Aplicaciones prácticas de la criptografía

Este artículo describe las aplicaciones fundamentales de la criptografía en la actualidad: cifrado y firma electrónica. Ambas aplicaciones son el núcleo del comercio electrónico y de cualquier transacción segura que se realice por Internet. También se introduce el concepto de certificado digital y su importancia para garantizar la seguridad, así como las aplicaciones que mayor beneficio obtienen de su utilización. Los algoritmos van acompañados de ejemplos prácticos basados en *openssl*, programa de libre distribución que está disponible para diversas arquitecturas.

Vera Delgado

Estudiante de quinto curso de la titulación de Ingeniería Informática de la Escuela Técnica Superior de Ingeniería (ICAI), Universidad Pontificia Comillas. Previsiblemente obtendrá el título oficial en junio de 2006.

Rafael Palacios

Ingeniero Industrial del ICAI (1990) y Doctor Ingeniero (1998). Es investigador del Instituto de Investigación Tecnológica y profesor del Departamento de Sistemas Informáticos de la Escuela Técnica Superior de Ingeniería (ICAI). Imparte clases de seguridad informática en la titulación de Ingeniero en Informática del ICAI.

Comentarios a:
comentarios@icai.es

Aplicaciones de cifrado

La aplicación principal de los algoritmos de cifrado es garantizar la confidencialidad de los documentos aunque estos resultasen accesibles a personas no autorizadas. La situación práctica en la que más se utiliza es la transferencia de información por canales de comunicación no seguros, como es Internet. Las técnicas de cifrado, además de garantizar la confidencialidad, garantizan colateralmente la integridad, ya que si el documento no es accesible para usuarios no autorizados tampoco es modificable.

Los algoritmos de cifrado juegan un papel decisivo en la transferencia de archivos, por ejemplo por correo electrónico, y en la transferencia de información mediante navegadores, por ejemplo durante el acceso a la página web de un banco. También se utilizan las técnicas de cifrado para proteger documentos importantes dentro del disco de duro o en cualquier medio de almacenamiento digital, por si se produce un acceso ilegal. Un acceso no autorizado a información confidencial puede tener lugar tanto por un acceso ilícito al sistema como por el robo de los medios físicos de almacenamiento (especialmente en ordenadores portátiles o *flash drive*).

Para garantizar la confidencialidad, se pueden utilizar tanto las técnicas de cifrado de tipo simétrico donde existe una clave secreta que se utiliza para cifrar y descifrar, como las técnicas de cifrado de tipo asimétrico donde existe una clave pública y otra privada. Las características principales de ambos tipos de cifrado, así como los algoritmos más utilizados han sido descritos en el artículo sobre criptografía el anterior número de la revista *Anales* [1].

El algoritmo más popular en la familia de algoritmos simétricos es Advanced Encryption Standard, AES [2], ya que es el estándar que fue seleccionado en 2002 por NIST (National Institute of Standards and Technology, USA). Mientras que el algoritmo más popular en la familia de algoritmos asimétricos es RSA [3], nombre que proviene de las iniciales de los apellidos de sus inventores (Ron Rivest, Adi Shamir y Len Adleman).

La gran ventaja de los métodos asimétricos es que permiten iniciar las comunicaciones sin tener que haber acordado previamente, y de manera segura, una clave secreta. Sin

embargo estos algoritmos son muy costosos desde el punto de vista computacional, por lo que se tiende a utilizar métodos simétricos. En la mayoría de las aplicaciones prácticas, se utilizan los métodos asimétricos para iniciar una comunicaión segura durante la cual se establece una clave secreta generada aleatoriamente y se acuerda un algoritmo de cifrado simétrico a partir de las preferencias de los dos interlocutores. A partir de ese momento la comunicación tiene lugar por medio de algoritmos simétricos que son más eficientes.

