Exercise: Make your own PKI

Become a certificate authority in minutes

Today certificates are widely used to verify, authenticate a client/user or server, to encrypt or sign emails or to sign other types of objects (e.g. source code). But what is the technical job of a certification authority?
In this exercise you'll learn how to create your own Root Certificate Authority (CA) and create certificates for a web server using the cryptography and TLS toolkit OpenSSL (https://www.openssl.org/).

Install OpenSSL on Windows

1. Download OpenSSL: There are plenty of ways to install OpenSSL on Windows. The easiest is to use an installer from one of the OpenSSL trusted parties on *https://wiki.openssl.org/index.php/Binaries*. I propose *https://slproweb.com/products/Win32OpenSSL.html* . The light version for your Windows is just fine for the exercise.
2. Execute the installer with default settings if applicable
3. The binary executable should be then available at
   `C:\Program Files\OpenSSL-Win64\bin`
4. To test the installation open a command prompt type
5. `C:\Program Files\OpenSSL-Win64\bin\openssl version`
6. (Optional) To not type the whole path all the time you can add „C:\Program Files\OpenSSL-Win64\bin" to you Windows path (see *https://docs.microsoft.com/en-us/previous-versions/office/developer/sharepoint-2010/ee537574(v=office.14*). Furhter steps assuming the openssl command to be in your PATH.

Unzip configuration & and management files

Unzip the config.zip file in the directory where you want to create the certificates. The three different files are:

- openssl.cnf: As the configuration file for openssl of where to find relevant parameters and source for certain actions
- db.txt: A small text file acting as a simple database to keep track of the certificate authority signings
- serial: A small text file to maintain a serial number for each signing activity

Store the three files in the directory where you want to create the PKI.

Create a self signed root certificate:

First of all we have to create a key for the root certificate authority. The following commands will create a file with random noise (8291 Bytes) and a 2048 bit RSA key which is encrypted with AES 256 Bit:

1. Create the random noise and put it into a file „.randRootCA"

   ```
   openssl rand -out .randRootCA 8192
   ```

2. Create the RSA key-pair and encrypt it with AES 256

   ```
   openssl genrsa -out rootca.key -aes256 -rand
   .randRootCA 2048
   ```

3. Create the self-signed root certificate in X.509 format using the create RSA key-pair's private key

```
openssl req -new -x509 -days 3650 -key rootca.key -
out rootca.crt
```

4. Enter the requested information in the wizard
5. You can view the content of the certificate

```
openssl x509 -text -in rootca.crt
```

Create a sub ca certificate (signed by the root ca):

The next step is to create a certificate signing request (CSR) for the Sub CA. To create the CSR we have to generate a key first. The generation of the key is equal to the key generation of the Root CA.

1. Again create some noise

```
openssl rand -out .randSubCA 8192
```

2. Create the RSA kay for the sub certification authority

```
openssl genrsa -out subca.key -aes256 -rand
.randSubCA 2048
```

3. Now create the certificate signing request

```
openssl req -new -key subca.key -out subca.csr
```

4. Now sign the sub certification authority with the root certification authority using the signing request. Set the configuration parameters as you find them appropriate

```
openssl ca -name CA_RootCA -in subca.csr -out
subca.crt -extensions subca_cert -config
openssl.cnf
```

Create a server certificate (singed by the sub ca):

To create a server certificate, e.g. for a webserver, you have to create a key and a certificate signing request. So the first step to create the key is equal to the key creation of the Root or Sub CA.

1. Again create some noise for the key

```
openssl rand -out .randServer 8192
```

2. Again, create another RSA key-pair and encrypt it.

```
3. openssl genrsa -out server.key -aes256 -rand
.randServer 2048
```

4. Now you create a certificate signing request (CSR) for the server certificate. The common name will be the domain name of your website. For example *www.nyname.com*.

```
openssl req -new -key server.key -out server.csr
```

5. Now you can sign the request and create the certificate for you webserver. Because some application also use the subject alternative name for verification it makes sense to define the field subjectAltName as: „DNS:www.myname.com“

```
openssl ca -name CA_SubCA -in server.csr -out
server.crt -config openssl.cnf
```

Congratulations you've now done the technical job of two certification authorities and created a TLS certificate for your webserver. No-one should trust this certificate as no-one really has verified that it belongs to you, but that's another story.

Disclaimer