swissbit®

Getting Started Guide

Swissbit iShield HSM

Document Version: 1.1

Table of Contents

1. Introduction

In this guide, you will learn how to use your Swissbit iShield HSM for AWS IoT Greengrass as hardware security module (HSM) using the PKCS#11 Hardware Security Integration (HSI). All AWS IoT Greengrass devices need to have a unique device certificate and corresponding private key for encrypted and authenticated communication with the AWS IoT Cloud and other local devices. The iShield HSM allows directly generating and storing private key on the HSM through a secure element.

In ch. 2, you will install and configure all prerequisites for the usage of the iShield HSM as hardware security module for Greengrass. Ch. 3 guides you through the setup of your AWS IoT Greengrass Core device with your iShield HSM and ch. 4 shows how to use the GUI application xca instead of the command line for required PKSC#11 API operations. Ch. 5 collects links referred to throughout this getting started guide.

2. Software Stack and Tools

2.1 Prerequisites

You need to install the following software so that Greengrass Core can communicate with your iShield HSM:

PCSC-Lite: Implements the PC/SC international standard for PC to smart card reader communication. The core component is the pcscd daemon program which coordinates communications with smart card readers, smart cards and cryptographic tokens like your iShield HSM that are connected to the system.

pcsc-tools: Can be used with smart cards and PC/SC. The tool pcsc_scan regularly checks for newly inserted or removed devices.

CCID: Allows the connection of a smartcard to a computer using a USB interface. The communication of iShield HSM with the Greengrass device is effected via the CCID (chip / smart card interface devices) driver. For automatic detection of your iShield HSM, the version 1.4.33 or later is required. The library libccid is installed as dependency of the PC/SC daemon.

libusb: Enables USB device access and is installed as dependency of libccid.

OpenSSL: Implements the TLS protocol and cryptographic functions. It can be used for the creation of keys, X.509 certificates, certificate signing requests and more.

OpenSC: Supports security applications like authentication, encryption and digital signature using smart cards. The library implements the PKCS#11 API and serves as PKCS#11 provider library on the Greengrass core.

libp11: Contains the library engine_pkcs11 that implements the OpenSSL engine interface for PKCS#11 modules.

Following this manual, you can either

- Install the prerequisites using a package manager or
- Use the provided software packages and install the software manually

If available on your device, you can install the prerequisites with a package manager as follows:

```
$sudo apt install libpcsclite-dev pcscd pcsc-tools libssl-dev libengine-pkcs11-openssl openssl
```

Download OpenSC 0.22.0 or later and configure and compile like

```
$tar -xf opensc-0.22.0.tar.gz
$cd opensc-0.22.0/
$./configure --prefix=/usr
$sudo make -j4 install
```

Alternatively, a software package containing all required components is available from https://www.swissbit.com/ishield-hsm/#tab-9-6900 for the architectures Armv7l, Aarch64 and x86_64

- pcsc
- pcsc-tools
- libccid
- libusb
- openssl
- libp11
- opensc

Download the archive, extract the content for all required component subdirectories built for your architecture to /usr, e.g. if your Greengrass device is a Raspberry PI with x64 OS copy the contents of aarch64/<component>/ into the /usr directory of your PI.

2.2 Start PC/SC daemon

This step is not required when you installed pcscd via a package manager as, in this case, it is running as systemd daemon. You can verify that the daemon is enabled with:

```
$systemctl status pcscd
● pcscd.service - PC/SC Smart Card Daemon
     Loaded:  loaded  (/lib/systemd/system/pcscd.service;  indirect;  vendor
preset: enabled)
     Active: inactive (dead)
TriggeredBy: ● pcscd.socket
       Docs: man:pcscd(8)
```

If you are using the provided software package instead, start the PC/SC daemon from /usr/bin with the following command:

```
$sudo ./pcscd -f --info
```

2.3 Detect iShield HSM

Now, insert your iShield HSM and verify that the device is recognized by running

```
$pcsc_scan
Using reader plug'n play mechanism
Scanning present readers...
0: Swissbit Secure USB PU-50n SE/PE (99000010EF000288) 00 00

Tue Apr 26 15:22:16 2022
 Reader 0: Swissbit Secure USB PU-50n SE/PE (99000010EF000288) 00 00
  Event number: 0
  Card state: Card inserted,
  ATR: 3B DC 18 FF 81 91 FE 1F C3 80 73 C8 21 13 66 05 03 63 51 00 02 50
```

If the device is not detected, install a more recent CCID driver version, e.g. ccid-1.4.33 or later. Alternatively, you can supplement the Info.plist of your CCID driver (usually located at /usr/lib/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist) with the missing entries from the respective file distributed as part of the libccid component of the provided software package. Restart the PC/SC daemon and check for the iShield HSM again.

