



# 05. Técnicas de programación segura

Programación de Servicios y procesos - 2º DAM

Luis del Moral Martínez

versión 20.10

Bajo licencia CC BY-NC-SA 4.0



# Contenidos del tema

## 1. Técnicas de programación segura

- 1.1 Introducción
- 1.2 Prácticas de programación segura
- 1.3 Criptografía y control de acceso
- 1.4 Seguridad en el entorno Java
- 1.5 Ficheros de políticas en Java
- 1.6 Criptografía en Java
- 1.7 Comunicaciones seguras con Java
- 1.8 Control de acceso con Java

# 1.1 Introducción

## La programación segura

- En este tema abordaremos buenas prácticas que permitan escribir código seguro
- En este tema usaremos las siguientes librerías de Java:
  - Librerías criptográficas (JCA y JCE)
  - Extensión de Sockets Seguros (JSSE)
  - Servicio de autenticación y autorización de Java (JAAS)

# 1.2 Prácticas de programación segura

## Recomendaciones generales

- Estudiar y comprender los errores que otros hayan cometido a la hora de desarrollar software
- Conocer el estado del arte antes de empezar a programar
- Acceder a foros y recursos de Internet (pueden ayudarnos a solucionar errores)
- Leer libros y artículos sobre prácticas de codificación
- Explorar proyectos de código abierto: [GitHub](#)

# 1.2 Prácticas de programación segura

## Precauciones en el manejo de datos (1)

- Limpiar los datos de entrada y controlarlos (inyección SQL, desbordamiento de búfer...)
- Realizar una comprobación de los límites (para evitar desbordamientos)
- Revisar los ficheros de configuración
- Comprobar los parámetros de línea de comandos
- No confiar en las URLs para intercambiar información (GET)
- No confiar del contenido web oculto (campos HTML ocultos en un formulario)
- Comprobar las cookies web

# 1.2 Prácticas de programación segura

## Precauciones en el manejo de datos (2)

- Comprobar las variables de entorno (suelen usarse para pasar preferencias a los programas)
- Establecer valores iniciales que sean válidos para todos los datos
- Comprender las referencias de nombre de fichero (rutas de acceso) y usarlas correctamente
- Prestar especial atención al almacenamiento de información sensible
  - Integridad, Confidencialidad y Disponibilidad de la información

# 1.2 Prácticas de programación segura

## Revisión de los procesos

- Los fallos de seguridad en el código fuente deben ser revisados de forma efectiva
- Realizar revisiones de código por pares (dos o más revisores)
- Realizar validación y verificación independiente (Puede que incluso línea a línea del código)
- Identificar y usar herramientas de seguridad disponibles (existe multitud de software)
- Utilizar librerías y metodologías de testeo efectivas

# 1.2 Prácticas de programación segura

## Utilizar listas de control de seguridad

- Permiten verificar si se han cubierto todas las fases durante la ejecución
- **Ejemplo**
  - La aplicación necesita una contraseña para que puedan acceder los usuarios
  - Los inicios de sesión son únicos
  - La aplicación usa el sistema de control de acceso basado en roles
  - Las contraseñas no se transmiten a través de la red en texto plano
  - Se emplean mecanismos de cifrado para proteger los mecanismos de transferencia de información



# 1.2 Prácticas de programación segura

## Lo que NO se debe hacer (1)

- Escribir código que usa nombres de ficheros relativos, deben usarse referencias completas
- Referir dos veces en un programa un mismo fichero por su nombre
- Invocar programas o servicios no confiables y/o obsoletos o sin mantenimiento
- Asumir que los usuarios no son maliciosos o no pueden actuar mal
- No realizar gestión de excepciones y asumir siempre el éxito
- Invocar un Shell o una línea de comandos desde la aplicación
- Autenticarse con criterios que no sean de confianza

# 1.2 Prácticas de programación segura

## Lo que NO se debe hacer (2)

- Usar áreas de almacenamiento con permisos de escritura (cuidado con esto)
- Guardar datos confidenciales en una base de datos sin contraseña o cifrado alguno
- Hacer eco de las contraseñas o mostrarlas en las vistas de usuario
- Emitir contraseñas o credenciales por correo electrónico (y olvidando el doble factor, etc.)
- Distribuir mediante programación alguna información confidencial por medio del email
- Guardar las contraseñas (o cualquier otra información sensible) sin cifrar

