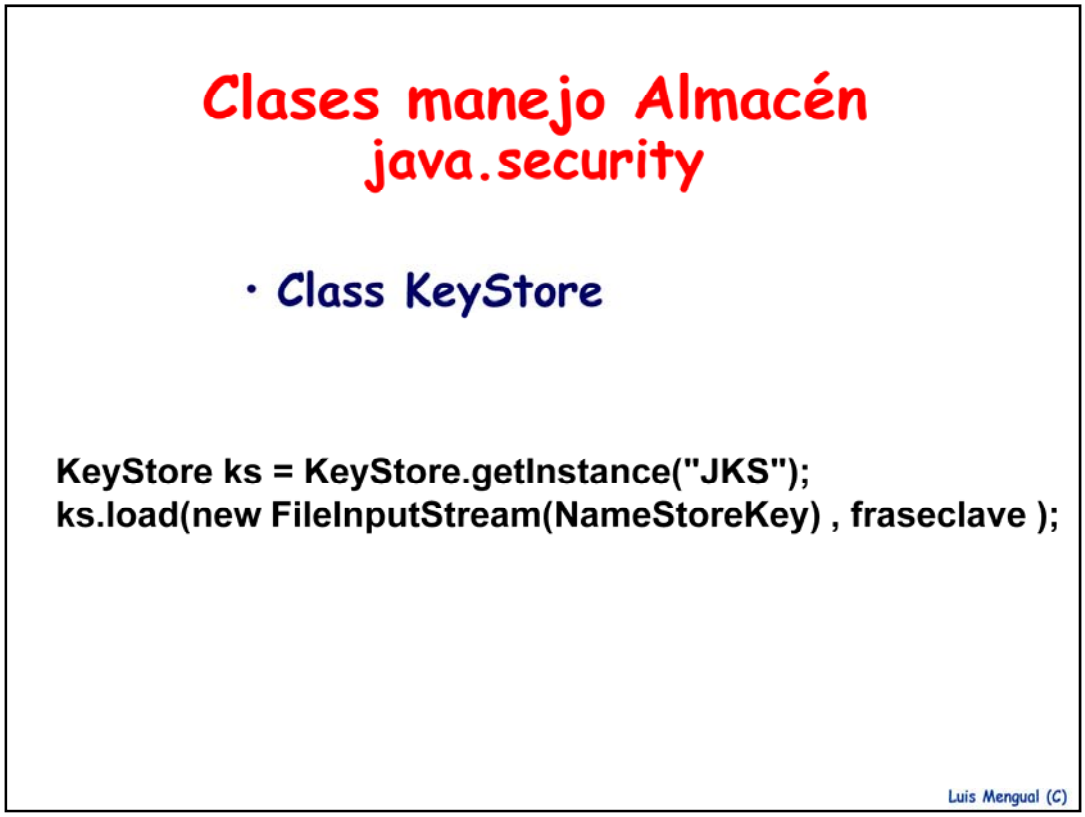
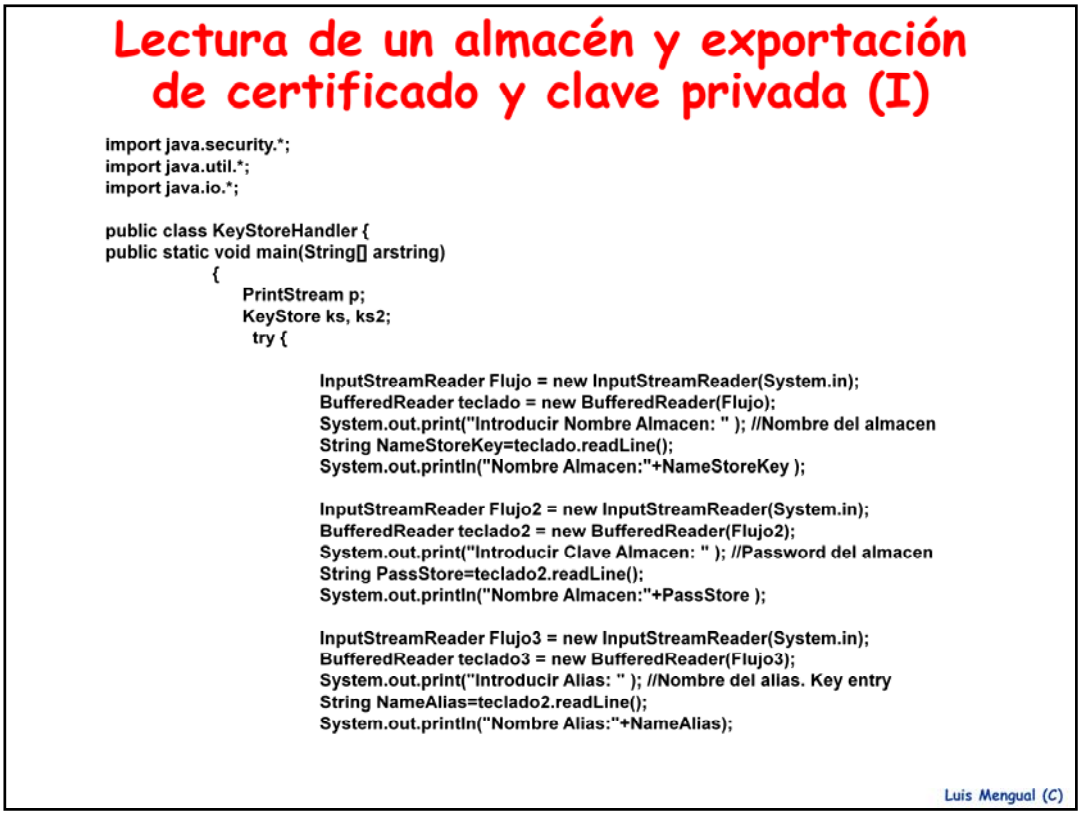


La clase "KeyStore" es fundamental para incorporar mecanismos de seguridad a nuestras aplicaciones. La clase "KeyStore" maneja los almacenes de certificados y claves.

