PKI amb OpenSSL

*Crear a la vegada una clau privada RSA i un CSR (anomenat "server.csr") signat per ella:

Recordem que un CSR és un document (contenint bàsicament una clau pública i signat amb la clau privada pròpia per garantir l'autenticitat) que representa una petició formal a una CA per a què aquesta el signi al seu torn amb la seva clau privada (i així donar fe de què aquesta clau pública es correspon a qui diu que és el propietari, el que a la pràctica es tradueix en obtenir-ne un CRT).

openssl req -newkey rsa:2048 -nodes -keyout private.key -new -subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:.domini.com,DNS:www.unaltre.com" -out server.csr*

NOTA: El paràmetre *-newkey rsa:2048* genera una clau privada de tipus RSA de 2048 bits Si només s'indiqués *-newkey rsa* (ometent el tamany), s'usaria el tamany per defecte indicat a l'arxiu de configuració d'OpenSSL.
NOTA: El paràmetre *-nodes* indica que la clau privada no estarà xifrada. Si no s'escrigués aquest paràmetre, la clau s'encriptaria amb una contrasenya ("passphrase") que es demanaria interactivament en aquell moment; això faria que cada cop que la màquina l'hagués de fer servir (com per exemple, a l'hora de posar en marxa un servidor web segur) demanaria aquesta contrasenya per desencriptar-la, cosa que no és molt pràctic. L'inconvenient de no encriptar la clau privada és que si algú la roba de la nostra màquina i se l'emporta a una altra, aquesta podrà utilitzar-se com a impostora.
NOTA: Si es vol xifrar la clau privada amb un algoritme robust, es recomana crear-la sense xifrar (fent servir el paràmetre *-nodes*) i llavors, posteriorment, afegir-li el "passphrase" amb la comanda *openssl pkey* ... adient (mostrada als paràgrafs anteriors). Això és perquè l'algoritme de xifrat que fa servir *openssl req* ... no és el més segur i aquesta comanda no té cap opció per canviar-lo.
NOTA: El paràmetre *-keyout* serveix per indicar el nom del fitxer on es guardarà la clau privada generada

NOTA: El paràmetre *-new* indica que es vol crear un nou CSR ("Certificate Signing Request")
NOTA: El paràmetre *-subj* serveix per indicar els diferents elements que composen el "Distinguised Name (DN)", el qual representa certa informació administrativa sobre l'individu/empresa/organització que vol fer la petició (és a dir, l'entitat origen del CSR). Aquesta informació (la qual, si no s'indica amb el paràmetre *-subj*, serà preguntada interactivament) és guardada a l'interior del CSR generat per tal de, un cop hagi sigut enviat a la CA signant escollida, pugui ser fàcilment consultada per aquesta. El DN es subdivideix en diversos camps, com ara "C" (de "Country"), "L" (de "Location"), etc que s'indiquen amb la sintaxi */nomcamp1=valor1/nomcamp2=valor2* ... però l'únic obligatori per evitar que es pregunti interactivament és "CN" (de "Common Name"), camp que antigament havia de tenir com a valor el nom DNS completament qualificat (FQDN) del servidor on s'utilitzaria el certificat (o, si aquest no tingués un FQDN, la seva adreça IP estàtica), però que actualment està en desús pels motius indicats a https://github.com/caddyserver/caddy/issues/3755 , entre els quals podem mencionar que no és prou flexible per admetre més d'un nom DNS o que la llargària d'aquests és limitada, entre altres. Així doncs, en realitat el valor que indiquem en el paràmetre *-subj* serà a la pràctica irrellevant, però l'hem d'afegir per a que no se'ns pregunti interactivament.

NOTA: En el cas d'optar per la creació interactiva (és a dir, per no afegir el paràmetre *-subj*), cal tenir en compte que es pot fer que alguna dels camps preguntats interactivament per la comanda anterior ("C", "ST", "L", "O", "OU" o "CN") estigui buit indicant el valor punt ("."), ja que si es pulsa "Enter" s'assignarà el valor per defecte existent a l'arxiu de configuració d'OpenSSL. En el cas concret del camp "C", el valor que es pot indicar ha de ser un "country code" de dues lletres; la llista completa es pot veure aquí: https://www.digicert.com/ssl-certificate-country-codes.htm D'altra banda, també es pregunta que indiquem una tal "challenge password"; aquest és un camp que es recomana deixar en blanc; va ser pensat per usar-se durant una revocació de certificats com una manera d'identificar l'entitat propietària del certificat (si s'indica, es grava tal qual dins del CSR per tal de posar-lo a disposició de la CA) però actualment és una dada que molt poques CAs implementen.

NOTA: El paràmetre *-addext* permet en general informació addicional (les anomenades "extensions") al CSR final (a la configuració general per defecte no s'hi afegeix cap. D'entre les extensions que poden afegir a un CRT "pelat" n'hi ha una, però, de molt important, l'extensió "**subjectAltName**" (o "SAN"). Aquesta extensió fa que el certificat que es generi estigui associat al nom DNS (o noms!) indicat/s. El fet que es puguin indicar més d'un nom DNS (ja sigui de dominis completament diferents o bé d'un mateix domini però fent servir comodins -"wildcards"- per indicar els eventuals noms que podrien aparéixer a l'esquerra d'aquest domini) permet protegir amb un sol certificat multitud de noms DNS diferents (sense aquesta característica caldria tenir llavors un certificat diferent per cada nom DNS diferent que volguéssim cobrir, que és el que passava antigament quan es feia servir el "CN").
NOTA: El paràmetre *-out* serveix per indicar el nom del fitxer (amb extensió .csr) que representa el CSR a enviar
NOTA: Es pot afegir un paràmetre "-YYY", el qual representarà el nom de l'algoritme "hash" escollit (precedit per un guió) que s'aplicarà al CSR per tal de garantir-ne la seva integritat. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com per exemple "**sha256**". De totes formes cal saber que, tot i que no s'indiqui el paràmetre "-YYY", sempre s'hi aplicarà un "hash" per defecte: l'indicat a l'arxiu de configuració general de OpenSSL (del qual en parlarem de seguida)
NOTA: Es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format del CSR Per defecte és PEM

