7. Infraestructura de clau pública

En el capítol anterior s'ha vist com l'aparició de la criptografia de clau pública permetia solucionar el problema de la distribució de claus. Ara bé, quan dues entitats volen iniciar una comunicació segura, com poden saber quina és la clau pública que correspon a cada una?

Per cobrir aquesta necessitat, és a dir, per vincular identitats amb les seves respectives claus públiques, apareixen els certificats digitals i, amb ells, la infraestructura de clau pública (en anglès *Public Key Infrastructure, PKI*), un conjunt de processos, rols i especificacions que permeten gestionar aquests certificats. Així, la infraestructura de clau pública permet transferir informació de manera segura, tot oferint serveis d'autenticació, integritat i confidencialitat.

En aquest capítol, es detallen quines són les principals entitats que formen part d'una infraestructura de clau pública i la seva funció, es descriu què són els certificats digitals i quin n'és el seu cicle de vida, s'exposen diferents estàndards que es fan servir en l'àmbit de les PKIs i, finalment, es discuteix sobre els problemes que pateixen els desplegaments de PKIs a nivell pràctic.

7.1 Entitats d'una PKI

Una PKI està formada per diverses entitats que interactuen entre elles. Algunes d'aquestes entitats existeixen en tots els desplegaments de PKI, mentre que d'altres en són opcionals i només existiran en segons quin tipus de desplegament. En aquesta secció, descriurem les diferents entitats que poden formar part d'una PKI, tot indicant si la seva existència és opcional si n'és el cas.

7.1.1 Autoritat de certificació

Una **autoritat de certificació** (CA per les seves sigles en anglès, *Certification Authority*) és una entitat que certifica el lligam entre un parell de claus i una identitat. Aquesta certificació es realitza mitjançant la signatura digital d'una estructura de dades que conté tant la identitat com la clau pública corresponent. L'autoritat de certificació també és l'encarregada de revocar aquest lligam si n'és el cas.

L'estructura de dades que signa una autoritat de certificació es coneix amb el nom de certificat de clau pública. Així:

Un **certificat digital** de clau pública és una estructura de dades que vincula una clau pública a una identitat.

Formats de certificats digitals	Tot i que l'ús de certificats X.509 es troba molt estès, existeixen altres formats de certificats digitals, per exemple, els certificats PGP o els certificats SPKI.

Aquesta vinculació es produeix fent que una autoritat de certificació de confiança signi el certificat digital, que conté tant la clau pública com la identitat. Addicionalment, un certificat digital acostuma a contenir altres camps com ara informació sobre l'emissor, la validesa, identificadors dels algorismes involucrats en la signatura del certificat, etc. A la Secció 7.3.1 es descriuen amb detall els certificats digitals X.509, un dels formats més utilitzats avui en dia.

Figura 7.1: Abstracció del contingut d'un certificat digital.

Les autoritats de certificació disposen d'un document anomenat Declaració de Pràctiques de Certificació (CPS, de l'anglès, *Certification Practice Statement*) que estableix les normes que regeixen l'emissió i gestió de certificats d'aquella CA. Alhora, la CPS deriva de la Política de Certificació (CP, per les seves sigles en anglès, *Certificate Policy*), que és un document més general que descriu l'arquitectura de la PKI en què s'engloba la CA i els actors que hi participen, els usos permesos per als certificats que emet la CA, la política de generació de claus, l'ús de CRLs i diversos processos que du a terme la CA.

7.1.2 Autoritat de registre

Tot i que les tasques de registre poden ser dutes a terme per les autoritats de certificació, en certs escenaris (com ara quan les entitats finals es troben dispersades geogràficament o bé quan el número de entitats a certificar és molt gran) pot ser d'interès que se separin i siguin realitzades per una altra entitat.

> Una **autoritat de registre** (RA per les seves sigles en anglès, *Registration Authority*) és una autoritat que verifica la identitat de l'entitat que se certifica en un certificat digital.

Aquesta verificació de la identitat pot ser duta a terme, per exemple, a través de la personificació física i mostrant el document nacional d'identitat o el passaport.

Addicionalment, l'autoritat de registre pot dur a terme altres tasques. Per exemple, la generació de claus o bé la iniciació del procés de revocació del certificat, ambdues en nom de l'usuari final.

L'autoritat de registre és una entitat opcional en una PKI, ja que les tasques que realitza poden ser fetes per la pròpia CA.

7.1.3 Autoritat de validació

Amb la utilització pràctica de certificats digitals apareix la necessitat de poder-los revocar, és a dir, d'anul·lar-ne la seva validesa abans de la data de caducitat del propi certificat. Aquesta necessitat pot sorgir, per exemple, quan les claus privades corresponents s'han vist compromeses.

> Una **llista de revocació de certificats** o CRL (de l'anglès, *Certificate Revocation List*) és una llista dels certificats que es troben revocats (però que no estan caducats) en un moment concret. Aquesta llista és signada per la CA (o bé per l'emissor de la CRL) i conté una marca de temps.

Així doncs, la publicació periòdica de llistes de certificats revocats és una manera de donar a conèixer els certificats que es troben revocats. A la Secció 7.3.2 es descriu amb detall les llistes de revocació de certificats definides per l'estàndard X.509 així com les diferents alternatives de publicació d'aquestes llistes.

Una alternativa (o un complement) a l'ús de CRLs és l'ús del protocol OSCP:

> El protocol **OCSP** (de l'anglès, *Online Certificate Status Protocol*) permet obtenir l'estat d'un certificat digital identificat de manera interactiva.

En concret, el protocol OCSP permet a un client emetre una petició sobre l'estat d'un certificat digital a un servidor OSCP. El servidor OCSP respondrà amb una resposta signada indicant l'identificador del certificat, el seu estat de revocació, l'interval de temps en el qual es considera vàlida la resposta i informació addicional. L'estat del certificat pot ser `bo`, `revocat` (ja sigui permanentment o en suspensió) o `desconegut`.

> Una **autoritat de validació** (VA per les seves sigles en anglès, *Validation Authority*) és una autoritat que ofereix un servei que permet verificar la validesa d'un certificat digital.

7.1.4 Autoritat de segellat de temps

Certes aplicacions requereixen de segells de temps segurs per operar. Dins d'una PKI, l'autoritat de segellat de temps és l'entitat encarregada de proporcionar aquests segells.

> Una **autoritat de segellat de temps** o TSA (per les seves sigles en anglès, *Time Stamping Authority*) és una entitat que crea segells de temps que permeten demostrar que una dada o document existia en un instant particular en el temps.

Un segell de temps és doncs una signatura digital realitzada per una TSA sobre una estructura de dades que conté, d'una banda, el hash d'un document i, d'altra banda, una representació d'un instant de temps (Figura 7.2).

És important notar que un segell de temps ens assegura que el document existia en la data esmentada en el segell, però no ens dóna informació sobre en quin moment va ser creat. Un ús habitual dels segells de temps es troba en la verificació que una signatura digital d'un missatge va ser creada abans que el corresponent certificat digital fos revocat. D'aquesta manera es permet que un certificat digital revocat pugui ser utilitzat per validar signatures digitals creades amb anterioritat a la revocació. Un altre dels usos habituals és en el lliurament de documents o sol·licituds que tenen una data límit: l'ús del segell de temps permet demostrar que el document o la sol·licitud existia en un instant de temps concret, anterior a la data límit.

Figura 7.2: Abstracció del contingut d'un segell de temps.

A la Secció 7.3.4 es descriu el protocol de timestamp definit per l'estàndard X.509.

7.1.5 Entitat final

> Una **entitat final** és una entitat que disposa d'un certificat en una PKI i que no és una autoritat de certificació. Les entitats finals poden ser individus, però també organitzacions, aplicacions o fins i tot dispositius.

Les entitats finals són titulars d'un certificat digital. S'anomenen entitats finals ja que apareixen al final de la cadena de certificació.

7.1.6 Repositori de certificats

L'existència d'un certificat digital emès per una entitat de confiança que vinculi una identitat amb un parell de claus és de poca utilitat si hom no es troba en disposició del certificat digital en qüestió.

La manera més senzilla de solucionar aquest problema consisteix a deixar que siguin els propis usuaris els que es facin arribar els certificats els uns als altres, per exemple, enviant-los per correu electrònic o trobant-se en persona i fent l'intercanvi en un suport físic. Aquest mètode es coneix com a **disseminació privada** i és viable quan el nombre d'usuaris és reduït i la majoria d'usuaris es coneixen, però no escala bé quan el nombre d'usuaris creix. Una de les alternatives consisteix a publicar els certificats digitals, de manera que els usuaris en tinguin accés.

> Els **repositoris de certificats** emmagatzemen i proporcionen accés als certificats digitals.

El terme repositori és un terme genèric que es fa servir per referir-se a qualsevol base de dades centralitzada (almenys a nivell lògic) capaç d'emmagatzemar informació i servir-la quan aquesta és sol·licitada.

Pel que fa a la seguretat d'aquests repositoris, la integritat dels certificats queda garantida per la pròpia signatura que inclouen, de manera que un atacant no pot modificar amb èxit els certificats digitals del repositori sense ser detectat. Tot i això, els repositoris poden ser susceptibles a altres atacs, per exemple, atacs de denegació de servei que inhabilitin l'accés al repositori als usuaris legítims.

7.1.7 Repositori de llistes de revocació de certificats

A part dels repositoris de certificats, les PKI també poden disposar de repositoris de llistes de revocació de certificats, on les aplicacions que necessitin validar un certificat digital puguin descarregar-se la CRL corresponent. De la mateixa manera que amb els repositoris de certificats digitals, les CRLs garanteixen la integritat del seu contingut a través de signatures digitals, però els repositoris poden ser susceptibles a altres atacs.

> Els **repositoris de llistes de revocació de certificats** emmagatzemen i proporcionen accés a les CRLs.

A la Secció 7.3.2 es descriuen les característiques i format de les CRLs segons l'estàndard X.509.

7.2 Cicle de vida d'un certificat digital

En aquesta secció es descriu el cicle de vida d'un certificat digital, és a dir, les etapes o processos per els quals passa un certificat, des dels preparatius que cal fer per poder-lo crear fins al processos que es duen a terme una vegada el certificat ja no està en ús.

Alguns d'aquests processos són intrínsecs a la vida d'un certificat digital i, per tant, sempre es duran a terme. Per exemple, sempre caldrà generar les claus que quedaran vinculades a un certificat generat, ja que les claus són essencials en la utilitat del certificat. En canvi, altres processos són opcionals, i només es duran a terme en circumstàncies concretes. N'és el cas, per exemple, de la revocació d'un certificat.

7.2.1 Generació del parell de claus

Existeixen, principalment, tres maneres de crear un parell de claus per a un usuari, totes elles amb els seus inconvenients i els seus avantatges. Tot i que alguns mètodes són preferits a altres, no hi ha una opinió

unànime sobre quin és el millor mètode a utilitzar de manera universal:

- L'usuari crea el seu propi parell de claus. Aquest mètode té com a principal avantatge que l'usuari és l'únic que coneix la seva clau privada, ja que aquesta ha estat generada pels seus propis mitjans en el seu propi equip. Aquest mètode és especialment indicat quan les claus generades es volen utilitzar per realitzar signatures digitals amb no-repudi, ja que en aquest cas és convenient que cap altra entitat conegui la clau privada. Com a inconvenient, però, l'usuari ha de ser prou competent a nivell tècnic per poder generar el parell de claus de manera segura i per gestionar-ne les còpies de seguretat de manera adequada. Així, per exemple, un dels problemes que es poden presentar en aquest cas és la generació de claus en sistemes informàtics domèstics infectats de malware.
- El parell de claus és generat per la CA (o l'autoritat de registre). Tenint en compte que la CA ja és un element de confiança dins la PKI, hi ha qui opina que s'hi pot confiar també fins al punt de deixar que sigui aquesta qui generi les claus. El principal avantatge d'aquest mètode és que les CAs acostumen a tenir personal especialitzat, que és capaç de garantir la seguretat dels equips que generen les claus. Com a inconvenients, a més del fet de tenir una segona entitat que coneix la clau privada de l'usuari, apareix la centralització de la generació de claus.
- El parell de claus és generat per una tercera part. La tercera part genera aleshores les claus i comunica de manera físicament segura la clau privada a l'usuari. Després, la tercera part destrueix tota la informació relativa a la creació de les claus.
- El parell de claus és generat en un dispositiu hardware especialitzat. En aquest cas, sovint les claus privades queden emmagatzemades en una zona de seguretat del propi hardware, de manera que no poden ser extretes.

Altres consideracions que poden influir sobre la decisió de qui ha de generar el parell de claus són les implicacions legals que comporten o la capacitat de còmput necessària per a realitzar el procés. Aquest últim punt no és gaire problemàtic avui en dia, ja que els dispositius informàtics actuals tenen recursos suficients per generar claus de les mides que s'utilitzen en aquests moments.

Estàndards de CSR	Una de les sintaxis més utilitzades per a peticions de certificat és la que es descriu al PKCS#10.

Quan el parell de claus no és generat per la CA, serà necessari fer-li arribar la clau pública per tal que pugui ser inclosa en el certificat digital. La Petició de Signatura de Certificat (o CSR, de l'anglès, Certificate Signing Request) és un missatge amb una estructura de dades coneguda que envia el sol·licitant a la CA per tal d'informar-la de la clau pública que demana certificar. A més de la clau pública, la CSR inclou informació addicional, com ara informació d'identificació del sol·licitant.

Exemple 7.1 La generació de claus per a l'idCAT

La política de l'idCAT detalla que el ciutadà ha de generar les seves pròpies claus per a la identificació i la signatura electrònica en el seu ordinador personal. Així, el ciutadà serà l'encarregat de generar les claus i enviarà a la CA la clau pública a incloure en el certificat digital.

Per tal de simplificar aquest procés i fer-lo accessible a usuaris sense coneixements tècnics, el procés de generació de claus es pot realitzar a través del navegador, de manera gairebé transparent per a l'usuari.

Exemple 7.2 La generació de claus del DNI electrònic

El DNIe 3.0 conté un xip amb informació sobre el ciutadà que n'és propietari. Entre la informació que incorpora aquest xip, s'hi troben dos parells de claus RSA, un parell d'autenticació i un parell de signatura. Aquestes claus són generades dins del propi xip, amb una llibreria criptogràfica que porta incorporada.

Exercici 7.1 Quin procediment s'ha de seguir alhora de generar les claus vinculades a un certificat digital per tal de poder garantir que les signatures realitzades amb la clau privada corresponent tenen la propietat de no-repudi?

1. El subscriptor ha de crear el seu propi parell de claus
2. El parell de claus ha de ser generat per l'autoritat de certificació
3. El parell de claus ha de ser generat per una tercera part
4. El parell de claus és generat en un dispositiu hardware especialitzat

7.2.2 Registre

El **registre** és el procés per el qual una entitat final, ja sigui un individu o una organització, és verificada.

El nivell de verificació necessària dependrà de la Política de Certificació o de la Declaració De Pràctiques de Certificació. Així, per exemple, per a certificats amb polítiques de verificació laxes, l'usuari pot simplement omplir un formulari per tal de sol·licitar el registre. En canvi, per a polítiques de certificat més estrictes, serà necessari que l'usuari es personi físicament davant de l'autoritat de registre amb algun document d'identitat reconegut que incorpori una fotografia.

