Code-Signing Tool – HSM

User's Guide

Rev. 3.0.1

May 2018

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	24/05/2018	Initial Work

USER'S GUIDE

TABLE OF CONTENTS

Page #

1.0 GENERAL INFORMATION

This document provides the information necessary for the user to effectively use Code-Signing Tool with Hardware Security Module backend. It is primarily intended for users who are familiar with CST tool to sign codes for the NXP High Assurance Boot (HAB).

A. INSTALLATION

1.1 Dependencies

The CST-HSM backend depends on:

- OpenSSL 1.0.2x (required by the Frontend)
- Libconfig

Make is required for building the software.

1) Install OpenSSL

```
$ apt-get install openssl libssl-dev
```

libconfig is C/C++ library for processing structured configuration files. It is used by CST to load HSM related configuration.

2) Install libconfig

```
$ apt-get install libconfig-dev
```

1.2 Prebuilt binaries

Pre-compiled CST and HSM backend can be found on the delivered package.

Unzip the CST-HSM package.

```
$ unzip cst-hsm-3.0.1.zip
```

For Linux 32-bits cst binary can be found under cst-hsm-3.0.1/release/linux32/bin
For Linux 64-bits cst binary can be found under cst-hsm-3.0.1/release/linux64/bin

1.3 Compile source

To compile backend and CST from souce:

1.3.1 Compile backend

Compile the backend source code using the following command:

```
$ make
```

Options:
OPENSSL_PATH Specify prefix of path of OpenSSL

1.3.1 Compile Code-Signing tool

Compile CST using the following command:

```
$ make all
```

2.0 DEPLOY Code-Signing Tool

B. DEPOLY

This section provides instructions and prerequisites to deploy Code-Signing Tool with 2 HSMs, these are SoftHSM2 and Utimaco HSM simulator.

2.1 Install HSM

2.1.1 SoftHSM2

Install SoftHSM2

```
  $ apt-get install softhsm2
```

Use either softhsm2-util to initialize a token. The SO PIN can e.g. be used to re-initialize the token and the user PIN is handed out to the application so it can interact with the token.

```
  $ softhsm2-util --init-token --slot 0 --label "CST-HSM"
```

Type in SO PIN and user PIN.

```
=== SO PIN (4-255 characters) ===
Please enter SO PIN: ******
Please reenter SO PIN: ******
=== User PIN (4-255 characters) ===
Please enter user PIN: ******
Please reenter user PIN: ******
The token has been initialized and is reassigned to slot 66362367
```

Initialized tokens will be reassigned to another slot (based on the token serial number). It is recommended to find and interact with the token by searching for the token label or serial number in the slot list / token info.

2.1.2 Utimaco HSM Simulator

Download and extract SecurityServerEvaluation package.

```
$ cd SecurityServerEvaluation-
V4.20.0.4/Software/Linux/Simulator/sim5_linux/bin
$ ./cs_sim.sh
```

Copy cs_pkcs11_R2.cfg to /etc

```
$ cp /root/Desktop/SecurityServerEvaluation-V4.20.0.4/Software/Linux/x86-
64/Crypto_APIs/PKCS11_R2/sample/cs_pkcs11_R2.cfg /etc
```

Edit **/etc/cs_pkcs11_R2.cfg** configuration file and

- Uncomment Logpath = /tmp in [Global] section
- Add Device = 3001@127.0.0.1 to [CryptoServer] section
- Save the file