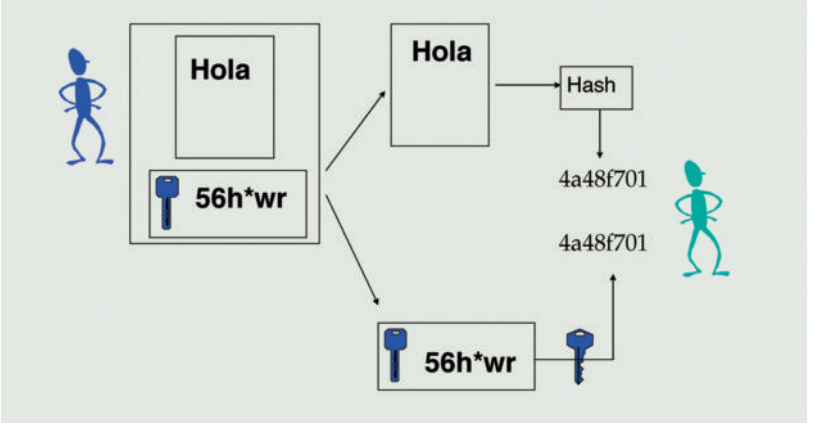
Una de las aplicaciones más extendidas de las técnicas de cifrado en la comunicación en modo seguro del **navegador de Internet**, es decir cuando el usuario pone https en lugar de http. Como se verá más adelante en este mismo artículo, al establecer una conexión https el navegador solicita la clave pública del servidor y luego se establece la comunicación mediante algoritmos simétricos. En el caso del **correo electrónico cifrado**, se sigue el estándar S/MIME que se basa en las normas PKCS#7. Análogamente se aplican cifrados simétricos con claves autogeneradas y luego dichas claves se cifran con algoritmos asimétricos.

Aplicaciones de firma electrónica

La firma electrónica se utiliza para conseguir dos de las características de seguridad: integridad y autenticación. Es importante destacar que un documento electrónico firmado puede ser público, porque integridad y autenticación no implican necesariamente confidencialidad. Por lo tanto los métodos de cifrado de clave secreta no son apropiados para realizar firma electrónica, y sin embargo los métodos asimétricos sí pueden aplicarse. Si un usuario cifra cierta información utilizando su clave privada, entonces cualquier otra persona tendrá acceso a dicha información mediante la clave pública del usuario. Dado que es imposible obtener la clave privada a partir de la clave pública y que la clave privada se mantiene segura, se garantiza la autoría (autenticación) y que nadie puede haber modificado el documento (integridad).

Un documento electrónico firmado es equivalente a un documento en papel y firmado a mano que se publica en un tablón de anuncios. La firma electrónica es uno de los aspectos más importantes de la criptografía porque permite realizar muchas transacciones

Figura 1. Verificación de la firma de un documento público



por Internet, evitando desplazamientos y pérdidas de tiempo. De hecho en España, a la firma electrónica aplicada sobre datos consignados en forma electrónica, se le otorga la equivalencia funcional con la firma manuscrita en virtud de la ley 59/2003, de 19 de diciembre, de Firma Electrónica.

Quizás la aplicación más conocida en España es la posibilidad de entregar la declaración de la renta en formato electrónico por Internet. Para ello el usuario se identifica ante el servidor web de la Agencia Tributaria mediante su certificado digital, y luego entrega el documento electrónico de la declaración firmado con su clave privada. Otro ejemplo son ciertas peticiones que se realizan mediante formularios en la web, que si se firman electrónicamente tiene la misma validez que una petición presencial. El correo electrónico también puede beneficiarse de los algoritmos de firma electrónica para poder enviar mensajes firmados que tiene la misma validez que una carta firmada, y evitando los problemas de falsificación del remite del correo[1]. Por último, un aspecto en el que se va más despacio son las transacciones bancarias, las cuales ganaría en nivel de seguridad si se realizasen firmadas digitalmente.

En la práctica no se suele cifrar toda la información del documento con la clave privada, ya que resulta muy pesado computacionalmente, sino que resulta mucho más eficiente (tanto para el emisor como para los receptores) obtener un resumen del documento mediante algoritmos HASH y luego cifrar exclusivamente el código obtenido. Finalmente se envía el documento, en principio sin cifrar, junto con unos códigos de seguridad que representan la firma. El receptor

(1) Los programas de correo electrónico más populares, Outlook y Mozilla/Firefox, soportan cifrado y firma electrónica, aunque Eudora todavía no lo soporta.

puede acceder al documento porque no está cifrado, y para verificar integridad y autenticación realiza las operaciones sintetizadas en la Figura 1. El documento enviado por el usuario azul está formado por el documento original y por la firma electrónica, que el HASH del documento cifrado con su clave privada. Entonces el receptor separa el documento de la firma y calcula el HASH del documento por un lado y obtiene el HASH original utilizando la clave pública sobre el código de la firma. Si el resultado que se obtiene por los dos caminos es el mismo significa que el documento no ha sido alterado.

