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

Luis Miguel Raña Cortizo

Descripción breve de la práctica

Consiste en el desarrollo de un conjunto de herramientas para dar soporte a la validación de las *"Compostelas"* de los peregrinos que hacen el Camino de Santiago.

Los peregrinos podrán generar su *Compostela Virtual* que será sellada por los albergues por donde vayan pasando. Finalmente, la *Oficina del Peregrino* podrá validar dicha *Compostela Virtual* para extraer los datos del peregrino y validar la autenticidad de los "sellos" emitidos por los albergues.

Estrategias criptográficas empleadas para asegurar los requisitos básicos exigidos y pasos del empaquetado, sellado y desempaquetado

Requisitos a cumplir:

(1) Asegurar la confidencialidad del contenido incluído por el peregrino y los albergues.

(2) La *Oficina del peregrino* debe saber que el peregrino que presenta la *Compostela Virtual* es quien dice ser.

(3) El contenido de la *Compostela Virtual* no haya sido modificado.

(4) Ni el peregrino ni los albergues podrán repudiar el contenido incluído por ellos en la *Compostela Virtual*.

(5) La *Oficina del peregrino* no podrá realizar cambios en el contenido de la Compostela Virtual.

(6) Los albergues tienen que poder garantizar la fecha y el lugar en que fue "sellada" la *Compostela Virtual*.

(7) Asegurar un coste computacional reducido en la creación, sellado y validación de la *Compostela Virtual* minimizando el uso de criptografía asimétrica.

Para cumplir estos requisitos se ha usado la siguiente estrategia para los datos del peregrino:

- Los datos del peregrino se encriptan con cifrado simétrico usando una clave aleatoria y luego se desencriptarán con esta misma clave(Requisitos 1, 3, 7).
- La clave usada para encriptar los datos se encriptará con cifrado asimétrico usando una clave pública de la *Oficina del peregrino,* y luego se usará la clave privada de la *Oficina del peregrino* para desencriptarlos(Requisitos 4, 7).
- Se incluirá tambien una firma digital del peregrino usando su clave privada, y luego se usará su clave pública para verificarla(Requisitos 2, 3, 4, 5, 7).

Para los albergues se sigue una estrategia similar:

- Los datos del albergue se encriptan con cifrado simétrico usando una clave aleatoria y luego se desencriptarán con esta misma clave(Requisitos 1, 3, 7).
- La clave usada para encriptar los datos se encriptará con cifrado asimétrico usando una clave pública de la *Oficina del peregrino,* y luego se usará la clave privada de la *Oficina del peregrino* para desencriptarlos(Requisitos 4, 7).
- Se incluirá tambien una firma digital de los datos del albergue(incluyendo fecha y lugar) concatenados con la firma del peregrino usando la clave privada de dicho albergue, y luego se usará su clave pública para verificarla(Requisitos 3, 4, 5, 6, 7).

De esta manera se obtienen tres ejecutables(*Imágenes 1,2,3*):

