INTRODUCCIÓ A LES COMUNICACIONS SEGURES I ELS CERTIFICATS DIGITALS

Requisits d'una comunicació "segura"

***Autenticació:** procés que dóna la garantia de conèixer sense error la identitat d'una persona (física o jurídica) o màquina. Aquest concepte guarda gran relació amb el de **no repudi**, que significa la impossibilitat de rebutjar l'autoria d'una determinada acció o document). Les eines principals per realitzar una autenticació actualment són els sistemes d'usuari/contrasenya (en el cas de persones) i els certificats digitals (en el cas de màquines).

***Integritat:** característica d'una comunicació; si existeix, significa que es pot confiar en què una determinada informació (per exemple l'existent dins d'un document electrònic) no ha sigut manipulada i, per tant, correspon al seu estat original. La principal eina per confirmar la integritat d'un missatge és la signatura digital.

***Confidencialitat:** característica d'una comunicació; si existeix, significa que es pot guardar de la vista de tercers el secret transferit a un determinat destinatari (normalment, aquest secret seran els missatges enviats i rebuts entre dues màquines) mitjançant el xifratge -i posterior desxifratge- dels missatges. L'eina principal per aconseguir aquest objectiu és la criptografia (simètrica o asimètrica).
NOTA: Els certificats digitals (que proporcionen autenticació) i les signatures digitals (que proporcionen integritat) són eines que només es troben disponibles a la criptografia asimètrica. Per tant, la criptografia simètrica pot xifrar i desxifrar informació però sense la garantia de què aquest missatge arribi o provingui de qui diu ser i/o que no hagi sigut manipulat.

Conceptes de clau simètrica (compartida) i asimètrica (pública)

Xifrar és el procés de convertir dades a un format alternatiu diferent de l'original. Aquest procés generalment requereix dos elements: una clau i un algoritme (d'entre diferents possibles, tots coneguts) que realitza, a partir d'aquesta clau, la transformació de les dades desitjada.

Els algoritmes de clau simètrica utilitzen la mateixa clau per xifrar i desxifrar dades. Això implica que,per a què sigui segura, aquesta clau ha de ser coneguda només per l'emissor i el receptor i ningú més. Els algoritmes de clau simètrica són molt ràpids, però necessiten transmetre prèviament la clau única per un canal segur entre l'emissor i el receptor. A més, el número de claus ha d'augmentar a mesura que augmenten les persones que es volen comunicar de forma segura entre sí.

En canvi, els algoritmes de clau asimètrica (també anomenats "de clau pública" degut a què una de les claus que fan servir pot ser distribuïda a tot el món) fan servir diferents claus pel xifrat i pel desxifrat. La idea de la criptografia asimètrica és que les dades que són xifrades amb una clau pública (la del receptor) només poden ser desxifrades amb la clau privada associada.

També es poden utilitzar -¡només!- les claus asimètriques per una altra acció molt important: signar un missatge (i així garantir-ne la seva integritat). En aquest cas l'emissor utilitzaria la seva clau privada i el receptor comprovarà la signatura amb la clau pública associada. Així doncs, resumint:

1) L'emissor fa servir la pròpia clau privada per signar el missatge.
(En realitat, el que es fa és xifrar el hash del missatge, i és aquest hash xifrat que s'inclou adjunt al missatge)
2) L'emissor fa servir la clau pública del receptor per xifrar el missatge (incloent el hash signat)
3) Es transmet el missatge xifrat i signat.
4) El receptor desxifra el missatge fent servir la pròpia clau privada.
5) El receptor comprova la signatura del missatge fent servir la clau pública de l'emissor.
(En realitat, el que es fa és desxifrar el hash i comprovar que aquest sigui el mateix que els hash recalculat del missatge rebut)

En aquest procediment mai es transmeten les claus privades (¡aquestes sí que han de ser secretes!) i l'intercanvi de les claus públiques és senzill, ja que no cal que es transmetin utilitzant un canal segur. A més, es garanteix que el missatge no pot ser compromès en la transmissió i tant l'emissor com el receptor comproven l'autenticitat de l'altra entitat. L'inconvenient principal de la criptografia de clau pública és el consum de recursos (és més costosa en temps de xifratge/desxifratge) i el missatge xifrat ocupa més que el text net.

