TLS security for straton

straton user guide – Rev. 5

sales@straton-plc.com

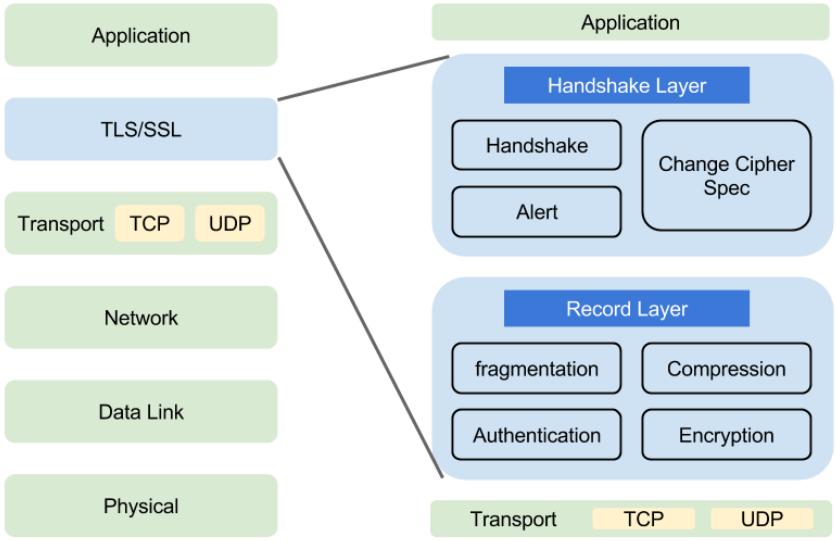
CP_R31_006_H

straton

Content

1. Introduction to SSL/TLS

SSL and TLS are data security protocols. They behave as an additional intermediate layer between the Transport layer (TCP) and the application layer (HTTP, FTP, SMTP, etc.) (see diagram).



This therefore means that they can be used to secure a web transaction as well as to send or receive email, or in our case, secure the communication between two devices (Master-Slave).

SSL (Secure Socket Layer) is an old protocol deprecated in favor of TLS (Transport Layer Security). TLS Is a protocol for the secure transmission of data based on SSLv3. It offers confidentiality, integrity, and authentication.

- Confidentiality: hides the content of the messages, it is impossible to spy on the information exchanged. The Client and the Server need to be assured that their conversation cannot be overheard by a third party. This functionality is provided by an encryption algorithm.
- Integrity: detects when the messages have been tampered with, it is impossible to fake the information exchanged. The Client and the Server must be able to ensure that the transmitted messages are neither truncated nor modified (integrity), that they arrive indeed from the sender. These functionalities are ensured by the signing of the data.
- Authentication: ensures that whoever is sending them is who he says he is. This point ensures the identity of the program, person or company with which we are communicating.

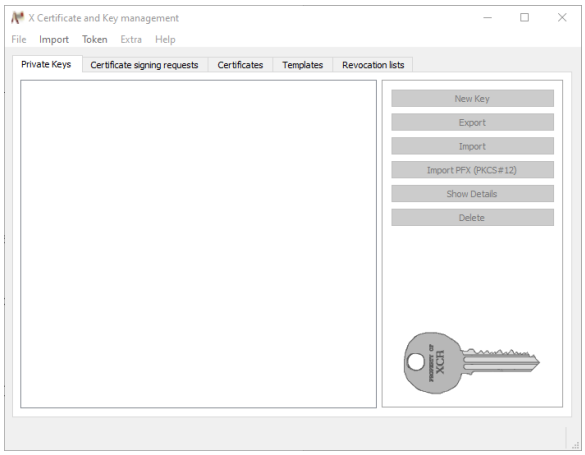
In the rest of this document, we will see how to configure the TLS in straton, using XCA Software to create the different files needed for our use case (CA, certificate, key…)

2. Use XCA Software for DNP3, IEC60870 and IEC61850 (ENERGY) certificates

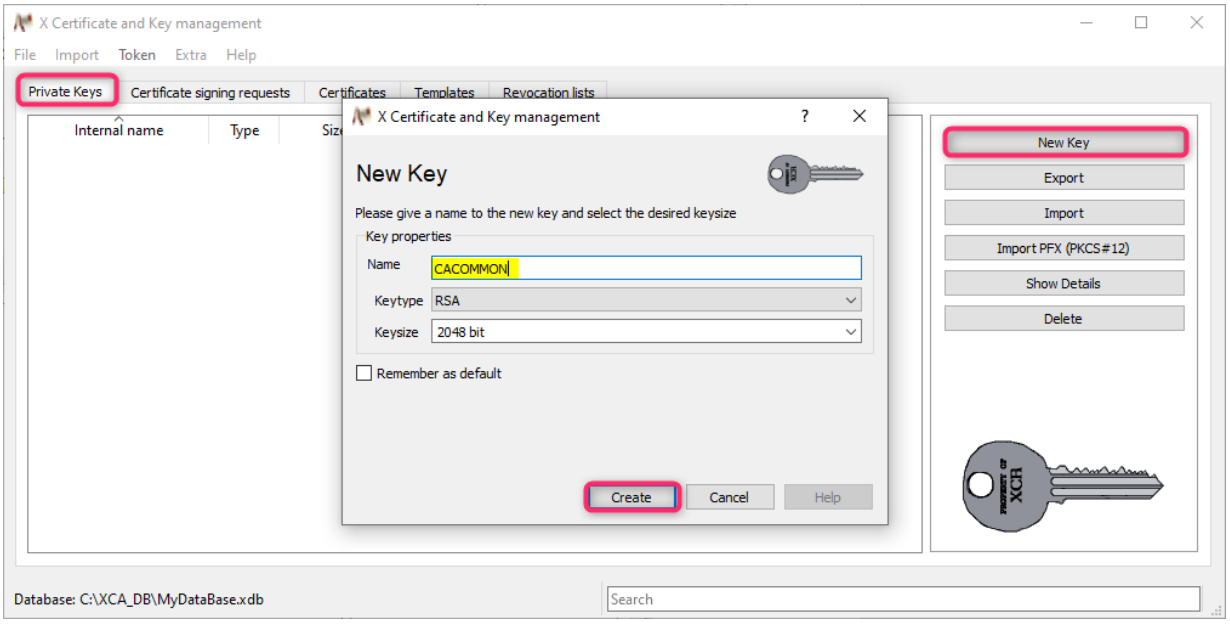
This application is intended for creating and managing X.509 certificates. Start your own PKI and create all kinds of private keys, certificates, requests or CRLs (Certificate Revocation List). You find more information on the official website and download the Software: https://hohnstaedt.de/xca/index.php

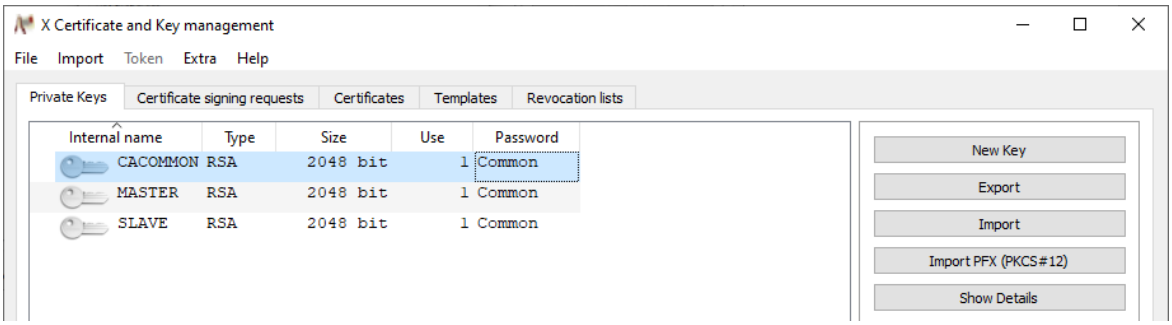
2.1. Create Private Keys

Let's start by opening XCA. In File > New DataBase, choose a location and a password (or no password).



Create 3 private keys for the CACOMMON, MASTER and SLAVE: Private Keys > New key





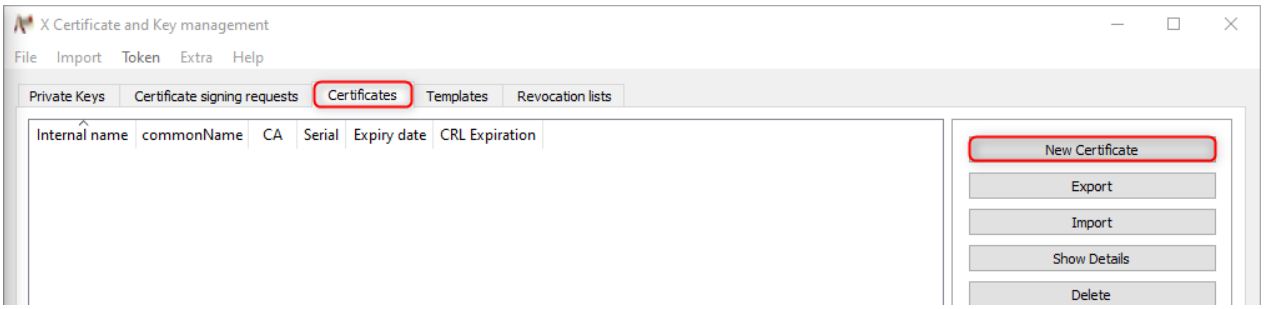
- CACOMMON: The key that is shared between Master/Outstation
- MASTER: Master authentication
- SLAVE: Slave authentication

2.2. Create self-signed Certificate Authority (CA)

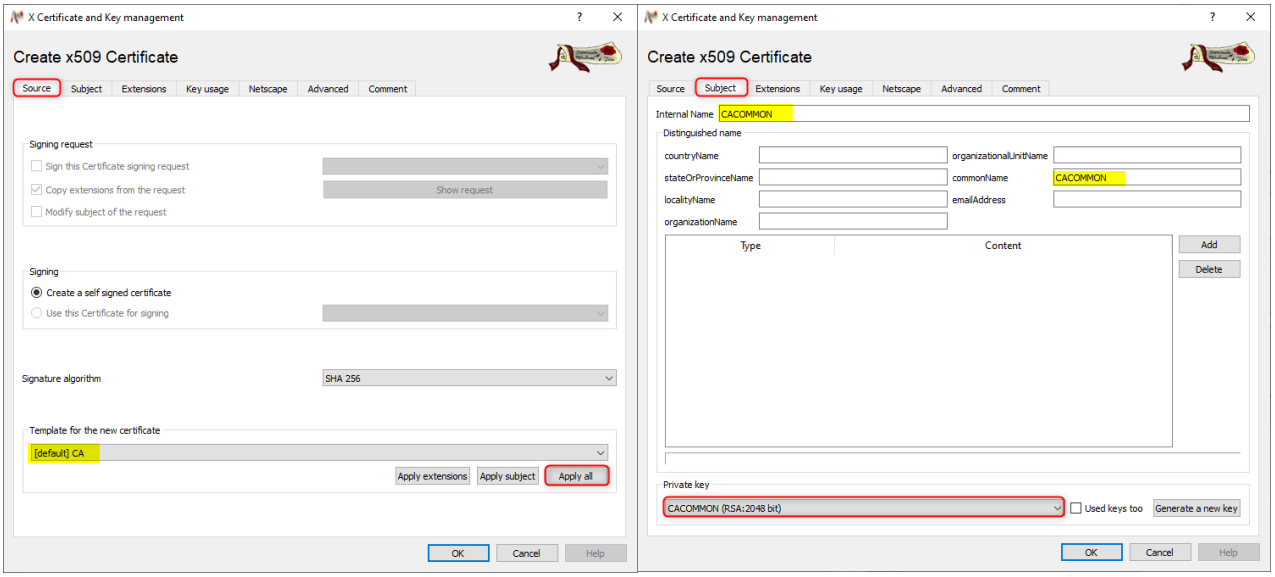
CERTIFICATE AUTHORITY - COMMON

Create a new self-signed certificate (CA): Certificates > New Certificate

This **CA** (Certificate Authority) will sign the other certificates. Note that here, this is done only for the test purposes.