2.4 Adding OpenSSL Engine Support

Through the engine mechanism, OpenSSL can access the objects on the iShield HSM directly through the PKCS#11 API. A PKCS#11 engine has to be registered with OpenSSL in the OpenSSL configuration file. The OpenSSL configuration file openssl.cnf can be found in the OpenSSL directory. This command displays OpenSSL directory

```
$openssl version -d
```

Add the following lines to your configuration file. This line has to be placed in the beginning of the configuration before any square brackets:

```
openssl_conf = openssl_init
```

This should be added at the end of the file:

```
[openssl_init]
engines = engine_section

[engine_section]
pkcs11 = pkcs11_section

[pkcs11_section]
engine_id = pkcs11
MODULE_PATH = /usr/lib/opensc-pkcs11.so
init = 0
```

Eventually adjust the value for MODULE_PATH according to your installation. The successful installation can be tested with this command

```
$openssl engine pkcs11 -t -c –vvvv
(pkcs11) pkcs11 engine
 [RSA, rsaEncryption, id-ecPublicKey]
     [ available ]
     SO_PATH: Specifies the path to the 'pkcs11' engine shared library
          (input flags): STRING
     MODULE_PATH: Specifies the path to the PKCS#11 module shared library
          (input flags): STRING
     PIN: Specifies the pin code
          (input flags): STRING
     VERBOSE: Print additional details
          (input flags): NO_INPUT
     QUIET: Remove additional details
          (input flags): NO_INPUT
     LOAD_CERT_CTRL: Get the certificate from card
          (input flags): [Internal]
     INIT_ARGS: Specifies additional initialization arguments to the PKCS#11
module
          (input flags): STRING
     SET_USER_INTERFACE: Set the global user interface (internal)
          (input flags): [Internal]
     SET_CALLBACK_DATA: Set the global user interface extra data (internal)
          (input flags): [Internal]
     FORCE_LOGIN: Force login to the PKCS#11 module
          (input flags): NO_INPUT
```

For more details on the PKCS#11 module configuration, see https://github.com/OpenSC/libp11#pkcs-11-module-configuration.

3. AWS IoT Greengrass Device Installation with iShield HSM

Follow the AWS IoT Greengrass developer guide to set up your Greengrass core device with private key and certificate in the HSM. For more details on hardware security integration with Greengrass, see https://docs.aws.amazon.com/greengrass/v2/developerguide/hardware-security.html. When installing the AWS IoT Greengrass Core software, for instance, with manual providing, create the AWS resources for your Greengrass core device in your AWS account, download the AWS IoT Greengrass Core software, PKCS#11 provider component and Amazon root CA certificate to your device, configure the hardware security module usage and start the installation. Following the Greengrass developer guide, create your device certificate from a private key generated on your iShield HSM and import the device certificate into the HSM as described in this chapter. If you prefer to use a GUI application instead of the command line, ch. 4 decribes the usage of the xca tool for all required PKCS#11 API operations. Follow ch. 4 instead of the subsections 3.2 3.3 and 3.5 during the setup of your Greengrass core device.

3.1 Initialize iShield HSM

Initialize your iShield HSM with the pkcs15-init tool from /usr/bin:

```
$./pkcs15-init --create-pkcs15
```

You will be asked to set a PIN and PUK. The PIN is required for further operations on the created token and can be reset with the help of the PUK.

The successful initialization can be verified with the pkcs11-tool:

```
$./pkcs11-tool --module ../lib/opensc-pkcs11.so -L
Available slots:
Slot 0 (0x0): Swissbit Secure USB PU-50n SE/PE (99000010EF000288) 01 00
  token label        : JavaCard isoApplet (User PIN)
  token manufacturer : unknown
  token model        : PKCS#15
  token flags        : login required, rng, token initialized, PIN initialized
  hardware version   : 0.0
  firmware version   : 0.0
  serial num         : 0000
  pin min/max        : 4/16
```

3.2 Create a key pair

Create a public private key pair on the token with the pkcs11-tool. Both, RSA and elliptic curve key pairs are supported. To generate a 2048-Bit RSA key pair with object id 1 and label `greengrass` run the following command:

```
$./pkcs11-tool --module ../lib/opensc-pkcs11.so --keypairgen --id "1" --label
"greengrass" --key-type rsa:2048 --login
Using slot 0 with a present token (0x0)
Logging in to "JavaCard isoApplet (User PIN)".
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
  label:      greengrass
  ID:         01
  Usage:      decrypt, sign
  Access:     none
Public Key Object; RSA 2048 bits
  label:      greengrass
  ID:         01
  Usage:      encrypt, verify
  Access:     none
```

3.3 Create a certificate signing request

Create a certificate signing request for the key pair:

```
$openssl req -engine pkcs11 -keyform engine -subj "/CN=greengrass/" -key "1" -
new -sha256 -out greengrass.csr
```

3.4 Register Device Certificate with AWS IoT

Your CSR can now be submitted to a Certicate Authority (CA) for signing. You can use your own CA or sign your CSR with the Amazon Root certificate authority in the AWS IoT Console or AWS Command Line Interface. For signing with the AWS IoT Console, log into your AWS account, navigate to the AWS IoT Console and go to `Secure` -> `Certificates` in the left side AWS IoT pane. Press `Add certificate` -> `Create certificate`, choose `Create certificate with certificate signing request (CSR)`, press `Choose file` and upload the .csr file that you just created (e.g. `greengrass.csr`). Select the value `Active` as `Certificate status` and press `Create`.

Add the certificate to the IoT Thing that you created for representing your device in the AWS IoT Cloud and attach the IoT Policy defining the device's permissions to the certificate.

