USO DEL API DE CIFRADO DE JAVA (JCA)

SEGURIDAD EN SISTEMAS INFORMÁTICOS

GRUPO-5

CIBRÁN CORES CABALEIRO
54506586J

ÍNDICE

1. INTRODUCCIÓN

El objetivo de esta práctica es aplicar los conocimientos adquiridos respecto a algoritmos criptográficos y usar el API Java Cryptography Architecture (JCA) y el provider BouncyCastle para 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. Los alumnos podrán generar su Examen Empaquetado que será remitido a la “autoridad de sellado”, que verificará la identidad del alumno que hace la entrega y le vinculará el timestamp que garantiza la fecha de entrega. Finalmente, el profesor podrá validar este Examen Empaquetado para verificar que procede del correspondiente alumno, extraer los datos aportados por el alumno y validar la autenticidad del “sello” emitido por la “autoridad de sellado”.

2. ESTRATEGIAS CRIPTOGRÁFICAS EMPLEADAS

En esta sección se describirán los pasos que se siguen en el empaquetado, sellado y desempaquetado del examen y se justificarán las estrategias criptográficas empleadas para asegurar que se cumplen los requisitos básicos exigidos.

Emplearemos el cifrado DES, el cifrado RSA y la firma digital.

EMPAQUETADO

El alumno cifrará el examen con una clave DES generada aleatoriamente de 56 bits y también cifrará esa clave con la pública del profesor en un cifrado RSA. Luego el examen cifrado y la clave cifrada se introducirán en un paquete en un bloque cada uno. Después se añadirá la firma digital del alumno (con su clave privada) del examen cifrado y la clave cifrada, añadiendo así un tercer bloque al paquete.

SELLADO

Luego este paquete es validado por la autoridad de sellado y lo rechaza si hay un error en la firma digital. Si la firma es válida, la autoridad realiza una firma digital (con su clave privada) en el que participarán los bloques del paquete más la fecha codificada en un array de bytes y este sello se añadirá en un bloque al paquete. También, se añadirá la fecha en el formato anterior en un bloque al paquete.

DESEMPAQUETADO

Finalmente, el nuevo paquete es validado por el profesor, comprueba si el sello y la firma digital del alumno son correctos y en ese caso obtenemos la fecha en la que el examen fue sellado junto con la clave DES cifrada y el examen cifrado. Con la clave privada del profesor se descifra la clave DES y con esta se descifra el examen, obteniendo el examen realizado por el alumno listo para corregir.

Se cumple el requisito **R1**, debido a que el alumno cifró la clave necesaria para cifrar y descifrar el examen con la clave pública del profesor por lo que suponemos que el profesor es el único que conoce su clave privada para poder descifrar el examen y así obtener el examen limpio.

También se cumple el **R2**, ya que el Examen Empaquetado fue firmado digitalmente por el alumno con su clave privada (suponemos que solo el alumno tiene acceso a su clave), por lo que cualquiera que tenga acceso a su clave pública pueda descifrarlo por lo que tanto la autoridad de sellado como el profesor tienen acceso a ver que fue presentado por ese alumno. Además se cumple el **R7** por lo anterior ya que lo verifica con la clave pública.

Como el sellado y la firma del alumno se han realizado como firmas digitales y los contenidos también están dentro del paquete se puede comprobar que no hubo modificaciones ya que si cambias la fecha del sellado no coincidirá con el sellado y habrá un error, por lo que se cumple el **R3**. Gracias a esto también se cumplen los **R4** y **R5** ya que una vez hacen las firmas digitales ya no pueden modificarse y si el profesor intentase modificar algo del examen empaquetado no coincidiría y saltaría una advertencia a la hora de validar el examen.

Cuando la autoridad de sellado recibe el paquete hace las comprobaciones necesarias y realiza el sellado con la fecha y hora de ese instante y gracias a que el sellado es como una firma digital si alguien cambiase la fecha habría un error al verificar el sellado por lo que se cumple el **R6**.

