Técnicas de Seguridad

Los sistemas criptográficos necesitan cumplir varias características:

- **Confidencialidad**: Sólo pueden tener acceso a la información las personas o sistemas autorizados.
- **Integridad**: La información gestionada por el sistema debe permanece íntegra, sin sufrir pérdidas o alteraciones.
- **Autenticación**: Se debe poder verificar la identidad de los participantes del sistema.
- **Vinculación**: o no repudio, Dispone de mecanismos para que el creador o emisor de una información no pueda negar su vínculo o autoría.

En esencia, todos los procesos criptográficos funcionan de la misma forma. A una entrada se le aplica un algoritmo, que producirá una salida.

- Si a un texto legible se le aplica un algoritmo de cifrado, que usualmente depende de **una clave**, nos dará un texto cifrado que es el que se envía al receptor. A este proceso de le llama **cifrado**.
- Si a ese texto cifrado se le aplica el mismo algoritmo, a veces la **misma clave**, se obtiene el texto legible original. A este proceso se le llama **descifrado**.

Algoritmos de cifrado: HASH o de resumen

Los algoritmos **de una sola vía** (o hash o de resumen) permiten mantener la integridad de los datos tanto en almacenamiento en BBDD como en el tráfico de redes. Reciben su nombre debido a su naturaleza matemática.

La lógica de estos algoritmos es la siguiente. Tú tienes una función y un mensaje. Es fácil calcular el resultado de la función f(x). A este f(x) se le denomina el **hash de x**, o "resumen" de x, o también *message digest* de x. La gracia es que, si bien hacer f(x) es fácil, es prácticamente imposible hacer el paso contrario: obtener x por mucho que conozcas la función.

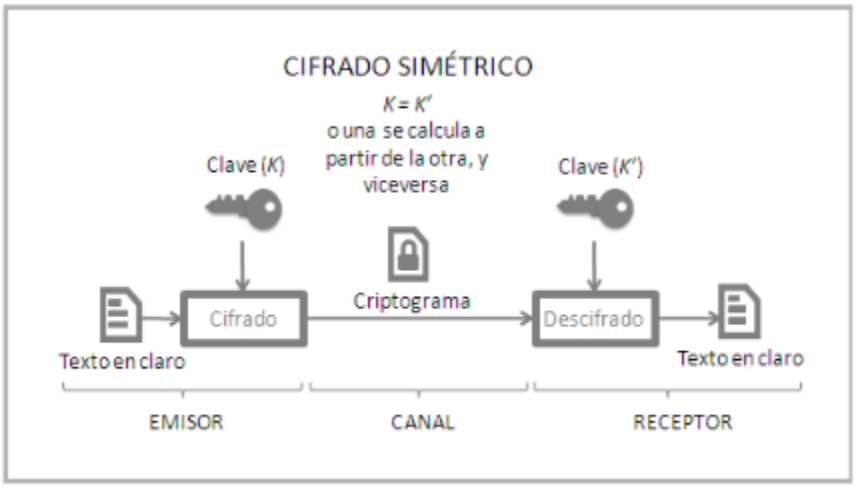
¿Para qué se usan los algoritmos HASH? Supongamos que me quiero registrar en una página web. Cuando me doy de alta, suministro un password que será cifrado y guardado en BBDD. Para logearme, le suministro un nuevo password que el Servidor cifra y después compara con el de BBDD. Si son iguales, dejo acceder al usuario. De esta forma, sólo yo conozco mi clave.

¿Cómo protege esto mi clave? Fácil… si un hacker accede a la BBDD e intenta robar las claves de la Tabla de Usuario, no podrá hacerse con mis claves, dado que es imposible obtener una clave por mucho que conozcas el algoritmo de cifrado.