3.5 Import X.509 certificate

Download the certificate from your AWS account to your device and load it into the token using the same id and label as for the key pair:

```
$./pkcs11-tool      --module      ../lib/opensc-pkcs11.so      --write-object
certificate.pem.crt --type cert --id "1" --label "greengrass" --login
```

3.6 Configure AWS IoT Greengrass Device with HSI

The AWS IoT Greengrass core device needs to be configured to use the HSM through the PKCS#11 interface. Create a config.yaml file in your Greengrass installer directory, copy the following content into the file and configure system parameters, the Greengrass nucleus and PKCS#11 provider.

```
---
system:
  certificateFilePath: "pkcs11:object=<Object Label>;type=cert"
  privateKeyPath: "pkcs11:object=<Object Label>;type=private"
  rootCaPath: "<Greengrass root folder>/AmazonRootCA1.pem"
  rootpath: "<Greengrass root folder>"
  thingName: "<IoT Thing name>"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "<Greengrass core software version>"
    configuration:
      awsRegion: "<AWS region with AWS device resources>"
      iotRoleAlias: "<token exchange role alias name>"
      iotDataEndpoint: "<AWS IoT data endpoint>"
      iotCredEndpoint: "<AWS IoT credentials endpoint>"
  aws.greengrass.crypto.Pkcs11Provider:
    configuration:
      name: "<Provider Name>"
      library: "<Provider Library>"
      slot: <Token Id>
      userPin: "<Token PIN>"
```

Fill with the missing information according to your settings and installation:

- <Object Label>: the label of your device certificate and private key, e.g. `greengrass`
- <Provider Name>: name of the PKCS#11 provider library, e.g. `opensc-pkcs11`
- <Provider Library>: absolute path of the PKCS#11 provider library, e.g. `/usr/lib/opensc-pkcs11.so`
- <Token Id>: id of the token containing your device's certificate and private key, e.g. 0
- <Token PIN>: pin that you set during token initialization

For more details on the configuration options, see the AWS IoT Greengrass Core software installation manual, Greengrass nucleus component configuration and PKCS#11 provider component configuration.

Now, you can run the installer and start AWS IoT Greengrass with HSI on your device.

4. PKCS#11 API Operations using XCA for AWS IoT Greengrass HIS

After successful initialization of your iShield HSM [3.1], all PKCS#11 API operation for Greengrass can be done either programmatically from the command line as shown in ch. 3 or you can use xca. xca is a GUI application managing X.509 certificates and cryptographic keys. The usage of PKCS#11 API allows xca to create key pairs on the iShield HSM, create certificate signing requests using protected key material and copy certificates to the device. xca is distributed as source code for Linux and can be downloaded from Sourceforge or you can install xca with the help of a package manager system with the command

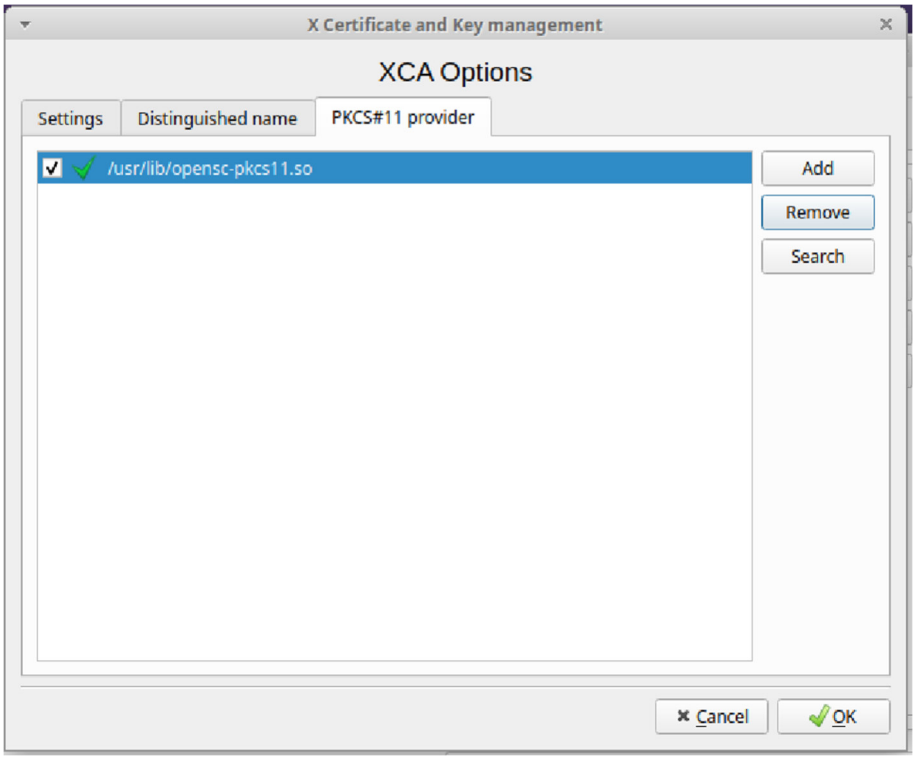
```
$sudo apt install xca
```

4.1 Create an Initial XCA Database

Start xca and create an initial database at `File` -> `New DataBase`. You will be asked to set a PIN for the database. When restarting xca, you can open your database at `File` -> `Open DataBase` with your assigned PIN.