Generar Compostela

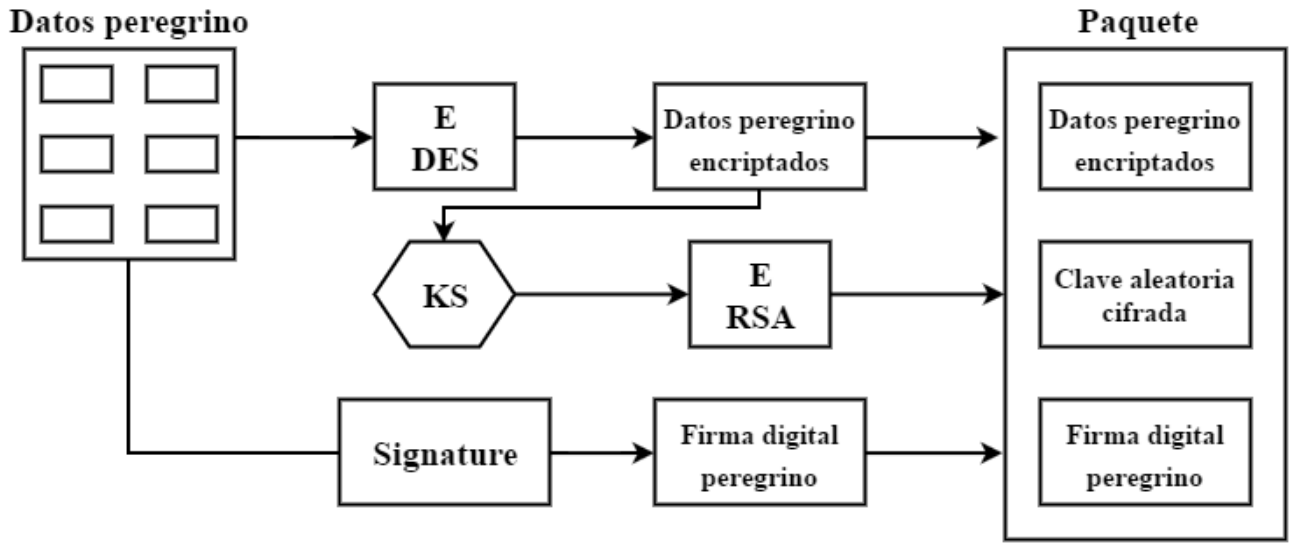

Imagen 1

Sellar Compostela

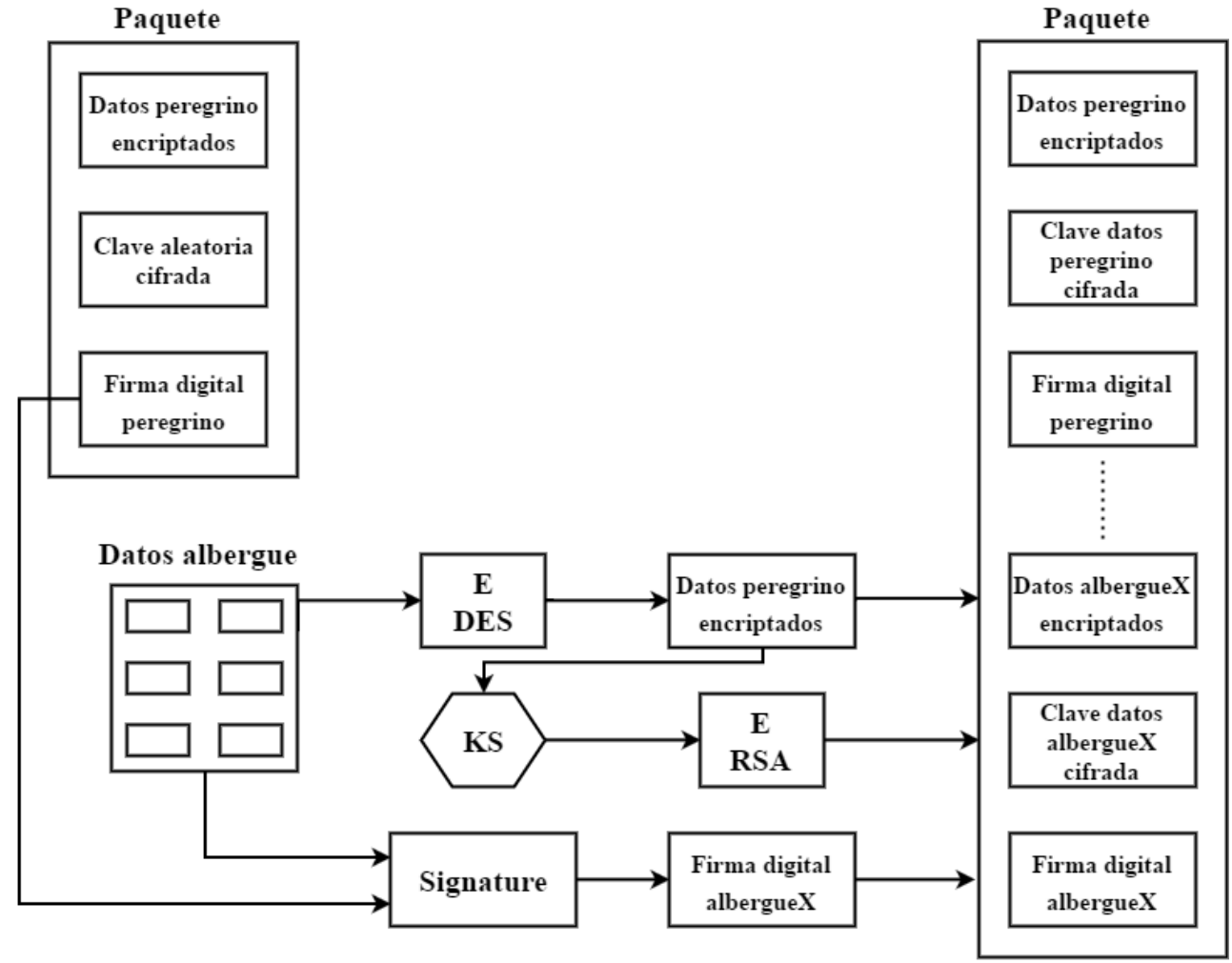

Imagen 2

Desempaquetar Compostela

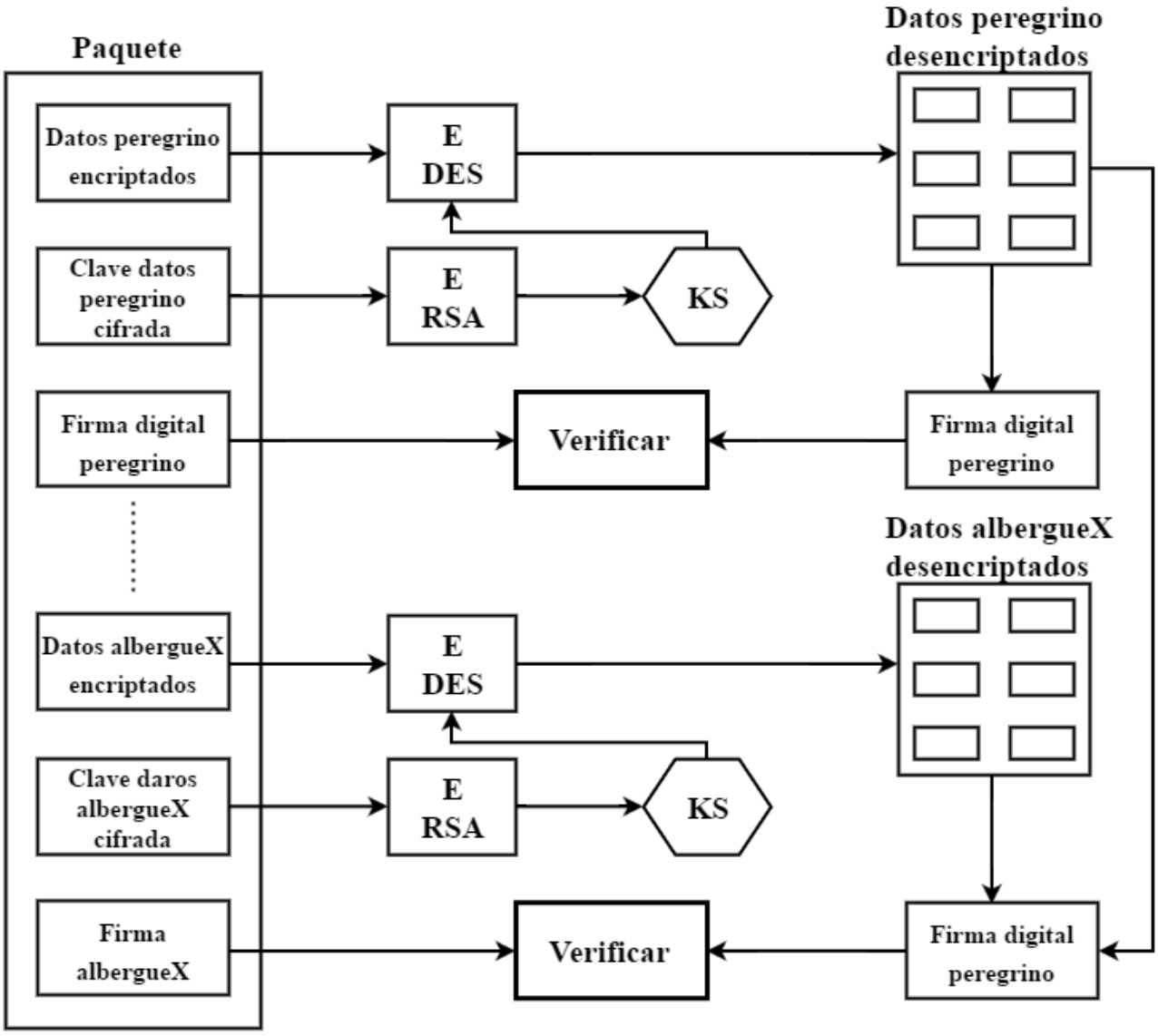

Imagen 3

Formato/estructura del "paquete" resultante

El paquete resultante que compone la *Compostela virtual* está formado por 3 bloques del peregrino y 3 bloques por cada uno de los albergues. Estos bloques son un bloque con los datos encriptados con cifrado simetrico, otro bloque con la clave utilizada para el encriptado de los datos encriptada con cifrado asimetrico y por útimo un bloque con una firma digital. Podemos verlo representado en el diagrama anterior(*Imagen 3*). No se puede asegurar que el contenido esté ordenado al tratarse de un Map(clave, valor), por tanto se usa una nomenclatura homogénea para los identificadores.

Implementación: clases y métodos más importantes

Ya se han nombrado los tres ejecutables implementados para generar, sellar y desempaquetar la *Compostela Virtual*:

- GenerarCompostela <nombre paquete> <ficheros con las claves necesarias>
- SellarCompostela <fichero paquete> <identificador albergue> <ficheros con las claves necesarias>
- DesempaquetarCompostela <fichero paquete> <num. albergues (N)> <identificador albergue 1> <ficheros claves albergue 1> ... <identificador albergue N> <ficheros claves albergue N> <ficheros con otras claves necesarias>

