



Especificación Técnica: PKI y ABI

DNI Electrónico

Identificación segura



Alvaro Cuno
GCRD -SGCD
22/07/2015

DNI Electrónico



Funcionalidades

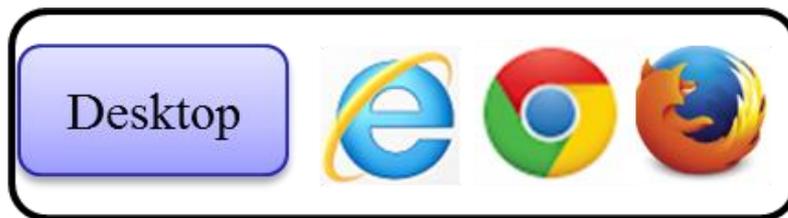


(en formato ICAO)

DNI Electrónico



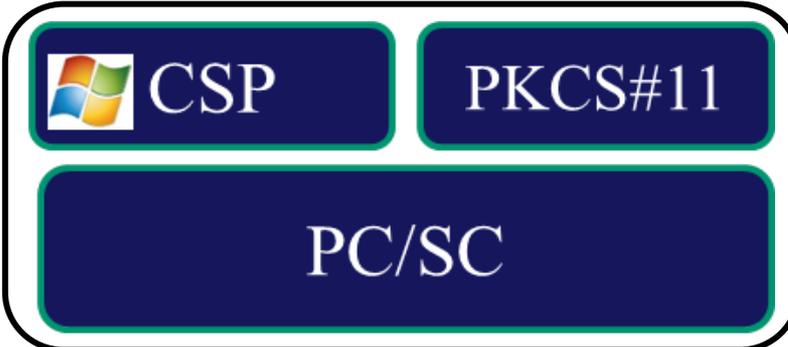
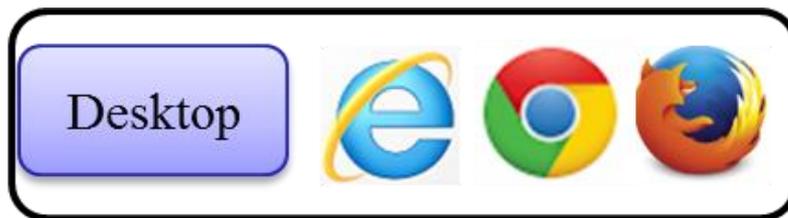
Accediendo a un DNle



