OPTIGA™ TPM Application Note

PKCS #11

Devices

- OPTIGA™ TPM SLB 9670 TPM2.0
- OPTIGA™ TPM SLI 9670 TPM2.0
- OPTIGA™ TPM SLM 9670 TPM2.0

About This Document

Scope and purpose

This document explains how an OPTIGA™ TPM SLx 9670 TPM2.0 can be integrated into a Raspberry Pi® to create a TPM-based PKCS #11 cryptographic token.

PKCS #11 is a Public-Key Cryptography Standard that defines a standard platform-independent API to access cryptographic services from tokens, such as hardware security modules (HSM) and smart cards. This document provides guidance on how to setup a TPM-based token on a Raspberry Pi®.

The OPTIGA™ TPM SLx 9670 TPM2.0 uses a SPI interface to communicate with the Raspberry Pi®. The OPTIGA™ TPM SLx 9670 TPM2.0 product family with SPI interface consists of 3 different products:

- OPTIGA™ TPM SLB 9670 TPM2.0 standard security applications
- OPTIGA™ TPM SLI 9670 TPM2.0 automotive security applications
- OPTIGA™ TPM SLM 9670 TPM2.0 industrial security applications

OPTIGA™ TPM SLx 9670 TPM2.0 products are fully TCG compliant TPM products with CC (EAL4+) and FIPS certification. The OPTIGA™ TPM SLx 9670 TPM2.0 products standard, automotive, and industrial differ with regards to supported temperature range, lifetime, quality grades, test environment, qualification, and reliability to fit the target applications requirements. An overview of all Infineon OPTIGA™ TPM products can be found on Infineon's website [2][3]. More information on TPM specification can be found on Trusted Computing Group (TCG) in reference [4].

Intended audience

This document is intended for customers who want to increase the security level of their platforms using a TPM 2.0 and like to evaluate the implementation of TPM-based PKCS #11 cryptographic token for their target applications.

Table of contents

List of figures

List of tables

Acronyms and Abbreviations

Acronym	Definition
API	Application Programming Interface
CSR	Certificate Signing Request
ECC	Elliptic Curve Cryptography
RSA	Rivest–Shamir–Adleman
SO	A Security Officer user
TPM	Trusted Platform Module

1 Prepare Raspberry Pi®

This section describes all the steps necessary for building a Raspberry Pi® bootable SD card image.

1.1 Prerequisites

- Raspberry Pi® 4
- Flash the Raspberry Pi® OS image (2021-01-11 release from [5]) on a micro-SD card (≥8GB)
- OPTIGA™ TPM (TPM2.0)
    - SLB 9670
    - SLI 9670
    - SLM 9670

