Conceptes de PKI

Comunicació segura entre extrems

Més enllà de les diferents tècniques que hi puguin haver (com ara LUKS, Gocryptfs, Age, etc) per assegurar la informació en repòs (és a dir, emmagatzemada en un dispositiu), quan hi intervé la comunicació entre dos extrems d'una xarxa cal afegir-hi l'assegurament de la pròpia comunicació en sí. És a dir, a banda del xifrat dels propis fitxers/carpetes que es puguin transmetre per un canal, el més òptim i segur seria que fos el propi canal el que estigués xifrat en sí, formant doncs un "túnel" segur per on pogués passar tota la informació per dins sense que aquesta calgués que estigués xifrada de per sí!.

Això és el que pretén, per exemple, el protocol TLS: aquesta tecnologia se centra en el xifratge del canal per on es transfereix la informació enlloc de centrar-se en el xifratge de la informació en sí . Això vol dir que, tal com acabem de dir, si es fa servir TLS en una connexió, es xifra tot el que hi passa “dins” d’aquesta connexió (comandes de petició/resposta, contrasenyes d´accés, capçaleres dels missatges, els propis missatges, etc).

El protocol TLS és un protocol híbrid (és a dir, que utilitza tant criptografia simètrica com asimètrica) el qual es basa en una infraestructura d’autoritats centrals de certificació reconegudes pel nostre sistema que possibiliten que aquest pugui confiar en l’autenticitat dels certificats oferts pels sistemes remots amb els quals es vulgui comunicar. Per entendre la frase anterior cal, doncs, conéixer primer uns quants conceptes importants, començant pels de "certificat" i "autoritat de certificació".

Què és un certificat? I una "autoritat de certificació (CA)"?

Ja sigui fent servir la criptografia de clau pública per xifrar tot el missatge o bé només per intercanviar de forma segura la clau simètrica que es farà servir per xifrar, tenim un problema: cal establir l'autenticitat de la clau pública del receptor per tal de què l'emissor envïi el missatge amb la tranquil·litat d'estar protegit contra impostors i atacs "MitM": si la clau pública fos fraudulenta (és a dir, associada a la clau privada d'un intrús), els missatges que l'emissor creu "secrets" podrien ser llegits per l'intrús que els interceptés (en lloc del destí legítim) sense cap protecció.

Una solució a aquest problema són els **certificats**. Un certificat simplement és una clau pública (és a dir, a la pràctica, un fitxer) que, bàsicament:

a) Està vinculat a un determinat nom, identificador del receptor (a partir d'ara l'anomenarem "entitat"). A la pràctica, aquest nom és de tipus DNS (més endavant veurem els mecanismes utilitzats per fer que un nom DNS serveixi per identificar una determinada entitat)

b) Aquesta vinculació ha estat signada internament per un tercer (l'anomenat "Certificate Authority" o **CA**) que garanteix ("certifica") que, efectivament, aquesta vinculació és correcta i, per tant, l'entitat és qui diu que és.

Certificats arrel, autosignats i "trust stores"

Ja sabem que "signar" un document (en aquest cas, una clau pública) implica fer servir una clau privada pròpia i, per tant, una signatura determinada només la pot realitzar un determinat agent (en aquest cas, una CA). Si l'emissor d'un missatge manté un magatzem de claus públiques de diverses CA podrà comprovar que la signatura associada al certificat a fer servir es correspongui efectivament a alguna d'aquestes CA "confiables"; si és així, el certificat llavors es donarà com a vàlid i es procedirà a l'enviament segur del missatge.En realitat, però, tècnicament l'emissor no té un magatzem de claus públiques de CAs sinó un magatzem de certificats de CAs, certificats que estan autosignats per elles mateixes (amb la seva pròpia clau privada) per evitar suplantacions. Aquests certificats (utilitzats, com hem dit, per verificar la signatura de la pròpia CA dins de certificats d'altres) són els que s'anomenen "**certificats arrel**".