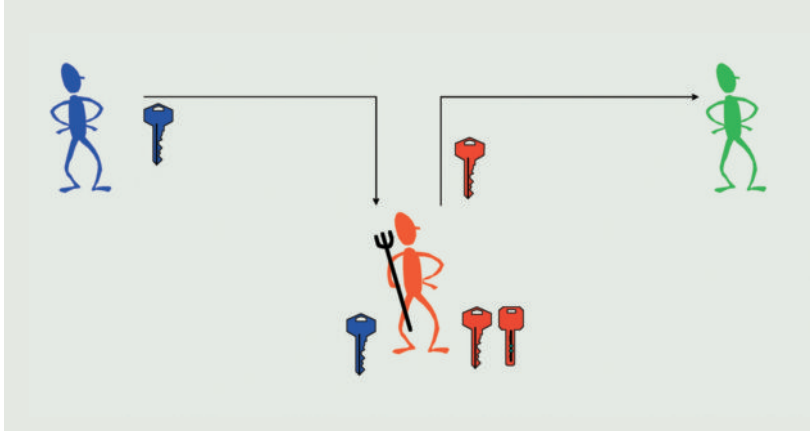
Para firmar un documento se pueden aplicar cualquiera de los algoritmos HASH y cualquiera de los algoritmos de cifrado asimétrico. Lo más utilizado es aplicar MD5 o SHA1 como función de resumen y luego RSA como algoritmo asimétrico para cifrar dicho resumen. La alternativa es utilizar el estándar de firma electrónica del NIST que se denomina DSA.

Certificados digitales

Los algoritmos asimétricos, es decir los que utilizan una clave pública y una clave privada, fueron diseñados para poder intercambiar información de manera segura sin necesidad de haber acordado previamente una clave secreta de cifrado. En efecto todo mensaje cifrado con la clave pública de un usuario sólo puede ser descifrado con la clave privada que el usuario mantiene segura.

Sin embargo existe una vulnerabilidad en este proceso que consiste en suplantar la identidad del receptor en un mecanismo de ataque conocido con el nombre de "man-in-the-middle attack" o bien MITM (ver Figura 2). Suponiendo que un atacante (representado en rojo) tiene capacidad de interceptar la comunicación entre dos usuarios, el atacante puede hacer una sustitución de claves que le dan capacidad para ver y modificar los mensajes sin que los usuarios sean conscientes de la intrusión. Según el ejemplo de la Figura, el usuario verde quiere enviar un mensaje confidencial al usuario azul, por lo que reclama su clave pública. Sin embargo la clave pública del usuario azul es interceptada por el atacante, que la almacena y la sustituye por una clave pública falsa (en rojo) que es enviada al usuario verde. Entonces los mensajes cifrados por el usuario verde con la clave pública falsa son interceptados y descifrados por el atacante, que los puede ver y modificar antes de reenviarlos al usuario azul cifrados con su clave pública auténtica. En este esquema el usuario verde piensa que está utilizando la clave pública de su destinatario cuando realmente está utilizando una clave pública falsa, y el usuario azul recibe un mensaje perfectamente cifrado con su clave pública auténtica y no detecta que el mensaje ha sido interceptado.

Figura 2. Esquema del ataque "Man-in-the-middle"



Para evitar este tipo de ataque, que haría perder toda la utilizada de los algoritmos asimétricos, es imprescindible que cada usuario tenga un mecanismo para verificar si las claves públicas de los demás usuarios son reales o falsas.

Un mecanismo para garantizar la autenticidad de las claves públicas es tomarlas de servidores de claves públicas en lugar de hacerlo directamente del destinatario del mensaje. Este mecanismo dificulta considerablemente el ataque "man-in-the-middle" ya que además de interceptar la comunicación entre emisor y receptor habría que interceptar la comunicación entre el emisor y el repositorio de claves. Este último canal de comunicación puede ser muy difícil de romper, especialmente si existen varios servidores de claves que además pueden estar incluidos en la Intranet del emisor.

