Si no hay correspondencia con ninguna regla, entonces se lleva a cabo una acción predeterminada. Hay dos políticas predeterminadas posibles:

- **Default = discard** lo que no está expresamente permitido está prohibido.
- **Default = forward** lo que no está expresamente prohibido está permitido.

La política de descartar por defecto (*default = discard*) es más conservadora. Inicialmente, todo está bloqueado, y los servicios deben añadirse en función de cada caso. Esta política es más visible a los usuarios, quienes probablemente verán el cortafuegos como un estorbo. La política de retransmitir por defecto (*default = forward*) aumenta la facilidad de uso para los usuarios finales, pero proporciona escasa seguridad; el administrador de seguridad debe, básicamente, reaccionar ante las nuevas amenazas de seguridad a medida que se van descubriendo.

La Tabla 11.1, extraída de [BELL94], muestra algunos ejemplos de grupos de reglas de filtrado de paquetes. En cada uno, las reglas se aplican de arriba hacia abajo. El «*» en un campo es un denominador comodín que se corresponde con cualquier cosa. Se asume que la política impuesta es *default = discard*.

- A) Se permite correo entrante (el puerto 25 es para SMTP entrante), pero sólo a una pasarela. Sin embargo, el correo desde el computador externo, SPIGOT, se bloquea porque tiene un historial que refleja el envío de ficheros masivos en los mensajes de correo electrónico.
- B) Esta es una instrucción explícita de política por defecto. Todos los conjuntos de reglas incluyen implícitamente esta regla como la última.
- C) Esta regla está destinada a especificar que cualquier computador interno puede enviar correo hacia el exterior. Un paquete TCP con puerto de destino 25 se encamina hacia el servidor SMTP en la máquina destino. El problema de esta regla es que el uso del puerto 25 para recepción de SMTP es sólo una opción por defecto; una máquina externa podría estar configurada para tener alguna otra aplicación asociada al puerto 25. Como está escrita esta regla, un atacante podría conseguir acceso a las máquinas internas enviando paquetes con el número de puerto TCP de origen 25.
- D) Estas reglas consiguen el resultado pretendido pero no conseguido en C. Las reglas se benefician de una característica de las conexiones TCP. Una vez que se ha establecido una conexión, se activa el indicador ACK del segmento TCP para reconocer los segmentos enviados desde el otro lado. Por tanto, estas reglas establecen que se permiten los paquetes IP cuya dirección IP de origen sea una de

Tabla 11.1. Ejemplos de filtrado de paquetes

	acción	nuestro host	puerto	otro host	puerto	comentario	
A	bloquear	*	*	SPIGOT	*	no confiamos en esas personas	
	permitir	OUR-GW	25	*	*	conexiones a nuestro puerto SMTP	
B	acción	nuestro host	puerto	otro host	puerto	comentario	
	bloquear	*	*	*	*	por defecto	
C	acción	nuestro host	puerto	otro host	puerto	comentario	
	permitir	*	*	*	25	conexiones a puerto SMTP de otros host	
	acción	origen	puerto	destino	puerto	indicador	comentario
D	permitir	{nuestros hosts}	*	*	25		nuestros paquetes a puerto SMTP de otros hosts
	permitir	*	25	*	*	ACK	sus respuestas
	acción	origen	puerto	destino	puerto	indicador	comentario
E	permitir	{nuestros hosts}	*	*	*		nuestras llamadas salientes
	permitir	*	*	*	*	ACK	respuestas a nuestras llamadas
	permitir	*	*	*	>1024		tráfico hacia sistemas no servidores

las de la lista de los computadores internos designados y cuyo número de puerto TCP de destino sea 25. También permiten paquetes entrantes cuyo número de puerto de origen sea 25 y que incluya el indicador ACK en el segmento TCP. Obsérvese que designamos explícitamente los sistemas origen y destino para definir esas reglas.

- E**) Este conjunto de reglas es un enfoque para manejar conexiones FTP. Con FTP se usan dos conexiones TCP: una conexión de control para preparar y solicitar la transferencia de fichero y una conexión de datos para la transferencia real del mismo. La conexión de datos utiliza un número de puerto diferente asignado dinámicamente para la transferencia. La mayoría de los servidores y, por tanto, la mayoría de los objetivos de ataques, se encuentran en números bajos de puerto; la mayor parte de llamadas salientes tienden a usar números altos de puerto, normalmente por encima del 1023. Por tanto, este conjunto de reglas permiten

- Paquetes originados internamente
- Paquetes de respuesta a una conexión iniciada por una máquina interna
- Paquetes destinados a un puerto con número alto de una máquina interna

Este esquema necesita que los sistemas se configuren de manera que solamente estén en uso los números de puerto apropiados.

El conjunto de reglas E refleja la dificultad de tratar con aplicaciones cuando se usa el filtrado de paquetes. Otra forma de tratar con FTP y aplicaciones similares es mediante una pasarela del nivel de aplicación, que se describirá más tarde en esta sección.

Una de las ventajas de un *router* de filtrado de paquetes es su simplicidad. Además, los filtradores de paquetes generalmente son transparentes a los usuarios y son muy rápidos. [WACK02] enumera las siguientes debilidades de los cortafuegos filtradores de paquetes:

- Debido a que los cortafuegos filtradores de paquetes no examinan los datos de niveles superiores, no pueden evitar ataques que emplean vulnerabilidades o funciones específicas de la aplicación. Por ejemplo, un cortafuegos filtrador de paquetes no puede bloquear comandos específicos de una aplicación; si el cortafuegos permite una aplicación dada, entonces todas las funciones disponibles dentro de dicha aplicación estarán permitidas.
- Debido a que el cortafuegos dispone de información limitada, la funcionalidad de registro (*logging*) presente en los cortafuegos filtradores de paquetes también es limitada. Los archivos de registro del filtrador de paquetes contienen normalmente la misma información utilizada para la toma de decisiones de control de acceso (dirección de origen, dirección de destino y tipo de tráfico).
- La mayoría de los cortafuegos filtradores de paquetes no admiten esquemas avanzados de autenticación de usuario. Una vez más, esta limitación se debe sobre todo a la falta de funcionalidad del cortafuegos respecto a niveles superiores.
- Generalmente, son vulnerables a ataques de seguridad y a programas (*exploits*) que se aprovechan de problemas en la especificación y en la pila de protocolos TCP/IP, como por ejemplo suplantación de dirección del nivel de red (*spoofing*).

La mayoría de los cortafuegos filtradores de paquetes no pueden detectar un paquete de red en el cual se ha alterado la información de direccionamiento del nivel 3 de OSI. Los ataques de suplantación de IP (*spoofing*) son utilizados por los intrusos para traspasar los controles de seguridad implementados en un cortafuegos.

- Finalmente, debido al pequeño número de variables utilizadas en las decisiones de control de acceso, los cortafuegos filtradores de paquetes son susceptibles de tener agujeros de seguridad causados por una configuración inadecuada. Es decir, es fácil configurar accidentalmente un cortafuegos filtrador de paquetes para permitir tipos de tráfico, orígenes y destinos que deberían ser denegados en función de la política de seguridad de la organización.

A continuación se presentan algunos de los ataques que pueden hacerse a los *routers* de filtrado de paquetes y las contramedidas apropiadas:

- **Suplantación de dirección IP:** el intruso transmite paquetes desde el exterior con un campo de dirección IP origen que contiene una dirección de un computador interno. El atacante espera que el uso de una dirección «robada» le permita la penetración de sistemas que emplean seguridad simple de dirección de origen, en la cual se aceptan paquetes desde determinados computadores internos de confianza. La medida para evitarlo es descartar los paquetes con una dirección origen interna si el paquete llega por una interfaz externa.
- **Ataques de enrutamiento en origen:** la estación de origen especifica la ruta que debería seguir un paquete mientras pasa a través de Internet, con la esperanza de

que el paquete evite las medidas de seguridad en las que no se analiza la información de enrutamiento en origen. La contramedida consiste en descartar todos los paquetes que usan esta opción.

- **Ataques de fragmento pequeño:** el intruso utiliza una opción de fragmentación para crear fragmentos extremadamente pequeños y forzar que la información de cabecera de TCP se ubique en un fragmento de paquete separado. Este ataque está diseñado para esquivar las reglas de filtrado que dependen de la información en la cabecera de TCP. El atacante espera que el *router* de filtrado examine solamente el primer fragmento y que los restantes se pasen sin analizar. Este ataque se puede frustrar descartando todos los paquetes donde el tipo de protocolo sea TCP y el campo de desplazamiento de fragmento IP (*IP Fragment Offset*) sea igual a uno.

Cortafuegos de inspección de estado

Un filtrador de paquetes tradicional toma decisiones de filtrado basadas en un paquete individual y no tiene en cuenta ningún contexto del nivel más alto. Para entender qué se entiende por contexto y por qué un filtrador de paquetes tradicional está limitado con respecto al contexto, es necesario hacer un pequeño repaso. La mayoría de las aplicaciones estandarizadas que funcionan encima de TCP siguen un modelo cliente/servidor. Por ejemplo, para SMTP (*Simple Mail Transfer Protocol*), el correo electrónico se transmite de un sistema cliente a un sistema servidor. El sistema cliente genera nuevos mensajes de correo electrónico, normalmente introducidos por usuarios. El sistema servidor acepta mensajes de correo electrónico entrantes y los aloja en los buzones de usuario adecuados. SMTP funciona estableciendo una conexión TCP entre cliente y servidor, en la que el número de puerto TCP del servidor, que identifica a la aplicación servidor SMTP, es 25. El número de puerto TCP para el cliente SMTP es un número entre 1024 y 16383 generado por el cliente SMTP.

En general, cuando una aplicación que utiliza TCP crea una sesión con un computador remoto, ésta crea una conexión TCP en la que el número de puerto TCP de la aplicación remota (el servidor) es un número menor que 1024 y el número de puerto TCP de la aplicación local (el cliente) es un número entre 1024 y 16383. Los números menores que 1024 son los números de puerto «conocidos» y están asignados permanentemente a aplicaciones particulares (por ejemplo, 25 para servidor SMTP). Los números de puerto entre 1024 y 16383 se generan dinámicamente y tienen significado temporal solamente durante el período de tiempo en que está establecida una conexión TCP.

Un cortafuegos simple de filtrado de paquetes debe permitir el tráfico de red entrante con números de puerto altos para que el tráfico TCP se pueda producir. Esto crea una vulnerabilidad que podrían explotar usuarios no autorizados.

Un filtrador de paquetes de inspección de estado aumenta las restricciones de las reglas para el tráfico TCP creando un directorio de conexiones TCP salientes, como se muestra en la Tabla 11.2. Hay una entrada para cada conexión actualmente establecida. El filtrador de paquetes permitirá ahora el tráfico entrante a puertos de número alto solamente a aquellos paquetes que encajan en el perfil de una de las entradas de este directorio.

Tabla 11.2 Ejemplo de una tabla de estado de conexiones de un cortafuegos de inspección de estado [WACK02]

Dirección origen	Puerto origen	Dirección destino	Puerto destino	Estado de la conexión
192.168.1.100	1030	210.9.88.29	80	Establecida
192.168.1.102	1031	216.32.42.123	80	Establecida
192.168.1.101	1033	173.66.32.122	25	Establecida
192.168.1.106	1035	177.231.32.12	79	Establecida
223.43.21.231	1990	192.168.1.6	80	Establecida
219.22.123.32	2112	192.168.1.6	80	Establecida
210.99.212.18	3321	192.168.1.6	80	Establecida
24.102.32.23	1025	192.168.1.6	80	Establecida
223.212.212	1046	192.168.1.6	80	Establecida

Pasarela del nivel de aplicación

Una pasarela del nivel de aplicación, también llamada servidor *proxy*, actúa como un repetidor del tráfico del nivel de aplicación (Figura 11.1b). El usuario contacta con la pasarela utilizando una aplicación TCP/IP como, por ejemplo, Telnet o FTP, y la pasarela solicita al usuario el nombre del computador remoto al que desea acceder. Cuando el usuario responde y proporciona un identificador de usuario e información de autenticación válidos, la pasarela contacta con la aplicación en el computador remoto y retransmite los segmentos TCP que contienen los datos de aplicación entre los dos puntos finales. Si la pasarela no implementa el código *proxy* de una aplicación específica, entonces el servicio no está permitido y no puede atravesar el cortafuegos. Además, la pasarela puede configurarse para permitir solamente algunas características específicas de una aplicación que el administrador de red considere aceptables mientras que deniega las otras características.

Las pasarelas del nivel de aplicación tienden a ser más seguras que los filtros de paquetes. En vez de intentar tratar con numerosas combinaciones posibles que se van a permitir y prohibir en el nivel TCP y en el IP, la pasarela del nivel de aplicación necesita solamente escrutar unas pocas aplicaciones permitidas. Además, es fácil registrar y auditar todo el tráfico entrante del nivel de aplicación.

La desventaja principal de este tipo de pasarelas es la sobrecarga de procesamiento adicional en cada conexión. En efecto, hay dos conexiones enlazadas entre los usuarios finales, con la pasarela como punto de enlace, y la pasarela debe examinar y reenviar todo el tráfico en ambas direcciones.

Pasarela del nivel de circuito

Un tercer tipo de cortafuegos es la pasarela del nivel de circuito (Figura 11.1c). Ésta puede ser un sistema autónomo o puede ser una función especializada realizada por una pasarela del nivel de aplicación para ciertas aplicaciones. Una pasarela del nivel de circui-

to no permite una conexión TCP extremo a extremo; en vez de eso, la pasarela establece dos conexiones TCP, una entre ella y un usuario TCP en un computador interno, y otra entre ella y un usuario TCP en un computador externo. Una vez se han establecido las dos conexiones, la pasarela normalmente retransmite segmentos TCP desde una conexión hacia la otra sin examinar los contenidos. La función de seguridad consiste en determinar qué conexiones serán permitidas.

Un uso común de una pasarela del nivel de circuito se da en una situación en la que el administrador del sistema confía en los usuarios internos. La pasarela se puede configurar para permitir servicio del nivel de aplicación o *proxy* para conexiones entrantes y funciones del nivel de circuito para conexiones salientes. En esta configuración, la pasarela puede sufrir de la sobrecarga del procesamiento necesario para examinar datos de aplicaciones entrantes para funciones prohibidas, pero no sufre de esa sobrecarga para los datos salientes.

Un ejemplo de implementación de pasarela del nivel de circuito es el paquete SOCKS [KOB92]; la versión 5 de SOCKS se define en el RFC 1928. El RFC define SOCKS de la siguiente manera:

El protocolo descrito aquí está diseñado para proporcionar un marco para aplicaciones cliente/servidor en los dominios TCP y UDP para utilizar los servicios de un cortafuegos de manera conveniente y segura. El protocolo, conceptualmente, es un «nivel cuña» entre el nivel de aplicación y el nivel de transporte, y como tal no proporciona servicios de pasarela del nivel de red, como por ejemplo retransmisión de mensajes ICMP.

Los componentes de SOCKS son los siguientes:

- El servidor SOCKS, que se ejecuta en un cortafuegos basado en UNIX.
- La librería del cliente SOCKS, que se implanta en los computadores internos protegidos por el cortafuegos.
- Versiones adaptadas a SOCKS de los diferentes programas cliente estándar, como FTP y TELNET. La implementación del protocolo SOCKS normalmente implica la recopilación o el reenlazado de las aplicaciones cliente basadas en TCP para que utilicen adecuadamente las rutinas de encapsulado de la librería de SOCKS.

Cuando un cliente basado en TCP desea establecer una conexión con un objeto al que sólo se puede acceder a través de un cortafuegos (tal determinación depende de la implementación), dicho cliente debe abrir una conexión TCP con el puerto SOCKS apropiado del sistema servidor de SOCKS. El servicio SOCKS está ubicado en el puerto TCP 1080. Si la solicitud de conexión tiene éxito, el cliente entabla una negociación para decidir el método de autenticación que se va a utilizar, se autentifica con el método elegido y luego envía una solicitud de retransmisión. El servidor SOCKS evalúa la solicitud y o bien establece la conexión apropiada o bien la deniega. Los intercambios UDP se manejan de manera similar. Básicamente, se abre una conexión TCP para autenticar a un usuario para que envíe y reciba segmentos UDP, y éstos serán retransmitidos mientras la conexión TCP permanezca abierta.

Computador bastión

Un computador bastión es un sistema identificado por el administrador del cortafuegos como un punto fuerte crucial de la seguridad de la red. Normalmente, sirve como base

para implantar una pasarela del nivel de aplicación o del nivel de circuito. Las características comunes de un computador bastión son las siguientes:

- La plataforma *hardware* del computador bastión ejecuta una versión segura de su sistema operativo, convirtiéndolo en un sistema de confianza.
- Solamente se instalarán los servicios que el administrador de la red considere esenciales. Entre ellos se incluyen aplicaciones *proxy* como, por ejemplo, Telnet, DNS, FTP, SMTP y autenticación de usuarios.
- El computador bastión podría solicitar autenticación adicional antes de permitir el acceso de un usuario a los servicios *proxy*. Además, cada servicio *proxy* podría solicitar su propia autenticación antes de conceder acceso al usuario.
- Cada *proxy* se configura para admitir solamente un subconjunto del conjunto de comandos de las aplicaciones estándar.
- Cada *proxy* se configura para permitir el acceso solamente a computadores específicos. Esto significa que el conjunto limitado de comandos/características podría aplicarse sólo a un subconjunto de sistemas de la red protegida.
- Cada *proxy* mantiene información de auditoría detallada registrando todo el tráfico, cada conexión y la duración de cada conexión. El registro de auditoría es una herramienta fundamental para descubrir y finalizar ataques de intrusos.
- Cada módulo *proxy* es un paquete de *software* muy pequeño diseñado específicamente para la seguridad de la red. Por su relativa simplicidad, es más fácil comprobar tales módulos para localizar agujeros de seguridad. Por ejemplo, una aplicación de correo normal de UNIX podría contener 20.000 líneas de código y un *proxy* de correo podría contener menos de 1.000.
- En el computador bastión, cada *proxy* es independiente de los otros. Si hay algún problema con la operación de un *proxy*, o si se descubre una vulnerabilidad futura, éste puede desinstalarse sin afectar a la operación de las otras aplicaciones *proxy*. También, si un número suficiente de usuarios solicita la implantación de algún nuevo servicio, el administrador de la red puede instalar fácilmente el *proxy* solicitado en el computador bastión.
- Generalmente, un *proxy* no realiza accesos a disco excepto para leer su fichero de configuración inicial. Esto dificulta que un intruso instale caballos de Troya para obtener información u otros ficheros dañinos en el computador bastión.
- Cada *proxy* se ejecuta como un usuario no privilegiado en un directorio privado y seguro del computador bastión.

CONFIGURACIONES DE CORTAFUEGOS

Además del uso de una configuración simple formada por un único sistema, como podría ser un único *router* de filtrado de paquetes o una única pasarela (Figura 11.1), son posibles configuraciones más complejas y más comunes. La Figura 11.2 muestra tres configuraciones habituales de cortafuegos. Examinaremos cada una de ellas.