También se cuenta con otro ejecutable para generar las claves(pública y privada) RSA de 512 bits necesarias:

- GenerarClaves <identificador>

Además de los ejecutables, se cuenta con una serie de clases a mayores:

- 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* que usa la codificación Base64 para representar datos binarios mediante caracteres imprimibles(leerPaquete(...) y escribirPaquete(...)).
- JSONUtils.java: métodos estáticos con utilidades para el formateo y parseo del formato JSON simplificado usado como soporte para el almacenamiento de los datos aportados por peregrinos y albergues(json2map(...) y map2json(...)).

También se incluye la librería bcprov-jdk15on-155.jar necesaria para llevar a cabo la práctica.

Instrucciones de compilación y ejemplos de uso

1. Para simplificar su uso van creados por defecto un par de claves PEREGRINO.publica y PEREGRINO.privada, OFICINA.publica y OFICINA.privada, ALBERGUE1.publica y ALBERGUE1.privada, ALBERGUE2.publica y ALBERGUE2.privada. Si se quieren generar mas pares de claves se tiene que ejecutar GenerarClaves(identificador) pasandole un identificador como parámetro.
2. Para generar la compostela se debe ejecutar GenerarCompostela(nombre, clave publica oficina, clave privada peregrino), por parametro se le pasa el nombre del paquete(por ejemplo Compostela.paquete), la clave publica de oficina(por defecto va creada OFICINA.publica) y la clave privada del peregrino(por defecto va creada PEREGRINO.privada) en ese orden.
3. A continuacion se debe sellar la compostela, se debe ejecutar SellarCompostela(paquete, identificador albergue, clave publica de oficina, clave privada albergue), por parametro se le pasa el nombre del paquete creado al generar la compostela, el identificador del albergue, la clave publica de oficina(por defecto va creada OFICINA.publica) y la clave privada del albergue(por defecto va creada ALBERGUE1.privada) en ese orden. Esto se debe ejecutar tantas veces como albergues queramos introducir, por defecto tambien van creadas las claves para ALBERGUE2. Si se desean mas albergues hay q genear mas claves.
4. Por último, para desempaquetar se ejecuta DesempaquetarCompostela(paquete,

numAlbergues, identificador albergue 1, clave publica albergue 1, ..., identificador albergue N, clave publica albergue N, clave publica peregrino, clave privada oficina), por parametros se pasan el nombre del pquete creado al generar la compostela, el numero de albergues que han realizado el sellado, los identificadores y claves publicas de los albergues(por defecto va creada ALBERGUE1.publica y ALBERGUE2.publica) que han realizado el sellado, la clave publica del peregrino(por defecto va creada PEREGRINO.publica) y la clave privada de oficina(por defecto va creada OFICINA.privada).