Algoritmes simètrics: AES, 3DES, IDEA, Blowfish, Twofish, Serpent , CAST, ...
Algortimes asimètrics: RSA, DSA, ElGamal...

La criptografia híbrida combina el sistema de clau compartida i el de clau pública per obtenir tots els avantatges sense els inconvenients. En aquest sistema només s'utilitza la criptografia de clau pública per transmetre una clau compartida, normalment generada de manera dinàmica per aquesta comunicació, al receptor. Una vegada s'ha comunicat la clau compartida a les dues estacions, s'utilitza xifratge simètric per xifrar el gros del missatge. D'entre els protocols possibles per realitzar aquesta negociació segura de claus (en un canal insegur i de manera no autenticada), destaca el protocol Diffie-Hellman, que és el sistema que utilitza, per exemple, Ssh.

Teoria de certificats

En el resum del procés descrit a la pàgina anterior hi ha un punt feble: ¿com sap l'emissor que la clau pública del receptor que farà servir per xifrar el missatge és realment del receptor i no d'un impostor que podria llavors desxifrar aquest missatge?. I igualment, ¿com sap el receptor que la clau pública de l'emissor que farà servir per comprovar la signatura del missatge és realment de l'emissor i no d'un impostor que podria fer-se passar per l'emissor vertader? Aquest punt feble es resol fent servir unes claus públiques "amb vitamines": els certificats digitals. Un certificado digital es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad y su clave pública. Puede ser usado, por ejemplo, para verificar la firma digital de dicha entidad. Un certificado contiene usualmente, entre otros datos:

*Nombre y datos burocráticos identificativos de la entidad certificada
*Número de serie
*Fecha de expiración
*Copia de la clave pública de la entidad certificada (utilizada para verificar su firma digital)
*Firma digital de la autoridad certificadora (de forma que el receptor pueda verificar que esta última ha establecido realmente la asociación)

En caso particular, por ejemplo, de un servidor web, el certificado que ofrece a los navegadores cuando un usuario visita su web garantiza que, durante la comunicación encriptada que realice el navegador gracias a la clave pública obtenida de este servidor web, dicho servidor es realmente la que dice ser (siempre y cuando dicho certificado venga firmado por una autoridad de certificación confiable por el cliente). Los navegadores ya vienen de serie con una lista de firmas reconocidas de varias autoridades de certificación confiables (los llamados "certificados raíz"; Mozilla ofrece su lista en https://wiki.mozilla.org/CA:IncludedCAs)...por eso en muchas páginas seguras no salta ningún popup extraño: porque su certificado fue firmado por una autoridad reconocida por el navegador. De hecho, el navegador comprueba varias cosas para dar por válida la conexión y comenzar la comunicación encriptada:

-El certificado obtenido no haya expirado ya
-El certificado obtenido ha sido creado por el mismo servidor web al que se está accediendo
-El certificado está firmado por alguien en el que se confía

Cualquier individuo o institución puede generar certificados digitales y convertirse así en una autoridad de certificación, pero si no es reconocido por quienes interactúen con el certificado, el valor del mismo (que en este caso se denomina "certificado autofirmado" porque tiene nuestra propia firma, aun no siendo ninguna CA) es prácticamente nulo (servirá para hacer pruebas o para un grupo de usuarios que confíen en nuestra máquina). Por ello las autoridades de certificación deben ser reconocidas como institución certificadoras por la entidad pública competente, de forma que su firma pueda ser reconocida como fiable, transmitiendo esa fiabilidad a los certificados emitidos por la citada institución. En España, las entidades que otorgan validez a las autoridades de certificación son la Fàbrica Nacional de Moneda y Timbre, el Ministerio de Industria, Turismo y Comercio, la Agència Catalana de Certificació, etc

Una autoridad de certificación gratuita es Cacert.org (http://www.cacert.org) . Otra es Startssl.com (http://www.startssl.com). O la que últimament té més èxit, LetsEncrypt (https://letsencrypt.org). La mayoría, no obstante, son comerciales: Thawte, Verisign, GlobalSign, Comodo, Ancert, RSA, FábricaNacionaldeMonedayTimbre, etc. Si queremos que nuestro servidor cuente con un certificado firmado por alguna de estas entidades, deberemos generar una solicitud de certificado (un "CSR") y enviársela para que, tras el pago correspondiente, nos hagan llegar el fichero que representa el certificado firmado convenientemente.