En la configuración **cortafuegos de computador protegido, bastión de interfaz única** (Figura 11.2a), el cortafuegos se compone de dos sistemas: un *router* de filtrado de paquetes y un computador bastión. Normalmente, el *router* se configura de manera que

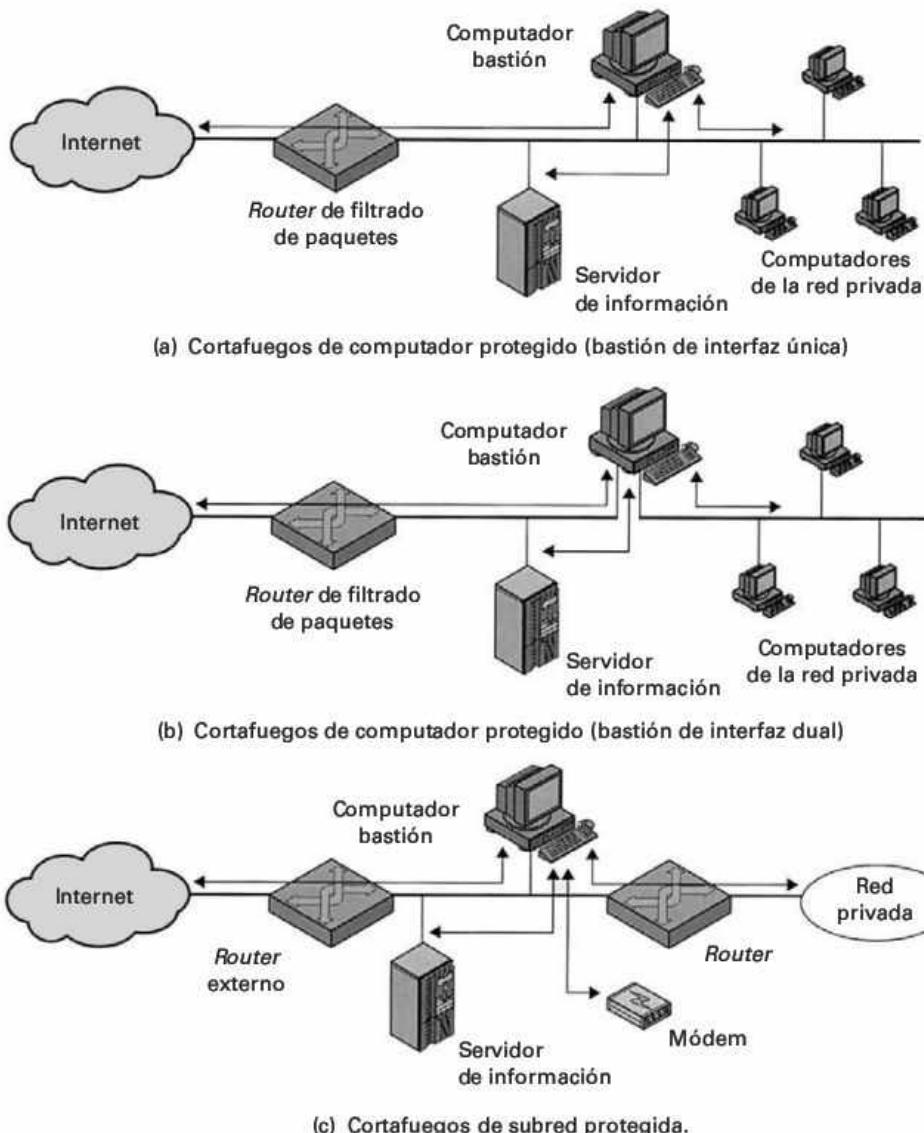


Figura 11.2 Configuraciones de cortafuegos

1. Para el tráfico desde Internet, sólo se permite la entrada de paquetes IP destinados al computador bastión.
2. Para el tráfico desde la red interna, sólo se permite la salida de paquetes IP procedentes del computador bastión.

El bastión realiza funciones de autentificación y de *proxy*. Esta configuración tiene mayor seguridad que un simple *router* de filtrado de paquetes o que una sola pasarela del nivel de aplicación, por dos razones. Primero, esta configuración implementa ambos filtros, el de paquetes y el del nivel de aplicación, permitiendo una flexibilidad considerable.

rable en la definición de políticas de seguridad. Segundo, un intruso generalmente debe penetrar dos sistemas separados antes de comprometer la seguridad de la red interna.

Esta configuración también provee flexibilidad para proporcionar acceso directo a Internet. Por ejemplo, la red interna podría incluir un servidor público de información como, por ejemplo, un servidor web, para el cual no se requiere un elevado grado de seguridad. En ese caso, el *router* se puede configurar para permitir el tráfico directo entre el servidor de información e Internet.

En la configuración de única interfaz del bastión que se acaba de describir, si se saquea el *router* de filtrado de paquetes, el tráfico podría fluir directamente a través del *router* entre Internet y otros computadores de la red privada. La configuración **corta-fuegos de computador protegido, bastión de interfaz dual** previene físicamente tal brecha de seguridad (Figura 11.2b). Las ventajas de los niveles duales de seguridad presentes en la configuración anterior también están presentes en ésta. Otra vez, puede permitirse comunicación directa con el *router* a un servidor de información o a otro computador, si esto se ajusta a la política de seguridad.

La configuración de **corta-fuegos de subred protegida** de la Figura 11.2c es la más segura de las que se han considerado. En esta configuración, se usan dos *routers* de filtrado de paquetes, uno entre el bastión e Internet y otro entre el bastión y la red interna. Esta configuración crea una subred aislada, que puede constar simplemente de un bastión, pero que también puede incluir uno o más servidores de información y módems para proporcionar acceso telefónico entrante. Normalmente, Internet y la red interna tienen acceso a los computadores de la subred protegida, pero el tráfico a través de esta subred está bloqueado. Esta configuración ofrece varias ventajas:

- Ahora hay tres niveles de defensa para frustrar a los intrusos.
- El *router* externo sólo revela a la Internet la existencia de la subred protegida; por lo que la red interna es invisible a la Internet.
- De igual forma, el *router* interno sólo revela a la red interna la existencia de la subred protegida; por lo tanto, los sistemas en la red interior no pueden construir rutas directas hacia la Internet.

11.2 SISTEMAS DE CONFIANZA

Una manera de mejorar la habilidad de un sistema para defenderse de intrusos y programas dañinos es implementar la tecnología de sistemas de confianza. Esta sección proporciona una breve descripción general de este asunto. Comenzaremos observando algunos conceptos básicos del control de acceso a los datos.

CONTROL DE ACCESO A LOS DATOS

Después de un inicio de sesión satisfactorio, al usuario se le concede acceso a uno o a un conjunto de computadores y de aplicaciones. Esto, generalmente, no es suficiente para un sistema que mantiene datos confidenciales en sus bases de datos. A través del procedimiento de control de acceso de usuario, éste puede ser identificado por el sistema. Puede haber un perfil asociado con cada usuario que especifique las operaciones y

accesos de ficheros permitidos. El sistema operativo puede entonces imponer las reglas en función del perfil de usuario. De todas formas, el sistema de gestión de bases de datos debe controlar el acceso a registros específicos o incluso a partes de los registros. Por ejemplo, puede ser permisible que alguien de la administración obtenga una lista del personal de la compañía, pero solamente algunos individuos seleccionados tendrían acceso a la información salarial. El asunto requiere un mayor nivel de detalle. Mientras que el sistema operativo puede conceder permiso a un usuario para acceder a un fichero o usar una aplicación, después de lo cual no hay más comprobaciones de seguridad, el sistema de gestión de bases de datos debe tomar una decisión para cada intento de acceso individual. Esta decisión dependerá no sólo de la identidad del usuario sino también de las partes específicas de los datos a los que se accede e, incluso, de la información que ya se ha proporcionado al usuario.

Un modelo general de control de acceso como el que realiza un sistema de gestión de ficheros o de bases de datos es el de una **matriz de acceso** (Figura 11.3a). Los elementos básicos del modelo son los siguientes:

- **Sujeto:** entidad capaz de acceder a objetos. Generalmente, el concepto de sujeto se equipara al de proceso. Cualquier usuario o aplicación consigue realmente acceso a un objeto por medio de un proceso que representa a ese usuario o a esa aplicación.
- **Objeto:** cualquier cosa cuyo acceso se controle como, por ejemplo, ficheros, partes de ficheros, programas y segmentos de memoria.
- **Derecho de acceso:** manera en que un sujeto accede a un objeto. Algunos ejemplos son lectura, escritura y ejecución.

Un eje de la matriz contiene los sujetos identificados que pueden intentar el acceso a los datos. Normalmente, esta lista estará formada por usuarios individuales o grupos de usuarios, aunque se podría controlar el acceso a terminales, computadores o aplicaciones en vez de, o además de, a los usuarios. El otro eje enumera los objetos a los que se podría acceder. Al máximo nivel de detalle, los objetos podrían ser campos de datos individuales. También podrían ser objetos de la matriz agrupamientos más amplios como, por ejemplo, registros, ficheros e, incluso, la base de datos completa. Cada entrada en la matriz indica los derechos de acceso de un sujeto para un objeto.

En la práctica, una matriz de acceso normalmente está dispersa y se implementa mediante descomposición, de una de dos maneras posibles. La matriz se puede descomponer por columnas produciendo **listas de control de acceso** (Figura 11.3b). Así, para cada objeto, una lista de control de acceso enumera usuarios y sus derechos de acceso permitidos; además, puede contener una entrada por defecto o pública. Esto permite que los usuarios que no estén explícitamente incluidos como poseedores de unos derechos especiales tengan un conjunto de derechos por defecto. Los elementos de la lista pueden incluir usuarios individuales así como grupos de usuarios.

La descomposición por columnas produce **tickets de capacidad** (Figura 11.3c). Un ticket de capacidad especifica los objetos y operaciones autorizados para un usuario. Cada usuario tiene una serie de tickets y podría estar autorizado para prestarlos o darlos a otros usuarios. Debido a que los tickets pueden dispersarse por todo el sistema, presentan un problema de seguridad mayor que las listas de control de acceso. En particular, los tickets no deben poder ser falsificados. Una manera de lograrlo es que el sistema operativo mantenga todos los tickets en nombre de los usuarios. Esos tickets tendrían que mantenerse en una región de memoria inaccesible a los usuarios.

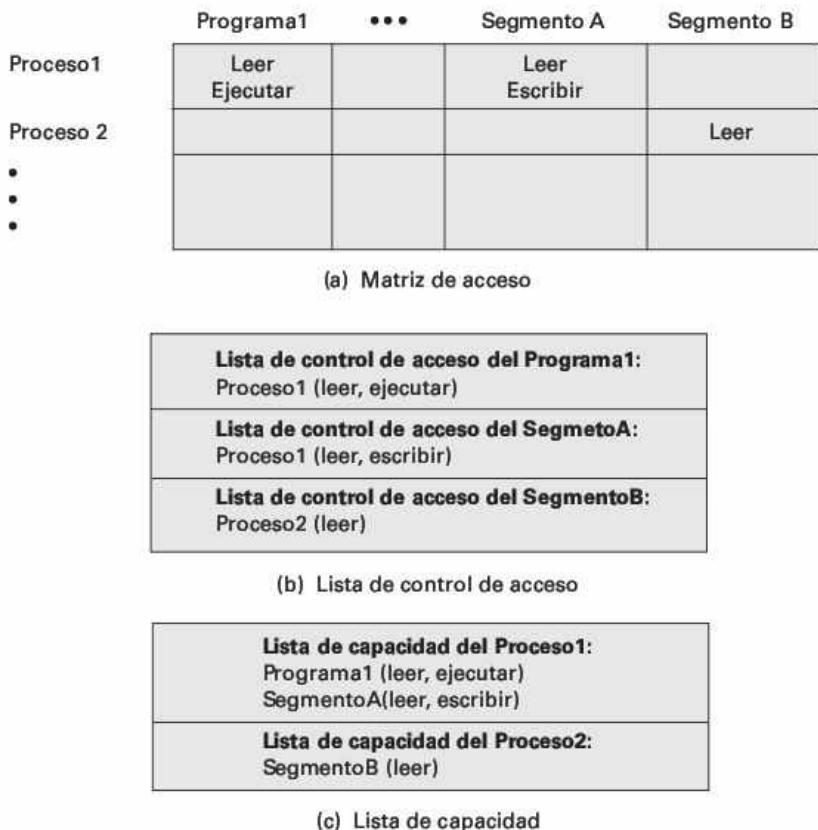


Figura 11.3 Estructura del control de acceso

EL CONCEPTO DE SISTEMA DE CONFIANZA

La mayor parte de lo discutido hasta el momento está relacionado con la protección de un mensaje o elemento frente a un ataque pasivo o activo por parte de un usuario dado. Un requisito ligeramente diferente pero muy aplicable es proteger datos o recursos en función de niveles de seguridad. Esto se suele encontrar en asuntos militares, donde la información se clasifica en no clasificada, confidencial, secreta, máximo secreto, etc. Este concepto se aplica igualmente en otras áreas, donde la información puede organizarse en grandes categorías y se les puede conceder a los usuarios autorización para acceder a ciertas categorías de datos. Por ejemplo, el nivel más alto de seguridad podría ser para documentos y datos de planificación estratégica de empresas, a los que sólo pueden acceder directivos de la empresa y su personal; a continuación podrían aparecer los datos financieros y de personal confidenciales, a los que accede únicamente el personal de administración, directivos, etc.

Cuando se definen múltiples categorías o niveles de datos, el requisito se conoce como **seguridad multinivel**. La regla general del requisito de seguridad multinivel consiste en que un sujeto de un alto nivel no puede transmitir información a un sujeto de un nivel inferior o no comparable a menos que ese flujo refleje con precisión la voluntad de un usuario autorizado. Para la implementación, este requisito consta de dos partes y se expresa de manera sencilla. Un sistema multinivel seguro debe imponer:

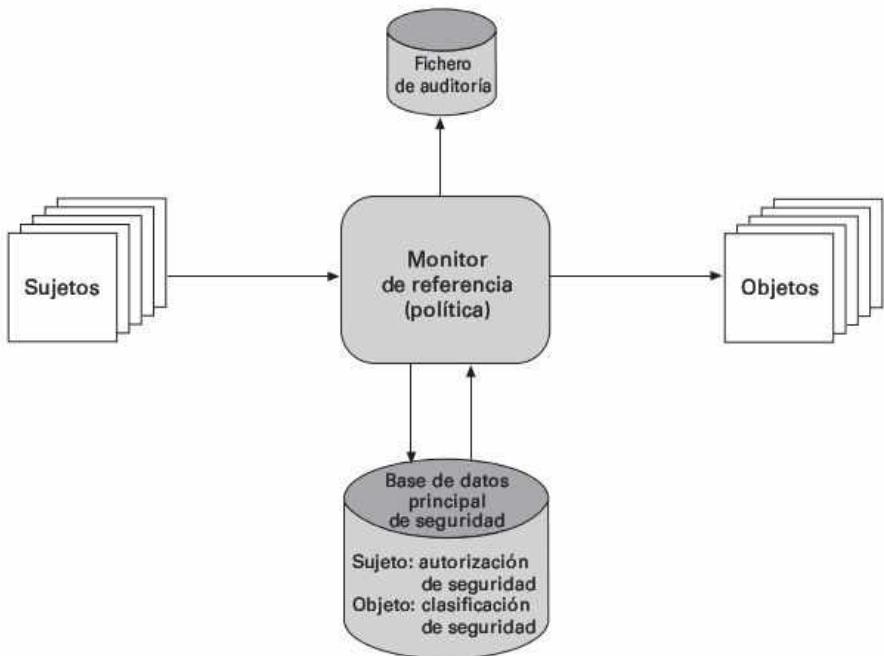


Figura 11.4 Concepto de monitor de referencia

- **No leer hacia arriba (no read up):** un sujeto solamente puede leer un objeto de un nivel de seguridad inferior o igual. A esto se le conoce como **propiedad de seguridad simple**.
- **No escribir hacia abajo (no write down):** un sujeto solamente puede escribir en un objeto de un nivel de seguridad superior o igual. A esto se le conoce como **propiedad.*¹** (que se lee *propiedad estrella*).

Estas dos reglas, si se aplican correctamente, proporcionan seguridad multinivel. Para un sistema de procesamiento de datos, el enfoque que se ha adoptado, y que ha sido objeto de investigación y desarrollo, se basa en el concepto de *monitor de referencia*. Este enfoque queda reflejado en la Figura 11.4. El monitor de referencia es un elemento de control en el *hardware* y en el sistema operativo de un computador que regula el acceso de sujetos a objetos en función de parámetros de seguridad del sujeto y del objeto. El monitor de referencia tiene acceso a un fichero, conocido como *base de datos fundamental de seguridad*, que enumera los privilegios de acceso (autorización de seguridad) de cada sujeto y los atributos de protección (nivel de clasificación) de cada objeto. El monitor de seguridad impone las reglas de seguridad (no leer hacia arriba, no escribir hacia abajo) y tiene las siguientes propiedades:

¹ El «*» no significa nada. Durante la redacción del primer informe sobre el modelo, a nadie se le ocurrió un nombre adecuado para la propiedad. El asterisco fue un carácter provisional introducido en el borrador para que un editor de texto pudiese encontrar y reemplazar rápidamente todas las instancias de su uso una vez que se diera nombre a la propiedad. A nadie se le ocurrió nunca un nombre, por lo que el informe se publicó con el «*» intacto.

- **Mediación completa:** las reglas de seguridad se aplican en cada acceso, no solamente, por ejemplo, cuando se abre un fichero.
- **Aislamiento:** el monitor de referencia y la base de datos están protegidos de modificaciones no autorizadas.
- **Verificabilidad:** el correcto funcionamiento del monitor de referencia debe ser demostrable. Es decir, debe ser posible demostrar matemáticamente que el monitor de referencia impone las reglas de seguridad y proporciona mediación completa y aislamiento.

Estos requisitos son consistentes. El requisito de mediación completa significa que debe mediarse para cada acceso a datos de memoria principal o de disco y de cinta magnética. Las implementaciones puramente *software* imponen una penalización demasiado alta de las prestaciones para que sea una opción práctica; la implementación debe ser, al menos parcialmente, *hardware*. El requisito de aislamiento significa que debe ser imposible que un atacante, no importa lo astuto que sea, cambie la lógica del monitor de referencia o los contenidos de la base de datos fundamental de seguridad. Finalmente, el requisito de demostración matemática es excelente para algo tan complejo como un computador de propósito general. A un sistema que puede proporcionar estos requisitos se le denomina **sistema de confianza**.

Un elemento final representado en la Figura 11.4 es el fichero de auditoría. En este fichero se almacenan los acontecimientos importantes de seguridad, como son las violaciones de seguridad detectadas y los cambios autorizados de la base de datos fundamental de seguridad.

