Tema 5, módulo 4: infraestructuras de clave pública

Programación Segura
grado en ingeniería informática

Curso académico 2020-2021

Objetivos del módulo

- Presentar algunas utilidades incluidas en la plataforma *Java SE* para facilitar la operación con claves criptográficas.
- Combinar su empleo con la utilidad openssl para crear infraestructuras corporativas de clave pública.

Índice

1. PKI

Seguridad basada en criptografía de clave pública

- Los sistemas de clave pública se emplean actualmente en diversas funciones de seguridad.
  - Identificar usuarios de cuentas de correo
  - Asegurar comunicación entre sitios *web*
  - Garantizar origen e integridad de *applets* de *Java*.
- Se hace necesaria una infraestructura que proporcione soporte al desarrollo de esas funciones.

PKI (*Public Key Infrastructure*)

- Designa genéricamente cualquier marco de trabajo basado en criptografía de clave pública preparado para el intercambio seguro de información.
  - Debe permitir asociar identidades de sujetos particulares u organizaciones a certificados digitales.
  - Debe proporcionar medios para verificar la autenticidad de esos certificados.
- Engloba elementos diversos como claves simétricas, pares criptográficos, certificados de clave pública y autoridades de certificación.
- *JCA* especifica una serie de servicios[1] para operar y gestionar con claves y con certificados digitales X.509.
- Toda plataforma *Java SE* debe proporcionar una colección de utilidades para creación y gestión de claves y firmado de documentos.

Keystores

- Son depósitos con material criptográfico generado para garantizar autenticidad, integridad y confidencialidad de datos manejados por aplicaciones desarrolladas en *Java*.
  - Su contenido se protege mediante encriptación y contraseñas de acceso.
- Cada usuario posee su propio *keystore*.
  - Con permisos de acceso restringidos a su propietario (denegados para cualquier otro usuario).
- Dos formatos de almacenamiento diferentes:
  - PKCS#12 que es el formato estándar; empleado por la utilidad `openssl`.
  - JKS que es un formato propio de *Java* empleado por defecto por la utilidad `keytools`; desaconsejado actualmente.

Tipos de entradas en un *keystore*

- *SecretKeyEntry*: clave secreta empleada en métodos criptográficos de clave simétrica; se puede proteger mediante contraseña particular.
- *PrivateKeyEntry*: clave privada acompañada de la cadena certificada para la correspondiente clave pública; se puede proteger mediante contraseña particular.
- *TrustedCertificateEntry*: certificado de clave pública de otra entidad.

Utilidades de *Java SE*

- Toda distribución regular de la plataforma *Java SE* debe incluir un conjunto de utilidades estándar para manejar claves criptográficas, certificados y firmas digitales.
- Tres de esas utilidades:
  - `keytool`: sirve para gestionar pares criptográficos y certificados de claves públicas.
  - `jar`: sirve para crear contenedores con los documentos[2] a transmitir.
  - `jarsigner`: sirve para firmar digitalmente contenedores de documentos.

[1] E incluye implementaciones de proveedores.
[2] Corrientemente documentos de código.

Documentación en línea

- Utilidad `keytool`:

  http://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html

- Utilidad `jar`:

  http://docs.oracle.com/javase/8/docs/technotes/tools/windows/jar.html

- Utilidad `jarsigner`:

  http://docs.oracle.com/javase/8/docs/technotes/tools/windows/jarsigner.html

- Utilidad `openssl`:

  https://www.openssl.org/docs/manmaster/man1/openssl.html

- Especificaciones `PKCS`:

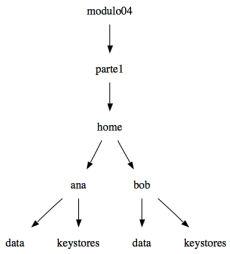
  https://en.wikipedia.org/wiki/PKCS

2. Gestión de certificados

2.1. Actividad 1: generación de claves

Actores y escenario

- Intervienen dos actores (con papeles intercambiables):
  - *Ana* firma como emisor
  - *Bob* verifica como receptor
- Los diferentes ficheros a crear durante este ejercicio se deben guardar en la estructura mostrada.



