# OpenSSL / Java

## openssl

hands-on

## 01

Bajar ultima version OpenSSL

Compilar (sin instalar)

Recuperar certificado SSL de google y guardarlo a disco

Visualizar el contenido del certificado con Windows

Visualizar el contenido del certificado con OpenSSL

Volcar individualmente issuer, subject y la fecha caducidad con OpenSSL

Que diferencias hay entre los subjects de OpenSSL y Windows?

Que diferencias hay entre formato PEM y DER?

## 01

```
> openssl x509 -inform DER -outform PEM -in cert.crt -out cert.
pem
> openssl x509 -inform DER -subject -in cert.crt -noout
> openssl x509 -inform DER -issuer -in cert.crt -noout
> openssl x509 -inform DER -enddate -in cert.crt -noout
```

## O2

Haz un dump total del certiﬁcado firma DNIe y mira el OID de política.

Localiza el documento oﬁcial donde dice que signiﬁca esa politica

Localiza la parte donde dice como esta codiﬁcado el numero de DNI

## 02

```
> openssl asn1parse -inform DER -i -in amb-auth.cer
> openssl asn1parse -inform DER -i -in amb-auth.cer -strparse
XXX
2.16.724.1.2.2.2.3
2.16.724.1.2.2.2.4
http://www.dnielectronico.es/PDFs/politicas_de_certificacion.pdf
```

## O3

Copia el certificado raíz y subordinado de tu dnie a disco

Haz que openssl verifique como correcto el certificado del DNIe (sin trampas)

Cambia otra subordinada del dni electrónico y verifica como no funciona si usas esa

## 03

```
> openssl x509 -inform DER -outform PEM -in ca-dgp.cer -out ca-
dgp.pem
> openssl x509 -inform DER -outform PEM -in ca-dni-001.cer -out
ca-dni-001.pem
> cat ca-dgp.pem > ca-all.pem
> cat echo ca-dni-001.pem >> ca-all.pem
> openssl verify -CAfile ca-all.pem amb-auth.pem
```

## 04

Mira donde está la url del OCSP indicado dentro de tu certificado, usa windows o la DPC

Verifica con OCSP que tu certificado no está revocado

Pon la opción -text, que contiene la petición y la respuesta?

Esta revocado el certificado que firma la respuesta OCSP?

## 04

```
> openssl ocsp -issuer ca-dni-001.pem -cert amb-auth.pem -url
http://ocsp.dnielectronico.es/ -CAfile ca-all.pem -text
```

## O5.1

(Minidemo xca)
Instala xca
Crea una CA raíz
Crea un certiﬁcado SSL cliente, duracion 1 año, issuer CN=jo, serial number = 11111111T
Exporta el certiﬁcado cliente con su clave privada a disco (pkcs#12) como "11111111T.p12" contraseña "111111"

## O5.2

Con openssl crea dos archivos, uno con la clave privada (en claro) y otro con el certiﬁcado que esta dentro del pkcs#12
Veriﬁca que es correcto el certiﬁcado con openssl

## 05

```
> openssl pkcs12 -in 11111111T.p12 -nokeys -passin pass:111111
> openssl pkcs12 -in 11111111T.p12 -nocerts -nodes -passin pass:
111111
```

## O6

Crea una firma pkcs#7 detached del archivo message.txt con la clave del pkcs#12

Verifica la firma

Parsea el archivo de firma, busca el hash dentro,haz el hash del archivo manualmente y comprueba que son iguales

Crea una firma pkcs#7 attached con sha256

Verifica la firma

## 06.1

```
>echo this is a message! > message.txt

>openssl smime -sign -in message.txt -signer 11111111T.pem -
inkey 11111111T.key -outform DER -binary -out message.txt.pkcs7

>openssl smime -verify -in message.txt.pkcs7 -binary -content
message.txt -inform DER -CAfile myca-root.cer
```

## M6.2

```
>openssl asn1parse -inform DER -i -in message.txt.pkcs7
```

```
  696:d=6  hl=2 l=  35 cons:       SEQUENCE
  698:d=7  hl=2 l=   9 prim:        OBJECT            :messageDigest
  709:d=7  hl=2 l=  22 cons:        SET
  711:d=8  hl=2 l=  20 prim:         OCTET STRING      [HEX DUMP]:1133E3ACF0A4CBB9D8B3BFD3F227731B8CD2650B
```

```
>openssl dgst -sha1 -hex  < message.txt
1133e3acf0a4cbb9d8b3bfd3f227731b8cd2650b
```

## 06.3

```
>openssl smime -sign -in message.txt -signer 11111111T.pem -
inkey 11111111T.key -outform DER -binary -out message.txt.pkcs7
-nodetach
```

```
>openssl smime -verify -in message.txt.pkcs7 -binary  -inform
DER -CAfile myca-root.cer
```

## O7

Crea un sello de tiempo del archivo message.txt contra una TSA `http://psis.catcert.net/psis/catcert/tsp` con openssl

Verifica el sello de tiempo con openssl

Que contiene?