Para cifrar el examen que es el contenido más grande se utilizará una clave simétrica DES de 56 bits ya que un cifrado asimétrico RSA nos llevaría muchos más cálculos y ralentizará el programa por lo que luego vamos a cifrar la clave DES mediante una clave RSA de 512 bits (que es muy segura) ya que solo estaríamos cifrando un contenido más pequeño. Y luego las firmas digitales son necesarias para poder verificar la autenticidad. Por todo esto también se cumple el **R8**.

3. FORMATO PAQUETE

Tras ejecutar EmpaquetarExamen.java, se obtiene un primer paquete con el examen empaquetado y tras SellarExamen.java, un paquete que tiene el sellado de tiempo incluido al anterior. Para que sea más fácil explicar como es el formato del paquete he decidido hacer el gráfico siguiente:

Siendo el de la izquierda el formato del paquete resultado tras ejecutar EmpaquetarExamen.java y el de la derecha tras ejecutar SellarExamen.java y realizar el sellado de tiempo.

Si una vez terminado el sellado tenemos un archivo <nombre_paquete>.paquete con el siguiente formato:

-----BEGIN PAQUETE-----
-----BEGIN BLOQUE CLAVE_CIFRADA-----
DQCSgi8jmfrbgkJkLO6IDplyAR3KcZVcNxP1XzB7jPYzDD+HF9ctQk1LUwQM1…
-----END BLOQUE CLAVE_CIFRADA-----
-----BEGIN BLOQUE EXAMEN_CIFRADO-----
ePagRTUUndTACIY6jEuMzNSASAICsrKJoleH1c/6Qd1pMMyD+Jqaf/jGLX6rs8i…
-----END BLOQUE EXAMEN_CIFRADO-----
-----BEGIN BLOQUE FECHA_HORA-----
V2VkIE9jdCAxMiAwMTowNDowOSBDRVNUIDIwMjI=
-----END BLOQUE FECHA_HORA-----
-----BEGIN BLOQUE HASH_ALUMNO-----
XgBWEcvne0OCx2pf5HLBBngOdWXwA0uifHvBpqcBnZNFUGLLT0ORqC28Sn3…
-----END BLOQUE HASH_ALUMNO-----
-----BEGIN BLOQUE SELLADO_TIEMPO-----
fph3BKzh34rMVJPVqfKgdQOjM1lnz+iSSSRk+N3OJJgmtL52NX+McJ+u8m4P9x…
-----END BLOQUE SELLADO_TIEMPO-----
-----END PAQUETE-----

En color verde se marca el inicio y final del paquete y el resto de colores hace referencia a un bloque del paquete.

CLAVE_CIFRADA hace referencia a la clave DES cifrada que se utilizó para cifrar el examen.
EXAMEN_CIFRADO es el examen cifrado
FECHA_HORA es la fecha y hora del sellado
HASH_ALUMNO es la firma digital del alumno
SELLADO_TIEMPO es la firma digital del sellado de tiempo

NOTA: *Se puede observar que dentro de los bloques hay una serie de caracteres sin significado ya que están en formato de array de bytes y que algunos tienen 3 puntos al final esto es para acortar espacio y hacerlo más entendible.*

4. IMPLEMENTACIÓN

Para la implementación he decidido crear 3 clases más aparte de las 2 proporcionadas por el profesor (GenerarClaves.java y Paquete.java) que son EmpaquetarExamen.java, SellarExamen.java y DesempaquetarExamen.java.

Estas clases tienen métodos que permiten al alumno crear el examen empaquetado, sellar el examen empaquetado a la autoridad de sellado y al profesor desempaquetar el examen empaquetado respectivamente.

Una vez creadas las claves privadas y públicas de los actores mediante la ejecución de GenerarClaves.java se ejecutarán los siguientes archivos java.