En la figura se muestra como cargar un almacén de claves.

A partir de la carga del almacén podemos extraer los certificados y la claves privadas (si existieran) del mismo con los métodos "getKey" y "getCertificate".

Con los objetos de las clases "Certificate" y "Key" podemos cargar los certificados y las claves privadas. Con los métodos "getEncoded" de estas clases codificamos en formato binario el certificado y la clave privada y los dejamos listos para ser impresos a fichero.



El ejercicio que se propone consiste en el acceso desde código java de un almacén de claves conteniendo un certificado autofirmado e imprimir el certificado de clave pública y la clave privada en formato *.cer. La herramienta Keystore no permite exportar la clave privada directamente por lo que hay que utilizar código Java.

Previamente hemos creado un almacén con la herramienta Keytool con un certificado autofirmado

keytool -genkey -alias CertificadoCL -keyalg RSA -validity "100" -keystore AlmacenCL -keypass oooooo -storepass oooooo

keytool -list -v -keystore AlmacenCL

Mediante el código java accedemos al almacén, extraemos el certificado autofirmado y la clave privada y los imprimimos en formato *.DER

Lectura de un almacén y exportación de certificado y clave privada (II)

```
char[] fraseclave = PassStore.toCharArray();
ks = KeyStore.getInstance("JKS");
ks.load(new FileInputStream(NameStoreKey) , fraseclave ); //cargamos el almacen que contiene el certificado

// OBTENEMOS LA CLAVE PRIVADA
Key key = ks.getKey(NameAlias, fraseclave);

if (key instanceof PrivateKey)
{
        java.security.cert.Certificate cert = ks.getCertificate(NameAlias);    // Get certificate //
        System.out.println(cert);
        byte[] buf1 = cert.getEncoded();
        FileOutputStream os1 = new FileOutputStream("certificado.der");
        os1.write(buf1);

        PublicKey publicKey = cert.getPublicKey(); // OBTENEMOS LA CLAVE PÚBLICA
        System.out.println(publicKey);

        System.out.println(key); // CREAMOS EL FICHERO CON LA CLAVE PRIVADA
        byte[] buf2 = key.getEncoded();
        FileOutputStream os2 = new FileOutputStream("private_key.der");
        os2.write(buf2);
}

    } //try
        catch (Exception exception) {exception.printStackTrace();          }
  }//main
}//public
```

Luis Mengual (C)

Para poder leer el almacén y extraer la clave privada y el certificado debemos en primer lugar crear una instancia de la clase "KeyStore" y cargar el fichero almacén en el objeto "keyStore".

ks = KeyStore.getInstance("JKS");
ks.load(new FileInputStream(NameStoreKey) , fraseclave);

Con el método "getCertificate" y tomando como parámetro el alias de la entrada podemos extraer el certificado de clave pública y llevarlo a un objeto de la clase "Certificate".

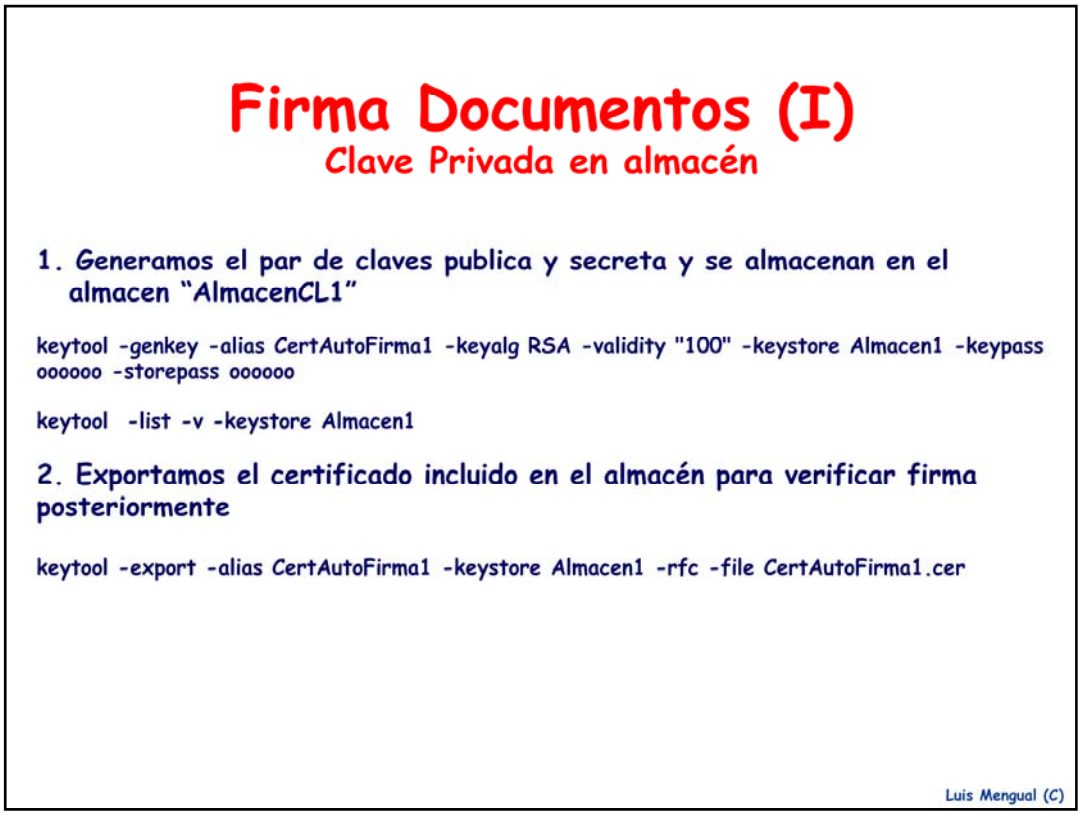
java.security.cert.Certificate cert = ks.getCertificate(NameAlias);
byte[] buf1 = cert.getEncoded();
FileOutputStream os1 = new FileOutputStream("certificado.der");
os1.write(buf1);