# 1.2 Prácticas de programación segura

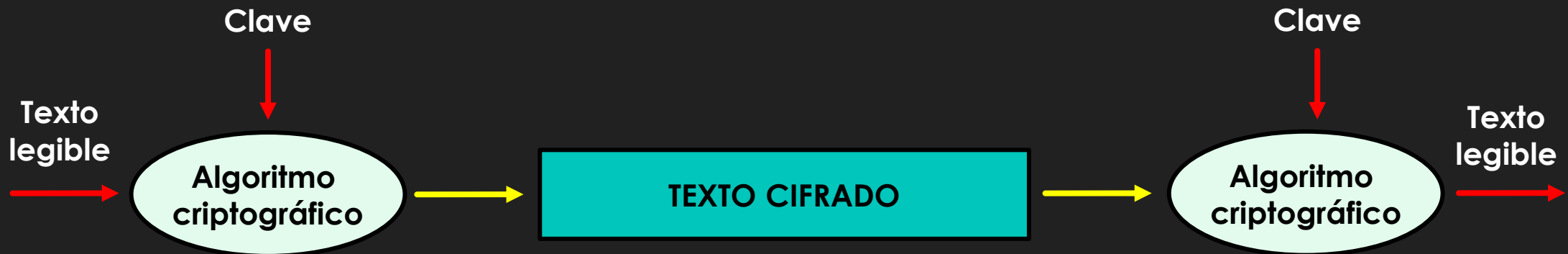
## Lo que NO se debe hacer (3)

- Distribuir contraseñas sin encriptar entre los sistemas (o cualquier información sensible)
- Tomar decisiones de acceso según variables o argumentos en tiempo de ejecución
- Confiar en exceso en software de terceros para las operaciones críticas

# 1.3 Criptografía y control de acceso

## La criptografía

- Este término deriva de la palabra griega *kryptos*, que significa oculto
- El objetivo de la criptografía es ocultar el significado de un mensaje



# 1.3 Criptografía y control de acceso

## La criptografía

- Existen tres clases de algoritmos criptográficos
  - Funciones de **una sola vía** (funciones hash)
  - Algoritmos de **clave privada** o de criptografía **simétrica**
  - Algoritmos de **clave pública** o de criptografía **asimétrica**

# 1.3 Criptografía y control de acceso

## Funciones Hash

- Estas funciones mantienen la integridad de los datos en el almacenamiento y en las redes
- A partir de un mensaje se calcula el **Message digest** o resumen
- Es prácticamente imposible realizar el camino opuesto
- Los algoritmos más usados son **MD5** y **SHA-1**

# 1.3 Criptografía y control de acceso

## Funciones Hash



# 1.3 Criptografía y control de acceso

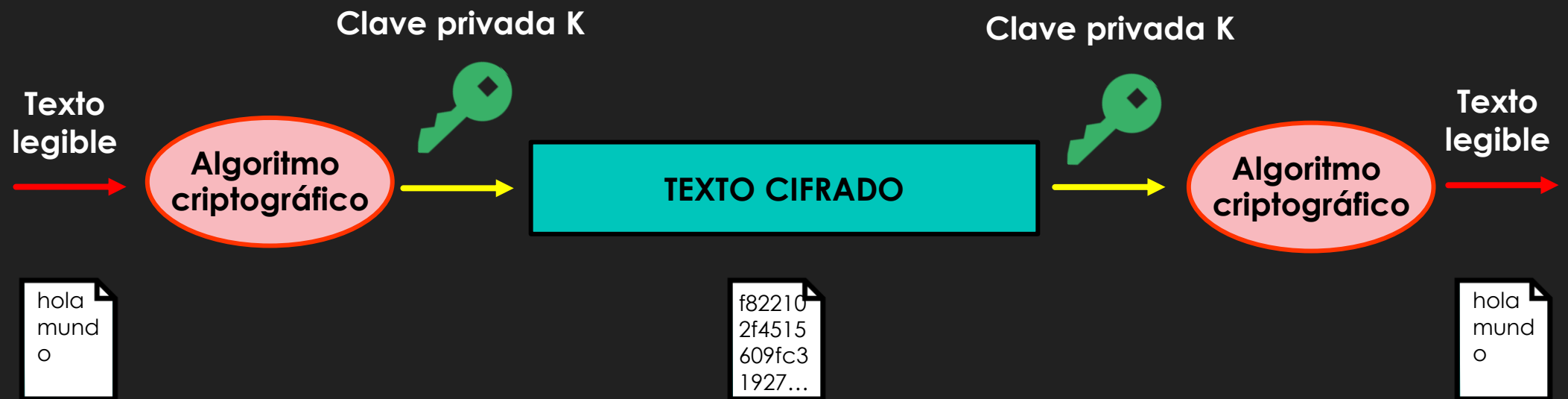
## Criptografía simétrica

- El emisor y el receptor comparten el conocimiento de una **clave privada**
- Esta clave no puede ser compartida con nadie más
- La clave permite **cifrar** y **descifrar** el mensaje
- El algoritmo más popular es el **DES** y su variante el **3DES**, además del cifrado **AES** de 256 bits



# 1.3 Criptografía y control de acceso

## Criptografía simétrica



# 1.3 Criptografía y control de acceso

## Criptografía simétrica

- **Ventajas**

- Cifra más rápido que los algoritmos de clave pública
- Sirven para los sistemas criptográficos basados en hardware

- **Inconvenientes**

- Requieren un sistema de distribución de claves muy seguro
- Si la clave privada es conocida, entonces el sistema no tiene ningún sentido
- Crecimiento exponencial del número de claves (si cada pareja requiere una clave privada)