EmpaquetarExamen.java se ejecuta de la siguiente manera:
java -cp [...] EmpaquetarExamen <fichero_examen> <nombre_paquete> <publicKeyProfesor> <privateKeyAlumno>

fichero_examen es el fichero a cifrar, *nombre_paquete* sin extensión paquete solo el nombre y las claves correspondientes: pública del profesor y privada del alumno.

1. En este archivo leemos el examen
2. Creamos la clave y el cifrador DES
3. Ciframos el examen
4. Leemos la clave pública del profesor
5. Creamos la clave y cifrador RSA
6. Ciframos la clave DES
7. Leemos la clave privada del alumno
8. Realizamos la firma digital
9. Creamos el paquete añadiendo el examen cifrado, la clave DES cifrada y la firma del alumno

SellarExamen.java se ejecuta de la siguiente manera:
java -cp [...] SellarExamen <paquete_examen> <publicKeyAlumno> <privateKeyAutoridad>
paquete_examen es el paquete creado con extensión paquete y las claves correspondientes: pública del alumno y privada de la autoridad de sellado.

1. En este archivo leemos la clave pública del alumno
2. Leemos los datos del paquete
3. Comprobamos si la firma del alumno es válida, si lo es seguimos
    a. Si es inválida, muestra error y no hace nada más
4. Leemos la clave privada de la autoridad de sellado
5. Obtenemos la hora y fecha y realizamos el sellado de tiempo
6. Sobreescribimos el examen empaquetado añadiendo la fecha/hora y el sellado de tiempo

DesempaquetarExamen.java se ejecuta de la siguiente manera:
java -cp [...] DesempaquetarExamen <paquete_examen> <examen_claro> <privateKeyProfesor> <publicKeyAlumno> <publicKeyAutoridad>

1. En este archivo leemos todos los datos del paquete
2. Creamos cifradorRSA
3. Leemos la clave pública de la autoridad de sellado
4. Comprobamos si la firma del alumno es válida, si lo es seguimos
    a. Si es inválida, muestra error y no hace nada más
5. Mostramos la fecha del sellado de tiempo
6. Leemos la clave pública del alumno
7. Comprobamos si la firma del alumno es válida, si lo es seguimos
    a. Si es inválida, muestra error y no hace nada más
8. Leemos la clave privada del profesor
9. Desciframos clave DES
10. Creamos cifrador DES y desciframos el examen
11. Creamos un archivo que sea el examen limpio

Las listas anteriores enumera el orden de acciones que se llevan a cabo en mi implementación, algunas se hacen en el main y otras en funciones. En esta práctica hice todo en main pero luego extraje algunas funciones como recuperarClave”private/public””actor” que se encarga de leer las claves que se pasan como argumentos por ejemplo recuperarClavePrivadaProfesor(args[2] , keyFactoryRSA) sirve para obtener la clave privada del profesor.

4.1 Ejemplo de entrada/salida de mi código sin error

Primero de todo abrimos una terminal y creamos un ejemplo de examen

Segundo, compilamos todos nuestras clases
Entrada:
PS > javac -cp [...] GenerarClaves.java EmpaquetarExamen.java Paquete.java DesempaquetarExamen.java SellarExamen.java
Salida: No hay

Tercero, creamos las claves de la autoridad, profesor y alumno
Entrada: PS > java -cp [...] GenerarClaves autoridad
Salida:
Generadas claves RSA pÃºblica y privada de 512 bits en ficheros autoridad.publica y autoridad.privada

Entrada: PS > java -cp [...] GenerarClaves profesor
Salida:
Generadas claves RSA pÃºblica y privada de 512 bits en ficheros profesor.publica y profesor.privada

Entrada: PS > java -cp [...] GenerarClaves alumno
Salida:
Generadas claves RSA pÃºblica y privada de 512 bits en ficheros alumno.publica y alumno.privada

Cuarto, empaquetamos el examen
Entrada:
PS > java -cp [...] EmpaquetarExamen examen.txt examen profesor.publica alumno.privada
Salida:
Examen cargado
Clave DES generada
Examen cifrado obtenido
Clave publica del profesor obtenida
Clave DES cifrada obtenida
Clave privada del alumno obtenida
Firma digital realizada
Examen cifrado aÃ±adido al paquete
Clave cifrada aÃ±adida al paquete
Hash alumno aÃ±adido al paquete
Examen Empaquetado creado