El departamento de defensa de los Estados Unidos, en un esfuerzo por satisfacer sus propias necesidades y para ofrecer un servicio público, estableció en 1981 el Computer Security Center perteneciente a la Agencia de Seguridad Nacional (NSA, National Security Agency) con el objetivo de promover la expansión de la disponibilidad de sistemas de computadores de confianza. Este objetivo se realiza a través del programa de Evaluación de Productos Comerciales del centro. Básicamente, el centro intenta evaluar si los productos comerciales disponibles cumplen los requisitos de seguridad descritos. El centro clasifica los productos evaluados en función del rango de características de seguridad que proporcionan. Las evaluaciones son necesarias para las adquisiciones del Departamento de Defensa pero luego se publican y se encuentran disponibles gratuitamente. Por lo tanto, pueden servir como guías a clientes comerciales para la compra de equipos disponibles directamente en los comercios.

DEFENSA FRENTE A CABALLOS DE TROYA

Una forma de protegerse de ataques de caballos de Troya es utilizar un sistema operativo seguro y de confianza. La Figura 11.5 muestra un ejemplo. En este caso, se utiliza un caballo de Troya para evitar el mecanismo de seguridad estándar que emplea la mayor parte de los gestores de ficheros y sistemas operativos: la lista de control de acceso. En este ejemplo, un usuario llamado Benito interactúa a través de un programa con un fichero de datos que contiene una ristra de caracteres de una confidencialidad crítica «CPE170KS». El usuario Benito ha creado el fichero proporcionando permisos de lectura y escritura solamente a los programas que se ejecuten en su propio nombre: es decir, solamente los procesos que son propiedad de Benito pueden acceder al fichero.

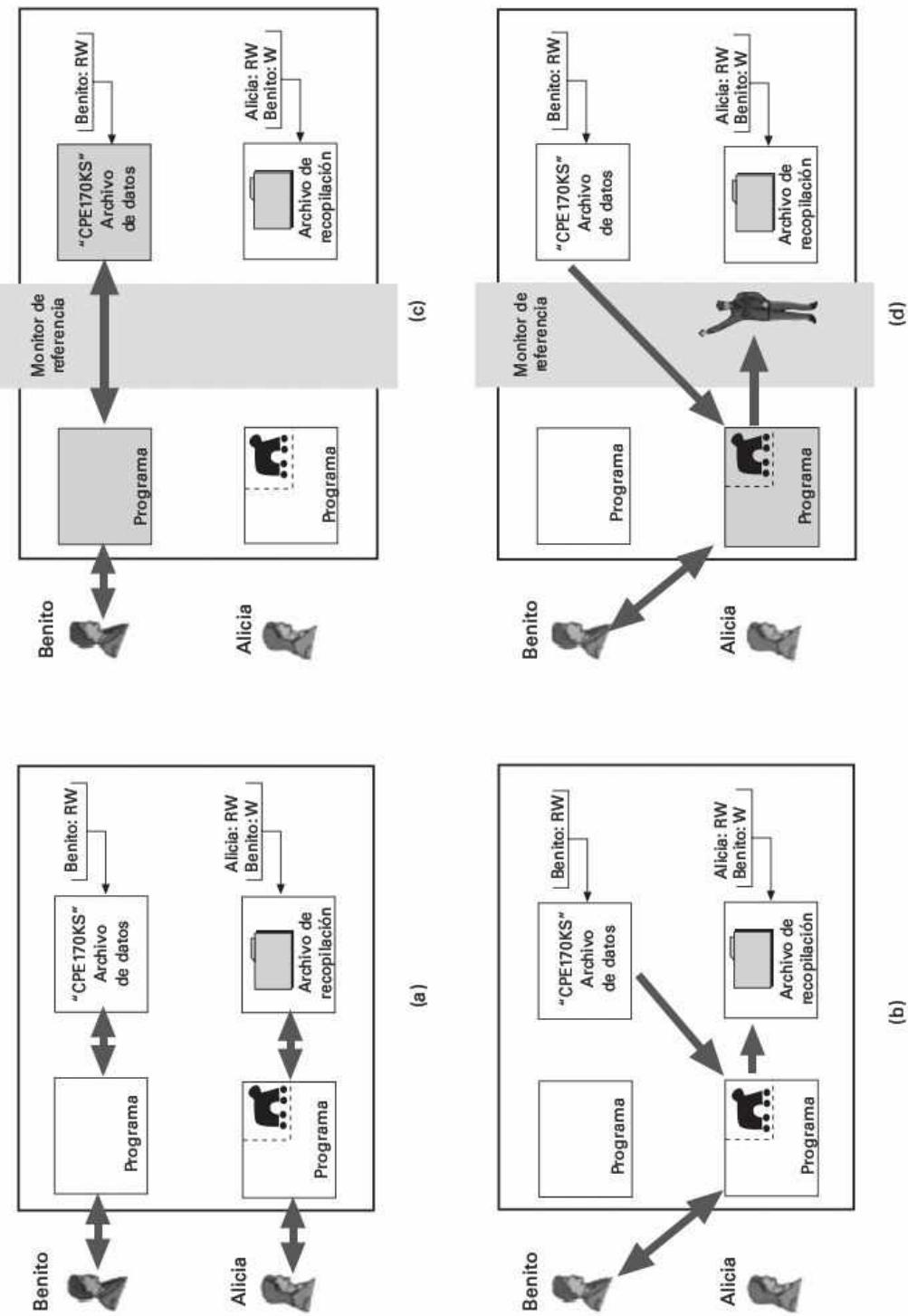


Figura 11.5 Caballo de Troya y sistema operativo seguro

El ataque del caballo de Troya comienza cuando un usuario hostil, llamado Alicia, obtiene acceso legítimo al sistema e instala un caballo de Troya y un fichero privado para utilizarlo en el ataque para almacenar información robada. Alicia se da a sí misma permiso de lectura y escritura para este fichero mientras que a Benito le da permiso de sólo escritura para el mismo fichero (Figura 11.5a). Alicia ahora induce a Benito para que invoque el programa caballo de Troya, quizás anunciándolo como una herramienta muy útil. Cuando el programa detecta que está siendo ejecutado por Benito, lee la ristra de caracteres confidencial del fichero de Benito y la copia en el fichero de recopilación de Alicia (Figura 11.5b). Ambas operaciones de lectura y escritura satisfacen las restricciones impuestas por las listas de control de acceso. Más tarde, Alicia solamente tiene que acceder al fichero de Benito almacenado en el archivo de recopilación para conocer el valor de la ristra.

Ahora considérese el uso de un sistema operativo seguro en este escenario (Figura 11.5c). Los niveles de seguridad se asignan a los sujetos en el momento de iniciar la sesión en función de criterios como desde qué terminal se accede al computador y qué usuario está implicado, identificado mediante identificador (ID) y contraseña. En este ejemplo, hay dos niveles de seguridad, confidencial y público, clasificados de manera que «confidencial» es de mayor nivel que «público». A los procesos y ficheros de datos propiedad de Benito se les asigna el nivel de seguridad confidencial. Los ficheros y procesos de Alicia están restringidos al nivel de seguridad público. Si Benito invoca el programa caballo de Troya (Figura 11.5d), el programa adquiere el nivel de seguridad de Benito. Por tanto, es capaz, bajo la propiedad de seguridad simple, de observar la ristra de caracteres confidencial. Cuando el programa intente almacenar la ristra en el fichero de nivel público (el fichero de recopilación), entonces, se viola la **propiedad*** y el intento es rechazado por el monitor de referencia. Así, el intento de escribir en el fichero de recopilación es denegado incluso aunque la lista de control de acceso lo permita; la política de seguridad tiene prioridad sobre el mecanismo de lista de control de acceso.

11.3 BIBLIOGRAFÍA Y SITIOS WEB RECOMENDADOS

Un tratamiento clásico de los cortafuegos, que todavía es válido, es [CHAP95]. Otro clásico, recientemente actualizado, es [CHES00]. [LODI98], [OPPL97] y [BELL94] son buenos artículos de descripción general sobre el asunto. [WACK02] es una excelente descripción general sobre la tecnología de cortafuegos y las políticas de los mismos.

[GASS88] proporciona un estudio comprensible de los sistemas de computadores de confianza. [PFLE97] y [GOLL99] también proporcionan cobertura.

- BELL94** Bellovin, S., y Cheswick, W. «Network Firewalls». *IEEE Communications Magazine*, septiembre 1994.
- CHAP95** Chapman, D. y Zwick, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 1995.
- CHES00** Cheswick, W. y Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2000.
- GASS88** Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

- GOLL99** Gollmann, D. *Computer Security*. New York: Wiley, 1999.
- LODI98** Lodin, S., y Schuba, C. «Firewalls Fend Off Invasions from the Net». *IEEE Spectrum*, febrero 1988.
- OPPL97** Oppiger, R. «Internet Security: Firewalls and Beyond». *Communications of the ACM*, mayo 1997.
- PFLE97** Pfeeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.
- WACK02** Wack, J.; Cutler, K.; y Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, enero 2002.

Sitios web recomendados:

- **Firewall.com:** numerosos enlaces referentes a cortafuegos y recursos *software*.

11.3 PALABRAS CLAVE, PREGUNTAS DE REPASO Y PROBLEMAS

PALABRAS CLAVE

computador bastión	matriz de acceso	router de filtrado de paquetes
cortafuegos	monitor de referencia	seguridad multinivel
cortafuegos de inspección de paquetes	objeto	sistema de confianza
derechos de acceso	pasarela del nivel de aplicación	sujeto
lista de control de acceso (ACL)	pasarela del nivel de circuito	ticket de capacidad

PREGUNTAS DE REPASO

- 11.1.** Enumera tres objetivos de diseño de los cortafuegos.
- 11.2.** Enumera cuatro técnicas utilizadas por los cortafuegos para controlar el acceso y reforzar la política de seguridad.
- 11.3.** ¿Qué información utiliza un router de filtrado de paquetes típico?
- 11.4.** ¿Cuáles son algunas de las debilidades de un router de filtrado de paquetes?
- 11.5.** ¿Cuáles son las diferencias entre un router de filtrado de paquetes y un cortafuegos de inspección de estado?
- 11.6.** ¿Qué es una pasarela del nivel de aplicación?
- 11.7.** ¿Qué es una pasarela del nivel de circuito?
- 11.8.** ¿Cuáles son las diferencias entre las tres configuraciones de la Figura 11.2?
- 11.9.** ¿Cuál es la diferencia entre un sujeto y un objeto, en el contexto del control de acceso?

- 11.10.** ¿Cuál es la diferencia entre una lista de control de acceso y un ticket de capacidad?
- 11.11.** ¿Cuáles son las dos reglas que impone un monitor de referencia?
- 11.12.** ¿Qué propiedades se requieren de un monitor de referencia?

PROBLEMAS

- 11.1.** En un sistema seguro con seguridad multinivel, la necesidad de la regla «no leer hacia arriba» es bastante obvia. ¿Cuál es la importancia de la regla «no escribir hacia abajo»?
- 11.2.** En la Figura 11.5 se rompe un enlace de la cadena copia-y-observa-después del caballo de Troya. Hay otros dos ángulos posibles de ataque de Drake: Drake inicia la sesión e intenta leer la ristra directamente, o Drake asigna un nivel de seguridad confidencial al fichero de «recopilación». ¿Previene el monitor de referencia esos ataques?

APÉNDICE A

Estándares citados en este libro

Hay perros que no degradarían lo que ellos mismos consideran formas sagradas. Un pastor alemán muy refinado y serio con el que trabajé, por ejemplo, protestaba ruidosamente cuando otros perros no obedecían. Cuando lo enseñaba a rescatar, de broma, se me ocurrió poner la pesa de pie sobre un extremo. El perro miró a la pesa y luego a mí con desaprobación, y luego la colocó en su posición correcta antes de recogerla y volver con ella, algo malhumorado.

Adam's Task Calling Animals by Name, VICKI HEARNE

Estándares ANSI

Número	Título	Fecha
X9.17	Gestión de claves de instituciones financieras (mayoristas)	1995

RFC de Internet

Número	Título	Fecha
RFC 822	Estándar para el formato de mensajes de texto de Internet ARPA	1982
RFC 1321	Algoritmo de resumen de mensaje MD5	1992
RFC 1510	Servicio de autentificación de red Kerberos (V5)	1993
RFC 1636	Seguridad en la arquitectura de Internet	1994
RFC 1928	Protocolo SOCKS versión 5	1996
RFC 2026	Proceso de normalización de Internet	1996
RFC 2040	Algoritmos RC5, RC5-CBC, RC5-CBC-Pad y RC5-CTS	1996

RFC de Internet (*continuación*)

Número	Título	Fecha
RFC 2045	MIME Primera Parte: formato del cuerpo de mensajes en Internet	1996
RFC 2046	MIME Segunda Parte: tipos de medios	1996
RFC 2047	MIME Tercera Parte: ampliaciones de la cabecera de mensaje para texto no ASCII	1996
RFC 2048	MIME Cuarta Parte: procedimientos de registro	1996
RFC 2049	MIME Quinta Parte: criterios de conformidad y ejemplos	1996
RFC 2104	HMAC: <i>Hash</i> con claves para autentificación de mensajes	1997
RFC 2119	Palabras fundamentales de uso en los RFC para indicar niveles de requisitos	1997
RFC 2246	El protocolo TLS	1999
RFC 2401	Arquitectura de seguridad para el protocolo de Internet	1998
RFC 2402	Cabecera de autentificación IP	1998
RFC 2406	Encapsulado de carga útil de seguridad IP (ESP)	1998
RFC 2408	Asociación de seguridad de Internet y protocolo de gestión de claves	1998
RFC 2459	Certificado de infraestructura de clave pública X.509 de Internet y perfil CRL	1999
RFC 2570	Introducción a la Versión 3 del estándar de Internet para el marco de la gestión de red	1999
RFC 2571	Arquitectura para describir los marcos de la gestión de SNMP	1999
RFC 2572	Procesamiento y gestión de mensajes para el protocolo sencillo de gestión de red (SNMP)	1999
RFC 2573	Aplicaciones SNMP	1999
RFC 2574	Modelo de seguridad basado en usuarios (USM)	1999
RFC 2575	Modelo de control de acceso basado en vistas (VACM)	1999
RFC 2576	Coexistencia entre las versiones 1, 2 y 3 del estándar de Internet para el marco de la gestión de red	2000
RFC 2630	Sintaxis criptográfica de mensajes	1999
RFC 2632	S/MIME Versión 3 Manejo de certificados	1999
RFC 2633	S/MIME Versión 3 Especificación de mensajes	1999
RFC 2828	Glosario de seguridad en Internet	2000
RFC 3156	Seguridad MIME con OpenPGP	2001
RFC 3174	Algoritmo <i>hash</i> seguro 1 US	2001

Recomendaciones ITU-T

Número	Título	Fecha
X.509	El directorio: marcos de clave pública y certificado de atributos	2000
X.800	Arquitectura de seguridad para la interconexión de sistemas abiertos	1991

Estándares de procesamiento de Información de NIST

Número	Título	Fecha
FIPS 46-3	Estándar de cifrado de datos (DES)	1999
FIPS 81	Modos de operación del DES	1980
FIPS 113	Autentificación de datos computacionales	1985
FIPS 180-1	Estándar <i>hash</i> seguro	1995
FIPS 180-2 (borrador)	Estándar <i>hash</i> seguro	2001
FIPS 181	Generador automático de contraseñas	1993
FIPS 186-2	Estándar de firma digital	2000
FIPS 197	Estándares de cifrado avanzado	2001
SP 800-38A	Recomendación para los modos de operación de cifrado de bloque	2001

A P É N D I C E B

Algunos aspectos de la teoría de números

B.1 Números primos y primos relativos

Divisores

Números primos

Números primos relativos

B.2 Aritmética modular

El diablo dijo a Daniel Webster: «Plantéame una tarea que no pueda llevar a cabo y te daré lo que me pidas.»

Daniel Webster: «Muy bien. Demuestra que para n mayor que 2, la ecuación $a^n + b^n = c^n$ no tiene solución no trivial en los enteros.»

Acordaron un periodo de tres días para resolver la tarea, y el diablo desapareció.

Al cabo de los tres días, el diablo se presentó nuevamente, ojeroso, nervioso, mordiéndose el labio. Daniel Webster le dijo: «¿Cómo fue la tarea? ¿Probaste el teorema?»

«Eh? No... no lo he demostrado.»

«Entonces ¿puedo pedirte lo que sea? ¿dinero? ¿la presidencia?»

«¿Cómo? Ah, sí, por supuesto. ¡Pero mira! Si pudiésemos demostrar estos dos lemas...»

The Mathematical Magpie, CLIFTON FADIMAN

En este Apéndice, se ofrece información sobre dos conceptos a los que hace referencia este libro: los números primos y la aritmética modular.

B.1 NÚMEROS PRIMOS Y PRIMOS RELATIVOS

En esta sección, a menos que se especifique lo contrario, se trata sólo con enteros no negativos. El uso de enteros negativos no introduciría diferencias significativas.

DIVISORES

Decimos que $b \neq 0$ divide a a si $a = mb$ para algún m , donde a , b y m son enteros. Es decir, b divide a a si no hay ningún resto en la división. La notación b/a se usa comúnmente para expresar que b divide a a . También, si b/a , decimos que b es un divisor de a . Por ejemplo, los divisores positivos de 24 son 1, 2, 3, 4, 6, 8, 12 y 24.

Se cumplen las siguientes relaciones:

- Si $a|1$, entonces $a = \pm 1$.
- Si $a|b$ y b/a , entonces $a = \pm b$.
- Cualquier $b \neq 0$ divide a 0.
- Si $b|g$ y $b|h$, entonces $b/(mg + nh)$ para cualquier entero m y n .

Para entender este último punto, obsérvese que

Si $b|g$, entonces g es de la forma $g = b \times g_1$ para algún entero g_1 .

Si $b|h$, entonces h es de la forma $h = b \times h_1$ para algún entero h_1 .

Así,

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

y, por lo tanto, b divide a $mg + nh$.

NÚMEROS PRIMOS

Un entero $p > 1$ es un número primo si sus únicos divisores son ± 1 y $\pm p$. Los números primos desempeñan un papel fundamental en la teoría de números y en las técnicas que se tratan en el Capítulo 3.

Cualquier entero $a > 1$ puede factorizarse de una única forma como

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_t^{\alpha_t}$$

donde $p_1 > p_2 > \dots > p_t$ son números primos y donde cada $\alpha_i > 0$. Por ejemplo, $91 = 7 \times 13$; y $11011 = 7 \times 11^2 \times 13$.

Es útil plantearlo de esta otra forma. Si P es el conjunto de todos los números primos, entonces cualquier entero positivo se puede escribir de manera única de la siguiente forma:

$$a = \prod_p p^{a_p} \quad \text{donde cada } a_p \geq 0$$

El lado derecho es el producto con todos los números primos posibles p ; para cualquier valor particular de a , la mayoría de los exponentes a_p serán 0.

