Proyecto SSI

04.11.2017

Adrián Simón Reboredo

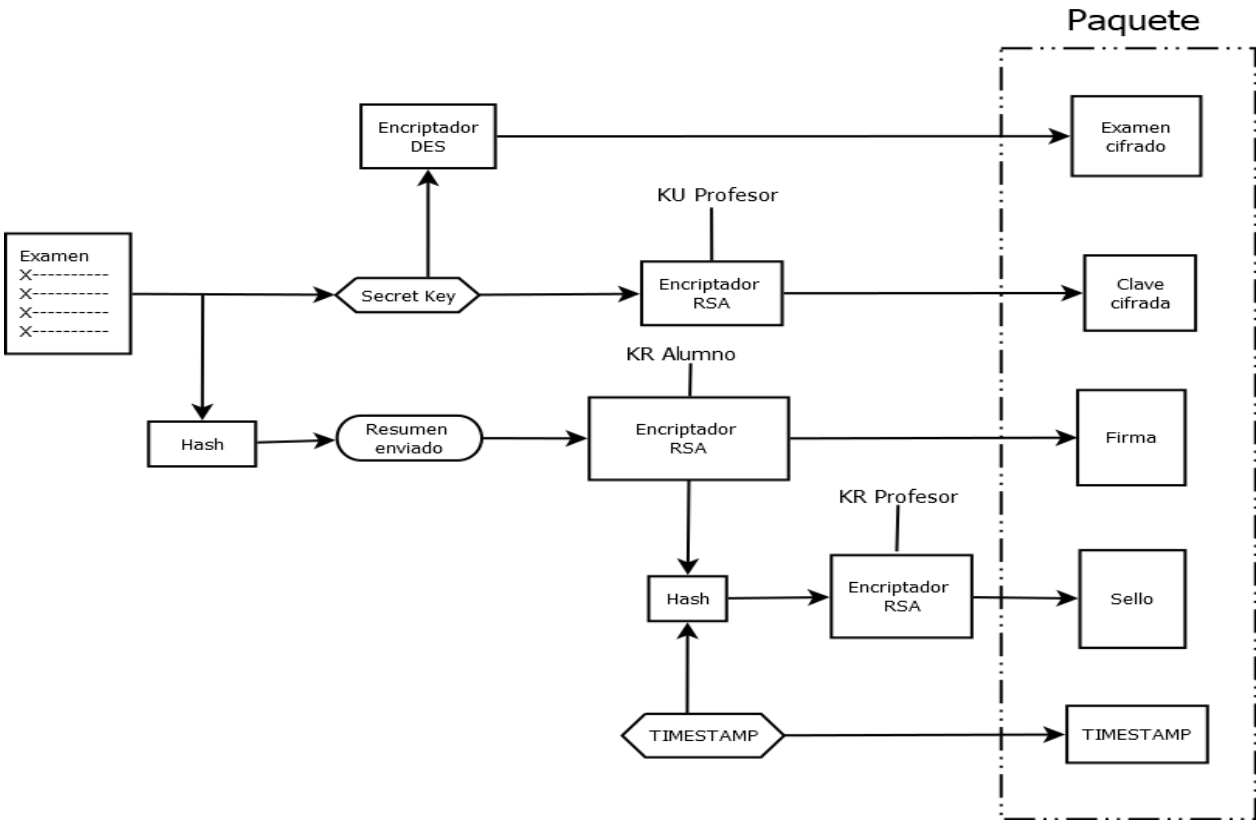
Luis Gerardo Romero Garcia

Descripción

Se pretende crear una aplicación cuya funcionalidad principal es el envío de documentos que en este caso serán exámenes enviados por el alumno y recibidos por un profesor de una forma fiable y segura. Para ello el programa dispondrá de diferentes métodos de encriptación y desencriptado para garantizar la confidencialidad, integridad, autenticación, autoridad y no repudio del documento.

