EXERCICIS:

1.-En aquest exercici caldrà que utilitzis dues màquines virtuals (amb un sistema operatiu de tipus "Server") que es puguin fer "ping" entre sí (per simplificar l'exercici, es recomana que les tarjes de xarxa d'ambdues màquines estiguin configurades en el mode "adaptador pont"): una màquina farà de servidor TLS i l'altra màquina farà de client TLS.

a) A la màquina servidora, executa les següents tres comandes per tenir una clau privada ("private.key") i un certificat x509 ("server.crt") que es faran servir al llarg de tots els exercicis d'aquest document. ¿Recordes per a què servia cadascuna?:

openssl req -newkey rsa:2048 -nodes -keyout ca.key -new -x509 -subj "/C=GB/CN=MiCA" -out ca.crt
openssl req -newkey rsa:2048 -nodes -keyout private.key -new -subj "/C=FR/CN=MiServer" -addext "subjectAltName=DNS:.pepe.com,DNS:pepe.com" -out server.csr*
openssl x509 -req -in server.csr -copy_extensions copyall -CA ca.crt -CAkey ca.key -out server.crt -set_serial 1

> NOTA: La primera comanda de les anteriors sol executar-se en una màquina diferent de la màquina on s'executen les altres dues (que és la que farà de servidor), ja que la primera comanda serveix per implementar una CA que podrà signar múltiples certificats, no només d'un servidor concret

b) A la màquina client, fes que es resolguin els noms "www.pepe.com", "moodle.pepe.com", "cloud.pepe.com" i "pepe.com" a la mateixa IP, la de la màquina servidora (que suposarem que és 192.168.12.123). Per fer-ho fàcil i no haver d'implementar un servidor DNS per això, simplement afegeix la següent línia a l'arxiu "/etc/hosts"...:

192.168.12.123 www.pepe.com moodle.pepe.com cloud.pepe.com pepe.com

...i tot seguit comprova que des de la màquina client es pot fer "ping" a la màquina servidor fent servir qualsevol dels noms anteriors.

c) A la màquina servidora, després d'assegurar-te que el tallafocs estigui apagat (*sudo systemctl stop ufw* en Ubuntu, *sudo systemctl stop firewalld* en Fedora), posa en marxa un servidor TLS v1.3 amb la següent comanda *openssl s_server -tls1_3 -key private.key -cert server.crt -port 2222*

> NOTA: Es podria restringir fins i tot les "cipher suites" que volem que pugui utilitzar el nostre servidor si afegim el paràmetre *-ciphersuites "xxx:yyy:zzz:..."* on "xxx", "yyy", "zzz" ... representent els valors concrets de "cipher suites" que volem oferir (l'ordre indicat és irrellevant perquè el marca la petició feta pel client). Exemples de valors serien "TLS_AES_256_GCM_SHA384" o "TLS_CHACHA20_POLY1305_SHA256"
> NOTA: En tot cas, per saber la llista de diferents "cipher suites" que la versió concreta d'OpenSSL instal·lada en un sistema pot reconéixer, es pot executar la comanda *openssl ciphers -s -v*

d) A la màquina client, executa la comanda *openssl s_client -tls1_2 -connect www.pepe.com:2222* ¿Què passa i per què? Executa tot seguit *openssl s_client -tls1_3 -connect www.pepe.com:2222* i comprova que ara no obtens cap error

> NOTA: Es podria restringir fins i tot les "cipher suites" que volem que pugui utilitzar el nostre client si afegim el paràmetre *-ciphersuites "xxx:yyy:zzz:..."* on "xxx", "yyy", "zzz" ... representent els valors concrets de "cipher suites" que volem oferir al servidor per a què ell en trïi la que li convingui (aquí l'ordre indicat és important perquè indicar l'ordre de preferència del client...la primera "cipher suite" que es trobi disponible al servidor, aquella s'utilitzarà). Exemples de valors serien "TLS_AES_256_GCM_SHA384" o "TLS_CHACHA20_POLY1305_SHA256"