A partir del método "getKey" tomando como parámetros el alias donde esta el certificado y la clave privada nos creamos un objeto de la clase "Key". En este objeto almacenamos la clave privada

byte[] buf2 = key.getEncoded();
FileOutputStream os2 = new FileOutputStream("private_key.der");
os2.write(buf2);

PARA CONVERTIR LA CLAVE PRIVADA A FORMATO pem
openssl> pkcs8 -nocrypt -inform DER -in private_key.der -outform PEM -out private_key.pem

PARA CONVERTIR EL CERTIFICADO A FORMATO pem
openssl> x509 –inform DER –in usrCertificado.crt/der –outform PEM –out usrCertificado.pem
openssl rsa -in private_key.pem -pubout -out pubkey.pem



Vamos a estudiar a continuación ejemplos de cómo poder realizar una firma digital utilizando la extensión de java JCE (Java Criptography Extension).

Para poder realizar la firma de un documento de texto necesitamos la clave privada de usuario.

Sabemos que la firma digital es una estructura que consta de dos elementos: Un elemento es el texto a firmar. El otro elemento es un bloque criptográfico formado por el cifrado con la clave privada del usuario del hash (resumen) del texto en claro a firmar:

Texto_claro, cifar(ClavePrivada, Hash(Texto_claro))

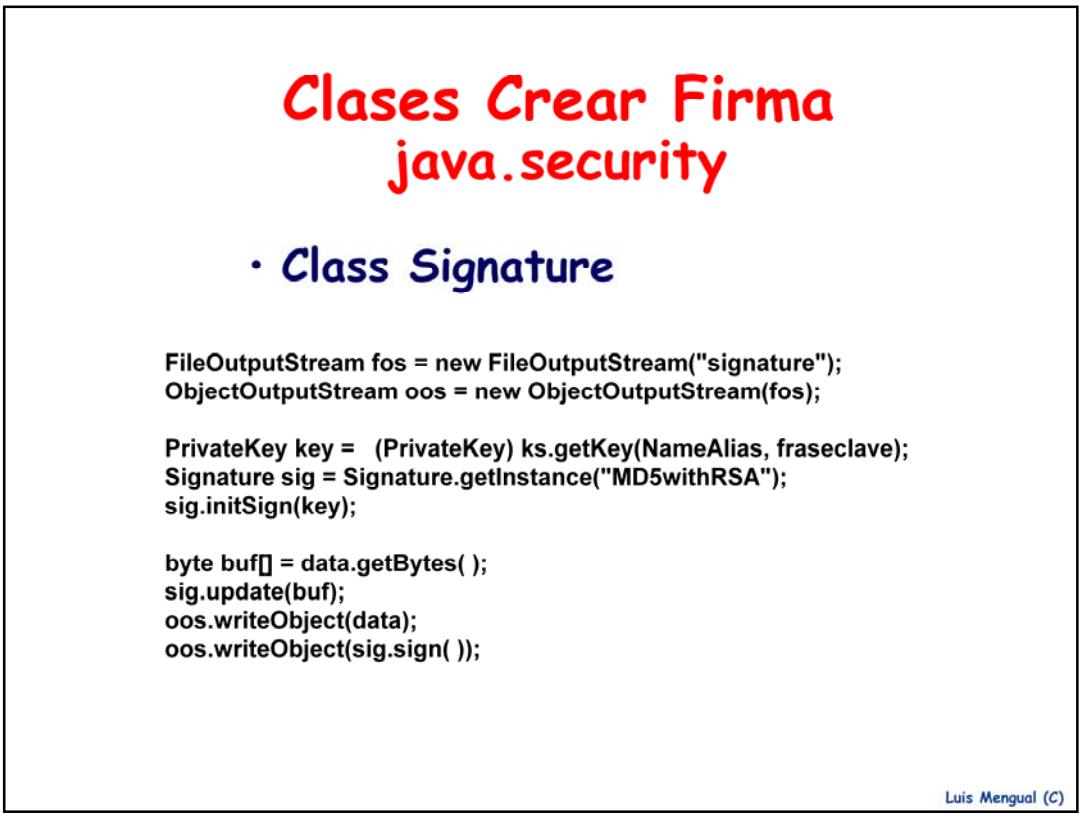
Para la verificación de una firma sería solo necesario el certificado de clave pública del usuario.

En el proceso de verificación de firma la clave pública del usuario se aplica al segundo elemento de la firma, obteniéndose un resumen (hash) que se compara con el resultado de aplicar la función hash al primer elemento de la firma. Si ambos valores coinciden se considera verificada la firma.

Proponemos dos ejemplos de firma de un texto utilizando las extensiones de seguridad de Java.

En el primer ejemplo asumimos que el usuario genera con la herramienta "keytool" un par de claves privada-secreta y el certificado asociado. El proceso de firma en este primer ejemplo lo haremos accediendo al almacén desde el código java y firmando con la clave privada. En el segundo ejemplo cogeremos la clave privada desde un fichero en formato binario (DER).

En cualquiera de los casos necesitaremos el certificado de clave publica para la verificación de la firma. Éste lo podemos obtener a partir del almacén con la herramienta "Keytool" como aparece descrito en la figura.



