
SPD - Autoritat Certificadora

Tutor de la pràctica : Antonio Bueno

Francesc Xavier Bullich Parra, GEINF (UDG-EPS)

5 de Gener de 2020

Contents

Objectiu	3
Introducció	3
Infraestructura PKI	3
Confiança	4
Jerarquia d'una CA	5
Construcció d'una CA amb OpenSSL	6
Creació del certificat arrel	6
Directoris	6
Fitxers base de dades	7
Fitxer de configuració OpenSSL	7
Creació de les claus	8
Autoritats intermedies	10
Estructura de Directoris	11
Fitxers	11
Creació dels certificats i les cadenes de confiança	11
Confiança entre serveis	15
Exemple de confiança	16
Revocació de certificats	18
CRL	18
OCSP	19
Referències	22

Objectiu

L'objectiu del treball és mostrar quins són els elements que componen una autoritat certificadora (CA). També mostraré una forma de crear una autoritat certificadora pròpia, per tal de poder crear certificats de confiança propis.

Introducció

Una autoritat certificadora és una entitat de confiança que és responsable d'emetre i revocar certificats digitals. Aquests certificats s'usen principalment per garantir la seguretat de les comunicacions digitals via TLS-HTTPS. S'utilitza criptografia de clau pública per generar les claus i els certificats.

La CA dona el servei de certificació, que garanteix la relació entre una persona (física o jurídica) i la seva clau pública (certificat), es a dir, un certificat digital ha de identificar a una persona i s'ha de poder confiar en aquesta relació.

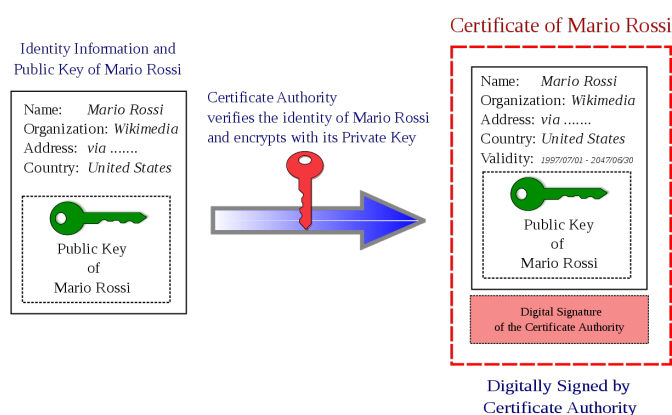


Figure 1: Certificat signat CA - Wikipedia

Infraestructura PKI

Tota la gestió de certificats digitals es fa a través d'un PKI (Public key infraestructure). De fet la CA és només una part del PKI.

El PKI es divideix en diversos subsistemes:

- CA: Rep les peticions i crea els certificats. Es responsable d'administrar el cicle de vida dels certificats (temps màxim de validesa o revocació).
- RA: Autoritat de registre és una interfície entre l'usuari i l'autoritat de certificació. Ha d'identificar el sol·licitants o titulars dels certificats.
- VA: Autoritat de validació emmagatzema els certificats digitals i administra l'allista de certificats caducats o revocats. També posa a disposició tots els certificats emesos per l'autoritat de certificació.
- Autoritat de custòdia: Emmagatzema de forma segura les claus de xifrat que utilitza l'autoritat certificadora.

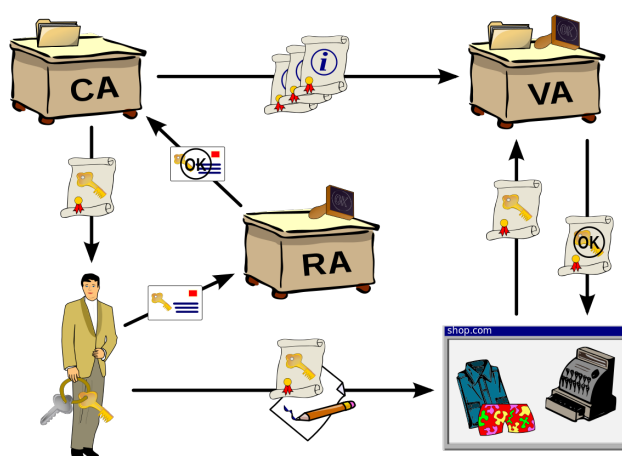


Figure 2: Esquema PKI - Wikipedia

Confiança

Existeixen 2 tipus de certificats: certificats autosignats o certificats signats per una autoritat certificadora.

Els certificats autosignats es poden crear fàcilment. Es tracten de certificats els quals el propietari i el signatari són el mateix. En alguns casos aquests certificats són suficients per determinades accions. Moltes aplicacions però, sobretot navegadors, no accepten aquest tipus de certificat i exigeixen que el certificat estigui signat per una autoritat certificadora de confiança.

Els navegadors confien en una sèrie de CA, que suposadament, compleixen tots els requeriments per actuar com a CA. Aquesta confiança fa que si es troben un lloc web amb un certificat signat per alguna d'aquestes entitats, validin el certificat i deixen navegar sense problemes. Si troba un certificat autosignat o signat per una CA que no té com a autoritat de confiança, no ens deixarà navegar.