No obstant, si ho penses una mica, hauries de veure que el fet que la darrera comanda anterior no doni cap error no hauria de ser així perquè el client està rebent un servidor signat per una CA no reconeguda per ell (ja que no poseeix el seu certificat arrel). Per tant, hauria d'haver aparegut, com a mínim, alguna mena d'advertència. Bé, en realitat, sí que ha aparegut, enmig de la resta d'informació sobre l'establiment de la connexió, però igualment, s'ha continuat com si res. Això és degut a què, tal com s'explica al seu manual, la comanda *openssl s_client* és una eina de proves i està dissenyada per finalitzar el handshake trobi els errors que trobi relacionats amb la verificació de certificats. No obstant, al mateix manual ja s'indica que si aquest comportament no és el desitjat, es pot afegir a la comanda client el paràmetre *-verify_return_error*

e) Executa *openssl s_client -tls1_3 -connect www.pepe.com:2222 -verify_return_error* ¿Què passa i quin missatge d'error obtens?

eII) Ara hauràs de passar al client el certificar arrel de la CA emprada per signar el certificat del servidor (és a dir, copiar el fitxer "ca.crt" del servidor al client). Això ho pots fer de qualsevol forma que et vagi bé....si per exemple triessis la comanda "netcat" per transferir el fitxer tan sols caldria executar en un terminal del client la comanda *nc -l -p 5555 > ca.crt* i en un terminal del servidor la comanda *nc 192.168.12.222 5555 < ca.crt* (on la IP indicada representa la IP del client), però qualsevol forma estarà bé. En tot cas, un cop ja tinguis el fitxer "ca.crt" al client, i mantenint en marxa el servidor de la mateixa manera, ara executa a la màquina client la següent comanda: *openssl s_client -tls1_3 -connect www.pepe.com:2222 -verify_return_error -CAfile ca.crt* ¿Què passa ara i per què? (observa que, efectivament, les línies "verify return" mostrades a la pantalla del client en establir-se la connexió ara ja no indiquen cap error sinó el valor "1", que significa que tot és correcte).

Un altre element que dona per pensar és el fet que si a la comanda client, en lloc d'indicar qualsevol dels noms DNS equivalents del servidor, s'indica la seva adreça IP, el client connecta igualment sense problemes (ho pots provar, si vols). En realitat, si es pensa una mica, sempre passa el mateix procés, ja que quan s'indica un nom DNS, aquest és resolt prèviament (en aquest cas, via arxiu "/etc/hosts"), així que, a la pràctica, el servidor sempre acaba rebent una petició dirigida a la seva IP, no pas cap nom DNS. Llavors, ¿com és que sap entregar el certificat que toca (associat, recordem, als noms DNS indicats a l'extensió SAN) si ni tan sols s'ha especificat cap nom DNS a la petició? Doncs en aquest cas perquè el nostre servidor TLS només té un certificat i és el certificat per defecte que entrega a totes les peticions, però en un servidor TLS amb múltiples certificats, la solució estàndar a aquesta pregunta és indicar el nom DNS desitjat dins de la petició "Client Hello", en un camp específic d'aquest missatge anomenat "**SNI**" (de "Server Name Indication"). Gràcies a aquest camp "SNI", doncs, un servidor TLS pot proporcionar certificats associats a diversos noms DNS sota una mateixa IP, oferint el certificat apropiat a cada petició segons el nom DNS demanat.

> NOTA: Fixeu-vos que aquest camp "SNI" fa la mateixa funció en el "TLS handshake" que la capçalera "Host:" en una petició HTTP, només que el "TLS handshake" succeeix abans que la petició HTTP (de fet, tot just després del "TCP 3-way handshake") i per això aquesta capçalera "Host:" no es pot utilitzar (perquè encara no existeix, bàsicament).

Tot i que, a la pràctica, ben poques vegades es fa servir directament en la comanda client (una altra cosa és als navegadors amb HTTPS), el valor d'aquest camp "SNI" es pot indicar mitjançant el seu paràmetre *-servername nomDNS*

> NOTA: Si s'indica un nom DNS com a valor del paràmetre *-connect* del client d'OpenSSL, aquest nom DNS serà l'utilitzat automàticament com a valor del camp "SNI"

f) Executa *openssl s_client -tls1_3 -connect 192.168.12.123:2222 -servername www.pepe.com* ¿Què passa?