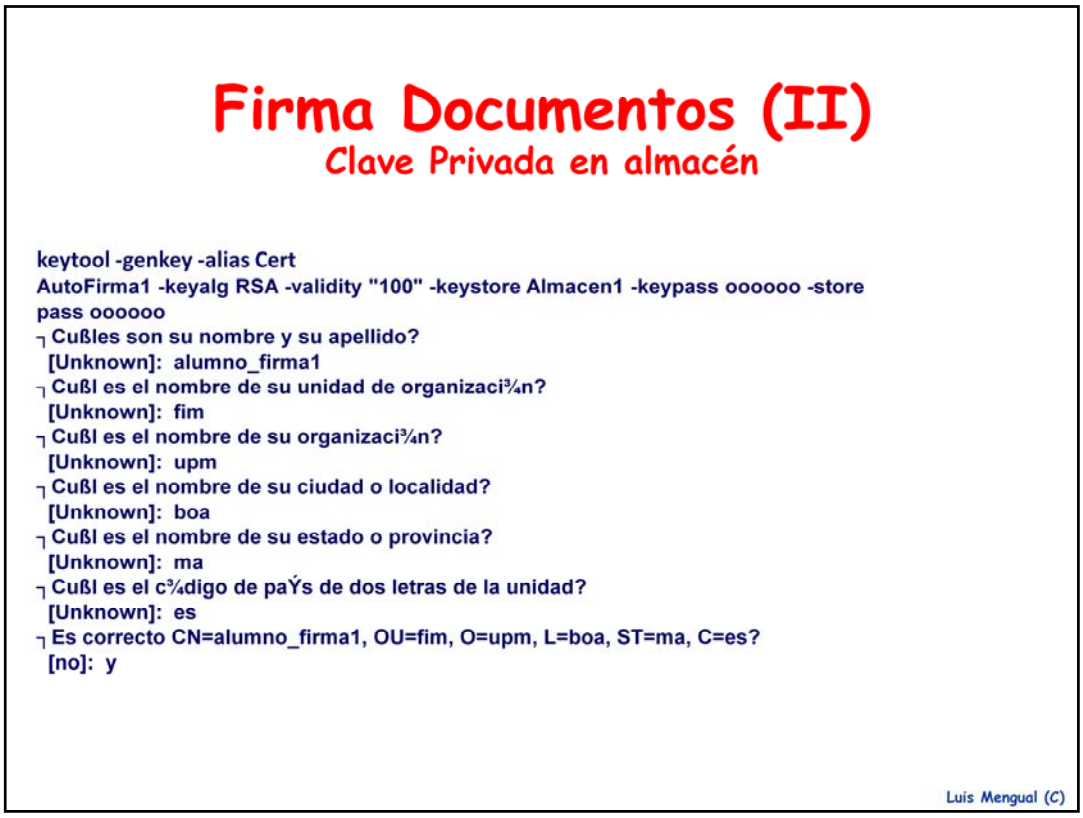
La clase "Signature" del paquete "java.security" nos permite crear un firma digital.

Con el método "update" de esta clase se actualizan los datos a ser firmados o verificados usando un array de bytes.

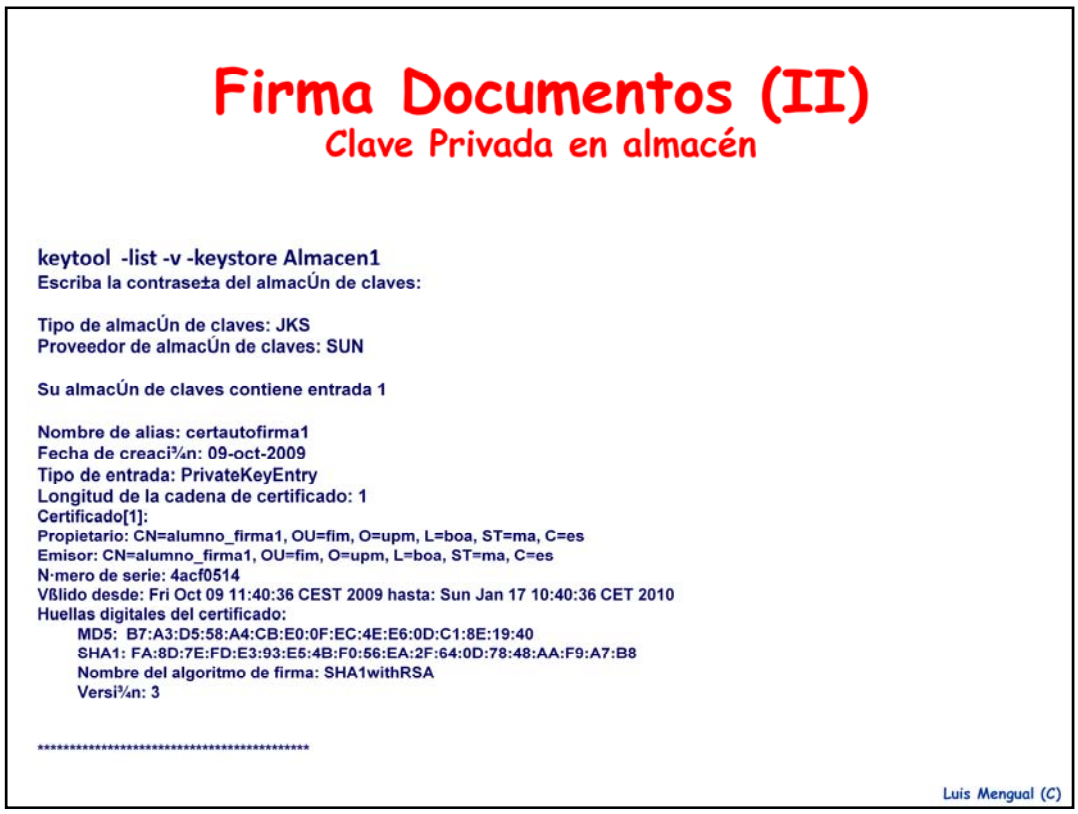
Los dos elementos de la firma son incorporados al fichero de firma con el método "writeObject" de la clase "ObjectOutputStream". Estos dos elementos son los datos a firmar y los datos firmados.

