Applet de realización de firma digital multiformato

Univeristat Jaume I
http://cryptoapplet.nisu.org/

Versión del documento:	1.2
Estado del documento:	Borrador.
Última Actualización:	22 de febrero de 2007
Tipo:	Externo, documentación de usuario.

Índice

1. Resumen

El applet de firma de la Univesitat Jaume I es un applet desarrollado en Java para la realización desde un navegador de firmas digitales en diferentes formatos, siendo éste capaz de emplear directamente los certificados accesibles desde el CryptoAPI de Windows (repositorio de Windows, "clauer"), o vía PKCS11, accediendo directamente al repositorio de Mozilla Firefox o al "clauer", ofreciendo una gestión transparente para el usuario.

2. Objetivos

El objeto principal de éste applet es el de realizar el proceso de firma electrónica avanzada de forma transparente e integrada tanto para el usuario como para el programador del sitio web que solicita la realización de la firma. El applet obtiene una entrada de datos determinada y ofrece como salida un objeto que representa la firma digital en los siguientes formatos:

- **Firma en bruto:** En este caso el applet utiliza el algoritmo RSA para firmar los datos que se le facilitan como entrada, sobre estos datos se realiza la operación $r = (sha1(msg))^d \bmod n$ ofreciendo como salida el resultado r, siendo msg los datos de entrada, d el exponente privado y n el módulo de la clave.

- **Firma en formato PKCS#7/CMS:** En este caso, el applet obtiene el resumen SHA-1 de los datos a firmar y realiza el proceso de firma ofreciendo como salida un PKCS#7 que representa la misma.

- **Firma en formato DigiDoc (XAdES-X-L):** En este caso, el applet obtiene los datos y ofrece una firma digital en formato DigiDoc (XAdES-X-L).

- **Firma de pdf:** En este caso, el applet recibe un pdf como entrada y ofrece un pdf firmado cuya firma digital es interpretada y validada correctamente por Adobe Reader.

El applet también es capaz de reconocer el navegador que lo está ejecutando obteniendo así los certificados de firma desde el almacén de certificados correspondiente. Los navegadores soportados son:

- Internet Explorer 6 y 7 obteniendo los certificados del CryptoApi.

- Navegadores de la familia Mozilla obteniendo los certificados de su almacen nativo mediante PKCS#11.

En ambos navegadores se permite utilizar certificados almacenados en el clauer[1].
Los sistemas operativos soportados son GNU/Linux y Windows XP.

[1] Véase http://clauer.uji.es

3. Comienzo rápido

Obtén la última versión del applet de http://cryptoapplet.nisu.org/. En esta dirección puedes encontrar las bibliotecas asociadas a cada funcionalidad del applet bajo el directorio /lib en /doc puedes encontrar éste documento y en /samples puedes encontrar algunos ejemplos de uso. En la raíz también encontrarás el fichero jar con el programa principal y un tar.gz con todos los ficheros jar del proyecto.

3.1. Instalación del applet en el servidor

Para instalar el applet en el servidor basta con crear un directorio accesible por el servidor web y copiar allí los ficheros *.jar de los que se compone el applet, el programa principal (el init del applet) se encuentra en CryptoApplet.jar. Para incluir el applet en la página web puedes utilizar las siguientes etiquetas:

- < Applet > (Recomendada por Sun por compatibilidad)
- < Object > (Sólo para Internet Explorer)
- < Embed > (Sólo para navegadores de la familia Mozilla)

A continuación se expondrá como utilizar la etiqueta Applet para instanciar el applet, suponiendo que los ficheros *.jar se encuentran en http://example.com/aps/stable, el código seria el siguiente:

```
<applet id="CryptoApplet" name="CryptoApplet"
        code="es.uji.dsign.applet.SignatureApplet"
        width="0" height="0" codebase="aps/stable"
        archive="CryptoApplet.jar,lbc.jar,jakarta-log4j-1.2.6.jar"
        mayscript>

   <param name="signFormat"
          value="es.uji.dsign.crypto.CMSSignatureFactory">

   <param name="signInputParams"
          value="es.uji.dsign.io.AppletInputNullParams"/>

   <param name="signOutputParams"
          value="es.uji.dsign.io.AppletOutputParams"/>

   <param name="progressBar" value="true"/>

</applet>
```

La primera línea contiene identificador, nombre, código que contiene la inicialización del applet, tamaño, directorio base del código en el servidor, que archivos jar se deben incluir

en la ejecución del applet y la etiqueta mayscript que indica que el applet puede interaccionar con la página mediante funciones JavaScript respectivamente.