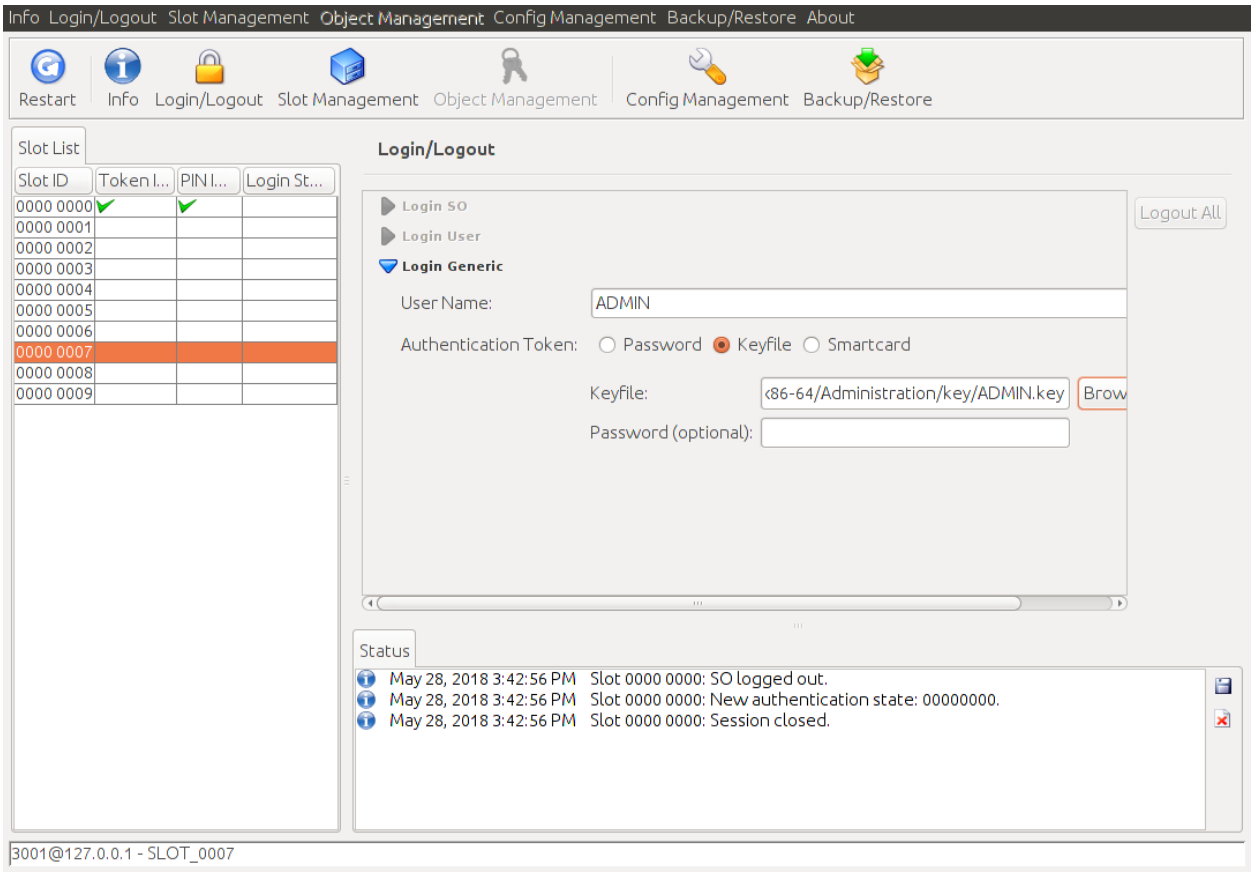
Install java if its is not installed on your system

```
$ apt-get install default-jre
```