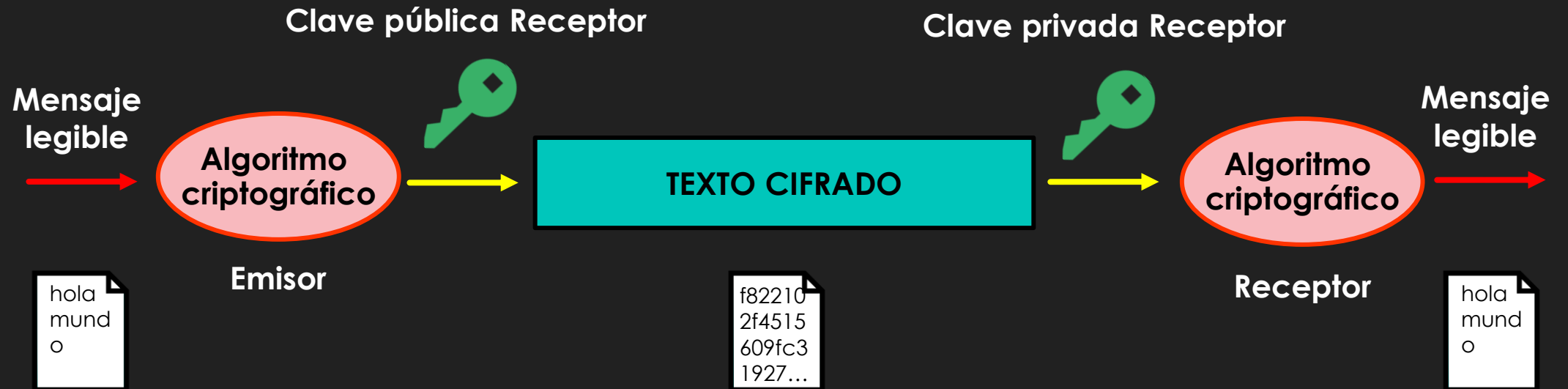
# 1.3 Criptografía y control de acceso

## Criptografía asimétrica

- El emisor emplea una **clave pública**, definida previamente por el receptor, para encriptar
- El receptor emplea la **clave privada** correspondiente para descryptar el mensaje
- De esta forma, solo el receptor puede descryptar el mensaje
- Estos algoritmos implican que cada participante tenga un **par de claves**
  - La **clave pública** es conocida por todos (para poder enviarle mensajes)
  - La **clave privada** es conocida solo por cada interesado (para poder descryptar los mensajes)
- El algoritmo más popular es el **RSA** (Rivest-Shamir-Adleman)
- Se emplea en el cifrado de protocolos y sistemas (IPSec, SSL, PGP...)

# 1.3 Criptografía y control de acceso

## Criptografía simétrica



# 1.3 Criptografía y control de acceso

## Criptografía asimétrica

### ■ Ventajas

- Permite autenticación y no repudio en muchos protocolos criptográficos
- Se emplean en colaboración con otros métodos criptográficos
- Administración sencilla de claves (no requieren de intercambio de claves seguro)

### ■ Inconvenientes

- Son algoritmos más lentos que los algoritmos de clave privada
- Sus implementaciones se suelen realizar en sistemas software
- Para una gran cantidad de usuarios se requiere de un sistema de certificación de autenticidad