El valor de cualquier entero positivo dado se puede especificar indicando todos los exponentes no cero en la formulación anterior. Así, el entero 12 se representa por $\{a_2 = 2, a_3 = 1\}$, y el entero 18 se representa por $\{a_2 = 1, a_3 = 2\}$. La multiplicación de dos números es equivalente a sumar los exponentes correspondientes:

$$k = mn \quad \rightarrow \quad k_p = m_p + n_p \quad \text{para todo } p$$

¿Qué significa, en lo que se refiere a estos factores primos, decir que a/b ? Cualquier entero de la forma p^k se puede dividir sólo por un entero que sea una potencia menor o igual del mismo número primo, p^j con $j \leq k$. Así, podemos decir

$$a/b \quad \rightarrow \quad a_p \leq b_p \quad \text{para todo } p$$

NÚMEROS PRIMOS RELATIVOS

Usaremos la notación $\gcd(a, b)$ para referirnos al **máximo común divisor** de a y b . Se dice que el entero positivo c es el máximo común divisor de a y b si

1. c es un divisor de a y de b ;
2. cualquier divisor de a y b es un divisor de c .

La siguiente es una definición equivalente:

$$\gcd(a, b) = \max[k, \text{tal que } k/a \text{ y } k/b]$$

Como se necesita que el máximo común divisor sea positivo, $\gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$. En general, $\gcd(a, b) = \gcd(|a|, |b|)$. Por ejemplo, $\text{mcd}(60, 24) = \gcd(60, -24) = 12$. También, como todos los enteros distintos de 0 dividen a 0, tenemos $\gcd(a, 0) = |a|$.

Es fácil determinar el máximo común divisor de dos enteros positivos si expresamos cada entero como el producto de números primos. Por ejemplo,

$$\begin{array}{ll} 300 & = 2^2 \times 3^1 \times 5^2 \\ 18 & = 2^1 \times 3^2 \\ \gcd(18, 300) & = 2^1 \times 3^1 \times 5^0 = 6 \end{array}$$

En general,

$$k = \gcd(a, b) \rightarrow k_p = \min(a_p, b_p) \text{ para todo } p$$

Determinar los factores primos de un número grande no es tarea fácil, así que la relación anterior no conduce directamente a una forma de calcular el máximo común divisor.

Los enteros a y b son primos relativos si no tienen ningún factor primo en común, es decir, si su único factor común es 1. Esto equivale a decir que a y b son primos relativos si $\gcd(a, b) = 1$. Por ejemplo, 8 y 15 son primos relativos porque los divisores de 8 son 1, 2, 4 y 8, y los divisores de 15 son 1, 3, 5 y 15, con lo cual, 1 es el único número presente en las dos listas.

B.2 ARITMÉTICA MODULAR

Dado cualquier entero positivo n y cualquier entero a , si dividimos a entre n , obtenemos un cociente q y un resto r que obedece a la siguiente relación:

$$a = qn + r \quad 0 \leq r < n; q = \lfloor a/n \rfloor$$

donde $\lfloor x \rfloor$ es el mayor entero menor o igual que x . El resto r con frecuencia se conoce como **residuo**.

Si a es un entero y n es un entero positivo, definimos $a \bmod n$ como el resto cuando a se divide entre n . Así, para cualquier entero a , siempre podemos escribir

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Se dice que dos enteros a y b son **congruentes módulo n** , si $(a \bmod n) = (b \bmod n)$. Esto se expresa $a \equiv b \pmod n$. Por ejemplo, $73 \equiv 4 \pmod {23}$; y $21 \equiv -9 \pmod {10}$. Obsérvese que si $a \equiv 0 \pmod n$, entonces $n \mid a$.

El operador *módulo* tiene las siguientes propiedades:

1. $a \equiv b \pmod n$ si $n \mid (a - b)$
2. $(a \bmod n) = (b \bmod n)$ implica que $a \equiv b \pmod n$
3. $a \equiv b \pmod n$ implica que $b \equiv a \pmod n$
4. $a \equiv b \pmod n$ y $b \equiv c \pmod n$ implica que $a \equiv c \pmod n$

Para demostrar el primer punto, si $n|(a - b)$, entonces $(a - b) = kn$ para algún k . Por lo que podemos escribir $a = b + kn$. Por lo tanto, $(a \bmod n) = (\text{resto cuando } b + kn \text{ se divide entre } n) = (\text{resto cuando } b \text{ se divide entre } n) = (b \bmod n)$. Los demás puntos se demuestran fácilmente.

El operador $(\bmod n)$ establece una correspondencia entre todos los enteros y el conjunto de enteros $\{0, 1, \dots, (n-1)\}$. Esto sugiere la pregunta: ¿podemos realizar operaciones aritméticas dentro de los límites de este conjunto? Resulta que podemos; la técnica se conoce como **aritmética modular**.

La aritmética modular presenta las siguientes propiedades:

- 1.** $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
- 2.** $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
- 3.** $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Demostramos la primera propiedad. Definamos $(a \bmod n) = r_a$ y $(b \bmod n) = r_b$. Entonces podemos escribir $a = r_a + jn$ para algún entero j y $b = r_b + kn$ para algún entero k .

Entonces

$$\begin{aligned}(a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\&= (r_a + r_b + (k + j)n) \bmod n \\&= (r_a + r_b) \bmod n \\&= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

Las demás propiedades se demuestran fácilmente.

Glosario

Al estudiar el Imperio, Arrakis y toda la cultura que produjo Maud'Dib, aparecieron muchos términos poco comunes. Favorecer la comprensión es un objetivo loable; de ahí las definiciones y aclaraciones que se ofrecen a continuación.

Duna, FRANK HERBERT

Algunos de los términos que aparecen en este glosario están extraídos del *Glossary of Computer Security Terminology* [NIST91]. Dichos términos se indican en el glosario mediante un asterisco.

Algoritmo RSA Algoritmo de cifrado de clave pública basado en exponenciación modular. Es el único algoritmo generalmente aceptado como práctico y seguro para el cifrado de clave pública.

Ataques de repetición Ataque a través del cual se falsifica un servicio ya autorizado y realizado mediante una «solicitud duplicada» con el fin de repetir comandos autorizados.

Autentificación* Proceso que se usa para verificar la integridad de los datos transmitidos, especialmente mensajes.

Autenticador Información adicional que se añade antepuesta a un mensaje para permitir al receptor verificar que el mensaje debería aceptarse como auténtico. El autenticador puede ser funcionalmente independiente del contenido del mensaje (por ejemplo, un *nonce* o un identificador de origen) o puede ser una función del contenido del mensaje (por ejemplo, un valor *hash* o una suma de prueba criptográfica).

Bacteria Programa que consume recursos del sistema replicándose a sí mismo.

Bomba lógica Lógica insertada en un programa que busca un conjunto determinado de condiciones en el sistema. Cuando se satisfacen dichas condiciones, ejecuta alguna función que da lugar a acciones no autorizadas.

Caballo de Troya (troyano)* Programa con una función aparente o realmente útil que contiene funciones adicionales (ocultas) que subrepticiamente explotan autorizaciones legítimas del proceso en detrimento de la seguridad.

Canal encubierto Canal de comunicación que permite la transferencia de información de forma no deliberada por parte de los diseñadores de la herramienta de comunicación.

Centro de distribución de claves Sistema autorizado para transmitir claves de sesión temporales a usuarios. Cada clave de sesión se transmite en forma cifrada, usando una clave maestra que el centro de distribución de claves comparte con el usuario de destino.

Cifrado asimétrico Forma de criptosema en la que se realiza cifrado y descifrado usando dos claves diferentes, una clave pública y una clave privada. Se conoce también como cifrado de clave pública.

Cifrado convencional Cifrado simétrico.

Cifrado de clave pública Cifrado asimétrico.

Cifrado múltiple Uso repetido de una función de cifrado, con diferentes claves, para producir una correlación más compleja de texto claro a texto cifrado.

Cifrado simétrico Forma de criptosistema en la que el cifrado y el descifrado se realizan usando la misma clave. También se conoce como cifrado convencional.

Cifrado La conversión de texto claro o datos a una forma ininteligible mediante una traducción reversible, basada en una tabla o algoritmo de traducción.

Cifrador de bloque Algoritmo de cifrado simétrico por el cual un bloque grande de bits de texto claro (normalmente 64) se transforma, como un todo, en un bloque cifrado de la misma longitud.

Cifrador de flujo Algoritmo de cifrado simétrico por el que la salida en texto cifrado se produce bit a bit o byte a byte partiendo de un flujo de entrada en texto claro.

Cifrador Algoritmo de cifrado y descifrado. Un cifrador sustituye una cierta información (un elemento en texto claro) por otro objeto, con la intención de ocultar significado. Normalmente, la regla de sustitución se rige por una clave secreta.

Clave de sesión Clave temporal de cifrado que se usa entre dos usuarios.

Clave maestra Clave de larga duración que se emplea entre un centro de distribución de claves y un usuario con el fin de codificar la transmisión de las claves de sesión. Normalmente, las claves maestras se distribuyen por medios no criptográficos. También se conoce como clave de cifrado de clave.

Clave privada Una de las dos claves que se usan en un sistema de cifrado asimétrico. En favor de una comunicación segura, la clave privada sólo debe ser conocida por su creador.

Clave pública Una de las dos claves que se emplean en un sistema de cifrado asimétrico. Dicha clave se hace pública para ser utilizada conjuntamente con su correspondiente clave privada.

Clave secreta Clave que se utiliza en un sistema de cifrado simétrico. Los dos participantes deben compartir la misma clave, y esta clave debe permanecer secreta para proteger la comunicación.

Código de autentificación de mensaje (MAC) Suma de comprobación criptográfica.

Código Regla invariable para sustituir una cierta información (por ejemplo, letra, palabra, frase) por otro objeto, no necesariamente del mismo tipo. En general, no hay intención de ocultar significado. Algunos ejemplos incluyen el código de caracteres ASCII (cada carácter se representa mediante siete bits) y la modulación por desplazamiento de frecuencias (cada valor binario se representa por una frecuencia particular).

Computacionalmente seguro Seguro porque el tiempo y/o el coste que implica vencer la seguridad es demasiado alto para que sea factible.

Confusión Técnica criptográfica que trata de hacer que la relación entre las estadísticas del texto cifrado y el valor de la clave de cifrado sea lo más compleja posible. Esto se logra mediante el uso de un algoritmo complejo de desordenamiento que depende de la clave y de la entrada.

Contraseña* Ristra de caracteres que se utiliza para autenticar una identidad. El conocimiento de la contraseña y su identificador de usuario asociado se consideran como prueba de autorización para el uso de las capacidades asociadas con ese identificador de usuario.

Control de acceso discrecional* Medio para restringir el acceso a objetos basados en la identidad de sujetos y/o grupos a los que pertenecen. Los controles son discrecionales en el sentido de que un sujeto con determinado permiso de acceso es capaz de traspasar ese permiso (quizás indirectamente) a cualquier otro sujeto (a menos que un control de acceso obligatorio lo impida).

Control de acceso obligatorio Medio para restringir el acceso a objetos basados en atributos fijos de seguridad asignados a usuarios, archivos y otros objetos. Los controles son obligatorios en el sentido de que no pueden ser modificados ni por usuarios ni por sus programas.

Cortafuegos Computador dedicado que interactúa con computadores externos a una red y que tiene incorporadas precauciones especiales de seguridad para proteger archivos confidenciales que se encuentran en computadores de dicha red.

Criptoanálisis diferencial Técnica mediante la cual se cifran textos claros elegidos con patrones particulares de diferencia XOR. Los patrones de diferencia del texto cifrado resultante proporcionan información que se puede usar para determinar la clave de cifrado.

Criptoanálisis Rama de la criptología que trata de descubrir un cifrado para obtener información, o de falsificar información cifrada para que sea aceptada como auténtica.

Criptografía Rama de la criptología que trata del diseño de algoritmos para el cifrado y el descifrado, diseñada para asegurar la confidencialidad y/o la autenticidad de los mensajes.

Criptología El estudio de las comunicaciones seguras, que abarca tanto la criptografía como el criptoanálisis.

Descifrado La traducción o conversión de texto o datos cifrados (llamados texto cifrado) a texto o datos originales (llamados texto claro).

Difusión Técnica criptográfica que trata de oscurecer la estructura estadística del texto claro expandiendo la influencia de cada dígito individual de texto claro en muchos dígitos de texto cifrado.

Digrama Secuencia de dos letras. En inglés y en otros idiomas, la frecuencia relativa de distintos digramas en texto claro se puede utilizar en el criptoanálisis de algunos cifradores.

Efecto avalancha Característica de un algoritmo de cifrado por la cual un pequeño cambio en el texto claro o en la clave da como resultado un gran cambio en el texto cifrado. Para un código *hash*, el efecto avalancha es una característica por la cual un cambio pequeño en un mensaje da como resultado un gran cambio en el resumen del mensaje.

Encadenamiento de bloques Procedimiento que se utiliza en el cifrado de bloques simétrico por el que un bloque de salida no depende sólo del bloque y la clave actuales de entrada en texto claro, sino también de la entrada y/o salida anteriores. El efecto del encadenamiento de bloques consiste en que dos instancias del mismo bloque de entrada en texto claro producirán diferentes bloques de texto cifrado, dificultando así el criptoanálisis.

Firma digital Mecanismo de autentificación que permite al creador de un mensaje adjuntar un código que funciona como firma. La firma garantiza la fuente y la integridad del mensaje.

Función hash Función que establece la correspondencia entre un bloque de datos o mensaje de longitud variable y un valor de longitud fija llamado código *hash*. La función está diseñada de forma que, al estar protegida, proporcione un autentificador a los datos o al mensaje. También se conoce como resumen de mensaje.

Función unidireccional Función que se calcula fácilmente, pero cuyo inverso es imposible de calcular.

Función unidireccional con trampa Función que se calcula fácilmente; el cálculo de su inverso es imposible, a menos que se conozca determinada información privilegiada.

Generador de números pseudoaleatorios Función que, de forma determinista, produce una secuencia de números que parecen ser estadísticamente aleatorios.

Gusano Programa que puede replicarse a sí mismo y enviar copias de computador a computador a través de conexiones en red. A su llegada, el gusano puede activarse para replicarse y propagarse nuevamente. Además de la propagación, el gusano, normalmente, realiza alguna función no deseada.

Honeypot Sistema anzuelo diseñado para apartar a los posibles atacantes de los sistemas críticos. Una forma de detección de la intrusión.

Incondicionalmente seguro Seguro, incluso, ante un oponente con tiempo y recursos ilimitados.

Intruso Individuo que obtiene, o intenta obtener, acceso no autorizado a un sistema de computación u obtener privilegios no autorizados en ese sistema.

Kerberos Nombre que se ha dado al servicio de autentificación de código del Proyecto Athena.

Name Identificador o número que sólo se usa una vez.

Raíz primitiva Si r y n son enteros primos relativos con $n > 0$, y si $\phi(n)$ es el menor exponente positivo m tal que $r^m \equiv 1 \pmod{n}$, entonces r se denomina raíz primitiva módulo n .

Resumen de mensaje Función *hash*.

Seguridad multinivel Capacidad que aplica el control de acceso en múltiples niveles de clasificación de datos.

Sistema confiable Computador y sistema operativo del que se puede verificar que implementa una política de seguridad determinada.

Sistema de detección de intrusos Conjunto de herramientas automatizadas diseñadas para detectar el acceso no autorizado a un sistema *host*.

Suma de comprobación criptográfica (checksum) Autentificador que consiste en una función criptográfica de datos que van a ser autenticados y de una clave secreta. También se conoce como código de autenticación de mensajes (MAC, *Message Authentication Code*).

Texto cifrado La salida de un algoritmo de cifrado; la forma cifrada de un mensaje o datos.

Texto claro Entrada a una función de cifrado o salida de una función de descifrado.

Trampa Punto secreto de entrada a un programa que se usa para conceder acceso sin los métodos habituales de autenticación de acceso.

Vector de inicialización Bloque de datos aleatorio que se utiliza para empezar el cifrado de múltiples bloques de texto claro, cuando se usa la técnica de cifrado de encadenamiento de bloques. Sirve para frustrar ataques de texto claro conocido.

Virus Código insertado en un programa que hace que una copia de sí mismo se introduzca en otros programas. Además de la propagación, el virus, normalmente, realiza alguna función no deseada.

Referencias

En temas de este tipo, todos creen tener justificación para escribir y publicar lo primero que les pasa por la cabeza cuando tienen un bolígrafo en la mano, y piensan que su idea es tan cierta como que dos más dos son cuatro. Si los críticos se preocuparan por pensar en el tema durante años y contrastar cada una de las conclusiones con la historia real de la guerra, tal y como yo he hecho, indudablemente tendrían más cuidado con lo que escriben.

De la guerra, CARL VON CLAUSEWITZ

Abreviaturas

ACM	Association for Computing Machinery
IEEE	Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology

ALVA90 Alvare, A. «How Crackers Crack Passwords or What Passwords to Avoid». *Proceedings, UNIX Security Workshop II*, August 1990.

ANDE80 Anderson, J. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co., April 1980.

AXEL00 Axelsson, S. «The Base-Rate Fallacy and the Difficulty of Intrusion Detection». *ACM Transactions and Information and System Security*, August 2000.

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems*. NIST Special Publication SP 800-31, November 2000.

BAUE88 Bauer, D., and Koblentz, M. «NIDX—AN Expert System for Real-Time Network Intrusion Detection». *Proceedings, Computer Networking Symposium*, April 1988.

BELL90 Bellovin, S., and Merritt, M. «Limitations of the Kerberos Authentication System». *Computer Communications Review*, October 1990.

BELL96a Bellare, M., Canetti, R., and Krawczyk, H. «Keying Hash Functions for Message Authentication». *Proceedings, CRYPTO '96*, August 1996; published by Springer-Verlag. An expanded version is available at <http://www.cse.ucsd.edu/users/mihir>.