- In the "Source" tab, create a self-signed certificate choosing "Create a self-signed certificate" in "Signing"
- Choose the "[default] CA" template for the new certificate then "Apply all"
- Then in the "Subject" tab, type the Internal Name and a commonName and do not forget to choose the **CACOMMON** private key

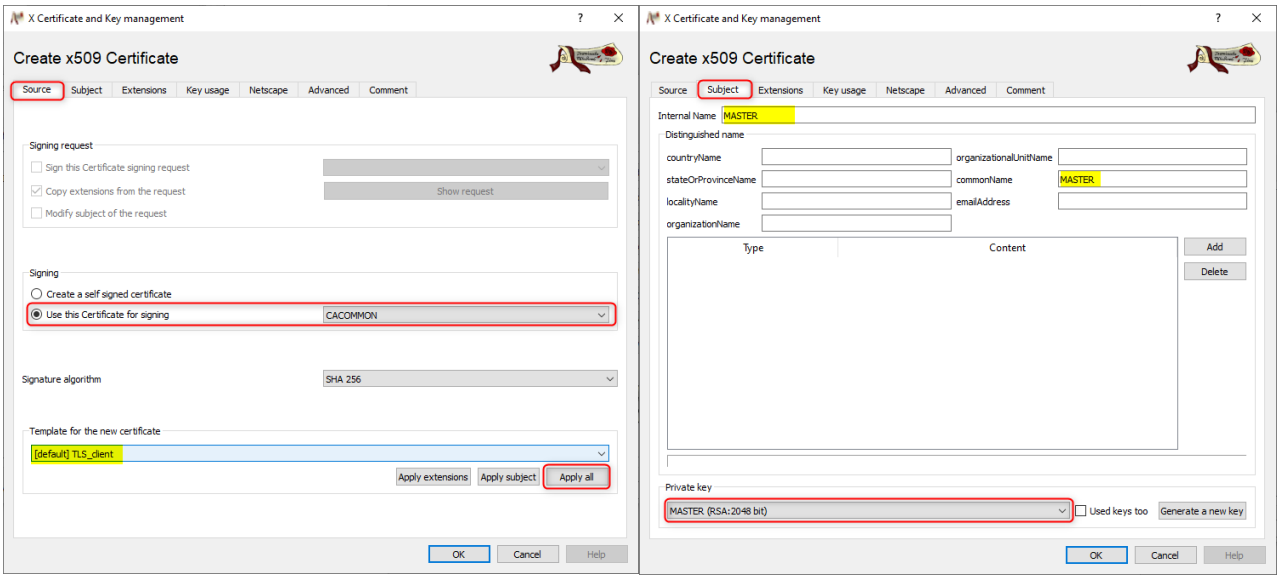


2.3. Create MASTER and SLAVE certificates

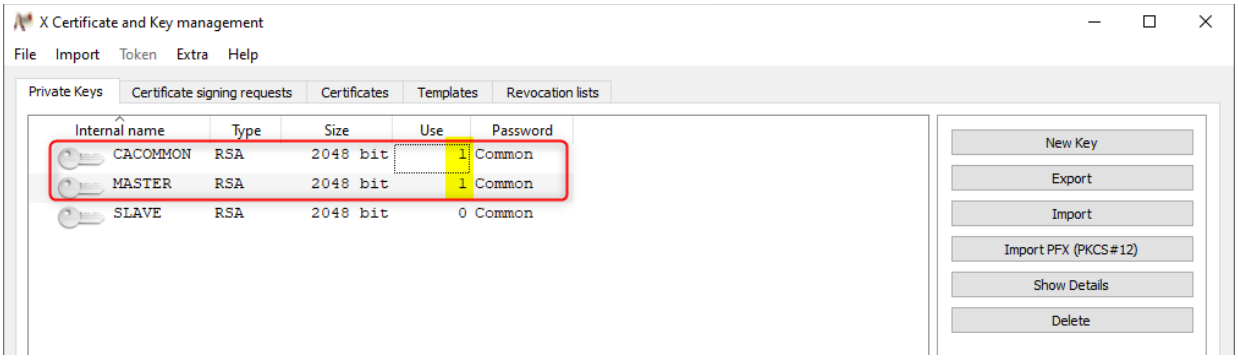
CERTIFICATE – MASTER

Do almost the same for the MASTER:

- Create a new certificate and use the "CACOMMON" CA for signing the MASTER certificate
- Choose the "[default] HTTPS_client" (or TLS_client) template for the new certificate, then apply all
- Then in "Subject", type the Internal Name and a commonName and do not forget to choose the MASTER private key.



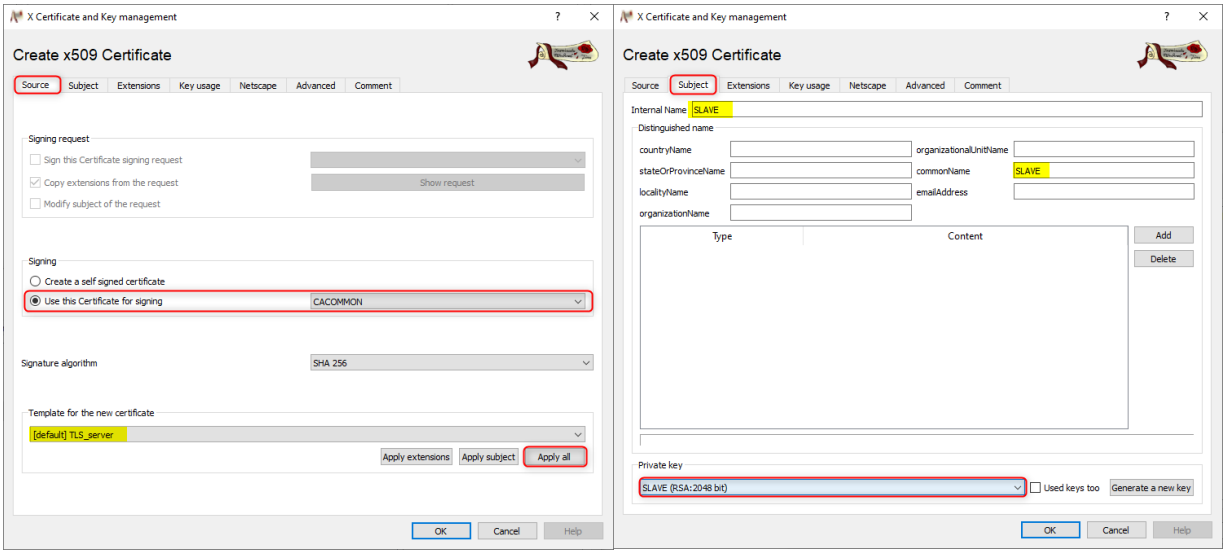
You can also verify that MASTER and CACOMMON are now used:



CERTIFICATE - SLAVE

Same principle for the SLAVE:

- Create a new certificate and use the "CACOMMON" CA to sign the certificate
- Choose the "[default] HTTPS_server" (or TLS_server) template for this certificate, then apply all
- Then in "Subject", type the Internal Name and a commonName and do not forget to choose the SLAVE private key.



2.4. Example of folders tree structure

For the next steps you need to create a new folders tree structures like this:

C:\PKI\IEC870M *(for example)*\

- CA
- PRIVATE

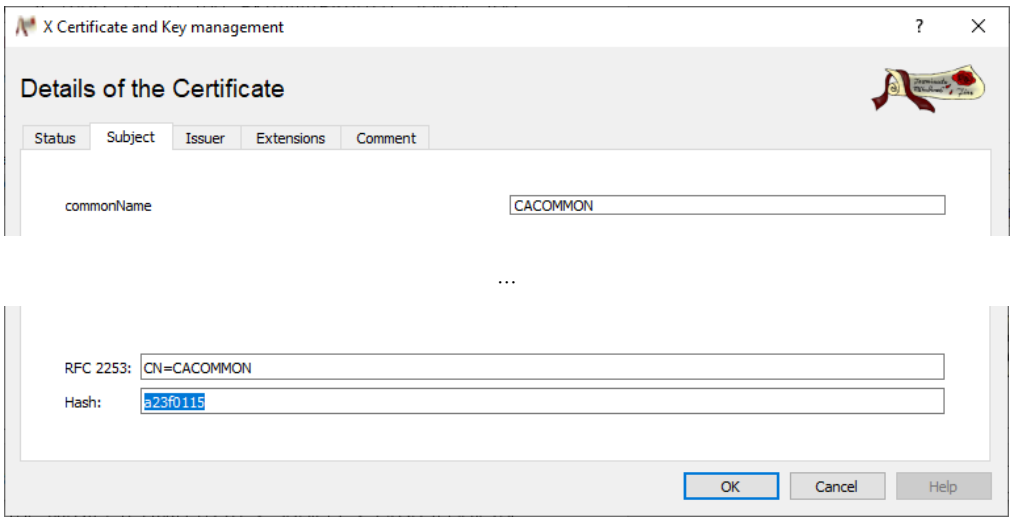
C:\PKI\IEC870S *(for example)*\

- CA
- PRIVATE

It's very important to write the name of the sub-folder "CA" and "PRIVATE" like in the example

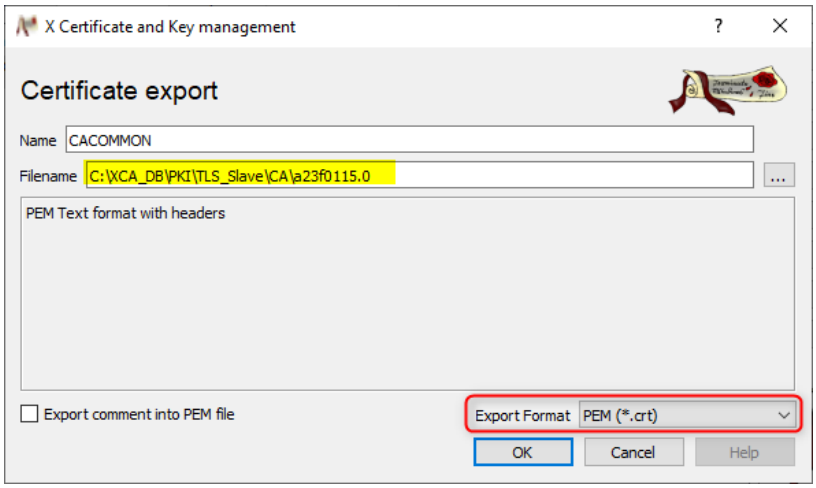
2.5. Trustlists (*.0)

