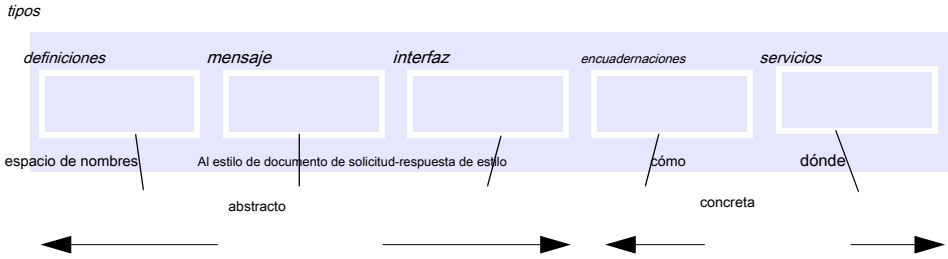
Figura 9.10 Los elementos principales de una descripción WSDL



9.3 Descripciones del servicio y IDL para servicios web

Se necesitan definiciones de interfaz para permitir que los clientes se comuniquen con los servicios. Para los servicios web, definiciones de interfaz se proporcionan como parte de una más general *Descripción del servicio,* que especifica otras dos características adicionales - cómo los mensajes deben ser comunicados (por ejemplo, mediante SOAP sobre HTTP) y el URI del servicio. Para atender para su uso en un entorno multi-idioma, las descripciones de servicio están escritos en XML.

Una descripción del servicio es la base de un acuerdo entre un cliente y un servidor como para el servicio que se ofrece. Reúne todos los hechos relacionados con el servicio que son relevantes para sus clientes. descripciones de los servicios se utilizan generalmente para generar recibos de clientes que implementan automáticamente el comportamiento correcto para el cliente.

El componente de IDL-como de una descripción de servicio es más flexible que otros IDLs, en que un servicio puede ser especificado, ya sea en términos de los tipos de mensajes que va a enviar y recibir o en términos de las operaciones que apoya, para permitir tanto intercambio de documentos y las interacciones de tipo respuesta petición.

Una variedad de diferentes métodos de comunicación puede ser utilizado por los servicios web y sus clientes. Por lo tanto se deja el método de comunicación que ser decidido por el proveedor de servicios y se especifica en la descripción del servicio, en lugar de integrada en el sistema, ya que es en CORBA, por ejemplo.

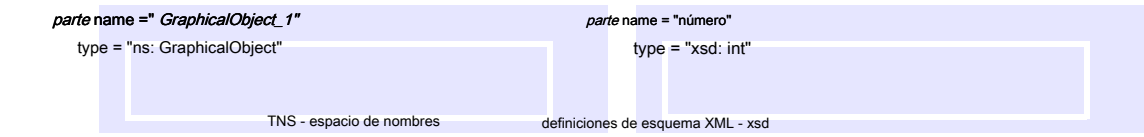
La capacidad para especificar el URI de un servicio como parte de la descripción del servicio evita la necesidad de que el servicio de aglutinante o de nombres separado utilizado por la mayoría de otros middleware. Cuenta con la implicación de que el URI no se puede cambiar una vez que la descripción del servicio se ha puesto a disposición de los clientes potenciales, pero el esquema URN atiende a un cambio de ubicación por lo que permite un direccionamiento indirecto en el nivel de referencia.

Por el contrario, en el enfoque de aglutinante, el cliente utiliza un nombre para buscar la referencia de servicio en tiempo de ejecución, lo que permite a las referencias de servicios cambian con el tiempo. Este enfoque requiere un direccionamiento indirecto de un nombre a una referencia de servicio para todos los servicios, a pesar de que muchos de ellos pueden permanecer siempre en el mismo lugar.

En el contexto de los servicios web, el Web Services Description Language (WSDL) se utiliza comúnmente para las descripciones de servicio. La versión actual, WSDL 2.0 [www.w3.org XI], Se convirtió en una Recomendación W3C en 2007. Se define un esquema XML para la representación de los componentes de una descripción del servicio, que incluyen, por ejemplo, los nombres de los elementos *definiciones, tipos, mensaje, interfaz, fijaciones* y *servicios*.

WSDL separa la parte abstracta de una descripción del servicio de la parte de hormigón, como se muestra en la Figura 9.10.

Figura 9.11 solicitud WSDL y mensajes de respuesta para el *nueva forma* operación



La parte resumen de la descripción incluye un conjunto de definiciones de los tipos utilizado por el servicio - en particular, los tipos de los valores intercambiados en los mensajes. El ejemplo Java de la Sección 9.2.3, cuya interfaz Java se muestra en la Figura 9.7, utiliza los tipos de Java *En t* y *GraphicalObject*. El primero (al igual que cualquier tipo básico) se puede traducir directamente en el equivalente XML, pero *GraphicalObject* se define en Java en términos de los tipos *int, String* y *booleano. GraphicalObject* se representa en XML, para su uso común por parte de los clientes heterogéneos, como una *tipoCompuesto* que consiste en una *secuencia* de tipos XML denominados incluyendo, por ejemplo:

<Elemento name = tipo "llena" = "boolean" /> <elemento name = tipo "originx" = "int" />

El conjunto de nombres definido dentro de la *tipos* sección de una definición WSDL se llama su *espacio de nombres.* los *mensaje* sección de la parte extracto contiene una descripción del conjunto de mensajes intercambiados. Para el estilo de documento de interacción, estos mensajes se pueden utilizar directamente. Para el estilo de petición-respuesta de la interacción, hay dos mensajes para cada operación, que se utilizan para describir las operaciones en el *interfaz* sección. La pieza de hormigón especifica cómo y dónde se puede contactar al servicio.

La modularidad inherente de una definición WSDL permite que sus componentes pueden combinar de diferentes maneras - por ejemplo, la misma interfaz se puede utilizar con diferentes fijaciones o lugares. Los tipos pueden ser definidos dentro de la *tipos* elemento o pueden ser definidos en un documento separado que se hace referencia por un URI de la *tipos* elemento. En el último caso, las definiciones de tipo pueden ser referenciados de varios diversos documentos WSDL.

Mensajes u operaciones • En los servicios web, todo lo que el cliente y el servidor es necesario tener una idea común acerca de los mensajes que se intercambian. Para un servicio basado en el intercambio de un pequeño número de diferentes tipos de documentos, simplemente WSDL describe los tipos de los diferentes mensajes que se intercambian. Cuando un cliente envía uno de estos mensajes a un servicio web, este último decide qué operación realizar y qué tipo de mensaje a enviar de vuelta al cliente sobre la base del tipo de mensaje recibido. En nuestro ejemplo de Java, dos mensajes serán definidos para cada una de las operaciones en la interfaz - una para la solicitud y uno por la respuesta. Por ejemplo, la Figura 9.11 muestra la petición y mensajes de respuesta para la *nueva forma* operación, que tiene un solo argumento de entrada de tipo *GraphicalObject* y un solo argumento de salida de tipo *En t*.

Para los servicios que soportan varias operaciones diferentes, es más eficaz para especificar los mensajes intercambiados como solicitudes de operaciones con los argumentos y sus respuestas correspondientes, lo que permite el servicio de despachar cada solicitud a la apropiada

Figura 9.12 patrones de intercambio de mensajes para operaciones WSDL

Nombre	*Los mensajes enviados por*			
	el Cliente	*Servidor*	*Entrega*	*mensaje de fallo*
En fuera	*Solicitud*	*Respuesta*		puede sustituir *Respuesta*
Sólo en	*Solicitud*			No hay ningún mensaje de fallo
En robusta de sólo *Solicitud*			Puede ser enviado garantizada	
Salir en	*Respuesta*	*Solicitud*		puede sustituir *Respuesta*
Sólo fuera		*Solicitud*		No hay ningún mensaje de fallo
Sólo robusta fuera		*Solicitud*	Garantizada la posibilidad de enviar culpa	

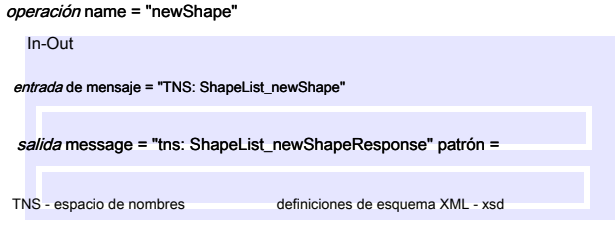
operación. Sin embargo, en una operación WSDL es un constructo para relacionar la solicitud y responder mensajes, en contraste con la definición de una operación en una interfaz de servicio.

Interfaz • El conjunto de operaciones que pertenecen a un servicio web se agrupan juntos en un elemento XML denominado *interfaz (* aveces llamado *portType).* Cada operación debe especificar el *patrón de intercambio de mensajes* entre el cliente y el servidor. Las opciones disponibles incluyen aquellos mostrados en la Figura 9.12 . El primero, *En fuera,* es el de uso común formulario de solicitud-respuesta de la comunicación cliente-servidor. En este modelo, el mensaje de respuesta puede ser sustituido por un mensaje de fallo. *Sólo en* es para los mensajes unidireccionales con *tal vez* y la semántica *Sólo fuera* es para *de una sola mano* mensajes desde el servidor al cliente; avisos de fallo no se pueden enviar con cualquiera. *En robusta de sólo* y *Sólo robusta fuera* son los mensajes correspondientes con entrega garantizada; mensajes de fallo pueden ser intercambiados.

Salir en es una interacción de petición-respuesta iniciada por el servidor. WSDL 2.0 también es extensible en que las organizaciones pueden introducir sus propios patrones de intercambio de mensajes si los predefinidos resultan ser insuficientes.

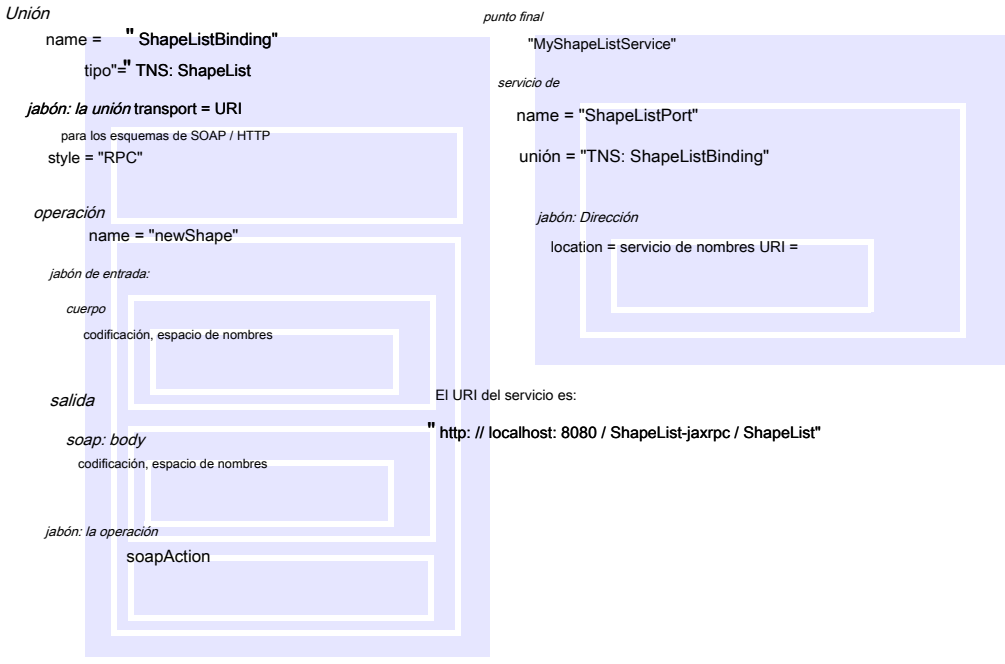
Volviendo a nuestro ejemplo de Java, cada una de las operaciones se define para tener una *En fuera* patrón. La operacion *nueva forma* se muestra en la figura 9.13, utilizando los mensajes definidos en la Figura 9.11 . Esta definición, junto con las definiciones de las otras cuatro operaciones se ser encerrado en un XML *interfaz* elemento. Una operación puede también especificar los mensajes de fallo que pueden ser enviadas.

Figura 9.13 operación WSDL *nueva forma*



Los nombres *operación,* patrón, *entrada* y *salida* se definen en el esquema XML de WSDL

Figura 9.14 unión de SOAP y de servicios definiciones



Si, por ejemplo, una operación tiene dos argumentos - decir, un entero y una cadena - no hay necesidad de definir un nuevo tipo de datos, ya que estos tipos se definen para los esquemas XML. Sin embargo, será necesario definir un mensaje que tiene estas dos partes. Este mensaje se puede usar entonces como una entrada o salida en la definición para la operación.

Herencia: Cualquier interfaz WSDL puede extenderse uno o más de otros interfaces WSDL. Esta es una forma simple de la herencia en la que una interfaz compatible con las operaciones de las interfaces se extiende además de los que se define. No se permite la definición recursiva de interfaces; es decir, si la interfaz de B se extiende interfaz A, a continuación, la interfaz A no puede extenderse interfaz B.

pieza de hormigón • La parte restante (hormigón) de un documento WSDL consiste en la *Unión (* la elección de protocolos) y la *Servicio (* La elección del punto final o la dirección del servidor). Los dos están relacionados, ya que la forma de la dirección depende del tipo de protocolo en uso. Por ejemplo, un extremo SOAP utilizará un URI mientras que un punto final de CORBA utilizará un identificador de objeto CORBA-específico.

Unión: los *Unión* sección de un documento WSDL dice qué formatos de mensaje y forma de representación externa de datos se van a utilizar. Por ejemplo, los servicios web a menudo utilizan SOAP, HTTP y MIME. Encuadernaciones pueden estar asociados con las operaciones o interfaces particulares, o pueden dejarse libres para el uso de una variedad de diferentes servicios web.

Figura 9.14 muestra un ejemplo de una *Unión* adjuntando *jabón: la unión* que especifica la dirección URL de un protocolo particular para la transmisión de sobres SOAP: el enlace HTTP para SOAP. Los atributos opcionales de este elemento también puede especificar lo siguiente:

- el patrón de intercambio de mensajes, que puede ser o bien *RPC (* solicitud-respuesta) o *documento* intercambiar - El valor predeterminado es *documento;*

- el esquema XML para los formatos de los mensajes - el valor predeterminado es SOAP *sobre;*
- el esquema XML para la representación de datos externos - el valor por defecto es la codificación SOAP XML.

Figura 9.14 también muestra los detalles de los enlaces para una de las operaciones (*nueva forma),* especificando que tanto el *entrada* y el *salida* mensaje debe viajar en una *soap: body,* usando un estilo de codificación particular, y que la operación debe ser transmitido como una *soapAction*.

Servicio: Cada *Servicio* elemento de un documento WSDL especifica el nombre del servicio y uno o más *(criterios de valoración* o puertos) donde puede poner en contacto una instancia del servicio. Cada una de las *punto final* elementos se refiere al nombre de la unión en uso y, en el caso de un jabón de unión, utiliza una *jabón: Dirección* elemento para especificar el URI de la ubicación del servicio.

• **documentación** Tanto la información humano-y-legible por máquina se puede insertar en una *documentación* elemento en la mayoría de los puntos dentro de un documento WSDL. Esta información puede ser retirado antes de WSDL se utiliza para el procesamiento automático, por ejemplo, por los compiladores de código auxiliar.

• **utilizar WSDL** documentos completos WSDL se puede acceder a través de sus URIs por los clientes y servidores, ya sea directa o indirectamente a través de un servicio de directorio, como UDDI. Las herramientas están disponibles para la generación de definiciones WSDL de información proporcionada a través de una interfaz gráfica de usuario, eliminando la necesidad de los usuarios a estar involucrados en los detalles complejos y la estructura de WSDL. Por ejemplo, el Web Services Description Language Toolkit para Java (WSDL4J) permite la creación, representación y manipulación de documentos WSDL que describe los servicios [wsdl4j.sourceforge.org]. definiciones WSDL también se pueden generar a partir de definiciones de interfaz escritas en otros lenguajes, como Java JAX-RPC, discutidos en la Sección 9.2.1.

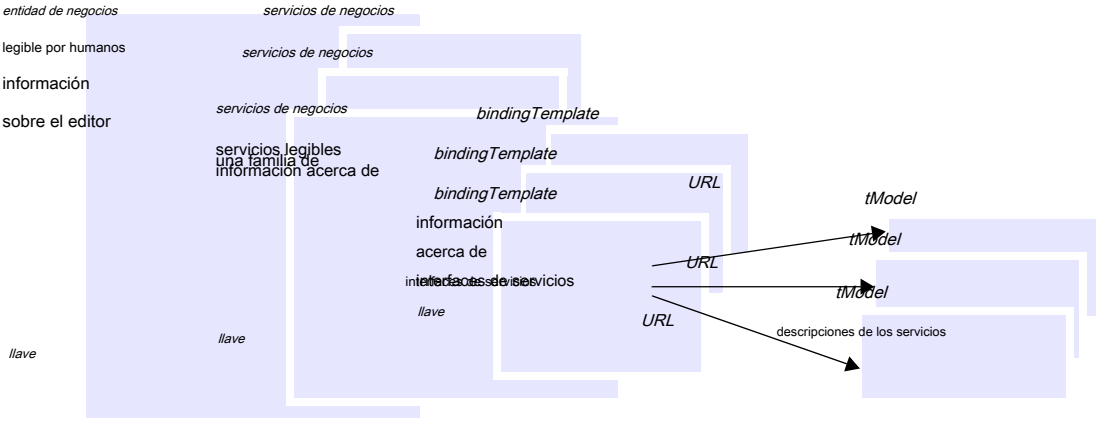
9.4 Un servicio de directorio para su uso con los servicios web

Hay muchas maneras en las que los clientes pueden obtener descripciones de los servicios. Por ejemplo, cualquier persona que preste un servicio web de nivel superior como el servicio de agente de viajes se discutió en la sección 9.1 es casi seguro que hacer una publicidad de páginas web del servicio y los clientes potenciales vendría a través de la página web en la búsqueda de los servicios de ese tipo.

Sin embargo, cualquier organización que planea basar sus aplicaciones en servicios web le resultará más conveniente utilizar un servicio de directorio para realizar estos servicios a disposición de los clientes. Este es el propósito de la Universal Description, Discovery and Integration servicio (UDDI) [Bellwood *et al.* 2003], que proporciona un servicio de nombres y un servicio de directorio (véase la Sección 13.3). Es decir, las descripciones de servicio WSDL pueden ser consultados por el nombre (un servicio de páginas en blanco) o por atributos (un servicio de páginas amarillas). También se puede acceder directamente a través de sus direcciones URL, que es conveniente para los desarrolladores que están diseñando programas de cliente que utilizan el servicio.

Los clientes pueden utilizar el enfoque de páginas amarillas para buscar una categoría particular de servicio, como agente de viajes o librería, o pueden usar las páginas blancas acercan a buscar un servicio con referencia a la organización que proporciona.

Figura 9.15 Las estructuras de datos UDDI principal



Estructuras de datos • Las estructuras de datos que apoyan UDDI están diseñados para permitir que todos los estilos anteriores de acceso y pueden incorporar cualquier cantidad de información legible por humanos. Los datos se organizan en términos de las cuatro estructuras que se muestran en la Figura 9.15 , cada uno de los cuales se puede acceder individualmente por medio de un identificador de llamada *llave (* aparte de *tModel,* que se puede acceder por una URL):

- *entidad de negocios* describe la organización que proporciona estos servicios web, dando a su nombre, dirección y actividades, etc .;
- *servicios de negocios* almacena información sobre un conjunto de instancias de un servicio web, como su nombre y una descripción de su propósito (por ejemplo, agente de viajes o librería);
- *bindingTemplate* ejerce en la dirección de una instancia de servicio web y referencias a descripciones de los servicios;
- *tModel* ejerce en las descripciones de servicios, por lo general los documentos WSDL, almacenados fuera de la base de datos y se accede por medio de direcciones URL.

Buscar • UDDI proporciona una API para buscar servicios basados en dos conjuntos de operaciones de consulta:

- los *obtener_xxx* conjunto de operaciones incluye *get_BusinessDetail, get_ServiceDetail, get_bindingDetail* y *get_tModelDetail;* que recuperan una entidad que corresponde a una determinada clave.
- los *find_xxx* conjunto de operaciones incluye *find_business, find_service, find_binding* y *find_tModel;* que recuperan el conjunto de entidades que coincide con una serie de criterios de búsqueda, proporcionando un resumen de los nombres, descripciones, las llaves y las direcciones URL. De esta manera los clientes en posesión de una clave particular pueden utilizar una *obtener_xxx* operación para recuperar la entidad correspondiente directamente, y otros clientes pueden utilizar la navegación para ayudar con las búsquedas, a partir de un gran conjunto de resultados y poco a poco reducir la lista. Por ejemplo, pueden comenzar utilizando la *find_business* operación con el fin de obtener una lista que contiene un resumen

de información sobre los proveedores de juego. A partir de este resumen, el usuario puede utilizar el *find_service* operación para limitar la búsqueda, haciendo coincidir el tipo de servicio requerido. En ambos casos, van a encontrar la clave de un adecuado *bindingTemplate* y por lo tanto encontrar la dirección URL para recuperar el documento WSDL para un servicio adecuado.

Además, UDDI proporciona una interfaz de notificar / suscripción por el que los clientes se registren interés en un determinado conjunto de entidades en un registro UDDI y obtener notificaciones de cambio, ya sea sincrónica o asincrónica.

• **publicación** UDDI proporciona una interfaz para publicar y actualizar información acerca de los servicios web. La primera vez que una estructura de datos (véase la figura 9.15) se publica en un servidor UDDI, se le da una clave en forma de un URI - por ejemplo, *UDDI: cdk5.net: 213*

- y ese servidor se convierte en su propietario.

• **registros** El servicio UDDI se basa en los datos replicados almacenados en registros. Un registro UDDI consiste en uno o más servidores UDDI, cada uno de los cuales tiene una copia del mismo conjunto de datos. Los datos se replica entre los miembros de un registro. Cada uno de ellos puede responder a consultas y publicar información. Los cambios en una estructura de datos deben presentarse a su dueño - es decir, el servidor en el que se publicó por primera vez. Es posible que un propietario de transmitir la propiedad a otro servidor UDDI en el mismo registro.

esquema de replicación: Los miembros de un registro propagan copias de las estructuras de datos entre sí de la siguiente manera: un servidor que ha hecho cambios notifica a los otros servidores en el registro, que luego solicitan los cambios. Una forma de marca de tiempo vector se utiliza para determinar cuál de los cambios debe ser propagado y se aplica. El esquema es simple en comparación con otros esquemas de replicación que utilizan las marcas de tiempo de vectores, tales como chisme (Sección 18.4.1) o Coda (Sección 18.4.3) por dos razones:

1. Todos los cambios en una estructura de datos particular, se realizan en el mismo servidor.
2. Las actualizaciones desde un servidor en particular se reciben en orden secuencial por los otros miembros, pero sin orden particular se impone entre las operaciones de actualización realizadas por diferentes servidores.

La interacción entre los servidores: Como se describió anteriormente, los servidores interactúan entre sí para llevar a cabo el esquema de replicación. También pueden interactuar con el fin de transferir la propiedad de las estructuras de datos. Sin embargo, la respuesta a una operación de búsqueda es realizada por un único servidor sin ninguna interacción con otros servidores en el registro, a diferencia de en el servicio de directorio X.500 (Sección 13.5), en la que está particionada de datos entre servidores que cooperan uno con el otro en encontrar el servidor relevante para una determinada solicitud.

la seguridad XML 9.5

la seguridad XML consiste en un conjunto de diseños del W3C relacionados sobre firma, gestión de claves y cifrado. Está destinado para su uso en el trabajo cooperativo a través de Internet que implica documentos cuyo contenido puede necesitar ser autenticada o cifrado. Típicamente se crean los documentos, intercambian, almacenan y luego intercambian de nuevo, posiblemente después de haber sido modificada por una serie de diferentes usuarios.

WS-Security [Kaler 2002] es otro enfoque de la seguridad que se ocupa de la aplicación de la integridad del mensaje, confidencialidad y autenticación de mensajes solo mensaje de SOAP.

Como un ejemplo de un contexto en el que la seguridad XML sería útil, considere un documento que contiene los registros médicos del paciente. Las diferentes partes de este documento se utilizan en la consulta del médico de cabecera y en las diversas clínicas especializadas y hospitales visitados por el paciente. Se actualizará por los médicos, enfermeras y consultores que hacen notas sobre la condición y el tratamiento del paciente, por los administradores de hacer citas y por los farmacéuticos que proporcionan la medicina. Diferentes partes del documento serán visibles por las diferentes funciones mencionadas anteriormente, y, posiblemente, el paciente también. Es esencial que ciertas partes del documento, por ejemplo, las recomendaciones en cuanto al tratamiento, se pueden atribuir a la persona que les hizo y se puede garantizar que no ha sido alterado.

Estas necesidades no pueden ser satisfechas por TLS (anteriormente conocido como SSL y describe en la Sección 11.6.3), que se utiliza para crear un canal seguro para la comunicación de información. Permite a los procesos en los dos extremos del canal para negociar como a la necesidad de autenticación o cifrado y las claves y algoritmos para ser utilizados, tanto cuando un canal está configurado y durante su vida útil. Por ejemplo, los datos sobre una transacción financiera podrían ser firmada y enviada a la clara hasta que la información confidencial, como datos de la tarjeta de crédito se debe dar, a la que se aplica el cifrado punto.

Para permitir que el nuevo tipo de uso se indica más arriba, la seguridad debe ser especificado dentro del propio documento y se aplica al documento en lugar de como una propiedad del canal que transmitirá desde un usuario a otro.

Esto es posible en XML u otros formatos de documentos estructurados, en los que los metadatos se puede utilizar. Las etiquetas XML se pueden utilizar para definir las propiedades de los datos en el documento. En particular, la seguridad XML depende de nuevas etiquetas que se pueden utilizar para indicar el comienzo y el final de las secciones de datos cifrados o firmados y de las firmas. Una vez que la seguridad necesaria se ha aplicado dentro de un documento, puede ser enviada a una variedad de diferentes usuarios, incluso por medio de multidifusión.

Requerimientos básicos • la seguridad XML debe proporcionar al menos el mismo nivel de protección que TLS. Es decir:

Para ser capaz de cifrar o bien un documento completo o sólo algunas partes seleccionadas de la misma: Por ejemplo, considere la información acerca de una transacción financiera, que incluye el nombre de una persona, el tipo de transacción y los detalles sobre el crédito o débito que se utiliza. En uno de los casos, solo los datos de la tarjeta podrían estar ocultos, por lo que es posible identificar la transacción antes de descifrar el registro. En otro caso, el tipo de transacción también podría ser oculto, por lo que los extranjeros no pueden saber si se trata, por ejemplo, una orden o un pago.

Para poder firmar ya sea un documento completo o sólo algunas partes seleccionadas de la misma: Cuando se pretende un documento que se utilizará para el trabajo cooperativo por un grupo de personas, no puede haber algunas partes críticas del documento que debe ser firmado con el fin de garantizar que se hicieron por una persona en particular o que no se han cambiado. Pero también es útil poder contar con otras partes que pueden ser alterados durante el uso del documento - estos no deben ser firmados.

requisitos básicos adicionales • Otros requisitos surgen de la necesidad de almacenar documentos, posiblemente modificarlos y luego enviarlos a una variedad de diferentes destinatarios:

Para agregar a un documento que ya está firmado y para firmar el resultado: Por ejemplo, Alice puede firmar un documento y pasarlo a Bob, que 'testigos de su firma' mediante la adición de una observación a tal efecto y después de firmar el documento completo. (Sección 11.1 se presentan los nombres, entre ellos Alice y Bob, que se utiliza para los protagonistas en los protocolos de seguridad.)

Para autorizar a los diferentes usuarios para ver distintas partes de un documento: En el caso de un registro médico, un investigador puede ver alguna sección en particular de los datos médicos, un administrador puede ver los detalles personales y un médico puede ver ambos.

Para agregar a un documento que ya contiene secciones cifrados y cifrar parte de la nueva versión, posiblemente incluyendo algunas de las secciones ya cifrados.

Las capacidades de flexibilidad y estructuración de la notación XML permiten hacer todo lo anterior, sin adiciones al régimen derivado de los requisitos básicos.

Disposiciones aplicables a los algoritmos • XML documentos seguros están firmados y / o cifrados mucho antes de cualquier consideración en cuanto a quién será el acceso a ellos. Si el autor ya no está involucrado, no es posible negociar los protocolos y si se debe utilizar la autenticación o cifrado. Por lo tanto:

La norma debe especificar un conjunto de algoritmos que pueden proporcionar en cualquier aplicación de seguridad XML: Al menos un cifrado y un algoritmo de firma debe ser obligatorio, para permitir que la mayor cantidad posible de interoperabilidad. Otros algoritmos opcionales deben ser proporcionados para su uso dentro de los grupos más pequeños.

Los algoritmos utilizados para el cifrado y la autenticación de un documento en particular deben ser seleccionados de esa suite y los nombres de los algoritmos utilizados deben ser referenciados dentro del propio documento XML: Si los lugares donde se utilizará el documento no se pueden predecir, entonces uno de los protocolos requeridos deben usarse. la seguridad XML define los nombres de los elementos que se pueden utilizar para especificar el URI del algoritmo utilizado para firmar o cifrado. Con el fin de ser capaz de seleccionar una variedad de algoritmos dentro del mismo documento XML, un elemento que especifica un algoritmo general se anida dentro de un elemento que contiene información firmada o datos cifrados.

Requisitos para la búsqueda de llaves • Cuando se crea un documento y cada vez que se actualiza, teclas correspondientes debe ser elegido, sin ningún tipo de negociación con las partes que pueden acceder al documento en el futuro. Esto lleva a los siguientes requisitos:

Para ayudar a los usuarios de los documentos seguros con la búsqueda de las claves necesarias: Por ejemplo, un documento que incluye datos firmados debe contener información sobre la clave pública que se utiliza para validar la firma, tales como un nombre que puede ser utilizado para obtener la clave o un certificado. UN *KeyInfo* elemento se puede usar para este propósito.

Para hacer posible que los usuarios que cooperaron para ayudar el uno al otro con las teclas: Siempre que el *KeyInfo* elemento no está obligado a criptográficamente la propia firma, la información puede ser añadido sin romper la firma digital. Por ejemplo, supongamos que Alicia firma un documento y lo envía a Bob con una *KeyInfo* elemento que especifica sólo el nombre de la clave. Cuando Bob recibe el documento que recupera la información necesaria para validar la firma y añade esto a la *KeyInfo* elemento cuando pasa el documento a Carol.

Figura 9.16 Algoritmos necesarios para la firma XML

Tipo de algoritmo	*Nombre del algoritmo*	*Requerido*	*referencia*
Resumen del mensaje	SHA-1	Necesario	sección 11.4.3
codificación	base64	Necesario	[Freed y Borenstein 1996]
Firma	DSA con SHA-1	Obligatorio	[NIST 1994]
(asimétrico)	RSA con SHA-1	Sección 11.3.2 Recomendado	
MAC firma (simétrica)	HMAC-SHA-1	Necesario	Sección 11.4.2 y Krawczyk *et al. [* 1997]
Canonicalización	Canonical XML	Necesario	página 409

El elemento KeyInfo • la seguridad XML especifica una *KeyInfo* elemento para indicar la clave que se utilizará para validar una firma o para descifrar algunos datos. Puede contener, por ejemplo, certificados, los nombres de las teclas o algoritmos de clave del acuerdo. Su uso es opcional: el firmante puede no querer revelar ninguna información clave para todas las partes que tienen acceso al documento, y en algunos casos la aplicación que utiliza la seguridad XML puede ya tener acceso a las llaves en uso.

• XML canónica Algunas aplicaciones pueden hacer cambios que no tienen ningún efecto sobre el contenido de información real de un documento XML. Esto se debe a que hay una variedad de diferentes formas de representar lo que es lógicamente el mismo documento XML. Por ejemplo, los atributos pueden estar en diferentes órdenes y diferentes codificaciones de caracteres se puede utilizar, sin embargo, el contenido de información es equivalente. Canonical XML [www.w3.org X] Fue diseñado para su uso con las firmas digitales, que se utilizan para garantizar que el contenido de información de un documento no ha sido modificado. elementos XML se canónicas antes de ser firmado y el nombre del algoritmo de canonización se almacena, junto con la firma. Esto permite que el mismo algoritmo que se utiliza cuando se valida la firma.

La forma canónica es una serialización estándar de XML como un flujo de bytes. Se añade atributos por defecto y elimina esquemas superfluos, poniendo los atributos y las declaraciones de esquema en orden lexicográfico en cada elemento. Se utiliza una forma estándar para saltos de línea y la codificación UTF-8 para los caracteres. Cualquiera de los dos documentos XML equivalentes tienen la misma forma canónica.

Cuando un subconjunto de un documento XML - decir un elemento - está en forma canónica, la forma canónica incluye el contexto antepasado, es decir, los espacios de nombres declarados y los valores de los atributos. Así, cuando XML canónica se utiliza en conjunción con la firma digital, la firma de un elemento no pasará su validación si ese elemento se coloca en un contexto diferente.

Una variación de este algoritmo, llamado Exclusivo Canonical XML, omite el contexto de la serialización. Esto podría ser utilizado si la aplicación tiene la intención firmado un elemento en particular para ser utilizado en diferentes contextos.

El uso de firmas digitales en XML • La especificación para las firmas digitales en XML [www.w3.org XII] Es una recomendación del W3C que define nuevos tipos de elementos XML para mantener las firmas, los nombres de los algoritmos, claves y referencias a información firmada. Los nombres proporcionados en esta memoria descriptiva se definen en el esquema XML Signature cuales

Figura 9.17 Algoritmos necesarios para el cifrado XML (también se requiere que los algoritmos en la Figura 9.16)

Tipo de algoritmo	*Nombre del algoritmo*	*Necesario*	*referencia*
cifrado de bloques	TripleDES, AES-128, AES-256	Necesario	sección 11.3.1
	AES-192	Opcional	
codificación	base64	Necesario	[Freed y Borenstein 1996]
transporte de claves	RSA-v1.5, RSA-OAEP	Necesario	Sección 11.3.2 [Kaliski y Staddon 1998]
Simétrica clave de recapitulación (firma de clave compartida)	TripleDES KeyWrap, AES-128 KeyWrap, AES-256KeyWrap	Necesario	[Housley 2002]
	AES-192 opcional KeyWrap		
el acuerdo de claves	Diffie-Hellman	Opcional	[Rescorla, 1999]

incluye los elementos *Firma, SignatureValue SignedInfo* y *KeyInfo*. Figura 9.16 se muestran los algoritmos que deben estar disponibles en una aplicación de firma XML.

• **servicio de gestión de claves** La especificación del servicio de gestión de claves XML [www.w3.org XIII] Contiene protocolos para la distribución y registro de claves públicas para su uso en firmas XML. A pesar de que no requiere ninguna infraestructura de clave pública particular, el servicio está diseñado para ser compatible con los ya existentes, por ejemplo,
certificados X.509 (Sección 11.4.4), SPKI (la simple Infraestructura de Clave Pública, Sección 11.4.4) o los identificadores de claves PGP (Pretty Good Privacy, Sección 11.5.2). Los clientes pueden utilizar este servicio para encontrar la clave pública de una persona. Por ejemplo, si Alice quiere enviar un correo electrónico cifrado a Bob, se puede utilizar este servicio para obtener su clave pública. En otro ejemplo, Bob recibe un documento firmado de Alice que contiene su
certificado X.509 y luego pide el servicio de información clave para extraer la clave pública.

• **cifrado XML** El estándar para la codificación en XML se define en una recomendación del W3C que especifica tanto la forma de representar los datos cifrados en XML y el proceso de cifrado y descifrado de que [www.w3.org XIV]. Se introduce una
EncryptedData elemento para encerrar las porciones de datos cifrados.

Figura 9.17 especifica los algoritmos de cifrado que se deben incluir en una implementación de cifrado XML. algoritmos de cifrado de bloque se utilizan para cifrar los datos, y la codificación base64 se utiliza en XML para la representación de las firmas digitales y datos cifrados. algoritmos de transporte clave son algoritmos de cifrado de clave pública diseñados para su uso en el cifrado y descifrado las claves mismos.

Los algoritmos de clave simétrica de envoltura son compartidos algoritmos de cifrado de clave secreta diseñados para cifrar y descifrar las claves simétricas por medio de otra de las claves. Esto podría ser utilizado si una clave se incluyera en una *KeyInfo* elemento.

Un algoritmo de clave acuerdo permite una clave secreta compartida que se deriva de un cálculo en un par de claves públicas. Este algoritmo se pone a disposición para su uso por las aplicaciones que necesitan para acordar una clave compartida sin ningún cambio. No se aplica por el propio sistema de seguridad XML.

Figura 9.18 escenario agente de viajes

1. El cliente solicita el servicio de agente de viajes para obtener información sobre un conjunto de servicios; Por ejemplo, vuelos, alquiler de coches y reservas de hotel.
2. El servicio de agente de viajes recoge los precios y la disponibilidad de información y la envía al cliente, que elige uno de estos procedimientos en nombre del usuario: (a) ajustar la consulta, posiblemente con más proveedores para obtener más información, a continuación, repita el paso 2. (b) hacer la reserva. (C) Salir.
3. El cliente solicita una reserva y la agencia de viajes cheques de servicio de disponibilidad.
4. *Ya sea* todos están disponibles; *o* para los servicios que no están disponibles; *ya sea* alternativas se ofrecen al cliente, que se remonta a la etapa 3; *o* el cliente vuelve al paso 1.
5. Tomar depósito.
6. No dar al cliente un número de reserva como confirmación.
7. Durante el período hasta el pago final, el cliente puede modificar o cancelar las reservas.

9.6 La coordinación de los servicios web

La infraestructura de SOAP soporta interacciones de solicitud-respuesta individuales entre los clientes y los servicios web. Sin embargo, muchas aplicaciones útiles implican varias solicitudes que necesitan ser hechas en un orden determinado. Por ejemplo, al reservar un vuelo, la información de precios y la disponibilidad se recoge antes de hacer las reservas. Cuando un usuario interactúa con las páginas web a través de un navegador, por ejemplo, para reservar un vuelo o para hacer una oferta en una subasta, la interfaz proporcionada por el navegador (que se basa en la información proporcionada por el servidor) controla la secuencia en el que se realizan las operaciones.

Sin embargo, si se trata de un servicio de web que está haciendo reservas, como el servicio de agente de viajes muestra en la Figura 9.2, que el servicio web necesita trabajar a partir de una descripción de la forma adecuada de proceder al interactuar con otros servicios se utilizan para, por ejemplo, coche contratar y reservas de hotel, así como las reservas de vuelos. Figura 9.18 muestra un ejemplo de tales una descripcion.

Este ejemplo ilustra la necesidad de servicios web como clientes a contar con una descripción de un determinado protocolo a seguir en la interacción con otros servicios web. Pero también existe el problema de mantener la consistencia en los datos del servidor cuando se está recibiendo y respondiendo a las peticiones de varios clientes. Los capítulos 16 y 17 discuten las transacciones, que ilustran los problemas por medio de una serie de transacciones bancarias. Como un

sencillo ejemplo, en una transferencia de dinero entre dos cuentas bancarias, consistencia requiere que tanto el depósito en una cuenta y la retirada de la otra debe ser realizada. Capítulo 17 presenta la confirmación de dos fases protocolo que se utiliza por los servidores que cooperan para garantizar la coherencia de las transacciones.

En algunos casos, las transacciones atómicas se adaptan a los requisitos de las aplicaciones que utilizan los servicios web. Sin embargo, las actividades como las de la agencia de viajes tardan mucho tiempo en completarse, y sería poco práctico usar una confirmación en dos fases protocolo para llevarlas a cabo, ya que implica mantener recursos bloqueados por largos períodos de tiempo. Una alternativa es utilizar un protocolo más relajado en el que cada participante realiza cambios en estado persistente a medida que ocurren. En el caso de fallo, un protocolo de nivel de aplicación se utiliza para deshacer estas acciones.

En middleware convencional, la infraestructura proporciona un simple protocolo de petición-respuesta, dejando a otros servicios tales como transacciones, persistencia y seguridad a implementarse como servicios de mayor nivel separadas que se pueden utilizar cuando se necesitan. Lo mismo es cierto para los servicios web, donde el W3C y otros han estado poniendo en el esfuerzo hacia la definición de servicios de mayor nivel.

Se ha trabajado sobre un modelo general para la coordinación de los servicios web, que es similar al modelo de transacción distribuida se describe en la Sección 17.2 en que tiene funciones de coordinador y participantes que son capaces de actuar a cabo protocolos particulares, por ejemplo, para llevar a cabo una transacción distribuida. Este trabajo, que se llama WSCoordination, es descrito por Langworthy [2004]. El mismo grupo también ha demostrado cómo se pueden llevar a cabo transacciones dentro de este modelo. Para un estudio exhaustivo de los protocolos de coordinación de servicios web, véase Alonso *et al.* [2004].

En el resto de esta sección, describimos las ideas detrás de la coreografía de servicios web. Considere el hecho de que sería posible describir todos los posibles caminos alternativos válidos a través del conjunto de interacciones entre pares de los servicios web que trabajan juntos en una tarea conjunta, tales como el escenario agente de viajes. Si una descripción estuviera disponible, que podría ser utilizado como una ayuda para la coordinación de las tareas conjuntas. También podría utilizarse como una especificación a seguir por las nuevas instancias de un servicio, como por ejemplo un nuevo servicio de reservas de vuelos que deseen unirse a una colaboración.

El W3C utiliza el término *coreografía* para referirse a un lenguaje basado en WSDL para definir la coordinación. Por ejemplo, el lenguaje puede especificar restricciones sobre el orden y las condiciones en que los mensajes se intercambian por los participantes. Una coreografía está destinado a proporcionar una descripción global de un conjunto de interacciones, que muestra el comportamiento de cada miembro de un conjunto de participantes, con el fin de mejorar la interoperabilidad.

Requisitos para la coreografía • La coreografía está destinado a apoyar las interacciones entre los servicios web que por lo general son administrados por diferentes empresas y organizaciones. Una colaboración que involucra múltiples servicios web y clientes debe ser descrito en términos de las series de interacciones observables entre pares de ellos. Tal descripción puede ser visto como un contrato entre los participantes, y se podría utilizar para los siguientes fines:

- para generar código esboza un nuevo servicio que quiere participar;
- como base para la generación de mensajes de prueba para un nuevo servicio;
- para promover un entendimiento común de la colaboración;
- para analizar la colaboración, por ejemplo, para identificar posibles situaciones de punto muerto.

El uso de una descripción coreografía común por parte de un conjunto de servicios web colaborando debe dar lugar a servicios más robustos con una mejor interoperabilidad. Además, debería ser más fácil para desarrollar e introducir nuevos servicios, por lo que el servicio en general más útil.

El proyecto de documento de trabajo del W3C [www.w3.org XV] Sugiere que una lengua coreografía debe incluir las siguientes características:

- composición jerárquica y recursiva de coreografías;
- la posibilidad de añadir nuevas instancias de un servicio existente y los nuevos servicios;
- caminos concurrentes, caminos alternativos y la capacidad de repetir una sección de una coreografía;
- tiempos de espera variables - por ejemplo, diferentes periodos para la realización de reservas;
- excepciones, por ejemplo, para hacer frente a los mensajes que llegan fuera de secuencia y de usuario acciones como cancelaciones;
- interacciones asíncronas (devoluciones de llamada);
- referencia de pasada, por ejemplo, para permitir que una empresa de alquiler de coches de consultar a un banco para una verificación de crédito en nombre de un usuario;
- marcado de los límites de las transacciones separadas que tienen lugar, por ejemplo, para permitir la recuperación;
- la capacidad de incluir documentación legible por humanos.

Un modelo basado en estos requisitos se describe en otro proyecto de documento de trabajo del W3C [www.w3.org XVI].

Idiomas para la coreografía • La intención es producir un lenguaje declarativo, basado en XML para la definición de coreografías que pueden hacer uso de las definiciones WSDL. El W3C ha hecho una recomendación de Coreografía de Servicios Web de lenguaje de definición Versión 1 [www.w3.org XVII]. Antes de esto, un grupo de empresas presentado al W3C una especificación para la interfaz de servicios web coreografía [www.w3.org XVIII].

9.7 Las solicitudes de servicios web

Los servicios web son ahora uno de los paradigmas dominantes para la programación de sistemas distribuidos. En esta sección, se discuten una serie de las principales áreas en las que los servicios web han sido empleadas ampliamente: en el apoyo a la arquitectura orientada a servicios, la cuadrícula y, más tarde, la computación en nube.

9.7.1 Arquitectura orientada a servicios

Arquitectura orientada a Servicios (SOA) es un conjunto de principios de diseño mediante el cual los sistemas distribuidos se desarrollan utilizando conjuntos de servicios débilmente acoplados que se pueden descubrir dinámicamente y luego se comunican entre sí o se coordinan a través coreografía para proporcionar servicios mejorados. arquitectura orientada a servicios es un resumen

concepto que puede ser implementado usando una variedad de tecnologías, incluyendo el objeto distribuido o enfoques basados en componentes discutidos en el capítulo 8. Los medios principales de la realización de arquitectura orientada a servicios, sin embargo, es a través del uso de servicios web, en gran parte debido al acoplamiento suelto inherente a este enfoque (como se discute en la Sección 9.2).

Este estilo de arquitectura se puede utilizar dentro de una empresa u organización para ofrecer una arquitectura de software flexible y para lograr la interoperabilidad entre los distintos servicios. Su uso principal, sin embargo se encuentra en la más amplia de Internet, que ofrece una vista común de servicios haciéndolos accesibles a nivel mundial y es susceptible de composición posterior. Esto hace que sea posible superar los niveles de heterogeneidad inherentes a la Internet y también para hacer frente al problema de las diferentes organizaciones de la adopción de diferentes productos de middleware internamente - es posible que una organización utiliza CORBA interna y otra para utilizar .NET, pero tanto entonces para exponer las interfaces utilizando servicios web, fomentando así la interoperabilidad global. La propiedad resultante es conocido como *de empresa a empresa (B2B) de integración.* Ya hemos visto un ejemplo de la necesidad de integración B2B en la figura 9.18 (el escenario agente de viajes), en el que el agente de viajes puede hacer frente a una amplia gama de compañías que ofrecen vuelos, alquiler de coches y alojamiento en un hotel.

Arquitectura orientada a servicios también permite y fomenta una *Triturar* enfoque para el desarrollo de software. Un mashup es un nuevo servicio creado por un desarrollador de terceros mediante la combinación de dos o más servicios disponibles en el entorno distribuido. La cultura mashup se basa en la disponibilidad de servicios útiles con interfaces bien definidas, junto con una comunidad de innovación abierta donde los individuos o grupos se involucran en el desarrollo de servicios combinados experimentales y ponerlos a disposición de otros para un mayor desarrollo. Ambas condiciones se cumplen ahora por el Internet, en particular con la aparición de la computación en nube y software como servicio (como se introdujo en la Sección

7.7.1), donde los principales desarrolladores de software, tales como Amazon, Flickr y eBay hacen los servicios disponibles a través de interfaces publicadas a otros desarrolladores. Como un ejemplo, consulte JBidwatcher [www.jbidwatcher.org], Un mashup basado en Java que se conecta a eBay para administrar las ofertas de forma proactiva en nombre de un cliente, por ejemplo el seguimiento de subastas y concursos en el último minuto para maximizar las posibilidades de éxito.

9.7.2 La cuadrícula

El nombre 'Guía' se utiliza para referirse a middleware que está diseñado para permitir el intercambio de recursos tales como archivos, computadoras, software, datos y sensores en una escala muy grande. Los recursos son compartidos normalmente por grupos de usuarios de diferentes organizaciones que están colaborando en la solución de los problemas que requieren un gran número de ordenadores para resolverlos, ya sea mediante el intercambio de datos o mediante el intercambio de potencia de cálculo. Estos recursos son necesariamente compatibles con el hardware heterogéneo equipo, sistemas operativos, lenguajes de programación y aplicaciones. Es necesaria una gestión para coordinar el uso de los recursos para asegurar que los clientes obtengan lo que necesitan y que los servicios pueden permitirse el lujo de suministrarlo. En algunos casos, se requieren técnicas sofisticadas de seguridad para garantizar que el uso correcto de los recursos se hace en este tipo de entorno.

El telescopio-Mundial: Una aplicación de cuadrícula

Este proyecto tiene que ver con el despliegue de los recursos de datos compartidos por la comunidad astronómica. Se describe en la obra de Szalay y Gray [2004], Szalay y Gray [2001] y gris y Szalay [2002]. datos astronomía consta de archivos de observaciones, cada una de las cuales cubre un período de tiempo determinado, una parte del espectro electromagnético (óptica, de rayos x, radio) y un área particular del cielo. Estas observaciones son hechas por diferentes instrumentos empleados en varios lugares en todo el mundo.

