RENIEC
REGISTRO NACIONAL DE IDENTIFICACIÓN Y ESTADO CIVIL

Especificación Técnica: PKI y ABI

DNI Electrónico

Identificación segura

Alvaro Cuno
GCRD -SGCD
22/07/2015

DNI Electrónico

Funcionalidades

- ABI
- AAI
- PKI
- MoC

(en formato ICAO)

DNI Electrónico

Funcionalidades

- ABI
- AAI
- PKI
- MoC

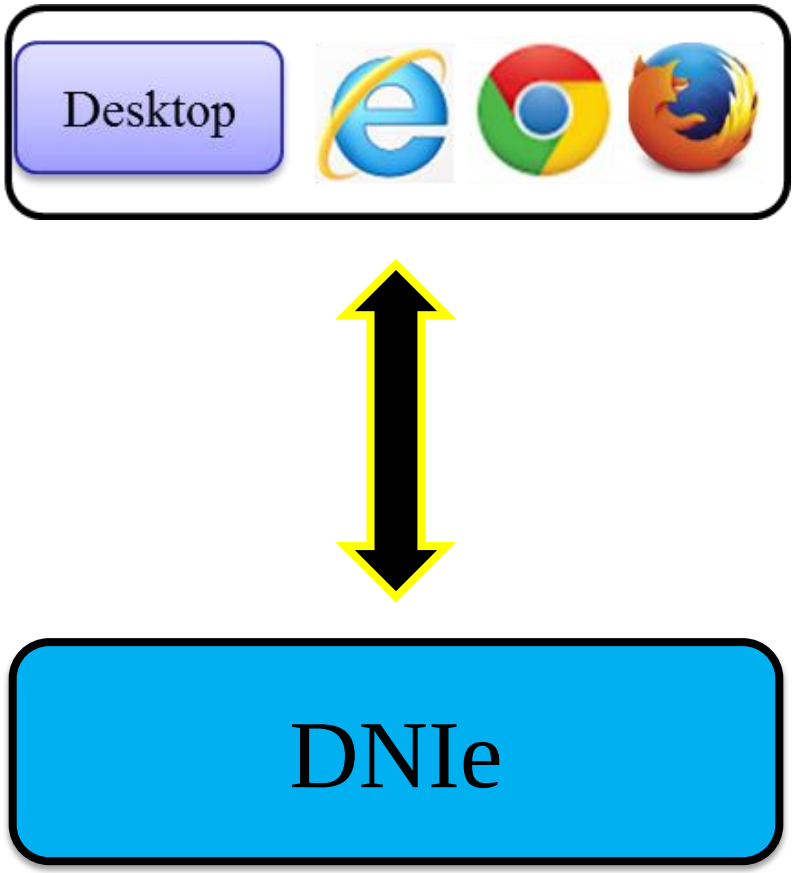
- Certificado raíz ECERNEP
- Certificado digital ECEP
- Certificado de Firma Digital
- Certificado de Autenticación
- Clave privada de Firma Digital → PIN 2
- Clave privada de Autenticación → PIN 1

Para hacer uso de las claves privadas del ciudadano se necesita conocer el PIN respectivo