Quinto, sellamos el examen
Entrada:
PS > java -cp [...] SellarExamen examen.paquete alumno.publica autoridad.privada
Salida:
Clave publica del alumno obtenida
Bloques necesarios del paquete cargados
Firma digital correcta!!
Clave privada de la autoridad obtenida
Fecha y hora actuales: Thu Oct 20 15:50:55 CEST 2022
Sellado de tiempo realizado
Fecha del sellado aÃ±adida al paquete

Sello aÃ±adido al paquete
Paquete sobreescrito con los 2 bloques del sellado

Finalmente, desempaquetamos el examen
Entrada:
PS > java -cp [...] DesempaquetarExamen examen.paquete examen_claro.txt profesor.privada alumno.publica autoridad.publica
Salida:
Bloques necesarios del paquete cargados
Clave publica de la autoridad obtenida
Sellado correcto!!
Fecha de sellado: Thu Oct 20 15:50:55 CEST 2022
Clave publica del alumno obtenida
Firma digital correcta!!
Clave privada del profesor obtenida
Clave DES descifrada obtenida
Examen en limpio obtenido

4.2 Ejemplo de entrada/salida de mi código con error

Aquí un ejemplo en el que la firma de la autoridad de sellado no es correcta tras la siguiente entrada:
PS > java -cp [...] DesempaquetarExamen examen.paquete examen_claro.txt profesor.privada alumno.publica autoridad_no_correcta.publica
Salida:
Bloques necesarios del paquete cargados
Clave publica de la autoridad obtenida
Error en el sellado.
ERROR: SELLADO NO VALIDO

Y otro ejemplo pero en el que la firma del alumno no es correcta tras la siguiente entrada:
PS > java -cp [...] DesempaquetarExamen examen.paquete examen_claro.txt profesor.privada alumno_no_correcta.publica autoridad.publica
Salida:
Bloques necesarios del paquete cargados
Clave publica de la autoridad obtenida
Sellado correcto!!
Fecha de sellado: Thu Oct 20 15:50:55 CEST 2022
Clave publica del alumno obtenida
Error en la firma digital.
ERROR: FIRMA ALUMNO NO VALIDA

También se podría comprobar modificando el paquete pero opté por crear otras claves para que a la hora de verificar la firma diese un error.

5. SIMPLIFICACIONES

Dado que se trata de una aplicación para aprender el uso del api de cifrado de Java (JCA) se asumirán una serie de simplificaciones:

1. 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)
2. No se contemplan los mecanismos de distribución fiable 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)
    a. El ALUMNO dispondrá de un fichero con la clave pública del PROFESOR al que enviará el Examen Empaquetado
    b. La AUTORIDAD SELLADO dispondrá del fichero con la clave pública del ALUMNO que procede a ”sellar” su Examen Empaquetado.
    c. 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.
3. Las respuestas del ALUMNO estarán almacenadas inicialmente en un fichero de texto aunque es indiferente el formato de ese fichero
4. 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.
5. 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.
6. 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.
7. 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.

6. CONCLUSIONES

Para concluir la práctica, quiero recalcar que me llamó bastante la atención como hacer los diferentes tipos de cifrados: simétricos, asimétricos y firmas digitales. Cómo leer las claves RSA, como se crea el par de claves, como generar una clave DES y usarla para cifrar y descifrar y como hacer una firma digital que tanto se usan hoy en día para firmar documentos digitales como pdfs. También me gustó la temática de los exámenes ya que una aplicación compleja llenando las simplificaciones de esta podría ser muy útil para los exámenes y/o trabajos online, ya que cada vez el teletrabajo va creciendo.

Finalmente, me sorprendió en general esta API que usamos ya que antes de esta asignatura no sabía que existía y me parece muy interesante.