Un estudio de cómo los astrónomos compartir sus datos es útil para derivar las características de una aplicación típica de cuadrícula, porque los astrónomos comparten libremente sus resultados con otros y cuestiones de seguridad se pueden omitir, por lo que esta discusión más simple.

Los astrónomos hacen estudios que necesitan combinar datos sobre los mismos objetos celestes sino que involucran a varios períodos de tiempo diferentes y múltiples partes del espectro. La capacidad de utilizar observaciones independientes de los datos es importante para la investigación. La visualización permite a los astrónomos ver los datos en 2D o en 3D gráficos de dispersión.

Los equipos de recolección del almacén de datos en archivos (inmensas terabytes actualmente), que son administrados localmente por cada equipo que reúne datos. Los instrumentos utilizados en la recopilación de datos están sujetos a la ley de Moore, por lo que la cantidad de datos recogidos crece exponencialmente. Como se reunieron, los datos se analizan mediante un procedimiento de tubería y se almacena como datos derivados para su uso por los astrónomos en todo el mundo. Pero antes de que los datos puedan ser utilizados por otros investigadores, los científicos que trabajan en un campo particular, deben ponerse de acuerdo en una forma común de etiquetar sus datos.

Szalay y Gray [2004] señalan que en el pasado, los datos de la investigación científica se incluyó por los autores en los artículos y publicado en revistas que vivían en las bibliotecas. Pero hoy en día, la cantidad de datos es demasiado grande como para ser incluido en una publicación. Esto se aplica no sólo a la astronomía, sino también a los campos de la física de partículas y la investigación del genoma y la biología. El papel de autor pertenece ahora a las colaboraciones, que tienen 5-10 años para construir sus experimentos antes de producir los datos que se publica en el mundo en los archivos basados en la web. Por lo tanto, los científicos que trabajan en los proyectos se convierten en proveedores de datos y bibliotecarios, así como autores.

Esta función adicional requiere cualquier proyecto que gestiona un archivo de datos para que sea accesible a otros investigadores. Esto implica una sobrecarga considerable, además de la tarea original de análisis de datos. Para que este intercambio sea posible, los datos en bruto requiere metadatos para describir, por ejemplo, el momento de su recogida, la parte del cielo y el instrumento utilizado. Además, los datos derivados de ir acompañada por los metadatos que describen los parámetros de las tuberías a través del cual se procesó.

El cálculo de los datos derivados requiere apoyo computacional pesada. A menudo tiene que volver a calcular como técnicas mejoran. Todo esto es un gasto considerable para el proyecto que posee los datos.

El objetivo del telescopio de la World Wide es unificar archivos astronomía del mundo en una gigantesca base de datos que contiene la literatura astronomía, imágenes, datos en bruto, conjuntos de datos derivados y los datos de simulación.

Requisitos de aplicaciones Grid • El telescopio de la World Wide es típico de una serie de *aplicaciones Grid uso intensivo de datos,* donde:

- los datos se recogen por medio de instrumentos científicos;
- los datos se almacenan en archivos separados en sitios cuyas ubicaciones pueden estar en diferentes lugares del mundo;
- los datos son gestionados por equipos de científicos pertenecientes a organizaciones independientes;
- una cantidad inmensa y creciente (terabytes o petabytes) de datos en bruto se generan a partir de los instrumentos;
- programas de ordenador se utilizan para analizar y hacer resúmenes de los datos en bruto, por ejemplo, clasificar, catalogar y calibrar los datos en bruto que representan objetos celestes.

El Internet hace que todos estos archivos de datos potencialmente disponibles para los científicos de todo el mundo, lo que les permite obtener datos de diferentes instrumentos recogidos en diferentes momentos y en diferentes lugares. Sin embargo, un científico particular usando estos datos para su propia investigación estará interesado en sólo un subconjunto de los objetos en los archivos.

La inmensa cantidad de datos en un archivo hace que sea inviable para transferir a la ubicación del usuario antes de procesar para extraer los objetos de interés, debido a consideraciones tales como el tiempo de transmisión y el espacio de disco local requerida. Por lo tanto, no es apropiado utilizar FTP o acceso a la web en este contexto. El procesamiento de los datos en bruto debe llevarse a cabo en el lugar donde se recoge y se almacena en una base de datos. Entonces, cuando un científico realiza una consulta acerca de los objetos particulares, la información en cada base de datos debería ser analizada y, si es necesario, las visualizaciones produce antes de devolver los resultados a la consulta a distancia.

El hecho de que los datos se procesan en muchos sitios diferentes proporciona un paralelismo incorporado que divide eficazmente la inmensa tarea está llevando a cabo.

A partir de las características anteriores, los siguientes requisitos se derivan: R1: El acceso remoto a los recursos - es decir, a la información requerida en los archivos. R2: Tratamiento de datos en el lugar donde se almacena y gestiona, ya sea cuando está recopilada o en respuesta a una solicitud. Una consulta típica podría resultar en una visualización basada en los datos recogidos para una región del cielo registrado por diferentes instrumentos en diferentes momentos. Esto implicará la selección de una pequeña cantidad de datos de cada archivo de datos masiva.

R3: El gestor de recursos de un archivo de datos debe ser capaz de crear instancias de servicio dinámicamente para hacer frente a la sección particular de los datos requeridos, al igual que en el modelo de objetos distribuidos, donde se crean servidores siempre que sean necesarios para manejar diferentes recursos gestionados por un servicio. R4: Los metadatos para describir:

- características de los datos en un archivo - por ejemplo, para la astronomía, la zona del cielo, la fecha y la hora de recogida y los instrumentos utilizados;
- características de un servicio de gestión de los datos - por ejemplo, su coste, su ubicación geográfica, de su editor o de su carga o espacio disponible.

Los servicios de directorio basados en los metadatos arriba: R5.

R6: Software para gestionar las consultas, transferencias de datos y reserva anticipada de los recursos, teniendo en cuenta que los recursos son generalmente administrados por los proyectos que generan los datos y que el acceso a ellos puede necesitar ser racionada. Los servicios Web pueden hacer frente a los dos primeros requisitos, proporcionando una manera conveniente para que los científicos tengan acceso a las operaciones en los datos en archivos remotos. Esto requerirá que cada aplicación particular proporcionar una descripción de servicio que incluye un conjunto de métodos para acceder a sus datos. El middleware Grid debe hacer frente a los requisitos restantes.

Las rejillas también se utilizan para *aplicaciones Grid computacionalmente intensivas* tales como el procesamiento de las enormes cantidades de datos producidos por el acelerador de partículas de alta energía CMS en el CERN [www.uscms.org], Probar los efectos de las moléculas de fármaco candidato [Taufer *et al.* 2003, Chien 2004] o el apoyo a los juegos multijugador masivos en línea utilizando la capacidad disponible en los ordenadores de racimo [www.butterfly.net]. Donde las aplicaciones computacionalmente intensivas, se despliegan en una cuadrícula, la gestión de recursos se refiere a la asignación de los recursos informáticos y el equilibrio de cargas.

Por último, será necesaria la seguridad para muchas aplicaciones Grid. Por ejemplo, la rejilla está en uso para la investigación médica y para aplicaciones de negocios. Incluso cuando la privacidad de los datos no es un problema, será importante establecer la identidad de las personas que crearon los datos.

• **middleware grid** La Arquitectura de Servicios Open Grid (OGSA) es un estándar para las aplicaciones basadas en la red [Foster, *et al.* 2001, 2002]. Proporciona un marco dentro del cual se pueden cumplir los requisitos anteriores, basado en servicios web. Los recursos son administrados por servicios Grid específicos de la aplicación. Globus continuación, implementa la arquitectura.

El Proyecto Globus comenzó en 1994 con el fin de proporcionar un software que integra y estandariza las funciones requeridas por una familia de aplicaciones científicas. Estas funciones incluyen servicios de directorio, seguridad y gestión de recursos. El primer juego de herramientas Globus apareció en 1997. El OGSA evolucionó a partir de la segunda versión de la caja de herramientas (llamado GT2), que se describe en Foster y Kesselman [2004]. La tercera versión (GT3), que apareció en 2002, se basó en OGSA y por lo tanto construido en servicios web. Fue desarrollado por la Alianza Globus (www.globus.org) y se describe en Sandholm y Gawor [2003]. Desde entonces, dos versiones adicionales han sido puestos en libertad - la última versión se conoce como GT5 y está disponible como software de fuente abierta [www.globus.org]).

Un estudio de caso de OGSA y Globus (hasta GT3) se puede encontrar en el sitio Web Companion [www.cdk5.net/web].

9.7.3 La computación en nube

La computación en nube se introdujo en el capítulo 1 como un conjunto de servicios basados en Internet informáticos suficientes para apoyar las necesidades de la mayoría de los usuarios de aplicaciones, almacenamiento y, lo que les permite prescindir en gran parte o totalmente con almacenamiento local de datos y software de aplicación. La computación en nube también promueve una visión de todo como un servicio, desde la infraestructura física o virtual a través de software, a menudo pagado en una base por-uso en lugar de comprar. El concepto es, por tanto, intrínsecamente ligada a un nuevo modelo de negocio para

la computación en nube proveedores ofrecen una gama de cálculo, datos y otros servicios a los clientes según sea necesario para su uso diario, por ejemplo, ofrecer suficiente capacidad de almacenamiento a través de Internet para actuar como un archivo o servicio de copia de seguridad.

Capítulo 1 también comenta sobre la superposición entre la nube y la rejilla. El desarrollo de la red precedió a la aparición de la computación en nube y fue un factor importante en su aparición. Ellos comparten el mismo objetivo de proporcionar recursos (servicios) que hay en el mayor de Internet. Mientras que la cuadrícula tiende a centrarse en aplicaciones de datos pesadas o computacionalmente costosos de alta gama, la nube de computación es más general, ofreciendo una gama de servicios para los usuarios de ordenadores individuales a través de los usuarios de gama alta. El modelo de negocio asociados con la computación en nube es también una característica distintiva. Por tanto, es justo decir que la cuadrícula es un ejemplo temprano de la computación en nube, pero la computación en nube ha desarrollado significativamente desde entonces.

Con la vista de todo como un servicio, los servicios web ofrecen un camino natural para la aplicación de computación en nube, y de hecho muchos vendedores pasan por este camino. La oferta más notable en este espacio es *Amazon Web Services (* AWS) [aws.amazon.com], Y aquí brevemente por debajo de esta tecnología. Veremos un enfoque alternativo para la computación en nube en el capítulo 21, cuando nos fijamos en la infraestructura de Google y el asociado de Google App Engine, que ambos cuentan con un lighterweight, el enfoque de mayor rendimiento que los servicios web.

Amazon Web Services son un conjunto de servicios en la nube implementadas en la extensa infraestructura física propiedad de Amazon.com. Originalmente desarrollado para fines internos en apoyo de su negocio minorista electrónico, Amazon ahora ofrece muchas de las instalaciones a los usuarios externos, lo que les permite ejecutar servicios independientes de la infraestructura. La implementación de AWS se ocupa de las cuestiones clave de sistemas distribuidos, tales como la disponibilidad de servicio de gestión, escalabilidad y rendimiento, lo que permite a los desarrolladores centrarse en el uso de sus servicios. Los servicios se ponen a disposición el uso de estándares de servicios web descritos anteriormente en este capítulo. Esto tiene la ventaja de que los programadores familiarizados con los servicios web pueden utilizar fácilmente AWS y pueden desarrollar mashups que incorporan servicios web de Amazon en su construcción. Más generalmente, el enfoque permite la interoperabilidad a través de Internet. Amazon también adopta el enfoque REST, como se propone en Fielding [2000] y discutido en la Sección 9.2.

Amazon ofrece una amplia y extensible conjunto de servicios, siendo la más significativa de las cuales se enumeran en la Figura 9.19. Contamos con EC2 con más detalle. EC2 es un servicio de computación elástica, donde el término 'elástico' se refiere a la capacidad de ofrecer la capacidad de computación que es de tamaño variable a las necesidades de los clientes. En lugar de una máquina real, EC2 ofrece al usuario una máquina virtual, llamado *ejemplo,* a su especificación deseada. Por ejemplo, un usuario puede solicitar una instancia de los siguientes tipos:

- un *instancia norma* diseñado para ser adecuado para la mayoría de aplicaciones;
- un *ejemplo de alta memoria* que ofrece la capacidad de memoria adicional, por ejemplo para aplicaciones que implican el almacenamiento en caché;
- un *alta CPU* instancia diseñada para apoyar tareas de cálculo;
- un *instancia de proceso clúster* ofreciendo un conjunto de procesadores virtuales con gran ancho de banda de interconexión para tareas de computación de alto rendimiento.

Figura 9.19 Una selección de Amazon Web Services

servicio web	*Descripción*
Amazon Elastic Compute Cloud (EC2)	basada en la web que ofrece el servicio de acceso a las máquinas virtuales de un rendimiento y capacidad de almacenamiento dada
Amazon Simple Storage Service (S3)	servicio de almacenamiento basado en web para datos no estructurados
Amazon simple DB	servicio de almacenamiento basado en la web para la consulta de datos estructurados
Amazon simple cola de servicio (SQS)	servicio alojado apoyo cola de mensajes (como se discute en el capítulo 6)
Amazon Elastic MapReduce	servicio basado en Web para la computación distribuida usando el modelo MapReduce (introducido en el Capítulo 21)
servicio basado en Web Amazon flexible al servicio de pagos (FPS) de soporte electrónico pagos	

Varios de estos puede ser refinado - por ejemplo, para una instancia estándar es posible solicitar una pequeña, mediana o gran ejemplo que representa diferentes especificaciones en términos de potencia de procesamiento, memoria, almacenamiento en disco y así sucesivamente.

EC2 se construye en la parte superior del hipervisor Xen, que se describe en la Sección 7.7.2. Las instancias se pueden configurar para ejecutar una variedad de sistemas operativos, como Windows Server 2008, Linux o OpenSolaris. También se pueden configurar con una variedad de software. Por ejemplo, es posible solicitar una instalación de Apache HTTP para apoyar alojamiento web.

EC2 apoya el concepto interesante de una dirección IP elástica, que se parece a una dirección IP tradicional, pero está asociada con la cuenta del usuario, no es un caso particular. Esto significa que si una máquina (virtual) falla, la dirección IP puede ser reasignado a otra máquina sin requerir la intervención de un administrador de red.

9.8 Resumen

En este capítulo hemos demostrado que los servicios web han surgido de la necesidad de proporcionar una infraestructura para apoyar el interfuncionamiento entre diferentes organizaciones. Esta infraestructura general, utiliza el protocolo HTTP utilizado para el transporte de mensajes entre clientes y servidores a través de Internet y se basa en el uso de URIs para referirse a los recursos. XML, un formato de texto, se utiliza para la representación de datos y de clasificación.

Dos influencias separadas condujeron a la aparición de servicios web. Una de ellas fue la adición de interfaces de servicio a servidores web con el fin de permitir que los recursos en un sitio para ser visitada por los programas cliente que no sean los navegadores y utilizando un modelo más rico

de interacción. El otro era el deseo de ofrecer algo así como RPC a través de Internet, en base a los protocolos existentes. Los servicios web resultantes proporcionan interfaces con los conjuntos de operaciones que se pueden llamar de forma remota. Al igual que cualquier otra forma de servicio, un servicio web puede ser el cliente de otro servicio web, lo que permite un servicio web para integrar o combinar un conjunto de otros servicios web.

SOAP es el protocolo de comunicación que se utiliza generalmente por los servicios web y sus clientes. Puede ser utilizado para transmitir mensajes de solicitud y sus respuestas entre el cliente y el servidor, ya sea mediante el intercambio asíncrono de documentos o por una forma de protocolo requestreply en base a un par de intercambios de mensajes asíncronos. En ambos casos, el mensaje de petición o respuesta está encerrada en un documento con formato XML llamado un sobre. El sobre SOAP se transmite generalmente a través del protocolo HTTP síncrona, aunque se pueden usar otros medios de transporte.

Los procesadores XML y SOAP están disponibles para todos los lenguajes de programación utilizados y sistemas operativos. Esto permite a los servicios web y sus clientes a ser desplegado en casi cualquier lugar. Esta forma de interfuncionamiento está habilitado de los hechos que los servicios web no están vinculados a ningún lenguaje de programación en particular y no son compatibles con el modelo de objetos distribuidos.

En los servicios de middleware convencionales, definiciones de interfaz proporcionan a los clientes con los detalles de los servicios. Sin embargo, en el caso de los servicios web, se utilizan las descripciones de servicio. Una descripción de servicio especifica el protocolo de comunicación para ser utilizado (por ejemplo, SOAP) y el URI del servicio, así como la descripción de su interfaz. La interfaz se puede describir bien como un conjunto de operaciones o como un conjunto de mensajes que se intercambian entre el cliente y el servidor.

la seguridad XML fue diseñado para proporcionar la protección necesaria de los contenidos de un documento intercambiado por los miembros de un grupo de personas, que tienen diferentes tareas a realizar en ese documento. Diferentes partes del documento estarán disponibles a diferentes personas, algunas de ellas con la posibilidad de añadir o alterar el contenido y otros sólo para leerlo. Para habilitar una completa flexibilidad en su uso en el futuro, las propiedades de seguridad se definen dentro del propio documento. Esto se logra por medio de XML, que es un formato de auto-descripción. elementos XML se utilizan para especificar partes del documento que están cifrados o firmados, así como detalles de los algoritmos utilizados e información para ayudar con la búsqueda de claves.

Los servicios Web se han utilizado para una variedad de propósitos en los sistemas distribuidos. Por ejemplo, los servicios web ofrecen una implementación natural del concepto de la arquitectura orientada a servicios, en los que su articulación flexible permite la interoperabilidad en aplicaciones Internetscale - incluyendo aplicaciones de negocio a negocio (B2B). Su acoplamiento débil inherente también es compatible con la aparición de un enfoque mashup en la construcción de servicios web. Los servicios Web también sustentan la cuadrícula, el apoyo a la colaboración entre los científicos o ingenieros en las organizaciones en diferentes partes del mundo. Su trabajo se basa muy a menudo en el uso de los datos brutos recogidos por los instrumentos en diferentes sitios y luego se procesa localmente. Globus es una implementación de la arquitectura que se ha utilizado en una variedad de aplicaciones de uso intensivo de datos y procesos informáticos. Por último, los servicios web se utilizan mucho en la computación en nube. Por ejemplo AWS de Amazon se basa totalmente en estándares de servicios web, junto con el resto de la filosofía de construcción servicio.

9.1 Comparar el protocolo de petición-respuesta como se describe en la Sección 5.2, con la puesta en práctica de la comunicación cliente-servidor en SOAP. Estado dos razones por las que el uso de mensajes asíncronos de jabón es más apropiado a través de Internet. ¿En qué medida el uso de HTTP SOAP de reducir la diferencia entre los dos enfoques?

página 388

9.2 Comparar la estructura de URLs que los utilizados para los servicios web con el de referencias a objetos remotos como se especifica en la Sección 4.3.4. Estado en cada caso la forma en que se utilizan para ejecutar una petición de cliente.

página 393

9.3 **Ilustrar los contenidos de un jabón *Solicitud* mensaje y correspondiente *Respuesta* mensaje en el *Elección* ejemplo** servicio de Ejercicio 5.11, utilizando la versión pictórica de XML como se muestra en la Figura 9.4 y Figura 9.5.

página 389

9.4 **Esquema de los cinco elementos principales de una descripción del servicio WSDL. En el caso de la** *Elección* de servicio definida en el ejercicio 5.11, indicar el tipo de información para ser utilizados por el ***Solicitud* y *Respuesta* Mensajes - ¿Algo de esta necesidad de ser incluidos en el espacio de nombres? Para el *votar*** operación, dibujar diagramas similares a la figura 9.11 y figura 9.13.

página 402

9.5 **Continuando con el ejemplo de la *Elección* servicio, explicar por qué la parte de la descripción WSDL del servicio definido** en el ejercicio 9.4 se conoce como 'abstracta'. ¿Qué tendría que ser añadido a la descripción del servicio para que sea completamente de concreto?

página 400

9.6 **Definir una interfaz Java para el *Elección* servicio adecuado para su uso como un servicio web. Estado** por qué cree que la interfaz que ha definido es adecuado. Explicar cómo se genera un documento WSDL para el servicio y la forma en que se puso a disposición de los clientes.

página 396

9.7 **Describir el contenido de un proxy de cliente de Java para el *Elección* Servicio. Explicar cómo el** de clasificación apropiado y métodos unmarshalling se pueden obtener para un proxy estática.

página 396

9.8 Explicar el papel de un contenedor de servlets en el despliegue de un servicio web y la ejecución de una petición del cliente.

página 396

9.9 En el ejemplo de Java se ilustra en la Figura 9.8 y la Figura 9.9, el cliente y el servidor están tratando con objetos, aunque los servicios Web no admiten objetos distribuidos. ¿Cómo puede ser el caso? ¿Cuáles son las limitaciones impuestas a las interfaces de servicios web Java?

página 395

9.10 **Delinear el esquema de replicación utilizado en UDDI. Suponiendo que las marcas de tiempo del vector son** utilizado para apoyar este esquema, definir un par de operaciones para el uso de los registros que necesitan intercambiar datos.

página 406

9.11 **Explicar por qué UDDI puede ser descrito como siendo a la vez un servicio de nombres y un directorio** servicio, mencionando los tipos de preguntas que se pueden hacer. La segunda 'D' en el nombre de UDDI se refiere al 'descubrimiento' - es realmente un servicio UDDI descubrimiento?

capítulo 13 y *página 404*

9.12 Delinear la diferencia principal entre TLS y seguridad XML. Explicar por qué es XML especialmente adecuado para el papel que desempeña, en función de estas diferencias.

Capítulo 11 y en la página 406

9.13 Documentos protegidos por la seguridad XML pueden ser firmados o cifrados antes de que nadie puede predecir quienes serán los destinatarios finales. ¿Qué medidas se adoptan para garantizar que estos últimos tienen acceso a los algoritmos utilizados por el anterior? *página 406*

9.14 Explicar la relevancia de XML canónica a las firmas digitales. lo contextual la información se puede incluir en la forma canónica? Dé un ejemplo de una violación de seguridad, donde se omite el contexto de la forma canónica. *página 409*

9.15 Un protocolo de coordinación podría llevarse a cabo con el fin de coordinar las acciones de Web servicios. Esquema de una arquitectura para (i) un centralizado y (ii) un protocolo de coordinación distribuida. En cada caso, describir las interacciones necesarias para establecer una coordinación entre un par de servicios web.

página 411

9.16 Comparar la semántica de llamadas RPC con la semántica de *WS-Reliable Messaging:*

i) afirman las entidades a las que se refiere cada uno.

ii) Comparar los diferentes significados de la semántica disponibles (por ejemplo, *a-menos una vez, a-más-una vez, exactamente una vez).*

Capítulo 5 y en la página 392

10

SISTEMAS peer-to-peer

10.1 Introducción

10.2 Napster y su legado

10.3 Peer-to-peer middleware

10.4 superposiciones de enrutamiento

10.5 estudios de casos de superposición: Pasteles, tapicería

10.6 casos prácticos de aplicación: ardilla, Oceanstore, Ivy

10.7 Resumen

sistemas peer-to-peer representan un paradigma para la construcción de sistemas y aplicaciones distribuidas en el que los recursos de datos y computacionales son aportados por muchos servidores en Internet, todos los cuales participan en la prestación de un servicio uniforme. Su aparición es una consecuencia del rápido crecimiento de Internet, que abarca muchos millones de ordenadores y un número similar de usuarios que requieren acceso a los recursos compartidos.

Un problema clave para los sistemas peer-to-peer es la colocación de objetos de datos a través de muchos anfitriones y la posterior disposición para el acceso a ellos de una manera que equilibra la carga de trabajo y asegura la disponibilidad sin añadir gastos indebidos. Se describen varios sistemas y aplicaciones que están diseñadas para lograr este reciente desarrollo.

sistemas middleware peer-to-peer están surgiendo que tienen la capacidad de compartir recursos informáticos, de almacenamiento y los datos presentes en los ordenadores 'en los bordes de Internet' en la escala global. Explotan existente de nombres, enrutamiento, replicación de datos y técnicas de seguridad en nuevas maneras de construir una capa de reparto de recursos confiable a través de una colección poco fiable y no fiable de los ordenadores y redes.

aplicaciones peer-to-peer se han utilizado para proporcionar el intercambio de archivos, almacenamiento en caché web, distribución de información y otros servicios, explotando los recursos de decenas de miles de máquinas a través de Internet. Ellos están en su más eficaz cuando se utiliza para almacenar grandes colecciones de datos inmutables. Su diseño disminuye su eficacia para aplicaciones que almacenan y actualizan los objetos de datos mutables.

10.1 Introducción

La demanda de servicios de Internet se puede espera que crezca a una escala que está limitado sólo por el tamaño de la población del mundo. El objetivo de los sistemas peer-to-peer es permitir el intercambio de datos y recursos en una escala muy grande, eliminando cualquier necesidad de servidores administrados por separado y su infraestructura asociada.

Las posibilidades de ampliar los servicios populares mediante la adición a la serie de los equipos que alojan ellos es menor cuando todos los hosts deben ser propiedad y gestionados por el proveedor de servicios. Los costes de administración y recuperación de fallos tienden a dominar. El ancho de banda de red que se puede proporcionar a un sitio único servidor a través de enlaces físicos disponibles es también una importante limitación. servicios a nivel de sistema, tales como Sun NFS (Sección 12.3), el sistema de archivos Andrew (Sección 12.4) o servidores de vídeo (Sección 20.6.1) y servicios a nivel de aplicación, tales como Google, Amazon o eBay toda presentar este problema en diversos grados.

sistemas peer-to-peer apuntan a servicios de apoyo distribuidos útiles y las aplicaciones que utilizan los recursos de computación de datos y está disponible en los ordenadores personales y estaciones de trabajo que están presentes en Internet y otras redes en números cada vez mayores. Esto es cada vez más atractiva como la diferencia de rendimiento entre las máquinas de escritorio y servidor se estrecha y banda ancha las conexiones a la red proliferan.

Pero hay otra, objetivo más amplio: un autor [Shirky 2000] ha definido aplicaciones Igualitaria como "aplicaciones que exploten los recursos disponibles en los bordes de la Internet - almacenamiento, ciclos, de contenido, de presencia humana. Cada tipo de distribución de los recursos mencionados en esta definición ya está representado por las aplicaciones distribuidas disponibles para la mayoría de tipos de ordenador personal. El propósito de este capítulo es describir algunas técnicas generales que simplifican la construcción de aplicaciones peer-to-peer y mejorar su escalabilidad, fiabilidad y seguridad.

sistemas cliente-servidor tradicionales gestionan y proporcionan acceso a recursos tales como archivos, páginas web u otros objetos de información ubicados en un equipo servidor único o un pequeño grupo de servidores fuertemente acoplados. Con este tipo de diseños centralizados, se requieren algunas decisiones acerca de la colocación de los recursos o la gestión de los recursos de hardware del servidor, pero la escala del servicio está limitado por la capacidad del hardware del servidor y la conectividad de red. sistemas peer-to-peer proporcionan acceso a los recursos de información ubicados en los ordenadores a través de una red (ya se trate de Internet o una red corporativa). Algoritmos para la colocación y posterior recuperación de objetos de información son un aspecto clave del diseño del sistema. El objetivo es ofrecer un servicio que está totalmente descentralizado y auto-organización,

sistemas peer-to-peer comparten estas características:

- Su diseño asegura que cada usuario aporta recursos al sistema.
- A pesar de que pueden diferir en los recursos que contribuyen, todos los nodos de un sistema peer-to-peer tienen las mismas capacidades y responsabilidades funcionales.
- Su correcto funcionamiento no depende de la existencia de cualquier sistema de administración centralizada.

- Pueden ser diseñados para ofrecer un grado limitado de anonimato a los proveedores y usuarios de los recursos.

- Una cuestión clave para su funcionamiento eficiente es la elección de un algoritmo para la colocación de los datos a través de muchos anfitriones y el posterior acceso a ella de una manera que equilibra la carga de trabajo y asegura la disponibilidad sin añadir gastos indebidos. Los ordenadores y las conexiones de red de propiedad y gestionados por una multitud de diferentes usuarios y organizaciones son recursos necesariamente volátiles; sus propietarios no garantizan para mantenerlos encendidos, conectados y sin fallos. Por lo que la disponibilidad de los procesos y equipos que participan en los sistemas peer-to-peer es impredecible. servicios peer-to-peer, por tanto, no pueden confiar en la garantía de acceso a los recursos individuales, aunque pueden ser diseñados para hacer que la probabilidad de fallo para acceder a una copia de un objeto replicado arbitrariamente pequeña.

Varios servicios basados en Internet temprana, incluyendo DNS (Sección 13.2.3) y Netnews / Usenet [Kantor y Lapsley 1986], adoptaron una arquitectura escalable y de alta disponibilidad de varios servidores. La Xerox vid registro de nombres y el servicio de entrega de correo [Birrell *et al.* 1982, Schroeder *et al.* 1984] proporciona un ejemplo temprano interesante de un, servicio distribuido tolerante a fallos escalable. algoritmo parlamento a tiempo parcial de Lamport de consenso distribuido [Lamport 1989], el Bayou replicado sistema de almacenamiento (véase la Sección

18.4.2) y el algoritmo de interdominio enrutamiento IP sin clase (véase la Sección 3.4.3), son ejemplos de algoritmos distribuidos para la colocación o ubicación de la información y se pueden considerar como antecedentes de sistemas peer-to-peer.

Pero el potencial para el despliegue de servicios peer-to-peer utilizando los recursos en los bordes de Internet surgió sólo cuando un número significativo de usuarios había adquirido siempre activa, conexiones de banda ancha a la red, por lo que sus ordenadores de sobremesa plataformas adecuadas para el uso compartido de recursos . Esto ocurrió por primera vez en los Estados Unidos alrededor
1999. A mediados de 2004 el número mundial de las conexiones a Internet de banda ancha había superado con creces los 100 millones [Internet World Stats 2004].

Tres generaciones de sistemas peer-to-peer y desarrollo de aplicaciones pueden ser identificados. La primera generación fue lanzada por el servicio de intercambio de música Napster [OpenNap 2001], que se describe en la siguiente sección. Una segunda generación de aplicaciones de intercambio de archivos que ofrecen una mayor escalabilidad, el anonimato y la tolerancia a fallos rápidamente seguido incluyendo Freenet [Clarke *et al.* 2000, freenetproject.org], Gnutella, Kazaa [Leibowitz *et al.* 2003] y BitTorrent [Cohen 2003].

Peer-to-peer middleware • La tercera generación se caracteriza por la aparición de capas de middleware para la gestión independiente de la aplicación de recursos distribuidos en una escala global. Varios grupos de investigación han completado el desarrollo, evaluación y perfeccionamiento de las plataformas middleware peer-to-peer y demostrada o ellos desplegados en una amplia gama de servicios de aplicaciones. Los ejemplos mejor conocidos y más completamente desarrolladas incluyen pasteles [Rowstron y Druschel 2001], la tapicería [Zhao *et al.* 2004], CAN [Ratnasamy *et al.* 2001], acorde [Stoica *et al.* 2001] y Kademlia [Maymounkov y Mazieres 2002].

Figura 10.1 distinciones entre IP y de superposición de encaminamiento para aplicaciones peer-to-peer

	IP	*superposición de enrutamiento a nivel de aplicación*
Escala	IPv4 está limitado a 2 32 nodos direccionables. El espacio de nombres IPv6 es mucho más generoso (2 128), pero direcciones en ambas versiones están jerárquicamente estructurados y gran parte del espacio se preasignan de acuerdo con los requisitos administrativos.	sistemas peer-to-peer pueden abordar más objetos. El espacio de nombres GUID es muy grande y plana (> 2 128), permitiendo que sea mucho más totalmente ocupado.
Balanceo de carga	Cargas en los routers están determinadas por la topología de red y los patrones de tráfico asociados.	escenarios de objetos pueden ser asignados al azar y por lo tanto los patrones de tráfico están divorciados de la topología de red.
dinámica de la red (adición / eliminación de objetos / nodos)	tablas de enrutamiento IP se actualizan de forma asíncrona en una base de mejor esfuerzo con constantes de tiempo del orden de 1 hora.	tablas de enrutamiento se pueden actualizar sincrónica o asincrónica con retrasos fracciones-de-un-segundo.
Tolerancia a fallos	La redundancia se diseña en la red IP por sus administradores, asegurando la tolerancia de un solo fallo de conectividad router o red. *norte*- veces es costoso.	Rutas y referencias a objetos pueden ser replicados *norte*-replicación para garantizar la tolerancia *norte* fallos de nodos o conexiones.
La identificación del objetivo	Cada dirección IP se asigna a exactamente una orientar nodo.	Los mensajes pueden ser enviados a la réplica más cercana de un objeto de destino.
La seguridad y el anonimato	Abordar sólo es seguro cuando se confía en todos los nodos. El anonimato de los propietarios de las direcciones no se puede lograr.	La seguridad se puede lograr incluso en entornos con confianza limitada. Un grado limitado de anonimato puede ser proporcionada.

Estas plataformas están diseñadas para colocar los recursos (objetos de datos, archivos) en un conjunto de equipos que están ampliamente distribuidos en todo el Internet y para enrutar mensajes a ellos en nombre de clientes, el alivio de los clientes de cualquier necesidad de tomar decisiones sobre los recursos de la colocación y de mantener información sobre el paradero de los recursos que requieren. A diferencia de los sistemas de segunda generación, que proporcionan garantías de entrega para las solicitudes en un número limitado de saltos de la red. Colocan réplicas de los recursos en los equipos host disponibles de una manera estructurada, teniendo en cuenta su disponibilidad volátil, su confiabilidad y los requisitos para el equilibrio de carga y localidad de almacenamiento y uso de información variable.

Los recursos son identificados por identificadores únicos globales (GUID), por lo general derivados como un hash seguro (descrito en la Sección 11.4.3) de todos o algunos de estado del recurso. El uso de un control seguro hace 'auto certificadora' un recurso - los clientes que reciben un recurso puede comprobar la validez de la matriz. Esto lo protege contra la manipulación por los nodos que no se confía en que se puede ser almacenado, pero esta técnica requiere que los estados de los recursos son inmutables, ya que un cambio en el estado resultaría en un valor hash diferente. Por lo tanto peerto-peer sistemas de almacenamiento están inherentemente más adecuados para el almacenamiento de objetos inmutables

(Como música o archivos de vídeo). Su uso para objetos con valores cambiantes es más difícil, pero esto puede ser acomodado por la adición de servidores de confianza para gestionar una secuencia de versiones e identificar la versión actual (como se hace por ejemplo en Oceanstore y Ivy, que se describe en las Secciones 10.6.2 y 10.6.3).

El uso de sistemas peer-to-peer para aplicaciones que exigen un alto nivel de disponibilidad de los objetos almacenados requiere el diseño de aplicaciones cuidado de evitar situaciones en las que todas las réplicas de un objeto son al mismo tiempo disponible. Hay un riesgo de que esto para los objetos almacenados en los ordenadores con la misma propiedad, ubicación geográfica, la administración, la conectividad de red, país o jurisdicción. El uso de GUID asistencias distribuidos al azar mediante la distribución de las réplicas objeto a encuentra azar nodos de la red subyacente. Si la red subyacente se extiende por muchas organizaciones en todo el mundo, entonces el riesgo de no disponibilidad simultánea se reduce mucho.

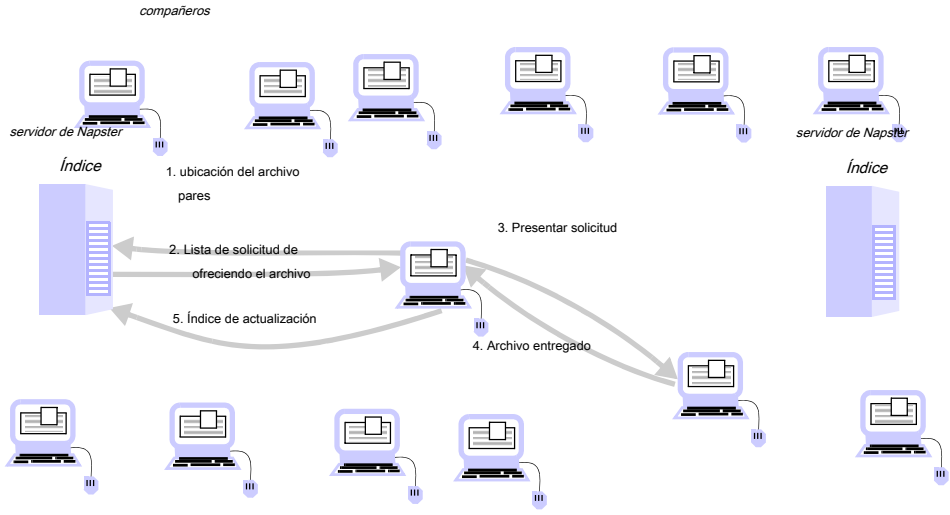
Superposición de enrutamiento frente enrutamiento IP • A primera vista, superposiciones de enrutamiento comparten muchas características con la infraestructura de enrutamiento de paquetes IP que constituye el mecanismo principal de comunicación de Internet (véase la Sección 3.4.3). Por tanto, es legítimo preguntarse por qué se requiere un mecanismo a nivel de aplicación de enrutamiento adicional en los sistemas peer-to-peer. La respuesta está en varias distinciones que se identifican en la figura 10.1. Se puede argumentar que algunas de estas distinciones se derivan de la naturaleza 'legado' de la PI como protocolo principal de Internet, pero el impacto del legado es demasiado fuerte para que pueda ser superada con el fin de apoyar las aplicaciones peer-to-peer de forma más directa.

• computación distribuida La explotación de la potencia de cálculo de repuesto en los ordenadores de los usuarios finales ha sido durante mucho tiempo un tema de interés y el experimento. Trabajar con los primeros ordenadores personales en Xerox PARC [Shoch y Hupp 1982] demostraron la factibilidad de realizar tareas de cálculo intensivo débilmente acoplados mediante la ejecución de procesos de fondo en ~ 100 computadoras personales conectadas por una red local. Más recientemente, un número mucho mayor de ordenadores han sido objeto de un uso para realizar varios cálculos científicos que requieren cantidades casi ilimitadas de potencia de cálculo.

El esfuerzo más ampliamente conocida de este tipo es el *SETI @ home* proyecto [Anderson *et al.* 2002], que es parte de un proyecto más amplio llamado el SETI. SETI @ home particiones un flujo de datos de radiotelescopios digitalizadas en unidades de trabajo 107second, cada una de alrededor de 350 Kbytes y los distribuye a los equipos cliente cuya potencia de cálculo es aportado por los voluntarios. Cada unidad de trabajo se distribuye de forma redundante a 3-4 ordenadores personales para protegerse de los nodos erróneos o maliciosos y se examina para patrones de señales significativas. La distribución de las unidades de trabajo y la coordinación de los resultados es manejado por un único servidor que se encarga de la comunicación con todos los clientes. Anderson *et al. [*2002] informó de que 3,91 millones de ordenadores personales habían participado en el proyecto SETI @ home para agosto de 2002, lo que resulta en el procesamiento de 221 millones de unidades de trabajo y que representa un promedio de 27,36 teraflops de potencia de cálculo durante los 12 meses hasta julio de 2002. El trabajo realizado hasta esa fecha, supone el mayor cálculo sencillo de la historia.

El cálculo de SETI @ home es inusual, ya que no implica ninguna comunicación o coordinación entre las computadoras mientras que están procesando las unidades de trabajo; los resultados se comunican a un servidor central en un solo mensaje corto que puede ser entregado siempre que se disponga el cliente y el servidor. Algunas otras tareas científicas de esta naturaleza han sido identificados, incluyendo la búsqueda de números primos grandes e intentos de fuerza bruta descifrado, pero el desencadenamiento de la potencia de cálculo en el

Figura 10.2 Napster: archivo peer-to-peer para compartir con un índice centralizado, replicado



Internet para una gama más amplia de tareas dependerá del desarrollo de una plataforma distribuida que soporta el intercambio de datos y la coordinación de cómputo entre los sistemas participantes a gran escala. Ese es el objetivo del proyecto de cuadrícula, discutido en el capítulo 19.

En este capítulo nos centramos en los algoritmos y sistemas desarrollados hasta la fecha para el intercambio de datos en redes peer-to-peer. En la Sección 10.2 se resumen diseño de Napster y una revisión de las lecciones aprendidas de la misma. En la sección 10.3 se describen los requisitos generales para capas de middleware peer-to-peer. Las siguientes secciones cubren el diseño y la aplicación de plataformas middleware-peer-to-peer, comenzando con una especificación abstracta en la Sección 10.4, seguido por una descripción detallada de dos ejemplos completamente desarrollados en la sección 10.5 y algunas aplicaciones de los mismos en la Sección 10.6.

10.2 Napster y su legado

La primera aplicación en la que surgió una demanda de un servicio de almacenamiento y recuperación de información a nivel mundial escalable era la descarga de archivos de música digital. Tanto la necesidad y la viabilidad de una solución de punto a punto se demostraron por primera vez por el sistema de intercambio de archivos Napster [OpenNap 2001], que proporciona un medio para que los usuarios compartir archivos. Napster se hizo muy popular para el intercambio de música poco después de su lanzamiento en 1999. En su pico, varios millones de usuarios se registra y miles fueron intercambiando archivos de música al mismo tiempo.

La arquitectura de Napster incluye índices centralizados, pero los usuarios suministra los archivos, los cuales fueron almacenamiento y acceso en sus ordenadores personales. método de funcionamiento de Napster se ilustra por la secuencia de pasos que se muestran en la figura 10.2. Tenga en cuenta que en el paso 5 se espera que los clientes añadir sus propios archivos de música en el conjunto de recursos compartidos por

la transmisión de un enlace con el servicio de indexación de Napster para cada archivo disponible. Por lo tanto la motivación para Napster y la clave de su éxito fue la puesta a disposición de un conjunto grande, ampliamente distribuida de archivos a los usuarios a través de Internet, el cumplimiento de la sentencia de Shirky, proporcionando acceso a recursos compartidos 'en los bordes de Internet'.

Napster fue cerrado como consecuencia de las acciones judiciales ejercitadas contra los operadores del servicio Napster por los propietarios de los derechos de autor de algunos de los materiales (es decir, la música codificada digitalmente) que se puso a disposición en el mismo (ver el cuadro de abajo).

El anonimato de los receptores y los proveedores de datos compartidos y otros recursos es una preocupación para los diseñadores de sistemas peer-to-peer. En sistemas con muchos nodos, el encaminamiento de las peticiones y los resultados se puede hacer suficientemente tortuoso para ocultar su fuente y el contenido de archivos puede ser distribuido a través de múltiples nodos, la difusión de la responsabilidad de su puesta a disposición. Los mecanismos para la comunicación anónima que son resistentes a la mayoría de las formas de análisis de tráfico están disponibles [Goldschlag *et al.* 1999]. Si los archivos también se encriptan antes de ser colocados en los servidores, los propietarios de los servidores pueden negar plausiblemente ningún conocimiento de los contenidos. Sin embargo, estas técnicas de anonimato aumentan el costo de la distribución de los recursos, y el trabajo reciente ha demostrado que el anonimato disponible es débil contra algunos ataques [Wright *et al.* 2002].

El Freenet [Clarke *et al.* 2000] y Freehaven [Dingledine *et al.* 2000] proyectos se centran en proporcionar servicios de archivos en todo el Internet que ofrecen el anonimato de los proveedores y usuarios de los archivos compartidos. Ross Anderson ha propuesto el Servicio Eternidad [Anderson 1996], un servicio de almacenamiento que ofrece garantías a largo plazo de los datos

sistemas peer-to-peer y problemas de propiedad de derechos de autor

Los creadores de Napster argumentaron que no eran responsables de la infracción de los titulares del Copyright derechos porque no estaban participando en el proceso de copia, que se realizó en su totalidad entre los usuarios máquinas. Su argumento fracasó debido a que los servidores de índice se consideraron una parte esencial del proceso. Dado que los servidores de índice se encuentran en direcciones conocidas, sus operadores fueron incapaces de permanecer en el anonimato y por lo tanto podrían ser objeto de demandas.

Un servicio de intercambio de archivos más plenamente distribuido podría haber logrado una mejor separación de las responsabilidades legales, la difusión de la responsabilidad a través de todos los usuarios y por lo tanto haciendo que el ejercicio de los recursos legales muy difícil, si no imposible. Cualquiera que sea vista que uno adopte sobre la legitimidad de la copia de archivos con el propósito de compartir material protegido por copyright, hay justificaciones sociales y políticas legítimas para el anonimato de los clientes y servidores en algunos contextos de aplicación. La justificación más convincente surge cuando se utiliza el anonimato para superar la censura y mantener la libertad de expresión de los individuos en las sociedades u organizaciones opresivos.

Se sabe que el correo electrónico y sitios web han jugado un papel importante en el logro de la conciencia pública en momentos de crisis política en tales sociedades; su papel podría reforzarse si los autores podrían ser protegidos por el anonimato. 'Denuncias' es un caso relacionado: un 'soplón' es un empleado que difunde o informa de las malas acciones de su empleador a las autoridades sin revelar su identidad por temor a sanciones o despido. En algunas circunstancias, es razonable para tal acción a ser protegida por el anonimato.

disponibilidad a través de la resistencia a todo tipo de pérdida accidental de datos y ataques de denegación de servicio. Basa la necesidad de un servicio de este tipo en la observación de que, mientras que la publicación es un estado permanente de información impresa - es prácticamente imposible eliminar material una vez que ha sido publicado y distribuido a unos pocos miles de bibliotecas en diversos organismos y jurisdicciones de todo el mundo - publicaciones electrónicas no pueden alcanzar fácilmente el mismo nivel de resistencia a la censura o supresión. Anderson cubre los requisitos técnicos y económicos para asegurar la integridad de la tienda y también señala que el anonimato es a menudo un requisito esencial para la persistencia de la información, ya que proporciona la mejor defensa contra los desafíos legales, así como las acciones ilegales como sobornos o ataques a los creadores,

Las lecciones aprendidas de Napster • Napster demostró la viabilidad de la construcción de un servicio a gran escala útil que depende casi por completo de datos y computadoras de los usuarios comunes de Internet. Para evitar inundar los recursos informáticos de los usuarios individuales (por ejemplo, el primer usuario para ofrecer una canción de las listas de éxitos) y sus conexiones de red, Napster tuvo en cuenta la localidad de red - el número de saltos entre el cliente y el servidor - la hora de asignar una servidor a un cliente que solicita una canción. Este mecanismo loaddistribution sencilla activar el servicio de escalar para satisfacer las necesidades de un gran número de usuarios.

limitaciones: Napster utiliza un (replicado) índice unificado de todos los archivos de música disponibles. Para la aplicación en cuestión, la necesidad de coherencia entre las réplicas no era fuerte, así que esto no obstaculizar el rendimiento, pero para muchas aplicaciones que constituiría una limitación. A menos que se distribuye la ruta de acceso a los objetos de datos, detección de objetos y abordar son propensos a convertirse en un cuello de botella.

dependencias de la aplicación: Napster se aprovechó de las características especiales de la aplicación para la que fue diseñado de otras maneras:

- Los archivos de música no se actualizan, evitando cualquier necesidad de asegurarse de que todas las réplicas de archivos sigue siendo proporcional después de las actualizaciones.
- No se requieren garantías relativas a la disponibilidad de los archivos individuales - Si un archivo de música no está disponible temporalmente, se puede descargar más tarde. Esto reduce la necesidad de la fiabilidad de los equipos individuales y sus conexiones a Internet.

10.3 Peer-to-peer middleware

Un problema clave en el diseño de las aplicaciones peer-to-peer está proporcionando un mecanismo para que los clientes puedan acceder a los recursos de datos de forma rápida y fiable dondequiera que se encuentran en toda la red. Napster mantiene un índice unificado de archivos disponibles para este propósito, dando las direcciones de red de sus anfitriones. sistemas de almacenamiento de archivos Igualitaria de segunda generación tales como Gnutella y Freenet emplean dividido y distribuido índices, pero los algoritmos utilizados son específicos para cada sistema.

Este problema de localización existía en varios servicios que son anteriores al paradigma peer-to-peer también. Por ejemplo, Sun NFS responde a esta necesidad con la ayuda de una capa de abstracción de sistema de archivos virtual en cada cliente que acepta solicitudes de acceso a los archivos almacenados en

varios servidores en términos de referencias de archivos virtuales (es decir, v-nodos, véase la Sección 12.3). Esta solución se basa en una cantidad sustancial de preconfiguración en cada cliente y la intervención manual cuando los patrones de distribución de archivos o cambios de provisión de servidor. Está claro que no es escalable más allá de un servicio gestionado por una sola organización. AFS (Sección 12.4) tiene propiedades similares.