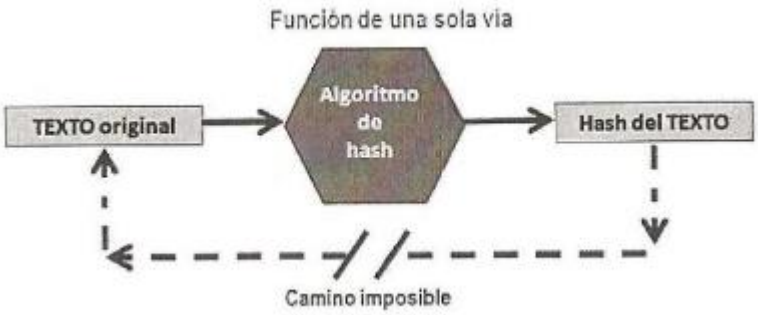
Los dos algoritmos más usados: **MD5** y **SHA**. Tienes un ejemplo en EjemploSHA.java

Algoritmos de cifrado: Clave Privada

En este tipo de cifrado (de clave secreta o de **criptografía simétrica**) el emisor y el receptor del mensaje comparten el conocimiento de **una clave** que no debe ser revelada a ningún otro. La clave se utiliza tanto para cifrar como descifrar el mensaje.



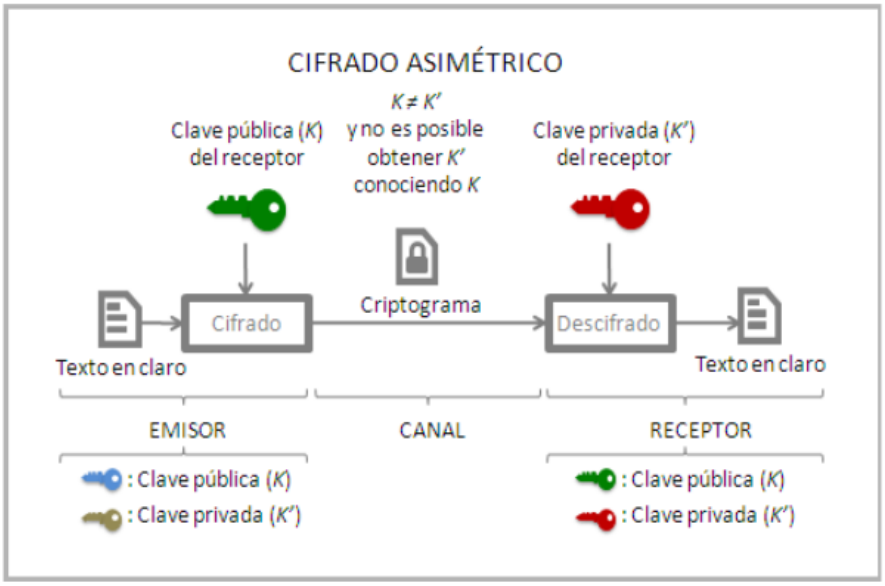
En este tipo de criptosistemas la clave para descifrar un criptograma (K') se calcula a partir de la clave (K) que se utilizó para cifrar el texto en claro, y viceversa. Lo habitual en los criptosistemas de este tipo es que ambas claves (K y K') coincidan, es decir, que se utilice la misma clave para cifrar un mensaje y para su descifrado.



Los dos algoritmos más usados: **DES** y **AES**. Tienes un ejemplo en **EjemploAES.java**

Algoritmos de cifrado: Clave Pública

Estos algoritmos, también llamados de **criptografía asimétrica**, son un poco rizar el rizo. El emisor de un mensaje (Participante B) emplea una clave pública, que puede ser conocida por el receptor (Participante A) **o por cualquier otra persona**, para cifrar el mensaje. Lo que sucede es que solamente el receptor puede descifrar el mensaje mediante una clave privada, que sólo él conoce.