Estructura PKCS#15

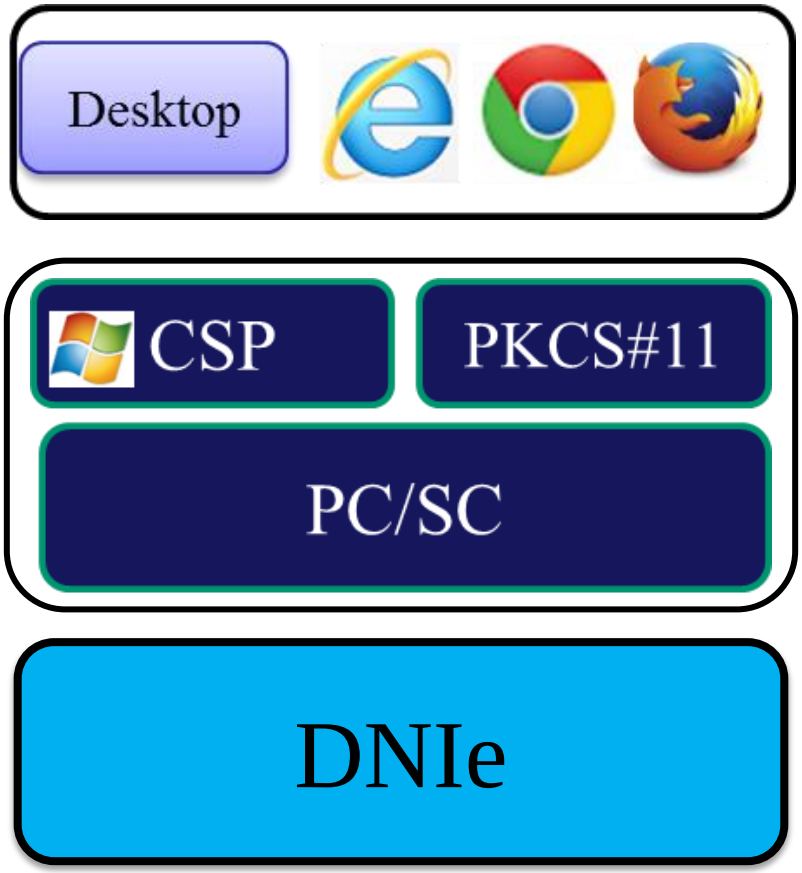
DNI Electrónico

Accediendo a un DNIe



DNI Electrónico

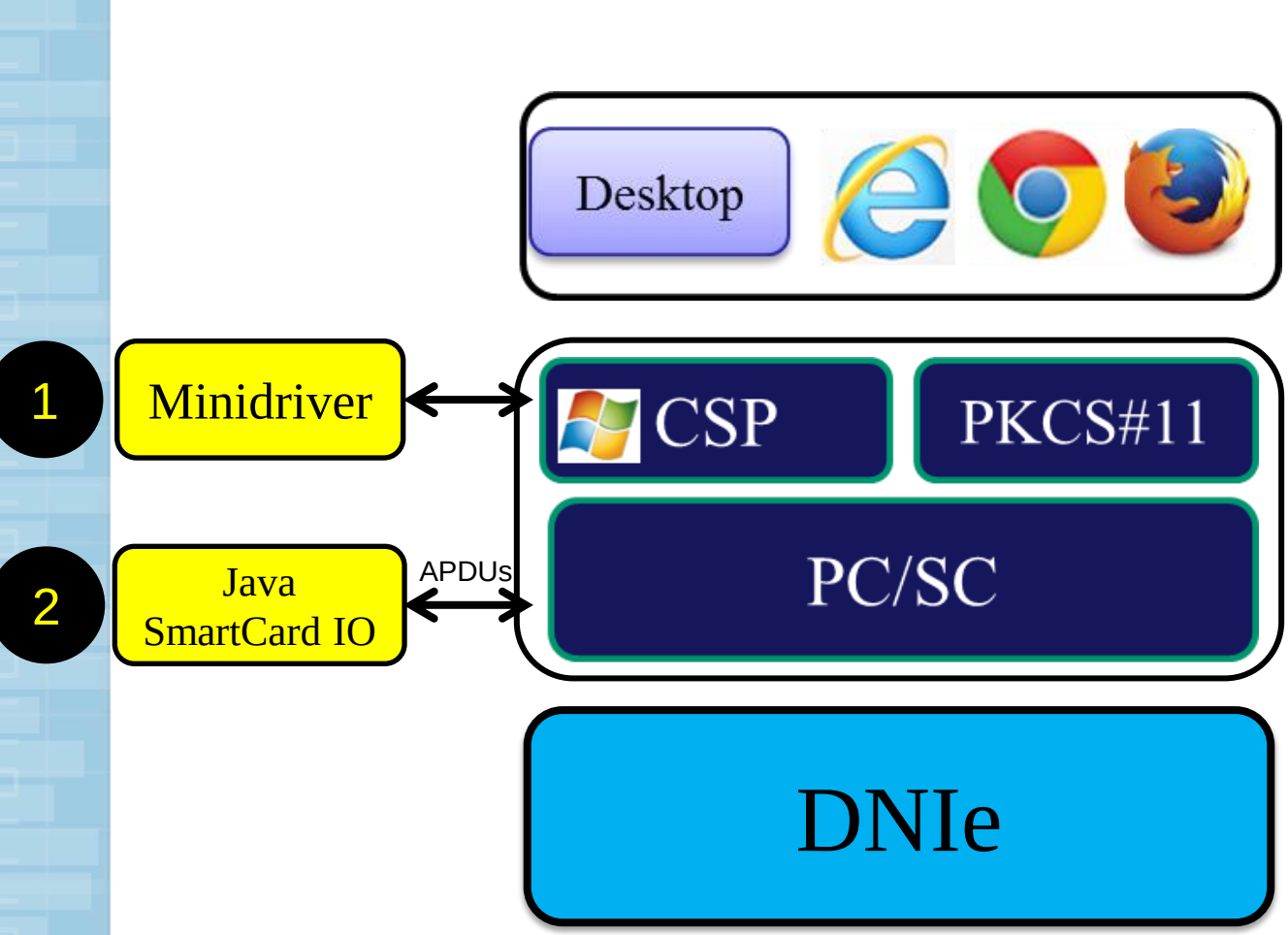
Accediendo a un DNIe



Plataforma PC

DNI Electrónico

Accediendo a un DNIe



DNI Electrónico

Minidriver

DNI Electrónico

Minidriver

Internet

Lector convencional de tarjetas de contactos (ISO 7816)