Launch p11cat

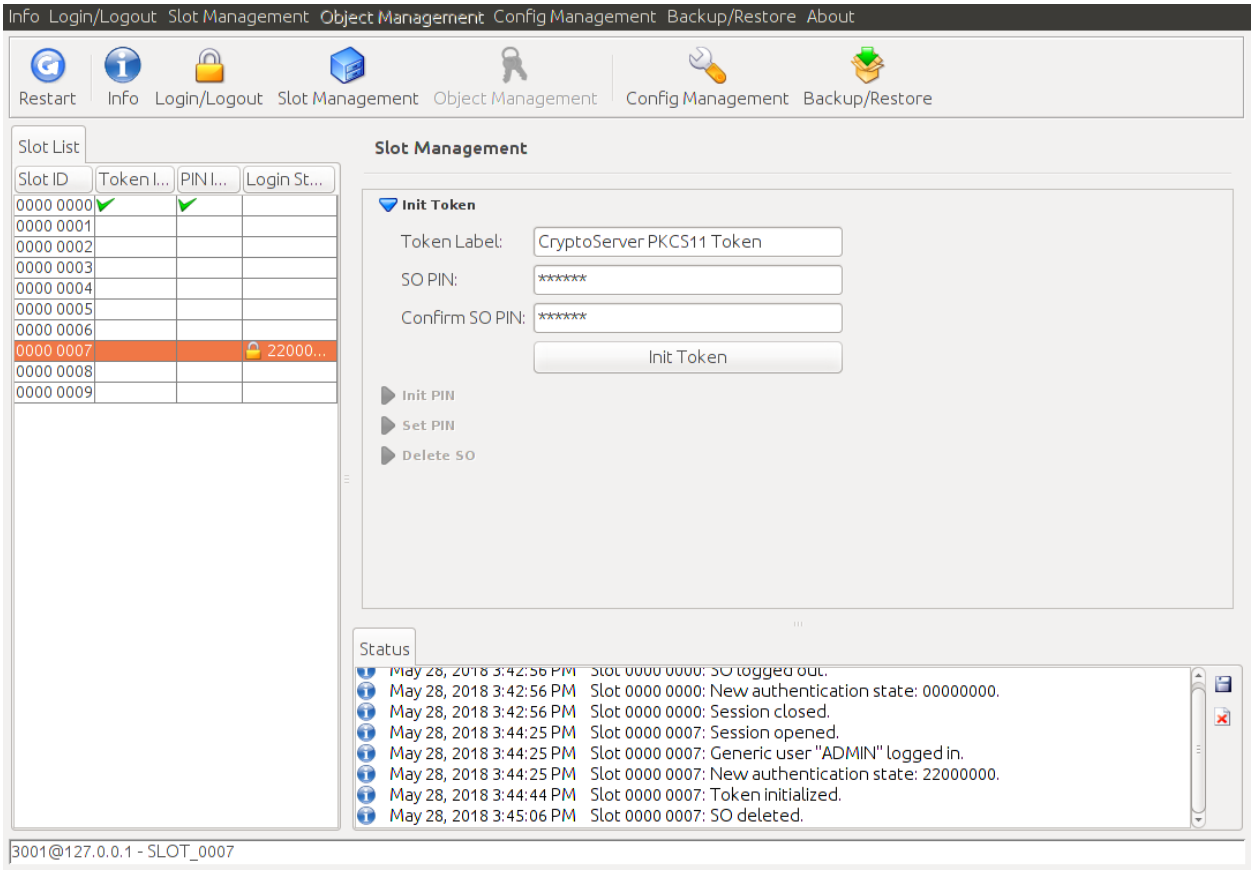
```
$ java -jar /root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-64/Administration/p11cat.jar
```

From the menu bar click **Login/Logout**
Select a slot from the **Slot List grid**
Select **Login Generic**
Type ***ADMIN*** as UserName
Select Keyfile as Authenticiation Token
Click browse button and fin the **ADMIN.key** file
The keyfile is under SecurityServerEvaluation-V4.20.0.4/Software/Linux/x86-64/Administration/key/ADMIN.key
Click Login