El formato más extendido para certificados digitales es el estándar UIT-T X.509 (son los ficheros con extensión .crt). Pero hay otros:

*.pem: Formato que se desarrolló específicamente en su momento para el uso de certificados con correo electrónico. Actualmente también se usa para distribución de claves privadas.
*.cer Formato muy frecuente para la distribución de certificados X.509. Es típico que una autoridad de certificación distribuya su certificado raíz con este formato.
*.key Formato para la distribución de claves privadas.

OpenSSL

OpenSSL (https://www.openssl.org) és una llibreria que permet als desenvolupadors de programes (com ara els de l'Apache) incloure un conjunt de funcions criptogràfica de propòsit general (és a dir, una "API" de criptografia) dins del codi font dels seus programes de manera que no s'hagin de preocupar dels detalls més tècnics sobre la seguretat de les comunicacions, ja que deleguen aquesta tasca en les funcions oferides per l'OpenSSL. L'OpenSSL també és, a més, una utilitat de línia de comandes que permet xifrar i desxifrar dades directament des del terminal, de forma interactiva, fent servir ja sigui criptografia de clau simètrica o bé asimètrica. De fet, el que ens interessarà més serà fer servir el segon tipus perquè ens permetrà crear certificats (és a dir, claus públiques certificades per una autoritat) associades al nostre servidor web i aconseguir així que aquest sigui segur pels nostres usuaris (és a dir, que les seves visites no puguin ser "esnifades" per hackers i que a més tinguin la garantia de no patir un atac "man-in-the-middle").Una eina alternativa a OpenSSL és el paquet GnuTLS (http://www.gnutls.org); una altra és LibreSSL (http://www.libressl.org); totes elles són lliures i multiplataforma.

Per aconseguir xifrar (amb l'algoritme que sigui) les comunicacions a través d'una xarxa TCP/IP, l'OpenSSL afegeix a sobre de la capa de transport TCP (i sota la capa d'aplicació, gestionada en el nostre cas pel protocol HTTP) una capa intermitja on s'implementa el protocol SSL (protocol obsolet) o TLS (versió millorada de SSL). D'aquí que les comunicacions webs segures s'anomenin "HTTPS" (HTTP + SSL).

A continuació s'indiquen les comandes OpenSSL més habituals (¡i mínimes: hi ha molts paràmetres interessants que no s'han indicat!) relacionades amb la gestió de certificats (moltes d'elles les haurem d'utilitzar quan vulguem implementar un servidor web segur; veure següent apartat). Totes aquestes comandes les haurem d'executar com a "root" dins de la carpeta "/etc/ssl" (a Ubuntu) o "/etc/pki/tls" (a Fedora); allà es pot veure com hi ha una subcarpeta per les claus privades que haguem creat (subcarpeta "private") i pels certificats (subcarpeta "certs"). ¡MOLT IMPORTANT!: per a què les comandes següents funcionin correctament, la màquina on s'executaran ha de tenir un nom DNS ja definit I AQUEST NO PODRÀ CANVIAR DESPRÉS. Això és degut a què els certificats estan estretament vinculats al nom de la màquina, i si aquest canvia el certificat deixarà de ser vàlid. Si no tenim nom DNS, a l'aula haurem de fer servir la ip del servidor directament.