> NOTA: Si intentes indicar un nom DNS no present al SAN del certificat del servidor, no obtindràs cap error perquè és com si no haguessis indicat cap valor SNI: el servidor *openssl s_server* proporcionarà el certificat per defecte (l'únic del que disposa)

2.-a) Instal·la el paquet "tshark" (a Ubuntu) o "wireshark-cli" (a Fedora) a la màquina client i executa tot seguit *sudo usermod -a -G wireshark elteuusuari* i reinicia la teva sessió d'usuari per poder començar a capturar paquets de xarxa sense haver de ser "root". Concretament, executa la comanda tshark -i *enp0s3* -f "ip host *192.168.1.123* and *192.168.1.222*" -w captura1.pcapng

> NOTA: El paràmetre *-i* indica el nom de la tarja de xarxa per on es capturaran els paquets, el paràmetre *-f* indica el filtre de captura que serveixi per només capturar allò desitjat (concretament, es diu que només es vol capturar el tràfic que tinguin com a origen o destí la màquina servidora i la màquina client, res més) i el paràmetre *-w* indica el nom del fitxer on es guardarà la captura. No confondre els filtres de captura amb els filtres de pantalla (indicats amb el paràmetre *-Y*, i amb una sintaxi completament diferent), els quals serveixen només per oferir una visualització de determinats paquets o uns altres segons quin filtre s'indiqui, però capturant-los tots.

b) Posa en marxa de nou el servidor TLS (amb la comanda habitual, *openssl s_server -tls1_3 -key private.key -cert server.crt -port 2222)* i connecta-t'hi des del client *(e*n un altre terminal diferent des d'on està executant el programa Tshark) també de la forma habitual, per exemple amb la comanda *openssl s_client -tls1_3 -connect www.pepe.com:2222 -CAfile ca.crt* Un cop establerta la connexió, intercanvia alguna cadena entre el client i servidor i finalment, talla la connexió pulsant CTRL+C al client.

c) Atura la comanda Tshark (també amb CTRL+C). L'arxiu "captura1.pcapng" que hauràs generat representa tot el tràfic TCP i TLS ocorregut entre els extrems. Per analitzar-lo podríem fer servir el Tshark igualment, però serà més còmode fer-ho amb el Wireshark. Per això, si la màquina client té entorn gràfic, només caldrà que obris aquesta captura amb el Wireshark; si la màquina client no té entorn gràfic, però, llavors o fas servir la comanda Tshark per inspeccionar la captura amb l'ajuda de les notes que apareixeran als següents apartats de l'exercici, o bé hauràs de copiar primer el fitxer "captura1.pcapng" a la màquina real (fent servir "netcat", per exemple, o scp, o via email...com vulguis) i llavors fer servir el Wireshark de la màquina real. En tot cas, obre la captura perquè procedirem a analitzar-la

> NOTA: La comanda per obrir una captura des del terminal i observar la llista de paquets allà presents amb informació bàsica de cadascun d'ell mostrada en columnes (similar a com ho fa el panell superior del Wireshark) és ***tshark -r captura1.pcapng***

d) Observa que, després dels tres primers paquets que apareixen a la captura (els de tipus TCP -SYN, SYN-ACK i ACK-, que representen el "TCP 3-way handshake", és a dir, l'establiment de la connexió TCP) apareix el primer paquet TLS, corresponent al missatge "Client hello". Sel·lecciona'l al panell superior del Wireshark i tot seguit observa, desplegant la línia "Transport Security Layer" que apareixerà al panel del mig, els següents valors:

> NOTA: La comanda equivalent de terminal seria ***tshark -O tls -Y "frame.number==4" -r captura1.pcapng*** ,on el paràmetre *-Y* indica que només es vol veure la informació del paquet nº4 (és a dir, en principi hauria de ser el paquet corresponent al "Client hello" després dels tres del "TCP 3-way handshake"...en tot cas, això es pot comprovar primer fent simplement *tshark -r captura1.pcapng* i observant el número de paquet a la columna de més a l'esquerra) i que, en concret, es vol veure la part disseccionada corresponent al protocol TLS només (si es volgués veure tota la informació disseccionada de tots els nivells del paquet -Ethernet, IP, TCP, ...- llavors caldria haver utilitzat el paràmetre *-V*)