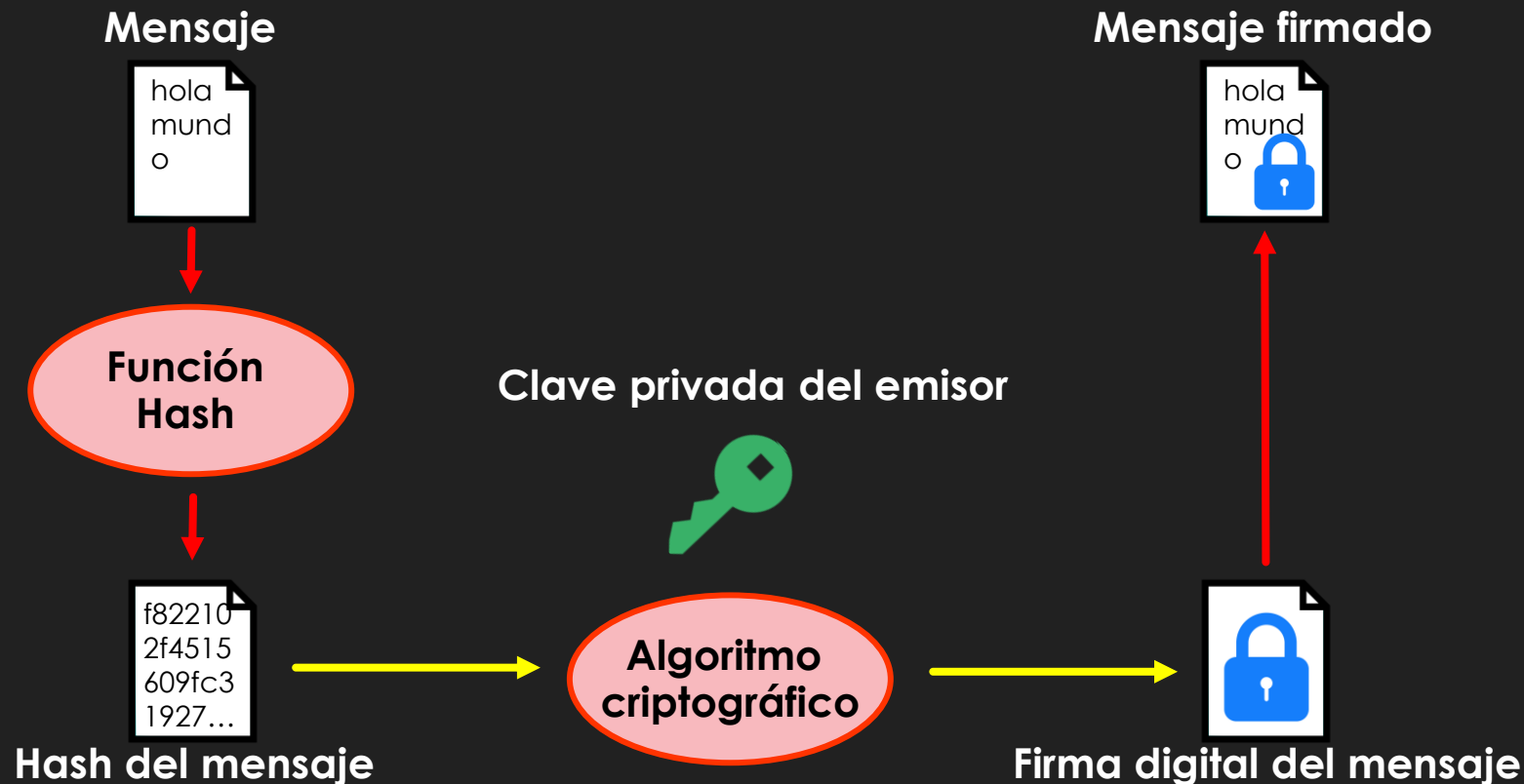
# 1.3 Criptografía y control de acceso

## ¿Cómo funciona una firma digital?

- Una **firma digital** es un compuesto de datos asociados a un mensaje
- Estos datos permiten asegurar la identidad del emisor y la integridad de la información
- El método de firma digital más extendido es el RSA