DNIe

JC Provider

App Java Desktop

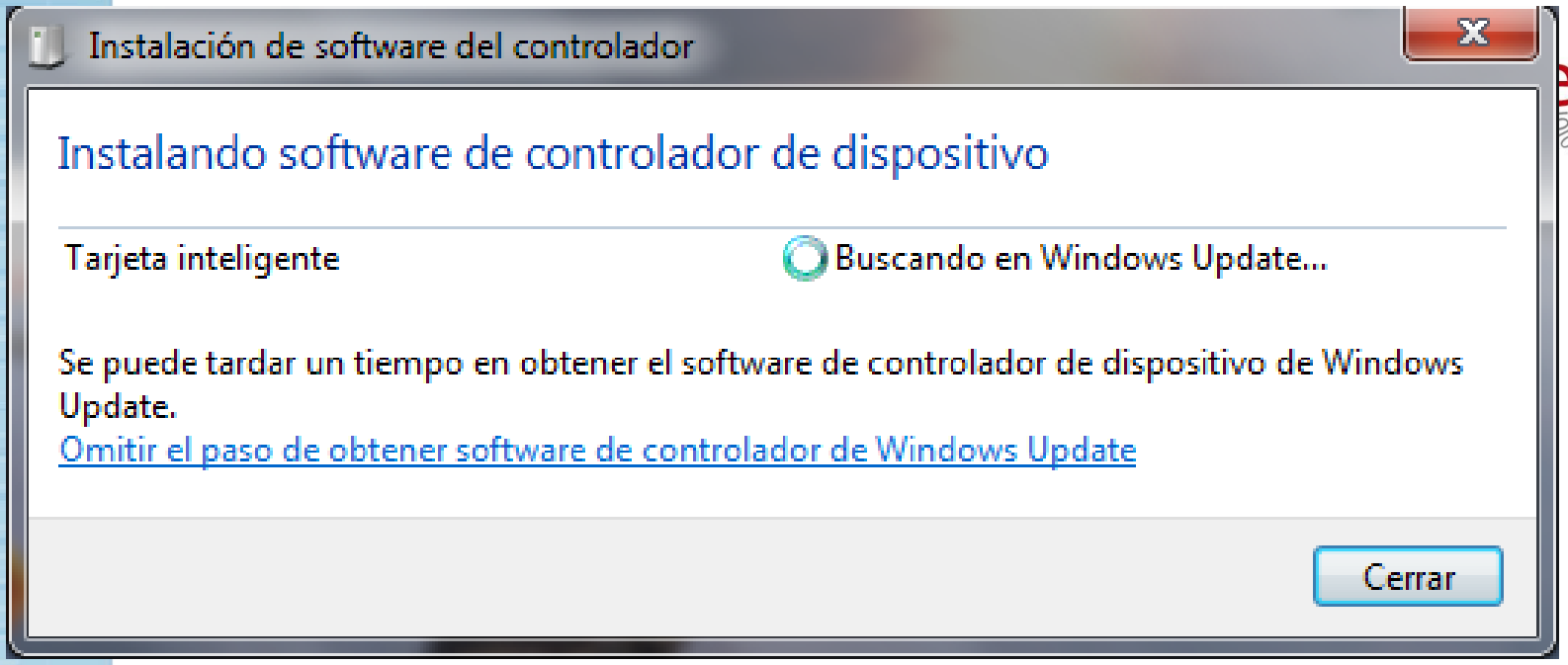
Java

Minidriver

SO Windows

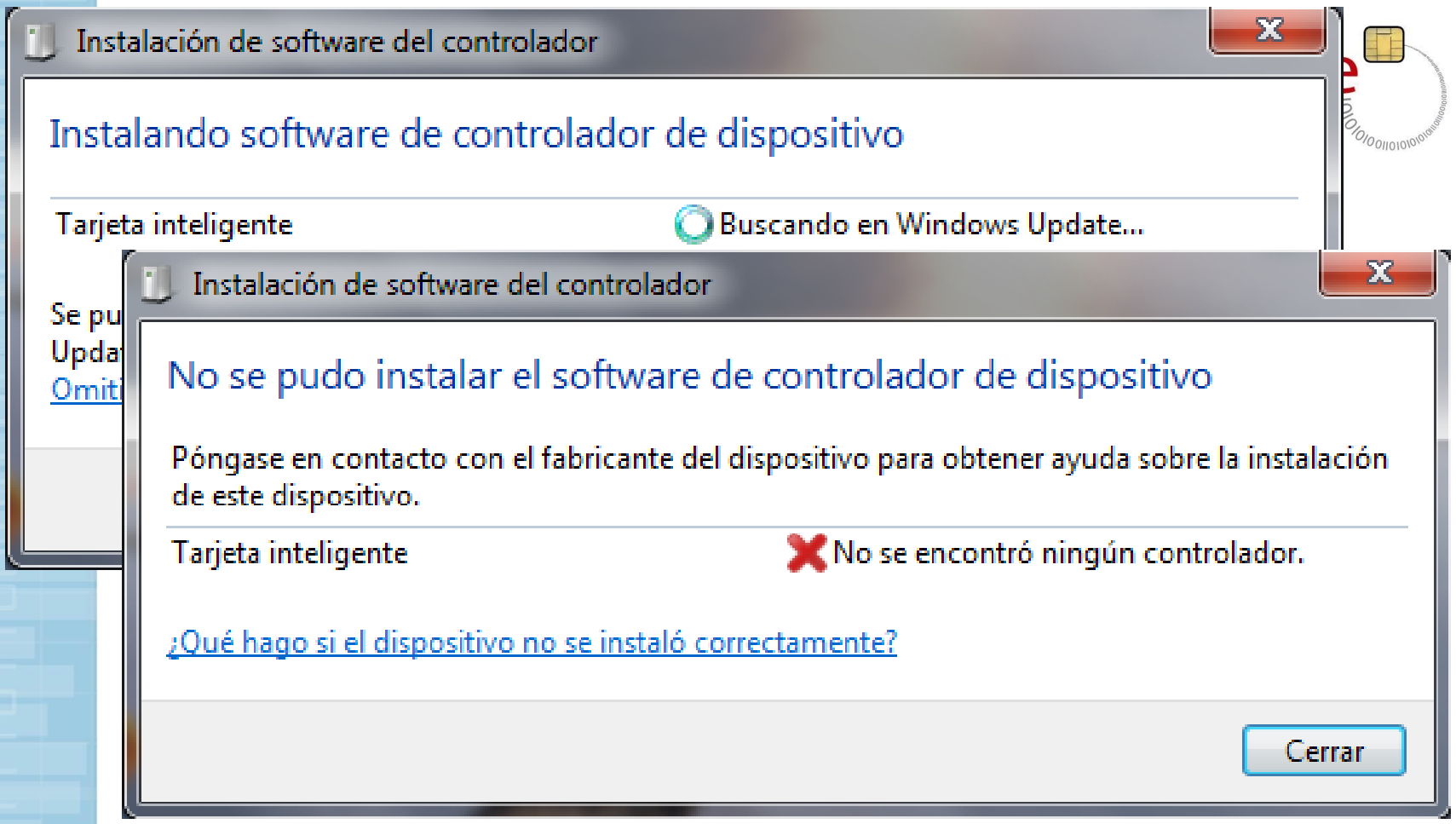
DNI Electrónico

Minidriver



DNI Electrónico

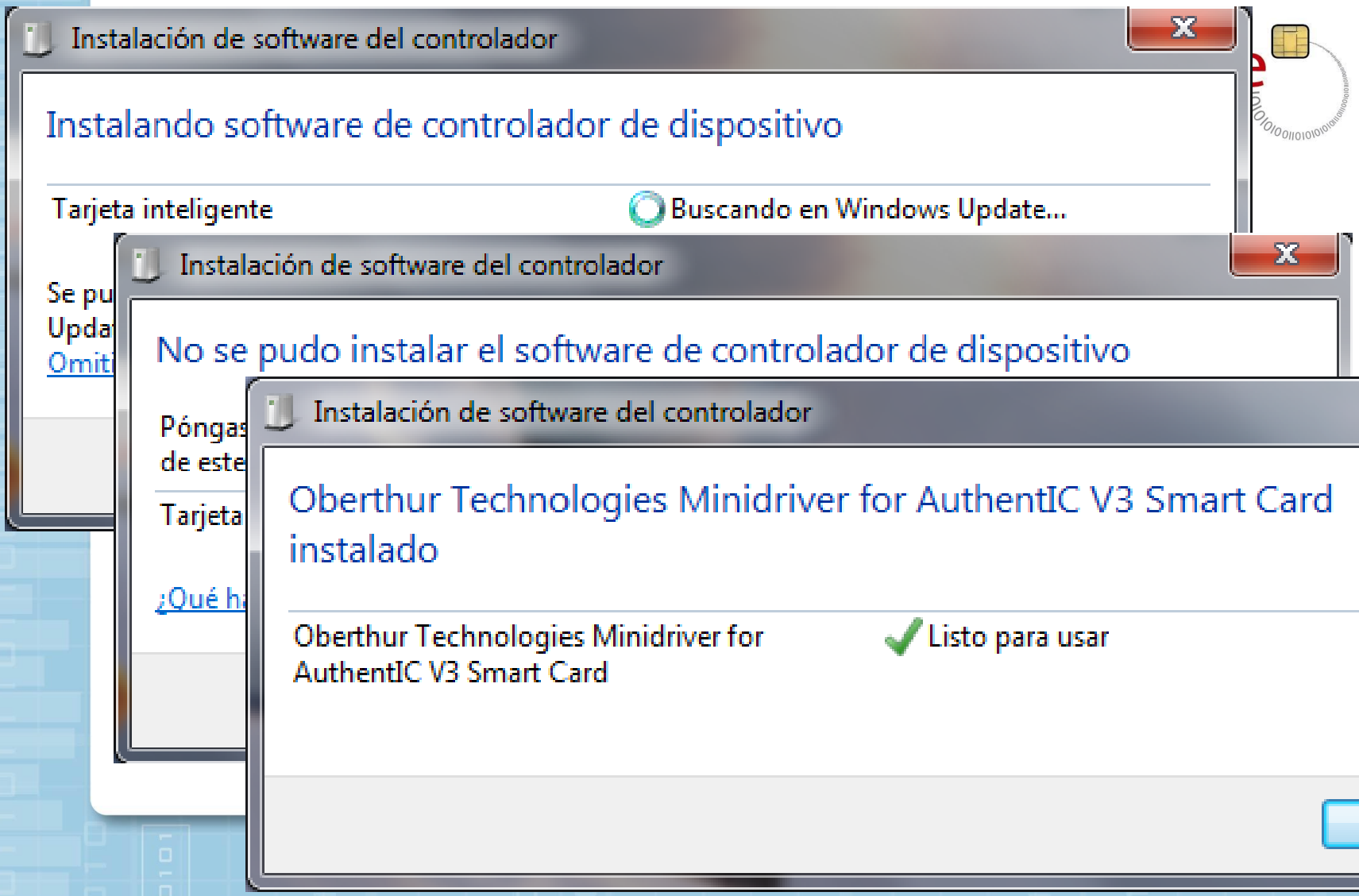
Minidriver



Habilitar acceso a internet o el Windows update para el usuario

DNI Electrónico

Minidriver



DNI Electrónico

Java cryptographic providers

- SunPKCS11
- SUN
- SunRsaSign
- SunJSSE
- SunJCE
- SunJGSS
- SunSASL
- XMLDSig
- SunPCSC
- SunMSCAPI
- SunEC (Java 7)

SunMSCAPI (Java 6)

Engine	Algorithm Name(s)
`Cipher`	RSA *RSA/ECB/PKCS1Padding only*
`KeyPairGenerator`	RSA
`KeyStore`	Windows-MY The keystore type that identifies the native Microsoft Windows MY keystore. It contains the user's personal certificates and associated private keys. Windows-ROOT The keystore type that identifies the native Microsoft Windows ROOT keystore. It contains the certificates of Root certificate authorities and other self-signed trusted certificates.
`SecureRandom`	Windows-PRNG The name of the native pseudo-random number generation (PRNG) algorithm.
`Signature`	MD2withRSA MD5withRSA SHA1withRSA

DNI Electrónico

Java cryptographic providers

- SunPKCS11
- SUN
- SunRsaSign
- SunJSSE
- SunJCE
- SunJGSS
- SunSASL
- XMLDSig
- SunPCSC
- SunMSCAPI
- SunEC (Java 7)

SunMSCAPI (Java 7)

Engine	Algorithm Names
`Cipher`	RSA *RSA/ECB/PKCS1Padding only*
`KeyPairGenerator`	RSA
`KeyStore`	Windows-MY The keystore type that identifies the native Microsoft Windows MY keystore. It contains the user's personal certificates and associated private keys. Windows-ROOT The keystore type that identifies the native Microsoft Windows ROOT keystore. It contains the certificates of Root certificate authorities and other self-signed trusted certificates.
`SecureRandom`	Windows-PRNG The name of the native pseudo-random number generation (PRNG) algorithm.
`Signature`	MD5withRSA MD2withRSA NONEwithRSA SHA1withRSA SHA256withRSA SHA384withRSA SHA512withRSA

DNI Electrónico

RENIEC
REGISTRO NACIONAL DE IDENTIFICACIÓN Y ESTADO CIVIL

Java cryptographic providers

- SunPKCS11
- SUN
- SunRsaSign
- SunJSSE
- SunJCE
- SunJGSS
- SunSASL
- XMLDSig
- SunPCSC
- SunMSCAPI
- SunEC
- OracleUcrypto
- Apple

SunMSCAPI (Java 8)

