Uso del API de cifrado de JAVA (JCA/JCE)

SSI 2018/19

18 de septiembre de 2018

Índice

1. Objetivos **1**

2. Descripción **1**

2.1. Simplificaciones 1

2.2. Requisitos 2

3. Desarrollo **3**

3.1. Actores 3

3.2. Programas/módulos a desarrollar 3

4. Entrega **4**

5. Herramientas a utilizar **5**

1. Objetivos

- Poner en práctica los conocimientos adquiridos respecto a algoritmos criptográficos
- Conocer y utilizar un API estándar para el desarrollo de aplicaciones criptográficas de complejidad media
  - En este caso se hará uso del API *Java Cryptography Architecture (JCA)* y del provider (implementación de JCA) BouncyCastle.

2. Descripción

Se trata de desarrollar una herramienta para el empaquetado y distribución de exámenes que sea fiable y segura y garantice las restricciones de entrega en plazo.

Para ello se contará con una especie de *Autoridad de sellado de tiempo* simplificada (ver Sellado de tiempo)

Los alumnos podrán generar su *Examen Empaquetado* que será remitido a la "autoridad de sellado" para vincularle el *timestamp* con la fecha de entrega. Finalmente, el profesor podrá validar este *Examen Empaquetado* para extraer los datos aportados por el alumno y validar la autenticidad del "sello" emitido por la "autoridad de sellado".

2.1. Simplificaciones

Dado que se trata de una aplicación "de juguete" se asumirán una serie de simplificaciones.

- Cada uno de los participantes (ALUMNO, AUTORIDAD SELLADO, PROFESOR) podrá generar sus propios pares de claves privada y pública que se almacenarán en ficheros (no se considera una protección del fichero con la clave privada, como sí ocurriría en una aplicación real)
- No se contemplan los mecanismos de distribución de claves públicas. Se asumirá que todas las claves públicas necesarias estarán en poder del usuario que las necesite (ALUMNO, AUTORIDAD SELLADO o PROFESOR) de forma confiable (en una aplicación real se haría uso de certificados digitales y de una *autoridad de certificación* común)
  - El ALUMNO dispondrá de un fichero con la clave pública del PROFESOR
  - La AUTORIDAD SELLADO podrá disponer de un fichero con la clave pública del PROFESOR y, eventualmente, también podrá tener acceso al fichero con la clave pública del ALUMNO que desee "sellar" su *Examen Empaquetado*.
  - El PROFESOR contará con la clave pública (almacenada en su respectivo fichero) de la AUTORIDAD SELLADO, así como con la clave pública del ALUMNO para el cual se vaya a realizar la validación de su *Examen Empaquetado*.
- Las respuestas del ALUMNO estarán almacenadas inicialmente en un fichero de texto (es indiferente la estructura de ese fichero)
- Dado que se trata de un ejemplo, las distintas piezas de información que aporte cada participante (ALUMNO o AUTORIDAD SELLADO) al *Examen Empaquetado* tendrán (antes del cifrado/firma y después del descifrado) la forma de *Strings* con codificación UTF8.
- El *Examen Empaquetado* se materializará físicamente en un fichero o "paquete" que contendrá toda la información que vayan incorporando los distintos participantes implicados: el ALUMNO que la generó y la AUTORIDAD SELLADO que da fé de la entrega del examen y del instante concreto en ésta que tuvo lugar.

  Nota:
  - Se aporta código para la gestión de "Paquetes" con múltiples partes codificadas en bloques de caracteres imprimibles empleando codificación Base64 (ver Codificación Base64)
  - **NECESARIAMENTE** se **deberá** emplear el código proporcionado o proporcionar una implementación propia equivalente en el caso de desarrollar la prácticas en otros lenguajes diferentes a Java.
- No se contempla un almacenamiento "físico" realista del *Examen Empaquetado*, sólo se trata de implementar los programas para generar, sellar y validar el *Examen Empaquetado* conforme a las especificaciones descritas en este documento.
- Al validar el *Examen Empaquetado*, si todas las comprobaciones de autenticidad respecto a ALUMNO y AUTORIDAD SELLADO son correctas, se mostrará al PROFESOR los datos aportados por el ALUMNO (el texto del examen) y los datos (*timestamp*) incorporados por la AUTORIDAD SELLADO que haya procesado dicho *Examen Empaquetado*. En caso contrario se indicarán las comprobaciones que no hayan sido satisfactorias.

2.2. Requisitos

Requisitos básicos a cumplir por el esquema criptográfico propuesto:

R1. Asegurar la **confidencialidad** del contenido incluido en la *Examen Empaquetado* por parte del ALUMNO (sólo el PROFESOR podrá tener acceso a estos contenidos).

- Opcionalmente se puede asegurar también la confidencialidad de la información incluida en el "sello" de la AUTORIDAD SELLADO (en caso de hacerlo así, sólo el PROFESOR podría tener acceso a estos contenidos), aunque no es imprescindible.