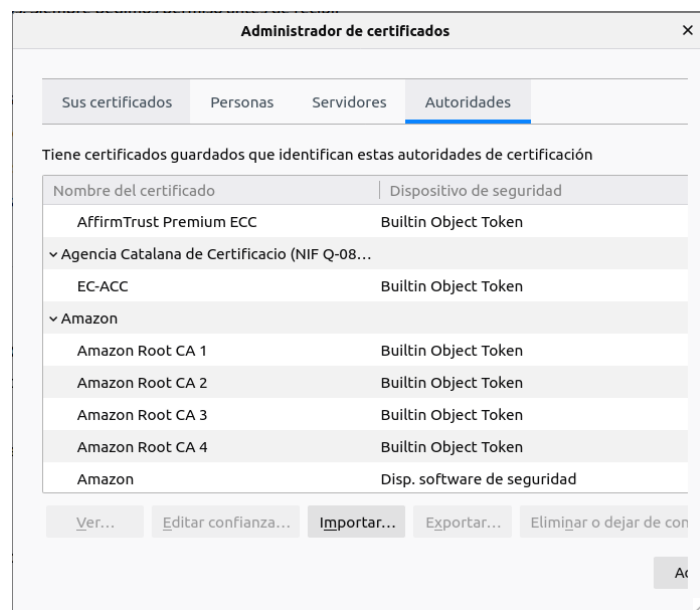


Figure 3: Autoritats confiança firefox

Si creem una autoritat de certificació propia, haurem d'incorporar-la a les autoritats de confiança dels nostres dispositius per tal que les reconeguim.

Jerarquia d'una CA

Com ja s'ha explicat, es necessari que els certificats estiguin signats per una CA. Aquesta CA tindrà el seu propi certificat per tal de poder signar les peticions que rebí.

Les CA solen treballar en jerarquia: tenen un certificat arrel, que està a sobre de tota la jerarquia. Aquest certificat s'utilitza per crear certificats intermedis que poden servir per diferents propòsits. Així forma una cadena de confiança

El que se sol fer és que el certificat arrel només firma els certificats intermedis. Els intermedis s'encarregen de signar peticions dels diferents usuaris de la CA. D'aquesta manera es pot protegir millor el certificat arrel, no cal que tinguem exposat el certificat arrel. Si algun dels certificats fos compromés, s'hauria de revocar, el que comportaria revocar també tots els certificats emesos. Per tant si utilitzem certificats intermedis és menys probable que es comprometi el certificat arrel. D'aquesta manera no perdriem tota la CA.

Els certificats arrel son autosignats degut a que no tenen cap instancia superior que el pugui signar. Per tant en aquest cas s'ha "confiar" que el certificat arrel sigui "correcte".

Construcció d'una CA amb OpenSSL

Podem trobar diversos proveïdors de certificats a Internet. Molts d'aquests proveïdors ja els tenen les aplicacions com a autoritats de confiança. Podriem utilitzar algun d'aquests proveïdors per obtenir certificats, però en alguns casos potser és més simple crear els nostres propis certificats, com per exemple si volem crear un tunel VPN.

Utilitzaré OpenSSL per mostrar els passos necessaris per montar la nostra propia autoritat certificadora. En aquest treball em centraré en l'apartat de la CA dins del PKI. No donaré una interfície RA i faré un VA molt simple utilitzant directoris i fitxers de linux.

Per aquesta part seguiré la documentació de [2], que precisament es una guia de com crear una CA. Dels que he trobat és el que està millor i té uns fitxers de configuració molt complets.

Tots els fitxers que composaran la CA són al **directori autoritat**.

Creació del certificat arrel

Com ja he explicat, per tenir una CA el primer de tot es crear el certificat arrel per poder crear una cadena de confiança. Aquest certificat només el farem servir per signar els certificats intermedis.

Per seguretat aquest certificat no l'hem d'utilitzar per res més per tant les claus haurien d'estar en un lloc separat de la resta i protegit. En aquest cas però com que és un cas acadèmic, tota la informació relacionada amb el certificat arrel estara en el subdirectori **autoritat/arrel**.

Directoris

Dins de **arrel** es creen els següents directoris per tal d'ordenar tots els fitxers que s'utilitzaran.

- **private**: Aquí hi aniran les claus privades. Com he comentat abans aquestes claus no s'haurien de guardar amb la resta d'informació sino que haurien d'estar separats i protegits.
- **certs**: Aquí si guardaran tots els certificats que signi el certificat arrel
- **crl**: Quan es revoqui algun certificat signat per l'arrel es guardarà en aquest directori.

Per posar una mica de protecció el directori private li treurem la resta de permisos

```
1 mkdir private certs crl
2 chmod 700 private
```

Fitxers base de dades

Utilitzarem fitxers a mode de base de dades.

- index.txt: Aquest fitxer contindrà tots els registres de certificats que es creïn. Tindran informació del número de sèrie i l'estat (valid, revocat o caducat)
- serial: Aquest fitxer conté un valor en format hexadecimal que utilitzarà openssl per generar els certificats. Cada certificat tindrà un número de sèrie. En aquest fitxer s'hi guarda el valor "autonumeric", per a cada nou certificat s'incrementa el valor.

```
1 touch index.txt
2 echo 1000 > serial
```

Fitxer de configuració OpenSSL

Al directori **autoritat** hi ha alguns fitxers plantilla de configuració. Aquests fitxers s'utilitzaran en les comandes de openssl i indiquen les opcions que ha d'utilitzar openssl per crear els certificats, com per exemple: la ruta dels directoris i fitxers que ha d'utilitzar, les polítiques de validació de dades o les dades per defecte per els certificats.