DNI Electrónico



Accediendo a un DNle



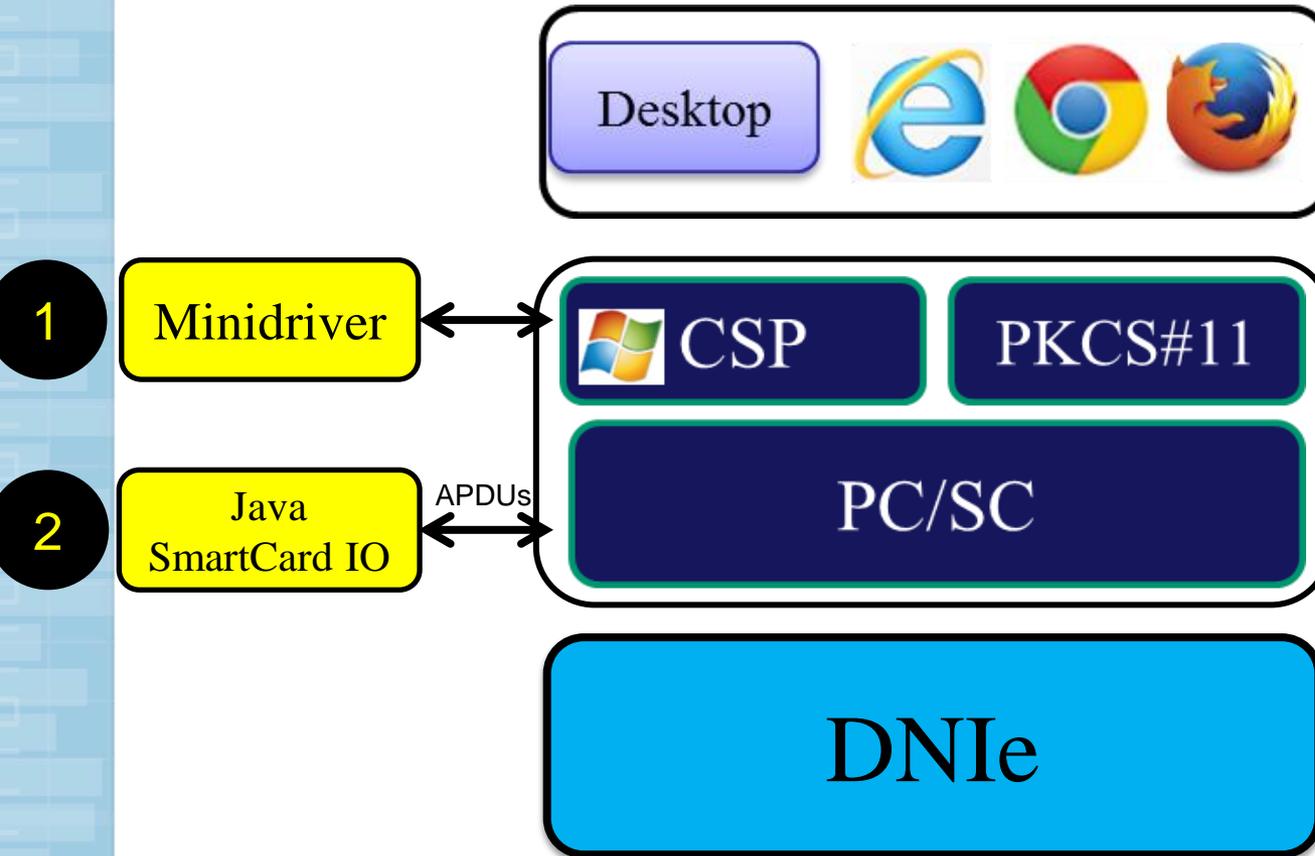
Plataforma PC

DNI Electrónico

Accediendo a un DNle



Plataforma PC



DNI Electrónico



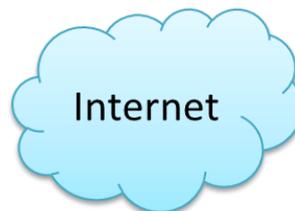
Minidriver



DNI Electrónico



Minidriver



Lector convencional de tarjetas de contactos (ISO 7816)