> * "Session ID" (útil per si es vol reemprendre una sessió TLS ja començada prèviament, el "0-RTT")
> * "Cipher suites" (¿quines apareixen llistades?)
> * L'extensió "server_name" amb el camp "Server Name" (¿què representa aquest valor?)
> * L'extensió "psk_key_ exchange_modes" (representa la proposta d'algoritme d'intercanvi de claus que el client ofereix per usar al servidor; ¿quin algoritme concret s'hi està indicant?)
> * L'extensió "key_share" (representa el valor que es vol que faci servir el servidor per generar la clau simètrica efímera)

> > NOTA: Pot sorprendre que aparegui que la versió del "handshake" és TLS v1.2 i no v.1.3. Això és perquè actualment la majoria de servidors encara fan servir TLS v1.2 en lloc de v1.3 i així ja no hi més problemes...però el client llavors indicarà que és capaç de fer servir TLSv1.3 (per si el servidor en qüestió sí que el pot entendre) en un altre lloc, concretament en l"extensió" "supported_versions" del missatge "Client Hello"

e) Tria ara el missatge "Server hello" al panell superior del Wireshark i tot seguit observa, desplegant la línia "Transport Security Layer" que apareixerà al panel del mig, els següents valors:

> NOTA: La comanda equivalent de terminal seria *tshark -O tls -Y "frame.number==6" -r captura1.pcapng* ja que en principi el paquet "Server hello" hauria de ser el sisè paquet de la captura, però en tot cas, això es pot comprovar primer fent simplement *tshark -r captura1.pcapng* i observant el número de paquet a la columna de més a l'esquerra

> * El mateix "Session ID" que l'indicat al paquet "Client hello"
> * La "cipher suite" triada d'entre les oferides pel client
> * L'extensió "supported_versions" (¿quin valor conté?)
> * L'extensió "key_share" (representa el valor que es vol que faci servir el client per generar la mateixa clau simètrica efímera que el servidor acaba de calcular)

Veuràs que poca cosa més pots esbrinar en aquest missatge per la gran majoria de dades venen xifrades (sota l'epígraf "Application Data Protocol". A TLS v1.3, tot el que ve a continuació del principi del paquet "Server hello" ve xifrat, incloent l'enviament del certificat de servidor al client (que és el que bàsicament representen aquests camps "Application Data Protocol" en aquest cas) i tots els paquets posteriors enviats pel servidor.

> NOTA: Noteu com encara, després del missatge "Server hello", a la llista de paquets intercanviats apareix un missatge TLS provinent del client i dirigit al servidor etiquetat com "Change cipher spec". A partir d'aquest missatge, el client també procedirà a enviar xifrats tots els seus paquets

f) A continuació jugarem amb alguns filtres de pantalla per veure només els paquets que concordin amb el filtre indicat en cada moment (recordeu que si féssim servir Tshark aquests filtres els hauríem d'indicar amb el paràmetre *-Y*)

* ¿Què veus si indiques el filtre "tls"?
* ¿Què veus si indiques el filtre "tls.handshake"?
* ¿Què veus si indiques el filtre "tls.handshake.type == 1"?
* ¿Què veus si indiques el filtre "tls.handshake.type == 2"?
* ¿Què veus si indiques el filtre "tls.handshake.extensions.supported_version == 0x0304"?
* ¿Què veus si indiques el filtre "tls.handshake.extension.type == server_name"?

> NOTA: Amb Tshark només caldria executar una comanda similar a ***tshark -Y "filtre" -r captura1.pcapng***
> NOTA: La referència completa de filtres relacionats amb el protocol TLS és https://www.wireshark.org/docs/dfref/t/tls.html

2BIS.-a) Atura el servidor TLS i ara posa'l en marxa afegint-hi un nou paràmetre, així: *openssl s_server -tls1_3 -key private.key -cert server.crt -port 2222* ***-keylogfile claus.txt***

El paràmetre anterior indica el fitxer on es guardaran les claus simètriques efímeres que va usant el servidor; aquesta informació ens servirà per poder desxifrar amb el Wireshark el tràfic capturat (però la captura anterior no ens servirà degut al "Pefect Foward Secrecy"...haurem de tornar a generar una nova captura).

b) Executa al client la comanda *tshark -i enp0s3 -f "ip host 192.168.1.123 and 192.168.1.222" -w captura2.pcapng* i tot seguit connecta-t'hi al servidor (en un altre terminal diferent des d'on està executant el programa Tshark) amb la comanda de sempre: *openssl s_client -tls1_3 -connect www.pepe.com:2222 -CAfile ca.crt* Un cop establerta la connexió, intercanvia alguna cadena entre el client i servidor i finalment, talla la connexió pulsant CTRL+C al client.

