EXERCICIS PKI amb OPENSSL (I):

1.-a) Obre el navegador Firefox de la màquina real i vés a https://elpuig.xeill.net Seguidament clica sobre la icona del cadenat que apareix a l'esquerra de la barra de direccions i, al quadre que apareix, clica sobre la fletxeta que surt al costat de la frase "Connexió segura". Apareixerà un altre quadre on podràs clicar sobre l'enllaç "Més informació", el qual et donarà pas a un altre quadre que, sota la pestanya "Seguretat", et mostrarà el botó "Mostra certificat". A partir d'aquí, esbrina:

*¿Quina és la CA emprada per la web del centre per signar el seu (aquest) certificat?
*¿Quina és la data d'expiració d'aquest certificat? En total, ¿quant de temps té de validesa?
*¿A quin (únic, en aquest cas) nom DNS està associat aquest certificat?
*¿Amb quin algoritme criptogràfic s'ha generat la clau pública del servidor web, incrustada dins del certificat?
*¿Quina longitud (en bits) té aquesta clau?
*¿Què et pots descarregar si cliques sobre l'enllaç "PEM(cert)"?

NOTA: Una altra manera més curta d'accedir al mateix quadre informatiu és clicant amb el botó dret sobre qualsevol punt de la pàgina web visitada i sel.leccionar l'opció "Informació de la pàgina" mostrada al menú contextual

b) A la mateixa pàgina on has trobat les respostes anteriors, vés ara a la pestanya "R3" i esbrina

*¿Quina és la CA emprada per LetsEncrypt per signar el seu propi certificat? (unes CAs depenen d'altres...pots veure finalment el certificat arrel de la CA "principal" a la pestanya "ISRG Root X1")
*¿Per a què a l'apartat "Usos de la clau" del certificat R3 apareix el valor "Certificate Signing" i al certificat del Puig no?¿Què té a veure la resposta a la pregunta anterior amb el valor que apareix a l'apartat "Restriccions bàsiques" en ambdues pestanyes?

c) Ara vés a la secció de "Privacitat" del quadre de Preferències del Firefox (bé a través del seu menú principal o bé escrivint *about:preferences#privacy* a la seva barra de direccions) i clica sobre el botó "Mostra els certificats". Comprova si la següent frase és veritat o mentida: *"A la pestanya "Entitats" apareix la llista de certificats arrel de les CA reconegudes pel Firefox i a la pestanya "Servidors" apareix la llista de certificats de servidors que han sigut signats per CA desconegudes".*

NOTA: En aquest quadre es poden fer moltes tasques interessants, com ara importar o exportar certificats, eliminar-los, editar la confiança, veure els seus detalls, etc. Anirem estudiant-ho en els propers dies

cII) Digues si aquesta frase és veritat o mentida: *"Un certificat arrel d'una CA és bàsicament la clau pública d'aquesta CA signada per la seva pròpia clau privada mentre que un certificat d'un servidor és bàsicament la clau pública d'aquest servidor signada per la clau privada d'una CA"*

2.-a) Crea un certificat autosignat, que s'anomeni "ca.crt" i que tingui un any de duració, a partir d'una clau privada que pots tenir generada ja d'abans (suposarem que s'anomena "ca,key") o bé generar-la en aquest moment, tant és. Aquest CRT no el farem servir com un certificat de servidor estàndar sinó que funcionarà com a certificat arrel (per poder actuar com una CA privada i així generar certificats pels nostres diferents servidors); per tant, no caldrà que tingui cap camp "SAN" ja que no el vincularem a cap nom DNS. Això ho podries aconseguir de forma no interactiva executant la comanda següent: *openssl req -newkey rsa:2048 -nodes -keyout ca.key -new -x509 -subj "/C=AD/CN=My CA" -days 365 -out ca.crt*

NOTA: No cal dir que "ca.key" no hauria de sortir mai de la màquina: si surt tot l'esquema de seguretat estarà compromés
NOTA: En comptes de generar un certificat arrel per després signar els certificats dels nostres servidors, es podria igualment generar un certificat autosignat diferent per cada servidor que administréssim (igual que s'acaba de fer a l'apartat anterior). No obstant, tenint una entitat certificadora centralitzada és molt més professional perquè els clients només hauran de posseir en un únic certificat arrel ("ca.crt") per poder confiar en els diferents certificats de servidor en comptes d'haver de posseir diferents certificats autosignats, un per cadascun d'aquests servidors.

