Code-Signing Tool – HSM

Quick Start Guide

Rev. 3.0.1

May 2018

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	29/05/2018	Initial Work

USER'S GUIDE

TABLE OF CONTENTS

Page #

1.0 CST-HSM
Install
Configure
Run

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). This guide assumes that the HSM in installed, configured and that all cryptographic materials are stored on it.

A. INSTALL

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
```

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

3) 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

B. CONFIGURE

CST requires having certificates on file system. To fulfill the requirement, you should pull the certificates used for signing to your file system using a utility which manage and use PKCS #11 security tokens. You can use the tool provided by your HSM vendor or any other alternative. In this manual we use p11tool.

2.1 Install p11tool

To perform interact with HSM you can use p11tool.

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

2.2 Locate Token URL

1) Make sure that a token is initialized on your HSM.
2) Locate the PKCS #11 interface implementation library provided by your HSM vendor. It will be used for p11tool and later by CST.
   (e.g)
   Utimaco library is *libcs_pkcs11_R2.so*
   SoftHSM2 library is *libsofthsm2.so*
3) Find the Token URL to interact with the token.

```
$ p11tool --provider <pkcs11_provider_library_path> --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

2.2 Pull the public certificates

Pull all certificates referenced on your Command Sequence File. Assume that a CSF and an IMG certificates were referenced.

1) List all certificates on HSM

Locate certificate URL

```
$ p11tool --provider <pkcs11_provider_library_path> "<token-url>" --list-all-
certs
```

2) Pull CSF certificate from HSM

Export the certificate to file system using its URL.

```
$ p11tool --provider <pkcs11_provider_library_path> --export "<cert_url>" --
outfile CSF1_1_sha256_2048_65537_v3_ca_crt.pem
```

3) Pull IMG certificate from HSM

Export the certificate to file system using its URL.

```
$ p11tool --provider <pkcs11_provider_library_path> --export "<cert_url>" --
outfile IMG1_1_sha256_2048_65537_v3_ca_crt.pem
```

2.3 Binding

In the same folder as CST, create a **hsm.cfg** file. This file is used by CST to determine parameters to access HSM and to locate cryptographic objects needed for signing.
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, for security reason you may not specify this value, it will be asked at runtime.
- **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.
    - **hsm.objects.file** path to certificate on file system.
    - **hsm.objects.id** Identifier of the same certificate on the HSM.
      Please note that this Id will be used also to load the private key corresponding to the certificate. HSM binds certificates to private keys by giving the same Id for both objects and in some HSM implementation they should have the same Label also.

```
$ nano hsm.cfg
```

```
# hsm stuff
hsm:
{
  module = <pkcs11_provider_library_path> ";
  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";
          });
};
```

C. RUN

Finally invoke the CST.

```
$ ./cst -i <infile> -o <outfile>
CSF Processed successfully and signed data available in <outfile>
```