NOTA: ¿Com comprova el client si el certificat rebut del servidor conté efectviament el nom DNS del servidor (ja sigui específicament o a través d'un nom "wildcard") al que precisament vol accedir (i, per tant, té la garantia d'iniciar una connexió segura) si qualsevol connexió sempre es fa a posteriori de la consulta DNS (és a dir, que sempre s'accedeix a qualsevol servidor a través de la seva IP)? En el cas dels clients web (el més habituals amb diferència) aquest problema es solventa amb la capçalera de client HTTP "Host:": un cop el client ha realitzat la petició DNS i ha obtingut la IP del servidor (web) al què es vol connectar, afegeix el seu nom DNS a la capçalera "Host:" de la petició HTTP. D'aquesta manera, a més de saber el servidor quin "virtualhost" ha de mostrar (si en té varis disponibles), també sabrà quin certificat concret ha d'oferir (si en té varis). El client llavors comprovarà que el valor SAN incrustat dins el certificat rebut coincideixi (ja sigui específicament o via "wildcard") amb el valor de la capçalera "Host:" enviada.

Com ja sabem, un cop tinguem el CSR, aquest s'haurà d'enviar a la CA escollida (via mail, formulari o un altre procediment que la CA en particular ofereixi) per a què ens retorni el fitxer ".crt" (és a dir, el nostre certificat signat per aquesta CA) llest per implementar-lo al nostre servidor.

Hi ha una alternativa per no haver d'omplir el "DN" interactivament que no és fer servir el paràmetre *-subj* sinó que és editar directament l'arxiu de configuració d'OpenSSL, de tipus INI, per indicar-hi allà els valors per defecte desitjats dels camps "Country", "State", "Locality", "Organization", "Common Name", etc. Aquest arxiu s'anomena "**/etc/pki/tls/openssl.cnf**" (a Fedora) i "**/etc/ssl/openssl.cnf**" (a Ubuntu). De totes formes, a la pràctica, per no modificar l'arxiu general, el que es fa és crear un arxiu diferent, creat "ad-hoc", que contingui (només) les directives i els seus valors que volguem que siguin diferents dels valors per defecte indicats a l'arxiu "openssl.cnf" (especificant, això sí, les seccions on pertany cada directiva). I llavors indicar que a la comanda *openssl req ...* es farà servir aquest arxiu "ad-hoc" (que anomenarem per exemple "lameva.cnf") mitjançant el paràmetre *-config /ruta/lameva.cnf*

Concretament, els valors que ens interessaran indicar per automatitzar la creació d'un CSR són unes quantes que pertanyen a una secció el nom de la qual ha de venir indicat a la directiva "distinguished_name" (a l'arxiu general aquest nom és "[req_distinguished_name]") pertanyent al seu torn a la secció "[req]". Aquestes directives estan documentades extensament als apartats "Configuration File Format" i "Distinguished Name and Attribute Section Format" de *man openssl-req* però un exemple de fitxer "lameva.conf" mínim podria ser, per exemple, el següent...:

```
[req]
distinguished_name = dnCSR
prompt = no
[dnCSR]
C = US
ST = New York
L = Brooklyn
O = Company
OU = Department
CN = www.example.com
```

...on la directiva "prompt" serveix, si val "no" (valor molt recomanable!), per poder indicar els valors desitjats en un format més automàtic del normal, valors que s'indicaran a les directives "C","ST","L","O","OU" o "CN", de significat evident. Un cop gravat el fitxer "lameva.cnf", podrem generar el mateix CSR, com hem dit, amb la comanda: *openssl req -newkey rsa:2048 -nodes -keyout private.key -new* ***-config lameva.cnf*** *-addext "..." -out server.csr*

NOTA: Hi ha més directives que es poden indicar a la secció on s'especifiquen els valors del DN. Aquestes directives, opcionals serveixen per completar més informació sobre l'entitat que genera el CSR. La llista completa es troba al RFC5280 però alguns exemples són *emailAddress* , *name* o *surname* D'altra banda, dir que els valors del DN també es poden especificar en la seva forma "llarga", així: countryName, *stateOrProvinceName*, *localityName*, *organizationName*, *organizationUnitName* i *commonName*.