DNIe

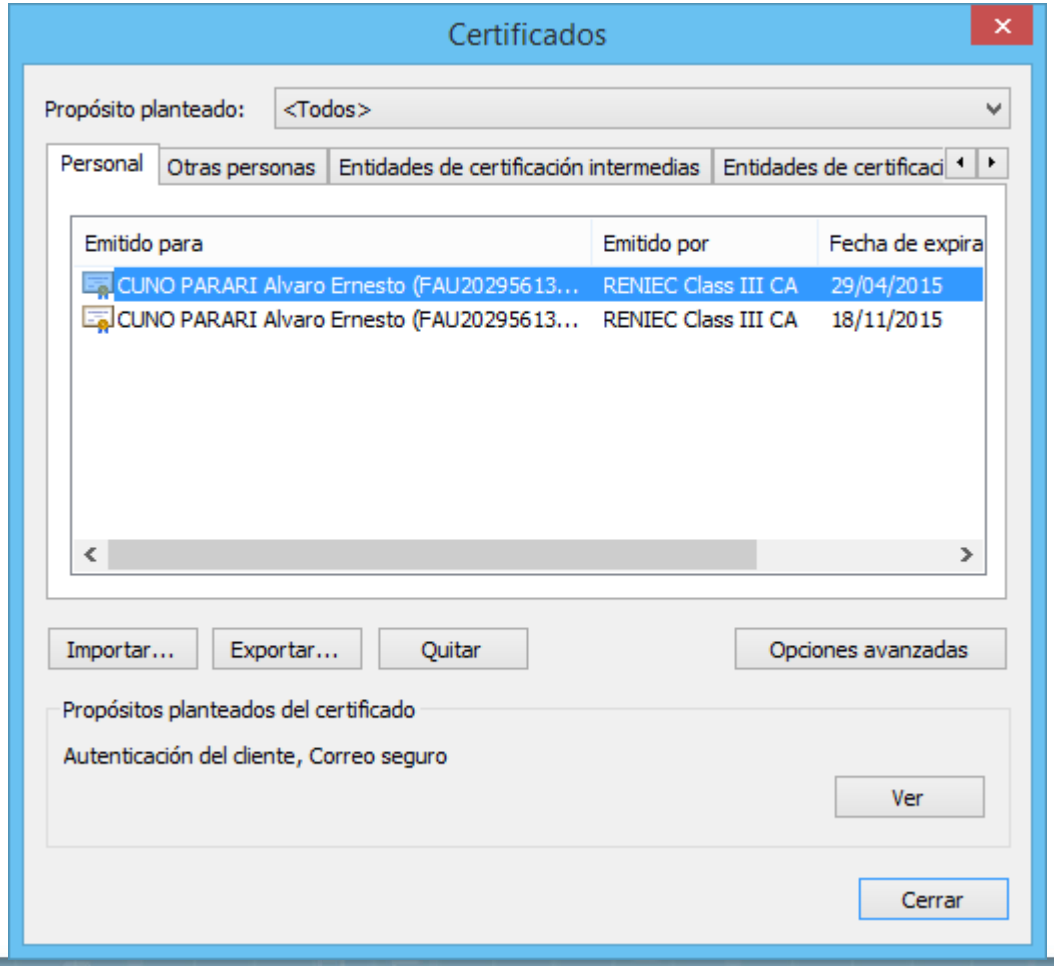
Engine	Algorithm Names
AlgorithmParameterGenerator	DSA
AlgorithmParameters	DSA
CertificateFactory	X.509
CertPathBuilder	PKIX
CertPathValidator	PKIX
CertStore	Collection LDAP
Configuration	JavaLoginConfig
KeyFactory	DSA
KeyPairGenerator	DSA
KeyStore	JKS DKS
MessageDigest	MD2 MD5 SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
Policy	JavaPolicy
SecureRandom	SHA1PRNG (Initial seeding is currently done via a combination of system attributes and the `java.security` entropy gathering device) NativePRNG (`nextBytes()` uses `/dev/urandom`, `generateSeed()` uses `/dev/random`) NativePRNGBlocking (`nextBytes()` and `generateSeed()` use `/dev/random`) NativePRNGNonBlocking (`nextBytes()` and `generateSeed()` use `/dev/urandom`)
Signature	NONEwithDSA SHA1withDSA SHA224withDSA SHA256withDSA

Ejemplo 1: iterando el keystore Windows-MY

KeyStore	Windows-MY The keystore type that identifies the native Microsoft Windows MY keystore. It contains the user's personal certificates and associated private keys. Windows-ROOT The keystore type that identifies the native Microsoft Windows ROOT keystore. It contains the certificates of Root certificate authorities and other self-signed trusted certificates.

