Contents

Objectiu

L'objectiu del treball és mostrar quins son els elements que composen 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 garantitzar la seguretat de les comunicacions digitals via TLS-HTTPS. S'utilitiza criptograrfia de clau pública per generar les claus i els certificats.

La CA dona el servei de certificació, que garantitza 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.

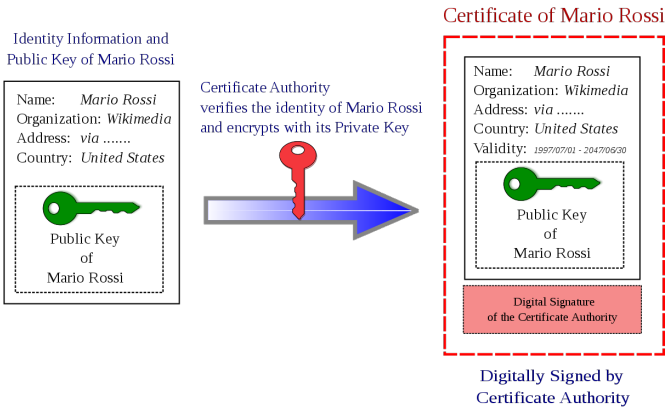

Figure 1: Certificat signat CA - Wikipedia

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 interficie entre l'usuari i l'autoritat de certificació. Ha d'identificar el solicitants o titulars dels certificats.
- VA: Autoritat de validació emmagztzema els certificats digitals i aministra l allista de certificats caducats o revocats. També posa a disposició tots els certificats emesos per l'autoritat de certificació.
- Autoritat de custodia: Emmagatzema de forma segura les claus de xifrat que utilitza l'autoritat certificadora.

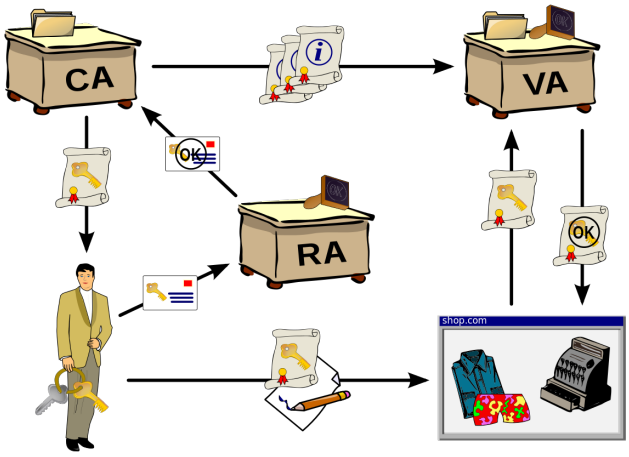

Figure 2: Esquema PKI - Wikipedia

Confiança

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

Els certificats autosignats es poden crear facilment. Es tracten de certificats els cuals el propietari i el signatari son el mateix. En alguns casos aquets certificats son suficients per determinades accions. Moltes aplicacions pero, 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 serie de CA, que suposadament, compleixen tots els requeriments per actuar com a CA. Aquesta confiança fa que si es troven un lloc web amb un certificat signat per alguna d'aquestes entitats, validin el certificat i deixa navegar sense problemes. Si trova un certificat autosignat o signat per una CA que no te com a autoritat de confiança, no ens deixerà navegar.

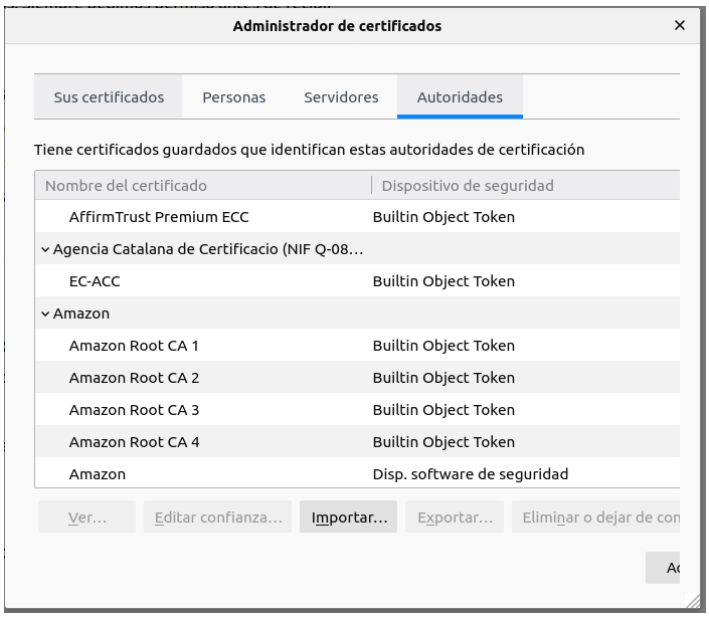

Figure 3: Autoritats confiança firefox

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

Jerarquia d'una CA

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

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. Això forma una cadena de confiança

El que se sol fer és que el cerfificat arrel només firma els cerfiticats 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 pugi signar. Per tant en aquest cas s'ha "confiar" que el certificat arrel sigui "correcte".