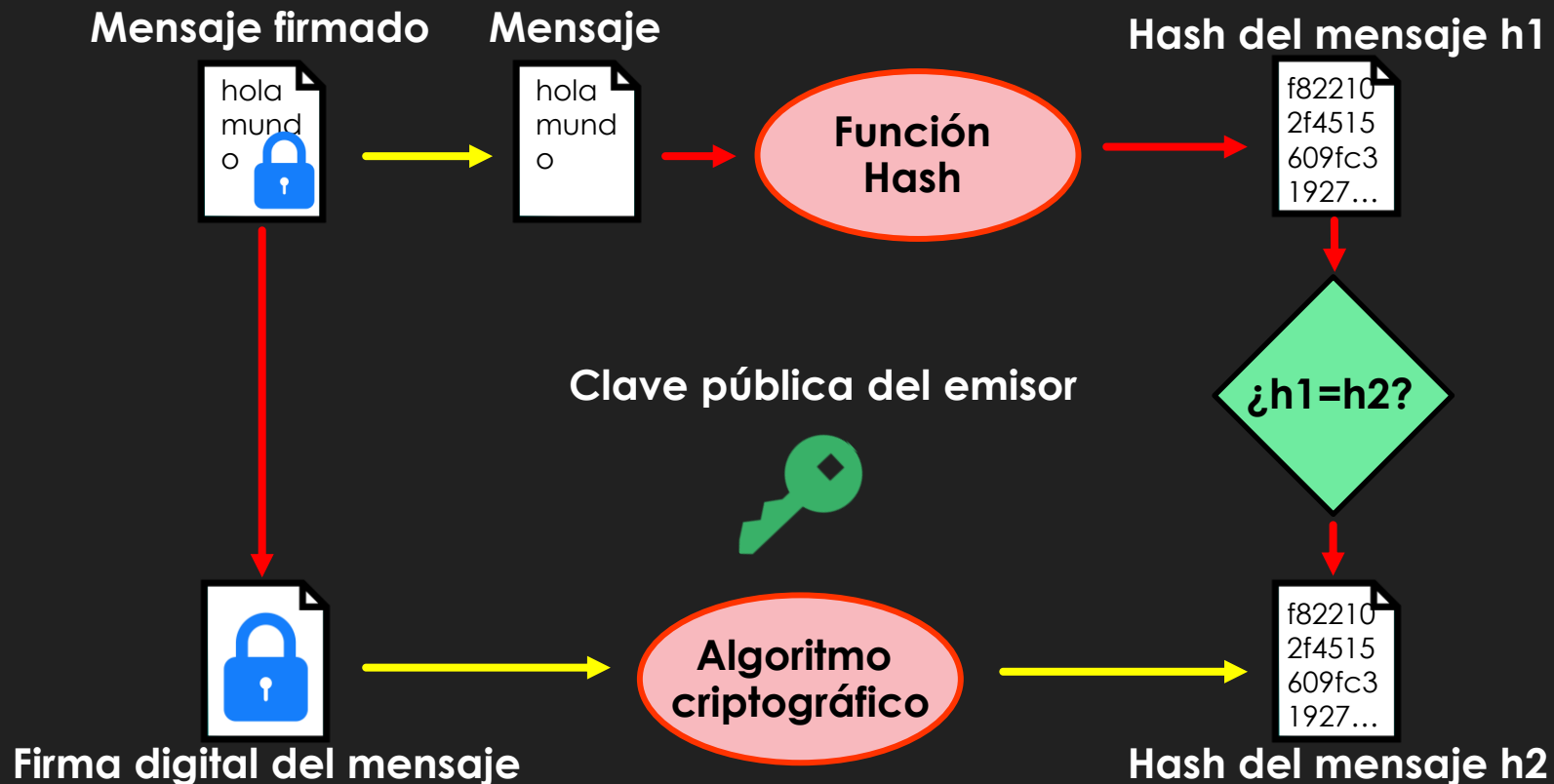
# 1.3 Criptografía y control de acceso

## Firma digital – Proceso de firmado



# 1.3 Criptografía y control de acceso

## Firma digital – Comprobación del mensaje firmado





# 1.3 Criptografía y control de acceso

## Certificados digitales

- Un **certificado digital** es un documento que certifica la clave pública de una entidad
- El certificado puede estar destinado a un usuario, una máquina, proceso o dispositivo
- Para certificar estos documentos se acude a las **Autoridades de Certificación** (CA)
- Las CA son entidades que emiten y gestionan los certificados y son **confiables**
- Los certificados digitales siguen el estándar X.509

# 1.3 Criptografía y control de acceso

## Estándar X.509

- Este estándar posee los siguientes campos:
  - Versión: normalmente X.509v3
  - Número de serie: Identificador numérico único dentro del dominio de la CA
  - Algoritmo de firma y parámetros
  - Emisor del certificado
  - Fechas de inicio y validez
  - Nombre del propietario de la clave pública
  - Identificador del algoritmo, clave pública y otros parámetros
  - Firma digital de la CA (resultado de cifrar el algoritmo simétrico y la clave privada, es el hash resultante)

# 1.3 Criptografía y control de acceso

## Estándar X.509

De este documento se obtiene el *hash* y se cifra con la clave privada de la CA para obtener la firma digital

Versión
Número de serie
ID algoritmo y parámetros
Entidad emisora (CA)
Inicio de validez
Fin de validez
Usuario
Algoritmo, parámetros y clave
Firma digital de la CA

Identificador del algoritmo

Período de validez

Clave pública del usuario

# 1.3 Criptografía y control de acceso

## Visualización de un certificado digital en el navegador

Información de la página - https://www.google.es/

General Medios Permisos Seguridad

**Identidad del sitio web**

Sitio web: www.google.es

Propietario: Este sitio web no proporciona información sobre su dueño.

Verificado por: Google Trust Services [Ver certificado](#)

Expira el: martes, 18 de agosto de 2020

**Privacidad e historial**

¿Se ha visitado este sitio web anteriormente? Sí, 4378 veces

¿Este sitio web almacena información en mi ordenador? Sí, cookies [Limpiar cookies y datos del sitio](#)

¿Se han guardado contraseñas de este sitio web? No [Ver contraseñas guardadas](#)

**Detalles técnicos**

Conexión cifrada (TLS\_AES\_128\_GCM\_SHA256, claves de 128 bits, TLS 1.3)

La página que está viendo fue cifrada antes de transmitirse por Internet.

El cifrado dificulta que personas no autorizadas vean la información que viaja entre sistemas. Es, por tanto, improbable que nadie lea esta página mientras viajó por la red.

[Ayuda](#)

### Certificado

*.google.es	GTS CA 101
-------------	------------

**Nombre del asunto**

**País** US

**Estado/Provincia** California

**Localidad** Mountain View

**Organización** Google LLC

**Nombre común** \*.google.es

**Nombre del emisor**

**País** US

**Organización** Google Trust Services

**Nombre común** GTS CA 101

**Validez**

**No antes** 26/5/2020 17:37:46 (hora de verano de Europa central)

**No después** 18/8/2020 17:37:46 (hora de verano de Europa central)

# 1.3 Criptografía y control de acceso

## Aplicaciones de los certificados digitales

- Autenticar la identidad del usuario de forma electrónica ante terceros
- Realizar trámites administrativos online (Agencia Tributaria, Seguridad Social, SAS...)
- Trabajar con otras facturas o servicios electrónicos
- Firmar electrónicamente todo tipo de documentos
- Cifrar datos para que sólo el destinatario pueda acceder a su contenido