Si es mira [2] explica per sobre que significa cada apartat.

Polítiques

Dins del fitxer de configuració podem definir diverses polítiques. Aquestes polítiques faran un seguit de comprovacions abans de signar un certificat, si no compleixen, no el signaran.

Per el certificat arrel s'utilitzara la politica estricte:

```
1 [ policy_strict ]
2 # The root CA should only sign intermediate certificates that match.
3 # See the POLICY FORMAT section of `man ca`.
4 countryName             = match
5 stateOrProvinceName     = match
6 organizationName        = match
7 organizationalUnitName  = optional
8 commonName               = supplied
9 emailAddress             = optional
```

Aquestes son les dades "Distinguished Name" necessaries per poder crear el certificat. Tenim 3 nivells:

- Match: Si es posa match en un atribut, aquest haurà de coincidir amb el que tingui el certificat arrel (o certificat amb el que es singi). Per exemple si a countryName del certificat arrel tenim "ES", només s'acceptaran peticions que tinguin "ES".

- supplied: Aquest camp pot tenir qualsevol valor però es requereix. Si la petició no el porta no es podrà signar el certificat.
- optional: Aquest camp es pot informar o no. No interfereix en la creació del certificat.

Creació de les claus

Fem una còpia de la plantilla al directori on tinguem l'arrel de la CA, en el meu cas **autoritat/arrel**, i modifiquem les dades convenients.

Un cop ja tenim tots els fitxers necessaris podem crear les claus privada i pública.

Per l'arrel i els certificats intermedis s'utilitzaran claus de 4096 bits i una contrasenya aes256 que ens demanarà cada cop que vulguem signar un certificat nou. De totes maneres podem signar certificats amb claus menors per tant no hi ha problema.

```
1 openssl genrsa -aes256 -out private/ca.key.pem 4096
2 chmod 400 private/ca.key.pem
```

```
1 Generating RSA private key, 4096 bit long modulus (2 primes)
2 .....++++
3 .....++++
4 e is 65537 (0x010001)
5 Enter pass phrase for private/ca.key.pem:
6 Verifying - Enter pass phrase for private/ca.key.pem:
```

Si es vol utilitzar aquesta clau, el password es "patates"

Finalment només s'ha de crear el certificat a partir de la clau pública. Quan s'utilitza el parametre "req" de Openssl es important indicar el fitxer de configuració -config, sino s'utilitzara el que te per defecte.

```
1 openssl req -config openssl.cnf -key private/ca.key.pem -new -x509 -
  days 7300 -sha256 -extensions v3_ca -out certs/ca.cert.pem
2
3 Enter pass phrase for private/ca.key.pem:
4 You are about to be asked to enter information that will be
  incorporated
5 into your certificate request.
6 What you are about to enter is what is called a Distinguished Name or a
  DN.
7 There are quite a few fields but you can leave some blank
8 For some fields there will be a default value,
9 If you enter '.', the field will be left blank.
10 -----
11 Country Name (2 letter code) [GB]:ES
12 State or Province Name [England]:Catalunya
13 Locality Name []:Girona
```



```
14 Organization Name [Alice Ltd]:SPD-UDG
15 Organizational Unit Name []:SPD
16 Common Name []:SPD Treball CA Arrel
17 Email Address []:
18
19 chmod 444 certs/ca.cert.pem
```

Aquesta comanda ens demanarà la contrasenya de la clau privada i després el distinguished name en el format del **rfc1779**.

En aquest cas s'utilitza sha256 com a algoritme de hash per signar els nous certificats però podem canviar-lo si volem, això dependrà de per quin entorn el volem utilitzar i quins algoritmes són acceptats en els nostres dispositius i aplicacions.

Com es pot veure també s'especifica el format del certificat **x509**. Aquest format és estàndard internacional per PKI. Podem veure l'especificació d'aquest format al **rfc5280**, en concret la versió 3 que és la que utilitzem.

Per comprovar que el certificat és correcte i veurem el contingut:

```
1 openssl x509 -noout -text -in certs/ca.cert.pem
2 Certificate:
3     Data:
4         Version: 3 (0x2)
5         Serial Number:
6             3b:c8:c8:9a:bd:3e:c4:57:cc:94:00:06:5d:0e:c7:3d:0b:e3:93:37
7         Signature Algorithm: sha256WithRSAEncryption
8         Issuer: C = ES, ST = Catalunya, L = Girona, O = SPD-UDG, OU =
9             SPD, CN = SPD Treball CA Arrel
10        Validity
11            Not Before: Jan  4 12:05:23 2020 GMT
12            Not After : Dec 30 12:05:23 2039 GMT
13        Subject: C = ES, ST = Catalunya, L = Girona, O = SPD-UDG, OU =
14            SPD, CN = SPD Treball CA Arrel
15        Subject Public Key Info:
16            Public Key Algorithm: rsaEncryption
17            RSA Public-Key: (4096 bit)
18            Modulus:
19                00:d3:18:a5:a7:61:de:4c:47:e4:d3:6d:d2:1b:c5:
20                c7:4a:f9:ed:f8:e8:5a:95:17:28:75:bc:4d:d9:87:
21                4f:eb....
22            Exponent: 65537 (0x10001)
23        X509v3 extensions:
24            X509v3 Subject Key Identifier:
25                3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1D:F9
26                :05:60
27            X509v3 Authority Key Identifier:
28                keyid:3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1
29                D:F9:05:60
```

```

27      X509v3 Basic Constraints: critical
28      CA:TRUE
29      X509v3 Key Usage: critical
30      Digital Signature, Certificate Sign, CRL Sign
31      Signature Algorithm: sha256WithRSAEncryption
32      bc:6d:71:a1:a0:6a:3f:9e:33:fb:e8:7f:f9:6d:2c:e7:59:0d:
33      50:69:49:81:f9:14:fc:8d:e5:55:ca:15:89:39:43:0c:05:f2:
34      cd:....