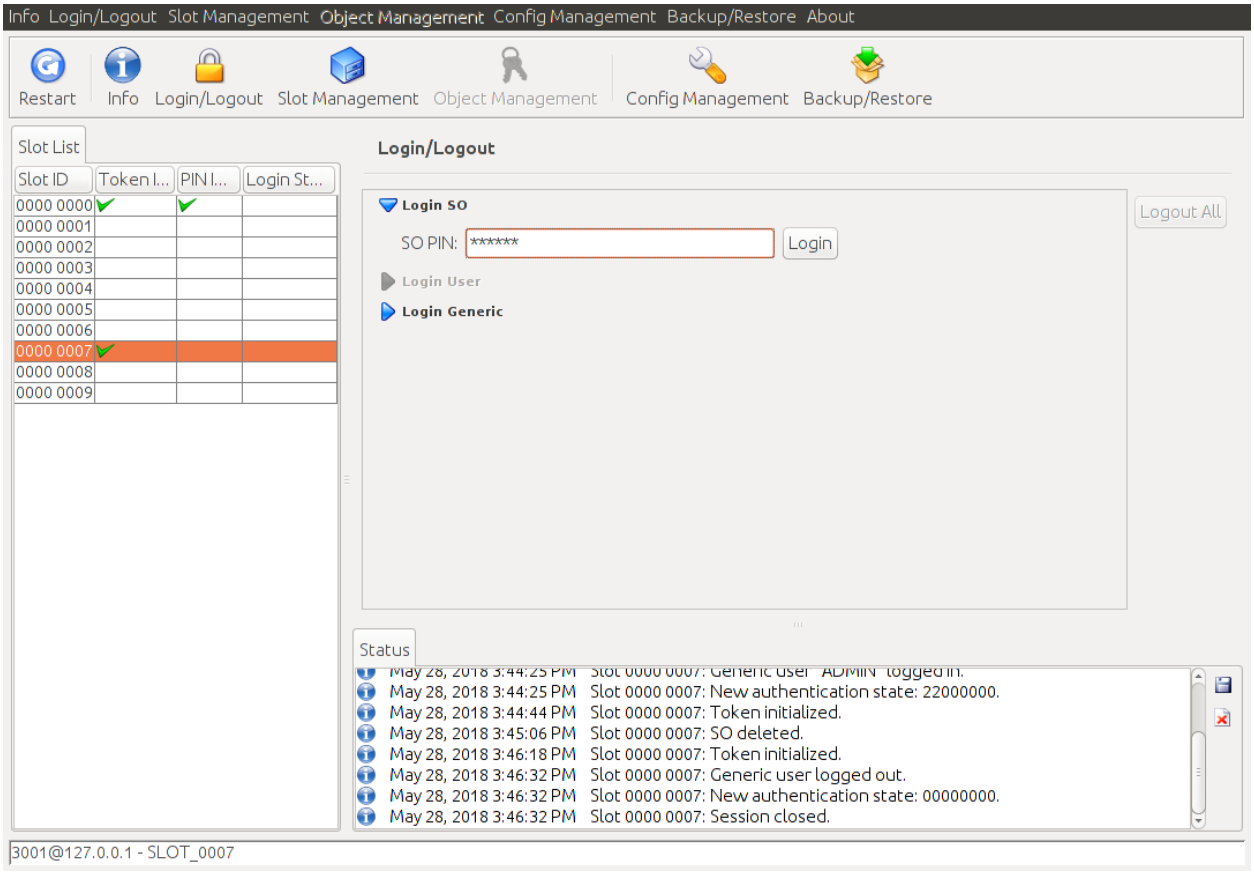


From **Slot Management** menu click

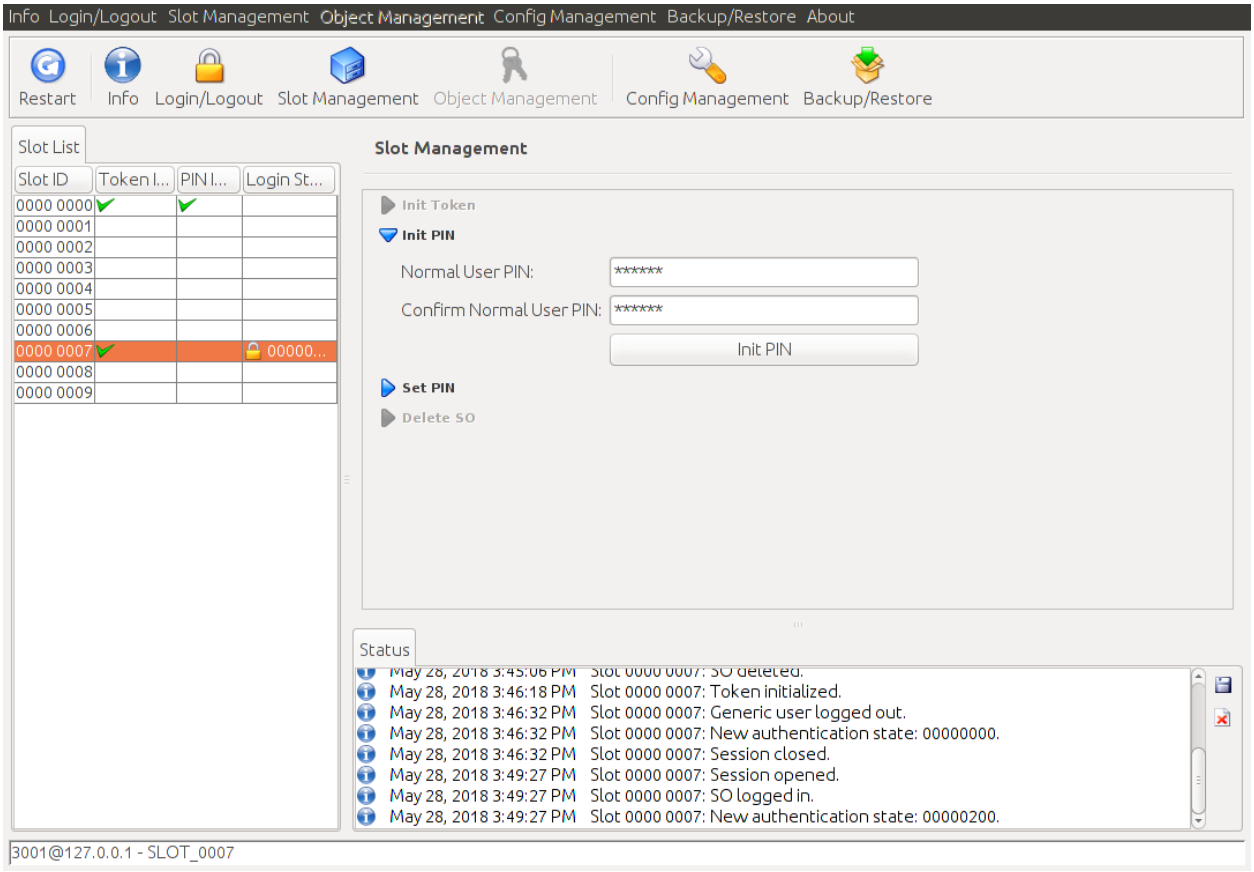
In **Init Token** form define the **Token Label** and **SO PIN** then click **Init Token**



From **Login/Logout** menu click **Logout All**
From **Login/Logout** menu click **Login SO**



From **Slot Management** menu click **Init PIN**
Set User PIN for example 123456



2.3 Create certificates & keys

Referring to **Key and Certificate Generation HAB Code Signing Tool User's guide**, generate keys and certificates which will be used to sign boot loader image.
If you already have the cryptographic material on the disk please go directly to *push cryptographic material to HSM* section

2.3.1 Create serial file

serial - 8-digit OpenSSL uses the contents of this file for the certificate serial numbers.

```
$ cd cst-hsm-3.0.1/release/keys
```

```
$ echo "28280581" > serial
```

2.3.2 Create passphrase file

key_pass.txt - Contains your pass phrase that will protect the HAB code signing private keys.

```