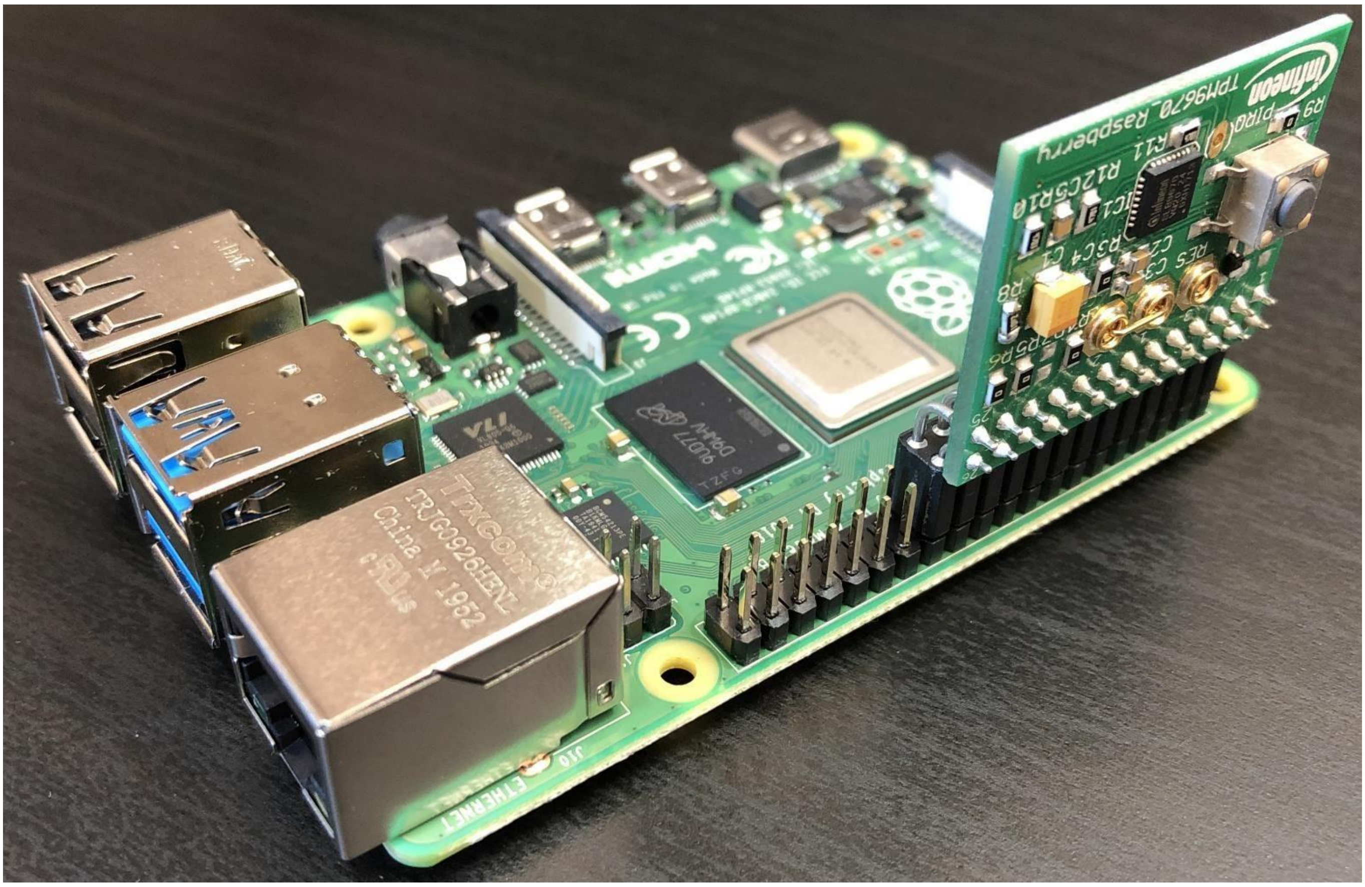

Figure 1 Infineon Iridium SLx 9670 TPM2.0 SPI Board on Raspberry Pi® 4

1.2 Enable TPM

Insert the flashed SD card and boot the Raspberry Pi®.

Open the configuration file in an editor:

Code Listing 1

```
$ sudo nano /boot/config.txt
```

Insert the following lines to enable SPI and TPM.

Code Listing 2

```
dtoverlay=tpm-slb9670
```

Save the file and exit the editor.

Reboot the Raspberry Pi® and check if TPM is activated.

Code Listing 3

```
$ ls /dev | grep tpm
tpm0
tpmrm0
```

1.3 Install TPM Software

Install the following software on the Raspberry Pi®:

Table 1 TPM 2.0 software

Software	Link	Version
tpm2-tss	https://github.com/tpm2-software/tpm2-tss	3.0.3
tpm2-tools	https://github.com/tpm2-software/tpm2-tools	5.0
tpm2-abrmd	https://github.com/tpm2-software/tpm2-abrmd	2.3.3
tpm2-pkcs11	https://github.com/tpm2-software/tpm2-pkcs11	1.5.0

Install dependencies:

Code Listing 4

```
$ sudo apt update
$ sudo apt -y install autoconf-archive libcmocka0 libcmocka-
  dev procps iproute2 build-essential git pkg-config gcc
  libtool automake libssl-dev uthash-dev autoconf doxygen
  libgcrypt-dev libjson-c-dev libcurl4-gnutls-dev uuid-dev
  pandoc libglib2.0-dev libsqlite3-dev libyaml-dev
```

First time Git setup, insert your username and email.