```

Si ens fixem, el Issuer (El que signa el certificat) i el Subject (Informació del certificat “client”) són el mateix. Com s’ha explicat abans, els certificats arrel són autosignats i contenen la mateixa informació en el Singant i el Client.

Amb aquests passos ja tindriem l’autoritat certificadora arrel. Hauriem de crear autoritats intermedies per tal que els clients no facin peticions directament a l’autoritat arrel. Haurem de crear certificats nous, signats per l’autoritat arrel que ja tenim i crear les cadenes de confiança.

Autoritats intermedies

En aquest cas seguirem aquest esquema de [3] per crear diverses autoritats intermedies i veure com poden confiar entre elles mitjançant els certificats.

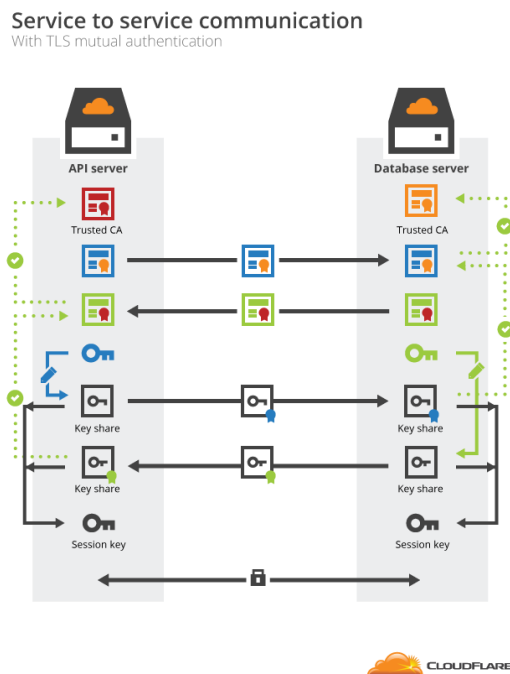


Figure 4: Esquema Confiança - CloudFlare

Estructura de Directoris

Utilitzarem per a cada autoritat un esquema similar al de l'arrel. En aquest cas però s'afagira el directori **csr** on s'hi guardaran els fitxers de petició per crear certificats dels "clients".

Dins de la carpeta **autoritat** hi ha les carpetes API i BDD on es crearan les dues autoritats intermedies.

```
1 # S'ha de fer el mateix pel directori API i per BDD
2
3 mkdir certs crl csr private
4 chmod 700 private
```

Fitxers

Com abans també tindrem els fitxers **index.txt** i **serial** per la gestió dels certificats que signin les autoritats intermitges. A més crearem un numero de serie per els certificats revocats (en l'autoritat arrel no s'ha fet, pero si haguessim de revocar algun certificat intermig també caldria).

```
1 # Aquesta part també s'ha de fer als directoris API i BDD
2 touch index.txt
3 echo 1000 > serial
4 echo 1000 > crlnumber
```

També caldrà un fitxer de configuració per les autoritats intermedies. Al directori **autoritat** hi ha una plantilla per els fitxers intermedis. Només s'ha de copiar i canviar les dades necessaries, com les dades per defecte o el directori on es troven tots els fitxers necessaris.

Creació dels certificats i les cadenes de confiança

Fare aquest pas amb la autoritat API. Per la BDD s'ha de fer el mateix amb els fitxers de BDD.

El primer pas és crear noves claus privades i públiques per l'autoritat intermedia API. Com abans creem un parell de claus xifrat amb una clau aes256.

```
1 openssl genrsa -aes256 -out API/private/api.key.pem 4096
2 Generating RSA private key, 4096 bit long modulus (2 primes)
3 .....+++++
4 .....
5 e is 65537 (0x010001)
6 Enter pass phrase for API/private/api.key.pem:
7 Verifying - Enter pass phrase for API/private/api.key.pem:
8
9 chmod 400 API/private/api.key.pem
```

Les contresenyyes seran:

- API: apipatata
- BDD: bddpatata

Ara tocara generar el certificat a partir de la clau pública de API. Aquest cop però no ho farem com abans ja que tornarem a aconseguir un certificat autosignat, i no tindriem la jerarquia ni la cadena de confiança que volíem.

Per obtenir el certificat aquest cop serà necessari crear un CSR (Certificate Signing Request). Aquest CSR contindrà la informació del sol·licitant (Distinguished Name) juntament amb la clau pública del sol·licitant. El format que utilitza el CSR és PKCS#10 i el podem veure al **rfc2986**.

```
1 openssl req -config API/openssl.cnf -new -sha256 -key API/private/api.  
key.pem -out API/csr/api.csr.pem  
2 Enter pass phrase for API/private/api.key.pem:  
3 You are about to be asked to enter information that will be  
incorporated  
4 into your certificate request.  
5 What you are about to enter is what is called a Distinguished Name or a  
DN.  
6 There are quite a few fields but you can leave some blank  
7 For some fields there will be a default value,  
8 If you enter '.', the field will be left blank.  
9 -----  
10 Country Name (2 letter code) [GB]:ES  
11 State or Province Name [England]:Catalunya  
12 Locality Name []:Girona  
13 Organization Name [Alice Ltd]:SPD-UDG  
14 Organizational Unit Name []:SPD-API  
15 Common Name []:SPD Treball CA API  
16 Email Address []:
```

Alhora de crear el CSR hem d'anar en compte en posar les mateixes dades que al certificat arrel en aquells camps on hi hagues la opció "match" a la privacitat.