DNI Electrónico

Ejemplo 1: iterando el keystore Windows-MY

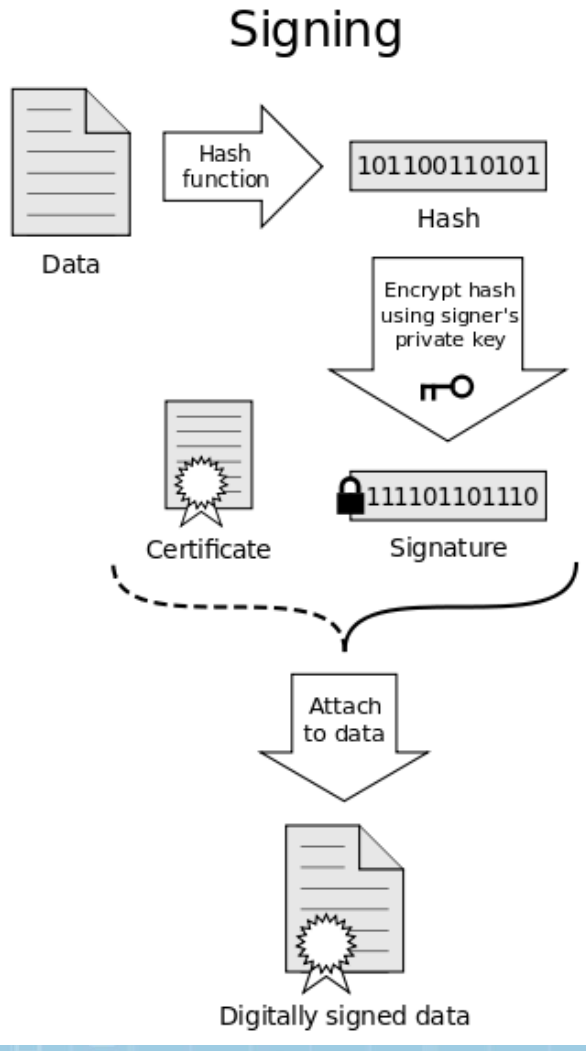


DNI Electrónico

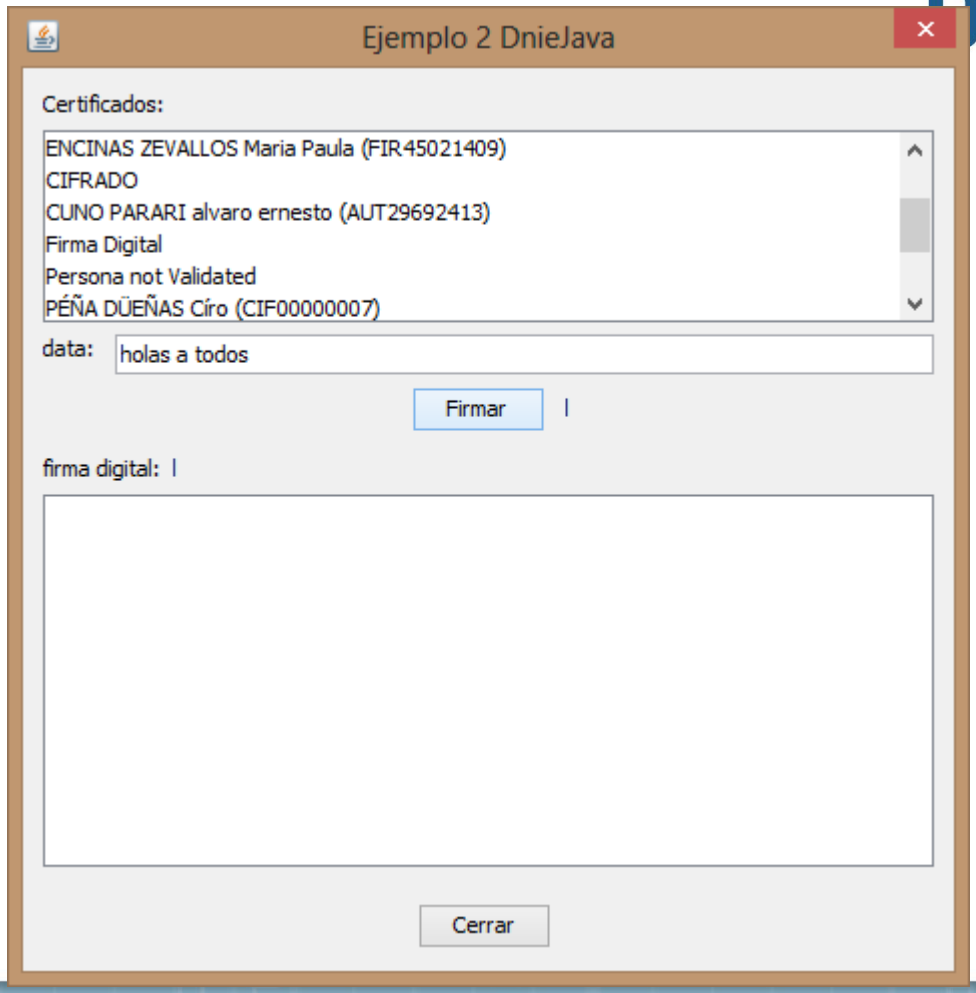
Ejemplo 1: iterando el keystore Windows-MY

```
static void ejemplo1() throws
        KeyStoreException, IOException, NoSuchAlgorithmException,
        CertificateException {

    SunMSCAPI providerMSCAPI = new SunMSCAPI();
    Security.addProvider(providerMSCAPI);
    KeyStore ks = KeyStore.getInstance("Windows-MY");
    ks.load(null,null);

    Enumeration<String> aliases = ks.aliases();
    while (aliases.hasMoreElements()) {
       String alias = aliases.nextElement();
       System.out.println("Alias: "+ alias);
       if (ks.isCertificateEntry(alias)){
          System.out.println(">>> is cert");
       } else {
          System.out.println(">>> is key");
       }
    }
}
```