*Creació d'una sol.licitud de certificat (un "CSR") -i de la clau privada associada, que quedarà gravada dins de "/etc/ssl/private"-. Aquest CSR s'haurà d'enviar seguidament 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; per a què tot sigui més segur, aquest fitxer ".crt" l'haurem de gravar dins de "/etc/ssl/certs" i donar-li -a l'igual que a la clau privada- permisos de 644 (és a dir, cal fer *chmod 644 fitxer.crt*) tot procurant que el seu propietari sigui l'usuari "root" (si no, cal fer *chown root:root fitxer.crt*)):
NOTA: Atenció: no podem enviar el CSR a qualsevol CA: només les reconegudes per la nostra màquina generaran un fitxer "crt" útil. Les CA més importants ja estan reconegudes gràcies a què el seu certificat arrel està per defecte instal.lat a les distribucions més importants gràcies a la instal.lació automàtica del paquet "ca-certificates" (a Ubuntu aquests certificats arrel es troben dins la carpeta "/usr/share/ca-certificates", i a Fedora es troben tots continguts dins d'un únic fitxer anomenat "ca-bundle" a la carpeta "/etc/pki/ca-trust"), però això pot no ser el cas de la CA que volguem fer servir.

openssl req -new -newkey rsa:2048 -nodes -keyout clauPriv.key -out solCert.csr

La comanda anterior demanarà una sèrie de preguntes: la resposta més important és a la de "Common Name": allà s'ha d'escriure obligatòriament el nom DNS complet del nostre servidor, el qual ha de ser igual sempre al valor de la directiva "ServerName"del VirtualHost que utilitzarà aquest certificat. La "challenge password" s'ha de deixar en blanc.
NOTA: Si volem que el certificat no només sigui vàlid per "www.elmeudomini.com" (per exemple) sinò també per qualsevol nom acabat en "elmeudomini.com", s'haurà d'escriure "*.elmeudomini.com".

El significat dels paràmetres anteriors és el següent:

req -new : indico que vull crear un nou CSR

-newkey rsa:2048 : genero una clau privada de tipus RSA amb una longitud de 2048 bits (a més llarga, més segura) que s'utilitzarà per la creació del CSR.

-nodes : la clau privada no estarà encriptada. Si no s'escrigués aquest paràmetre, la clau s'encriptaria amb una contrasenya ("passphrase") que demanaria 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. Si ens trobéssim en la situació de tenir una clau privada amb una contrasenya i la volguéssim eliminar, amb la següent comanda podríem crear-ne una nova clau privada equivalent a l'original però sense passphrase: *openssl rsa -in clauPriv.key -out novaClauPriv.key*

-*keyout clauPriv.key* : indico el nom del fitxer (amb extensió .key) que representa la clau privada
-*out solCert.csr* : indico el nom del fitxer (amb extensió .csr) que representa el CSR a enviar
-*days nº* : un paràmetre molt comú és aquest, que indica el temps que seran vàlids els fitxers generats (arribat el moment, caducaran)

La comanda anterior generava una clau privada per tal de poder signar el CSR corresponent. Però si volem podem fer els dos passos (crear la clau privada i crear el CSR) de forma separada. D'aquesta manera, si ja tinguéssim la clau privada creada d'abans, només caldria realitzar el segon pas:

openssl genrsa -nodes -out clauPriv.key 2048 #Crea una clau privada

openssl req -new -key clauPriv.key -out solCert.csr #Crea el CSR signant-lo amb aquesta clau privada

En qualsevol cas, es pot verificar el contingut del CSR generat amb la comanda...:

openssl req -noout -text -in solCert.csr

...i el de qualsevol clau privada:

openssl rsa -noout -text -in clauPriv.key

*Creació d'un certificat autosignat : com abans, es pot generar al mateix temps una clau privada i la clau pública que serà signada amb ella (generant així el certificat autosignat)...

openssl req -new -x509 -newkey rsa:2048 -nodes -keyout clauPriv.key -out cert.crt

...o bé generar el certificat a partir d'una clau privada (prèviament existent o bé creada amb la comanda indicada una línies més adalt) amb la comanda següent:

openssl req -new -x509 -key clauPriv.key -out cert.crt

En realitat, com es pot veure, l'únic paràmetre que canviar respecte a la comanda que generava el CSR és "-x509", el qual serveix per indicar que la sol.licitud de certificat que es genera internament vagi dirigida a la clau privada indicada (realitzant així l'autosignatura). De fet, si tinguéssim un CSR ja creat, també podríem generar el certificat a partir d'ell, així:

openssl req -x509 -in solCertcsr -out cert.crt -signkey clauPriv.key

En tot cas, es pot verificar el contingut del certificat retornat amb la comanda:

openssl x509 -noout -text -in cert.crt

Si en algun moment volguéssim comprovar si un certificat està vinculat (o no) a una clau privada determinada, podríem comparar si el seu "modulus" (una ristra de lletres i números identificatius) és el mateix executant les dues comandes següents:

openssl x509 -noout -modulus -in cert.crt ; openssl rsa -noout -modulus -in clauPriv.key