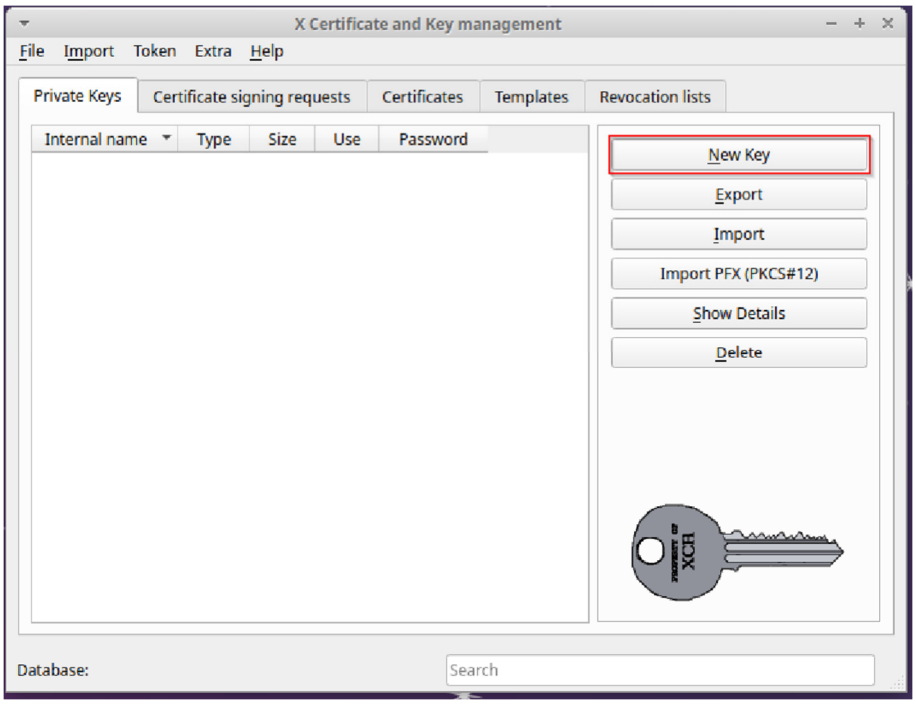
4.2 Configure PKCS#11 Provider

Open the xca options dialog with `File` -> `Options`, add your PKCS#11 provider at the respective tab and then press `OK`.

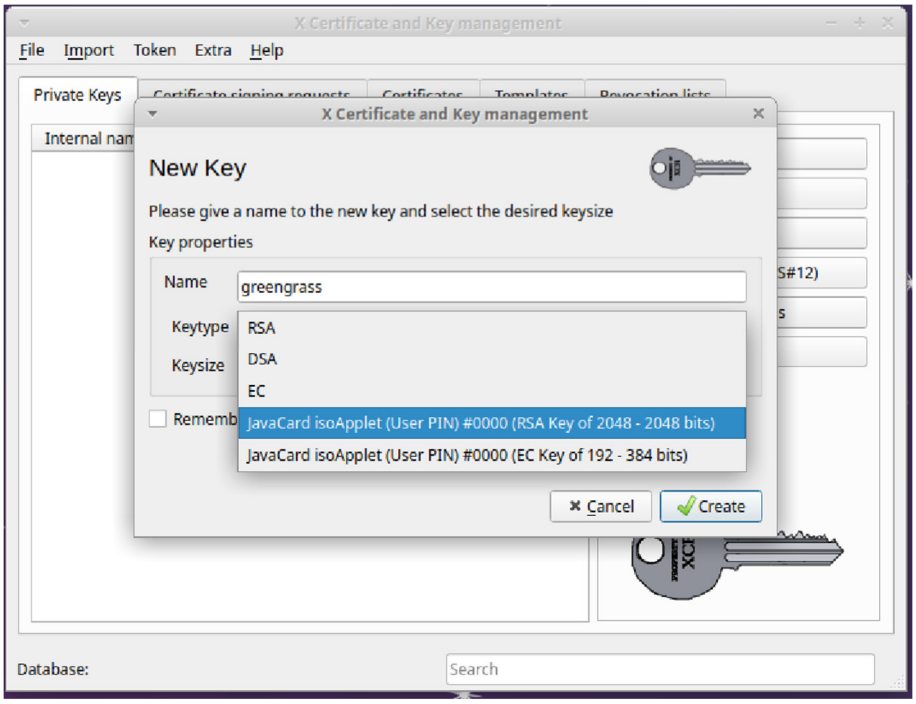


4.3 Create a key pair

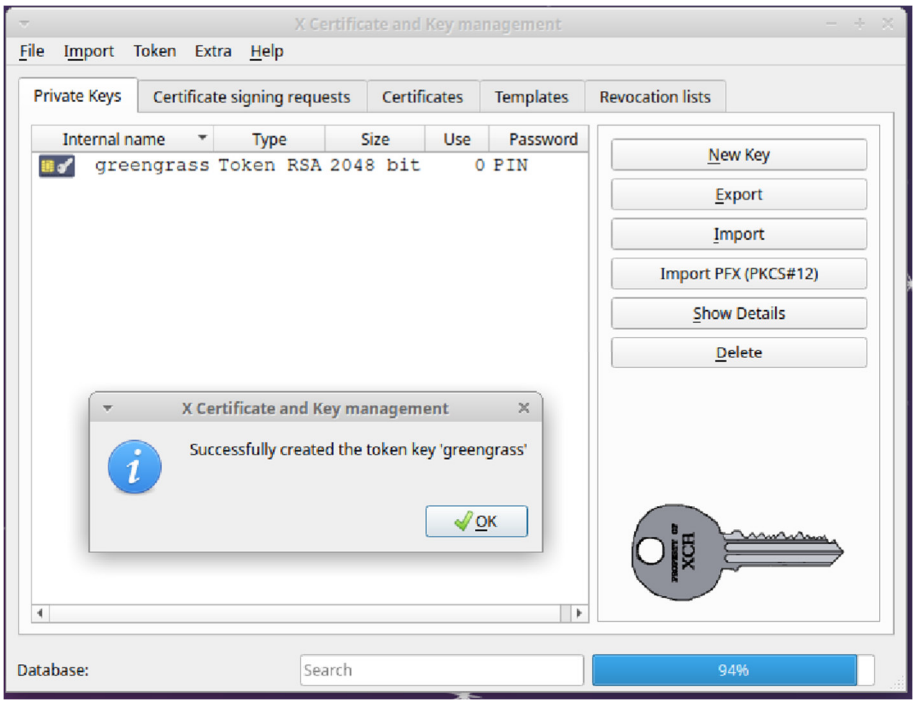
To create a new key pair on the token, open the `Private Keys` tab and press the `New Key` button.



