logoURJCnegativo1.eps

Certificados

Seguridad Informática – Jorge Á. Colinas G.

URJC

[A. Certificados 1](#_Toc25779318)

[A.1. Creando un certificado para “autofirmar”. 1](#_Toc25779319)

[A.2. Creando una Autoridad de Certificación 3](#_Toc25779320)

[B. Anexo codificaciones de certificados 7](#_Toc25779321)

[B.1. Codificación en formato PEM (Privacy Enhanced Mail). 7](#_Toc25779322)

[B.2. Codificación en formato DER (Distinguished Encoding Rules). 7](#_Toc25779323)

[B.3. P7B/PKCS#7 8](#_Toc25779324)

[B.4. Formato PFX/P12/PKCS#12 8](#_Toc25779325)

[B.5. Convertir formaros de Certificados. 8](#_Toc25779326)

# Certificados

Para asegurar que una determinada clave pública pertenece a un usuario en concreto se utilizan los certificados digitales.

Un certificado digital es un documento electrónico que asocia una clave pública con la identidad de su propietario.

Podemos clasificar los certificados según la autoridad que los certifica:

* los certificados emitidos por una autoridad de certificación (Certification Authority) ,

es la CA quién certifica dichos datos.

* Los certificados autofirmados, el propio servidor crea su certificado, asegurando el propio servidor su validez.

Algunos comandos de OpenSSL :

**ca:** permite crear y gestionar una Autoridad de Certificación basada en el modelo de confianza jerárquico.

**req**: permite crear y gestionar peticiones de emisión de certificados X.509.

**x509**: permite gestionar certificados X.509.

**verify**: permite verificar certificados X.509.

## Creando un certificado para “autofirmar”.

Usando *openssl req* creamos un certificado, con las opciones:

-nodes indicamos que es sin contraseña,

-days la caducidad del certificado en días,

-keyout contiene la claveprivadapublica utilizada en el certificado, (también se puede utilizar una creada previamente, para ello se utiliza la opción -key):

**Common Name (CN)**. Nombre de dominio para el que se va a solicitar el certificado. El nombre de dominio (o subdominio) de tu sitio web, deberá coincidir exactamente con el dominio que escribirán los visitantes al acceder a tu web (*www.tudominio.com ; \*.tudominio.com ; secure.tudominio.com*).

Los campos destinados al **email**,**optional company name** y **challenge password**pueden dejarse en blanco.

openssl req -new -nodes -keyout claveprivpub -out certificado.csr -days 365

Se generan dos archivos (si hemos utilizado la opción -keyout), un certificado y una clave privada.

Vemos el contenido de la clave privada creada:

openssl rsa -in claveprivpub -text -out claveprivpub.txt

writing RSA key

cat claveprivpub.txt

RSA Private-Key: (2048 bit, 2 primes)

modulus:

00:cd:a6:de:20:46:52:b6:0d:52:d2:c6:5b:f8:44:

08:86:ec:25:a9:f6:fb:11:0f:87:70:2e:20:8e:80:

3c:0a:ef:8a:7d:de:05:c3:f1:71:39:14:4a:d4:f4:

… … …

Para ver el contenido del certificado

openssl req -subject -noout -in certificado.csr

openssl req -text -noout -in certificado.csr

El siguiente paso es enviar el CSR (Certificate Signing Request) a una entidad certificadora para que lo firme y emita el certificado correspondiente.

Nosotros vamos a realizarlo ***autofirmado***, para ello lo deberemos firmar con la clave creada anteriormente.

Opciones utilizadas:

x509 se utiliza para firmar solicitudes y verificar certificados.

-req indica que se intenta crear un certificado a partir de una solicitud desde el archivo especificado por -in.

-days días que tardará en caducar.

-signkey indica la clave con la que será firmado el certificado, aquí es la propia clave privada del certificado emitido.

openssl x509 -req -days 365 -in certificado.csr -signkey claveprivpub -out certificado.crt

Para ver el contenido del certificado:

openssl x509 -text -noout -in certificado.crt

El *issuer* es el emisor del certificado, es decir, el que lo firma con su clave privada.

El *subject* es el propietario del certificado, es decir, el que lo crea y solicita a la CA que lo firme.