b) Ara suposarem que estem en una màquina que farà de servidor HTTPS i, per tant, necessitarà dues coses per implementar la seguretat requerida: la seva pròpia clau privada i el seu certificat corresponent. Pots obrir una altra màquina virtual diferent de la què farà de CA per fer-ho més realista però no és necessari si no vols. En qualsevol cas, en aquest hipotètic servidor HTTPS cal que executis primer la comanda següent per crear la seva pròpia clau privada (la qual ha de ser hiperprotegida!), per exemple així: *openssl genpkey -algorithm rsa -out serverHttps.key* i tot seguit, cal que executis la comanda següent per crear un CSR, el qual s'haurà d'enviar d'alguna manera (via mail, ncat, scp...és igual) a la màquina que fa de CA: *openssl req -key serverHttps.key -new -subj "/C=AD/CN=My Server" -addext*

"subjectAltName=DNS:www.elmeunom.com" -out serverHttps.csr (on "www.*elmeunom*.com" -com per exemple, "www.oscar.com"-, representa el nom DNS del servidors HTTPS; tingues en compte que encara que ara mateix aquest nom no estigui definit en cap servidor DNS, és important que sigui aquest pels propers exercicis).

c) Un cop la màquina que fa de CA tingui al seu poder l'arxiu "serverHttps.csr", caldrà que allí executis la següent comanda per tal de generar el certificat del servidor HTTPS corresponent, amb una duració d'un any (i se suposa que se li retornarà per a què ell el pugui fer servir a la seva configuració): *openssl x509 -req -in serverHttps.csr -copy_extensions copyall -CA ca.crt -CAkey ca.key -out serverHttps.crt -days 365 -CAcreateserial*

> NOTA: Tal com s'ha explicat a la teoria, la comanda anterior generarà, a més del fitxer "serverHttps.crt", un fitxer anomenat "ca.srl" incloent el nº de sèrie aleatori usat per generar el primer. És molt important no perdre aquest fitxer!!

3.-a) Aprofita la clau privada que vas crear a l'apartat b) de l'exercici anterior per tornar a generar un altre CSR que ara inclogui a l'extensió "subjectAltName", a més del valor "www.elmeunom.com", els valors addicionals "elmeunom.com", "*.elmeunom.org" i "*.cocacola.com". I fes-ho a partir d'un arxiu de configuració "ad-hoc". Això ho pots aconseguir executant la comanda *openssl req -key serverHttps.key -new -out serverHttps2.csr -config serverHttps2.cnf* tenint en compte que l'arxiu "serverHttps2.cnf" ha de tenir un contingut com el següent:

```
[req]
distinguished_name = dnCSR
req_extensions = extensionsCSR
prompt = no
[dnCSR]
C = AD
CN = My Server 2
emailAddress = admin@elmeunom.com
[extensionsCSR]
subjectAltName  = DNS:www.elmeunom.com,DNS:elmeunom.com,DNS:*.elmeunom.org,DNS:*.cocacola.com
```

b) Torna a actuar ara com a CA; és a dir: signa el CSR generat a l'apartat anterior executant la comanda *openssl x509 -req -in serverHttps2.csr -copy_extensions copyall -CA ca.crt -CAkey ca.key -out serverHttps2.crt -set_serial 12345* ¿Per a quins dominis aquest certificat serà vàlid?