$ echo "maro1337" > key_pass.txt
$ echo " maro1337" >> key_pass.txt
```

2.3.3 Generate the PKI tree

```
$ ./hab4_pki_tree.sh
```

When prompts please answer according to the following:

```
Do you want to use an existing CA key (y/n)?: n
Do you want to use Elliptic Curve Cryptography (y/n)?: n
Enter key length in bits for PKI tree: 2048
Enter PKI tree duration (years): 10
How many Super Root Keys should be generated? 4
Do you want the SRK certificates to have the CA flag set? (y/n)?: y
```

Private keys will generate on keys directory and corresponding Certificates are placed in the crts directory.

2.3.4 Generate SRK table

```
cd ../crts
```

```
$ ../linux64/bin/srktool -h 4 -t SRK_1_2_3_4_table.bin -e
SRK_1_2_3_4_fuse.bin -d sha256 -c
./SRK1_sha256_2048_65537_v3_ca_crt.pem,./SRK2_sha256_2048_65537_v3_ca_crt.pem
,./SRK3_sha256_2048_65537_v3_ca_crt.pem,./SRK4_sha256_2048_65537_v3_ca_crt.pe
m -f 1
```

SRK_1_2_3_4_table.bin - SRK table contents with HAB data
SRK_1_2_3_4_fuse.bin - contains SHA256 result to be burned to fuse