Mantenir la integritat d'aquests magatzems de certificats arrel (anomenats també "**trust stores**") és clau ja que, com aviat veurem, és molt fàcil generar-se un "**certificat autosignat**" propi (és a dir, un certificat signat per la pròpia clau privada i no pas per la de cap entitat externa) sense cap intervenció de tercers, autoassignant-se, per tant, qualsevol nom. Els "trusts stores" permeten identificar els certificats autosignats ja que la seva signatura no serà reconeguda (si no han sigut manipulats per introduïr-ne certificats arrel fraudulents).

En un sistema hi poden haver diversos "trust stores", depenent de per a què es facin servir. La majoria d'ells venen integrats dins dels navegadors web, ja que aquests programes són l'entorn on els certificats se solen fer servir més habitualment (el protocol HTTPS utilitza la tecnologia TLS per assegurar la comunicació entre els navegadors i els servidors web i aquesta basa el seu funcionament en l'ús de certificats) però també existeixen, de forma integral, a nivell de sistema operatiu . Els "trust stores" més importants són mantinguts bàsicament per quatre organitzacions:

Apple (http://www.apple.com/certificateauthority/ca_program.html)	Manté el "trust store" de iOS i macOS
Microsoft (https://aka.ms/RootCert)	Manté el "trust store" de Windows
Google (https://g.co/chrome/root-policy)	Manté el "trust store" de Chrome (en totes les plataformes excepte iOS)
Mozilla (https://www.mozilla.org/en-US/about/governance/policies/security-group/certs)	Manté el "trust store" dels seus productes i, degut a la seva transparència, també és la base d'altres "trust stores", com els propis de moltes distribucions de Linux, usats a nivell de sistema operatiu integralment.

NOTA: En general, tot allò que no són navegadors web (és a dir, altres programes nadius, com per exemple la comanda *curl*, entre altres) fan servir el "trus store" integral del sistema operatiu. En tot cas, els "trust stores", ja sigui de sistema o els propis de cada navegador, ja venen incorporats de sèrie i són actualitzats mitjançant el mecanisme estàndard d'actualització de paquets.

Hi ha més de 100 CAs incloses en els "trust stores" anteriors: Symantec, DigiCert, Entrust, Let's Encrypt...són només algunes. El document "Baseline Requirements" de l'organització CA/Browser Forum (https://cabforum.org) especifica els requisits i regles que ha de complir un empresa o organització per poder actuar com a CA (és a dir, per començar a emetre certificats d'entitats signats per ella) i esser inclosa (o més tècnicament parlant, esser inclós el seu certificat autosignat) dins d'un "trust store".

En resum: en el cas particular, per exemple, d'un servidor web segur (HTTPS), el certificat que ofereix als navegadors quan un usuari visita la seva web garanteix que aquest servidor sigui realment qui diu ser, sempre i quan el seu certificat vingui firmat per una autoritat de certificació confiable pel navegador. Tal com hem dit, per a què la firma d'una autoritat de certificació sigui confiable pel navegador, aquest ha de tenir integrat en la seva llista de certificats arrel reconeguts el corresponent certificat arrel. Mitjançant aquest mecanisme, els navegadors sempre validen qualsevol certificat del món signat per qualsevol de les CA presents en aquesta llista i, en cas positiu, llavors procedeixen a enviar, de forma segura, la petició HTTP escaient en cada cas. En el cas de que visitéssim un servidor HTTPS que oferís al navegador un certificat firmat per una CA no reconeguda (com és el cas dels anomenats "certificats autosignats") es mostraria un missatge d'advertència, ja que aquest fet pot ser sospitós: rebre un certificat autosignat és equivalent a que algú digui que es pot confiar en ell perquè ell mateix ho certifica...si més no és estrany i, molt probablement, perillós.

NOTA: De totes formes, fins l'arribada de Let's Encrypt (veure més avall), els certificats autosignats eran l'única manera de disposar gratuïtament d'un certificat, i ho continua essent si no tenim registrat cap domini DNS (veure més avall).