El common name serveix per distingir univocament la identitat de cada certificat per tant ha de ser diferent al d'abans. Normalment podem posar-hi noms DNS.

El següent pas serà que l'autoritat arrel signi la petició CSR de API. Aquí s'ha d'anar en compte, com que el que signa es l'arrel, hem de posar el fitxer de configuració de l'arrel i no el de API.

```
1 openssl ca -config arrel/openssl.cnf -extensions v3_intermediate_ca -  
days 3650 -notext -md sha256 -in API/csr/api.csr.pem -out API/certs/  
api.cert.pem  
2  
3 Using configuration from arrel/openssl.cnf  
4 Enter pass phrase for /home/francesc/repos/spd-autoritat-certificadora/  
autoritat/arrel/private/ca.key.pem:
```

```
5 Can't open /home/francesc/repos/spd-autoritat-certificadora/autoritat/
  arrel/index.txt.attr for reading, No such file or directory
6 139794191585728:error:02001002:system library:fopen:No such file or
  directory:../crypto/bio/bss_file.c:72:fopen('/home/francesc/repos/
  spd-autoritat-certificadora/autoritat/arrel/index.txt.attr','r')
7 139794191585728:error:2006D080:BIIO routines:BIIO_new_file:no such file
  :../crypto/bio/bss_file.c:79:
8 Check that the request matches the signature
9 Signature ok
10 Certificate Details:
11     Serial Number: 4096 (0x1000)
12     Validity
13         Not Before: Jan  4 14:38:09 2020 GMT
14         Not After : Jan  1 14:38:09 2030 GMT
15     Subject:
16         countryName           = ES
17         stateOrProvinceName   = Catalunya
18         organizationName      = SPD-UDG
19         organizationalUnitName = SPD-API
20         commonName            = SPD Treball CA API
21     X509v3 extensions:
22         X509v3 Subject Key Identifier:
23             DD:35:35:0E:F1:5D:D0:1F:5A:2A:68:77:24:CC:87:0A:AB:3D:
             E5:80
24         X509v3 Authority Key Identifier:
25             keyid:3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1
             D:F9:05:60
26
27         X509v3 Basic Constraints: critical
28             CA:TRUE, pathlen:0
29         X509v3 Key Usage: critical
30             Digital Signature, Certificate Sign, CRL Sign
31 Certificate is to be certified until Jan  1 14:38:09 2030 GMT (3650
  days)
32 Sign the certificate? [y/n]:y
33
34
35 1 out of 1 certificate requests certified, commit? [y/n]y
36 Write out database with 1 new entries
37 Data Base Updated
```

Ens demana la clau per utilitzar la clau privada de l'arrel i genera el certificat. Ens demana si el volem signar i finalment veiem com l'afegeix a la base de dades.

En el proces li faltaven alguns fitxers que com no existien, els ha creat.

Si ara mirem el fitxer **arrel/index.txt** veurem que ha creat un “registre” de tipus “V” (valid) per el certificat nou que s’ha creat.

```
1 cat arrel/index.txt
2
3 V    300101143809Z          1000    unknown /C=ES/ST=Catalunya/O=SPD-UDG/OU
    =SPD-API/CN=SPD Treball CA API
```

Per comprovar el certificat podem fer la mateixa comanda d’abans:

```
1 openssl x509 -noout -text -in API/certs/api.cert.pem
2 Certificate:
3     Data:
4         Version: 3 (0x2)
5         Serial Number: 4096 (0x1000)
6         Signature Algorithm: sha256WithRSAEncryption
7         Issuer: C = ES, ST = Catalunya, L = Girona, O = SPD-UDG, OU =
            SPD, CN = SPD Treball CA Arrel
8         Validity
9             Not Before: Jan  4 14:38:09 2020 GMT
10            Not After : Jan  1 14:38:09 2030 GMT
11         Subject: C = ES, ST = Catalunya, O = SPD-UDG, OU = SPD-API, CN
            = SPD Treball CA API
12         Subject Public Key Info:
13             Public Key Algorithm: rsaEncryption
14             RSA Public-Key: (4096 bit)
15             Modulus:
16                 00:ad:33:bd:ca:13:da:f8:97:2e:e2:c1:5b:3f:20:
17                 c2:56:1e:99:3e:4c:34:04:80:47:03:9c:8f:a7:db:
18                 5f:....
19             Exponent: 65537 (0x10001)
20         X509v3 extensions:
21             X509v3 Subject Key Identifier:
22                 DD:35:35:0E:F1:5D:D0:1F:5A:2A:68:77:24:CC:87:0A:AB:3D:
                E5:80
23             X509v3 Authority Key Identifier:
24                 keyid:3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1
                D:F9:05:60
25
26             X509v3 Basic Constraints: critical
27                 CA:TRUE, pathlen:0
28             X509v3 Key Usage: critical
29                 Digital Signature, Certificate Sign, CRL Sign
30         Signature Algorithm: sha256WithRSAEncryption
31             2b:e1:70:80:95:ef:20:d1:6e:79:9e:9b:9a:7c:a0:b7:42:1f:
32             8c:e0:13:b6:04:19:09:a2:e7:83:0b:fa:51:3b:d0:2f:26:0d:
33             1d:....
```

Ara podem veure que el Subject (client del certificat) i el Issuer (Signant del certificat → CA Arrel) són diferents.