Code Listing 5

```
$ git config --global user.name "your name"
$ git config --global user.email your-email@example.com
```

Download the Git repository pkcs11-optiga-tpm [1].

Code Listing 6

```
$ cd ~
$ git clone https://github.com/Infineon/pkcs11-optiga-tpm
```

Install TPM software stack:

Code Listing 7

```
$ cd ~
$ git clone https://github.com/tpm2-software/tpm2-tss.git
$ cd tpm2-tss
$ git checkout 3.0.3
$ ./bootstrap
$ ./configure
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

Install TPM tools:

Code Listing 8

```
$ cd ~
$ git clone https://github.com/tpm2-software/tpm2-tools.git
$ cd tpm2-tools
$ git checkout 5.0
$ ./bootstrap
$ ./configure
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

Install TPM access broker & resource manager:

Code Listing 9

```
$ cd ~
$ git clone https://github.com/tpm2-software/tpm2-abrmd.git
$ cd tpm2-abrmd
$ git checkout 2.3.3
$ ./bootstrap
$ ./configure --with-dbuspolicydir=/etc/dbus-1/system.d
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

Configure D-Bus:

Code Listing 10

```
$ sudo useradd --system --user-group tss
$ sudo pkill -HUP dbus-daemon
$ sudo systemctl daemon-reload
```

Allow TPM device node to be accessed by tpm2-abrmd user 'tss'. Take note that this effect is not persistent.

Code Listing 11

```
$ sudo chown tss /dev/tpm0
```

To verify that D-Bus is configured correctly:

Code Listing 12

```
$ tpm2_clear -T tabrmd:bus_name=com.intel.tss2.Tabrmd -c p
```

Install TPM PKCS #11. Apply the patch "support-existing-TPM2-persistent-objects.patch" for section 2.2.4 to work.

Code Listing 13

```
$ cd ~
$ git clone https://github.com/tpm2-software/tpm2-pkcs11.git
$ cd tpm2-pkcs11
$ git checkout 1.5.0
$ git am ~/pkcs11-optiga-tpm/patches/interoperability-with-
  existing-TPM2-persistent-objec.patch
$ ./bootstrap
$ ./configure --disable-fapi
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

1.4 Setup Python3

Set Python 3.7 as default:

Code Listing 14

```
$ sudo rm /usr/bin/python
$ sudo ln -s python3.7 /usr/bin/python
```

To verify Python is set correctly:

Code Listing 15

```
$ python -V
Python 3.7.3
```

Install Python libraries:

Code Listing 16

```
$ pip3 install pyyaml
$ pip3 install pyasn1-modules
```

1.5 Install PKCS #11 Software

Install dependencies:

Code Listing 17

```
$ sudo apt install libpcsclite-dev
```

Install OpenSC:

Code Listing 18

```
$ cd ~
$ git clone https://github.com/OpenSC/OpenSC.git
$ cd OpenSC
$ git checkout 0.21.0
$ ./bootstrap
$ ./configure
$ make -j$(nproc)
```

Code Listing 18

```
$ sudo make install
$ sudo ldconfig
```

Install OpenSSL PKCS #11 engine:

Code Listing 19

```
$ cd ~
$ git clone https://github.com/OpenSC/libp11.git
$ cd libp11
$ git checkout libp11-0.4.11
$ ./bootstrap
$ ./configure
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

Check if the engine pkcs11.so is correctly installed in /usr/lib/arm-linux-gnueabihf/engines-1.1/.

2 Operation Guide

This section describes how OpenSC and OpenSSL can be used to interact with TPM-based PKCS #11 token.

The OpenSC software dependencies:

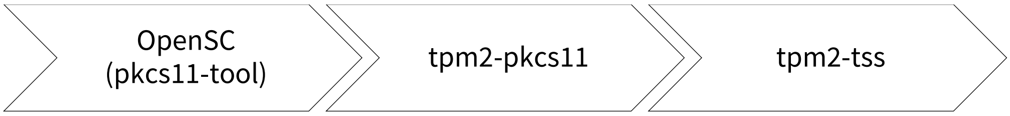

Figure 2 tpm2pkcs11-tool software dependencies

The OpenSSL software dependencies:

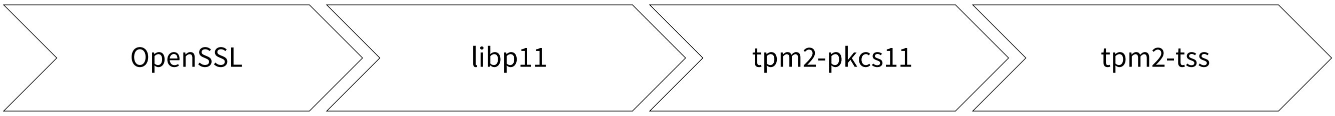

Figure 3 tpm2ssl software dependencies

2.1 Environment Setup

Make a copy of the sample configuration file (~/tpm2-pkcs11/misc/tpm2-pkcs11.openssl.sample.conf) and place it at ~/tpm2-pkcs11.openssl.conf. Update the engine path and tpm2-pkcs11 library path in the file:

- dynamic_path = /usr/lib/arm-linux-gnueabihf/engines-1.1/pkcs11.so
- MODULE_PATH = /usr/local/lib/libtpm2_pkcs11.so

Set abbreviations. Remember to update the path:

Code Listing 20

```
$ alias tpm2pkcs11-tool='pkcs11-tool --module
  /usr/local/lib/libtpm2_pkcs11.so'
$ alias tpm2ssl='OPENSSL_CONF=~/tpm2-pkcs11.openssl.conf
  openssl'
$ alias tpm2_ptool='~/tpm2-pkcs11/tools/tpm2_ptool.py'
```

Set environment variable:

Code Listing 21

```
$ mkdir ~/pkcs11-store
$ export TPM2_PKCS11_STORE=~/pkcs11-store
```

2.2 PKCS #11 Token Creation

There are two ways of creating a PKCS #11 token, to create a token from a blank TPM, or to create a token and link it with existing TPM objects. Find more information at [6].

2.2.1 Start Blank

Reset the TPM:

Code Listing 22

```
$ tpm2_clear -c p
```

Initialize a store at path ~/pkcs11-store and provision the TPM:

Code Listing 23

```
$ tpm2_ptool init --path ~/pkcs11-store
```

Create a TPM-based PKCS #11 token:

Code Listing 24

```
$ tpm2_ptool addtoken --pid 1 --sopin sopin --userpin userpin --label tpm-token --path ~/pkcs11-store
```

2.2.2 Link Existing (Keys stored outside of a TPM)

Option 1: Provision the TPM owner hierarchy:

Code Listing 25

```
$ tpm2_clear -c p
$ tpm2_createprimary -G ecc -c primary.ctx
$ tpm2_evictcontrol -c primary.ctx 0x81000001
$ tpm2_create -G rsa2048 -C 0x81000001 -u rsakey.pub -r rsakey.priv
$ tpm2_create -G ecc -C 0x81000001 -u ecckey.pub -r ecckey.priv
```

Option 2: Provision the TPM platform hierarchy:

Code Listing 26

```
$ tpm2_clear -c p
$ tpm2_createprimary -C p -G ecc -c primary.ctx
$ tpm2_evictcontrol -C p -c primary.ctx 0x81800001
$ tpm2_create -G rsa2048 -C 0x81800001 -u rsakey.pub -r rsakey.priv
$ tpm2_create -G ecc -C 0x81800001 -u ecckey.pub -r ecckey.priv
```

If platform hierarchy is used, the command tpm2_clear is not able to remove platform persistent handles. Instead, use the following commands:

Code Listing 27

