

OpenSSL y certificados digitales – Práctica

Publicado por [Diego Córdoba](#) en 2 enero, 2023

En esta entrada trabajaremos con OpenSSL y certificados digitales, cómo generar certificados, firmas, CAs, verificación, y cómo extraer información.

Este artículo forma parte de la [Serie sobre Criptografía Aplicada](#) publicada en este blog.
Pueden visitar el [índice de la serie](#) para acceder a todo el contenido.

[Anteriormente](#) aprendimos sobre la Infraestructura de clave pública, o PKI. Analizamos los conceptos principales para garantizar la seguridad basados en criptografía asimétrica, funciones hash, y la confianza en una tercera entidad en la comunicación: la autoridad certificante, o CA.

Hoy vamos a aprender cómo trabajar con certificados digitales utilizando OpenSSL, y cómo implementar autoridades certificantes.

Creación de certificados digitales

En un artículo anterior estudiamos los diferentes formatos comunes que pueden tener los certificados digitales [x.509](#). Ahora veremos cómo trabajar con ellos usando OpenSSL.

Recordemos que la criptografía asimétrica se basa en **dos claves, una pública y una privada**. Cuando hablamos de PKI y mencionamos que lo que utilizan los nodos de una comunicación segura son **certificados digitales**, cabe resaltar que un certificado digital es la clave pública de una entidad, a la que se ha añadido **información de validez** (hashes, firmas digitales, fechas de expiración, etc).

No tenemos todavía una autoridad certificante, o CA, por lo que vamos a crear nuestro primer certificado firmado por la misma clave privada par de la pública incluida en el certificado. A esto se lo



Primero creamos nuestra clave privada RSA, llamémosle, `privkey.pem`.

```
openssl genrsa -out privkey.pem 2048
```

El paso siguiente es generar la petición de firma de certificado, que idealmente utilizará una autoridad certificante para darnos un certificado firmado que incluya nuestra clave pública.

El archivo **CSR (Certificate Sign Request)**, en nuestro caso inicial, vamos a firmarlo nosotros mismos con la clave privada que acabamos de crear.

```
openssl req -new -key privkey.pem -out cert.csr
```

Este comando nos va a solicitar los medatados que serán incorporados al certificado digital firmado.

Veamos un ejemplo de carga de datos:

```
$ openssl req -new -key privkey.pem -out cert.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AR
State or Province Name (full name) [Some-State]:Mendoza
Locality Name (eg, city) []:San Rafael
Organization Name (eg, company) [Internet Widgits Pty Ltd]:JuncoTIC
Organizational Unit Name (eg, section) []:Contenido
Common Name (e.g. server FQDN or YOUR name) []:juncotic.com
Email Address []:diego@juncotic.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

El archivo de salida CSR tiene un contenido similar a este (luego veremos cómo interpretarlo):

```
diego@cryptos:~$ cat cert.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIC2zCCAcMCAQAwgZUxCzAJBgNVBAYTAkFSMRAwDgYDVQQIDAgnZW5kb3phMRMw
EQYDVQQHDApTYW4gUmFmYVVsMREwDwYDVQQKDAhKdW5jb1RJQzESMBAGA1UECwwJ
Q29udGVuaWVwMRUwEwYDVQQDDAxqdW5jb3RpYy5jb20xITAfBgkqhkiG9w0BCQEW
EmRpZwdvQGp1bmNvdGljLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBANNTBikAzu76rblIwayXMHZ6mJD2MkvJY0vQp+gAt9LVGrz7Wd39KFZdzH
AuPI5nXleX2imin7CI50dwAFF0R+aTuDaEfwoXNtDaD6ZzaKTRKNcUj0ZatToV0
XEYE9saYhTtur/bao0N+NX9AcxKVJiBwl4FPHR9XaUG9sQu2v5Y/yHS+gKNash2c
oAOH+iD8KxynwbIIv09Rh0smHmWze84rxbTX1GEfN7+Mld1NpJ9QZ419MrkBvV5u
dkL5fFqB3qyhMMMMmAluTQV5C+bUecs0t141A17kCto22gE7XJdYtXEZijMUx0FI
OggSHIyAv0VXTke0fAIjb+AUNzkCAwEAaAAMA0GCSqGSIb3DQEBCwUA4IBAQAY
TMiinNa1V8q1yqen1J6BYGzAv3MVRobtPDMMGzHvi+iUOV+kE11m36NBzSk1.iBmZ
```