En la figura el fichero que contiene la firma se llama "signature".

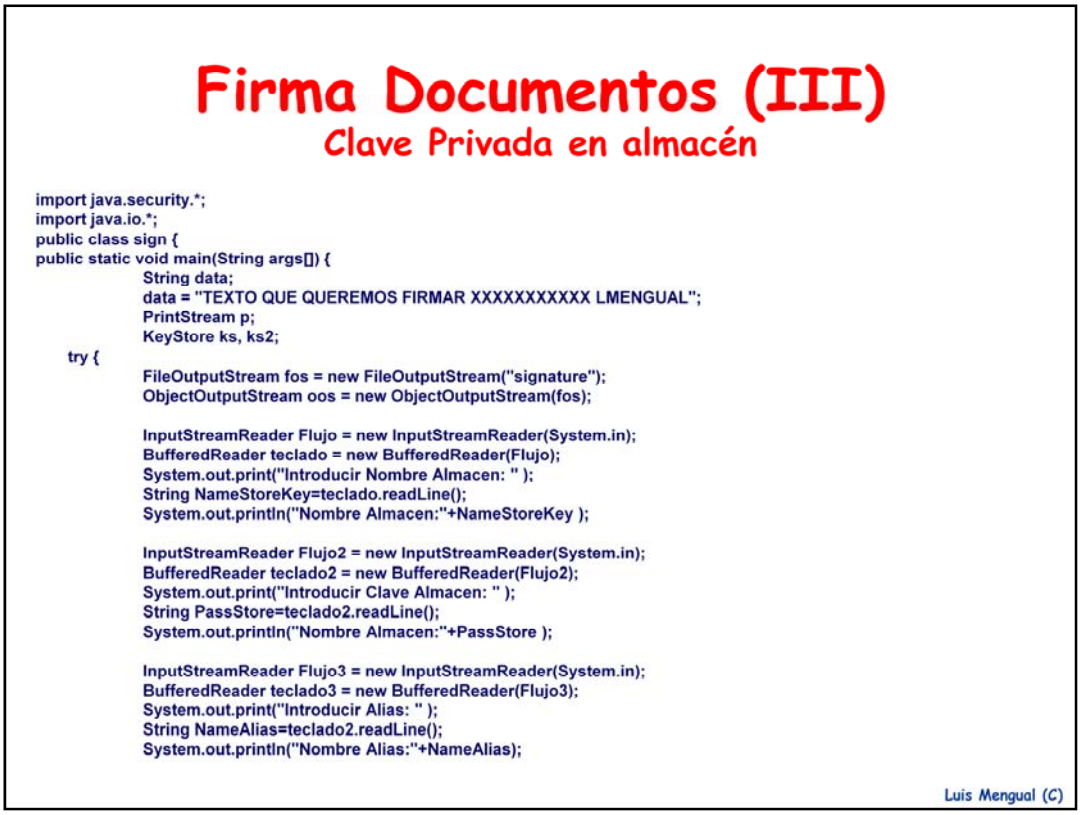
Notar que el método "sing" de la clase "Signature" devuelve los bytes firmados de todos los datos incorporados con el método "update".



En la figura vemos una vez más el proceso de generación de un par de claves privada-publica RSA y su certificado asociado, así como su almacenamiento en el almacén "Almacen1".

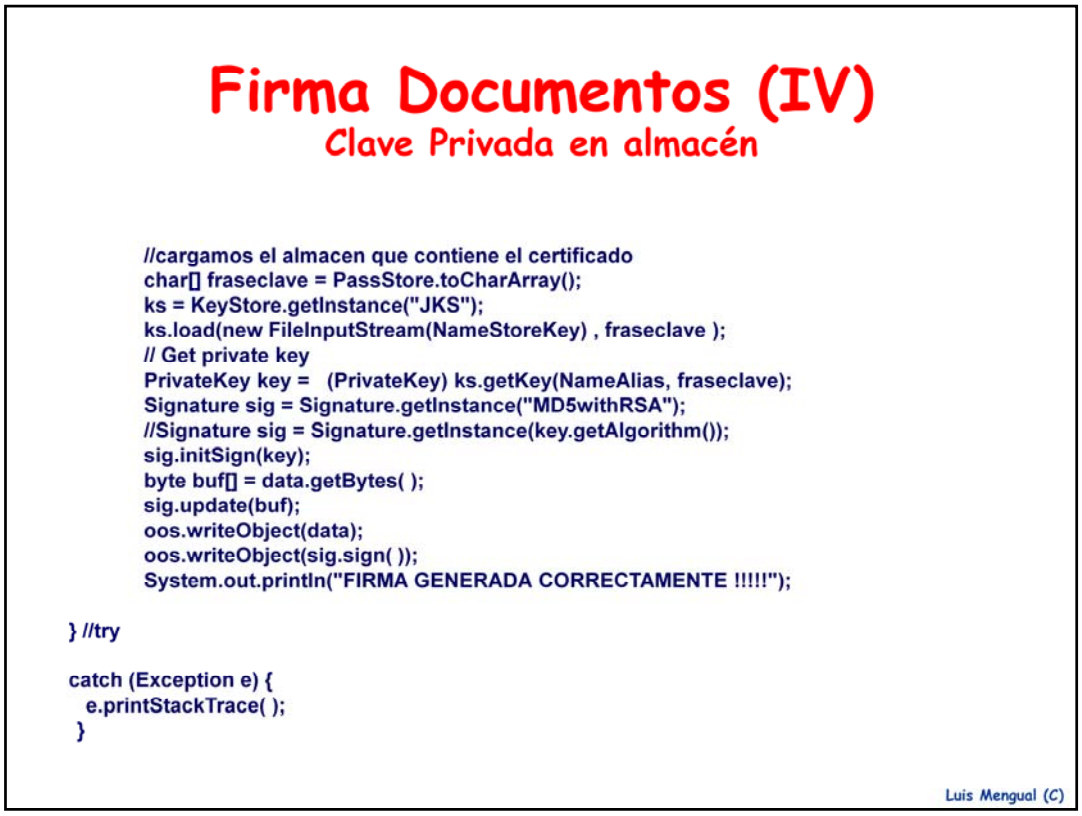


Visualizamos el contenido del "Almacen1"



En el código fuente de este primer ejemplo vamos a generar la firma accediendo directamente al almacén de claves.