Una altra directiva interessant de conèixer a l'hora de generar CSRs és "req_extensions", pertanyent a la secció "[req]" del fitxer de configuració. El valor d'aquesta directiva és el nom de la secció sota la qual s'indicaran les diferents extensions que es vol que tingui el CRT final (a l'arxiu general aquesta directiva per defecte no s'utilitza) . D'entre les extensions que poden afegir a un CRT "pelat" ja hem parlat d'una que actualment és imprescindible: l'extensió "**subjectAltName**". Un exemple de fitxer "lameva.conf" mínim però incorporant l'extensió "subjectAltName" podria ser, per exemple, el següent...:

```
[req]
distinguished_name = dnCSR
req_extensions = extensionsCSR
prompt = no
[dnCSR]
C = US
```

ST = New York
L = Brooklyn
O = Company
OU = Department
CN = www.example.com
emailAddress = admin@example.com
[extensionsCSR]
subjectAltName = *DNS:*.domini.com,DNS=domini.com,DNS=www.unaltre.com*

...on s'han indicat diversos noms DNS que el certificat cobrirà, com ara el del domini principal ("domini.com"), un nom wildcard penjant d'aquest domini principal ("*.domini.com") i un nom de primer nivell però penjant d'un altre domini principal diferent ("www.unaltre.com")

Existeix una sintaxi alternativa per indicar les extensions que és més fàcil de llegir ja que crea una secció específica per l' extensió en particular que es desitgi. Així, l'exemple anterior es podria escriure d'aquesta altra manera:

[req]
distinguished_name = dnCSR
req_extensions = extensionsCSR
prompt = no
[dnCSR]
C = US
ST = New York
L = Brooklyn
O = Company
OU = Department
CN = www.example.com
emailAddress = admin@example.com
[extensionsCSR]
subjectAltName = ***@seccioParticular***
[seccioParticular]
DNS.1=*.domini.com
DNS.2=domini.com
DNS.3=www.unaltre.com

A més de la directiva *DNS.x* (on "x" representa un número sencer), en una secció de tipus *subjectAltName* poden haver més directives, com ara ***IP.x*** (per poder indicar IPs en comptes de noms DNS), ***email.x*** (per poder indicar direccions de correu) o ***URI.x*** (per poder indicar URIs), entre d'altres que es poden consultar a https://tools.ietf.org/html/rfc5280

En qualsevol cas, un cop gravat el fitxer "lameva.cnf" amb l'extensió "subjectAltName" afegida (sigui amb una sintaxi o amb una altra), ara podrem generar el CSR corresponent amb una comanda similar a la que vam fer servir per generar un CSR sense extensions: *openssl req -newkey rsa:2048 -nodes -keyout private.key -new* ***-config lameva.cnf*** *-out server.csr*

> NOTA: Existeix un paràmetre de *openssl req* que permet obviar el valor indicat a la directiva *req_extensions* de l'arxiu de configuració per tal d'utilitzar un altre valor. Es tracta del paràmetre *-reqexts extensionsCSR* Això té sentit quan a l'arxiu de configuració hi apareixen vàries seccions ("[extensionsCSR]","[altresExtensionsCSR]", etc) que contenen diferents extensions cadascuna ("subjectAltName" o altres) i es vol generar un CSR amb un d'aquests conjunts en concret.

Altres extensions diferents interessants que es poden afegir, a més de "subjectAltName", sota la mateixa secció "[extensionsCSR]" (o en una altra secció diferent: acabem de veure que la secció a utilitzar en un moment donat per generar un CSR concret es pot escollir mitjançant la directiva *req_extensions* del fitxer de configuració o bé directament mitjançant el paràmetre *-reqexts* de la comanda *openssl req* ...) són, per exemple (teniu la llista completa d'extensions possibles juntament amb una explicació de cadascuna a *man x509v3_config*):

**basicConstraints* = Aquesta extensió només apareix en certificats arrel amb el valor "CA:TRUE" (sense cometes). També podria valer "CA:FALSE" (sense cometes) però llavors seria equivalent a no haver escrit aquesta extensió. Serveix per indicar precisament que el certificat en qüestió ha de ser de tipus arrel (és a dir, dissenyat per signar altres certificats).

**keyUsage* = Aquesta extensió serveix per restringir els possibles usos que pot tenir el certificat en qüestió als indicats (si aquesta extensió no apareix, el certificat podrà ser emprat per qualsevol ús). Pot valer diversos valors, separats per comes. D'entre els valors més comuns (sense cometes) podem destacar: "*keyEncipherment*", "*dataEncipherment*", "*digitalSignature*" o "*nonRepudiation*" entre altres. El significat de cadascun es pot consulutar a https://roll.urown.net/ca/x509.html#key-usage

*extendedKeyUsage = Aquesta extensió serveix per restringir els possibles usos que pot tenir el certificat en qüestió als indicats (si aquesta extensió no apareix, el certificat podrà ser emprat per qualsevol ús). Pot valer diversos valors, separats per comes. D'entre els valors més comuns (sense cometes) podem destacar: "*serverAuth*", "*clientAuth*", "*codeSigning*" o "*emailProtection*", entre altres.

*Crear a la vegada una clau privada P-256 i un CSR (anomenat "server.csr") signat per ella:

openssl req -newkey ec -pkeyopt ec_paramgen_curve:prime256v1 -nodes -keyout private.key -new -subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:.domini.com,DNS:www.unaltre.com" -out server.csr*

NOTA: Una manera alternativa d'aconseguir el mateix seria fent-ho en dos passos: primer creant un "fitxer de paràmetres" (que anomenarem "fitxer.param"), adient pel tipus de clau el·líptica que voldrem generar, amb la comanda *openssl genpkey* ***-genparam*** *-algorithm ec -pkeyopt ec_paramgen_curve:prime256v1 -out fitxer.param* i, un cop fet això, executant llavors la comanda efectiva pròpiament dita, així: *openssl req -newkey* ***ec:fitxer.param*** *-nodes -keyout private.key -new -subj "..." -addext "...." -out server.csr*