- Double-click on the CA root certificate (CACOMMON) > Subject > Hash (copy the Hash)



SLAVE TRUSTLIST:

Right-click on the CACOMMON certificate: Export > File



Put it in a PKI/xxx/CA folder with the <HASH>.0 extension (for example in C:\PKI\IEC870S\CA\a23f0115.0)

Take care about the extension, the export format must be *.crt but the file extension must be *.0

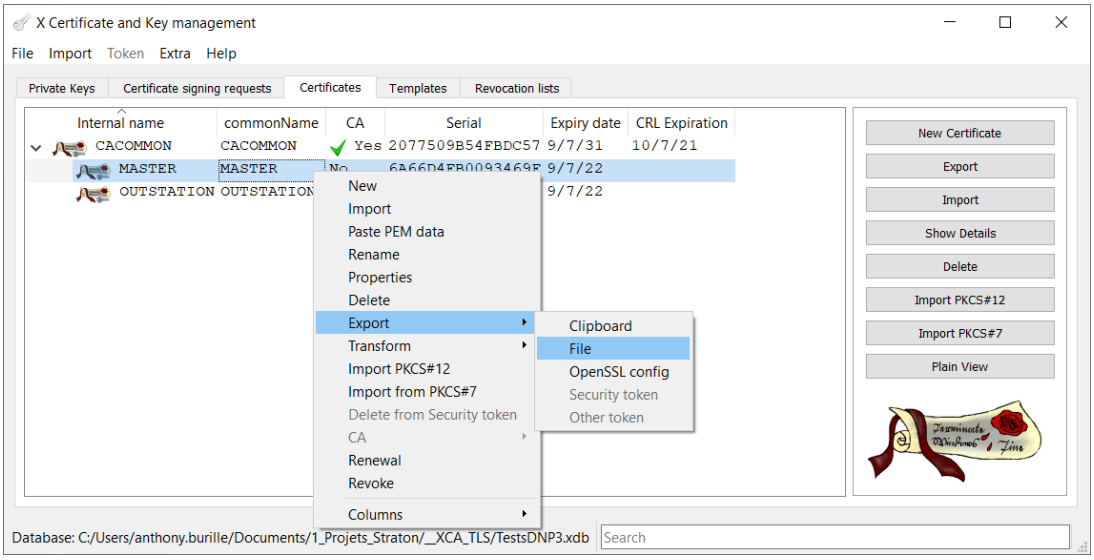
MASTER TRUSTLIST:

The CA is the same, so the trustlist for the MASTER is the same too. Redo the same thing for the MASTER or copy/paste the trustlist (*.0) in the MASTER's folder (for example in C:\PKI\IEC870M\CA\a23f0115.0)

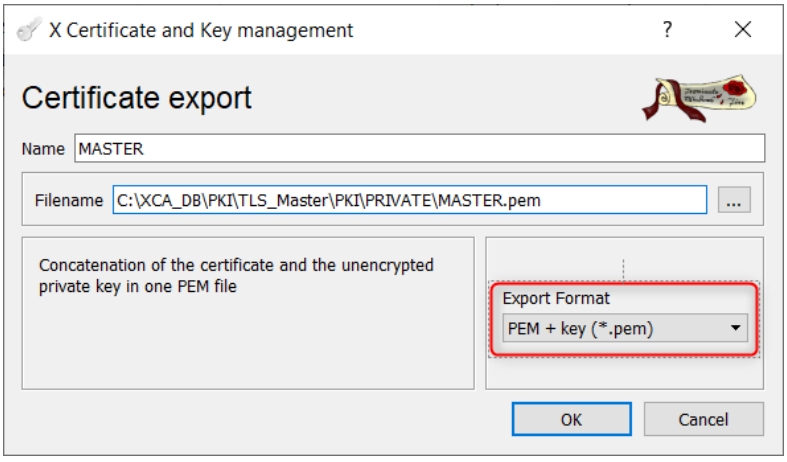
2.6. Private certificates (*.pem)

MASTER'S PRIVATE CERTIFICATE

Export the MASTER's private certificate doing a right click on the MASTER certificate > Export > File



Then choose the format "PEM + key (*.pem)". The file name does not matter. It must be in the PKI/xxx/PRIVATE folder (eg. C:\PKI\IEC870M\PRIVATE\MASTER.pem)



SLAVE'S PRIVATE CERTIFICATE

Export the SLAVE's private certificate doing a right click on the SLAVE certificate > Export > File

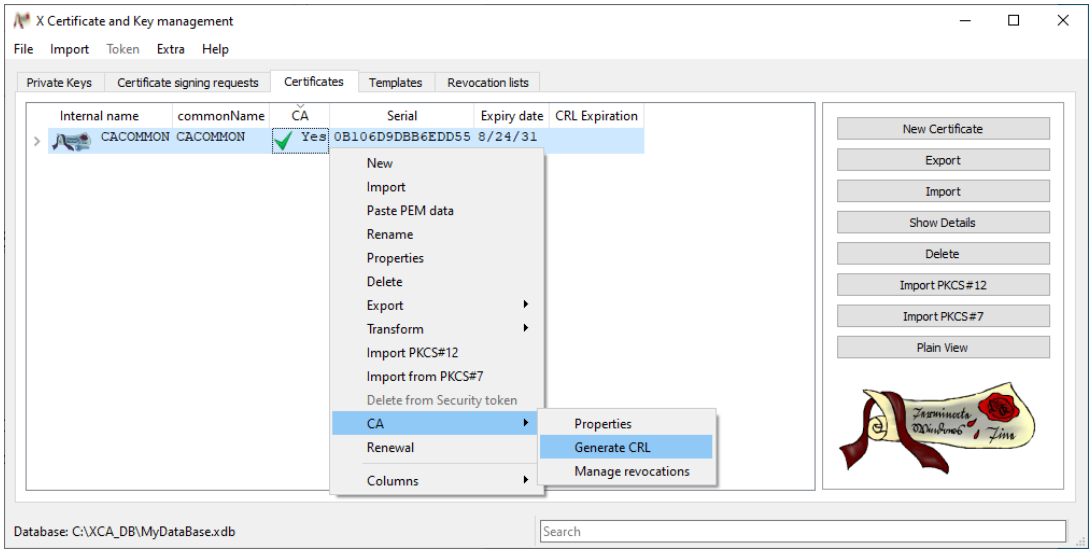
Then as for the MASTER, choose the format "PEM + key (*.pem)". It must be in the PKI/yyy/PRIVATE folder (eg. C:\PKI\IEC870S\PRIVATE\SLAVE.pem)

2.7. Certificate revocation list (*.r0)

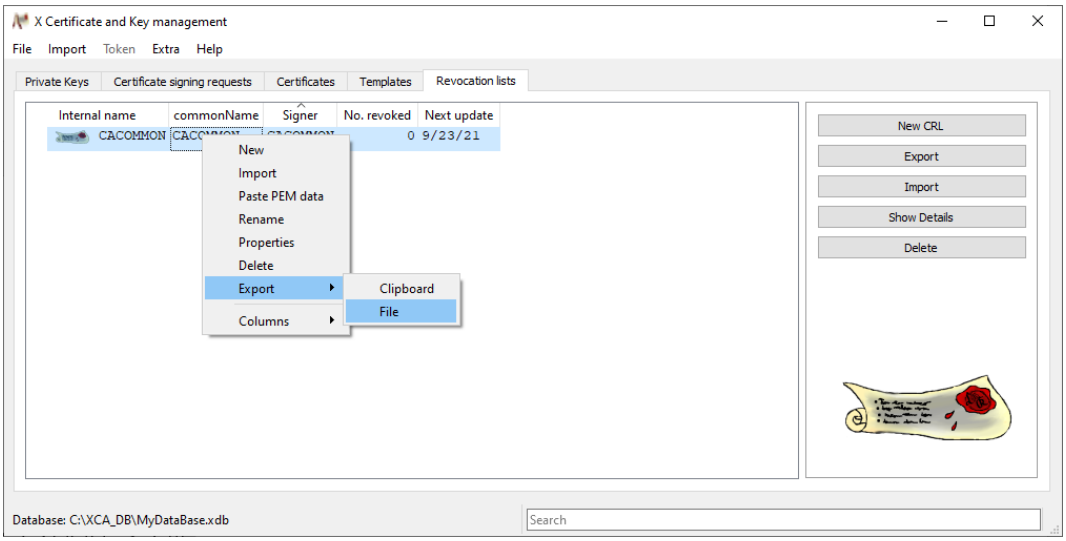
Generate the certificate revocation list containing the list of revoked certificates. This CRL must be manually updated if certificates are revoked.

Right-click on the CACOMMON CA > Generate CRL

- Then press OK to validate, the CRL is now in the "Revocation lists" tab

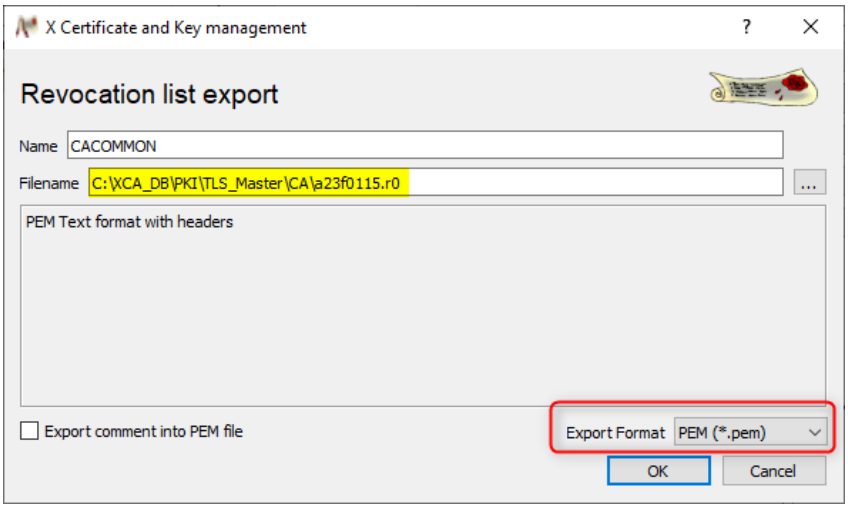


In the revocation lists tab, right-click on the CRL > Export > File

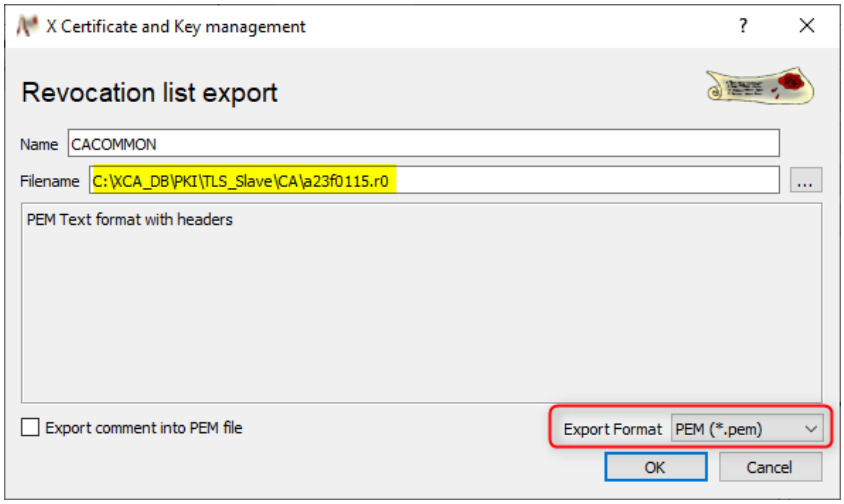


Put this file in the SLAVE's CA folder, with the hash of the CACOMMON, with extension *.r0 (eg. C:\PKI\IEC870S\CA\a23f0115.r0)