## 07

```
./openssl ts -query -data message.txt -out message.txt.ts-query
```

```
cat message.txt.ts-query | curl -s -S -H 'Content-Type:
application/timestamp-query' --data-binary @- http://psis.
catcert.net/psis/catcert/tsp  -o message.txt.ts-response
```

```
./openssl ts -reply -in message.txt.ts-response -text
```

## java

hands-on

## J1_ParseDNIe()

Leer el certificado del DNI del disco
Mostrar por pantalla el issuer del certificado
Mostrar por pantalla la fecha caducidad del certificado
Mostrar por pantalla el subject del certificado de tres maneras diferentes

## J1_ParseDNIe()

```java
public static void J1_ParseDNI() throws Exception {
  FileInputStream fs = new FileInputStream(SIGNATURE_CERT_PATH);
  CertificateFactory cf = CertificateFactory.getInstance("X.509");
  X509Certificate cert = (X509Certificate)cf.generateCertificate(fs);
  println(cert.getNotAfter());
  println(cert.getIssuerDN().toString());
  println(cert.getSubjectDN().toString());
  println(cert.getSubjectX500Principal().getName(X500Principal.RFC1779));
  println(cert.getSubjectX500Principal().getName(X500Principal.RFC2253));
}
```

## J2_VerifyDNIe()

Comprobar que no está caducado
Comprobar que dentro de dos años no estará caducado
Comprobar que es un certificado de no repudio
Comprobar la firma de su SUBCA emisora
Comprobar la firma de su CAROOT emisora
Comprobar la autofirma de la CA ROOT

# J2_VerifyDNIe

```java
public static X509Certificate readCertificate(String filePath) throws Exception {
   FileInputStream fs = new FileInputStream(filePath);
   CertificateFactory cf = CertificateFactory.getInstance("X.509");
   X509Certificate cert = (X509Certificate)cf.generateCertificate(fs);
   fs.close();
   return cert;
}
public static void E2_VerifyDNIe() throws Exception {
   X509Certificate certDNI = readCertificate(SIGNATURE_CERT_PATH);
   certDNI.checkValidity();
   certDNI.checkValidity(new Date(new Date().getTime()+2*365*24*3600000L));
   if (!certDNI.getKeyUsage()[1]) throw new Exception ("No tiene nonRepudiation");
   X509Certificate certCA1 = readCertificate(SUB1_CERT_PATH);
   X509Certificate certRoot = readCertificate(ROOT_CERT_PATH);
   certDNI.verify(certCA1.getPublicKey());
   certCA1.verify(certRoot.getPublicKey());
   certRoot.verify(certRoot.getPublicKey());
}
```

## J3_UseP12()

Cargar el PKCS#12
Comprobar que solo existe un certificado y que este tiene la clave privada asociada.
Obtener el certificado (X509Certificate) y la clave privada (PrivateKey)

## J3_UseP12

```
public static void J3_UseP12() throws Exception {
   char[] password = "111111".toCharArray();
   FileInputStream fs = new FileInputStream(MYCERT_P12_PATH);
   KeyStore keystore = KeyStore.getInstance("PKCS12");
   keystore.load(fs, password);
   fs.close();
   Enumeration<?> aliases =  keystore.aliases();
   if (!aliases.hasMoreElements()) throw new Exception("Too few elements!");
   String certificateAlias = (String) aliases.nextElement();
   if (aliases.hasMoreElements()) throw new Exception("Too many elements!");
   if (!keystore.isKeyEntry(certificateAlias))
      throw new Exception("Doesn't have the pvk!");
   X509Certificate cert = (X509Certificate)
       keystore.getCertificate(certificateAlias);
   PrivateKey pvk = (PrivateKey) keystore.getKey(certificateAlias, password);
}
```