*Crear un CSR signat amb una clau privada pròpia ja existent:

openssl req -key private.key -new -subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:.domini.com,DNS:www.unaltre.com" -out server.csr*

NOTA: Es pot afegir el paràmetre *-keyform {PEM|DER}* per indicar el format de la clau usada (per defecte PEM)
NOTA: Si la clau utilitzada tingués una "passphrase" integrada, es demanarà interactivament. Per evitar això, es pot escriure (suposant que el "passphrase" és "hola") el paràmetre *-passin pass:hola* També es pot indicar així: *-passin env:VAR* si la variable d'entorn VAR especificada té com a valor la "passphrase" adient, o també així: *-passin file:/ruta/fitxer.txt* si el fitxer especificat (el qual podria estar al seu torn xifrat amb algun algoritme simètric...) té com a primera línia el valor de la "passphrase" adient, o també així: *-passin stdin* si la "passphrase" es rebrà a través d'una canonada. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*.
NOTA: Es pot afegir un paràmetre "-YYY", el qual representarà el nom de l'algoritme "hash" escollit (precedit per un guió) que s'aplicarà al CSR per tal de garantir-ne la seva integritat. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com per exemple "**sha256**". De totes formes cal saber que, tot i que no s'indiqui el paràmetre "-YYY", sempre s'hi aplicarà un "hash" per defecte: l'indicat a l'arxiu de configuració general de OpenSSL (del qual en parlarem de seguida)
NOTA: Es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format del CSR Per defecte és PEM

*Comprovar el contingut d'un Certificate Signing Request (CSR):

openssl req -in server.csr -text -noout

NOTA: El paràmetre *-text* serveix per veure els camps que formen les metadades del CSR. Per saber el significat de les més importants es pot consultar https://knowledge.digicert.com/solution/SO18140.html o fer els exercicis
NOTA: El paràmetre *-noout* serveix per no mostrar el contingut del CSR en sí
NOTA: Es pot afegir el paràmetre *-verify* (o escriure'l en comptes de *-text*) per simplement comprovar que el CSR estigui correctament emplenat. Només mostrarà el missatge "Verify is OK" (o "Verify is not OK")
NOTA: Es pot afegir el paràmetre *-inform {PEM|DER}* per especificar el format del CSR (per defecte és PEM)

---

*Crear a la vegada una clau privada RSA i un certificat (anomenat "server.crt") **autosignat** per ella:

La comanda següent és molt útil per generar certificats per servidors en proves en els que els clients hauran de confiar explícitament, ja que estarien signats per una clau a priori no reconeguda (el cas més comú d'això és quan apareix un missatge d'advertència als navegadors en el moment que s'hi vol accedir per primera vegada a un servidor HTTPS que empra aquest tipus de certificats, missatge que deixa a voluntat del visitant la decisió de confiar-ne o no):

openssl req -newkey rsa:2048 -nodes -keyout private.key -new ***-x509*** *-subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:*.domini.com,DNS:www.unaltre.com" -out server.crt*

NOTA: És la mateixa comanda que s'usa per crear un CSR (veure l'apartat corresponent per saber el significat dels diferents paràmetres utilitzats en ella) només afegint un paràmetre extra: ***-x509***. Aquesta opció és la que fa que la sortida de la comanda (fitxer indicat al paràmetre *-out*) no sigui un CSR sinó un CRT (autosignat).
NOTA: Igual que vam veure en la creació de CSRs, es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format del CRT Per defecte és PEM
NOTA: Igual que vam veure en la creació de CSRs, es pot afegir un paràmetre "-YYY", el qual representarà el nom de l'algoritme "hash" escollit (precedit per un guió) que s'aplicarà al CRT per tal de garantir-ne la seva integritat. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com per exemple "**sha256**". De totes formes cal saber que, tot i que no s'indiqui el paràmetre "-YYY", sempre s'hi aplicarà un "hash" per defecte: l'indicat a l'arxiu de configuració general de OpenSSL (del qual en parlarem de seguida)
NOTA: Es pot afegir el paràmetre *-days nº* per indicar el número de dies que el certificat tindrà validesa, més enllà dels quals caducarà. Per defecte són 30 dies.
NOTA: Es pot afegir el paràmetre *-set_serial nº* per indicar manualment un número de sèrie al certificat. Si no s'escriu, s'assignarà un número aleatori.

Si el que volem és actuar com a CA de la nostra organització, la comanda anterior, de fet, es pot utilitzar, gairebé de forma idèntica, per generar certificats arrel, perquè penseu que un certificat arrel no deixa de ser un certificat autosignat amb una característica especial: que no ha d'estar associat a cap nom DNS en particular (és a dir, que no ha de tenir l'extensió "subjectAltName"). En aquest cas, el certificat arrel "server.crt" (que molts cops és nombrat "**ca.crt**" per a identificar-lo) és el que hauran de tenir els clients al seu abast -veurem als exercicis com- per tal de reconèixer a partir de llavors els certificats signats per la nostra CA sense cap missatge d'advertència, i "private.key" (que molts cops és nombrada "**ca.key**" per a identificar-la) és la clau privada de la CA que servirà per signar tots els CSR dels diferents servidors finals de la nostra organització -més endavant veurem com-. En definitiva: la comanda a executar hauria de ser...:

*openssl req -newkey rsa:2048 -nodes -keyout ca.key -new **-x509** -subj "/C=GB/CN=Pep" -out ca.crt*

NOTA: En qualsevol cas, si realment es vol implementar una CA a nivell "empresarial", és convenient instal.lar i executar un software més específic com ara DogTagPKI (https://www.dogtagpki.org)