In the `New Key` dialog choose a name for the new key pair. Select the key type `JavaCard isoApplet (User PIN) #0000 (RSA Key of 2048 – 2048 bits)` to create a 2048-Bit RSA key pair and close the dialog with the `Create` button.

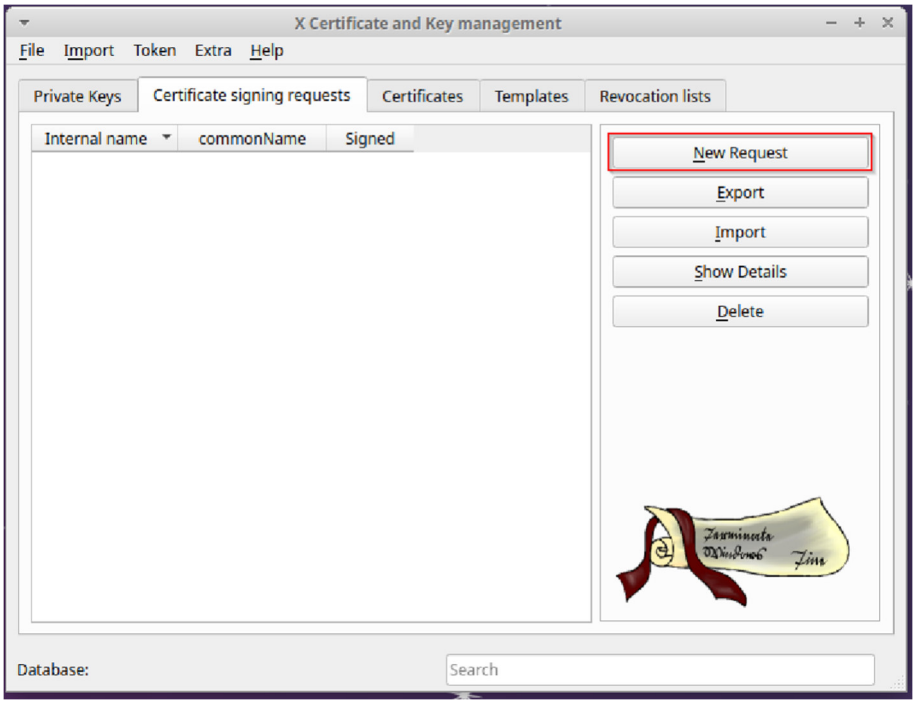


You will be prompted to enter the PIN for your token. After successful key creation, the key is listed in the xca application.

