Table of Contents

1 Prácticas de programación segura

A la hora de escribir un código seguro, se tendrán en cuenta las siguientes pautas:

- Documentarse: Estudiar los fallos que otros hayan podido cometer al escribir rutinas similares a las que se están programando. Conviene investigar software libre de proyectos ya asentados para aprender de estas técnicas.
- Comprobar los datos de entrada: No fiarse de que los datos que vaya a proveer el usuario sean lo que se espera, comprobar longitud (para no superar los límites de la memoria o de los buffers), revisar los tipos de datos (por ejemplo, si se espera que el usuario escriba un número verificar que realmente es un número y no una palabra),…
- Verificar que las variables se inicializan correctamente. No dejar variables sin inicializar e inicializarlas a valores consistentes.
- Proteger el almacenamiento de datos sensibles. Por ejemplo, las contraseñas se deben guardar cifradas (lo habitual es aplicar una función hash).
- Reusar código verificado o usar bibliotecas ampliamente probadas.
- Revisar el código. Sobre todos después de muchas modificaciones por un equipo de programadores un código puede presentar inconsistencias.
- Comentar el código y mantenerlo legible. A veces una implementación que sea fácil de entender es preferible a una implementación más eficiente, pero difícil de entender.

2 Criptografía

La criptografía según la Wikipedia es:

La criptografía (del griego κρύπτος (kryptós), «secreto», y γραφή (graphé), «grafo» o «escritura», literalmente «escritura secreta») se ha definido, tradicionalmente, como el ámbito de la criptología que se ocupa de las *técnicas de cifrado o codificado destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados*.

Existen tres clases de algoritmos criptográficos:

- **Funciones hash**: Consiste en una función de la forma f(x) → y que trasforma x en un valor y. El cálculo de f(x) es muy rápido obteniendo y, pero el proceso inverso es muy lento o imposible. Se usan para verificar que la información de un mensaje no ha sido alterada. Entre los algoritmos disponibles se dispone del **MD5** o del **SHA1**.

En principio el resultado de la operación f(x) → y debería ser único, pero hay situaciones en que varios valores producen el mismo resultado. A esto se le denomina **colisión hash**. Por ejemplo, supongamos que tenemos arrays de números y la función hash consiste en sumarlos. Los arrays {1, 2, 3} y {3, 3} producen el mismo resultado, 6, por lo que se tiene una colisión hash.

- **Algoritmos de criptografía simétrica**: Son aquellos que usan la misma clave para cifrar como para descifrar el mensaje. Entre los algoritmos de cifrado simétricos se tienen el **DES**, **Triple DES** y el **AES**.
- **Algoritmos de criptografía asimétrica**: Son aquellos en los que se usan dos claves que están relacionadas entre sí. A una de estas claves se la llama la **clave pública** y se puede difundir entre los sistemas con los que se desea mantener comunicación. La otra clave se denomina **clave privada** y como su nombre indica no debe ser difundida. Los mensajes cifrados con la clave privada, sólo pueden ser descifrados con la clave pública. Los mensajes cifrados con la clave pública, sólo pueden ser descifrados con la clave privada. Entre los algoritmos más usados está el **RSA**.

La ventaja de los algoritmos de criptografía simétrica respecto a los de criptografía asimétrica, es que son más rápidos.

Si se quiere realizar una comunicación entre varios interlocutores usando criptografía simétrica, primero se debe encontrar un mecanismo para pasar la contraseña de forma segura entre los interlocutores. Esto es un problema cuando se usan medios de comunicación inseguros.

Otro problema es que el cifrado de las mismos mensajes con la misma clave, hace que se generen las mismas porciones de código, por lo que un “espía” podría identificar esas porciones en los mensajes e intentar descifrar los mensajes. Esto se puede resolver fácilmente dividiendo el mensaje en bloques y haciendo una operación de XOR de un bloque con el anterior y cifrándolo. De esta forma el mensaje cifrado irá variando de forma aleatoria según vaya creciendo el mensaje.

Uno de los usos de la criptografía simétrica es que puede convertir los canales inseguros en canales seguros de la siguiente forma. Supongamos que varios interlocutores quieren comunicarse, lo primero que hacen es compartir sus claves públicas. Si A quiere enviar un mensaje a B, lo cifra usando la clave pública de B, de forma que sólo B lo podrá descifrar usando su clave privada.