*Crear a la vegada una clau privada P-256 i un certificat (anomenat "server.crt") **autosignat** per ella:

*openssl req -newkey ec -pkeyopt ec_paramgen_curve:prime256v1 -nodes -keyout private.key -new **-x509** -subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:*.domini.com,DNS:www.unaltre.com" -out server.crt*

NOTA: Veure notes de l'apartat anterior per saber de possibles paràmetres extra a l'hora de crear un CRT
NOTA: Igual que a l'apartat anterior, si no afegim l'extensió "subjectAltName", la comanda anterior podria servir per llavors generar un certificat arrel

*Crear un certificat **autosignat** per una clau privada pròpia ja existent:

*openssl req -key private.key -new **-x509** -subj "/C=GB/CN=Pepe" -addext "subjectAltName=DNS:*.domini.com,DNS:www.unaltre.com" -out server.crt*

NOTA: És la mateixa comanda que s'usa per crear un CSR amb una clau pre-existent (veure l'apartat corresponent per saber el significat dels diferents paràmetres utilitzats en ella) només afegint un paràmetre extra: ***-x509***. Aquesta opció és la que fa que la sortida de la comanda (fitxer indicat al paràmetre *-out*) no sigui un CSR sinó un CRT (autosignat).
NOTA: Igual que vam veure en la creació de CSRs amb clau pre-existent, es pot afegir el paràmetre *-keyform {PEM|DER}* per indicar el format de la clau usada (per defecte PEM)
NOTA: Igual que vam veure en la creació de CSRs amb clau pre-existent, si aquesta clau tingués una "passphrase" integrada, es demanarà interactivament. Per evitar això, es pot escriure (suposant que el "passphrase" és "hola") el paràmetre *-passin pass:hola* També es pot indicar així: *-passin env:VAR* si la variable d'entorn VAR especificada té com a valor la "passphrase" adient, o també així: *-passin file:/ruta/fitxer.txt* si el fitxer especificat (el qual podria estar al seu torn xifrat amb algun algoritme simètric...) té com a primera línia el valor de la "passphrase" adient, o també així: *-passin stdin* si la "passphrase" es rebrà a través d'una canonada. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*
NOTA: Igual que vam veure en la creació de CSRs, es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format del CRT Per defecte és PEM

NOTA: Igual que vam veure en la creació de CSRs, es pot afegir un paràmetre "-YYY", el qual representarà el nom de l'algoritme "hash" escollit (precedit per un guió) que s'aplicarà al CRT per tal de garantir-ne la seva integritat. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com per exemple "**sha256**". De totes formes cal saber que, tot i que no s'indiqui el paràmetre "-YYY", sempre s'hi aplicarà un "hash" per defecte: l'indicat a l'arxiu de configuració general de OpenSSL (del qual en parlarem de seguida)
NOTA: Es pot afegir el paràmetre *-days nº* per indicar el número de dies que el certificat tindrà validesa, més enllà dels quals caducarà. Per defecte són 30 dies.
NOTA: Es pot afegir el paràmetre *-set_serial nº* per indicar manualment un número de sèrie al certificat. Si no s'escriu, s'assignarà un número aleatori.
NOTA: Igual que a l'apartat anterior, si no afegim l'extensió "subjectAltName", la comanda anterior podria servir per llavors generar un certificat arrel

Per tal d'afegir extensions des d'un fitxer de configuració a un CRT autogenerat, es fa de la mateixa manera que vam veure per generar CSRs però amb una diferència només: la línia "*req_extensions*" de l'arxiu "lameva.cnf" deixa de tenir sentit i cal ara substituïr-la per una línia anomenada "*x509_extensions*" (amb el mateix valor, això sí: el nom de la secció que agruparà, dins de l'arxiu, les extensions a incorporar). Per tant, un cop gravat el fitxer "lameva.cnf" amb un contingut similar al mostrat a continuació, podrem executar la mateixa comanda que faríem servir per generar el CRT normalment però afegint el paràmetre (ja conegut) *-config /ruta/lameva.cnf* ; és a dir, per exemple, així: *openssl req -key private.key -new -x509* ***-config /ruta/lameva.cnf*** *-out server.crt*

[req]
distinguished_name = dnCSR
***x509_extensions** = extensionsCSR*
prompt = no
[dnCSR]
C = US
ST = New York
L = Brooklyn
O = Company
OU = Department
CN = www.example.com
emailAddress = admin@example.com
[extensionsCSR]
subjectAltName = @seccioParticular
[seccioParticular]
DNS.1=www.domini.com
DNS.2=www.unaltre.org

NOTA: Existeix un paràmetre de *openssl req -x509* que permet obviar el valor indicat a la directiva *x509_extensions* de l'arxiu de configuració per tal d'utilitzar un altre valor. Es tracta del paràmetre *-extensions extensionsCSR* Això té sentit quan a l'arxiu de configuració hi apareixen vàries seccions ("[extensionsCSR]","[altresExtensionsCSR]", etc) que contenen diferents extensions cadascuna ("subjectAltName" o altres) i es vol generar un CRT amb un d'aquests conjunts en concret.