Otro mecanismo para reforzar la seguridad de las claves públicas es apoyarse en una tercera figura, independiente del emisor y del receptor, que se encarga de firmar electrónicamente las claves. A esta nueva figura se la denomina Entidad de Certificación y su función principal es certificar que una clave es válida y pertenece a una determinada persona. La implementación práctica de este mecanismo se realiza guardando la clave pública junto con cierta información adicional (nombre del usuario, fechas de validad, número de serie…) dentro de un certificado

digital según el estándar X.509, y solicitar la firma electrónica de dicho certificado por parte de la entidad de certificación. La entidad de certificación verifica los datos de la persona y entonces firma el certificado, es decir, que calcula un resumen del certificado mediante SHA1 o MD5 y lo cifra con su propia clave privada (que es el activo mejor guardado de la compañía). Además, la entidad de certificación mantiene una lista de certificados que han sido revocados, es decir certificados que han sido anulados a petición de sus propietarios, por ejemplo en caso detectar el robo del certificado.

Un certificado digital que esté firmado por una entidad de certificación se puede verificar en cualquier momento utilizando la clave pública de la entidad de certificación y consultando las listas de revocación. La mayoría de los navegadores de Internet y de los programas de correo incorporan de forma segura las claves públicas de entidades de certificación de reconocido prestigio como VeriSign, Thawte, Entrust y FNMT (Fábrica Nacional de Moneda y Timbre de España).

En definitiva, la distribución de claves públicas por medio de certificados digitales firmados es un mecanismo seguro y que permite realizar las siguientes operaciones prácticas:

- **Cifrar comunicaciones**. La aplicación más extendida es la comunicación segura con servidores web (por ejemplo el banco) cuando se utiliza https. En este caso el navegador solicita un certificado digital al banco, y este debe presentar un certificado firmado por una de las entidades reconocidas por el navegador. Entonces se establece una comunicación cifrada asimétrica para acordar una clave secreta y un algoritmo de cifrado y a partir de ese momento se establece una comunicación cifrada simétrica[2].
- **Firmar mensajes y documentos**. Si un usuario dispone de un certificado digital instalado en su programa de correo, puede enviar mensajes de correo firmados electrónicamente. Esto permite a los destinatarios verificar la autenticidad del correo y por lo tanto se obtiene la garantía de que nadie ha suplantado el remite del emisor. Dado que suplantar el remite del correo electrónico resulta especialmente sencillo debido a la baja seguridad de los protocolos utilizados, esta aplicación resulta especialmente interesante.
- **Identificación ante un sistema o autenticación de usuarios**. Todavía se suele utilizar un nombre de usuario y una clave para identificarse ante muchos sistemas informáticos, como por ejemplo el banco por Internet. Estos nombres de usuario y contraseñas en modo texto son bastante vulnerables y la suplantación del usuario sería inmediata. Alternativamente los servidores web pueden exigir un certificado digital al usuario para comprobar su identidad, lo cual resulta mucho más seguro.

Al combinar la capacidad de firmar información con la posibilidad de identificar usuarios de manera segura, aparece la posibilidad de realizar transacciones por Internet que tradicionalmente se realizaban de manera presencial. Quizás el ejemplo más conocido en España es la posibilidad de entregar por Internet la declaración de Renta.

Cada vez es mayor el número de administraciones españolas que admiten certificados personales firmados por la Fabrica Nacional de Moneda y Timbre para realizar trámites por Internet. El sistema se ha visto promovido por la Agencia Tributaria (http://www.aeat.es/) del Ministerio de Economía y Hacienda, para permitir la entrega de la declaración de la Renta por Internet. Posteriormente otros organismos han ido incorporando servicios accesibles mediante certificados de usuario, por ejemplo la Seguridad Social (http://www.segsocial.es/), Ayuntamiento de Madrid (http://www.munimadrid.es/), Ministerio de Fomento (http://www.fomento.es/).