- Se debe dejar una copia del fichero de datos *Ligeia.txt* en el directorio *ana/data*.

Generación de pares criptográficos

- *Ana* opera en su directorio *keystores*.
- Teclea

```
lunik:> keytool -genkeypair -storetype pkcs12
          -alias ana -keystore ana.keystore
```

para generar un par criptográfico para firma *DSA*.

- La orden para generar un par es `-genkeypair`.

- Teclea

```
lunik:> keytool -genkeypair -keyalg rsa -keysize 4096
          -alias ana-rsa -keystore ana.keystore
```

para generar un par criptográfico de cifra *RSA*.

- La opción `-keyalg` permite indicar para qué algoritmo se debe generar el par.

- Si en el momento de generar el par criptográfico no existe ningún *keystore* con el nombre indicado, la utilidad `keytool` automáticamente se encarga de crear uno.
  - Por defecto, el formato de almacenamiento empleado es JKS.
  - Si se prefiere el formato de almacenamiento definido en el estándar *PKCS#12* hay que indicarlo.
  - Una vez creado el *keystore*, toda nueva entrada se almacena en el formato establecido.

- Los diversos elementos contenidos dentro de un *keystore* se identifican mediante etiquetas denominadas *alias*.
  - En el *keystore* creado hay una entrada identificada como *ana* y otra identificada como *ana-rsa*.

- Se puede teclear

```
lunik:> keytool -list -v -keystore ana.keystore
```

para obtener un listado con las diferentes entradas de un *keystore*.

Exportación de certificados

- *Ana* dispone ahora de un par criptográfico con el que firmar los documentos que quiera transmitir.

- Para que los receptores de sus documentos puedan verificar su autenticidad es necesario haberles hecho llegar previamente la clave pública del par.

- Esa clave pública se proporciona en forma de certificado.
  - *Bob* deberá guardar ese certificado en un *keystore* local.
  - Recuperará ese certificado cada vez que reciba un documento firmado remitido por *Ana*.

- *Ana* teclea

```
lunik:> keytool -export -file ana.cert
          -alias ana -keystore ana.keystore
```

para extraer el certificado de clave pública asociado al alias *ana*.

- Se puede teclear

```
lunik:> keytool -printcert -v -file ana.cert
```

para ver el contenido del certificado.

Importación del certificado

- *Ana debe hacer llegar el fichero con el certificado a Bob a través de un medio seguro.*
- Una vez que *Bob* dispone del certificado expedido por *Ana*, lo deposita en uno de sus *keystores* de confianza.
- Operando en su directorio *keystores*, *Bob* teclea

```
lunik:> keytool -import -file ana.cert -storetype pkcs12
          -alias ana -keystore bob.truststore
```

  para almacenar el certificado recibido con alias *ana*.
- Ese certificado se mantiene mientras no caduque y *Ana* no notifique un cambio de su par criptográfico para firma.

2.2. Actividad 2: operación con documentos

Creación del contenedor de documentos

- El documento a firmar es *Ligeia.txt*.
- Operando en el directorio *data*, *Ana* teclea

```
lunik:> jar cvf Ligeia.jar Ligeia.txt
```

  para crear un fichero compacto que sirva como contenedor de los diferentes documentos a firmar.
- Se puede teclear

```
lunik:> jar tvf Ligeia.jar
```

  para obtener un listado de los elementos embalados dentro del fichero *jar*.
- Documentación en línea:

http://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html

Firma del contenedor de documentos

- Para firmar el fichero *Ligeia.jar* se emplea la clave privada del par criptográfico identificado por el alias *ana* guardado en el *keystore* *ana.keystore*.
- *Ana* teclea

```
lunik:> jarsigner -keystore ../keystores/ana.keystore
          -signedjar sLigeia.jar Ligeia.jar ana
```

  - Para acceder al *keystore* *ana.keystore* se solicita la contraseña general.

