

Memoria Práctica 4: Certificados Digitales

Seguridad y Protección de Sistemas Informáticos

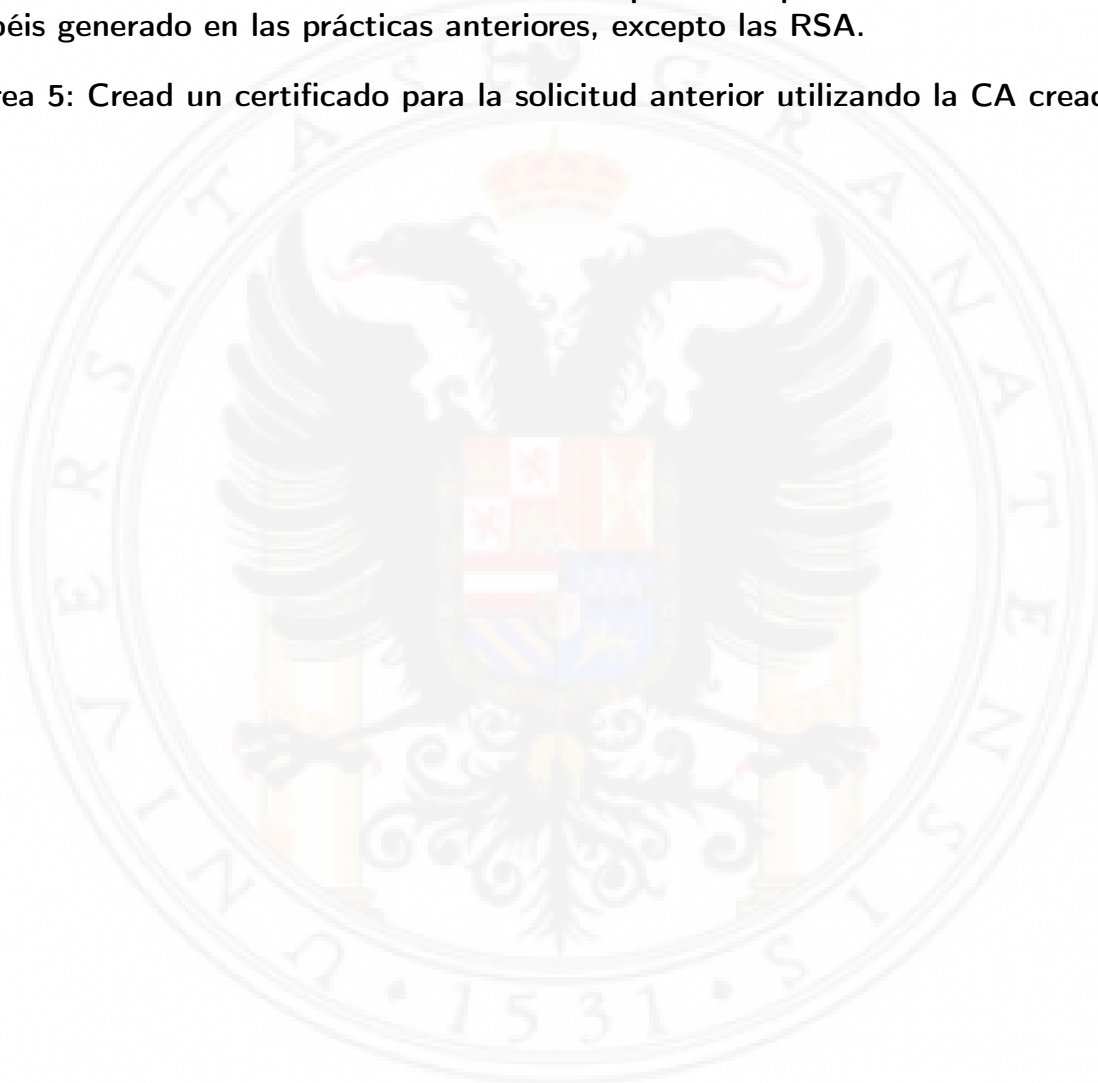
CURSO 2017-2018

Arthur Rodríguez Nesterenko - DNI Y1680851W

11 de diciembre de 2017

Índice

1. Tarea 1: Cread una autoridad certificadora. En este caso se premiará el uso de *openssl ca* frente a *CA.pl*, aunque este último comando es admisible. 3
2. Tarea 2: Cread una solicitud de certificado que incluya la generación de claves en la misma. 7
3. Tarea 3: Cread un certificado para la solicitud anterior empleando la CA creada en el primer punto. 8
4. Tarea 4: Cread una solicitud de certificado para cualquiera de las claves que habéis generado en las prácticas anteriores, excepto las RSA. 11
5. Tarea 5: Cread un certificado para la solicitud anterior utilizando la CA creada. 12



En esta práctica vamos a emplear OpenSSL para gestionar **Autoridades de Certificación (CA)** y **certificados X509**.

1. Tarea 1: Cread una autoridad certificadora. En este caso se premiará el uso de *openssl ca* frente a *CA.pl*, aunque este último comando es admisible.

Una **autoridad certificadora (CA)** es una **entidad de confianza** (ya sea en el ámbito público o privado) cuyos objetivos principales son: **verificar identidades** y **emitir certificados**, éste último a través de **autoridades subordinadas** que son las que realmente se encargan de emitir certificados, de forma que la entidad certificadora raíz se vea lo **menos comprometida posible**, evitando cualquier tipo de corrupción en la misma; si esto ocurriese, cualquier certificado emitido por cualquiera de las autoridades subordinadas **podría perder su validez**.

Esta tarea consiste en la creación de una CA utilizando preferiblemente el comando *openssl ca* frente al script de perl *CA.pl*. Para lograr este cometido hemos de seguir una serie de pasos que ilustraremos a continuación, basándonos principalmente en la entrada *Create the root pair*[1] y en menor medida en el Gist *How to setup your own CA with OpenSSL*[2].