Take care of the extension, the export format must be *.pem but extension must be *.r0



Do exactly the same thing to export the CRL based on the common CA in the SLAVE's PKI folder (or simply copy/paste the *.r0 into the SLAVE's PKI\CA folder)



3. Use OpenSSL for "ENERGY" certificates

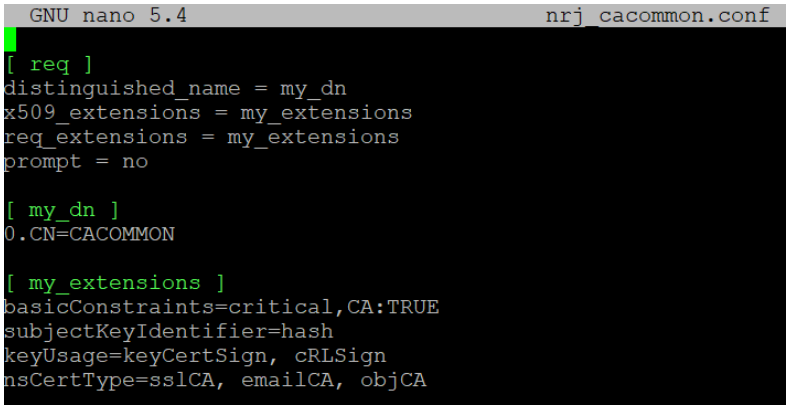
3.1. Configuration files (*.conf)

With OpenSSL you have to create configuration files which contains parameters/extensions you want to add to each certificates.

3.1.1. Create the CACOMMON configuration file

Use the following command and create the configuration file of the CACOMMON as shown on the capture below :

```
# sudo nano nrj_cacommon.conf
```



Text to copy/paste :

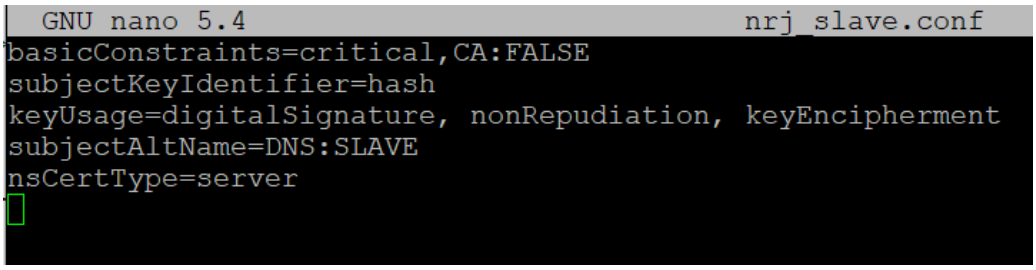
```
[ req ]
distinguished_name = my_dn
x509_extensions = my_extensions
req_extensions = my_extensions
prompt = no

[ my_dn ]
0.CN=CACOMMON

[ my_extensions ]
basicConstraints=critical,CA:TRUE
subjectKeyIdentifier=hash
keyUsage=keyCertSign, cRLSign
nsCertType=sslCA, emailCA, objCA
```

NOTE: To save and quit : *CTRL+X* then *Y (Yes)* and *ENTER*

3.1.2. Create the SLAVE configuration file

Create the configuration file for the SLAVE as shown on the capture below :

```
# sudo nano nrj_slave.conf
```



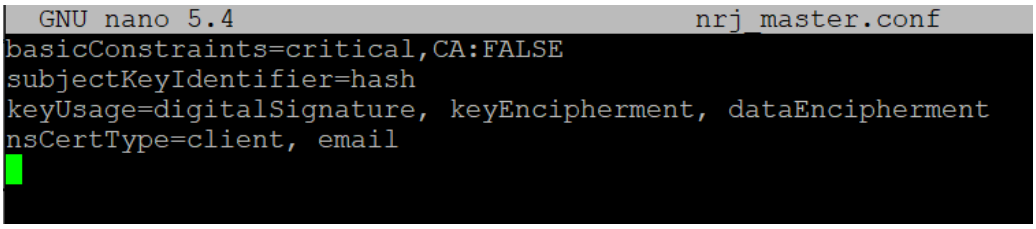
Text to copy/paste :

```
basicConstraints=critical,CA:FALSE
subjectKeyIdentifier=hash
keyUsage=digitalSignature, nonRepudiation, keyEncipherment
subjectAltName=DNS:SLAVE
nsCertType=server
```

3.1.3. Create the MASTER configuration file

Create the configuration file for the MASTER as shown on the capture below :

```
# sudo nano nrj_master.conf
```



Text to copy/paste :

```
basicConstraints=critical,CA:FALSE
subjectKeyIdentifier=hash
keyUsage=digitalSignature, keyEncipherment, dataEncipherment
nsCertType=client, email
```

3.2. Create Private Keys (*.key)

Use the following command to create the private key of the CACOMMON:

```
# openssl genrsa -out CACOMMON.key 2048
```

Create the private key of the SLAVE:

```
# openssl genrsa -out SLAVE.key 2048
```

Create the private key of the MASTER:

```
# openssl genrsa -out MASTER.key 2048
```

3.3. Create the CA certificate (*.crt)

Use the following command to create a certificate for the CA (Certificate Authority) :

```
# openssl req -sha256 -nodes -new -x509 -key CACOMMON.key -out ca.crt
```

NOTE: It will demand to complete some fields (Country Name, State or Province Name, …). The only one needed is the "Common Name". In this case use "CACOMMON".

3.4. Create and Sign SLAVE base certificate (*.crt)

Use the following command to create a certificate signing request for the SLAVE :

```
# openssl req -new -out slave.csr -key SLAVE.key
```

And this one to create and sign the SLAVE certificate using the configuration file created previously :

```
# openssl x509 -req -in slave.csr -CA ca.crt -CAkey CACOMMON.key -CAcreateserial -out slave.crt -extfile nrj_slave.conf
```

NOTE: It will demand to complete some fields (Country Name, State or Province Name, …). The only one needed is the "Common Name". In this case use "SLAVE".

3.5. Create and Sign MASTER base certificate (*.crt)

Create a certificate signing request for the MASTER :

```
# openssl req -new -out master.csr -key MASTER.key
```

Create and sign the MASTER certificate (using the configuration file created previously) :

```
# openssl x509 -req -in master.csr -CA ca.crt -CAkey CACOMMON.key -CAcreateserial -out master.crt -extfile nrj_master.conf
```

NOTE: It will demand to complete some fields (Country Name, State or Province Name, …). The only one needed is the "Common Name". In this case use "MASTER".

3.6. Create SLAVE and MASTER private certificates (*.pem)

To create private certificates for the SLAVE and the MASTER you'll to use the `cat` command to concatenate the private key and the base certificate into a *.pem file.

For the SLAVE :

```
# cat SLAVE.key SLAVE.crt > SLAVE.pem
```

For the MASTER :

```
# cat MASTER.key MASTER.crt > MASTER.pem
```

3.7. Create the Trustlist (*.0) (CACOMMON "certificate")

Use the following command to create a "trustlist" using the CACOMMON configuration file :

```
# openssl req -x509 -sha256 -nodes -new -out a23f0115.0 -key CACOMMON.key -config NRJ_CACOMMON.conf
```

3.8. Create and Organize the PKI folder

Now you will need to create the following files architecture :

home\pi\PKI\

- IEC870S\
  - CA
  - PRIVATE
- IEC870M\
  - CA
  - PRIVATE

/!\ Do not change names of folders CA and PRIVATE or the straton editor will not be able to correctly recover the files

To create each folder use the command `mkdir.`

Example with the SLAVE:

```
# mkdir ~/PKI
# mkdir ~/PKI/IEC870S
# mkdir ~/PKI/IEC870S/CA
# mkdir ~/PKI/IEC870S/PRIVATE
```

3.9. Add certificates and trustlist in the correct folders

To add certificates and the truslist to the right place you can use the command `cp <source> <destination_folder>` as shown in the example below.

Example:

```
# cp SLAVE.pem PKI/IEC870S/PRIVATE/
```

For the SLAVE :

- Put the Trustlist (a23f0115.0) under *PKI\IEC870S\CA*
- Put the SLAVE certificate (SLAVE.pem) under *PKI\IEC870S\PRIVATE*

For the MASTER :

- Put the Trustlist (a23f0115.0) under *PKI\IEC870M\CA*
- Put the MASTER certificate (MASTER.pem) under *PKI\IEC870M\PRIVATE*

4. Example of implementation in straton with the IEC60870

The configuration of TLS in straton follows the same principle for the following drivers : DNP3, IEC60870 and IEC61850. Only the location of the different parameters differs.

In the following parts, we will take the example of an IEC60870 configuration.

The MASTER will be on Windows.

The SLAVE on a Raspberry.

Always think to put the right files at the right places and configure the different path accordingly in the Fieldbus Configurators.