Encriptación del examen

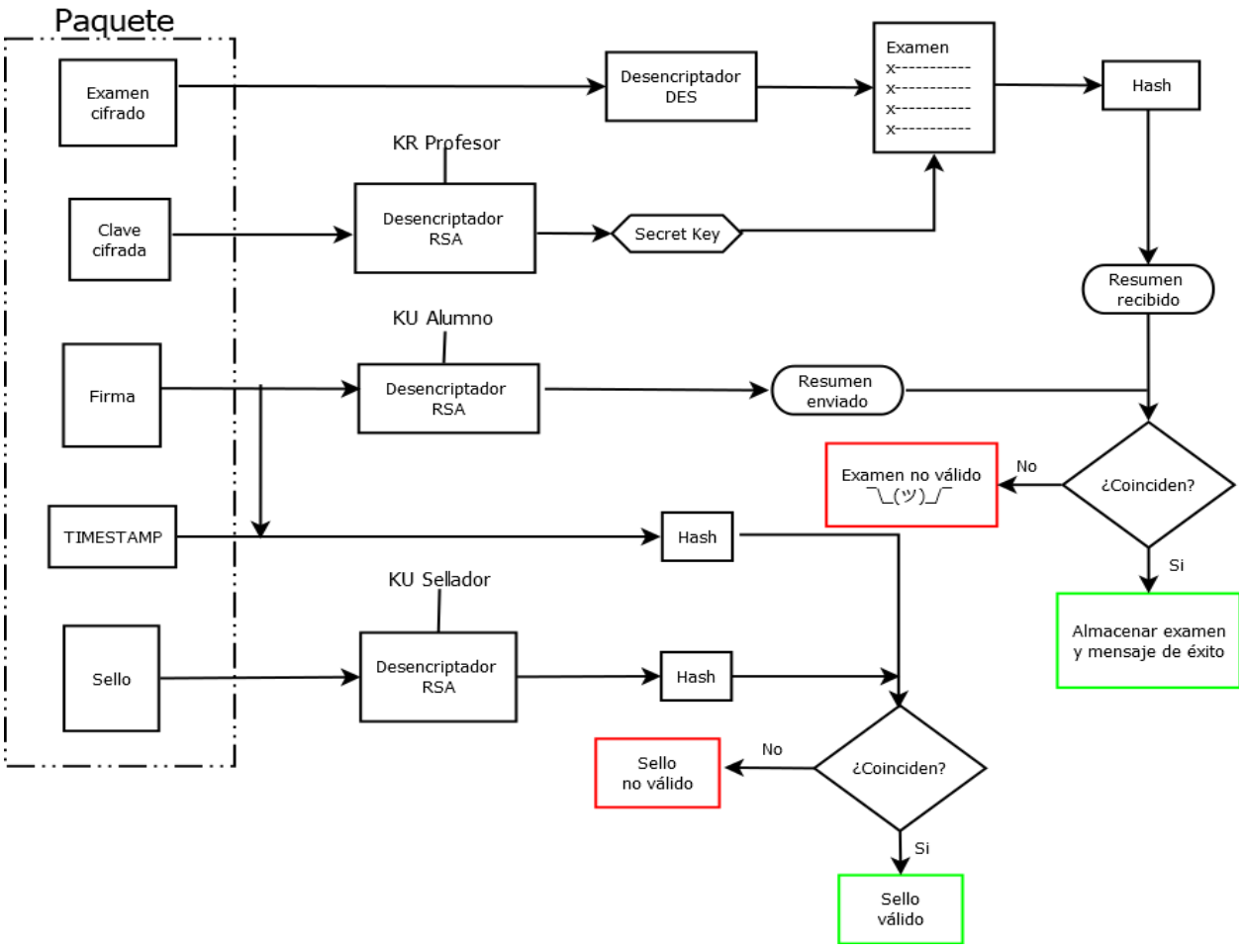
Para ello generaremos un paquete que constará del examen cifrado, una clave secreta generada a partir del cifrado DES del examen y que está estará cifrada con la clave pública del profesor mediante RSA y una firma digital que se generará a partir del resumen generado por función Hash del propio examen y que será encriptado mediante RSA con la clave privada del alumno. Una vez montado el paquete, otra entidad a mayores se encargará de introducir el TIMESTAMP y un sello en el paquete que será la combinación de la firma digital y el TIMESTAMP, todo ello se le aplicará una función Hash y se encriptará con la clave privada de la entidad selladora.



Desencriptado del examen

Al ser recibido el paquete, se desencriptarán la clave cifrada con la clave privada del profesor dándonos la clave con la que hemos encriptado el examen desde un principio y con ella podremos desencriptar el examen mediante DES. Antes de dar el examen recibido como válido tenemos que comprobar antes si el resumen del examen que habremos recibido coincide con el resumen que fue enviado por la firma digital. Para ello desencriptaremos mediante RSA la firma con la clave pública del alumno dándonos el resumen que recordemos, le hemos aplicado las funciones Hash con anterioridad antes de ser enviado y comprobaremos que coinciden para validar finalmente la integridad del examen.

Del mismo modo haremos lo mismo con el TIMESTAMP y el sello, al TIMESTAMP le aplicaremos la función Hash y al sello lo desencriptaremos mediante RSA con la clave pública de la entidad que lo ha sellado y le aplicaremos la función Hash al resultado. Si ambos resultados coinciden es que el sello se ha mantenido intacto.