1. Creamos la **estructura de directorios** sobre la cual se realizarán todas las operaciones relacionadas con los certificados. Esta estructura colgará de un directorio local:

```
mkdir -p root/ca
```

Además crearemos el resto de directorios a utilizar por CA junto con dos ficheros de texto plano relacionados con los certificados firmados, **index.txt** y **serial**.

```
mkdir certs crt newcerts private
chmod 700 private
touch index.txt
echo 1024 > serial
```

2. Copiamos el fichero de configuración de OpenSSL **openssl.cnf** que se encuentra en el path **/System/Library/OpenSSL** en nuestro path local **root/ca**. Este fichero de configuración será el que modificaremos ligeramente para añadir lo necesario. Elegimos hacer una copia en local del fichero de configuración para evitar modificaciones no deseadas que afecten a otras funcionalidades de OpenSSL.

```
cp /System/Library/OpenSSL/openssl.cnf ./root/ca/
```

3. El siguiente paso consiste en **modificar el fichero de configuración en nuestro path local** para añadir o cambiar ciertos aspectos relacionados con la autoridad de certificación. Estos aspectos se muestran a continuación, y están relacionados con distintas secciones del mismo que atienden a valores por defecto, tipos de hash utilizados o el DN (Distinguished Name)

- **Dentro de la sección [CA_default]:** ficheros, directorios y otras configuraciones por defecto que utiliza el comando *openssl ca* para la creación de certificados. Sus valores se corresponden principalmente con aquellos **directorios que creamos en el paso 1**, los cuales dictan los paths donde se almacenarán los certificados.

```
#Ficheros y directorios
dir    = /Users/Arthur18/root/ca
certs  = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
...
```

Paths donde se almacenan los **certificados y las claves privadas** de la autoridad raíz. Tanto la clave como el certificado serán verificados por la propia autoridad cuando se intente expedir un certificado.

```
#Par root key y certificado
certificate = $dir/certs/cacert.crt
private_key = $dir/private/cakey.pem
```

Funciones hash a utilizar: SHA-2. La familia de SHA-1 está obsoleta.

```
default_md = sha256
```

- **Dentro de la sección [req]:** opciones que se aplican en la creación de certificados y en las solicitudes de creación de los mismos.

```
default_bits = 2048
```

- **Dentro de la sección [req_distinguished_name]:** esta sección contiene la **información** que normalmente se **requiere a la hora de crear un certificado**. Dado que la modificación de este fichero es con fines educativos, vamos a cambiar aquellos valores por defecto que tendrá nuestra CA a la hora de la emisión de certificados, principalmente aquellos correspondientes al DN:

```
countryName_default = ES
stateOrProvinceName_default = Granada
0.organizationName_default = UGR-Universidad de Granada
emailAddress = arthur18@correo.ugr.es
```

4. Una vez modificado nuestro fichero de configuración estamos listos para **crear la root key** a través del comando **openssl genrsa**. El objetivo consiste en crear un par de claves RSA que conformarán las claves de la autoridad de certificación, utilizando la clave privada para firmar los certificados expedidos.

Los parámetros de éste comando son similares a los que utilizabamos para la práctica anterior, con la salvedad de que esta pareja de claves será **almacenada en el subdirectorio /root/ca/private**, será de **4096 bits** (buscando una cota superior que permita firmar certificados de menor tamaño) y que además estará **protegida con un cifrado AES-256** con una clave generada de forma aleatoria.(p.e: 5AFB138EC6B3DF3E81E55F6CE0FEA2BE). Tras generarla será necesario darle permisos de lectura con **chmod 400 /private/cakey.pem**.

Comando:

```
openssl genrsa -aes256 -out private/cakey.pem 4096
```

5. El último paso para crear nuestra CA es crear el denominado **root certificate** (certificado raíz) que junto con la **root key** conforman la autoridad de certificación. En este caso, lo que hemos creado es la CA raíz pero para el cometido de la práctica no haremos uso de autoridades subordinadas, por lo que será esta propia CA quien cree los certificados y los firme.

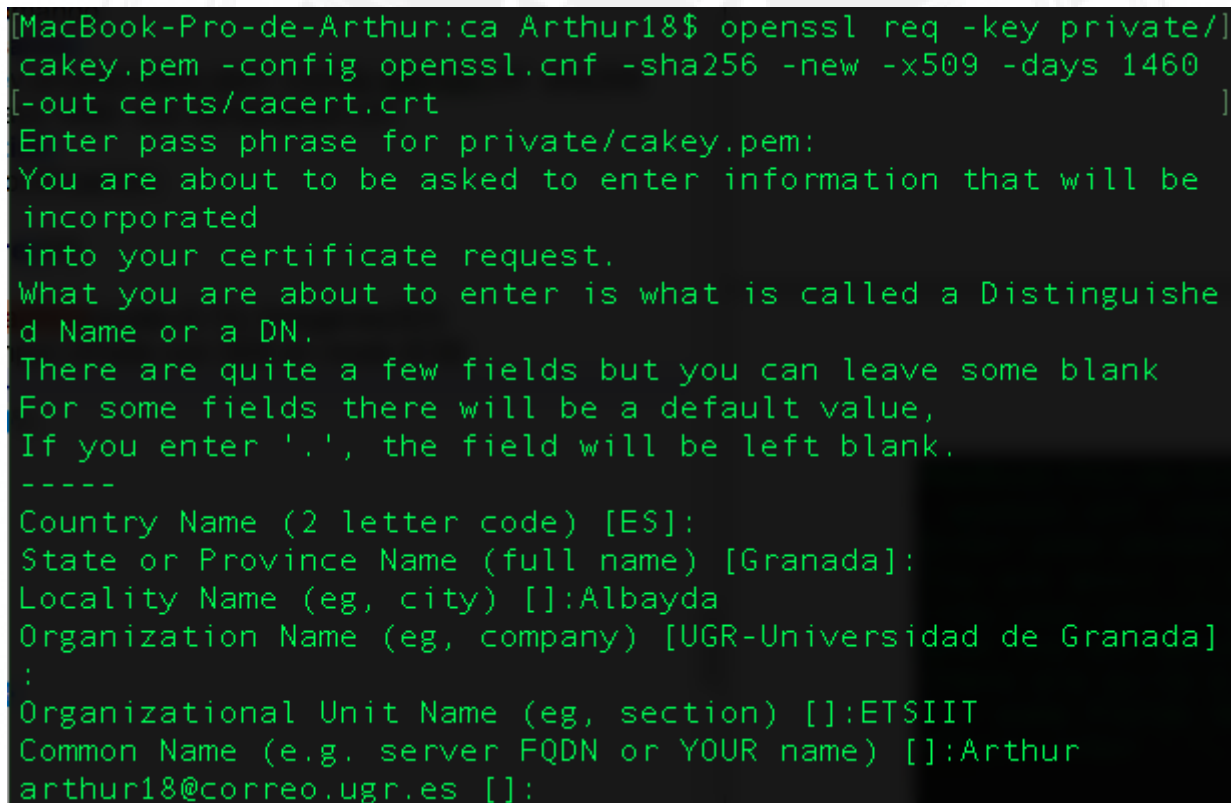
Para crear el **root certificate** haremos uso del comando `openssl req` junto con los siguientes parámetros:

- `-key`: fichero de claves RSA de nuestra CA, `cakey.pem`.
- `-config`: la familia de funciones hash que hemos especificado en el fichero de configuración.
- `-sha256`: dado que hemos especificado en el fichero de configuración que será esta la familia de funciones hash a utilizar.
- `-new`: crea una nueva solicitud de certificado, en este caso, para nuestra propia CA.
- `-x509`: esta opción permite crear un certificado *autofirmado* en vez de una solicitud en sí.
- `-days`: indica la validez del certificado. En nuestro caso pondremos 4 años (1460 días) que es la validez de un certificado emitido por la CA FNMT Usuarios.
- `-out`: el certificado `cacert.crt` que queremos obtener.

La salida del comando

```
openssl req -key private/cakey.pem -config openssl.cnf -sha256 -new -x509
-days 1460 -out certs/cacert.crt
```

se muestra a continuación.



```
[MacBook-Pro-de-Arthur:ca Arthur18$ openssl req -key private/]
cakey.pem -config openssl.cnf -sha256 -new -x509 -days 1460
[-out certs/cacert.crt ]
Enter pass phrase for private/cakey.pem:
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 Distinguishe
d 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) [ES]:
State or Province Name (full name) [Granada]:
Locality Name (eg, city) []:Albayda
Organization Name (eg, company) [UGR-Universidad de Granada]
:
Organizational Unit Name (eg, section) []:ETSIIT
Common Name (e.g. server FQDN or YOUR name) []:Arthur
arthur18@correo.ugr.es []:
```

Figura 1.1: Creación de la CA con el fichero `openssl.cnf`

A partir de este momento se considera creada nuestra autoridad de certificación, cuyo certificado mostramos en la siguiente figura. Para mostrar el certificado utilizamos el comando `openssl x509 -noout -text -in certs/cacert.crt`

```

[MacBook-Pro-de-Arthur:ca Arthur18$ openssl x509 -noout -text -in certs/cacert.crt
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      c8:8f:f3:21:ce:b1:b0:ed
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=ES, ST=Granada, L=Albayda, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur
    Validity
      Not Before: Dec 10 19:20:30 2017 GMT
      Not After : Dec  9 19:20:30 2021 GMT
    Subject: C=ES, ST=Granada, L=Albayda, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (4096 bit)

```

Figura 1.2: Certificado de la autoridad de certificación. Parte 1

```

Modulus:
  00:b3:fb:c8:1f:ec:fe:d2:9e:b6:bb:b4:cd:b0:7f:
  f6:11:5a:4a:4e:3c:e7:5d:37:db:bb:9e:1d:a7:a9:
  8b:2a:69:44:ef:a4:de:89:9e:40:41:64:2a:04:86:
  b2:a1:74:a9:a4:db:db:96:a8:18:36:0a:6a:4d:1f:
  e4:db:7a:66:2c:68:2c:fc:6c:9d:de:31:e6:4a:ad:
  4a:b4:af:91:e3:cc:22:97:14:83:cd:07:26:83:01:
  9a:84:c0:02:41:97:29:05:86:b5:20:22:5e:3b:73:
  54:a2:86:94:0b:3d:2b:d3:3e:36:dd:4a:0a:38:ae:
  b9:a1:6d:59:07:6a:be:f7:b8:72:58:de:2e:1e:b0:
  07:e2:f1:66:4d:a5:6e:39:1d:a0:f8:dc:da:95:a2:
  1f:db:ab:80:d0:c1:49:8b:0c:d7:d9:2a:ed:53:d9:
  d1:a6:16:8b:e7:56:c5:53:a1:c6:3b:b6:7c:6a:1c:
  10:c5:17:53:07:f9:28:43:cc:25:f4:ad:8c:30:15:
  33:01:d6:e7:6a:e3:58:b8:5f:0d:64:e3:14:33:3b:
  20:06:78:0b:c2:d5:ae:1d:c8:8d:13:0f:4d:fa:80:
  eb:59:ab:26:44:72:ee:cd:34:9a:33:83:69:d5:b8:
  d9:84:5b:41:7a:53:11:0c:b1:d7:af:78:97:e5:26:
  89:8f:17:ac:52:5e:c8:db:9c:9b:b9:70:05:e8:10:
  af:4c:76:0c:46:4b:92:44:79:13:03:b8:29:ce:a7:
  5b:cf:82:db:6d:28:91:a0:10:90:69:8d:1b:be:5c:
  22:1b:06:e6:67:66:7e:51:6b:f5:a5:79:5b:7d:4d:
  5e:b5:48:5b:8f:7e:ea:b0:77:66:1f:f6:70:9e:9a:
  22:b9:a1:a9:98:d2:5a:bc:46:11:59:ee:a4:96:fa:
  59:4e:c9:ca:fa:47:10:8d:32:3f:38:20:3f:9d:e4:
  17:c4:eb:12:91:2e:fe:0a:3f:39:57:3d:b8:d8:be:
  c7:5e:35:94:ac:de:dd:fc:84:94:cd:55:e8:2a:ac:
  cf:43:68:87:33:a4:3f:f1:fb:a0:73:c3:5b:e5:90:
  38:c5:f5:6d:0c:9b:6a:0a:af:2f:27:75:bf:54:5d:
  b2:cd:53:c9:3a:f8:9b:1e:0d:01:d6:58:e0:be:1d:
  93:43:3c:d0:07:60:57:9e:6f:32:7e:e5:6b:42:00:
  ec:19:87:f6:db:fc:23:73:6e:46:78:64:46:4c:88:
  43:db:8d:f4:38:f5:78:b7:a7:a5:a4:87:1e:bc:9f:
  95:f8:dd:cc:e9:22:e4:19:16:51:33:e6:a4:67:79:
  46:66:86:10:a3:13:52:51:29:06:28:a6:27:ef:36:
  a4:1c:c3
Exponent: 65537 (0x10001)