Construcció d'una CA amb OpenSSL

Podem trobar diversos proveidors de certificats a Internet. Molts d'aquests proveidors ja els tenen les aplicacions com a autoritats de confiança. Podriem utilitzar algun d'aquests proveidors per obtenir certificats, però en alguns casos potser és més simple crear els nostres propis certifcats, 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 centrarè en l'apartat de la CA dins del PKI. No donarè una interficie RA i farè un VA molt simple utilitzant directoris i fitxers de linux.

Per aquesta part seguirè 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'utilizaran.

- 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 guardaràn 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 creein. Tindran informació del numero de serie 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 numero de serie. En aquest fitxer s'hi guarda el valor "autonumeric", per a cada nou certificat s'incrementa el valor.

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

Fitxer de configuració OpenSSL

Al directori **autoritat** hi ha alguns fitxers plantilla de configuració. Aquests fitxers s'ulitlizaran 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 politiques de validació de dades o les dades per defecte per els certificats.

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

Politiques

Dins del fitxer de configuració podem definir diverses politiques. Aquestes politiques 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:

```
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
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'acceparan peticions que tinguin "ES".

- supplied: Aquest camp pot tenir qualsevol valor però es requerit. 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 copia 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 publica.

Per l'arrel i els certificats intermedis s'utilitzaran claus de 4096 bits i una contrasenya aes265 que ens demanarà cada cop que volguem signar un certificat nou. De totes maneres podem singar certificats amb claus menors per tant no hi ha problema.

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

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

```
openssl req -config openssl.cnf -key private/ca.key.pem -new -x509 -
    days 7300 -sha256 -extensions v3_ca -out certs/ca.cert.pem

Enter pass phrase for private/ca.key.pem:
You are about to be asked to enter information that will be
    incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
     DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name [England]:Catalunya
Locality Name []:Girona
```

```
Organization Name [Alice Ltd]:SPD-UDG
Organizational Unit Name []:SPD
Common Name []:SPD Treball CA Arrel
Email Address []:

chmod 444 certs/ca.cert.pem
```

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

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

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

Per comprovar que el certificat es correcte i veuren el contingut:

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

```

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

Si ens fixem, el Issuer (El que signa el certificat) i el Subject(Informació del certificat "client") són el matiex. Com s'ha explicat abans, els certificats arrel son 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.

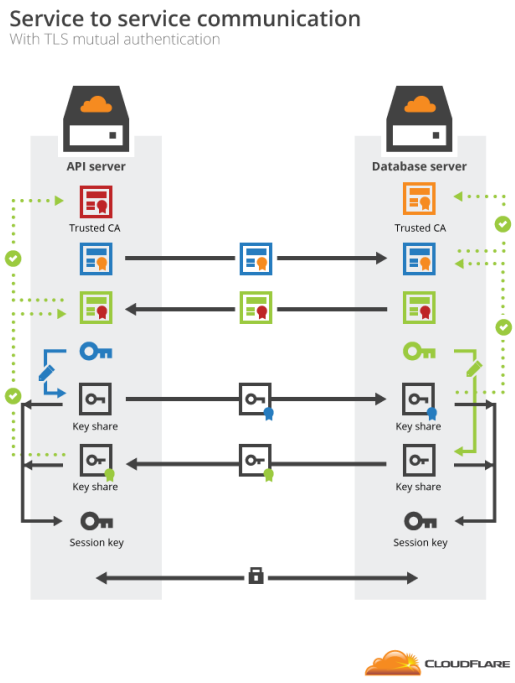

Figure 4: Esquema Confiança - CloudFlare

Estructura de Directoris

Utilitzarem per a cada autoritat un esquema similar al de l'arrel. En aquest cas peró 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.

```
# S'ha de fer el matiex pel directori API i per BDD

mkdir certs crl csr private
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).

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

```
openssl genrsa -aes256 -out API/private/api.key.pem 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................++++
.....................................................................................

e is 65537 (0x010001)
Enter pass phrase for API/private/api.key.pem:
Verifying - Enter pass phrase for API/private/api.key.pem:

chmod 400 API/private/api.key.pem
```

Les contresenyes seran:

- API: apipatata
- BDD: bddpatata

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

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

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

Alhora de crear el CSR hem d'anar en compte en possar 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 seguent 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.