A continuación, en el cuerpo de la etiqueta applet, encontramos los parámetros, definidos con más detalle en el punto 4.1. en este caso la configuración de parámetros que se ha utilizado indica que el tipo de salida es un objeto PKCS#7 (parámetro[2] *es.uji.dsign.crypto.CMSSignatureFactory*), que el hash de entrada para la firma se obtendrá como parámetro en la invocación de la función de firma (parámetro *AppletInputNullParams*). El siguiente parámetro indica dónde va la salida, en este caso, el valor *es.uji.dsign.io.AppletOutputParams* indica que debe pasar el resultado como parámetro a una función JavaScript definida en el HTML de la página y por último el parámetro progressBar indica que se muestre al usuario algún tipo de progreso durante la carga de certificados y la realización de la firma.

Para que el applet comience el proceso de firma, debemos invocar la función:

```
document.CryptoApplet.sign(hash);
```

El resultado se devuelve como parámetro de la invocación a la función JavaScript onSignOk que debe estar definida en el HTML de la página. Un ejemplo podría ser el siguiente:

```
function onSignOk(signature){
    document.formu.firma.value=signature;
    document.formu.submit();
}
```

En el directorio indicado por el atributo *codebase* debe contener, además de los ficheros jar indicados en el atributo *archive*, la dll *MicrosoftCryptoApi_0_3.dll*.

3.2. Instalación del applet en el cliente

El Applet se instala de forma transparente en el cliente cuando éste se instancia, el applet crea dos ficheros temporales en el directorio temporal del usuario, uno denominado MicrosoftCryptoApi_0_3.dll en el caso de Internet Explorer y uno denominado p11.cfg.1166440118 en caso de que el cliente utilize Mozilla.

3.3. Firma en bruto

Para realizar éste tipo de firma, debemos utilizar al menos los siguientes parámetros.

```
<param name="signFormat"
         value="es.uji.dsign.crypto.RawSignatureFactory"/>
<param name="signInputParams"
         value="es.uji.dsign.io.AppletInputNullParams"/>
<param name="signOutputParams"
         value="es.uji.dsign.io.AppletOutputParams"/>
```

[2]Para más información con respecto a los parámetros posibles, ver sección 4.1 más adelante.

para realizar la firma invocaremos, como anteriormente:

```
document.CryptoApplet.sign(datos);
```

Y el resultado de la misma será nuevamente pasado como parámetro a la función JavaScript onSignOk que debe estar definida.

3.4. Firma en formato PKCS#7/CMS

Para realizar la firma en este formato debemos utilizar la configuración expuesta en el punto 3.1.

El procedimiento de firma del applet que obtiene este formato de representación de firma es especialmente interesante, su interés reside en el hecho de que para realizar la firma de unos datos, el applet sólo requiere el resumen SHA-1 de los datos, de esta forma un cliente puede firmar datos con gran tamaño tranfiriendo únicamente 20 bytes por la red y éste puede auditar posteriormente la firma realizada si se le ofrece el objeto PKCS#7. Un escenario típico de esto es un webmail[3] , en el que se utiliza S/MIME para la encapsulación de firmas digitales donde un correo electrónico puede contener varios adjuntos que hagan que éste tenga un tamaño considerable, el proceso de firma en este caso sería el siguiente:

1. El cliente compone el mensaje con sus adjuntos y lo envía al servidor.
2. El servidor compone los datos que forman la firma, los resume con una función hash y devuelve los 20 bytes resultantes al cliente para ser firmados.
3. El cliente devuelve al servidor el objeto PKCS#7.
4. El servidor compone el mail S/MIME con la firma y lo envía.

