

Universidad del País Vasco

FACULTAD DE INFORMÁTICA

SEGURIDAD, RENDIMIENTO Y
DISPONIBILIDAD DE SERVICIOS E
INFRAESTRUCTURAS

Proyecto PKI

Autores:
Gorka Álvarez
Jon Gámiz
Asier Zubia

Donostia, a 4 de Marzo de 2021

Índice

1. Descripción del Proyecto	2
2. Desarrollo del Proyecto	2
2.1. Creación de la CA Raíz	2
2.2. Creación de la CA subordinada	3
2.3. Emisión de un Certificado	4
2.4. Revocación de un Certificado	5
2.5. Validación de un Certificado con OCSP responder	5
2.6. Exportar un Certificado y Clave Privada	7
2.7. Verificación de Certificados en Base a la CRL	7
2.8. Creación del entorno web	8
3. Root-CA conf_file.cnf	9
4. Intermed-CA con_file.cnf	11

Lista de Scripts

1. Root-CA Conf_file.cnf	10
2. Intermed-CA Conf_file.cnf	12

Índice de figuras

1. Árbol de directorios de la CA Raíz	2
2. RootCa Certificate Signing Request	3
3. Certificado autofirmado de la Ca raíz	3
4. Certificado de la CA subordinada	4
5. Verificación del certificado de la CA subordinada	4
6. Certificado firmado para el User1	5
7. Error de inicio del cliente OCSP	6
8. Error en la validación de la cadena de certificados	6
9. Funcionamiento correcto utilizando el User 1	6
10. Funcionamiento correcto utilizando el User 2	7
11. Exportación del certificado y la clave privada del User 1 a un fichero con formato PKCS#12	7
12. Extracción de clave y certificado de un fichero .p12	7
13. Verificación de User1 frente a la CRL	8

1. Descripción del Proyecto

El objetivo del proyecto es consolidar los conceptos trabajados en clase sobre los certificados digitales. Por tanto, la finalidad del proyecto será crear una PKI¹ simple compuesta por una CA² que permita emitir, revocar y verificar certificados a personas o servidores de una red privada.

Las horas totales dedicadas a la realización del proyecto han sido **10 horas**.

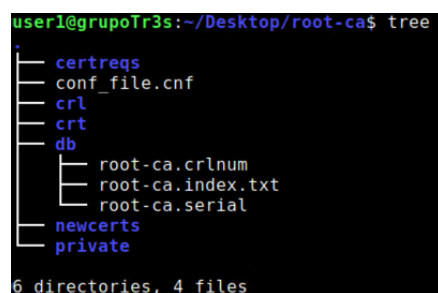
2. Desarrollo del Proyecto

En esta sección se describirán las tareas realizadas en el proyecto así como la resolución de las mismas. Todo el proyecto, entorno web, junto con certificados, estructura completa de la PKI y maquina virtual que aloja la estructura, queda accesible en [GitHub](#).

En cuanto a las claves de los certificados, todos están protegidos con una clave que es su mismo nombre, es decir, la contraseña de la clave correspondiente al certificado *root-ca.crt.pem* es *root-ca*, la del *ocsp-server.crt.pem* es *ocsp-server*, para *user1.crt.pem* es *user1* y así sucesivamente.

2.1. Creación de la CA Raíz

En primer lugar, antes de poder realizar nada, debemos crear toda la estructura de trabajo de la CA.



```
user1@grupoTr3s:~/Desktop/root-ca$ tree
.
├── certreqs
│   └── conf_file.cnf
├── crl
├── crt
├── db
│   ├── root-ca.crlnum
│   ├── root-ca.index.txt
│   └── root-ca.serial
├── newcerts
└── private

6 directories, 4 files
```

Figura 1: Árbol de directorios de la CA Raíz

Como se ve en la figura 1 ya se han creado algunos de los ficheros necesarios para poner en marcha la CA. El primero *conf_file.cnf*, ver [3. Root-CA conf file.cnf](#), contiene la configuración necesaria para garantizar el correcto funcionamiento de nuestra CA raíz. Dentro del directorio *db*, *Data Base*, se encuentran los archivos relacionados con la gestión de números de serie de los certificados así como el fichero *root-ca.index.txt* que contiene el registro de los certificados válidos, revocados y expirados.