Peer-to-peer sistemas de middleware están diseñados específicamente para satisfacer la necesidad de la colocación automática y posterior localización de los objetos distribuidos que gestiona los sistemas y aplicaciones peer-to-peer.

Requerimientos funcionales • La función de middleware peer-to-peer es simplificar la construcción de los servicios que se implementan a través de muchos hosts en una red ampliamente distribuida. Para lograr esto se debe permitir a los clientes para localizar y comunicarse con cualquier recurso disponible individuo a un servicio, a pesar de que los recursos están ampliamente distribuidos entre los anfitriones. Otros requisitos importantes incluyen la posibilidad de añadir nuevos recursos y para eliminarlos a voluntad y para agregar hosts para el servicio y eliminarlos. Al igual que otros middleware, middleware peer-to-peer debe ofrecer una interfaz de programación sencillo para los programadores de aplicaciones que es independiente de los tipos de recursos distribuidos que manipula la aplicación.

Requerimientos no funcionales • Para realizar con eficacia, middleware-peer-to-peer también debe abordar los siguientes requisitos no funcionales [cf. Kubiatowicz 2003]:

escalabilidad global: Uno de los objetivos de las aplicaciones peer-to-peer es explotar los recursos de hardware de un gran número de ordenadores conectados a Internet. Por lo tanto, el middleware Igualitaria debe estar diseñado para soportar aplicaciones que tienen acceso a millones de objetos en decenas de miles o cientos de miles de hosts.

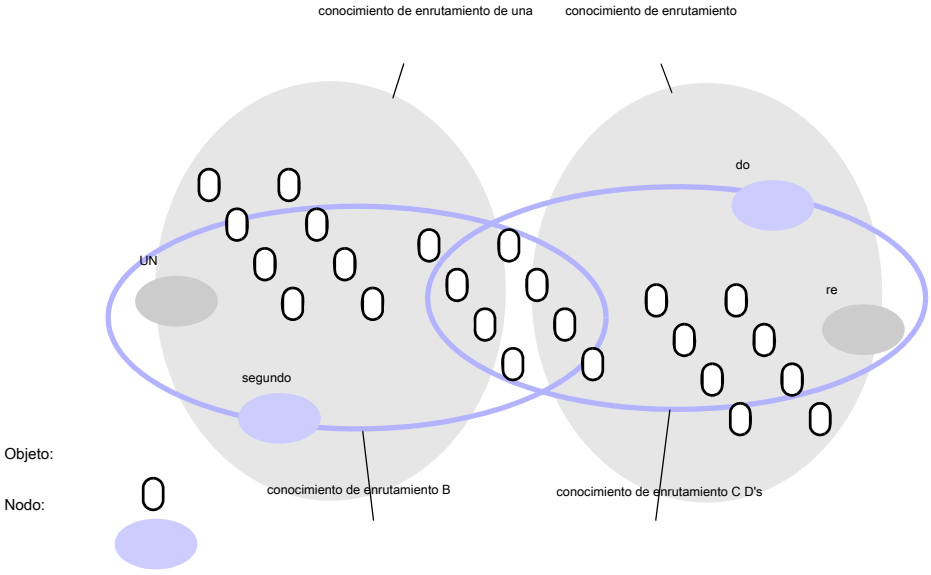
Balanceo de carga: El funcionamiento de cualquier sistema diseñado para aprovechar una gran cantidad de equipos depende de la distribución equilibrada de la carga de trabajo entre ellos. Para los sistemas que estamos considerando, esto se consigue mediante una colocación aleatoria de los recursos junto con el uso de réplicas de los recursos más usados.

La optimización de las interacciones locales entre pares vecinos: La 'distancia de red' entre los nodos que interactúan tiene un impacto sustancial en la latencia de las interacciones individuales, como las solicitudes de cliente para el acceso a los recursos. cargas de tráfico de red también se ven afectados por ella. El middleware debe tener como objetivo colocar recursos cercanos a los nodos que acceden a ellos los más.

Acomodar a la disponibilidad de hospedantes altamente dinámico: La mayoría de los sistemas peer-to-peer se construyen a partir de equipos host que son libres de unirse o dejar el sistema en cualquier momento. Los anfitriones y segmentos de red utilizados en sistemas peer-to-peer no son propiedad o administrados por cualquier autoridad única; ni su fiabilidad ni su participación continua en la prestación de un servicio está garantizada. Un gran reto para los sistemas de peerto-pares es proporcionar un servicio confiable a pesar de estos hechos. Como anfitriones unirse al sistema, que deben integrarse en el sistema y la carga deben ser redistribuidos para explotar sus recursos. Cuando abandonan el sistema sea voluntaria o involuntariamente, el sistema debe detectar su salida y redistribuir la carga y los recursos.

Los estudios sobre las aplicaciones y sistemas como Gnutella y Overnet peer-to-peer han mostrado un importante volumen de negocio de los ejércitos que participan [Saroiu *et al.* 2002, Bhagwan *et al.* 2003]. Para el sistema de intercambio de archivos peer-to-peer Overnet, con 85.000

Figura 10.3 Distribución de información en una superposición de enrutamiento



hosts activos a través de Internet, Bhagwan *et al.* medido una longitud media de sesión de 135 minutos (y una media de 79 minutos) para una muestra aleatoria de 1.468 hosts sobre un período de 7 días, con 260 al 650 de los hosts 1.468 disponibles para el servicio en cualquier momento. (Una sesión representa un período durante el cual un host está disponible antes de que se desconecta voluntariamente o inevitablemente.)

Por otro lado, los investigadores de Microsoft miden una duración de la sesión de 37,7 horas para una muestra aleatoria de 20.000 máquinas conectadas a la red corporativa de Microsoft, con entre 14.700 y 15.600 de las máquinas disponibles para el servicio en un momento dado [Castro *et al.* 2003]. Estas medidas se basan en un estudio de viabilidad para el sistema de archivos peer-to-peer FARSITE [Bolosky *et al.* 2000]. La gran variación entre los valores obtenidos en estos estudios se debe principalmente a las diferencias en el entorno y el comportamiento de la red entre los usuarios individuales de Internet y los usuarios en una red corporativa, tales como los de Microsoft.

Seguridad de los datos en un entorno con confianza heterogénea: En los sistemas a escala mundial con los ejércitos de propiedad diversa que participa, la confianza debe construirse mediante el uso de mecanismos de autenticación y cifrado para garantizar la integridad y privacidad de la información.

El anonimato, la negación y la resistencia a la censura: Hemos observado (en el recuadro de la página 429) que el anonimato de los titulares y los destinatarios de los datos es una preocupación legítima en muchas situaciones de la resistencia a la censura exigentes. Un requisito relacionado es que los anfitriones que contienen datos deben ser capaces de negar plausiblemente responsabilidad de mantener o suministrarlo. El uso de un gran número de hosts en los sistemas de punto a punto puede ser útil en el logro de estas propiedades.

Por lo tanto, la mejor manera de diseñar una capa de middleware para apoyar los sistemas peer-to-peer escala global es un problema difícil. Los requisitos para la escalabilidad y disponibilidad hacen

inviable para mantener una base de datos en todos los nodos cliente que dan las ubicaciones de todos los recursos (objetos) de interés.

El conocimiento de las ubicaciones de los objetos deben ser dividido y distribuido por toda la red. Cada nodo se hace responsable de mantener un conocimiento detallado de las ubicaciones de los nodos y los objetos en una parte del espacio de nombres, así como un conocimiento general de la topología de toda el espacio de nombres (Figura 10.3). Un alto grado de replicación de este conocimiento es necesario para garantizar la fiabilidad de cara a la disponibilidad volátil de los ejércitos y la conectividad de red intermitente. En los sistemas que describimos a continuación, factores de replicación tan altas como 16 se utilizan normalmente.

10.4 superposiciones de enrutamiento

El desarrollo de middleware que cumple con los requisitos funcionales y no funcionales descritos en la sección anterior es un área activa de investigación, y ya han surgido varios sistemas middleware significativos. En este capítulo se describen varios de ellos en detalle.

En los sistemas de punto a punto un algoritmo distribuido conoce como una *superposición de enrutamiento* asume la responsabilidad para la localización de nodos y objetos. El nombre denota el hecho de que el middleware toma la forma de una capa que es responsable de encaminar las peticiones de cualquier cliente a un host que tiene el objeto al que se dirige la solicitud. Los objetos de interés pueden ser colocados en y posteriormente trasladados a cualquier nodo de la red sin la participación del cliente. Se denomina una superposición ya que implementa un mecanismo de encaminamiento en la capa de aplicación que es bastante separada de cualesquiera otros mecanismos de enrutamiento desplegados a nivel de red como de enrutamiento IP. Este enfoque a la gestión y la ubicación de los objetos replicados se analizó primero y demostrado ser eficaz para las redes que implican suficientemente muchos nodos en un documento innovador por Plaxton *et al.* [1997].

La superposición de enrutamiento asegura que cualquier nodo puede acceder a cualquier objeto encaminando cada solicitud a través de una secuencia de nodos, explotando conocimiento en cada uno de ellos para localizar el objeto de destino. sistemas peer-to-peer suelen almacenar varias réplicas de objetos para garantizar la disponibilidad. En ese caso, la superposición de enrutamiento mantiene conocimiento de la ubicación de todas las réplicas disponibles y proporciona peticiones al nodo más cercano 'en vivo' (es decir, uno que no ha fallado) que tiene una copia del objeto en cuestión.

Los GUID utilizados para identificar los nodos y los objetos son un ejemplo de los nombres 'puros' se hace referencia en la Sección 13.1.1. Estos también son conocidos como identificadores opacos, ya que revelan nada acerca de las ubicaciones de los objetos a los que se refieren.

La tarea principal de una capa de enrutamiento es la siguiente:

Encaminamiento de peticiones a objetos: Un cliente que desee invocar una operación en un objeto envía una solicitud incluyendo el GUID del objeto a la superposición de enrutamiento, que encamina la petición a un nodo en el que reside una réplica del objeto.

Sin embargo, la superposición de enrutamiento también debe realizar algunas otras tareas:

La inserción de objetos: Un nodo que desee hacer un nuevo objeto a disposición de un servicio peer-to-peer calcula un GUID para el objeto y lo anuncia a la superposición de enrutamiento, que a su vez garantiza que el objeto es accesible por todos los demás clientes.

Figura 10.4 interfaz de programación básica para una tabla de dispersión distribuida (DHT) como implementado por el API del pasado sobre los pasteles

poner (GUID, datos)

Víveres *datos* en réplicas en todos los nodos responsable del objeto identificado por *GUID. eliminar (GUID)*

Elimina todas las referencias a *GUID* y los datos asociados.

value = conseguir (GUID)

Recupera los datos asociados con *GUID* desde uno de los nodos responsables de la misma.

Eliminación de objetos: Cuando los clientes solicitan la eliminación de objetos del servicio de la superposición de enrutamiento debe hacerlos disponibles.

Además de nodo y eliminación: Los nodos (es decir, ordenadores) pueden unirse y dejar el servicio. Cuando un nodo se une al servicio, la superposición de enrutamiento se encarga de que pueda asumir algunas de las responsabilidades de otros nodos. Cuando las hojas de un nodo (ya sea voluntariamente o como resultado de un fallo del sistema o de la red), sus responsabilidades se distribuyen entre los otros nodos.

GUID de un objeto se calcula a partir de todo o parte del estado del objeto usando una función que proporciona un valor que es, con muy alta probabilidad, único. La unicidad se verifica mediante la búsqueda de otro objeto con el mismo GUID. Una función de hash (como el SHA-1, véase la Sección 11.4) se utiliza para generar el GUID de valor del objeto. Debido a que estos identificadores distribuidos al azar se utilizan para determinar la colocación de objetos y para recuperar de ellos, sistemas de superposición de enrutamiento se describen a veces como *tablas hash distribuidas (* DHT). Esto se refleja en la forma más simple de API utilizado para acceder a ellos, como se muestra en la figura 10.4. Con esta API, la *poner()* operación se utiliza para presentar un elemento de datos para ser almacenados junto con su GUID. La capa de DHT asume la responsabilidad de elegir un lugar para ello, almacenándola (con réplicas para garantizar la disponibilidad) y el acceso a ella a través de la *obtener()* operación.

Una forma ligeramente más flexible de API es proporcionado por una *ubicación de objetos distribuidos y de encaminamiento (* DOLR) capa, como se muestra en la figura 10.5. Con esta interfaz objetos se pueden almacenar en cualquier lugar y la capa DOLR es responsable de mantener un mapeo entre identificadores de objeto (GUID) y las direcciones de los nodos en la que están situados réplicas de los objetos. Los objetos pueden ser replicados y se almacenan con el mismo GUID en diferentes anfitriones, y la superposición de enrutamiento se hace responsable de encaminar peticiones a la réplica más cercana disponible.

Con el modelo de DHT, un elemento de datos con los GUID *X* se almacena en el nodo cuyo GUID es numéricamente más cerca de *X* y en el *r* cuyos anfitriones son los GUID de próxima más cercana a ella numéricamente, donde *r* es un factor de replicación elegido para asegurar una probabilidad muy alta de la disponibilidad. Con el modelo de DOLR, ubicaciones para las réplicas de los objetos de datos se deciden fuera de la capa de enrutamiento y la capa DOLR se notifica de la dirección de host de cada réplica usando el *publicar()* operación.

Figura 10.5 interfaz de programación básica para la localización distribuido objeto y de encaminamiento (DOLR) como implementado por la tapicería

publicar (GUID)

GUID puede ser calculado a partir del objeto (o una parte de ella, por ejemplo, su nombre). Esta función hace que el nodo de la realización de una *publicar* operación del anfitrión para el objeto correspondiente a *GUID. anular la publicación (GUID)*

Hace que el objeto correspondiente a *GUID* inaccesible.

sendToObj (msg, GUID, [n])

Siguiendo el paradigma orientado a objetos, un mensaje de invocación se envía a un objeto con el fin de acceder a él. Esto podría ser una solicitud para abrir una conexión TCP para la transferencia de datos o para devolver un mensaje que contiene la totalidad o parte del estado del objeto. El parámetro final opcional [*norte],* si está presente, solicita la entrega del mismo mensaje a *norte*
réplicas del objeto.

Las interfaces de las figuras 10.4 y 10.5 se basan en un conjunto de representaciones abstractas propuesto por Dabek *et al. [*2003] para demostrar que la mayoría de las implementaciones de superposición de enrutamiento peer-to-peer desarrollado hasta la fecha proporcionan una funcionalidad muy similar.

La investigación sobre el diseño de sistemas de recubrimiento de enrutamiento se inició en 2000 y había llegado a buen término para el año 2005, con el desarrollo y la evaluación de varios prototipos exitosos. Las evaluaciones demostraron que su rendimiento y fiabilidad eran adecuadas para su uso en muchos entornos de producción. En la siguiente sección se describen dos de estos en detalle: Pasteles, que implementa una API de tabla de dispersión distribuida similar a la presentada en la figura 10.4, y la tapicería, que implementa una API similar a la mostrada en la figura 10.5. Tanto los pasteles y la tapicería emplean un mecanismo de encaminamiento conocido como *enrutamiento prefijo* para determinar las rutas para la entrega de mensajes en función de los valores de los GUID a la que se dirigen. enrutamiento prefijo se estrecha la búsqueda del siguiente nodo a lo largo de la ruta mediante la aplicación de una máscara binaria que selecciona un número creciente de dígitos hexadecimales desde el destino GUID después de cada salto. (Esta técnica también se emplea en entre dominios sin clase de encaminamiento para paquetes IP, como se indica en la Sección 3.4.3.)

Otros esquemas de enrutamiento han sido desarrollados que explotan diferentes medidas de distancia para reducir la búsqueda del destino del siguiente salto. Acorde [Stoica *et al.* 2001] basa la elección de la diferencia numérica entre los GUID del nodo seleccionado y el nodo de destino. CAN [Ratnasamy *et al.* 2001] utiliza distancia en una *re-* hiperespacio dimensional en la que se colocan los nodos. Kademlia [Maymounkov y Mazieres 2002] utiliza el XOR de pares de GUID como una métrica para la distancia entre nodos. Debido XOR es simétrica, Kademlia puede mantener tablas de enrutamiento de los participantes de manera muy sencilla; que siempre reciben las peticiones de los mismos nodos contenidos en sus tablas de enrutamiento.

GUID no son legible, por lo que las aplicaciones cliente deben obtener los GUID de los recursos de interés a través de algún tipo de servicio de indexación utilizando nombres legibles por humanos o solicitudes de búsqueda. Idealmente, estos índices también se almacenan de una manera punto a punto de superar las debilidades de los índices centralizados evidenciadas por Napster. Sin embargo, en casos sencillos, tales como archivos de música o publicaciones disponibles para su descarga peer-to-peer, simplemente pueden ser indexados en las páginas web (*cf.* BitTorrent [Cohen 2003]). En una red BitTorrent

búsqueda de artículos conduce a un archivo apéndice que contiene los detalles del recurso deseado, incluyendo su GUID y la dirección URL de un *rastreador* - una serie que mantiene una lista actualizada de direcciones de red para los proveedores que están dispuestos a suministrar el archivo (véase el Capítulo 20 para más detalles del protocolo BitTorrent).

La descripción anterior de las superposiciones de enrutamiento probablemente han planteado dudas en la mente del lector acerca de su rendimiento y fiabilidad. Las respuestas a estas preguntas se desprenden de las descripciones de los sistemas de recubrimiento de enrutamiento prácticos en la siguiente sección.

10.5 estudios de casos de superposición: Pasteles, tapicería

El enfoque de enrutamiento prefijo se adoptó por tanto pastelería y la tapicería. Pastelería es la infraestructura de enrutamiento de mensajes desplegados en varias aplicaciones, incluyendo pasadas [Druschel y Rowstron 2001], un (inmutable) sistema de almacenamiento de archivos de archivos implementado como una tabla de dispersión distribuida con la API en la figura 10.4, y la ardilla, un servicio de almacenamiento en caché web peerto-peer describe en la Sección 10.6.1. Pastelería tiene un diseño sencillo, pero eficaz, que hace que sea un buen primer ejemplo para nosotros estudiar en detalle.

Tapiz es la base para el sistema de almacenamiento Oceanstore, que se describe en la Sección 10.6.2. Tiene una arquitectura más compleja que la de pastelería, ya que tiene como objetivo apoyar una gama más amplia de enfoques localidad. Describimos la tapicería en la Sección 10.5.2.

También tenemos en cuenta los enfoques alternativos no estructurados en la Sección 10.5.3, mirando en detalle en el estilo adoptado por superposición de Gnutella.

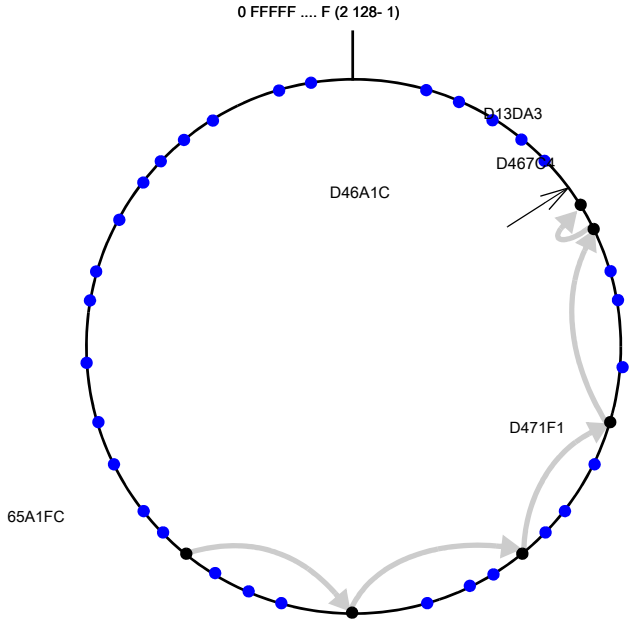
10.5.1 pastelería

Pastelería [Rowstron y Druschel 2001, Castro *et al.* 2002a, freepastry.org] Es una superposición de enrutamiento con las características que se describe en la Sección 10.4. Todos los nodos y los objetos que se puede acceder a través de los pasteles se asignan los GUID de 128 bits. Para nodos, éstos se calculan mediante la aplicación de una función hash seguro (como el SHA-1; véase la Sección
11.4.3) a la clave pública con la que se proporciona cada nodo. Para los objetos tales como archivos, el GUID se calcula mediante la aplicación de una función hash seguro a nombre del objeto o de alguna parte del estado de almacenamiento del objeto. Los GUID resultantes tienen las propiedades usuales de valores de hash seguras - es decir, que se distribuyen aleatoriamente en el rango de 0 a 2^{128} -1. Proporcionan no hay pistas sobre el valor de las que se calcularon, y los enfrentamientos entre GUID para diferentes nodos u objetos son extremadamente poco probable. (Si se produce un choque, pastelería lo detecta y toma medidas correctivas.)

En una red con *norte* nodos participantes, el algoritmo de encaminamiento de pastelería se enrutar correctamente un mensaje dirigido a cualquier GUID en *O (log N)* pasos. Si el GUID identifica un nodo que está actualmente activo, el mensaje se entrega a ese nodo; de lo contrario, el mensaje se entrega al nodo activo cuya GUID es numéricamente más cercano a él. nodos activos asumen la responsabilidad de procesar las solicitudes dirigidas a todos los objetos en su vecindad numérica.

pasos de enrutamiento implican el uso de un protocolo de transporte subyacente (normalmente UDP) para transferir el mensaje a un nodo de repostería que es 'más cerca' a su destino. Pero tenga en cuenta que la cercanía hace referencia aquí es en un espacio completamente artificial - el espacio de GUID. los

Figura 10.6 enrutamiento Circular solo es correcta pero ineficiente *Basado en Rowstron y Druschel [2001]*



Los puntos representan los nodos en vivo. El espacio se considera como circular: el nodo 0 es adyacente al nodo (2 128- 1). El diagrama ilustra el encaminamiento de un mensaje de 65A1FC nodo a D46A1C utilizar la hoja de información del conjunto de solo, asumiendo conjuntos de hojas de tamaño 8 (*l* = 4). Este es un tipo degenerado de enrutamiento que escalar muy mal; no se utiliza en la práctica.

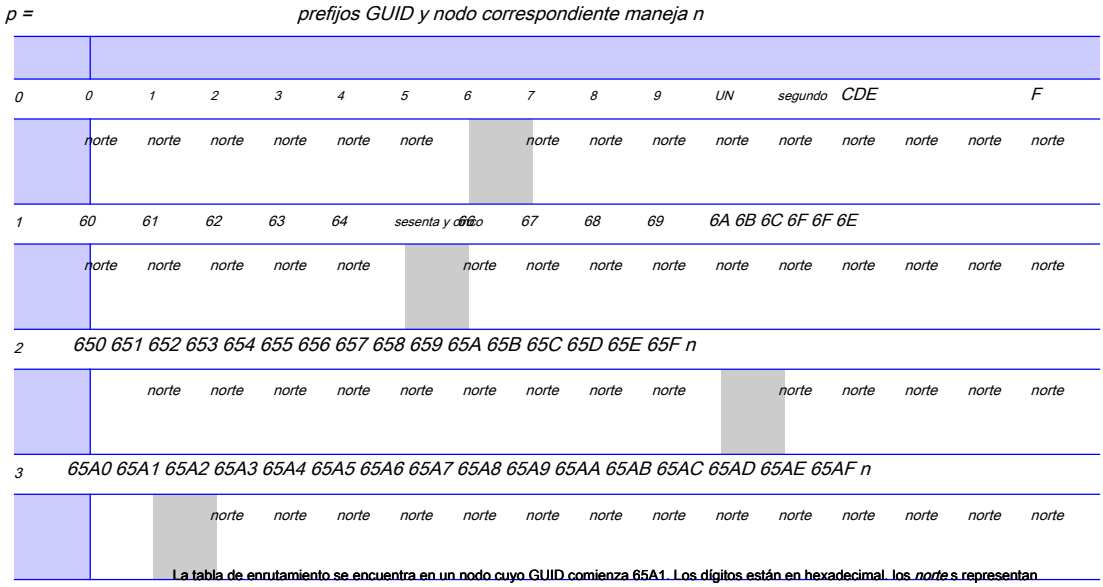
el transporte real de un mensaje a través de Internet entre dos nodos de pastelería puede requerir un número sustancial de IP lúpulo. Para minimizar el riesgo de vías de transporte innecesariamente prolongados, pastelería utiliza una métrica localidad en función de la distancia de red en la red subyacente (tal como la cantidad de saltos o mediciones de latencia de ida y vuelta) para seleccionar vecinos apropiadas cuando la creación de las tablas de enrutamiento utilizados en cada nodo .

Miles de servidores ubicados en sitios muy dispersas pueden participar en una superposición de pasteles. Es totalmente auto-organización: cuando los nuevos nodos se unen a la superposición obtienen los datos necesarios para construir una tabla de enrutamiento y otro estado requerida de los miembros existentes en

O (log N) mensajes, en donde *norte* es el número de los ejércitos que participan en la superposición. En el caso de un fallo de nodo o de salida, los nodos restantes pueden detectar su ausencia y cooperativamente reconfigurar para reflejar los cambios requeridos en la estructura de enrutamiento en un número similar de mensajes.

• Algoritmo de enrutamiento El algoritmo de encaminamiento completo implica el uso de una tabla de encaminamiento en cada nodo para enrutar mensajes de manera eficiente, pero para los propósitos de explicación, se describe el algoritmo de enrutamiento en dos etapas. La primera etapa se describe una forma simplificada del algoritmo que enruta los mensajes correctamente, pero de manera ineficiente sin una tabla de enrutamiento, y

Figura 10.7 Las primeras cuatro filas de una tabla de enrutamiento de pastelería



La tabla de enrutamiento se encuentra en un nodo cuyo GUID comienza 65A1. Los dígitos están en hexadecimal. los *norte* s representan [GUID, dirección IP] pares que actúan como nodo asas especificando el siguiente salto para ser tomada por los mensajes dirigidos a los GUID que responden a cada prefijo dado. entradas sombreadas de color gris en el cuerpo de la tabla indican que el prefijo coincide con el GUID actual hasta el valor determinado de *pag:* la siguiente fila hacia abajo o el conjunto de hojas deben ser examinados para encontrar una ruta. Aunque hay un máximo de 128 filas de la tabla, registrar sólo 16 *norte* filas se rellenarán en promedio en una red con *norte* nodos activos.

la segunda etapa se describe el algoritmo de enrutamiento completo, que rutas una solicitud a cualquier nodo en *O (log N)* mensajes:

Etapa I: Cada nodo activo almacena una *de hoja de conjunto* - un vector *L (* de tamaño 2 *I)* que contiene los GUID y direcciones IP de los nodos cuyos GUID son numéricamente más cercano a ambos lados de su propio (*I* arriba y *I* abajo). conjuntos de hojas son mantenidas por los pasteles como nodos se unen y se van. Incluso después de un fallo de nodo, que serán corregidos dentro de un corto período de tiempo. (Recuperación de fallos se discute a continuación.) Por tanto, es un invariante del sistema de pastelería que los conjuntos de hojas reflejan un estado reciente del sistema y que convergen en el estado actual en la cara de fallos hasta algunos tasa máxima de fracaso.

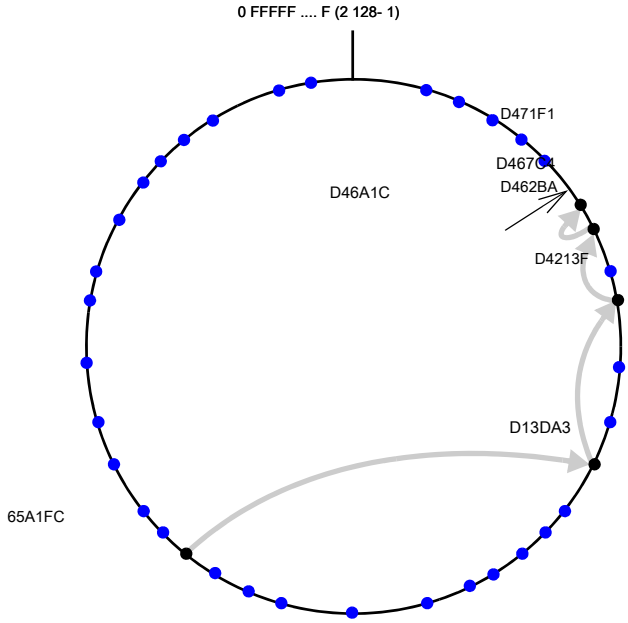
El espacio GUID es tratado como circular: vecino menor de GUID es 0 2 128 -1.

Figura 10.6 da una vista de nodos activos distribuidos en este espacio de direcciones circular. Puesto que cada hoja de conjunto incluye los GUID y direcciones IP de los vecinos inmediatos del nodo actual, un sistema de pastelería con conjuntos de hojas correctas de tamaño de al menos 2 enrutar los mensajes CAN a cualquier GUID trivialmente de la siguiente manera: cualquier nodo *UN* que recibe un mensaje

METRO con la dirección de destino *re* las rutas de mensajes mediante la comparación de *re* con su propio GUID

UN y con cada uno de los GUID en su hoja fija y expedición *METRO* al nodo entre ellos que es numéricamente más cerca de *RE.*

Figura ejemplo de enrutamiento 10,8 pasteles *Basado en Rowstron y Druschel [2001]*



Distribución de un mensaje de 65A1FC nodo a D46A1C. Con la ayuda de una tabla de enrutamiento bien poblado el mensaje se puede entregar en ~ log 16 (*N)* lúpulo.

La figura 10.6 ilustra esto para un sistema de pastelería con *l* = 4. (En bienes típica instalaciones de pastelería, *l* = 8.) En base a la definición de conjuntos de hojas se puede concluir que en cada paso *METRO* se reenvía a un nodo que está más cerca de *re* que el nodo actual y que este proceso finalmente entregará *METRO* al nodo activo más cerca de *RE*. Pero tal esquema de enrutamiento es claramente muy ineficientes, requiriendo ~ *N / 2l* lúpulo para entregar un mensaje en una red con *norte* linfáticos.

Etapa II: La segunda parte de nuestra explicación describe el algoritmo de pastelería completa y muestra cómo enrutamiento eficaz se consigue con la ayuda de las tablas de enrutamiento.

Cada nodo de pastelería mantiene una tabla de enrutamiento con estructura de árbol que da GUID y direcciones IP para un conjunto de nodos de difundir a través de todo el rango de 2 128 posibles valores GUID, con aumento de la densidad de la cobertura de los GUID numéricamente cerca de su propia.

La figura 10.7 muestra la estructura de la tabla de enrutamiento para un nodo específico, y la figura 10.8 ilustra las acciones del algoritmo de encaminamiento. La tabla de enrutamiento se estructura de la siguiente manera: GUID son vistos como valores hexadecimales y la tabla clasifica los GUID de la base de sus prefijos hexadecimales. La tabla tiene tantas filas como dígitos hexadecimales en un GUID, por lo que para el sistema de pastelería prototipo que estamos describiendo, hay 128/4 = 32 filas. cualquier fila *norte* contiene 15 entradas - una para cada valor posible de la *norte*º dígito hexadecimal, excluyendo el valor en el mercado local

Figura 10.9 Algoritmo de enrutamiento de pastelería

Para manejar un mensaje *METRO* dirigida a un nodo *D (* dónde *R [p, i]* es el elemento en la columna *yo,* fila *pag* de la tabla de enrutamiento):

1. *Si (L- l < D <L l) {//* el destino está dentro del conjunto de hojas o es el nodo actual.
2. Adelante *METRO* al elemento L_{yo} de la hoja de conjunto con GUID más cerca de *re* o el nodo actual *A.*

3.} *else {//* utilizar la tabla de enrutamiento para despachar *METRO* a un nodo con un GUID más cerca
4. Encontrar *pag,* la longitud del prefijo común más larga de *re* y *UN,.* y *yo,* el (*p* + 1) º dígito hexadecimal de *RE.*
5. *Si (R [p, i] null)* adelante *METRO* a *R [p, i] //* ruta *METRO* a un nodo con un prefijo más largo común.
6. else {// no hay ninguna entrada en la tabla de enrutamiento.
7. Adelante *METRO* a cualquier nodo en *L* o *R* con un prefijo común de longitud *pag* pero un GUID que es numéricamente más cerca. }}

GUID del nodo. Cada entrada en los puntos de la tabla para uno de los potencialmente muchos nodos cuyos GUID tener el prefijo correspondiente.

El proceso de enrutamiento en cualquier nodo *UN* utiliza la información en su tabla de enrutamiento *R* y el conjunto de hoja *L* para manejar cada solicitud desde una aplicación y cada mensaje entrante desde otro nodo de acuerdo con el algoritmo mostrado en la figura 10.9. Podemos estar seguros de que el algoritmo tendrá éxito en la entrega *METRO* a su destino porque las líneas 1, 2 y 7 realizan las acciones descritas en la Etapa I de nuestra descripción anterior, y hemos demostrado que esto es un completo, aunque ineficiente, algoritmo de encaminamiento. Los pasos restantes están diseñados para utilizar la tabla de enrutamiento para mejorar el rendimiento del algoritmo mediante la reducción del número de saltos necesarios.

Líneas 4-5 entran en juego cada vez *re* no está comprendido en el rango numérico de la hoja de conjunto del nodo actual y entradas de la tabla de enrutamiento pertinentes están disponibles. La selección de un destino para el siguiente salto implica comparar los dígitos hexadecimales de *re* con los de *UN (* el GUID del nodo actual) de izquierda a derecha para descubrir la longitud, *pag,* más largo de su prefijo común. Esta longitud se utiliza entonces como una fila de desplazamiento, junto con el primer dígito no coincidente de *re* como una columna compensado, para acceder al elemento necesario de la tabla de enrutamiento. La construcción de la mesa asegura que este elemento (si no está vacío) contiene la dirección IP de un nodo cuyo GUID tiene *p* + 1 dígitos de prefijo en común con *RE.*

La línea 7 se utiliza cuando *re* cae fuera del rango numérico de la hoja de conjunto y no hay una entrada de tabla de encaminamiento pertinente. Este caso es raro; que surge sólo cuando los nodos recientemente han fracasado y la mesa aún no se ha actualizado. El algoritmo de enrutamiento es capaz de proceder mediante el escaneo de tanto el conjunto de la hoja y la tabla de encaminamiento y de reenvío *METRO* a otro nodo cuyo GUID tiene *pag* búsqueda de dígitos del prefijo pero es numéricamente más cerca de *RE.* Si ese nodo está en *L,* entonces estamos siguiendo el procedimiento de la Etapa I se ilustra en la figura 10.6. Si es en *R,* entonces tiene que estar más cerca de *re* que cualquier nodo en *L,* por lo tanto, estamos mejorando en la Etapa I.

• **Host Integration** Los nuevos nodos utilizan un protocolo de unirse con el fin de adquirir sus contenidos de la tabla y el conjunto de la hoja de ruta y notificar a otros nodos de los cambios que se deben hacer a su

mesas. En primer lugar, el nuevo nodo calcula un GUID adecuado (normalmente aplicando la función hash SHA-1 a la clave pública del nodo), entonces se pone en contacto con un nodo de pastelería cercana. (Aquí se utiliza el término *cerca* para referirse a distancia de red, es decir, un pequeño número de saltos de red o bajo retardo de transmisión; ver el cuadro de abajo).

Supongamos que el GUID del nuevo nodo es *X* y los que entre en contacto nodo cerca tiene GUID *A*. Nodo *X* envía una nota especial *unirse* mensaje de solicitud de *UN,* dando *X* como su destino. *UN* despacha el *unirse* mensaje a través de los pasteles de la forma habitual. ruta pastelería voluntad del *unirse* mensaje al nodo existente cuyo GUID es numéricamente más cerca de *X;* vamos a llamar a este nodo de destino *Z.*

A, Z y todos los nodos (*SEGUNDO, DO,...)* a través del cual el *unirse* mensaje se enruta en su camino hacia *Z* añadir pasos adicionales al algoritmo de pastelería encaminamiento normal, que resultan en la transmisión de los contenidos de las partes pertinentes de sus tablas de enrutamiento y conjuntos de hojas para *X*. *X* las examina y construye su propia tabla de enrutamiento y la hoja de conjunto de ellos, solicitando información adicional de otros nodos si es necesario.

Para ver como *X* construye su tabla de enrutamiento, tenga en cuenta que la primera fila de la tabla depende del valor de *X* Es por GUID, y para minimizar las distancias de enrutamiento, la mesa debe ser construido para enrutar mensajes a través de los nodos vecinos siempre que sea posible. *UN* es un vecino de *X,* por lo que la primera fila de *UN* 'S de mesa es una buena elección inicial para la primera fila de *X* 'S de mesa, *X 0.* Por otra parte, *UN* 'S tabla no es probablemente relevante para la segunda fila, *X 1,* porque *X* 'S y *UN* 'S GUID no pueden compartir el mismo primer dígito hexadecimal. El algoritmo de encaminamiento asegura que *X* 'S y *segundo* 's GUID comparten el mismo primer dígito, sin embargo, lo que implica que la segunda fila de *segundo* 'S tabla de enrutamiento, *segundo 1,* es un valor inicial adecuado para *X 1.* Similar, *do 2* es adecuado para *X 2,* y así.

Además, recordando las propiedades de los conjuntos de hojas, tenga en cuenta que, desde *Z* Es por GUID es numéricamente más cerca de *X* 'S, *X* 'S de hoja de conjunto debe ser similar a *Z* 'S. De hecho, *X* 'S de hoja de conjunto ideal deberá diferir de *Z* 'S por un solo miembro. *Z* por lo tanto, 's de hoja de conjunto se toma como una primera aproximación adecuada, que con el tiempo se optimiza a través de la interacción con sus vecinos como se describe en la subsección tolerancia a fallos a continuación.

Por último, una vez *X* ha construido su conjunto de la hoja y la tabla de enrutamiento de la manera indicada anteriormente, se envía su contenido a todos los nodos identificados en el conjunto de la hoja y la tabla de enrutamiento y ajustar sus propios cuadros para incorporar el nuevo nodo. Toda la tarea de incorporar un nuevo nodo en la infraestructura de pastelería requiere la transmisión de O (log *NORTE)* mensajes.

el fracaso o la salida Anfitrión • Los nodos de la infraestructura de repostería pueden fallar o salir sin previo aviso. Se considera que un nodo de pastelería fallado cuando sus vecinos inmediatos (en el espacio) GUID ya no pueden comunicarse con él. Cuando esto ocurre, es necesario reparar los conjuntos de hojas que contienen el GUID del nodo que ha fallado.

algoritmo del vecino más cercano

El nuevo nodo debe tener la dirección del nodo de pastelería al menos uno ya existente, pero podría no estar cerca. Para asegurar que linfáticos cercanos son conocidos pastelería incluye un algoritmo de 'vecino más cercano' para encontrar un nodo cercano al medir de forma recursiva el retardo de ida y vuelta para un mensaje de sondeo enviado periódicamente a cada miembro del conjunto de la hoja del nodo de pastelería cercana actualmente conocido.

Para reparar su hoja de conjunto *L*, el nodo que descubre el fracaso busca un nodo vivo cerca del nodo que ha fallado en *L* y pide una copia del conjunto de la hoja de ese nodo, *L'*. *L* 'Contendrá una secuencia de GUID que solapan parcialmente los de *L*, incluyendo uno con un valor apropiado para reemplazar el nodo que ha fallado. Otros nodos vecinos son entonces informados de la falla y que realizan un procedimiento similar. Este procedimiento garantiza que la reparación de conjuntos de hojas serán reparados a menos *l* nodos numerados de forma adyacente fallan simultáneamente.

Las reparaciones en las tablas de enrutamiento se hacen sobre una base 'cuando descubierto'. El enrutamiento de mensajes puede proceder con algunas entradas de la tabla de enrutamiento que ya no están en vivo - fracasado los intentos de enrutamiento dan lugar a la utilización de una entrada diferente de la misma fila de una tabla de enrutamiento.

• localidad La estructura de enrutamiento de repostería es altamente redundante: hay muchas rutas entre cada par de nodos. La construcción de las tablas de enrutamiento pretende tomar ventaja de esta redundancia para reducir los tiempos de transmisión de mensajes reales mediante la explotación de las propiedades localidad de nodos en la red de transporte subyacente (que normalmente es un subconjunto de nodos en el Internet).

Recordemos que cada fila de una tabla de enrutamiento contiene 16 entradas. Las entradas de la *yo*°
fila dan las direcciones de 16 nodos con GUID con *yo* -1 dígitos iniciales hexadecimales que coincidan con el GUID del nodo actual y una *yo*° dígitos que toma cada uno de los posibles valores hexadecimales. Una superposición de pastelería bien poblado contendrá muchos más nodos que pueden estar contenidos en una tabla de enrutamiento individual; siempre que una nueva tabla de enrutamiento se está construyendo una elección se hace para cada posición entre varios candidatos (tomado de información de encaminamiento suministrada por otros nodos) en base a un algoritmo de selección vecino proximidad [Gummadi *et al.* 2003]. Una localidad métrica (número de IP lúpulo o latencia medido) se utiliza para comparar candidatos y se elige el nodo más cercano disponible. Dado que la información disponible no es exhaustiva, este mecanismo no puede producir rutas óptimas a nivel mundial, pero las simulaciones han demostrado que resulta en rutas que son, en promedio, sólo alrededor del 30-50% más largo que el óptimo.

Tolerancia a fallos • Como se describió anteriormente, el algoritmo de encaminamiento pasteles asume que todas las entradas en las tablas y los conjuntos de hoja de encaminamiento se refieren a vivir, funcionando correctamente nodos. Todos los nodos envían ' *Los mensajes de transacciones'(* es decir, los mensajes enviados a intervalos de tiempo fijos para indicar que el emisor está vivo) a nodos vecinos en sus conjuntos de hojas, pero la información acerca de los nodos fallidos detectados de esta manera no puede difundirse con rapidez suficiente para eliminar los errores de enrutamiento. Tampoco se representan los nodos maliciosos que pueden intentar interferir con el enrutamiento correcto. Para superar estos problemas, se espera que los clientes que dependen de la entrega de mensajes fiable emplear una *al menos una vez* mecanismo de entrega (Sección ver

5.3.1) y repetir sus peticiones de varias veces en la ausencia de una respuesta. Esto permitirá que los pasteles una ventana de tiempo más largo para detectar fallos en los nodos y reparación.

Para hacer frente a los fallos restantes o nodos maliciosos, un pequeño grado de aleatoriedad se introduce en el algoritmo de selección de ruta descrita en la figura 10.9. Esencialmente, el paso en la línea 5 de la Figura 10.9 se modifica en una pequeña proporción seleccionada al azar de los casos para producir un prefijo común que es menor que la longitud máxima. Esto se traduce en el uso de un enrutamiento tomado de una fila anterior de la tabla de enrutamiento, produciendo menos óptimo, pero diferente de enrutamiento que la versión estándar del algoritmo. Con esta variación aleatoria en el algoritmo de encaminamiento, retransmisiones cliente deben finalmente tener éxito incluso en presencia de un pequeño número de nodos maliciosos.

• **fiabilidad** Los autores de pastelería han desarrollado una versión actualizada denominada MSPastry [Castro *et al.* 2003] que utiliza el mismo algoritmo de encaminamiento y los métodos de gestión de acogida similares, pero también incluye algunas medidas de seguridad de funcionamiento adicionales y algunas optimizaciones de rendimiento de los algoritmos de gestión de acogida.

Fiabilidad medidas incluyen el uso de reconocimientos en cada salto en el algoritmo de encaminamiento. Si el host emisor no recibe un acuse de recibo después de un tiempo de espera especificado, selecciona una ruta alternativa y retransmite el mensaje. El nodo que no pudo enviar un acuse de recibo se observa a continuación, como un fallo sospechado.

Como se mencionó anteriormente, para detectar los nodos fallidos cada nodo de pastelería envía periódicamente un mensaje de latido a su vecino inmediato a la izquierda (es decir, con un GUID inferior) en el conjunto de hoja. Cada nodo también registra la hora del último mensaje recibido de los latidos del corazón de su vecino inmediato a la derecha (con un GUID superior). Si el intervalo desde el último latido del corazón excede un umbral de tiempo de espera, el nodo de detección inicia un procedimiento de reparación que implica poner en contacto los nodos restantes de la hoja de conjunto con una notificación sobre el nodo que ha fallado y una solicitud de sustituciones sugeridas. Incluso en el caso de múltiples fallos simultáneos, este procedimiento termina con todos los nodos en el lado izquierdo del nodo que tiene conjuntos de hojas fallidos que contienen la *l* nodos en vivo con los GUID más cercanos.

Hemos visto que el algoritmo de encaminamiento puede funcionar correctamente con los conjuntos de hojas solamente; pero el mantenimiento de las tablas de enrutamiento es importante para el rendimiento. nodos fallidos sospechosos en tablas de enrutamiento se sondean de una manera similar a la utilizada para el conjunto de la hoja y si no logran responder, sus entradas de la tabla de enrutamiento se reemplazan con una alternativa adecuada obtenida a partir de un nodo cercano. Además, un protocolo de chismes sencilla (véase la Sección 18.4.1) se utiliza para intercambiar periódicamente información de la tabla de encaminamiento entre nodos con el fin de reparar entradas erróneas y prevenir el deterioro lento de las propiedades localidad. El protocolo chismes se ejecuta cada 20 minutos.

El trabajo de evaluación • Castro y sus colegas han llevado a cabo una evaluación exhaustiva del rendimiento MSPastry, el objetivo de determinar el impacto en el rendimiento y la fiabilidad del huésped unirse a *l* tasa de licencia y los mecanismos de confiabilidad asociados [Castro *et al.* 2003].

La evaluación se llevó a cabo mediante la ejecución del sistema MSPastry bajo el control de un simulador se ejecuta en una sola máquina que simula una gran red de ordenadores, con el mensaje que pasaban sustituido por retardos de transmisión simuladas. La simulación modela de forma realista el comportamiento de inclusión / exclusión de los ejércitos y los retardos de transmisión IP basado en parámetros de instalaciones reales.

Todos los mecanismos de fiabilidad de la MSPastry se incluyeron, con intervalos realistas para la sonda y mensajes de latido. El trabajo de simulación fue validado por comparación con las mediciones tomadas con MSPastry la ejecución de un carga de la aplicación real a través de una red interna con 52 nodos.

A continuación se resumen los resultados clave.

Confianza: Con una tasa de pérdida de mensajes asumida IP del 0%, MSPastry no pudo entregar 1,5 en 100.000 solicitudes (presumiblemente debido a la no disponibilidad de hosts de destino), y todas las solicitudes que se entregaron llegó al nodo correcto.

Con una tasa de pérdida de mensajes asumida IP del 5%, MSPastry perdió alrededor de 3,3 en 100.000 solicitudes y 1,6 de cada 100.000 solicitudes fueron entregados al nodo mal. El uso de reconocimientos por salto en MSPastry asegura que todos los mensajes perdidos o mal dirigidos eventualmente son retransmitidos y llegan al nodo correcto.

Actuación: La métrica utilizada para evaluar el desempeño de MSPastry se llama *castigo por demora relativa (* RDP) [Chu *et al.* 2000], o *tramo.* RDP es una medida directa de la coste adicional incurrido en el empleo de una capa de superposición de enrutamiento. Es la relación entre el retraso medio en la entrega de una petición de la superposición de encaminamiento y en la entrega de un mensaje similar entre los mismos dos nodos a través de UDP / IP. Los valores observados para RDP MSPastry bajo cargas simuladas variaron de ~ 1,8 con cero pérdida de mensaje de red a ~ 2,2 con 5% de pérdida de mensaje de red.

Gastos generales: La carga de red adicional generada por *control de tráfico* - los mensajes implicados en el mantenimiento de conjuntos de hojas y las tablas de enrutamiento - era menos de 2 mensajes por minuto por nodo. El tráfico RDP y el control se aumentó significativamente tanto para las longitudes de sesión de menos de unos 60 minutos, debido a los gastos generales de configuración inicial. En general, estos resultados muestran que la superposición de capas de trazado se pueden construir que lograr un buen rendimiento y alta fiabilidad con miles de nodos que operan en entornos realistas. Incluso con longitudes medias más cortas sesiones de 60 minutos y las altas tasas de error de red del sistema se degrada con gracia, sin dejar de ofrecer un servicio eficaz.