Remember: if you followed the previous steps, you must have this file architecture:

...\PKI\IEC870S\

- CA\
  - a23f0115.0 = Trustlist from the CA
  - a23f0115.r0 = Certificate revocation list from the CA *(optional)*
- PRIVATE\
  - SLAVE.pem = Slave's certificate (in *.pem format = pem+key file)

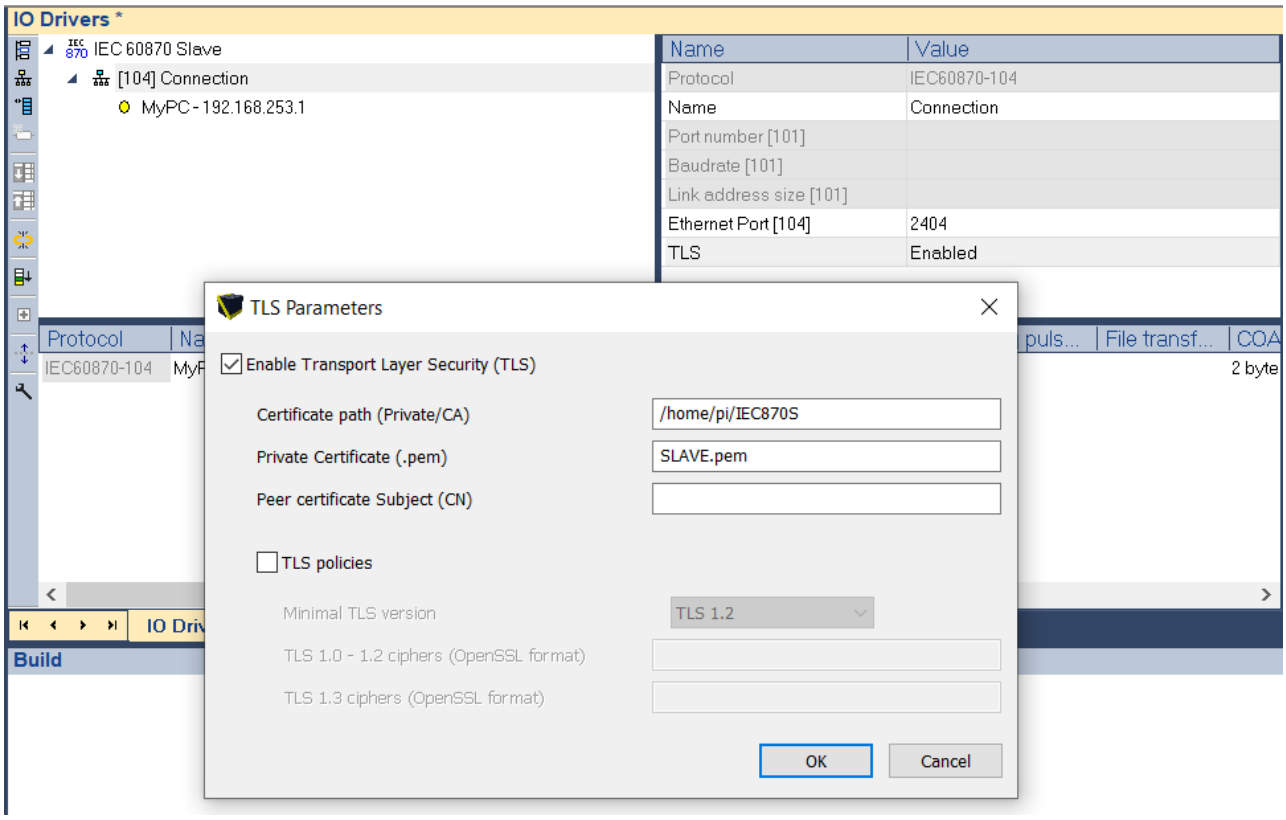
...\PKI\IEC870M\

- CA\
  - a23f0115.0 = Trustlist from the CA
  - a23f0115.r0 = Certificate revocation list from the CA *(optional)*
- PRIVATE\
  - MASTER.pem = Master's certificate (in *.pem format = pem+key file)

4.1. Slave configuration

- In the driver's configuration
  - Enable TLS
  - Put the certificate path (eg. /home/pi/PKI/IEC870S if you are on a Raspberry)
  - Put the certificate name (SLAVE.pem)

If you want to use the Common Name, put for example on the SLAVE side /CN=MASTER (because it is the "peer" CN)



4.2. Master configuration

Follow the same principle as for the 870 Server configuration.

- On the driver's configuration
  - Enable TLS
  - Put the certificate path (eg. C:\PKI\IEC870M)
  - Put the certificate name (MASTER.pem)

4.3. Peer certificate Subject

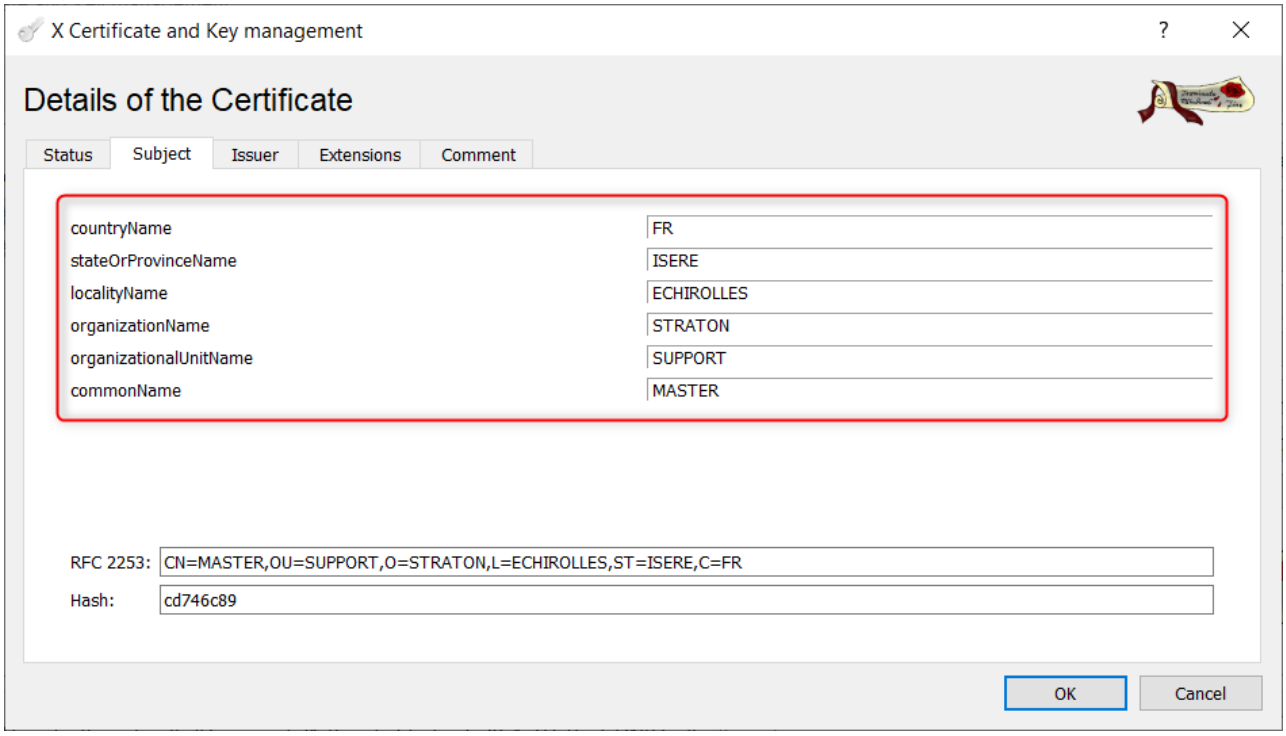
The peer Certificate Subject is the one from the partner. For example on the SLAVE side, the parameter can be /CN=MASTER

These are the parameters entered when creating the certificates from the private keys, in the part *2.3* of this tutorial.

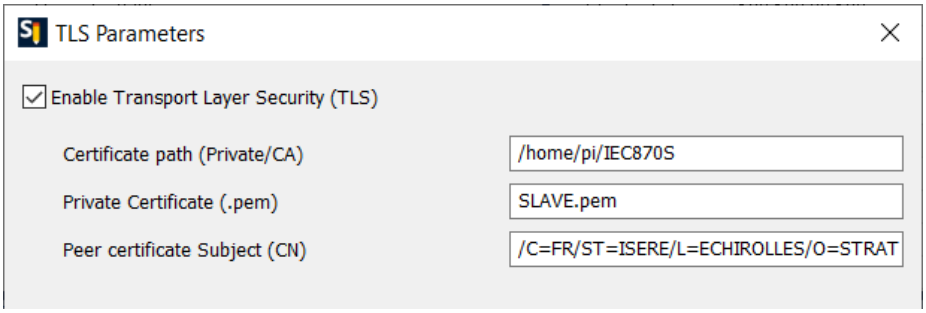
You can recover it double-clicking on the certificate in the XCA "Certificate" tab.

Using the peer certificate subject is optional.

For example if we have the following parameters on the MASTER certificate:



Then in the SLAVE configuration (because it expects the peer subject), you must fill all these parameters:



If parameters are not filled in the right order, then after the MASTER tried to connect, the SLAVE's Runtime will output an error like:

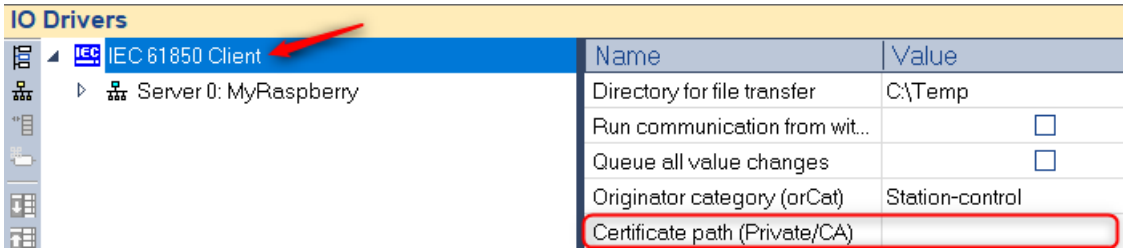
Module '(null)' - error: 0.0.0.0:20000<=192.168.33.139:58609: Error peer certificate has unexpected subject '/C=FR/ST=ISERE/L=ECHIROLLES/O=STRATON/OU=SUPPORT/CN=MASTER'

5. Frequently Asked Questions about ENERGY certificates

Where can I find the parameter to add my certificate path in the IEC61850 Client driver?

Here (IEC61850 Client) it does not work like the 850 Server, you will not be able to find the "Certificate path (Private/CA)" in the TLS parameter (the one which can have the state : Enable/Disable).