3.5. Firma en formato DigiDoc (XAdES-X-L)

Para realizar una firma en este formato, en primer lugar debemos incluir todas las bibliotecas necesarias en el atributo archive de la etiqueta applet, éstas son:

```
archive="CryptoApplet.jar,bcprov-jdk14-133.jar,bcmail-jdk14-133.jar,
         jakarta-log4j-1.2.6.jar,bctsp-jdk14-133.jar,xmlsec.jar,
         myxmlsec.jar,xalan.jar,iaik_jce.jar,iaikPkcs11Wrapper.jar"
```

Además de esto, debemos utilizar los siguientes parámetros del applet:

```
<param name="signFormat" value="es.uji.dsign.crypto.XAdESSignatureFactory">
```

En cuanto a los parámetros relativos a la entrada de datos en el applet, es aconsejable, dado que normalmente los datos a firmar suelen ser extensos, utilizar los siguientes:

[3]Puede verse un ejemplo simplificado en http://cryptoapplet.nisu.org/applet/samples/sigmail.php

```
<param name="signInputParams" value="es.uji.dsign.io.URLInputParams"/>
<param name="signOutputParams" value="es.uji.dsign.io.URLOutputParams"/>
<param name="urlInput" value="http://example.com/datos"/>
<param name="urlOutput" value="https://example.com/resultado.php"/>
```

De esta forma, el applet obtendrá los datos a firmar de la URL indicada por urlInput enviando un POST con la variable content conteniendo el resultado de la firma ***codificado en base64*** a la URL indicada por urlOutput.

También deben incluirse algunos ficheros extra que se describen en el punto 4.2.

3.6. Firma de PDF

Para la realización de firma de pdf se aconseja utilizar los mismos parámetros que en el ejemplo anterior 3.5, a excepción de signFormat cuyo valor debe ser:

```
<param name="signFormat" value="es.uji.dsign.crypto.PDFSignatureFactory">
```

El proceso realizado por el applet con los parámetros recomendados sería análogo al expuesto en el punto anterior.

En la versión 1 del applet la firma de PDF permite únicamente firmar documentos con certificados expedidos por la ACCV[4].

La versión 2 del applet facilitará mediante un fichero de configuración la posibilidad de indicar otras autoridades de certificación con la que permitir la firma de documentos PDF.

En la sección de ejemplos puede verse uno para este formato de firma.

4. Configuración avanzada

4.1. Parámetros

El applet soporta los parámetros que mostramos a continuación, estos pueden combinarse para obtener distintos tipos de formato de firma y obtener o devolver los datos a firmar de maneras diferentes.

[4]Véase http://www.accv.es

Nombre	Posible valor / Descripción
"signFormat"	**Valor:** es.uji.dsign.crypto.RawSignatureFactory **Descripción:** Formato de la firma "en bruto" **Valor:** es.uji.dsign.crypto.CMSSignatureFactory **Descripción:** Formato de la firma en PKCS#7/CMS **Valor:** es.uji.dsign.crypto.PDFSignatureFactory **Descripción:** Formato de la firma PDF **Valor:** es.uji.dsign.crypto.XAdESSignatureFactory **Descripción:** Formato de la firma DigiDoc (XAdES-X-L).
"signInputParams"	**Valor:** es.uji.dsign.io.AppletInputNullParams **Descripción:** Los datos de entrada se obtienen de la función sign(datos); **Valor:** es.uji.dsign.io.AppletInputParams **Descripción:** El resumen de entrada de la firma se obtiene del parámetro signHash. **Valor:** es.uji.dsign.io.URLInputParams **Descripción:** Los datos de entrada para la firma se obtienen de la URL indicada por el parámetro urlInput.
"signOutputParams"	**Valor:** es.uji.dsign.io.AppletOutputParams **Descripción:** El resultado se pasa como argumento a la función JavaScript onSignOk definida en el cuerpo de la página. **Valor:** es.uji.dsign.io.URLOutputParams **Descripción:** El resultado se envia mediante POST al al URL indicada en el parámetro urlOutput. La variable que contiene el resultado es content.