  Para la funcionalidad requerida en este proyecto bastará con dar soporte al "sellado" y esos datos (*timestamp*) pueden dejarse en claro en el "paquete" que da soporte físico al *Examen Empaquetado*.

R2. Garantizar que el PROFESOR tenga la **seguridad** de que el ALUMNO que presenta el *Examen Empaquetado* **es quien dice ser**.

R3. **Asegurar** que el contenido del "paquete" con el *Examen Empaquetado* (datos del ALUMNO y sello de AUTORIDAD SELLADO) que se ha recibido **no** haya sido **modificado**.

R4. Asegurar que ni el ALUMNO ni la AUTORIDAD SELLADO podrán **repudiar** el contenido incluido por ellos en el *Examen Empaquetado*

R5. **Asegurar** que el PROFESOR **no** podrá realizar **cambios** en el contenido del *Examen Empaquetado* que ha recibido.

R6. Contar con un mecanismo mediante el cuál la AUTORIDAD pueda **garantizar la fecha** en que fue "recibido" el *Examen Empaquetado* generado por un determinado ALUMNO.

Se pretende que esta vinculación entre *Examen Empaquetado* y "sello" pueda ser validada por el PROFESOR y que no pueda ser falsificada ni por el ALUMNO, ni por la AUTORIDAD SELLADO (o por otras AUTORIDADES SELLADO ajenas), ni por el propio PROFESOR

- **Nota:** En este caso estas AUTORIDADES SELLADO funcionarán de un modo parecido (aunque simplificado) a las **autoridades de sellado de tiempo** de las infraestructuras de clave pública (ver más en Sellado de tiempo [wikipedia]).

R7. Asegurar un **coste computacional reducido** en la creación, sellado y validación del *Examen Empaquetado* **minimizando** el uso de **criptografía asimétrica**

3. Desarrollo

En primer lugar se deberán de analizar los requisitos anteriores, para determinar qué estrategias seguir para conseguir cada uno de ellos.

Se debe decidir qué acciones realizar en el origen (ALUMNO), qué tareas realizará la AUTORIDAD SELLADO y qué comprobaciones se llevarán a cabo en el destino (PROFESOR), además de decidir qué algoritmos concretos se emplearán.

3.1. Actores

Alumnos. podrá generar su propio par de claves (pública y privada) y será responsable de **generar** el *Examen Empaquetado* a partir del fichero de texto con el examen en claro original.

Autoridad Sellado. podrá generar su propio par de claves (pública y privada) y será responsable de **sellar** el *Examen Empaquetado* de un ALUMNO dado, aportando los datos que correspondan.

Profesor. podrá generar su propio par de claves (pública y privada) y será responsable de extraer los datos aportados por el ALUMNO en el *Examen Empaquetado* que le haya enviado y validar la información incluida en el mismo por la AUTORIDAD SELLADO

3.2. Programas/módulos a desarrollar

Una vez decidido cómo garantizar los requisitos exigidos, el resultado final será **obligatoriamente** el desarrollo de 4 ejecutables:

- `java -cp [...] GenerarClaves <identificador>` (**ya** está **implementado**)
  - Usado para generar los pares de claves de los participantes: ALUMNO, AUTORIDAD SELLADO y PROFESOR
  - Se le pasa como argumento de línea de comando un `identificador` que se usará para componer los nombre de los archivos que se generarán
  - Genera dos ficheros: `''identificador.publica''` e `''identificador.privada''`, conteniendo, respectivamente, las claves pública y privada de ese usuario
- `java -cp [...] EmpaquetarExamen <fichero examen><nombre paquete><ficheros con las claves necesarias>`
  - Usado por el ALUMNO

- Se le pasa en línea de comandos un fichero de texto con el contenido del examen a enviar, el nombre del paquete resultante y los ficheros con las claves necesarias para el empaquetado (el número y tipo exacto de los ficheros de claves dependerá de que estrategia se haya decidido seguir).
- Genera el fichero `<nombre paquete>` (por ejemplo `examen.paquete`) con el resultado de "empaquetar" los datos que conforman el *Examen Empaquetado*.

- `java -cp [...] SellarExamen <nombre paquete><ficheros con las claves necesarias>`
  - Usado por la AUTORIDAD SELLADO
  - Se le pasa en línea de comandos el fichero con el "paquete" a sellar y los ficheros con las clave/s criptográficas necesaria/s.
  - Al "paquete" recibido como argumento le vincula (añade) los bloques que correspondan para incorporar los datos aportados por la AUTORIDAD SELLADO y para garantizar la autenticidad (y opcionalmente también la confidencialidad) de los datos de "sellado".
  - El resultado será el mismo fichero del "paquete" pasado como parámetro con los nuevos datos incorporados en forma de nuevos bloques.