```
openssl x509 -req -days 10000 -in cert.csr -signkey privkey.pem -out c
```

Esto nos genera el archivo **cert.crt** (.crt, no confundir con el .csr generado recién). Veamos un ejemplo de su contenido:

```
[diego@cryptos ~]$ cat cert.crt
-----BEGIN CERTIFICATE-----
MIIDszCCCapsCFAV1JkkR1myd+ca7+wFCVhfr...bZ9MA0GCSqGSIb3DQEBCwUAMIGV
MQswCQYDVQQGEwJB...jEQMA4GA1UECAwHTWVuZG96YTETMBEGA1UEBwwKU2FuIFJh
ZmFlbDERMA8GA1UECgwISnVuY29USUMx...EjAQBgNVBA...MCUNvb...RlbmlkbzEVMBMG
A1UEAw...ManVuY290a...MuY29tMSEwHwYJKoZIhv...NAQkBFhJka...Vnb...BqdW5jb3Rp
Yy5jb20wHhcNMjMwMTAyMjE1MjAxWhcNMzIxMjMwMjE1MjAxWjCB...TELMAkGA1UE
BhMCQVIXEDA0BgNVBAgMB01l...bmRvemExEzARBgNVBAcMClNhbiBSY...ZhZ...wxE...TAP
BgNVBAoMCEp1bmNv...V...L...R...D...250...Z...5p...Z...8xFTATB...NV...BAMMDGp1
bmNvdGljLmNvbTEhMB8GCSqGSIb3DQEJARYS...Z...29AanVuY290a...MuY29tMII...B
IjANBgkqhkiG9w...BAQEF...AAOCAQ8AM...IBCgKCAQEA...o1MGKQD...07v...tuUjBrJcw...fP
qYkPYyS8lg69Cn6AC32UavPtZ3f0oVl3McC48j...deV5faKaKfsIj...nR3AAU8U5H5
b04NoR/C...h...20...N...P...n...N...F...o...1...x...SP...1...g...10...b...15...c...B...T...2...x...j...E...026...9...t...g...i...0341...f...R...z...E...n...l...m
```

Este es un certificado x509 perfectamente válido para utilizar en nuestros servicios!

NOTA: El navegador no confía en mí!

Hay que aclarar un tema: la confianza genera problemas.

Si cargamos este certificado en un servidor web, para proveer HTTPS, el cliente seguramente obtendrá una advertencia de sitio dañino o certificado no válido. Esto es porque el navegador del cliente no posee la clave pública (en nuestro caso, el mismo certificado) para verificar la firma digital (al ser autofirmado, se usa el mismo certificado para verificar su firma digital).

Automatizando algunas cosas

Hemos visto cómo crear el certificado digital, la petición de firma, y el certificado firmado paso a paso. No obstante, si debemos realizar este procedimiento repetidamente, puede resultar engorroso.

OpenSSL provee una forma de crear un certificado digital x509 autofirmado y su clave privada en un solo comando, de manera muy sencilla:

```
openssl req -x509 -nodes -days 1000 -newkey rsa:2048 -keyout privada.
```

Veamos un ejemplo de su salida:



三

Verificando la firma

Tenemos un certificado autofirmado, por lo que la firma digital se puede verificar utilizando el mismo certificado digital. En nuestro caso, `cert.crt` es el certificado firmado usando la clave privada par de la pública incluida en el mismo certificado. Así, podemos verificar la firma de la siguiente manera:

```
openssl verify -CAfile cert.crt cert.crt
```

```
[diego@cryptos ~]# openssl verify -CAfile cert.crt cert.crt  
cert.crt: OK
```

Basta de autofirmados, vamos con la CA

Supongamos que queremos crear un certificado digital firmado por una CA creada por nosotros mismos.

Primero debemos tener clave privada y certificado de la CA... en nuestro ejemplo, creamos una clave y certificado autofirmado de la CA (supongamos que confiamos plenamente en nosotros mismos :P).