Estructura interna d'un certificat

Un navegador (i, en general, qualsevol programa) comprova diverses coses del 'certificat obtingut de l'altre extrem per donar-lo per vàlid i començar la comunicació encriptada:

*-Ha d'estar signat per algú en qui es confia (és el que hem comentat)

*-No deu haver expirat (tots tenen una data d'expiració)

*-Ha d'haver sigut creat pel mateix nom DNS de l'entitat al què s'està accedint. Com ja hem dit, cada certificat, en generar-se, es vincula a un nom DNS de l'entitat per al qual serà vàlid i només aquest; si en connectar a un extrem s'obtingués un certificat on el nom del DNS allà registrat no concordés amb el nom DNS de l'entitat (servidor) al qual es vol accedir, la connexió no es farà. En altres paraules: es fa servir el nom DNS d'una entitat (prèviament registrat a la IANA) com a identificador del propietari del certificat corresponent a aquesta entitat.

Tradicionalment, aquest nom DNS estava indicat dins del certificat formant part d'una dada interna anomenada "Distinguished Name" ("**DN**"); aquesta dada identificava el propietari del certificat i incloia no només el nom DNS (el qual, concretament, s'indicava en un camp concret del DN anomenat "Common Name", "**CN**") sinó també altres dades com ara la ciutat, país i organització del propietari (tot i que, en realitat, totes aquestes dades ja des del principi no servien a la pràctica per res perquè són un residu d'un projecte anomenat X.500 que no tenia res a veure amb la web sinó amb un llistat telefònic mundial que no es va arribar a implementar mai). No obstant, actualment, ni tan sols el camp "CN" és rellevant perquè ara el nom DNS (o noms!, perquè es pot escriure el comodí "*" per indicar més d'un subdomini, com per exemple "*.domini.com") cal que estigui indicat en la dada interna "Subject Alternative Name" ("**SAN**"), apareguda després d'una modificació (una "extensió") que l'organització CABForum va realitzar fa anys de l'estàndard de creació de certificats (que estudiarem de seguida).

NOTA: En un certificat hi poden haver més d'una dada "SAN" i no totes han de tenir com a valor un nom DNS; també poden contenir adreces IP, adreces de correu i URIs. D'aquesta manera, els certificats poden servir per identificar altres tipus d'"entitats" més enllà de màquines, com ara persones o recursos.
NOTA: No només els servidors són susceptibles d'enviar certificats als clients per a què aquests verifiquin la seva autenticitat; també pot ser a l'inrevés: els clients també poden tenir certificats per enviar-los a l'altre extrem (en aquest cas el servidor). Aquesta situació ocorre quan volem que només hi hagi clients determinats (els autenticats mitjançant el seu certificat) que puguin contactar amb el servidor i no clients indiscriminats. Un exemple molt habitual (encara que en aquest cas no s'usen certificats sinó simplement claus públiques perquè no existeix el concepte de CA) és el de les connexions SSH on volem que només clients concrets puguin accedir a un servidor SSH.

El format que defineix l'estructura interna de les dades presents dins certificats (com per exemple els camps "DN" o "SAN", entre altres), ja estiguin distribuïts per una CA o autosignats (és igual), pot variar, encara que el més habitual és el format UIT-T **X.509v3**, el qual es basa al seu torn en un estàndard de definició de dades anomenat **ASN.1** La codificació concreta de les dades ASN.1 incloses dins d'un certificat X.509 pot ser binària (en el que s'anomena format "**DER**") o basada en Base64 (en el que s'anomena format "**PEM**"); els primers solen tenir extensió ".der" i els segons solen tenir extensió ".pem", "crt" o ".cer". En tot cas, un certificat X.509 està estructurat internament en vàries seccions, com són:

*Número de sèrie
*Data de creació i d'expiració
*Còpia de la clau pública de l'entitat certificada
*Signatura de l'autoritat certificadora (CA)
*Nom i dades burocràtiques identificatives de l'entitat certificada, incloent el seu domini DNS

NOTA: A més dels certificats X.509 amb forma de fitxers individuals, existeixen altres formats envolvents (amb la seva corresponent extensió) que serveixen per incloure diversos elements (varis certificats, claus, etc) tot en un fitxer. Aquests formats envolvents estan definits en una família d'estàndards anomenats PKCS (també basats en ASN.1 i per tant, codificables tant en "PEM" com en "DER"), i en podem destacar els següents:

* **PKCS#7** (també anomenat CMS) : Conté diversos certificats, un rera l'altre (pertanyents a una mateixa cadena de certificació, llegir més endavant). D'ús comú en entorns Java, les seves extensions comunes són ".p7b" i ".p7c"
* **PKCS#12** : Conté una (o més) clau/s privada/es (normalment xifrada/es amb una passphrase) juntament amb el seu corresponent certificat (o cadena de certificats PKCS#7). D'ús habitual en productes Microsoft, les seves extensions comunes són ".p12" i ".pfx"
***PKCS#8** : Conté només una clau privada (normalment xifrada amb una passphrase) i metadades relacionades com el seu tipus, etc. Les seves extensions comunes són ".pem" i ".key"
***PKCS#10** : Conté la sol·licitud de certificació a enviar a una CA (l'anomenat "CSR"). La seva extensió comuna és ".csr"

Procés de creació d'un certificat

Com ja hem dit, qualsevol individu o institució pot signar claus públiques (és a dir, generar certificats digitals) i convertir-se així en una autoritat de certificació, però si no és reconegut pels que interactuïn amb aquest certificat (és a dir, si no s'ofereix un certificat arrel propi), el valor de el mateix és pràcticament nul (servirà per fer proves o per a un grup d'usuaris que confiïn en la nostra màquina). Per això les autoritats de certificació han de ser reconegudes oficialment com a institució certificadores de manera que el seu certificat arrel (i com a conseqüència, la seva signatura) pugui ser reconeguda com a fiable, transmetent aquesta fiabilitat als certificats de tercers signats per l'esmentada institució.

Per això el més habitual com a administradors de sistemes informàtics serà simplement enviar la nostra clau pública (o més tècnicament, un fitxer específic anomenat "sol·licitud de certificació" que inclou la clau pública) a una autoritat de certificació oficial perquè sigui aquesta la que signi la nostra clau i ens retorni el certificat generat pertinent, el qual ens permetrà implementar, per exemple, un lloc segur sense que els clients tinguin problemes de confiança. Antigament només existien CA comercials, les quals cobraven una certa quantitat de diners per realitzar la signatura de la nostra clau pública; exemples són les empreses Thawte, Comodo, Verisign, Ancert, GlobeSSL o RSA. Afortunadament, des de fa un temps hi ha més CAs que realitzen aquest procés gratuïtament, com ara **Cacert** (http://www.cacert.org) i, sobretot, **Let's Encrypt** (https://letsencrypt.org). L'únic requisit que necessiten aquestes CA gratuïtes (com qualsevol altra CA, de fet) és que disposem d'un domini DNS registrat a Internet, ja que el vincle entre el certificat i el propietari de la mateixa el realitzen a través d'aquesta dada.

> NOTA: A nivell legal, les administracions públiques que otorguen validesa i oficialitat a les autoritats de certificació són la Fàbrica Nacional de Moneda y Timbre, el Ministerio de Industria, Turismo y Comercio o la Agència Catalana de Certificació, entre altres.
> NOTA: Les CAs poden realitzar més tasques a més de generar certificats a partir de sol.licituds, com ara renovar-los quan s'arribi a la seva data d'expiració, desactivar-los o fins i tot revocar-los definitivament si hi hagués algun problema (com per exemple el robatori de la clau privada usada per signar) mitjançant o bé el mètode CRL o bé el mètode **OCSP**, més modern En parlarem més endavant

---

Més en detall, si volem tenir un certificat signat per una determinada CA que validi que, per exemple, el nostre servidor HTTPS "www.domini.com" és, efectivament, nostre (no cal que sigui un servidor HTTPS, però és el cas més habitual...recordem que un servidor HTTPS no és res més que un servidor HTTP al qual se li ha "acoplat" el protocol TLS per assegurar-lo), els passos que cal seguir són els següents:

1.- Creació del parell de claus del servidor. En la versió actual del protocol TLS s'admeten claus de tipus RSA (amb el format PSS un mínim de 2048 bits) o bé claus de tipus el·líptic, com ara ECDSA o EdDSA (ambdues amb un mínim de 256 bits i basades en diferents algoritmes matemàtics, com "secp256k1" o "prime256v1" o també "Curve25519"). Veurem properament com realitzar aquest pas (en el servidor) amb l'eina OpenSSL

