Práctica 3. Cifrado con clave pública y firma con la JCA de Java

Práctica 3 de BySS

Proyecto BySS

Documentación de la Práctica 3:

Título: BySSLab
Autor: Lorenzo M. Martínez Bravo
Empresa: DISIT de la UEx
Descripción: Prácticas de BySS

Realización:

- Individual o grupos de 2 estudiantes.

Descripción:

Propósito:

- Se trata de proteger ficheros utilizando algoritmos de clave pública. Se permiten las operaciones:
    - Generación y almacenamiento en un fichero de un par de claves privada/pública.
    - Firma de ficheros con la clave privada. Se firma el contenido de un fichero con la clave privada. Se genera un fichero de salida en el que se inserta una cabecera, que contiene la identificación del algoritmo de firma empleado, y la firma calculada, junto con el contenido original del fichero.
    - Verificación de la firma de un fichero. Esta operación, complementaria a la anterior, verifica que la firma almacenada en un fichero coincide con la que se calcula. Además, recupera el contenido del fichero original.
    - Cifrado de un fichero con la clave pública.
    - Descifrado de un fichero con la clave privada. Esta operación complementa a la anterior.

Implementación:

- J2SDK v1.4.x.

Clases a utilizar (de la librería J2SDK):

KeyPairGenerator: Representa un generador de pares de claves. Se usa de la siguiente forma:

1. Se obtiene una instancia: KeyPairGenerator kpg = KeyPairGenerator.getInstance("algoritmo");
2. Se inicia el generador: kpg.initialize(tamaño); // 512 bits

KeyPair: Sirve para representar un par de claves (privada/pública). Operaciones que necesitamos utilizar:

- Generación de un par de claves: Se realiza utilizando la clase anterior:

  keyPair = kpg.generateKeyPair();

- Almacenamiento del par de claves: No hay nada específico. Se puede utilizar alguna de las posibilidades siguientes:
  - Serialización hacia un fichero, sin cifrar y sin proteger la clave privada.
  - Serialización hacia un fichero, cifrando con un algoritmo de cifrado de claves.
  - Conversión a su forma codificada y cifrado posterior.
- Acceso a los componentes:

  PublicKey pku = keyPair.getPublic();

  PrivateKey pkr= keyPair.getPrivate();

Signature: Clase que sirve para construir y verificar firmas digitales. Veamos como se utiliza para crear y verificar firmas:

- Creación de una firma:

1. Obtener instancia del objeto: Signature dsa = Signature.getInstance("algoritmoFirma");
2. Iniciar para crear firma con Clave Privada: dsa.initSign(ClavePrivada);
3. Procesar información a firmar: dsa.update(byte[]);
4. Obtener la firma: byte[] sig = dsa.sign();

- Verificación de una firma:

1. Obtener instancia del objeto: Signature dsa = Signature.getInstance("algoritmoFirma");
2. Iniciar para verificar firma con Clave Pública: dsa.initVerify(ClavePublica);
3. Procesar información a firmar: dsa.update(byte[]);
4. Verificar la firma: boolean verifies = dsa.verify(sig);

Cipher: Se trata de la clase utilizada para cifrar/descifrar (ya usada con algoritmos simétricos). Aquí la usaremos para cifrar con la Clave Pública y para descifrar con la Clave Privada correspondiente. A diferencia de lo que ya hemos usamos anteriormente, ahora no podremos usar las clases CipherInputStream/CipherOutputStream, pues no se implementa su funcionalidad con algoritmos de clave pública. Por tanto, tendremos que usar la clase Cipher, y solventar algunos problemas.

- Cifrado:

1. Obtener instancia: Cipher c = Cipher.getInstance("Algoritmo");
2. Iniciar para cifrado con la Clave Pública: c.init(c.ENCRYPT_MODE,ClavePublica);
3. Cifrar cada bloque de información por separado. Los bloques a cifrar están limitados en tamaño. El algoritmo se encarga de rellenar cada bloque si es necesario:

```
int blockSize = 53; c.getBlockSize() 11;
byte out[] = c.doFinal(byte[blockSize]);
```

- Descifrado:

1. Obtener instancia: Cipher c = Cipher.getInstance("Algoritmo");
2. Iniciar para cifrado con la Clave Pública: c.init(c.DECRYPT_MODE,ClavePrivada);
3. Descifrar cada bloque de información por separado. El algoritmo se encarga de eliminar el relleno de cada bloque si es necesario:

```
int blockSize = 64; //c.getBlockSize();
byte out[] = c.doFinal(byte[blockSize]);
```

Algoritmos disponibles:

- Generación de claves: "RSA".
- Cifrado: "RSA/ECB/PKCS1Padding".
- Firma: "SHA1withRSA", "MD2withRSA", "MD5withRSA".

Referencias:

- Prototipo de la práctica.
- Clases para gestionar la cabecera (Header.java, Options.java).
- Documentación de la JCA de Java.
- Nombres de algoritmos.