Es importante que los nombres de los identificadores que se usan para sellar la compostela coincidan exactamente con los q se usan para desempaquetar, así como las claves que se usan para encriptar se correspondan con las que se usan para desencriptar, sino el programa no funcionará correctamente. Tambien es importante que el orden de los parámetros sea exactamente el indicado en cada ejecutable.

Tanto al generar la compostela como al sellarla se pedirán por pantalla unos datos que el usuario debe ir introduciendo, al desempaquetar se hará un comprobación para ver si los datos han sido manipulados(firma digital), si no han sido manipulados los mostrará por pantalla, sino mostrará un mensaje alertando de su manipulación.

Simplificaciones realizadas y sus consecuencias en una aplicación real

Se asumen una serie de simplificaciones:

- Cada uno de los participantes (PEREGRINO, ALBERGUE, OFICINA DEL PEREGRINO) 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 (PEREGRINO, ALBERGUE u OFICINA DEL PEREGRINO) de forma confiable (en una aplicación real se haría uso de certificados digitales y de una *autoridad de certificación* común).
    - EL PEREGRINO y dispondrá de un fichero con la clave pública de la OFICINA DEL PEREGRINO.
    - Cada ALBERGUE dispondrá de un fichero con la clave pública de la OFICINA DEL PEREGRINO y también tandrá acceso al fichero con la clave pública del PEREGRINO que desee "sellar" su *Compostela Virtual*.
    - La OFICINA DEL PEREGRINO contará con la clave pública (almacenada en su respectivo fichero) de cada uno de los ALBERGUES, así como con la clave pública del PEREGRINO para el cual se vaya a realizar la validación de su *Compostela Virtual*.
- Las distintas piezas de información que aporte cada participante (PEREGRINO o ALBERGUE) tendrán (antes del cifrado/firma y después del descifrado) la forma de *Strings* en formato JSON con codificación UTF8.
- La *Compostela Virtual* se materializará en un "paquete" que contendrá toda la información que vayan incorporando los distintos participantes implicados: el PEREGRINO que la generó y los ALBERGUESque la hayan "sellado".
- No se contempla un almacenamiento "físico" realista de la *Compostela Virtual*, sólo se trata de implementar los programas para generar, sellar y validar la *Compostela Virtual* conforme a las especificaciones descritas en este documento.
- Al validar la *Compostela Virtual*, si todas las comprobaciones son correctas, se mostrará a la OFICINA DEL PEREGRINO los datos aportados por el PEREGRINO y los datos

incorporados en cada uno de los "sellos" estampados por cada ALBERGUE que haya procesado dicha *Compostela Virtual*. En caso contrario se indicarán las comprobaciones que no hayan sido satisfactorias.

- No se contemplan comprobaciones adicionales más allá de las anteriores, como puede ser verificar el orden correcto de los "sellados" o la imposibilidad física/temporal de los "sellados" realizados.
- Se asume que este tipo de comprobaciones "lógicas" las hará el personal de la OFICINA DEL PEREGRINO

Resultados obtenidos y conclusiones

El resultado es un programa que simula el funcionamiento descrito bajo las simplificaciones comentadas y los requisitos impuestos. De esta manera se obtiene un programa compuesto por cuatro ejecutables donde cada uno cumple una función o paso del sistema implementado.

En esta práctica se llevan a cabo tareas en java tales como leer y escribir ficheros, conversiones de datos(String, byte, Map), cifrado y descrifrado usando los algoritmos DES y RSA, generación de par de claves RSA, firmas digitales(Signature), manejo de clases y funciones, entre otras.

Es una práctica que sirve para familiarizarse con el manejo de datos encriptados, ser capaz de volcar unos datos encriptados en un fichero para posteriormente extraer y obtener los datos iniciales.