Exemple 7.3 El registre per a l'idCAT

L'Agència Catalana de Certificació ofereix els certificats digitals idCAT a la ciutadania. Per tal de realitzar el procés de registre, el ciutadà ha de personar-se a qualsevol de les Entitats de Registre idCAT (per exemple, als ajuntaments) i mostrar un document identificatiu (DNI, NIE o passaport). Prèviament, el ciutadà pot omplir un formulari amb les seves dades personals, de manera que el tramit s'agilitza.

Exemple 7.4 El registre per als certificats de persona física de la FNMT

La Fábrica Nacional de la Moneda y Timbre ofereix diferents tipus de certificats digitals, entre els quals hi ha els certificats de persona física. Per realitzar el registre d'un certificat de persona física de la FNMT, cal omplir un formulari per Internet i personar-se en alguna de les oficines de registre amb el codi de sol·licitud que s'obté al complimentar el formulari i un document d'identitat (DNI, passaport, carnet de conduir o NIE). Entre les oficines de registre disponibles s'hi troben les oficines de la Seguretat Social i les delegacions de l'Agència Tributària.

7.2.3 Creació del certificat

Una vegada s'ha generat el parell de claus i s'ha verificat la identitat de l'entitat, es crea el certificat digital. El certificat digital contindrà, principalment, la clau pública del titular (que haurà de ser enviada a la CA si el parell de claus no ha estat generat directament per la CA), la identitat verificada durant el registre i la signatura digital de la CA.

Exemple 7.5 La creació del certificat idCAT

Després de comprovar la identitat i les dades incloses en el certificat, i una vegada rebuda la clau pública,

l'Agència Catalana de Certificació pot generar el certificat digital del sol·licitant. El titular del certificat el podrà obtenir mitjançant el procediment telemàtic d'obtenció del certificat, que també es realitza a través del navegador.

7.2.4 Disseminació i recuperació del certificat

El procés de disseminació del certificat dependrà del cas d'ús concret i de la Política de Certificació. En alguns casos, el certificat es lliura directament al seu titular, que serà l'encarregat de distribuir-lo a terceres parts sota la seva discreció. En altres casos, el certificat es publica en algun repositori públic, de manera que tothom en té accés lliure.

Addicionalment, si la generació de claus no ha estat realitzada per l'entitat final del certificat, caldrà que la clau privada sigui enviada també a l'usuari.

D'altra banda, quan parlem de recuperació del certificat ens referim a l'habilitat d'obtenir un certificat d'entitat final quan aquest és necessari. Els casos d'ús més habituals són quan es necessita enviar informació xifrada a un destinatari o bé quan és necessari verificar una signatura digital rebuda d'una altra entitat.

Exemple 7.6 El repositori de claus públiques del MIT

El MIT PGP Public Key Server (`http://pgp.mit.edu/`) és un dels repositoris de claus públiques més popular. Actualment, el repositori forma part de la xarxa SKS de servidors de claus públiques, de manera que totes les claus que s'hi publiquen es disseminen cap al centenar de servidors que formen part de la xarxa.

7.2.5 Validació del certificat

La validació d'un certificat digital és, en realitat, un procés constituït per un conjunt de validacions, que caldrà realitzar abans de permetre fer cap operació criptogràfica amb la clau que aquest certifica. Aquestes validacions passen per comprovar que:

- La **signatura digital** del certificat és vàlida, és a dir, la signatura ha estat realitzada amb la clau de l'emissor del certificat i és correcta per al contingut del certificat. Noteu que aquesta validació ens garanteix la integritat de les dades del certificat.
- La data actual es troba dins del **període de validesa** del certificat digital, és a dir, el certificat no ha expirat.
- El certificat no ha estat **revocat**.
- L'**ús** que s'està donant al certificat digital és correcte (tenint en compte les restriccions d'utilització, de nom, de política, les extensions d'utilització de la clau, etc.).
- El certificat ha estat emès per una **entitat de confiança**.

Per tal de validar un certificat digital, caldrà construir la cadena de certificats entre el certificat que es vol validar i l'entitat de confiança.

Una **cadena de certificats** és una llista ordenada de certificats de clau pública començant per un certificat signat per una entitat de confiança i acabant amb el certificat que es vol validar. Tots els certificats intermedis són certificats de CA en els quals el titular d'un certificat correspon amb l'emissor del següent.

Per tal de validar doncs un certificat, caldrà comprovar tots els certificats de la cadena.

7.2.6 Expiració del certificat

Els certificats digitals tenen un període de validesa en el qual es consideren vàlids per a la seva funció. Quan aquest període de validesa acaba, diem que el certificat digital ha **expirat**.

Abans que finalitzi el període de validesa d'un certificat digital, es pot **actualitzar** el certificat digital, de manera que l'entitat final certificada pugui seguir gaudint dels serveis de la PKI ininterrompudament. El procés d'actualització passa per la creació d'un nou parell de claus i d'un nou certificat associat a les noves claus, amb un nou període de validesa. Aquest procés s'acostuma a dur a terme quan s'aproxima la data d'expiració del certificat original. Com que el titular del certificat ja ha passat pel procés de registre i disposa d'un certificat vàlid en el moment d'actualitzar-lo, el procés d'actualització no requereix que el titular del certificat torni a passar pel procés de registre.

Una alternativa és la **renovació** del certificat digital. En aquest cas, la mateixa clau pública que hi ha al certificat que està a punt d'expirar s'inclou en un nou certificat, amb un nou període de validesa. Cal anar en compte, però, a l'hora de renovar certificats digitals, ja que pot suposar problemes de seguretat en certes circumstàncies.

Exemple 7.7 L'expiració dels certificats del DNI electrònic

Els certificats del DNIe 3.0 tenen una validesa de 60 mesos des de la data d'emissió (o inferior si la data de caducitat del dni és anterior a aquests 60 mesos). Els certificats del DNIe es poden actualitzar personant-se en un *Punt d'Actualització del DNIe* dins d'una oficina d'expedició. El procés d'actualització és un procés automatitzat on el ciutadà s'autentica amb dades biomètriques i introdueix el seu pin, i noves claus i certificats són creats.

7.2.7 Revocació del certificat

Els certificats digitals tenen un període de validesa indicat en el propi certificat. A vegades però, és necessari poder invalidar un certificat abans que finalitzi aquest període. N'és el cas, per exemple, quan la clau privada corresponent és compromesa, quan es produeix un canvi de nom o quan canvia l'associació entre un titular i la CA (en particular, quan un treballador es desvincula d'una empresa). En aquest casos, serà necessari revocar el certificat digital.

Un certificat digital **revocat** és aquell que ha estat cancel·lat abans de la seva data d'expiració.

Exemple 7.8 La revocació certificats del DNI electrònic

El certificat de signatura digital del DNIe pot ser revocat personificant-se físicament en qualsevol de les oficines d'expedició del DNIe.

7.2.8 Història i arxivament de claus

Tot i que els certificats digitals tenen una data d'expiració, això no implica que totes les dades xifrades amb les claus d'aquests certificats hagin de deixar de ser accessibles quan el certificat caduca. Per tant, és necessari que les claus siguin emmagatzemades, encara que el corresponent certificat digital hagi caducat. Aquest procés es coneix amb el nom d'**història de claus** i és dut a terme, principalment, per a emmagatzemar claus privades que permetin desxifrar contingut que va ser xifrat en el passat. Normalment, la pròpia entitat final realitza aquest procés de manera local.

En canvi, quan es parla d'**arxivament de claus**, normalment es parla d'un servei ofert per una tercera part, que emmagatzema material de claus de vàries entitats finals. L'arxivament de claus consisteix en l'emmagatzemament a llarg plaç de claus (ja siguin de xifratge o de verificació de certificats). L'arxivament de claus és útil, per exemple, quan s'intenta validar una signatura digital creada amb una clau associada a un certificat que ja ha expirat.

7.3 Els estàndards X.509

L'estàndard X.509 de l'ITU-T defineix un framework per als certificats de clau pública, incloent l'especificació de les dades utilitzades per representar els certificats en sí, així com la informació sobre revocacions de certificats. Addicionalment, l'estàndard defineix també frameworks per a certificats d'atributs i serveis d'autenticació. Així, l'estàndard no ofereix la descripció de tots els components d'una PKI, sinó només d'una part, amb la intenció de servir com a base per a l'especificació i construcció de PKIs completes.

Certificats d'atributs

Un **certificat d'atributs** o AC (de l'anglès, *attribute certificate* és una estructura de dades signada digitalment que vincula uns valors d'uns atributs amb informació d'identificació del seu propietari.

D'altra banda, l'IETF dedica esforços a l'estandardització de les infraestructures de clau pública basades en X.509 a través del grup de treball PKIX. Inicialment, la feina del grup se centrava en perfilar les normes X.509 que produïa la ITU-T, però posteriorment el grup també va començar a desenvolupar iniciatives independents adreçades cobrir les necessitats de la PKI a Internet. La IETF publica els seus documents tècnics en les anomenades RFC (de l'anglès, *Request For Comments*), algunes de les quals tenen caràcter estandarditzador.

ITU

La ITU (de l'anglès, *International Telecommunication Union* és l'agència de les Nacions Unides especialitzada en l'àmbit de les telecomunicacions, la informació i les tecnologies de la comunicació.

IETF

La IETF (de l'anglès, *International Engineering Task Force* és una comunitat internacional que té com a objectiu millorar i evolucionar Internet, a través de la producció de documents tècnics que guiïn aquesta evolució.

La primera versió de l'estàndard internacional ITU-T X.509 va ser publicada al 1988 com a part de les recomanacions per directori X.500 i defineix un format estàndard per a certificats digitals. La versió descrita en aquest estàndard es coneix com a versió 1. Al 1993, l'estàndard va ser revisat i es van afegir dos camps més als certificats, amb el que es coneix com la versió 2 del format. També al 1993 es van publicar les RFCs relacionades de Internet Privacy Enhanced Mail (PEM), que inclouen especificacions per a una PKI basada en els certificats x.509 v1 (RFC1422). L'experiència obtinguda a l'intentar fer desplegaments d'aquesta RFC va servir per mostrar les deficiències del format v1 i v2 dels certificats X.509. En resposta a les deficiències detectades, es va crear la versió 3 del format del certificats, que extèn la versió 2 afegint la possibilitat de crear camps d'extensions addicionals. L'estandardització del format v3 va ser completada al juny de 1996 i es considera vigent en l'actualitat.

7.3.1 Certificats de clau pública

Com hem vist, un certificat digital de clau pública és una estructura de dades que vincula una clau pública amb una identitat a través d'una signatura d'aquest vincle feta per una autoritat. En aquesta secció, descriurem l'estructura d'un certificat X.509.

Gestió de certificats X.509

L'Openssl inclou l'eina `x509` que incorpora tot de funcionalitats relatives als certificats X.509. Així, per exemple, l'eina permet visualitzar el contingut dels certificats, convertir certificats a diferents formats o signar peticions de certificat.

La Taula 7.1 mostra els camps d'un certificat X.509 versió 3. L'estructura de dades del certificat és similar a moltes de les estructures de dades on s'hi emmagatzema contingut signat: s'inclou un camp amb el contingut a signar, un camp amb l'identificador de l'algorisme que s'ha fet servir per resumir i signar el contingut i, finalment, el valor de la signatura.

Camp	Descripció
`TBSCertificate`	El certificat a signar (en anglès, *to-be-signed certificate*).
`signatureAlgorithm`	Identificador de l'algorisme de signatura utilitzat per l'emissor del certificat per signar-lo.
`signatureValue`	Cadena de bits amb el valor de la signatura.

Taula 7.1: Camps d'un certificat X.509.

L'identificador de l'algorisme de signatura

L'identificador de l'algorisme de signatura acostuma a especificar una funció hash amb la qual es fa un resum del contingut a signar i un algorisme de signatura.

La Taula 7.3.1 mostra els camps del certificat a signar d'un certificat X.509 versió 3.

Els nom distingits (o DN, de l'anglès, *Distinguished Name*) acostumen a ser representats com una cadena de caràcters, on diferents atributs i el seu valor són llistats separats per comes. Les claus reconegudes són el nom comú (CN, de l'anglès, *CommonName*); el nom de la localitat (L, de l'anglès, *LocalityName*); el nom de l'estat o província (ST, de l'anglès *StateOrProvinceName*); el nom de l'organització (O, de l'anglès, *OrganizationName*; el nom de la unitat organitzativa (OU, de l'anglès, *OrganizationalUnitName*); el nom del país (C, de l'anglès, *CountryName*); i el carrer (STREET, de l'anglès, *StreetAddres*).

Exemple 7.9 Exemple de certificat x.509

A continuació s'inclou un exemple d'un certificat x.509 emès per a un estudiant de l'assignatura de criptografia de la UOC.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 8793 (0x2259)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=CAT, ST=Barcelona, L=Barcelona, O=UOC, OU=EIMT,
            CN=Consultor Criptografia
        Validity
            Not Before: May 23 13:27:19 2016 GMT
            Not After : May 23 13:27:19 2018 GMT
        Subject: C=CAT, ST=Barcelona, O=UOC,
            OU=EstudiantsCriptografia,
            CN=estudiant/emailAddress=estudiant@uoc.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (361 bit)
                Modulus:
                    01:b4:50:f5:bc:50:66:5e:80:0f:a3:85:07:de:c5:
                    d0:d4:36:c6:54:b1:66:db:46:49:06:37:4d:85:e2:
                    e7:b3:e8:b4:39:d7:05:77:20:67:8c:68:be:f9:37:
                    9d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
```

```
            Netscape Cert Type:
                SSL Client, S/MIME
            X509v3 Key Usage:
                Digital Signature, Non Repudiation
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                32:6C:46:E0:A5:7A:97:E3:EC:E6:0F:3D:23:14:13:7B:
                    5B:E0:97:F3
            X509v3 Authority Key Identifier:
                keyid:D2:D1:3D:A7:69:53:C6:B3:8A:10:D6:3A:51:87:
                    EB:56:4C:7C:99:7A
                DirName:/C=CAT/ST=Barcelona/L=Barcelona/
                    O=UOC/OU=EIMT/CN=Consultor Criptografia
                serial:D5:16:AD:04:20:AA:8C:26