Una vez tengamos el entorno de trabajo de la CA creado, procederemos a construir la propia CA. El primer paso será crear una clave privada para nuestra CA, la cual almacenaremos en la carpeta *private*, a la cual le hemos limitado los permisos para que solo pueda acceder el propietario. Para crear la clave privada hemos ejecutado el comando `openssl genrsa -des3 -out private/root-ca.key 2048`, con clave *root-ca*. Este comando genera una clave con el algoritmo triple Des y con tamaño de clave 2048b. Para garantizar una mayor seguridad de la clave, se le han limitado los permisos a la misma, dejando únicamente los permisos de lectura para el propietario.

El siguiente paso será crear el *Certificate Signing Request* o CSR de la CA, el cual posteriormente se autofirmará. Para ello ejecutaremos el comando `openssl req -new -key private/root-ca.key -out certreqs/root-ca.csr -config conf_file.cnf`, el cual nos generará el siguiente fichero.

¹Public Key Infrastructure

²Certificate Authority

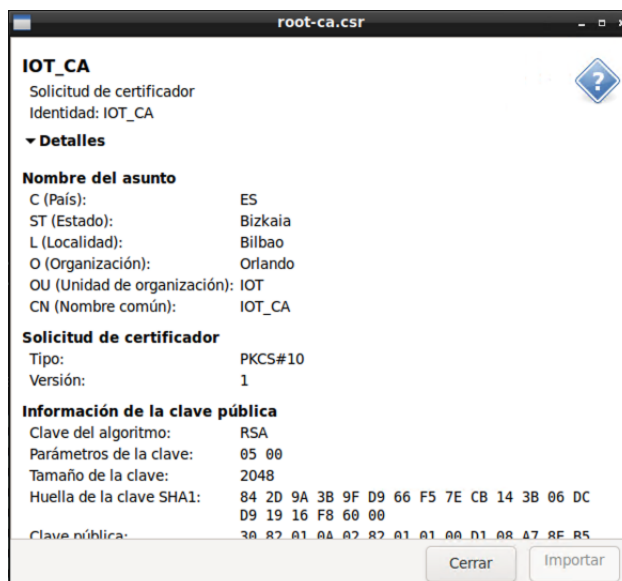


Figura 2: RootCa Certificate Signing Request

Puesto que se trata del certificado de la CA raíz, el certificado debe ser autofirmado. Para realizar esta firma ejecutaremos el comando `openssl ca -selfsign -in certreqs/root-ca.csr -out crt/root-ca.crt.pem -config conf_file.cnf -extensions ca_ext`. El certificado resultante es el siguiente.

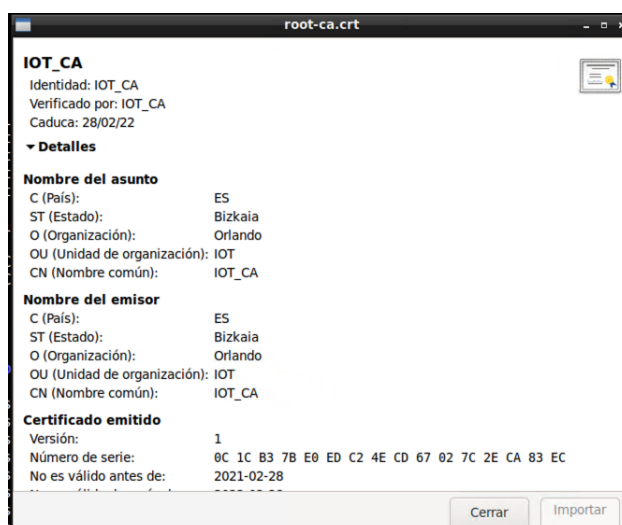


Figura 3: Certificado autofirmado de la Ca raíz

De esta manera, ya tendremos nuestra CA lista para emitir certificados.

2.2. Creación de la CA subordinada

Para esta tarea se pide crear una CA subordinada, la cual, firmada y aprobada por la CA raíz, permite firmar certificados a los usuarios o paginas web creando de esta manera una cadena de certificación.

Para poder crear esta CA intermedia, lo primero que haremos será crear el entorno en el que trabajará esta CA. La estructura de los directorios y ficheros de configuración será la misma que la de nuestra CA raíz, ver figura 1.