- `java -cp [...] DesempaquetarExamen <nombre paquete><fichero examen><ficheros con las claves necesaria`
  - Usado por el PROFESOR
  - Se le pasa en línea de comandos el fichero con el "paquete" que representa al *Examen Empaquetado* (donde se incluyen los datos aportados por el ALUMNO y los datos de la AUTORIDAD SELLADO), el nombre del fichero donde se almacenará el examen en claro y los nombres de los ficheros con las claves que sean necesarias para desempaquetar y verificar la información que contiene el mencionado "paquete".
  - Al usuario (PROFESOR) se le indicará por pantalla el resultado de las comprobaciones que se hayan realizado sobre el *Examen Empaquetado* y se mostrarán los datos que incluye.
    - se indicará si los datos incluidos por el ALUMNO o por la AUTORIDAD SELLADO han sufrido modificaciones o no
    - se indicará si el "sello" de la AUTORIDAD SELLADO es válido/auténtico y, de ser así, se mostrará la fecha de sellado.
    - una vez verificado que el ALUMNO que generó el "paquete" es quien realmente corresponde, se descifrará el examen enviado y se almacenará en el fichero indicado el texto en claro incluido por el ALUMNO en su *Examen Empaquetado* (opcionalmente, también se puede presentar por pantalla)

4. Entrega

- Práctica **individual** o en **parejas**.
- **Documentación a entregar:**

  Memoria Contendrá al menos los siguiente puntos
  - Descripción breve de la práctica
  - Descripción y justificación de las estrategias criptográficas empleadas para asegurar los requisitos básicos exigidos, junto con los pasos que se siguen en el empaquetado, sellado y desempaquetado del examen.
  - Descripción del formato/estructura del "paquete" resultante
  - Descripción breve de la implementación: clases y métodos más importantes.
  - Instrucciones de compilación y ejemplos de uso.
  - Breve comentario sobre las simplificaciones realizadas (apartado 2.1) y sus consecuencias en una aplicación real.
  - Resultados obtenidos y conclusiones

Código fuente + (opcionalmente) casos de prueba

- **Aviso importante:**
  Salvo razones debidamente justificadas en la documentación (uso de otros lenguajes de programación o similares), los ejecutables desarrollados deben de seguir las restricciones respecto a nombre, parámetros de línea de comandos y orden de los mismos especificadas en la sección 3.2 ("Programas a desarrollar"). Del mismo modo, los datos aportados por ALUMNO y AUTORIDAD SELLADO han de al almacenarse en el "Paquete" creado.

- **Fecha de entrega:** por determinar
- Entrega en FAITIC (un único archivo con documentación, código, etc) [evitar incluir JAR de BouncyCastle y ficheros .class]
- **Defensa:** esta previsto que alrededor de un 10 %-15 % de las prácticas entregadas en tiempo y forma serán seleccionadas para una pequeña defensa (5-10 min.) ante el profesor
  - fundamentalmente serán prácticas con dudas de funcionamiento o implementación, aquellas que presentes dudas en cuanto a su autoría o entregas elegidas al azar
  - la defensa se realizará bien en horario de laboratorio o en horario de tutorias (se acordará con cada grupo)
- **Evaluación:**
  - Documentación correcta + empaquetado básico (examen del ALUMNO) y validación/descifrado por el PROFESOR: hasta 70 %
  - Adición de la generación y validación de los "sellos" de la *autoridad sellado*: hasta 100 %

5. Herramientas a utilizar

La práctica se implementará en Java utilizando el API de criptografía JCA y el provider Bouncy Castle. Se podrá realizar tanto en Windows como en Linux.

- API Java Cryptography Architecture (JCA) [antes JCE (Java Cryptography Extension)]
- Paquete de criptografia Bouncy Castle (JCA provider)
- Tutorial del API JCA (Java Cryptography Architecture) [pdf]

Código de partida (ver ejemplo de uso en `main()` de `PaqueteDAO.java`):

- Paquete.java: encapsula un paquete formado por varios bloques. Cada boque tiene un nombre (de tipo `String`) y un contenido (de tipo `byte[]`). La clase provee de métodos para añadir, listar y eliminar los bloques que conforman el paquete.
- Bloque.java: encapsula cada uno de los bloques que forman parte de un paquete.
- PaqueteDAO.java: métodos estáticos que dan soporte a las operaciones de lectura y escritura de paquetes empleando un formato similar al PGP *ASCII armor* (ver PGP) que usa la codificación Base64 para representar datos binarios mediante caracteres imprimibles.
  - Métodos públicos estáticos: `leerPaquete(...)` y `escribirPaquete(...)`
  - Ver en el método `main()` ejemplos de escritura y lectura de Paquetes.
  - Esta clase usa los objetos `org.bouncycastle.util.encoders.Base64` de la librería BouncyCastle para la conversión a/desde Base64, por lo que par compilar y ejecutar es necesario incluir en el CLASSPATH elcorrespondiente paquete JAR.
- GenerarClaves.java: genera un par de claves RSA de 512 bits