            Netscape CA Revocation Url:
                http://www.uoc.edu/criptografia/ca-crl.pem
    Signature Algorithm: sha1WithRSAEncryption
         a4:6f:89:4e:2c:fe:85:0b:a2:7e:02:e6:45:3f:81:79:22:fa:
         2f:a1:d8:bf:43:f8:42:b9:b1:6f:6c:66:93:96:a6:2e:af:cc:
         c0:40:5f:21:69:60:77:0b:4f:00:06:40:61:f7:ad:09:1a:f2:
         1d:55:3c:a6:f5:dc:c2:f6:39:81:57:59:d6:cc:c6:b5:ad:00:
         78:be:2f:ae:d4:b6:e6:71:ab:5a:03:76:3d:0c:55:3d:87:b7:
         ab:a8:8c:2a:ef:87:09:3e:f8:50:71:b4:67:5b:a2:72:8e:a2:
         3d:3c:06:d4:09:93:c6:d7:df:4c:b3:a9:6f:ba:b2:f9:3b:95:
         44:e3:15:3c:15:ce:24:1f:23:16:c9:07:72:91:90:ff:8d:e2:
         c6:1c:95:22:18:b1:d9:39:a1:31:97:4f:cb:cc:71:23:94:4d:
         ef:0b:f0:64:3d:f7:a0:70:4c:2e:0f:6c:54:f1:95:52:00:85:
         62:9c:a3:b2:28:ea:f0:21:58:ba:4c:24:38:d7:9b:9c:78:6a:
         a6:fc:cc:11:62:11:9b:55:59:66:08:9d:98:11:3b:4c:20:e0:
         31:81:ef:1b:6d:3b:97:75:de:1f:75:6c:e5:6a:95:96:a5:9b:
         2d:f9:78:f2:31:88:f3:36:b4:21:cd:20:d4:91:e2:b0:0b:48:
         ab:fc:64:57
```

Extensions dels certificats de clau pública

El camp d'extensions permet afegir nous camps a l'estructura de dades d'un certificat sense haver de modificar-ne la seva definició en ASN.1.

Un camp d'extensió consisteix en:

- Un identificador d'extensió.
- Un flag de criticitat.
- Un valor codificat.

Les extensions, al ser camps addicionals, poden no ser reconegudes per totes les entitats que poden processar el certificat digital. El valor booleà del flag de criticitat condiciona com afectarà el reconeixement de l'extensió a la validació del certificat. Si el flag de criticitat conté el valor `FALSE` i l'entitat que ha de processar el certificat no reconeix l'extensió, aleshores pot ignorar-la a l'hora de realitzar la validació. En canvi, si el flag té el valor `TRUE`, una extensió no reconeguda causa que el certificat es consideri invàlid. Si l'entitat que ha de processar el certificat reconeix l'extensió, aleshores l'estàndard especifica que aquesta hauria de processar l'extensió, independentment del valor de criticitat que tingui.

Taula 7.2: Camps del certificat a signar.

Codi	Descripció
`version`	Indica la versió del certificat. Pot contenir els valors 0 (v1), 1 (v2) o 2 (v3).
`serialNumber`	És un valor enter designat per la CA que és únic per cada certificat emès per aquella CA en concret, és a dir, el parell de valors emissor i número de serie identifica de manera única un certificat digital de clau pública.
`signature`	Conté l'identificador de l'algorisme i la funció de hash fets servir per la CA per signar el certificat, per exemple, `sha-1WithRSAEncryption`. Aquest valor ha de ser el mateix que el del camp signatureAlgorithm.
`issuer`	Conté el nom distingit de la CA que emet el certificat, que no pot ser una cadena buida.
`validity`	Indica el període de validesa del certificat digital. Durant aquest interval, la CA garanteix que mantindrà informació sobre l'estat del certificat. El període de validesa s'indica amb dos camps de temps: `notBefore` i `notAfter`.
`subject`	Nom distingit que identifica al titular de la clau pública que està sent certificada. El camp pot estar buit si es tracta d'un certificat d'entitat final amb l'extensió `subjectAltName` inclosa i marcada com a crítica. En cas contrari, el camp no pot contenir una cadena buida.
`subjectPublicKeyInfo`	Conté dos components: `algorithm` i `subjectPublicKey`. El camp `algorithm` ha de contenir l'algorisme al que pertany la clau pública. El camp `subjectPublicKey` ha de contenir la clau pública que està sent certificada.
`issuerUniqueIdentifier`	Camp opcional que es fa servir per identificar de forma única l'emissor en cas de reutilització de noms.
`subjectUniqueIdentifier`	Camp opcional que es fa servir per identificar de forma única el titular del certificat en cas de reutilització de noms.
`extensions`	Camp opcional que permet afegir nous camps a l'estructura de dades.

Per tant, totes les extensions que tenen el flag de criticitat a `FALSE` poden causar comportaments inconsistents entre les entitats que reconeixen l'extensió (i que, per tant, la processaran) i aquelles que no la reconeixen (i que, per tant, la poden ignorar).

L'estàndard X.509 defineix algunes extensions. En el següents paràgrafs, es descriuen algunes d'aquestes extensions.

L'extensió de ús de la clau (`KeyUsage`) permet descriure la intenció d'ús del certificat. La Taula 7.3 descriu els possibles usos reconeguts per l'extensió. Un mateix certificat pot indicar diversos usos, tot i que aquest comportament pot suposar riscos de seguretat.

Concordança de les extensions	Si el booleà de `cA` de l'extensió `BasicConstraints` s'indica com a `FALSE`, aleshores l'extensió `KeyUsage` no pot tenir l'ús de `keyCertSign` actiu.

L'extensió de restriccions bàsiques (`BasicConstraints`) permet identificar si el titular del certificat és una autoritat de certificació i la profunditat màxima de les cadenes de certificació que inclouen el certificat en qüestió. D'una banda, l'extensió permet incloure un valor booleà que indica si la clau pública certificada pot ser utilitzada per verificar signatures de certificats digitals (camp `cA`). D'altra banda, i si el valor booleà revela que és un certificat de CA, l'extensió inclou un enter que indica el número màxim de certificats intermedis (no autoemesos) que poden seguir a aquest certificat en una cadena de certificació vàlida (camp `pathLenConstraint`).

Taula 7.3: Usos de clau reconeguts.

Codi	Descripció
`digitalSignature`	Verificació de signatures digitals.
`contentCommitment`	Verificació de signatures digitals on el signatari es compromet amb el contingut signat.
`keyEncipherment`	Xifratge de claus o d'algun altre tipus d'informació de seguretat.
`dataEncipherment`	Xifratge de dades d'usuari.
`keyAgreement`	Ús com a clau pública en un protocol d'establiment de claus.
`keyCertSign`	Verificació de signatures de certificats realitzades per Autoritats de Certificació.
`cRLSign`	Verificació de signatures de CRLs realitzades per autoritats.
`encipherOnly`	Ús com a clau pública en un protocol d'establiment de claus per utilitzar únicament xifratge de dades (cal indicar també l'ús `keyAgreement`).
`decipherOnly`	Ús com a clau pública en un protocol d'establiment de claus per utilitzar únicament desxifratge de dades (cal indicar també l'ús `keyAgreement`).

L'extensió d'identificador de la clau de l'autoritat (`AuthorityKeyIdentifier`) permet identificar la clau privada utilitzada per signar un certificat digital. Això és particularment útil quan l'emissor del certificat disposa de diverses claus.

L'extensió d'identificador de la clau del titular (`SubjectKeyIdentifier`) permet identificar els certificats que tenen una clau pública donada. Així, quan una entitat final té diversos certificats (per exemple, de diferents emissors) amb la mateixa clau pública, el conjunt de certificats pot ser identificat fàcilment.

Exercici 7.2 Una aplicació ha de validar un certificat digital que conté una extensió marcada com a crítica. Indiqueu quin hauria de ser el resultat de la validació per les següents casuístiques (suposant que la resta de comprovacions que es realitzen per validar el certificat són correctes).

Nota: Un resultat `TRUE` indica una validació satisfactòria, mentre que un resultat `FALSE` indica que la validació no és correcta.

	Resultat de processar l'extensió	
	`TRUE`	`FALSE`
L'aplicació reconeix l'extensió		
L'aplicació no reconeix l'extensió		

Exercici 7.3 Una aplicació ha de validar un certificat digital que conté una extensió marcada com a **no** crítica. Indiqueu quin hauria de ser el resultat de la validació per les següents casuístiques (suposant que la resta de comprovacions que es realitzen per validar el certificat són correctes).

	Resultat de processar l'extensió	
	`TRUE`	`FALSE`
L'aplicació reconeix l'extensió		
L'aplicació no reconeix l'extensió		

Tipus de certificats de clau pública

Existeixen, principalment, dos tipus de certificats de clau pública:

- Certificats de clau pública d'**entitats finals**.
- Certificats de clau pública d'**autoritats de certificació** (CA).

Un certificat d'entitat final és un certificat emès per una autoritat de certificació a una entitat que no és emissora d'altres certificats. En canvi, un certificat de CA és un certificat emès per una CA a una entitat que és també una CA i, per tant, també és capaç d'emetre certificats. Un certificat de CA ha d'incloure l'extensió `basicConstraints` amb el component `cA` a `True`.

Els certificats de CA poden ser alhora classificats en tres tipus diferents:

- Un certificat **autoemès** és un certificat de CA on el titular i l'emissor són la mateixa CA. Aquest tipus de certificat es pot fer servir, per exemple, per a realitzar un canvi de claus, transferint la confiança de la clau antiga a la nova clau.
- Un certificat **autosignat** és un cas especial d'un certificat autoemès on la clau privada utilitzada per signar el certificat correspon a la clau pública que se certifica amb el certificat. Es poden fer servir certificats autosignats, per exemple, per donar a conèixer una clau pública o altra informació.
- Un certificat **creuat** és un certificat de CA on l'emissor i el titular són autoritats de certificació diferents. Un certificat creuat es pot fer servir, per exemple, per a reconèixer l'existència de la CA titular o bé per autoritzar-la.

7.3.2 Llistes de revocació de certificats

Les autoritats de certificació són responsables d'informar sobre l'estat de revocació dels certificats que emeten. Un dels mètodes per oferir aquesta informació de revocació és la publicació de llistes de revocació de certificats o CRLs (per les seves sigles en anglès, *Certificate Revocation List*). Normalment, la pròpia autoritat de certificació emet les CRLs, però també pot delegar aquesta responsabilitat a alguna altra entitat.

Una CRL és una llista dels números de sèrie dels certificats revocats, juntament amb la signatura de la CA (o l'emissor de la CRL) i una marca de temps. Normalment, les CRLs es publiquen periòdicament, per exemple, cada hora o un cop al dia. Quan un certificat és revocat, el seu número de sèrie s'afegeix a la CRL que s'emet després de la revocació. El número de sèrie no s'ha d'eliminar de la CRL fins que hagi aparegut en una CRL emesa amb posterioritat a la fi del període de validesa del certificat.

Com que les CRLs contenen una signatura de l'entitat que les emet, la integritat del seu contingut està garantida. D'aquesta manera, no és necessari confiar en què els servidors o els processos que distribueixen les CRLs no intentaran modificar-les.

Un dels inconvenients que presenta l'ús de CRLs és l'endarreriment que es crea a l'hora d'informar sobre la revocació d'un certificat. Entre la revocació d'un certificat i l'addició del seu número de sèrie a la propera CRL que es publica passa un interval de temps, que podrà ser menor o major en funció de la periodicitat de la publicació de la CRL. Durant aquest període, tot i que el certificat es troba revocat, la informació de revocació no estarà disponible. Aquest problema es pot minimitzar amb l'ús de protocols interactius que consulten l'estat d'un certificat digital en un moment concret.

Un altre dels inconvenients que presenta l'ús de CRLs és que són susceptibles a atacs de denegació de servei. Un atacant pot evitar que la informació de revocació d'un certificat arribi a les aplicacions blocant la distribució de la CRL. Així, mentre que un atacant no podrà modificar el contingut de la CRL (degut a la signatura sobre aquest), sí que pot atacar la disponibilitat del servei.

Generació de CRLs	L'Openssl inclou l'eina `ca`, que incorpora funcionalitats per gestionar una autoritat de certificació, entre les quals hi ha la creació de CRLs.

Així doncs, quan un sistema necessita validar un certificat digital, a més de comprovar-ne la seva signatura i el període de validesa, també serà necessari que es descarregui una CRL prou recent i comprovi que el

número de sèrie del certificat no hi figura. La definició del que es considera prou recent dependrà de la política de cada sistema.

Els certificats poden contenir informació sobre com obtenir informació de revocació a través de CRLs fent servir l'extensió `crlDistributionPoint`.

Tipus de CRLs

Cada llista de revocació de certificats té un **abast**, és a dir, un conjunt de certificats que poden aparèixer en aquella CRL. Per exemple, l'abast d'una CRL poden ser tots els certificats emesos per una determinada CA o bé tots els certificats emesos per una CA per un motiu concret.

Una llista CRL **completa** enumera tots els certificats no expirats dins del seu abast que han estat revocats per alguna de les raons cobertes per l'abast. D'altra banda, direm que una CRL és **plena i completa** quan conté tots els certificats no expirats emesos per la CA que s'han revocat per qualsevol raó.

Els termes plena i completa	En anglès, es fan servir els termes *complete* i *full and complete* per diferenciar entre CRLs que contenen únicament certificats revocats per un dels motius indicats a l'abast o bé independentment del motiu.

Una CRL **indirecta** és una CRL amb un abast que inclou almenys un certificat emès per una entitat de certificació diferent de l'emissor de la CRL. Una CRL indirecta pot incloure en el seu abast certificats emesos per diverses autoritats de certificació. A més, si l'emissor de la CRL és una CA, aleshores l'abast de la CRL pot incloure també els certificats emesos per aquesta CA.

Una **delta** CRL només enumera els certificats dins del seu abast que han canviat d'estat de revocació des de l'emissió d'una CRL completa referenciada. La CRL completa referenciada s'anomena CRL **base**. Es considera que l'estat d'un certificat ha canviat si aquest està revocat, si ha deixat d'estar suspès o si la raó per la qual el certificat ha estat revocat ha canviat. L'abast de la delta CRL ha de ser el mateix que la CRL base que referencia. A més, la clau privada utilitzada per signar la delta CRL també ha de ser la mateixa que la feta servir per signar qualsevol CRL completa que pugui actualitzar. Generalment, les delta CRLs són més petites que les CRLs que actualitzen, de manera que fer servir delta CRLs pot ajudar a reduir el consum d'ample de banda d'un sistema que faci servir CRLs.

Una aplicació que fa servir delta CRLs ha de ser capaç de construir una CRL completa combinant una CRL completa emesa amb anterioritat i la delta CRL més recent. Addicionalment, l'aplicació també pot construir una CRL completa a partir de la delta CRL més recent i d'una CRL construïda localment que és completa per aquest abast.

És considera que una delta CRL és **actual** si el temps actual es troba en el període comprés entre els camps `thisUpdate` i `nextUpdate`. Per tant, podria passar que l'emissor de CRLs emetés més d'una delta CRL abans del `nextUpdate`, existint aleshores més d'una delta CRL considerada actual. En aquests casos, l'estàndard recomana (però no exigeix) que es faci servir la CRL que té el `thisUpdate` més actual.

Contingut d'una CRL

Una CRL és una llista de certificats revocats signada per una autoritat. Així, els camps que defineixen una CRL segons l'estàndard X.509 es descriuen a la Taula 7.4.

La llista de certificats revocats és alhora una seqüència de diversos camps. La Taula 7.5 descriu els camps que la componen.

Extensions de les CRLs

El camp d'extensions és un camp opcional que pot aparèixer a partir de la versió 2 i que conté una seqüència d'una o més extensions, que permeten afegir atributs addicionals a les CRLs. De manera anàloga a les extensions dels certificats X.509, cada extensió d'una CRL pot ser marcada com a crítica o com a no crítica.

Taula 7.4: Camps d'una CRL.

Camp	Descripció
`tbsCertList`	La llista de certificats revocats (en anglès, *to-be-signed certificate list*).
`signatureAlgorithm`	Identificador de l'algorisme de signatura utilitzat per l'emissor de la CRL per signar-la.
`signatureValue`	Cadena de bits amb el valor de la signatura.

Taula 7.5: Camps de la llista a signar.

Camp	Descripció
`version`	Opcional, descriu la versió de la CRL codificada.
`signature`	Identificador de l'algorisme de signatura utilitzat per l'emissor de la CRL per signar-la. El valor d'aquest camp ha de coincidir amb el valor del camp `signatureAlgorithm`.
`issuer`	Nom de l'entitat que ha emès i signat la CRL.
`thisUpdate`	Data d'emissió d'aquesta CRL.
`nextUpdate`	Opcional, data d'emissió de la propera CRL. La propera CRL pot ser emesa abans de la data indicada pel camp `nextUpdate`, però ho hauria de ser emesa més tard d'aquesta.
`revokedCertificates`	Opcional, la llista amb els certificats revocats. Si no hi ha certificats revocats, aleshores la llista no s'inclou. En cas contrari, els certificats revocats s'enumeren en base al seu número de sèrie, i s'especifica la data en la qual han estat revocats.
`crlExtensions`	Opcional, extensions que poden contenir atributs addicionals.

Les aplicacions que no siguin capaces de reconèixer o processar una extensió marcada com a crítica en una CRL no han de fer ús d'aquella CRL. En canvi, si l'extensió està marcada com a no crítica, les aplicacions poden ignorar-la. En els següents paràgrafs, descriurem algunes de les extensions més populars que es fan servir en CRLs a Internet.

De la mateixa manera que als certificats de clau pública, l'extensió d'identificador de la clau de l'autoritat (`AuthorityKeyIdentifier`) és una extensió que permet afegir informació sobre la clau pública corresponent a la clau privada utilitzada per signar la CRL. Aquesta extensió és especialment útil quan un mateix emissor té vàries claus de signatura.

El número de CRL (`CRLNumber`) és una extensió que permet incloure un número de seqüència a la CRL. Donats un emissor de CRL i un abast concret, els números de seqüència són valors estrictament creixents.

L'indicador de delta CRL permet indicar que una CRL és una delta CRL. Per fer-ho, l'extensió inclou el número de CRL de la CRL base (`BaseCRLNumber`) que va ser utilitzada per crear aquesta delta CRL.

El codi de motiu (`CRLReason`) és una extensió que permet identificar el motiu per el qual s'ha revocat el certificat. La Taula 7.6 mostra els codis reconeguts. Exceptuant el codi `removeFromCRL`, la resta de codis indiquen que el certificat ha estat revocat.

Altres models d'emissió de delta CRLs

A l'exemple es mostra el model d'emissió de delta CRLs *tradicional*. Existeixen altres models, com ara el model de finestres lliscants (en anglès, *sliding windows*) que permeten estalviar més ample de banda en alguns escenaris.

Taula 7.6: Codis per especificar el motiu de revocació de certificats digitals.

Codi	Descripció
`unspecified`	Es pot fer servir per revocar certificats per algun motiu diferent als que tenen un codi específic.
`keyCompromise`	Es fa servir per revocar un certificat d'una entitat final i indica que la clau privada del titular del certificat ha estat compromesa.
`cACompromise`	Es fa servir per revocar certificats d'autoritats de certificació i indica que la clau privada de la CA ha estat compromesa.
`affiliationChanged`	Indica que el nom del titular o bé alguna altra informació del certificat ha estat modificada (però que no hi ha motiu per sospitar que la clau privada ha estat compromesa).
`superseded`	Indica que el certificat ha estat substituït (però que no hi ha motiu per sospitar que la clau privada ha estat compromesa).
`cessationOfOperation`	Indica que el certificat ja no és necessari per a l'objectiu pel qual va ser emès (però que no hi ha motiu per sospitar que la clau privada ha estat compromesa).
`certificateHold`	Indica que el certificat es troba suspès temporalment.
`removeFromCRL`	Aquest codi només pot aparèixer en delta CRLs i permet indicar que el certificat s'ha d'eliminar de la CRL, ja sigui perquè ha expirat o perquè ja no es troba suspès.
`privilegeWithdrawn`	Indica que el certificat (de clau pública o d'atributs) es troba revocat perquè un privilegi contingut al certificat ha estat retirat.
`aACompromise`	Es fa servir per revocar certificats d'autoritats d'atributs i indica que la clau privada de l'autoritat d'atributs ha estat compromesa.

Exemple 7.10 Exemple d'ús tradicional de CRLs

La següent taula es mostra un exemple de la manera tradicional d'emetre delta CRLs. En aquest exemple, s'emeten CRLs completes cada dues hores i delta CRLs cada 30 minuts. L'abast de la CRL comprèn tots els certificats emesos per l'autoritat de certificació C per qualsevol motiu excepte `superseded`.

La notació C_x indica el certificat emès per l'autoritat C amb número de sèrie x. Es fa servir l'hora amb precisió de minuts per indicar el temps, però en un cas real es faria servir la data i l'hora completa. A la llista de certificats revocats s'inclou el número de sèrie del certificat i el motiu de la revocació. S'omet la data en la qual ha estat revocat el certificat per simplificar-ne la visualització.

El certificat C_1 s'ha revocat per compromís de la clau privada en algun instant anterior a les `8:00`. Quan s'emet la CRL completa de les `8:00`, C_1 hi apareix amb motiu `keyCompromise`. Les delta CRL que tenen com a base la CRL 1 no inclouen aquest certificat, ja que ja apareix a la CRL base.

En algun moment entre les `8:30` i les `9:00`, C_{33} es revoca amb motiu `privilegeWithdrawn`, i apareix per tant a la primera delta CRL que es publica després de la revocació, la CRL 3. Entre les `9:00` i les `9:30` l'estat de revocació de C_{33} canvia a `affiliationChanged`, i aquest canvi es veu reflectit a la següent delta CRL que es publica, la CRL 4. En el mateix interval de temps el certificat C_{25} es revoca amb motiu `superseded`, però aquest certificat no apareix en cap de les CRLs ja que està fora de l'abast definit.

Entre les `9:30` i les `10:00` el certificat C_{22} es posa en espera. Aquest canvi es veu reflectit tant a la CRL completa com a la delta CRL que es publica a les `10:00`.

Entre les 10:00 i les 10:30 C_{55} és revocat per compromís de la clau privada, i apareix per tant a la delta CRL de les 10:30. Tot i que el certificat expira a les 10:45, C_{55} segueix apareixent a les delta CRL fins que apareix en una CRL completa (la CRL 9).

En l'interval de temps entre les 10:30 i les 11:00, s'aixeca la suspensió sobre el certificat C_{22}. Per tant, a partir de la CRL 7, les delta CRL indiquen que es pot eliminar el certificat de la CRL amb el codi `removeFromCRL`. A partir de la CRL 10 aquesta notificació ja no s'inclou en les delta CRLs, ja que ja s'ha publicat una CRL completa on no hi apareix C_{22} (la CRL 9). És important notar que la CRL no inclou el certificat C_{22} amb codi `removeFromCRL` sinó que simplement no s'inclou el certificat a la llista de certificats revocats. El codi `removeFromCRL` no apareix en CRLs completes.

Temps actual t	Estat dels certificats en l'instant t	Contingut de la CRL completa	Contingut de la Delta CRL
8:00	$\{C_1 : keyCompromise\}$	CRLNumber : 1 thisUpdate: 8:00 nextUpdate: 10:00 revokedCertificates: $\{C_1 : keyCompromise\}$	CRLNumber : 1 thisUpdate: 8:00 nextUpdate: 8:30 BaseCRLNumber: 1 revokedCertificates: {}
8:30	$\{C_1 : keyCompromise\}$		CRLNumber : 2 thisUpdate: 8:30 nextUpdate: 9:00 BaseCRLNumber: 1 revokedCertificates: {}
9:00	$\{C_1 : keyCompromise,$ $C_{33} : privilegeWithdrawn\}$		CRLNumber : 3 thisUpdate: 9:00 nextUpdate: 9:30 BaseCRLNumber: 1 revokedCertificates: $\{C_{33} : privilegeWithdrawn\}$
9:30	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded\}$		CRLNumber : 4 thisUpdate: 9:30 nextUpdate: 10:00 BaseCRLNumber: 1 revokedCertificates: $\{C_{33} : affiliationChanged\}$
10:00	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded,$ $C_{22} : certificateHold\}$	CRLNumber : 5 thisUpdate: 10:00 nextUpdate: 12:00 revokedCertificates: $\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{22} : certificateHold\}$	CRLNumber : 5 thisUpdate: 10:00 nextUpdate: 10:30 BaseCRLNumber: 1 revokedCertificates: $\{C_{33} : affiliationChanged,$ $C_{22} : certificateHold\}$
10:30	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded,$ $C_{22} : certificateHold,$ $C_{55} : keyCompromise\}$		CRLNumber : 6 thisUpdate: 10:30 nextUpdate: 11:00 BaseCRLNumber: 5 revokedCertificates: $\{C_{55} : keyCompromise\}$
11:00	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded\}$ C_{55} expired on $t =$ 10:45		CRLNumber : 7 thisUpdate: 11:00 nextUpdate: 11:30 BaseCRLNumber: 5 revokedCertificates: $\{C_{55} : keyCompromise,$ $C_{22} : removeFromCRL\}$
11:30	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded\}$		CRLNumber : 8 thisUpdate: 11:30 nextUpdate: 12:00 BaseCRLNumber: 5 revokedCertificates: $\{C_{55} : keyCompromise,$ $C_{22} : removeFromCRL\}$
12:00	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded\}$	CRLNumber : 9 thisUpdate: 12:00 nextUpdate: 14:00 revokedCertificates: $\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{55} : keyCompromise\}$	CRLNumber : 9 thisUpdate: 12:00 nextUpdate: 12:30 BaseCRLNumber: 5 revokedCertificates: $\{C_{55} : keyCompromise,$ $C_{22} : removeFromCRL\}$
12:30	$\{C_1 : keyCompromise,$ $C_{33} : affiliationChanged,$ $C_{25} : superseded\}$		CRLNumber : 10 thisUpdate: 12:30 nextUpdate: 13:00 BaseCRLNumber: 9 revokedCertificates: {}

7.3.3 Online Certificate Status Protocol

El protocol OCSP (de l'anglès, *Online Certificate Status Protocol*) permet determinar l'estat de revocació actual d'un certificat digital a través d'un protocol interactiu. L'OCSP es pot fer servir ja sigui com a substitut o com a complement de les CRLs.

En el protocol hi participen dues parts, el client OSCP (que està interessat en saber l'estat d'un certificat digital) i el servidor OCSP (que respondrà a les consultes del client).

Quan un client necessita validar l'estat d'un certificat digital, aleshores envia una petició al servidor OCSP i suspèn l'acceptació del certificat fins que arriba la resposta. Una petició OCSP conté els següents camps:

- Versió del protocol.
- Identificador del certificat o certificats per als quals es demana l'estat de revocació.
- Opcionalment, extensions.

Una resposta OCSP (de tipus bàsic) conté la següent informació:

- Versió de la sintaxi de la resposta.
- Instant de temps en què s'ha generat la resposta.
- Conjunt de respostes (per cada un dels certificats que s'han demanat a la petició).
- Identificador de l'algorisme de signatura.
- Valor de la signatura.

Cada una de les respostes conté alhora quatre camps: l'identificador del certificat a la qual fa referència, l'estat del certificat, l'interval de validesa de la resposta i, opcionalment, les extensions.

Implementació de l'OCSP	L'Openssl inclou l'eina `ocsp` que incorpora funcionalitats per interactuar amb un servidor OCSP i, fins i tot, per operar com un petit servidor OCSP.

L'especificació d'OCSP de la RFC6960 defineix tres alternatives per indicar l'estat del certificat: bo (`good`), revocat (`revoked`) o desconegut (`unknown`). Cal anar amb compte, però, a l'hora d'interpretar aquestes respostes, ja que els noms utilitzats per designar-les poden generar confusió sobre els detalls del seu significat.

Una resposta bona indica que no hi ha cap certificat amb el número de sèrie especificat a la petició que es trobi revocat (i que estigui dins del seu periode de validesa). Això no implica necessàriament que el certificat existeixi (podria ser que mai hagués estat emès) ni que el certificat sigui vàlid en aquell moment (podria ser que la resposta del servidor OCSP no estigui compresa en l'interval de validesa del certificat). Una resposta de certificat revocat indica que el certificat ha estat revocat, ja sigui temporalment (especificant el codi de motiu `certificateHold`) o bé permanentment. Tot i això, un servidor OCSP també pot retornar aquesta resposta si la CA corresponent no ha emès mai un certificat amb aquest número de sèrie. Una resposta d'estat desconegut indica que el servidor no coneix el certificat.

Exercici 7.4 Indiqueu quines de les següents afirmacions són certes:

1. Per tal de comprovar la data de caducitat d'un certificat digital, podem utilitzar el protocol OCSP.
2. Per tal de comprovar la data de caducitat d'un certificat digital, podem utilitzar CRLs.
3. Per tal de comprovar la data de caducitat d'un certificat digital, només ens cal el propi certificat.
4. Una resposta OCSP `good` sempre indica que el certificat no està revocat.
5. Una resposta OCSP `revoked` sempre indica que el certificat està revocat.

7.3.4 Time Stamp Protocol

Com hem vist, les autoritats de segellat de temps o TSA són les autoritats d'una PKI encarregades de crear segells de temps. Per fer-ho, la TSA signa els segells que emet amb una clau específicament reservada per a aquest propòsit. En concret, el certificat digital corresponent ha de tenir una única instància del camp

extended key usage amb keyPurposeId `id-kp-timeStamping` i l'extensió marcada com a crítica.

El TSP (Time-Stamp Protocol) és el protocol que es fa servir per interactuar amb una TSA. El procediment per aconseguir un segell de temps és el següent. En primer lloc, el sol·licitant envia una petició de segell de temps a la TSA. Després, la TSA respon a la petició amb un missatge de resposta. Per acabar, el sol·licitant hauria de comprovar l'estat d'error de la resposta. Si no hi ha errors, caldria validar el segell de temps retornat.

Nonce

Un nonce (de l'anglès, *number used once*) és un nombre arbitrari que es fa servir una única vegada en un protocol criptogràfic.

La petició de segell de temps que el sol·licitant envia a la TSA conté, entre d'altres:

- El hash de la dada que es vol segellar.
- L'identificador de l'algorisme de hash utilitzat.
- Opcionalment, un nonce. El nonce és un valor aleatori que té una probabilitat alta de ser generat una única vegada pel client. El nonce és opcional però, si s'inclou, la resposta de la TSA ha de contenir aquest mateix valor. El nonce permet detectar el reenviament d'un segell de temps, ja sigui realitzat de manera involuntària per errors en la transmissió o bé com a conseqüència d'un atac.
- Opcionalment, un booleà `certReq` que permet indicar que es desitja que la TSA inclogui el certificat corresponent a la signatura en la resposta. Si no s'inclou el camp o aquest és `False`, aleshores la resposta de la TSA no ha de contenir el certificat.

La resposta de la TSA conté l'estat de la petició i, en alguns casos, el segell de temps demanat.

L'estat està format per tres camps: un codi que l'identifica i que sempre es troba present i, opcionalment, un text i una explicació del motiu per el qual la petició ha fallat (si n'és el cas). Els codis d'estat reconeguts són:

- *concedit*: el segell de temps demanat s'adjunta a la resposta.
- *concedit amb modificacions*: s'adjunta un segell de temps, però aquest conté alguna modificació.
- *rebutjat*: es rebutja la petició de segell de temps.
- *en espera*: la creació del segell de temps es troba en espera.
- *en alerta per revocació*: el missatge conté un avís que la revocació és imminent.
- *notificació de revocació*: s'ha revocat el certificat.

La resposta ha de contenir el segell de temps només si l'estat és concedit o concedit amb modificacions. En cas contrari, caldrà indicar el motiu de la fallada. Els motius de fallada poden ser diversos, com ara per exemple, que l'algorisme de hash indicat no es reconegui, que el format de la petició sigui incorrecte, o que la font utilitzada per la TSA per aconseguir el temps no estigui disponible en aquell moment.

Sintaxi dels segells de temps

L'explicació sobre el contingut dels segells de temps abstrau els detalls reals de la sintaxis d'aquests. Els segells de temps són informació signada, que segueix la sintaxi CMS (Cryptographic Message Syntax).

El camp de temps

La codificació del temps en una resposta de la TSA sempre acaba en Z, el que indica que representa el temps Zulu. El temps Zulu és un sinònim del Temps Universal Coordinat que es fa servir en aviació civil.

Per la seva banda, el segell de temps estarà format per la signatura de la TSA (a la que s'adjuntarà l'identificador del certificat utilitzat per realitzar-la) i el contingut del segell de temps, que tindrà, entre d'altres, els següents camps:

- El hash de la dada que s'ha rebut.
- L'identificador de l'algorisme de hash utilitzat.
- Un número de sèrie, que serà assignat per la TSA a cada segell de temps i que serà únic entre els segells emesos per aquesta TSA.
- El temps, indicant el moment en què la TSA ha generat el timestamp expressat en UTC (Temps

Universal Coordinat).
- Opcionalment, la precisió, que ens permet determinar l'interval de temps exacte en el qual s'ha creat el segell de temps. Si el camp no s'inclou, la precisió es pot anunciar d'altres maneres, com ara a partir de la política de la TSA.
- Opcionalment, el nonce. Si la petició contenia nonce, aleshores la resposta cal que també la contingui i que el valor sigui igual que el de la petició.
- Opcionalment, el nom de la TSA, que permet ajudar en la identificació de la TSA.

Implementació del TSP	L'Openssl inclou l'eina `ts` que incorpora les funcions bàsiques de client i servidor de TSA.

Finalment, quan es rep la resposta de la TSA, caldria validar-la. Per fer-ho, en primer lloc es comprova l'estat d'error i, si no s'ha produït cap error, aleshores es valida el segell de temps: es comproven els camps, es valida la signatura digital, es comprova que el segell de temps correspongui al que es va demanar (revisant tant el valor del hash com l'identificador de l'algorisme fet servir), l'estat del certificat de la TSA i el temps. El temps pot ser validat a partir de la referència d'un servei de temps local de confiança, o bé comprovant que el nonce inclòs en la petició es troba també en la resposta.

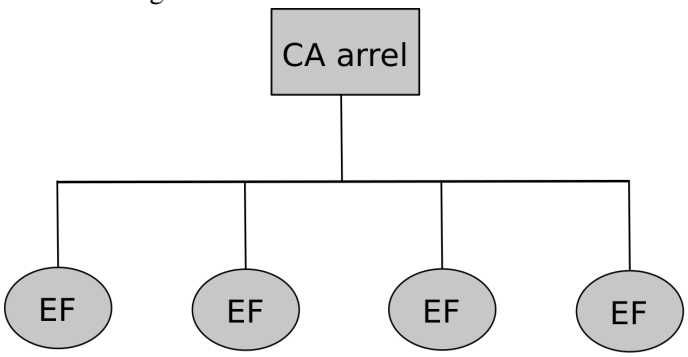
Exercici 7.5 Un ciutadà està recollint signatures digitals per a recolzar una proposta. Per tal que les signatures digitals siguin considerades vàlides, cal que estiguin realitzades en un interval de temps concret (t_{inici}, t_{fi}), és a dir, que s'hagin més tard de t_{inici} i abans de t_{fi}. Quin d'aquests segells de temps seria vàlid per demostrar que el conjunt de signatures s'han creat en el període establert?

1. Un segell de temps sobre totes les signatures realitzat en $t_s < t_{inici}$
2. Un segell de temps sobre totes les signatures realitzat en $t_s > t_{fi}$
3. Un segell de temps sobre totes les signatures realitzat en $t_{inici} < t_s < t_{fi}$

7.3.5 Estructures de PKI

Potser el model d'estructura de PKI més simple és el model de **CA única**. En aquest model, hi ha una única autoritat de certificació, que emet certificats per a totes les entitats finals que participen de la PKI. La Figura 7.3 mostra un exemple d'aquest model.

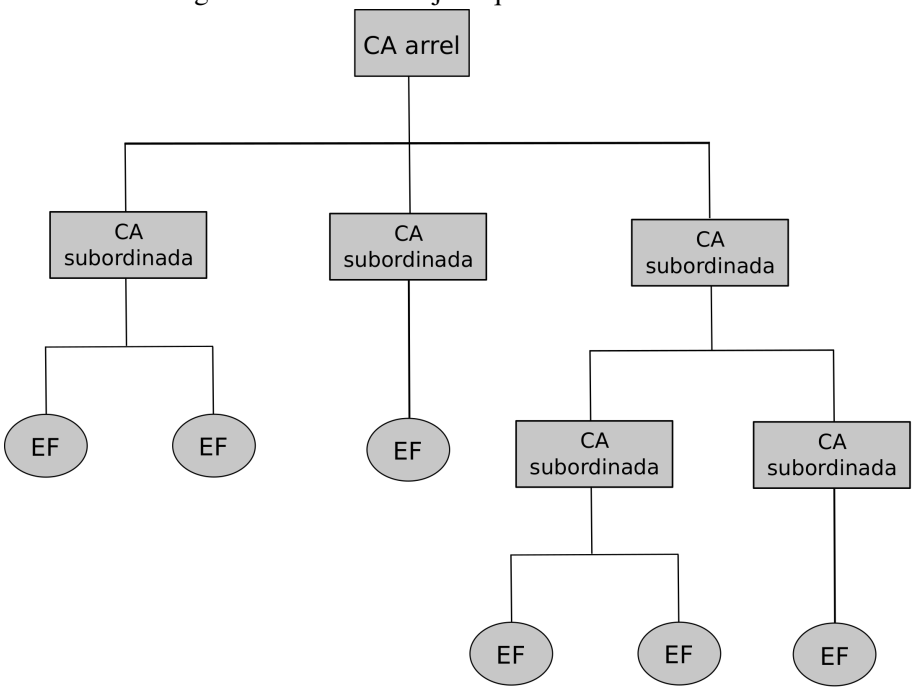
Figura 7.3: Estructura amb CA única.



Una PKI amb un model **jeràrquic amb arrel única** té una una única CA arrel, però pot tenir altres CAs. Les diferents CAs es troben estructurades seguint una jerarquia, on la CA arrel certifica a un conjunt d'autoritats de certificació, que alhora poden crear certificats per a altres CAs (o per a entitats finals), creant els diferents nivells de la jerarquia. Les CAs intermèdies també són conegudes amb el nom d'autoritats subordinades. El model de CA única pot ser vist com un cas específic del model jeràrquic, on la única CA existent és la CA arrel. La Figura 7.4 mostra un exemple d'aquest model.

Una variant d'aquesta estructura és el model **jeràrquic amb llista de confiança**, que disposa d'una llista

Figura 7.4: Estructura jeràrquica amb arrel única.



amb vàries CAs arrel. Cadascuna de les CAs arrel pot tenir autoritats subordinades seguint una jerarquia. L'ús més conegut d'aquest model és en els navegadors web, que contenen una llista amb uns centenars de CAs arrel. La Figura 7.5 mostra un exemple d'aquest model.

Existeixen altres estructures de PKI, com ara l'estructura de **malla**, on les autoritats de certificació emeten certificats creuats, o la interconnexió a través de **bridge** CAs, on una autoritat fa de nexe entre les diferents PKIs.

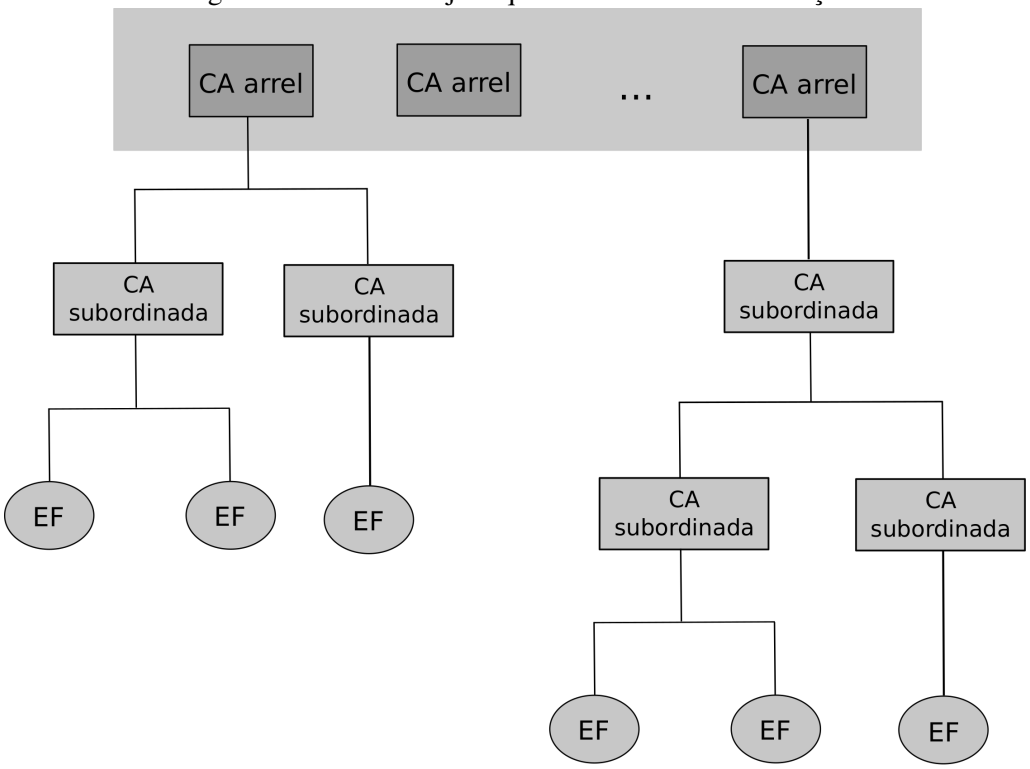
7.4 Les normes PKCS

Les normes PKCS (de l'anglès, *Public Key Cryptography Standards*) són un conjunt d'especificacions sobre criptografia de clau pública. Les especificacions les publica l'empresa RSA Laboratories i són elaborades conjuntament amb altres empreses del sector, com ara Apple, Microsoft, DEC, Lotus, Sun i MIT. Algunes d'aquestes especificacions han acabat esdevenint estàndards d'organismes internacionals com ara la IETF. L'objectiu d'aquestes publicacions és fomentar l'ús de la criptografia de clau pública i accelerar-ne el seu desplegament.

Actualment, hi ha 10 normes PKCS. Addicionalment, els PKCS#13 i #14, que cobreixen el xifrat i signatura fent servir criptografia de corbes el·líptiques i la generació de nombres pseudo-aleatoris, respectivament, no estan encara publicats. Els PKCS#2 i #4 es troben retirats des de 2010, quan van ser incorporats al PKCS#1 (tots dos descrivien també detalls sobre l'ús de l'RSA). El PKCS#6 descrivia la versió 1 dels certificats X.509 i està sent eliminat en favor de la versió 3 de X.509.

- PKCS#1: defineix mecanismes per xifrar i signar dades fent servir el criptosistema de clau pública RSA.
- PKCS#3: defineix el protocol d'establiment de claus de Diffie-Hellman.
- PKCS#5: descriu un mètode per xifrar una cadena amb una clau secreta derivada d'una contrasenya.
- PKCS#7: defineix una sintaxis general per missatges que inclouen atributs criptogràfics com ara signatures digitals o xifrat.
- PKCS#8: descriu un format per informació sobre claus privades, que inclou la clau privada per alguns algorismes de clau pública i, opcionalment, un conjunt d'atributs.

Figura 7.5: Estructura jeràrquica amb llista de confiança.



- PKCS#9: defineix tipus d'atributs per fer servir en altres estàndards PKCS.
- PKCS#10: descriu una sintaxi per peticions de certificació.
- PKCS#11: defineix una interfície de programació anomenada Cryptoki per a dispositius criptogràfics com targetes intel·ligents i targetes PCMCIA.
- PKCS#12 especifica un format portable per a emmagatzemar o transportar claus privades d'usuari, certificats, secrets, etc.
- PKCS#15 és un complement al PKCS#11 proveint un estàndard per credencials criptogràfiques emmagatzemades en tokens criptogràfics.

En aquest capítol, es detallen tres d'aquestes especificacions, els PKCS#1, #5 i #12.

7.4.1 PKCS#1

En el capítol anterior hem vist el funcionament bàsic de l'RSA. A la pràctica, però, no es fa servir l'RSA directament com s'ha descrit al capítol, ja que això resultaria insegur. Dos dels problemes més coneguts que té l'RSA amb la formulació que hem vist al capítol de Criptografia de Clau Pública són els atacs per l'ús d'exponents petits i el determinisme de l'algorisme.

D'una banda, fer servir exponents petits en combinació amb missatges m també petits fa que l'esquema sigui vulnerable. En concret, si $m^e < n$, aleshores podem desxifrar el missatge directament sense necessitat de conèixer la clau privada, calculant l'arrel e-èsima d'm sobre els enters. És a dir, si $c = m^e \mod n$ i $m^e < n$, aleshores $c = m^e$ i per tant $m = \sqrt[e]{c}$.

Exemple 7.11 Exemple d'atac per l'ús d'exponents petits

Suposem que tenim un parell de claus RSA de 64 bits formada pels valors:

```
PubK = (e, n) = (3, 18230703860219055503)
PrivK = (d, n) = (616012317821603203, 18230703860219055503)
```

Si volem xifrar un missatge $m = 55$, procediríem a elevar el missatge a l'exponent públic, com s'ha vist al capítol anterior:
$\texttt{c} = m^e \mod n = 55^3 \mod 18230703860219055503 = 166375$

Noteu com $55^3 = 166375$ i, per tant, $55^3 < n$.

Aleshores, un atacant que captura el missatge `c` i que coneix la clau pública, pot procedir a desxifrar el missatge, fent:

$\texttt{m} = c^{1/e} = 166375^{1/3} = 55$

D'altra banda, l'RSA és un algorisme determinista: el resultat de xifrar un text pla m amb una clau pública pub_k és sempre un mateix valor xifrat c. Si repetim el xifrat del mateix missatge amb la mateixa clau, el resultat és sempre el mateix valor c. Això fa que un atacant tingui un cert avantatge a l'hora d'intentar esbrinar el text pla m corresponent a un cert text xifrat c.

Suposem que un missatge m s'ha xifrat amb la clau pública pub_k fent servir RSA, donant com a resultat el valor xifrat c. Un atacant que coneix el valor xifrat c i la clau pública pub_k pot intentar desxifrar el missatge xifrant repetidament diversos valors m_i i anar comprovant si el resultat correspon al valor c que vol desxifrar. Si el conjunt de possibles missatges és petit, aquest atac sempre tindria èxit ja que l'atacant seria capaç de provar tots els possibles textos plans.

Exemple 7.12 Exemple d'atac pel determinisme de l'algorisme

Suposem que un elector que participa en unes eleccions on hi ha 4 candidats ($\{c_2, c_3, c_4, c_5\}$) ha d'enviar el seu vot xifrat amb la clau pública de la mesa electoral, indicant l'identificador del candidat votat.

Suposant que la mesa electoral té el següent parell de claus RSA també de 64 bits (on els exponents no són petits per evitar l'atac de l'exemple anterior):
`PubK = (e, n) = (7387905850005970831, 16517425874601317047)`
`PrivK = (d, n) = (6515200225287789487, 16517425874601317047)`
i que l'elector vol votar al candidat 4, el vot a enviar seria el valor:

$\texttt{vot} = m^e \mod n = 4^{7387905850005970831} \mod 16517425874601317047 = 8163478232469599798$

Un atacant que capturi el vot i vulgui conèixer-ne el seu contingut, només cal que generi ell mateix tots els possibles vots xifrats. Com que la clau pública de la mesa electoral és coneguda per tothom, l'atacant pot aconseguir aquesta informació i calcular:

$\texttt{v}_2 = 2^{7387905850005970831} \mod 16517425874601317047 = 11673347059272354770$
$\texttt{v}_3 = 3^{7387905850005970831} \mod 16517425874601317047 = 10980320764598560840$
$\texttt{v}_4 = 4^{7387905850005970831} \mod 16517425874601317047 = 8163478232469599798$
$\texttt{v}_5 = 5^{7387905850005970831} \mod 16517425874601317047 = 3701559658846578058$

Aleshores, l'atacant descobreix que l'elector ha votat al candidat 4, sense necessitat d'haver de desxifrar el vot.

Addicionalment, l'RSA ofereix xifratge homomòrfic. Donats dos missatges en pla, m_1 i m_2 i els seus corresponents textos xifrats c_1 i c_2 (amb la mateixa clau), el resultat de multiplicar els dos textos xifrats ($c_1 c_2$) és precisament el mateix valor que s'obté al multiplicar els dos textos plans m_1 i m_2 i xifrar-los

posteriorment.

En efecte, si tenim que:
$c_1 = E(m_1) = {m_1}^e \mod n$
$c_2 = E(m_2) = {m_2}^e \mod n$
aleshores:
$E(m_1) * E(m_2) = c_1 * c_2 = ({m_1}^e)({m_2}^e) \mod n = (m_1 * m_2)^e \mod n = E(m_1 * m_2)$

La propietat d'homomorfisme de l'RSA pot ser utilitzada per construir esquemes amb propietats interessants, però, d'altra banda, també pot suposar problemes de seguretat en segons quins escenaris. A partir d'un missatge xifrat $c_1 = E(m_1)$, un atacant podrà construir un segon missatge xifrat $c_2 = c_1 * E(\alpha)$ que correspondrà al missatge en pla $m_1 * \alpha$ sense conèixer la clau de xifratge ni el missatge en pla original m_1.

Així doncs, mentre que la propietat d'homomorfisme és útil per construir esquemes que manipulin dades mentre en preserven la seva confidencialitat, també pot ser una debilitat en segons quins desplegaments i depenent de l'ús que se'n faci.

Per tal de solucionar aquests problemes que apareixen amb la utilització de l'RSA en la seva definició bàsica, s'acostuma a afegir un conjunt de bits aleatoris com a *padding* al missatge en pla abans de xifrar-lo. D'aquesta manera, s'aconsegueix que el mateix missatge m pugui correspondre a diversos textos xifrats c, s'eviten els missatges m vulnerables per la seva representació i, alhora, es pot eliminar la propietat d'homomorfisme que presenta l'RSA.

Noteu que amb la introducció de bits aleatoris, l'RSA passa a ser un criptosistema probabilístic, on un mateix missatge xifrat amb una mateixa clau pot donar lloc a diversos textos xifrats. En canvi, el criptosistema d'ElGamal ja és probabilístic per definició.

L'estàndard PKCS#1 defineix tot un conjunt de recomanacions per implementar l'RSA. En concret, l'estàndard descriu primitives criptogràfiques, esquemes de xifrat, esquemes de signatura i detalls de codificació.

La versió 2.1 de l'estàndard PKCS#1 va ser republicada com a RFC 3447, amb unes petites correccions.

Esquemes de xifrat

Seguint la definició de l'estàndard, un esquema de xifrat consisteix en una operació de xifrat i una operació de desxifrat.

El PKCS#1 defineix dos esquemes de xifrat: RSAES-OAEP i RSAES-PKCS1-v1_5. En aquest capítol, veurem la descripció de l'esquema RSAES-OAEP. L'esquema RSAES-PKCS1-v1_5 s'inclou només per mantenir la compatibilitat amb les aplicacions ja existents, ja que actualment es coneixen atacs que fan que el seu ús no sigui recomanable. Abans però de descriure l'RSAES-OAEP, definirem i veurem un exemple de funció de generació de màscara, una construcció que es fa servir en RSAES-OAEP.

Funcions de generació de màscara

RSAES-OAEP	Tot i incloure les sigles AES, aquest esquema no te res a veure amb el criptosistema de bloc.

L'esquema de xifrat RSAES-OAEP fa ús d'una funció de generació de màscara en dues ocasions per tal de generar el valor EM a xifrar.

> Una **funció de generació de màscara** (MGF per les seves sigles en anglès, *Mask Generation Function*) és una funció que rep com a paràmetres una cadena de mida variable i la mida de sortida desitjada i retorna una cadena de la mida especificada a l'entrada.

És a dir, una funció MGF és una funció que rep una entrada de mida m_i bits i un valor de mida de sortida m_o i retorna una sortida de mida m_o bits:

$$MGF(\{0,1\}^{m_i}, m_o) \rightarrow \{0,1\}^{m_o}$$

Les funcions de generació de màscara són deterministes, ja que la sortida de la funció queda determinada de manera única per la seva entrada. A més, la seva sortida ha de ser pseudoaleatòria, de manera que coneixent una part de la sortida no se'n pugui generar la resta.

L'MGF1 és una funció de generació de màscara basada en una funció hash. Donada una funció hash H amb sortida de mida *hLen*, l'MGF1 es defineix de la següent manera:

```
definició mgf1(seed, maskLen)
    si maskLen > 2^32*hLen aleshores
        retornar "Error: màscara massa llarga"
    fi si
    T = ""
    per iteracio = 0 fins a ceil(maskLen/hLen) - 1 fes
     comptador =  I2OSP(iteracio, 4)
        T = T || H( seed || comptador )
    fi per
    retornar maskLen octets més significatius de T
fi definició
```

La funció I2OSP	La funció `I2OSP(x,y)` retorna el valor x representat fent servir y octets.

És a dir, la funció va concatenant el resultat d'aplicar una funció hash a la concatenació del valor que rep com a llavor (`seed`) i un comptador d'iteracions que es va incrementant d'un en un. Aquest procés es repeteix fins a tenir prou octets com per generar una sortida de la mida especificada com a paràmetre (`maskLen`). Finalment, com que `maskLen` pot no ser múltiple de la mida del hash (`hLen`), es possible que s'hagin de descartar els octets menys significatius del resultat acumulat. A l'hora de concatenar el comptador amb la llavor, es concatena la seva representació en quatre octets.

La funció ceil	La funció `ceil(x)` retorna el menor enter major o igual que x.

Exemple 7.13 **Exemple d'ús de l'MGF1** Suposem que volem fer servir l'MGF1 amb la funció hash sha-1 amb els següents valors d'entrada:

`maskLen = 45`
`seed = 0x5307`

La funció sha-1 produeix una sortida de 160 bits, és a dir, 20 octets. Per tant,
`hLen = 20`

En primer lloc, es comprova que la mida de sortida desitjada no sigui superior a $2^{32}hLen$. Com que no ho és ($maskLen < 2^{32}hLen$), es continua l'execució normalment, calculant el valor màxim que assumirà el comptador en el bucle:

`ceil(maskLen/hLen) - 1 = ceil(45/20) - 1 = 3 - 1 = 2`

Després, es procedeix a calcular el valor `T` a cada una de les iteracions del bucle:

Iteració 0:
`comptador = 0x00000000`
`seed || comptador = 0x530700000000`
`H(seed || comptador) = 0xc5230b3bcb615dbc0b9a63f6e975b0f327fc576c`