# 1.3 Criptografía y control de acceso

## Cómo solicitar un certificado digital

- Si es un certificado de persona física, a través de la FNMT o usando el DNI-e
- Si es un certificado para una aplicación o empresa se debe solicitar a través de la CA:
  1. Crear un Certificate Signing Request (CSR): <https://www.digicert.com/csr-creation.htm>
  2. Guardar a buen recaudo todos los datos suministrados y las claves generadas
  3. Acceder a un proveedor de certificados de confianza (Starfield, etcétera)
  4. Comprar el certificado SSL
  5. Validar el dominio (acceder a la zona DNS o validación mediante **well-known** y **pki-validation**)
  6. Si es wildcard (\*), instalar en el resto de dispositivos, servicios o aplicaciones donde se vaya a usar

# 1.3 Criptografía y control de acceso

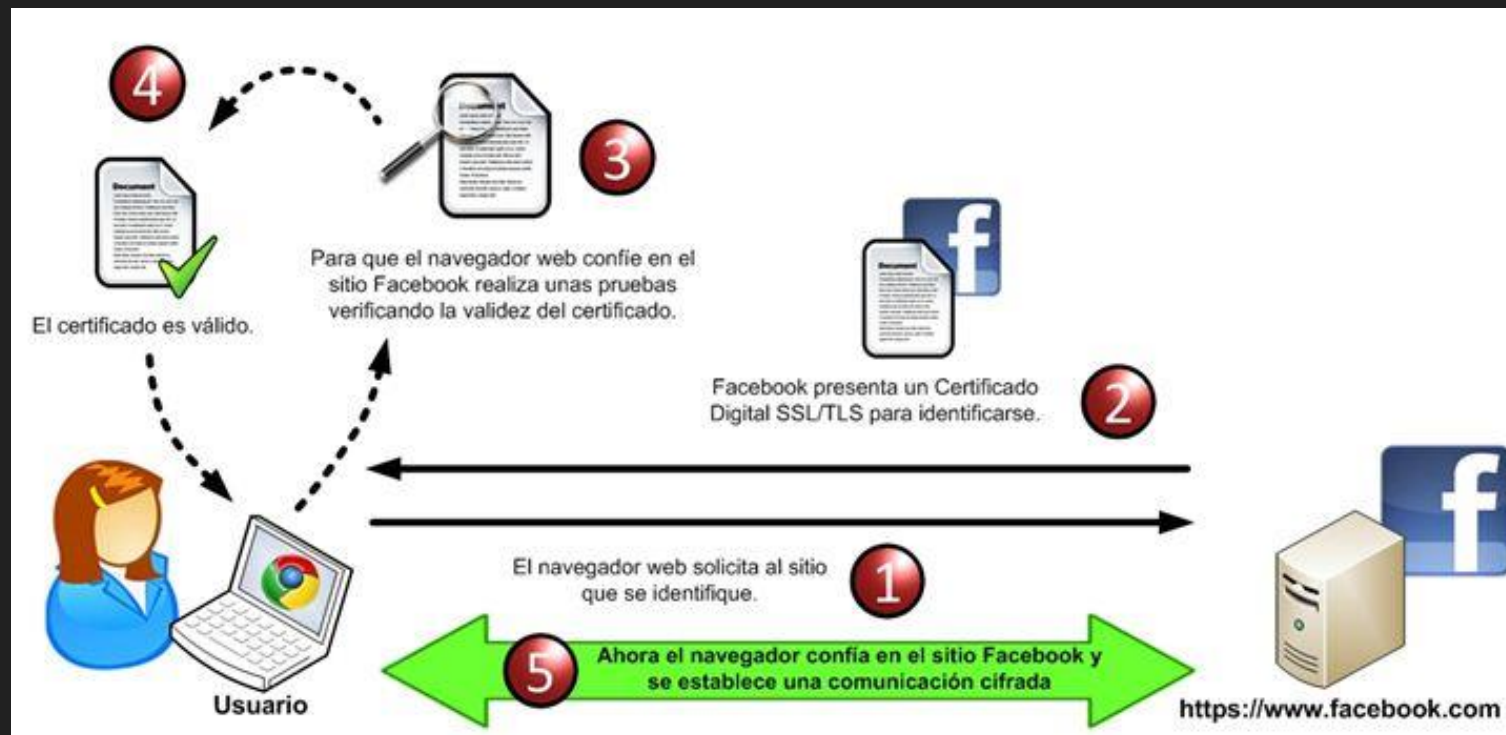
## ¿Dónde comprar un certificado SSL?

- Lugares de confianza para obtener un certificado SSL (mi recomendación):
  - **Gratuitos:** Let's Encrypt
    - <https://letsencrypt.org/es/>
      - Duración de 4 meses
      - No los soportan los cortafuegos y algunos servidores web
  - **De pago:** Domains Priced Right (desde 70\$ al año)
    - <https://www.domainspricedright.com/products/ssl>
      - Duración de 1 o 2 años (siempre hay 3-4 meses de prórroga)
      - Posibilidad de comprar un certificado Wildcard (\*)
      - Permite descargar el certificado en múltiples formatos
      - Es una CA de alta seguridad y reputación (antigua Starfield)



# 1.3 Criptografía y control de acceso

## Caso real de uso de certificado SSL





# 1.3 Criptografía y control de acceso

## Control de acceso

- El control de acceso gestiona el acceso a los recursos de un sistema
- Posee tres componentes esenciales:
  - **Identificación**: proceso que sigue un usuario o sujeto para indicar su identidad en el sistema
  - **Autenticación**: proceso que verifica que un usuario o sujeto es quien dice ser
  - **Autorización**: proceso que determina si el sujeto autenticado tiene permiso para acceder al recurso
- Este mecanismo debe impedir que los usuarios no autenticados accedan al sistema
- Generalmente, se utilizan algunas de estas medidas: contraseñas, biometría, tokens...