## J4_SignEnveloped.1

Firma XMLDSig eveloped del documento

```
<a><b id='sign'></b><c /></a>
```

## J4_SignEnveloped.2

```
public static Document str2xml(String str) throws Exception {
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
DocumentBuilder builder = dbf.newDocumentBuilder();
return builder.parse(
new ByteArrayInputStream(str.getBytes(StandardCharsets.UTF_8))); }

public static String xml2str(Document doc) throws Exception {
ByteArrayOutputStream baos = new ByteArrayOutputStream();
TransformerFactory tf = TransformerFactory.newInstance();
Transformer trans = tf.newTransformer();
trans.setOutputProperty(OutputKeys.INDENT, "yes");
trans.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");
trans.transform(new DOMSource(doc), new StreamResult(baos));
return baos.toString(StandardCharsets.UTF_8.name()); }
```

# J4_SignEnveloped.3

```
public static void J4_SignEnveloped(X509Certificate cert, PrivateKey pvk) throws
Exception {
Document doc = ...
DOMSignContext dsc = new DOMSignContext(..);
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
Reference ref = fac.newReference("",
   fac.newDigestMethod(..., null),Collections.singletonList
   (fac.newTransform(...,
          (TransformParameterSpec) null)), null, null);
SignedInfo si = fac.newSignedInfo(fac.newCanonicalizationMethod
   (...,(C14NMethodParameterSpec)
   null),fac.newSignatureMethod(...,
   null),Collections.singletonList(...));
KeyInfoFactory kif = fac.getKeyInfoFactory();
X509Data x509Data = kif.newX509Data(Collections.singletonList(...));
KeyInfo ki = kif.newKeyInfo(Collections.singletonList(...));
XMLSignature signature = fac.newXMLSignature(..., ...);
signature.sign(...);
System.out.println(xml2str(doc));}
```

## J4_SignEnveloped

```
public static void J4_SignEnveloped(X509Certificate cert, PrivateKey pvk) throws
Exception {
Document doc = str2xml("<a><b id='sign'></b><c /></a>");
DOMSignContext dsc = new DOMSignContext(pvk, doc.getDocumentElement());
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
Reference ref = fac.newReference("",
   fac.newDigestMethod(DigestMethod.SHA1, null),Collections.singletonList
           (fac.newTransform(Transform.ENVELOPED,
          (TransformParameterSpec) null)), null, null);
SignedInfo si = fac.newSignedInfo(fac.newCanonicalizationMethod
   (CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,(C14NMethodParameterSpec)
   null),fac.newSignatureMethod(SignatureMethod.RSA_SHA1,
   null),Collections.singletonList(ref));
KeyInfoFactory kif = fac.getKeyInfoFactory();
X509Data x509Data = kif.newX509Data(Collections.singletonList(cert));
KeyInfo ki = kif.newKeyInfo(Collections.singletonList(x509Data));
XMLSignature signature = fac.newXMLSignature(si, ki);
signature.sign(dsc);
System.out.println(xml2str(doc));}
```

## J5_VerifyEnveloped.1

Verificar la firma hecha con J4_SignEnveloped

Hacer prueba modificando la firma

## J5_VerifyEnveloped.2

```
public static void J5_VerifyEnveloped(final Document doc, final X509Certificate
cert) throws Exception {
   final KeySelector ks = new KeySelector() {
   public KeySelectorResult select(KeyInfo keyInfo, Purpose purpose,
AlgorithmMethod method, XMLCryptoContext context) throws KeySelectorException {
        return new KeySelectorResult() {
          public Key getKey() { return cert.xxx(); }
        }; }; };
   Node node = doc.getElementsByTagNameNS(XMLSignature.XMLNS, ...).item(0);
   DOMValidateContext valContext = new DOMValidateContext(...);
   XMLSignatureFactory factory = ...
   XMLSignature signature = factory.unmarshalXMLSignature(...);
   if (!signature.validate(...)) throw new Exception("XML Validation failed");
}

                                     doc.getElementsByTagName("b").item(0).
getAttributes().removeNamedItem("id");
```

## J5_VerifyEnveloped

```
public static void J5_VerifyEnveloped(final Document doc, final X509Certificate
cert) throws Exception
{
   final KeySelector ks = new KeySelector() {
   public KeySelectorResult select(KeyInfo keyInfo, Purpose purpose,
AlgorithmMethod method, XMLCryptoContext context) throws KeySelectorException {
        return new KeySelectorResult() {
          public Key getKey() { return cert.getPublicKey(); }
        };
      };
   };
   Node node = doc.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature").item
(0);
   DOMValidateContext valContext = new DOMValidateContext(ks,node);
   XMLSignatureFactory factory = XMLSignatureFactory.getInstance("DOM");
   XMLSignature signature = factory.unmarshalXMLSignature(valContext);
   if (!signature.validate(valContext))
      throw new Exception("XML Validation failed");
}
```