La optimización de latencia de búsqueda de superposición • Zhang *et al. [* 2005a] han demostrado que el rendimiento de la búsqueda de una clase importante de redes superpuestas (incluyendo pasteles, acordes y la tapicería) puede ser mejorada sustancialmente mediante la inclusión de un simple algoritmo de aprendizaje que mide las latencias en realidad experimentado para acceder a los nodos de superposición y de este modo modifica de forma incremental la superposición de las tablas de enrutamiento para optimizar las latencias de acceso.

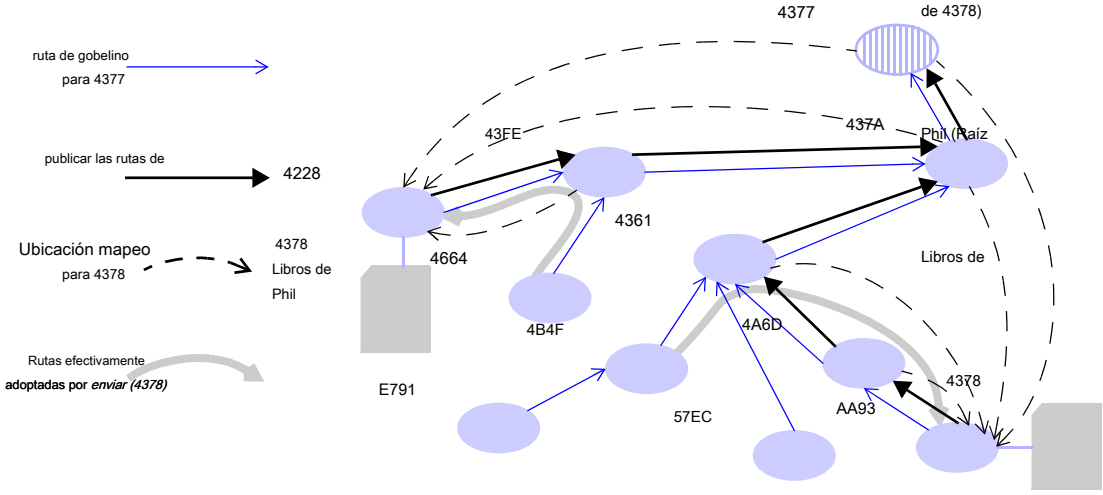
10.5.2 Tapiz

Tapiz implementa una tabla de dispersión distribuida y de las rutas de mensajes a nodos basados en GUID asociados a los recursos utilizando enrutamiento prefijo de una manera similar a pasteles. Pero el API del Tapiz oculta la tabla de dispersión distribuida de aplicaciones detrás de una interfaz DOLR como el que se muestra en la figura 10.5. Los nodos que poseen recursos utilizan el *publicar (GUID)* primitiva para darlos a conocer a la tapicería, y los titulares de los recursos siguen siendo responsables de su almacenamiento. recursos replicados se publican con el mismo GUID por cada nodo que contiene una réplica, dando como resultado múltiples entradas en la estructura de la tapicería de enrutamiento.

Esto da a las aplicaciones de la tapicería flexibilidad adicional: se puede colocar réplicas cerca (en la distancia de red) para usuarios frecuentes de los recursos con el fin de reducir la latencia y reducir al mínimo la carga de red o para asegurar la tolerancia de fallos de la red y de acogida. Pero esta distinción entre los pasteles y la tapicería no es fundamental: aplicaciones de repostería pueden lograr una flexibilidad similar al hacer que los objetos asociados con GUID simplemente actuar como representantes de los objetos de nivel de aplicación más complejos y la tapicería puede ser utilizado para implementar una tabla de dispersión distribuida en términos de su DOLR API [Dabek *et al.* 2003].

En la tapicería de identificadores de 160 bits se utilizan para referirse tanto a los objetos y a los nodos que realizan acciones de enrutamiento. Los identificadores son bien *NodeIds,* que se refieren a los ordenadores que realizan operaciones de encaminamiento, o *GUID,* que se refieren a los objetos. Para cualquier recurso con GUID *GRAMO* hay un nodo raíz único con GUID R_{GRAMO} que es numéricamente más cerca de *GRAMO.* Hospedadores *H* la celebración de las réplicas de *GRAMO* invocar periódicamente *publicar (G)* para asegurar que los ejércitos recién llegados se dan cuenta de la existencia de *GRAMO.* En cada invocación de *publicar (G)* un

Figura 10.10 la tapicería de enrutamiento *De Zhao et al. [2004]*



Réplicas del archivo *Libros de Phil* (G = 4378) se alojan en los nodos 4228 y AA93. Nodo 4377 es el nodo raíz para el objeto 4378. Las rutas de la tapicería que se muestran son algunas de las entradas en las tablas de enrutamiento. Los caminos publicar muestran rutas seguidas por los mensajes de publicar el que se establecen las asignaciones de ubicación en caché de objetos 4378. Las asignaciones de localización se utilizan posteriormente para enrutar los mensajes enviados a 4378.

publicar el mensaje se encamina desde el invocador hacia el nodo *R GRAMO*. Tras la recepción de un mensaje de publicar *R GRAMO* entra (*G, IP H),* la asignación entre *GRAMO* y la dirección IP del host de envío en su tabla de enrutamiento, y cada nodo a lo largo de la trayectoria de publicación almacena en caché la misma asignación. Este proceso se ilustra en la figura 10.10. Cuando los nodos **tienen múltiples (*G, IP)* asignaciones para el mismo GUID, que están ordenados por la distancia a la red (tiempo de ida y vuelta) a** la dirección IP. Para los objetos replicados esto se traduce en la selección de la réplica disponible más cercano del objeto como el destino para los mensajes posteriores enviados al objeto.

Zhao *et al. [*2004] dan detalles completos de los algoritmos de encaminamiento de la tapicería y la gestión de las tablas de enrutamiento de la tapicería de cara a la llegada y salida del nodo. Su trabajo incluye datos completos de evaluación del desempeño basado en la simulación de redes de gobelino a gran escala, lo que demuestra que su rendimiento es similar al de pastelería. En la sección 10.6.2 se describe el almacén de archivos Oceanstore, que ha sido construido y desplegado más de la tapicería.

10.5.3 De estructurado a estructurado peer-to-peer

La discusión hasta ahora se ha centrado exclusivamente en lo que se conoce como *estructurada Igualitaria* enfoques. En enfoques estructurados, existe una política global general que rige la topología de la red, la colocación de los objetos en la funciones de enrutamiento o de búsqueda utilizados para localizar objetos en la red de redes y. En otras palabras, hay una (distribuido) estructura de datos específica que sustenta la superposición asociado y un conjunto de algoritmos

Figura 10.11 estructurados o no estructurados sistemas-peer-to-peer

	peer-to-peer estructurado	*No estructurada peer-to-peer*
ventajas	Garantizada para localizar objetos (suponiendo que existan) y pueden ofrecer los límites de tiempo y la complejidad de esta operación; relativamente baja sobrecarga mensaje.	De auto-organización y naturalmente resistente a un fallo de nodo.
desventajas	Necesidad de mantener estructuras de superposición a menudo complejas, que pueden ser difícil y costoso de conseguir, especialmente en ambientes altamente dinámicos.	Probabilístico y por lo tanto no pueden ofrecer garantías absolutas sobre la localización de objetos; propensos a la sobrecarga de mensajería excesiva que puede afectar a la escalabilidad.

operando sobre esa estructura de datos. Esto puede verse claramente en los ejemplos de pastelería y de la tapicería en base a la tabla de dispersión distribuida subyacente y estructuras de anillo asociados. Debido a la estructura impuesta, tales algoritmos son eficientes y ofrecen límites de tiempo en la localización de objetos, pero al costo de mantenimiento de las estructuras subyacentes, a menudo en entornos altamente dinámicos.

Debido a este argumento de mantenimiento, *estructurado peer-to-peer* enfoques También se han desarrollado. En los enfoques no estructurados, no existe un control total sobre la topología o la colocación de los objetos dentro de la red. Por el contrario, la superposición se crea de una manera ad hoc, con cada nodo que se une a la red siguiendo algunas reglas simples y locales para establecer la conectividad. En particular, un nodo que se une establecerá contacto con un conjunto de *vecinos* sabiendo que los vecinos también estarán conectados a otros vecinos y así sucesivamente, formando una red que es fundamentalmente descentralizada y auto-organización y por lo tanto resistente a un fallo de nodo. Para localizar un objeto dado, entonces es necesario llevar a cabo una búsqueda de la topología de la red resultante; Claramente, este enfoque no puede ofrecer ninguna garantía de ser capaz de encontrar el objeto y el rendimiento será impredecible. Además, existe un riesgo real de generar tráfico de mensajes excesiva para localizar objetos.

Un resumen de las fuerzas relativas de los sistemas estructurados y no estructurados-peer-to-peer se proporciona en la figura 10.11. Es interesante reflexionar que, a pesar de las aparentes desventajas de los sistemas no estructurados peer-to-peer, este enfoque es dominante en Internet, sobre todo en el apoyo de intercambio de archivos peer-to-peer (con sistemas como Gnutella, Freenet y BitTorrent toda la adopción enfoques no estructurados). Como se verá, los avances significativos se han hecho en este tipo de sistemas para mejorar el rendimiento de los enfoques no estructurados y este trabajo es importante dada la cantidad de tráfico generado por intercambio de archivos peer-to-peer en Internet (por ejemplo, un estudio llevado a cabo en los años 2008/9 indica que el intercambio de archivos peer-to-peer representan entre el 43% y el 70% de todo el tráfico de Internet, www.ipoque.com]).

Estrategias para la búsqueda efectiva • En intercambio de archivos peer-to-peer, todos los nodos en los archivos de la oferta de red a la mayor medio ambiente. Como se mencionó anteriormente, el problema de la localización de un fichero mapea entonces en una búsqueda de toda la red para localizar archivos correspondientes. Si se implementa ingenuamente, esto se llevaría a cabo al inundar la red con peticiones. Esta es precisamente la estrategia adoptada en las primeras versiones de Gnutella. En particular, en Gnutella

0.4, cada nodo remitió una solicitud para cada uno de sus vecinos, que luego a su vez transmiten esta información a sus vecinos, y así sucesivamente hasta que se encuentre una coincidencia. Cada búsqueda también se vio limitada con un campo de tiempo de vida que limita el número de saltos. En el tiempo de Gnutella
0.4 se desplegó, la conectividad promedio fue de alrededor de 5 vecinos por nodo. Este enfoque es simple, pero no escala y rápidamente se inunda la red con tráfico basada en búsqueda.

Una serie de mejoras se han desarrollado para la búsqueda de redes no estructuradas [Lv *et al.* 2002, Tsoumakos y Roussopoulos 2006], incluyendo:

Búsqueda anillo ampliado: En este enfoque, el nodo iniciar lleva a cabo una serie de búsquedas con el aumento de los valores en el campo-tiempo de vida, reconociendo que un número significativo de las solicitudes se cumplirá localmente (especialmente si se combina con una estrategia de replicación eficaz, como se discute más adelante).

paseos aleatorios: Con paseos aleatorios, el nodo de iniciación pone en marcha una serie de caminantes que siguen sus propios caminos aleatorios a través del gráfico interconectado ofrecido por la superposición no estructurada.

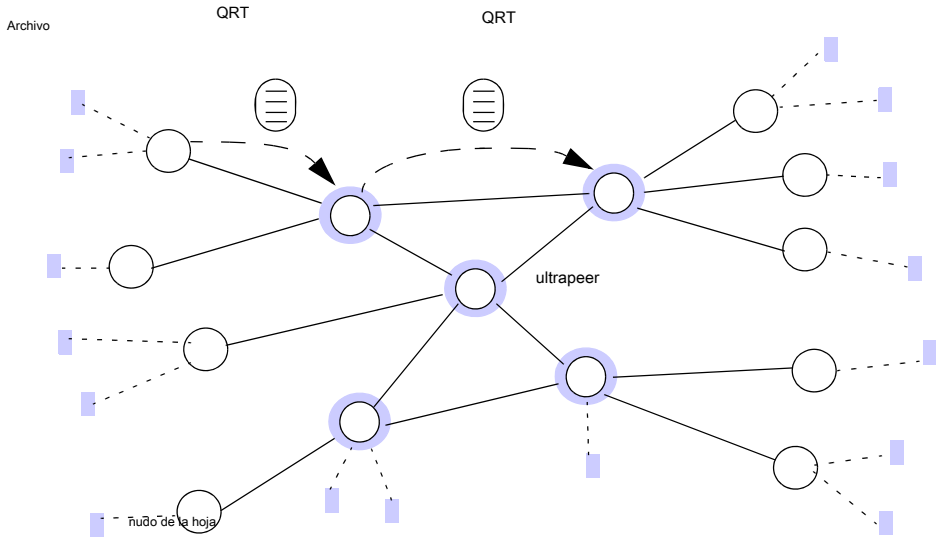
Chismoso: En los enfoques Cotilleo, un nodo envía una solicitud a un vecino dado con una cierta probabilidad, y por lo tanto las solicitudes se propagan a través de la red de una manera similar a un virus a través de una población (por ejemplo, protocolos de chismes también se denominan *protocolos de epidemia).* La probabilidad o bien puede ser fijado para una red dada o calcularse dinámicamente basándose en la experiencia previa y / o el contexto actual. (Tenga en cuenta que chismes es una técnica común en sistemas distribuidos; aplicaciones adicionales se pueden encontrar en los capítulos 6 y 18).

Tales estrategias pueden reducir significativamente la sobrecarga de búsqueda en las redes estructuradas y por lo tanto aumentar la escalabilidad de los algoritmos. Tales estrategias son también a menudo apoyados por apropiado *replicación* técnicas. Al replicar contenido a través de una serie de pares, la probabilidad de descubrimiento eficiente de los archivos en particular se mejora de manera significativa. Las técnicas incluyen la replicación del conjunto de archivos y la dispersión de fragmentos de archivos a través de Internet - un enfoque que se utiliza con eficacia en BitTorrent, por ejemplo, para reducir la carga en cualquiera de pares en la descarga de archivos de gran tamaño (véase el Capítulo 20).

Estudio de caso: • Gnutella Gnutella fue lanzado originalmente en 2000 y desde entonces ha crecido hasta convertirse en una de las aplicaciones dominantes y más influyentes de intercambio de archivos peer-to-peer. Como se mencionó anteriormente, en un principio el protocolo adoptado una estrategia inundaciones bastante simple que no tuviesen el tamaño particularmente bien. En respuesta a esto, Gnutella 0,6 introducido una serie de modificaciones que han mejorado significativamente el rendimiento del protocolo.

La primera modificación importante fue pasar de una pura arquitectura de igual a igual, donde todos los nodos son iguales a uno donde todos los compañeros siguen cooperando para ofrecer el servicio, pero algunos nodos, designados tener recursos adicionales, como son elegidos *ultrapeers,* y formar el corazón de la red, con otros pares, teniendo sobre el papel de nodos hoja (u hojas). Las hojas se conectan a un pequeño número de ultrapeers, que están fuertemente conectadas a otras ultrapeers (con más de 32 conexiones de cada uno). Esto reduce drásticamente el número máximo de saltos necesarios para la búsqueda exhaustiva. Este estilo de arquitectura de punto a punto se conoce como una arquitectura híbrida y también es el enfoque adoptado en Skype (como se discute en la Sección 4.5.2).

Figura 10.12 Los elementos clave en el protocolo Gnutella



La otra mejora clave fue la introducción de un protocolo de enrutamiento de consulta (QRP) diseñado para reducir el número de consultas emitidas por los nodos. El protocolo se basa en el intercambio de información acerca de los archivos contenidos en los nodos y por tanto sólo reenviar consultas por caminos donde el sistema cree que habrá un resultado positivo. En lugar de compartir información sobre los archivos directamente, el protocolo produce un conjunto de números de hash en las palabras individuales en un nombre de archivo. Por ejemplo, un nombre de archivo, tales como “El capítulo diez en P2P” estará representada por cuatro números, por ejemplo <65, 47, 09, 76>. Un nodo produce una tabla de enrutamiento de consulta (QRT) que contiene los valores hash que representan cada uno de los archivos contenidos en ese nodo que posteriormente envía a todos sus ultrapeers asociados. Ultrapeers entonces producen sus propias tablas de consulta de enrutamiento basado en la unión de todas las entradas de todas las hojas conectados junto con las entradas de los archivos contenidos en ese nodo, y el intercambio de éstos con otros ultrapeers conectadas. De esta manera, ultrapeers pueden determinar qué rutas ofrecen una ruta válida para una determinada consulta, lo que reduce significativamente la cantidad de tráfico innecesario. Más específicamente, un ultrapeer enviará una consulta a un nodo si se encuentra una coincidencia (lo que indica que el nodo tiene el archivo) y llevará a cabo el mismo cheque antes de pasarlo a otro ultrapeer si es el último salto en el fichero. Tenga en cuenta que, con el fin de evitar la sobrecarga de ultrapeers, nodos enviará una consulta a una ultrapeer a la vez y luego esperar por un período determinado para ver si consiguen una respuesta positiva.

Por último, una consulta en Gnutella contiene la dirección de red de la ultrapeer iniciar, lo que implica que una vez que se encuentra un archivo puede ser enviado directamente a la ultrapeer asociado (utilizando UDP), evitando un recorrido inverso a través de la gráfica.

Los principales elementos asociados con Gnutella 0.6 se resumen en la figura 10.12.

10.6 casos prácticos de aplicación: ardilla, Oceanstore, Ivy

Las capas de superposición de encaminamiento descritos en la sección anterior se han explotado en varios experimentos de aplicación y de las aplicaciones resultantes han sido ampliamente evaluado. Hemos escogido tres de ellos para el estudio adicional, el servicio de almacenamiento en caché ardilla web basado en pastelería, y los almacenes de archivos Oceanstore y la hiedra.

caché web 10.6.1 ardilla

Los autores de pastelería han desarrollado el servicio de ardilla peer-to-peer de almacenamiento en caché web para su uso en redes locales de ordenadores personales [Iyer *et al.* 2002]. En las redes locales medianas y grandes de almacenamiento en caché web se realiza típicamente usando un ordenador servidor dedicado o clúster. El sistema de ardilla realiza la misma tarea mediante la explotación de los recursos informáticos y de almacenamiento ya disponibles en los ordenadores de sobremesa en la red local. En primer lugar, damos una breve descripción general del funcionamiento de un servicio de almacenamiento en caché web, a continuación, describimos el diseño de ardilla y una revisión de su eficacia.

• **almacenamiento en caché Web** navegadores web generan HTTP *OBTENER* solicitudes de objetos de Internet como páginas HTML, imágenes, etc. Estos pueden ser atendidas desde una memoria caché del navegador en la máquina cliente, a partir de una *caché de proxy web (* un servicio que se ejecuta en otro equipo en la misma red local o en un nodo de cerca en Internet) o de la *servidor Web de origen (* el servidor cuyo nombre de dominio está incluida en los parámetros de la *OBTENER* solicitud), dependiendo de que contiene una copia nueva del objeto. Los cachés de proxy local y cada uno contiene un conjunto de objetos recuperados recientemente organizados para la búsqueda rápida por URL. Algunos objetos son no almacenable en caché ya que son generados dinámicamente por el servidor en respuesta a cada solicitud.

Cuando un caché del navegador o caché de proxy web recibe una *OBTENER* solicitud, hay tres posibilidades: el objeto solicitado es almacenable en caché, hay un error de caché o el objeto se encuentra en la caché. En los dos primeros casos la solicitud se envía al siguiente nivel hacia el servidor Web de origen. Cuando el objeto solicitado se encuentra en una memoria caché, la copia en caché debe ser probado para la frescura.

Los objetos Web se almacenan en servidores web y servidores de caché con algunos valores de metadatos adicionales, incluyendo una marca de tiempo que da una *fecha de la última modificación (T)* y posiblemente una *tiempo-a-vivo (t)* o un *etag (* un hash calculado a partir de los contenidos de una página web). Estos elementos de metadatos son suministrados por el servidor de origen siempre que un objeto se devuelve a un cliente.

Los objetos que tienen un tiempo a vivir asociado *t,* si se consideran frescos *T + t* es posterior a la corriente en tiempo real. Para los objetos sin un time-to-live, un valor estimado de *t* se utiliza (a menudo sólo unos pocos segundos). Si el resultado de esta evaluación es positiva frescura, el objeto almacenado en caché se devuelve al cliente sin contactar con el servidor Web de origen. De lo contrario, un condicional *GET (CGET)* invitación procede al siguiente nivel para su validación. Hay dos tipos básicos de *CGET* solicitudes: una *Si-Modified-Since* solicitud que contiene la indicación de la hora de la última modificación conocida, y una *If-None-Match* solicitud que contiene una

etag que representa el contenido de objeto. Esta *CGET* solicitud puede ser realizado ya sea por otro caché web o por el servidor de origen. Un caché web que recibe una *CGET* solicitud y no tiene una copia nueva del objeto envía la solicitud a la web origen

servidor. La respuesta contiene o bien todo el objeto, o una *no modificado* mensaje si el objeto en caché no se ha modificado.

Siempre que un objeto puede almacenar en caché recién modificado se recibe desde el servidor de origen, que se añade al conjunto de objetos en la memoria caché local (desplazamiento de los objetos más antiguos que siguen siendo válidos si es necesario) junto con una marca de tiempo, un tiempo a vivir y una *etag* si está disponible.

El esquema descrito anteriormente es la base del funcionamiento de los servicios de almacenamiento en caché de proxy web centralizados desplegados en la mayoría de las redes locales de apoyo a un gran número de clientes web. Proxy cachés web se implementan normalmente como un proceso de múltiples subprocesos que se ejecutan en un único host dedicado o un conjunto de procesos que se ejecutan en un clúster de ordenadores y requieren una cantidad considerable de recursos informáticos dedicados en ambos casos.

Ardilla • El servicio de almacenamiento en caché web ardilla realiza las mismas funciones que utilizan una pequeña parte de los recursos de cada equipo cliente en una red local. La función de hash seguro SHA-1 se aplica a la URL de cada objeto almacenado en caché para producir una pastelería GUID de 128 bits. Desde el GUID no se utiliza para validar el contenido, no tiene que basarse en el contenido completo de objetos, como lo es en otras aplicaciones de pastelería. Los autores de ardilla basan su justificación para esto en el argumento de extremo a extremo (Sección 2.3.3), con el argumento de que la autenticidad de una página web puede verse comprometida en muchos puntos en su viaje desde el host al cliente; autentificación de páginas en caché añade poco a ninguna garantía general de la autenticidad y el protocolo HTTPS (incorporación de extremo a extremo Transport Layer Security, discutido en la Sección 11.6.

En la implementación más simple de ardilla - que resultó ser la más efectiva - el nodo cuyo GUID es numéricamente más cercano al GUID de un objeto se convierte en ese objeto de *nodo de casa,* responsable de mantener la copia en caché del objeto cuando hay uno.

Los nodos cliente están configurados para incluir un proceso ardilla proxy local, que asume la responsabilidad de tanto el almacenamiento en caché local y remota de objetos web. Si una nueva copia de un objeto deseado no está en la caché local, rutas ardilla una *Obtener* petición o una *CGET* solicitud (cuando hay una copia inactiva del objeto en la caché local) a través de pastelería al nodo casa. Si el nodo casa tiene una copia nueva, que responde directamente al cliente con una *no modificado*
mensaje o una nueva copia, según sea apropiado. Si el nodo casa tiene una copia inactiva o ninguna copia del objeto, emite una *CGET* o una *Obtener* al servidor de origen, respectivamente. El servidor de origen puede responder con una *no modificado* mensaje o una copia del objeto. En el primer caso, el nodo de casa revalida su entrada en la caché y envía una copia del objeto al cliente. En este último caso, se envía una copia del nuevo valor para el cliente y coloca una copia en su caché local si el objeto es cacheable.

• Evaluación de la ardilla Squirrel fue evaluada por simulación utilizando cargas modeladas derivadas de residuos de la actividad de las memorias caché de proxy web existentes centralizada en dos entornos reales de trabajo dentro de Microsoft, uno con 105 clientes activos (en Cambridge) y el otro con más de 36 000 (en Redmond). La evaluación comparó el rendimiento de un caché web de ardilla con una centralizada en tres aspectos:

La reducción en ancho de banda externa total utilizado: El ancho de banda externa total está inversamente relacionada con la proporción de aciertos, porque se encuentra a fallos de caché que generan solicitudes a los servidores web externos. Las proporciones de golpe observados para servidores de caché web centralizado eran 29% (para Redmond) y 38% (para Cambridge). Cuando los mismos registros de actividad

se utilizaron para generar una carga simulada para la memoria caché de la ardilla, con cada cliente contribuyendo 100 Mbytes de almacenamiento en disco, se alcanzaron proporciones de éxito muy similares de 28% (Redmond) y 37% (Cambridge). De ello se desprende que el ancho de banda externa se reduciría en una proporción similar.

La latencia percibida por los usuarios para acceder a los objetos web: El uso de un enrutamiento resultados de superposición de varias transferencias de mensajes (saltos de enrutamiento) a través de la red local para transmitir una petición de un cliente al host responsable de almacenamiento en caché el objeto relevante (el nodo de origen). La media del número de saltos de enrutamiento observados en la simulación fueron 4.11 saltos para ofrecer una *OBTENER* solicitud en el caso de Redmond y 1,8 lúpulo en el caso de Cambridge, mientras que sólo se requiere una única transferencia de mensajes para acceder a un servicio de caché centralizada.

Sin embargo las transferencias locales toman sólo unos pocos milisegundos con hardware Ethernet moderna, incluyendo el tiempo de establecimiento de conexión TCP, mientras que las transferencias de mensajes TCP área de ancho a través de Internet requieren 10-100 ms. Por lo tanto, los autores argumentan que la ardilla de la latencia de acceso a los objetos encontrados en la caché está inundado por el mucho mayor latencia de acceso a los objetos que no se encuentran en la memoria caché, dando una experiencia de usuario similar a la proporcionada con un caché centralizada.

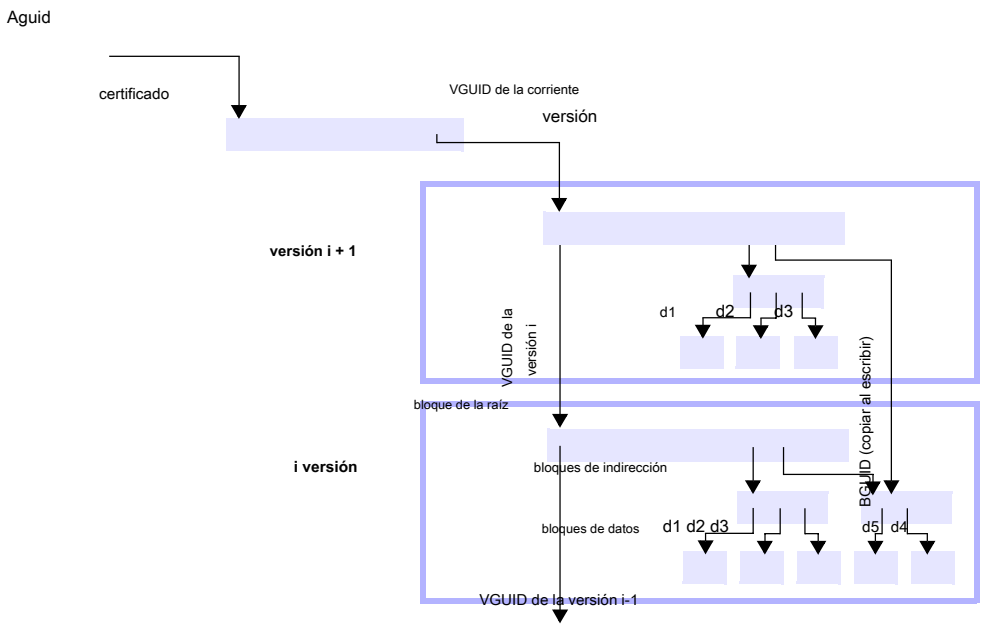
La carga de cálculo y almacenamiento impuesta a los nodos cliente: El número medio de solicitudes de caché servido para otros nodos de cada nodo durante todo el período de la evaluación era extremadamente baja, de sólo 0,31 por minuto (Redmond), lo que indica que la proporción global de recursos del sistema se consumen es extremadamente bajo. Basándose en las mediciones descritas anteriormente, los autores de la ardilla concluyeron que su rendimiento es comparable a la de una memoria caché centralizada. Ardilla logra una reducción en la latencia observada para el acceso a página Web similar a la alcanzable por un servidor caché centralizada con una caché dedicada de tamaño similar. La carga adicional impuesta a los nodos cliente es baja y probable que sea imperceptible para los usuarios. El sistema de ardilla se desplegó posteriormente como la caché web principal en una red local con 52 máquinas de cliente utilizando ardilla, y los resultados confirmaron sus conclusiones.

almacén de archivos 10.6.2 Oceanstore

Los desarrolladores de la tapicería han diseñado y construido un prototipo para un almacén de archivos peer-to-peer. A diferencia pasado, se admite el almacenamiento de archivos mutables. El diseño Oceanstore [Kubiatowicz *et al.* 2000; Kubiatowicz 2003; ñandú *et al.* 2001, 2003] tiene como objetivo proporcionar una escala muy grande, instalación de almacenamiento persistente de forma incremental escalable para objetos de datos mutables con la persistencia y la fiabilidad a largo plazo en un entorno en constante cambio de la red y los recursos informáticos. Oceanstore está destinado para su uso en una variedad de aplicaciones, incluyendo la implementación de un servicio NFS-como archivo, alojamiento de correo electrónico, bases de datos y otras aplicaciones que implican el intercambio y el almacenamiento persistente de un gran número de objetos de datos.

El diseño incluye la disposición para el almacenamiento replicado de ambos objetos de datos mutables e inmutables. El mecanismo para mantener la coherencia entre las réplicas se puede adaptar a las necesidades de aplicación de una manera que fue inspirado por el sistema de Bayou (Sección 18.4.2). Privacidad y la integridad se logran a través de la encriptación de los datos y el uso de un protocolo de acuerdo bizantino (véase la Sección 15.5) para cambios a replicado

Figura 10.13 organización de almacenamiento de objetos Oceanstore



Versión *i* + 1 se ha actualizado en los bloques *d1, d2* y *d3*. El certificado y los bloques de raíces incluyen algunos metadatos que no se muestra.
Todas las flechas sin etiqueta son BGUIDs.

objetos. Esto es necesario porque la fiabilidad de los hosts individuales no se puede asumir.

Un prototipo Oceanstore, llamado Estanque [Rhea *et al.* 2003], se ha construido. Es lo suficientemente completa para soportar aplicaciones y su rendimiento ha sido evaluado contra una variedad de puntos de referencia con el fin de validar el diseño Oceanstore y comparar su rendimiento con los enfoques más tradicionales. En el resto de esta sección damos una visión general del diseño Oceanstore / Estanque y resumir los resultados de la evaluación.

Pond utiliza el mecanismo de superposición de enrutamiento Tapiz para colocar bloques de datos en los nodos distribuidos a través de Internet y de despachar las solicitudes a ellos.

• organización de almacenamiento objetos de datos Oceanstore / estanque son análogos a los archivos, con sus datos almacenados en un conjunto de bloques. Pero cada objeto se representa como una secuencia ordenada de versiones inmutables que son (en principio) Se mantuvo siempre. Cualquier actualización a un objeto como resultado la generación de una nueva versión. Las versiones comparten todos los bloques sin cambios, siguiendo la técnica de copia por escritura para crear y actualizar objetos se describe en la Sección

7.4.2. Por lo que una pequeña diferencia entre las versiones requiere sólo una pequeña cantidad de espacio de almacenamiento adicional.

Los objetos están estructurados de una manera que es una reminiscencia del sistema de archivo Unix, con los bloques de datos organizados y se accede a través de un bloque de metadatos llamado el bloque de raíz y bloques de **indirección adicionales si es necesario (*cf.* Unix *i-nodos).* Otro nivel de indirección se utiliza para asociar un textual persistente o** otro nombre externamente visible (por ejemplo, el nombre de ruta de un archivo) con la secuencia de versiones de un objeto de datos. Figura

Figura 10.14 Tipos de identificador usado en Oceanstore

Nombre	*Sentido*	*Descripción*
BGUID	bloque de GUID	control seguro de un bloque de datos
VGUID	versión GUID	BGUID del bloque de la raíz de una versión
Aguid	GUID activa	Identifica de forma exclusiva todas las versiones de un objeto

10.13 ilustra esta organización. GUID se asocian con el objeto (una Aguid), el bloque de raíz para cada versión del objeto (una VGUID), los bloques de indirección y los bloques de datos (BGUIDs). Varias réplicas de cada bloque se almacenan en los nodos pares seleccionados de acuerdo a criterios de localidad y disponibilidad de almacenamiento, y sus GUID se publican (utilizando el *publicar()* primitivo de la Figura 10.5) por cada uno de los nodos que contiene una réplica de manera que la tapicería puede ser utilizado por los clientes acceder a los bloques.

Se utilizan tres tipos de GUID, como se resume en la figura 10.14. Los dos primeros son los GUID de tipo normalmente asignados a los objetos almacenados en la tapicería - que se calculan a partir de los contenidos del bloque pertinente utilizando una función de hash seguro para que puedan ser utilizados más tarde para autenticar y verificar la integridad de los contenidos. Los bloques de los que hacen referencia son necesariamente inmutable, ya que cualquier cambio en el contenido de un bloque de invalidaría el uso del GUID como un token de autenticación.

El tercer tipo de identificador usado es AGUIDs. Estos se refieren (indirectamente) a toda la corriente de versiones de un objeto, permitiendo a los clientes acceder a la versión actual del objeto o de cualquier versión anterior. Dado que los objetos almacenados son mutables, los GUID utilizan para identificar ellos no se pueden derivar de su contenido, porque eso sería hacer que los GUID celebradas en los índices, etc., siempre que un objeto obsoleto cambió.

En su lugar, cada vez que se crea un nuevo objeto de almacenamiento de un Aguid permanente se genera mediante la aplicación de una función hash seguro a un nombre específico de la aplicación (por ejemplo, un nombre de archivo) suministrado por el cliente de crear el objeto y una clave pública que representa el propietario del objeto (véase la sección 11.2.5). En una aplicación de sistema de archivo, un Aguid se almacena en los directorios en contra de cada nombre de archivo.

La asociación entre un Aguid y la secuencia de versiones del objeto que identifica se registra en un certificado firmado que se almacena y se replica por un esquema de replicación copia principal (también llamada replicación pasiva; véase la Sección 18.3.1). El certificado incluye el VGUID de la versión actual y el bloque de raíz para cada versión contiene la VGUID de la versión anterior, por lo que no es una cadena de referencias permite a los clientes que tienen un certificado para atravesar toda la cadena de versiones (Figura 10.13). Se necesita un certificado firmado para asegurarse de que la asociación es auténtico y se ha hecho por un director autorizado. Los clientes deben comprobar esto. Siempre que se crea una nueva versión de un objeto, se genera un nuevo certificado de la celebración de la VGUID de la nueva versión, junto con una marca de tiempo y un número de secuencia de versión.

El modelo de confianza para los sistemas de punto a punto requiere que la construcción de cada nuevo certificado se acordó (como se describe más abajo) entre un pequeño conjunto de hosts llamado el *anillo interior*. Cada vez que un nuevo objeto se almacena en Oceanstore, se selecciona un conjunto de hosts para actuar como el anillo interior para ese objeto. Utilizan tapicería de *publicar()* primitiva para hacer el Aguid para el objeto conocido hasta la tapicería. Los clientes pueden utilizar la tapicería a las peticiones de ruta para el certificado del objeto a uno de los nodos en el anillo interior.

El nuevo certificado sustituye a la antigua copia primaria celebrada en cada nodo anillo interior y se difunde a un mayor número de copias secundarias. Se deja a los clientes para determinar la frecuencia con que comprobar si hay una nueva versión (una decisión similar tiene que ser tomada por las copias caché de los archivos en NFS, la mayoría de las instalaciones operan con una ventana consistencia de 30 segundos entre el cliente y el servidor; véase la Sección 12.3) .

Como es habitual en los sistemas de punto a punto, la confianza no puede ser colocado en cualquier huésped individual. La actualización de las copias principales requiere acuerdo de consenso entre los hosts en el anillo interior. Ellos usan una versión de un algoritmo bizantino acuerdo basado en el estado de la máquina descrito por Castro y Liskov [2000] para actualizar el objeto y firmar el certificado. El uso de un protocolo de acuerdo bizantino asegura que el certificado se mantiene correctamente, incluso si algunos miembros del anillo interior fallan o se comportan de forma malintencionada. Debido a que los costes computacionales y de comunicación de subida acuerdo bizantino con el cuadrado del número de hosts que participan, el número de hosts en el anillo interior se mantiene pequeño y el certificado resultante se replica más ampliamente utilizando el esquema de copia primaria mencionada anteriormente.

Realización de una actualización también implica la comprobación de los derechos de acceso y la serialización de la actualización con cualquier otro escribe pendientes. Una vez que el proceso de actualización se ha completado para la copia principal, los resultados se difunden a las réplicas secundarias almacenados en hosts fuera del anillo interior mediante un árbol de encaminamiento de multidifusión que es administrado por la tapicería.

Debido a su naturaleza de sólo lectura, los bloques de datos se replican por un mecanismo diferente más, eficiente almacenamiento. Este mecanismo se basa en la división de cada bloque en *metro* fragmentos de igual tamaño, que se codifican usando *códigos de borrado [*Weatherspoon y Kubiatowicz 2002] para *norte* fragmentos, donde *n*> *m*. La propiedad clave de codificación de borrado es que es posible reconstruir un bloque de cualquier *metro* de sus fragmentos. En un sistema que utiliza codificación de borrado todos los objetos de datos permanecen disponibles con la pérdida de hasta *norte* - *metro* Hospedadores. En la implementación de la charca *m* = 16 y *n* = 32, por lo que para una duplicación del costo de almacenamiento, el sistema puede tolerar el fallo de hasta 16 hosts sin pérdida de datos. Tapiz se utiliza para almacenar fragmentos en y recuperarlas de la red.

Este alto nivel de tolerancia a fallos y la disponibilidad de datos se consigue con cierto coste en términos de reconstrucción de bloques de fragmentos de borrado-codificada. Para minimizar el impacto de esta, todo los bloques también se almacenan en la red utilizando la tapicería. Puesto que pueden ser reconstruidos a partir de sus fragmentos, estos bloques se tratan como una cache - que no son tolerantes a errores y que pueden ser desechadas cuando se requiere espacio de almacenamiento.

Actuación • Estanque fue desarrollado como un prototipo para demostrar la viabilidad de un servicio de archivos escalable peer-to-peer, en lugar de como una aplicación de producción. Está implementado en Java, e incluye casi todo el diseño esbozado anteriormente. Se evaluó contra varios puntos de referencia de diseño especial y de una forma sencilla la emulación de un cliente NFS y el servidor en términos de objetos Oceanstore. Los desarrolladores probaron la emulación de NFS con el baremo Andrew [Howard *et al.* 1988], que emula a una carga de trabajo de desarrollo de software. La tabla de la figura 10.15 muestra los resultados para este último. Ellos se obtuvieron utilizando 1 GHz PC Pentium III que ejecuta Linux. Las pruebas de LAN se realizaron utilizando un Ethernet Gigabit y los resultados WAN se obtuvieron utilizando dos conjuntos de nodos conectados por el Internet.

Las conclusiones extraídas por los autores fueron que el rendimiento de Oceanstore / Estanque cuando se opera a través de una red de área amplia (es decir, Internet) excede sustancialmente la de NFS para la lectura y está dentro de un factor de tres de NFS para

Figura 10.15 Evaluación del desempeño del prototipo Pond emulando NFS

	LAN		*PÁLIDO*		*predominantes en las*
Fase	*Linux NFS*	*Estanque*	*Linux NFS*	*Estanque*	*operaciones de referencia*
1	0.0	1.9	0.9	2.8	Lee y escribe
2	0.3	11.0	9.4	16.8	Lee y escribe
3	1.1	1.8	8.3	1,8	Leer
4	0.5	1.5	6.9	1,5	Leer
5	2.6	21.0	21.5	32.0	Lee y escribe
Total	4.5	37.2	47.0	54.9	

Las cifras muestran los tiempos en segundos para ejecutar diferentes fases de la referencia Andrew. Tiene cinco fases: (1) Crea subdirectorios recursivamente, (2) copias de un árbol de fuentes, (3) examina el estado de todos los archivos en el árbol sin examinar sus datos, (4) examina cada byte de datos en todos los archivos y (5) compila y enlaza los archivos.

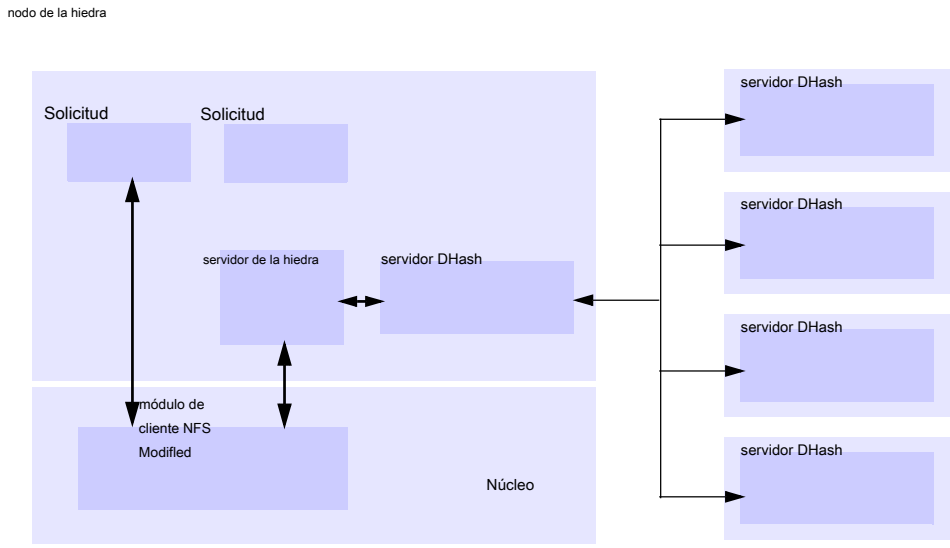
actualización de los archivos y directorios; los resultados de LAN fueron sustancialmente peor. En general, los resultados sugieren que un servicio de archivos peer-to-peer a escala de Internet basado en el diseño Oceanstore sería una solución eficaz para la distribución de archivos que no cambian muy rápidamente (como las copias en caché de páginas web). Su potencial para su uso como una alternativa a NFS es cuestionable incluso para redes de área amplia y es claramente no competitivo para su uso puramente local.

Estos resultados se obtuvieron con los bloques de datos almacenados sin fragmentación y replicación basada en el borrado de código. El uso de claves públicas contribuye sustancialmente al coste computacional de la operación de la charca. Las cifras indicadas son para las claves de 512 bits, cuya seguridad es buena, pero menos que perfecto. Los resultados para las claves de 1024 bits fueron sustancialmente peores para las fases de esos puntos de referencia que participan actualizaciones de archivos. Otros resultados obtenidos con los puntos de referencia de diseño especial incluyen la medición del impacto del proceso de acuerdo bizantino sobre la latencia de cambios. Estos estaban en el rango de 100 ms a 10 segundos. Una prueba de la actualización de rendimiento logra un máximo de 100 actualizaciones / segundo.

sistema de archivos 10.6.3 Ivy

Al igual que Oceanstore, Ivy [Muthitacharoen *et al.* 2002] es un sistema de archivos de lectura / escritura que soporta múltiples lectores y escritores implementadas a través de una capa de enrutamiento de superposición y una distribución de hachís dirigida almacén de datos. A diferencia de Oceanstore, el sistema de archivos de la hiedra emula un servidor Sun NFS. Ivy almacena el estado de los archivos como los registros de las solicitudes de actualización de archivos emitidos por clientes de la hiedra y reconstruye los archivos mediante el escaneo de los registros siempre que sea incapaz de satisfacer una petición de acceso desde su caché local. Las entradas de registro se llevan a cabo en el **servicio de almacenamiento hashaddressed DHash distribuido [Dabek *et al.* 2001]. (Registros primero fueron utilizados para registrar las** actualizaciones de archivos en el sistema operativo distribuido Sprite [Rosenblum y Ooosterhout 1992], tal como se describe brevemente en la Sección 12.5, pero no se utilizan simplemente para optimizar el rendimiento de la actualización del sistema de archivos.)

Figura 10.16 arquitectura del sistema Ivy



El diseño de la hiedra resuelve varios problemas no resueltos anteriormente que surgen de la necesidad de alojar archivos en máquinas de confianza parcial o poco fiables, incluyendo:

- El mantenimiento de los metadatos del archivo consistente (*cf. i-nodo* contenidos en los sistemas de archivos Unix / NFS) con actualizaciones de archivos potencialmente concurrentes a diferentes nodos. Bloqueo no se utiliza debido a la falta de nodos o conectividad de red podría causar el bloqueo indefinido.

- confianza parcial entre los participantes y la vulnerabilidad a los ataques de las máquinas participantes. La recuperación de fallos de integridad causadas por este tipo de ataques se basa en la noción de *puntos de vista* del sistema de archivos. Una vista es una representación del estado construido a partir de los registros de las actualizaciones hechas por un conjunto de participantes. Los participantes pueden ser removidos y una vista sin vuelven a calcular sus actualizaciones. Así, un sistema de archivos compartidos es visto como el resultado de la fusión de todas las actualizaciones realizadas por un conjunto de participantes (dinámicamente seleccionado).

- La operación continua durante las particiones de la red, lo que puede dar lugar a cambios en conflicto a los archivos compartidos. cambios conflictivos se resuelven utilizando métodos similares a las usadas en el sistema de archivos Coda (Sección 18.4.3). Ivy implementa una API en cada nodo cliente que se basa en el protocolo de servidor NFS (similar al conjunto de las operaciones listadas en la Sección 12.3, figura 12.9). Los nodos cliente incluyen un proceso de servidor que utiliza Ivy DHash para almacenar y registros de registro de acceso en los nodos a través de una red de área local o amplia basado en claves (GUID) que se calculan como el hash de los contenidos de registro (véase la figura 10.16). DHash implementa una interfaz de programación como la que se muestra en la figura 10.4 y se replica todas las entradas en varios nodos para

resistencia y disponibilidad. Los autores señalan que la hiedra DHash en principio, podría ser reemplazado por otro almacén de hash con su dirección y distribuido, como los pasteles, la tapicería o CAN.

Un almacén de archivos Ivy consiste en un conjunto de registros de actualización del registro, uno por participante. Cada participante Ivy agrega solamente a su propio registro, pero puede leer todos los registros que componen el sistema de archivos. Las actualizaciones se almacenan en registros separados por cada participante de manera que se pueden revertir en caso de fallos de seguridad o fallos de consistencia.

Un registro de Ivy es una lista ordenada por tiempo inverso relacionado de entradas de registro. Cada entrada del registro es un registro de sellos de tiempo de una petición de cliente a cambiar el contenido o metadatos de un archivo o directorio. DHash utiliza el 160-bit hash SHA-1 de un registro como una clave para colocar y recuperar el registro. Cada participante también mantiene un bloque DHash mutable (llamado *log-cabeza)* que apunta al registro de registro más reciente del participante. bloques mutables se les asigna un par de claves públicas de cifrado por su propietario. Los contenidos del bloque se firman con la clave privada y, por tanto, pueden ser autenticados con la clave pública correspondiente. Ivy utiliza vectores de versión (es decir, las marcas de tiempo del vector; véase la Sección 14.4) para imponer un orden total en las entradas de registro cuando se lee a partir de múltiples registros.

DHash almacena un registro de registro usando un hash SHA-1 de su contenido como la clave. entradas de registro están encadenados en el orden de llegada usando la tecla DHash como un enlace. El registro de cabeza es la clave para la entrada de registro más reciente. Para almacenar y recuperar log-cabezas, un par de claves pública se calcula por el titular del registro. El valor de la clave pública se utiliza como clave DHash y la clave privada es utilizada por el propietario para firmar el registro de cabeza. Cualquier participante que tiene la clave pública puede recuperar el registro de cabeza y lo utilizan para acceder a todos los registros en el registro.

Suponiendo un sistema de archivos integrado por un único registro por el momento, el método de ejecución canónica para una solicitud para leer una secuencia de bytes de un archivo requiere un análisis secuencial del registro para encontrar los registros que contienen actualizaciones para la parte correspondiente de la archivo. Los registros son de longitud ilimitada, pero la exploración termina cuando el primer registro o registros se encontraron que cubre la secuencia requerida de bytes.

El algoritmo canónica para acceder a un multi-usuario, sistema de archivo de varias de registro consiste en la comparación de las marcas de tiempo de vectores en las entradas de registro para determinar el orden de cambios (ya que un reloj global no se puede suponer).