```
openssl req -x509 -nodes -days 1000 -newkey rsa:2048 -keyout privada
```



Ahora creamos una **clave privada** para, por ejemplo, un servidor web:

```
openssl genrsa -out privada_sitio.pem 2048
```

Ya tenemos la clave privada, creamos una **petición de firma** para la CA:

```
openssl req -new -key privada_sitio.pem -out cert_sitio.csr
```

Finalmente, creamos el certificado basado en el CSR que acabamos de generar, firmado con la clave privada de la CA:

```
openssl x509 -req -in cert_sitio.csr -CA cert_CA.crt -CAkey privada_CA
```

Este comando puede fallar en algunas versiones de openssl, con el mensaje de que no encuentra un determinado archivo **.srl**.

Cuando se emite un certificado mediante una Autoridad de Certificación (CA), se debe asignar un número de serie único a cada certificado emitido. Este número de serie se incluye en el certificado y se utiliza para identificarlo de manera única. Cuando se trabaja con CAs, algunas versiones de openssl requieren que se cree un archivo donde se almacene el número de serie de los certificados firmados.

La opción **-CAcreateserial** se utiliza cuando se genera un nuevo certificado firmado por una CA existente. Si el archivo de serie aún no existe, esta opción creará un nuevo archivo y asignará el número de serie 01 al primer certificado emitido. Si el archivo ya existe, se continuará utilizando el último número de serie disponible en el archivo existente.

Por esto es que, si el comando anterior falla al no encontrar el archivo de serie con extensión **.srl**, esta opción lo creará y podrá terminar el comando correctamente.

Y sí, podemos verificar la firma si quisiéramos:

```
$ openssl verify -CAfile cert_CA.crt cert_sitio.crt  
cert_sitio.crt: OK
```

En este caso he especificado un hash SHA512 para la generación de la firma. Recuerden no usar nada menor a SHA256 por seguridad.

Ahora, lo que se estarán preguntando: el navegador del cliente debe confiar en la CA creada por nosotros, por lo que deberemos **compartirle el certificado de la CA** (el que usamos para verificar la firma), de modo que el cliente lo pueda añadir a la lista de certificados válidos en su navegador.



NO, cuando montamos un servidor en producción, lo más recomendable es que no creamos nuestra propia CA para firmar los certificados, sino que creamos la petición de firma (el CSR), y le solicitemos a una autoridad certificante válida la generación del certificado firmado. Estas CAs son conocidas, y los navegadores confían en ellas (es decir, ya disponen de su certificado para verificar la firma).

Algunas CAs conocidas son las siguientes:

- Comodo (<https://www.comodo.com/>)
- GeoTrust (<https://www.geotrust.com/>)
- GoDaddy (<https://www.godaddy.com/web-security/ssl-certificate>)
- DigiCert (<https://www.websecurity.digicert.com/ssl-certificate>)

Todas estas CAs son de pago, los precios, para tener una idea, rondan los 400 dólares por año por dominio.

No obstante, una autoridad certificante válida gratuita muy utilizada es [Let's Encrypt](#), una organización sin fines de lucro que se encarga de firmar certificados por períodos más cortos de tiempo, y dispone de herramientas para la actualización automática de certificados digitales.

En mi sitio estoy usando esta CA, va muy bien, y es relativamente fácil de configurar. En otro artículo les explicaré cómo hacerlo 😊

Extrayendo información de los certificados

Tanto en certificados digitales como en peticiones de firma CSR, podemos extraer información utilizando OpenSSL. Veamos un ejemplo:

```
$ openssl x509 -in cert_sitio.crt -noout -text
Certificate:
Data:
    Version: 1 (0x0)
    Serial Number:
        1f:d1:96:29:80:f8:21:82:82:bf:da:15:fe:11:f1:ea:a6:55:f0:a
    Signature Algorithm: sha512WithRSAEncryption
    Issuer: C = AR, ST = Mendoza, L = San Rafael, O = MI_CA, OU =
    Validity
        Not Before: Jan 2 22:29:36 2023 GMT
        Not After : Dec 30 22:29:36 2032 GMT
    Subject: C = AR, ST = Mendoza, L = San Rafael, O = Nueva Empre
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
            Modulus:
                00:ca:ee:16:df:49:8b:cd:2c:5a:d9:9a:c1:70:99:
                79:e5:27:c6:fd:7f:2a:14:2a:e8:38:7e:0f:0b:e4:
```