2.-Creació del CSR del servidor: Aquest fitxer bàsicament està format per la clau pública generada a l'apartat anterior (vinculada unívocament amb una sola clau privada, recordem-ho) juntament amb una sèrie de dades administratives del servidor que caldrà afegir-hi per a què, quan s'envïi aquest CSR a la CA triada (veure següent pas), aquesta pugui validar correctament que el propietari d'aquesta clau pública és qui diu que és. Concretament, la dada a indicar més important en aquest sentit és el nom DNS (o noms, perquè hi poden haver varis) del servidor (en el nostre exemple, "www.domini.com"), ja que aquesta dada és la que la CA utilitza per comprovar (de seguida veurem com) la vinculació entre el CSR del servidor en qüestió i la identitat real d'aquest servidor. Veurem properament com realitzar aquest pas (en el servidor) amb l'eina OpenSSL

3.-Enviament del CSR del servidor a la CA triada: Normalment aquest pas es fa via formulari de contacte o email. En tot cas, això dependrà del mètode que indiqui la CA escollida en concret.

4.-Entrega al servidor per part de la CA del seu certificat: Normalment aquest pas es fa via email. El certificat contindrà les mateixes dades que el CSR (és a dir, la clau pública del servidor i les seves dades administratives incloent els noms DNS als quals estarà vinculat) només que, a més, tindrà una data de validesa i, sobre tot, estarà signat per la clau privada de la CA. La signatura prova que el

servidor propietari del CSR (ho és perquè el CSR està vinculat a una clau privada que només pot tenir ell) és el servidor amb el/s nom/s indicat/s al CSR.

NOTA: En el cas dels certificats autosignats, el pas 3 òbviament s'omet i la signatura feta al pas 4 es realitza mitjançant la pròpia clau privada del servidor generada al pas 1
NOTA: Ja hem comentat que el problema dels certificats autosignats és que els clients que es connectin al servidor segur no coneixeran la signatura, així que no podran confiar en què la informació aportada pel certificat sigui autèntica. És per això que, per exemple, en els navegadors, quan es troben una situació així, apareix un missatge d'error. Aquest mateix problema, d'altra banda, també apareix quan la signatura d'una determinada CA tampoc és reconeguda pels clients degut a què no la tenen dins de la llista de CAs "de confiança" (és a dir, tècnicament parlant, no tenen incorporat el seu certificat arrel dins del seu "trust store").

5.-Configuració segura del servidor: Un cop el servidor ja té el certificat (i la seva clau privada, que mai haurà sortit d'ell), el darrer pas és indicar al software en qüestió (per exemple, si estem posant en marxa un servidor HTTPS, podria ser Apache2 o Nginx), de la forma que aquest software tingui estipulat (això ja dependrà de com es configuri cadascun), aquests dos fitxers per tal de, a partir de llavors, fer-los servir (i així, doncs, poder establir una comunicació segura). És a dir, al final l'objectiu de tot aquest procés és poder disposar de dos fitxers: una clau privada i un certificat; a partir d'aquí, el software que es vulgui implementar n'haurà fer ús d'aquests dos fitxers i ja està.

Validació d'un CSR

Després de rebre un CSR i de verificar que la clau pública allà embeguda es correspongui efectivament a la del servidor remitent (gràcies a la vinculació amb la seva clau privada), el més important que una CA ha de fer és esbrinar si el nom DNS (o noms) indicats en el CSR realment és/són el/s nom/s veritables del servidor remitent. Segons com es faci aquesta verificació, ens podem trobar amb diferents "categories" de certificats:

***Certificats DV** (Validació de domini): Aquest és el nivell més bàsic de validació. La CA només assegura que el servidor que proporciona el CSR és també el propietari d'un domini DNS específic. Això ho fa consultant la informació continguda a la base de dades pública WHOIS i enviant llavors un email de confirmació a l'adreça que figura al registre WHOIS del domini. En tot cas, aquests mètodes confirmen que el domini està controlat per la part que sol·licita el certificat.