4.-a) Crea un certificat autosignat anomenat "serverHttps3.crt" executant la comanda *openssl req -key serverHttps.key -new -x509 -subj "/C=AD/CN=My Server" -addext "subjectAltName=DNS:www.elmeunom.com" -out serverHttps3.crt*

> NOTA: Com que aquet certificat autosignat no és un certificat arrel d'una CA sinó un certificat de servidor, haurà d'incorporar l'extensió "subjectAltName" per tal d'associar-lo al/s nom/s DNS d'aquest servidor, cosa que no passava amb els certificats arrel

b) Crea un certificat autosignat anomenat "serverHttps3BIS.crt" executant la comanda *openssl req -key serverHttps.key -new -x509 -out serverHttps3BIS.crt -config serverHttps3.cnf* (on l'arxiu "serverHttps3.cnf" haurà de tenir el següent contingut):

```
[req]
distinguished_name = dnCSR
x509_extensions = extensionsCSR
prompt = no
[dnCSR]
C = AD
CN = My Server 3
emailAddress = admin@elmeunom.com
[extensionsCSR]
subjectAltName  = DNS:www.elmeunom.com,DNS:elmeunom.com,DNS:*.elmeunom.org,DNS:*.cocacola.com
```

5.-a) Executa la comanda *openssl x509 -in ca.crt -text -noout* (on "ca.crt" representa el certificat autosignat creat a l'apartat *a)* de l'exercici 2) i confirma que:

-El DN de la CA signant del certificat i el DN del propi certificat tinguin el mateix valor
-Apareguin, sota l'apartat "X509v3 Extensions" les extensions "SKI", "AKI" i "CA:TRUE" (però no pas la "SAN"). Consulta la pàgina *man x509v3_config* per esbrinar per a què serveix cadascuna

b) Executa la comanda *openssl req -in serverHttps.csr -text -noout* (on "serverHttps.csr" representa el CSR creat a l'apartat *b)* de l'exercici 2) i localitza el valor dels següents camps entre els que el formen:

-El DN i el SAN del CSR
-¿Per què no hi apareix cap camp "Issuer:"?
-L'algoritme emprat per generar la clau pública inclosa dins del CSR (com que aquesta deriva de la clau privada emprada, serà el mateix algoritme que l'utilitzat en la creació d'aquesta)

c) Executa la comanda *openssl x509 -in serverHttps.crt -text -noout* (on "serverHttps.crt" representa el certificat creat a l'apartat *c)* de l'exercici 2) i localitza el valor dels següents camps entre els que el formen:

-El Número de sèrie del certificat
-El DN de la CA signant del certificat
-La data d'inici i de final de validesa del certificat
-El DN i el SAN del certificat
-L'algoritme emprat per generar la clau pública inclosa dins del certificat (com que aquesta deriva de la clau privada emprada, serà el mateix algoritme que l'utilitzat en la creació d'aquesta)

d) Executa la comanda *openssl req -in serverHttps2.csr -text -noout* (on "serverHttps2.csr" representa el CSR creat a l'apartat *a)* de l'exercici 3) i confirma que apareix una secció dins d'aquest CSR anomenada "Request Extensions", que inclou l'extensió "X509v3 Subject Alternative Name", on apareixen la llista completa de tots els noms DNS indicats en el moment de la creació del CSR.

e) Executa la comanda *openssl x509 -in serverHttps2.crt -text -noout* (on "serverHttps2.crt" representa el certificat creat a l'apartat *b)* de l'exercici 3) i confirma que apareix una secció dins d'aquest CRT anomenada "X509v3 extensions", que inclou l'extensió "X509v3 Subject Alternative Name", on apareixen la llista completa de tots els noms DNS indicats en el moment de la creació del certificat.

NOTA: Potser no veus cap extensió perquè estiguis afectat pel següent bug: "Extensions in certificate requests are not transferred to certificates" (https://www.openssl.org/docs/man1.1.0/man1/x509.html#BUGS) que fa que "serverhttp2.crt" pugui no incloure extensions.

f) Executa la comanda *openssl x509 -in serverHttps3.crt -text -noout* (on "serverHttps3.crt" representa el certificat creat a l'apartat *a)* de l'exercici 4) i confirma que apareix una secció dins d'aquest CRT anomenada "X509v3 extensions", que inclou l'extensió "X509v3 Subject Alternative Name", on apareixen la llista completa de tots els noms DNS indicats en el moment de la creació del certificat.

g) Executa la comanda *openssl x509 -in serverHttps3BIS.crt -text -noout* (on "serverHttps3BIS.crt" representa el certificat creat a l'apartat *b)* de l'exercici 4) i confirma que apareix una secció dins d'aquest CRT anomenada "X509v3 extensions", que inclou l'extensió "X509v3 Subject Alternative Name", on apareixen la llista completa de tots els noms DNS indicats en el moment de la creació del certificat.