La diferencia con respecto a la CA raíz es que, obviamente, el certificado final no será autofirmado, sino que será firmado por la CA raíz. Esto lo haremos utilizando el comando `openssl ca -in certreqs/intermed-ca.csr -out crt/intermed-ca.crt.pem -config conf_file.cnf -extensions v3_intermediate_ca`. Cabe destacar que la extension que se le aplica a la CA intermedia no es la misma que la que se aplica a la CA raíz, para ver las diferencias entre ambas ver [Root-CA conf_file.cnf](#). El certificado resultante.

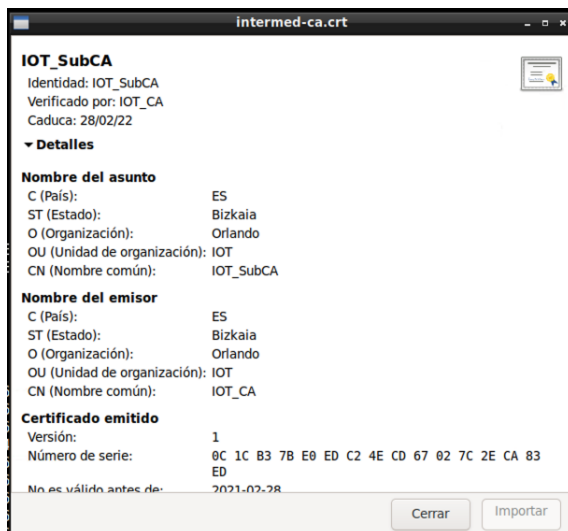


Figura 4: Certificado de la CA subordinada

Para verificar que efectivamente la el certificado es válido y que está respaldado por la CA raíz, podemos ejecutar el siguiente comando `openssl verify -verbose -CAfile crt/root-ca.crt crt/intermed-ca.crt` el cual devuelve el siguiente resultado.

```
user1@grupoTr3s:~/Desktop/PKI/root_ca$ openssl verify -verbose -CAfile crt/root-ca.crt crt/intermed-ca.crt
crt/intermed-ca.crt: OK
```

Figura 5: Verificación del certificado de la CA subordinada

2.3. Emisión de un Certificado

A partir de aquí todos los pasos se realizan con la CA subordinada, por lo que se tratará a esta CA como la CA raíz.

Ahora que tenemos lista nuestra CA ya podemos empezar a firmar certificados, pero antes, necesitamos que los usuarios o servidores nos envíen su solicitud de firma de certificado (CSR). El usuario o servidor, creará su clave privada y solicitud con el comando `openssl req -out csr/user1.csr -new -newkey rsa:2048 -keyout private_keys/user1.key`, dando como resultado el `.csr` que enviará a la CA.

La CA, validará los datos enviados por el usuario y en caso de que sean válidos firmará el certificado utilizando el comando `openssl ca -in certreqs/user1.csr -out crt/user1.crt -config conf_file.cnf -extensions client_ext`. Cabe destacar, que la extensión aplicada al certificado es `client_ext`, si el certificado fuese a ser emitido para un servidor se utilizaría la extensión `server_ext`.