2.3.4 Command Sequence File (CSF)

CST tool requires CSF script. This file describes certificates, keys and the data ranges used in sign and encryption functions. Create **u-boot.csf** file.

```
$ cd ../linux64/bin/
$ nano u-boot.csf
```

```
[Header]
Version = 4.1
Security Configuration = Open
Hash Algorithm = sha256
Engine Configuration = 0
Certificate Format = X509
Signature Format = CMS
Engine = CAAM

[Install SRK]
File = "../../crts/SRK_1_2_3_4_table.bin"
Source index = 0

[Install CSFK]
File = "../../crts/CSF1_1_sha256_4096_65537_v3_usr_crt.pem"

[Authenticate CSF]

[Install Key]
# Key slot index used to authenticate the key to be installed
Verification index = 0

# Key to install
Target index = 2
File = "../../crts/IMG1_1_sha256_4096_65537_v3_usr_crt.pem"

[Authenticate Data]
Verification index = 2
Blocks = 0x00000000 0 0xd4454 "flash.bin"
```

2.4 Push cryptographic material to HSM

2.4.1 Install p11tool

To perform interact with HSM you can use p11tool.

```
$ apt-get install gnutls-bin
```

2.4.2 Find Token URL

You need to locate your HSM vendor's PKCS#11 interface implementation in order to use it with p11tool and CST later.

Utimaco pkcs#11 library is *SecurityServerEvaluation-V4.20.0.4/Software/Linux/x86-64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so*

SoftHSM2 pkcs11# library is */usr/lib/softhsm/libsofthsm2.so*

Find the Token URL to interact with the token.

```
$ p11tool --provider <pkcs11_lib_path> --list-tokens
```

Example using SoftHSM2

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so --list-tokens
```

```
Token 0:
        URL:
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;serial=9c1aeb00e05a3
48b;token=CST-HSM
        Label: CST-HSM
        Type: Generic token
        Manufacturer: SoftHSM project
        Model: SoftHSM v2
        Serial: 9c1aeb00e05a348b
        Module: (null)
```

In this case Token URL is
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;serial=9c1aeb00e05a348b;token=CST-HSM

2.4.3 Push CSF certificate & key

To match a certificate to a private key, both should have the same ID, for some other HSM implementations they should have the same label.

Utimaco HSM Simulator

Private Key

```
$ p11tool --provider /root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so
"pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=UTIMACO%20
CS000000;token=CryptoServer%20PKCS11%20Token" so  --login --write --load-
privkey ../../keys/CSF1_1_sha256_2048_65537_v3_usr_key.pem --label
CSF1_1_sha256_2048_65537_v3_usr --id ec705018e9bf8ad60096e13cb2f0fbad
```

Certificate

```
$ p11tool --provider /root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so
"pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=UTIMACO%20
CS000000;token=CryptoServer%20PKCS11%20Token" so  --login --write --load-
certificate ../../crts/CSF1_1_sha256_2048_65537_v3_usr_crt.pem --label
CSF1_1_sha256_2048_65537_v3_usr --id ec705018e9bf8ad60096e13cb2f0fbad
```

SoftHSM2

Private Key

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so  --login --write
"<token-url>" --load-privkey
../../keys/CSF1_1_sha256_2048_65537_v3_usr_key.pem --label
CSF1_1_sha256_2048_65537_v3_usr --id ec705018e9bf8ad60096e13cb2f0fbad
```

Certificate

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so  --login --write
"<token-url>" --load-certificate
../../crts/CSF1_1_sha256_2048_65537_v3_usr_crt.pem --label
CSF1_1_sha256_2048_65537_v3_usr --id ec705018e9bf8ad60096e13cb2f0fbad
```

2.4.4 Push IMG certificate & key

Utimaco HSM Simulator

Private key

```
$ p11tool --provider /root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so
"pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=UTIMACO%20
CS000000;token=CryptoServer%20PKCS11%20Token" so  --login --write --load-
privkey ../../keys/IMG1_1_sha256_2048_65537_v3_usr_key.pem --label
IMG1_1_sha256_2048_65537_v3_usr --id a0c8cac03985fb6dced29c97dc83aef7
```

Certificate