En la figura introducimos el nombre del almacén de certificado, su clave y el alias …donde se haya la entrada



Creamos una instancia de la clase "keystore", cargamos el almacén del fichero "Alamcen1" y obtenemos la clave privada de la entrada con el método "getKey" de la clase "keystore":

PrivateKey key = (PrivateKey) ks.getKey(NameAlias, fraseclave);

A continuación nos creamos un instancia de la clase "Signature" y la inicializamos con la clave privada que usaremos para firmar el texto. El texto a firmar será introducido como un array con el método "update". Los datos firmados se obtendrán con el método "sing".

En paralelo escribimos en forma de objetos los datos a firmar y la firma generada en el fichero de firma "oos":

```
sig.initSign(key);
byte buf[] = data.getBytes( );
sig.update(buf);
oos.writeObject(data);
oos.writeObject(sig.sign( ));
```

La verificación de firma es el proceso inverso al de la creación de la firma. Para ello necesitamos en primer lugar la clave pública par de la clave privada que firma el texto.

Esta clave pública la podemos obtener a partir de un fichero de certificado como hemos visto en ejemplos anteriores "Lectura de Certificado a partir de un fichero".

Para ello utilizamos la clase "CertificateFactory" y el método "generateCertificate" de esta clase para crear un certificado (un objeto de la clase "X509Certificate") a partir de un fichero.

A continuación con el método "getPublicKey()" de la clase "X509Certificate" obtenemos la clave publica que utlizaremos en el proceso de verificación.

El proceso de verificación comienza al instanciar un objeto de la clase signature. Con el método "initVerify(pk)" introducimos la clave pública que utilizaremos para la verificación de la firma.

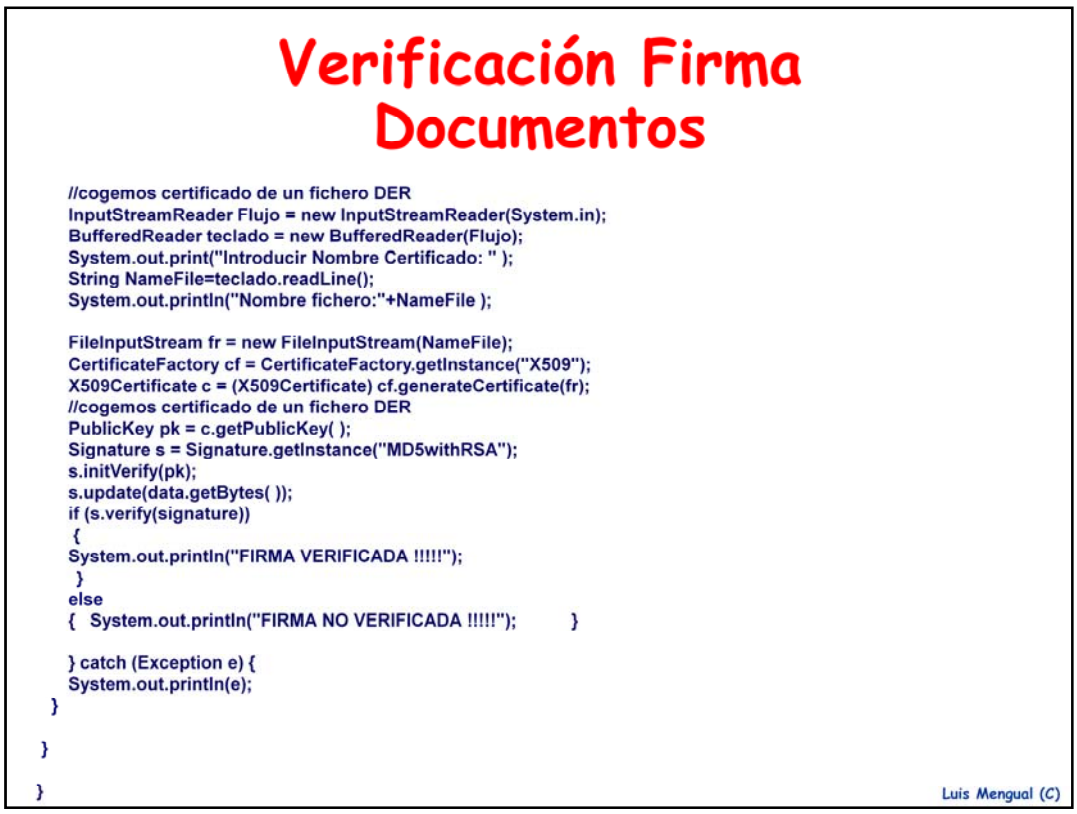
Los contenidos de firma van a estar en un fichero. De este fichero leeremos los dos elementos de la firma digital: los datos en claro y el resumen de los datos cifrados con la clave privada. En la figura estos dos elementos serían "data" y "signature"

Con el método "update" de la clase "Signature" introducimos los datos en claro en forma de array. Finalmente con el método "verify" se verifica la veracidad de la firma.