6.-a) Executa la comanda següent (on el certificat "serverHttps.crt" i la clau privada "serverHttps.key" els vas haver de crear a l'apartat *c)* i *b)* de l'exercici 2, respectivament) : *openssl s_server -tls1_3 -key serverHttps.key -cert serverHttps.crt* . A continuació (en un altre terminal) observa la sortida de la comanda *sudo ss -tlnp* i digues en quin port i per a quines IPs rebrà peticions TLS el servidor que acabes de posar en marxa.

NOTA: Recorda que per posar en marxa un servidor segur amb TLS (ja sigui per muntar un servidor HTTP**S**, o SMTP**S**, o el que sigui) necessitaràs SEMPRE aquets dos elements que s'han indicar a la comanda *openssl s_server*: una clau privada (que no ha de sortir mai del servidor i que serveix per desxifrar el que provingui dels clients) i un certificat (que no deixa de ser la clau pública associada, la qual hauran d'obtenir els clients per xifrar els seus missatges cap el servidor). Així doncs, a la configuració del software pertinent (per exemple, si volem implementar un servidor HTTPS, podríem triar l'Apache) caldrà indicar SEMPRE la ubicació d'aquests dos elements (de la forma que aquell software tingui previst)

aII) Connecta al servidor anterior executant (en un terminal de la mateixa màquina on el servidor està en marxa) la comanda *openssl s_client -connect 127.0.0.1:4433* i confirma que la connexió s'ha realitzat correctament enviant la paraula "Hola" al servidor. A partir d'aquí, troba a la pantalla del client els següents valors informatius (mostrats durant el procés d'establiment de la connexió TLS) i digues què signifiquen:

-La frase *"verify error:num=20:unable to get local issuer certificate*" (i com a conseqüència, la frase "*verify error:num=21:unable to verify the first certificate*"
-Les línies que comencen per "s:" i "i:" sota l'apartat "Certificate chain"
-El contingut Base64 mostrat entre les línies ---*BEGIN CERTIFICATE*--- i ---*END CERTIFICATE*---
-El valor que segueix a la paraula "Server Public Key is" al principi d'una línia
-El valor "Session ID", "TLS Session Ticket" i "Timeout" presents sota la secció "SSL-Session" (consulta les notes de teoria sobre la comanda *openssl s_client* per saber-ho)
-El valor que segueix a la paraula "New," al principi d'una línia

NOTA: El significat del valor que segueix a "Cipher is" en aquesta mateixa línia l'estudiarem més endavant
En qualsevol cas, es pot comprovar que aquest valor canvia si afegim a la comanda *openssl s_client* ... anterior els paràmetres *-tls1_3 -ciphersuites TLS_CHACHA20_POLY1305_SHA256* , per exemple
NOTA: Tant el valor que segueix a "New," com el que segueix a "Cipher is" els tornem a trobar en línies següents, sota la secció "SSL-Session", acompanyant a les opcions "Protocol" i "Cipher", respectivament

aIII) Desconnecta el client del servidor (pulsant CTRL+C) i torna-hi a connectar però ara indicant el certificat arrel amb el qual es va signar el seu certificat (en el nostre cas, "ca.crt"). Això ho pots fer amb la comanda *openssl s_client -connect 127.0.0.1:4433* ***-CAfile ca.crt*** ¿Quina diferència veus en els valors informatius que ara apareixen durant el procés d'establiment d'aquesta nova connexió TLS respecte els que vas veure a l'apartat anterior (fixa't sobre tot en les línies que comencen per "Verification error" (o "Verification: OK") i "Verify return code")?

NOTA: Tingues present, per tant, que per a què els clients no notin cap errada en intentar connectar amb un servidor TLS amb un certificat signat per una CA en principi no reconeguda, caldrà aconseguir que prèviament cada client tingui a mà i localitzat (en alguna ruta dins del seu disc dur, normalment) el certificat arrel de la CA en qüestió. Això implica, doncs, haver-lo de descarregar abans d'iniciar la (primera) connexió.