Una alternativa més "professional" que utilitzar un certificat autosignat és convertir una màquina de la nostra organització en una CA per a què sigui ella la què signi els nostres certificats. Això no ho veurem a classe, però cal saber que és una altra possibilitat que ofereix l'OpenSSL. Muntar una CA pròpia no ens estalviarà el missatge d'advertència als navegadors si no fem res, però si prèviament instal.lem el certificat signat a les màquines clients (cada navegador té un procediment diferent des del seu menú d'opcions), llavors sí que ja no sortirà aquest missatge (això té sentit en un entorn controlat, com una LAN).

La suite OpenSSL té moltíssimes comandes més que no veurem per no sortir-nos del temà. No obstant, a continuació mostrem una brevíssima taula amb algunes de les utilitats més habituals:

openssl help	Per obtenir una llista de totes les comandes possibles es pot executar: *openssl list-standard-commands* *openssl list-message-digest-commands* *openssl list-cipher-commands*
openssl dgst -md5 /ruta/fitxer	Obté el hash MD5 del fitxer. Altres algoritmes són: *sha1*,*ripemd160* i més
echo bla | openssl passwd -stdin -1	Genera una contrasenya MD5 (-1) a partir del que arriba per l'entrada estàndar
openssl enc -e -bf -in /ruta/fitxerin -out /ruta/fitxerout	Xifra un arxiu amb Blowfish. Altres algoritmes són *3des*, *aes128*... Demana la clau interactivament, la qual es guarda encriptada a /etc/secure/passwd.enc.bf Es pot afegir el paràmetre -base64 (-a) per transformar el binari xifrat en texte ascii.
openssl enc -d -bf -in /ruta/fitxerin -out /ruta/fitxerout	Desxifra un arxiu amb Blowfish. Demana la clau interactivament. Si es va crear armadura Ascii al xifrar, s'haurà d'afegir el paràmetre -a aquí també.
openssl s_client -connect ipserv:nºp [-CAPath /ruta/certificatCA]	Funciona com un Netcat però conectant-se a un servidor segur (Postfix, Apache, etc). Les opcions -showcerts y -debug també valen la pena.

Implementació de servidors web segurs (HTTPS)

El mòdul "ssl" és una interfaç entre l'Apache i la llibreria OpenSSL que permet que el primer pugui fer servir la segona per autentificar màquines, gestionar firmes i certificats digitals i encriptar la comunicació amb els clients, entre altres aspectes relacionats amb la seguretat de les conexions. A l'Ubuntu aquest mòdul s'instal.la per defecte juntament amb el propi servidor Apache (està dins del paquet "apache2-bin") però a Fedora cal instal.lar-lo específicament, amb el paquet "mod_ssl".
NOTA: També existeix el mòdul "gnutls" (a l'Ubuntu el paquet que l'instal.la es diu "libapache2-mod-gnutls" i a Fedora, "mod_gnutls") similar en objectius al "ssl" però que utilitza GnuTLS.

Per crear un certificat autosignat útil per Apache es poden executar directament les comandes adients oferides per la suite OpenSSL (veure apartat anterior) o bé utilitzar algun programa "wrapper" (embolcall) que oculta la complexitat del procés, facilitant la feina. Els programes OpenVPN o Postfix, per exemple, ofereixen cadascun un "wrapper" propi que podríem utilitzar si tinguéssim instal.lat algun dels dos, però el més habitual és fer servir un "wrapper" genèric que ofereix la pròpia distribució. A Fedora tenim el paquet "crypto-utils", el qual conté la comanda *genkey* i a Ubuntu hi ha el paquet "ssl-cert", el qual conté la comanda *make-ssl-cert* : la primera s'executa simplement indicant com a paràmetre el nom DNS complet del nostre servidor (es posarà en marxa un quadre interactiu de preguntes que permet creat tant CSRs com certificats autosignats) i la segona s'executa així: *make-ssl-cert /usr/share/ssl-cert/ssleay.cnf server.crt* (se'ns preguntarà pel nom DNS del servidor i es guardarà el certificat i la clau privada dins l'arxiu "server.crt" -¡sí, dins el mateix fitxer!-; l'arxiu "ssleay.cnf" fa de plantilla). De totes maneres, nosaltres seguirem la forma "estàndar" de creació de certificats explicada a l'apartat anterior, fent servir directament les comandes OpenSSL-