- BELL96b** Bellare, M., Canetti, R., and Krawczyk, H. «The HMAC Construction». *CryptoBytes*, Spring 1996.
- BERS92** Berson, T. «Differential Cryptanalysis Mod 2^{32} with Applications to MD5». *Proceedings, EUROCRYPT '92*, May 1992; published by Springer-Verlag.
- BLOO70** Bloom, B. «Space/time Trade-offs in Hash Coding with Allowable Errors». *Communications of the ACM*, July 1970.
- BLUM97a** Blumenthal, U.; Hein, N.; and Wijnen, B. «Key Derivation for Network Management Applications». *IEEE Network*, May/June, 1997.
- BLUM97b** Blumenthal, U., and Wijnen, B. «Security Features for SNMPv3». *The Simple Times*, December 1997.
- BOER93** Boer, B., and Bosselaers, A. «Collisions for the Compression Function of MD5». *Proceedings, EUROCRYPT '93*, 1993; published by Springer-Verlag.
- BOSS97** Bosselaers, A., Dobbertin, H., and Preneel, B. «The RIPEMD-160 Cryptographic Hash Function». *Dr. Dobb's Journal*, January 1997.
- BRYA88** Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Project Athena document, February 1988. Available at <http://web.mit.edu/kerberos/www/dialogue.html>.
- CASS01** Cass, S. «Anatomy of Malice». *IEEE Spectrum*, November 2001.
- CERT02** CERT Coordination Center. «Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol». CERT Advisory CA-2002-03, 25 June 2002. www.cert.org/advisories/CA-2002-03.html.
- CHAP95** Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 1995.
- CHEN98** Cheng, P., et al. «A Security Architecture for the Internet Protocol». *IBM Systems Journal*, Number 1, 1998.
- CHES97** Chess, D. «The Future of Viruses on the Internet». *Proceedings, Virus Bulletin International Conference*, October 1997.
- CHES00** Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2000.
- COHE94** Cohen, F. *A Short Course on Computer Viruses*. New York: Wiley, 1994.
- COME00** Comer, D. *Internetworking with TCP/IP, Volume I. Principles, Protocols and Architecture*. Upper Saddle River, NJ: Prentice Hall, 2000.
- CORM01** Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
- DAMG89** Damgård, I. «A Design Principle for Hash Functions». *Proceedings, CRYPTO '89*, 1989; published by Springer-Verlag.
- DAVI89** Davies, D., and Price, W. *Security for Computer Networks*. New York: Wiley, 1989.
- DAVI93** Davies, C., and Ganesan, R. «BApasswd: A New Proactive Password Checker». *Proceedings, 16th National Computer Security Conference*, September 1993.
- DENN87** Denning, D. «An Intrusion-Detection Model». *IEEE Transactions on Software Engineering*, February 1987.
- DIFF76** Diffie, W., and Hellman, M. «Multiuser Cryptographic Techniques». *IEEE Transactions on Information Theory*, November 1976.
- DIFF88** Diffie, W. «The First Ten Years of Public-Key Cryptography». *Proceedings of the IEEE*, May 1988. Reprinted in [SIMP92].

- DOBB96a** Dobbertin, H. «The Status of MD5 After a Recent Attack». *CryptoBytes*, Summer 1996.
- DOBB96b** Dobbertin, H., Bosselaers, A., and Preneel, B. «RIPEMD-160: A Strengthened Version of RIPEMD». *Proceedings, Third International Workshop on Fast Software Encryption*, 1996; published by Springer-Verlag.
- DORA99** Doraswamy, N., and Harkins, D. *IPSec*. Upper Saddle River, NJ: Prentice Hall, 1999.
- DREW99** Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.
- EFF98** Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998.
- ENGE80** Enger, N., and Howerton, P. *Computer Security*. New York: Amacom, 1980.
- FEIS73** Feistel, H. «Cryptography and Computer Privacy». *Scientific American*, May 1973.
- FORD95** Ford, W. «Advances in Public-Key Certificate Standards». *ACM SIGSAC Review*, July 1995.
- FORR97** Forrest, S., Hofmeyr, S., and Somayaji, A. «Computer Immunology». *Communications of the ACM*, October 1997.
- FRAN01** Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.
- GAUD00** Gaudin, S. «The Omega Files». *Network World*, June 26, 2000.
- HARL01** Harley, D., Slade, R., and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.
- HEBE92** Heberlein, L., Mukherjee, B., and Levitt, K. «Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks». *Proceedings, 15th National Computer Security Conference*, October 1992.
- GARD77** Gardner, M. «A New Kind of Cipher That Would Take Millions of Years to Break». *Scientific American*, August 1977.
- GARF97** Garfinkel, S., and Spafford, G. *Web Security & Commerce*. Cambridge, MA: O'Reilly and Associates, 1997.
- GASS88** Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.
- GOLL99** Gollmann, D. *Computer Security*. New York: Wiley, 1999.
- HELD96** Held, G. *Data and Image Compression: Tools and Techniques*. New York: Wiley, 1996.
- HONE01** The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.
- HUIT98** Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.
- IANS90** I'Anson, C., and Mitchell, C. «Security Defects in CCITT Recommendation X.509-The Directory Authentication Framework». *Computer Communications Review*, April 1990.
- ILGU93** Ilgun, K. «USTAT: A Real-Time Intrusion Detection System for UNIX». *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.
- JAVI91** Javit, H., and Valdes, A. «The SRI IDES Statistical Anomaly Detector». *Proceedings 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.
- JIAN02** Jiang, G. «Multiple Vulnerabilities in SNMP». *Security and Privacy Supplement to Computer Magazine*, 2002.

- JUEN85** Jueneman, R., Matyas, S., and Meyer, C. «Message Authentication». *IEEE Communications Magazine*, September 1988.
- KENT00** Kent, S. «On the Trail of Intrusions into Information Systems». *IEEE Spectrum*, December 2000.
- KEPH97a** Kephart, J., Sorkin, G., Chess, D., and White, S. «Fighting Computer Viruses». *Scientific American*, November 1997.
- KEPH97b** Kephart, J., Sorkin, G., Swimmer, B., and White, S. «Blueprint for a Computer Immune System». *Proceedings, Virus Bulletin International Conference*, October 1997.
- KLEI90** Klein, D. «Foiling the Cracker: A Survey of, and Improvements to, Password Security». *Proceedings, UNIX Security Workshop II*, August 1990.
- KOBL92** Koblas, D., and Koblas, M. «SOCKS». *Proceedings, UNIX Security Symposium III*, September 1992.
- KOHL89** Kohl, J. «The Use of Encryption in Kerberos for Network Authentication». *Proceedings, Crypto '89*, 1989; published by Springer-Verlag.
- KOHL94** Kohl, J., Neuman, B., and Ts'o, T. «The Evolution of the Kerberos Authentication Service», In Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Available at <http://web.mit.edu/kerberos/www/papers.html>.
- LAI91** Lai, X., and Massey, J. «Markov Ciphers and Differential Cryptanalysis». *Proceedings, EUROCRYPT '91*, 1991; published by Springer-Verlag.
- LEUT94** Leutwyler, K. «Superhack». *Scientific American*, July 1994.
- LODI98** Lodin, S., and Schuba, C. «Firewalls Fend Off Invasions from the Net». *IEEE Spectrum*, February 1998.
- LUNT88** Lunt, T., and Jagannathan, R. «A Prototype Real-Time Intrusion-Detection Expert System». *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, April 1988.
- MACG97** Macgregor, R., Ezvan, C., Liguori, L., and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.
- MADS93** Madsen, J. «World Record in Password Checking». *Usenet, comp.security.misc newsgroup*, August 18, 1993.
- MARK97** Markham, T. «Internet Security Protocol». *Dr. Dobb's Journal*, June 1997.
- MCHU00** McHugh, J., Christie, A., and Allen, J. «The Role of Intrusion Detection Systems». *IEEE Software*, September/October 2000.
- MEIN01** Meinel, C. «Code Red for the Web». *Scientific American*, October 2001.
- MENE97** Menezes, A., Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
- MERK79** Merkle, R. *Secrecy, Authentication, and Public Key Systems*. Ph.D. Thesis, Stanford University, June 1979.
- MERK89** Merkle, R. «One Way Hash Functions and DES». *Proceedings, CRYPTO '89*, 1989; published by Springer-Verlag.
- MEYE82** Meyer, C., and Matyas, S. *Cryptography: A New Dimension in Computer Data Security*. New York: Wiley, 1982.

- MILL88** Miller, S., Neuman, B., Schiller, J., and Saltzer, J. «Kerberos Authentication and Authorization System». *Section E.2.1, Project Athena Technical Plan*, M.I.T. Project Athena, Cambridge, MA, 27 October 1988.
- MILL98** Miller, S. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.
- MITC90** Mitchell, C., Walker, M., and Rush, D. «CCITT/ISO Standards for Secure Message Handling». *IEEE Journal on Selected Areas in Communications*, May 1989.
- MURH98** Murhammer, M., et al. *TCP/IP: Tutorial and Technical Overview*. Upper Saddle River, NJ: Prentice Hall, 1998.
- NACH97** Nachenberg, C. «Computer Virus-Antivirus Coevolution». *Communications of the ACM*, January 1997.
- NEED78** Needham, R., and Schroeder, M. «Using Encryption for Authentication in Large Networks of Computers». *Communications of the ACM*, December 1978.
- NICH99** Nichols, R. ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.
- OPPL97** Oppliger, R. «Internet Security: Firewalls and Beyond». *Communications of the ACM*, May 1997.
- PFLE97** Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.
- PIAT91** Piattelli-Palmarini, M. «Probability: Neither Rational nor Capricious». *Bostonia*, March 1991.
- PORR92** Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection*. Master's Thesis, University of California at Santa Barbara, July 1992.
- PROC01** Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.
- RESC01** Rescorla, E. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.
- RIVE78** Rivest, R., Shamir, A., and Adleman, L. «A Method for Obtaining Digital Signatures and Public Key Cryptosystems». *Communications of the ACM*, February 1978.
- RIVE94** Rivest, R. «The RC5 Encryption Algorithm». *Proceedings, Second International Workshop on Fast Software Encryption*, December 1994; published by Springer-Verlag.
- RIVE95** Rivest, R. «The RC5 Encryption Algorithm». *Dr. Dobb's Journal*, January 1995.
- RUBI97** Rubin, A.; Geer, D.; and Ranum, M. *Web Security Sourcebook*. New York: Wiley, 1997.
- SAFF93** Safford, D., Schales, D., and Hess, D. «The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment». *Proceedings, UNIX Security Symposium IV*, October 1993.
- SALO96** Salomaa, A. *Public-Key Cryptography*. New York: Springer-Verlag, 1996.
- SCHN93** Schneier, B. «Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)». *Proceedings, Workshop on Fast Software Encryption*, December 1993; published by Springer-Verlag.
- SCHN94** Schneier, R. «The Blowfish Encryption Algorithm». *Dr. Dobb's Journal*, April 1994.
- SCHN96** Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.
- SCHN00** Schneier, B. *Secrets, and Lies: Digital Security in a Networked World*. New York: Wiley 2000.
- SIMM92** Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.

- SING99** Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Anchor Books, 1999.
- SMIT97** Smith, R. *Internet Cryptography*. Reading, MA: Addison-Wesley, 1997.
- SNAP91** Snapp, S., et al. «A System for Distributed Intrusion Detection». *Proceedings, COMPCON Spring '91*, 1991.
- SPAF92a** Spafford, E. «Observing Reusable Password Choices». *Proceedings, UNIX Security Symposium III*, September 1992.
- SPAF92b** Spafford, E. «OPUS: Preventing Weak Password Choices». *Computers and Security*, No. 3, 1992.
- STAL99** Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.
- STAL00** Stallings, W. *Data and Computer Communications, 6th edition*. Upper Saddle River, NJ: Prentice Hall, 2000.
- STAL03** Stallings, W. *Cryptography and Network Security: Principles and Practice, 3rd edition*. Upper Saddle River, NJ: Prentice Hall, 2003.
- STEI88** Steiner, J., Neuman, C., and Schiller, J. «Kerberos: An Authentication Service for Open Networked Systems». *Proceedings of the Winter 1988 USENIX Conference*, February 1988.
- STEP93** Stephenson, P. «Preventive Medicine». *LAN Magazine*, November 1993.
- STER92** Sterling, B. *The Hacker Crackdown: Law and Disorder on the Electronic Frontier*. New York: Bantam, 1992.
- STEV94** Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994.
- STIN02** Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.
- STOL88** Stoll, C. «Stalking the Wily Hacker». *Communications of the ACM*, May 1988.
- STOL89** Stoll, C. *The Cuckoo's Egg*. New York: Doubleday, 1989.
- THOM84** Thompson, K. «Reflections on Trusting Trust (Deliberate Software Bugs)». *Communications of the ACM*, August 1984.
- TIME90** Time, Inc. *Computer Security, Understanding Computers Series*. Alexandria, VA: Time-Life Books, 1990.
- TSUD92** Tsudik, G. «Message Authentication with One-Way Hash Functions». *Proceedings, INFOCOM '92*, May 1992.
- TUNG99** Tung, B. *Kerberos: A Network Authentication System*. Reading, MA: Addison-Wesley, 1999.
- VACC89** Vaccaro, H., and Liepins, G. «Detection of Anomalous Computer Session Activity». *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1989.
- WACK02** Wack, J., Cutler, K., and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, January 2002.
- ZIV77** Ziv, J., and Lempel, A. «A Universal Algorithm for Sequential Data Compression». *IEEE Transactions on Information Theory*, May 1977.

Índice analítico

3DES, Véase TripleDES (3DES)

A

Adleman, Len, 75

Adler, Mark, 168

Adquisidor o banco del vendedor, SET, 248-49

AES (estándar de cifrado avanzado), 38-42

cifrado/descifrado, 39-40

descripción de, 39-42

etapa de cifrado, 41

página web, 51

Agente de gestión, 263

SNMP, 263

Alerta de certificado caducado, 233

Alerta de certificado desconocido, 234

Alerta de certificado erróneo, 233

Alerta de certificado no permitido, 233

Alerta de certificado revocado, 233

Alerta de fallo de descompresión, 233

Alerta de fallo de negociación, 233

Alerta de mac de registro erróneo, 233

Alerta de mensaje inesperado, 233

Alerta de no certificado, 233

Alerta de notificación de cierre, 233

Alerta de parámetro ilegal, 233

Algoritmo de cifrado de clave pública RSA, 75-77, 132-33, 237-39, 245

definición, 75

ejemplo de, 77

y S/MIME, 159

Algoritmo de cifrado y cifrado de clave pública, 72

Algoritmo de compresión, ZIP, 168-70

Algoritmo de descifrado y cifrado de clave pública, 72

Algoritmo de descompresión, ZIP, 170

Algoritmo de generación de subclave, 34

Algoritmo de resumen de mensaje MD5, 67, 239, 292

y S/MIME, 158-60

Algoritmo DES (*Data Encryption Standard*), 32, 34-36, 67

descripción de, 35

máquina «DES cracker», 35

robustez del, 35-36

uso del término, 35 nota al pie

Algoritmos criptográficos:

S/MIME, 158-59

DSS (*Digital Signature Standard*), 158

algoritmo de resumen de mensaje MD5, 159

algoritmos de clave pública, 158

ElGamal, 159

SHA-1, 159

Triple DES (3DES), 159

Algoritmos de cifrado simétrico, 34-44

AES (*Advanced Encryption Standard*), 38-43

Blowfish, 43

DES (*Data Encryption Standard*), 34-37

IDEA (*International Data Encryption Algorithm*), 42-43

RC5, 43-44

Triple DES (3DES), 37-38

Algoritmos de clave pública y S/MIME, 158-59

Amenazas, 5, 226, 342-53

acceso a la información, 16

autentificación, 225

comparación de, 225

confidencialidad, 225

denegación de servicio, 225

integridad, 225

servicio, 16

ubicación de, 226

Amenazas de acceso a la información, 16

Amenazas de denegación de servicio, 225

Amenazas de servicio, 16

Análisis de tráfico, 7

y USM, 284

Análisis de umbrales, 314

AntiVirus Online, 358

Aplicación de generación de comandos, SNMPv3, 279-82

Aplicación de generación de notificación, SNMPv3, 281

Aplicación de recepción de notificación, SNMPv3, 281

Aplicación de reenvío mediante proxy, SNMPv3, 281-82

Aplicación de respuesta a comandos, SNMPv3, 279

Aplicaciones de autentificación, 91-126

certificados de usuario, 112-17

Kerberos, 92-111

servicio de autentificación X.509, 111-21

áreas, 18

Aritmética modular, 390-91

congruente módulo n, 390

definición, 391

residuo, 390

Arquitectura de gestión de red, 262-64

Arquitectura de seguridad OSI (recomendación X.800 de la ITU-T), 4, 9, 12
 Arquitectura de seguridad OSI, 4-5
 Arquitectura del protocolo de gestión de red, 264-65
 Asociaciones de seguridad (SA), 183-86
 anidamiento de túneles, 198
 autentificación más confidencialidad, 198-99
 combinación, 198-201
 combinaciones básicas, 199-201
 ESP con opción de autentificación, 199
 grupo transporte/túnel, 199
 transporte adyacente, 198, 199
 dirección IP de destino, 184
 Identificador de protocolo de seguridad, 184
 parámetros, 184-85
 selectores, 185-86
 SPI (*Security Parameters Index*), 184
 Ataque de palabra probable, 30-32
 Ataque de sólo texto cifrado, 31
 Ataque de texto cifrado elegido, 31
 Ataque de texto claro conocido, 30
 Ataque de texto claro elegido, 31
 Ataque de texto elegido, 31
 Ataques a la seguridad, 5-9, *Véase*
 Ataques
 ataques activos, 8-9, 56, 226
 ataques pasivos, 6-8, 56, 226
 Ataques activos, 8-9, 56, 226
 de interrupción de servicio, 9
 de modificación de mensajes, 9
 de repetición, 9
 de suplantación, 8-9
 Ataques de enrutamiento en origen, y *routers* de filtrado de paquetes, 388-69
 Ataques de fragmentos pequeños, y *routers* de filtrado de paquetes, 389
 Ataques de interrupción de servicio, 9
 y USM, 284
 y zombis, 344-45
 Ataques, 5
 activos, 8-9, 56, 226
 de interrupción de servicio, 9
 de suplantación, 8-9
 de modificación de mensajes, 9
 de repetición, 9
 de texto cifrado elegido, 31

de texto claro elegido, 31
 de texto elegido, 31
 de sólo texto cifrado, 31
 de interrupción de servicio, 9
 y USM, 283
 y zombis, 344-45
 de texto claro conocido, 30-31
 de palabra probable, 30-31
 de enrutamiento en origen, 368-69
 de fragmento pequeño, 389
 Ataques pasivos, 6-8, 56, 226
 detección de, 7-8
 obtención de contenido de mensaje, 6
 patrón de tráfico, 8
 Autentificación, 11
 autentificación mediante cifrado convencional, 56-57
 entidad origen/destino, 11
 origen de los datos, 11
 protocolo de determinación de claves Oakley, 202
 Autentificación bidireccional, 118
 Autentificación de la entidad par, 11
 Autentificación de mensaje, 57-60
 código de, 57-59
 definición, 56
 función *hash* unidireccional, 59-60
 usando cifrado convencional, 56-60
 sin cifrado de mensaje, 57-59
 Autentificación del origen de datos, 11
 Autentificación tridireccional, 118
 Autentificación unidireccional, 118
 Autoridad de certificación, 82, 112, 236
 SET, 249
 y SET, 249
 Autorización de pago, SET, 256-57

B

Base de datos fundamental de seguridad, 377
 BCP (*Best Current Practice*), 21
 BER (*Basic Encoding Rules*), 161
Blowfish, 43
 y ESP, 193
 Bomba lógica, 344
 Búsqueda exhaustiva de claves, tiempo medio para, 32
Byte indicador de confianza, contenido de, 145