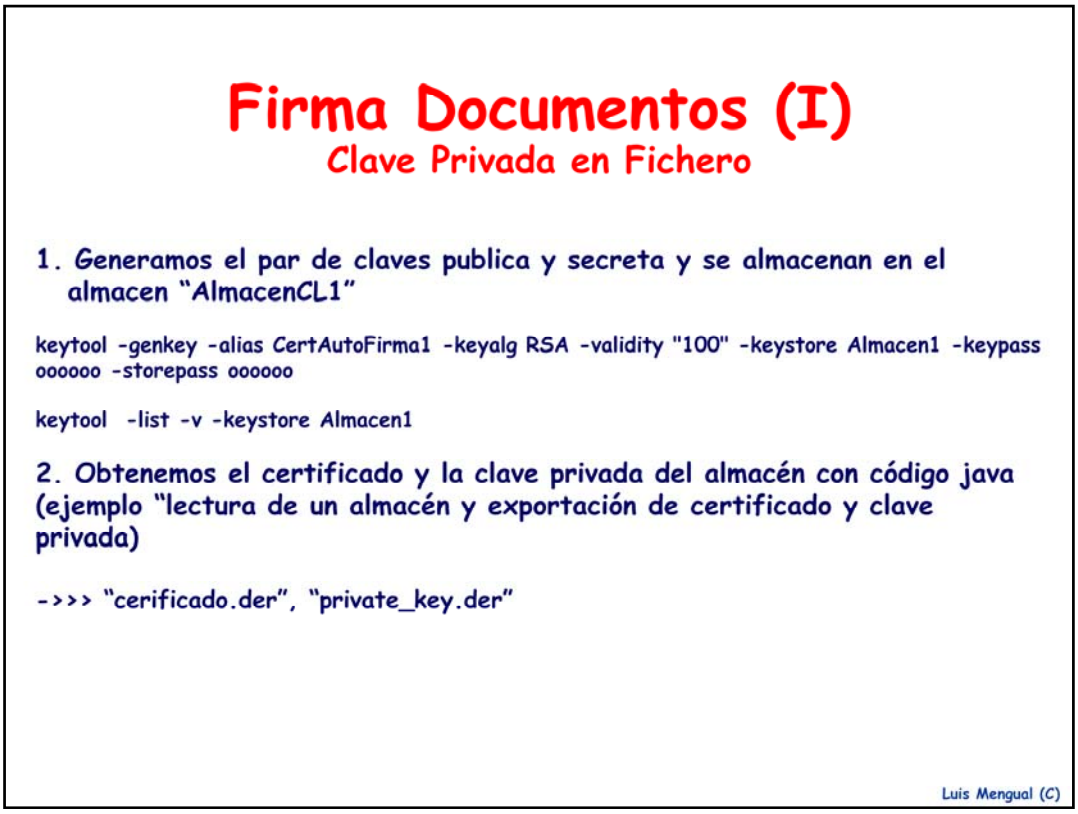
```
X509Certificate c = (X509Certificate) cf.generateCertificate(fr);
PublicKey pk = c.getPublicKey( );
Signature s = Signature.getInstance("MD5withRSA");
s.initVerify(pk);
s.update(data.getBytes( ));
if (s.verify(signature))System.out.println("FIRMA VERIFICADA !!!!!");
```

En la figura se tiene el código fuente completo de un ejemplo de verificación de firma.

En la figura lo primero que se observa es la extracción de un fichero de firma de los dos elementos que componen la firma. Para ello se utiliza el método "readObject" de la clase "ObjectInputStream"



En la figura se muestra el código fuente correspondiente a la lectura del certificado de clave pública, la creación del objeto de la clase "Signature" y el proceso ya descrito de verificación de firma.



En este ejemplo proponemos otra alternativa para la generación de una firma en java. En este caso la clave privada de firma esta disponible en un fichero.

Para ello seguimos en el planteamiento de un usuario que genera un par de claves privada-pública y el certificado asociado con la herramienta "keytool". Hacemos la hipótesis de que el usuario ha exportado esa clave secreta a un fichero. Para ello nos remitimos al ejemplo que vimos con anterioridad "lectura de un almacén y exportación de certificado y clave privada".

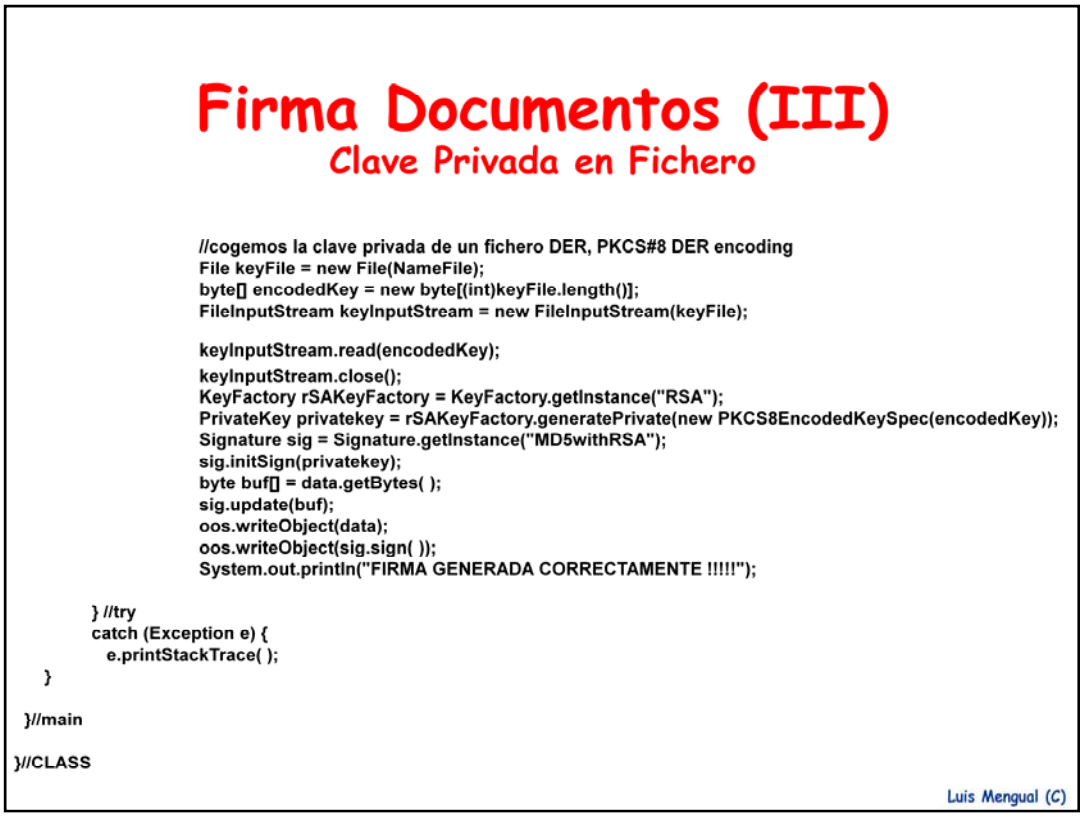
Firma Documentos (II)