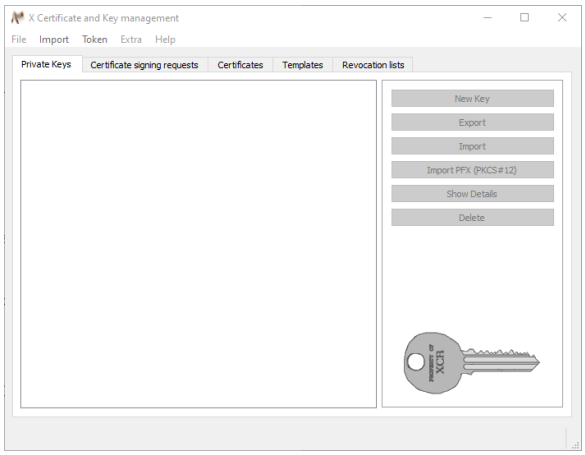
The "Certifiacte path (Private/CA)" can be found directly from the first level of the configurator (IEC 61850 Client)



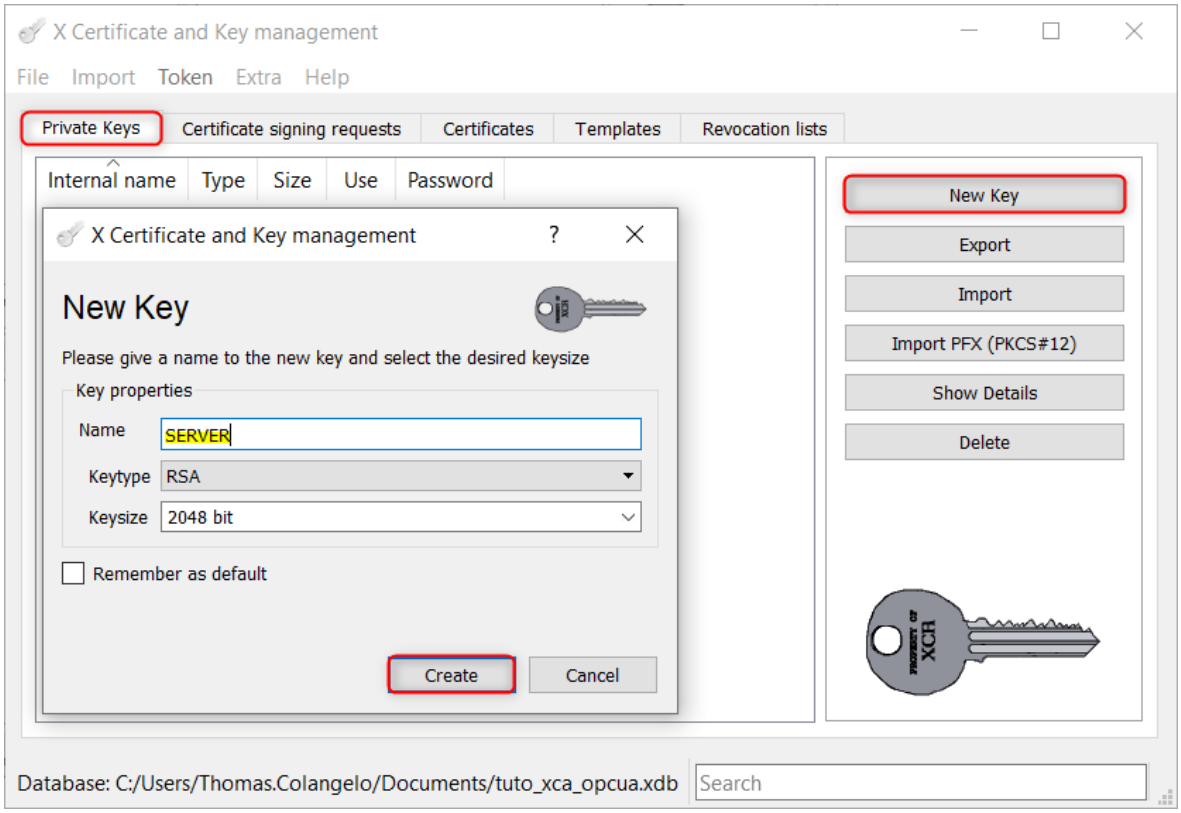
6. Use XCA Software for OPC UA certificates

6.1. Private Keys

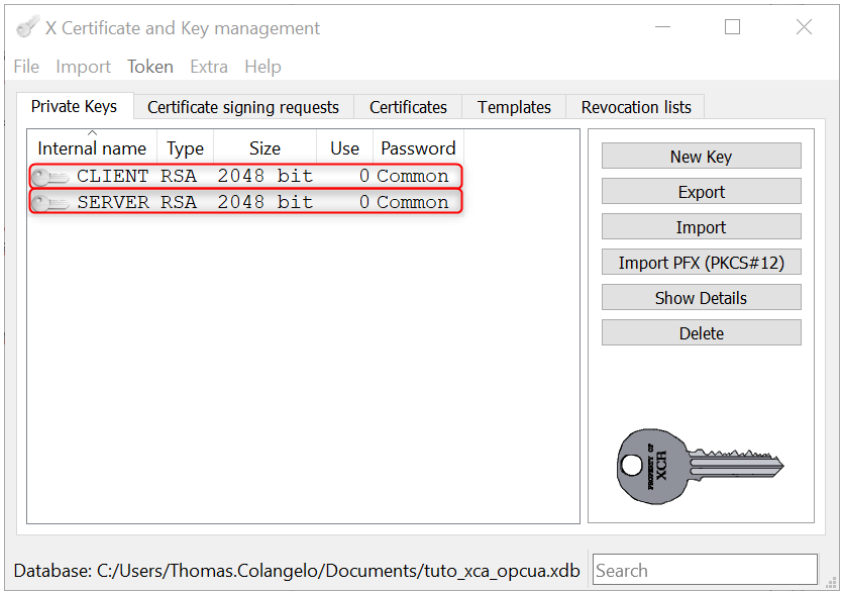
Let's start by opening XCA. In File > New DataBase, choose a location and a password (or no password)



Create 2 private keys for the SERVER and the CLIENT

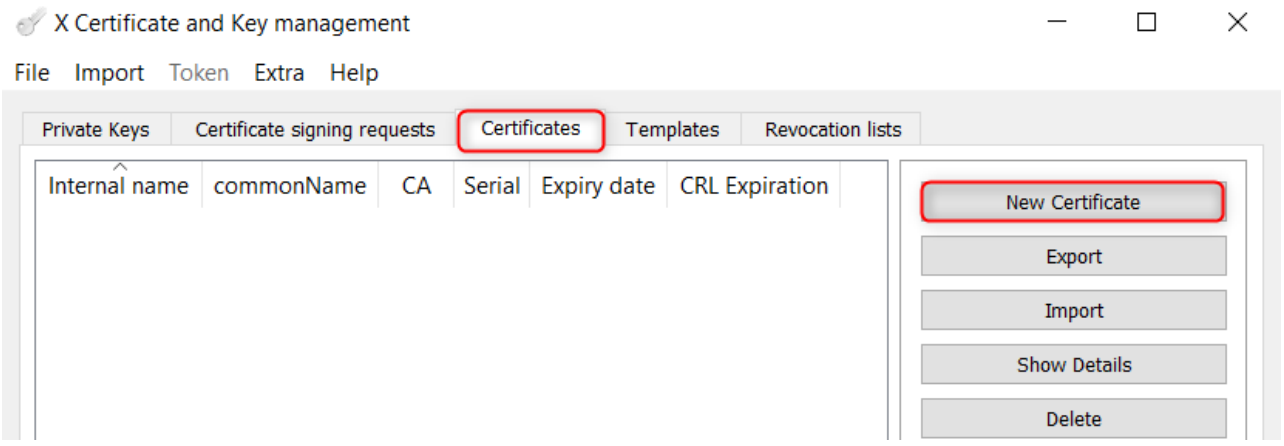


You should obtain something like that :