*Crear un certificat **signat** a partir d'un CSR d'altri i un CRT i clau privada pròpia (fer de CA, vaja):

openssl x509 -req -in server.csr -copy_extensions copyall
-CA ca.crt -CAkey ca.key -out server.crt -set_serial nº

NOTA: El paràmetre *-copy_extensions copyall* serveix per copiar totes les extensions incloses en el CSR al certificat (si no s'indica, no es copiarà cap). Òbviament, ens interessa que com a mínim l'extensió SAN hi aparegui al certificat que crearem
NOTA: El paràmetre *-CAkey* serveix per indicar la clau privada a usar per signar el CSR proporcionat
NOTA: El paràmetre *-CA* serveix per indicar el certificat arrel que identifica la CA que signarà el CSR
NOTA: El paràmetre *-set_serial* serveix per indicar manualment un número de sèrie al certificat.
NOTA: Ens podem estalviar d'escriure el paràmetre *-set_serial nº* cada cop que volguem crear un certificat si escrivim un sol cop (la primera vegada que creem un primer certificat) el paràmetre *-CAcreateserial* . Aquest paràmetre assigna un número aleatori com a número de sèrie del certificat en qüestió i guarda aquest número en un fitxer anomenat "ca.srl" (ubicat a la mateixa carpeta on s'hagi executat la comanda *openssl*). Per crear els següents certificats a partir d'aquest primer ja no caldrà indicar cap paràmetre específic (ni *-set_serial nº* ni *-Cacreateserial*, és a dir, que es pot escriure simplement això: *openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -out server.crt*) perquè OpenSSL usarà el valor actual guardat a "ca.srl" per augmentar-lo automàticament i assignar aquest nou valor augmentat al nou certificat (guardant llavors aquest nou valor dins de "ca.srl" -sobreescrivint el valor anterior-, per tenir-lo disponible per la propera vegada que volem crear un nou certificat).
NOTA: Es pot indicar un fitxer "ca.srl" explícitament per tal de fer-lo servir a l'hora de crear nous certificats amb el paràmetre *-CAserial /ruta/unaltrefitxer.csl*

NOTA: Igual que vam veure a l'apartat anterior, es pot afegir el paràmetre *-keyform {PEM|DER}* per indicar el format de la clau usada (per defecte PEM)
NOTA: Igual que vam veure a l'apartat anterior, si aquesta clau tingués una "passphrase" integrada, es demanarà interactivament. Per evitar això, es pot escriure (suposant que el "passphrase" és "hola") el paràmetre *-passin pass:hola* També es pot indicar així: *-passin env:VAR* si la variable d'entorn VAR especificada té com a valor la "passphrase" adient, o també així: *-passin file:/ruta/fitxer.txt* si el fitxer especificat (el qual podria estar al seu torn xifrat amb algun algoritme simètric...) té com a primera línia el valor de la "passphrase" adient, o també així: *-passin stdin* si la "passphrase" es rebrà a través d'una canonada. Per saber més opcions consulta l'apartat "Pass Phrase Options" de *man openssl*
NOTA: Igual que vam veure en la creació de CSRs i CRTs, es pot afegir un paràmetre "-YYY", el qual representarà el nom de l'algoritme "hash" escollit (precedit per un guió) que s'aplicarà al CRT per tal de garantir-ne la seva integritat. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com per exemple "**sha256**". De totes formes cal saber que, tot i que no s'indiqui el paràmetre "-YYY", sempre s'hi aplicarà un "hash" per defecte: l'indicat a l'arxiu de configuració general de OpenSSL (del qual en parlarem de seguida).
NOTA: Igual que vam veure en la creació de CSRs i CRTs, es pot afegir el paràmetre *-outform {PEM|DER}* per especificar el format del CRT Per defecte és PEM
NOTA: Com sempre que volguem crear un CRT, es pot afegir el paràmetre *-days nº* per indicar el número de dies que el certificat tindrà validesa, més enllà dels quals caducarà. Per defecte són 30 dies.
NOTA: Una altra comanda alternativa que permet obtenir el mateix resultat és *openssl ca ...* però no l'estudiarem

Si es volen afegir extensions al CRT generat independentment de les demanades al CSR, es pot escriure un fitxer específic (que anomenarem "mesExt.cnf") on només apareguin llistades les extensions desitjades (per exemple, així)...

basicContraints = CA:FALSE
subjectAltName = @seccioParticular
[seccioParticular]
DNS.1=www.domini.com
DNS.2=.unaltre.org*

...i llavors executar la mateixa comanda que faríem servir però afegint el paràmetre *-extfile /ruta/mesExt.cnf* ; és a dir, per exemple: *openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key* ***-extfile mesExt.cnf*** *-out ca.crt -set_serial nº*

NOTA: Dins de l'arxiu "mesExt.cnf" es poden agrupar conjunts d'extensions sota seccions. Si es fa així, per indicar quina secció concreta es vol fer servir a l'hora de generar el certificat cal afegir, a més del paràmetre *-extfile*, el paràmetre *-extensions nomSeccio* (si es deixen extensions fora de qualsevol secció, es consideraran dins d'una secció anomenada "default")

*Comprovar el contingut d'un certificat: *openssl x509 -in server.crt -text -noout*

NOTA: El paràmetre *-text* mostra els camps que formen les metadades del CRT. Per saber el significat de les més importants es pot consultar https://knowledge.digicert.com/solution/SO18140.html o fer els exercicis del final.
NOTA: El paràmetre *-noout* serveix per no mostrar el contingut del certificat en sí
NOTA: Es pot afegir el paràmetre *-inform {PEM|DER}* per especificar el format del CRT (per defecte és PEM)

---

*Saber quina/es CA/s han signat un certificat: *openssl x509 -in server.crt -issuer -noout*