JC Provider

App Java Desktop



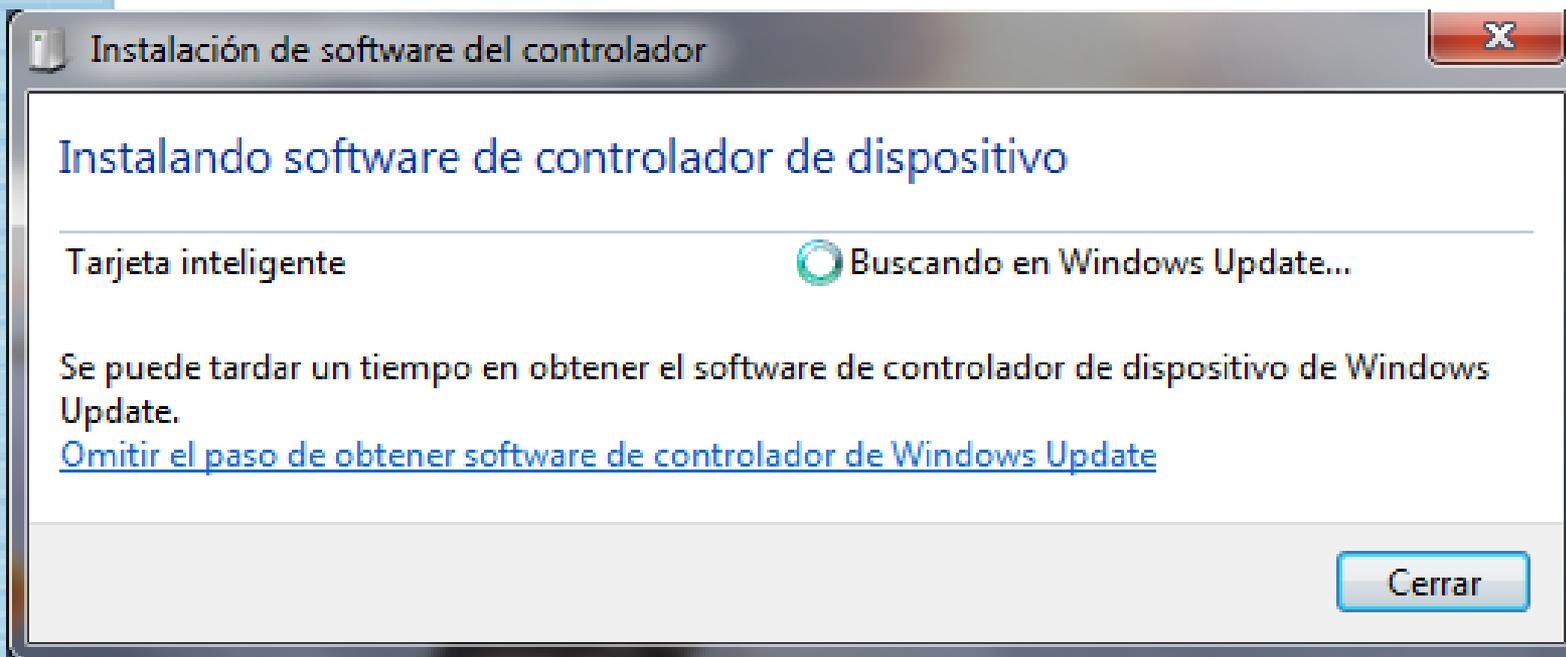
Minidriver

SO Windows

DNI Electrónico



Minidriver



DNI Electrónico



Minidriver

Instalación de software del controlador

Instalando software de controlador de dispositivo

Tarjeta inteligente Buscando en Windows Update...

Se pu
Upda
Omit

Instalación de software del controlador

No se pudo instalar el software de controlador de dispositivo

Póngase en contacto con el fabricante del dispositivo para obtener ayuda sobre la instalación de este dispositivo.

Tarjeta inteligente **X** No se encontró ningún controlador.

[¿Qué hago si el dispositivo no se instaló correctamente?](#)

Cerrar

Habilitar acceso a internet o el Windows update para el usuario¹⁰

DNI Electrónico



Minidriver

Instalación de software del controlador

Instalando software de controlador de dispositivo

Tarjeta inteligente Buscando en Windows Update...

Instalación de software del controlador

No se pudo instalar el software de controlador de dispositivo

Instalación de software del controlador

Oberthur Technologies Minidriver for AuthentIC V3 Smart Card instalado

Oberthur Technologies Minidriver for AuthentIC V3 Smart Card Listo para usar

Cerrar

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

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



Java cryptographic providers

SunMSCAPI (Java 8)



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

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



Certificados

Propósito planteado: <Todos>

Personal Otras personas Entidades de certificación intermedias Entidades de certificación

Emitido para	Emitido por	Fecha de expira
CUNO PARARI Alvaro Ernesto (FAU20295613...	RENIEC Class III CA	29/04/2015
CUNO PARARI Alvaro Ernesto (FAU20295613...	RENIEC Class III CA	18/11/2015