# 1.4 Seguridad en el entorno Java

## Seguridad en la máquina virtual de Java (JVM)

- La **máquina virtual de Java (JVM)** se encarga de ejecutar las aplicaciones Java
- El programa es interpretado y después se ejecutan los *bytecodes* (código objeto)
- Antes de realizar este proceso, la JVM realiza ciertas tareas previas
- Es en esta etapa donde intervienen los mecanismos de seguridad interna de Java

# 1.4 Seguridad en el entorno Java

## Seguridad en la máquina virtual de Java (JVM)

- Existen tres componentes en el proceso de seguridad:
  1. **Cargador de clases**: encuentra y carga los bytecodes que definen las diferentes clases
    - **Cargador Bootstrap** (carga las clases del sistema desde el **JAR rt.jar**)
    - **Cargador de clases de extensión**: carga una aplicación estándar desde el directorio **jre/lib/ext**
    - **Cargador de clases de aplicación**: localiza las clases y los ficheros **jar/zip** del **CLASSPATH**
  2. **Verificador de ficheros de clases**: valida los bytecodes (inicialización, comprueba referencias...)
  3. **Gestor de seguridad**: controla si ciertas operaciones están o no permitidas
- Normalmente no se instala un gestor de seguridad al ejecutar una aplicación Java

# 1.4 Seguridad en el entorno Java

## Seguridad en la máquina virtual de Java (JVM)

- Las propiedades del sistema se pueden leer con **System.getProperty(propiedad)**
- Existen dos formas de instalar un gestor de seguridad en la aplicación:
  - Iniciando la máquina virtual con la opción **-Djava.security.manager**
  - Invocando el método **setSecurityManager**: **System.setSecurityManager(new SecurityManager())**
- Más información sobre las propiedades del sistema: [enlace](#)

# 1.4 Seguridad en el entorno Java

## Seguridad en la máquina virtual de Java (JVM)

- Al ejecutar un programa Java por defecto se carga un fichero de políticas predeterminado
- Este otorga todos los permisos al código para acceder a propiedades comunes
- Otras propiedades, sin embargo, no reciben permisos de lectura
- Si el programa no tiene acceso, se producirá una excepción **AccessControlException**
- La plataforma define ciertas APIs para las áreas de seguridad de la aplicación:
  - **JCA** (*Java Cryptography Architecture*): desarrollo de funciones criptográficas
  - **JSSE** (*Java Secure Socket Extension*): comunicaciones seguras en Internet (**SSL** y **TLS**)
  - **JAAS** (*Java Authentication and Authorization Service*): control, autenticación y acceso

# 1.5 Ficheros de políticas en Java

## El fichero `java.policy`

- Los ficheros de políticas especifican los permisos disponibles para el código
- El valor por defecto es tener un solo fichero de políticas en todo el sistema y otro en el home
  - En sistemas **Windows**: `java.home\conf\security\java.policy`
  - En sistemas **UNIX/Linux**: `java.home /conf/security/java.policy`
- Es posible iniciar una aplicación con un fichero personalizado:
  - `java -Djava.security.policy=fichero.policy aplicación`

# 1.5 Ficheros de políticas en Java

## Formato del fichero de políticas

- **codeBase**: indica la ubicación del código base sobre la que se definen los permisos (**URL**)
- **nombreClase**: nombre de la clase de permisos (*java.io.FilePermission*, *java.net.SocketPermission*)
- **nombreDestino**: especifica el destino del permiso (depende de la clase de permiso)
- **acción**: indica una lista de acciones (separadas por coma)
  - *read*, *write*, *delete* o *execute* (para ficheros)
  - *accept*, *listen*, *connect*, *resolve* (para sockets)
  - *read* o *write* (para propiedades)

# 1.5 Ficheros de políticas en Java

## Formato del fichero de políticas

```
grant codeBase "file:C:/" {  
    permission java.io.FilePermission "C:\\\\*", "write";  
};
```

- Este fichero permite a los programas de **C:/** escribir y crear ficheros en **C:/**
- **Más información sobre las políticas:** [enlace](#)



# 1.6 Criptografía en Java

## JCA (Java Cryptography Architecture)

- El componente **JCA** es una parte importante de la plataforma Java
- Contiene una arquitectura proveedor y un conjunto de **APIs** para firmas, resúmenes, cifrado...
- El **API JCA** incluye la extensión criptográfica **JCE – Java Cryptography Extension**
  - Implementa servicios criptográficos: **java.security.\*** y **javax.crypto.\***
  - Proporciona proveedores de implementaciones criptográficas reales