C

Caballos de Troya, 343, 344
 defensa, 378-80
 Cabecera de autentificación, 182, 188-92
 cabecera de autentificación en modo transporte, 191
 cabecera de autentificación en modo túnel, 191-92
 campo de cabecera siguiente, 188
 campo de datos de autentificación, 189-90
 campo de índice de parámetros de seguridad, 188
 campo de longitud de carga útil, 188
 campo de número de secuencia, 188
 campo reservado, 188
 servicio contra repetición, 189
 valor de comprobación de la integridad, 189-90
 Cabecera de autentificación, IPv6, 219
 Cabecera de encapsulamiento de carga útil de seguridad, IPv6, 219
 Cabecera de enrutamiento, IPv6, 219-21
 Cabecera de fragmento, IPv6, 218-19, 221-22
 Cabecera de opciones de destino, IPv6, 218, 221
 Cabecera de opciones salto en salto, IPv6, 219, 221
 Cabeceras de extensión: IPv6, 219-22
 cabecera de autentificación, 219
 cabecera de encapsulamiento de carga útil de seguridad, 219
 cabecera de fragmento, 219, 221-22
 cabecera de opciones de destino, 219-22
 cabecera de opciones salto en salto, 219, 221
 cabecera de enrutamiento, 219-21
 orden de, 219-20
 Campo cabecera siguiente:
 cabecera de autentificación (AH), 188-89
 cabecera de opciones salto en salto, 221

- cabecera IPv6, 219
 encapsulamiento de la carga útil de seguridad (ESP), 193
 Campo de acción, registros de auditoría, 312
 Campo de algoritmo de cifrado, CipherSpec (especificación de cifrado), 237
 Campo de algoritmo MAC, CipherSpec, 237
 Campo de atributos de directorio de sujeto, 120
 Campo de clase de tráfico, cabecera IPv6, 218
 Campo de condición de excepción, registros de auditoría, 312
 Campo de confianza en el dueño, 145
 Campo de confianza en la firma, 145
 Campo de *cookie* del iniciador, cabecera ISAKMP, 206
 Campo de *cookie* del replicante, cabecera ISAKMP, 206
 Campo de datos de autenticación: cabecera de autenticación, 188-90
 encapsulamiento de la carga útil de seguridad, 192
 Campo de datos de carga útil, encapsulamiento de carga útil de seguridad (ESP), 192-93
 Campo de desplazamiento de fragmento:
 cabecera de fragmento, 222
 cabecera IPv4, 216
 Campo de dirección de destino:
 cabecera IPv4, 218
 cabecera IPv6, 218-19
 Campo de dirección fuente:
 cabecera IPv4, 217
 cabecera IPv6, 218-19
 Campo de etiqueta de flujo, cabecera IPv6, 218-19
 Campo de extensión de cabecera, cabecera de opciones salto en salto, 221
 Campo de identificación:
 cabecera de fragmento, 222
 cabecera IPv4, 216-18
 Campo de identificador de mensaje, cabecera ISAKMP, 207
 Campo de identificador de sesión, mensaje client_hello, 236
 Campo de indicador M, cabecera de fragmento, 222
 Campo de indicadores:
 cabecera IPv4, 216-18
 cabecera ISAKMP, 205-7
 Campo de índice de parámetros de seguridad:
 cabecera de autenticación (AH), 188-89
 encapsulamiento de la carga útil de seguridad (ESP), 192-93
 Campo de legitimidad de la clave, 145
 Campo de límite de salto, cabecera IPv6, 219
 Campo de longitud de cabecera de Internet, cabecera IPv4, 216
 Campo de longitud de carga útil:
 cabecera de autenticación (AH), 188
 cabecera IPv6, 219-20
 Campo de longitud, cabecera ISAKMP, 207
 Campo de longitud de relleno, encapsulamiento de la carga útil de seguridad (ESP), 194
 Campo de material de clave, CipherSpec, 237
 Campo de nombre alternativo de emisor, 120
 Campo de nombre alternativo de sujeto, 120
 Campo de número de secuencia:
 cabecera de autenticación (AH), 188-89
 encapsulamiento de la carga útil de seguridad (ESP), 192-93
 Campo de objeto, registros de auditoría, 312
 Campo de opciones:
 cabecera de opciones salto en salto, 221
 cabecera IPv4, 218
 Campo de protocolo, cabecera IPv4, 217
 Campo de relleno:
 cabecera IPv4, 218
 encapsulamiento de carga útil de seguridad (ESP), 192-94
 Campo de segmentos restantes, cabecera de enrutamiento, 221
 Campo de sello de tiempo, registros de auditoría, 313
 Campo de siguiente carga útil, cabecera ISAKMP, 206
 Campo de suite de cifrado, mensaje client_hello, 236
 Campo de sujeto, registros de auditoría, 312
 Campo de suma de prueba de cabecera, cabecera IPv4, 218
 Campo de tamaño de *hash*, CipherSpec, 237
 Campo de tamaño de vector de inicialización (IV), CipherSpec, 237
 Campo de tiempo de vida, cabecera IPv4, 217
 Campo de tipo de cifrado, CipherSpec, 237
 Campo de tipo de enrutamiento, cabecera de enrutamiento, 221
 Campo de tipo de intercambio, cabecera ISAKMP, 205-7
 Campo de tipo de servicio, cabecera IPv4, 216
 Campo de uso de recursos, registros de auditoría, 313
 Campo de valor aleatorio, mensaje client_hello, 236
 Campo de versión:
 cabecera IPv4, 216
 cabecera IPv6, 218
 mensaje client_hello, 235
 Campo de versión mayor, cabecera ISAKMP, 206
 Campo de versión menor, cabecera ISAKMP, 206
 Campo Es exportable, CipherSpec, 237
 Campo método de compresión, mensaje client_hello, 236
 Campo msgFlags, 282-83
 Campo msgID, 282
 Campo msgMaxSize, 282
 Campo msgSecurityModel, 283
 Campo msgVersion, 282
 Campo Res, cabecera de fragmento, 222
 Campo Reservado, cabecera de autenticación (AH), 188
 Captura de pago, SET, 25
 Carga útil de certificado, ISAKMP, 209
 Carga útil de eliminación, ISAKMP, 210
 Carga útil de firma, ISAKMP, 209
 Carga útil de identificación, ISAKMP, 207
 Carga útil de intercambio de clave, ISAKMP, 207
 Carga útil de notificación, ISAKMP, 209

- Carga útil de propuesta, ISAKMP, 207
 Carga útil de SA, ISAKMP, 207
 Carga útil de solicitud de certificado, ISAKMP, 209
 Carga útil de transformación, ISAKMP, 207
 Carga útil *hash*, ISAKMP, 209
 Carga útil *nonce*, ISAKMP, 209
 CAST-128, 131, 132, 133, 137, 141
 CAST y ESP, 193, 194
 Centro de distribución de claves (KDC), 51
 CERT (*Computer Emergency Response Team*), 178, 308
 CERT Coordination Center, 333
 Certificados, 112-17
 correspondencia de políticas, 120
 forward, 115
 obtención de, 114-16
 políticas, 119
 reverse, 115
 revocación de, 116-17
 Certificados de usuario:
 obtención de, 114-18
 revocación de, 116-17
 Certificados *forward*, 115
 Certificados *reverse*, 115
 Cifrado, 14-15
 Cifrado convencional, usando autenticación, 56-61
 Cifrado de bloque, relleno, 231
 Cifrado de clave pública, 15 *nota al pie*
 y Protocolo de determinación de claves Oakley, 202
 Cifrado de clave secreta, *Véase* Cifrado simétrico:
 Cifrado de clave simétrica y protocolo de determinación de clave Oakley, 204
 Cifrado de enlace, 46-49
 Cifrado extremo a extremo, 48-49
 Cifrado simétrico, 28
 algoritmo de cifrado, 28
 algoritmo de descifrado, 29
 ataque de palabra probable, 30
 ataque de sólo texto cifrado, 30-31
 ataque de texto cifrado elegido, 31
 ataque de texto claro conocido, 30-31
 ataque de texto claro elegido, 31
 ataque de texto elegido, 31
 clave secreta, 28
 criptoanálisis, 30-32
 criptografía, 29-30
 dispositivos de cifrado, ubicación de, 48-49
 distribución de claves, 49-51
 en modo CBC, 44-46
 en modo CFB, 46-48
 esquemas de cifrado computacionalmente seguro, 31
 estructura del cifrado de Feistel, 32-34
 principios, 28-34
 requisitos para el uso seguro de, 29
 texto cifrado, 28
 texto claro, 28
 Cifrado unidireccional e intrusos, 308
 Cifrado/descifrado y criptosistemas de clave pública, 73-74
 Cifrador de bloque, 30
 Cifrador de flujo, 30
 CipherSpec, (especificación de cifrado) 237, 240, 241
 Clave de sesión, 50
 Clave maestra, 239, 240-41
 Clave permanente, 50
 Clave privada, 72, 130
 fichero de clave privada, 140
 Clave pública, 72
 fichero de clave privada, 140
 fichero de clave pública, 140
 Claves secretas, 72
 distribución de clave pública de, 83-84
 COAST, 23
 Codificación de transferencia binaria, 155
 Codificación de transferencia de 7 bits, 155
 Codificación de transferencia de 8 bits, 155
 Codificación de transferencia x-token, 155
 Código de alerta de acceso denegado, TLS, 244
 Código de alerta de autoridad de certificación desconocida, TLS, 244
 Código de alerta de cancelado por usuario, TLS, 244
 Código de alerta de error de decodificación, TLS, 245
 Código de alerta de error de descifrado, TLS, 245
 Código de alerta de error interno, TLS, 245
 Código de alerta de fallo de descifrado, TLS, 244
 Código alerta de no renegociación, TLS, 244
 Código alerta de restricción de exportación, TLS, 245
 Código alerta de seguridad insuficiente, TLS, 244-45
 Código alerta de sobrecarga de registro, TLS, 244
 Componente de clave de sesión, 139
 Componente de firma, 138-39
 dos octetos iniciales de, 138
 identificador de clave de las claves públicas del emisor, 141
 resumen de mensaje, 138
 sello de tiempo, 138
 Componente de mensaje, 138
 Compresión, 229
 Comprobación de contraseña proactiva, 329
 Comprobación de contraseña reactiva, 329
 Comprobación de la integridad, 354
 Computador bastión, 371-72
Computer and Network Security Reference Index, 23
Computer Security Resource Center, 23
 Comunidad SNMP, definición, 270
 Concepto de monitor de referencia, 377
 Congruente módulo n , 390-91
 Contexto MIB y determinación de acceso, 295
 Contraseñas generadas por computador, 328
 Control de acceso, 10-11, 14
 e intrusos, 308-309
 Control de acceso a los datos, 374-76
 listas de control de acceso, 375
 matriz de acceso, 375
 tickets de capacidad, 375
 Compresión de datos usando ZIP, 168-70
 Confidencialidad de los datos, 10-11
 Integridad de los datos, 10-12, 14
 Control de acceso basado en vistas (VAC), 273, 293-98
 elementos del modelo VACM, 293-95
 contexto MIB, 294
 grupo, 293
 nivel de seguridad, 294
 política de acceso, 295
 vista MIB, 294
 motivación, 296

- procesamiento del control de acceso, 295-98
- Control de comportamiento, cortafuegos, 364
- Control de dirección, cortafuegos, 363
- Control de enrutamiento, 14
- Control de servicio, cortafuegos, 363
- Control de usuarios, cortafuegos, 363
- Conversión radix 64, 134, 135, 171-72
- Correspondencias de política, 120
- Cortafuegos, 361-82
- capacidades, 364
 - características de, 363-64
 - configuraciones, 372-74
 - control de comportamiento, 364
 - cortafuegos de computador protegido, bastión de interfaz dual, 374
 - cortafuegos de computador protegido, bastión de interfaz única, 372
 - cortafuegos de subred protegida, 374
 - control de dirección, 363
 - control de servicio, 363
 - control de usuario, 363-64
 - principios de diseño, 362-74
 - sistemas de confianza, 374-80
 - concepto de, 376-78
 - control de acceso a los datos, 374-76
 - defensa frente a caballo de Troya, 378-80
 - definición, 376-78
 - tipos de, 364-72
 - computador bastión, 371-72
 - cortafuegos de inspección de estado, 369-70
 - pasarela del nivel de aplicación, 370
 - pasarela del nivel de circuito, 370-71
 - routers* de filtrado de paquetes, 365-69
- Cortafuegos de computador protegido, bastión de interfaz dual, 372-74
- Cortafuegos de computador protegido, bastión de interfaz única, 372-74
- Cortafuegos de inspección de estado, 369-70
- tabla de estado de conexión, ejemplo de, 370
- Cortafuegos de subred protegida, 374
- Criptografía de clave pública:
- algoritmo de cifrado de clave pública RSA, 75-77
 - algoritmos, 75-81
 - criptografía de curva elíptica (ECC), 81
 - DSS (*Digital Signature Standard*), 81
 - firmas digitales, 81-82
 - gestión de claves, 82-84
 - intercambio de claves Diffie-Hellman, 78-80
 - principios, 71-75
 - aplicaciones de criptosistemas de clave pública, 73-74
 - estructura del cifrado de clave pública, 71-73
 - requisitos para, 74-75
- Criptografía de curva elíptica (ECC), 81
- Criptosistemas de clave pública:
- aplicaciones, 73-74
 - cifrado/descifrado, 73
 - firma digital, 74
 - intercambio de clave, 74
- D**
- Detección de acciones, 14
- Detección de anomalías basada en perfiles, 314
- métricas, 314
 - calibre, 314
 - contador, 314
 - intervalo de tiempo, 314
 - utilización de recursos, 314
- Detección estadística de anomalías, 313-5, véase Detección de anomalías basada en perfiles
- media y desviación estándar, 314
- métricas, 314-16
- modelo de procesos de Markov, 315
- modelo de serie temporal, 315
- modelo multivariable, 315
- modelo operativo, 315
- Diffie-Hellman anónimo, 237, 239
- Diffie-Hellman efímero, 236, 237, 239
- Dispositivos de cifrado, ubicación, 48-49
- Distribución de claves, 49-51
- Divisores, 388-89
- DNS, 372
- DSS (*Digital Signature Standard*), 81 y S/MIME, 159
- E**
- EFF (*Electronic Frontier Foundation*), 35-36
- ElGamal, 133
- y S/MIME, 157, 158, 159
- Emisor o banco del comprador, SET, 248, 249
- Emulador de CPU, explorador GD, 355
- Encapsulamiento de la carga útil de seguridad (ESP), 182, 192-98
- algoritmos de cifrado y autenticación, 193-94
 - campo de cabecera siguiente, 193
 - campo de datos de autenticación, 193
 - campo de datos de carga útil, 193
 - campo de índice de parámetros de seguridad, 193
 - campo de longitud de relleno, 193
 - campo de número de secuencia, 193
 - campo de relleno, 193
 - ESP en modo transporte, 194, 195-97
 - ESP en modo túnel, 194, 197-98
 - formato, 193
- Enfoque de fuerza bruta, al criptanálisis, 32
- Enfoques de antivirus, 353-55
- ESP (*Encapsulating Security Payload*), Véase Encapsulamiento de la carga útil de seguridad;
- Especificación técnica, 20
- Esquemas de cifrado computacionalmente seguro, 31
- Esquemas de dirección y routers, 214
- Estación de gestión, 263
- SNMP, 263
- Estado de conexión, 229
- Estándar para la criptografía de clave pública (IEEE P1363), 81
- Estándares, 383-85
- Estructura del cifrado de Feistel, 32-34, 67
- algoritmo de generación de subclaves, 34
 - cifrado/descifrado mediante software rápido, 34
 - facilidad de análisis, 34
 - función de etapa, 34

número de etapas, 34
tamaño de bloque, 34
tamaño de clave, 34
Etiqueta de seguridad, 14
Etiquetas de seguridad, 166
Explorador de firma de virus, tecnología GD, 355
Exploradores de cuarta generación, 354-55
Exploradores de primera generación, 354
Exploradores de segunda generación, 354
Exploradores de tercera generación, 354
Extensiones, certificado de usuario, 114-15

F

Falacia de la tasa base, 336-39
demostración de, 338-39
independencia, 336-37
probabilidad condicional, 336-37
teorema de Bayes, 337-38
Falsos IP, 178
Fase de activación, virus, 345
Fase de ejecución, virus, 345
Fase de propagación, virus, 345
Fase inactiva, virus, 345
Feistel, Horst, 32
Flabilidad y routers, 214
fichero de clave privada, 140-41
fichero de clave pública, 140
Fichero de claves públicas, 139-43
Filtro de Bloom, 330
Firewall.com, 381
Firma, certificado de usuario, 113
Firmas digitales, 14, 59, 81-82, 157
y criptosistemas de clave pública, 74
y Protocolo de determinación de claves Oakley, 204
Firma dual, SET, 252
Fragmentación, 214, 229
FTP, 265, 370, 371
Función de compresión, 65
Función de etapa, estructura del cifrado de Feistel, 32-34
Función hash resistente a la colisión, 61 *nota al pie*
Función hash unidireccional, 59-60, 61 *nota al pie*

Función hash unidireccional robusta, 61 *nota al pie*
Funcionalidad fiable, 14
Funciones hash seguras, 61-71
algoritmo de resumen de mensaje MD5, 67
algoritmo de resumen de mensaje RIPEMD-160, 67-68
comparación de, 68
HMAC, 68-71
requisitos de las funciones hash, 61-62
SHA-1, 64-67
simple, 62-64
Funciones, S/MIME, 157-58

G

Gailly, Jean-Iup, 168
Generadora de mutaciones, 349
Gestión de claves, 82-4
Gestión de contraseñas, 323-32
control de acceso, 327-28
estrategias de selección de contraseñas, 328-32
filtro de Bloom, 330-31
modelo de Markov, 330-32
protección de contraseñas, 323-28
comprobación de contraseña reactiva, 329
comprobador de contraseña proactivo, 329
contraseñas generadas por computador, 328
técnica de educación del usuario, 328
vulnerabilidad de contraseña, 323-27
Grupo de asociaciones de seguridad, 198
Grupo de trabajo SNMPv3 de la IETF, 298
Gusano Código Rojo/Código Rojo II, 352-53
Gusano Morris, 352

H

HAMC, 69-71, 289
algoritmo, 69-71
estructura, 70
objetivos de diseño, 69
HTTP (*Hypertext Transfer Protocol*, 227