NOTA: Let's Encrypt, que és una CA molt popular perquè emeten certificats a tothom que tingui un domini sense cap cost econòmic, només ofereix certificats DV. Això es deu principalment a que no poden automatitzar l'emissió dels altres tipus de certificats (ja que requereixen esforç humà, i això requeriria pagar algú).
NOTA: El protocol ACME, desenvolupat per Let's Encrypt, millora el mètode de verificació descrit al paràgraf anterior amb una automatització més eficient: en comptes de confirmar la identitat via correu electrònic, una CA ACME emet un "repte" (també anomenat "desafiament") al servidor sol·licitant de certificació que l'ha de completar per demostrar que controla un domini. El tipus de repte a realitzar no està especificat a l'estàndard ACME, però els més habituals inclouen publicar un número aleatori en un URL determinada (el desafiament HTTP) i col·locar un número aleatori en un registre DNS TXT (el desafiament DNS).
NOTA: És important tenir en compte, no obstant, què demostra realment un certificat DV; se suposa que ha de demostrar que l'entitat que sol·licita el certificat és propietari del domini rellevant però, de fet, l'únic que demostra és que, en algun moment (si més no, cada tres mesos, que és el termini de validesa que tenen els certificats de Let's Encrypt), l'entitat que sol·licita el certificat pot llegir un correu electrònic o pot configurar el DNS o pot servir un determinat valor mitjançant HTTP, però la seguretat subjacent d'aquests mètodes no és excel·lent.

***Certificats OV** (Validació d'organització): Aquest tipus de certificat autentica el propietari del lloc i requereix informació comercial legítima per a aquesta empresa. El seu procés de validació és més llarg i detallat: la CA no només verifica el fet de ser propietari del domini, sinó també el fet de ser el propietari jurídic de l'empresa, el qual ha de constar en les bases de dades oficials de registre d'empreses. Així doncs, els estafadors no poden obtenir un certificat OV perquè la seva organització no es pot validar. El principal avantatge d'obtenir un certificat OV és que les dades legítimes de l'empresa apareixeran al mateix certificat, donant més confiança que tant el lloc web com l'empresa. són de bona reputació. Els OV solen ser utilitzats per corporacions, governs i altres entitats que volen oferir una capa addicional de confiança als seus visitants.

*__Certificats EV__ (Validació ampliada): El procés de verificació dels certificats OV no és coherent entre les diverses CA. Per això, el "CABForum" va introduir els certificats EV. Aquests certificats contenen la mateixa informació que la resta però exigeixen estrictes requisits per verificar la identitat del propietari. El procés d'EV pot durar dies o setmanes ja que normalment inclou intercanvi de burocràcia en paper signada (amb bolígrafs) per responsables de l'empresa. A la pràctica, com que els navegadors (i la immensa majoria de programes, de fet) només requereixen una garantia de nivell DV (basada en proves ràpides i automatizades del control d'un domini), els certificats EV a penes s'utilitzen.

Renovacions

Si encara s'està utilitzant un certificat que està a punt de caducar, caldrà renovar-lo abans que això passi perquè si no els clients deixaran de confiar-hi. No hi ha cap procés de renovació estàndard; és a dir, no hi ha cap manera formal d'ampliar el període de validesa d'un certificat. En lloc d'això, el que es fa és simplement substituir el certificat que caduca per un de nou. Per tant, el procés de renovació és el mateix que el procés d'emissió: generar i presentar un CSR i complir qualsevol obligació de prova d'identitat.

Revocacions

No obstant, a vegades es pot desitjar que es deixi de confiar en un certificat abans fins i tot que arribi la seva data de caducitat (és a dir, es pot voler marcar un certificat com a "invàlid" per tal que ningú en continuï confiant). Això és necessari quan, per exemple, la clau privada associada ha sigut compromesa. Per aconseguir això, cal realitzar un procés anomenat "revocació".

A diferència de la caducitat, l'estat de revocació no es pot codificar al certificat. El client ha de determinar l'estat de revocació del certificat mitjançant algun procés "fora de banda". Com que això suposa una complicació afegida a la infraestructura de tot plegat, molt sovint el que s'acaba implementant, sobre tot a nivell de xarxa local fora de la web és una "revocació passiva"; és a dir, l'emissió de certificats amb una data de caducitat tan curta que la revocació no sigui necessària i que només amb la no-renovació ja sigui suficient. Sobre l'interval de temps que es considera "curt" per no renovar, això depèn, però les vint-i-quatre hores són força habituals en aquest tipus d'infraestructures internes fora de la web.