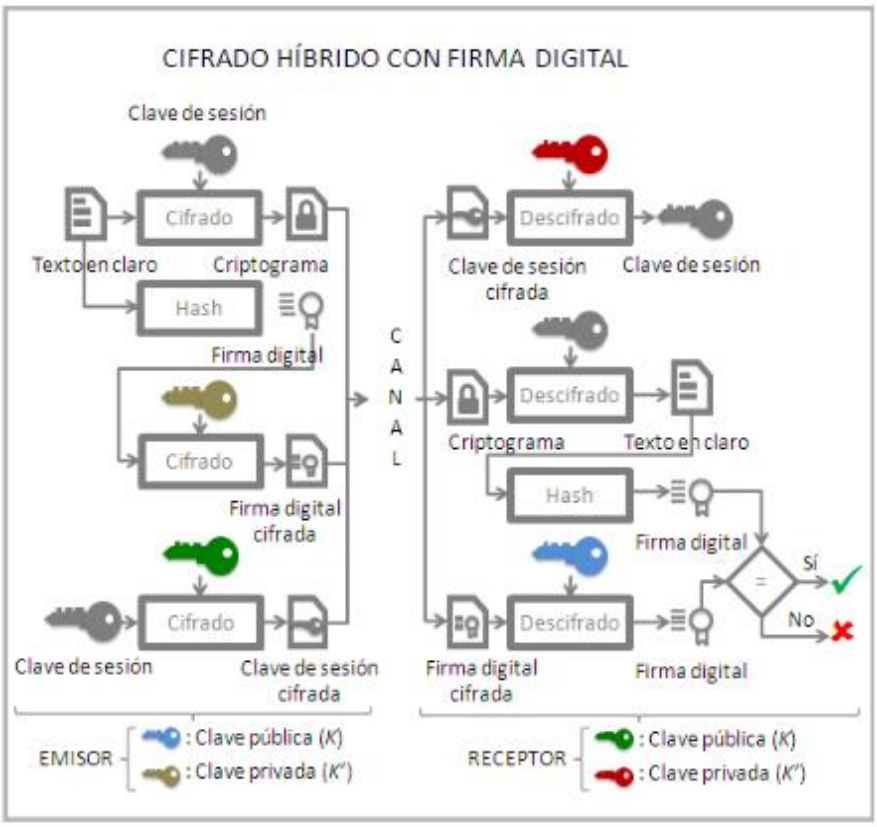
En este tipo de criptosistemas, la clave para descifrar un criptograma (K') es diferente de la clave (K) que se utilizó para cifrar el texto en claro (de ahí el nombre de cifrado asimétrico) y, además, aunque no son independientes, del conocimiento de K **no es posible** deducir K', o al menos hasta el momento no se conoce que nadie lo haya logrado.

Este tipo de criptosistemas se caracterizan porque tanto el emisor como el receptor tienen un **par de claves** (K y K') para asegurarse una comunicación bidireccional.

El algoritmo más conocido es **RSA**. Tienes un ejemplo en **EjemploRSA.java**, junto a un generador de claves llamado **EjemploRSA_KeyGenerator.java**