Archivos generados:

claveprivpub : claves asimétricas utilizadas en el certificado.

certificado.csr : Certificado creado

certificado.crt : Certificado autofirmado, con la clave privada claveprivpub

openssl x509 -subject -issuer -enddate -noout -in certificado.crt

## Creando una Autoridad de Certificación

Para crear una AC, necesitamos crear una estructura de directorios y ficheros:

Directorios:

$directorio/CA/certs:

Directorio donde se guardan los certificados ya firmados y enviados a los clientes. $directorio/CA/newcerts

Directorio donde se guardan los certificados que acaban de ser firmados.

$directorio/CA/crl (Certificate Revokation List).

Directorio donde los certificados revocados son almacenados.

$directorio/CA/csr (Certificate Signing Request).

Directorio donde están las peticiones de certificados que tenemos pendientes de firmar. $directorio/CA/private

Directorio donde se guarda la clave privada de la autoridad de certificación. (También contiene el resto de claves privadas que sean generadas para los diferentes servicios).

Ficheros necesarios:

$directorio/CA/serial

Indica el número de serie que tiene el próximo certificado a firmar. Para ello deberá contener dentro del archivo el número 01 y un salto de línea.

$directorio/CA/crlnumber

Indica el número de serie que tendrá el próximo certificado revocado. Para ello deberá contener dentro del archivo el número 01 y un salto de línea.

$directorio/CA/index.txt

Fichero vacío y contendrá un listado de los certificados firmados.

Comandos Linux:

*directorio=.*

*mkdir -p $directorio/CA/certs*

*mkdir -p $directorio/CA/newcerts*

*mkdir -p $directorio/CA/crl*

*mkdir -p $directorio/CA/csr*

*mkdir -p $directorio/CA/private*

*echo 01 > $directorio/CA/serial*

*echo 01 > $directorio/CA/crlnumber*

*touch $directorio/CA/index.txt*

*directorio=`pwd`*

*dir=$directorio*

Hacemos una copia del fichero de configuración antes de modificarlo. El fichero de configuración es: **/etc/ssl/openssl.cnf lo copiamos en $dir/CA/openssl.cnf**

Dentro de este fichero debemos modificar según nuestras necesidades:

[ CA\_default ]

dir $dir/CA Directorio raíz de la autoridad de certificación.

certs $dir/certs Directorio donde se almacenarán los certificados ya firmados.

crl\_dir $dir/crl Directorio donde los certificados revocados son almacenados.

database $dir/index.txt Archivo con la base de datos de los certificados.

new\_certs\_dir $dir/newcerts Directorio donde se guardan los certificados que acaban de ser firmados.

certificate $dir/irtic.pem Archivo con la clave pública de la autoridad de certificación.

serial $dir/serial Archivo con el número de serie de los certificados.

crlnumber $dir/crlnumber Archivo con el número de serie de revocación.

crl $dir/crl.pem Lista de los certificados revocados.

private\_key $dir/private/irtic.key Archivo con la clave privada de la autoridad de certificación.

RANDFILE $dir/private/.rand Archivo con el número aleatorio privado.

x509\_extensions usr\_cert Extensiones que han de añadirse al certificado.

name\_opt ca\_default Formato en que se mostrará el nombre del certificado antes de que sea firmado.

cert\_opt ca\_default Formato en que se mostrará un certificado antes de que sea firmado. default\_days 365 Días por defecto para los que se firma el archivo.

default\_crl\_days 30 Días por defecto en que debe ser actualizada la lista de certificados revocados de esta autoridad de certificación.

default\_md default Compendio de mensaje utilizado, por defecto es md5 (valor default).

preserve no Indica si se ha de mantener o no el orden Domain Name.

policy policy\_match Política por defecto a aplicar si no se especifica ninguna.