Mitjançant el certificat arrel podem verificar que el nou certificat per l'autoritat intermitja és vàlid.

```
1 openssl verify -CAfile arrel/certs/ca.cert.pem API/certs/api.cert.pem
2
3 API/certs/api.cert.pem: OK
```

Ara el problema que hi ha és que només amb el certificat de API no n'hi ha prou perquè, per exemple, els navegadors confiïn en el certificat, tot i està signat per una autoritat certificadora superior.

El que s'ha de fer és donar-li el certificat arrel a més del que estem utilitzant a l'aplicació que necessiti verificar el certificats.

Una altre opció es crear una cadena de confiança entre l'autoirat arrel i l'autoritat API.

```
1 cat API/certs/api.cert.pem arrel/certs/ca.cert.pem > API/certs/ca-api-
  chain.cert.pem
2 chmod 444 API/certs/ca-api-chain.cert.pem
```

Les 2 opcions son vàlides. Si decidim “instal·lar” el certificat de l'autoritat arrel, el fitxer de cadena només ha de contenir el certificat de l'autoritat API.

Confiança entre serveis

Seguint l'estructura de la figura 4:

Ara tenim dues autoritats intermitges. Cada una servirà certificats per els seu conjunt de serveis. L'autoritat API servirà certificats les diferents aplicacions que tinguem a la nostre xarxa. L'autoritat BDD servirà els certificats als servidors de bases de dades que tinguem.

Que s'ha de fer:

En els servidors/aplicacions que siguin del conjunt de les API (que tinguin certificats signats per API CA) s'instal·larà el certificat de la BDD CA (amb el root o amb la cadena de confiança).

D'altra banda en els servidors que siguin del conjunt de BDD (que tinguin certificats signats per BDD CA) s'instal·larà el certificat de API CA.

D'aquesta manera, les aplicacions/servidors API només confiaran en BDD CA, es a dir que només acceptaran certificats que s'hagin estat signats per BDD CA. Amb les aplicacions de BDD pasará el mateix pero amb la signatura de API CA.

Així doncs quan s'estableixi una connexió entre un aplicació API i una aplicació BDD, totes dues aplicacions veuren que el certificat de l'altre aplicació és valid i de confiança.

Exemple de confiança

En aquest apartat simularé 2 aplicacions, una per cada CA intermitja, amb directoris diferents.

Es generarà un certificat per a cada aplicació a partir de les CA intermitges. Posteriorment “s’instal·larà a cada aplicació el certificat de l’autoritat de l’altre part i es verificaran els certificats.

Com abans s’ha de generar un parell de claus, privada i pública, per a cada aplicació.

Al directori **exemple** hi haurà els directoris de les aplicacions api1 i bdd1. Cada aplicació tindrà la seva clau privada al subdirector clau, i els certificats que necessiti al subdirector certs.

```
1 openssl genrsa -out api1/clau/api1.key.pem 2048
2 Generating RSA private key, 2048 bit long modulus (2 primes)
3 .....+++++
4 .....+++++
5 e is 65537 (0x010001)
6
7 chmod 400 api1/clau/api1.key.pem
8
9 openssl genrsa -out bdd1/clau/bdd1.key.pem 2048
10 Generating RSA private key, 2048 bit long modulus (2 primes)
11 .....+++++
12 .....+++++
13 e is 65537 (0x010001)
14
15 chmod 400 bdd1/clau/bdd1.key.pem
```

Ara per a cada un es creen els CSR amb la Autoritat corresponent i es signen. Ho mostro només per api1, per bdd1 és el mateix amb la seva autoritat.

```
1 openssl req -config ../autoritat/API/openssl.cnf -key api1/clau/api1.
   key.pem -new -sha256 -out ../autoritat/API/csr/api1.csr.pem
2
3 You are about to be asked to enter information that will be
   incorporated
4 into your certificate request.
5 What you are about to enter is what is called a Distinguished Name or a
   DN.
6 There are quite a few fields but you can leave some blank
7 For some fields there will be a default value,
8 If you enter '.', the field will be left blank.
9 -----
10 Country Name (2 letter code) [GB]:ES
11 State or Province Name [England]:Catalunya
12 Locality Name []:Girona
13 Organization Name [Alice Ltd]:SPD-Api1
14 Organizational Unit Name []:
15 Common Name []:SPD Treball api1
16 Email Address []:
```



```
17
18 openssl ca -config ../autoritat/API/openssl.cnf -extensions server_cert
    -days 375 -notext -md sha256 -in ../autoritat/API/csr/api1.csr.pem
    -out ../autoritat/API/certs/api1.cert.pem
19
20 # La CA (RA) ens hauria de retornar el certificat, Faig un cp per
    simular-ho
21 cp ../autoritat/API/certs/api1.cert.pem api1/certs/
```