    - Para emplear el par criptográfico asociado al alias *ana* se solicita la contraseña particular de esa entrada.

- La firma se genera empleando la clave privada del par asociado al alias indicado.
- Documento original, firma generada y certificado de clave pública son embalados en el fichero firmado *sLigeia.jar*.
- Se puede comprobar listando su contenido:

```
lunik:> jar tvf sLigeia.jar
```

Verificación del documento

- *Bob* necesita el certificado de clave pública emitido por *Ana* para verificar el documento firmado enviado por ella.
- Operando desde su directorio *data*, *Bob* teclea

```
lunik:> jarsigner -verify
          -keystore ../keystores/bob.truststore sLigeia.jar
```

- Las opciones `-verbose -certs`

```
lunik:> jarsigner -verify -verbose -certs
          -keystore ../keystores/bob.truststore sLigeia.jar
```

  proporcionan información de salida que indica:

    - Si los documentos de datos se encuentran en el listado resumen (*manifiesto*).
    - Si clave pública empleada para verificar la firma se encuentra entre las claves almacenadas en el *keystore*.
    - Información sobre el certificado de clave pública empleado en la verificación.

3. Certificación de confianza

3.1. Relación de confianza

Confianza

Confianza
Esperanza firme que se tiene en la integridad de alguien.

- Genera una relación especial entre personas.
- Esa relación puede ser transitiva.
    - Si *Ana* confía en *Bob* y *Bob* confía en *Charlie*, entonces *Ana* podría acceder a confiar en *Charlie*.
    - Es la base de las cartas de recomendación.
- Puede llevar a abusos.

- Se puede generar ilusoria o falsamente.

Ingeniería social
Elaboración y puesta en práctica de métodos de manipulación persuasiva para inducir en las víctimas una confianza infundada.

- Los ingenieros sociales crean situaciones que inducen a bajar la guardia.
  - Cuento del lobo y los siete cabritillos
- La falsificación de claves por parte de un suplantador anula la seguridad proporcionada por los métodos basados en sistemas clave pública.
  - Ataque MITM

Ataque MITM (*meet in the middle*)

- *Ana* y *Bob* son dos sujetos que no se conocen.
- *Ana* y *Bob* en algún momento podrían querer intercambiar información a través de un canal de transmisión público.
- *Eva* se anticipa y crea un par criptográfico.
- Haciéndose pasar por *Ana*, envía la clave pública a *Bob*.
- Y viceversa; haciéndose pasar por *Bob*, envía la clave pública a *Ana*.
- Ahora *Eva* está en situación de interceptar y manipular cualquier transmisión de información entre *Ana* y *Bob* sin que éstos lo adviertan.

Primer principio: *trust no one*

¡Aprende la lección, sigue mis principios!

Excepción al primer principio

Principio de confianza
Cuando se reciben datos de una entidad cuya identidad es posible acreditar y se confía en esa entidad, entonces se puede asumir que esos datos son genuinos.

- Si por alguna razón se omite la obligada acreditación de la entidad emisora entonces un suplantador podría violar la seguridad que se deriva del principio de confianza.

Principio de confianza respecto a claves

- El material a depositar en un *keystore* se puede separar en dos categorías distintas en función del grado de confianza.
  - Claves propias: consisten en una identificación de la propia entidad junto con la clave privada asociada a esa identificación.
  - Certificados de confianza externos: consisten en una clave pública junto con la identificación de la entidad que la generó/proporcionó.
- Para reasaltar la separación que se deriva de la distinción entre certificados propios[3] y certificados externos se ha visto útil emplear dos *keystores*.
  - Uno contiene claves y certificados de clave propios.
  - Otro contiene certificados externos de confianza así como certificados de autoridades de certificación.