Importar... Exportar... Quitar Opciones avanzadas

Propósitos planteados del certificado

Autenticación del cliente, Correo seguro

Ver

Cerrar

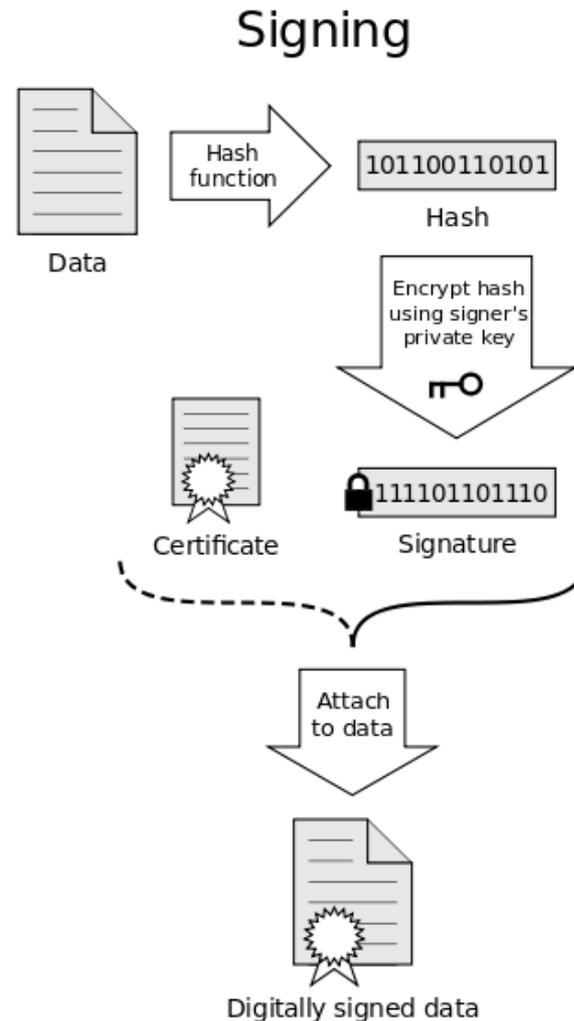
Ejemplo 1: iterando el keystore Windows-MY

```
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



DNI Electrónico



Ejemplo 2: generación de firma digital

Ejemplo 2 DnieJava

Certificados:

- ENCINAS ZEVALLOS Maria Paula (FIR45021409)
- CIFRADO
- CUNO PARARI alvaro ernesto (AUT29692413)
- Firma Digital
- Persona not Validated
- PEÑA DÜEÑAS Ciro (CIF00000007)

data:

firma digital:



DNI Electrónico



Ejemplo 2: generación de firma digital

The screenshot shows two overlapping windows. The background window, titled "Ejemplo 2 DnieJava", displays a list of certificates under the heading "Certificados:". The list includes:

- ENCINAS ZEVALLOS Maria Paula (FIR45021409)
- CIFRADO
- CUNO PARARI alvaro ernesto (AUT29692413)
- Firma Digital
- Persona not Validated
- PEÑA DÜEÑAS Ciro (CIF00000000)

Below the list, there is a text field labeled "data:" containing the text "holas a todos". At the bottom of the window, there is a label "firma digital:" followed by a large empty rectangular box.

The foreground window is a Windows Security dialog box titled "Seguridad de Windows". It contains the following text:

- Tarjeta inteligente**
- Escriba su PIN de firma digital.

Below this text is a light blue box containing:

- A smart card icon.
- The label "PIN" above a text input field with six dots.
- A link labeled "Más información".

At the bottom of the dialog box are two buttons: "Aceptar" and "Cancelar". At the bottom of the Java window, there is a "Cerrar" button.

DNI Electrónico



Ejemplo 2: generación de firma digital

Ejemplo 2 DnieJava

Certificados:

- ENCINAS ZEVALLOS Maria Paula (FIR45021409)
- CIFRADO
- CUNO PARARI alvaro ernesto (AUT29692413)
- Firma Digital
- Persona not Validated
- PÉÑA DUEÑAS Ciro (CIF00000007)

data:

firma digital: 256 bytes, 512 chars