Lo más habitual es mezclar la criptografía simétrica y asimétrica. Los interlocutores intercambian sus claves públicas y las usan para pasarse una clave que se usará posteriormente para cifrar los mensajes usando criptografía simétrica. Al ser más rápida la criptografía simétrica que la asimétrica, se ganará en rendimiento.

2.1 Firma digital

Uno de los usos de las funciones hash y la criptografía asimétrica es la firma digital de mensajes. Imaginemos que se desea verificar que un mensaje no ha sido alterado y que proviene del emisor correcto y no ha sido suplantada su identidad. El procedimiento para firmar digitalmente el mensaje sería el siguiente:

1. Se calcula el valor hash del mensaje.

2. Se cifra dicho valor hash usando la clave privada del emisor del mensaje. Este valor hash cifrado es la firma digital.
3. Se difunde el mensaje con la firma digital.

Si alguien quiere verificar la autenticidad del mensaje:

1. Calcula el valor hash del mensaje.
2. Descifra la firma digital usando la clave pública del emisor.
3. Si los dos valores hash coinciden, el mensaje no ha sido alterado y proviene del emisor.

Queda un punto pendiente, que es garantizar que la clave pública proviene de quien dice ser. Para ello se usan las **Autoridades de certificación** que son organismos que se encargan de verificar la identidad de los propietarios de las claves públicas dando al usuario un par de claves, pública y privada, el **certificado digital**. Hay numerosos organismos de este tipo. En España se dispone de la FNMT (Fabrica Nacional de Moneda y Timbre) entre otros.

Para obtener el certificado digital de la FNMT, hay que visitar su página web y solicitarlo. Una vez solicitado hay que personarse físicamente en una de las oficinas con las que tienen concierto y allí validarán el certificado. Después hay que descargar el certificado en un equipo usando un enlace y una contraseña que la FNMT proveerá.

En España, además, se dispone del DNI electrónico que no es más que un certificado digital almacenado en el chip del DNI que puede ser leído con un lector.

El certificado digital se almacena en un formato denominado **X.509** que está ampliamente aceptado. Más información de este formato se tiene en:

https://es.wikipedia.org/wiki/X.509

Uno de los campos del formato X.509 lo que hace es incluir el valor hash del certificado digital del usuario cifrado con la clave pública de la Autoridad de certificación. Así el destinatario puede verificar que la clave pública es correcta y su autenticidad está verificada.

Además el X.509, también incluye algunos datos del propietario de la clave pública.

3 Control de acceso

A veces es necesario controlar quien accede a un determinado recurso. Hay tres puntos a tener en cuenta en el control de acceso:

1. Identificación: es el proceso por el cual alguien indica quien es.
2. Autentificación: consiste en verificar si la información proporcionada por el usuario es cierta.
3. Autorización: consiste en determinar si el usuario, una vez autentificado, tiene acceso al recurso.

La forma de realizar la autentificación puede ser mediante un sistema de usuario y contraseña, usar elementos biométricos (huella dactilar,…), por tarjetas de acceso o por tokens.

Un **token** es una forma de realizar el control de acceso en el cual el servidor cifra (por clave simétrica o asimétrica) una información que identifica al usuario y la sesión que tiene abierta. Este token se envía al cliente que no es capaz de leer su contenido. Cada vez que el usuario quiere usar su sesión, se envía el token, el servidor lo descifra y obtiene la información de la sesión.

La identificación por token es muy usada en las APIs REST, ya que no es necesario enviar el token al mismo servidor que lo generó y cualquier servidor del cual dependa el servicio puede procesarlo.

Curiosidad:
REST significa de "REpresentational State Transfer", “transferencia de representación de estado”, la idea es que el servidor no recuerde el estado del cliente. Entre dos llamadas, el servidor borra el estado en el que se encontraba el cliente y debe ser el cliente el que mande los datos sobre su estado.
Por ejemplo, si el cliente inicia sesión en el servidor, el servidor crea un token y se lo envía al cliente. El servidor no recuerda si el usuario ha iniciado sesión o no y es el cliente a través del token el que se lo indica.
Una aplicación escrita con REST es útil en el caso de que se prevea que vaya a crecer mucho, pues es fácil añadir nuevos servidores y que la carga de trabajo se reparta entre ellos de forma equilibrada.

4 Política de seguridad

Una política de seguridad es un conjunto de reglas para el mantenimiento de un cierto nivel de seguridad. Estas políticas pueden ser reglas para fijar contraseñas, planes de actualizaciones, normas sobre el uso de equipos, técnicas de cifrado,...