En qualsevol cas, per fer que l'Apache utilitzi el certificat (autosignat) que haguem creat, primer cal habilitar el SSL; a l'Ubuntu això simplement es fa així: *a2enmod ssl* Seguidament caldrà afegir a l'arxiu de configuració del VirtualHost "a assegurar" les següents línies (sempre dins de la secció <VirtualHost> a la mateixa alçada que la directiva *DocumentRoot*; es pot fer servir l'arxiu "default-ssl.conf" com a plantilla si es vol):

SSLEngine on
SSLCertificateFile /ruta/al/certificat/cert.crt
SSLCertificateKeyFile /ruta/a/la/clau/privada/clauPriv.key

NOTA:Si estiguéssim en el cas de tenir un fitxer que inclogui tant el certificat com la clau privada tot en un (això també és possible), les línies a escriure serien llavors:
SSLEngine on
SSLCertificateFile /ruta/al/fitxer/que/inclou/certificat/mes/clau/privada/server.crt

En realitat, el VirtualHost "default-ssl" de l'Ubuntu ja ver configurat per defecte llest per funcionar (escoltant al port 443; comproveu que la directiva *Listen 433* a "ports.conf" està activa). Només cal assegurar-se de les rutes a utilitzar pel certificat i clau privada del servidor i ja està. Podràs observar que en aquest fitxer (¡i en el fitxer de configuració del mòdul, ssl.conf!) hi han més directives relacionades amb l'SSL, que no estudiarem.

NOTA: Antigament, la tecnologia TLS no podia oferir múltiples VirtualHosts des d'una mateixa IP perquè el servidor no sabia quin VirtualHost estava sol.licitant el client a la capçalera"Host" (ja que aquesta també estava encriptada) i per tant no sabia quin certificat -el corresponent a cada lloc- oferir. El que es feia llavors era tenir un únic certificat compartit per tots els llocs amb la mateixa IP (una sol.lució gens professional). Afortunadament, avui dia aquest problema ja no hi és gràcies a la tecnologia Server Name Indication (https://en.wikipedia.org/wiki/Server_Name_Indication)

Un cop configurat el VirtualHost segur i activat mitjançant *a2ensite nomVH*, es pot comprovar, abans de reiniciar el servidor, si tots els fitxers de configuració de l'Apache estan correctament escrits amb la comanda *apache2ctl configtest* . Si tot és correcte, podrem iniciar el servidor i accedir al lloc segur escrivint a la barra del navegador una direcció tal com https://www.nom.servidor (¡important indicar la "s" a "https"...si no l'Apache es queixarà de què volem accedir a un lloc HTTPS fent servir el protocol HTTP!; el port 443 no cal afegir-lo perquè indicant"https" ja se sobrentén igual que se sobrentenia el 80 quan accedíem a HTTP). De totes formes, si volguéssim redireccionar automàticament al lloc HTTPS un usuari que hagués escrit "http" en comptes de "https" a la barra de direccions, el més fàcil és utilitzar la directiva *Redirect* dins del VirtualHost "estàndar" per reencaminar-lo a l'altre; és a dir, fer així:

```
<VirtualHost *:80>
        ServerName nom.dns.servidor
        Redirect permanent / https://nom.dns.servidor    #Aquí es pot posar la IP també
</VirtualHost>

<VirtualHost _default_:443>
        ServerName nom.dns.servidor
        DocumentRoot /var/www/html/ssl
        SSLEngine On
        # etc..
</VirtualHost>
```

Una altra manera, més complicada, d'aconseguir el mateix seria amb el mòdul "Rewrite", tal com s'explica aquí: https://wiki.apache.org/httpd/RewriteHTTPToHTTPS