Prácticas de Seguridad Informática

—

Práctica 5

—

Grado Ingeniería Informática

Curso 2014-2015

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Departamento de Informática e
Ingeniería de Sistemas
Area de Lenguajes y Sistemas Informáticos

3 de diciembre de 2014

Introducción

Vamos a realizar algunas pruebas con la biblioteca criptográfica de Java. En particular, la idea es probar diferentes componentes y realizar medidas para adquirir conocimiento no sólo sobre como usar estas bibliotecas, sino también acerca de los costes que implica el uso de determinadas tecnologías.

Utilizaremos la *Java Cryptography Extension* (JCE) que forma parte de la JDK a partir de la versión 1.4.

Como se comentó en clase el sistema es ampliable con bibliotecas de terceros, que se denominan proveedores (*providers*).

Algunas consideraciones previas (recordatorio)

Resumen de un mensaje (*hash*)

Sirve para asegurar la integridad. A partir de un mensaje se genera un bloque de bits que representan una 'firma' (o resumen, no confundir con firma digital). Un cambio (incluso pequeño) en el texto original, producirá una firma diferente.

Se utilizan funciones de una sola dirección: es fácil generar la firma, pero casi imposible (o, al menos, muy difícil; o al menos, muy costoso) generar el mensaje que corresponde a una firma dada.

La autentificación de un mensaje se consigue mediante el uso de resúmenes criptográficos, que son más difíciles de falsificar.

Criptografía de clave secreta

El objetivo es proporcionar confidencialidad. Queremos impedir que lectores no deseados puedan examinar el texto.

El cifrado se puede hacer en bloques, o bit a bit. Si se hace en bloques, y no tenemos una cantidad de bits que sea múltiplo del tamaño del bit, hace falta completar, hasta alcanzar el tamaño adecuado (*padding*). Los cifrados bit a bit se suelen denominar cifrados de flujo.

Los modos de cifrado sirven para decidir la forma en que se realizarán las operaciones (básicamente, si se cifran bloques de modo independiente, o si hay dependencias entre el cifrado de los distintos bloques).

Criptografía de clave pública

En la criptografía de clave pública, cada usuario tiene sólo dos claves, que le sirven para comunicarse con cualquier otro. Una de ellas es privada, y debe permanecer en secreto. La otra es pública y debe estar a disposición del que la requiera. Permite muchos esquemas de comunicación, basados en el uso de la clave privada, la pública, o combinaciones de ambas. Es lenta, comparada con la criptografía de clave privada. Esa es una de las características que queremos medir en esta práctica. Habitualmente se utiliza

una combinación de ambas: criptografía de clave pública para 'negociar' una clave privada con la que comunicarse.

Firma digital

Para garantizar la autenticidad de un mensaje se puede utilizar el siguiente procedimiento:

- Se hace un resumen del mensaje
- Se firma con la clave pública del receptor
- Se envía el mensaje, junto con el resumen firmado

El receptor:

- Descifra el resumen con su clave privada
- Obtiene el resumen del mensaje
- Compara

¿Qué hay que hacer?

Vamos a programar una aplicación que nos permita experimentar con todas estas funciones, y además medir el tiempo que emplea cada una en realizar su función.

Deberemos elegir en cada caso alguno de los algoritmos disponibles y justificar (**muy** brevemente) la elección como comentario en el propio código al principio, junto con la información sobre el grupo que ha realizado la práctica.

[Ejemplo]

- Vamos a utilizar para las pruebas DES/ECB/PCKS5Padding (algoritmo DES, en modo *Electronic Code Book*, con relleno PCKS5).
  El tamaño de la clave es ....
  (Otros parámetros relevantes).
  Los motivos de esta decisión son:
  ...
- Vamos a utilizar para las pruebas RSA/ECB/PCKS1Padding (algoritmo RSA, en modo *Electronic Code Book*, con relleno PCKS1).
  El tamaño de la clave es...
  (Otros parámetros relevantes).
  Los motivos de esta decisión son... ...

El programa deberá incluir:

- Obtener un *hash*;
- Generación de claves de cifrado, tanto de clave secreta, como de clave pública. Almacenamiento adecuado de las mismas para su utilización posterior.
- Cifrar y descifrar, tanto con clave secreta como con clave pública, utilizando las claves generadas con nuestro programa. En el caso de criptografía de clave pública determinar el tamaño máximo del mensaje que se puede cifrar.
- Firma digital[1].

Para todos los casos, se deberá poder medir los tiempos:

- Tiempo de generación de claves (para tres tamaños diferentes de la clave; por ejemplo, el que viene en los programas proporcionados como ejemplo, el doble y el triple)[2].
- Tiempo de firmado (hash y firma digital).
- Tiempo de cifrado y descifrado:
  - Para un conjunto de documentos de texto con criptografía de clave secreta[3].
  - Para una lista de cien mensajes aleatorios del tamaño máximo determinado anteriormente. Comparar los tiempos para la criptografía de clave secreta y la de clave pública.

Para esto será útil tener en cuenta el método `getTime` de la clase `Date`. Este método devuelve el número de milisegundos desde el 1 de enero de 1970. Llamándolo antes y después de hacer una operación, podremos medir fácilmente el tiempo transcurrido.

Bolas extra

- Comprobar que somos capaces de intercambiar información con otros grupos acordando una clave y viendo que somos capaces de descifrar los documentos (ficheros de texto) cifrados por ellos y también enviarles mensajes cifrados.
- Cifrar un documento/mensaje con nuestro programa para luego descifratlo con alguno de los disponibles que implementen los mismos sistemas de cifrado[4]. Descifrar un documento/mensaje cifrado con alguno de estos programas mediante el programa que hemos realizado.
- Añadir algún proveedor externo[5].

[1] Podemos hacerlo 'a mano' (programando los pasos nosotros, con lo que ya sabemos de criptografía de clave pública), y utilizando la biblioteca de java nos proporciona herramientas adecuadas, mediante la clase `Signature`.

[2] En algunos casos el tiempo puede ser pequeño. Una manera más fiable de medir sería ejecutar un cierto número de veces el proceso de generación de la clave y calcular la media.

[3] Por ejemplo, 25. Ideas: páginas de la Wikipedia -en formato texto-; páginas del manual de Unix; ...

[4] Por ejemplo, `openssl`. Este trabajo puede implicar estudiar los formatos de almacenamiento que se utilizan, tanto para lo que se quiere cifrar como para las claves.

[5] Por ejemplo BouncyCastle, `https://www.bouncycastle.org/`. Comparar las medidas de tiempo para ambos proveedores.

Documentación adicional

- Java SE Documentantion,

  `http://docs.oracle.com/javase/7/docs/technotes/guides/security/`

  Para personas con prisa: 'Java(TM) Cryptography Architecture Standard Algorithm Name Documentation'

  `http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html`

- Será de utilidad este documento, donde hay ejemplos de uso de todas las clases necesarias.

  `http://www.ibm.com/developerworks/java/tutorials/j-sec1/`

- El código de ese documento está en

  `http://www.ibm.com/developerworks/java/tutorials/j-sec1/JavaSecurity1-source.jar`