```

Figura 1.3: Certificado de la autoridad de certificación. Parte 2


```

X509v3 extensions:
  X509v3 Subject Key Identifier:
    67:1D:B9:54:84:7D:66:D7:7D:C9:C8:6F:E8:4B:0F:F5:29:D1:05:DB
  X509v3 Authority Key Identifier:
    keyid:67:1D:B9:54:84:7D:66:D7:7D:C9:C8:6F:E8:4B:0F:F5:29:D1:05:DB
    DirName:/C=ES/ST=Granada/L=Albayda/O=UGR-Universidad de Granada/OU=ETSIIT/CN=Arthur
    serial:C8:8F:F3:21:CE:B1:B0:ED

  X509v3 Basic Constraints:
    CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
6a:30:5f:2c:d7:7e:70:bc:cb:1d:ca:60:4a:f9:af:68:16:0e:
80:94:b8:c7:b3:c8:f6:84:32:5c:73:ee:38:05:fe:41:7f:d2:
20:04:34:31:9b:c5:f0:74:77:1e:15:3e:1a:f3:0d:4b:ce:e4:
1f:48:0d:b6:6c:f9:43:61:c0:cd:7e:c5:84:4c:45:b5:ea:20:
6c:06:60:e2:38:25:07:47:df:21:dc:7f:82:27:ca:d7:2e:e0:
dd:48:90:5e:9c:81:90:b0:3a:db:d9:9d:66:7d:af:af:93:69:
46:d0:0a:43:23:74:aa:ce:2d:f2:88:c6:71:a5:0d:76:28:02:
a7:a0:9d:a0:09:86:56:a1:d4:b6:a2:65:7b:ed:84:5c:32:c5:
e2:a3:cc:bf:22:b3:38:f3:13:12:c4:ed:00:4a:f7:25:ff:4a:
cc:c8:2f:8f:19:e3:08:9c:6a:51:76:20:36:cc:6d:98:a8:e1:
5b:90:cd:31:8e:b2:df:8d:e1:eb:7e:c2:b3:07:50:ad:11:59:
42:ed:68:f6:d8:f6:20:44:2f:1d:da:e5:30:64:b7:0c:ec:72:
98:30:5f:7d:7f:81:67:70:91:e4:2b:e0:ee:f2:6c:47:66:1e:
ef:cd:ff:e4:39:ee:68:9e:95:22:41:02:c5:4f:86:0a:b9:ff:
b9:5d:83:34:69:b9:f8:3f:cb:23:d6:09:e0:30:83:3b:77:fa:
e8:1c:05:d2:43:c7:f4:0e:cc:61:09:cb:fa:eb:48:6f:60:ac:
be:69:e3:21:fd:ce:ff:03:80:9b:d2:64:d7:2a:bb:6b:4f:1a:
db:9d:f1:18:29:18:c8:9e:28:02:5f:23:9f:92:f0:c0:47:e2:
32:90:ef:9c:2f:d4:41:1a:6e:62:fa:a8:cc:3a:32:0d:a6:8e:
54:96:f8:b4:64:98:79:7d:37:84:b7:55:8a:35:5a:85:f6:de:
3f:46:40:57:9f:18:35:f2:e9:06:01:2c:f7:ad:19:d0:6d:3d:
78:69:4a:79:ef:1a:5b:1c:36:e2:52:f9:a6:42:89:eb:87:62:
b4:f1:e7:8f:a1:42:c2:4a:ef:89:ec:76:da:8c:81:5f:50:05:
05:33:9a:ca:30:4b:99:68:df:7b:1b:02:de:a6:2e:e7:24:f2:
68:4a:6e:44:74:01:fb:a2:14:33:55:a0:cb:2c:c3:16:17:c6:
c9:ed:b4:a9:59:42:3c:9e:d0:09:c3:9f:6d:c3:f8:f5:d6:4d:
a9:ef:5b:e2:2d:45:8c:f4:c5:ba:a9:9d:0c:ed:eb:9b:13:24:
6c:7a:a0:c7:d3:35:23:d0:8c:05:44:2f:71:6c:4e:74:c9:30:
02:0d:d2:73:2f:dd:99:2a

```

Figura 1.4: Certificado de la autoridad de certificación. Parte 3

2. Tarea 2: Cread una solicitud de certificado que incluya la generación de claves en la misma.

Una solicitud de certificado se realiza mediante el comando `openssl req` y para que incluya generación de claves en la misma basta con **no indicarle fichero de clave RSA alguno** sino que se le especifica que cree uno a través de los siguientes argumentos: `-newkey rsa:2048` para que genere un fichero de claves RSA de 2048 bits, `-keyout private/CSRkey.pem` para indicar que queremos guardar estas claves en nuestro directorio y además cifrar la misma con la misma clave que en la tarea anterior: `5AFB138EC6B3DF3E81E55F6CE0FEA2BE`.