NOTA: També es pot usar la comanda *openssl verify [-CAfile ca.crt] server.crt* , la qual mostra amb un "OK" si el certificat indicat ha sigut signat per una CA determinada (identificada pel certificat arrel indicat, si aquesta no fos ja reconeguda per defecte pel sistema)

*Saber si el certificat és de tipus servidor, client i/o CA: *openssl x509 -in server.crt -purpose -noout*

NOTA: També es pot usar la comanda *openssl verify -purpose sslserver -CAfile ca.crt server.crt* , la qual mostra amb un "OK" si el certificat indicat serveix com a certificat de servidor de tercers

*Comprovar la data de caducitat d'un certificat: *openssl x509 -in server.crt -dates -noout*

NOTA: Es mostraran les dates "notBefore" i "notAfter". La que ens interessa conèixer per saber si un certificat ha expirat o no (i, per tant, si continua sent vàlid o no) és la de "notAfter".

*Comprovar el número de sèrie d'un certificat: *openssl x509 -in server.crt -serial -noout*

*Obtenir el "fingerprint" d'un certificat: *openssl x509 -in server.crt -YYY -fingerprint -noout*

NOTA: El paràmetre "-YYY" representa el nom de l'algoritme "hash" escollit (precedit per un guió) per obtenir el fingerprint. Aquest nom pot ser un de la llista mostrada per *openssl list -digest-commands* , com p. ex."**sha256**"

*Convertir un certificat PEM a DER: *openssl x509 -in server.pem -out server.der -outform der*

NOTA: Recordar que el format DER és binari (en format ASN.1) i el format PEM és de tipus text (conté la representació en Base64 del certificat DER en sí, precedit de la línia ---BEGIN CERTIFICATE--- i finalitzat per la línia ---END CERTIFICATE---) En un mateix fitxer PEM poden haver més d'un certificat (separats entre sí per les línies descrites) però no és habitual

*Convertir un certificat DER a PEM: *openssl x509 -in server.der -out server.pem -inform der*

NOTA: If you need to use a cert with the java application (or any other who accept only PKCS#7 format) or with a Microsoft application (or any other who accept only PKCS#12 -PFX-), you can use similar commands to convert from/into this format

Si en algun moment volguéssim comprovar si la clau pública inclosa dins d'un certificat és la mateixa que la que està dins d'un eventual CSR o és la mateixa que es va generar a partir d'una determinada clau privada, podríem comparar la clau pública extreta de cadascun d'aquests artefactes (certificat, CSR, clau privada) és la mateixa executant les comandes següents (on a cada clau pública obtinguda se li aplica un hash SHA256 per facilitar-ne la comparació -això darrer no és imprescindible però sí més còmode-):

openssl x509 -in server.crt -pubkey -noout | openssl sha256
openssl req -in server.csr -pubkey -noout | openssl sha256
openssl pkey -in private.key -pubout | openssl sha256

---

*Conectar a un servidor TLS (HTTP**S**, SMTP**S**, LDAP**S**,...)

openssl s_client -connect www.marca.com:443

NOTA: Per saber si una versió concreta del protocol TLS és acceptada pel servidor, es pot forçar a realitzar la connexió amb una versió concreta afegint el paràmetre respectiu *-ssl3*, *-tls1*, *-tls1_1*, *-tls1_2*, *-tls1_3* (o *-no_ssl3*, *-no_tls1*, *-no_tls1_1*, etc). Si aquesta versió és acceptada pel servidor, s'obtindrà la resposta "CONNECTED"; si no, un "handshake failure." (a les línies anteriors es pot comprovar el valor "Protocol" i "Cipher" per esbrinar realment amb quina versió de TLS i quina ciphersuite està contestant el servidor per establir la connexió.
NOTA: Per saber si una "ciphersuite" concreta és acceptada pel servidor, es pot forçar a realitzar la connexió amb una "ciphersuite" determinada, així *-cipher 'ECDHE-ECDSA-AES256-SHA'* (si s'usa TLS1.2 o inferior) o *-ciphersuites 'TLS_AES_256_GCM_SHA384'* (si s'usa TLS1.3 o superior; es poden provar vàries "ciphersuites" en ordre de preferència si s'escriuen una darrera de l'altra separades per ":"). Si és acceptada, s'obtindrà la resposta "CONNECTED"; si no, un “handshake failure.”. Sobre el concepte de "ciphersuite" en parlarem més endavant.
NOTA: Si el certificat del servidor fos signat per una CA no reconeguda pel nostre sistema, caldrà indicar "a mà" el certificat arrel adient "a mà" amb el paràmetre *-CAfile /ruta/ca.crt*
NOTA: També és possible utilitzar la sintaxis *openssl s_client -host www.marca.com -port 443*
NOTA: A la secció "Certificate chain" es poden veure els detalls del "subject" (línies que comencen per "s") i de l'"issuer" (línies que comencen per "i") de tots els certificats de la cadena (les primeres línies es corresponen al certificat final i les darreres a l'arrel). Si escrivim el paràmetre *-showcerts* podrem veure, a més, tots els certificats pròpiament dits que formen aquesta cadena. També és interessant el paràmetre *-tlsextdebug* per obtenir informació sobre les extensions detectades al certificat del servidor (el darrer de la cadena)
NOTA: Es pot verificar si el certificat ofert per un servidor remot cobreix un determinat nom DNS a més del canònic (això és útil per comprovar si el certificat és multidomini realment) amb la comanda *openssl s_client -verify_hostname www.example.com -connect example.com:443*
NOTA: Entre altres informacions que ens mostra la comanda, podem veure un "Session ID" i un "Session Ticket", els quals permeten reempredre sessions sense que el servidor hagi de mantenir l'estat. En aquest sentit, és interessant el paràmetre *-reconnect* , el qual crea una connexió i la reutilitza cinc vegades més, mostrant per pantalla la informació pertinent de la sessió inicial i després la dels restabliments d'aquesta.
NOTA: En el cas que el client utilitzi un certificat i clau propis per autenticar-se contra el servidor, caldrà afegir els corresponents paràmetre *-cert* i *-key* (veieu l'enunciat de l'exercici nº10 per més informació)