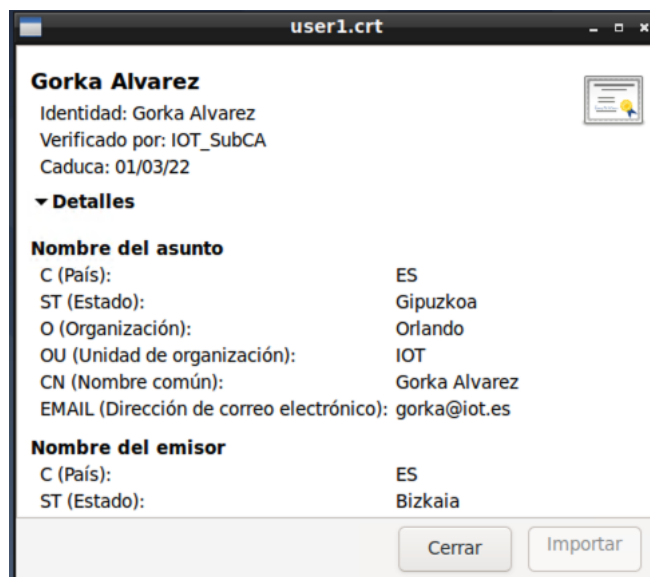


Figura 6: Certificado firmado para el User1

2.4. Revocación de un Certificado

Nuestra autoridad de certificación, a parte de crear certificados, también debe poder revocar certificados emitidos por la razón que sea. Para ello, deberá ejecutar el comando `openssl ca -revoke crt/user1.crt -crl_reason superseded -config conf_file.cnf`. A través del parámetro *crl_reason*, la CA puede especificar el motivo por el que se le ha revocado el certificado al usuario, siendo los posibles motivos los siguientes.

- **Unspecified.** No se especifica ninguna razón concreta que justifique la revocación del certificado.
- **KeyCompromise.** La clave privada del usuario se ha visto comprometida, por lo que su certificado puede ser usado ilícitamente.
- **CACompromise.** La clave de la CA se ha visto comprometida, por lo que su certificado, y todos los certificados emitidos deben dejar de ser válidos.
- **AffiliationChanged.** El usuario ya no pertenece a la misma afiliación, concretamente a la proporcionada en el campo *Distinguished Name*, por lo que su certificado deja de ser válido.
- **Superseded.** Se ha reemplazado el certificado de un usuario, por lo que su certificado viejo deja de ser válido. Una de las razones más comunes son un cambio de contraseña.
- **CessationOfOperation.** En caso de que la CA se deje de utilizar su certificado se revocará con este código.
- **CertificateHold.** Se revoca el certificado debido a una retención del mismo.
- **RemoveFromCRL.** Imposibilita la opción de desrevocar el certificado.

La CA cada X tiempo publicará la lista de certificados revocados, con el objetivo de tenerla lo más actualizada posible. Para poder obtener esta lista, la CA ejecutará el comando `openssl ca -gencrl -out crl/intermed-ca.der -config conf_file.cnf`

2.5. Validación de un Certificado con OCSP responder

Para la validación de un Certificado con OCSP responder hace falta realizar cuatro pasos.

En el primer paso deberemos crear la clave privada junto con el *csr* para el servidor OSCP. Para ello hemos ejecutado el siguiente comando `openssl req -out csr/ocsp-server.csr -new -newkey rsa:2048 -keyout private_keys/ocsp-server.key`

El segundo paso a realizar sería firmar el certificado desde la CA Intermedia. Por tanto, ejecutamos el siguiente comando `openssl ca -in certreqs/ocsp-server.csr -out crt/ocsp-server.crt -config conf_file.cnf -extensions ocsp_ext`. Destacar que en este caso se utiliza la extensión *ocsp_ext*.

Una vez generada las claves y firmado el certificado hemos procedido a iniciar el servidor OSCP. Hemos ejecutado el comando `openssl ocsp -port 12000 -index db/intermed-ca.index.txt -CA crt/intermed-ca.crt -rsigner crt/ocsp-server.crt -rkey private/ocsp-server.key -text -out log.txt`

Por último, se han realizado dos pruebas para la conexión del cliente con el servidor OSCP, uno de ellos, User 2, con un Certificado que había sido revocado.

En el primer intento se introdujo mal la URL donde se encontraba el servidor mediante el comando `openssl ocsp -CAfile crt/intermed-ca.crt -issuer crt/intermed-ca.crt -url http://ocsp.grupoTr3s:12000 -cert crt/user1.crt -resp_text`, que nos dio el siguiente error.

```
user1@grupoTr3s:~/Desktop/PKI/intermed-ca$ openssl ocsp -CAfile crt/intermed-ca.crt -issuer crt/intermed-ca.crt -url http://ocsp.grupoTr3s:12000 -cert crt/user1.crt -resp_text
Error connecting BIO
Error querying OSCP responder
139893903467648:error:2008F002:BIO routines:BIO_lookup_ex:system lib:../crypto/bio/b_addr.c:724:Name or service not known
```