```
AAFF6507649D599E7C56D979BB4242AD0B221DDC5C59BAF843CD0
5507AE47F794614B0DF5C06672FEBC3236EED2A61F368CBF71143
057A3F40C83408F129A8E7DEB13DDE6E09BD541B83196791352CA
46296DD0B607E8AC302C2B90AE9BB89BEE8D28102D1A7F97B8049
6B02677DCD3B50A423FFD2E6C7BE485D3DB63FEA643BDDC67B7E5
F75ECD23890070532194BB5C9FB5B0BEB7F13255BEE84DCB8CF7D
169234E9CB32E05C6BD36F61D7CCBB8475E2AED0DBA0513E739E0
33BDF5966EA650AA84C93277D46ACD77B39796C126533DB942F53
7AD0BE85DCA8E11EF99161A7E74CE2BE73A2A1C8C33EE29EBE0D8
57ACAA9BFF50AF8E74A34C20C5FDA571BF4
```

Seguridad de Windows

ente
firma digital.

PIN

.....

[Más información](#)

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 = byteArrayToHex(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();
    }
}
```

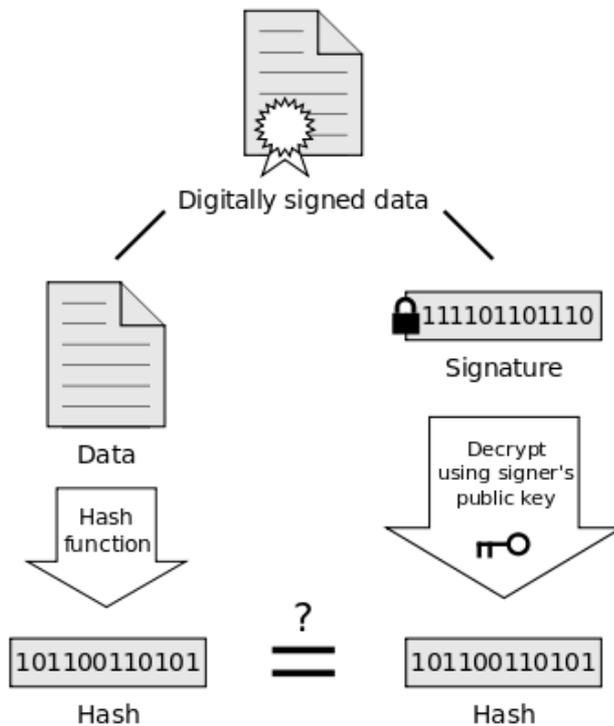
DNI



Ejemplo 3: verificación firma digital



Verification



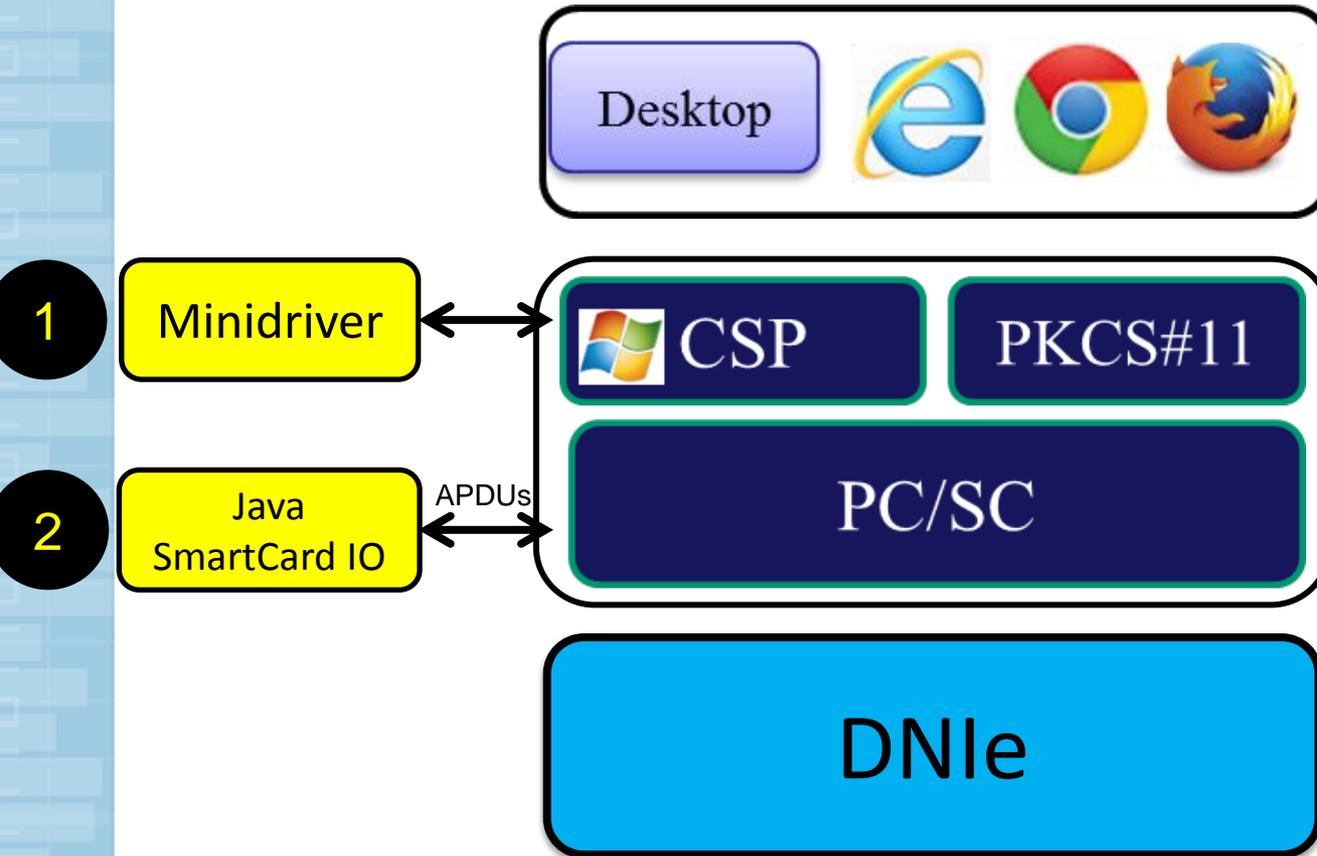
If the hashes are equal, the signature is valid.