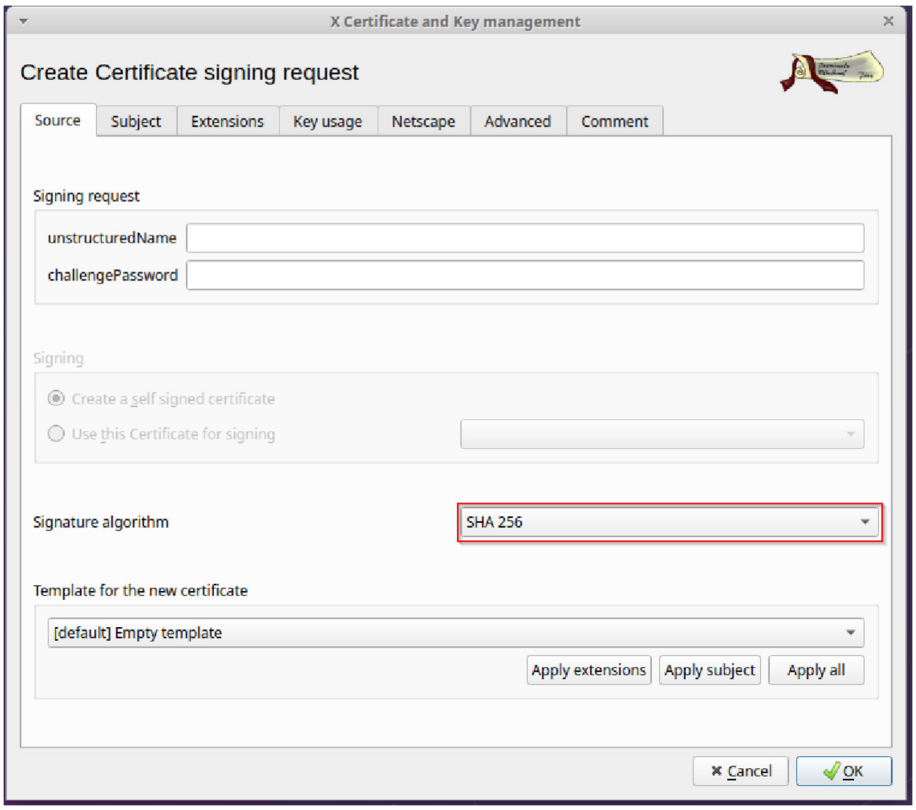


4.4 Create a certificate signing request

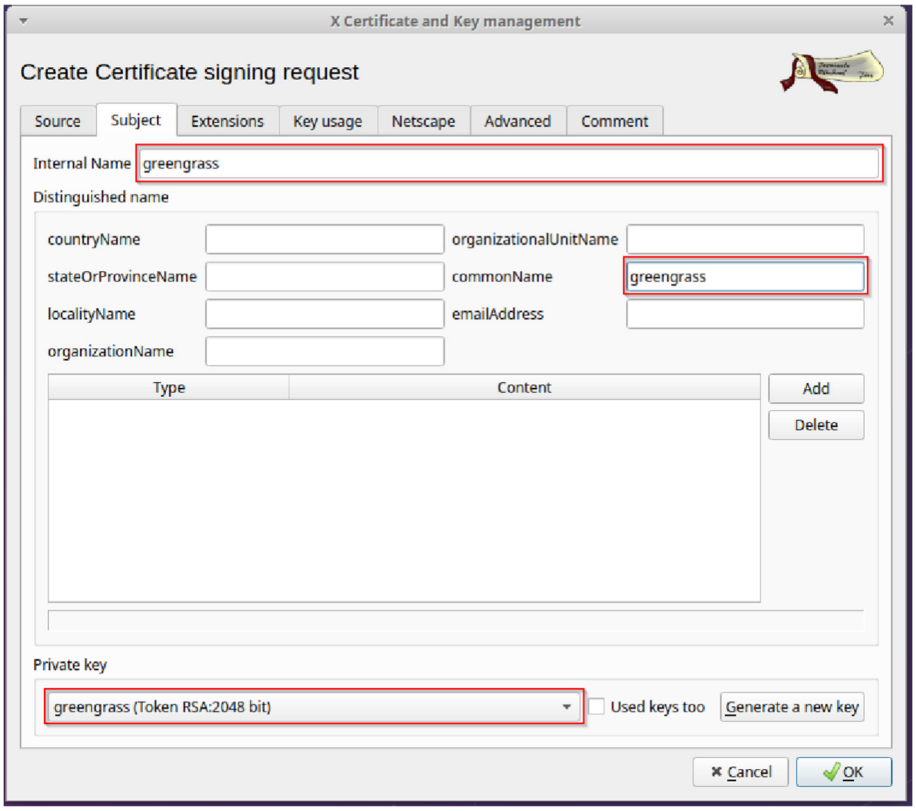
Go to the `Certificate signing requests` tab and choose `New Request` to create a CSR from xca.



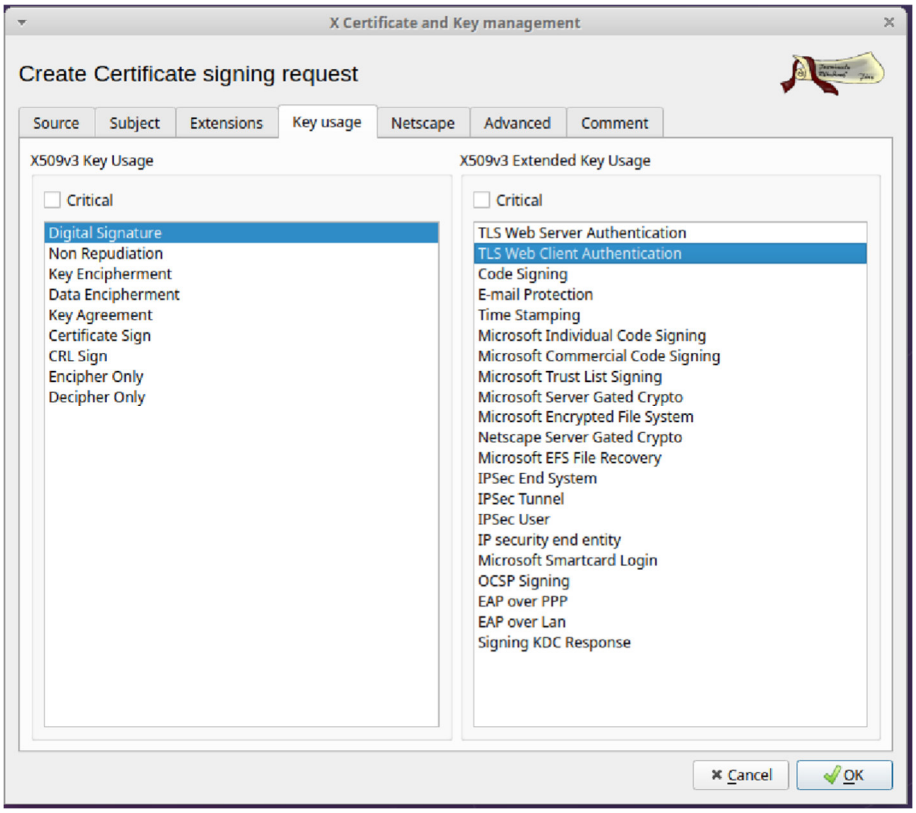
Select `SHA 256` as signature algorithm in the `Source` tab of the `Create Certificate signing request` dialog.