Ejemplo 2: generación de firma digital

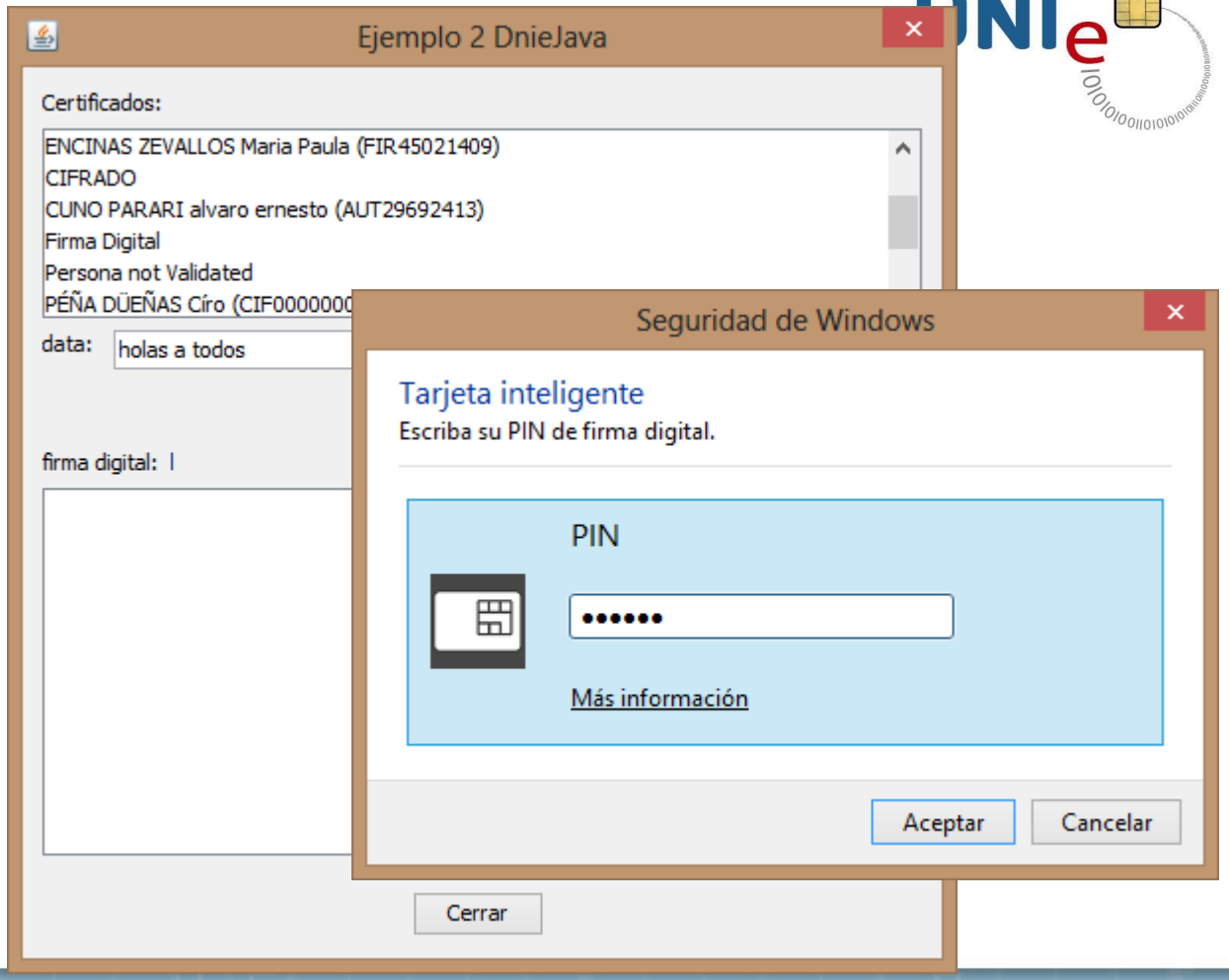


Ejemplo 2: generación de firma digital



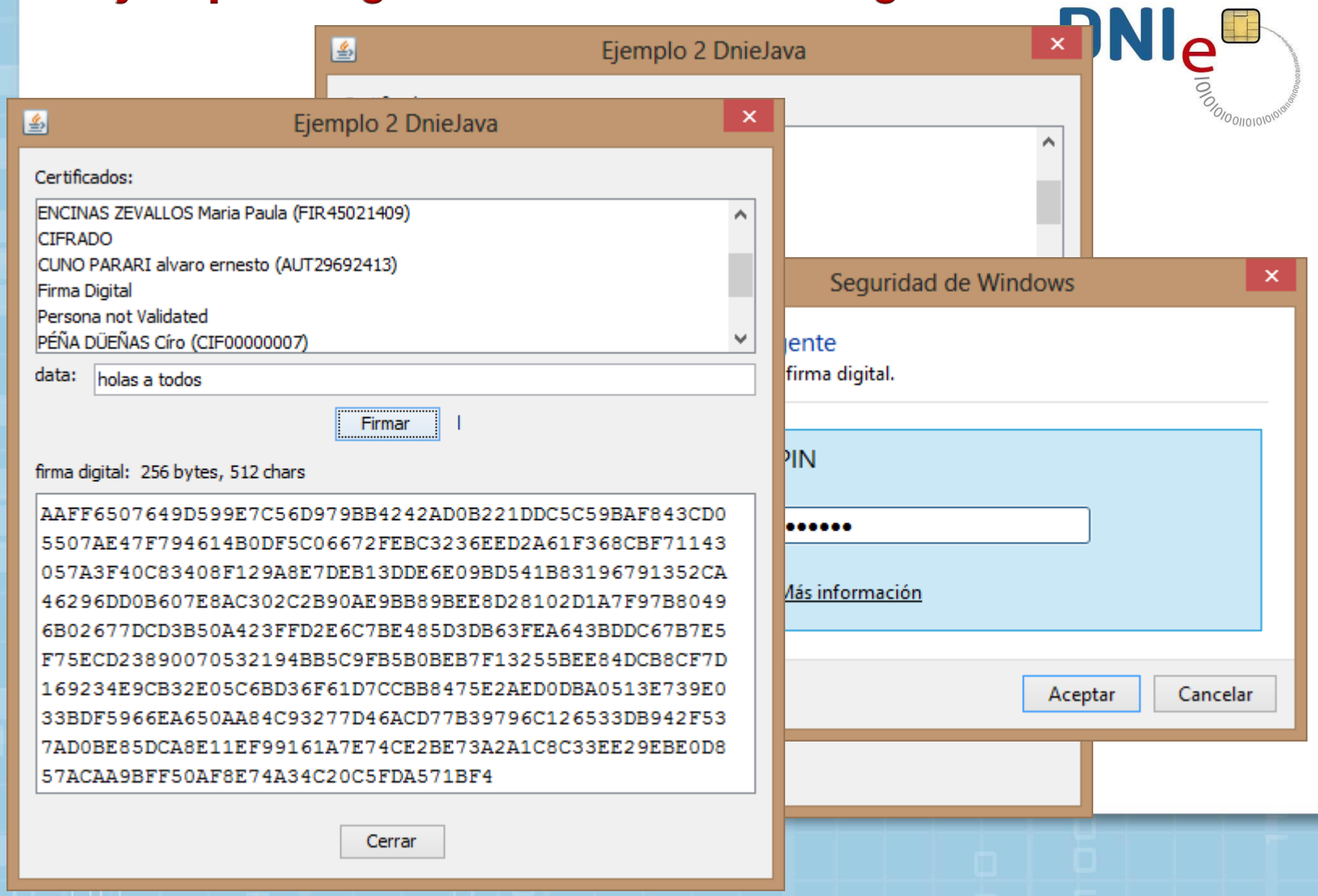
DNI Electrónico

Ejemplo 2: generación de firma digital



DNI Electrónico

Ejemplo 2: generación de firma digital



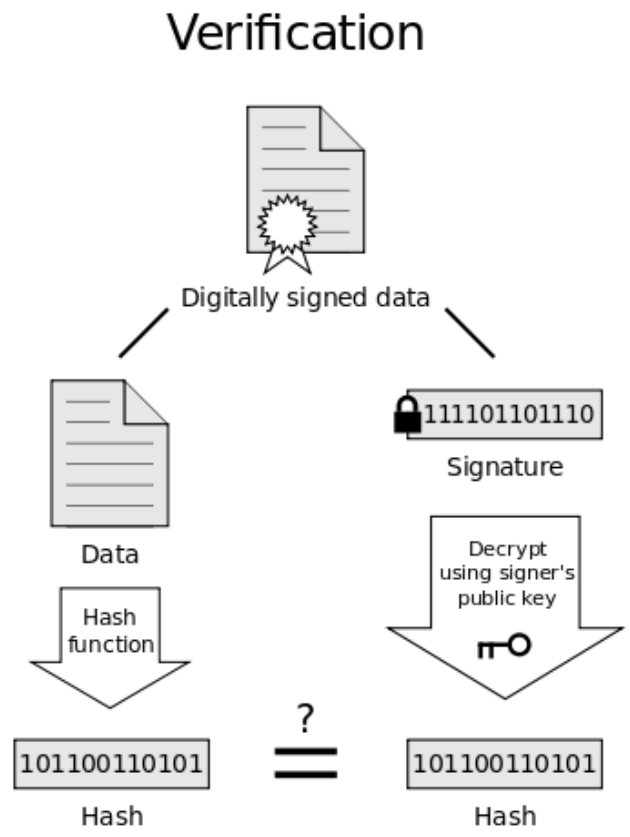
Ejemplo 2: generación de firma digital


```
JList certsList = new JList();
CertificateListModel certsListModel = new CertificateListModel();
List<PrivateKey> privateKeyList = new ArrayList<PrivateKey>();
PrivateKey selectedPrivateKey = null;

        class CertificateListModel extends DefaultListModel {

           public void refresh() throws KeyStoreException, RuntimeException,
                   IOException, NoSuchAlgorithmException, CertificateException,
                   UnrecoverableKeyException, UnrecoverableEntryException {

             this.removeAllElements();

             SunMSCAPI providerMSCAPI = new SunMSCAPI();
             Security.addProvider(providerMSCAPI);
             KeyStore ks = KeyStore.getInstance("Windows-MY");
             ks.load(null,null);

             Enumeration<String> aliases = ks.aliases();
             while (aliases.hasMoreElements()) {
                String alias = aliases.nextElement();
                if (ks.isKeyEntry(alias)) {
                   PrivateKey privKey = (PrivateKey) ks.getKey(alias, null);
                   Certificate cert = ks.getCertificate(alias);
                   this.addElement(cert);
                   privateKeyList.add(privKey);
                }
             }

           }
}
```

Ejemplo 2: generación de firma digital

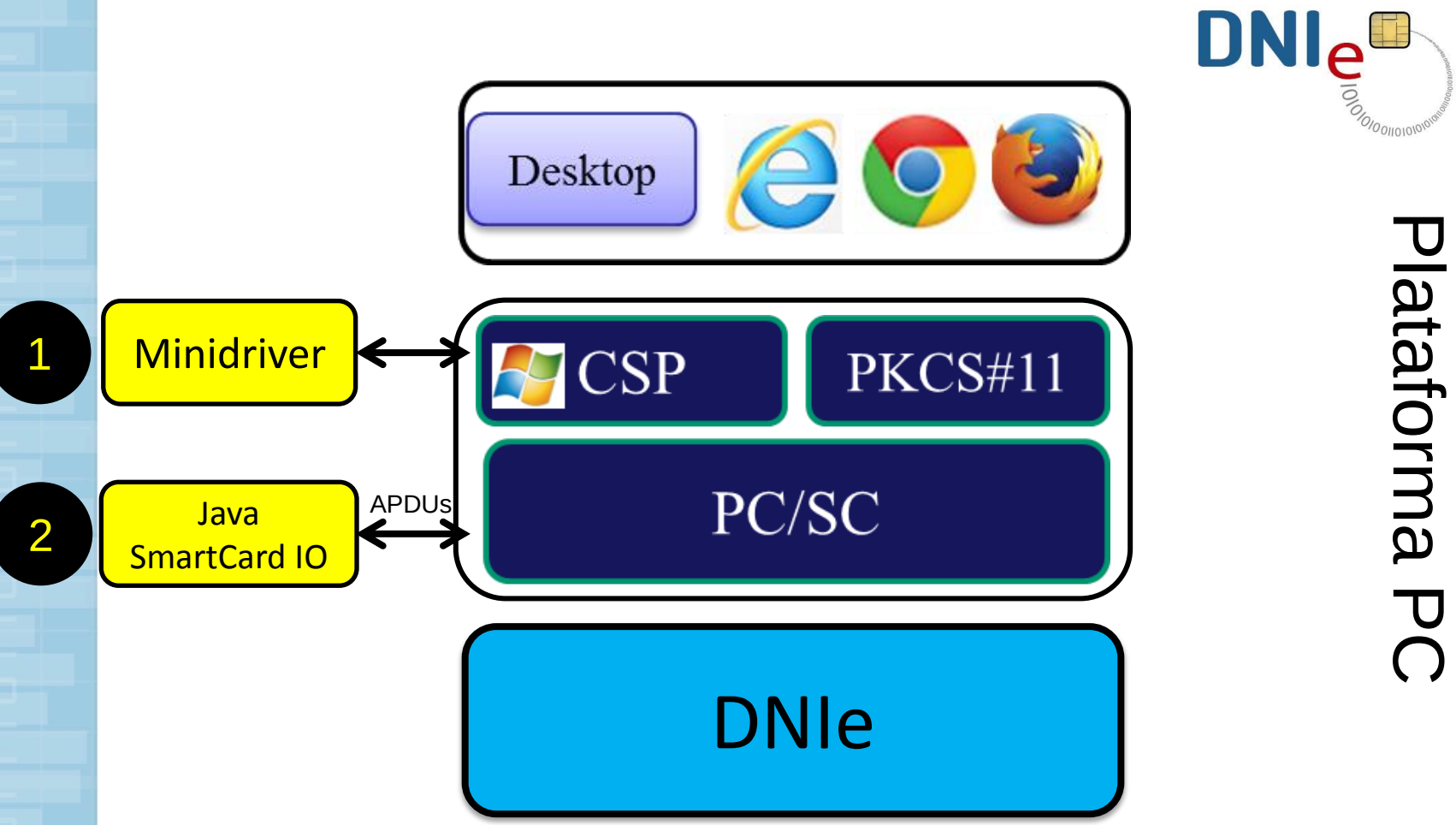
```
private void doClose(int retStatus) {
   returnStatus = retStatus;

   if (returnStatus == RET_OK) {
      try {
         if (selectedPrivateKey==null) return;
         if (dataTextField.getText().isEmpty()) return;

         Signature sig = Signature.getInstance("SHA1withRSA");
         sig.initSign(selectedPrivateKey);
         sig.update(dataTextField.getText().getBytes());

         byte[] signatureValue = sig.sign();
         String signatureString = bytArrayToHex(signatureValue);

         signatureTextArea.setText(signatureString);
         signInfoLabel.setText(signatureValue.length + " bytes, " +
                               signatureString.length() + " chars");

      } catch (NoSuchAlgorithmException ex) {
         Logger.getLogger(FirmaDialog.class.getName()).log(Level.SEVERE, null, ex);
      } catch (InvalidKeyException ex) {
         Logger.getLogger(FirmaDialog.class.getName()).log(Level.SEVERE, null, ex);
      } catch (SignatureException ex) {
         Logger.getLogger(FirmaDialog.class.getName()).log(Level.SEVERE, null, ex);
      }
   } else { // cerrar
      setVisible(false);
      dispose();
   }
}
```