I

IAB (*Internet Architecture Board*, 19, 178-79
IDEA (*International Data Encryption Algorithm*), 42-43, 133, 137, 141
y ESP, 193-94
Identificador de algoritmo de firma, certificado de usuario, 114
Identificador de clave:
Identificador de clave de autoridad, 119
Identificador de clave del sujeto, 120
Identificador de usuario:

fichero de claves privadas, 140
fichero de claves públicas, 140
Identificador único de emisor de certificado, certificado de usuario, 113
Identificador único de sujeto, certificado de usuario, 113
IESG (*Internet Engineering Steering Group*), 19
IETF (*Internet Engineering Task Force*), 19, 181, 218
IETF Security Area, 23
Información de clave pública del sujeto, certificado de usuario, 113
Informe de aplicabilidad, 21
Informe para la auditoría de seguridad, 14
Instancia de objeto y determinación de acceso, 295
Intercambio agresivo, ISAKMP, 210
Intercambio base, ISAKMP, 210
Intercambio de autenticación, 14
Intercambio de clave Diffie-Hellman fijo, 236, 239
Intercambio de clave RSA, 237-38
Intercambio de claves Diffie-Hellman, 78-80, 84, 132
Diffie-Hellman anónimo, 237-39
Diffie-Hellman efímero, 236, 239
Diffie-Hellman fijo, 236, 239
seguridad de, 79
Intercambio de claves y criptosistemas de clave pública, 74
Intercambio de protección de identidad, ISAKMP, 210
Intercambio de sólo autenticación, ISAKMP, 210
Intercambio informativo, ISAKMP, 211
Interfaces y routers, 214

- Internet:
 categorías de estándares, 20-21
 proceso de estandarización, 19-20
 proceso de publicación de RFC,
 20
 tipos de RFC, 21
 uso del término, 2
- Intrusion Detection Working Group*
 (Grupo de trabajo para la detección de intrusos), 333
- Intrusos, 205-39
 detección de intrusos, 310-23
 gestión de contraseñas, 323-32
 incidente de Wily Hacker (1986-87), 307
 suplantador, 306
 técnicas de intrusión, 308-10
 usuario clandestino, 306
 usuario fraudulento, 306
 y cifrado unidireccional, 308
 control de acceso, 327-28
 estrategias de selección de
 contraseñas, 328-32
 protección de contraseña, 323-
 28
 vulnerabilidad de la contraseña, 323-28
- y control de acceso, 308
 detección basada en perfiles,
 311
 detección basada en reglas,
 311
 detección de anomalías, 311
 detección de anomalías basada
 en reglas, 317
 detección de la intrusión basa-
 da en reglas, 315-19
 detección de umbrales, 311
 detección distribuida de la in-
 trusión, 320-22
 detección estadística de ano-
 malías, 311, 313-15
 falacia de la tasa base, 319
 formato de intercambio de de-
 tección de intrusos, 322-23
honeypots, 322
 identificación de la penetra-
 ción, 311
 identificación de la penetración
 basada en reglas, 317-18
 registros de auditoría, 312-13
- IPSEC, carta del protocolo de segu-
 ridad IP, 212
- IPSEC, sitio de recursos de seguri-
 dad IP, 211
- IPv4, 216-18
 cabecera IP, 217
- campo de desplazamiento de
 fragmento, 217
- campo de dirección de desti-
 no, 218
- campo de dirección de origen,
 218
- campo de identificación, 216
- campo de indicadores, 216
- campo de longitud de cabece-
 ra de Internet, 216
- campo de opciones, 218
- campo de protocolo, 217
- campo de relleno, 218
- campo de suma de comproba-
 ción de cabecera, 218
- campo de tiempo de vida, 217
- campo de tipo de servicio, 216
- campo de versión, 216
- IPv4, selector del tipo de servicio,
 186
- IPv6, 218-22
 cabecera, 219-19
 campo de cabecera siguiente,
 219
- campo de clase de tráfico, 218
- campo de dirección de desti-
 no, 219
- campo de dirección de origen,
 219
- campo de etiqueta de flujo,
 218-19
- campo de límite de salto, 219
- campo de longitud de carga
 útil, 219
- campo de versión, 218
- cabeceras de extensión, 219-22
- cabecera de autenticación, 219
- cabecera de enrutamiento,
 219, 220
- cabecera de fragmento, 219,
 221-22
- cabecera de opciones de desti-
 no, 219-22
- cabecera de opciones salto en
 salto, 219-21
- encapsulamiento de la carga
 útil de seguridad, 219
- orden de, 219-20
- ISAKMP (*Internet Security Asso-
 ciation and Key Management
 Protocol*), 205-6, 210
 definición, 201
 Protocolo de determinación de
 claves Oakley, 202-5
- ISAKMP/Protocolo Oakley
- ITU-T, Sector de estandarización de
 telecomunicaciones de la ITU, 4
- J**
- Juegos de guerra, 343
- K**
- Kerberos, 92-111
 escalabilidad de, 94
 fiabilidad de, 94

motivación, 93-94
 múltiples, 104-5
 seguridad de, 94
 técnicas de cifrado, 123-26
 modo PCBC (*Propagating Cipher Block Chaining*), 125-26
 transformación de contraseña a clave, 123-25
 transparencia de, 94
 uso del término, 92, nota al pie
 versión 4 de, 94-105
 ataques de contraseña, 107
 autentificación entre dominios, 106
 base lógica para los elementos del protocolo, 100-3
 cifrado doble, 107
 cifrado PCBC, 107
 claves de sesión, 107
 comparado con la versión 5, 106-7
 deficiencias técnicas, 106-7
 dependencia del protocolo de Internet, 106
 dependencia del sistema de cifrado, 106
 diálogo de autentificación, 98-103
 dominios, 104-5
 envío de autentificación, 106
 Kerberos múltiples, 104-5
 ordenación de bytes del mensaje, 106
 tiempo de vida del ticket, 106
 versión 5, 105-11
 ataques de contraseña, 107
 autentificación entre dominios, 106
 cifrado doble, 107
 cifrado PCBC, 107
 claves de sesión, 107
 comparado con la versión 4, 106-7
 dependencia del protocolo de Internet, 106
 dependencia del sistema de cifrado, 106
 diálogo de autentificación, 107-9
 envío de autentificación, 106
 indicadores de ticket, 109-11
 intercambio de autentificación cliente/servidor, 109
 intercambio de servicio de autentificación, 108
 intercambio de TGT o servicio que concede un ticket, 109

ordenación de bytes del mensaje, 106
 tiempo de vida del ticket, 106

L

Laboratorios RSA, 85
 Lai, Xuejia, 42
 Lempel, Abraham, 168
 Libro de códigos, 44
 Limitaciones básicas, ruta de certificación, 120-21
 Limitaciones de nombre, ruta de certificación, 120-21
 Limitaciones de política, ruta de certificación, 121
 Limitaciones en la ruta de certificación, 120-21
 Lista de revocación de certificados (CRL), 116-17, 163
 Listas de control de acceso, 375, 376
 Listas de correo seguras, 166
 Lloyd, Tim, 344
 LZ77, 168-70

M

MAC, Código de autentificación de mensajes, 229-31, 241-42
 Massey, James, 42
 MasterCard SET, página web, 258
 Matriz de acceso, 375
 Máximo común divisor, 389-90
 Mecanismo de seguridad, 5
 Mecanismos de seguridad, 13-14
 Mensaje certificate_verify, 239
 TLS, 245
 Mensaje client_key_exchange, 239
 Mensaje change_cipher_spec, 239
 Mensaje de certificado, 237, 238
 Mensaje de respuesta de autorización, 257
 Mensaje de respuesta de captura, 258
 Mensaje de respuesta de compra, 255
 Mensaje de solicitud de autorización, 256-57
 Mensaje de solicitud de captura, 258
 Mensaje de solicitud de certificado, 238
 Mensaje de solicitud de compra, 254-55

Mensaje server_done, 238
 Mensaje server_key_exchange, 237-38

Mensajes:

 S/MIME, 159-63
 entidad MIME, asegurar, 160-61
 firma en claro, 162
 mensaje de sólo certificado, 163
 solicitud de registro, 163
 tipo smime envelopedData, 161-62
 tipo smime signedData, 161-62

Método de compresión, estado de sesión, 228

Métricas, detección estadística de anomalías, 313-15

MIB (*Management Information Base*), 263, 264

MIB II (*Management Information Base for Network Management of TCP/IP-based Internets* (RFC 1213)), 264

Microsoft y virus de macro, 349-50

MIME (*Multipurpose Internet Mail Extensions*), 149-58

campo de codificación de transferencia de contenido, 151, 154-55

campo de descripción de contenido, 151

campo de identificación de contenido, 151

campo de tipo de contenido, 151

campo de versión MIME, 151

campos de cabecera, 151

codificación de transferencia base, 64, 155

codificaciones de transferencia, 154-55

datos firmados, 157

datos firmados y empaquetados, 158

definición, 149-57

ejemplo multiparte, 155

elementos, 151

estructura de mensaje, ejemplo de, 156

forma canónica, 156, 157

forma nativa, 157

registro y usuario, 164

RFC 822, 150

subtipo mensaje/parcial, 154

subtipo mensaje/fcc822, 154

subtipo multiparte/alternativo, 153
 subtipo multiparte/mezclado, 153
 subtipo multiparte/paralelo, 153
 subtipo multiparte/resumen, 154
 tipo de aplicación, 154
 tipo de mensaje, 154
 tipo de texto, 153
 tipo multiparte, 153
 tipos de contenido, 152-54
 campo de versión MIME, 151
 campos de cabecera, 151
 codificaciones de transferencia, 154-55
 definición, 150-57
 ejemplo multiparte, 155
 elementos, 151
 estructura de mensaje, ejemplo de, 156
 forma nativa, 157
 subtipo mensaje/parcial, 154
 subtipo mensaje/rfc822, 154
 subtipo multiparte/alternativo, 153-54
 subtipo multiparte/mezclado, 153
 subtipo multiparte/paralelo, 153
 subtipo multiparte/resumen, 154
 tipo de mensaje, 154
 tipo de texto, 153
 tipo multiparte, 153
 transferencia imprimible textualmente, 155
MIT Distribution Site for PGP, 167
MIT Kerberos Site, 121
MLA (Mail List Agent), S/MIME, 166
 Modelo de Markov, para la selección de contraseñas, 330-32
 Modelo de seguridad de usuarios (USM), 273, 283-84
 y análisis de tráfico, 284
 y ataques de denegación de servicio, 284
 y modo CBC, 285
 Modelo de seguridad y determinación de acceso, 295
 Modelo de serie de tiempo, detección estadística de anomalías, 315
 Modelo multivariable, detección estadística de anomalías, 315
 Modelo operativo, detección estadística de anomalías, 315
 Modificación de mensajes, 9
 Modo CBC (*Cipher Block Chaining*), 44-46, 63, 237

Modo CFB (*Cipher Feedback*), 46, 131
 Modo ECB (*Electronic Codebook*), 44
 Modo PCBC (*Propagating Cipher Block Chaining*), Kerberos, 125-26
 Modo transporte, 186
 ESP, 194-97
 Modo túnel, 186-87
 ESP, 194, 197-98
 Módulo de control de emulación, explorador GD, 355
 Morris, Robert, 352
 Multics, 343

N

NCSA (*National Computer Security Agency*), 349
 Nimda, 353
 NIST (*National Institute of Standards and Technology*), 34, 38, 64
 NIST Secure Hashing Page, 85
 Nivel de seguridad y determinación de acceso, 295
 No repudio, 10, 12
 Nombre de emisor de certificado, certificado de usuario, 112
 Nombre de sujeto, certificado de usuario, 112
 Nonces, Protocolo de determinación de claves Oakley, 204
 Notarización, 14
 Noticias del grupo de trabajo para la seguridad IP, 212
 Número de serie, certificado de usuario, 112
 Números primos, 388-90
 definición, 389
 divisores, 388-89
 máximo común divisor, 389-90
 números primos relativos, 389-90

O

Obtención de contenido de mensaje, 6-8
 Omega Engineering, 344
 Orden, IPv6, 219-20

P

Paquete IP, uso del término, 184, *nota al pie*
 Paquete SOCKS, 371
 Parámetro AuthoritativeEngineID, USM, 286, 289
 Parámetro de certificado par, estado de sesión, 227-28
 Parámetro de clave de escritura del cliente, estado de conexión, 229
 Parámetro de clave de escritura del servidor, estado de conexión, 229
 Parámetro de clave maestra, estado de sesión, 227
 Parámetro de clave secreta para MAC de escritura del cliente, estado de conexión, 229
 Parámetro de especificación de cifrado, estado de sesión, 227-28
 Parámetro de identificador de sesión, estado de sesión, 228
 Parámetro de número de secuencia, estado de conexión, 229
 Parámetro de valores aleatorios de servidor y cliente, estado de conexión, 229
 Parámetro de vectores de inicialización, estado de conexión, 229
 Parámetro Es reanudable, estado de sesión, 228
 Parámetro latestReceivedEngineTime, USM, 286, 288
 Parámetro msgAuthenticationParameters, USM, 286
 Parámetro msgAuthoritativeEngineBoots, USM, 286, 289
 Parámetro msgAuthoritativeEngineTime, USM, 286, 289
 Parámetro msgPrivacyParameters, USM, 286
 Parámetro msgUserName, USM, 286
 Parámetro secreto de MAC de escritura del servidor, estado de conexión, 229
 Parámetro snmpEngineBoots, USM, 288-90
 Parámetro snmpEngineTime, USM, 288-90
 Participantes:
 SET, 248-50

- adquisidor o banco del vendedor, 249
 autoridad de certificación (CA), 249
 emisor o banco del comprador, 249
 pasarela de pago, 249
 titular de tarjeta, 248
 vendedor, 248
- Pasarela de pago, SET, 249
 Pasarela del nivel de aplicación, 370
 Pasarela del nivel de circuito, 370-71
 PDU extendida, 282
 Perfil de acceso SNMP, 271
 Perfil de comunidad SNMP, 271
 Período de uso de clave privada, 120
 Período de validez, certificado de usuario, 112
 PGP (*Pretty Good Privacy*), 43, 128-49
 claves criptográficas, 135-43
 definición de, 128
 descripción operativa, 130-35
 documentación, 130
 ficheros de claves, 139-43
 fichero de clave privada, 141
 fichero de clave pública, 141
 generación de clave de sesión, 137
 generación de mensajes, 142
 generación de números aleatorios, 173-75
 números aleatorios, 173
 números pseudoaleatorios, 173-75
 gestión de clave pública, 143-48
 confianza, uso de, 145-49
 enfoques para, 144-45
 revocación de claves públicas, 148-48
 Identificadores de clave de sesión, 137-39
 componente clave de sesión, 139
 componente de firma, 138-39
 componente mensaje, 138
 notación, 129-30
 autentificación, 130-31
 compatibilidad con correo electrónico, 134-35
 compresión, 133-34
 confidencialidad, 131-33
 confidencialidad y autentificación, 133
- segmentación y reensamblado, 135
 razones para el crecimiento de, 129
 recepción, 143
 resumen de servicios, 131
 PGP Home Page, 167
 Procesador frontal (*front-end*) (FEP), 51
 Procesamiento de certificado:
 S/MIME, 164
 certificados Verisign, 164-66
 papel de agente usuario, 164
 Procesamiento de mensajes, SHA-1, 64-67
 añadir bits de relleno, 65
 añadir longitud, 65
 Inicializar *buffer MD*, 65
 procesar mensaje en bloques de 512 bits (16 palabras), 65
 salida, 66
 Procesamiento del pago, SET, 252-58
 Proceso de estandarización, 19-20
 Programa de evaluación de productos comerciales, 378
 Programa de evaluación de productos comerciales, Agencia de seguridad nacional (NSA), 378
 Programas gusanos de red, 351
 Propiedad unidireccional, 61
 Protocolo Alert, 227, 232-34
 Protocolo Change Cipher Spec, 227, 228, 232, 239
 Protocolo de datagramas de usuario (UDP), 180, 264-65
 Protocolo de determinación de claves Oakley, 202-5
 características de, 203-4
 definición, 201
 ejemplo de, 204-5
 Intercambio de cookie, 203
 nonce, 204
 requisitos para la generación de cookies, 203-4
 y cifrado de clave pública, 204
 y cifrado de clave simétrica, 204
 y firmas digitales, 204
 Protocolo de gestión de red, 264
 Protocolo de Internet (IP), 214-16
 Protocolo Handshake, 227, 229, 234-41
 acción, 235
 autentificación de cliente e intercambio de clave (fase 3), 238-39
- autentificación de servidor e intercambio de clave (fase 2), 237-38
 campo de contenido, 234
 establecimiento de las capacidades de seguridad (fase 1), 235-37
 fases de intercambio inicial, 234-40
 finalización (fase 4), 239-40
 mensaje certificate_verify, 239
 mensaje certificate_request, 238
 mensaje client_hello, 235-36
 mensaje client_key_exchange, 239
 mensaje change_cipher_spec, 239
 campo de longitud, 234
 campo de tipo, 234
 mensaje server_done, 238
 mensaje server_hello, 236
 mensaje server_key_exchange, 237-38
 tipos de mensaje, 234
 Protocolo Record SSL, 229-32
 Protocolo Simple de Gestión de Red, Véase SNMP:
 Proyecto Honeypot, 333
 Proyecto RIPE, 67-68
Public-Key Infrastructure Working Group, 121
- ## R
- RC5, 43-44
 y ESP, 193
 Recibos firmados, 166
 Recuperación de la seguridad, 14
 Red virtual privada, 194
 Registros de auditoría, 312-13
 campos en, 312
 específicos para detección, 312
 nativos, 312
 Registros de auditoría específicos para la detección, 312
 Registros nativos de auditoría, 312
 Regla de no escribir hacia abajo, seguridad multinivel, 376
 Regla de no leer hacia arriba, seguridad multinivel, 376
 Relleno del tráfico, 14
 Repetición, 9
 Residuo, 390
 Resistencia débil a la colisión, 61
 Resistencia fuerte a la colisión, 61

RFC experimental, 21
 Rijndael, página web, 52
RIPE (RACE Integrity Primitives Evaluation), 67
 RIPEMD-160, 67
 Rivest, Ron, 43, 75
Routers, 214-16
Routers de filtrado de paquetes, 365-69
 ataques de enrutamiento en origen, 368-69
 ataques de fragmentos pequeños, 369
 ataques/contramedidas, 369
 debilidades de, 368
 ejemplos de, 367
 simplicidad de, 368
 suplantación de dirección IP, 368