Un cop tenim els certificats de api1 i bdd1 necessitem que les aplicacions confiïn en “l’altre” autoritat certificadora. Al directori certs de api1 hi posarem la cadena de confiança de BDD i al directori “certs” de bdd1 hi posarem la cadena de confiança de API.

```
1 cp ../autoritat/API/certs/ca-api-chain.cert.pem bdd1/certs/
2 cp ../autoritat/BDD/certs/ca-bdd-chain.cert.pem api1/certs/
```

Simulem una connexió entre els dos serveis amb un subdirectori c’onnexio’ a cada una de les aplicacions. En aquest directori “s’hauran guardat els certificats intercanviats”.

```
1 exemple/
2   api1/
3     certs/
4       api1.cert.pem
5       ca-bdd-chain.cert.pem
6     clau/
7       api1.key.pem
8     connexio/
9       bdd1.cert.pem
10  bdd1/
11    certs/
12      bdd1.cert.pem
13      ca-api-chain.cert.pem
14    clau/
15      bdd1.key.pem
16    connexio/
17      api1.cert.pem
```

Ara només ens queda verificar que el certificat és de confiança mitjançant les cadenes de confiança que hem instal·lat a cada aplicació. Obviament només comprovant les signatures no en fem prou però serveix per il·lustrar com s’hauria de fer.

```
1 openssl verify -CAfile exemple/api1/certs/ca-bdd-chain.cert.pem exemple
  /api1/connexio/bdd1.cert.pem
2 exemple/api1/connexio/bdd1.cert.pem: OK
3
4 openssl verify -CAfile exemple/bdd1/certs/ca-api-chain.cert.pem exemple
  /bdd1/connexio/api1.cert.pem
5 exemple/bdd1/connexio/api1.cert.pem: OK
```

Revocació de certificats

Es possible que, tot i intentar assegurar les claus dels certificats, aquestes siguin compromeses. També pot ser que un certificat ja no sigui útil, per qualsevol motiu. En els dos casos el que s'ha de fer és demanar a l'autoritat (la que ens ha signat el certificat) que revoqui els certificats corresponents.

Quan es fa la validació d'un certificat s'hauria de comprovar que aquest no estigui caducat o que s'hagi revocat abans de la data d'expiració. Si passa algun d'aquests dos casos no hauriem de confiar en el certificat.

Les autoritats de certificació, d'altra banda, han d'oferir algun servei per veure si un certificat ha estat revocat.

OpenSSL permet treballar amb llistes de certificats revocats (CRL) i també amb Online Certificate Status Protocol (OCSP).

CRL

En el cas de les llistes de revocació cada cop que es revoqui un certificat s'haurà de modificar un fitxer que contindrà tots els certificats revocats. Aquest fitxer haurà d'estar exposat en un lloc públic.

Cada autoritat certificadora intermitja haurà de publicar la seva ruta al seu CRL. Caldrà modificar el fitxer de configuració de les autoritats intermitges per tal que tothom que tingui un certificat expedit per aquella autoritat sàpiga on anar a buscar la llista CRL.

En l'apartat [server_cert] dels fitxers de configuració hem d'afegir una línia com aquesta:

```
1  crlDistributionPoints = URI:http://api.exemple/api.crl.pem
```

Per generar el fitxer de cada autoritat:

```
1  openssl ca -config autoritat/API/openssl.cnf -gencrl -out autoritat/API
   /crl/api.crl.pem
2
3  openssl ca -config autoritat/BDD/openssl.cnf -gencrl -out autoritat/BDD
   /crl/bdd.crl.pem
```

Per revocar utilitzarem el certificat de bdd1 que s'ha utilitzat en l'exemple anterior. Per fer-ho hauriem de passar el certificat a l'autoritat i demanar-li (via RA) que el revokes. Jo ja tinc una còpia del certificat a BDD, ho faré directament amb aquesta.

```
1  openssl ca -config autoritat/BDD/openssl.cnf -revoke autoritat/BDD/
   certs/bdd1.cert.pem
2
3  Using configuration from autoritat/BDD/openssl.cnf
```

```
4 Enter pass phrase for /home/francesc/repos/spd-autoritat-certificadora/
  autoritat/BDD/private/bdd.key.pem:
5 Revoking Certificate 1000.
6 Data Base Updated
7
8 # Si s'observa el fitxer 'base de dades' index.txt de l'autoritat BDD
  es pot veure que ara el certificat apareix com a revocat 'R'
9
10 cat autoritat/BDD/index.txt
11 R    210113162319Z    200105111503Z    1000    unknown /C=ES/ST=Catalunya/
    L=Girona/O=SPD-Bdd1/CN=SPD Treball bdd1
```

Per que els canvis tinguin es vegin reflectits s'ha de tornar a fer la comanda que genera el fitxer de certificats revocats. Després podem veure el contingut del fitxer i veurem que, efectivament, s'ha afegit el certificat que s'ha revocat.

```
1 openssl ca -config autoritat/BDD/openssl.cnf -gencrl -out autoritat/BDD
  /crl/bdd.crl.pem
2
3 openssl crl -in autoritat/BDD/crl/bdd.crl.pem -noout -text
4
5 Certificate Revocation List (CRL):
6     Version 2 (0x1)
7     Signature Algorithm: sha256WithRSAEncryption
8     Issuer: C = ES, ST = Catalunya, O = SPD-UDG, OU = SPD-BDD, CN =
        SPD Treball CA BDD
9     Last Update: Jan  5 11:22:24 2020 GMT
10    Next Update: Feb  4 11:22:24 2020 GMT
11    CRL extensions:
12        X509v3 Authority Key Identifier:
13            keyid:51:7E:A8:70:8B:9E:BB:2F:10:93:AE:29:03:8E:C5:20:
            D3:0A:B6:FB
14
15        X509v3 CRL Number:
16            4097
17    Revoked Certificates:
18        Serial Number: 1000
19        Revocation Date: Jan  5 11:15:03 2020 GMT
20        Signature Algorithm: sha256WithRSAEncryption
21            55:f6:87:b2:3a:ec:cf:4e:1e:ed:58:d4:ed:44:2e:b2:53:a7:
22            c7:....
```

OCSP

OCSP és similar fa una funció similar a CRL. En comptes d'anar a buscar un fitxer de que conté tots els llistats revocats, el que es fa és habilitar una crida web per un certificat en concret.