```
68:78:2e:8c:03:c3:82:24:26:6e:bc:2f:a2:f3:b3:  
ce:68:af:f2:14:b1:03:ba:28:3b:4e:01:7a:f7:64:  
df:e2:2d:5d:6d:0e:20:55:77:26:89:58:31:a4:8c:  
44:cc:7f:75:64:3e:f3:b8:43:84:e5:08:86:a8:b5:  
03:f9:fb:41:c9:84:30:00:1d:a4:3d:0d:85:fe:af:  
4b:5e:f8:ed:5a:3c:8e:74:92:f3:29:20:18:82:2c:  
e5:20:df:d5:b0:9a:72:ab:3c:ad:e3:e0:d4:f3:52:  
bd:ab:7d:61:12:b9:47:97:8e:0a:04:19:c5:41:37:  
22:2c:1c:e0:52:d3:dc:92:92:bb:14:57:cf:70:33:  
91:63:81:8e:5a:58:17:c9:da:4b:4f:92:13:c2:e2:  
37:24:7a:07:0f:eb:91:57:d6:85:c1:7b:84:f9:79:  
f3:c1
```

Exponent: 65537 (0x10001)

Signature Algorithm: sha512WithRSAEncryption

Signature Value:

```
0a:bc:01:e1:81:f5:63:db:ea:34:1e:80:86:0d:83:73:89:af:  
6b:38:a4:e2:c8:d8:8a:de:10:b0:3b:55:7f:4d:e1:ad:9f:19:  
fa:18:00:ae:e2:45:30:60:b5:80:54:a4:48:45:9c:e3:fd:47:  
6f:75:3f:0f:4a:bb:5c:f6:a6:a2:03:5c:bc:01:b7:5e:2d:30:  
a0:d0:ba:93:60:12:a1:a5:8a:a2:2c:05:e3:f3:9e:d8:0a:20:  
c5:94:8f:5b:7c:11:7e:0f:de:25:0b:fc:3b:ca:91:c0:41:f2:  
17:18:70:b0:81:e2:62:48:b9:88:a0:5f:d1:76:db:6d:78:65:  
7e:91:28:c0:a1:75:69:d7:28:97:9c:c0:ac:27:0d:fd:9a:a7:  
63:34:32:6e:59:54:92:e1:63:bd:24:12:18:20:88:3f:95:34:  
68:3c:18:01:f8:de:f8:75:d4:5c:41:d2:ee:dc:a0:b7:33:de:  
de:06:23:b3:64:12:b2:0c:f3:9f:f6:78:a6:03:08:3a:12:9d:  
40:c7:25:e3:9b:b7:79:41:aa:a1:96:d7:44:3b:a1:0c:9c:ce:  
ce:37:e7:95:39:dd:fa:c7:9a:81:73:3f:8a:b8:f1:19:06:c0:  
0c:d9:57:96:ff:4a:11:76:dd:99:1a:d0:1a:13:1b:14:9e:35:  
48:df:b3:81
```

A su vez, si alguien nos envía su certificado digital, y requiere que encriptemos un archivo utilizando la clave que tiene incorporada, podemos extraerla de la siguiente forma:

```
openssl x509 -inform pem -in cert.crt -pubkey -noout > publica.key
```

Luego podemos usar la clave pública [como hemos aprendido con anterioridad](#).

OpenSSL y certificados digitales: Conclusiones

Hemos realizado un recorrido por las principales opciones de OpenSSL para trabajar con certificados digitales.

Un tema interesante, más para quien trabaja como CA, es la revocación ([revoke](#)) de certificados

digitales. No lo he incluido en este artículo para no extenderme demasiado, además de que es

general, un administrador de sistemas suele solicitar las firmas o revocaciones de certificados a las CAs autorizadas.



Por otro lado, aquí he realizado todos los pasos utilizando la utilidad OpenSSL por línea de comandos, pero recordemos que OpenSSL tiene librerías de código para casi cualquier lenguaje (C, Python, etc), por lo que podríamos implementar nosotros mismos una aplicación que genere certificados, firme, verifique firmas, etc.

Sin más, me comprometo a documentar el caso de uso de Let's Encrypt para quien le interese montar su propio servicio seguro SSL/TLS de manera gratuita.

Como siempre, ya saben dónde encontrarme para consultas y sugerencias.

Hasta la próxima!

¿Preguntas? ¿Comentarios?

Si tenés dudas, o querés dejarnos tus comentarios y consultas, sumate al [grupo de Telegram de la comunidad JuncoTIC!](#)

¡Te esperamos!

Artículos relacionados:

[Criptografía Aplicada - Índice de la serie](#)

[PKI: ¿Qué es la infraestructura de clave pública?](#)

[Apache2 y SSL/TLS en Debian: ejemplo de configuración](#)

[x.509: Certificados digitales DER, CRT y CER](#)

[OpenSSL y Criptografía Asimétrica - Práctica](#)

[XCA: interesante front-end para openssl](#)

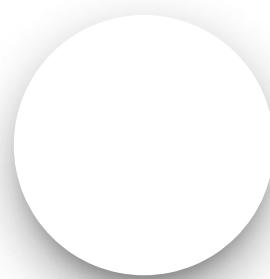


[Let's Encrypt y Certbot
en GNU/Linux - HTTPS](#)

[OpenSSL: Cifrado un
archivo
simétricamente](#)

Categorías: [APLICACIONES](#) [CRIPTOGRAFÍA](#)

Etiquetas: [openssl](#) [ssl](#) [tls](#) [x509](#)



Diego Córdoba

- Ingeniero en Informática - Mg. Teleinformática - Tesis pendiente - Docente universitario - Investigador

¡Seguinos!

[Telegram Channel](#)

[RSS Feed](#)



Nuestros cursos

[Administra GNU/Linux por línea de comandos](#)

[GNU/LINUX Básico online](#)

[Editor VIM online](#)

Buscar

Buscar ...



Entradas recientes

[Blueprints en Flask](#)[DataFrame con Pandas a partir de CSV](#)[Archivos en formulario de Flask](#)[Cálculo de raíz cuadrada dígito a dígito](#)[Histogramas con Matplotlib](#)[Función lambda en Python](#)[Grillas en Matplotlib](#)

Categorías

[Aplicaciones](#)[Criptografía](#)[Educación](#)[InfoSec](#)[Inteligencia Artificial](#)[Linux](#)[Noticias](#)[Programación](#)[Redes](#)



Videos

Etiquetas

algoritmo algoritmos antiguos algoritmos de ordenamiento bash command criptografia crypto debian
Flask fsf git gnu hack hacking htop infosec Inteligencia Artificial kernel linux mount networking
open source openssl password privacy procesos prog programacion programming python red
security seguridad software libre ssh ssl sysadmin systemd sysvinit tcip top torvalds tunneling vim
vulnerability

Entradas relacionadas



APLICACIONES

XCA: interesante front-end para openssl

Hoy les voy a mostrar algunas características de XCA, un interesante front-end grafico para realizar varias tareas con claves asimétricas y certificados digitales utilizando openssl. Se trata de un front-end gráfico basado en Qt que [Leer más...](#)



CRIPTOGRAFÍA

Criptografía Aplicada – Índice de la serie

En esta oportunidad les comparto un breve artículo a modo de índice de la serie de posts sobre criptografía aplicada que he publicado recientemente en el blog. Quienes me conocen saben que la criptografía aplicada [Leer más...](#)



CRPTOGRAFÍA

rsync: Un vistazo a este extraordinario comando

Hoy veremos los puntos principales de la gran utilidad rsync para copiar/sincronizar archivos tanto locales como remotos mediante SSH. Hace tiempo que quería escribir alguna breve documentación sobre el comando rsync en GNU/Linux, así que [Leer más...](#)

[BLOG](#)[CURSOS](#)[EVENTOS](#)[SERVICIOS](#)[NOSOTROS](#)[CONTACTO](#)