Justificación de los métodos utilizados

Mediante la firma digital con función Hash garantizamos la integridad del examen ya que en caso de que el documento sea modificado dentro del paquete antes de ser recibido, cuando el profesor vaya a desencriptarlo, el resumen generado por la función hash a partir del documento y por otro lado el resumen generado a partir de la firma una vez desencriptada no coincidirán. Además garantizamos la confidencialidad porque en caso de que el paquete acabe en manos equivocadas, al no tener la clave privada del profesor lo único que obtendrá será un examen corrupto que no corresponderá con el examen original al ser desencriptado.

Además al utilizar cifrado asíncrono garantizamos tanto la autenticación del mensaje como el no repudio ya que la firma al ser desencriptada por la clave pública del alumno sabemos que por un lado solo el alumno pudo haber firmado el paquete y por otro el alumno no podría negar que él ha firmado el paquete porque no tendría sentido que al hacer el desencriptado de la firma con su clave pública nos de el resumen correcto si no ha sido encriptado desde un principio con la propia clave privada del alumno.

Hemos utilizado para implementar la firma métodos Hash ya que estos nos permiten encriptar y desencriptar con un valor de tamaño específico (resumen) en vez de trabajar directamente con el documento a encriptar ahorrándonos un alto coste computacional y permitiéndonos poder trabajar con documentos muy grandes si el sistema lo requiriese.

Por último mediante el sello y el TIMESTAMP podemos comprobar si el TIMESTAMP ha sido cambiado o no al comprobar que si el resultado de aplicar, desencriptando el sello previamente, la función hash en ambos elementos coinciden. Mediante esta técnica sabremos si el paquete ha sido enviado a tiempo.

Clases y métodos utilizados

EmpaquetarExamen.java: Clase encargada de generar el paquete. Encripta el examen con un cifrador DES cuya clave es generada aleatoriamente. Acto seguido es cifrada la clave DES con RSA mediante la clave pública del profesor. Después genera un resumen Hash del examen cifrado y lo cifra con la clave privada del alumno, generando la firma del paquete.

SellarPaquete.java: Clase encargada de crear un sello de autenticación para el paquete. En primer lugar añade al paquete una marca de tiempo, conocido como TIMESTAMP. Acto seguido realiza un resumen Hash de concatenación de la firma cifrada y el TIMESTAMP con su clave privada RSA .

DesempaquetarExamen.java: Clase encargada descifrar el paquete y comprobar su integridad, no repudio y autenticidad. Esta clase consta con dos métodos

- **comprobarIntegridad:** Obtiene la clave DES descifrándola mediante RSA con la clave privada del profesor para a continuación descifrar el contenido del examen mediante DES. A continuación realiza un resumen hash a partir del examen descifrado y lo compara con la firma obtenida al descifrar con RSA la firma cifrada del paquete utilizando la clave pública del alumno.
- **comprobarSello:** Descifra el sello usando la clave pública de la autoridad de sellado y lo compara con un resumen hash de la combinación recibida de la firma y el TIMESTAMP.

Instrucciones de compilación y ejemplo de usos

Prerrequisitos:

- Disponer de la implementación BouncyCastle en classpath.
- Haber generado claves públicas y privadas para cada uno de los actores del sistema (alumno, profesor y autoridad de sellado).

Compilación de EmpaquetarExamen.java:

- java EmpaquetarExamen <nombreExamen> <nombrePaquete> <KUProfesor> <KRAlumno>
- Genera un archivo <nombrePaquete> que contiene la representación DAO del paquete.

Compilación de SellarExamen.java:

- java SellarExamen <nombrePaquete> <KRSellado>
- Modifica el archivo <nombrePaquete> incluyendo el TIMESTAMP y el sello

Compilación de DesempaquetarExamen.java:

- Java DesempaquetarExamen <nombrePaquete> <nombreExamen> <KUSellado> <KUAlumno> <KRProfesor>
- Muestra por pantalla los resultados del desempaquetado
- En caso de que el examen no esté corrupto, genera el archivo “examenDescifrado”

Conclusiones

Tras analizar la arquitectura de la aplicación, nos ha costado encontrar puntos débiles en general debido a que las carencias o defectos que puedan tener las técnicas de cifrado, es contrarrestado por el uso de otras técnicas de cifrado que se han logrado integrar correctamente en el programa dando como resultado una aplicación que, a pesar de no disponer de otras medidas que no entran en el alcance del proyecto, cumplen de forma satisfactoria con los requisitos del mismo.