El tiempo necesario para llevar a cabo este proceso para una operación tan simple como una *leer* solicitud es potencialmente muy larga. Se reduce a una duración más tolerable y predicable mediante el uso de una combinación de cachés locales y *instantáneas*. Las instantáneas son representaciones del sistema de archivos calcula y almacena localmente por cada participante como un subproducto de su uso de los registros. Constituyen una representación suave del sistema de archivos en el sentido de que pueden ser invalidadas si un participante es expulsado del sistema.

Actualización de la consistencia es *cerca de abrir;* es decir, los cambios realizados en un archivo por una aplicación no son visibles para otros procesos hasta que el archivo está cerrado. El uso de un modelo de consistencia closeto-abierta permite *escribir* operaciones en un archivo que se guardan en el nodo cliente hasta que el archivo está cerrado; a continuación, todo el conjunto de *escribir* las operaciones se escribe como un único registro de registro y se genera un nuevo registro de log-cabeza y escritos (una extensión al protocolo NFS permite la aparición de una *cerca* el funcionamiento de la aplicación que se le notifique al servidor Ivy).

Puesto que no es un servidor de Ivy separada en cada nodo y cada uno almacena de forma autónoma sus actualizaciones en un registro separado y sin coordinación con los otros servidores, la serialización de cambios debe hacerse en el momento cuando los registros se leen con el fin de construir el contenido de los archivos. Los vectores versión escrita en los registros de registro se pueden utilizar para ordenar mayoría de las actualizaciones, pero

actualizaciones en conflicto son posibles y deben ser resueltos por métodos automáticos o manuales específicos de la aplicación, como se hace en Coda (véase la Sección 18.4.3).

integridad de los datos se logra mediante una combinación de los mecanismos que ya hemos mencionado: los registros se inmutable y su dirección es un control seguro de su contenido; log-cabezas se verifican mediante la comprobación de una firma de clave pública de su contenido. Pero el modelo de confianza permite la posibilidad de que un participante malicioso puede obtener acceso a un sistema de archivos. Por ejemplo, podrían eliminar los archivos que poseen maliciosamente. Cuando esto se detecta, el participante malicioso se expulsa de la vista; su registro ya no se utiliza para calcular el contenido del sistema de archivos y los archivos que se han borrado son una vez más visible en la nueva vista.

Los autores utilizaron un Ivy Andrew indice de referencia [Howard *et al.* 1988] para comparar el rendimiento de hiedra con un servidor estándar NFS en entornos de red de área local y amplia. Consideraron (a) Ivy usando servidores locales DHash en comparación con un solo servidor y NFS local (b) el uso de servidores Ivy DHash ubicados en varios sitios de Internet remotos en comparación con un único servidor NFS remoto. También consideraron las características de rendimiento en función del número de participantes en una vista, el número de participantes que escriben al mismo tiempo y el número de servidores DHash utilizados para almacenar los registros.

Encontraron que los tiempos de ejecución de la hiedra se encontraban dentro de un factor de dos de NFS tiempos de ejecución para la mayoría de las pruebas en el punto de referencia y en un factor de tres para todos ellos. Los tiempos de ejecución para el amplio despliegue de red de área superaron los del caso local por un factor de 10 o más, pero se obtuvieron proporciones similares para un servidor de NFS remoto. Los detalles completos de la evaluación del desempeño se pueden encontrar en el documento de la hiedra [Muthitacharoen *et al.* 2002]. Cabe señalar, sin embargo, que NFS no fue diseñado para el uso de área amplia; otros sistemas basados en servidores más recientemente desarrollados, como XFS Sistema de Archivo y Andrew [Anderson *et al.* 1996] ofrecen un mayor rendimiento en las implementaciones de área amplia y podría haber hecho mejores bases para la comparación. La principal contribución de la hiedra se encuentra en su nuevo enfoque para la gestión de la seguridad y la integridad en un entorno de confianza parcial - una característica inevitable de grandes sistemas distribuidos que abarcan muchas organizaciones y jurisdicciones.

10.7 Resumen

arquitecturas peer-to-peer se muestran en primer lugar para apoyar el intercambio de datos muy grande escala con el uso de toda la Internet de Napster y sus descendientes para compartir música digital. El hecho de que gran parte de su uso en conflicto con las leyes de derechos de autor no disminuye su importancia técnica, aunque también tenían inconvenientes técnicos que restringían su despliegue en aplicaciones en las que las garantías de integridad de los datos y la disponibilidad no eran importantes.

La investigación posterior dio como resultado el desarrollo de plataformas middleware peer-to-peer que entregan peticiones a objetos de datos dondequiera que se encuentren en Internet. En enfoques estructurados, los objetos se tratan usando GUID, que son puros nombres que no contengan información de direccionamiento IP. Los objetos se colocan en los nodos de acuerdo con alguna función de mapeo que es específica para cada sistema de middleware. La entrega se realiza mediante una superposición de enrutamiento en el middleware que mantiene las tablas de enrutamiento y solicitudes hacia delante

a lo largo de una ruta determinada por el cálculo de la distancia de acuerdo con la función de mapeo elegido. En los enfoques no estructurados, los nodos forman a sí mismos en una red ad hoc y luego se propagan a través de búsquedas vecinos para encontrar los recursos apropiados. Varias estrategias se han desarrollado para mejorar el desempeño de esta función de búsqueda e incrementar la escalabilidad global del sistema.

Las plataformas de middleware añaden garantías de integridad basados en el uso de una función hash seguro para generar los GUID y garantías disponibilidad basados en la replicación de objetos en varios nodos y en algoritmos de encaminamiento tolerantes a fallos.

Las plataformas se han desplegado en varias aplicaciones piloto a gran escala, refinado y evaluado. Los recientes resultados de la evaluación indican que la tecnología está lista para su despliegue en aplicaciones que implican un gran número de usuarios que comparten muchos objetos de datos. Los beneficios de los sistemas peer-to-peer incluyen:

- su capacidad para explotar los recursos no utilizados (almacenamiento, procesamiento) en los equipos host;
- su escalabilidad para soportar un gran número de clientes y anfitriones con un excelente equilibrio de las cargas sobre los enlaces de red y recursos informáticos anfitrión;
- las propiedades de auto-organización de las plataformas middleware que dan lugar a los gastos de apoyo, que son independientes del número de clientes y anfitriones desplegados. Debilidades y los sujetos de la investigación actual incluyen:
- su uso para el almacenamiento de datos mutable es relativamente costoso en comparación con un servicio de confianza, centralizada;
- la base prometedora que proporcionan para el cliente y el anonimato anfitrión todavía no ha dado lugar a fuertes garantías de anonimato.

CEREMONIAS

10.1 aplicaciones de intercambio de archivos como Napster primeros fueron restringidos en su capacidad de ampliación por la necesitará mantener un índice central de los recursos y los anfitriones que las tienen. ¿Qué otras soluciones al problema de indexación puede identificar?

páginas 428- 430, 435, Sección 18.4

10.2 El problema de mantener los índices de los recursos disponibles depende de la aplicación.

Tenga en cuenta la idoneidad de cada una de sus respuestas al ejercicio 10.1 para:

i) el intercambio de música y los medios de comunicación de archivos;

ii) el almacenamiento a largo plazo del material archivado como revista o periódico de contenido;

iii) de almacenamiento de la red de archivos de lectura-escritura de propósito general.

10.3 ¿Cuáles son las principales garantías de que los usuarios esperan servidores convencionales (por ejemplo, servidores web o servidores de archivos) para ofrecer? *sección 1.5.5*

10.4 Las garantías ofrecidas por los servidores convencionales pueden ser violados como resultado de:

i) daño físico a la acogida;

ii) Los errores o inconsistencias introducidas por los administradores del sistema o sus gestores;

iii) ataques con éxito en la seguridad del software del sistema;

iv) errores de hardware o software.

Da dos ejemplos de posibles incidentes para cada tipo de violación. ¿Cuál de ellos podría ser descrito como un abuso de confianza o un acto criminal? ¿Serían abusos de confianza si se produjeron en un ordenador personal que estaba contribuyendo algunos recursos para un servicio Igualitaria? ¿Por qué es esto importante para los sistemas peer-to-peer?

sección 11.1.1

10.5 sistemas peer-to-peer suelen depender de *no es de confianza* y *volátil* sistemas informáticos para la mayor parte de sus recursos. La confianza es un fenómeno social con consecuencias técnicas. La volatilidad (disponibilidad es decir, impredecible) también es a menudo debido a las acciones humanas. Elaborar sus respuestas al ejercicio 10,4 por discutir las posibles formas en que cada uno de ellos es probable que difieran de acuerdo con los siguientes atributos de los ordenadores utilizados:

i) la propiedad;

ii) la ubicación geográfica;

iii) la conectividad de red;

iv) el país o jurisdicción.

¿Qué sugiere esto acerca de las políticas para la colocación de objetos de datos en un servicio de almacenamiento peer-to-peer?

10.6 Evaluar la disponibilidad y confiabilidad de los ordenadores personales en su ambiente. Se debe estimar:

el tiempo de actividad: ¿Cuántas horas por día es el operativo del ordenador y conectado a Internet?

la coherencia del programa: Es el software gestionado por un técnico competente?

Seguridad: ¿Está el equipo totalmente protegido contra la manipulación por parte de sus usuarios u otras personas? Sobre la base de su evaluación, analizar la viabilidad de la ejecución de un servicio datasharing en el conjunto de equipos que haya evaluado y resumen de los problemas que deben ser abordados en un servicio de intercambio de datos peer-to-peer.

páginas 431-432

10.7 Explicar por qué usar el control seguro de un objeto de identificar y direccionar mensajes a ella es a prueba de manipulaciones. ¿Qué propiedades se requieren de la función hash? ¿Cómo se puede mantener la integridad, aunque una proporción sustancial de nodos pares se subvierten?

páginas 426, 453, Sección 11.4.3

10.8 A menudo se argumenta que los sistemas peer-to-peer pueden ofrecer anonimato para:

i) clientes que acceden a los recursos;

ii) los anfitriones que proporcionan acceso a los recursos.

Discutir cada una de estas proposiciones. Sugerir una manera en la que la resistencia a los ataques contra el anonimato podría mejorarse. *página 429*

10.9 Los algoritmos de enrutamiento eligen un salto siguiente de acuerdo con una estimación de la distancia de alguna espacio de direccionamiento. Pasteles y la tapicería tanto utilizar espacios de direcciones lineal circulares en la que una función basada en la diferencia numérica aproximada entre GUID determina

su separación. Kademlia utiliza el XOR de los GUID. ¿Cómo funciona esto ayuda en el mantenimiento de las tablas de enrutamiento? ¿La operación XOR proporcionan propiedades adecuadas para una distancia métrica?

435 páginas, [Maymounkov y Mazieres 2002]

10.10 Cuando el servicio de almacenamiento en caché web peer-to-peer ardilla se evaluó mediante simulación, 4,11

Se requieren saltos en promedio para encaminar una petición de una entrada de caché cuando se simula el tráfico de Redmond, mientras que sólo el 1.8 fueron requeridos para el tráfico de Cambridge. Explicar esto y demostrar que es compatible con el rendimiento teórico reclamó para pasteles.

páginas 436, 450

10.11 En los sistemas no estructurados peer-to-peer, mejoras significativas en los resultados de búsqueda pueden ser

proporcionado por la adopción de determinadas estrategias de búsqueda. Comparar y contrastar las estrategias de búsqueda de anillo expandido y caminata aleatoria, destacando que cada enfoque es probable que sea eficaz.

página 446

Esta página se ha dejado intencionadamente en blanco

1

SEGURIDAD

11.1 Introducción

11.2 Visión general de las técnicas de seguridad

11.3 Los algoritmos criptográficos

11.4 Las firmas digitales

11.5 pragmática criptografía

11.6 Estudios de casos: Needham-Schroeder, Kerberos, TLS, 802.11 WiFi

11.7 Resumen

Existe una necesidad generalizada de medidas para garantizar la privacidad, integridad y disponibilidad de recursos en los sistemas distribuidos. ataques a la seguridad toman las formas de espionaje, enmascaramiento, la manipulación y la denegación de servicio. Los diseñadores de sistemas distribuidos seguros deben hacer frente a interfaces de servicios expuestos y redes no seguras en un entorno donde los atacantes puedan tener conocimiento de los algoritmos utilizados y desplegar los recursos de computación.

La criptografía es la base para la autenticación de mensajes, así como su carácter secreto e integridad; Se requieren protocolos de seguridad diseñados cuidadosamente para explotarla. La selección de algoritmos criptográficos y la gestión de claves son fundamentales para la eficacia, el rendimiento y la facilidad de uso de los mecanismos de seguridad. criptografía de clave pública hace que sea fácil de distribuir claves criptográficas pero su rendimiento es inadecuado para el cifrado de datos en masa. La criptografía de clave secreta es más adecuado para las tareas de cifrado a granel. protocolos híbridos como Transport Layer Security (TLS) establecen un canal seguro usando criptografía de clave pública y luego lo utilizan para intercambiar claves secretas para su uso en los intercambios de datos subsiguientes.

La información digital puede ser firmado, la producción de certificados digitales. Certificados permiten a la confianza que se establece entre los usuarios y las organizaciones.

El capítulo concluye con estudios de caso sobre los enfoques para el diseño del sistema de seguridad y los mecanismos de seguridad implementados en Kerberos, TLS / SSL y 802.11 Wi-Fi.

11.1 Introducción

En la Sección 2.4.3 se introdujo un modelo simple para el examen de los requisitos de seguridad en los sistemas distribuidos. Llegamos a la conclusión de que la necesidad de mecanismos de seguridad en los sistemas distribuidos surge del deseo de compartir recursos. (Recursos que no se comparten por lo general se pueden proteger aislándolas de acceso externo.) Si consideramos los recursos compartidos como objetos, a continuación, el requisito es evitar que los procesos que encapsulan los objetos compartidos y los canales de comunicación que se utilizan para interactuar con ellos contra todas las formas concebibles de ataque. El modelo presentado en la Sección 2.4.3 proporciona un buen punto de partida para la identificación de requisitos de seguridad. Se puede resumir de la siguiente manera:

- Procesos encapsulan los recursos (tanto objetos de nivel de lenguaje de programación y recursos definidos por el sistema) y permiten a los clientes acceder a ellos a través de las interfaces. Los directores (usuarios u otros procesos) están autorizados para operar en los recursos. Los recursos deben ser protegidos contra el acceso no autorizado (Figura 2.17).
- Procesos interactúan a través de una red que es compartido por muchos usuarios. Enemigos (los atacantes) pueden tener acceso a la red. Pueden copiar o intentar leer cualquier mensaje transmitido a través de la red y pueden inyectar mensajes arbitrarios, dirigirse a cualquier destino y que pretende provenir de cualquier fuente, en la red (Figura 2.18).

La necesidad de proteger la integridad y privacidad de la información y otros recursos que pertenecen a los individuos y las organizaciones es un fenómeno generalizado, tanto en el mundo físico y el digital. Surge del deseo de compartir recursos. En el mundo físico, las organizaciones adoptan *políticas de seguridad* que dispondrá la distribución de los recursos dentro de los límites especificados. Por ejemplo, una empresa puede permitir la entrada a los edificios sólo a sus empleados y visitantes acreditados. Una política de seguridad para documentos puede especificar grupos de empleados que tienen acceso a clases de documentos, o puede ser definida para documentos y usuarios individuales.

Las políticas de seguridad se aplican con la ayuda de *mecanismos de seguridad.* Por ejemplo, el acceso a un edificio puede ser controlado por un empleado de recepción, que emite tarjetas de identificación para visitantes acreditados, y aplicado por un guardia de seguridad o cerraduras electrónicas. El acceso a los documentos en papel se controla normalmente mediante la ocultación y la distribución restringida. En el mundo electrónico, la distinción entre las políticas y los mecanismos de seguridad es igualmente importante; sin ella, sería difícil determinar si un sistema particular era seguro. Las políticas de seguridad son independientes de la tecnología utilizada, así como la provisión de una cerradura en una puerta no garantiza la seguridad de un edificio a menos que exista una política para su uso (por ejemplo, que la puerta se bloqueará cada vez nadie está vigilando la Entrada). Los mecanismos de seguridad que describimos aquí en sí mismas no garantizan la seguridad de un sistema. En la Sección 11.1.2,

La provisión de mecanismos para la protección de datos y otros recursos en los sistemas distribuidos al tiempo que permite la interacción entre los equipos que están permitidos por las políticas de seguridad es la preocupación de este capítulo. Los mecanismos que describiremos están diseñados para cumplir las políticas de seguridad contra los ataques más decididos.

El papel de la criptografía • La criptografía digital proporciona la base para la mayoría de los mecanismos de seguridad informática, pero es importante tener en cuenta que la seguridad informática y criptografía son temas distintos. La criptografía es el arte de la codificación de la información en un formato que sólo los destinatarios pueden descifrar. La criptografía también se puede emplear para proporcionar la prueba de la autenticidad de la información, de una manera análoga a la utilización de firmas en las transacciones convencionales.

La criptografía tiene una historia larga y fascinante. La necesidad militar para la comunicación segura y la correspondiente necesidad de un enemigo para interceptar y descifrar que condujo a la inversión de mucho esfuerzo intelectual por parte de algunos de los mejores cerebros matemáticos de su tiempo. Los lectores interesados en la exploración de esta historia encontrarán absorción de lectura en libros sobre el tema por David Kahn [Kahn 1967, 1983, 1991] y Simon Singh [Singh 1999]. Whitfield Diffie, uno de los inventores de la criptografía de clave pública, ha escrito con conocimiento de primera mano sobre la historia y la política de la criptografía reciente [Diffie

1988, Diffie y Landau 1998].

Es sólo en los últimos tiempos que la criptografía ha surgido de las envolturas previamente puestas en ella por las instituciones políticas y militares que utilizan para controlar su desarrollo y uso. Ahora es el sujeto de la investigación abierta por una comunidad grande y activa, con los resultados que se presentan en muchos libros, revistas y conferencias. La publicación del libro de Schneier *Applied Cryptography [* Schneier 1996] fue un hito en la apertura de los conocimientos en el campo. Fue el primer libro para publicar muchos algoritmos importantes con el código fuente - un paso valiente, porque cuando la primera edición apareció en 1994 la situación jurídica de dicha publicación no estaba clara. El libro de Schneier sigue siendo la referencia definitiva en la mayoría de los aspectos de la criptografía moderna. Un libro co-escrito por Schneier [Ferguson y Schneier 2003] más reciente ofrece una excelente introducción a la criptografía de ordenador y una visión discursiva de prácticamente todos los algoritmos y técnicas importantes en el uso actual, incluyendo varios publicados desde libro anterior de Schneier. Además, Menezes *et al. [* 1997] proporcionan un buen manual práctico con una sólida base teórica y la Biblioteca seguridad de la red [www.secinf.net] Es una excelente fuente en línea de conocimientos y experiencias prácticas.

Ross Anderson *Ingeniería de Seguridad [* Anderson 2008] También es excepcional. Está repleto de lecciones prácticas sobre el diseño de sistemas seguros, extraídos de situaciones del mundo real y los fallos de seguridad del sistema.

La nueva apertura es en gran parte resultado del enorme crecimiento del interés en las aplicaciones no militares de la criptografía y los requisitos de seguridad de los sistemas informáticos distribuidos, lo que ha dado lugar a la existencia por primera vez de una comunidad auto-sostenible de investigadores criptográficos fuera del dominio militar .

Irónicamente, la apertura de la criptografía para acceso y uso público ha dado como resultado una gran mejora en las técnicas criptográficas, tanto en su fuerza para resistir los ataques de los enemigos y en la comodidad con la que se pueden implementar. criptografía de clave pública es uno de los frutos de esta apertura. Como otro ejemplo, observamos que el algoritmo de cifrado DES que fue adoptado y utilizado por las agencias militares y gubernamentales de Estados Unidos fue inicialmente un secreto militar. Su eventual publicación y exitosos esfuerzos para reprimir el resultado fue el desarrollo de algoritmos de cifrado de clave secreta mucho más fuertes.

Otra utilidad spin-off ha sido el desarrollo de una terminología común y el enfoque. Un ejemplo de esto último es la adopción de un conjunto de nombres familiares para los protagonistas (principales) que participan en las operaciones que han de ser asegurado. El uso de

Figura 11.1 nombres familiares para los protagonistas de los protocolos de seguridad

Alicia	En primer participante
Chelín	En segundo participante
Villancico	Participante en los protocolos de tres y cuatro bandas
David	Participante en los protocolos a cuatro bandas
Víspera	fisgón
Mallory	atacante malicioso
Sara	Un servidor

nombres familiares para los directores y los atacantes ayuda a aclarar y llevar a las descripciones de la vida de los protocolos de seguridad y los posibles ataques contra ellos, que ayuda en la identificación de sus debilidades. Los nombres que se muestran en la figura 11.1 se utilizan ampliamente en la seguridad literatura y los usamos aquí libremente. No hemos sido capaces de descubrir sus orígenes; la primera aparición de la que somos conscientes es en el documento original de la criptografía de clave pública RSA [Rivest *et al.* 1978]. Un comentario divertido sobre su uso se puede encontrar en Gordon [1984].

11.1.1 Las amenazas y ataques

Algunas amenazas son obvias - por ejemplo, en la mayoría de los tipos de red local es fácil de construir y ejecutar un programa en un ordenador conectado que obtenga copias de los mensajes transmitidos entre otros equipos. Otras amenazas son más sutiles - si los clientes no pueden autenticar servidores, un programa puede instalarse en lugar de un servidor de archivos auténtico y de ese modo obtener copias de la información confidencial que los clientes sin darse cuenta de que envían para su almacenamiento.

Además del peligro de pérdida o daño de información o recursos a través de violaciónes directos, reclamaciones fraudulentas pueden hacerse contra el propietario de un sistema que no es demostrablemente seguro. Para evitar este tipo de reclamaciones, el propietario debe estar en condiciones de refutar la afirmación al mostrar que el sistema es seguro contra tales violaciónes o mediante la producción de un registro de todas las transacciones para el período en cuestión. Un ejemplo común es el problema 'retirada fantasma' en cajeros automáticos (cajeros). La mejor respuesta que un banco puede suministrar a tal afirmación es proporcionar un registro de la transacción que está firmado digitalmente por el titular de la cuenta de una manera que no puede ser forjada por un tercero.

El objetivo principal de la seguridad es restringir el acceso a la información y recursos para sólo aquellos directores que están autorizados a tener acceso. Las amenazas de seguridad se dividen en tres grandes categorías:

Fuga: Se refiere a la adquisición de información por los receptores no autorizados.

La manipulación: Se refiere a la alteración no autorizada de información.

Vandalismo: Se refiere a la interferencia con el funcionamiento correcto de un sistema sin ganancia para el autor.

Los ataques a sistemas distribuidos dependen de la obtención de acceso a los canales de comunicación existentes o el establecimiento de nuevos canales que se hacen pasar por conexiones autorizadas. (Utilizamos el término *canal* para referirse a cualquier mecanismo de comunicación entre procesos) Métodos de ataque pueden clasificarse adicionalmente de acuerdo con la forma en que se emplea mal un canal.:

escuchas ilegales: La obtención de copias de los mensajes sin autorización.

enmascaramiento: Enviar o recibir mensajes usando la identidad de otra principal sin su autoridad.

Mensaje manipulación: Interceptar mensajes y alterar su contenido antes de pasarlos al destinatario previsto. los *man-in-the-middle* es una forma de mensaje de manipulación en la que un atacante intercepta el primer mensaje en un intercambio de claves de cifrado para establecer un canal seguro. Los sustitutos atacante claves que les permitan descifrar los mensajes posteriores antes reencrypting en las teclas correctas y pasarlos comprometidas.

Reproducción: El almacenamiento de los mensajes interceptados y enviarlos en una fecha posterior. Este ataque puede ser eficaz incluso con mensajes autenticados y encriptados.

Negación de servicio: Las inundaciones de un canal u otro recurso con mensajes con el fin de negar el acceso a los demás.

Estos son los peligros en la teoría, pero ¿cómo se llevan a cabo ataques en la práctica? ataques con éxito dependen del descubrimiento de las lagunas en la seguridad de los sistemas. Por desgracia, estos son muy comunes en los sistemas de hoy en día, y no son necesariamente particularmente oscuro. Cheswick y Bellovin [1994] identifican 42 debilidades que consideran que presentan riesgos graves en los sistemas de Internet ampliamente utilizados y componentes. Se extienden de adivinar la contraseña a los ataques contra los programas que realizan el Protocolo de tiempo de red o manejan la transmisión de correo. Algunos de estos han dado lugar a ataques con éxito y bien publicitadas [Stoll 1989, Spafford 1989], y muchos de ellos han sido explotados con fines maliciosos o criminales.

Cuando el Internet y los sistemas que están conectados a él fueron diseñados, la seguridad no era una prioridad. Los diseñadores probablemente tenían la menor idea de la escala a la que crecería Internet, y el diseño básico de los sistemas como UNIX es anterior a la llegada de las redes informáticas. Como veremos más adelante, la incorporación de medidas de seguridad tiene que ser cuidadosamente pensado en la etapa de diseño básico y el material de este capítulo tiene por objeto proporcionar la base para este tipo de pensamiento.

Nos centramos en las amenazas a los sistemas distribuidos que surgen de la exposición de sus canales de comunicación y sus interfaces. Para muchos sistemas, estas son las únicas amenazas que deben tenerse en cuenta (distintos de los que surgen de errores humanos - los mecanismos de seguridad no pueden proteger contra una contraseña mal elegido o uno que sea divulgada por descuido). Sin embargo, para sistemas que incluyen programas y sistemas cuya seguridad es particularmente sensible a la fuga de información, hay más amenazas móviles.

Las amenazas de código móvil • Varios lenguajes de programación desarrollados recientemente se han diseñado para permitir que los programas que se cargan en un proceso de un servidor remoto y luego ejecutada localmente. En ese caso, las interfaces internas y objetos dentro de un proceso de ejecución pueden estar expuestos al ataque de código móvil.

Java es el lenguaje más utilizado de este tipo, y sus diseñadores prestó especial atención al diseño y construcción del lenguaje y los mecanismos de cobro a distancia, en un esfuerzo para restringir la exposición (la *salvadera* modelo de protección contra código móvil).

La máquina virtual de Java (JVM) está diseñado con código móvil a la vista. Da a cada aplicación su propio entorno en el que se ejecutará. Cada entorno tiene un controlador de seguridad que determina qué están disponibles para la aplicación de recursos. Por ejemplo, el gerente de seguridad podría dejar un reproductor de archivos de lectura y escritura de aplicaciones, o darle acceso limitado a las conexiones de red. Una vez que un administrador de seguridad se ha establecido, no puede ser reemplazado. Cuando un usuario ejecuta un programa como un navegador que descarga el código móvil para ejecutar de forma local en su nombre, no tienen muy buenas razones para confiar en el código que se comporten de una manera responsable. De hecho, existe el peligro de descargar y ejecutar código malicioso que elimina archivos o accede a la información privada. Para proteger a los usuarios contra el código no confiable, la mayoría de los navegadores especifican que los applets no pueden acceder a los archivos locales, impresoras o conexiones de red. Algunas aplicaciones de código móvil son capaces de asumir diferentes niveles de confianza en código descargado. En este caso, los administradores de seguridad están configuradas para proporcionar un mayor acceso a los recursos locales.

La JVM toma dos medidas adicionales para proteger el medio ambiente local:

1. Las clases descargados se almacenan por separado de las clases locales, lo que les impide la sustitución de las clases locales con versiones espurias.
2. Los códigos de bytes se comprueba su validez. Válido código de bytes de Java se compone de Java instrucciones de máquina virtual a partir de un conjunto especificado. Las instrucciones también se comprueban para asegurar que no se producen ciertos errores cuando se ejecuta el programa, como el acceso a las direcciones de memoria ilegales.

La seguridad de Java ha sido objeto de mucha investigación posterior, en el curso de la cual se hizo evidente que los mecanismos adoptados originales no eran libres de lagunas [McGraw y Felden 1999]. Las lagunas identificadas fueron corregidas y el sistema de protección de Java fueron refinados para permitir que el código móvil para acceder a los recursos locales cuando sea autorizado para hacerlo [java.sun.com V].

A pesar de la inclusión de mecanismos del tipo de comprobación y el código de validación, los mecanismos de seguridad incorporados en los sistemas de códigos móviles todavía no producen el mismo nivel de confianza en su eficacia como los utilizados para proteger los canales de comunicación y las interfaces. Esto se debe a la construcción de un entorno para la ejecución de programas ofrece muchas oportunidades para el error, y es difícil estar seguro de que todos han sido evitado. Volpano y Smith [1999] han señalado que un enfoque alternativo, basado en pruebas de que el comportamiento del código móvil es sonido, podría ofrecer una mejor solución.

fuga de información • Si se observa la transmisión de un mensaje entre dos procesos, alguna información puede ser obtenida de su mera existencia - por ejemplo, una inundación de mensajes a un distribuidor en una acción en particular podría indicar un alto nivel de la negociación de esa población. Hay muchas formas más sutiles de la fuga de información, algunos maliciosos y otros derivados de error involuntario. El potencial de fugas surge siempre que se pueden observar los resultados de un cálculo. Se trabajó en la prevención de este tipo de amenaza a la seguridad en los años 1970 [Denning y Denning 1977]. El enfoque adoptado es asignar niveles de seguridad a la información y canales y analizar el flujo de información

en canales con el objetivo de garantizar que la información de alto nivel no puede fluir dentro de los canales de nivel inferior. Un método para el control seguro de los flujos de información fue descrito por primera vez por Bell y LaPadula [1975]. La extensión de este enfoque a los sistemas distribuidos con desconfianza mutua entre los componentes es el sujeto de la investigación reciente [Myers y Liskov 1997].

11.1.2 transacciones electrónicas Asegurando

Muchos usos de Internet en la industria, el comercio y en otros lugares implican transacciones que dependen de manera crucial de la seguridad. Por ejemplo:

Email: Aunque los sistemas de correo electrónico no incluyeron originalmente apoyo a la seguridad, hay muchas aplicaciones de correo electrónico en el que el contenido de los mensajes deben ser mantenidas en secreto (por ejemplo, cuando se envía un número de tarjeta de crédito) o el contenido y el remitente de un mensaje deben ser autenticados (por ejemplo, cuando la presentación de una subasta por correo electrónico). seguridad criptográfica basada en las técnicas descritas en este capítulo se incluye ahora en muchos clientes de correo.

Compra de bienes y servicios: Tales transacciones son ahora comunes. Los compradores seleccionan los bienes y pagar por ellos a través de la Web y los bienes son entregados a través de un mecanismo de suministro adecuado. Software y otros productos digitales (tales como grabaciones y videos) pueden ser entregados mediante la descarga. bienes tangibles, tales como libros, CDs y casi cualquier otro tipo de producto, también son vendidos por los vendedores de Internet; estos se suministran a través de un servicio de entrega.

Las transacciones bancarias: bancos electrónicos ofrecen ahora los usuarios prácticamente todas las instalaciones proporcionadas por los bancos convencionales. Se pueden consultar sus saldos y estados, transferir dinero entre cuentas, establecer pagos automáticos regulares y así sucesivamente.

Micro-transacciones: Internet se presta a la entrega de pequeñas cantidades de información y otros servicios a muchos clientes. La mayoría de las páginas web actualmente se pueden ver de forma gratuita, pero el desarrollo de la Web como un medio de publicación de alta calidad depende sin duda de la capacidad de los proveedores de información para obtener el pago de los consumidores de la información. Voz y videoconferencias a través de Internet es actualmente también es libre, pero se carga para cuando también está implicada una red telefónica. El precio de tales servicios puede equivale a sólo una fracción de un centavo, y los gastos generales de pago debe ser correspondientemente baja. En general, los esquemas basados en la participación de un servidor bancario o tarjeta de crédito por cada transacción no pueden lograr esto. Las transacciones de este tipo se pueden realizar de forma segura sólo cuando están protegidos por las políticas y los mecanismos de seguridad adecuados. Un comprador debe estar protegida contra la divulgación de códigos de crédito (números de tarjetas) durante la transmisión y en contra de los vendedores fraudulentos que obtengan el pago sin la intención de suministrar las mercancías. Los vendedores deben obtener el pago antes de entregar la mercancía, y para productos descargables deben asegurarse de que los clientes sólo pagan obtienen los datos en una forma utilizable. La protección requerida debe lograrse a un costo que es razonable en comparación con el valor de la transacción.

políticas de seguridad razonables para los vendedores y compradores de Internet conducen a los siguientes requisitos para obtener la compras por Internet:

1. autenticar el vendedor al comprador, por lo que el comprador puede estar seguro de que están en contacto con un servidor operado por el proveedor con el que tenían la intención de tratar.

2. Mantenga el número de tarjeta de crédito del comprador y otros detalles de pago de caer en manos de terceros y asegurar que se transmiten sin modificaciones del comprador al vendedor.

3. Si las mercancías están en una forma adecuada para la descarga, asegúrese de que su contenido es entregado al comprador sin alteración y sin dar a conocer a terceros. La identidad del comprador no se requiere normalmente por el vendedor (salvo con el fin de entregar la mercancía, si no se descargan). El proveedor deberá verificar que el comprador tiene los fondos suficientes para pagar la compra, pero esto generalmente se hace por exigir el pago del banco del comprador antes de la entrega de la mercancía.

Las necesidades de seguridad de las transacciones bancarias utilizando una red abierta son similares a los de las operaciones de compra, con el comprador como el titular de la cuenta y el banco como el proveedor, pero no hay un cuarto requisito, así:

4. autenticar la identidad del titular de la cuenta al banco antes de darles acceso a su cuenta.

Tenga en cuenta que en esta situación, es importante para el banco para asegurarse de que el titular de la cuenta no se puede negar que participaron en una transacción. *No repudio* es el nombre dado a este requisito.

Además de los requisitos anteriores, los cuales son dictados por las políticas de seguridad, hay algunos requisitos del sistema. Estos surgen de la gran escala de Internet, lo que hace que sea poco práctica para requerir a los compradores a entrar en relaciones especiales con los proveedores (mediante el registro de claves de cifrado para su uso posterior, etc.). Debería ser posible para un comprador para completar una transacción segura con un vendedor, incluso si no ha habido ningún contacto previo entre el comprador y el vendedor y sin la participación de un tercero. Técnicas tales como el uso de 'cookies' - los registros de transacciones anteriores almacenados en la máquina cliente del usuario - tienen debilidades obvias de seguridad; escritorio y móviles organizadora se encuentran a menudo en entornos físicos inseguros.

Debido a la importancia de la seguridad para el comercio por Internet y el rápido crecimiento en el comercio por Internet, hemos elegido para ilustrar el uso de técnicas de seguridad criptográfica mediante la descripción en la Sección 11.6 del *de facto* protocolo de seguridad estándar que se utiliza en la mayor parte del comercio electrónico - Seguridad de la capa de transporte (TLS). Una descripción de Millicent, un protocolo diseñado específicamente para micro-transacciones, se puede encontrar en
www.cdk5.net/security .

El comercio por Internet es una importante aplicación de las técnicas de seguridad, pero ciertamente no es el único. Es necesario siempre que los ordenadores son utilizados por individuos u organizaciones para almacenar y comunicar información importante. El uso del correo electrónico cifrado para una comunicación privada entre individuos es un ejemplo de ello que ha sido objeto de considerable debate político. Nos referimos a este debate en la Sección 11.5.2.

11.1.3 El diseño de sistemas seguros

zancadas inmensas se han hecho en los últimos años en el desarrollo de técnicas criptográficas y su aplicación, sin embargo, el diseño de los sistemas de seguridad sigue siendo una tarea inherentemente difícil. En el corazón de este dilema es el hecho de que el objetivo del diseñador es excluir *todas* posibles ataques y las lagunas. La situación es análoga a la del programador cuyo objetivo es excluir todos los errores de su programa. En ningún caso existe un método concreto para asegurar este objetivo se cumple durante el diseño. Se diseña con los mejores estándares disponibles y se aplica el análisis informal y cheques. Una vez completado un diseño, validación formal es una opción. El trabajo sobre la validación formal de los protocolos de seguridad ha producido algunos resultados importantes [Lampson *et al.* 1992, Schneider 1996, Abadi y Gordon 1999]. Una descripción de uno de los primeros pasos en esta dirección, la lógica BAN de autenticación [Burrows *et al.* 1990], y su aplicación se puede encontrar en www.cdk5.net/security . La seguridad es acerca de cómo evitar los desastres y reducir al mínimo los accidentes. En el diseño de la seguridad es necesario asumir lo peor. El recuadro de la página 472 muestra un conjunto de supuestos útiles y directrices de diseño. Estos supuestos subyacen a la idea detrás de las técnicas que describimos en este capítulo.

Para demostrar la validez de los mecanismos de seguridad empleadas en un sistema, los diseñadores del sistema deben construir primero una lista de amenazas - métodos por los cuales las políticas de seguridad podrían ser violados - y demostrar que cada uno de ellos se evita mediante los mecanismos empleados. Esta demostración puede tomar la forma de una discusión informal o, mejor, una prueba lógica.

No hay una lista de las amenazas es probable que sea exhaustiva, por lo que los métodos de auditoría también deben ser utilizados en aplicaciones sensibles a la seguridad para detectar violaciónes. Estos son fáciles de implementar si un registro seguro de las acciones del sistema sensibles a la seguridad siempre se graba con los detalles de los usuarios que realizan las acciones y su autoridad.

Un registro de seguridad contendrá una secuencia de registros con sellos de tiempo de las acciones de los usuarios. Como mínimo los registros incluirán la identidad de un principal, la operación realizada (por ejemplo, borrar archivos, actualizar el registro contable), la identidad del objeto operado y una marca de tiempo. Cuando se sospeche la violaciónes particulares, los registros pueden ampliarse para incluir la utilización de recursos físicos (ancho de banda de red, periféricos), o el proceso de registro se pueden dirigir a las operaciones sobre objetos particulares. El análisis posterior puede ser estadística o basados en la búsqueda. Incluso cuando se sospecha que hay violaciónes, las estadísticas se pueden comparar con el tiempo para ayudar a descubrir las tendencias o eventos inusuales.

El diseño de sistemas seguros es un ejercicio de equilibrio de costos contra las amenazas. La gama de técnicas que se pueden implementar para la protección de los procesos y asegurar la comunicación entre procesos son suficientes para soportar casi cualquier ataque fuerte, pero su uso incurre en gastos e inconvenientes:

- Un coste (en términos de esfuerzo computacional y uso de la red) se incurre para su uso. Los costos deben ser equilibrados contra las amenazas.
- las medidas de seguridad especificadas de manera inapropiada pueden excluir a los usuarios legítimos de la realización de las acciones necesarias.

Tales compensaciones son difíciles de identificar sin comprometer la seguridad y puede parecer estar en conflicto con el consejo en el primer párrafo de este apartado, pero la fuerza de las técnicas de seguridad requeridas pueden ser cuantificados y las técnicas se pueden basar en el seleccionado

el costo estimado de los ataques. Las técnicas de costo relativamente bajo empleados en el protocolo Millicent, descritos en www.cdk5.net/security proporcionar un ejemplo.

A modo de ejemplo de las dificultades y contratiempos que pueden surgir en el diseño de sistemas seguros, se revisan las dificultades que surgieron con el diseño de seguridad incorporados en los correspondientes estándar de red inalámbrica IEEE 802.11 en la Sección 11.6.4.

11.2 Visión general de las técnicas de seguridad

El propósito de esta sección es introducir al lector en algunas de las técnicas y mecanismos más importantes para asegurar los sistemas y aplicaciones distribuidas. A continuación se describe de manera informal, reservando descripciones más rigurosas para las secciones 11.3 y
11.4. Usamos los nombres familiares para los principales introducidas en la figura 11.1 y las notaciones para elementos cifrados y firmados que se muestran en la figura 11.2.

hipótesis del peor caso y pautas de diseño

Las interfaces se exponen: Los sistemas distribuidos se componen de procesos que ofrecen servicios o compartir información. Sus interfaces de comunicación son necesariamente abierta (para permitir nuevos clientes acceder a ellos) - un atacante puede enviar un mensaje a cualquier interfaz.

Las redes son inseguros: Por ejemplo, fuentes de mensajes pueden falsificarse - los mensajes se pueden hacer para parecer como si vinieron de Alice, cuando en realidad eran enviados por Mallory. direcciones de host se pueden 'falsificados' - Mallory se puede conectar a la red con la misma dirección que Alice y recibir copias de los mensajes destinados a ella.

Limitar la duración y el alcance de cada secreto: Cuando se genera primero una clave secreta que podemos estar seguros de que no ha sido comprometida. Cuanto más larga sea la usamos y la más ampliamente es sabido, mayor es el riesgo. El uso de secretos como contraseñas y claves secretas compartidas debe ser limitada en el tiempo, y el intercambio debe ser restringida.

Algoritmos y código del programa están disponibles para los atacantes: El más grande y el más ampliamente distribuido es un secreto, mayor es el riesgo de que se difunda. algoritmos de cifrado secretas son totalmente inadecuadas para entornos de red a gran escala de hoy en día. La mejor práctica es publicar los algoritmos utilizados para el cifrado y la autenticación, basándose únicamente en el secreto de las claves criptográficas. Esto ayuda a asegurar que los algoritmos son fuertes lanzándolos abierto al escrutinio por parte de terceros.

Los atacantes pueden tener acceso a grandes recursos: El costo de potencia de cálculo está disminuyendo rápidamente. Debemos asumir que los atacantes tendrán acceso a los ordenadores más grandes y poderosas proyectados en el tiempo de vida de un sistema, a continuación, añadir algunos órdenes de magnitud para permitir desarrollos inesperados.

Minimizar la base de confianza: Las partes de un sistema que son responsables de la aplicación de su seguridad, *y todo el hardware y componentes de software de los que dependen,* tienen que ser de confianza - esto se refiere a menudo como el *base informática de confianza.* Cualquier defecto o error de programación en esta base de confianza pueden producir fallos de seguridad, por lo que deberán reducirse al mínimo su tamaño. Por ejemplo, los programas de aplicación no se debe confiar para proteger los datos de sus usuarios.

Figura 11.2 notaciones de criptografía

K_{UN}	clave secreta de Alice
$K_{segundo}$	clave secreta de Bob
K_{AB}	clave secreta compartida entre Alice y Bob
K_{Apriv}	la clave privada de Alice (sólo se conoce a Alice)
$K_{Un\ pub}$	la clave pública de Alice (publicado por Alice para que todos puedan leer)
$\{METRO\}_K$	Mensaje *METRO* cifrada con la clave *K*
$[METRO]_K$	Mensaje *METRO* firmado con la clave *K*

11.2.1 criptografía

El cifrado es el proceso de codificación de un mensaje de una manera tal como para ocultar su contenido. La criptografía moderna incluye varios algoritmos de seguridad para cifrar y descifrar mensajes. Todos ellos se basan en el uso de secretos llamada *llaves*. Una clave criptográfica es un parámetro utilizado en un algoritmo de cifrado de tal manera que el cifrado no se puede invertir sin conocimiento de la clave.

Hay dos clases principales de algoritmo de cifrado de uso general. Los primeros usos *claves secretas compartidas* - el remitente y el destinatario deben compartir un conocimiento de la clave y no debe ser revelada a nadie más. La segunda clase de algoritmos de cifrado utiliza */ pares de claves pública y privada*. Aquí el remitente de un mensaje utiliza una *Llave pública* - uno que ya ha sido publicado por el destinatario - para cifrar el mensaje. El destinatario utiliza un correspondiente *llave privada* para descifrar el mensaje. Aunque muchos directores pueden examinar la clave pública, sólo el destinatario puede descifrar el mensaje, porque tienen la clave privada.

Ambas clases de algoritmo de cifrado son extremadamente útiles y se utilizan ampliamente en la construcción de sistemas distribuidos seguras. algoritmos de cifrado de clave pública típicamente requieren de 100 a 1.000 veces más potencia de procesamiento como algoritmos de clave secreta, pero hay situaciones en su conveniencia supera esta desventaja.

11.2.2 Usos de la criptografía

La criptografía juega tres papeles importantes en la implementación de sistemas seguros. Los introducimos aquí a grandes rasgos por medio de algunos escenarios simples. En secciones posteriores de este capítulo, se describen estos y otros protocolos con mayor detalle, abordar algunos problemas sin resolver que no son más destacadas aquí.

En todos nuestros escenarios siguientes, podemos suponer que Alice, Bob y cualesquiera otros participantes ya se han puesto de acuerdo sobre los algoritmos de encriptación que desea utilizar y disponer de implementaciones de ellos. También suponemos que cualquier claves secretas o claves privadas que sean titulares se pueden almacenar de forma segura para evitar que los atacantes obtención de las mismas.

El secreto y la integridad • La criptografía se utiliza para mantener la confidencialidad e integridad de la información cada vez que se expone a ataques potenciales - por ejemplo, durante la transmisión a través de redes que son vulnerables a las escuchas y mensaje de manipulación. Este uso de la criptografía corresponde a su papel tradicional en la militar y

las actividades de inteligencia. Se aprovecha el hecho de que un mensaje cifrado con una clave de cifrado en particular sólo puede ser descifrado por un receptor que conoce la clave de descifrado correspondiente. Por lo tanto, mantiene el secreto del mensaje cifrado, siempre y cuando la clave de descifrado no es *comprometida (* da a conocer a los no participantes en la comunicación) y siempre que el algoritmo de cifrado es lo suficientemente fuerte para vencer cualquier intento posibles para Agriétela. El cifrado también mantiene la integridad de la información cifrada, a condición de que alguna información redundante, tal como una suma de control está incluido y marcada.

Escenario 1. La comunicación secreta con una clave secreta compartida: Alice desea enviar alguna información secreta a Bob. Alice y Bob comparten una clave secreta K_{AB}.

1. usos Alice K_{AB} y una función de cifrado acordada $E(K_{AB}, METRO)$ para cifrar y enviar cualquier número de mensajes $\{METRO_i\}_{KAB}$ a Bob. (Alice puede seguir usando K_{AB} con tal de que es seguro asumir que K_{AB} No se ha visto comprometida.)
2. Bob descifra los mensajes cifrados usando la función de descifrado correspondiente $D(K_{AB}, METRO)$.

Bob ahora puede leer el mensaje original *METRO*. Si el mensaje descifrado tiene sentido, o mejor, si incluye algún valor acordado entre Alice y Bob (como una suma de comprobación del mensaje), entonces Bob sabe que el mensaje es de Javier, y que no ha sido manipulado. Sin embargo, todavía hay algunos problemas:

Problema 1: ¿Cómo puede Alice enviar una clave compartida K_{AB} a Bob de forma segura?

Problema 2: ¿Cómo sabe que cualquier Bob $\{METRO_{yo}\}$ No es una copia de un mensaje cifrado a principios de Alice que fue capturado por Mallory y se repite más adelante? Mallory no tiene que tener la clave K_{AB} para llevar a cabo este ataque - puede simplemente copiar el patrón de bits que representa el mensaje y enviarlo a Bob más adelante. Por ejemplo, si el mensaje es una solicitud para pagar algo de dinero a alguien, Mallory podría engañar a Bob a pagar dos veces. Mostramos cómo estos problemas se pueden resolver más adelante en este capítulo.

• **Autenticación** La criptografía se utiliza en apoyo de los mecanismos para la autenticación de la comunicación entre pares de directores. Un director que descifra un mensaje con éxito utilizando una clave particular puede asumir que el mensaje es auténtico si contiene una suma de comprobación correcta o (si el modo de bloque de encadenamiento de cifrado, que se describe en la Sección 11.3, es usado) algún otro valor esperado. Pueden inferir que el remitente del mensaje poseía la clave de cifrado correspondiente y por lo tanto deducir la identidad del remitente si la clave es conocida sólo por dos partes. Así, si las llaves se llevan a cabo en privado, un descifrado con éxito autentica el mensaje descifrado como procedente de un remitente en particular.

Escenario 2. comunicación autentica con un servidor: Alice desea acceder a los archivos en poder de Bob, un servidor de archivos en la red local de la organización en la que trabaja. Sara es un servidor de autenticación que se gestiona de forma segura. Sara emite usuarios con contraseñas y mantiene las claves secretas actuales para todos los directores en el sistema que sirve (generados mediante la aplicación de algún tipo de transformación de la contraseña del usuario). Por ejemplo, se sabe que la clave de Alice K_{UN} y Bob de $K_{SEGUNDO}$. En nuestro escenario nos referimos a una *boleto*. Un billete es un elemento cifrado emitido por un servidor de autenticación, que contiene la identidad del principal a la que se expide y una clave compartida que se ha generado para la sesión de comunicación actual.