Las políticas vienen definidas en los apartados policy\_math (para firmar los propios certificados de la CA) de o policy\_anything (para firmar certificados de otras organizaciones). Cuando queremos que el campo no se modifique, es decir que sea el indicado por la entidad certificadora pondremos la opción *match*, si queremos que sea obligatorio pondremos *supplied* y para los opcionales *optional.*

[ req\_distinguished\_name ]

countryName\_default ES País de emisión del certificado

stateOrProvinceName\_default Madrid Estado o provincia de emisión.

localityName\_default Vicalvaro Localidad de emisión del certificado

0.organizationName\_default URJC Nombre de la organización

organizationalUnitName\_default SEGINF Nombre de la sección

commonName\_default AC de URJC Nombre de la autoridad

emailAddress\_default [info@urjc.es](mailto:info@urjc.es) Dirección de correo del responsable de la autoridad de certificación

[ req ]

default\_bits 2048 Bits por defecto de la clave privada.

default\_md sha1 Resumen HAS utilizado por defecto.

Para crear el certificado de la autoridad de certificación, certificado raíz, recordemos que este certificado es el que nos convertirá en una autoridad certificadora CA, nos solicitará una "pass phrase", una contraseña, pero en forma de una frase que es con la que validaremos nuestra autoridad. Se solicitará en las operaciones de validación, baja de certificados, ….

openssl req -new -x509 -days 365 -config $dir/CA/openssl.cnf -keyout $dir/CA/private/claveCA.key -out $dir/CA/certificadoCA.pem

Podemos ver clave privada generada con:

openssl rsa -in $dir/CA/private/ claveCA.key -text

El certificado generado:

openssl x509 -in $dir/CA/certificadoCA.pem -text

Para ver el propósito del certificado con:

openssl x509 -in $dir/CA/certificadoCA.pem -purpose

### Firmando certificados:

Suponemos que nos ha llegado una solicitud de certificado, que es el fichero $dir/CA/csr/servidor.csr. Dicha solicitud ha sido generada tal y como vimos con anterioridad.

Podemos examinar dicha solicitud que nos ha llegado mediante el comando:

openssl req -in $dir/CA/csr/certificado.csr -text

Si consideramos que dicha solicitud es correcta, podemos pasar a firmarla mediante:

openssl ca -config $dir/CA/openssl.cnf -in $dir/CA/csr/certificado.csr -verbose

Como salida del comando anterior, obtendremos, dentro del directorio $dir/CA/newcerts un fichero con el nombre <número de serie>.pem, donde <número de serie> es el número de serie que en el instante de la firma tuviera el fichero serial.

Al no especificar ninguna política, utilizamos la política por defecto, por lo que solo podemos firmar certificados, de nuestra propia organización, que es la política por defecto. Si deseamos firmar certificados con nuestra autoridad de certificación para cualquier organización externa, debemos ejecutar el comando (ver apartado policy\_anything del fichero de configuración):

openssl ca -config $dir/CA/openssl.cnf -in $dir/CA/csr/certificado.csr -verbose -policy policy\_anything

El cual indica que no es necesario que coincida el nombre de la autoridad de certificación con el nombre de la autoridad que solicita la firma del certificado.

Verificar el certificado creado:

openssl verify -CAfile $dir/CA/certificadoCA.pem $dir/CA/newcerts/01.pem

cp $dir/CA/newcerts/01.pem $dir/CA/certs/certificadi1.pem

Para revocar un certificado:

openssl ca -revoke $dir/CA/certs/certificado1.pem –config $dir/CA/openssl.cnf

Actualizamos la lista de certificados revocados:

openssl ca –config $dir/CA/openssl.cnf -gencrl -out $dir/CA/crl/certificado1.pem

Consultar los certificados revocados:

openssl crl -in $dir/CA/crl/certificado1.pem -text

Toda autoridad certificadora tiene la posibilidad de revocar algún certificado emitido. El motivo de revocar un certificado puede ser tan sencillo como la caducidad del mismo, el extravío de algún certificado, etc.

La lista de certificados revocados debe ser puesta a disposición de los usuarios de la red de forma similar a como ponemos nuestra autoridad de certificación. Esto permite a los usuarios descargarse la lista de certificados revocados e instalarla en su ordenador, pudiendo comprobar que los certificados que le llegan, además de haber sido firmados por nuestra CA, continúan siendo válidos para nuestra CA que los ha firmado.

# Anexo codificaciones de certificados

El estándar X.509 solo define la sintaxis de los certificados.