```
$ tpm2_evictcontrol -C p -c 0x81800001
```

Create a TPM-based PKCS #11 token associated with the TPM primary key. Example given here is using the primary key from the owner hierarchy, for the platform hierarchy change the handle to 0x81800001:

Code Listing 28

```
$ tpm2_ptool init --primary-handle 0x81000001 --path=~/pkcs11-store
$ tpm2_ptool addtoken --pid 1 --sopin sopin --userpin userpin --label tpm-token --path ~/pkcs11-store
```

Link existing TPM objects (RSA and ECC key objects):

Code Listing 29

```
$ tpm2_ptool link --label tpm-token --id 0 --key-label linkrsa2048 --userpin userpin --path ~/pkcs11-store rsakey.pub rsakey.priv
$ tpm2_ptool link --label tpm-token --id 1 --key-label linkeccp256 --userpin userpin --path ~/pkcs11-store ecckey.pub ecckey.priv
```

Show linked key objects:

Code Listing 30

```
$ tpm2pkcs11-tool --slot 1 --list-objects --login --pin userpin
```

Reset the token:

Code Listing 31

```
$ rm ~/pkcs11-store/tpm2_pkcs11.sqlite3
```

2.2.3 Link Existing (Keys stored in a TPM NV area)

Option 1: Provision TPM owner hierarchy:

Code Listing 32

```
$ tpm2_clear -c p
$ tpm2_createprimary -G ecc -c primary.ctx
$ tpm2_evictcontrol -c primary.ctx 0x81000001

### Create RSA keypair and store it in NV
$ tpm2_create -G rsa2048 -C 0x81000001 -u rsakey.pub -r rsakey.priv
$ tpm2_nvdefine -C o 0x1000000 -s `cat rsakey.priv | wc -c` -a "ownerread|ownerwrite"
$ tpm2_nvwrite -C o 0x1000000 -i rsakey.priv
$ tpm2_nvdefine -C o 0x1000001 -s `cat rsakey.pub | wc -c` -a "ownerread|ownerwrite"
$ tpm2_nvwrite -C o 0x1000001 -i rsakey.pub

### Create ECC keypair and store it in NV
$ tpm2_create -G ecc -C 0x81000001 -u ecckey.pub -r ecckey.priv
$ tpm2_nvdefine -C o 0x1000002 -s `cat ecckey.priv | wc -c` -a "ownerread|ownerwrite"
$ tpm2_nvwrite -C o 0x1000002 -i ecckey.priv
$ tpm2_nvdefine -C o 0x1000003 -s `cat ecckey.pub | wc -c` -a "ownerread|ownerwrite"
$ tpm2_nvwrite -C o 0x1000003 -i ecckey.pub
```

Option 2: Provision TPM platform hierarchy:

Code Listing 33

```
$ tpm2_clear -c p
$ tpm2_createprimary -C p -G ecc -c primary.ctx
$ tpm2_evictcontrol -C p -c primary.ctx 0x81800001

### Create RSA keypair and store it in NV
$ tpm2_create -G rsa2048 -C 0x81800001 -u rsakey.pub -r
  rsakey.priv
$ tpm2_nvdefine -C p 0x1000000 -s `cat rsakey.priv | wc -c`
  -a "platformcreate|ppread|ppwrite"
$ tpm2_nvwrite -C p 0x1000000 -i rsakey.priv
$ tpm2_nvdefine -C p 0x1000001 -s `cat rsakey.pub | wc -c` -
  a "platformcreate|ppread|ppwrite"
$ tpm2_nvwrite -C p 0x1000001 -i rsakey.pub

### Create ECC keypair and store it in NV
$ tpm2_create -G ecc -C 0x81800001 -u ecckey.pub -r
  ecckey.priv
$ tpm2_nvdefine -C p 0x1000002 -s `cat ecckey.priv | wc -c`
  -a "platformcreate|ppread|ppwrite"
$ tpm2_nvwrite -C p 0x1000002 -i ecckey.priv
$ tpm2_nvdefine -C p 0x1000003 -s `cat ecckey.pub | wc -c` -
  a "platformcreate|ppread|ppwrite"
$ tpm2_nvwrite -C p 0x1000003 -i ecckey.pub
```

If platform hierarchy is used, the command “tpm2_clear” is not able to remove platform persistent handles. Instead, use the following commands:

Code Listing 34