Los certificados digitales podrían evitar muchos de los problemas que se producen actualmente de suplantación de identidad por Internet. Hay dos técnicas, cada vez más extendidas, para conseguir obtener el nombre de usuario y la clave de acceso de una persona: *keyloggers y phishing*. Los primeros son programas que se auto-instalan en el ordenador (como si fuesen un virus) y registran las pulsaciones del teclado del usuario. Dado que el nombre de usuario y la clave son cadenas de texto, es sencillo para el *keylogger* detectarlas y enviarlas por Internet al atacante. La técnica de *phishing* consiste en engañar al usuario, normalmente mediante un correo electrónico, para que introduzca sus datos en una página web. Dicha página ha sido creada por el atacante imitando la página principal de un banco. La utilización

(2) Algunos navegadores, por ejemplo Firefox, muestran el algoritmo de cifrado simétrico que se está utilizando en una comunicación segura en protocolo https. Basta pulsar sobre el candado de seguridad para obtener esta información y la información de certificado del servidor. En Internet Explorer esta información aparece en Archivo/Propiedades.

del *phishing* se ha duplicado entre octubre de 2004 y octubre de 2005 según la organización Anti-Phishing Working Group (http://www.antiphishing.org/).

Ejemplos de utilización de los algoritmos

El proyecto *openssl* [4] ha desarrollado un sistema criptográfico de carácter general, basado en software abierto y de uso libre tanto para aplicaciones comerciales como no comerciales. Los programas ejecutables están disponibles para sistemas basados en Unix y Windows. A continuación se muestran ejemplos de utilización de los algoritmos criptográficos mediante el comando *openssl*:

Cifrado simétrico

El programa *openssl* soporta de manera estándar muchos algoritmos de cifrado, incluidas diferentes variantes de cada uno. El comando necesario para cifrar con el algoritmo DES el documento texto.txt y generar texto_cifrado.txt es el siguiente:

```
openssl des -in texto.txt -out texto_cifrado.txt
-pass pass:clave
```

Para utilizar otro algoritmo basta cambiar "des" por el nombre del algoritmo deseado (des3, rc2, rc4, rc5, idea, bf, aes-256-ecb, etc.). Para descifrar el archivo se utiliza el mismo comando, pero añadiendo la opción *–d* (descifrar), como por ejemplo:

```
openssl des –d -in texto_cifrado.txt -out
texto_original.txt -pass pass:clave
```

Cifrado asimétrico

Los cifrados asimétricos mediante el algoritmo RSA se pueden realizar con el comando *rsautl* de *openssl*. Es muy sencillo de utilizar si disponemos de un certificado que incluya tanto la clave pública como la clave privada, es decir el típico archivo que se obtiene al exportar certificados desde el navegador. En los siguientes ejemplos supondremos que tenemos el certificado en formato PEM en el archivo *cert.pem*.

PARA CIFRAR:

```
openssl rsautl -in texto.txt -out texto.rsa -inkey
cert.pem -certin -encrypt
```

PARA DESCIFRAR:

```
openssl rsautl -in texto.rsa -out texto_original.txt
-inkey cert.pem -certin -decrypt
```

En cada caso se utiliza la parte necesaria del certificado, es decir la clave pública para cifrar y la clave privada para descifrar. Alternativamente se pueden proporcionar archivos tipo PEM, que según el caso sólo contengan la parte pública o la parte privada, entonces se omite la opción *-certin*.

A diferencia del cifrado simétrico, el comando *rsautl* no admite textos largos, tan sólo de unos 100 caracteres. Esto es debido a que normalmente no se cifran grandes documentos mediante algoritmos asimétricos. Como ya se ha comentado los documentos se cifran con algoritmos simétricos sólo la clave secreta se cifra con algoritmos asimétricos. Una manera directa para cifrar un documento real es utilizar el comando *smime* que realiza el cifrado simétrico mediante des, 3des, rc2 o aes y luego protege la clave secreta utilizando la clave pública del destinatario:

```
openssl smime -encrypt -aes128 -in bigfile.doc
-out bigfile.msg user.pem
```

Este comando genera un archivo de acuerdo al estándar S/MIME que se utiliza típicamente en el correo electrónico y que contiene la información de acuerdo a PKCS#7 (el nombre MIME del contenido es "application/x-pkcs7-mime").