1. Alice envía un mensaje (sin cifrar) a Sara indicando su identidad y solicitar una entrada para el acceso a Bob.

2. Sara envía una respuesta a Alice encriptada en K_{UN} que consiste en un billete (para ser enviado a Bob con cada solicitud de acceso a archivos) cifrada en $K_{segundo}$ y una nueva clave secreta K_{AB} para su uso en la comunicación con Bob. Así que la respuesta que recibe Alice se ve así: {{ *Boleto*$\}_{KB}$; $K_{AB}\}_{KA}$.

3. Alice descifra la respuesta utilizando K_{UN} (el cual se genera a partir de su contraseña usando la misma transformación; la contraseña no se transmite a través de la red, y una vez que se ha utilizado se elimina de almacenamiento local para evitar comprometer ella). Si Alice tiene la clave correcta contraseña derivados K_{UN}, se obtiene un billete válido para usar el servicio de Bob y una nueva clave de cifrado para su uso en la comunicación con Bob. Alice no puede descifrar o manipular el billete, ya que se cifra en $K_{SEGUNDO}$. Si el destinatario no es Alice, entonces no se sabe la contraseña de Alice, por lo que no serán capaces de descifrar el mensaje.

4. Alice envía el boleto a Bob junto con su identidad y una solicitud *R* acceder un archivo: { *Boleto*$\}_{KB}$; *Alice, R*.

5. El billete, originalmente creado por Sara, es en realidad: { K_{AB}, *Alicia*$\}_{KB}$. Bob descifra el boleto usando su llave $K_{SEGUNDO}$. Así que Bob obtiene la identidad auténtica de Alice (basado en el conocimiento compartido entre Alicia y Sara de la contraseña de Alice) y una nueva clave secreta compartida K_{AB} para su uso en la interacción con Alice. (Esto se llama una *clave de sesión* porque con seguridad puede ser utilizado por Alice y Bob para una secuencia de interacciones.) Este escenario es una versión simplificada del protocolo de autenticación desarrollado originalmente por Roger Needham y Michael Schroeder [1978] y posteriormente usado en el sistema Kerberos desarrollado y utilizado en el MIT [Steiner *et al.* 1988], que se describe en la Sección 11.6.2. En nuestra descripción simplificada de su protocolo anterior, no hay protección contra la repetición de los mensajes de autenticación de edad. Esto y algunas otras debilidades se tratan en nuestra descripción del protocolo completo de Needham-Schroeder en la Sección 11.6.1.

El protocolo de autenticación que hemos descrito depende del conocimiento previo por parte del servidor de autenticación de Sara Alice y teclas de Bob, K_{UN} y $K_{SEGUNDO}$. Esto es factible en una sola organización en la que Sara se ejecuta en un ordenador físicamente seguro y es administrado por un director de confianza que genera valores iniciales de las llaves y los transmite a los usuarios por un canal seguro independiente. Pero no es apropiada para el comercio electrónico u otras aplicaciones de área amplia, donde el uso de un canal separado es extremadamente inconveniente y la exigencia de una tercera parte de confianza no es realista. criptografía de clave pública nos rescata de este dilema.

La utilidad de los retos: Un aspecto importante de Needham y Schroeder 1978 gran avance fue la constatación de que la contraseña de un usuario no tiene que ser sometido a un servicio de autenticación (y por lo tanto expuesta en la red) cada vez que se autentica. En su lugar, se introdujo el concepto de criptografía *reto*. Esto se puede ver en el paso 2 de nuestro escenario anterior, donde el servidor, Sara, emite un ticket a Alice *cifrado en clave secreta de Alice,* K_A. Esto constituye un reto porque Alice no puede hacer uso del billete a menos que pueda descifrarlo, y ella sólo puede descifrarlo si ella

puede determinar K_{UN}, que se deriva de la contraseña de Alice. Un impostor que dice ser Alice sería derrotado en este punto.

Escenario 3. comunicación con las claves públicas autenticado: Suponiendo que Bob ha generado un par de claves pública / privada, el siguiente diálogo permite Bob y Alice para establecer una clave secreta compartida, K_{AB}:

1. Alice accede a un servicio de distribución de claves para obtener una *certificado de clave pública* dando la clave pública de Bob. Se llama un certificado, ya que está firmado por una autoridad de confianza
   - una persona u organización que es ampliamente conocido por ser fiable. Después de comprobar la firma, se lee la clave pública de Bob, K_{BPUB}, a partir del certificado. (Se discute la construcción y uso de certificados de clave pública en la Sección 11.2.3.)
2. Alice crea una nueva clave compartida, K_{AB}, y encripta usando K_{BPUB} con un público- algoritmo de clave. Ella envía el resultado a Bob, junto con un nombre que identifica de forma exclusiva un / pair pública clave privada (desde que Bob puede tener varios de ellos) - es decir, Alice envía *nombre de clave,* $\{K_{AB}\}_{KBpub}$.
3. Bob selecciona la clave privada correspondiente, K_{Bpriv}, de su almacén de claves privada y lo utiliza para descifrar K_{AB}. Tenga en cuenta que el mensaje de Alice a Bob podría haber sido dañado o alterado en tránsito. La consecuencia sería simplemente que Bob y Alice no comparten la misma clave K_{AB}. Si esto es un problema, se puede evitar mediante la adición de un valor acordado o cadena al mensaje, tales como nombres o direcciones de correo electrónico de Bob y Alice, que Bob puede comprobar después de descifrar. El escenario anterior ilustra el uso de la criptografía de clave pública para distribuir una clave secreta compartida. Esta técnica se conoce como una *protocolo criptográfico híbrido* y está muy ampliamente utilizado, ya que se aprovecha de las características útiles tanto de clave pública y algoritmos de cifrado de clave secreta.

Problema: Este intercambio de claves es vulnerable a ataques man-in-the-middle. Mallory puede interceptar la petición inicial de Alice para el servicio de distribución de claves para el certificado de clave pública de Bob y enviar una respuesta que contiene su propia clave pública. A continuación, puede interceptar todos los mensajes posteriores. En nuestra descripción anterior, protegemos contra este ataque al exigir el certificado de Bob para ser firmado por una autoridad bien conocida. Para protegerse de este ataque, Alice debe asegurarse de que el certificado de clave pública de Bob está firmado con una clave pública (como se describe más adelante) que ha recibido de una manera totalmente segura.

Firmas digitales • La criptografía se utiliza para implementar un mecanismo conocido como *firma digital*. Esto emula el papel de una firma convencional, la verificación de un tercero que un mensaje o un documento es una copia sin modificación de uno producido por el firmante.

técnicas de firma digital se basan en una unión irreversible al mensaje o documento de un secreto que sólo conoce el firmante. Esto puede lograrse mediante la encriptación del mensaje - o mejor, una forma comprimida del mensaje llama una *digerir* - el uso de una clave que sólo conocen el firmante. Un compendio es un valor de longitud fija calculada por la aplicación de una *asegurar la función de digerir*. Una función digesto seguro es similar a una función de suma de comprobación, pero es muy poco probable que produzca un valor digerir similar para dos mensajes diferentes. El digesto cifrada resultante actúa como una firma que acompaña al mensaje. criptografía de clave pública se utiliza generalmente para esto: el creador genera una firma con su clave privada, y la firma puede ser descifrado por cualquier destinatario mediante el correspondiente

Figura 11.3 certificado de la cuenta bancaria de Alice

1. *Tipo de certificado:*	Número de cuenta
2. *Nombre:*	Alicia
3. *Cuenta:*	6262626
4. *Autoridad de certificación:*	Banco de Bob
5. *Firma:*	{ *Recopilación (campo campo 2 + 3)*} $_{KBpriv}$

Llave pública. Hay un requisito adicional: el verificador debe estar seguro de que la clave pública es realmente la de la directora que afirma ser el firmante - esto es tratado por el uso de certificados de clave pública, que se describe en la Sección 11.2.3.

Escenario 4. Las firmas digitales con una función de digerir seguro: Alice quiere firmar un documento *METRO* de modo que cualquier receptor ulterior puede verificar que es el autor de la misma. Así, cuando Bob tarde accede al documento firmado después de su recepción por cualquier vía y de cualquier fuente (por ejemplo, se podría enviar en un mensaje o podría ser recuperada a partir de una base de datos), se puede verificar que Alice es el originador.

1. Alice calcula una longitud fija compendio del documento, *Recopilación (M)*.
2. Alicia encripta el digesto de su clave privada, lo anexa a *METRO* y hace que el resultado, *M, {Compendio (M)}* $_{KApriv,}$ a disposición de los usuarios previstos.
3. Bob obtiene el documento firmado, extractos *METRO* y calcula *Recopilación (M)*.
4. Bob desencripta { *Recopilación (M)*} $_{KApriv}$ utilizando la clave pública de Alice, $K_{Un\ pub,}$ y compara el resultado con su calculada *Recopilación (M)*. Si coinciden, la firma es válida.

11.2.3 Certificados

Un certificado digital es un documento que contiene una declaración (por lo general corta) firmado por un director. Se ilustra el concepto de un escenario.

Escenario 5. El uso de certificados: Bob es un banco. Cuando sus clientes a establecer contacto con él tienen que estar seguros de que están hablando con Bob el banco, incluso si nunca le han puesto en contacto antes. Bob necesita para autenticar a sus clientes antes de que se les da acceso a sus cuentas.

Por ejemplo, Alice podría ser útil para obtener un certificado de su banco indicando su número de cuenta bancaria (figura 11.3). Alice podría utilizar este certificado al hacer compras para certificar que tiene una cuenta en el Banco de Bob. El certificado está firmado con la clave privada de Bob, $K_{Bpriv.}$ Un vendedor, Carol, puede aceptar un certificado de este tipo para cargar artículos a la cuenta de Alice siempre que ella puede validar la firma en el campo 5. Para ello, Carol necesita tener la clave pública de Bob y ella necesita estar seguro de que es auténtico para proteger contra la posibilidad de que Alice podría firmar un certificado falso asociar su nombre con la cuenta de otra persona. Para llevar a cabo este ataque, Alice simplemente generar un nuevo par de claves, $K_{B'pub,}$ $K_{B'priv,}$ y utilizarlos para generar un certificado falso que pretende provenir del Banco de Bob.

Figura 11.4 certificado de clave pública para el Banco de Bob

1. *Tipo de certificado:*	Llave pública
2. *Nombre:*	Banco de Bob
3. *Llave pública:*	K_{BPUB}
4. *Autoridad de certificación:*	Fred - La Federación de Banqueros
5. *Firma:*	$\{ Recopilación (campo campo 2 + 3)\}_{KFpriv}$

Lo que Carol necesita es un certificado de clave pública de Bob, firmado por una autoridad bien conocida y de confianza. Supongamos que Fred representa a la Federación de Banqueros, una de cuyas funciones es certificar las claves públicas de los bancos. Fred podría emitir una *certificado de clave pública* para Bob (Figura 11.4).

Por supuesto, este certificado depende de la autenticidad de la clave pública de Fred, K_{Fpub}, por lo que tenemos un problema recurrente de autenticidad - Carol sólo puede confiar en este certificado si ella puede estar seguro de que conoce la clave pública auténtica de Fred, K_{Fpub}. Podemos romper esta recursividad, garantizando que obtiene Carol K_{Fpub} de alguna manera en la que pueden confiar - que podría ser entregado por un representante de Fred o ella podría recibir una copia firmada de la misma de alguien que conoce y confía en que dice que la obtuvo directamente de Fred. Nuestro ejemplo ilustra una cadena de certificación - una con dos enlaces, en este caso.

Ya hemos aludido a uno de los problemas que surgen con los certificados - la dificultad de elegir una autoridad de confianza de la que una cadena de autenticaciones puede comenzar. La confianza es absoluta rara vez, por lo que la elección de una autoridad debe dependerá de la finalidad a la que el certificado se va a poner. Otros problemas surgen por el riesgo de claves privadas sean comprometidos (comunicado) y la longitud admisible de una cadena de certificación - el más largo de la cadena, mayor es el riesgo de un eslabón débil.

Siempre que se tenga cuidado para abordar estas cuestiones, las cadenas de certificados son una importante piedra angular para el comercio electrónico y otros tipos de transacciones en el mundo real. Ellos ayudan a abordar el problema de la escala: hay seis mil millones de personas en el mundo, así que ¿cómo podemos construir un entorno electrónico en el que podemos establecer las credenciales de cualquiera de ellos?

Los certificados pueden ser utilizados para establecer la autenticidad de muchos tipos de declaración. Por ejemplo, los miembros de un grupo o asociación podrían desear mantener una lista de correo electrónico que está abierto sólo a los miembros del grupo. Una buena manera de hacer esto sería que el gestor de pertenencia (Bob) para emitir un certificado de miembro ($S, Bob, \{Compendio (S)\}_{KBpriv}$) a cada miembro, en donde S es una declaración de la forma *Alice es un miembro de la Sociedad friendly* y K_{Bpriv} es la clave privada de Bob. Un miembro de aplicar para unirse a la lista de correo electrónico Sociedad de Amigos tendría que suministrar una copia de este certificado al sistema de gestión de listas, que comprueba el certificado antes de permitir que el miembro para unirse a la lista.

Para que los certificados de utilidad, se necesitan dos cosas:

- un formato estándar y representación por ellos para que los emisores de certificados y usuarios de certificados pueden construir con éxito e interpretarlos;

- un acuerdo sobre la manera en que las cadenas de los certificados se construyen, y, en particular, la noción de una autoridad de confianza. Volvemos a estos requisitos en la Sección 11.4.4.

A veces hay una necesidad de revocar un certificado - por ejemplo, Alice podría suspender su pertenencia a la sociedad de amigos, pero ella y los demás probablemente seguirá manteniendo las copias almacenadas de su certificado de membresía. Sería costoso, si no imposible, para localizar y eliminar todos esos certificados, y que no es fácil para invalidar un certificado - que sería necesario notificar a todos los posibles destinatarios de la revocación. La solución habitual a este problema consiste en incluir una fecha de caducidad en el certificado. Cualquier persona que reciba un certificado caducado debe rechazarla, y el sujeto del certificado deberá solicitar su renovación. Si se requiere una revocación más rápida, entonces uno de los mecanismos más complicados mencionados anteriormente se debe recurrir a.

11.2.4 Control de Acceso

A continuación describimos los conceptos en los que el control de acceso a los recursos se basa en los sistemas distribuidos y las técnicas mediante las cuales se ejecuta. La base conceptual para la protección y control de acceso se estableció muy claramente en un trabajo clásico de Lampson [1971], y los detalles de las implementaciones no distribuidos se puede encontrar en muchos libros sobre sistemas operativos (véase, por ejemplo, [Stallings 2008]).

Históricamente, la protección de los recursos en los sistemas distribuidos ha sido en gran parte a servicios específicos. Servidores reciben mensajes de solicitud de la forma < *op, el director de recursos>,* dónde *op* es la operación solicitada, *director de escuela* es una identidad o un conjunto de credenciales para el sujeto que realiza la solicitud y *recurso* identifica el recurso a la que la operación se va a aplicar. El servidor debe autenticarse primero el mensaje de solicitud y las credenciales del principal y luego aplicar el control de acceso, rechazando cualquier solicitud de la que la entidad solicitante no tiene los derechos de acceso necesarios para realizar la operación solicitada en el recurso especificado.

En sistemas distribuidos orientados a objetos puede haber muchos tipos de objeto al que se debe aplicar control de acceso, y las decisiones son a menudo específica de la aplicación. Por ejemplo, Alice puede ser permitido solamente un retiro de dinero de su cuenta bancaria por día, mientras que Bob tiene derecho a tres. las decisiones de control de acceso se suelen dejar al código de nivel de aplicación, pero no se proporciona soporte genérico para gran parte de la maquinaria que apoya las decisiones. Esto incluye la autenticación de los directores, la firma y la autenticación de las solicitudes, y la gestión de credenciales y datos de derechos de acceso.

dominios de protección • Un dominio de protección es un entorno de ejecución compartida por un conjunto de procesos: contiene un conjunto de < *los recursos, los derechos>* pares, una lista de los recursos que puede accederse por todos los procesos que se ejecutan dentro del dominio y que especifican las operaciones permitidas en cada recurso. Un dominio de protección se asocia generalmente con un director determinado - cuando un usuario inicia una sesión, su identidad es autenticada y se crea un dominio de protección para los procesos que van a ejecutar. Conceptualmente, el dominio incluye todos los derechos de acceso que posee el director, incluyendo cualquier derecho que se adquieren a través de la composición de diversos grupos. Por ejemplo, en UNIX, el dominio de protección de un proceso está determinado por los identificadores de usuario y grupo unido al proceso en el tiempo de inicio de sesión. Los derechos se especifican en términos de operaciones permitidas. Por ejemplo, un archivo podría ser lectura y escritura para un proceso y sólo es legible por otro.

Un dominio de protección es más que una abstracción. Dos implementaciones alternativas se utilizan comúnmente en sistemas distribuidos: *capacidades* y *Las listas de control de acceso.*

capacidades: Un conjunto de capacidades se lleva a cabo por cada proceso de acuerdo con el dominio en el que se encuentra. Una capacidad es un valor binario que actúa como una tecla de acceso, permitiendo el acceso al soporte de determinadas operaciones en un recurso especificado. Para el uso en sistemas distribuidos, donde las capacidades deben ser infalsificable, toman una forma tal como:

identificador de recursos	Un identificador único para el recurso de destino
operaciones	Una lista de las operaciones permitidas en el recurso
Código de autenticación	Una firma digital haciendo que la capacidad de infalsificable

Servicios sólo proporcionan capacidades para usuarios cuando les han autenticado como pertenecientes al dominio de protección que se recaba. La lista de operaciones en la capacidad es un subconjunto de las operaciones definidas para el recurso de destino y, a menudo se codifica como un mapa de bits. Capacidades diferentes se utilizan para diferentes combinaciones de derechos de acceso al mismo recurso.

Cuando se utilizan las capacidades, las solicitudes de cliente son de la forma < *op, ID de usuario, capacidad de>*. Es decir, que incluyen una capacidad para el recurso que se accede en lugar de un simple identificador, dando al servidor de prueba inmediata de que el cliente está autorizado a acceder al recurso identificado por la capacidad de las operaciones especificadas por la capacidad. Un registro de entrada, control de acceso en una solicitud que se acompaña de una capacidad de sólo implica la validación de la capacidad y la comprobación de que la operación solicitada está en el conjunto permitido por la capacidad. Esta característica es la principal ventaja de las capacidades

- constituyen una clave de acceso en sí misma, al igual que una llave física a una cerradura de la puerta es una llave de acceso al edificio que protege la cerradura.

Capacidades comparten dos inconvenientes de llaves de una cerradura física:

el robo de clave: Cualquier persona que tiene la llave de un edificio puede utilizarlo para acceder, si son o no son un titular autorizado de la clave - que pueden haber robado la llave u obtenida de alguna manera fraudulenta.

El problema de revocación: El derecho de celebrar una tecla cambia con el tiempo. Por ejemplo, el soporte puede dejar de ser un empleado del propietario del edificio, pero podría retener la llave, o una copia del mismo, y utilizarlo de forma no autorizada. Las únicas soluciones disponibles para estos problemas de teclas físicas son (a) para poner el titular de la clave ilícito de la cárcel - no siempre es factible en una escala de tiempo que les impedirá hacer daño - o (b) para cambiar el bloqueo y vuelva a emitir claves para toda la llave - los titulares de una operación torpe y costoso.

Los problemas análogos para las capacidades son claras:

- Capacidades pueden, por descuido o como resultado de un ataque de espionaje, caen en manos de los directores distintos de aquellos a los que se emitieron. Si esto sucede, los servidores pueden hacer nada para evitar su utilización ilícita.
- Es difícil para cancelar capacidades. La situación del titular puede cambiar y sus derechos de acceso debe cambiar en consecuencia, pero todavía puede utilizar sus capacidades. Las soluciones a estos dos problemas, basándose en la inclusión de información de identificación del titular y en los tiempos de espera más listas de capacidades revocados, respectivamente, se han propuesto y desarrollado [Gong 1989, Hayton *et al.* 1998]. Aunque añaden

complejidad a un concepto de otro modo simple, capacidades siguen siendo una técnica importante

- por ejemplo, pueden ser usados en conjunción con las listas de control de acceso para optimizar el control de acceso en el acceso repetido al mismo recurso, y proporcionan el mecanismo más bonito para la aplicación de delegación (véase la Sección 11.2.5).

Es interesante notar la similitud entre las capacidades y certificados. Considere certificado de Alicia de la propiedad de su cuenta bancaria introducida en la Sección
11.2.3. Se diferencia de capacidades como se describe aquí sólo en que no hay una lista de operaciones permitidas y que el emisor se identifica. Certificados y capacidades pueden ser conceptos intercambiables en algunas circunstancias. certificado de Alice podría ser considerada como una clave de acceso que le permite realizar todas las operaciones permitidas a los titulares de cuentas en su cuenta bancaria, siempre que su identidad puede ser probada.

listas de control de acceso: Una lista se almacena con cada recurso, que contiene una entrada de la forma *<Dominio, operaciones>* para cada dominio que tiene acceso al recurso y dando las operaciones permitidas en el dominio. Un dominio puede ser especificado por un identificador para un principal o puede ser una expresión que puede ser utilizado para determinar la pertenencia de un director del dominio. Por ejemplo, *El propietario de este archivo* es una expresión que se puede evaluar mediante la comparación de la identidad del representado solicitante con la identidad del propietario almacenada en un archivo.

Este es el esquema adoptado en la mayoría de los sistemas de archivos, incluyendo UNIX y Windows NT, donde un conjunto de bits de permiso de acceso está asociado con cada archivo, y los dominios a los que se conceden los permisos se definen por referencia a la información de propiedad almacenado con cada archivo .

Solicitudes a los servidores son de la forma < *op, el director de recursos>*. Para cada solicitud, el servidor autentica el principal y comprueba que la operación solicitada se incluye en la entrada del director en la lista de control de acceso del recurso correspondiente.

• **implementación** Las firmas digitales, credenciales y certificados de clave pública proporcionan la base criptográfica para control de acceso seguro. canales seguros ofrecen beneficios de rendimiento, lo que permite múltiples peticiones para ser manejado sin necesidad de comprobación repetida de los directores y las credenciales [Wobber *et al.* 1994].

Tanto CORBA y Java ofrecen API de seguridad. Soporte para control de acceso es uno de sus principales propósitos. Java proporciona soporte para objetos distribuidos para gestionar su propio control de acceso con *Director, firmante* y *ACL* clases y métodos predeterminados para la autenticación y soporte para certificados, cheques y validación de la firma de control de acceso. De clave secreta y la criptografía de clave pública también se apoyan. Farley [1998] proporciona una buena introducción a estas características de Java. La protección de los programas Java que incluyen el código móvil se basa en el concepto de dominio de protección - código local y el código descargado se proporcionan con diferentes dominios de protección en el que va a ejecutar. Puede haber un dominio de protección para cada fuente de descarga, con los derechos de acceso a diferentes conjuntos de recursos locales, dependiendo del nivel de confianza que se coloca en el código descargado.

Corba ofrece una especificación de Servicio de Seguridad [Blakley 1999, OMG 2002b] con un modelo para los ORB para proporcionar una comunicación segura, autenticación, control de acceso con credenciales, ACL y auditoría; estos se describen con más detalle en la Sección 8.3.

11.2.5 Credenciales

Las credenciales son un conjunto de pruebas aportadas por un director al momento de solicitar el acceso a un recurso. En el caso más simple, un certificado de una autoridad competente indicando la identidad del representado es suficiente, y esto podría ser utilizado para comprobar los permisos del principal en una lista de control de acceso (véase la sección 11.2.4). Esto es a menudo todo lo que se requiere o se proporciona, pero el concepto se puede generalizar para hacer frente a muchos requisitos más sutiles.

No es conveniente para requerir a los usuarios interactuar con el sistema y autenticarse cada vez que se requiere su autoridad para realizar una operación en un recurso protegido. En su lugar, la noción de que una credencial *habla por* un principal se introduce. Así certificado de clave pública de un usuario habla por ese usuario - cualquier proceso de recibir una solicitud autentica con la clave privada del usuario se puede asumir que la solicitud fue emitida por ese usuario.

los *habla por* idea se puede llevar mucho más lejos. Por ejemplo, en una tarea cooperativa, podría ser necesario que ciertas acciones sensibles sólo deben llevarse a cabo con la autoridad de dos miembros del equipo; en ese caso, el director solicitando la acción presentaría su propia credencial de identificación y credenciales de respaldo de otro miembro del equipo, junto con una indicación de que van a tomarse juntos al comprobar las credenciales.

Del mismo modo, a votar en una elección, una solicitud de voto iría acompañado de un certificado de elector, así como un certificado de identificación. Un certificado de delegación permite a un director para actuar en nombre de otro, y así sucesivamente. En general, un registro de entrada de control de acceso implica la evaluación de una fórmula lógica que combina los certificados suministrados. Lampson *et al. [*1992] han desarrollado una lógica integral de autenticación para su uso en la evaluación de la *habla por* autoridad llevada por un conjunto de credenciales. Wobber *et al. [*1994] describen un sistema que apoya este enfoque muy general. Seguir trabajando en formas útiles de credenciales para su uso en tareas cooperativas en el mundo real se puede encontrar en Rowley [1998].

credenciales basadas en funciones parecen ser particularmente útil en el diseño de esquemas de control de acceso práctico [Sandhu *et al.* 1996]. Conjuntos de credenciales basadas en roles están definidos para las organizaciones o para tareas de cooperación y derechos de acceso a nivel de aplicación se construyen con referencia a ellos. Los roles se pueden asignar a los directores específicos por la generación de certificados de papel que asocian con los directores de papeles con nombre en tareas específicas u organizaciones [Coulouris *et al.* 1998].

Delegación • Una forma particularmente útil de credencial es uno que da derecho un principal, o un proceso de actuación para un principal, para realizar una acción con la autoridad de otro principal. Una necesidad para la delegación puede surgir en cualquier situación en la que un servicio necesita acceder a un recurso protegido con el fin de completar una acción en nombre de su cliente. Considere el ejemplo de un servidor de impresión que acepta peticiones para imprimir archivos. Sería un desperdicio de recursos para copiar el archivo, por lo que el nombre del archivo se pasa al servidor de impresión y se accede a ella por el servidor de impresión en nombre del usuario que realiza la solicitud. Si el archivo está protegido contra lectura, esto no funciona a menos que el servidor de impresión puede adquirir derechos temporales para leer el archivo. La delegación es un mecanismo diseñado para resolver problemas como éste.

Delegación se puede lograr utilizando un certificado de delegación o una capacidad. El certificado está firmado por el director de la solicitud y que autoriza a otra principal (el servidor de impresión en nuestro ejemplo) para acceder a un recurso con nombre (el archivo a imprimir). En los sistemas que los soportan, capacidades pueden lograr el mismo resultado sin la necesidad de

identificar los directores - una capacidad de acceder a un recurso se puede pasar en una petición a un servidor. La capacidad es un conjunto infalsificable, codificado de los derechos para acceder al recurso.

Cuando se delegan los derechos, es común para restringir a un subconjunto de los derechos mantenidos por el director de emisión, por lo que el principal delegado no puede abusar de ellos. En nuestro ejemplo, el certificado podría ser limitada en el tiempo para reducir el riesgo de códigos del servidor de impresión posteriormente sean comprometidos y el archivo cedidos a terceros. El Servicio de Seguridad CORBA incluye un mecanismo para la delegación de derechos basado en certificados, con el apoyo de la restricción de los derechos realizadas.

11.2.6 Los cortafuegos

Firewalls se introdujeron y se describen en la Sección 3.4.8. Protegen intranets, las acciones de filtrado que actuarán en las comunicaciones entrantes y salientes. Aquí hablamos de sus ventajas e inconvenientes como los mecanismos de seguridad.

En un mundo ideal, la comunicación siempre estaría entre los procesos de confianza mutua y canales seguros sería siempre ser utilizado. Hay muchas razones por las que este ideal no es alcanzable, algunos se pueden corregir, pero otros inherentes a la naturaleza abierta de sistemas distribuidos o resultantes de los errores que están presentes en la mayoría del software. La facilidad con que los mensajes de solicitud se pueden enviar a cualquier servidor, en cualquier lugar, y el hecho de que muchos servidores no están diseñados para resistir los ataques de piratas informáticos o errores accidentales, hace que sea fácil para que la información que se pretende que sea confidencial a fugarse del ser dueño de los servidores de la organización. elementos indeseables pueden penetrar también en la red de una organización, permitiendo que los programas de gusanos y virus para entrar en sus ordenadores. Ver [web.mit.edu ll] Para una crítica adicional de cortafuegos.

Firewalls producen un entorno de comunicación local en la que se intercepta toda la comunicación externa. Los mensajes se envían al destinatario local destinado únicamente para las comunicaciones que están autorizados explícitamente.

El acceso a las redes internas puede ser controlado por los cortafuegos, pero el acceso a los servicios públicos en Internet es libre porque su propósito es ofrecer servicios a una amplia gama de usuarios. El uso de servidores de seguridad no ofrece protección contra ataques desde el interior de una organización, y es crudo en su control de acceso externo. Hay una necesidad de mecanismos de seguridad finergrained, permitiendo a los usuarios individuales para compartir información con otros seleccionados sin comprometer la privacidad e integridad. Abadi *et al. [* 1998] describen un enfoque para la provisión de acceso a los datos privados de Internet para los usuarios externos en base a una *túnel web* mecanismo que puede ser integrado con un servidor de seguridad. Ofrece acceso a los usuarios de confianza y autenticados a los servidores web internos a través de un proxy seguro basado en el HTTPS (HTTP sobre TLS).

Los cortafuegos no son particularmente eficaces contra los ataques de denegación de servicio, como la basada en IP spoofing que se describe en la Sección 3.4.2. El problema es que la gran cantidad de mensajes generados por este tipo de ataques abruma cualquier punto único de defensa tales como cortafuegos. Cualquier remedio para las inundaciones de mensajes entrantes se debe aplicar también aguas arriba de la meta. Remedios basados en el uso de mecanismos de calidad de servicio para restringir el flujo de mensajes desde la red a un nivel que el objetivo puede manejar parece la más prometedora.

11.3 Los algoritmos criptográficos

Un mensaje es encriptado por el remitente aplica alguna regla para transformar el *Texto sin formato* mensaje (cualquier secuencia de bits) a una *texto cifrado (* una secuencia diferente de bits). El receptor debe conocer la regla inversa con el fin de transformar el texto cifrado en el texto normal. Otros directores no son capaces de descifrar el mensaje a menos que también conocen la regla inversa. La transformación de cifrado se define con dos partes, una *función de correo* y una *clave K.* El mensaje cifrado resultante se escribe *METRO K.*

$$EKM = METRO_K$$

La función de cifrado *mi* define un algoritmo que transforma los elementos de datos en texto plano en los elementos de datos cifrados por la combinación de ellos con la llave y la transposición de ellos de una manera que es dependiente en gran medida del valor de la clave. Podemos pensar en un algoritmo de cifrado como la especificación de una gran familia de funciones de la cual un miembro en particular se selecciona por cualquier tecla determinada. El descifrado se lleva a cabo utilizando una función inversa *RE,* que también tiene una clave como un parámetro. Para el cifrado de clave secreta, la clave utilizada para el descifrado es el mismo que el utilizado para el cifrado:

$$DKEKMM =$$

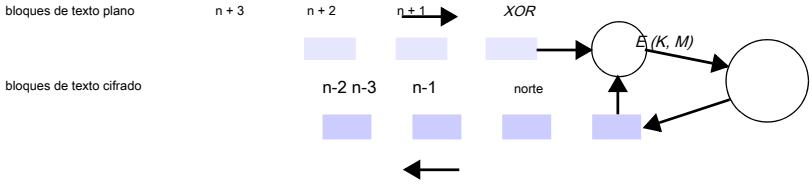
Debido a su uso simétrico de claves, la criptografía de clave secreta se refiere a menudo como *criptografía simétrica,* mientras que la criptografía de clave pública que se conoce como *asimétrico* debido a que las claves utilizadas para el cifrado y descifrado son diferentes, como veremos a continuación. En la siguiente sección, se describen varias funciones de cifrado ampliamente utilizada de ambos tipos.

• Los algoritmos simétricos Si quitamos el parámetro clave de la consideración mediante la definición $F_K METRO = EKM$, entonces es una propiedad de las funciones de cifrado fuerte que $F_K METRO$ es relativamente fácil de calcular, mientras que la inversa, $F_K{}^{-1} M$, es tan difícil calcular que no es factible. Tales funciones se conocen como funciones unidireccionales. La eficacia de cualquier método para cifrar la información depende de la utilización de una función de cifrado F_K que tiene esta propiedad de una sola vía. Esto es lo que protege contra los ataques diseñados para descubrir *METRO* dado $METRO_K$.

Para bien diseñado algoritmos simétricos tales como los descritos en la sección siguiente, su fuerza contra los intentos de descubrir *K* dado un texto plano *METRO* y el texto cifrado correspondiente $METRO_K$ depende del tamaño de *K.* Esto se debe a la forma general más eficaz de ataque es el más cruda, conocida como una *ataque de fuerza bruta.* El enfoque de fuerza bruta es ejecutar a través de todos los valores posibles de *K,* informática *EKM* hasta que el resultado coincide con el valor de $METRO_K$ que ya se sabe. Si *K* tiene *norte* bits de este tipo de ataque requiere $2^{norte-1}$ iteraciones en promedio, y un máximo de 2^{norte} iteraciones, para encontrar *K.* Por lo tanto el tiempo de roer *K* es exponencial en el número de bits en *K.*

• Los algoritmos asimétricos Cuando se utiliza un par de claves pública / privada, funciones unidireccionales son explotados de otra manera. La viabilidad de un sistema de clave pública fue propuesto por primera vez por Diffie y Hellman [1976] como un método criptográfico que elimina la necesidad de confianza entre las partes que se comunican. La base de todos los esquemas de clave pública es la existencia de *funciones trampilla.* Una función de la trampilla es una función unidireccional con una salida secreta - es fácil de calcular en una dirección pero no factible para calcular la inversa menos que se conozca un secreto. Era la posibilidad de encontrar este tipo de funciones y su uso

Figura 11.5 encadenamiento de bloques de cifrado



en criptografía práctica que Diffie y Hellman sugeridos por primera vez. Desde entonces, se han propuesto y desarrollado varios esquemas de clave pública prácticos. Todos ellos dependen de la utilización de las funciones trampilla que involucran grandes cantidades.

El par de claves necesarias para los algoritmos asimétricos se deriva de una raíz común. Para el algoritmo RSA, que se describe en la Sección 11.3.2, la raíz es un par de números primos muy grandes elegido arbitrariamente. La derivación del par de claves de la raíz es una función unidireccional. En el caso del algoritmo RSA, los números primos grandes se multiplican entre sí - un cálculo que toma sólo unos segundos, incluso para los muy grandes números primos utilizados. El producto resultante, *NORTE,* es, por supuesto, mucho más grandes que los multiplicanda. Este uso de la multiplicación es una función unidireccional en el sentido de que es computacionalmente imposible derivar los multiplicanda originales del producto - es decir, que factorizar el producto.

Uno de los pares de claves se utiliza para el cifrado. Para RSA, la función de cifrado oscurece el texto en claro mediante el tratamiento de cada bloque de bits como un número binario y elevarlo a la potencia de la clave, de módulo *NORTE.* El número resultante es el bloque de texto cifrado correspondiente.

La talla de *norte* y al menos uno del par de claves es mucho más grande que el tamaño de clave segura para las claves simétricas para asegurar que *norte* No es factorizable. Por esta razón, el potencial de ataques de fuerza bruta en RSA es pequeña; su resistencia a los ataques depende de la inviabilidad de factorización *NORTE.* Discutimos tamaños seguras para *norte* en la Sección 11.3.2.

cifrados en bloque • La mayoría de los algoritmos de cifrado operan en bloques de tamaño fijo de datos; 64 bits es un tamaño popular para los bloques. Un mensaje se subdivide en bloques, el último bloque se rellena a la longitud estándar, si es necesario y cada bloque se cifra de forma independiente. El primer bloque está disponible para la transmisión tan pronto como ha sido cifrado.

Para un simple cifrado de bloque, el valor de cada bloque de texto cifrado no depende de los bloques anteriores. Esto constituye una debilidad, ya que un atacante puede reconocer patrones repetidos e inferir su relación con el texto en claro. Tampoco es la integridad de los mensajes garantizados a menos que una suma de comprobación o se utiliza mecanismo seguro digerir. La mayoría de los algoritmos de cifrado de bloques emplean el encadenamiento de bloques de cifrado (CBC)

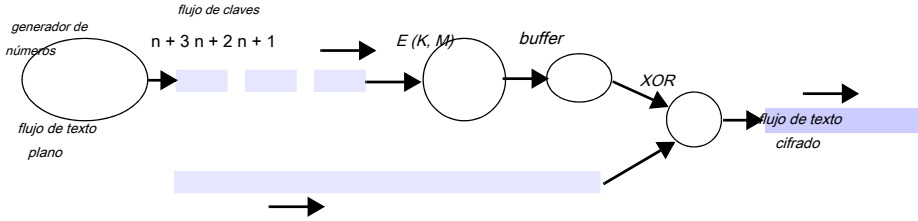
para superar estas debilidades.

Cifrado encadenamiento de bloques: En el modo de encadenamiento de bloques de cifrado, cada bloque de texto claro se combina con el bloque de texto cifrado anterior utilizando el o-exclusiva operación (XOR) antes de que se cifra (figura 11.5). En el descifrado, el bloque se descifra y entonces el bloque cifrado anterior (que debería haber sido almacenada para este propósito) es XOR-ed con ella para obtener el nuevo bloque de texto claro. Esto funciona porque la operación XOR es su propio inverso

- dos aplicaciones de que producen el valor original.

CBC se pretende evitar porciones idénticas de cifrado de texto claro a piezas idénticas de texto cifrado. Pero hay una debilidad en el inicio de cada secuencia de bloques - si

Figura 11.6 Cifrado de flujo



abrimos las conexiones cifradas a dos destinos y enviar el mismo mensaje, las secuencias cifradas de bloques serán los mismos, y un espía podrían obtener alguna información útil a partir de esto. Para evitar esto, es necesario insertar una pieza diferente de texto plano delante de cada mensaje. Dicho texto se denomina *vector de inicialización.* Una marca de tiempo hace que un buen vector de inicialización, obligando a cada mensaje para comenzar con un bloque de texto plano diferente. Esto, combinado con la operación CBC, resultará en diferentes textos cifrados incluso para dos textos planos idénticos.

El uso del modo CBC se limita a la encriptación de datos que se transfieren a través de una conexión fiable. Descifrado fallará si se pierden todos los bloques de texto cifrado, ya que el proceso de descifrado no será capaz de descifrar todas las demás secuencias. Por tanto, es inadecuado para su uso en aplicaciones tales como los descritos en el Capítulo 18, en el que algo de pérdida de datos puede ser tolerada. Un cifrado de flujo se debe utilizar en tales circunstancias.

cifrados de flujo • Para algunas aplicaciones, como el cifrado de las conversaciones telefónicas, cifrado en bloques es inapropiada debido a que los flujos de datos se producen en tiempo real en pequeños trozos. muestras de datos puede ser tan pequeño como 8 bits o incluso un solo bit, y sería un desperdicio de la almohadilla de cada uno de estos a 64 bits antes de la codificación y la transmisión de ellos. cifrados de flujo son algoritmos de cifrado que pueden realizar el cifrado de forma incremental, la conversión de texto plano en texto cifrado de un bit a la vez.

Esto suena difícil de lograr, pero en realidad es muy simple para convertir un algoritmo de cifrado de bloques para su uso como un cifrado de flujo. El truco es construir una *generador de flujo de clave.* Una cadena de claves es una secuencia de longitud arbitraria de bits que pueden utilizarse para ocultar el contenido de un flujo de datos por XOR-ing la cadena de claves con el flujo de datos (Figura 11.6)

. Si el flujo de claves es seguro, entonces también lo es el flujo de datos encriptado resultante. La idea es análoga a una técnica utilizada en la comunidad de inteligencia para frustrar los fisgones, donde se juega 'ruido blanco' para ocultar la conversación en una habitación mientras sigue grabando la conversación. Si el sonido ambiente ruidoso y el ruido blanco se graban por separado, la conversación se puede reproducir sin ruido restando la grabación de la grabación habitación ruidosa ruido blanco.

Un generador de flujo de clave puede ser construido por iteración de una función matemática sobre un rango de valores de entrada para producir una corriente continua de valores de salida. Los valores de salida se concatenan para hacer bloques de texto plano, y los bloques son encriptados usando una clave compartida por el emisor y el receptor. La cadena de claves se puede disimular aún más mediante la aplicación de CBC. Los bloques cifrados resultantes se utilizan como la cadena de claves. Una iteración de casi cualquier función que ofrece una gama de diferentes valores no enteros va a hacer por el material de origen, sino un generador de números aleatorios se utiliza generalmente con un valor de partida para la iteración acordados entre el emisor y el receptor. Para mantener la calidad del servicio para el flujo de datos, los bloques de corriente de clave se deben producir un poco

por delante de la hora a la que se van a utilizar, y el proceso que los produce no debe exigir mucho esfuerzo de procesamiento que el flujo de datos se retrasa.

Por lo tanto, en principio, los flujos de datos en tiempo real pueden ser encriptados tan segura como datos por lotes, siempre que suficiente potencia de procesamiento está disponible para cifrar el flujo de claves en tiempo real. Por supuesto, algunos de los dispositivos que podrían beneficiarse de cifrado en tiempo real, tales como teléfonos móviles, no están equipados con procesadores muy potentes, y en ese caso, puede ser necesario reducir la seguridad del algoritmo de serie de claves.

Diseño de algoritmos criptográficos • Hay muchos algoritmos criptográficos bien diseñados de tal manera que $EKM = METRO_K$ oculta el valor de *METRO* y lo hace prácticamente imposible recuperar *K* más rápidamente que por la fuerza bruta. Todos los algoritmos de cifrado se basan en la manipulación de información de preservación de *METRO* utilizando los principios basados en la teoría de la información [Shannon 1949]. Schneier [1996] describe los principios de la de Shannon *Confusión* y *difusión* para ocultar el contenido de un bloque de texto cifrado *METRO,* combinándolo con una llave *K* de tamaño suficiente para que sea a prueba de ataques de fuerza bruta.

Confusión: operaciones no destructivos tales como XOR y de desplazamiento circular se utilizan para combinar cada bloque de texto claro con la llave, la producción de un nuevo patrón de bits que oscurece la relación entre los bloques en *METRO* y $\{METRO\}_K$. Si los bloques son más grandes que algunos caracteres que esto perjudica el análisis basado en el conocimiento de las frecuencias de caracteres. (La máquina de la Segunda Guerra Mundial alemán Enigma utilizó encadenado bloques de una sola letra, y finalmente fue derrotado por el análisis estadístico.)

Difusión: Por lo general hay repetición y la redundancia en el texto en claro. Difusión disipa los patrones regulares que resultan mediante la transposición de porciones de cada bloque de texto claro. Si se utiliza CBC, la redundancia también se distribuye a lo largo de un texto más largo. cifrados de flujo no pueden utilizar la difusión ya que no hay bloques.

En las dos secciones siguientes, se describe el diseño de varios algoritmos prácticos importantes. Todos ellos han sido diseñados a la luz de los principios antes mencionados han sido objeto de un análisis riguroso y se considera que es seguro contra todos los ataques conocidos con un considerable margen de seguridad. Con la excepción del algoritmo de TEA, que se describe con fines ilustrativos, los algoritmos descritos aquí se encuentran entre los más ampliamente utilizado en aplicaciones donde se requiere una gran seguridad. En algunos de ellos quedan algunas debilidades o áreas de preocupación de menor importancia; el espacio no nos permite describir todas esas preocupaciones aquí, y se remite al lector a Schneier [1996] para más información. Resumimos y comparar la seguridad y el rendimiento de los algoritmos en la Sección 11.5.1.

Los lectores que no requieren una comprensión del funcionamiento de los algoritmos criptográficos pueden omitir las Secciones 11.3.1 y 11.3.2.

11.3.1-clave secreta (simétrica) algoritmos

Muchos algoritmos criptográficos se han desarrollado y publicado en los últimos años. Schneier [1996] describe más de 25 algoritmos simétricos, muchos de los cuales se identifica como seguro contra ataques conocidos. Aquí tenemos espacio para describir sólo tres de ellos. Hemos optado por la primera, TEA, por la simplicidad de su diseño e implementación, y utilizarlo para dar un ejemplo concreto de la naturaleza de este tipo de algoritmos. Vamos a discutir la DES y algoritmos de IDEA con menos detalle. DES era

Figura 11.7 función de encriptación TEA

```
void cifrar (sin signo k largo [], sin signo de texto largo []) {
        unsigned long y = texto [0], z = texto [1];                          1
        unsigned delta largo = 0x9e3779b9, suma = 0; int n;                  2
        para (n = 0; n <32; n ++) {                                          3
                suma + = delta;                                              4
                y + = ((z << 4) + k [0]) ^ (z + suma) ^ ((z >> 5) + k [1]);  5
                z + = ((y << 4) + k [2]) ^ (y + suma) ^ ((y >> 5) + k [3]);  6
        }
        texto [0] = y; texto [1] = z;                                        7
}
```

un estándar nacional de Estados Unidos durante muchos años, pero es ahora en gran parte del interés histórico, ya que sus claves de 56 bits son demasiado pequeños para resistir el ataque de fuerza bruta con el hardware moderno. IDEA utiliza una clave de 128 bits. Es uno de los algoritmos de cifrado simétrico de bloques más eficaces y una buena opción integral para el cifrado a granel.

En 1997, el Instituto Nacional de Estándares y Tecnología (NIST) emitió una invitación a presentar propuestas para un algoritmo para sustituir a DES como un nuevo estándar de cifrado avanzado de Estados Unidos (AES). En octubre de 2000 el ganador fue seleccionado entre 21 algoritmos presentados por los criptógrafos de 11 países. El algoritmo Rijndael ganadora fue elegida por su combinación de fuerza y eficiencia. Más información sobre ella es la siguiente.

TÉ • Los principios de diseño de algoritmos simétricos descritos anteriormente se ilustran bien en el cifrado Tiny Algoritmo (TEA), desarrollado en la Universidad de Cambridge [Wheeler y Needham 1994]. La función de cifrado, programado en C, se da en su totalidad en

Figura 11.7.

El algoritmo TEA utiliza rondas de adición número entero, XOR (el operador ^) y bit a bit cambios lógicos (<< y >>) para lograr la difusión y la confusión de los patrones de bits en el texto en claro. El texto plano es un bloque de 64 bits representado como dos enteros de 32 bits en el vector *texto[]*. La clave es de 128 bits de largo, representado como cuatro enteros de 32 bits.

En cada una de las 32 rondas, las dos mitades del texto se combinan varias veces con porciones desplazadas de la llave y la otra en las líneas 5 y 6. El uso de XOR y desplazan porciones del texto proporciona confusión, y el desplazamiento y el intercambio de las dos porciones del texto proporciona difusión. La constante no repetitivo *delta* se combina con cada parte del texto en cada ciclo para ocultar la clave en caso de que se manifieste por una sección de texto que no varía. La función de descifrado es la inversa de que para el cifrado y se da en la figura 11.8.

Este corto programa proporciona cifrado de clave secreta segura y razonablemente rápido. Es algo más rápido que el algoritmo DES, y la concisión del programa se presta a la optimización e implementación de hardware. La clave de 128 bits es seguro contra ataques de fuerza bruta. Los estudios realizados por sus autores y otros han puesto de manifiesto sólo dos debilidades mínimas, en el que los autores abordan en una nota posterior [Wheeler y Needham 1997].

Figura 11.8 función TEA descifrado

```
void decrypt (sin signo k largo [], sin signo de texto largo []) {
    unsigned long y = texto [0], z = texto [1];
    unsigned long delta = 0x9e3779b9, suma = delta << 5; int n; para (n = 0; n
    <32; n ++) {
        z - = ((y << 4) + k [2]) ^ (y + suma) ^ ((y >> 5) + k [3]); y - = ((z << 4)
      + k [0]) ^ (z + suma) ^ ((z >> 5) + k [1]); suma - = delta; }


    texto [0] = y; texto [1] = z; }
```

Para ilustrar su uso , La figura 11.9 muestra un procedimiento simple que utiliza TEA a cifrar y descifrar un par de archivos abiertos anteriormente (utilizando el C *stdio* biblioteca).