```
$ tpm2_evictcontrol -C p -c 0x81800001
```

Create a TPM-based PKCS #11 token by associating it with the primary key. Example given here is using the primary key from the owner hierarchy, for the platform hierarchy change the handle to 0x81800001:

Code Listing 35

```
$ tpm2_ptool init --primary-handle 0x81000001 --path=~/pkcs11-
  store
$ tpm2_ptool addtoken --pid 1 --sopin sopin --userpin userpin
  --label tpm-token --path ~/pkcs11-store
```

Link existing TPM objects (RSA and ECC key objects); for the platform hierarchy replace the parameter “-C o” with “-C p”:

Code Listing 36

```
### Read RSA keypair from NV and link it to the PKCS #11 token
$ tpm2_nvread -C o 0x1000000 -o rsakey.priv
$ tpm2_nvread -C o 0x1000001 -o rsakey.pub
$ tpm2_ptool link --label tpm-token --id 0 --key-label
  linkrsa2048 --userpin userpin --path ~/pkcs11-store
  rsakey.pub rsakey.priv
$ tpm2_nvundefine -C o 0x1000000
$ tpm2_nvundefine -C o 0x1000001
```

Code Listing 36

```

### Read ECC keypair from NV and link it to the PKCS #11 token
$ tpm2_nvread -C o 0x1000002 -o ecckey.priv
$ tpm2_nvread -C o 0x1000003 -o ecckey.pub
$ tpm2_ptool link --label tpm-token --id 1 --key-label linkeccp256 --userpin userpin --path ~/pkcs11-store ecckey.pub ecckey.priv
$ tpm2_nvundefine -C o 0x1000002
$ tpm2_nvundefine -C o 0x1000003
```

Show linked key objects:

Code Listing 37

```
$ tpm2pkcs11-tool --slot 1 --list-objects --login --pin userpin
```

Reset the token:

Code Listing 38

```
$ rm ~/pkcs11-store/tpm2_pkcs11.sqlite3
```

2.2.4 Link Existing (Key persisted in a TPM)

Provision TPM owner hierarchy:

Code Listing 39

```
$ tpm2_clear -c p
$ tpm2_createprimary -G ecc -c primary.ctx
$ tpm2_evictcontrol -c primary.ctx 0x81000001
$ tpm2_create -G rsa2048 -C 0x81000001 -u rsakey.pub -r rsakey.priv -p keyauth
$ tpm2_load -C 0x81000001 -u rsakey.pub -r rsakey.priv -c rsakey.ctx
$ tpm2_evictcontrol -c rsakey.ctx 0x81000002
$ tpm2_create -G ecc -C 0x81000001 -u ecckey.pub -r ecckey.priv
$ tpm2_load -C 0x81000001 -u ecckey.pub -r ecckey.priv -c ecckey.ctx
$ tpm2_evictcontrol -c ecckey.ctx 0x81000003
```

Create a TPM-based PKCS #11 token associated with the TPM primary key:

Code Listing 40

```
$ tpm2_ptool init --primary-handle 0x81000001 --path=~/pkcs11-store
$ tpm2_ptool addtoken --pid 1 --sopin sopin --userpin userpin --label tpm-token --path ~/pkcs11-store
```

Link existing TPM objects (RSA and ECC key objects):

Code Listing 41

```
$ tpm2_ptool link-persist --label tpm-token --id 0 --key-label linkrsa2048 --userpin userpin --path ~/pkcs11-store 0x81000002 --auth keyauth
$ tpm2_ptool link-persist --label tpm-token --id 1 --key-label linkecc2048 --userpin userpin --path ~/pkcs11-store 0x81000003
```

Show linked key objects:

Code Listing 42