Truststores

- Son *keystores* que sólo guardan certificados externos que sirven para determinar en qué entidades externas se confía.
  - Sirven como lista blanca.
- El propietario de un *truststore* sólo debiera añadir al mismo entradas de aquellas entidades en las que haya decidido confiar.
- Para eso debiera estar seguro de que no ha sido burlado con una artimaña del tipo MITM.

3.2. Certificación autorizada

Claves certificadas

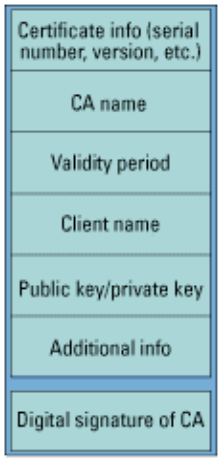
- Es necesario garantizar la propiedad y autenticidad de las claves públicas empleadas.
- Para gozar de suficiente confianza toda clave pública debe estar certificada.
  - Debe haber sido emitida por una entidad autorizada para su reconocido y exclusivo empleo por parte de un cierto particular.
  - El certificado asegura que la clave pública es propiedad del titular indicado en el certificado.
  - Es la forma de asegurar al receptor de una clave pública que no está siendo engañado con una artimaña MITM.
- Con ese objetivo se han desarrollado los conceptos de certificado digital y autoridad de certificación.

[3] Y sus correspondientes claves privadas.

- Autoridad de certificación: organismo público o empresa privada especializada en seguridad digital, de confianza reconocida nacional o internacionalmente (*trust center*), que expide certificados que asocian claves públicas con usuarios.
- Certificado digital: documento expedido y firmado digitalmente por una autoridad de certificación que acredita que cierta clave pública ha sido emitida por cierto usuario[4].
  - Sirve como credencial digital.
  - Basta verificar la firma que incluye para saber si es auténtico o ha sido falsificado.

Componentes de un certificado digital

- CA emisora (*issuer*): si un usuario confía en la entidad que expide el certificado y éste es válido (es posible verificar la firma que incluye), entonces el usuario puede confiar en el certificado.
- Propietario (*owner*): información sobre la identidad de la entidad titular a la que se refiere el certificado.
- Periodo de validez: sus fechas deben ser examinadas cuando se evalúa la validez de un certificado.
- Clave pública que se certifica: todos lo demás campos intentan asegurar la validez de éste.
- Firma digital de todo lo anterior: generada por la entidad emisora usando su clave privada.



Autocertificados

- Cada vez que `keytool` genera un nuevo par criptográfico, la clave pública queda autocertificada durante un plazo limitado.
- La utilidad práctica de ese autocertificado se reduce al ámbito del desarrollo y prueba de aplicaciones.
  - Los autocertificados no generan confianza y no suelen ser aceptados.

Autoridad de certificación (*Certification Authority*, CA)

- Respaldada por una bien fundada relación de confianza.
  - Si se confía en cierta CA, automáticamente se confía en todos los certificados expedidos por esa CA.

[4] Una autoridad de certificación puede también expedir un par criptográfico para que sea empleado por un usuario titular y proporcionarle un certificado de la componente pública.

- Expide una serie de certificados raíz que distribuye controladamente en su ámbito de acción.
- Un certificado raíz de una CA es un autocertificado de la clave pública con la que se puede verificar la firma de los certificados expedidos por esa CA.
- La confianza convenida en la CA permite que cualquier usuario acepte un certificado cuya firma pueda ser verificada con la clave pública incluida en el certificado raíz de una CA.
- Un *web browser* usualmente incorpora una colección de certificados de varias autoridades raíz.
- Existen compañías que basan su negocio en la emisión de certificados.
  - VeriSign, Entrust o GTE CyberTrust expiden comercialmente (bajo pago) certificados a propietarios.
  - CaCert expide certificados gratuitamente.
- La FNMT-RCM expide gratuitamente certificados a ciudadanos particulares que son aceptados por los servicios telemáticos de todas las administraciones del estado español.