• **DES** El estándar de cifrado de datos (DES) [Oficina Nacional de Normalización 1977] fue desarrollado por IBM y posteriormente adoptado como un estándar nacional de Estados Unidos para aplicaciones gubernamentales y empresariales. En este estándar, la función de cifrado mapea una entrada de texto plano de 64 bits en una salida de cifrado de 64 bits usando una clave de 56 bits. El algoritmo tiene 16 etapas clave-dependientes conocidos como *rondas,* en el que los datos a ser cifrados es bit girado por un número de bits determinados por la llave y tres transposiciones clave-independientes. El algoritmo fue mucho tiempo para realizar en el software en los ordenadores de los años 1970 y 1980, pero fue implementada en hardware VLSI rápido y puede ser fácilmente incorporado en la interfaz de red y otros chips de comunicación.

Figura 11.9 TEA en uso

```
té vacío (modo char, FILE * archivoentrada, FILE * archivo de salida, sin firmar larga k []) {/ * es el modo
'e' para cifrar, 'd' para descifrar, k [] es la clave. * /
    Char ch, Texto [8]; int i; while (! feof
    (archivoentrada)) {
        i = fread (Texto, 1, 8, infile);          / * Leer 8 bytes del archivo de entrada en texto * /
        si (i <= 0) break;
        mientras que (i <8) {Texto [i ++] = ' ';}          / * Almohadilla último bloque con espacios * /
        interruptor (modo)
        {case 'e':
            cifrar (k, (unsigned long *) Texto); descanso; caso 'd':


            descifrar (k, (unsigned long *) Texto); descanso; }


        fwrite (Texto, 1, 8, archivosalida);          / * 8 bytes de escritura de texto al archivo de salida * /
    }}
```

En junio de 1997, se quebró con éxito en un ataque de fuerza bruta ampliamente publicitado. El ataque se llevó a cabo en el contexto de una competición para demostrar la falta de seguridad de cifrado con claves más cortas de 128 bits [www.rsasecurity.com II]. Un consorcio de usuarios de Internet publicó un programa de aplicación de cliente en una de sus PCs y otras estaciones de trabajo, cuyo número llegó a 14.000 durante un período de 24 horas [Curtin y Dolske 1998].

El programa cliente se dirige a agrietarse la clave particular usado en un texto plano muestra conocida / texto cifrado y luego usarla para descifrar un mensaje secreto desafío. Los clientes interactuaron con un único servidor que coordina su trabajo, la emisión de cada cliente con rangos de valores clave para comprobar y recibir informes sobre la marcha de los mismos. El equipo cliente típico corrió el programa cliente sólo como una actividad de fondo y tenía un rendimiento aproximadamente igual a un procesador Pentium de 200 MHz. La clave estaba roto en aproximadamente 12 semanas, después de aproximadamente 25% de la posible 2^{56} o 6 10 dieciséis

Los valores habían sido verificados. En 1998, una máquina fue desarrollada por la Fundación Frontera Electrónica [FEP 1998] que pueden agrietarse con éxito claves DES en torno a tres días.

A pesar de que todavía se utiliza en muchas aplicaciones comerciales y otros, DES en su forma básica debe ser considerado obsoleto para la protección de toda la información, pero de bajo valor. Una solución que se conoce como que se utiliza con frecuencia *triple DES (* o *3DES) [* ANSI 1985, Schneier 1996]. Esto implica la aplicación de DES tres veces con dos llaves, K_1 y K_2:

$$mi_{3\,DES}\,K_1\,K_2\;YO_{DES}\,K_1\;re_{DES}\,K_2\;mi_{DES}\,K_1\;METRO$$

Esto da una fortaleza contra los ataques de fuerza bruta equivalente a una longitud de clave de 112 bits - adecuados para el futuro previsible - pero tiene el inconveniente de bajo rendimiento resultante de la triple aplicación de un algoritmo que ya es lento para los estándares modernos.

IDEA • El Algoritmo de Encriptación de Datos Internacional (IDEA) fue desarrollado en la década de 1990 [Lai y Massey 1990, Lai 1992] como sucesor de DES. Como el té, que utiliza una clave de 128 bits para cifrar bloques de 64 bits. Su algoritmo se basa en el álgebra de grupos y tiene ocho rondas de XOR, adición módulo $2^{dieciséis}$ y la multiplicación. Por tanto DES e IDEA, la misma función se utiliza para el cifrado y el descifrado: una propiedad útil para los algoritmos que se van a implementar en hardware.

La fuerza de la idea ha sido ampliamente analizado, y no se han encontrado deficiencias significativas. Se lleva a cabo el cifrado y descifrado en aproximadamente tres veces la velocidad de la DES.

• RC4 RC4 es un cifrado de flujo desarrollado por Ronald Rivest [Rivest 1992b]. Las claves pueden ser de cualquier longitud de hasta 256 bytes. RC4 es fácil de implementar [Schneier 1996, pp. 397-8] y realiza el cifrado y descifrado de aproximadamente 10 veces más rápido que DES. Fue, por tanto, ampliamente adoptado en aplicaciones que incluyen las redes IEEE 802.11 WiFi, pero una debilidad fue descubierta posteriormente por Fluhrer *et al. [* 2001] que permitió a los atacantes para romper algunas de las claves. Esto condujo a un rediseño de la seguridad 802.11 (véase la Sección 11.6.4 para más detalles).

• AES El algoritmo Rijndael seleccionado para convertirse en el algoritmo estándar de cifrado avanzado por el NIST fue desarrollado por Joan Daemen y Vincent Rijmen [Daemen y Rijmen 2000, 2002]. El sistema de cifrado tiene una longitud variable de bloque y la longitud de clave, con las especificaciones para las claves con una longitud de 128, 192 o 256 bits para cifrar bloques con una

longitud de 128, 192 o 256 bits. Tanto la longitud de bloque y la longitud de clave pueden ser extendidos por múltiplos de 32 bits. El número de rondas en el algoritmo varía del 9 al 13 en función de los tamaños de clave y de bloque. Rijndael puede ser implementado de manera eficiente en una amplia gama de procesadores y hardware.

11.3.2-clave pública (asimétrica) algoritmos

Sólo unos pocos esquemas de clave pública prácticas se han desarrollado hasta la fecha. Dependen de la utilización de las funciones trampilla de un gran número de producir las llaves. Las llaves *Ke* y *kd* son un par de números muy grandes, y la función de cifrado realiza una operación, tal como la exponenciación en *METRO,* utilizando uno de ellos. El descifrado es una función similar utilizando la otra clave. Si la exponenciación utiliza aritmética modular, se puede demostrar que el resultado es el mismo que el valor original de *METRO;* es decir:

$$D(K_{re}, E(K_{mi}, M)) = M$$

Un director que deseen participar en la comunicación segura con los demás hace un par de claves, *Ke* y *kd,* y mantiene la clave de descifrado *kd* un secreto. La clave de cifrado *Ke* puede ser dado a conocer públicamente para ser utilizado por cualquier persona que desee comunicarse. La clave de cifrado *Ke* puede ser visto como una parte de la función de cifrado unidireccional *MI,* y la clave de descifrado *kd* es la pieza de conocimiento secreto que permite directora *pag* para revertir la encriptación. Cualquier titular de *Ke (* que está ampliamente disponible) puede cifrar mensajes { *METRO}* $_{Ke,}$

pero sólo el director que tiene el secreto *kd* puede operar la trampilla.

El uso de las funciones de un gran número conduce a grandes costos de procesamiento en el cálculo de las funciones *mi* y *RE.* Más adelante veremos que este es un problema que ha de abordarse mediante el uso de claves públicas sólo en las etapas iniciales de las sesiones de comunicaciones seguras. El algoritmo RSA es sin duda el más ampliamente conocido algoritmo de clave pública y que lo describen con cierto detalle aquí. Otra clase de algoritmos se basa en funciones derivadas del comportamiento de las curvas elípticas en un plano. Estos algoritmos ofrecen la posibilidad de funciones de cifrado y descifrado menos costosas con el mismo nivel de seguridad, pero su aplicación práctica es menos avanzado y nos ocupamos de ellos sólo brevemente.

• **RSA** El Rivest, Shamir y Adelman (RSA) para el diseño de un sistema de cifrado de clave pública [Rivest *et al.* 1978] se basa en el uso del producto de dos números primos muy grandes (mayor que 10^{100}), basándose en el hecho de que la determinación de los factores primos de un número tan grande es tan computacionalmente difícil como sea efectivamente imposible.

A pesar de extensas investigaciones sin defectos se han encontrado en ella, y ahora se utiliza muy ampliamente. Un esquema del método de la siguiente manera. Para encontrar un par de claves < *e, d*>:

1. Elija dos números primos grandes, *PAG* y *Q (* cada uno mayor que 10^{100}), y la forma
   $N = PQZ =$
   $(P - 1) \quad (Q - 1)$
2. Para *re,* elegir cualquier número que es relativamente privilegiada con *Z (* es decir, tal que *re* tiene no hay factores comunes con *Z).*

Nos ilustran los cálculos necesarios utilizando valores enteros pequeños para *PAG* y *Q:*

$P = 13,\ Q = 17\ N = 221,\ Z = 192$

$d = 5$

3. Para encontrar *mi,* resuelve la ecuación:

$$ed = 1 \bmod Z$$

Es decir, *ed* es el elemento más pequeño divisible por *re* en la serie $Z + 1, 2Z + 1, 3Z + 1, \ldots$.

$ed = 1 \bmod 192 = 1, 193, 385, 385 \ldots$ es divisible por $de = 385/5 = 77$

Para cifrar el texto usando el método RSA, el texto plano se divide en bloques de longitud iguales *k* los bits, donde $2^k < N$ (es decir, de tal manera que el valor numérico de un bloque es siempre menor que *NORTE;* en aplicaciones prácticas, *k* es por lo general en el rango de 512 a 1024).

$k = 7$, ya que $2^7 = 128$

La función para el cifrado de un solo bloque de texto claro *METRO* es:

$$E'(E, N, M) = M^{mi} \bmod N$$

para un mensaje *METRO,* es el texto cifrado $METRO^{77} \bmod 221$

La función de descifrado de un bloque de texto encriptado *do* para producir el bloque de texto claro original es:

$$D'(d, N, c) = c^{re} \bmod N$$

Rivest, Shamir y Adelman demostraron que *MI'* y *RE'* son inversas de inversión (es decir, $E'(D'(x)) = D'(E'(x)) = X$) para todos los valores de *PAG* en el rango de 0 *P N.*

Los dos parámetros *e, N* puede ser considerado como una clave para la función de cifrado, e igualmente los parámetros *d, N* representar una clave para la función de descifrado. Así podemos escribir *Ke* = <e, N> y *Kd* = <d, N>, y obtenemos las funciones de cifrado *Suplir las deficiencias, M)* = $\{M\}_K$ (la notación aquí lo que indica que el mensaje cifrado puede ser descifrado por el titular de la *privado* llave *kd)* y *D (Kd,* $\{M\}_K$) = *METRO.*

Vale la pena señalar una debilidad potencial de todos los algoritmos de clave pública - porque la clave pública está disponible para los atacantes, pueden generar fácilmente los mensajes cifrados. Por lo tanto pueden intentar descifrar un mensaje desconocido mediante el cifrado exhaustivamente secuencias de bits arbitrarias hasta que se consigue una coincidencia con el mensaje de destino. Este ataque, que se conoce como una *ataque de texto elegido,* es derrotado por asegurar que todos los mensajes son más largos que la longitud de la clave, por lo que esta forma de ataque de fuerza bruta es menos factible que un ataque directo a la tecla.

Un destinatario de la intención de la información secreta debe publicar o distribuir el par < *E, N*> mientras se mantiene *re* secreto. La publicación de < *E, N*> no pone en peligro el secreto de *re,* porque cualquier intento de determinar *re* requiere el conocimiento de los números primos originales *PAG* y *Q,* y estos sólo se pueden obtener mediante la factorización de *NORTE.*

Factoring de un gran número (recordamos que *PAG* y *Q* fueron elegidos para ser> 10^{100}, así que *norte* > 10^{200}) es extremadamente lento, incluso en equipos de muy alto rendimiento. En 1978, Rivest *et al.* concluido que factorizar un número tan grande como 10^{200} llevaría más de cuatro mil millones de años con el algoritmo más conocido en un equipo que lleva a cabo una

millones de instrucciones por segundo. Un cálculo similar para los ordenadores de hoy en día sería reducir este tiempo a alrededor de un millón de años,

El RSA Corporation ha emitido una serie de retos para factorizar números de más de 100 dígitos decimales [www.rsasecurity.com III]. En el momento de la escritura, los números de hasta 174 dígitos decimales (576 dígitos binarios) se han contabilizado con éxito, por lo que el uso del algoritmo RSA con claves de 512 bits es claramente inaceptable débil para muchos propósitos. El RSA Corporation (titulares de las patentes en el algoritmo RSA) recomienda una longitud de clave de al menos 768 bits, o alrededor de 230 dígitos decimales, para largo plazo (~ 20 años) de seguridad. Llaves tan grandes como 2048 bits se utilizan en algunas aplicaciones.

Los cálculos de resistencia anteriores se supone que los algoritmos de factorización conocidas en la actualidad son los mejores disponibles. RSA y otras formas de criptografía asimétrica que utilizan la multiplicación del número primo como su función unidireccional será vulnerable si se descubre un algoritmo de factorización más rápido.

algoritmos de curva elíptica • Un método para generar los pares de claves públicas / privadas en base a las propiedades de las curvas elípticas se ha desarrollado y probado. Los detalles completos se pueden encontrar en el libro de Menezes dedicados al tema [Menezes 1993]. Las llaves se derivan de una rama diferente de las matemáticas, ya diferencia de RSA su seguridad no depende de la dificultad de factorizar números grandes. claves más cortas son seguras, y los requisitos de procesamiento para el cifrado y el descifrado son inferiores a los de RSA. algoritmos de encriptación de curva elíptica es probable que se adoptado más ampliamente en el futuro, especialmente en sistemas tales como los dispositivos móviles que contengan, que tienen recursos de procesamiento limitados. La matemática relevante implica algunas propiedades muy complejas de curvas elípticas y está más allá del alcance de este libro.

11.3.3 protocolos criptográficos Híbridos

criptografía de clave pública es conveniente para el comercio electrónico, porque no hay necesidad de un mecanismo de distribución de claves seguras. (Hay una necesidad para autenticar las claves públicas, pero esto es mucho menos onerosa, que sólo requiere un certificado de clave pública para ser enviado con la llave.) Sin embargo, los costos de procesamiento de criptografía de clave pública son demasiado altos para el cifrado de hasta el mensajes de tamaño medio se encuentran normalmente en el comercio electrónico. La solución adoptada en la mayoría de los sistemas distribuidos a gran escala es usar un esquema de cifrado híbrido en el que se utiliza la criptografía de clave pública para autenticar las partes y para cifrar un intercambio de claves secretas, que se utilizan para todas las comunicaciones posteriores. Se describe la implementación de un protocolo híbrido en el estudio de caso TLS en la Sección 11.6.3.

11.4 Las firmas digitales

firmas digitales fuertes son un requisito esencial para sistemas seguros. Son necesarias con el fin de certificar ciertas piezas de información - por ejemplo, para proporcionar declaraciones de confianza identidades de sus claves públicas o de unión de algunos derechos de acceso o funciones a los usuarios de los usuarios de unión de identidades.

La necesidad de firmas en muchos tipos de negocio y operación personal está fuera de discusión. firmas manuscritas se han utilizado como un medio para verificar los documentos durante el tiempo que han existido documentos. firmas manuscritas se utilizan para satisfacer las necesidades de los receptores de documentos para verificar que el documento es:

Auténtico: Convence al destinatario que el firmante firmó deliberadamente el documento y no ha sido alterado por cualquier otra persona.

infalsificable: Proporciona una prueba de que el firmante, y nadie más, firmaron el documento de forma deliberada. La firma no puede ser copiado y colocado en otro documento.

No repudiable: El firmante no puede creíblemente negar que el documento fue firmado por ellos.

En realidad, ninguna de estas propiedades deseables de la firma se logra en su totalidad por firmas convencionales - falsificaciones y copias son difíciles de detectar, los documentos pueden ser alterados después de la firma y firmantes a veces son engañados en la firma de un documento de forma involuntaria o inconscientemente - pero estamos dispuestos a vivir con su imperfección debido a la dificultad de hacer trampa y el riesgo de detección. Al igual que las firmas manuscritas, firmas digitales dependen de la unión de un atributo único y secreto del firmante de un documento. En el caso de las firmas manuscritas, el secreto es el patrón de escritura a mano del firmante.

Las propiedades de los documentos digitales conservados en ficheros o mensajes almacenados son completamente diferentes de las de los documentos en papel. Los documentos digitales son trivialmente fácil de generar, copiar y modificar. Simplemente añadiendo la identidad del originador, ya sea como una cadena de texto, una fotografía o una imagen escrita a mano, no tiene ningún valor para fines de verificación.

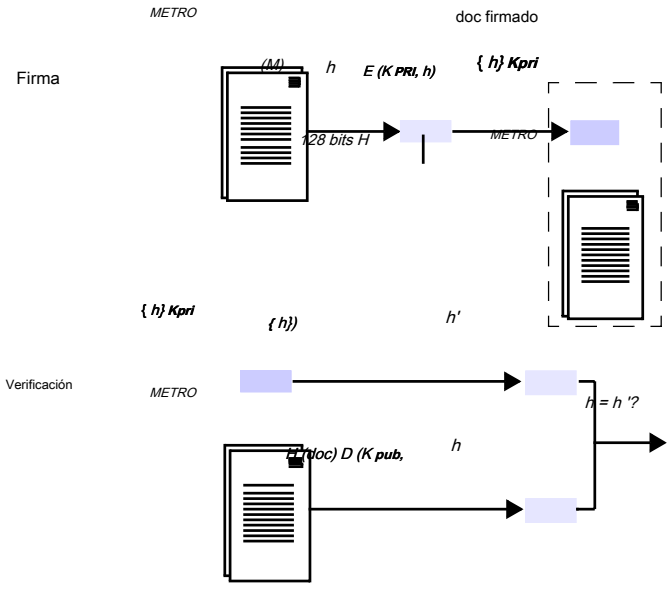
Lo que se necesita es un medio para unir de forma irrevocable la identidad del firmante a toda la secuencia de bits que representan un documento. Esto debe cumplir con el primer requisito anterior, para confirmar su autenticidad. Al igual que con las firmas manuscritas, sin embargo, la fecha de un documento no puede ser garantizada por una firma. El destinatario de un documento firmado sólo sabe que el documento fue firmado antes de que lo recibió.

En cuanto a el no repudio, hay un problema que no surge con las firmas manuscritas. ¿Qué pasa si el firmante revela deliberadamente su clave privada y, posteriormente, niega haberse adherido, diciendo que hay otros que podrían haber hecho, ya que la clave privada no era? Algunos protocolos se han desarrollado para hacer frente a este problema bajo el epígrafe de *firmas digitales innegables [*Schneier 1996], sino que aumentan considerablemente la complejidad.

Un documento con una firma digital puede ser considerablemente más resistente a la falsificación que uno escrito a mano. Pero la palabra 'original' no tiene mucho sentido con referencia a los documentos digitales. Como veremos en nuestra discusión de las necesidades del comercio electrónico, firmas digitales solos no pueden, por ejemplo, evitar la doble gasto de dinero electrónico - se necesitan otras medidas para evitarlo. Ahora se describen dos técnicas para firmar digitalmente documentos, encuadernación identidad de un director para el documento. Ambos dependen del uso de la criptografía.

• La firma digital Un documento electrónico o mensaje *METRO* puede ser firmado por un director *UN* mediante el cifrado de una copia de *METRO* con una llave K_{UN} y lo conecta a una copia de texto claro de *METRO* y *UN*'S identificador. El documento firmado a continuación, consiste en: $METRO, A, [M]_{KA}$. La firma puede ser

Figura 11.10 Las firmas digitales con claves públicas



verificada por un director que posteriormente recibe el documento para comprobar que se originó por *UN* y que su contenido, *METRO,* posteriormente no se han alterado.

Si se utiliza una clave secreta para cifrar el documento, sólo los directores que comparten el secreto puede verificar la firma. Pero si se utiliza la criptografía de clave pública, a continuación, el firmante utiliza su clave privada y cualquier persona que tenga la clave pública correspondiente puede verificar la firma. Esta es una mejor análogo para las firmas convencionales y cumple con una amplia gama de necesidades de los usuarios. La verificación de firmas produce de manera diferente dependiendo de si la criptografía de clave secreta o de clave pública se utiliza para producir la firma. Se describen los dos casos en las Secciones 11.4.1 y 11.4.2.

• **funciones Digest** Digest funciones también se denominan *funciones hash seguras* y denotado *H (M).* Ellos deben ser diseñados cuidadosamente para asegurar que *H (M)* es diferente de *H (M ')* para todos los pares posibles de mensajes *METRO* y *METRO'*. Si hay algunos pares de diferentes mensajes *METRO* y *METRO'* de tal manera que *H (M) = H (M ')*, a continuación, un director de duplicidad podría enviar una copia firmada de *METRO,* pero cuando se enfrentan a afirmar que se *METRO'* fue originalmente enviado y que debe haber sido alterado durante la transmisión. Se discuten algunas funciones hash seguras en la Sección 11.4.3.

11.4.1 Firmas digitales con claves públicas

criptografía de clave pública está particularmente bien adaptado para la generación de firmas digitales, ya que es relativamente simple y no requiere ninguna comunicación entre el receptor de un documento firmado y el firmante o terceros.

El método para *UN* para firmar un mensaje *METRO* y *segundo* para verificar que es la siguiente (y se ilustra gráficamente en la figura 11.10):

1. *UN* genera un par de claves *Kpub* y *Kpriv* y publica la clave pública *Kpub* por colocándolo en una localización bien conocida.

2. *UN* calcula el compendio de *M, H (M)* utilizando una función hash seguro acordado *H* y cifra con la clave privada *Kpriv* para producir la firma $S = \{H(M)\}_{Kpriv}$.

3. *UN* envía el mensaje firmado $[METRO]_{K} = SRA$ a *SEGUNDO.*

4. *segundo* desencripta *S* utilizando *Kpub* y calcula el compendio de *M, H (M).* Si coinciden, el firma es válida.

El algoritmo RSA es muy adecuado para su uso en la construcción de las firmas digitales. Tenga en cuenta que la *privado* clave del firmante se utiliza para cifrar la firma, en contraste con el uso del receptor de *público* clave para el cifrado cuando el objetivo es transmitir la información en secreto. La explicación de esta diferencia es sencillo - una firma debe crearse con un secreto que sólo conoce el firmante y que debe ser accesible a todos para su verificación.

11.4.2 firmas digitales con claves secretas - MAC

No hay ninguna razón técnica por un algoritmo de cifrado de clave secreta no debe ser utilizado para cifrar una firma digital, pero con el fin de verificar dichas firmas debe ser revelada la clave, y esto causa algunos problemas:

- El firmante debe hacer los arreglos para el verificador para recibir la clave secreta utilizada para la firma de forma segura.

- Puede que sea necesario para verificar una firma en varios contextos y en diferentes momentos - en el momento de la firma, el firmante no puede conocer las identidades de los verificadores. Para resolver esto, la verificación podría delegarse a un tercero de confianza que mantiene las claves secretas de todos los firmantes, pero esto añade complejidad al modelo de seguridad y requiere una comunicación segura con la tercera parte de confianza.

- La divulgación de una clave secreta utilizada para la firma no es deseable, ya que debilita la seguridad de las firmas hechas con esa clave - una firma podría ser forjado por un poseedor de la clave que no es el dueño de la misma.

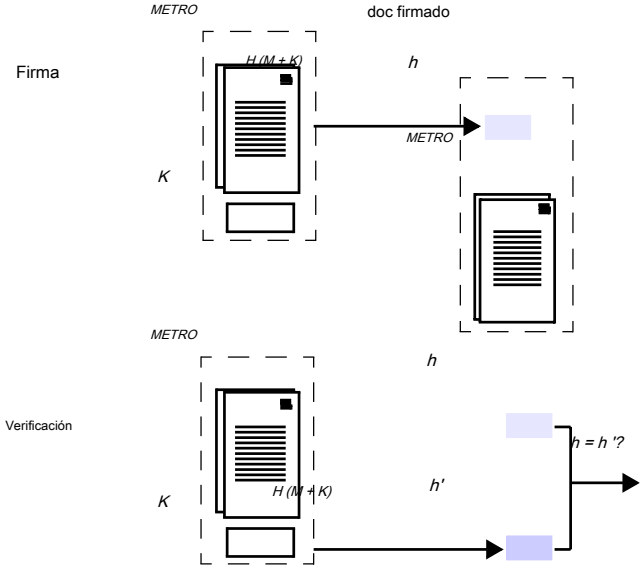
Por todas estas razones, el método de clave pública para la generación y verificación de firmas ofrece la solución más conveniente en la mayoría de las situaciones.

Una excepción surge cuando un canal seguro se utiliza para transmitir mensajes no cifrados, pero hay una necesidad de verificar la autenticidad de los mensajes. Desde un canal seguro proporciona una comunicación segura entre un par de procesos, una clave secreta compartida se puede establecer utilizando el método híbrido descrito en la Sección 11.3.3 y utilizado para producir firmas de bajo costo. Estas firmas se denominan *códigos de autenticación de mensajes* (MAC) para reflejar su propósito más limitado - se autentican la comunicación entre pares de directores sobre la base de un secreto compartido.

Una técnica de firma de bajo costo basado en claves secretas compartidas que tiene la seguridad adecuada para muchos propósitos se ilustra en la figura 11.11 y describe a continuación. El método depende de la existencia de un canal seguro a través del cual se puede distribuir la clave compartida:

1. *UN* genera una clave aleatoria *K* para la firma y la distribuye utilizando canales seguros a uno o más directores que van a necesitar para autenticar los mensajes recibidos desde *A.* Los directores son *de confianza* no revelar la clave compartida.

Figura 11.11 firmas de bajo costo con una clave secreta compartida



2. Para cualquier documento *METRO* ese *UN* desea firmar, *UN* concatena *METRO* con *K*, computa el producto de digestión del resultado, *h* = *HMK* + , y envía el documento firmado *METRO* + *M h* a cualquiera que desee verificar la firma. (El digesto *h* es un MAC.) *K* no se ve comprometida por la divulgación de *h,* ya que la función hash ha oscurecido totalmente su valor.

3. El receptor, *SEGUNDO,* concatena la clave secreta *K* con el documento recibido *METRO* y calcula el digesto *h'* = *HMK* + . La firma se verifica si *h* = *h'* .

Aunque este método adolece de las desventajas enumeradas anteriormente, que tiene una ventaja de rendimiento, ya que no implica ningún cifrado. (Hashing Secure es típicamente de aproximadamente 3-10 veces más rápido que el cifrado simétrico; véase la Sección 11.5.1.) El protocolo de canal seguro TLS describe en la Sección 11.6.3 apoya el uso de una amplia variedad de MACs, incluyendo el esquema descrito aquí. El método también se utiliza en el protocolo de dinero electrónico Millicent descrito en www.cdk5.net/security , Donde es importante mantener el bajo costo de procesamiento de transacciones de bajo valor.

11.4.3 funciones seguros Digest

Hay muchas maneras de producir un patrón de bits de longitud fija que caracteriza a un mensaje de longitud arbitraria o documento. Tal vez el más simple es utilizar la operación XOR iterativamente para combinar piezas de longitud fija del documento de origen. Tal función se utiliza a menudo en los protocolos de comunicación para producir un corto de hash de longitud fija para caracterizar un mensaje para los propósitos de detección de errores, pero es insuficiente como base para

un esquema de firma digital. Una función digesto seguro *h = H (M)* debe tener las siguientes propiedades:

1. Teniendo en cuenta *METRO,* es fácil de calcular *h.*
2. Teniendo en cuenta *h,* es difícil calcular *METRO.*
3. Teniendo en cuenta *METRO,* es difícil encontrar otro mensaje *METRO',* de tal manera que *H (M) = H (M ').*

Tales funciones también se llaman *funciones hash unidireccionales.* La razón de este nombre es evidente por sí misma sobre la base de las dos primeras propiedades. Establecimiento de 3 exige una característica adicional: a pesar de que sabemos que el resultado de una función hash no puede ser único (porque la digestión es una transformación de la información reductoras), tenemos que estar seguros de que un atacante, da un mensaje *METRO* que produce un hash *h,* no puede descubrir otro mensaje *METRO'* que también produce *h.* Si un atacante *podría* hacer esto, entonces podrían falsificar un documento firmado *METRO'*

sin el conocimiento de la clave de firma mediante la copia de la firma del documento firmado *METRO* y añadiendo a *METRO'.*

Es cierto que el conjunto de mensajes que hash al mismo valor está restringido y el atacante tendría dificultades en la producción de una falsificación significativa, pero con paciencia se podía hacer, por lo que debe tener vigilancia en contra. La viabilidad de este modo se mejora considerablemente en el caso de una llamada *ataque de cumpleaños:*

1. Alice prepara dos versiones, *METRO* y *METRO',* de un contrato de Bob. *METRO* es favorable a Bob y *METRO'* no es.
2. Alice hace varias versiones sutilmente diferentes de ambos *METRO* y *METRO'* que son visualmente indistinguibles entre sí por métodos tales como la adición de los espacios en los extremos de líneas. Ella compara los hashes de todo el *METRO* s con toda la *METRO'* s. Si se encuentra con dos que son el mismo, se puede proceder al siguiente paso; si no, se va a producir versiones visualmente indistinguibles de los dos documentos hasta que se obtiene una coincidencia.
3. Cuando ella tiene un par de documentos *METRO* y *METRO'* que hash al mismo valor, se da el documento favorables *METRO* a Bob para que firme con una firma digital utilizando su clave privada. Cuando regresa, ella sustituye la versión coincidente desfavorable *METRO',* conservando la firma de *METRO.*

Si nuestros valores hash son 64 bits de longitud, se requiere solamente 2^{32} versiones de *METRO* y *METRO'* de media. Esto es demasiado pequeña para una mayor comodidad. Tenemos que hacer que nuestros valores hash al menos 128 bits de largo para protegerse contra este tipo de ataque.

El ataque se basa en una paradoja estadística conocida como el *paradoja del cumpleaños* - el probabilidad de encontrar un par coincidente en un conjunto dado es mucho mayor que para encontrar una coincidencia para un individuo dado. Stallings [2005] da la derivación estadística de la probabilidad de que habrá dos personas con la misma fecha de nacimiento en un conjunto de *norte* gente. El resultado es que para un conjunto de sólo 23 personas lo más probable es uniforme, mientras que se requiere un conjunto de 253 personas de la misma probabilidad de que habrá uno con un cumpleaños en un día determinado.

Para satisfacer las propiedades mencionadas anteriormente, una función compendio seguro tiene que ser diseñado cuidadosamente. Las operaciones a nivel de bits utilizados y su secuenciación son similares a los encontrados en la criptografía simétrica, pero en este caso las operaciones no tienen por qué ser la información de preservación, ya que la función es, sin duda no pretende ser reversible. Por lo tanto una función de compendio seguro puede hacer uso de toda la gama de operaciones aritméticas y lógicas bit a bit. La longitud del texto de origen generalmente se incluye en los datos digeridos.

Figura 11.12 formato de certificado X509

Tema	Nombre Distinguido, de clave pública
Editor	Nombre Distinguido, Firma
Periodo de validez	No antes de la fecha, no después de la fecha
Información administrativa	Versión, número de serie
información ampliada	

Dos funciones digerir ampliamente utilizados para aplicaciones prácticas son el algoritmo MD5 (llamado así porque es el quinto de una secuencia de resumen de mensaje algoritmos desarrollados por Ron Rivest) y SHA-1 (Secure Hash Algorithm), que ha sido adoptado para la normalización de el Instituto Nacional de Estándares y Tecnología (NIST). Ambos han sido cuidadosamente probados y analizados y se puede considerar de manera adecuada conseguir en el futuro previsible, mientras que sus implementaciones son razonablemente eficiente. Los describimos brevemente aquí. Schneier [1996] y Mitchell *et al. [*1992] encuesta técnicas de firma digital y de resumen de mensaje funciones en profundidad.

• MD5 El algoritmo MD5 [1992a Rivest] utiliza cuatro rondas, cada aplicación de una de las cuatro funciones no lineales a cada uno de 16 segmentos de 32 bits de un bloque de 512 bits de texto de origen. El resultado es un resumen de 128 bits. MD5 es uno de los algoritmos más eficientes disponibles en la actualidad.

SHA-1 • SHA-1 [NIST 2002] es un algoritmo que produce una de 160 bits. Se basa en el algoritmo MD4 de Rivest (que es similar a MD5), con algunas operaciones adicionales. Es considerablemente más lento que MD5, pero el de 160 bits es que ofrece una mayor seguridad contra la fuerza bruta y ataques de estilo cumpleaños. algoritmos SHA que entregan digiere más largos (224, 256 y 512 bits) también se incluyen en la norma [NIST 2002]. Por supuesto, su longitud adicional implica costos adicionales para la generación, almacenamiento y comunicaciones de la firma digital y MAC, pero a raíz de la publicación de los ataques a los predecesores de SHA-1, lo que sugiere que SHA-1 es vulnerable [Randall y Szydlo 2004], el NIST anunció que va a ser sustituida por el SCS ya digerir las versiones de software de gobierno de Estados Unidos para el año 2010 [NIST 2004].

El uso de un algoritmo de cifrado para hacer un digesto • Es posible utilizar un algoritmo de cifrado simétrico, tal como los que se detallan en la Sección 11.3.1 para producir un digesto seguro. En este caso, la clave debe ser publicado para que el algoritmo de compendio puede ser aplicado por cualquier persona que desee verificar una firma digital. El algoritmo de cifrado se utiliza en modo CBC, y el producto de digestión es el resultado de combinar el valor CBC penúltima con el bloque de cifrado final.

11.4.4 Certificado de normas y autoridades de certificación

X.509 es el formato estándar más ampliamente utilizado para los certificados [CCITT 1988b]. Aunque el formato de certificado X.509 es una parte del estándar X.500 para la construcción de directorios globales de nombres y atributos [1988a CCITT], que se utiliza comúnmente en el trabajo criptográfico como una definición de formato de los certificados independientes. Se describe la

X.500 norma de denominación en el capítulo 13.

La estructura y contenido de un certificado X.509 se ilustran en la figura 11.12. Como puede verse, se une una clave pública a una entidad llamada llama *tema*. La unión es en la firma, que se emite por otra entidad llamada llama *editor*. El certificado tiene una *Periodo de validez,* que se define por un par de fechas. los

<Nombre de reconocimiento> las entradas están destinados a ser el nombre de una persona, organización u otra entidad, junto con la suficiente información contextual para que sea único. En una completa aplicación X.500 esta información contextual se extrae de una jerarquía de directorios en la que aparece el nombre entidad, pero en ausencia de implementaciones X.500 globales sólo puede ser una cadena descriptiva.

Este formato está incluido en el protocolo TLS para el comercio electrónico y es ampliamente utilizado en la práctica para autenticar las claves públicas de servicios y sus clientes. Ciertas empresas y organizaciones bien conocidas han establecido para actuar como *(autoridades de certificación* por ejemplo, Verisign [www.verisign.com] Y CREN [www.cren.net]), Y otras compañías e individuos pueden obtener X.509 certificados de clave pública a partir de ellas mediante la presentación de pruebas satisfactorias de su identidad. Esto conduce a un procedimiento de verificación de dos pasos para cualquier certificado X.509:

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

2. validar la firma.

• **El enfoque SPKI** El enfoque X.509 se basa en la singularidad mundial de nombres distinguidos. Se ha señalado que este es un objetivo práctico que no refleja la realidad de la práctica legal y comercial actual [Ellison 1996], en el que las identidades de los individuos no se supone que son únicas, pero se hacen único por referencia a otras personas y organizaciones. Esto se puede ver en el uso de un permiso de conducir o una carta de un banco para autenticar el nombre de un individuo y la dirección (un nombre por sí solo es poco probable que sea único entre la población mundial). Esto conduce a cadenas más largas de verificación, porque hay muchas posibles emisores de certificados de clave pública, y sus firmas deben ser validados a través de una cadena de verificación que conduce de nuevo a alguien conocido y de confianza por el director de la realización de la verificación.

Los argumentos anteriores son la base para la Infraestructura de Clave Pública simple desarrollado recientemente (SPKI) propuestas (ver RFC 2693 [Ellison *et al.* 1999]). Este es un esquema para la creación y gestión de conjuntos de certificados públicos. Permite a cadenas de certificados para ser procesados usando inferencia lógica para producir certificados derivados. Por ejemplo, 'Bob cree que la clave pública de Alice es $K_{Un\ pub}$ 'Y 'Carol confía en Bob teclas de Alice' implica' Carol cree que la clave pública de Alice es $K_{Un\ pub}$ '.

11.5 pragmática criptografía

En la Sección 11.5.1, se compara el rendimiento de los algoritmos de cifrado y hash seguro descritos o mencionados anteriormente. Consideramos algoritmos de cifrado junto con funciones hash seguras ya que el cifrado también se puede utilizar como un método para la firma digital.

Figura 11.13 El rendimiento de cifrado simétrico y asegure digerir algoritmos

	tamaño de la clave / tamaño de hash (bits)	*PRB optimizado 90 MHz Pentium 1 (Mbytes / s)*	*Crypto ++ 2.1 GHz Pentium 4 (Mbytes / s)*
TÉ	128	-	23.801
DES	56	2.113	21.340
Triple-DES	112	0,775	9,848
IDEA	128	1.219	18.963
AES	128	-	61.010
AES	192	-	53.145
AES	256	-	48.229
MD5	128	17.025	216.674
SHA-1	160	-	67.977

En la Sección 11.5.2, se discuten algunos aspectos no técnicos que rodean el uso de la criptografía. No hay espacio para hacer justicia a la gran cantidad de discusión política que ha tenido lugar sobre este tema desde algoritmos criptográficos fuertes aparecieron por primera vez en el dominio público, ni se han alcanzado los debates todavía muchas conclusiones definitivas. Nuestro objetivo es simplemente para dar al lector una conciencia de este debate en curso.

11.5.1 rendimiento de los algoritmos criptográficos

La figura 11.13 compara las velocidades de los algoritmos de cifrado simétrico y seguro digerir funciones que hemos discutido en este capítulo. Cuando esté disponible, nos damos dos mediciones de velocidad. En la columna etiquetada *PRB optimizado* damos cifras basadas en los publicados por Preneel *et al. [*1998]. Las cifras en la columna marcada *Crypto ++* se obtuvieron mucho más recientemente por los autores de la Crypto ++ biblioteca de código abierto en sistemas criptográficos [www.cryptopp.com]. Los títulos de las columnas indican la velocidad del hardware utilizado para estos puntos de referencia. Las implementaciones de PRB eran programas de ensamblador handoptimized mientras que los del Crypto ++ eran programas de C ++ generadas con un compilador de optimización.

Las longitudes de clave dan una indicación del coste computacional de un ataque de fuerza bruta en la llave; la verdadera fuerza de los algoritmos criptográficos es mucho más difícil de evaluar y se basa en el razonamiento sobre el éxito del algoritmo en oscurecer el texto sin formato. Preneel *et al.* proporcionar una útil discusión sobre la fuerza y el rendimiento de los principales algoritmos simétricos.

¿Qué tienen estas cifras de rendimiento significan para las aplicaciones reales de la criptografía, como su uso en el régimen TLS para las interacciones web seguras (la *https* protocolo, descrito en la Sección 11.6.3)? Las páginas Web son rara vez de más de 100 kilobytes, por lo que el contenido de una página se pueden cifrar utilizando cualquiera de los algoritmos simétricos en unos pocos milisegundos, incluso con un procesador que es muy lento para los estándares de hoy en día. RSA se utiliza principalmente para las firmas digitales, y que el paso también se puede realizar en unos pocos milisegundos. Así, el impacto de rendimiento de los algoritmos de la velocidad percibida del

https aplicación es mínima.

Los algoritmos asimétricos como RSA rara vez se utilizan para el cifrado de datos, pero su rendimiento para la firma es de interés. El Crypto ++ páginas de biblioteca indican que con el hardware mencionado en la última columna de la figura 11.13 se tarda unos 4,75 ms utilizando RSA con una clave de 1024 bits para firmar un hash seguro (presumiblemente utilizando 160 bits SHA-1) y alrededor de 0,18 ms a verificar la firma.

11.5.2 Aplicaciones de la criptografía y políticos obstáculos

Los algoritmos descritos, sobre todo, surgieron durante los años 1980 y 1990, cuando las redes de computadoras comenzaron a ser utilizados con fines comerciales y se hacía evidente que su falta de seguridad era un problema importante. Como mencionamos en la introducción de este capítulo, la aparición de software criptográfico fue fuertemente resistido por el gobierno de Estados Unidos. La resistencia tuvo dos fuentes: la Agencia de Seguridad Nacional de Estados Unidos (NSA), que se cree que tienen una política para restringir la fuerza de la criptografía a disposición de otras naciones a un nivel en el que la NSA podría descifrar cualquier comunicación secreta con fines de inteligencia militar;

software criptográfico se clasificó como una munición en los Estados Unidos y estaba sujeto a restricciones de exportación estrictos. Otros países, especialmente los aliados de los EE.UU., aplicados similar (o en algunos casos incluso más estrictas) restricciones. El problema se ve agravado por el desconocimiento general entre los políticos y el público en general en cuanto a qué software criptográfico era y sus potenciales aplicaciones no militares. compañías de software de Estados Unidos protestaron que las restricciones estaban impidiendo la exportación de software, tales como navegadores, y las restricciones a la exportación fueron finalmente formuladas en una forma que permitió la exportación de código usando las teclas de no más de 40 bits - apenas criptografía fuerte!

Las restricciones a la exportación pueden haber obstaculizado el crecimiento del comercio electrónico, pero no eran particularmente eficaces en la prevención de la propagación de conocimientos o criptográfico de acuerdo software criptográfico de las manos de los usuarios de otros países, ya que muchos programadores dentro y fuera de los EE.UU. estaban ansiosos y capaz de implementar y distribuir el código criptográfico. La posición actual es que el software que implementa la mayor parte de los principales algoritmos criptográficos ha estado disponible en todo el mundo durante varios años, en la impresión [Schneier 1996] y en línea, en las versiones comerciales y freeware [me www.rsasecurity.com , cryptography.org , privacy.nb.ca , www.openssl.org].

Un ejemplo es el programa llamado PGP (Pretty Good Privacy) [Garfinkel 1994, Zimmermann 1995], originalmente desarrollado por Philip Zimmermann y llevado adelante por él y otros. Esto es parte de una campaña técnico y político para garantizar que la disponibilidad de métodos criptográficos no está controlada por el gobierno de Estados Unidos. PGP se ha desarrollado y distribuido con el objetivo de permitir a todos los usuarios de la computadora para disfrutar del nivel de privacidad e integridad que ofrece el uso de la criptografía de clave pública en sus comunicaciones. PGP genera y gestiona las claves públicas y secretas en nombre de un usuario. Utiliza RSA cifrado de clave pública para la autenticación y para transmitir claves secretas con el interlocutor deseado, y utiliza el IDEA o 3DES algoritmos de cifrado de clave secreta para cifrar mensajes de correo y otros documentos. (En el momento de PGP fue desarrollado por primera vez,

ampliamente disponible en versiones libres y comerciales. Se distribuye a través de los centros de distribución independientes para los usuarios de América del Norte [www.pgp.com] Y los de otras partes del mundo [PGP Internacional] Para eludir (totalmente legal) las regulaciones de exportación de Estados Unidos.

El gobierno de Estados Unidos finalmente reconoció la inutilidad de la posición de la NSA y el daño que estaba causando a la industria de la computación de Estados Unidos (que no fue capaz de comercializar versiones seguras de los navegadores web, sistemas operativos distribuidos y muchos otros productos en todo el mundo). En enero de 2000, el gobierno de Estados Unidos introdujo una nueva política [www.bxa.doc.gov] Destinado a permitir a los proveedores de software de Estados Unidos para la exportación de software que incorpora el cifrado fuerte. Pero una barra legal fue retenido en el envío a ciertos países y los usuarios finales, ver www.rsa.com para mayor información. Por supuesto, los EE.UU. no tiene el monopolio de la producción o la publicación de software criptográfico; implementaciones de código abierto están disponibles para todos los algoritmos bien conocidos [www.cryptopp.com]. El efecto de las regulaciones es simplemente un obstáculo para la comercialización de algunos productos de software comerciales producidos en Estados Unidos.

Otras iniciativas políticas han tenido como objetivo mantener el control sobre el uso de la criptografía mediante la introducción de la legislación insistiendo en la inclusión de las lagunas o trampillas disponibles sólo para el gobierno policiales y de organismos de seguridad. Estas propuestas surgen de la percepción de que los canales de comunicación secretos pueden ser muy útiles a los criminales de todo tipo. Antes de la llegada de la criptografía digital, los gobiernos siempre tenían los medios para interceptar y analizar las comunicaciones entre los miembros del público. La criptografía digital de fuerte altera radicalmente esa situación. Pero estas propuestas para legislar para evitar el uso de la criptografía fuerte, uncompromized han sido fuertemente resistido por los ciudadanos y los órganos de las libertades civiles, que se preocupan por su impacto en los derechos de privacidad de los ciudadanos. Hasta el momento, ninguna de estas propuestas legislativas ha sido adoptado,

11.6 Estudios de casos: Needham-Schroeder, Kerberos, TLS, 802.11 WiFi

Los protocolos de autenticación publicadas originalmente por Needham y Schroeder [1978] están en el corazón de muchas técnicas de seguridad. Los presentamos en detalle en la Sección 11.6.1. Una de las aplicaciones más importantes de su protocolo de clave secreta de autenticación es el sistema Kerberos [Neuman y Ts'o 1994], que es el tema de nuestro segundo estudio de caso (Sección 11.6.2). Kerberos fue diseñado para proporcionar autenticación entre clientes y servidores en las redes que forman un único dominio de gestión (intranets).

Nuestro tercer estudio de caso (Sección 11.6.3) trata el protocolo Transport Layer Security (TLS). Este fue diseñado específicamente para satisfacer la necesidad de transacciones electrónicas seguras. Ahora es apoyada por la mayoría de los navegadores web y servidores, y se emplea en la mayoría de las transacciones comerciales que se realizan a través de la Web.

Nuestro último estudio de caso (Sección 11.6.4) ilustra la dificultad de la ingeniería de sistemas seguros. El estándar WiFi IEEE 802.11 fue publicado en 1999 con una especificación de seguridad incluido. Pero el análisis y los ataques posteriores han demostrado la especificación a ser severamente inadecuada. Identificamos los puntos débiles y las relacionamos con los principios criptográficos se tratan en este capítulo.

protocolo de autenticación 11.6.1 El Needham-Schroeder

Los protocolos descritos aquí se desarrollaron en respuesta a la necesidad de un medio seguro para gestionar las claves y contraseñas () en una red. En el momento de la publicación del trabajo [Needham y Schroeder 1978], los servicios de archivos de red acababan de aparecer y había una necesidad urgente de mejores formas de gestionar la seguridad en las redes locales.

En las redes que están integrados con fines de gestión, esta necesidad puede satisfacerse mediante un servicio clave segura que emite las claves de sesión en forma de retos (ver Sección 11.2.2). Ese es el propósito del protocolo de clave secreta desarrollada por Needham y Schroeder. En el mismo documento, Needham y Schroeder también establecen un protocolo basado en el uso de claves públicas para la autenticación y distribución de claves que no depende de la existencia de los servidores de claves seguras y es por lo tanto más adecuado para su uso en redes con muchos dominios de gestión independientes , tales como Internet. No describimos la versión de clave pública aquí, pero el protocolo TLS se describe en la Sección 11.6.3 es una variación de la misma.

Needham y Schroeder proponen una solución para la autenticación y distribución de claves basado en una *servidor de autenticación* que suministra las claves secretas a los clientes. El trabajo del servidor de autenticación es proporcionar una manera segura para pares de procesos para la obtención de claves compartidas. Para ello, debe comunicarse con sus clientes mediante mensajes cifrados.

Needham y Schroeder con claves secretas • En su modelo, un proceso que actúa en nombre de un director de A que desea iniciar una comunicación segura con otro proceso que actúa en nombre de un director de B puede obtener una clave para este propósito. El protocolo se describe por dos procesos arbitrarios A y B, pero en los sistemas cliente-servidor, A es probable que sea un cliente de iniciar una secuencia de peticiones a algún servidor B. se suministra la clave de una de dos formas: una que Una lata usar para cifrar los mensajes que envía a B y uno que puede transmitir de forma segura a B. (Este último están codificados en una clave que se sabe que B pero no a a, de modo que B puede descifrarlo y la clave no se vea comprometida durante la transmisión).