Nombre	Posible valor / Descripción
"urlInput"	**Valor:** Una URL válida **Descripción:** Se tiene en cuenta sólo si se indica como signOutputParams el valor es.uji.dsign.io.URLInputParams y éste define la url desde la que obtener los datos.
"urlOutput"	**Valor:** Una URL válida **Descripción:** Se tiene en cuenta sólo si se indica como signOutputParams el valor es.uji.dsign.io.URLOutputParams y éste define la url a la que se debe hacer el POST con el resultado. Éste resultado se retorna codificado base64.
"signHash"	**Valor:** El hash a firmar en hexadecimal
"downloadUrl"	**Valor:** Una URL válida **Descripción:** Indica la dirección desde la que el applet debe obtener la dll MicrosoftCryptoApi_0_3.dll sino se indica se intenta obtener desde el directorio presente en el atributo codebase del tag applet.
"progressBar"	**Valor:** true/false **Descripción:** Muestra una barra de progreso indeterminado si esta a true.
"encoding"	**Valor:** hex **Descripción:** Indica que los datos de entrada se encuentran representados en hexadecimal como un string y deben ser descodificados antes de considerarlos para la firma. **Valor:** base64 **Descripción:** Indica que los datos de entrada se encuentran representados en base64 como un string y deben ser descodificados antes de considerarlos para la firma.

Tabla 1: Descripción de los parámetros

4.2. Uso del formato DigiDoc

Si se utiliza el formato DigiDoc para representar la firma digital, se debe incluir en CryptoApplet.jar un fichero de configuración de este formato, el fichero se compone de pares campo=valor y éstos son un subconjunto de los permitidos por las biblioteca que imple-

menta la funcionalidad DigiDoc [5]. A continuación pasamos a comentar éste fichero de configuración[6] :

```
# Configuración de las clases a utilizar.
DIGIDOC_SIGN_IMPL=ee.sk.digidoc.factory.PKCS11SignatureFactory
DIGIDOC_SIGN_IMPL_PKCS11=ee.sk.digidoc.factory.PKCS11SignatureFactory
DIGIDOC_NOTARY_IMPL=ee.sk.digidoc.factory.BouncyCastleNotaryFactory
DIGIDOC_FACTORY_IMPL=ee.sk.digidoc.factory.SAXDigiDocFactory
DIGIDOC_TIMESTAMP_IMPL=es.uji.dsign.digidoc.factory.BouncyCastleSignatureTimestampFac
CANONICALIZATION_FACTORY_IMPL=ee.sk.digidoc.factory.DOMCanonicalizationFactory
CRL_FACTORY_IMPL=ee.sk.digidoc.factory.CRLCheckerFactory
ENCRYPTED_DATA_PARSER_IMPL=ee.sk.xmlenc.factory.EncryptedDataSAXParser
ENCRYPTED_STREAM_PARSER_IMPL=ee.sk.xmlenc.factory.EncryptedStreamSAXParser

# Configuración del proveedor principal.
DIGIDOC_SECURITY_PROVIDER=org.bouncycastle.jce.provider.BouncyCastleProvider
DIGIDOC_SECURITY_PROVIDER_NAME=BC

# OCSP responder URL.
DIGIDOC_OCSP_RESPONDER_URL=http://ocsp.pki.gva.es

# Si se firma la petición OCSP.
SIGN_OCSP_REQUESTS=false

#Certificados con los cuales realizar una comprobación preliminar
#del firmante, si para el certificado firmante no podemos encontrar
#su cadena de expedición aquí, se generará un error.
DIGIDOC_CA_CERTS=2
DIGIDOC_CA_CERT1=jar://cagva.pem
DIGIDOC_CA_CERT2=jar://rootca.pem


# Certificados del responder OCSP.
DIGIDOC_OCSP_COUNT=1
DIGIDOC_OCSP1_CN=ocsp-gva
DIGIDOC_OCSP1_CERT=jar://ocsp-gva_pem.crt
DIGIDOC_OCSP1_CA_CERT=jar://cagva.pem
DIGIDOC_OCSP1_CA_CN=CAGVA

# OCSP or CRL selectors
# Debemos indicar siempre OCSP
DIGIDOC_CERT_VERIFIER=OCSP
DIGIDOC_SIGNATURE_VERIFIER=OCSP
```