Obtención de una certificado expedido por una CA

- Con la solicitud es necesario aportar una prueba de identidad a la CA.
  - Esa identidad queda asociada/representada a un nombre de usuario que debe ser único en el contexto de la CA.
- Una autoridad de registro (RA) debe verificar suficientemente que el solicitante representa a la organización o persona física que afirma que representa.
- Conseguida esa verificación, la CA expide y firma un certificado que testimonia la validez de la información contenida dentro del certificado.
- En general, es necesario pagar por el servicio.

Cadenas de certificados

- La propiedad transitiva de una relación de confianza permite enlazar múltiples certificados en una cadena.
- En un certificado que incluye una cadena de certificación:
  - El primer certificado C_0 es siempre del usuario que informa sobre su clave pública.
  - Cada certificado intermedio C_j, con $1 \leq j \leq n$, en la cadena pertenece a la autoridad que expidió el previo.
  - El certificado final es el certificado raíz de una autoridad de certificación.
- La transitividad de una relación de confianza permite confiar en la validez del primer certificado:

$$\text{CA raíz} \overset{\text{certifica}}{\rightsquigarrow} C_n \overset{\text{certifica}}{\rightsquigarrow} \cdots \overset{\text{certifica}}{\rightsquigarrow} C_1 \overset{\text{certifica}}{\rightsquigarrow} C_0$$

Revocación de certificados

- Es obligatorio que una CA revoque un certificado expedido cuando:
  - La clave privada del usuario ha sido comprometida.
  - Se descubre que ha expedido un certificado a un usuario erróneo o que ha falsificado su identidad.
  - El usuario ha perdido el derecho a acceder a un servicio para cuya identificación solicitó el certificado.
  - La estructura de la CA ha sido comprometida de forma que terceros puedan expedir certificados falsos.
- Dos modos de gestionar revocaciones:
  - Búsqueda en listas de revocación de certificados (CRL) mantenidas por las autoridades de certificación.
    - Son listas negras.
  - Verificación en tiempo real recorriendo la cadena de certificación.

Utilidad `openssl`

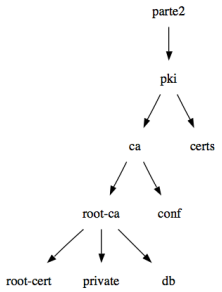
- No siempre es obligatorio pagar por un certificado expedido por una CA reconocida externamente.
- En la red interna de una organización basta con certificados expedidos por una autoridad local.
  - Servidor *web* interno que opere con protocolo *https*
  - Red privada virtual
- En esa situación es aceptable que una autoridad local expida con la utilidad `openssl` certificados válidos para operar en ese entorno.
- Documentación *online*:

`https://www.openssl.org/docs/apps/openssl.html`

3.3. Actividad 3: creación de una CA local

Escenario

- Los diferentes ficheros a crear durante este ejercicio se deben guardar en una estructura como la mostrada.



- Se deben dejar copias de los ficheros de configuración `root-ca.conf` y `signing-ca.conf` proporcionados en el directorio *pki/ca/conf*.

Certificados emitidos y números de serie

- Un fichero de texto con información del estado de todos los certificados expedidos por la autoridad de certificación es mantenido automáticamente por la utilidad `openssl`.
  - Fichero `pki/ca/root-ca/db/root-ca.db`
- Otro fichero guarda el número de serie que se asignará al siguiente certificado a expedir.
  - Fichero `pki/ca/root-ca/db/root-ca.crt.srl`
- Ambos ficheros deben existir antes de crear la CA.
- Desde el directorio *pki*, el administrador de certificados teclea:

```
lunik:> touch ca/root-ca/db/root-ca.db
lunik:> touch ca/root-ca/db/root-ca.crt.srl
lunik:> echo 01 > ca/root-ca/db/root-ca.crt.srl
```