Figura 7: Error de inicio del cliente OSCP

Una vez corregido la URL donde se encuentra el servidor, hemos procedido a volver a ejecutar el comando `openssl ocsp -CAfile crt/intermed-ca.crt -issuer crt/intermed-ca.crt -url http://localhost:12000 -cert crt/user1.crt -resp_text`. En esta ocasión nos encontramos un nuevo error, pero esta vez era debido a un error en la validación en la cadena de certificados.

```
Response Verify Failure
139828776957056:error:27069065:OCSP routines:OCSP_basic_verify:certificate verify error:../crypto/ocsp/ocsp_vfy.c:93:Verify error:
unable to get issuer certificate
139828776957056:error:27069065:OCSP routines:OCSP_basic_verify:certificate verify error:../crypto/ocsp/ocsp_vfy.c:93:Verify error:
unable to get issuer certificate
crt/user2.crt: revoked
This Update: Mar 1 11:18:26 2021 GMT
Reason: superseded
Revocation Time: Feb 28 13:45:46 2021 GMT
```

Figura 8: Error en la validación de la cadena de certificados

Tras actualizar el parámetro *CAfile* sustituyéndolo por el que contenía nuestra cadena, *root-intermed-ca.crt.pem*, ejecutamos el comando `openssl ocsp -CAfile crt/root-intermed-ca.crt.pem -issuer crt/intermed-ca.crt -url http://localhost:12000 -cert crt/user1.crt -resp_text`.

```
crt/user1.crt: good
This Update: Mar 1 11:26:31 2021 GMT
```

Figura 9: Funcionamiento correcto utilizando el User 1

```
crt/user2.crt: revoked
This Update: Mar  1 11:25:30 2021 GMT
Reason: superseded
Revocation Time: Feb 28 13:45:46 2021 GMT
```

Figura 10: Funcionamiento correcto utilizando el User 2

2.6. Exportar un Certificado y Clave Privada

En cuanto a la exportación de un certificado y clave privada del *User 1* hemos ejecutado el siguiente comando `openssl pkcs12 -export -out user1.p12 -in crt/user1.crt -inkey ../users/private_keys/user1.key` obteniendo como resultado un fichero en formato PKCS#12, utilizando su certificado y su clave.

```
user1@grupoTr3s:~/Desktop/PKI/intermed-ca$ openssl pkcs12 -export -out user1.p12 -in crt/user1.crt -inkey ../users/private_keys/user1.key
Enter pass phrase for ../users/private_keys/user1.key:
Enter Export Password:
Verifying - Enter Export Password:
```

Figura 11: Exportación del certificado y la clave privada del User 1 a un fichero con formato PKCS#12

Una vez hecho eso, para extraer el certificado y la clave del fichero `pks12` que acabamos de crear, hemos ejecutado el siguiente comando `openssl pkcs12 -in user1.p12 -out crt/user1_v2.crt`. Después de ejecutar el comando nos pide la contraseña de importación, la frase contraseña de la *PEM* y por último repetir la frase contraseña de la *PEM* que acabamos de introducir. Una vez hecho eso se nos genera perfectamente el archivo `.crt`. Sin embargo, no todo lo que reluce es oro, y nos hemos encontrado con que no podíamos visualizar la clave privada introduciendo la contraseña que habíamos establecido. Esto se debe a un *bug* del visor de contraseñas de *Debian*.

```
user1@grupoTr3s:~/Desktop/PKI/intermed-ca$ openssl pkcs12 -in user1.p12 -out crt/user1_v2.crt
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Figura 12: Extracción de clave y certificado de un fichero `.p12`

Para intentar descubrir cual ha sido el problema tratamos de exportar el certificado y la clave del fichero `.p12` por separado. Para extraer únicamente el certificado hemos ejecutado el siguiente comando `openssl pkcs12 -nokeys -in user1.p12 -out crt/user1_v2.crt`, generando como resultado el certificado sin ningún tipo de problema.

A continuación hemos extraído únicamente la clave haciendo uso del comando `openssl pkcs12 -nocerts -in user1.p12 -out crt/user1_v2.pem` generando un fichero *PEM*. Al tratar de abrir este fichero con la clave seguíamos teniendo el mismo problema de antes. Sin embargo al visualizar la clave por consola, la podíamos ver sin ningún problema, concluyendo en el *bug* antes mencionado.

2.7. Verificación de Certificados en Base a la CRL

Como último servicio ofrecido por la CA, se podrá realizar la verificación de un certificado con respecto a la CRL (*Certificate Revocation List*), para ello lo primero será obtener la lista de certificados revocados, el comando es el mismo que el utilizado en la sección 2.4. [Revocación de un Certificado](#), y que recordemos que era `openssl ca -gencrl -out crl/intermed-ca.der -config conf_file.cnf`.

Una vez obtenida la última versión de la CRL, ejecutaremos el comando `openssl verify -crl_check -CRLfile crl/intermed-ca.der -CAfile crt/root-intermed-ca.crt.pem crt/user1.crt.pem`, donde `root-intermed-ca.crt.pem` es el fichero que contiene la cadena de verificación y `user1.crt.pem` el certificado a validar frente a la CRL.

Puesto que el certificado del *User 1* había sido revocado con anterioridad, al realizar la validación nos da error.