Ejemplo 3: verificación firma digital



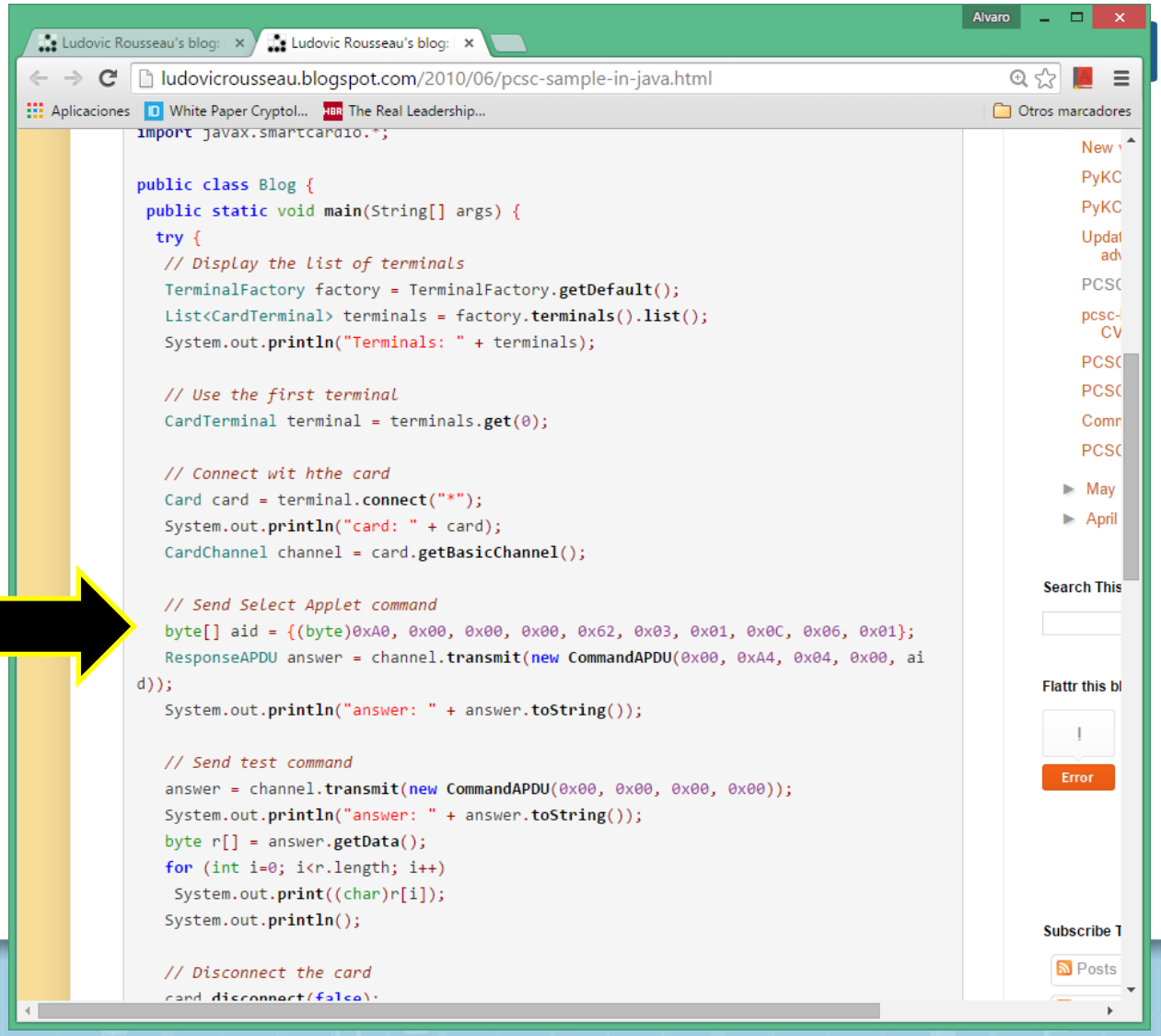
DNI Electrónico

Accediendo a un DNIe



DNI Electrónico

Java SmartCard IO



DNI Electrónico

Java SmartCard IO

DNI Electrónico

Identificadores

Identificador de la tarjeta

La tarjeta responde con el siguiente identificador:

ATR	3BDD18008131FE4580F9A0000000770100700A90008Bh	22 bytes

ATR del smart card – DNIe

Identificador de aplicación PKI

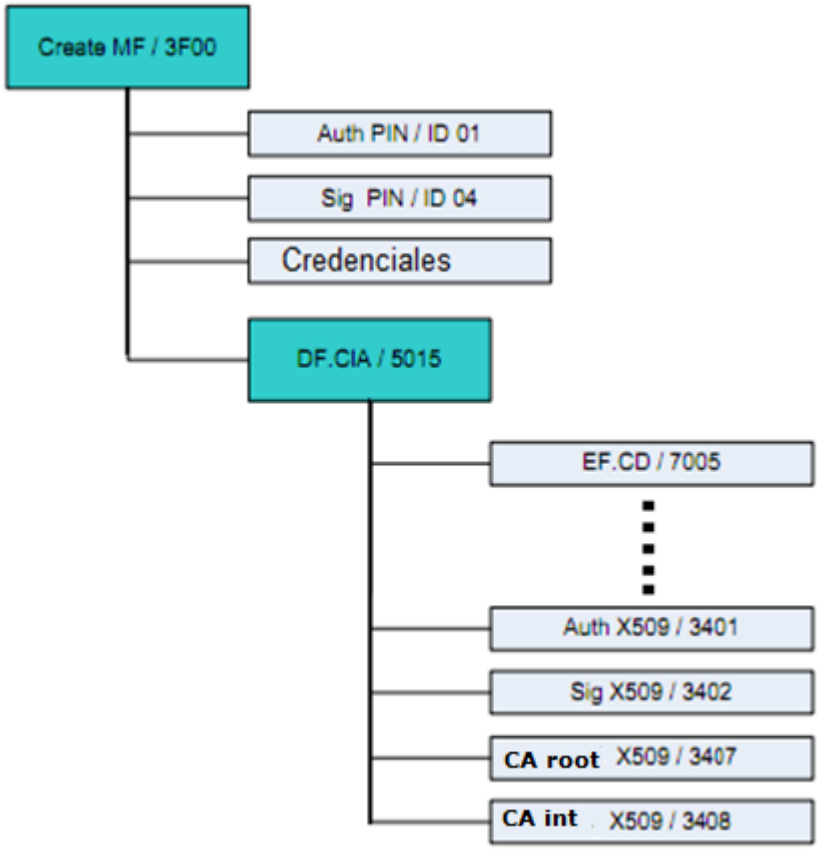
El contexto PKI viene contenido en un paquete con el siguiente identificador de instancia:

AID	A0000000770100700A1000F100000100h	16 bytes

AID de la instancia del contexto PKI del DNIe

DNI Electrónico

Estructura lógica PKI



- Master File (MF) Dirección 3F00
- Elementary File (EF) de la estructura PKI : 5015

CERTIFICADOS DIGITALES

- Certificado Autenticación: EF-3401
- Certificado Firma: EF-3402.
- Certificado CA : EF-3407
- Certificado CA intermedia: EF-3408
- Registro ABI: EF-FD01

- Descripción de los certificados contenidos en el token (ISO/IEC 7816-15) : EF-70050

Conformada por archivos y directorios: Master file, Directory file, Elementary file

DNI Electrónico

Operaciones

1. Obtención de certificados
2. Firmar hash
3. Creación de canal seguro
4. Verificación biométrica
5. Obtención del ABI

Comandos APDU

La estructura general para un comando APDU es:

Cabecera (4 bytes)				Campos opcionales		
CLA	INS	P1	P2	Lc	Data	Le

Estructura de un comando APDU

La cabecera describe el comando que la aplicación ejecuta. Los primeros cuatro bytes del comando APDU representan la cabecera, donde:

CLA	Representa la clase, indicando si el comando es un mensaje conforme al ISO 7816-4
INS	Indica la instrucción
P1, P2	Indica parámetros adicionales

Cabecera de un APDU

Los otros campos del comando APDU son opcionales (lo que significa que ellos pueden estar ausentes en algunos casos). Estos campos definen datos adicionales suministrados con el comando, los cuales son:

Lc	Indica la longitud del campo DATA
DATA	Indica los datos presentes en el comando
Le	Indica la longitud de la respuesta esperada

Campos opcionales de un APDU

CLA	INS	P1	P2

Formato APDU Nº 1

CLA	INS	P1	P2	Le

Formato APDU Nº 2

CLA	INS	P1	P2	Lc	DATA

Formato APDU Nº 3

CLA	INS	P1	P2	Lc	DATA	Le

Formato APDU Nº 4

DNI Electrónico

Comandos APDU

1. Comando Seleccionar Contexto
2. Comando Seleccionar Archivo
3. Comando de Lectura Binaria
4. Comando Verificar PIN
5. Comando Verificación Biométrica
6. Comando MSE – set
7. Comando PSO
8. Generar Par de Claves SMA
9. Obtener desafío
10. Autenticación General

DNI Electrónico

Obtener certificado de autenticación

1. Iniciar contexto
2. Seleccionar el master file: MF-3F00
3. Seleccionar el EF-5015 (contenedor de la estructura PKI del DNIe)
4. Seleccionar el EF-3401.
5. Obtener tamaño del certificado de autenticación (EF-3401).
6. Lectura del EF-3401 (según el tamaño del certificado).

Input (p.e Certificado de 1736 bytes)	Output (SW1 SW2)
1.- APDU: 00 A4 04 00 10 A0 00 00 00 77 01 00 70 0A 10 00 F1 00 00 01 00	1.- 90 00
2.- APDU: 00 A4 00 00 02 3F 00	2.- 90 00
3.- APDU: 00 A4 00 00 02 50 15	3.- 90 00
4.- APDU: 00 A4 02 00 02 34 01	4.- 90 00
5.- APDU: 00 B0 00 00 00 //lectura primeros 256 bytes	5.- 90 00
00 B0 01 00 00 //lectura del 256 al 512 bytes	90 00
…	…
00 B0 06 00 C8 //lectura del 1536 al 1224 bytes	90 00

DNI Electrónico

Cifrar hash con clave privada de firma

1. Iniciar contexto
2. Verificar PIN de Firma (VERIFY PIN).
3. Establecer el MANAGE SECURITY ENVIRONMENT (MSE), para el credencial de Firma.
4. Ejecutar el PERFORM SECURITY OPERATION (PSO), con el credencial de Firma y los Bytes que desean firmar ("hash").
5. Recuperar los bytes firmados.

Input	Output (SW1 SW2)
1.- APDU: 00 A4 04 00 10 A0 00 00 00 77 01 00 70 0A 10 00 F1 00 00 01 00	1.- 90 00
2.- PIN Autenticación: 00 20 00 01 08 34 33 32 31 FF FF FF FF	2.- 90 00
3.- MSE Autenticación: 00 22 41 B6 06 80 01 11 83 01 01	3.- 90 00
4.- PSO: 00 2A 9E 9A 21 301F300706052B0E03021A041434A995E1C275A36FB184E4CD9664F1DA56878681	4.- 90 00
5.- Recuperar bytes firmados (ResponseAPDU.getData() en Java)	5.- 90 00
DONDE: PIN Firma = 4321 Bytes a firmar (hash 33 bytes): 301F300706052B0E03021A041434A995E1C275A36FB184E4CD9664F1DA56878681	

PIN

Longitud HEX, de los bytes a firmar

Bytes a Firmar con el credencial de Autenticación

Estructura del registro ABI

Las respuestas siempre son requeridas, aún si ellas no contienen datos. La estructura de una respuesta APDU es:

Data	SW1	SW2

Formato APDU Nº 5

La combinación (SW1, SW2), comúnmente llamada *status word,* siempre está presente en la respuesta.

Cada combinación tiene una interpretación acorde al último comando APDU procesado.

Estructura del registro ABI

El campo Data del APDU respuesta a la lectura del EF *FD01* contiene los datos del registro ABI del DNIe.

La cadena hexadecimal obtenida en el campo Data del APDU respuesta está organizada en objetos TLV según la siguiente estructura:

Campo Data:

T	L	V	
78h	Longitud total (variable)	TLV_A	TLV_B

TLV del Campo Data de la respuesta a la lectura del registro ABI

El TLV_A tiene los siguientes campos:

T_A	L_A	V_A
5Ch	18h	V_A

Primera trama TLV contenida en el campo Valor de la respuesta a la lectura del registro ABI

El campo TLV_B es la concatenación de los siguientes TLVs listados a continuación:

T_B	L_B	V_B
5F60h	08h	Valor CUI
5F61h	01h	Valor Dígito de verificación
5F62h	Longitud Primer Apellido	Valor Primer Apellido
5F63h	Longitud Segundo Apellido	Valor Segundo Apellido
5F64h	Longitud Pre Nombres	Valor Pre Nombres
5F6Ah	01h	Valor Género (46h o 4Fh)
5F21h	06h	Valor Ubigeo
5F22h	06h	Valor Grupo de Votación
5F23h	04h	Valor
5F7Dh	00h	-
5F7Bh	00h	-
5F7Ch	04h	Valor

Objetos TLV contenidos en la segunda trama del campo Valor de la respuesta a la lectura del registro ABI

Muchas gracias