Comando:

```
openssl req -out newcerts/CSR1.crt -config openssl.cnf -sha256 -new -newkey
rsa:2048 -keyout private/CSRkey.pem
```

Mostramos la solicitud con el comando `openssl req -in newcerts/CSR1.crt -noout -text`.

```

[MacBook-Pro-de-Arthur:ca Arthur18$ openssl req -in newcerts/CSR1.csr -noout ]
-text
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: C=ES, ST=Granada, L=Barrio de la Cruz, O=UGR-Universidad de
Granada, OU=ETSIIT-DECSAI, CN=Arthur M Rodriguez N
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:c6:d4:8b:dc:77:9d:97:a7:b5:a1:88:35:53:40:
        59:02:eb:6f:28:14:95:ad:fb:9d:6f:ee:99:c4:47:
        53:73:3e:ed:1e:2d:40:b8:38:dc:56:58:43:29:c9:
        6a:ef:03:e5:51:e4:af:8e:42:57:fa:18:50:c7:8d:
        cf:95:65:1f:ef:d0:c2:34:02:44:fc:e0:6c:26:b7:
        23:41:b8:f2:82:97:ec:b3:8c:65:c4:19:7c:2e:8c:
        60:03:1d:ce:80:3b:d1:dc:5d:d3:c3:1a:a6:04:ac:
        8f:5f:34:78:ac:0a:67:45:be:54:58:0c:a8:79:0e:
        6a:b9:15:f1:38:43:0f:af:d8:03:8b:5e:67:e3:a3:
        90:7f:7a:96:50:aa:3c:45:a9:81:9f:1f:e5:8e:91:
        13:7e:0b:8d:44:6c:2c:87:7f:15:42:50:0c:22:bc:
        dc:38:4e:82:91:84:bb:a3:40:af:85:22:a4:84:20:
        38:ba:15:3a:a2:7d:d0:85:42:11:b4:33:71:35:73:
        56:7b:32:21:70:30:41:fd:a1:19:ec:fd:01:37:13:
        72:1d:0a:f7:7e:ec:bc:db:d9:f2:2e:d7:6d:f1:b0:
        40:6e:b8:a6:61:cc:9e:46:c9:f1:63:1f:5c:f1:a8:
        b2:82:2f:7d:12:40:5a:72:43:b1:e4:fa:e1:ea:9d:
        b1:95
      Exponent: 65537 (0x10001)
    Attributes:
      challengePassword      :buenosdias

```

Figura 2.1: Solicitud de certificado hecha a la CA

```

Signature Algorithm: sha256WithRSAEncryption
37:8d:a8:3d:79:fd:11:27:ac:23:eb:20:50:e3:d4:91:75:45:
3f:a6:fb:c8:af:47:ba:36:be:d4:d8:85:cb:fa:4f:54:eb:d4:
6f:5b:ae:2c:ea:7d:76:a7:b9:18:31:39:2a:0a:54:52:0e:bb:
4d:6a:93:98:cf:aa:50:55:d8:cf:82:2c:17:6a:35:dd:2a:cd:
ae:59:e6:c2:b1:a6:ad:ee:b8:fe:0d:ba:0a:2d:f5:ee:a2:38:
be:c4:8d:f3:8b:c0:56:39:14:d4:4f:53:55:9b:e0:fb:c4:c7:
07:9a:de:5c:79:3f:88:60:1c:69:4f:85:69:01:df:b3:84:45:
a2:40:6e:f9:95:9c:e9:34:1c:c3:99:d0:f3:c2:13:da:fe:b8:
e7:b3:c9:67:7f:92:d8:84:a4:49:73:d0:07:bb:70:61:9d:12:
da:c4:1d:19:db:4b:e5:90:45:1f:9d:b4:3b:c9:22:9b:5e:8a:
25:0e:7d:34:44:92:99:be:0d:a2:c8:ed:e4:24:16:75:ee:14:
8d:e5:7b:d9:22:a9:47:8f:92:75:b9:22:c0:fa:0f:bc:30:7f:
58:2c:0a:a3:36:dc:0a:1e:4b:7a:ca:b8:87:93:04:9a:fc:bd:
6b:c5:1a:c9:1f:74:8d:72:71:93:d9:df:6b:a6:b4:5f:b1:9c:
c8:67:ca:db

```

Figura 2.2: Solicitud de certificado hecha a la CA

3. Tarea 3: Cread un certificado para la solicitud anterior empleando la CA creada en el primer punto.

La creación de un certificado a partir de una solicitud es trivial siempre y cuando sepamos qué comando utilizar. Para ello recuperaremos el manual del comando `openssl ca` y atendiendo a los distintos argumentos que acepta construimos el siguiente comando:

```
openssl ca -config openssl.cnf -out newcerts/CSRsign.crt -infiles newcerts/CSR1.csr
```


Donde `-config` indica nuevamente el fichero de configuración sobre el que se basa la CA, el `-out` para indicar el fichero de salida (un certificado) y `-infile`s para indicar la solicitud sobre la que debe expedir un certificado. La salida del comando se puede ver a continuación:

```
MacBook-Pro-de-Arthur:ca Arthur18$ openssl ca -config openssl.cnf -out newcerts/CSRs.sign.crt -infile newcerts/CSR1.csr
Using configuration from openssl.cnf
Enter pass phrase for /Users/Arthur18/root/ca/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4132 (0x1024)
  Validity
    Not Before: Dec 10 20:29:19 2017 GMT
    Not After : Dec 10 20:29:19 2018 GMT
  Subject:
    countryName           = ES
    stateOrProvinceName   = Granada
    organizationName      = UGR-Universidad de Granada
    organizationalUnitName = ETSIIT-DECSAI
    commonName            = Arthur M Rodriguez N
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      8D:9F:42:60:FF:B5:5F:C1:07:25:66:7D:38:6B:CA:45:60:DB:BF:DC
    X509v3 Authority Key Identifier:
      keyid:67:1D:B9:54:84:7D:66:D7:7D:C9:C8:6F:E8:4B:0F:F5:29:D1:
05:DB
Certificate is to be certified until Dec 10 20:29:19 2018 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Figura 3.1: Creación de certificado para la solicitud anterior

El certificado se muestra con el comando:

```
openssl x509 -in newcerts/CSRs.sign.crt -text -noout.
```

```

MacBook-Pro-de-Arthur:ca Arthur18$ openssl x509 -in newcerts/CSRsign.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4132 (0x1024)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=ES, ST=Granada, L=Albayda, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur M
        Validity
            Not Before: Dec 10 20:29:19 2017 GMT
            Not After : Dec 10 20:29:19 2018 GMT
        Subject: C=ES, ST=Granada, O=UGR-Universidad de Granada, OU=ETSIIT-DECSAI, CN=Arthur M
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:c6:d4:8b:dc:77:9d:97:a7:b5:a1:88:35:53:40:
                59:02:eb:6f:28:14:95:ad:fb:9d:6f:ee:99:c4:47:
                53:73:3e:ed:1e:2d:40:b8:38:dc:56:58:43:29:c9:
                6a:ef:03:e5:51:e4:af:8e:42:57:fa:18:50:c7:8d:
                cf:95:65:1f:ef:d0:c2:34:02:44:fc:e0:6c:26:b7:
                23:41:b8:f2:82:97:ec:b3:8c:65:c4:19:7c:2e:8c:
                60:03:1d:ce:80:3b:d1:dc:5d:d3:c3:1a:a6:04:ac:
                8f:5f:34:78:ac:0a:67:45:be:54:58:0c:a8:79:0e:
                6a:b9:15:f1:38:43:0f:af:d8:03:8b:5e:67:e3:a3:
                90:7f:7a:96:50:aa:3c:45:a9:81:9f:1f:e5:8e:91:
                13:7e:0b:8d:44:6c:2c:87:7f:15:42:50:0c:22:bc:
                dc:38:4e:82:91:84:bb:a3:40:af:85:22:a4:84:20:
                38:ba:15:3a:a2:7d:d0:85:42:11:b4:33:71:35:73:
                56:7b:32:21:70:30:41:fd:a1:19:ec:fd:01:37:13:
                72:1d:0a:f7:7e:ec:bc:db:d9:f2:2e:d7:6d:f1:b0:
                40:6e:b8:a6:61:cc:9e:46:c9:f1:63:1f:5c:f1:a8:
                b2:82:2f:7d:12:40:5a:72:43:b1:e4:fa:e1:ea:9d:
                b1:95
            Exponent: 65537 (0x10001)

```

Figura 3.2: Certificado creado para la solicitud anterior

```

X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        8D:9F:42:60:FF:B5:5F:C1:07:25:66:7D:38:6B:CA:45:60:DB:BF:DC
    X509v3 Authority Key Identifier:
        keyid:67:1D:B9:54:84:7D:66:D7:7D:C9:C8:6F:E8:4B:0F:F5:29:D1:05:DB