DNI Electrónico

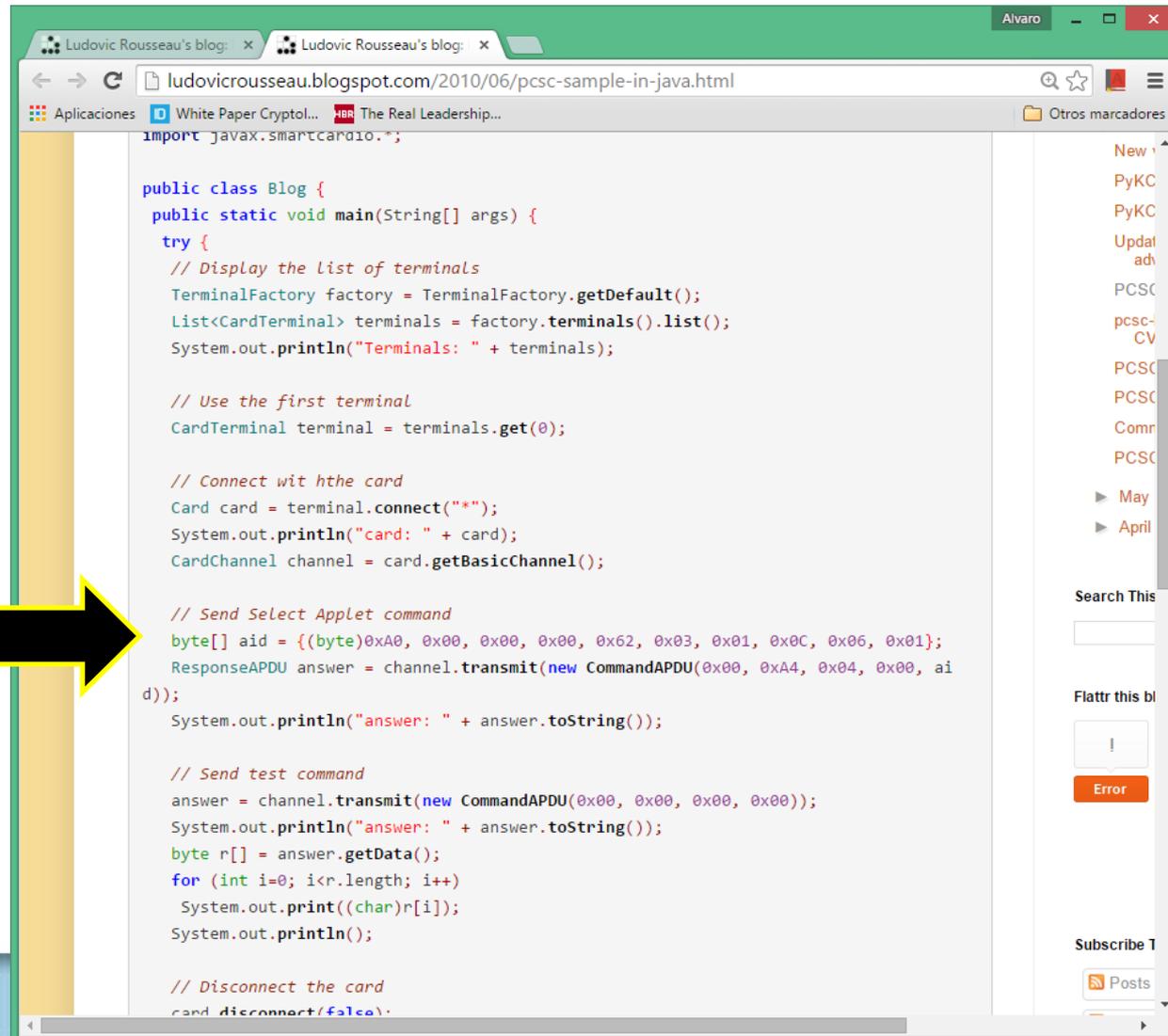
Accediendo a un DNle



Plataforma PC



Java SmartCard IO



```
import javax.smartcardio.*;

public class Blog {
    public static void main(String[] args) {
        try {
            // Display the List of terminals
            TerminalFactory factory = TerminalFactory.getDefault();
            List<CardTerminal> terminals = factory.terminals().list();
            System.out.println("Terminals: " + terminals);

            // Use the first terminal
            CardTerminal terminal = terminals.get(0);

            // Connect with the card
            Card card = terminal.connect("");
            System.out.println("card: " + card);
            CardChannel channel = card.getBasicChannel();

            // Send Select Applet command
            byte[] aid = {(byte)0xA0, 0x00, 0x00, 0x00, 0x62, 0x03, 0x01, 0x0C, 0x06, 0x01};
            ResponseAPDU answer = channel.transmit(new CommandAPDU(0x00, 0xA4, 0x04, 0x00, aid));
            System.out.println("answer: " + answer.toString());

            // Send test command
            answer = channel.transmit(new CommandAPDU(0x00, 0x00, 0x00, 0x00));
            System.out.println("answer: " + answer.toString());
            byte r[] = answer.getData();
            for (int i=0; i<r.length; i++)
                System.out.print((char)r[i]);
            System.out.println();

            // Disconnect the card
            card.disconnect(false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



Java SmartCard IO



The screenshot shows a web browser window displaying the blog post "PC/SC sample in different languages" by Ludovic Rousseau. The browser's address bar shows the URL: `ludovicrousseau.blogspot.com/2010/04/pcsc-sample-in-different-languages.html`. The page title is "Ludovic Rousseau's blog" with the subtitle "My activities related to smart card and Free Software (as in free speech)". The post date is "Thursday, April 8, 2010". The main heading is "PC/SC sample in different languages". The text of the post states: "The PC/SC API also called WinSCard can be used from a large variety of languages. I will start a serie of blogs to present the same program (functionnally) but using different languages." Below this text is a section titled "Languages:" which is highlighted with a blue border. This section contains a list of programming languages. To the right of the main content is a "Blog Archive" sidebar showing a list of years and months with post counts.