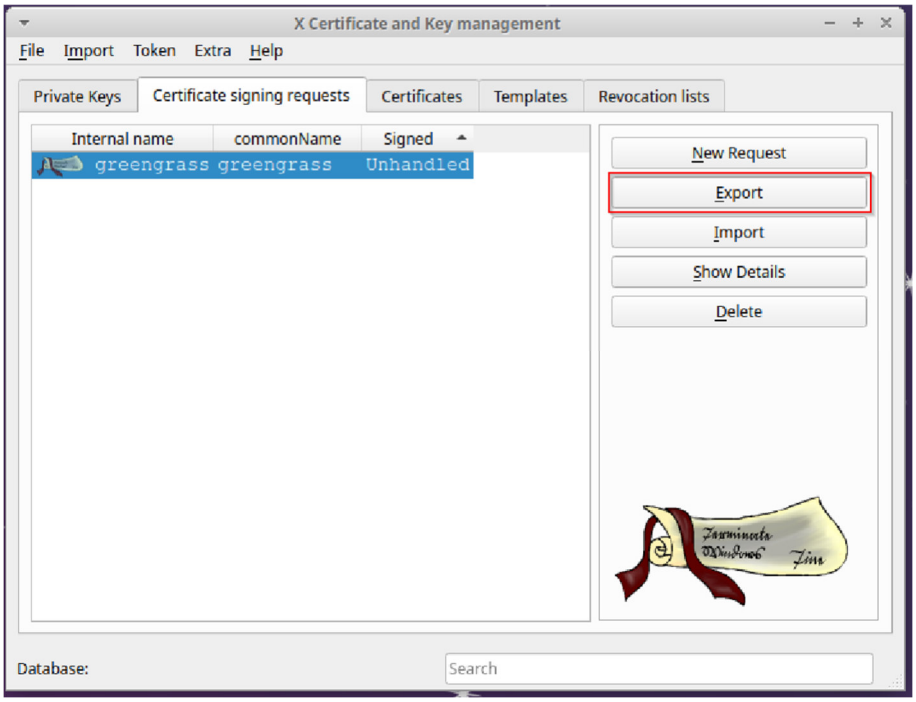
Set an internal name and common name in the `Subject` tab and choose the private key for which you want to create the CSR.



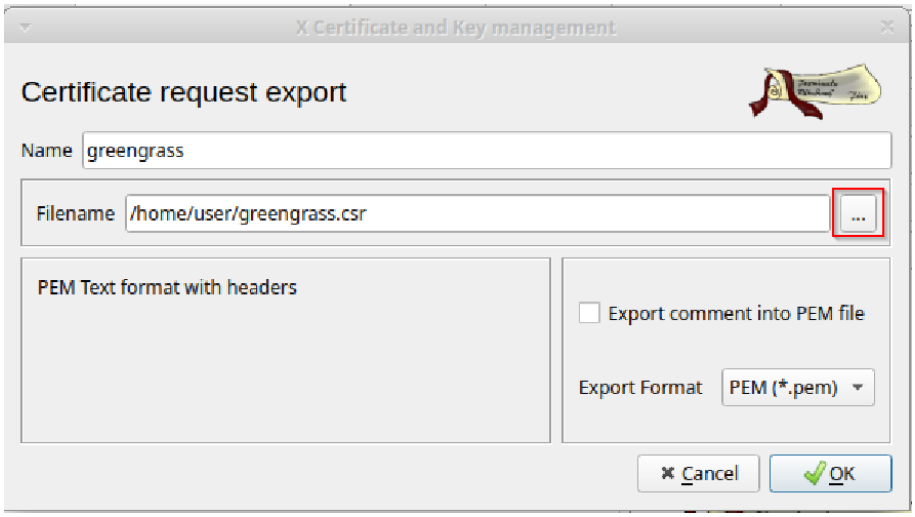
In the tab `Key Usage` select `Digital Signature`and `TLS Web Client Authentication`. Close the dialog with OK and enter the PIN for your token to create the CSR in the xca data base.



Export the CSR to a file to be able to submit it to your CA. In the tab `Certificate signing requests` select the newly created CSR and press the button `Export`.

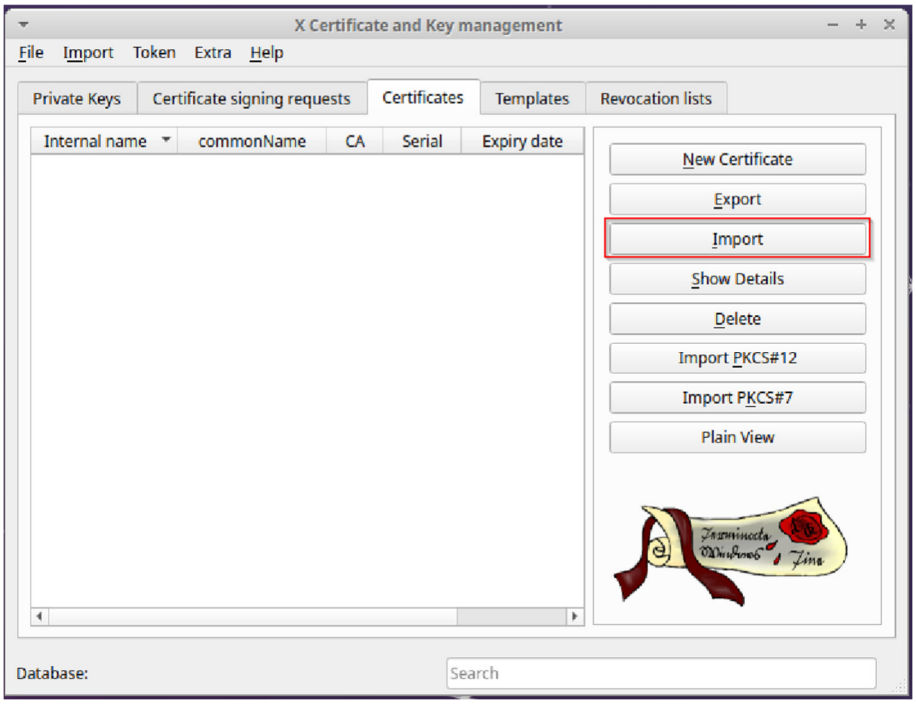


Choose your export file location and make sure to save your CSR in *.csr format.