Molts navegadors utilitzen aquest mètode i ja no fan servir les CRL.

Com abans primer s'ha de modificar el fitxer de configuració de l'autoritat. S'ha de posar la línia que indica on es pot consultar el OCSP. En el meu cas indicaré directament localhost.

```
1 [server-cert]
2 ....
3 authorityInfoAccess = OCSP;URI:http://localhost
```

Per respondre les peticions OCSP necessitem un certificat. Per tant crearem tant el parell de claus com el certificat signat per l'autoritat intermitja.

```
1 openssl genrsa -aes256 -out autoritat/BDD/private/ocsp.bdd.key.pem 4096
2 openssl genrsa -aes256 -out autoritat/API/private/ocsp.api.key.pem 4096
3
4 openssl req -config autoritat/BDD/openssl.cnf -new -sha256 -key
  autoritat/BDD/private/ocsp.bdd.key.pem -out autoritat/BDD/csr/ocsp.
  bdd.csr.pem
5
6 openssl ca -config autoritat/BDD/openssl.cnf -extensions ocsp -days 375
  -notext -md sha256 -in autoritat/BDD/csr/ocsp.bdd.csr.pem -out
  autoritat/BDD/certs/ocsp.bdd.cert.pem
```

Les contrasenyes són:

- API: apio
- BDD: bddo

El OCSP mira directament el fitxer de base de dades index.txt de cada autoritat certificadora. El que es fa és preguntar per un certificat en concret. Si li passem una còpia d'un certificat que ha estat revocat en algun moment, ens contestarà que aquell certificat va ser revocat. De fet ens serveix també per verificar si és correcte, OCSP ens indica l'estatus del certificat.

Utilitzaré el certificat de bdd1 que ja ha estat revocat en l'apartat CRL.

Primer s'ha de tenir un servei que vagi escoltant les peticions ocsp d'una autoritat en concret. Per veure totes les opcions que hi ha **Manual OCSP**.

```
1 #S'indica el fitxer index, el port, el certificat per signar del ocsp i
  el certificat de l'autoritat
2
3 openssl ocsp -index autoritat/BDD/index.txt -port 2560 -rsigner
  autoritat/BDD/certs/ocsp.bdd.cert.pem -rkey autoritat/BDD/private/
  ocsp.bdd.key.pem -CA autoritat/BDD/certs/ca-bdd-chain.cert.pem -text
  -out log.txt
4
5 Enter pass phrase for autoritat/BDD/private/ocsp.bdd.key.pem:
6 ocsp: waiting for OCSP client connections...
```

Des del client que intenta verificar un certificat: (en aquest cas és localhost pero serviria qualsevol url que fos correcte)

```
1 # Des dels fitxers exemple mirem el certificat que hem rebut de bdd1 a
   api1
2
3 openssl ocsp -CAfile exemple/api1/certs/ca-bdd-chain.cert.pem -issuer
   exemple/api1/certs/ca-bdd-chain.cert.pem -cert exemple/api1/connexio
   /bdd1.cert.pem -url http://127.0.0.1:2560 -resp_text -noverify
```

Aquesta comanda farà la petició http i al final rebrem com a resposta, el certificat del OCSP i la resposta signada per aquest:

```
1 OSCP Response Data:
2   OSCP Response Status: successful (0x0)
3   Response Type: Basic OSCP Response
4   Version: 1 (0x0)
5   Responder Id: C = ES, ST = Catalunya, L = Girona, O = SPD-BDD, OU =
   OSCP-BDD, CN = SPD treball OSCP BDD
6   Produced At: Jan  5 12:12:09 2020 GMT
7   Responses:
8   Certificate ID:
9     Hash Algorithm: sha1
10    Issuer Name Hash: 423837A4CE5A773E1487548465C7C807A5F9E181
11    Issuer Key Hash: 517EA8708B9EBB2F1093AE29038EC520D30AB6FB
12    Serial Number: 1000
13    Cert Status: revoked
14    Revocation Time: Jan  5 11:15:03 2020 GMT
15    This Update: Jan  5 12:12:09 2020 GMT
16
17 .....
18 -----END CERTIFICATE-----
19 exemple/api1/connexio/bdd1.cert.pem: revoked
20   This Update: Jan  5 12:12:09 2020 GMT
21   Revocation Time: Jan  5 11:15:03 2020 GMT
```

Referències

- [1] Que és una CA - https://es.wikipedia.org/wiki/Autoridad_de_certificaci%C3%B3n
- [2] Crear una CA - <https://jamielinux.com/docs/openssl-certificate-authority/index.html>
- [3] Crear un PKI - <https://blog.cloudflare.com/how-to-build-your-own-public-key-infrastructure/>