```
T || H( seed || comptador ) = 0xc5230b3bcb615dbc0b9a63f6e975b0f327fc576c


Iteració 1:
comptador = 0x00000001
seed || comptador = 0x530700000001
H(seed || comptador) = 0x12f189b1d17e5b688d856fc700ffd2b20aabb14e
T || H( seed || comptador ) = 0xc5230b3bcb615dbc0b9a63f6e975b0f327fc57
     6c12f189b1d17e5b688d856fc700ffd2b20aabb14e


Iteració 2:
comptador = 0x00000002
seed || comptador = 0x530700000002
H(seed || comptador) = 0x62f2e51fe23fdbfc3d200346f30157d5985d24ef
T || H( seed || comptador ) = 0xc5230b3bcb615dbc0b9a63f6e975b0f327fc57
     6c12f189b1d17e5b688d856fc700ffd2b20aabb14e62f2e51fe23fdbfc3d2003
     46f30157d5985d24ef


Finalment, es retornen els 45 octets més significatius, de l'últim valor T calculat:
c5230b3bcb615dbc0b9a63f6e975b0f327fc576c12f189b1d17e5b688d856fc700ffd
     2b20aabb14e62f2e51fe2
```

L'esquema de xifrat RSAES-OAEP

L'esquema de xifrat RSAES-OAEP defineix com xifrar un missatge M amb RSA afegint *padding* i introduint aleatorietat. L'esquema rep com a entrades tres valors, i retorna un valor xifrat:

```
C = RSAES-OAEP(M, (e,n), L)
```

on `M` és el missatge a xifrar; `(e,n)` és la clau pública RSA; i `L` és una etiqueta opcional (si no s'inclou, es pren com a valor la cadena buida).

A més de la funció de xifratge amb RSA que hem vist al capítol anterior, l'esquema també fa ús de dues funcions addicionals: `H`, una funció hash i `MGF`, una funció de generació de màscara

Octet

Un **octet** són 8 bits. En alguns estàndards es fa servir la paraula octet en comptes de byte per definir conjunts de 8 bits ja que, en realitat, la mida d'un byte depèn de la plataforma. Tot i que actualment gairebé tots els sistemes interpreten un byte com a 8 bits, en certs contextos el seu ús podria provocar ambigüitats.

Per descriure l'esquema, farem servir els termes `mLen` i `hLen` per referir-nos, respectivament, a la mida en octets del missatge `m` i de la sortida de la funció hash `H`. A més, el valor `k` representa la mida (també en octets) del capítol de la clau RSA utilitzada.

La Figura 7.6 mostra l'esquema de xifratge de l'RSA-OEAP. Com es pot apreciar, el primer pas de l'esquema és construir el *padding* per al missatge `M`. Això es fa concatenant quatre valors: `lHash`, el resultat d'aplicar la funció hash a l'etiqueta `L` (`lHash = H(L)`); `PS`, una cadena d'octets fixats a zero; un octet representant el valor `1`; i `M`, el missatge a xifrar. La longitud de la cadena `PS` és variable, i dependrà de la mida del missatge i de la sortida de la funció hash. De fet, es podria donar el cas que la cadena `PS` tingués longitud zero.

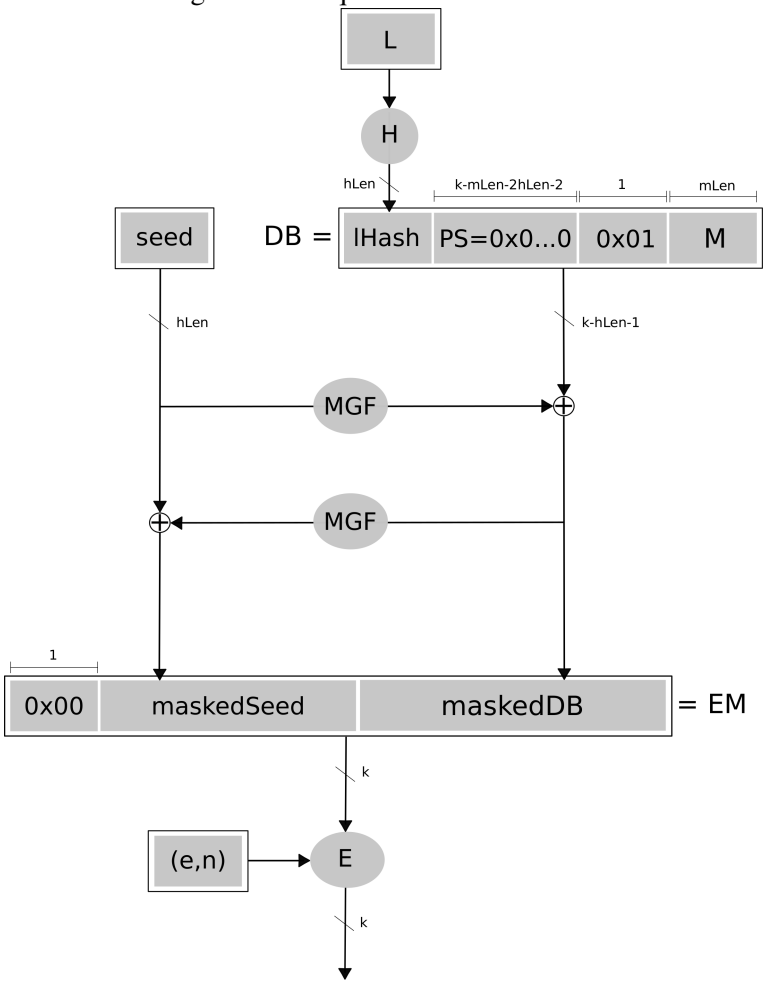
Així doncs,

```
DB = lHash || PS || 0x01 || M
```

En segon lloc, es genera una llavor aleatòria de mida *hLen*, representada com a *seed* a l'esquema.

En els següents passos hi entren en joc, d'una banda, la funció xor i, d'altra banda, la funció de generació de

Figura 7.6: Esquema de l'RSA-OAEP.



màscara *MGF* triada. Utilitzant aquestes dues funcions es calculen els valors *maskedDB* i *maskedSeed*:

```
maskedDB = DB ⊕ MGF(seed, k-hLen-1)
maskedSeed = seed ⊕ MGF(DB, hLen)
```

i es calcula el valor *EM* com la concatenació d'un octet de zeros amb *maskedSeed* i *maskedDB*:

```
EM = 0x00 ‖ maskedSeed ‖ maskedDB
```

Aquest valor `EM` té ara exactament `k` octets, i és el que es farà servir com a entrada de la funció de xifrat de l'RSA que hem vist al capítol anterior. Noteu que fent servir l'esquema RSAES-OAEP per xifrar se solucionen els problemes de l'RSA comentats anteriorment ja que, d'una banda, els valors a xifrar tenen ara sempre `k` octets i, d'altra banda, el xifrat deixa de ser determinista amb la inclusió del valor aleatori `seed`.

Per desxifrar un valor `C` xifrat amb RSAES-OAEP, es procedeix a desfer el camí realitzat a l'hora de xifrar-lo: en primer lloc, es desxifra el valor rebut fent servir la funció de desxifrat de l'RSA vista al capítol anterior. En segon lloc, assignarem el resultat del desxifrat a `EM`, i desfarem els passos realitzats a l'hora de xifrar fins a obtenir el valor original del missatge `M`.

Exemple 7.14 Exemple de xifratge fent servir RSAES-OAEP

En aquest exemple farem servir RSAES-OAEP per xifrar un missatge amb una clau de 512 bits i fent servir la funció sha1 com a funció hash. La clau pública que farem servir és:

```
n = 121745231563782351101796726796023575630699986274537835886656686101963
    667490409998929148563876090024139251265643218735316441793103154702608
    085900075427092633
e = 65537
```

El missatge a xifrar serà:
`message = 0x484f4c41`

Deixarem l'etiqueta buida (`L=''`) i farem servir com a llavor el valor:
`seed = 000000000000000000000000000000000051ead`

Així doncs, la clau pública a utilitzar té 512 bits de manera que $k = 64$ octets. Alhora, la funció sha1 té una sortida de 160 bits, pel que `hlen = 20`. El missatge a xifrar té mida `mLen = 4` octets. Per tant, la mida de `PS` serà:
`k-mLen-2hLen-2 = 64-4-2*20-2 = 18` octets.

Sabent la mida de `PS`, podem calcular el valor `DB`:

```
lHash = H('') = da39a3ee5e6b4b0d3255bfef95601890afd80709
DB = lHash || PS || 0x01 || M = da39a3ee5e6b4b0d3255bfef95601890afd807090000
    000000000000000000000000000000001484f4c41
```

Ara, procedim a aplicar la funció de generació de màscara:

```
dbMask = MGF(seed, k-hLen-1) = MGF(seed, 43) = e68e82475a8216e69b2ac31ab2
    0b60a168c155da68b1e64c8972335f90ebd96d57905b0ffe49ae92b14f3f
maskedDB = DB ⊕ dbMask =
    da39a3ee5e6b4b0d3255bfef95601890afd80709000...001484f4c41 ⊕
    e68e82475a8216e69b2ac31ab20b60a168c155da68b...9ae92b14f3f =
    3cb721a904e95deba97f7cf5276b7831c71952d368b1e64c8972335f90ebd96d5790
    5b0ffe49afdafe037e

seedMask = MGF(maskedDB, hLen) = MGF(maskedDB, 20) =
    9e883239d0bc279884730611a7f07b5e65f17474
maskedSeed = seed ⊕ seedMask =
    000000000000000000000000000000000051ead ⊕
    9e883239d0bc279884730611a7f07b5e65f17474 =
    9e883239d0bc279884730611a7f07b5e65f46ad9
```

Per tant, el valor que xifrarem amb la primitiva bàsica de xifratge RSA serà:
`EM = 0x00 || maskedSeed || maskedDB = 009e883239d0bc279884730611a7f07b5e65f`
`46ad93cb721a904e95deba97f7cf5276b7831c71952d368b1e64c8972335f90ebd96`
`d57905b0ffe49afdafe037e`

Per acabar, es procedeix a realitzar el xifratge. Com que tenim la clau pública expressada en enters en base decimal, una alternativa per a realitzar el càlcul és convertir prèviament la representació d'EM a base 10:
`c = E(e,n)(EM) =` $m^e \mod n$ `=`
`536217038208916855157918998855334509587413202413907763459400353831594476491127080549474416255526656440643378986467124634863899985157139786237339951511550 6`

Finalment, convertim de nou el resultat del xifratge a la representació hexadecimal de k octets:
`C = 6661be2c1357ba21cbc074362ae9e8e08898779e7df987d4ae2993e1e02ab051cf9f7`
`a109fc3389c96779e6206c49e695da846efa3945ba0c9e43adaddcc2ff2`

7.4.2 PKCS#5

La norma PKCS#5 proveeix recomanacions per a la implementació de criptografia basada en contrasenyes. En concret, la norma descriu funcions de derivació de claus, esquemes de xifrat, esquemes d'autenticació de missatges i la sintaxi ASN.1 que identifica les diferents tècniques.

La versió 2.0 de l'estàndard PKCS#5 va ser republicada com a RFC 2898.

Sal criptogràfica i número d'iteracions

Avui en dia, l'ús de contrasenyes és un mètode molt freqüent per protegir accessos a sistemes o secrets. Les contrasenyes escollides pels usuaris, però, acostumen a no ser adequades per a ser utilitzades directament com a claus d'esquemes criptogràfics segurs. D'una banda, solen ser massa curtes i, d'altra banda, poden ser susceptibles a atacs per diccionari.

Bits de sal	La denominació de bits de sal fa referència a què la sal fa variar el gust dels aliments, de la mateixa manera que els bits de sal fan variar les contrasenyes.

L'ús d'una sal en criptografia basada en contrasenyes s'ha utilitzat tradicionalment per produir conjunts de claus grans a partir d'una única contrasenya. La clau corresponent a una contrasenya se selecciona de dins d'aquest conjunt de manera aleatòria a partir del valor de sal. Una clau individual se selecciona aplicant una funció de derivació de claus (KDF per les seves sigles en anglès, *Key Derivation Function*).

L'ús de bits de sal té, principalment, dos beneficis:

1. Es dificulta que un atacant pugui precalcular totes les claus corresponents a un diccionari de contrasenyes. Amb l'ús de n bits de sal, cada contrasenya té 2^n possibles claus, pel que el cost de precalcular-les augmenta considerablement.
2. S'aconsegueix que la probabilitat que la mateixa clau sigui seleccionada dues vegades sigui molt baixa. Això soluciona alguns dels problemes que apareixen quan es reutilitzen claus.

A més de fer servir uns bits de sal, una altra tècnica que s'utilitza habitualment en criptografia basada en contrasenyes és incrementar deliberadament el temps de càlcul necessari per calcular cada clau, de manera que aquest increment no sigui significatiu per a un usuari que necessita calcular una clau però sí que ho sigui per a un atacant que es troba fent una cerca exhaustiva. Aquest increment del temps de càlcul es fa augmentant el número d'iteracions que realitza la funció de derivació de clau. La norma PKCS#5 recomana fer servir com a mínim 1.000 iteracions. De totes maneres, moltes de les implementacions actuals ja superen amb escreix aquest valor.

Exemple 7.15 Emmagatzemament de contrasenyes en sistemes basats en Unix

Els sistemes basats en Unix tradicionalment fan servir sal criptogràfica per emmagatzemar les contrasenyes dels usuaris del sistema. Així, normalment les contrasenyes dels usuaris es troben emmagatzemades en el següent format:
`$id$salt$hashed`

on `id` és l'identificador de l'algorisme de hash utilitzat, `salt` és el valor de sal i `hashed` és el valor del hash contrasenya aplicant la sal.

Per exemple, les següents tres entrades serien vàlides per emmagatzemar la informació necessària per validar un usuari que faci servir la contrasenya *criptografia* per entrar al sistema:

```
$6$76YTAM$mbfTXZY1.D7qaIPbzKcQX8yBXuhwTr4D9u3vpctvU7XFcqPkUzgK3.z.93DTJV
      6.zzoMf9GaoDIugnJuY99CC1
$6$86YTAM$qG0.v6FvNTz9j4RKAXB5TMh1twEGhGPxvNlfZiCkpNhBYv4B4MBOYmUkdFUKIR
      IEB.Qfs2Gyqv2ohmxLtgN170
$1$86YTAM$VD8sYNaSgNCoEzFlf2OhK.
```

Les dues primeres correspondrien a representacions de la mateixa contrasenya fent servir sha-512 com a funció hash i dos valors de sal diferents. És important notar com un petit canvi en el valor de sal (`76YTAM` en comptes de `86YTAM`) canvia radicalment el valor resultant.

En canvi la tercera entrada correspondria a l'emmagatzematge de la mateixa contrasenya amb la segona sal, però fent servir MD5. En aquest cas, podem veure com canviar la funció de hash també fa variar el resultat, encara que la contrasenya i la sal coincideixin.

Funcions de derivació de clau

Una **funció de derivació de claus** produeix una clau derivada a partir d'una clau base i d'altres paràmetres.

Una funció de derivació de claus basada en contrasenyes és un cas específic d'una funció de derivació de claus on la clau base és la contrasenya i els altres paràmetres són un valor de sal i un número d'iteracions.

La norma PKCS#5 defineix dues funcions de derivació de claus basades en contrasenyes: PKKDF1 i PBKDF2. La funció PBKDF1 s'inclou només per mantenir la compatibilitat amb aplicacions ja existents i es recomana l'ús de PBKDF2 per a les noves aplicacions.

7.4.3 PKCS#12

L'estàndard PKCS#12 especifica un format per emmagatzemar i transportar informació sobre l'identitat de persones, com ara claus privades, certificats i secrets, entre d'altres dades criptogràfiques.

Un dels usos més habituals d'aquest format és l'emmagatzemament d'una clau privada i del seu corresponent certificat digital x.509 o bé per emmagatzemar tots els certificats d'una cadena de confiança.

La versió 1.1 de l'estàndard PKCS#12 va ser republicada com a RFC 7292.

L'estàndard suporta diferents modes de privacitat i integritat per a la transferència d'informació personal. En concret, l'estàndard suporta quatre combinacions, dos modes de privacitat i dos d'integritat:

- Mode de privacitat de clau pública: la informació personal s'empaqueta i es xifra amb una clau pública de la plataforma de destí. Es poden recuperar les dades amb la clau privada corresponent.
- Mode de privacitat amb contrasenya: la informació personal es xifra amb una clau simètrica derivada del nom d'usuari i d'una contrasenya.
- Mode d'integritat amb clau pública: la integritat es garanteix amb una signatura digital sobre el contingut, realitzada amb la clau privada de la plataforma d'origen. La signatura és verificada a la destinació, fent servir la clau pública corresponent.
- Mode d'integritat amb contrasenya: la integritat és garanteix a través d'un Codi d'Autenticació de Missatge (MAC) derivat d'una contrasenya.

7.5 Formats de representació de dades

L'ASN.1 (per les seves sigles en anglès, *Abstract Syntax Notation number One*) és un estàndard que descriu una notació formal utilitzada per descriure dades en protocols de comunicacions. Aquesta notació permet representar dades de manera independent de les codificacions específiques de cada màquina, del sistema operatiu o del llenguatge de programació utilitzat. Les normes PKCS fan servir aquest estàndard per descriure com emmagatzemar i transmetre claus i altres tipus de material criptogràfic.

Organitzacions darrere l'ASN.1

L'ASN.1 és un estàndard conjunt de la ISO (*International Organization for Standardization*), l'IEC (*International Electrotechnical Commission*) i de l'ITU-T (*International Telecommunication Union Telecommunication Standardization Sector*).

L'estàndard permet definir **tipus de dades** i **valors**. Un tipus de dades és una categoria d'informació (per exemple, numèrica o textual). Un valor és una instància d'un tipus concret. La sintaxi ASN.1 defineix tres categories de tipus de dades: tipus simples, que són atòmics; tipus estructurats, que tenen components; i tipus etiquetats, que són derivats d'altres. Es pot assignar un nom als tipus i valors ASN.1, de manera que aquest nom es pot fer servir per definir altres tipus i valors.

Així, per exemple, els enters (`INTEGER`), les cadenes de bits (`BIT STRING`) o el valor null (`NULL`) són tipus simples. En canvi, la seqüència (`SEQUENCE`), una col·lecció ordenada d'un o més valors d'altres tipus, i el conjunt (`SET`), una col·lecció desordenada d'un o més valors d'altres tipus, són tipus estructurats.

La notació ASN.1 es complementa per l'especificació d'un conjunt d'algorismes anomenats **regles de codificació** (en anglès es coneixen com *encoding rules*) que determinen la representació exacta de cada missatge en octets. Tres de les famílies de regles de codificació estandarditzades són: *Basic Encoding Rules* (BER), *Packed Encoding Rules* (PER) i *XML Encoding Rules* (XER).

Codificacions canòniques

De la mateixa manera que la codificació DER ens ofereix una codificació única dins de BER, existeixen representacions canòniques de les regles PER (anomenades CANONICAL-PER) i XER (conegudes com a Canonical XML Encoding Rules o CXER).

En criptografia sovint necessitem una representació única d'una certa dada. Els estàndards de codificació comentats ofereixen però diverses maneres de codificar un mateix valor, pel que no són adients per fer servir en criptografia. Les regles de codificació DER (de l'anglès, *Distinguished Encoding Rules*) són un subconjunt de les regles BER que ofereixen una codificació única a cada valor ASN.1. En concret, les regles DER especifiquen, per a cada valor a codificar, quin dels possibles mètodes de codificació BER s'ha d'utilitzar en aquell cas, assegurant així una codificació única. Així doncs, codificar fent servir les regles DER ens permet garantir que els processos criptogràfics que realitzem no es veuen alterats per l'ús de diferents representacions d'una mateix valor.

Exemple 7.16 Exemple del resultat de codificar en BER i DER

Fent servir la codificació BER, el valor booleà `Cert` pot ser codificat de 255 maneres diferents, que corresponen als 255 valors diferents de zero que es poden representar amb un byte ($2^8 - 1 = 255$). La sintaxi DER ens indica quina d'aquestes 255 maneres hem de triar per tal de codificar el booleà `Cert`.

Així, molts estàndards relacionats amb la criptografia fan servir DER per codificar dades. Per exemple, les dades que són signades dels certificats X.509 que hem vist a la Secció 7.3.1 o les de les llistes de revocació de certificats descrites a la Secció 7.3.2 es codifiquen en DER. Això permet que les comprovacions de les signatures digitals retornin els resultats esperats.

A vegades però, fer servir fitxers binaris per transmetre contingut criptogràfic no és el més adient. PEM és una codificació printable que fa servir 64 caràcters que són universalment representables: les lletres de la `a` a la `z` en minúscula i majúscula, els dígits del `0` al `9` i els símbols `+` i `/`. Així, cada caràcter permet codificar 6 bits d'informació ($2^6 = 64$). Addicionalment, el caràcter `=` es fa servir com a caràcter especial per a indicar com tractar el padding de cada missatge. El text resultant d'una codificació PEM consisteix en un conjunt de línies de 64 caràcters, exceptuant l'última línia que pot contenir un nombre de caràcters menor.

L'origen del format PEM

El protocol Privacy-enhanced Electronic Mail (PEM) va ser el primer estàndard en proposar l'ús d'una codificació en base 64 amb caràcters printables i línies curtes. Tot i que el protocol va caure en desús, la codificació encara es fa servir per transferir material criptogràfic de manera printable. Actualment, es fa servir el nom fitxer PEM per indicar fitxers amb material criptogràfic codificats en base 64 (inspirats amb la codificació original de l'RFC PEM), però que van més enllà del que s'especifica a l'estàndard. Per exemple, l'ús de les etiquetes `--BEGIN CERTIFICATE--` i `--END CERTIFICATE--` no es troba especificada a l'estàndard.

Així doncs, el sistema de codificació PEM ens permet representar una cadena d'octets qualsevol en un conjunt de caràcters printables representats en línies curtes, la majoria de les quals tenen la mateixa mida. La cadena de caràcters representant el material criptogràfic codificat en base 64 es troba emmarcada dins d'unes etiquetes que n'indiquen l'inici i el final. Per exemple, un certificat digital representat en PEM es trobaria emmarcat entre les etiquetes `--BEGIN CERTIFICATE--` i `--END CERTIFICATE--`:

```
--BEGIN CERTIFICATE--
MIIB0TCCATqgAwIBAgIQUq+2SdEkLr5K6xqjSEvRsDANBgkqhkiG9w0BAQUFADAU
MRIwEAYDVQQDEwlsb2NhbGhvc3QwHhcNMTIwODA0MDA0OTEyWhcNMTcwODA0MDAw
MDAwWjAUMRIwEAYDVQQDEwlsb2NhbGhvc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
[...]
Y2nd44bYEpmaBy7XJ5UIGEkuD3VIxT2S+2bCwkRR+9/+7vggR2q7l7YEktM2mFBI
yqOMOroAw+5cdc06c/B7UimwKFczsyhi9LUIr3rXI42FdXBHWw==
--END CERTIFICATE--
```

7.6 Els problemes de la PKI en desplegaments reals

Tot i que la PKI semblava que hauria de resoldre molts dels problemes de seguretat als quals ens enfrontem, el seu desplegament no ha arribat a complir amb les expectatives que s'hi havien dipositat: avui en dia les PKIs no són tant populars com es creia que arribarien a ser i les garanties de seguretat que ofereixen les infraestructures de clau pública no sempre estan a l'alçada del que, a nivell teòric, haurien d'oferir.

Així, per exemple, l'ús del protocol HTTPS es troba molt estès avui en dia. L'HTTPS fa servir una infraestructura de clau pública per autenticar els servidors web, normalment a través del seu domini. A més, la clau pública del certificat del servidor es fa servir per establir un canal de comunicació segur entre el servidor i el client.

Qui controla les CAs de confiança dels navegadors?

Un estudi publicat l'any 2013 sobre l'estat de l'ecosistema HTTPS reporta que només un 20% dels certificats d'autoritats de certificació de confiança dels principals navegadors corresponen a CAs comercials. La resta d'autoritats es troben controlades per empreses, institucions financeres, institucions religioses, museus i biblioteques.

Per tal de poder validar els certificats dels servidors, els navegadors tenen una llista d'autoritats de certificació de confiança en les quals confien. Aquesta llista acostuma a tenir uns pocs centenars de certificats arrel, el que porta a confiar en uns pocs milers de certificats de CA. Depenent de la configuració del sistema, la llista pot provenir del sistema operatiu o del navegador, i hi ha diferències notables en les llistes de certificats de les diferents configuracions.

Els sectors més crítics amb aquest model defensen que un dels principals problemes que hi ha és que qualsevol CA de confiança té el poder de signar qualsevol domini. Certament, amb el model actual, una CA d'algun país remot que tingui el seu certificat arrel en el navegador d'un usuari gaudirà de la mateixa confiança que una CA espanyola a l'hora d'emetre un certificat per a una pàgina web amb un domini del govern espanyol. Però no només això, les CAs arrel tenen també el poder de crear autoritats de certificació intermèdies que, excepte en entorns molt específics, també tindran el poder d'emetre certificats per a qualsevol domini. En aquesta línia, en els últims anys hi ha hagut diversos incidents de seguretat relacionats amb la PKI de la Web.

Al 2012, l'autoritat de certificació Trustwave (en la qual confien els principals navegadors) va emetre un certificat de CA i va incloure la clau privada corresponent dins d'un dispositiu hardware segur, que va llogar a una empresa amb l'objectiu que aquesta pogués espiar les connexions xifrades amb TLS dels seus empleats. La pròpia CA va reconèixer que havia dut a terme aquesta pràctica. Aquest incident va alertar d'una pràctica que es rumoreja que és habitual entre les autoritats de certificació, i que posa en perill la seguretat d'Internet.

Google ha detectat en nombroses ocasions certificats fraudulents afectant a algun dels seus dominis. Així, al 2014 va denunciar el National Informatics Centre (NIC) de l'Índia (una autoritat subordinada de l'Indian Controller of Certifying Authorities) estava emetent certificats no autoritzats que afectaven alguns dels seus dominis, algun domini de Yahoo i altres dominis. Al desembre de 2013, Google ja havia detectat certificats fraudulents per als seus dominis emesos per l'entitat de certificació francesa ANSSI i, uns mesos abans, al gener del mateix any, per l'entitat turca Türktrust.

Google és capaç de detectar aquests atacs gràcies a vàries mesures que ha anat desplegant amb el temps. Així, per exemple, el navegador Chrome porta incorporat *pinning* de certificats per a alguns dels dominis de google.

El *pinning* és un procediment pel qual s'associa un host a una identitat criptogràfica (una o vàries claus públiques o certificats en una cadena de certificats X.509). La validació d'un pin consisteix a comprovar que almenys una de les claus públiques especificades es troba en la cadena de certificació del host.

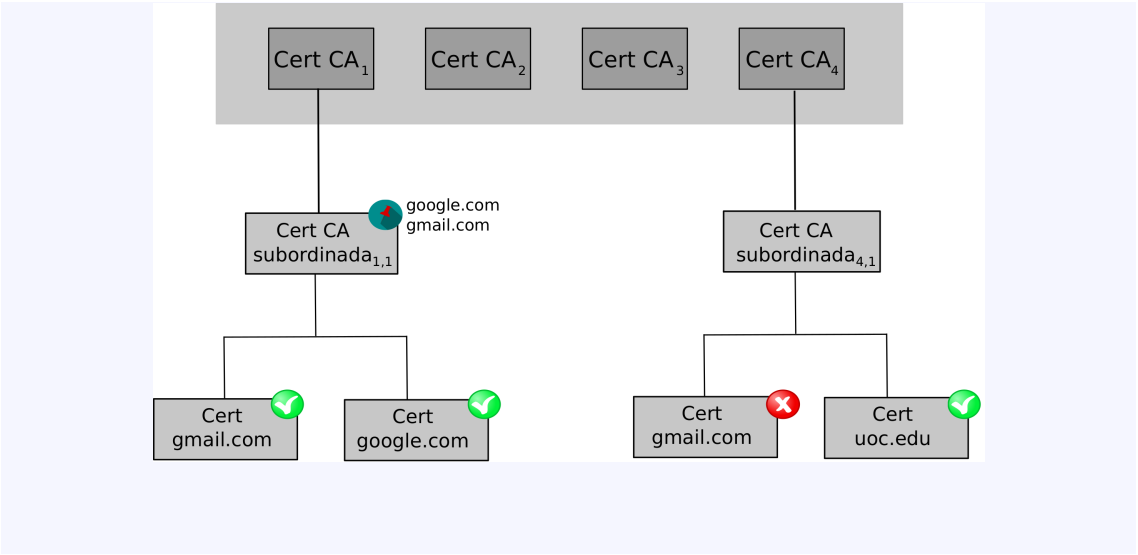
Exemple 7.17 Pinning de certificats en el navegador Chrome

El navegador Chrome inclou un conjunt de claus públiques per a dominis de Google, fet que permet als seus usuaris detectar, per exemple, quan es troben davant d'un certificat de gmail que aparenta ser vàlid (es troba dins d'una cadena de certificació vàlida atenent als certificats de confiança del navegador) però que no inclou cap de les claus públiques especificades, com es donaria en els casos comentats anteriorment.

Així, per exemple, suposem una instància del navegador Chrome que té quatre certificats a la seva llista de confiança, com es mostra a la figura següent. Si no hi ha més restriccions, qualsevol certificat signat amb la clau privada corresponent a algun dels certificats de la llista confiança (o alguna autoritat de certificació subordinada) serà considerat com a vàlid (si totes les comprovacions esmentades a la Secció 7.2.5 són satisfactòries).

Ara bé, si s'afegeix un pin que vinculi el certificat de la CA subordinada$_{1,1}$ amb els dominis google.com i gmail.com, caldrà que la cadena de certificació dels certificats d'aquests dominis passi pel certificat amb el pin per donar el certificat per vàlid. Així, els certificats de gmail.com i google.com emesos per la CA subordinada$_{1,1}$ són vàlids, mentre que el certificat de gmail.com emès per la CA subordinada$_{4,1}$ no ho serà. En canvi, un certificat d'un altre domini que no tingui cap pin (com ara uoc.edu) emès per la mateixa CA subordinada$_{4,1}$, serà considerat com a vàlid.

Noteu que l'existència d'un pin no elimina el requeriment de validar la cadena de certificació, és a dir, el pin afegeix comprovacions alhora de validar un certificat, però no n'elimina.



Altres dels problemes que sorgeixen amb els desplegaments pràctics de les infraestructures de clau pública i que no estan limitats a l'https són fruit dels conflictes d'interès entre els diferents actors que participen de la PKI, la falta d'incentius per a segons quines accions crítiques per al bon funcionament de la infraestructura, la confusió que generen certes especificacions o la falta d'usabilitat. Així, per exemple, si analitzem les alternatives de consulta de l'estat de revocació d'un certificat, trobem que el protocol OCSP comporta dificultats alhora d'interpretar les seves respostes (com es comentava a la Secció 7.3.3), mentre que l'emissió de CRLs és un procediment costós per a la CA per al qual no en té un incentiu directe. Pel que fa la usabilitat, potser un dels exemples més clars de sistemes que no arriben a ser utilitzats en tot el seu potencial és el dni electrònic: mentre que gairebé tota la ciutadania espanyola disposa de certificats digitals en el seu document d'identitat, la necessitat de tenir un dipositiu hardware capaç de llegir el dni i la dificultat d'instal·lar-lo, configurar-lo i fer-lo servir en un equip domèstic, fan que en molts casos aquests certificats no es facin servir.

7.7 Resum

En aquest capítol, s'ha presentat la infraestructura de clau pública, tot repassant les entitats que hi participen i el seu paper dins de la infraestructura, les fases per les quals passa un certificat digital des de la seva creació fins a la finalització del seu ús i els estàndards més importants que detallen diferents processos de la PKI. Finalment, s'han repassat els formats més habituals per codificar informació criptogràfica i s'ha discutit sobre els problemes que presenten els deplegaments reals de les infraestructures de clau pública, més enllà dels conceptes teòrics detallats als estàndards.

7.8 Solucions dels exercicis

Exercici 7.1:

a i d: Per tal de garantir el no-repudi, la clau privada no pot ser coneguda per ningú, a part del subscriptor del certificat. Per tant, caldrà que les claus s'hagin generat o bé pel propi subscriptor, o bé en un dispositiu hardware segur.

Exercici 7.2:

c: Les afirmacions a i b són falses, ja que tant l'OCSP com les CRLs ens permeten comprovar l'estat de revocació d'un certificat digital, però no la seva validesa (que es troba explicitada en el propi certificat). Les respostes d i e també són falses, ja que podem obtenir també aquestes respostes en altres situacions, per exemple, preguntant per certificats que no hagin estat mai emesos.

Exercici 7.3:

El resultat de la validació del certificat és:

	Resultat de processar l'extensió	
	`TRUE`	`FALSE`
L'aplicació reconeix l'extensió	`TRUE`	`FALSE`
L'aplicació no reconeix l'extensió	`FALSE`	`FALSE`

Exercici 7.4:

El resultat de la validació del certificat és:

	Resultat de processar l'extensió	
	`TRUE`	`FALSE`
L'aplicació reconeix l'extensió	`TRUE`	`FALSE`
L'aplicació no reconeix l'extensió	`TRUE`	`TRUE`

Exercici 7.5:

Cap dels segells de temps proposats ens permetria demostrar que les signatures han estat recollides en el període de temps indicat. El segell de temps a demostraria que les signatures han estat creades abans de l'inici del període establert. El segell de temps b no podria garantir que no s'han creat signatures ni abans ni després del període de temps establert. El segell de temps c no garantiria que les signatures no han estat creades abans de l'inici del període. Un segell de temps només permet garantir que una dada existeix en un instant de temps concret.

7.9 Bibliografia

Adams, C., Steve L. (2003). *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional.

Choudhury, S. Bhatnagar, K., Haque, W. (2002). *Public key infrastructure implementation and design.*. John Wiley & Sons, Inc.

Kuhn, D. Richard, et al. (2001). *Introduction to public key technology and the federal PKI infrastructure*. National Institute of Standards and Technology.

Obaidat, M., Boudriga, N. (2007). *Security of E-Systems and Computer Networks*. Cambridge University Press.

International Telecommunication Union (2000). *Recommendation X.509 - The Directory: Public-key and attribute certificate frameworks.*

International Telecommunication Union (2015). *Recommendation X.680 - Abstract Syntax Notation One (ASN. 1): Specification of Basic Notation*

Jonsson, J., Kaliski, B. (2003). *Public-key cryptography standards (PKCS)# 1: RSA cryptography specifications version 2.1.*

Nystrom, M., et al. (2014). *PKCS# 12: Personal information exchange syntax v. 1.1.*

Kaliski, B. (2000). *PKCS# 5: Password-based cryptography specification version 2.0.*

J. Linn (1993). *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*.
`https://tools.ietf.org/html/rfc1421`

S. Kent (1993). *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*.
`https://tools.ietf.org/html/rfc1422`

D. Balenson (1993). *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*.
`https://tools.ietf.org/html/rfc1423`

B. Kaliski (1993). *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*.
`https://tools.ietf.org/html/rfc1424`

S. Kille (1995). *A String Representation of Distinguished Names*.
`https://tools.ietf.org/html/rfc1779`

C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen (1999). *SPKI Certificate Theory*.
`https://tools.ietf.org/html/rfc2693`

B. Kaliski (2000). *PKCS #5: Password-Based Cryptography Specification Version 2.0*.
`https://tools.ietf.org/html/rfc2898`

C. Adams, P. Cain, D. Pinkas, R. Zuccherato (2001). *Internet X.509 Public Key Infrastructure Time-Stamp Protocol*.
`https://tools.ietf.org/html/rfc3161`

W. Polk, R. Housley, L. Bassham (2002). *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.
`https://tools.ietf.org/html/rfc3279`

J. Jonsson, B. Kaliski (2003). *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography*

Specifications Version 2.1.
https://tools.ietf.org/html/rfc3447

M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, R. Nicholas (2005). *Internet X.509 Public Key Infrastructure: Certification Path Building*.
https://tools.ietf.org/html/rfc4158

C. Adams, S. Farrell, T. Kause, T. Mononen (2005). *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*.
https://tools.ietf.org/html/rfc4210

D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.
https://tools.ietf.org/html/rfc5280

R. Housley (2009). *Cryptographic Message Syntax (CMS)*.
https://tools.ietf.org/html/rfc5652

K. Moriarty, Ed., M. Nystrom, S. Parkinson, A. Rusch, M. Scott (2014). *PKCS #12: Personal Information Exchange Syntax v1.1*.
https://tools.ietf.org/html/rfc7292

OpenSSL Software Foundation (Consultat per última vegada: setembre de 2016). *OpenSSL Cryptography and SSL/TLS Toolkit Documentation*.
https://www.openssl.org/docs/

S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams (2013). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*.
https://tools.ietf.org/html/rfc6960