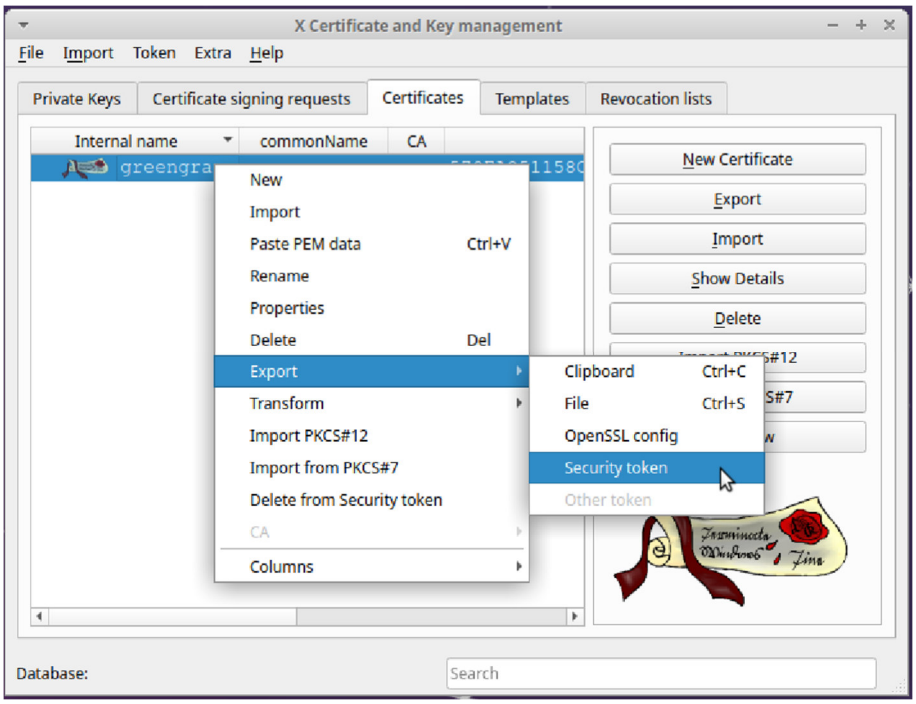


4.5 Import X.509 certificate

After you received the X.509 certificate from your CSR, load the certificate into the xca database: In the tab `Certificates`, click the button `Import` and select your certificate file.



The certificate is now listed in xca and can be imported into the token. Select it with the right mouse click to open the context menu. Select `Export` -> `Security token` and enter your token PIN when prompted.


X Certificate and Key management
File Import Token Extra Help
Private Keys
Certificate signing requests
Certificates
Templates
Revocation lists
Internal name
commonName
CA
New Certificate
Export
Import
Show Details
Delete
New
Import
Paste PEM data Ctrl+V
Rename
Properties
Delete Del
Export
Transform
Import PKCS#12
Import from PKCS#7
Delete from Security token
CA
Columns
Clipboard Ctrl+C
File Ctrl+S
OpenSSL config
Security token
Other token
Database:
Search

5. Appendix

5.1 Links

5.1.1 Swissbit Product Landing Page with Downloads

https://www.swissbit.com/en/products/security-products/ishield-hsm/

5.1.2 PCSC-Lite and Tools Documentation

https://pcsclite.apdu.fr/
http://ludovic.rousseau.free.fr/softwares/pcsc-tools/

5.1.3 CCID Documentation

https://ccid.apdu.fr/

5.1.4 libusb Homepage

https://libusb.info/

5.1.5 OpenSSL Documentation

https://www.openssl.org/

5.1.6 OpenSC Documentation and Download

https://github.com/OpenSC/OpenSC/wiki
https://github.com/OpenSC/OpenSC/archive/refs/tags/0.22.0.tar.gz

5.1.7 libp11 Download and Engine Configuration

https://github.com/OpenSC/libp11
https://github.com/OpenSC/libp11#pkcs-11-module-configuration

5.1.8 Software Packages for HSM Setup for AWS IoT Greengrass

https://www.swissbit.com/ishield-hsm/#tab-9-6900

5.1.9 AWS IoT Greengrass Device Guide References

https://docs.aws.amazon.com/greengrass/v2/developerguide/setting-up.html
https://docs.aws.amazon.com/greengrass/v2/developerguide/hardware-security.html
https://docs.aws.amazon.com/greengrass/v2/developerguide/manual-installation.html
https://docs.aws.amazon.com/greengrass/v2/developerguide/install-greengrass-core-v2.html
https://docs.aws.amazon.com/greengrass/v2/developerguide/greengrass-nucleus-component.html#greengrass-nucleus-component-configuration
https://docs.aws.amazon.com/greengrass/v2/developerguide/pkcs11-provider-component.html#pkcs11-provider-component-configuration

5.1.10 XCA Application Download

https://sourceforge.net/projects/xca/

6. Document History

Version	Updated on	Updated by	Short description
1.0	May 13th, 2022	Swissbit AG	First draft
1.1	May 18th, 2022	Swissbit AG	Updated Links