c) Atura la comanda Tshark (també amb CTRL+C). Obre l'arxiu "captura2.pcapng" amb el Wireshark (ja sigui el de la pròpia màquina client si aquesta té entorn gràfic o bé el de la màquina real, previ traspàs del fitxer de captura...si vol fer servir el Tshark, llegeix la "nota" següent) i tot seguit, vés al menú "Edit->Preferences->Protocols->TLS" i al quadre que t'hi apareix, al valor "(Pre)-master-secret log filename" indica la ruta del fitxer "claus.txt". ¿Què passa tot seguit a la finestra principal del Wireshark? Concretament, sel·lecciona el paquet "Server hello" i observa les diferents línies "Record layer" que apareixen ara al panell central sota la línia "Transport Security Layer"....¿segueix apareixent l'etiqueta "Application Data Protocol"?

> NOTA: Si fessis servir la comanda *tshark* per llegir el contingut d'una captura, l'acció equivalent a desxifrar el contingut d'un tràfic TLS mitjançant les claus guardadesa "claus.txt" seria afegir-ne el paràmetre ***-o "ssl.keylog_file:/ruta/claus.txt"***, així: *tshark -o "ssl.keylog_file:/ruta/claus.txt" -r captura2.pcapng* (per veure la "finestra principal" de tots els paquets ara desxifrats) o *tshark -o "ssl.keylog_file:/ruta/claus.txt" -O tls -Y "frame.number==6" -r captura2.pcapng* (per veure la dissecció desxifrada del paquet "Server hello")

d) Més en concret, dins de la línia "Record layer: Handshake protocol: Certificate", ¿quina informació ara tens disponible (que abans no) sota l'apartat "Handshake protocol: Certificate" -> "Certificates" -> "Certificate" -> "signedCertificate" (i allí, més camps, com "issuer", "validity", "subject", "extensions",...)?

e) Troba les cadenes intercanviades entre el client i servidor (i/o viceversa) en la sessió capturada. Pista: busca els paquets etiquetats com "Application Data"

> NOTA: Amb Tshark caldria fer ***tshark -o "ssl.keylog_file:/ruta/claus.txt" -O data -r captura2.pcapng***

La comanda *openssl s_server* ofereix també el paràmetre *-WWW* , el qual emula un servidor HTTPS senzill que podrà servir pàgines web (ubicades de forma relativa al directori des d'on s'hagi executat; per exemple, si la URL és *https://www.pepe.com/pagina.html*, serà enviada la pàgina "./pagina.html"). Cal dir que existeix una URL específica, "/stats", que permet visualitzar informació de configuració i estat del servidor.

> NOTA: Les capçaleres de resposta HTTP són generades pel servidor automàticament. En aquest sentit, l'extensió del fitxer demanat serveix per determinar el valor de la capçalera de resposta "Content-Type": extensions "html", "htm" i "php" fan que aquesta capçalera valgui "text/html" mentre que tota la resta fan que valgui "text/plain".
> the response headers

3.-a) Atura la màquina virtual client (no la faràs servir més). I a la mateixa màquina servidora dels exercicis anteriors...:

> * Crea una pàgina web anomenada "pag.html" amb un contingut qualsevol, com per exemple aquest: *<html><body>Hola amics! </body></html>*
> * Arrenca un servidor TLS així: *openssl s_server -key private.key -cert server.crt -port 2222 -WWW*

b) Obre el navegador Firefox de la teva màquina real i vés a https://192.168.12.123:2222/pag.html ¿Què passa? ¿Quin tipus d'error "SEC_ERROR_xxx" t'hi apareix? No li donis al botó d'"Accepto el risc".

> NOTA: Podríem haver fet servir algun nom DNS dels presents al certificat, però la màquina real no els pot resoldre (ja que no hem modificat el seu "/etc/hosts", i per fer-ho necessitem ser "root")

c) Vés al menú "Preferències"->"Privadesa i seguretat" -> botó "Mostra els certificats" (sota la secció "Certificats") i al quadre que t'hi apareix, vés a la pestanya "Servidors" i clica en el botó "Afegeix una excepció". Al quadre que se t'apareix indica "https://192.168.12.123:2222", clica en el botó "Obté el certificat" i finalment, en el botó "Confirma l'excepció de seguretat". Torna a anar ara a la mateixa URL que l'apartat anterior des de la pantalla principal del navegador. ¿Què veus ara?