En el cas del certificats públics en Internet, hi ha dos mecanismes principals de revocació:

* __Protocol CRL__ : Es basa en oferir una llista pública signada de números de sèrie que identifiquen els certificats revocats. Aquesta llista es publica des d'un punt de distribució CRL, la URL del qual haurà d'estar inclosa al certificat. La idea és que els clients descarreguin aquesta llista i la consultin per obtenir l'estat de revocació quan verifiquen un certificat. No obstant, hi ha alguns problemes evidents amb aquest sistema; per exemple, els punts de distribució poden no estar disponibles en un moment donat (per exemple, degut a un atac DDoS, una falla d'Internet, etc) i això és un forat de seguretat molt important perquè els certificats revocats no podran ser consultats i, normalment en aquests casos, llavors els clients els accepten tots. A més, les llistes CRL poden ser gran i per aquest motiu, aquestes llistes, un cop comprovades, se solen guardar a la memòria cau dels navegadors durant uns quants dies (per no haver-les d'estar descarregant contínuament), cosa que també és un problema perquè en els clients poden no sincronitzar-se a temps.

* __Protocol OCSP__ : Permet als clients consultar a un "servidor OCSP" el número de sèrie d'un determinat certificat per obtenir el seu estat de revocació. Igual que el punt de distribució CRL, la URL del servidor OCSP cal incloure'l al certificat. Aquest mecanisme té els seus propis problemes; per exemple: el responsable del servidor OCSP pot veure quins llocs està visitant el client en funció de les comprovacions de l'estat del certificat que envïi (per tant, no hi ha privadesa). També afegeix una sobrecàrrega a cada connexió TLS perquè s'ha de fer una sol·licitud addicional per comprovar l'estat de revocació. D'altra banda, igual que CRL, molts navegadors assumeixen que un certificat és vàlid si la resposta OCSP està inactiva o retorna un error.

El "OCSP stapling" és una variant d'OCSP que soluciona el problema de privadesa que hi ha amb el protocol OCSP bàsic. En lloc que sigui el client qui pregunti al servidor OCSP, en aquest cas és el servidor propietari del certificat qui ho fa. La resposta OCSP és una acreditació signada (amb una data de caducitat brevíssima) que indica que el certificat del servidor en qüestió no es troba revocat. Llavors, aquesta acreditació "es grapa" al certificat del servidor ofert en el "TLS handshake" al client. Això proporciona al client un estat de revocació raonablement actualitzat sense haver de consultar directament al servidor OCSP, i el servidor pot utilitzar una resposta OCSP signada diverses vegades, fins que caduqui, reduint així la càrrega del servidor OCSP

Les sigles "**PKI**" (de "Public Key Infrastructure") sovint s'utilitzen per referir-se a tot el necessari (tant a nivell de hardware com de software) per mantenir les comunicacions segures mitjançant l´ús de certificats, CAs i signatures digitals. És a dir, és un terme "paraigües" per tot allò que cal (llibreries, comandes protocols, convencions, clients, servidors, mecanismes de descoberta, etc). per crear, distribuir, emmagatzemar, verificar, revocar i, en definitiva, gestionar i interactuar amb certificats, CAs i claus.

Cal tenir en compte que el sistema PKI no és perfecte; les vulnerabilitats més importants que pot patir són dues: que un malware alteri a la nostra màquina client el depòsit de certificats arrel de forma que tinguem importats certificats signats per CAs fraudulentes (això es fa sovint manualment per realitzar atacs de tipus "MitM" amb eines com Mitmproxy o similars) i que un atac als servidors d'una CA robi la seva clau privada, podent així l'atacant firmar les claus públiques de servidors fraudulents que no ho semblarien.

NOTA: La infraestructura PKI no és l'única que permet confirmar que la clau pública d'un extrem és de qui diu que és. Una altra tecnologia com a "Web of Trust" també persegueix el mateix objectiu, però en aquest cas d'una forma descentralitzada (és a dir, sense fer servir CAs). Només per saber-ho: el protocol TLS fa servir PKI mentre que el protocol GPG fa servir "Web of Trust"