```
$ tpm2pkcs11-tool --slot 1 --list-objects --login --pin userpin
```

Reset the token:

Code Listing 43

```
$ rm ~/pkcs11-store/tpm2_pkcs11.sqlite3
```

2.3 OpenSC

For simplicity, follow section 2.2.1 to initialize a PKCS #11 token before continuing.

Show available token:

Code Listing 44

```
$ tpm2pkcs11-tool --list-token-slots
```

Change user pin (from “userpin” to “upin”):

Code Listing 45

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --change-pin --new-pin upin
```

Change user pin with SO pin (from “upin” to “userpin”):

Code Listing 46

```
$ tpm2pkcs11-tool --slot 1 --init-pin --so-pin sopin --pin userpin
```

Show supported key types:

Code Listing 47

```
$ tpm2pkcs11-tool --slot 1 --list-mechanisms
```

Create an RSA key object:

Code Listing 48

```
$ tpm2pkcs11-tool --slot 1 --id 00 --label rsa2048 --login --pin userpin --keypairgen --key-type RSA:2048
```

Create an ECC key object:

Code Listing 49

```
$ tpm2pkcs11-tool --slot 1 --id 01 --label eccp256 --login --pin userpin --keypairgen --usage-sign --key-type EC:secp256r1
```

Show created key objects:

Code Listing 50

```
$ tpm2pkcs11-tool --slot 1 --list-objects --login --pin userpin
```

Read the public component of the RSA key:

Code Listing 51

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --id 00 --type pubkey --read-object > rsa.pub.der
$ openssl rsa -inform DER -outform PEM -in rsa.pub.der -pubin > rsa.pub.pem
```

Read the public component of the ECC key:

Code Listing 52

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --id 01 --type pubkey --read-object > ecc.pub.der
$ openssl ec -inform DER -outform PEM -in ecc.pub.der -pubin > ecc.pub.pem
```

Generate random data:

Code Listing 53

```
$ tpm2pkcs11-tool --slot 1 --generate-random 32 > data
```

RSA encryption and decryption:

Code Listing 54

```
$ openssl rsautl -encrypt -inkey rsa.pub.pem -in data -pubin -out data.crypt
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --id 00 --decrypt --mechanism RSA-PKCS --input-file data.crypt --output-file data.plain
$ diff data data.plain
```

RSA signing and verification:

Code Listing 55

```
$ tpm2pkcs11-tool --slot 1 --id 00 --login --pin userpin --sign --mechanism SHA256-RSA-PKCS --input-file data --output-file data.rsa.sig
$ openssl dgst -sha256 -verify rsa.pub.pem -signature data.rsa.sig data
```

ECC signing and verification:

Code Listing 56

```
$ tpm2pkcs11-tool --slot 1 --id 01 --login --pin userpin --sign --mechanism ECDSA-SHA1 --signature-format openssl --input-file data --output-file data.ecc.sig
$ openssl dgst -sha1 -verify ecc.pub.pem -signature data.ecc.sig data
```

Destroy RSA key object:

Code Listing 57

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --delete-object --type privkey --id 00
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --delete-object --type pubkey --id 00
```

Destroy ECC key object:

Code Listing 58

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --delete-object --type privkey --id 01
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --delete-object --type pubkey --id 01
```

2.4 OpenSSL

To verify that the PKCS #11 engine is accessible:

Code Listing 59

```
$ openssl version
OpenSSL 1.1.1d  10 Sep 2019
$ openssl engine pkcs11 -t
(pkcs11) pkcs11 engine
     [ available ]
```

Create RSA and ECC key objects:

Code Listing 60

```
$ tpm2pkcs11-tool --slot 1 --id 02 --label osslrsa2048 --login --pin userpin --keypairgen --key-type RSA:2048
$ tpm2pkcs11-tool --slot 1 --id 03 --label ossleccp256 --login --pin userpin --keypairgen --usage-sign --key-type EC:secp256r1
```

RSA signing and verification:

Code Listing 61

```
$ echo "beefcafe" > data
$ tpm2ssl dgst -engine pkcs11 -keyform engine -sign "pkcs11:token=tpm-token;object=osslrsa2048;pin-value=userpin" -out data.rsa.sig data
$ tpm2ssl dgst -engine pkcs11 -keyform engine -verify "pkcs11:token=tpm-token;object=osslrsa2048;pin-value=userpin" -signature data.rsa.sig data
```

ECC signing and verification:

Code Listing 62

```
$ echo "beefcafe" > data
$ tpm2ssl dgst -engine pkcs11 -keyform engine -sign "pkcs11:token=tpm-token;object=ossleccp256;pin-value=userpin" -out data.ecc.sig data
$ tpm2ssl dgst -engine pkcs11 -keyform engine -verify "pkcs11:token=tpm-token;object=ossleccp256;pin-value=userpin" -signature data.ecc.sig data
```

RSA encryption and decryption:

Code Listing 63

```
$ echo "beefcafe" > data
$ tpm2ssl rsautl -engine pkcs11 -keyform engine -inkey "pkcs11:token=tpm-token;object=osslrsa2048;pin-value=userpin" -encrypt -in data -out data.crypt
$ tpm2ssl rsautl -engine pkcs11 -keyform engine -inkey "pkcs11:token=tpm-token;object=osslrsa2048;pin-value=userpin" -decrypt -in data.crypt -out data.plain
$ diff data data.plain
```

Generate a self-signed certificate:

Code Listing 64

```
$ tpm2ssl req -engine pkcs11 -keyform engine -key "pkcs11:token=tpm-token;object=ossleccp256;pin-value=userpin" -new -x509 -days 365 -subj '/CN=TPM CA/' -sha256 -out ca.crt.pem
### view the certificate
$ openssl x509 -in ca.crt.pem -text -noout
```

Import a certificate:

Code Listing 65

```
$ tpm2_ptool addcert --label=tpm-token --key-label=ossleccp256 --path ~/pkcs11-store ca.crt.pem
```

Show imported certificates:

Code Listing 66

```
$ tpm2pkcs11-tool --slot 1 --list-objects --login --pin userpin
```

Read a certificate, use the Code Listing 66 to get the object id.

Code Listing 67

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --read-object --type cert --id <object id> cert.der
$ openssl x509 -inform der -in cert.der -out cert.pem
$ diff cert.pem ca.crt.pem
```

Delete a certificate, use the Code Listing 66 to get the object id.

Code Listing 68

```
$ tpm2pkcs11-tool --slot 1 --login --pin userpin --delete-object --type cert --id <object id>
```

Generate a CSR:

Code Listing 69

```
$ tpm2ssl req -engine pkcs11 -keyform engine -key "pkcs11:token=tpm-token;object=ossleccp256;pin-value=userpin" -new -subj '/CN=TPM device/' -out csr.pem
### view the certificate
$ openssl req -in csr.pem -text -noout
```

Sign the CSR:

Code Listing 70

```
$ tpm2ssl x509 -engine pkcs11 -CAkeyform engine -CAkey "pkcs11:token=tpm-token;object=ossleccp256;pin-value=userpin" -req -CA ca.crt.pem -sha256 -set_serial 1 -in csr.pem -out crt.pem
### view the certificate
$ openssl x509 -in crt.pem -text -noout
```

References

[1] https://github.com/Infineon/pkcs11-optiga-tpm

[2] https://www.infineon.com/cms/en/product/evaluation-boards/iridium9670-tpm2.0-linux/

[3] http://www.infineon.com/tpm

[4] https://trustedcomputinggroup.org/resource/tpm-main-specification/

[5] https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2021-01-12/2021-01-11-raspios-buster-armhf.zip

[6] https://github.com/tpm2-software/tpm2-pkcs11/blob/master/docs/INTEROPERABILITY.md

Revision history

Reference	Description
Revision 1.0, 2021-02-29	
all	Initial version

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-02-09

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:
cssсustomerservice@infineon.com

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.