S

SMIME, 83, 112, 149-66
 algoritmos criptográficos, 158-59
 algoritmo de cifrado de clave pública RSA, 159
 algoritmo de resumen de mensaje MD5, 159
 algoritmos de clave pública, 158
DSS (Digital Signature Standard), 158
 ElGamal, 159
 SHA-1, 159
 TripleDES (3DES), 159
 almacenamiento y recuperación de certificado y usuario, 164
 datos empaquetados, 157
 datos firmados en claro, 157
 funcionalidad, 157-59
 funciones, 157-58
 generación de claves y usuario, 164
MLA (Mailing List Agent), 166
 mensajes, 159-63
 entidad MIME, asegurar, 160-61
 firma en claro, 162-63
 mensaje de sólo certificado, 163
 solicitud de registro, 163
 tipo smime envelopedData, 161
 tipo smime signedData, 161-62
 procesamiento de certificado, 164-66
 certificados Verisign, 164-66

función de agente usuario, 164
 servicios de seguridad avanzada, 166
 etiquetas de seguridad, 166
 listas de correo seguras, 166
 recibos firmados, 166
 tipos de contenido, 160
SMIME Central Site, 167
SMIME Charter Site, 167
 Secuencia cifrar-descifrar-cifrar (EDE), 37
 Seguridad de Internet, 2
 Seguridad de la información, 2
 Seguridad de la red, 2
 grupos de trabajo USENET, 24
 modelo de, 14-17
 Seguridad de la web, 223-60
 amenazas, 226
 amenazas a la autenticación, 225
 amenazas a la confidencialidad, 225
 amenazas a la integridad, 225
 amenazas de denegación de servicio, 225
 cálculos criptográficos, 240-41
 creación de clave maestra, 240
 generación de parámetros criptográficos, 241
 enfoques para, 226-27
 retos de, 224-25
SET (Secure Electronic Transaction), 227, 246-58
 autorización de pago, 256-57
 captura de pago, 258
 características fundamentales, 248
 firma dual, 250-52
 introducción a, 247-50
 mensaje de respuesta de autorización, 257
 mensaje de respuesta de captura, 258
 mensaje de respuesta de compra, 255
 mensaje de solicitud de autorización, 256-57
 mensaje de solicitud de captura, 258
 mensaje de solicitud de compra, 252-56
 participantes, 248-50
 procesamiento del pago, 252-58
 requisitos, 247-48
 secuencia de hechos para una transacción, 250
 solicitud de compra, 252-56
 tipos de transacción, 253
SSL (Secure Sockets Layer), 226, 227-41
 arquitectura, 227-29
 protocolo Alert, 227, 232-34
 protocolo Change Cipher Spec, 232
 protocolo Handshake, 234-40
 protocolo Record SSL, 229-32
TLS (Transport Layer Security), 226, 241-46
 algoritmos de cifrado simétricos, 245
 cálculos criptográficos, 245-46
 código de autenticación de mensaje (MAC), 241-42
 códigos de alerta, 244
 función pseudoaleatoria, 242-43
 intercambios de clave, 245
 mensaje certificate_verify, 245
 mensaje finished, 246
 número de versión, 241
 relleno, 246
 suites de cifrado, 245
 tipos de certificados de clientes, 245
 Seguridad de los sistemas:
 ataques, 361-82
 intrusos, 305-39
 software dañino, 341-59
 Seguridad de redes, 3-4
 Seguridad del correo electrónico, 127-75
 compresión de los datos usando ZIP, 168-70
 conversión radix 64, 171-72
 generación de números aleatorios, 173-75
PGP (Pretty Good Privacy), 128-49
SMIME, 149-67
 Seguridad en la gestión de redes, 261-302
SNMP (Simple Network Management Protocol), 262-98
 agente de gestión, 263
 arquitectura de gestión de red, 262-64
 arquitectura del protocolo de gestión de red, 264-65
 base de información de gestión (MIB), 263
 conceptos básicos de, 262-70

- estación de gestión, 263
 papel de, 265
 protocolo de gestión de red, 264
proxy, 266
 SNMPv2, 266-70
- Seguridad informática, 2
- Seguridad IP (IPSec), 83, 112, 177-222
- aplicaciones de, 179
 - aplicaciones de enrutamiento, 181
 - áreas fundamentales, 178
 - arquitectura, 181-87
 - asociaciones de seguridad, 183-86
 - combinación de, 198-201
 - dirección de destino IP, 184
 - identificador de protocolo de seguridad, 184
 - índice de parámetros de seguridad (SPI), 184
 - modo transporte, 186
 - modo túnel, 186-87
 - parámetros, 184-85
 - selectores, 185-86
 - servicios, 182-83
 - beneficios de, 180
 - cabecera de autentificación (AH), 188-92
 - cabecera de autentificación (AH) en modo transporte, 191
 - cabecera de autentificación (AH) en modo túnel, 191-92
 - campo cabecera siguiente, 188
 - campo de datos de autentificación, 189
 - campo de índice de parámetros de seguridad, 188
 - campo de número de secuencia, 188
 - campo longitud de carga útil, 188
 - campo reservado, 188
 - servicio contra repeticiones, 189
 - valor de comprobación de la integridad, 189-90
 - documentos, 181-82
 - algoritmo de autentificación, 182
 - algoritmo de cifrado, 182
 - arquitectura, 181
 - cabecera de autentificación (AH), 182
 - dominio de Interpretación (DOI), 182
 - encapsulamiento de la carga útil de seguridad (ESP), 182
 - gestión de claves, 182
 - Encapsulamiento de la carga útil de seguridad (ESP), 192-98
 - algoritmos de cifrado y autenticación, 193-94
 - campo de cabecera siguiente, 193
 - campo de datos de autentificación, 193
 - campo de datos de carga útil, 193
 - campo de longitud de relleno, 193
 - campo de número de secuencia, 193
 - campo de relleno, 193
 - campo Índice de parámetros de seguridad, 193
 - ESP en modo transporte, 195-97
 - ESP en modo túnel, 194, 197-98
 - formato, 193
 - gestión de claves, 201-11
 - automática, 201
 - manual, 201
 - protocolo Oakley/ISAKMP, 201-11
 - introducción a, 178-81
 - Seguridad multivel, 376-77
 - Selector de clase IPv6, 186
 - Selector de dirección de destino, 185
 - Selector de dirección IP fuente, 185
 - Selector de etiqueta de flujo IPv6, 186
 - Selector de identificador de usuario, 186
 - Selector de puertos de origen y de destino, 185
 - Selector del nivel de confidencialidad de los datos, 186
 - Selector del Protocolo de la capa de transporte, 186
 - Selector del protocolo IPSec, 186
 - Sello de tiempo:
 - fichero de claves privadas, 140-41
 - fichero de claves públicas, 140-41
 - Servicio de autentificación X.509, 111-21
 - autentificación bidireccional, 118
 - autentificación tridireccional, 118
 - autentificación unidireccional, 118
 - certificados de usuarios, 112-17
 - formato de la versión 3, 119-21
 - información de claves y política, 119-20
 - limitaciones en la ruta de certificación, 120-21
 - sujeto de certificado y atributos del emisor de certificado, 120
 - procedimientos de autenticación, 117-18
 - revocación de, 116-17
 - obtención de, 114-16
 - Servicio de disponibilidad, 12
 - Servicio de seguridad, 5
 - Servicios de seguridad, 9-13
 - autentificación, 11
 - confidencialidad de los datos, 10
 - control de acceso, 10-11
 - integridad de los datos, 10-12
 - no repudio, 10-12
 - servicio de disponibilidad, 12-13
 - Servicios de seguridad avanzada: S/MIME, 166-67
 - etiquetas de seguridad, 166
 - listas de correo seguras, 166
 - recibos firmados, 166
 - SET, 112
 - SET (*Secure Electronic Transaction*), 62, 83, 227, 246-58
 - autorización de pago, 256-57
 - captura de pago, 258
 - características fundamentales, 248
 - firma dual, 250-52
 - introducción a, 247-50
 - mensaje de respuesta de autorización a, 257
 - mensaje de respuesta de captura, 258
 - mensaje de respuesta de compra, 255
 - mensaje de solicitud de autorización a, 256-57
 - mensaje de solicitud de captura, 258
 - mensaje de solicitud de compra, 254-55
 - participantes, 248-50
 - adquisidor o banco del vendedor, 249
 - autoridad de certificación (CA), 249
 - emisor o banco del comprador, 249
 - pasarela de pago, 249
 - titular de tarjeta, 248
 - vendedor, 248
 - procesamiento del pago, 252-58
 - requisitos, 247-48
 - secuencia de acciones para una transacción, 253

- solicitud de compra, 252-56
 tipos de transacción, 241
SHA (Secure Hash Algorithm), 64,
Véase SHA-1
SHA-1, 67, 130, 238-39, 252, 292
 procesamiento de mensaje, 64-67
 añadir bits de relleno, 65
 añadir longitud, 65
 initializar *buffer MD*, 65
 procesar mensaje en bloques
 de 512 bits (16 palabras), 65
 salida, 66
 y S/MIME, 158-59
Shamir, Adi, 75
Simple Network Management Protocol (RFC 1157), 264
Simple Web (Universidad de Twente), 298
Sistema de gestión de red (NMS), 291
Sistemas confiables o de confianza, 374-80
 concepto de, 376-78
 control de acceso a los datos, 374-78
 defensa frente a caballo de Troya, 378-80
 definición, 376-78
Sitio SETCo, 258
SMTP (Simple Mail Transfer Protocol), 369-72
SNMP (Simple Network Management Protocol), 262-98
 administrador tradicional SNMP, 274-77
 agente de gestión, 263
 agente tradicional SNMP, 276
 agentes *proxy*, 266
 arquitectura de gestión de red, 262-64
 arquitectura del protocolo de gestión de red, 264-65
 conceptos básicos de, 262-70
 estación de gestión, 263
 función de, 265
 MIB, 263, 264
 protocolo de gestión de red, 264
SNMPv1, 269-72
SNMPv2, 266-70
SNMPv3, 272-98
SNMPv1, 269-72
 comunidades y nombres de comunidad, 270-71
 perfil de comunidad SNMP, 271
 política de acceso, 271-72
 política de acceso SNMP, 271-72
 servicio de autenticación, 270-71
 servicio *proxy*, 272
SNMPv2, 266-70
 gestión de red distribuida, 267-68
 mejoras funcionales, 268-70
SNMPv3, 272-98
 aplicación de generación de comandos, 279
 aplicación de generación de notificación, 281
 aplicación de recepción de notificación, 281
 aplicación de reenvío mediante *proxy*, 281
 aplicación de respuesta a comandos, 279
 aplicaciones, 279-82
 arquitectura, 274
 control de acceso basado en vistas (VACM), 273, 293-98
 elementos del modelo VACM, 293-95
 entidad, 274-77
 funciones criptográficas, 284-85
 localización de claves, 290-93
 mecanismos de validez temporal USM, 288-90
 modelo de procesamiento de mensaje, 282-83
 modelo de seguridad de usuarios (USM), 273, 283-84
 motivación, 296-98
 motores autoritativos y no autoritativos, 285-86
 parámetros de mensaje USM, 286-88
 procesamiento de control de acceso, 295-96
 terminología, 277-79
SNMPv3, sitio web, 298
 Sociedad de Internet, 19
Software dañino, 341-59, *Véase Gusanos; Amenazas; Virus*
 amenazas, 342-53
 bomba lógica, 344
 caballos de Troya, 344
 gusanos, 351-53
 ataques actuales, 352-53
 gusano Código Rojo/Código Rojo II, 352-53
 gusano Morris, 352
 Nimda, 353
 vehículo de red, 351
 trampas, 342-43
 virus, 342-53
 contramedidas, 353-58
 estructura, 346-48
 fase de activación, 345
 fase de ejecución, 345
 fase de propagación, 345
 fase inactiva, 345
 naturaleza de, 345-50
 tipos de, 348-49
 virus de correo electrónico, 350-51
 virus de macro, 349-50
 virus furtivo, 348
 virus parásito, 348
 virus polimórfico, 348
 virus residente en memoria, 348
zombis, 344-45
Software de bloqueo de acciones, 357-58
SSL (Secure Sockets Layer), 83, 227, 227-41
 arquitectura, 227-29
 conexión SSL, 227
 estado de conexión, 229
 estado de sesión, 228
 sesión SSL, 227-28
 protocolo Alert, 227, 232-34
 protocolo Change Cipher Spec, 232
 protocolo Handshake, 234-40
 protocolo Record SSL, 229-32
SSL de Netscape, página web, 258
SSL/TLS, 112
Stoll, Cliff, 307
Structure and Identification of Management Information for TCP/IP-Based Networks (RFC 1155), 264
Suplantación, 8-9
Suplantación de dirección de IP, 368
Suplantación de dirección IP y routers de filtrado de paquetes, 368
Suplantador, 306

T

- Tamaños máximos de paquetes y *routers*, 214
TCP/IP (Transmission Control Protocol/Internet Protocol), 4, 15, 17, 180, 218, 265, 272, 370
Technical Committee on Security and Privacy de IEEE (Comité técnico para la seguridad y la privacidad), 23
 Técnica de educación del usuario, para selección de claves, 328
 Técnica Fortezza, 237-39

Técnicas avanzadas de antivirus, 355-57
 Telnet, 370, 371
 Teoría de números, 387-91
 aritmética modular, 390-91
 congruente módulo n, 390
 definición, 391
 residuo, 390
 números primos, 388-90
 definición, 389
 divisores, 388-89
 números primos relativos, 389-90
 máximo común divisor, 389-90
 Texto cifrado y cifrado de clave pública, 72
 Texto claro y cifrado de clave pública, 72
The Cryptography FAQ (sitio web de preguntas más frecuentes sobre criptografía), 23
 Tickets de capacidad, 375
 Tipo de acceso y determinación de acceso, 295
 Tipos de carga útil, ISAKMP, 207-10
 carga útil de certificado, 209
 carga útil de firma, 209
 carga útil de identificación, 207
 carga útil de intercambio de clave, 207
 carga útil de notificación, 209
 carga útil de propuesta, 207
 carga útil de SA, 207
 carga útil de solicitud de certificado, 209
 carga útil de transformación, 207
 carga útil *hash*, 209
 carga útil *nonce*, 209
 eliminación de carga útil, 210
 Tipos de contenido, S/MIME, 159-60
 Tipos de transacción, SET, 253
 Titular de tarjeta, SET, 248
TLS (Transport Layer Security), 68, 226, 241-46
 algoritmos de cifrado simétrico, 245
 cálculos criptográficos, 245-46
 códigos de alerta, 244
 formato de registro de TLS, 241
 función pseudoaleatoria, 242-43
 intercambios de claves, 245
 MAC, 241-42
 mensaje *certificate_verify*, 245
 mensaje *finished*, 245
 número de versión, 241
 relleno, 246

suites de cifrado, 245
 tipos de certificados de cliente, 245
 Tom Dunigan, página web sobre seguridad, 23
 Trampas, 342-43
 Transformación de contraseña a clave, Kerberos, 123
Transport Layer Security Charter, 258
 TripleDES (3DES), 37-38, 133, 141
 uso del término, 35 nota al pie y S/MIME, 159
 Triple DES de tres claves, y ESP, 194

U

UDP (*User Datagram Protocol*), Véase Protocolo de datagramas de usuario:
 Unidades de datos de protocolo (PDU), 214, 268, 274, 275
 extendida, 282
 USC/ISI, página web de Kerberos, 121
 USM, Véase Modelo de seguridad de usuarios:
 Usuario clandestino, 206
 Usuario fraudulento, 306

V

VAC, Véase Control de acceso basado en vistas:
 Valor previo de clave maestra, 239, 241
 Vehículos de red, gusanos, 351
 Vendedor, SET, 248
Verisign, 125
 certificados, 164-66
 Versión, certificado de usuario, 112
 Violaciones de la seguridad, 3
 Virus, 2-3, 16, 342-53
 enfoques de antivirus, 353-55
 exploradores de cuarta generación, 354
 exploradores de primera generación, 354
 exploradores de segunda generación, 354
 exploradores de tercera generación, 354
 software de bloqueo de acciones, 357-58

acciones supervisadas, 357-58
 contramedidas, 353-58
 desventajas de, 358
 estructura, 346-48
 fase de activación, 345
 fase de ejecución, 345
 fase de propagación, 345
 fase inactiva, 345
 naturaleza de, 345-51
 tipos de, 348-49
 ventajas de, 358
 virus de correo electrónico, 350-51
 virus de macro, 349-50
 virus furtivo, 348
 virus parásito, 348
 virus polimórfico, 348
 virus residente en memoria, 348
 técnicas avanzadas antivirus, 355-57
 descifrado genérico (GD), 355
 sistema de inmunidad digital, 356-57
 Virus de compresión, lógica de, 347
 Virus de correo electrónico, 350-51
 Virus de macro, 349-50
 Virus furtivo, 348
 Virus Melissa, 350
 Virus parásito, 348
 Virus polimórfico, 348
 Virus residente en memoria, 348
 Vista MIB, 294

W

Wales, Richard, 168
 Wily Hacker, incidente, (1986-87), 307
 Gusanos, 16, 352-53
 ataques actuales, 352-53
 Código rojo/Código Rojo II, 353
 gusano Morris, 352
 vehículos de red, 351

Z

ZIP, 134, 168-70
 algoritmo de compresión, 168-70
buffer de entrada, 170
buffer histórico deslizante, 169-70
 algoritmo de descompresión, 170
 Ziv, Jacob, 168
 Zombis, 344-45



Fundamentos de Seguridad en Redes

Aplicaciones y Estándares

Stallings

2^a Edición

En esta era de la conectividad electrónica universal, de virus y *hackers*, de escuchas y fraudes electrónicos, no hay un momento en el que no importe la seguridad. Dos tendencias han confluído para hacer de interés vital el tema de este libro. En primer lugar, el enorme crecimiento de los sistemas de computadores y sus interconexiones mediante redes ha hecho que organizaciones e individuos dependan cada vez más de la información que se almacena y se transmite a través de estos sistemas. Esto, a su vez, ha llevado a un aumento de la conciencia de la necesidad de proteger los datos y los recursos, de garantizar la autenticidad de los datos y los mensajes y de proteger los sistemas frente a ataques a la red. En segundo lugar, las disciplinas de la criptografía y la seguridad de la red han madurado, dando como resultado el desarrollo de aplicaciones prácticas, ya disponibles, para la seguridad de la red.

El propósito de este libro es proporcionar un estudio práctico sobre las aplicaciones y los estándares relativos a la seguridad de la red. Se resaltan, por una parte, las aplicaciones que más se utilizan en Internet y en las redes corporativas y, por otra, los estándares más extendidos, especialmente los de Internet.

El libro está destinado a una audiencia tanto académica como profesional. Como libro de texto, está diseñado para cubrir un curso de un semestre sobre seguridad en redes para estudiantes universitarios de ciencias de la computación, ingeniería de la computación e ingeniería eléctrica. También sirve como libro básico de referencia y es adecuado para el aprendizaje autónomo.

El libro se divide en tres partes:

- Primera parte. Criptografía
- Segunda parte. Aplicaciones de seguridad en redes
- Tercera parte. Seguridad de los sistemas

LibroSite es una página web, en *castellano*, propia del libro que ofrece un respaldo académico exhaustivo, tanto para los docentes como para los alumnos.

Los profesores pueden encontrar respuestas a los ejercicios, material adicional, sala de profesores, área de investigación, contribuciones, etc.

Para los estudiantes existen ejercicios adicionales, enlaces a recursos de Internet sobre el tema, buscadores de trabajo, sala de estudio y mucho más.



www.librosite.net/stallings4

PEARSON
Educación

www.pearsoneducacion.com

ISBN 978-84-832-2922-4



9 788483 229224