```
user1@grupo1r3s:~/Desktop/PKI/intermed-ca$ openssl verify -crl_check -CRLfile crt/intermed-ca.der -CAfile crt/root-intermed-ca.crt.pem crt/user1.crt.pem
C = ES, ST = Gipuzkoa, O = Orlando, OU = IOT, CN = Gorka Alvarez, emailAddress = gorka@iot.es
error 23 at 0 depth lookup: certificate revoked
error crt/user1.crt.pem: verification failed
```

Figura 13: Verificación de User1 frente a la CRL

2.8. Creación del entorno web

El entorno web destinado realizar todas las tareas antes descritas queda accesible a través del siguiente [enlace](#). Para poder acceder a las tareas de la CA o de la CA subordinada, se puede acceder con el usuario *admin* y contraseña *admin*. Para acceder a las funciones de los usuarios se puede realizar un breve registro y posteriormente iniciar sesión, o también se puede acceder mediante el usuario *user* y contraseña *user*.

Cabe destacar que las tareas que puede realizar la CA en este entorno web son aquellas que puede realizar la CA subordinada, es decir, el certificado de descarga, la emisión de certificados y ver la lista de certificados revocados, entre otras cosas, son utilizando la CA subordinada como la CA raíz.

3. Root-CA conf_file.cnf

```
### ca_openssl.cnf ###
[default]
name           = grupo3-CA
domain_suffix  = grupoTr3s
aia_url        = http://$name.$domain_suffix/$name.crt
crl_url        = http://$name.$domain_suffix/$name.crl
ocsp_url       = http://ocsp.$domain_suffix:12000
default_ca     = CA_default # Sección por defecto para CA
name_opt       = utf8,esc_ctrl,multiline,lname,align

[ca_dn]
countryName      = ES
stateOrProvinceName = Bizkaia
localityName     = Bilbao
organizationName  = Orlando
organizationalUnitName = IOT
commonName       = IOT_CA

[CA_default]
home           = /home/user1/Desktop/PKI/root_ca
db             = $home/db
privado        = $home/private
database       = $db/root-ca.index.txt
serial        = $db/root-ca.serial
crlnumber      = $db/root-ca.crlnum
certificate     = $home/crt/root-ca.crt.pem
private_key    = $privado/root-ca.key
crl            = $db/crl/root-ca.der          # (DER format)
RANDFILE       = $privado/.rand
new_certs_dir  = $home/newcerts
unique_subject = no
copy_extensions = none
default_days   = 365
default_crl_days = 30
default_md     = sha256
policy         = policy_c_o_match

# Política para la CA
[policy_c_o_match]
countryName      = match
stateOrProvinceName = optional
organizationName  = match
organizationalUnitName = optional
commonName       = supplied
emailAddress      = optional

[req]
default_bits      = 2048          # 4096
encrypt_key       = yes
default_md        = sha256
utf8              = yes
string_mask       = utf8only
```

```
prompt                = no
distinguished_name    = ca_dn
req_extensions        = ca_ext
```

```
[ca_ext]
basicConstraints      = critical,CA:true
keyUsage              = critical,keyCertSign,cRLSign, digitalSignature
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
```

```
[crl_info]
URI.0                = $crl_url
```

```
[issuer_info]
caIssuers;URI.0      = $aia_url
OCSP;URI.0           = $ocsp_url
```

```
[name_constraints]
permitted;DNS.0=$name
permitted;DNS.1=$name
excluded;IP.0=0.0.0.0/0.0.0.0
excluded;IP.1=0:0:0:0:0:0:0:0/0:0:0:0:0:0:0:0
```