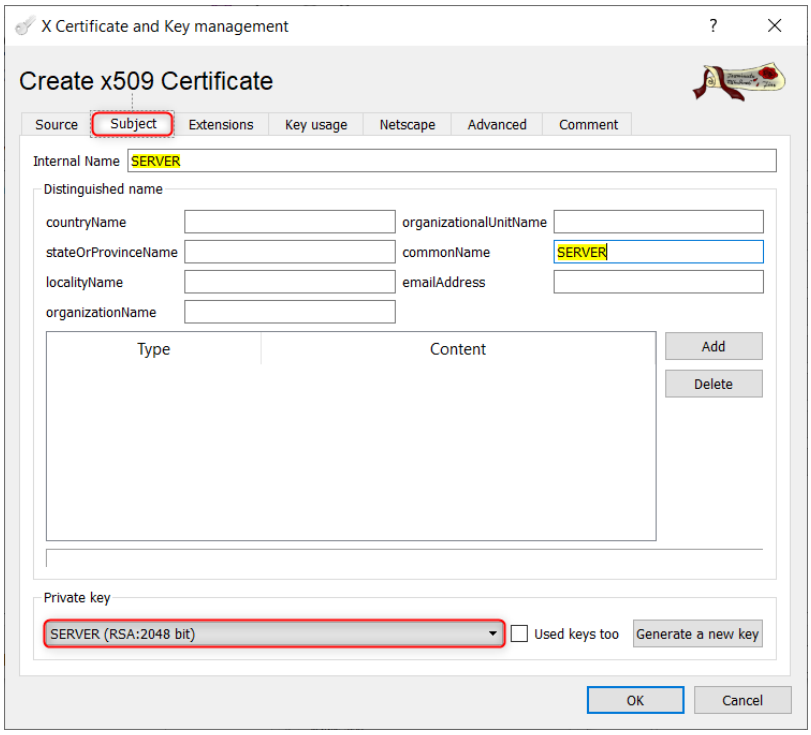
6.2. Create self-signed certificates

Now we will create 2 self-signed certificate : Certificates > New Certificate

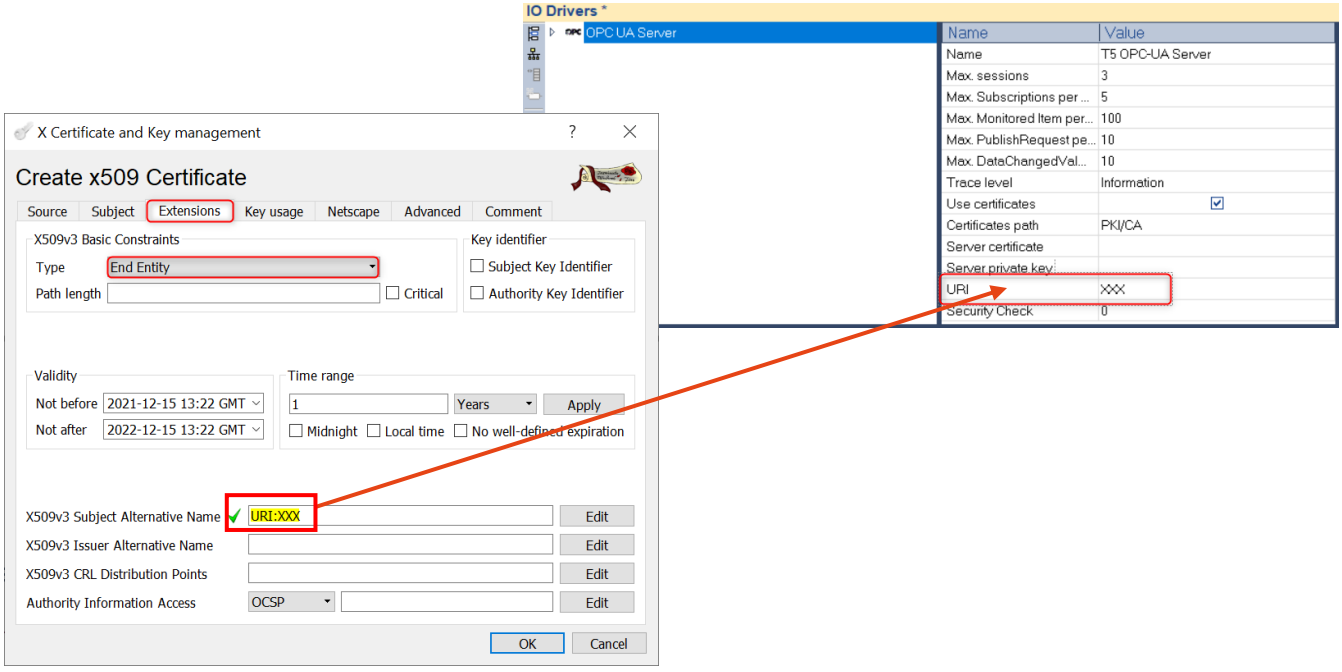


6.2.1. Create the SERVER certificate

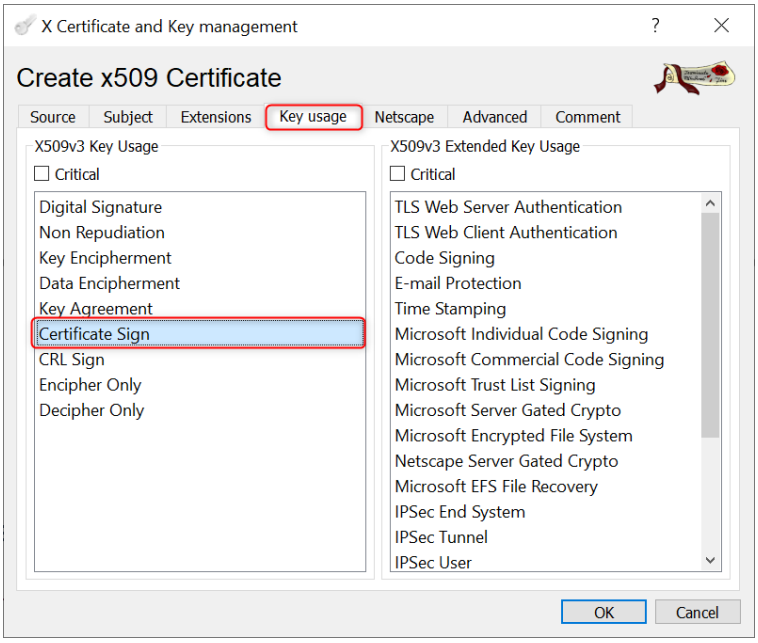
Under the "Subject" tab, add an *Internal Name* and a *commonName* and select the private key **SERVER** if it's not use by default



Under the "Extensions" tab, change the *Type* to "End Entity" and put an **URI** in the field *X509v3 Subject Alternative Name* (here the "XXX" correspond to the **URI** chosen in the straton editor)



And finally, under the "Key Usage" tab, select "Certificate Sign" then press "OK" to create the **SERVER** certificate



6.2.2. Create the CLIENT certificate

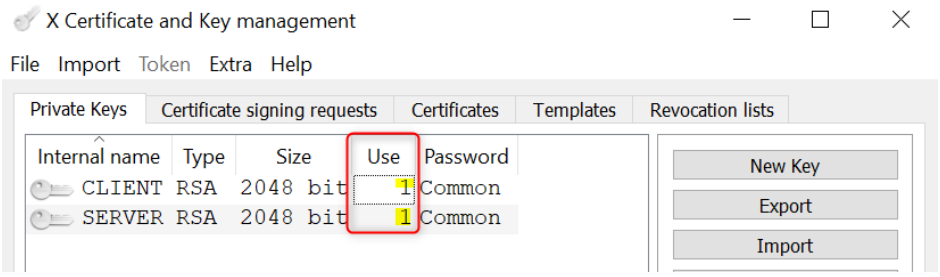
For the CLIENT certificate the steps are pretty identical

Under the "Subject" tab, add an *Internal Name* and a *commonName* (something different than the one choose for the server) and select the private key **CLIENT** if it's not use by default

Under the "Extensions" tab, change the *Type* to "End Entity" and in the field *X509v3 Subject Alternative Name* enter "**URI:urn:stratonopcua.client.application**"

And finally, under the "Key Usage" tab, select "Certificate Sign" then press "OK" to create the **CLIENT** certificate

Note that you can check if the private keys are correctly used



6.3. Create the PKI folder

For the next steps you'll need to create the following file architectures :

C:\CLIENT_PKI\PKI\CA\

- certs
- crl
- private

C:\SERVER_PKI\PKI\CA\

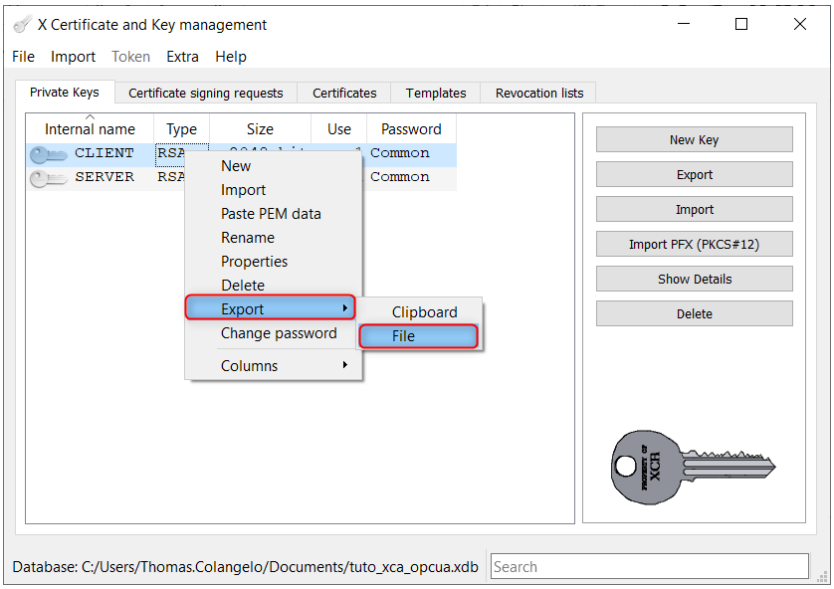
- certs
- crl
- private

/!\ Do not change the folders' names or the straton editor will not be able to correctly recover the files that we will create in the next steps

6.4. Private Keys export (*.pem)

Export the **SERVER** and **CLIENT** private keys by doing a right click on it then Export > File

Use the *Export Format* "PEM private (*.pem)"

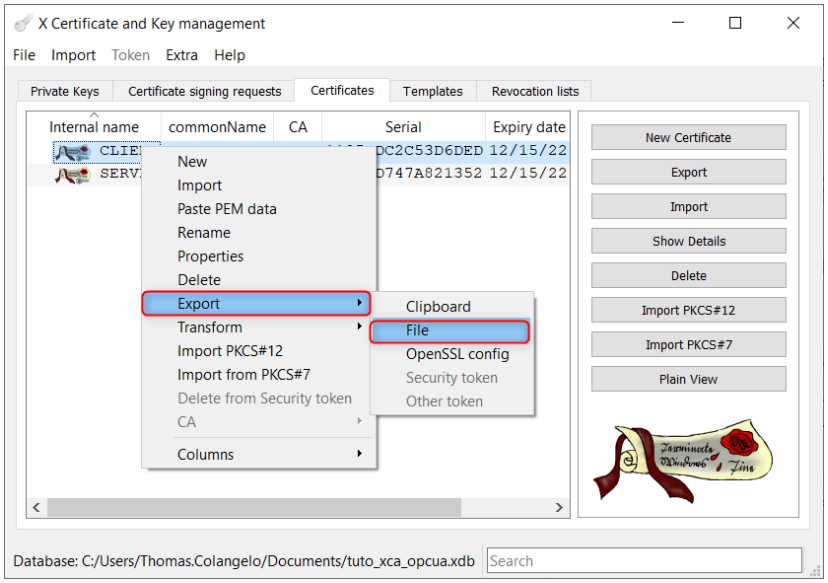


For the **SERVER**, put the private key under *SERVER_PKI\PKI\CA\private*

For the **CLIENT**, put the private key under *CLIENT_PKI\PKI\CA\private*

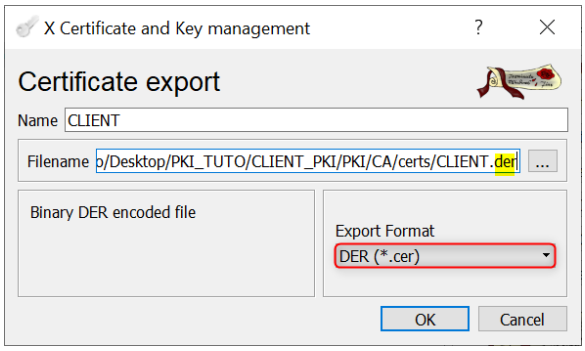
6.5. Certificates export (*.der)

Export the **SERVER** and **CLIENT** certificates by doing a right click on it then Export > File



Use the *Export Format* "DER (*.cer)"

*/!\ It's needed to change the extension *.cer in *.der (straton works only with the *.der certificates)*



For the **SERVER**, put the certificate under *SERVER_PKI\PKI\CA\certs* and *CLIENT_PKI\PKI\CA\certs*

For the **CLIENT**, put the certificate under *CLIENT_PKI\PKI\CA\certs* and *SERVER_PKI\PKI\CA\certs*

7. Use OpenSSL for OPC UA certificates

7.1. Configuration file (*.conf)

With OpenSSL you have to create a configuration file which contains the whole parameters of the certificates you want to create.