La obtención del documento original se obtiene con el argumento *–decrypt* del comando *smime* utilizando el certificado completo del destinatario, que contiene su clave privada:

```
>openssl smime -decrypt -in bigfile.msg -recip
mymit.pem -inkey mymit.pem -out original.doc
Enter pass phrase for mymit.pem: xxxxxxx
```

Conversión de certificados

Existen varios formatos para almacenar los certificados en archivos, pero openssl proporciona comandos para convertir el formato de dichos archivos. Esto es útil ya que algunas aplicaciones requieren un formato específico para los certificados.

Hoy en día todos los certificados siguen el estándar X.509 de ITU-T (International Telecommunication Union-Telecommunication Standardization Sector). Sin embargo los certificados pueden estar contenidos en otras estructuras como son PKCS#7 y PKCS#12 (Public Key Cryptography Standards definidos por RSA Data Security Inc). A su vez

todos estos formatos pueden almacenarse en archivos binarios (formato DER) o en archivos de texto[3] (formato PEM). Las extensiones de archivos más utilizadas son las siguientes:

- *.crt cer* o *.der* para certificados X.509 puros en formato DER.
- *.pem* para certificados X.509 puros en formato PEM.
- *.p7c .p7b .p7s .p7m* … para certificados X.509 dentro de una estructura PKCS#7 en formato DER o PEM.
- *.p12* o *.pfx* para certificados X.509 dentro de una estructura PKCS#12 en formato DER.

El estándar PKCS#7 sólo puede contener certificados públicos (que contienen la clave pública, pero no la clave privada) y los archivos *crt, cer, der* y *pem* también se suelen utilizar exclusivamente para almacenar claves públicas.

Normalmente se utilizan archivos p12 para almacenar en un único archivo cifrado la clave pública y la clave privada que definen la identidad de una persona. La extensión alternativa, pfx, se deriva de Personal inFormation Exchange.

El comando ***x509*** de *openssl* permite convertir entre PEM y DER cualquier certificado público tipo X.509, consultar sus datos o firmar el certificado. Esto último es algo que nunca realiza un usuario final sino que es una de las funciones de las entidades de certificación. Para convertir un certificado de formato DER a PEM, por ejemplo fnmt_root.crt, se utiliza el siguiente comando (la conversión inversa es análoga):

```
openssl x509 –inform DER –in fnmt_root.crt
–outform PEM –out fnmt_root.pem
```

Este comando es útil, por ejemplo, para poder incorporar el certificado de FNMT en la lista de entidades de confianza del servidor web Apache, que requiere formato PEM.

Un comando para consultar los datos básicos de un certificado (nombre del propietario, fechas de validez…) es el siguiente:

```
openssl x509 -inform DER -in fnmt_root.crt
-text –noout
```

Cuando el certificado público está contenido en una estructura PKCS#7 se debe utilizar el comando ***pkcs7*** de *openssl* de acuerdo a los siguientes ejemplos:

Conversión DER/PEM. La opción print_certs extrae los certificados en X.509 puro.

```
openssl pkcs7 –inform DER –in nombre.p7
–outform PEM –out nombre.p7.pem
–print_certs
```

MOSTRAR DATOS BÁSICOS:

```
openssl pkcs7 –inform DER –in nombre.p7
–print_certs –noout
```

Los navegadores de Internet (tanto Internet Explorer como Netscape/Mozilla/Firefox) exportar los certificados completos en formato p12. Al contener la clave privada, estos archivos van cifrados con una clave que se solicita al usuario en el momento de crear el archivo. Los archivos tipo p12 se pueden manejar en *openssl* mediante el comando pkcs12, que permite exportar certificados a formato PEM con bastante control sobre las claves que se exportan. Por defecto exporta a formato PEM el certificado público del propietario, y la clave privada del propietario y los certificados de las entidades de certificación (CA) que lo han firmado. Hay bastantes opciones para controlar qué información se extrae, las opciones más importantes de este comando son las siguientes:

```
openssl pkcs12 -in nombre.p12 –out
nombre.pem
```

-nokeys no exporta la clave privada.
-clcerts sólo exporta el certificado personal, no la CA que lo ha firmado.
-cacerts sólo exporta los certificados de las CA.

En función de la información que se exporte el comando pedirá las claves de cifrado del archivo p12 y del archivo PEM, que pueden ser distintas.