```
[ocsp_ext]
authorityKeyIdentifier = keyid:always
basicConstraints        = critical,CA:false
extendedKeyUsage        = OCSPSigning
keyUsage                = critical,digitalSignature
subjectKeyIdentifier    = hash
```

```
[server_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth,serverAuth
keyUsage                = critical,digitalSignature,keyEncipherment
subjectKeyIdentifier    = hash
```

```
[client_ext]
authorityInfoAccess     = @issuer_info
authorityKeyIdentifier  = keyid:always
basicConstraints        = critical,CA:false
crlDistributionPoints   = @crl_info
extendedKeyUsage        = clientAuth
keyUsage                = critical,digitalSignature
subjectKeyIdentifier    = hash
```

```
[ v3_intermediate_ca ]
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer
basicConstraints        = critical, CA:true, pathlen:0
keyUsage                = critical, digitalSignature, cRLSign, keyCertSign
```

Script 1: Root-CA Conf.file.cnf

4. Intermed-CA con_file.cnf

```

### ca_openssl.cnf ###
[default]
name                = grupo3-CA
domain_suffix       = grupoTr3s
aia_url              = http://$name.$domain_suffix/$name.crt
crl_url              = http://$name.$domain_suffix/$name.crl
ocsp_url             = http://ocsp.$domain_suffix:12000
default_ca           = CA_default # Sección por defecto para CA
name_opt             = utf8,esc_ctrl,multiline,lname,align

[ca_dn]
countryName          = ES
stateOrProvinceName  = Bizkaia
localityName          = Bilbao
organizationName      = Orlando
organizationalUnitName = IOT
commonName            = IOT_SubCA

[CA_default]
home                = /home/user1/Desktop/PKI/intermed-ca
db                  = $home/db
privado              = $home/private
database            = $db/intermed-ca.index.txt
serial              = $db/intermed-ca.serial
crlnumber            = $db/intermed-ca.crlnum
certificate          = $home/crt/intermed-ca.crt.pem
private_key          = $privado/intermed-ca.key
crl                  = $db/crl/intermed-ca.der # (DER format)
RANDFILE            = $privado/.rand
new_certs_dir        = $home/newcerts
unique_subject       = no
copy_extensions      = none
default_days         = 365
default_crl_days     = 30
default_md            = sha256
policy               = policy_c_o_match

# Política para la CA
[policy_c_o_match]
countryName          = match
stateOrProvinceName  = optional
organizationName      = match
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional

[req]
default_bits         = 2048 # 4096
encrypt_key          = yes
default_md            = sha256

```

```
utf8                = yes
string_mask         = utf8only
prompt              = no
distinguished_name  = ca_dn
req_extensions      = ca_ext

[ca_ext]
basicConstraints     = critical,CA:true
keyUsage             = critical,keyCertSign,cRLSign, digitalSignature
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer

[crl_info]
URI.0               = $crl_url

[issuer_info]
caIssuers;URI.0     = $aia_url
OCSP;URI.0           = $ocsp_url

[name_constraints]
permitted;DNS.0=$name
permitted;DNS.1=$name
excluded;IP.0=0.0.0.0/0.0.0.0
excluded;IP.1=0:0:0:0:0:0:0:0/0:0:0:0:0:0:0:0

[ocsp_ext]
authorityKeyIdentifier = keyid:always
basicConstraints       = critical,CA:false
extendedKeyUsage       = OCSPSigning
keyUsage               = critical,digitalSignature
subjectKeyIdentifier   = hash

[server_ext]
authorityInfoAccess    = @issuer_info
authorityKeyIdentifier = keyid:always
basicConstraints       = critical,CA:false
crlDistributionPoints  = @crl_info
extendedKeyUsage       = clientAuth,serverAuth
keyUsage               = critical,digitalSignature,keyEncipherment
subjectKeyIdentifier   = hash

[client_ext]
authorityInfoAccess    = @issuer_info
authorityKeyIdentifier = keyid:always
basicConstraints       = critical,CA:false
crlDistributionPoints  = @crl_info
extendedKeyUsage       = clientAuth
keyUsage               = critical,digitalSignature
subjectKeyIdentifier   = hash
```

Script 2: Intermed-CA Conf.file.cnf