```
$ p11tool --provider /root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so
"pkcs11:model=CryptoServer;manufacturer=Utimaco%20IS%20GmbH;serial=UTIMACO%20
CS000000;token=CryptoServer%20PKCS11%20Token" so  --login --write --load-
certificate ../../crts/IMG1_1_sha256_2048_65537_v3_usr_crt.pem --label
IMG1_1_sha256_2048_65537_v3_usr --id a0c8cac03985fb6dced29c97dc83aef7
```

SoftHSM2

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so  --login --write
"<token-url>"  --load-privkey
../../keys/IMG1_1_sha256_2048_65537_v3_usr_key.pem --label
IMG1_1_sha256_2048_65537_v3_usr --id a0c8cac03985fb6dced29c97dc83aef7
```

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so  --login --write
"<token-url>"  --load-certificate
```

```
../../crts/IMG1_1_sha256_2048_65537_v3_usr_crt.pem --label
IMG1_1_sha256_2048_65537_v3_usr --id a0c8cac03985fb6dced29c97dc83aef7
```

2.4.5 Verification

Verify that all cryptographic materials were written correctly to token by running.

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so "<token-url>"  --list-
all --login
```

Enter PIN when prompts

2.5 Pull certificates from HSM

2.5.1 Install p11tool

To perform interact with HSM you can use p11tool.

```
$ apt-get install gnutls-bin
```

2.5.1 Find Token URL

Find the Token URL to interact with the token.

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so --list-tokens
```

```
Token 0:
        URL:
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;serial=9c1aeb00e05a3
48b;token=CST-HSM
        Label: CST-HSM
        Type: Generic token
        Manufacturer: SoftHSM project
        Model: SoftHSM v2
        Serial: 9c1aeb00e05a348b
        Module: (null)
```

In this case Token URL is
pkcs11:model=SoftHSM%20v2;manufacturer=SoftHSM%20project;serial=9c1aeb00e05a348b;token=CST-HSM

2.4.3 Pull CSF & IMG certificate

List all certificates on HSM

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so "<token-url>"  --list-
all-certs
```

Pull CSF certificate from HSM

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so --export "<cert-url>" --
outfile CSF1_1_sha256_2048_65537_v3_ca_crt.pem
```

Pull IMG certificate from HSM

```
$ p11tool --provider /usr/lib/softhsm/libsofthsm2.so --export "<cert-url>" --
outfile IMG1_1_sha256_2048_65537_v3_ca_crt.pem
```

2.6 Create HSM configuration file

Create a **hsm.cfg** file in the same folder as CST. The configuration file specifies the following parameters.

- **hsm.module** contains the path to vendor's PKCS#11 implementation library
- **hsm.pin** specifies the User PIN
- **hsm.slot** specifies the SLOT ID
- **hsm.objects** is a lookup table which bind a cryptographic material on the file system to its Object ID on the HSM. This is needed since the current frontend implementation loads and checks certificates from the file system and then provide as argument the certificate path to the backend. The backend can see a path only. To be load the required certificate and its private key from the HSM, the backend uses this lookup.

```
$ nano hsm.cfg
```

```
# hsm stuff
hsm:
{
  module = "/root/Desktop/SecurityServerEvaluation-
V4.20.0.4/Software/Linux/x86-
64/Crypto_APIs/PKCS11_R2/lib/libcs_pkcs11_R2.so";
  pin = "123456";
  slot = 0;
  objects = (
          { file  = "../../crts/CSF1_1_sha256_2048_65537_v3_usr_crt.pem";
            id = "ec705018e9bf8ad60096e13cb2f0fbad";
          },
          { file  = "../../crts/IMG1_1_sha256_2048_65537_v3_usr_crt.pem";
            id = "a0c8cac03985fb6dced29c97dc83aef7";
          });
};
```

For SRK tool you may need to pull all SRK certificates.

C. RUN

Generate the CSF binary signature

```
$ ./cst -i u-boot.csf -o u-boot.csf.bin
CSF Processed successfully and signed data available in u-boot.csf.bin
```