> NOTA: El mateix procediment hauries de fer si el certificat del servidor fos autosignat

cII) ¿Què estaries fent, però, si en lloc de fer els passos indicats a l'apartat anterior, haguessis descarregat el fitxer "ca.crt" del servidor TLS a la teva màquina real i llavors haguessis anat al menú "Paràmetres"->apartat "Privadesa i seguretat" -> botó "Mostra els certificats" (sota la secció "Certificats") i al quadre que t'hi apareix, haguessis anat a la pestanya "Entitats" i haguessis clicat en el botó "Importa..." per tal de triar aquest fitxer "ca.crt"?

d) Executa a la màquina real la comanda *curl https://192.168.12.123:2222/pag.html* ¿Què passa? Pista: tingues en compte que el magatzem de certificats arrel i certificats confiables de servidors (és a dir, el "trus store") que utilitza el Firefox no és el mateix que el que utilitza el sistema en general (que és on, precisament, consulta la comanda *curl*)

Una manera ràpida de solucionar el problema anterior seria descarregar l'arxiu "ca.crt" del servidor TLS i emprar-lo llavors en executar la comanda *curl* a la màquina real així: *curl -cacert ca.crt https://192.168.12.123:2222/pag.html* ; d'aquesta forma, el client HTTPS confiaria en el certificat presentat degut a haver indicat un certificat arrel "ad-hoc". Ho podríem fer així, però optarem per una altra "solució" molt menys elegant i segura, explicada al següent apartat.

> NOTA: Una altra cosa que es podria fer és aconseguir que el sistema client acceptés el certificat arrel de forma permanent i general (abans fins i tot d'anar al nostre servidor) en lloc d'haver indicar-lo explícitament en cada petició. Concretament, això es pot aconseguir copiant el fitxer "ca.crt" a la carpeta "**/usr/local/share/ca-certificates**" (a Ubuntu) o "**/etc/pki/ca-trust/extracted**" (a Fedora) i tot seguit executant la comanda *sudo update-ca-certificates* (a Ubuntu) *o sudo update-ca-trust* (a Fedora). Amb això el que estarem fent és afegir aquest certificat arrel al "trust store" del sistema, del qual confien la majoria d'aplicacions (com *curl*)

dII) Executa tot seguit la comanda *curl -k https://192.168.12.123:2222/pag.html* ¿Què passa ara? (consulta la pàgina del seu manual per confirmar la teva suposició). Apaga finalment el servidor TLS de la màquina virtual.

4.-a) A la màquina servidora, instal·la el paquet "apache2" (a Ubuntu) o "httpd" (a Fedora) per així disposar d'un servidor HTTP/S Apache. Si estàs a Fedora, hauràs d'instal·lar també el paquet "mod_ssl" i, a més, executar la comanda *sudo setsebool -P httpd_read_user_content 1* (per tal d'indicar al framework de seguretat SELinux que permeti a lApache llegir el certificats+claus)

b) Crea un arxiu anomenat "pepe.com.conf" dins de la carpeta "/etc/httpd/conf.d" (a Fedora) o "/etc/apache2/sites-available" (a Ubuntu) amb el següent contingut (on "server.crt" i "private.key" són el mateix certificat i clau privada utilitzats als exercicis anteriors (llegeix en aquest sentit la següent nota):

*<VirtualHost *:443>*
ServerName pepe.com
*ServerAlias *.pepe.com*
DocumentRoot /var/www/pepe
SSLEngine on
SSLCertificateFile /ruta/server.crt
SSLCertificateKeyFile /ruta/private.key
</VirtualHost>

NOTA: El valor de les directives *ServerName* i *ServerAlias* ha de coincidir obligatòriament amb algun dels noms DNS indicats a l'extensió "subjectAltName" del certificat "server.crt".
NOTA: Per seguretat, la ruta on haurien d'ubicar-se la clau privada i el certificat del servidor hauria de ser preferiblement "/etc/ssl". A més, els permisos de la clau haurien de ser 640 amb el propietari "root" (és a dir, com administrador caldria fer *chmod 640 private.key* i, si calgués també, *chown root:root private.key*) i els permisos del certificat haurien de ser 644 (és a dir, com administrador caldria fer *chmod 644 server.crt* i, si calgués també, *chown root:root server.crt*)