Clave Privada en Fichero

```
import java.security.*;
import java.security.spec.*;
import java.security.cert.*;
import java.io.*;
import java.util.*;

public class sign {

public static void main(String args[]) {

                String privateKeyFileName ;
                String data;
                data = "TEXTO QUE QUEREMOS FIRMAR XXXXXXXXXX LMENGUAL";
                PrintStream p;
                KeyStore ks, ks2;

        try {
                FileOutputStream fos = new FileOutputStream("signature");
                ObjectOutputStream oos = new ObjectOutputStream(fos);

        //cogemos la clave privada de un fichero DER
                InputStreamReader Flujo = new InputStreamReader(System.in);
                BufferedReader teclado = new BufferedReader(Flujo);
                System.out.print("Introducir Nombre Fichero Clave Privada: " );
                String NameFile=teclado.readLine();
                System.out.println("Nombre fichero:"+NameFile );
```

Luis Mengual (C)

Teniendo disponible la clave privada en un fichero, se puede leer desde código java ..



..para ello creamos un objeto de la clase "KeyFactory", y utilizamos el método "generatePrivate" para obtener un objeto de la clase "privatekey" que podamos introducir en el objeto de firma.

Para hacer esto además hay que invocar al objeto PKCS8EncodedKeySpec para interprete la codificación de la clave privada.

KeyFactory rSAKeyFactory = KeyFactory.getInstance("RSA");

PrivateKey privatekey = rSAKeyFactory.generatePrivate(new PKCS8EncodedKeySpec(encodedKey));

Introducir Clave Privada y Certificado en un almacen (I)

```
import java.security.*; //Para manejar los almacenes de Claves
import java.security.spec.*; // conversion PKCS8EncodedKeySpec
import java.security.cert.*;
import java.io.*;
import java.util.*;

public class ImportCertKey {
public static void main(String args[]) {
        try {
                KeyStore keyStore = null; // Creamos un keystore vacio
                keyStore = KeyStore.getInstance("JKS");
                keyStore.load(null, "oooooo".toCharArray());

                // cargamos el certificado
                InputStreamReader Flujo1 = new InputStreamReader(System.in);
                BufferedReader teclado1= new BufferedReader(Flujo1);
                System.out.print("Introducir Nombre Certificado: " );
                String NameFile1=teclado1.readLine();
                FileInputStream certificateStream = new FileInputStream(NameFile1);
                CertificateFactory certificateFactory = CertificateFactory.getInstance("X509");
                java.security.cert.Certificate[]  chain = {};
                chain = certificateFactory.generateCertificates(certificateStream).toArray(chain);
                certificateStream.close();
```

Luis Mengual (C)

Introducir Clave Privada y Certificado en un almacen (II)

```
            //cogemos la clave privada de un fichero DER
            InputStreamReader Flujo2 = new InputStreamReader(System.in);
            BufferedReader teclado2 = new BufferedReader(Flujo2);
            System.out.print("Introducir Nombre Fichero Clave Privada: " );
            String NameFile2=teclado2.readLine();
            File keyFile = new File(NameFile2);
            byte[] encodedKey = new byte[(int)keyFile.length()];
            FileInputStream keyInputStream = new FileInputStream(keyFile);
            keyInputStream.read(encodedKey);
            keyInputStream.close();
            KeyFactory rSAKeyFactory = KeyFactory.getInstance("RSA");
            PrivateKey privatekey = rSAKeyFactory.generatePrivate(new PKCS8EncodedKeySpec(encodedKey));

            System.out.println("password de clave privada : ");
            String privateKeyEntryPassword = (new BufferedReader(new InputStreamReader(System.in))).readLine();
            char[] fraseclave2 = privateKeyEntryPassword.toCharArray();

            InputStreamReader Flujo5 = new InputStreamReader(System.in);
            BufferedReader teclado5 = new BufferedReader(Flujo5);
            System.out.print("Introducir Nombre Alias: " );
            String Alias=teclado5.readLine();
            //la clave privada tiene que corresponden con el certificado que enta em el array 0 de chaihn
            keyStore.setEntry( Alias,  new KeyStore.PrivateKeyEntry( privatekey, chain ), new KeyStore.PasswordProtection(fraseclave2) );
            // con estas lineas se escribe en el almacen nuevo el certificado y la clave privada
            FileOutputStream keyStoreOutputStream = new FileOutputStream("AlmacenNuevo");
             keyStore.store(keyStoreOutputStream, "oooooo".toCharArray());
               keyStoreOutputStream.close();

        } catch (Exception e) {
            e.printStackTrace( );
        }
    }//main
}//class
```

Luis Mengual (C)