Se utiliza la extensión en el nombre del certificado para identificar cómo está codificado internamente el certificado y con esto evitaremos problemas luego a la hora de manipular los certificados digitales.

## Codificación en formato PEM (Privacy Enhanced Mail).

Cuando los certificados están codificados en Base64 / ASCII.

Estos certificados al abrirlos con un editor de texto comienzan y terminan con líneas similares a las siguientes:

-----BEGIN CERTIFICATE-----

...

-----END CERTIFICATE-----

En los certificados codificados PEM se suele utilizar las extensiones: .pem, .crt, .cer, y .key   
El certificado PEM y la clave privada pueden combinarse en un solo archivo, pero para la mayoría de las plataformas, el certificado y la clave privada están estar separados.  
  
Para ver el contenido del certificado codificado PEM:

openssl x509 -text -noout -in certificado.pem

## Codificación en formato DER (Distinguished Encoding Rules).

La extensión DER es utilizada para certificados digitales codificados en forma binaria.

Los archivos codificados en formato DER se utilizan las extensiones: .der, y .cer

Comando para ver el contenido del certificado codificado DER:

openssl x509 -in certificado.der -inform der -text -noout

**Utilizando Certificados para conexiones Seguras.**

En la práctica, los certificados que presenta un servidor para establecer conexiones no están directamente firmados por las CA de orden superior (son aquellas que firman ellas mismas sus certificados), sino que existe una jerarquía de **certificados intermedios** entre el más bajo (el del dominio del servidor al que se conecta la app) y el superior de la cadena (el certificado raíz de la CA). Esta cadena de certificados es lo que se denomina una cadena de confianza’ (chain of trust) y que se utiliza en el proceso de verificación.

## P7B/PKCS#7

Esta codificación puede contener solo uno certificado o también los certificados intermedios en formato BASE64/ASCII.

Los archivos codificados en formato DER se utilizan las extensiones: .p7b, y .p7c

Contiene el certificado y certificados intermedios, pero contiene no la clave privada.

## Formato PFX/P12/PKCS#12

El formato PKCS#12 o PFX/P12 es una codificación en binario con el certificado (incluso también con los intermedios) con la clave privada. Los certificados y la clave privada en el archivo PFX son protegidos por la contraseña.

La extensión del formato más utilizada es .pfx y .p12.

PKCS#12 se utiliza muy a menudo para dispositivos con Windows a fin de importar y exportar los certificados junto con la clave privada.

Los certificados colocados en PFX también se utilizan para firmar en Microsoft Authenticode.

## Convertir formaros de Certificados.

### Convertir de formato PEM a DER:

openssl x509 -in certificado.crt -outform der -out certificado.der

### Convertir de formato PEM a P7B:

Agrega los certificados PEM *certificado.cer* y *CertificadoCA.cer* (o cadena de certificados) a *certificado.p7b*

openssl crl2pkcs7 -nocrl -certfile certificado.cer -out certificado.p7b -certfile CertificadoCA.cer

El parámetro -certfile se puede añadir tantas veces como certificados desee agregar al archivo.

### Convertir de formato PEM a PFX:

openssl pkcs12 -export -out cetrificado.pfx -inkey ClavePrivada.key -in certificado.crt -certfile CertificadoCA.crt

### Convertir de formato DER a PEM:

openssl x509 -in certificado.crt -inform der -outform pem -out certificado.pem

### Convertir de formato P7B a PEM:

Tener en cuenta que PKCS tiene múltiples ítems contenidos (p. ej. un certificado y un certificado intermedio firmado CA)

openssl pkcs7 -print\_certs -in certificado.p7b -out certificado.cer

### Convertir de formato P7B a PFX:

openssl pkcs7 -print\_certs -in certificado.p7b -out certificado.cer

openssl pkcs12 -export -in certificado.cer -inkey ClavePrivada.key -out certificado.pfx -certfile CertificadoCA.cer

### Convertir certificado PFX a PEM

openssl pkcs12 -in certificado.pfx -out certificado.crt -nodes

openssl rsa -in certificado.crt -out certificado.key

openssl pkcs12 -in certificado.pfx -out certificado.crt -nodes -nokeys

-nocerts para únicamente obtener la clave privada o añadir

-nokeys para obtener solo los certificados.