Thursday, April 8, 2010

PC/SC sample in different languages

The PC/SC API also called WinSCard can be used from a large variety of languages. I will start a serie of blogs to present the same program (functionnally) but using different languages.

Languages:

- C
- Perl
- Python
- scriptor
- OCaml
- Prolog
- Ruby
- Java
- C#
- Ada
- PHP (dead upstream as 11 January 2015)
- PHP5
- lua
- JavaScript (Node.js)
- Python (using python-pcsc-lite)
- Common Lisp
- C for UEFI

If you know a wrapper for a language not listed above please contact me.

Blog Archive

- ▶ 2015 (25)
- ▶ 2014 (61)
- ▶ 2013 (38)
- ▶ 2012 (27)
- ▶ 2011 (46)
- ▼ 2010 (55)
 - ▶ December (5)
 - ▶ November (5)
 - ▶ October (9)
 - ▶ September (1)
 - ▶ August (8)
 - ▶ July (1)
 - ▶ June (10)
 - ▶ May (6)
 - ▼ April (10)
 - PCSC sample in Pyth...
 - Free software Token...
 - PKCS#11 (for Mac...
 - Source code of PKCS...
 - .NET cards
 - PCSC sample in Perl...
 - My Ohloh page

Identificadores



Identificador de la tarjeta

La tarjeta responde con el siguiente identificador:

ATR	3BDD18008131FE4580F9A0000000770100700A90008Bh	22 bytes
-----	---	----------

ATR del smart card – DNIe

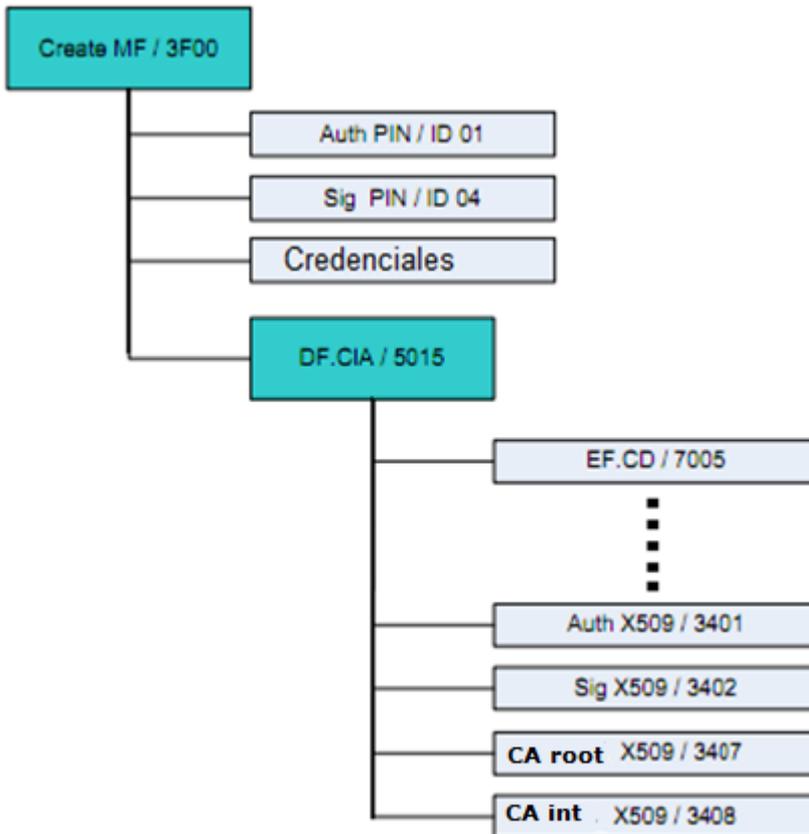
Identificador de aplicación PKI

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

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

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 N° 1

CLA	INS	P1	P2	Le
-----	-----	----	----	----

Formato APDU N° 2

CLA	INS	P1	P2	Lc	DATA
-----	-----	----	----	----	------

Formato APDU N° 3

CLA	INS	P1	P2	Lc	DATA	Le
-----	-----	----	----	----	------	----

Formato APDU N° 4

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 DNle)
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 ← PIN	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 ← Bytes a Firmar con el credencial de Autenticación	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

Longitud HEX, de los bytes a firmar

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 N° 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