```
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

Using configuration from arrel/openssl.cnf
Enter pass phrase for /home/francesc/repos/spd-autoritat-certificadora/autoritat/arrel/private/ca.key.pem:
```

```
Can't open /home/francesc/repos/spd-autoritat-certificadora/autoritat/arrel/index.txt.attr for reading, No such file or directory
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')
139794191585728:error:2006D080:BIO routines:BIO_new_file:no such file:../crypto/bio/bss_file.c:79:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Jan  4 14:38:09 2020 GMT
            Not After : Jan  1 14:38:09 2030 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Catalunya
            organizationName          = SPD-UDG
            organizationalUnitName    = SPD-API
            commonName                = SPD Treball CA API
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                DD:35:35:0E:F1:5D:D0:1F:5A:2A:68:77:24:CC:87:0A:AB:3D:E5:80
            X509v3 Authority Key Identifier:
                keyid:3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1D:F9:05:60

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Jan  1 14:38:09 2030 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
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 exisiten, 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.

```
cat arrel/index.txt

```

```
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:

```
openssl x509 -noout -text -in API/certs/api.cert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = ES, ST = Catalunya, L = Girona, O = SPD-UDG, OU =
           SPD, CN = SPD Treball CA Arrel
        Validity
            Not Before: Jan  4 14:38:09 2020 GMT
            Not After : Jan  1 14:38:09 2030 GMT
        Subject: C = ES, ST = Catalunya, O = SPD-UDG, OU = SPD-API, CN
           = SPD Treball CA API
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:ad:33:bd:ca:13:da:f8:97:2e:e2:c1:5b:3f:20:
                    c2:56:1e:99:3e:4c:34:04:80:47:03:9c:8f:a7:db:
                    5f:....
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                DD:35:35:0E:F1:5D:D0:1F:5A:2A:68:77:24:CC:87:0A:AB:3D:
                   E5:80
            X509v3 Authority Key Identifier:
                keyid:3B:AE:80:4A:E4:39:9F:3A:B1:02:10:78:5F:56:A0:8E:1
                   D:F9:05:60

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         2b:e1:70:80:95:ef:20:d1:6e:79:9e:9b:9a:7c:a0:b7:42:1f:
         8c:e0:13:b6:04:19:09:a2:e7:83:0b:fa:51:3b:d0:2f:26:0d:
         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.

```
openssl verify -CAfile arrel/certs/ca.cert.pem API/certs/api.cert.pem

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 perque, per exemple, els navegadors confiin 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 pasarà 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 direcoris de les aplicacions api1 i bdd1. Cada aplicació tindrà la seva clau privada al subdirectori claus, i els certificats que necessiti al subdirectoris certs.

```
openssl genrsa -out api1/clau/api1.key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..............................+++++
........................+++++
e is 65537 (0x010001)

chmod 400 api1/clau/api1.key.pem

openssl genrsa -out bdd1/clau/bdd1.key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
....+++++
................+++++
e is 65537 (0x010001)

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.

```
openssl req -config ../autoritat/API/openssl.cnf -key api1/clau/api1.key.pem -new -sha256 -out ../autoritat/API/csr/api1.csr.pem

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:ES
State or Province Name [England]:Catalunya
Locality Name []:Girona
Organization Name [Alice Ltd]:SPD-Api1
Organizational Unit Name []:
Common Name []:SPD Treball api1
Email Address []:

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

# La CA (RA) ens hauria de retornar el certificat, Faig un cp per simular-ho
cp ../autoritat/API/certs/api1.cert.pem api1/certs/
```

Un cop tenim els certificats de api1 i bdd1 necesitem que les aplicacions confiin 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.

```
cp ../autoritat/API/certs/ca-api-chain.cert.pem bdd1/certs/
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".

```
exemple/
    api1
        certs
            api1.cert.pem
            ca-bdd-chain.cert.pem
        clau
            api1.key.pem
        connexio
            bdd1.cert.pem
    bdd1
        certs
            bdd1.cert.pem
            ca-api-chain.cert.pem
        clau
            bdd1.key.pem
        connexio
            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 peró serveix per ilustrar com s'hauria de fer.

```
openssl verify -CAfile exemple/api1/certs/ca-bdd-chain.cert.pem exemple
    /api1/connexio/bdd1.cert.pem
exemple/api1/connexio/bdd1.cert.pem: OK

openssl verify -CAfile exemple/bdd1/certs/ca-api-chain.cert.pem exemple
    /bdd1/connexio/api1.cert.pem
exemple/bdd1/connexio/api1.cert.pem: OK
```

Revocació de certificats

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/