NOTA: En el cas de fer servir certificats de clients (per autenticar a ells també), caldria afegir a la configuració anterior la línia *SSLCACertificateFile /ruta/ca.crt* (o, alternativament, *SSLCACertificatePath /ruta*) per tal d'apuntar al/s certificat/s arrel que els permeti verificar. En aquest cas, si el client fos la comanda *curl*, caldria afegir-li els paràmetre ***--key privateclient.key*** i ***--cert client.crt*** per fer-li saber quina clau privada del client hauria d'utilitzar i quin certificat del client haurà d'enviar al servidor en qüestió

c) Realitza les següents tasques auxiliars:

*Crea (com a root) la carpeta "/var/www/pepe"
*Crea (com a root) dins de la carpeta anterior un arxiu anomenat "pag.html" amb el contingut *<html><body>Hola, sóc Pepe</body></html>*
*(Només a Ubuntu): Executa (com a root) les comandes *a2enmod ssl* i *a2ensite pepe.com*
*Reinicia (com a root) la configuració del servei (*systemctl reload apache2* o *systemctl reload httpd*)

d) Repeteix l'apartat b) de l'exercici anterior (si funcionés la resolució de noms, podries indicar en lloc de la IP del servidor, algun dels seus possibles noms, com "pepe.com" o "www.pepe.com"...en qualsevol cas obtindries el mateix resultat)

dII) Accepta l'excepció mostrada en el missatge d'error anterior i torna a intentar tot seguit visualitzar la pàgina "pag.html". ¿Què passa ara?

dIII) ¿Què és el que trobes a la finestra mostrada en anar al menú "Paràmetres"-> apartat "Privadesa i Seguretat"-> botó "Mostra els certificats"-> pestanya "Servidors", i per què? ¿Què passa si, esborres d'allà el certificat del teu servidor Apache i tornes a intentar de nou visualitzar la pàgina "pag.html", i per què?

e) Intenta ara visualitzar "pag.html" però indicant el protocol "http://" (en lloc de "https://") a la barra de direccions del navegador. ¿Què veus i per què?

NOTA: En aquest sentit, podries provar d'afegir les següents línies a la configuració del "virtualhost" i veure el seu efecte:

RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.)$ https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]*

f) Consulta la documentació oficial del mòdul "mod_ssl" de l'Apache (disponible a https://httpd.apache.org/docs/2.4/mod/mod_ssl.html) per esbrinar per a què serveixen les següents directives:

***** *SSLRequireSSL* (només vàlida dins d'una secció <*Directory*> o <*Location*>)

***** *SSLCipherSuite* amb el valor *PROFILE=SYSTEM* (consulta en aquest cas els comentaris escrits a l'arxiu "/etc/httpd/conf.d/ssl.conf" -a Fedora-) o bé amb el valor *HIGH:!aNULL* (consulta en aquest cas els comentaris escrits a l'arxiu "/etc/apache2/mods-available/ssl.conf" -a Ubuntu-)

* *SSLProtocol* amb el valor all -SSLv3 (consulta en aquest cas els comentaris escrits a l'arxiu "/etc/httpd/conf.d/ssl.conf" -a Fedora- o a l'arxiu "/etc/apache2/mods-available/ssl.conf" -a Ubuntu-)

* *SSLPashPhraseDialog* (especialment amb el valor *builtin*), indicada a nivell de "virtualhost"

* *SSLOptions +StdEnvVars* (consulta també l'apartat "Environment Variables" de la documentació) indicada a nivell de "virtualhost", el seu possible ús combinant-la amb la directiva *Require expr*

NOTA: Tot i que si s'indica pot provocar un consum de recursos alt en el servidor, la directiva anterior pot ser interessant activar-la per pàgines PHP (per tal de què aquestes puguin fer servir les variables d'entorn de tipus SSL_*). És per això que sovint s'escriu dins d'una secció específica com <*FilesMatch "\.php$*> o similar

* *SSLHonorCipherOrder* (amb el valor *off*)

* *SSLSessionTickets* (amb el valor *off*)