Aquesta comanda funciona com un client *ncat --ssl ...* però més enllà del seu possible ús com a client de consola (bé interactiu bé agafant l'entrada d'una canonada), és útil sobre tot per comprovar els detalls de la connexió TLS (sobre tot si s'usa el paràmetre *-showcerts* o també *-debug*) perquè mostra els detalls de tots els certificats involucrats en cadena durant la connexió TLS fins que aquesta s'estableix finalment (event que s'indica amb la impressió de tres guions ("---") per pantalla), moment en el qual ja es pot "començar a treballar" (fent servir el protocol superior que hi hagi implementat al servidor: si aquest fos HTTP, per exemple, es podria escriure directament una petició GET, etc).

NOTA: Sota l'apartat "Diagnostics" de la pàgina *man openssl-verify* es troben llistats els diferents codis de retorn possibles en l'establiment d'una connexió TLS i una explicació del què significa cadascú (molt útil si hi ha hagut algun error i no sabem per què)
NOTA: D'altra banda, es poden fer diferents proves de velocitat en les connexions TLS amb un servidor remot amb les comandes *openssl s_time -connect www.marca.com:443 -new* (per una nova connexió) o *openssl s_time -connect www.marca.com:443 -reuse* (per una connexió ja establerta)

*Posar en marxa un servidor TLS "a seques":

openssl s_server -key private.key -cert server.crt [-port nº] [-quiet]

NOTA: Per defecte, si no s'indica, el port a usar serà el 4433. Una altra manera d'indicar el port (i la IP concreta a la qual estarà associada és utilitzant el paràmetre *-accept IP:nºport*
NOTA: Es pot afegir el paràmetre *-WWW* per a simular un servidor web senzill que pot carregar pàgines HTML.
NOTA: Per forçar als clients a utilitzar una versió concreta del protocol TLS (per exemple, TLS 1.3) es pot utilitzar el paràmetre *-tls1_3* (o paràmetres similars). Si no s'indica cap, el servidor negociarà amb el client la versió més moderna que sigui entesa per ambdós extrems. Un cop indicada la versió del protocol, es pot restringir fins i tot la/es ciphersuite/s a utilitzar mitjançant el paràmetre *-ciphersuites una:unaAltra:...* (noteu que quan un client envia la seva llista de ciphersuites suportades, el primer valor inclòs en aquesta llista que apareixi com a valor d'aquest paràmetre *-ciphersuites* serà l'utilitzat; és a dir, que l'ordre el marca la llista del client, no pas com s'hagin escrit al paràmetre *-ciphersuites*)
NOTA: Per saber el significat i utilitat dels paràmetre *-verify nº* i *-Verify nº* consulteu l'enunciat de l'exercici nº6 En aquest sentit, si el client que s'hi connecta s'autentiqués mitjançant un certificat de client i aquest fos signat per una CA no reconeguda pel nostre servidor, caldrà indicar "a mà" el certificat arrel adient "a mà" amb el paràmetre *-CAfile /ruta/ca.crt*

Aquesta comanda funciona com un servidor *ncat --ssl -l -p ...* encara que més enllà del seu possible ús com a servidor interactiu de consola, és útil sobre tot per comprovar els detalls de la connexió TLS.

---

*Saber la versió d'OpenSSL instal.lada al sistema: *openssl version [-a]*

*Saber les "ciphersuites" TLS que el nostre OpenSSL pot gestionar: *openssl ciphers -s [-v]*

NOTA: El paràmetre *-s* serveix per efectivament mostrar només les "ciphersuites" suportades pel sistema actual concret, no totes les teòricament existents (que és el que es veu sense aquest paràmetre).
NOTA: El paràmetre *-v* és el mode verbós: mostra més dades sobre cada "ciphersuite" mostrada
NOTA: Es pot restringir la sortida de la comanda anterior segons el tipus de "ciphersuite" desitjada si s'indica alguna característica concreta com a paràmetre. Per exemple, es pot indicar *-tls1_3* per mostrar només les "ciphersuites" reconegudes usades en el protocol TLS1.3 (les úniques recomanables).

*Comprovar la velocitat de diferents algoritmes de xifrat al nostre sistema: *openssl speed*

NOTA: There are three relevant parts to the output. The first part consists of the OpenSSL version number and compile-time configuration. This information is useful if you're testing several different versions of OpenSSL with varying compile-time options. The second part contains symmetric cryptography benchmarks (i.e., hash functions and private cryptography). Finally, the third part contains the asymmetric (public) cryptography benchmarks
NOTA: Es pot fer que els tests es realitzin en vàries CPU a la vegada amb el paràmetre *-multi nº*
NOTA: Es pot restringir la prova a un determinat algoritme (de xifrat o de "hash"), el nom del qual s'haurà d'indicar com a darrer paràmetre. Per saber els noms possibles es pot escriure *openssl list -cipher-commands* o *openssl list -digest-commands*