# 1.6 Criptografía en Java

## Resumen de mensajes

- Un resumen de un mensaje (también conocido como hash) es una marca digital
- Como hemos comentado, existen multitud de algoritmos (SHA-1, MD5, SHA-256, SHA-512...)
- La clase **MessageDigest** permite aplicar algoritmos de resumen: [enlace](#)

# 1.6 Criptografía en Java

## Generación y verificación de firmas digitales

- Como ya se ha estudiado, la arquitectura de firmas digitales proporciona mayor seguridad
- La clase **KeyPairGenerator** permite generar claves públicas y privadas: [enlace](#)
- La clase **KeyPair** es una clase soporte para generar claves pública y privada: [enlace](#)
- Las interfaces **PrivateKey** y **PublicKey** agrupan la funcionalidad de las claves: [enlace](#) y [enlace](#)
- Finalmente, la clase **Signature** permite realizar la firma de los datos: [enlace](#)

# 1.6 Criptografía en Java

## La herramienta keytool

- Permite generar un par de claves pública y privada

```
C:\Users\Luis>keytool
Key and Certificate Management Tool

Commands:

-certreq          Generates a certificate request
-changealias     Changes an entry's alias
-delete          Deletes an entry
-exportcert       Exports certificate
-genkeypair       Generates a key pair
-genseckey        Generates a secret key
-gencert          Generates certificate from a certificate request
-importcert       Imports a certificate or a certificate chain
-importpass       Imports a password
-importkeystore   Imports one or all entries from another keystore
-keypasswd        Changes the key password of an entry
-list            Lists entries in a keystore
-printcert        Prints the content of a certificate
-printcertreq     Prints the content of a certificate request
-printcrl         Prints the content of a CRL file
-storepasswd      Changes the store password of a keystore
-showinfo         Displays security-related information
```

# 1.6 Criptografía en Java

## Encriptación y desencriptación con clave secreta

- La clase **Cipher** permite encriptar y desencriptar información: [enlace](#)
- La clase **KeyGenerator** genera claves secretas (para algoritmos simétricos): [enlace](#)

# 1.6 Criptografía en Java

## Encriptación y desencriptación con clave pública

- En el cifrado de clave pública se puede compartir sin problemas la clave para encriptar
- La clave para desencriptar solo la conoce el receptor
- Pasos para transmitir información:
  - A crea su par de claves y manda la clave pública a B
  - B crea su par de claves y manda su clave pública a A
  - A crea un mensaje, lo cifra con la clave pública del receptor y lo envía a B
  - B crea un mensaje, lo cifra con la clave pública de A y lo envía a A

# 1.6 Criptografía en Java

## Encriptar y desencriptar flujos de datos

- JCA proporciona varias clases que encriptan o desencriptan flujos de datos
- La clase **CipherOutputStream** se compone de un **OutputStream** y un **Cipher**: [enlace](#)
- La clase **CipherInputStream** se compone de un **InputStream** y un **Cipher**: [enlace](#)

# 1.7 Comunicaciones seguras con Java

## JSSE (Java Secure Socket Extension)

- Estos paquetes permiten desarrollar aplicaciones seguras en Internet
- Proporcionan una implementación Java de los protocolos SSL y TLS
- Las clases **SSLSocket** y **SSLServerSocket** representan sockets seguros: [enlace](#) y [enlace](#)



# 1.8 Control de acceso con Java

## **JAAS (Java Authentication and Authorization Service)**

- Esta interfaz permite a las aplicaciones Java acceder a servicios autenticación y acceso
- Puede usarse para autenticar usuarios o para autorizar usuarios
- Proporcionan una implementación Java de los protocolos SSL y TLS
- Ahora analizaremos varios ejemplos complejos de JAAS

# Créditos de las imágenes y figuras

## Cliparts e iconos

- **Obtenidos mediante la herramienta web [IconFinder](#)** (según sus disposiciones):
  - Diapositivas 1
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free comercial use)
  - A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

## Diagramas, gráficas e imágenes

- Se han desarrollado en PowerPoint y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
- Para el resto de recursos se han especificado sus fabricantes, propietarios o enlaces
- Si no se especifica copyright con la imagen, entonces es de desarrollo propio o CC0

# Créditos de los proyectos referenciados

## Proyecto Cifrador

- Proyecto desarrollado por José Luis González Sánchez (2019)
- Enlace de GitHub: <https://github.com/joseluisgs/Cifrador2019>
- A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

## Proyecto Servidor-Cliente Seguro

- Proyecto desarrollado por José Luis González Sánchez (2019)
- Enlace de GitHub: <https://github.com/joseluisgs/ServidorClienteSeguro2019>

## Licencia de los proyectos

- En todos los proyectos referenciados se ha citado debidamente a su autor o autores
- Portal GitHub del desarrollador (o desarrolladores): <https://github.com/joseluisgs?tab=repositories>