7.1.1. Create the SERVER configuration file

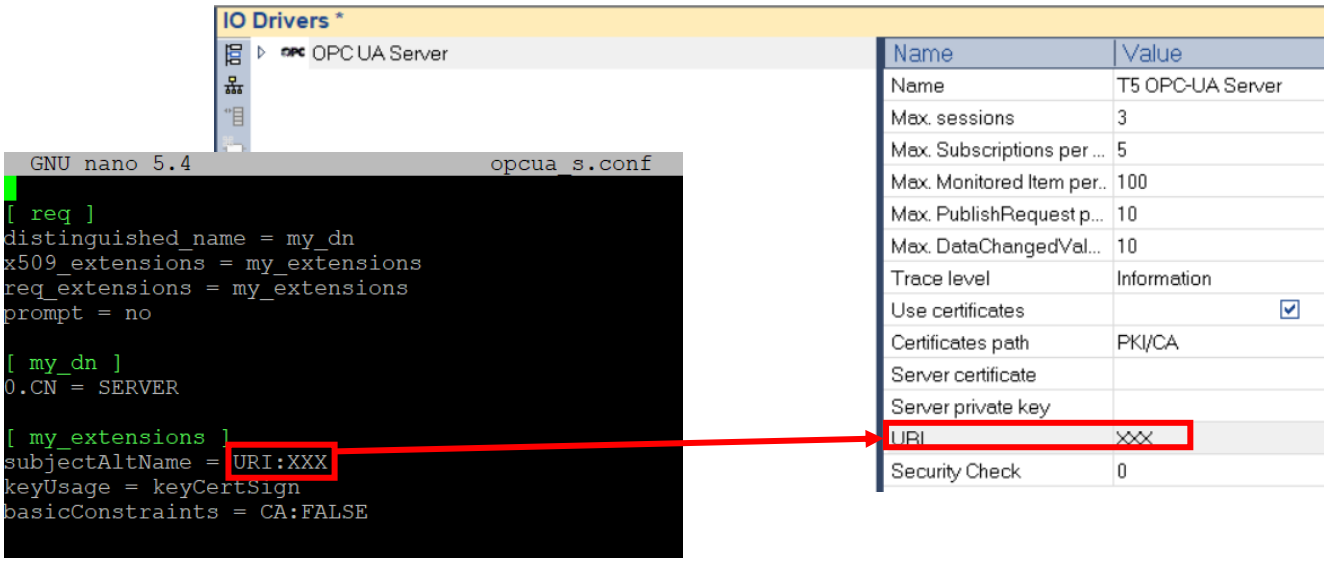
Use the following command and create the configuration file of the SERVER as shown on the capture below :

```
# sudo nano opcua_s.conf
```

```
  GNU nano 5.4                                opcua_s.conf

[ req ]
distinguished_name = my_dn
x509_extensions = my_extensions
req_extensions = my_extensions
prompt = no

[ my_dn ]
0.CN = SERVER

[ my_extensions ]
subjectAltName = URI:XXX
keyUsage = keyCertSign
basicConstraints = CA:FALSE
```

⇨

Under `[ my_extensions ]` the URI chosen as **`subjectAltName`** should correspond to the URI field of the straton editor



Text to copy/paste :

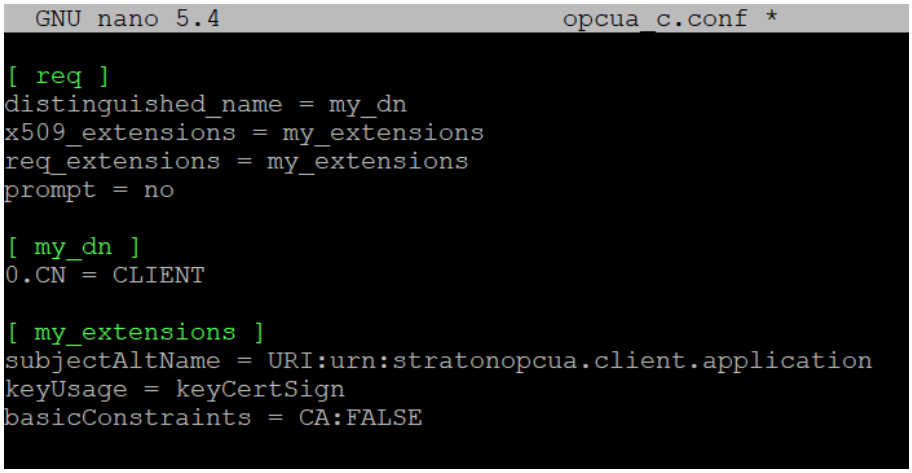
```
[ req ]
distinguished_name = my_dn
x509_extensions = my_extensions
req_extensions = my_extensions
prompt = no

[ my_dn ]
0.CN = SERVER

[ my_extensions ]
subjectAltName = URI:XXX
keyUsage = keyCertSign
basicConstraints = CA:FALSE
```

NOTE: To save and quit : *CTRL+X* **then** *Y (Yes)* **and** *ENTER*

7.1.2. Create the CLIENT configuration file

Use the following command and create the configuration file of the CLIENT as shown on the capture:

```
# sudo nano opcua_c.conf
```



Text to copy/paste :

```
[ req ]
distinguished_name = my_dn
x509_extensions = my_extensions
req_extensions = my_extensions
prompt = no

[ my_dn ]
0.CN = CLIENT

[ my_extensions ]
subjectAltName = URI:urn:stratonopcua.client.application
keyUsage = keyCertSign
basicConstraints = CA:FALSE
```

7.2. Create Private Keys and self-signed Certificates

7.2.1. Create the SERVER Private Key and Certificate

Use the following command to create both, the private key and certificate of the SERVER:

```
# openssl req -x509 -sha256 -nodes -newkey rsa:2048 -keyout SERVER.pem -out SERVER.der -config opcua_s.conf
```

NOTE: It will demand to complete some fields (Country Name, State or Province Name, …). The only one needed is the "Common Name". In this case use "SERVER".

7.2.2. Create the CLIENT Private Key and Certificate

As for the SERVER, use the following command to create the private key and the certificate of the CLIENT:

```
# openssl req -x509 -sha256 -nodes -newkey rsa:2048 -keyout CLIENT.pem -out CLIENT.der -config opcua_c.conf
```

NOTE: It will demand to complete some fields (Country Name, State or Province Name, …). The only one needed is the "Common Name". In this case use "CLIENT".

7.3. Create and Organize PKI folders

7.3.1. Creation of each PKIs

Now you'll need to create the following file architectures :

home\pi\CLIENT_PKI\PKI\CA\

- certs
- crl
- private

home\pi\SERVER_PKI\PKI\CA\

- certs
- crl
- private

/!\ Do not change the folders' names or the straton editor will not be able to correctly recover the files

To create each folder use the command `mkdir.`

Example with the SERVER:

```
# mkdir ~/SERVER_PKI
# mkdir ~/SERVER_PKI/PKI
# mkdir ~/SERVER_PKI/PKI/CA
# mkdir ~/SERVER_PKI/PKI/CA/certs
# mkdir ~/SERVER_PKI/PKI/CA/crl
# mkdir ~/SERVER_PKI/PKI/CA/private
```

7.3.2. Add certificates and keys in the correct folders

To add certificates and private keys to the right place you can use the command `cp <source> <destination_folder>` as shown in the example below.

Example:

```
# cp CLIENT.pem CLIENT_PKI/PKI/CA/private/
```

For the SERVER:

- Put the SERVER private key (SERVER.pem) under *SERVER_PKI\PKI\CA\private*
- Put the SERVER certificate (SERVER.der) under *SERVER_PKI\PKI\CA\certs*
- Put the CLIENT certificate (CLIENT.der) under *SERVER_PKI\PKI\CA\certs*

For the CLIENT:

- Put the CLIENT private key (CLIENT.pem) under *CLIENT_PKI\PKI\CA\private*
- Put the CLIENT certificate (CLIENT.der) under *CLIENT_PKI\PKI\CA\certs*
- Put the SERVER certificate (SERVER.der) under *CLIENT_PKI\PKI\CA\certs*

8. Example of implementation in straton for the OPC UA

In the following parts, the SERVER will be on Windows and the CLIENT on a Raspberry.

Always think to put the right files at the right places and configure the different path accordingly in the Fieldbus Configurators.

Remember, if you've followed the previous steps, you must have this kind of file architecture :

...\CLIENT_PKI\PKI\CA\

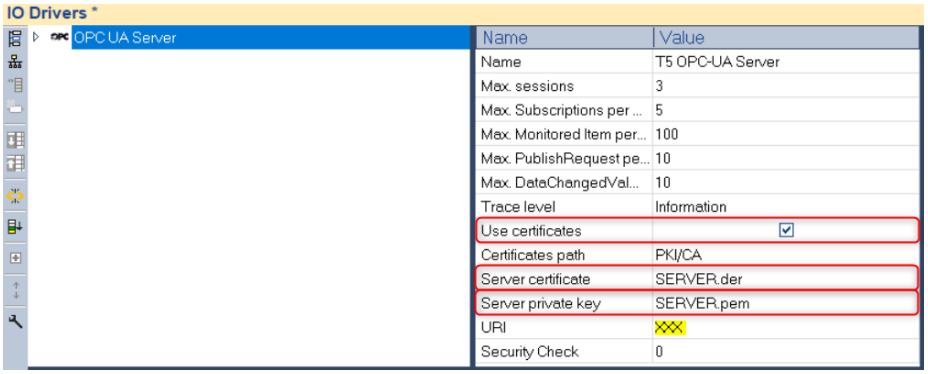
- certs\
  - CLIENT.der = Client's certificate
  - SERVER.der = Server's certificate
- crl\
- private\
  - CLIENT.pem = Client's private key

...\SERVER_PKI\PKI\CA\

- certs\
  - CLIENT.der = Client's certificate
  - SERVER.der = Server's certificate
- crl\
- private\
  - SERVER.pem = Server's private key

8.1. SERVER configuration

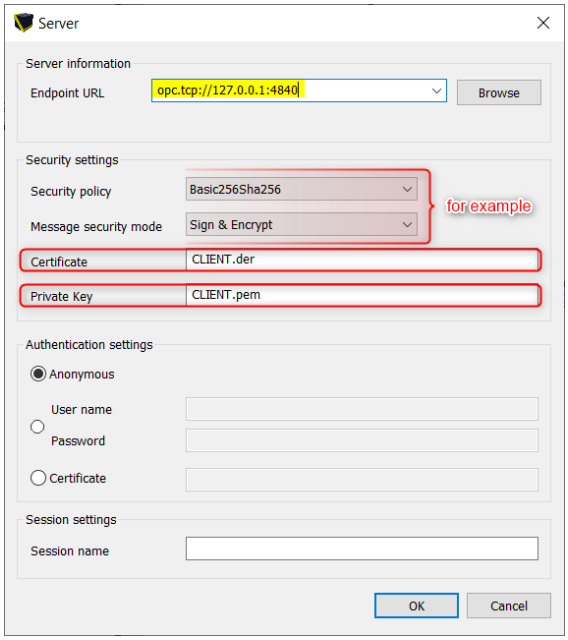
- In the driver's configuration
  - Check "Use certificates"
  - Under "Server certificate", wright the complete name of the certificate you're using (if you've followed the example of this tutorial it will be "SERVER.der")
  - Under "Server private key", wright the complete name of the server private key (if you've followed the example, it will be "SERVER.pem")
  - Also check that the URI is the same that you put in the server's certificate



- Certificate path
  - As we will run the SERVER on a Windows runtime it is necessary to put the PKI folder (created in the previous steps) into the Runtime's data

8.2. CLIENT configuration

- In the driver's configuration
  - Start by adding a "Master/Port"
  - Complete the "Endpoint URL" according to the server's one
  - Select a "Security policy" according to those accepted by the server **(/!\ do not use "None")**
  - Under "Certificate", wright the complete name of the certificate you're using (if you've followed the example of this tutorial it will be "CLIENT.der")
  - Under "Private Key",wright the complete name of the client private key (if you've followed the example, it will be "CLIENT.pem")



- Certificate path
  - As we will run the **CLIENT** on a Raspberry runtime it is necessary to put the PKI folder (created in the previous steps) at the same path than the runtime (ex.: /home/pi/)
  - Also need to put it into the editor's data (note that you can find it in the section "Help" of the editor under "About…" then click on the three dots "[…]")