El servidor de autenticación S mantiene una tabla que contiene un nombre y una clave secreta para cada principal conocida por el sistema. La clave secreta se utiliza sólo para autenticar los procesos de cliente al servidor de autenticación y para transmitir mensajes de forma segura entre los procesos cliente y el servidor de autenticación. Nunca se da a conocer a terceros y se transmite a través de la red de a lo sumo una vez, cuando se genera. (Idealmente, una clave siempre debe ser transmitida por algunos otros medios, tales como en papel o en un mensaje verbal, evitando cualquier exposición en la red). Una clave secreta es el equivalente de la contraseña utilizada para autenticar a los usuarios en los sistemas centralizados. Por componentes humanos, el nombre en poder del servicio de autenticación es su nombre de usuario y la clave secreta es su contraseña.

El protocolo se basa en la generación y transmisión de boletos por el servidor de autenticación. Un billete es un mensaje cifrado que contiene una clave secreta para su uso en la comunicación entre A y B. tabulamos los mensajes en el protocolo de clave secreta Needham y Schroeder en la figura 11.14. El servidor de autenticación es S.

norte UN y *norte segundo* son *nonces*. Un nonce es un valor entero que se añade a un mensaje para demostrar su frescura. Nonces se utilizan sólo una vez y se generan bajo demanda. Por ejemplo, los valores de uso único se pueden generar como una secuencia de valores enteros o mediante la lectura del reloj en la máquina de envío.

Si el protocolo se completa con éxito, tanto A como B pueden estar seguros de que cualquier mensaje cifrado en K_{AB} que reciben viene del otro, y que cualquier mensaje

Figura 11.14 El protocolo de autenticación de clave secreta Needham-Schroeder

Encabezamiento	*Mensaje*	*notas*
1. AS:	A, B, N_{UN}	A solicita S para el suministro de una clave para la comunicación con B.
2. S UN:	$\{ norte_{UN}, B, K_{AB}, \{ K_{AB}, UN \}_{KB} \}_{KA}$	S devuelve un mensaje cifrado en la clave secreta del A, que contiene una clave recién generada K_{AB}, y un 'billete' encriptada en la clave secreta del B. el nonce $norte_{UN}$ demuestra que el mensaje fue enviado en respuesta a la precedente. A cree que S envió el mensaje porque sólo S conoce la clave secreta del A.
3. AB:	$\{ K_{AB}, UN \}_{KB}$	A envía el billete a B.
4. BA:	$\{ norte_{B} \}_{KAB}$	B descifra el vale y utiliza la nueva clave, K_{AB}, para cifrar otro valor de uso único, $norte_{SEGUNDO}$.
5. AB:	$\{ norte_{segundo} - 1 \}_{KAB}$	Un demuestra a B que era el remitente del mensaje anterior mediante la devolución de una transformación de acuerdo $norte_{SEGUNDO}$.

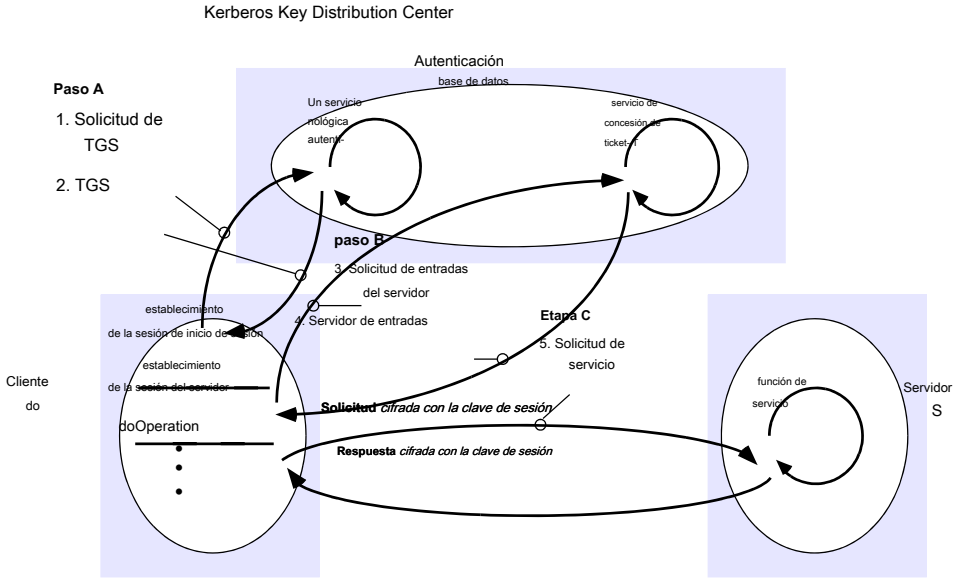
cifrada en K_{AB} que envían sólo puede entenderse por la otra o por S (y S se supone que es digno de confianza). Esto es así porque los únicos mensajes que se han enviado contiene K_{AB} fueron cifrados en la clave secreta del A o B de la clave secreta.

Hay una debilidad en este protocolo en el que B no tiene razones para creer que el mensaje 3 es fresco. Un intruso que se las arregla para obtener la clave K_{AB} y hacer una copia del billete y el autenticador $\{ K_{AB}, A \}_{KB}$ (ambos de los cuales podrían haber sido dejado en un lugar de almacenamiento expuesta por un un programa cliente no se ejecuta bajo la autoridad de un descuido o) se puede utilizar para iniciar un intercambio posterior con B, hacerse pasar por A. Para este ataque que se produzca un valor antiguo de K_{AB} tiene que ser comprometida; en la terminología de hoy, Needham y Schroeder no incluyen esta posibilidad en su lista de amenaza, y la opinión general es que hay que hacerlo. La debilidad puede remediarse mediante la adición de un nonce o marca de tiempo para el mensaje 3, por lo que se convierte en: $\{ K_{AB}, A \}_{KBpub}$. B descifra este mensaje y comprueba que t es reciente. Esta es la solución adoptada en Kerberos.

11.6.2 Kerberos

Kerberos fue desarrollado en el MIT en los años 1980 [Steiner *et al.* 1988] para proporcionar una gama de servicios de autenticación y seguridad para su uso en la red informática del campus del MIT y otras intranets. Ha sido objeto de varias revisiones y mejoras a la luz de la experiencia y la retroalimentación de las organizaciones de usuarios. Kerberos versión 5 [Neuman y Ts'o 1994], que se describe aquí, es un estándar de Internet (ver RFC 4120 [Neuman *et al.* 2005]) y es utilizado por muchas empresas y organizaciones. El código fuente de una implementación de Kerberos del MIT está disponible [me web.mit.edu]; que está incluido en el OSF Distributed Computing Environment (DCE) [OSF 1997] y como el servicio de autenticación predeterminado en Microsoft Windows [www.microsoft.com II]. Una extensión

Figura 11.15 arquitectura del sistema de Kerberos



se incluyó para incorporar el uso de certificados de clave pública para la autentificación inicial de directores (Paso A en la figura 11.15) [Neuman *et al.* 1999].

Figura 11.15 muestra la arquitectura de proceso. Kerberos se ocupa de tres tipos de objeto de seguridad:

Boleto: Un token emitido a un cliente por el servicio de concesión de vales Kerberos para la presentación a un servidor en particular, la verificación de que el remitente recientemente ha sido autenticado por Kerberos. Los boletos incluyen un tiempo de caducidad y una clave de sesión recién generada para su uso por el cliente y el servidor.

autenticador: Un token construido por un cliente y se envía a un servidor para probar la identidad del usuario y la moneda de cualquier comunicación con el servidor. Un autenticador se puede utilizar sólo una vez. Contiene el nombre del cliente y una marca de tiempo y está encriptada en la clave de sesión correspondiente.

clave de la sesión: Una clave secreta generada aleatoriamente por Kerberos y emitido a un cliente para su uso en la comunicación con un servidor determinado. El cifrado no es obligatorio para todas las comunicaciones con los servidores; la clave de sesión se utiliza para cifrar la comunicación con los servidores que lo demanden y para cifrar todos los autenticadores (véase más arriba).

Los procesos de cliente deben poseer un billete y una clave de sesión para cada servidor que utilizan. No sería práctico para suministrar un nuevo billete y la clave para cada interacción cliente-servidor, por lo que la mayoría de los boletos se conceden a los clientes con una duración de varias horas para que puedan ser utilizados para la interacción con un servidor en particular hasta que expiren.

Un servidor de Kerberos es conocido como un centro de distribución de claves (KDC). Cada KDC ofrece un servicio de autenticación (AS) y un servicio de concesión de vales (TGS). Al iniciar la sesión,

los usuarios se autentican por el AS, usando una variación de la red segura del método de contraseña, y el proceso de cliente que actúa en nombre del usuario se suministra con una *billete ticketgranting* y una clave de sesión para la comunicación con el TGS. Posteriormente, el proceso del cliente original y sus descendientes pueden utilizar el ticket de acceso para obtener entradas y claves de sesión para servicios específicos de la TGS.

El [1978] protocolo Needham y Schroeder es seguida muy de cerca en Kerberos, con valores de tiempo (números enteros que representan una fecha y hora) utilizado como nonces. Esto tiene dos propósitos:

- para protegerse de repetición de los mensajes antiguos interceptados en la red o la reutilización de billetes viejos que se encuentran situadas en la memoria de las máquinas de la cual el usuario autorizado ha cerrado la sesión (nonces se utilizaron para lograr este propósito en Needham y Schroeder);

- aplicar una vida para billetes, permitiendo que el sistema de revocar los derechos de los usuarios cuando, por ejemplo, dejan de ser usuarios del sistema autorizado.

A continuación se describen los protocolos de Kerberos en detalle, usando la notación definida en la parte inferior de la página. En primer lugar, se describe el protocolo mediante el cual el cliente obtiene un boleto y una clave de sesión para acceder al TGS.

Un billete de Kerberos tiene un período fijo de tiempo de validez a partir de las $t1$ y terminando en el momento $t2$. Un billete para un cliente C para acceder a un servidor de S toma la forma:

$$CS\ t1\ t2\ K_{CS}\ K_S$$, que denotamos como $billete\ CS\ K_S$

el nombre del cliente está incluido en el billete para evitar su posible uso por impostores, como veremos más adelante. Los números de los pasos y mensaje de la Figura 11,15 corresponden a los de la descripción tabulada *A*. Tenga en cuenta que el mensaje 1 no está cifrada y no incluye la contraseña del C. Contiene un valor de uso único que se utiliza para comprobar la validez de la respuesta.

A. *Obtener clave de sesión Kerberos y TGS, una vez por sesión de inicio de sesión de cabecera*

	Mensaje	*notas*
1. CA Solicitud de TGS	*C, T, n*	Client C pide al servidor de autenticación Kerberos A para suministrar un billete para la comunicación con el T. servicio de concesión de vales
2. AC: TGS clave de sesión y el boleto	$\{K_{CONNECTICUT,\ norte}\}_{KC}, \{billete\ (C,\ T)\}_{KT}$ que contiene $C, T, t1, t2, K_{CT}$	Un devuelve un mensaje que contiene un billete de cifrado en su clave secreta y una clave de sesión para C para su uso con T. La inclusión de la nonce *norte* cifrada en K_{do} muestra que el mensaje proviene del destinatario del mensaje 1, que debe conocer K_{DO}.

Notación:

UN Nombre del servicio de autenticación Kerberos.
T Nombre del servicio Kerberos de concesión de vales.
do Nombre del cliente.
norte Un valor de uso único.
t Una marca de tiempo.
$t1$ tiempo de validez del billete de partida.
$t2$ tiempo de validez del billete fin.

Mensaje 2 veces se llama un 'desafío' porque presenta el solicitante con información que sólo es útil si se conoce la clave secreta del C, K_{DO}. Un impostor que intenta hacerse pasar por C mediante el envío de mensajes 1 no puede ir más, ya que no pueden descifrar el mensaje 2. Para los directores que son usuarios, K_{do} es una versión mezclada de la contraseña del usuario. El proceso cliente le pedirá al usuario que escriba su contraseña y tratará de descifrar el mensaje 2 con ella. Si el usuario da la contraseña correcta, el proceso cliente obtiene la clave de sesión $K_{Connecticut}$ y un billete válido para el servicio de concesión de vales; si no, se obtiene un galimatías. Los servidores tienen claves secretas de su propia, que sólo conoce el proceso de servidor correspondiente y al servidor de autenticación.

Cuando un billete válido se ha obtenido del servicio de autenticación, el cliente C puede usarlo para comunicarse con el servicio de concesión de vales para obtener entradas para otros servidores de cualquier número de veces hasta que el boleto expira. Por lo tanto para obtener un boleto para cualquier servidor

S, C construye un autenticador cifrado en $K_{Connecticut}$ de la forma:

$\{ C, t\}_{KCT,}$ que denotamos como $\{ auth\ (C)\}_{KCT,}$ y envía una solicitud a T:

SEGUNDO. *Obtener entradas para un servidor S, una vez por sesión cliente-servidor*

3. CT: Billete de solicitud de servicio S	$\{ auth\ (C)\}$ KCT, $\{ billete\ (C, T)\}$ KT, S, n	C solicita al servidor T vale de concesión para suministrar un billete para la comunicación con otro servidor, S.
4. TC: Venta de entradas	$\{ KCS, n\}$ KCT, $\{billete\ (C, S)\}$ KS	T verifica el billete. Si se trata de T válido genera una nueva clave de sesión aleatoria, *KCS,* y lo devuelve con un billete de S (cifrado en la clave secreta del servidor, *KANSAS).*

C es entonces listo para emitir mensajes de petición al servidor, S:

DO. *Emitir una solicitud del servidor con un billete*

Solicitud de servicios: 5. CS	$\{ auth\ (C)\}_{KCS,}$ $\{ billete\ (C, S)\}_{KS,}$ *solicitud, n*	C envía el billete de S con un autenticador recién generado para C y una petición. La solicitud se puede cifrar en K_{CS} si se requiere confidencialidad de los datos.

Para el cliente para asegurarse de la autenticidad del servidor, S debe devolver el valor de uso único *norte* a C (para reducir el número de mensajes necesarios, esto podría incluirse en los mensajes que contienen la respuesta del servidor a la solicitud):

RE. *Autenticar servidor (opcional)*

6. SC: La autenticación del servidor	$\{ norte\}_{KCS}$	(Opcional): S envía el nonce a C, cifrada en K_{CS}.

• Aplicación de Kerberos Kerberos fue desarrollado para su uso en el Proyecto Athena en el MIT - una instalación informática en red en todo el campus de la educación universitaria con muchas estaciones de trabajo y servidores que proporcionan un servicio a más de 5000 usuarios. El entorno es tal que ni la fiabilidad de los clientes ni la seguridad de la red y las máquinas que ofrecen servicios de red pueden ser asumidas - por ejemplo, las estaciones de trabajo no están protegidos contra la instalación del software del sistema desarrolladas por el usuario, y las máquinas de servidor (que no sea el servidor Kerberos) no están necesariamente asegurado contra interferencia física con su configuración de software.

Kerberos proporciona prácticamente la totalidad de la seguridad en el sistema de Athena. Se utiliza para autenticar usuarios y otros directores. La mayor parte de los servidores que se ejecutan en la red se han extendido a requerir un billete de cada cliente en el inicio de cada interacción con el cliente-servidor. Estos incluyen el almacenamiento de archivos (NFS y Andrew File System), correo electrónico, acceso remoto y la impresión. las contraseñas de los usuarios se conocen sólo para el usuario y para el servicio de autenticación Kerberos. Los servicios tienen claves secretas que sólo se conocen a Kerberos y los servidores que proporcionan el servicio.

Se describe aquí el modo en que se aplica Kerberos para la autentificación de usuarios en inicio de sesión. Su uso para asegurar el servicio de archivos NFS se describe en el capítulo 12.

Sesión con Kerberos • Cuando un usuario inicia sesión en una estación de trabajo, el programa de inicio de sesión envía el nombre del usuario al servicio de autenticación Kerberos. Si el usuario es conocido en el servicio de autenticación, responde con una clave de sesión, un nonce encriptado de la contraseña del usuario y una entrada para el TGS. El programa de login a continuación, intenta descifrar la clave de sesión y el valor de uso único usando la contraseña que el usuario escribió en respuesta a la solicitud de contraseña. Si la contraseña es correcta, el programa de inicio de sesión obtiene la clave de sesión y el valor de uso único. Se comprueba el valor de uso único y almacena la clave de sesión con el billete para su posterior uso en la comunicación con el TGS. En este punto, el programa de inicio de sesión puede borrar la contraseña del usuario desde su memoria, ya que el billete ahora sirve para autenticar al usuario. A continuación se inicia una sesión de inicio de sesión del usuario en la estación de trabajo.

Acceso a los servidores con Kerberos • Cada vez que un programa que se ejecuta en una estación de trabajo necesita tener acceso a un nuevo servicio, solicita un billete para el servicio del servicio de otorgamiento de tickets. Por ejemplo, cuando un usuario UNIX desea iniciar sesión en un equipo remoto, el *rlogin* programa de comandos en el equipo del usuario obtiene un billete desde el servicio ticketgranting Kerberos para el acceso a la *rlogind* servicio de red. los *rlogin* programa de comando envía el billete, junto con un nuevo autenticador, en una solicitud a la *rlogind* proceso en el equipo donde el usuario desea conectarse. La *rlogind* programa descifra el boleto con el *rlogin* la clave secreta del servicio y comprueba la validez del billete (es decir, que toda la vida del billete no ha expirado). máquinas de servidores deben tener cuidado para almacenar sus claves secretas en el almacenamiento que es inaccesible a los intrusos.

los *rlogind* programa utiliza la clave de sesión incluido en el boleto para descifrar el autenticador y comprueba que el autenticador es fresco (autenticadores pueden usarse sólo una vez). Una vez el *rlogind* programa está convencido de que el billete y el autenticador son válidos, no hay necesidad de ella para comprobar el nombre del usuario y la contraseña, porque la identidad del usuario es conocido en el *rlogind* programa y una sesión de inicio se establece para ese usuario en la máquina remota.

Implementación de Kerberos • Kerberos se implementa como un servidor que se ejecuta en una máquina segura. Se proporciona un conjunto de bibliotecas para el uso de aplicaciones y servicios al cliente. Se utiliza el algoritmo de cifrado DES, pero esto se implementa como un módulo separado que puede ser reemplazado fácilmente.

El servicio Kerberos es escalable - el mundo se divide en dominios separados de autoridad de autenticación, llamadas *reinos,* cada uno con su propio servidor de Kerberos. La mayoría de los directores están registrados en un solo reino, pero los servidores de concesión de vales Kerberos se registran en todos los reinos. Los directores pueden autenticarse en servidores de otros reinos a través de su servidor de otorgamiento de tickets local.

Dentro de un mismo ámbito, no puede haber varios servidores de autenticación, todos los cuales tienen copias de la misma base de datos de autenticación. La base de datos de autenticación se replica mediante un simple procedimiento de maestro-esclavo. Las actualizaciones se aplican a la copia maestra por un único servicio de gestión de base de datos de Kerberos (KDBM) que se ejecuta sólo en la máquina principal. El KDBM gestiona las peticiones de los usuarios cambiar sus contraseñas y las peticiones de los administradores del sistema para agregar o eliminar directores y cambiar sus contraseñas.

Para que este esquema transparente para los usuarios, la duración de los tiquets TGS debería ser tan largo como la más larga sesión de inicio es posible, ya que el uso de un billete caducado resultará en el rechazo de las solicitudes de servicio; el único remedio es para el usuario a autenticar la sesión de inicio de sesión y luego solicitar nuevas entradas de servidor para todos los servicios en uso. En la práctica, se utilizan tiempos de vida de billetes en la región de 12 horas.

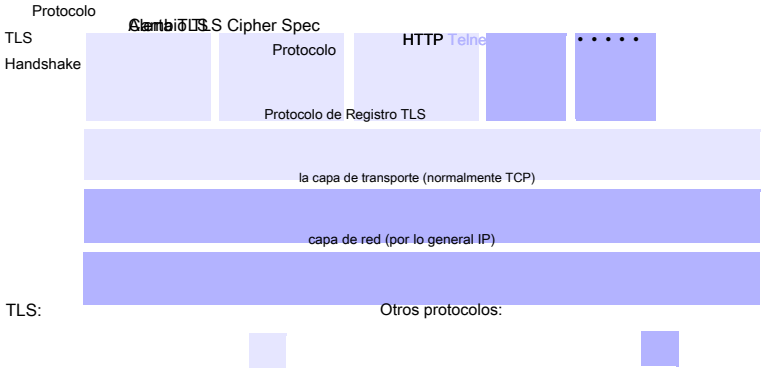
Las críticas a la Kerberos • El protocolo para Kerberos versión 5 se ha descrito anteriormente contiene varias mejoras diseñadas para hacer frente a las críticas de las versiones anteriores [Bellovin y Merritt 1990, Burrows *et al.* 1990]. La crítica más importante de la versión 4 fue que los valores de uso único utilizados en autenticadores se implementan como marcas de tiempo, y la protección contra la repetición de autenticadores dependían de sincronización al menos suelto de clientes y los servidores de relojes. Por otra parte, si un protocolo de sincronización se utiliza para llevar relojes de cliente y servidor en sincronía suelta, el protocolo de sincronización debe ser en sí mismo seguro contra ataques de seguridad. Véase el Capítulo 14 para obtener información sobre los protocolos de sincronización de reloj.

La definición de protocolo para la versión 5 permite que los valores de uso único en autenticadores a ser implementados como marcas de tiempo o como números de secuencia. En ambos casos, se requiere que sean únicos y que los servidores tienen una lista de valores de uso único recientemente recibidos de cada cliente para comprobar que no se reproducen. Este es un requisito aplicación inconveniente y es difícil para los servidores para garantizar en caso de fallas. Kehne *et al. [*1992] han publicado una propuesta de mejora para el protocolo Kerberos que no se basa en los relojes sincronizados.

La seguridad de Kerberos depende de tiempos de vida limitado para las sesiones. El período de validez de los billetes de TGS está generalmente limitada a unas pocas horas; el período debe ser elegido para ser lo suficientemente largo para evitar interrupciones inoportunas de servicio, pero lo suficientemente corto como para asegurar que los usuarios que han sido dados de baja o degradados no se siguen utilizando los recursos por más de un período corto. Esto podría causar dificultades en algunos entornos comerciales, porque la consiguiente necesidad de que el usuario para el suministro de un nuevo conjunto de datos de autentificación en un punto arbitrario en la interacción podría entrometerse en la aplicación.

Figura 11.16 pila de protocolo TLS

(Figuras 11.16 a 11.19 se basan en diagramas en Hirsch [1997] y se utilizan con el permiso de Frederick Hirsch)



11.6.3 transacciones electrónicas fijándolo con Secure Sockets

El Secure Sockets Layer (SSL) protocolo fue desarrollado originalmente por el Netscape Corporación [www.mozilla.org] Y propuesto como un estándar específicamente para satisfacer las necesidades descritas a continuación. Una versión ampliada de SSL ha sido adoptado como un estándar de Internet bajo el nombre de capa de transporte (TLS), se describe en el RFC 2246 [Dierks y Allen 1999]. TLS es apoyada por la mayoría de los navegadores y es ampliamente utilizado en el comercio por Internet. Exploramos sus principales características a continuación:

encriptación y autenticación algoritmos negociables • En una red abierta no debemos suponer que todas las partes utilizan el mismo software de cliente o de que todo el software cliente y servidor incluye un algoritmo de cifrado en particular. De hecho, las leyes de algunos países intentan restringir el uso de ciertos algoritmos de cifrado a esos países solo. TLS ha sido diseñado de manera que los algoritmos utilizados para el cifrado y la autenticación se negocian entre los procesos en los dos extremos de la conexión durante el apretón de manos inicial. Puede resultar que no tienen suficientes algoritmos en común, y en ese caso el intento de conexión fallará.

Bootstrapped comunicación segura • Para satisfacer la necesidad de una comunicación segura sin negociación anterior o ayuda de terceros, el canal seguro se establece utilizando un protocolo similar al esquema híbrido mencionado en la Sección 11.3.3. la comunicación sin cifrar se utiliza para los intercambios iniciales, a continuación, criptografía de clave pública y la criptografía de clave secreta, finalmente, una vez que se ha establecido una clave secreta compartida. Cada interruptor es opcional y precedido de una negociación.

Así, el canal seguro es totalmente configurable, permitiendo la comunicación en cada dirección para ser encriptado y autenticado, pero que no requieren, de manera que los recursos informáticos no tienen que ser consumidos en la realización de cifrado innecesario.

Los detalles del protocolo TLS se publican y estandarizado, y varias bibliotecas de software y herramientas están disponibles para apoyar eso [Hirsch 1997, www.openssl.org], Algunos de ellos en el dominio público. Se ha incorporado en una amplia gama de software de aplicación, y su seguridad ha sido verificado por la revisión independiente.

TLS consiste en dos capas (figura 11.16): TLS Record capa de protocolo, que implementa un canal seguro, el cifrado y la autenticación de los mensajes transmitidos a través de cualquier protocolo orientado a la conexión; y una capa de apretón de manos, que contiene el protocolo TLS apretón de manos y otros dos protocolos relacionados que establecen y mantienen una sesión TLS (es decir, un canal seguro) entre un cliente y un servidor. Tanto por lo general son implementados por las bibliotecas de software a nivel de aplicación en el cliente y el servidor. El protocolo de registro TLS es una capa de nivel de sesión; que puede ser utilizado para el transporte de applicationlevel datos de forma transparente entre un par de procesos garantizando al mismo tiempo su carácter secreto, la integridad y la autenticidad. Estas son exactamente las propiedades que especifican para los canales seguros en nuestro modelo de seguridad (Sección 2.4.3), pero en TLS hay opciones para las parejas que se comunican a elegir si desea o no desplegar el descifrado y autenticación de mensajes en cada dirección. Cada sesión segura se da un identificador, y cada socio puede almacenar identificadores de sesión en una memoria caché para su posterior reutilización, evitando la sobrecarga de establecer una nueva sesión cuando se requiere otra sesión segura con la misma pareja.

TLS es ampliamente utilizado para añadir una capa de comunicación segura por debajo de los protocolos de nivel de aplicación existentes. Es probablemente el más ampliamente usado para asegurar las interacciones HTTP para su uso en el comercio por Internet y otras aplicaciones sensibles a la seguridad. Está implementado por prácticamente todos los navegadores y servidores web: el uso del prefijo de protocolo *https:* en las direcciones URL inicia el establecimiento de un canal seguro TLS entre un navegador y un servidor web. También ha sido ampliamente utilizados para proporcionar implementaciones seguras de Telnet, FTP y muchos otros protocolos de aplicación. TLS es el *de facto* estándar para el uso en aplicaciones que requieren canales seguros; hay una amplia variedad de implementaciones disponibles, tanto comerciales como de dominio público, con las API de CORBA y Java.

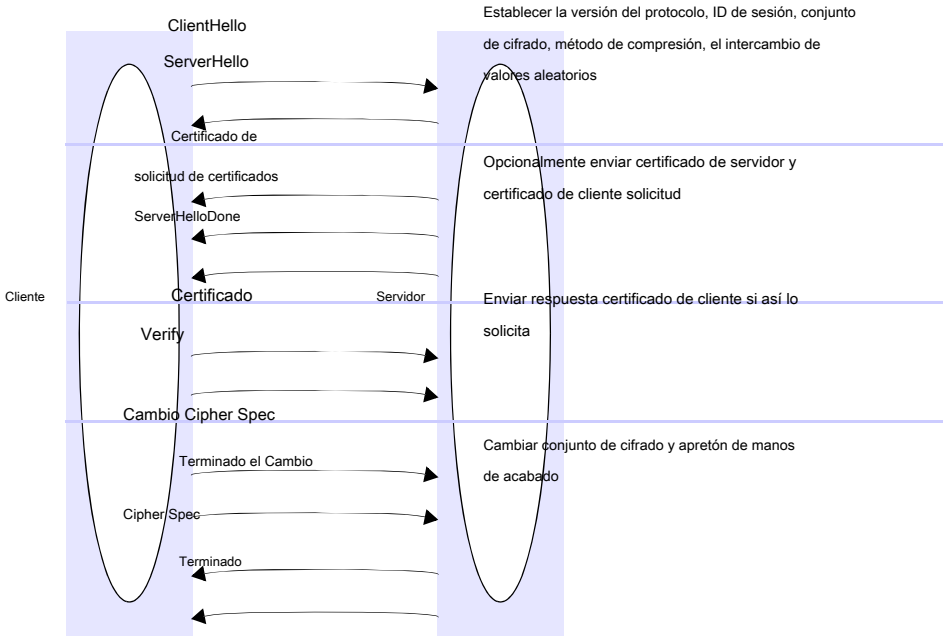
El protocolo de enlace TLS se ilustra en la figura 11.17. El apretón de manos se realiza a través de una conexión existente. Se inicia en la clara y establece una sesión TLS mediante el intercambio de las opciones y parámetros acordados necesarios para realizar el cifrado y la autenticación. La secuencia apretón de manos varía dependiendo de si se requiere autenticación de cliente y servidor. El protocolo de enlace también puede ser invocada en un momento posterior para cambiar la especificación de un canal seguro - por ejemplo, la comunicación puede comenzar con la autenticación de mensajes usando solamente códigos de autenticación de mensaje, y en un punto posterior, el cifrado puede ser añadido. Esto se consigue mediante la realización del protocolo de enlace de nuevo para negociar una nueva especificación de cifrado utilizando el canal existente.

El apretón de manos inicial TLS es potencialmente vulnerable a man-in-the-middle, tal como se describe en la Sección 11.2.2, Escenario 3. Para protegerse contra ellos, la clave pública utilizada para verificar el primer certificado recibido puede ser entregado por un canal separado - por ejemplo, los navegadores de Internet y otros programas entregados en un CD-ROM pueden incluir un conjunto de claves públicas para algunas autoridades de certificación conocidos. Otra defensa para los clientes de los servicios conocidos se basa en la inclusión del nombre de dominio del servicio en sus certificados de clave pública - los clientes sólo deben tratar con el servicio a la dirección IP correspondiente a ese nombre de dominio.

TLS es compatible con una variedad de opciones para las funciones de cifrado a utilizar. Estos son conocidos colectivamente como una *conjunto de cifrado*. Un conjunto de cifrado incluye una única opción para cada uno de las características mostradas en la figura 11.18.

Una variedad de paquetes de cifrado populares están precargados, con identificadores estándar en el cliente y el servidor. Durante el apretón de manos, el servidor ofrece al cliente una lista de los identificadores de conjuntos de cifrado que tiene a su disposición, y el cliente responde seleccionando uno de ellos (o

Figura 11.17 protocolo de enlace TLS



dando una indicación de error si no tiene ninguno que coincidan). En esta etapa también están de acuerdo en un método de compresión (opcional) y un valor inicial aleatorio para funciones de cifrado de bloque CBC (véase la Sección 11.3).

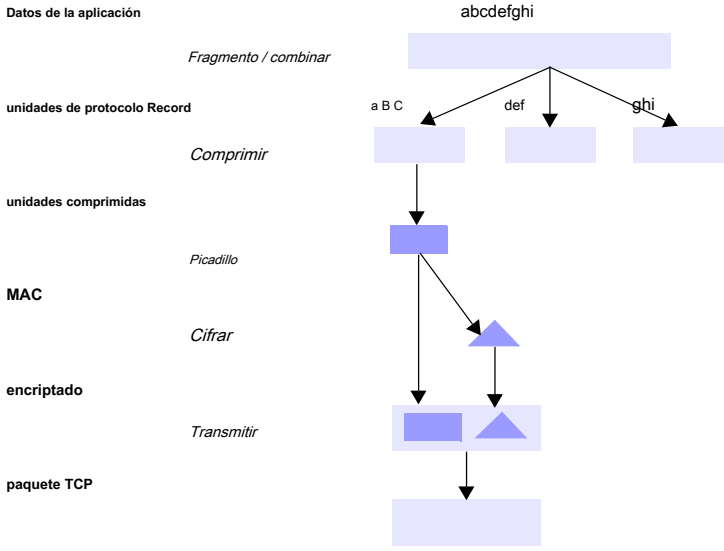
A continuación, los socios se autentican entre sí opcionalmente mediante el intercambio de certificados publickey firmados en formato X.509. Estos certificados se pueden obtener de una autoridad de clave pública o simplemente pueden ser generados temporalmente para el propósito. En cualquier caso, al menos una clave pública debe estar disponible para su uso en la siguiente etapa del apretón de manos.

Una pareja, genera una *secreto pre-maestro* y lo envía a la otra parte cifrada con la clave pública. Un secreto pre-maestro es un valor aleatorio grande que es utilizado por ambos socios para generar las dos claves de sesión (llamado *escribir* teclas) para el cifrado de datos

Figura 11.18 TLS opciones de configuración apretón de manos

Componente	*Descripción*	*Ejemplo*
método de intercambio de claves	El método que se utilizará para el intercambio de una clave de sesión	RSA con certificados de clave pública
Cifrado para transferencia de datos	El bloque o corriente de cifra que se utiliza para los datos	IDEA
función de compendio de mensajes	Para la creación de códigos de autenticación de mensajes (MAC)	SHA-1

Figura 11.19 protocolo de registro de TLS



en cada dirección y los secretos de autenticación de mensajes que se utilizará para la autenticación de mensajes. Cuando todo esto se ha hecho, se inicia una sesión segura. Esto es provocado por el *ChangeCipherSpec* mensajes intercambiados entre los socios. Estos son seguidos por *Terminado* mensajes. Una vez el *Terminado* Se han intercambiado mensajes, toda comunicación posterior se encriptado y firmado de acuerdo con el conjunto de cifrado elegido con las teclas acordados.

Figura 11.19 muestra el funcionamiento del protocolo de grabación. Un mensaje para la transmisión se fragmenta primero en bloques de un tamaño manejable, a continuación, los bloques se comprimen opcionalmente. La compresión no es estrictamente una función de comunicación segura, pero se proporciona aquí porque un algoritmo de compresión puede compartir de forma eficaz una parte del trabajo de procesar los datos en bloque con el cifrado y algoritmos de firma digital. En otras palabras, una tubería de transformaciones de datos se puede configurar dentro de la capa de registro TLS que llevará a cabo todas las transformaciones necesarias más eficiente que se podría hacer de forma independiente.

El cifrado y la autenticación de mensajes (MAC) transformaciones implementar los algoritmos especificados en el conjunto de cifrado acordada exactamente como se describe en las Secciones 11.3.1 y 11.4.2. Por último, el bloque firmado y cifrado se transmite a la pareja a través de la conexión TCP asociado, donde las transformaciones se invierten para producir el bloque de datos original.

Resumen • TLS proporciona una implementación práctica de un esquema de cifrado híbrido con la autenticación y el intercambio de claves basado en claves públicas. Debido a que los sistemas de cifrado se negocian en el apretón de manos, que no depende de la disponibilidad de los algoritmos particulares. Tampoco depende de ningún servicio de seguridad en el momento de la sesión

establecimiento. El único requisito es que los certificados de clave pública son emitidos por una autoridad reconocida por ambas partes.

Debido a que la base de SSL para TLS y su implementación de referencia se publicaron [www.mozilla.org], Que fue objeto de revisión y debate. Algunas modificaciones se hicieron a principios de los diseños, y fue ampliamente aceptada como norma de valor. TLS está ahora integrado en la mayoría de los navegadores web y servidores web y se utiliza en otras aplicaciones tales como Telnet y FTP seguro. Comercial y de dominio público [me www.rsasecurity.com , Hirsch 1997, www.openssl.org] Implementaciones están ampliamente disponibles en forma de bibliotecas y los complementos del navegador.

11.6.4 Las deficiencias en el diseño de la seguridad WiFi IEEE 802.11 original

El estándar IEEE 802.11 para redes LAN inalámbricas descritas en la Sección 3.5.2 primero fue lanzado en 1999 [IEEE 1999]. Se llevó a cabo en las estaciones base, ordenadores portátiles y dispositivos portátiles a partir de una fecha similar y ampliamente utilizado para la comunicación móvil. Desafortunadamente, el diseño de la seguridad en el estándar fue encontrado posteriormente ser severamente inadecuada en varios aspectos. Describimos que el diseño inicial y sus debilidades como un estudio de caso en las dificultades de diseño de seguridad ya que se hace referencia en la Sección 11.1.3.

Se reconoció que las redes inalámbricas son por su naturaleza más expuesto al ataque de las redes de cable, porque el los datos transmitidos a la red y están disponibles para el espionaje y haciéndose pasar por cualquier dispositivo equipado con un transmisor / receptor dentro del alcance. El diseño inicial 802.11, por tanto, como objetivo proporcionar control de acceso para redes WiFi y la privacidad e integridad de los datos transmitidos en ellos a través de una especificación de seguridad titulado Wired Equivalent Privacy (WEP), que incorpora las siguientes medidas, que opcionalmente pueden ser activados por un administrador de red :

Control de acceso por un protocolo de desafío-respuesta (*cf.* Kerberos, Sección 11.6.2), en el que un nodo que se une es desafiada por la estación base para demostrar que tiene la clave compartida correcta. Una clave única, *K,* es asignado por un administrador de red y compartido entre la estación base y todos los dispositivos autorizados.

Privacidad e integridad el uso de un mecanismo de cifrado opcional basado en el cifrado de flujo RC4. La misma clave, *K,* utilizados para el control de acceso también se utiliza en el cifrado. Hay opciones de longitud de clave de 40, 64 o 128 bits. Una suma de comprobación encriptada está incluido en cada paquete para proteger su integridad.

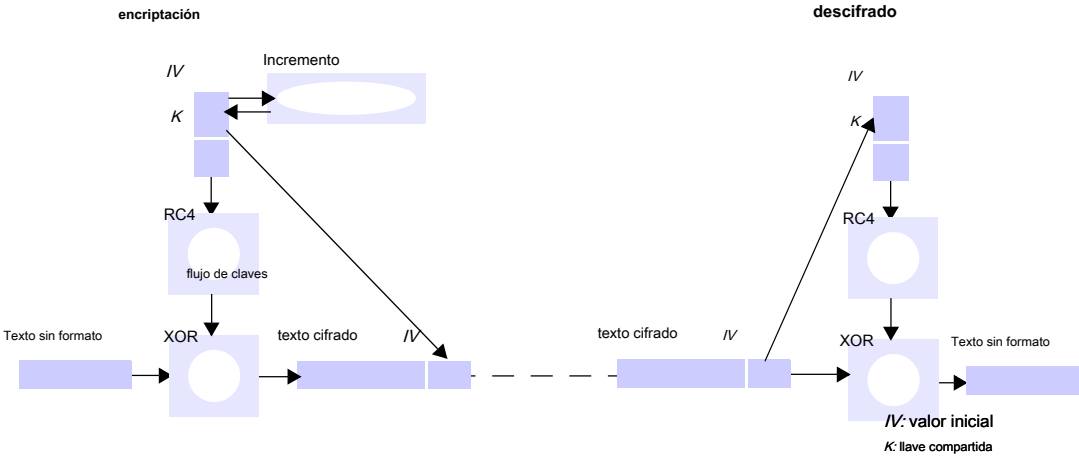
Los siguientes deficiencias y debilidades de diseño fueron descubiertos poco después de la norma fue desplegado:

La puesta en común de una sola tecla por todos los usuarios de una red hace que el diseño débil en la práctica, ya que:

- La clave es susceptible de ser transmitida a los nuevos usuarios en los canales no protegidos.
- Un usuario malintencionado descuidado o sola (como un ex empleado descontento) que ha obtenido acceso a la clave puede poner en peligro la seguridad de toda la red, y esto puede no ser descubierto.

Solución: Utilizar un protocolo de clave pública basado en la negociación de claves individuales, como se hace en TLS / SSL (ver Sección 11.6.3).

Figura 11.20 El uso de cifrado de flujo RC4 en IEEE 802.11 WEP



Las estaciones de base no se autentican, por lo que un atacante que conoce la clave compartida actual podría introducir una estación base parodia y escuchar a escondidas, insertar o manipular cualquier tráfico.

Solución: Las estaciones de base deben suministrar un certificado que se puede autenticar mediante el uso de una clave pública obtenida de un tercero.

uso inadecuado de un cifrado de flujo en vez de un cifrado de bloques (ver las descripciones de bloque y corriente a sistemas de cifrado en la Sección 11.3). Figura 11.20 muestra el proceso de cifrado y decyption en 802,11 seguridad WEP. Cada paquete está cifrado por XOR-ing su contenido con un flujo de clave producida por el algoritmo RC4. La estación receptora utiliza RC4 para generar el mismo flujo de claves y descifra el paquete con otra XOR. Para evitar errores de sincronización de corriente de clave cuando los paquetes se pierden o dañan, RC4 se reanuda con un valor inicial que consiste en un 24-bit *valor inicial*

concatenado con la clave compartida a nivel mundial. El valor inicial se actualiza y se incluye (en el claro) en cada paquete transmitido. La clave compartida no puede ser fácilmente cambiado en el uso normal, por lo que el valor inicial sólo tiene $s = 2^{24}$ (o alrededor de 10^7) diferentes estados, lo que resulta en la repetición del valor de arranque y por tanto el flujo de clave, después de 10^7 los paquetes se envían. En la práctica, esto puede ocurrir dentro de unas pocas horas, y los ciclos de repetición aún más cortos puede surgir si se pierden paquetes. Un atacante recibir los paquetes cifrados siempre puede detectar repeticiones ya que el valor inicial se envía en el claro.

La especificación RC4 advierte explícitamente contra la repetición de serie de claves. Esto se debe a un atacante que recibe un paquete encriptado do_{yo} y conoce el texto sin formato PAG_{yo} (Por ejemplo, adivinar que se trata de una consulta a un servidor estándar) se puede calcular el flujo de claves K_{yo} utilizada para cifrar el paquete. El mismo valor de K_{yo} se repetirá después de s los paquetes se transmiten, y el atacante puede utilizar para descifrar el recién transmitida

paquete. El atacante puede llegar a tener éxito en el descifrado de una alta proporción de los paquetes de esta manera al adivinar los paquetes de texto plano correctamente. Esta debilidad se señaló por primera vez por Borisov *et al. [*2001] y dio lugar a una importante revalorización de seguridad WEP y su sustitución en versiones posteriores de 802.11.

Solución: Negociar un duplicado de la llave después de un tiempo menor que el peor de los casos para la repetición. sería necesario un código de terminación explícita, como es el caso de TLS.

longitudes de clave de 40 bits y 64 bits se incluyeron en el estándar para permitir que los productos para ser enviados al exterior por los proveedores de los Estados Unidos en un momento en las regulaciones del gobierno de Estados Unidos mencionadas en la Sección 11.5.2 restringidos longitudes de clave para los dispositivos exportados a 40 bits (y posteriormente 64 bits). Pero claves de 40 bits son tan fácilmente rotas por la fuerza bruta que ofrecen muy poca seguridad, e incluso claves de 64 bits son potencialmente manipulable con un ataque determinado.

Solución: Utilice únicamente claves de 128 bits. Esto ha sido adoptado en muchos productos WiFi recientes.

El cifrado de flujo RC4 se mostró, después de la publicación de la norma 802.11, de tener debilidades que permitieron la clave para ser descubierta después de la observación de una cantidad considerable de tráfico, incluso sin repetición de la secuencia de claves [Fluhrer *et al.* 2001]. Esta debilidad se demostró en la práctica. Se rindió el esquema WEP insegura incluso con claves de 128 bits y llevó a algunas empresas a prohibir el uso de redes WiFi por sus empleados.

Solución: Proporcionar para la negociación de las especificaciones de cifrado como se hace en TLS, dando una selección de algoritmos de cifrado. RC4 está cableada en el estándar WEP, con ninguna disposición para la negociación de algoritmos de cifrado.

Los usuarios a menudo no desplegó la protección ofrecida por el sistema WEP, probablemente porque no se dan cuenta de lo expuesto sus datos era. Esto no era una debilidad en el diseño de la norma, pero en la comercialización de productos basados en ella. La mayoría fueron diseñados para poner en marcha con deshabilitado la seguridad y la documentación de los riesgos de seguridad a menudo era débil.

Solución: Mejores ajustes por defecto y la documentación pueden ayudar. Los usuarios quieren obtener un rendimiento óptimo, sin embargo, y la comunicación era perceptiblemente más lento con el cifrado activado usando el hardware disponible en el momento. Los intentos de evitar el uso de cifrado WEP llevado a la adición a las estaciones base de características para la represión de normalmente transmitidos por estaciones de base las que identifican los paquetes y el rechazo de los paquetes no se ha enviado desde una dirección MAC autorizada (véase la Sección

3.5.1). Ninguno de estos tanta seguridad que ofrece, ya que una red puede ser descubierto mediante la interceptación ('sniffing') paquetes en la transmisión y las direcciones MAC se puede suplantar por modificaciones del sistema operativo.

IEEE creó un grupo de trabajo específico para crear una solución de seguridad de reemplazo y su trabajo dio lugar a un nuevo protocolo de seguridad conocido como Wi-Fi Protected Access (WPA). Esto se especifica en el proyecto IEEE 802.11i [IEEE 2004b, Edney y Arbaugh 2003], y comenzó a aparecer en los productos a mediados de 2003. IEEE 802.11i (también conocido como WPA2) en sí se ratificó en junio de 2004, y utiliza el cifrado AES, en lugar de RC4, que fue utilizado en WEP. Los desarrollos posteriores de IEEE 802.11 también incorporan WPA2.

11.7 Resumen

Las amenazas a la seguridad de los sistemas distribuidos son omnipresentes. Es esencial para proteger los canales de comunicación y las interfaces de cualquier sistema que se encarga de la información que podría ser objeto de ataques. El correo personal, el comercio electrónico y otras transacciones financieras, son ejemplos de dicha información. Los protocolos de seguridad son cuidadosamente diseñados para evitar lagunas. El diseño de los sistemas de seguridad se inicia a partir de una lista de amenazas y un conjunto de supuestos 'peor caso'.

Los mecanismos de seguridad se basan en la criptografía de clave pública y clave secreta. Los algoritmos criptográficos revuelven los mensajes de una manera que no se puede invertir sin conocer la clave de descifrado. La criptografía de clave secreta es simétrica - la misma tecla sirve para el cifrado y el descifrado. Si dos partes comparten una clave secreta, pueden intercambiar información cifrada sin riesgo de espionaje y la manipulación y con garantías de autenticidad.

criptografía de clave pública es asimétrica - teclas separadas se utilizan para el cifrado y descifrado, y el conocimiento de uno no revela la otra. Una clave es pública, permitiendo que cualquiera pueda enviar mensajes seguros al titular de la clave privada correspondiente y permitiendo que el poseedor de la clave privada para firmar los mensajes y certificados. Los certificados pueden actuar como credenciales para el uso de los recursos protegidos.

Los recursos están protegidos por mecanismos de control de acceso. los sistemas de control de acceso asignan derechos a los directores (es decir, los titulares de credenciales) para realizar operaciones en objetos distribuidos y colecciones de objetos. Los derechos se pueden mantener en las listas de control de acceso (ACL) asociadas con colecciones de objetos o pueden estar en manos de los directores en forma de capacidades - unforgeable claves para el acceso a las colecciones de recursos. Las capacidades son convenientes para la delegación de derechos de acceso, pero son difíciles de revocar. Los cambios en las ACL se aplican inmediatamente, revocar los derechos de acceso anteriores, pero son más complejos y costosos de administrar que las capacidades.

Hasta hace poco, el algoritmo de cifrado DES fue el esquema de cifrado simétrico más utilizado, pero sus claves de 56 bits ya no están a salvo de ataques de fuerza bruta. La versión triple DES da fortaleza clave de 112 bits, que es segura, pero otros algoritmos modernos (como IDEA y AES) son mucho más rápidos y ofrecer una mayor resistencia.

RSA es el esquema de cifrado asimétrico más ampliamente utilizado. Para la seguridad contra los ataques de factoraje, que debe ser usado con claves de 768 bits o superior. De clave pública (asimétrica) algoritmos son superados por los de clave secreta (simétrica) algoritmos en varios órdenes de magnitud, por lo que generalmente se utilizan sólo en protocolos híbridos, tales como TLS, para el establecimiento de canales seguros que utilizan claves compartidas para los intercambios posteriores.

El protocolo de autenticación de Needham-Schroeder fue el primero de propósito general, el protocolo de seguridad práctica, y todavía proporciona la base para muchos sistemas prácticos. Kerberos es un sistema bien diseñado para la autenticación de usuarios y la protección de los servicios dentro de una sola organización. Kerberos se basa en Needham-Schroeder y la criptografía simétrica. TLS es el protocolo de seguridad diseñado para y ampliamente utilizado en el comercio electrónico. Es un protocolo flexible para el establecimiento y utilización de canales seguros basados tanto en la criptografía simétrica y asimétrica. Los puntos débiles de la seguridad IEEE 802.11 Wi-Fi proporcionan una lección sobre las dificultades de diseño de seguridad.