Signature Algorithm: sha256WithRSAEncryption
    28:f9:94:85:74:70:f5:57:54:3f:33:5e:05:41:08:72:5d:e5:
    6c:36:a8:36:6d:05:e5:6d:02:7d:64:5f:d0:16:54:fc:1d:61:
    c8:72:24:fd:ee:d0:f1:27:c4:62:0b:4c:ce:44:84:21:e9:d0:
    16:c8:dd:e9:00:f3:9b:c5:4d:11:64:9f:15:69:8f:f6:0a:22:
    e0:b1:37:8f:ec:c7:e1:3c:0f:d1:27:0a:f7:02:8e:06:18:04:
    f9:f6:92:91:78:0f:f5:a1:a6:fe:64:7f:4e:78:2c:82:9a:b3:
    aa:4f:88:13:3a:5b:44:3c:b0:57:65:4b:8d:72:b3:e5:14:5f:
    eb:e6:c7:62:1e:b9:f9:02:ed:ba:9c:86:45:0e:c3:57:18:6b:
    20:4b:9c:c4:45:6f:84:f4:98:be:d2:1f:78:22:48:51:6f:18:
    78:f8:f4:c7:62:e5:a0:fc:e7:6f:23:5d:40:bf:d2:9f:38:d6:
    76:54:f4:39:50:cf:ba:8e:f1:b5:54:71:f7:08:01:cb:5f:41:
    17:26:ad:28:56:a7:13:b3:77:0f:2c:ea:e8:7a:86:16:d8:86:
    4e:f9:7e:40:6d:dc:90:ba:0a:7a:ba:76:32:f9:92:76:11:c9:
    46:b3:69:49:d9:4d:b2:a9:4d:12:57:f8:a1:0d:f4:43:bf:a5:
    7b:43:65:16:27:40:0d:d0:8e:34:4a:21:1a:bf:e8:e0:c3:6e:
    49:ae:87:23:3e:25:96:33:b0:fd:0f:f4:3c:cf:2d:f2:f2:31:
    00:e6:dd:0d:6d:89:c5:e2:dd:9e:ca:2b:72:d8:93:62:d9:e1:
    24:84:b5:eb:6c:29:96:8b:21:7d:6b:4f:b6:08:f3:c9:79:10:
    28:c8:dd:82:95:c0:35:51:1d:ee:71:78:4b:a5:68:e8:71:91:
    4e:93:25:c5:c4:34:f6:3e:21:4d:0e:f3:23:f4:bf:33:e6:d9:
    af:49:51:3d:09:60:1e:e3:1e:df:89:2b:48:79:cc:2e:55:2c:
    30:8b:0a:92:cf:56:89:44:6f:f1:e6:76:98:3a:77:9e:a9:14:
    5a:ce:49:da:4e:de:74:0a:5d:9e:6e:93:af:bc:a8:b5:09:2c:
    70:be:97:d2:5f:ce:00:c8:cb:80:a1:7b:ae:a1:a9:a9:2f:9d:
    56:84:eb:f9:c6:c3:dd:83:eb:e8:11:c8:61:d4:da:bc:ea:2e:
    d6:56:c5:44:9a:7c:1f:56:6c:12:f8:02:8f:d7:56:2b:2e:24:
    da:ef:73:ff:2e:46:94:04:90:11:c6:0f:1a:52:d0:bd:60:47:
    52:13:95:53:00:f5:13:55:6d:43:a7:a4:26:9a:7c:b6:71:40:
    a6:95:6d:e0:55:0d:54:df

```

Figura 3.3: Certificado creado para la solicitud anterior

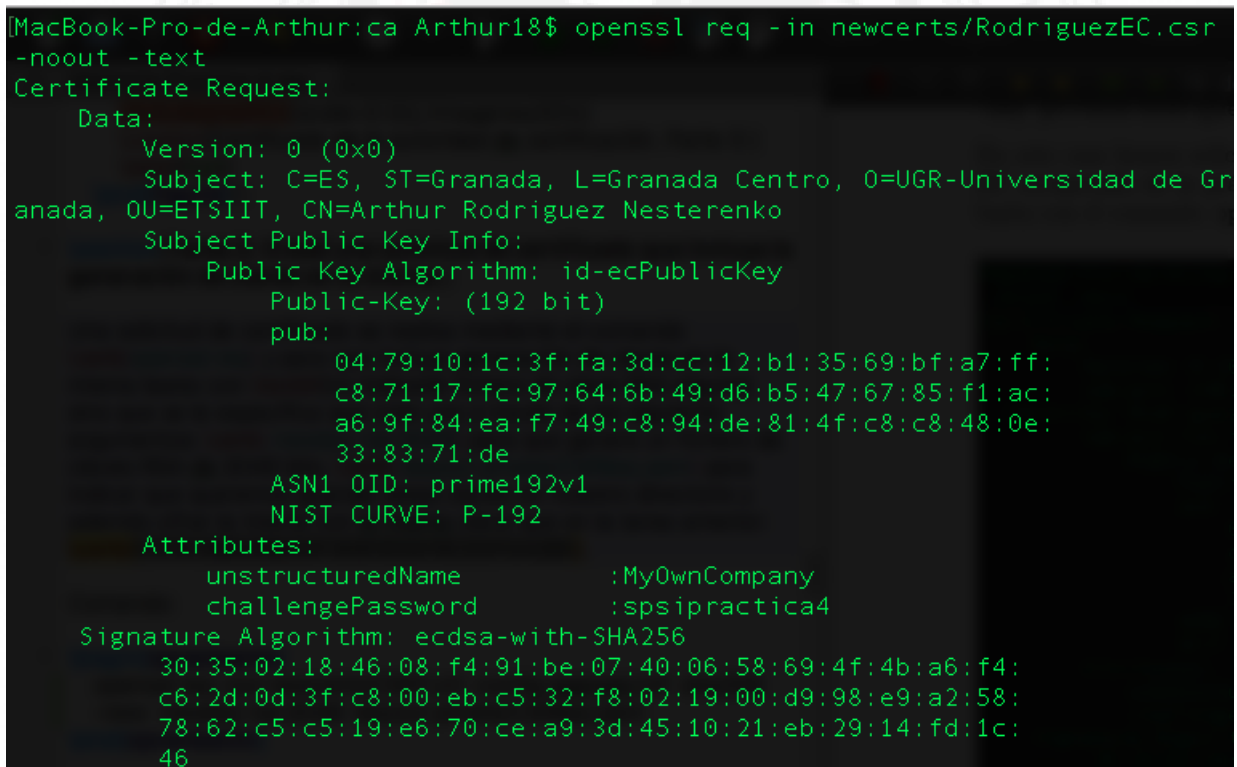
4. Tarea 4: Cread una solicitud de certificado para cualquiera de las claves que habéis generado en las prácticas anteriores, excepto las RSA.

Crear una solicitud para claves generadas en practicas anteriores es sencillo. Basta con recuperar los ficheros de parejas de claves, cualquiera excepto de RSA, y reconfigurar el comando de la tarea 2 para que acepte un fichero de claves de entrada y genere una solicitud.

Recuperamos el fichero de claves **RodriguezECkey.pem** que contiene la pareja de claves generada a partir de los parametros de una EC. Por último configuramos el comando para generar la solicitud de certificado:

```
openssl req -out newcerts/RodriguezEC.csr -config openssl.cnf -sha256 -new -key private/RodriguezECkey.pem -days 720
```

En este caso hemos solicitado un certificado por 720 dias a partir del fichero de claves EC recuperado de la práctica anterior. Para ver la solicitud que acabamos de realizar basta con el comando: `openssl req -in newcerts/RodriguezEC.csr -noout -text`.



```
[MacBook-Pro-de-Arthur:ca Arthur18$ openssl req -in newcerts/RodriguezEC.csr -noout -text
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: C=ES, ST=Granada, L=Granada Centro, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur Rodriguez Nesterenko
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (192 bit)
      pub:
        04:79:10:1c:3f:fa:3d:cc:12:b1:35:69:bf:a7:ff:
        c8:71:17:fc:97:64:6b:49:d6:b5:47:67:85:f1:ac:
        a6:9f:84:ea:f7:49:c8:94:de:81:4f:c8:c8:48:0e:
        33:83:71:de
      ASN1 OID: prime192v1
      NIST CURVE: P-192
    Attributes:
      unstructuredName          :MyOwnCompany
      challengePassword         :spsipractica4
  Signature Algorithm: ecdsa-with-SHA256
    30:35:02:18:46:08:f4:91:be:07:40:06:58:69:4f:4b:a6:f4:
    c6:2d:0d:3f:c8:00:eb:c5:32:f8:02:19:00:d9:98:e9:a2:58:
    78:62:c5:c5:19:e6:70:ce:a9:3d:45:10:21:eb:29:14:fd:1c:
    46
```

Figura 4.1: Solicitud de certificado para el fichero de claves recuperado (Curva utilizada: P-192)

5. Tarea 5: Cread un certificado para la solicitud anterior utilizando la CA creada.

El paso final consiste en crear un certificado para esta solicitud, acción análoga a la realizada en la tarea 3, por lo que basta con recuperar el comando:

```
openssl ca -config openssl.cnf -out newcerts/RodriguezEC.crt  
-infiles newcerts/RodriguezEC.csr
```

El certificado se puede ver con el siguiente comando:

```
openssl x509 -in newcerts/RodriguezEC.crt -text -noout
```

```
MacBook-Pro-de-Arthur:ca Arthur18$ openssl x509 -in newcerts/RodriguezEC.crt -text -noout  
Certificate:  
  Data:  
    Version: 3 (0x2)  
    Serial Number: 4133 (0x1025)  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: C=ES, ST=Granada, L=Albayda, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur  
    Validity  
      Not Before: Dec 10 21:02:27 2017 GMT  
      Not After : Dec 10 21:02:27 2018 GMT  
    Subject: C=ES, ST=Granada, O=UGR-Universidad de Granada, OU=ETSIIT, CN=Arthur Rodriguez  
    Nesterenko  
    Subject Public Key Info:  
      Public Key Algorithm: id-ecPublicKey  
      Public-Key: (192 bit)  
      pub:  
        04:79:10:1c:3f:fa:3d:cc:12:b1:35:69:bf:a7:ff:  
        c8:71:17:fc:97:64:6b:49:d6:b5:47:67:85:f1:ac:  
        a6:9f:84:ea:f7:49:c8:94:de:81:4f:c8:c8:48:0e:  
        33:83:71:de  
      ASN1 OID: prime192v1  
      NIST CURVE: P-192  
    X509v3 extensions:  
      X509v3 Basic Constraints:  
        CA:FALSE  
      Netscape Comment:  
        OpenSSL Generated Certificate  
      X509v3 Subject Key Identifier:  
        C5:74:76:D4:EF:52:39:8C:96:61:A5:DB:77:A0:2D:7A:65:6E:45:23  
      X509v3 Authority Key Identifier:  
        keyid:67:1D:B9:54:84:7D:66:D7:7D:C9:C8:6F:E8:4B:0F:F5:29:D1:05:DB
```

Figura 5.1: Certificado creado para la solicitud anterior **RodriguezEC.csr**

```
Signature Algorithm: sha256WithRSAEncryption
aa:62:48:07:a2:e1:02:39:9c:33:9d:d9:82:e1:3a:f4:42:ea:
4b:42:2b:aa:2b:ec:55:79:81:c1:2a:61:1a:e4:8e:db:91:bd:
bb:74:29:57:4b:66:e5:69:ef:52:1f:2d:d4:a4:ad:64:40:b1:
c2:f3:4a:de:e9:72:b6:c1:04:81:36:2d:99:07:5e:f6:f0:39:
c2:29:32:64:3c:71:30:cb:1a:3d:05:ff:d6:24:06:c1:5a:d4:
f2:6f:47:8e:02:31:59:0d:51:48:3d:c2:b7:7b:7f:50:d8:6a:
1f:e6:a2:f2:22:dc:98:f8:d9:57:22:f1:5a:7f:16:36:ac:20:
e7:1f:52:12:8a:78:5b:d0:f5:1f:08:8e:48:99:a9:dc:4c:66:
af:9d:0e:3a:bb:52:c1:d1:13:93:a9:96:42:be:9c:de:f6:19:
8b:c5:1f:06:fe:dc:96:77:fe:63:a2:2f:50:64:7d:9b:c6:d6:
35:a2:fa:3c:f3:82:12:ec:d2:ff:01:f5:77:f7:94:50:e9:75:
16:c8:61:b3:53:49:94:71:bd:db:ae:1c:3f:82:32:0d:6a:32:
b7:fe:76:36:58:3a:51:cc:1f:b5:67:b6:da:81:bc:2f:9b:6d:
23:d1:eb:3d:6d:99:f2:bb:5f:79:8c:7f:ba:35:40:7d:e1:09:
39:e7:fd:5c:4b:28:d6:fb:cd:bd:18:4f:6c:ea:ea:16:1f:06:
d8:d7:4b:f8:b5:7f:92:63:fb:a5:8c:7d:53:85:b2:a5:72:13:
03:ae:be:5d:6a:3b:8b:64:cc:e6:34:3f:b1:3c:f0:a7:87:a6:
d2:76:d9:e5:ed:c4:85:3f:4f:19:d8:f5:f2:1b:d9:18:86:e8:
ca:b4:38:4c:65:66:8b:56:ec:29:cf:6c:99:54:db:c7:5e:b3:
c0:77:a8:30:da:6b:fe:23:5c:d7:46:ee:33:39:82:11:23:05:
87:1a:04:b1:cc:a7:02:9d:38:8e:f4:2e:3f:92:f8:9f:55:c2:
50:c5:15:d3:85:da:66:3a:95:d0:6f:3a:04:89:d9:ae:83:52:
ef:3c:6d:d7:78:d8:90:ff:ad:ce:97:5b:99:7b:5d:13:ec:5d:
e1:a3:ff:4f:af:11:8d:49:c1:6c:9f:55:34:01:f7:dc:c4:a2:
6e:41:be:ac:31:34:63:30:ae:e4:aa:3a:aa:f4:0e:d6:60:be:
04:b9:4f:cd:78:08:66:17:32:f4:37:20:48:7b:d9:8f:35:6f:
5d:1b:0f:62:b1:04:38:c1:3f:d7:b0:05:aa:bb:63:95:9e:63:
6f:01:ae:73:dd:12:4c:5e:0f:1f:8a:aa:2c:43:4c:8b:c7:b8:
76:00:68:6f:a0:ad:12:e9
```

Figura 5.2: Certificado creado para la solicitud anterior **RodriguezEC.csr**

Referencias

- [1] <https://jamielinux.com/docs/openssl-certificate-authority/create-the-root-pair.html>, Create the root pair.
- [2] <https://gist.github.com/Soarez/9688998>, How to setup your own CA with OpenSSL.