[5] Véase http://openxades.org/files/JDigiDoc-2.0-eng.pdf

[6] Esta configuración es válida para firma con certificados de la ACCV http://www.accv.es

```
# Relativo al Timestamp
DIGIDOC_TSA_COUNT=1
DIGIDOC_TSA1_CERT=jar://tsa1_accv.der
DIGIDOC_TSA1_CA_CERT=jar://tsa1_accv.der
DIGIDOC_TSA1_USE_NONCE=true
DIGIDOC_TSA1_ASK_CERT=false
DIGIDOC_TSA1_URL=http://tss.accv.es:8318/tsa
DIGIDOC_TSA1_CN=TSA1 ACCV
DIGIDOC_TSA1_CA_CN=Root CA Generalitat Valenciana
DIGIDOC_TSA1_SN=1164131574
MAX_TSA_TIME_ERR_SECS=60
```

Todos los certificados que se referencian desde el fichero de configuración deben ser también incluidos en el fichero jar principal.

4.3. Firma del applet

Para que el applet se ejecute con los privilegios necesarios para realizar las operaciones relativas a la firma, éste debe estar firmado y el cliente debe depositar su confiaza en él. Para realizar la importación de la clave privada con la que se firmará el applet, se recomienda que se utilize el programa que se puede obtener de http://forum.java.sun.com/thread.jspa?forumID=60&tstart=30&threadID=484622&trange=15. Una vez importada la clave privada al Java KeyStore, firmar el jar con el siguiente comando:

$ jarsigner -keystore ruta_hasta_el_store -storepass password_del_store *.jar Alias_de_la_clave

Es conveniente que firmemos todas las bibliotecas *.jar para que el applet funcione correctamente, sin la firma, el applet no tendrá suficientes privilegios y fallará en su ejecución.

4.4. Entrada/Salida

La selección de la entrada/salida de datos en applet se realiza mediante parámetros, estos son: *signInputParams* y *signOutputParams* que se encuentran documentados en el punto anterior.

4.5. Idiomas

En el fichero jar principal del applet CryptoApplet.jar se encuentra un fichero denominado i18n.properties en el que se definen las cadenas que el applet debe mostrar en los distintos puntos del proceso de firma, estas cadenas vienen determinadas en el fichero como pares identificador=cadena, a continuación indicamos a que parte del proceso corresponde cada indicador:

```
# Applet Window
WINDOW_TITLE          Título de la ventana principal.
```

```
WINDOW_MESSAGE          Aviso introducir password Mozilla

# Widgets
LABEL_CERTIFICATE_SELECTION       Selección de certificado.
LABEL_VIEW_CERTIFICATES           Ver Certificados.
LABEL_SIGN                        Botón de firma.
LABEL_PIN                         Pedir pin del Clauer.
LABEL_PASSWORD                    Pedir password.
LABEL_TREE_ROOT                   Lista de certificados.

# Information
SIGN_PROCESS_OK                   Firma realizada correcta.
PROGRESS_VIEW_CERTIFICATES        Muestra de certificados en progreso.
PROGRESS_RELOADING_CERTIFICATES   Actualización lista de certificados
                                  en progreso.
PROGRESS_SIGN                     Aviso realización de firma en progreso.
PROGRESS_WINDOW_TITLE             Título de la ventana indicadora del
                                  progreso.


# Error
ERROR_OPEN_CERTIFICATE_STORE      No se puede acceder al almacén
                                  de certificados.
ERROR_ACCESS_CERTIFICATE_STORE    Error al acceder al almacén
                                  de certificados.
ERROR_RETRIEVE_CERTIFICATE_LIST   No se ha podido recuperar la
                                  lista de certificados.
ERROR_ACCESS_CLAUER               No se puede acceder al clauer.
ERROR_STORE_EMPTY                 No existen certificados en el
                                  almacén.
ERROR_CERTIFICATE_USE             Sólo puedes utilizar certificados
                                   con funciones de firma.
ERROR_CERTIFICATE_NOT_SELECTED    Es obligatorio seleccionar un
                                  certificado de la lista.
ERROR_COMPUTING_SIGNATURE         No se ha podido calcular la firma.
ERROR_ACCESS_NO_STORES            No se ha encontrado ningún almacén
                                  de certificados válido en tu ordenador.

# Secciones de errores por formato de salida.

# CMS Errors
ERROR_CMS_NOCERT=No se pudo recuperar el certificado.
ERROR_CMS_NOKEY=No se pudo recuperar la referencia a la clave privada.
ERROR_CMS_SIGNATURE=No se pudo calcular la firma.

# RAW Errors
```

```
ERR_RAW_NOKEY=No se pudo recuperar la referencia a la clave privada.
ERROR_RAW_SIGNATURE=No se pudo calcular la firma.

# DigiDoc Errors
ERROR_DDOC_NOCERT=No se pudo recuperar el certificado.
ERROR_DDOC_NOKEY=No se pudo recuperar la referencia a la clave privada.
ERROR_DDOC_NODIGEST=No se pudo calcular el resumen.
ERROR_DDOC_SIGNATURE=No se pudo calcular la firma.
ERROR_DDOC_TSDIGEST=No se pudo calcular la marca de tiempo.
ERROR_DDOC_TSATIMEOUT=la conexión expiró obteniendo la marca de timepo.
ERROR_DDOC_TSARESPONSE=Se obtuvo una marca de tiempo inválida.
ERROR_DDOC_TSACA=No se pudo obtener el certificado raíz de la autoridad
                 de certificación.
ERROR_DDOC_CERTREVOKED=El certificado con el que intenta firmar está revocado.
ERROR_DDOC_CERTEXPIRED=El certificado con el que intenta firmar ha expirado.
ERROR_DDOC_CACERTREAD=No se pudo comprobar la validez de su certificado.
ERROR_DDOC_CERTGENERIC=No se pudo comprobar la validez de su certificado.
```

