# SSL management

This page documents some things I had to do to manage my SSL[1] certificates and how I did them using OpenSSL[2]

## Contents



## 1 Creating your own CA and certificate

### 1.1 Creating the root certificate

This creates the root certificate that you will use to sign everything else:

```
openssl req -newkey rsa:4096 -sha512 -days 9999 -x509 -nodes -out example_root.cer
```

This also create a `privkey.pem` file, containing your root private key: keep this in a secure place !

### 1.2 Creating the certificate signing request (CSR)

This will be used to sign your certificate:

```
openssl req -newkey rsa:4096 -sha512 -nodes -out example_com.csr \
    -keyout example_com.key
```

[1] `http://en.wikipedia.org/wiki/Transport_Layer_Security`
[2] `http://www.openssl.org/`

### 1.3 Creating a configuration file

Certificate creation in openssl is so complicated that you need a configuration file to indicate all the options. This is the one I used:

```
# Mainly copied from:
# http://swearingscience.com/2009/01/18/openssl-self-signed-ca/

[ ca ]
default_ca = myca

[ crl_ext ]
# issuerAltName=issuer:copy  #this would copy the issuer name to altname
authorityKeyIdentifier=keyid:always

 [ myca ]
 dir = ./
 new_certs_dir = $dir
 unique_subject = no
 certificate = $dir/example_root.cer
 database = $dir/certindex
 private_key = $dir/privkey.pem
 serial = $dir/certserial
 default_days = 9999
 default_md = sha512
 policy = myca_policy
 x509_extensions = myca_extensions
 crlnumber = $dir/crlnumber
 default_crl_days = 9999

 [ myca_policy ]
 commonName = supplied
 stateOrProvinceName = supplied
 countryName = optional
 emailAddress = optional
 organizationName = supplied
 organizationalUnitName = optional

 [ myca_extensions ]
 basicConstraints = CA:false
 subjectKeyIdentifier = hash
 authorityKeyIdentifier = keyid:always
 keyUsage = digitalSignature,keyEncipherment
 extendedKeyUsage = serverAuth
 crlDistributionPoints = URI:http://certs.example.com/example_root.crl
 subjectAltName  = @alt_names

 [alt_names]
```

```
DNS.1 = example.com
DNS.2 = *.example.com
```

The important lines are the DNS.1 and DNS.2 lines: put in all the domains you want the certificate to be valid with (of course, you can add DNS.3, DNS.4 etc... as you like). The star (*) means all subdomains, so `*.example.com` will match `www.example.com` as well as `very.complicated.example.com`

Also pay attention to the `default_md` and `default_days` variables.

Save your configuration files to e.g `example_root.conf`.

### 1.4 Creating the index files

Once you have created the configuration file, you should create an empty index file, and a serial number file for the certificate index and the revocation list:

```
touch certindex
echo 000a > certserial
echo 000a > crlnumber
```

### 1.5 Generating the certificate and the CRL

Finally, generate your certificate:

```
openssl ca -batch -config example_root.conf -notext -in example.com.csr \
    -out example.com.cer
```

And generate your (empty) certificate revocation list:

```
openssl ca -config example_root.conf -gencrl -keyfile privkey.pem \
    -cert example_root.cer -out example_root.crl.pem
 openssl crl -inform PEM -in example_root.crl.pem -outform DER -out \
    example_root.crl && rm example_root.crl.pem
```

The last line is necessary because RFC 5280[3] requires the CRL to be encoded using `DER`.

### 1.6 Checking the content of your certificate

Use this command to see that the content of your certificate is what you expect:

```
openssl x509 -text -noout < example_com.cer
```

To check the fingerprint, use:

```
openssl x509 -fingerprint -sha1 -noout < example_com.cer
```

### 1.7 Revoking a certificate

To revoke a bad certificate (here `example.com.cer`), update your index using:

```
openssl ca -config example_root.conf -revoke example_com.cer \
    -keyfile privkey.pem -cert example_root.cer
```

Then re-generate the revocation list using the command mentionned above.

[3] `http://www.ietf.org/rfc/rfc5280.txt`

## 2 References

- To create my own CA and use it to sign my own certificate, I followed this self signed CA howto[4].
- To make the certificate over multiple domains, I used the SAN/UCC certificate generation howto[5].
- To manage the certificates, I refered to this certificate management howto[6]

[4] http://langui.sh/2009/01/18/openssl-self-signed-ca/
[5] http://langui.sh/2009/02/28/openssl-sanucc-certificate-generation/
[6] http://gagravarr.org/writing/openssl-certs/ca.shtml