## J6_SignPDF.1

Poner librerias iText en el POM

Firmar un PDF

Hacer una firma visible

Guardarlo en el disco

## J6_SignPDF.2

```
public static void J6_SignPDF(X509Certificate cert, PrivateKey pvk) throws
Exception
{
  PdfReader reader = ...
  ...
  PdfStamper stp = PdfStamper.createSignature(...);
  PdfSignatureAppearance sap = stp.getSignatureAppearance();
  sap.setCrypto(...);
  sap.setReason(..);
  sap.setLocation(...);
  ...
  stp.close();
  signedPDFStream.close();
}
```

# J6_SignPDF

```
public static void J6_SignPDF(X509Certificate cert, PrivateKey pvk) throws
Exception
{
  PdfReader reader = new PdfReader(BLANKPDF_PATH);
  FileOutputStream signedPDFStream =
    new FileOutputStream(BLANKPDF_PATH+".signed.pdf");
  PdfStamper stp = PdfStamper.createSignature(reader, signedPDFStream, '\0');
  PdfSignatureAppearance sap = stp.getSignatureAppearance();
  sap.setCrypto(pvk, new Certificate[] {cert}, null,
     PdfSignatureAppearance.WINCER_SIGNED);
  sap.setReason("Adrià Massanet");
  sap.setLocation("Barcelona");
  sap.setVisibleSignature(new Rectangle(100, 100, 200, 200), 1, null);
  stp.close();
  signedPDFStream.close();
}
```

## J7_VerifyPDF.1

Verificar la firma del PDF

# J7_VerifyPDF.2

```
public static void E7_VerifyPDF() throws Exception
{
   PdfReader reader = new PdfReader(...);
   AcroFields acroFields =...
   System.out.println("Number of signatures:"+ ...);
   for (Object signatureName : … )
   {
       PdfPKCS7 pkcs7 = ...
       System.out.println("Signed by subject:"+ … );
       System.out.println("Signature ok:"+...);
   }
}
```

## J7_VerifyPDF

```
public static void E7_VerifyPDF() throws Exception
{
   PdfReader reader = new PdfReader(BLANKPDF_PATH+".signed.pdf");
   AcroFields acroFields = reader.getAcroFields();
   System.out.println("Number of signatures:"+
      acroFields.getSignatureNames().size());
   for (Object signatureName : acroFields.getSignatureNames())
   {
       PdfPKCS7 pkcs7 = acroFields.verifySignature((String)signatureName);
       System.out.println("Signed by subject:"+
          ((X509Certificate)pkcs7.getCertificates()[0]).getSubjectDN());
       System.out.println("Signature ok:"+pkcs7.verify());
   }
}
```

## J8_UseDNIe.1

Usar la clave del DNIe para firmar un PDF

# J8_UseDNIe.2

```
public static void E8_ReadDNIe() throws Exception {
  KeyStore keystore = KeyStore.getInstance(...);
  keystore.load(...);
  Enumeration<?> aliases =  keystore.xxx()
  while (aliases.hasMoreElements()) {
     String alias = (String)aliases.nextElement();
     if (...) { // has private key
        X509Certificate cert = (X509Certificate) keystore.getCertificate(alias);
        if (...) { // non-rep certificate
           System.out.println(cert.getSubjectDN().toString());
           PrivateKey pvk = …;
           J6_SignPDF(cert,pvk);
        }
      }
   }
}
```

# J8_UseDNIe

```
public static void E8_ReadDNIe() throws Exception {
  KeyStore keystore = KeyStore.getInstance("WINDOWS-MY");
  keystore.load(null, null);
  Enumeration<?> aliases =  keystore.aliases();
  while (aliases.hasMoreElements()) {
     String alias = (String)aliases.nextElement();
     if (keystore.isKeyEntry(alias)) {
        X509Certificate cert = (X509Certificate) keystore.getCertificate(alias);
        if (cert.getKeyUsage()[1]) {
           System.out.println(cert.getSubjectDN().toString());
           PrivateKey pvk = (PrivateKey) keystore.getKey(alias, null);
           J6_SignPDF(cert,pvk);
        }
      }
   }
}
```