Firma Digital

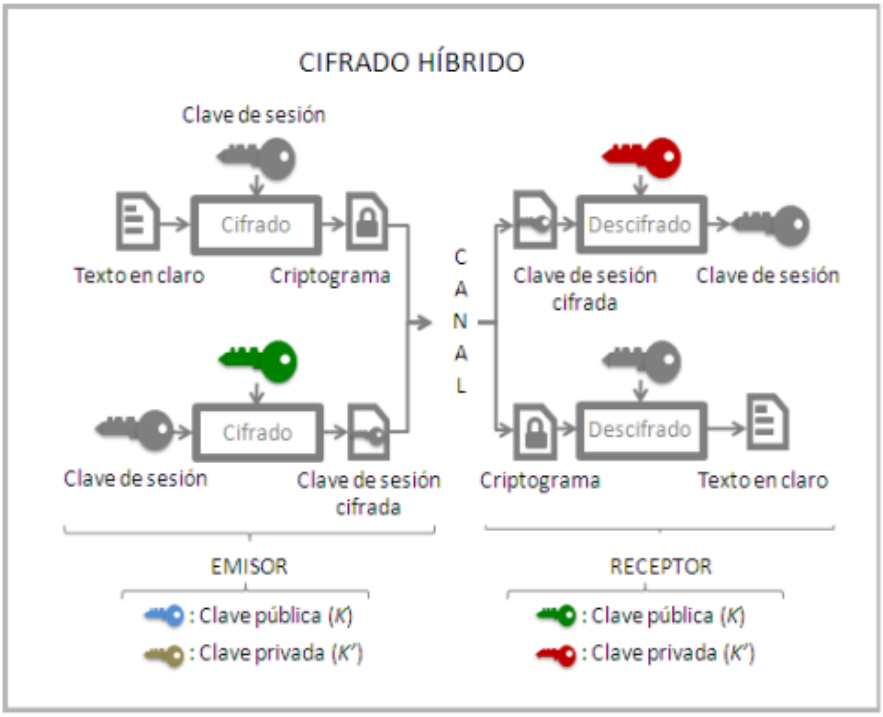
Una **firma digital** es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar a la entidad originadora de dicho mensaje (autenticación de origen y no repudio), y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador (integridad).



El algoritmo que se usa es **DSA**. Tienes un ejemplo en **EjemploDSAjava**, junto a un generador de ficheros de firma llamado **EjemploDSA_DigitalSignatureGenerator.java**

Criptografía híbrida

No hay nada que nos impida mezclar diferentes tipos de cifrado a la vez. En este tipo de criptosistemas el texto en claro se cifra mediante una clave de sesión (simétrica), correspondiente a cada mensaje particular y generada aleatoriamente, y la clave de sesión se cifra utilizando la clave pública del receptor (asimétrica), de tal manera que sólo el receptor puede obtener la clave de sesión mediante su clave privada (asimétrica) y posteriormente descifrar el texto cifrado mediante la clave de sesión (simétrica).



Protocolos seguros de comunicaciones

Las comunicaciones mediante sockets TCP/UDP y HTTP son potencialmente inseguras, dado que la información que se envía mediante estos protocolos puede ser interceptada y leída sin problemas. En Java, es posible utilizar las clases del paquete **java.net.ssl** para utilizar sockets seguros basados en SSL. Por su parte, siempre podemos utilizar HTTPS en lugar de HTTP.

Para poder utilizar las comunicaciones basadas en SSL se necesita de un **Certificado Digital**. La obtención de un certificado reconocido como válido debe de ser proporcionado por una Autoridad Certificadora. Como alternativa, Java proporciona herramientas para crea tu propio certificado digital.

Desde la **consola de Windows** podemos hacer:

keytool -genkeypair -alias borja -keyPass Mad23@ -validity 100 -storepass Mad23@ -keyalg "DSA"

Esta instrucción genera un fichero en la ruta c:\Usuarios\borja llamado **.KeyStore**, donde se almacena la información relevante del certificado.

```
C:\Users\borja>keytool -genkeypair -alias borja -keyPass Mad23@ -validity 100 -storepass Mad23@ -keyalg "DSA"
¿Cuáles son su nombre y su apellido?
  [Unknown]:  borja
¿Cuál es el nombre de su unidad de organización?
  [Unknown]:  borja
¿Cuál es el nombre de su organización?
  [Unknown]:  borja
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:  borja
```

Estos certificados no son detectados como fiables, pero pueden usarse para realizar comunicaciones seguras entre aplicaciones propias. Para comunicar dos sockets SSL entre ellos, primero se debe de crear un certificado para cada uno de ellos, exportarlos e importarlos de forma cruzada (el del cliente al servidor y viceversa). Una vez realizada la configuración adecuada, ya podemos utilizar las clases SSLServerSocket y SSLSocket de la forma habitual.