Solicitud de certificado y clave base

- Se necesita una clave raíz que sirva como base para firmar todos los certificados que se expidan.
- Se debe mantener extremadamente protegida para evitar que un suplantador pueda expedir certificados no autorizados que no se distingan de los válidos.
- Además se debe generar una petición de certificación de esa clave (*certification signing request*)
- Se obtiene operando como administrador de certificados:

```
lunik:> openssl req -new
      -out ca/root-ca/root-ca.csr
      -config ca/conf/root-ca.conf
      -keyout ca/root-ca/private/root-ca.key
lunik:> chmod 400 ca/root-ca/private/root-ca.key
```

Expedición de certificado raíz

- El certficado raíz es el punto de partida de todas las relaciones de confianza basadas en esta infraestructura.
- Se genera en base al fichero *csr* producido al generar la clave privada de la autoridad.
- Va autofirmado.
- Se obtiene operando como administrador de certificados:

```
lunik:> openssl ca
      -selfsign -verbose
      -config ca/conf/root-ca.conf
      -extensions root_ca_ext
      -in ca/root-ca/root-ca.csr
      -out ca/root-ca/root-cert/root-ca.pem
```

- Ahora ya se dispone de un certificado raíz para actuar como una autoridad local de certificación.
- La configuración especificada `root-ca.conf` indica que el certificado raíz se deposite en el directorio `pki/ca/root-ca/root-cert`.
- Se pueden hacer las comprobaciones siguientes:
  - El fichero `pki/ca/root-ca/db/root-ca.db` incluye una entrada que corresponde al certificado raíz expedido; su número de serie es 01.
  - El número de serie guardado en el fichero `pki/ca/root-ca/db/root-ca.crt.srl` ahora es 02.
  - El fichero `pki/ca/root-ca/root-cert/01.pem` muestra la información del certificado raíz generado.
- El fichero `pki/ca/root-ca/root-ca.csr` con la solicitud de certificación ya no se necesita.
  - Se debe eliminar.

3.4. Actividad 4: actuación como CA local

Solicitud de certificación de clave pública particular

- *Ana* necesita un certificado válido, por lo que genera una petición de certificación para una de sus claves públicas.
- Desde el directorio de *Ana*:

```
lunik:> keytool -certreq
        -alias ana -keystore keystores/ana.keystore
        -file ana.csr
```

- *Ana* envía esa petición al administrador de certificados:

```
lunik:> mv ana.csr ../pki/certs
```

Expedición de certificado de clave pública

- El administrador de certificados expide un certificado en respuesta a la petición:

```
lunik:> openssl ca
      -verbose
      -in certs/ana.csr
      -out certs/ana.pem
      -config ca/conf/signing-ca.conf
      -extensions client_ext
```

- Obsérvese que en este caso se especifica el fichero de configuración `ca/conf/signing-ca.conf` preparado para expedir certificados de usuario.
- Se puede verificar el certificado expedido:

```
lunik:> openssl verify
      -CAfile ca/root-ca/root-cert/root-ca.pem
      certs/ana.pem
```

- El administrador de certificados hace llegar de forma segura a *Ana* una copia de su certificado de clave pública.

Recepción de certificado de clave pública

- *Ana* recibe el certificado en formato *PEM*.
- El formato *PEM* es un formato legible que no puede ser depositado en un *keystore*.
- Se requiere una conversión a formato *DER* para almacenamiento:

```
lunik:> openssl x509 -in ana.pem -out ana.crt
```

- *Ana* ahora puede almacenar el certificado válido en su *keystore*:

```
lunik:> keytool -import -keystore ana/keystores/ana.keystore
      -alias ana-cert -file ana.crt
```

- Más importante aún, *Ana* puede distribuir ese certificado entre aquellos agentes (que confien en la autoridad local) a los que envíe documentos firmados.