Para traducir el applet a otro idioma, basta con sustituir estas cadenas en el fichero y volver a empaquetar el fichreo .jar.

5. Ejemplos de uso

- Firma de datos en bruto.
  Ver: http://cryptoapplet.nisu.org/samples/signraw.php
- Firma de un hash en formato CMS para el envio de mail.
  Ver: http://cryptoapplet.nisu.org/samples/signmail.php
- Firma de datos en formato digidoc (XAdES-X-L).
  Ver: http://cryptoapplet.nisu.org/samples/signxades.php
- Firma de datos en formato PDF.
  Ver: http://cryptoapplet.nisu.org/samples/signpdf.php

6. Roadmap

- Firma formato pdf. El desarrollo de la clase de firma de PDF se prevee para la versión 2 del applet, a pesar de estar ya diseñada, ésta se encuentra en fase de pruebas. Implementación de los parámetros relativos a esta clase:
  - **PDFReason:** Campo “Reason” de la firma de PDF.
  - **PDFLocation:** Campo “Location” de la firma de PDF.
  - **PDFContact:** Campo “Contact” de la firma de PDF.

  - **PDFVisibleSignature:** Valor true/false, indica si queremos que aparezca un rectángulo indicando la información sobre la firma en una página del PDF.
  - **PDFVisibleSignaturePage:** Si el valor anterior es true, indica en que página debe aparecer el rectángulo, en caso contrario es ignorado.

- Funcionalidad de firma por lotes. Esta característica, permitirá realizar varias firmas desde una misma página web con una configuración determinada por el usuario. Se prevee disponible para la versión 2.
- Interfaz gráfico con progreso integrado. La indicación del progreso de las operaciones se muestra en una ventana a parte, se prevee, para la versión 2, integrar ésta ventana en la ventana principal de la aplicación.
- Integración firma factura electrónica. En versiones posteriores, se prevee incluir una clase de firma capaz de generar un XML totalmente compatible con las principales aplicaciones de validación.
- Soporte para Safari en MAC. En versiones posteriores se estudiará también dotar al applet de soporte para gestionar al almacén de certificados de Safari y así poder firmar desde él.
- Versión aplicación de escritorio del applet e integración de validación. Dado que las clases de bajo nivel del applet implementan toda la funcionalidad de acceso a CryptoApi, PKCS#11, etc. Se estudiará la posibilidad de crear una clase que implemente un interfaz de escritorio de firma capaz de ofrecer la misma funcionalidad que el applet. También se estudiará la posibilidad de que ésta aplicación sea capaz de validar los documentos firmados por él.