Una vez obtenido el certificado en formato PEM (nombre.pem en este ejemplo) es posible extraer exclusivamente la clave pública, aislándola del resto de la información del certificado (nombre, fechas…):

```
openssl rsa –in nombre.pem –out
clave_publica.pem –pubout
```

[3] En realidad el formato PEM es la conversion base64 del formato DER a la que se añaden unos delimitadores del tipo "–BEGIN CERTIFICATE–" y "–END CERTIFICATE–".

Firma electrónica

Como ya se ha visto anteriormente, la firma electrónica de un documento no se realiza cifrado todo el documento con un algoritmo asimétrico, ya que esto sería muy costoso computacionalmente tanto para el emisor como para los receptores. En la práctica la firma electrónica se realiza en dos pasos: obtención de resumen o HASH del documento y cifrado del mismo.

La herramienta *openssl* proporciona el comando ***dgst*** para obtener el resumen de un documento. Este comando recibe como argumento el nombre del algoritmo HASH que se quiera utilizar. A continuación se muestran dos ejemplos:

```
>openssl dgst -md5 bigfile.doc
MD5(bigfile.doc)=
f1c9645dbc14efddc7d8a322685f26eb
>openssl dgst -sha1 bigfile.doc
SHA1(bigfile.doc)=
8c206a1a87599f532ce68675536f0b1546900d7a
```

Estas funciones también son útiles para obtener códigos de control de integridad de archivos que son muy utilizados en detección de intrusiones, virus y otras amenazas.

Para realizar la firma electrónica como tal sería necesario cifrar el código obtenido, sin embargo toda la operación se puede realizar de manera directa mediante unos parámetros adicionales

```
openssl dgst -sign mymit.pem -out bigfile.signa-
ture bigfile.doc
```

donde *mymit.pem* es el certificado completo (incluyendo la clave privada) en formato PEM, tal y como se obtiene al aplicar el comando *pkcs12* descrito anteriormente. Este comando genera un archivo pequeño llamado *bigfile.signature* que contiene exclusivamente la firma digital del documento.

El receptor puede realizar la verificación del documento mediante el siguiente comando:

```
openssl dgst -verify mymit_public.pem
-signature bigfile.signature bigfile.doc
```

donde *mymit_public.pem* es exclusivamente la clave pública del autor, no el certificado completo.

Alternativamente se puede firmar un documento electrónicamente utilizando el comando ***smime***, lo cual genera un documento en el formato estándar de los programas de correo S/MIME que se basa en PKCS#7. Con este comando se obtiene un archivo MIME cuyo contenido es de tipo "multipart/signed" donde la primera parte es el documento original y la segunda parte es de tipo "application/x-pkcs7-signature". Este documento también se puede verificar con el comando *smime*:

```
>openssl smime -sign -in bigfile.doc -out bigfile.msg
-signer mymit.pem
Enter pass phrase for mymit.pem: xxxxxx
>openssl smime -verify -in bigfile.msg -CAfile
mit_ca.pem -out extraido.doc
Verification successful
```

Si la aplicación no necesita verificar el certificado del autor de documento, entonces se puede utilizar la opción *–noverify* en lugar de la opción *–CAfile*, como se muestra en el siguiente ejemplo:

```
>openssl smime -verify -noverify -in bigfile.msg
-out extraido.doc
Verification successful
```

Conclusiones

En este artículo se han descrito las principales aplicaciones de la criptografía. También se ha descrito lo que es un certificado digital y su importancia para poder garantizar comunicaciones seguras entre dos interlocutores, por ejemplo un usuario y su banco. Por último se han presentado ejemplos de utilización de los algoritmos mediante la herramienta de uso gratuito *openssl* que permite cifrar, descifrar, manipular certificados, firmar electrónicamente y verificar la autenticidad. ∎

Referencias

[1] V. Delgado, R. Palacios: "Introducción a la Criptografía: Tipos de algoritmos", *Anales de Mecánica y Electricidad*, Vol. LXXXIII, Fascículo 1, pp. 42-46, 2006.

[2] National Bureau of Standards, Data Encryption Standard, FIPS-Pub.197. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., Nov. 2001.

[3] R. Rivest, A. Shamir, L. Adleman: "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Vol. 21 (2), pp.120–126. 1978. Previously released as an MIT "Technical Memo" in April 1977.

[4] "OpenSSL: The Open Source toolkit for SSL/TLS", http://www.openssl.org.