|  |  |  |  |
| --- | --- | --- | --- |
| NXP Semiconductors | Document Number: AN00000000000000 | | |
| Application Notes | Rev. | , |  |

LPC55Sxx PSA Secure Boot, Debug Authentication, Provisioning

Contents

Stanislav Jancik – Optional

[1. Introduction 1](#_Toc34657596)

[2. Software Requirements 2](#_Toc34657597)

[3. Secure Boot 2](#_Toc34657598)

[3.1. Private Key Generation 2](#_Toc34657599)

[3.2. Certification Signing Request 2](#_Toc34657600)

[3.3. Self-Signed Certificate 3](#_Toc34657601)

[3.4. Prepare Binary Files 3](#_Toc34657602)

[3.5. Signed Images Preparation 6](#_Toc34657603)

[3.6. Erasing a Device 8](#_Toc34657604)

[3.7. Loading Signed Images (only when PRINCE is not used) 8](#_Toc34657605)

[3.8. CFPA Page Preparation 8](#_Toc34657606)

[3.9. CMPA Page Preparation 9](#_Toc34657607)

[3.10. Flash Encryption Using PRINCE 10](#_Toc34657608)

[4. Debug Authentication 11](#_Toc34657609)

[5. Provisioning 11](#_Toc34657610)

# Introduction

The LPC55xx/LPC55Sxx is an Arm Cortex® M33-based microcontroller for embedded applications:

* App to 320KB of on-chip SRAM
* App to 640KB of on-chip flash
* High-speed and full-speed USB host
* Device interface with crystal-less operation for full-speed
* Five general-purpose timers
* One SCTimer/PWM
* One RTC/alarm timer
* One 24-bit Multi-Rate Timer (MRT)
* Windowed Watchdog Timer (WWDT)
* Eight flexible serial communication peripherals, each of it is a USART, SPI, I2C, or I2S interface
* One 16-bit 1.0 Msamples/sec ADC, temperature sensor

LPC55xx/LPC55Sxx devices offer support for real-time encryption and decryption for on-chip flash using the PRINCE encryption engine. The following sections

explain the use of LPC55Sxx PRINCE for data encryption of on-chip flash***.***

# Software Requirements

OpenSSL 1.1.1

* [Download](https://slproweb.com/products/Win32OpenSSL.html)

elftosb-gui, blhost

1. [Download](http://kex-stage.freescale.net/en/builder)
2. Board LPCxpresso55S69
3. Choose All toolchains or MCUXpresso IDE in Toolchain / IDA list
4. The checkbox mcu-boot has to be checked (use select all)
5. Generate SDK package
6. The elftosb-gui and blhost are located here: SDK\_2.7.0\_LPCXpresso55S69\middleware\mcu-boot\bin\Tools\
7. To run elftosb-gui use elfosb-gui(win).cmd script

MCUXpresso IDE v11.1.0\_3209 or later

Dokumentation: UM11126\_LPC556x\_v.1.7.pdf

AN12283.pdf

Attached files: CustomerManufacturingFactoryConfigurationArea\_0x9e400.bin

InFieldROMAPIScratch\_0x9de00.bin

# Secure Boot

## Private Key Generation

The four private keys need to be generated:

openssl genrsa -out rotk1.pem 4096

openssl genrsa -out rotk2.pem 4096

openssl genrsa -out rotk1\_2048.pem 2048

openssl genrsa -out rotk2\_2048.pem 2048

## Certification Signing Request

After creation of private keys, certificate signing request (csr) is created:

openssl req -new -key rotk1.pem -out certificate\_rotk1.csr -extensions v3\_ca

openssl req -new -key rotk2.pem -out certificate\_rotk2.csr -extensions v3\_ca

openssl req -new -key rotk1\_2048.pem -out certificate\_rotk1\_2048.csr -extensions v3\_ca

openssl req -new -key rotk2\_2048.pem -out certificate\_rotk2\_2048.csr -extensions v3\_ca

After this command, the user has to fill the following data:

Country Name:

State or Province Name:

Locality Name:

Organization Name:

Organization Unit Name:

Common Name:

Email address:

Password:

Company Name:

## Self-Signed Certificate

openssl x509 -req -days 365 -in certificate\_rotk1.csr -signkey rotk1.pem -sha256 -outform der -out certificate\_rotk1.der.crt -extfile x509\_v3.ext -set\_serial 0x3cc30000abababab

openssl x509 -req -days 365 -in certificate\_rotk2.csr -signkey rotk2.pem -sha256 -outform der -out certificate\_rotk2.der.crt -extfile x509\_v3.ext -set\_serial 0x3cc30000abababab

openssl x509 -req -days 365 -in certificate\_rotk1\_2048.csr -signkey rotk1\_2048.pem -sha256 -outform der -out certificate\_rotk1\_2048.der.crt -extfile x509\_v3.ext -set\_serial 0x3cc30000abababab

openssl x509 -req -days 365 -in certificate\_rotk2\_2048.csr -signkey rotk2\_2048.pem -sha256 -outform der -out certificate\_rotk2\_2048.der.crt -extfile x509\_v3.ext -set\_serial 0x3cc30000abababab

The x509\_v3.ext file has to be created to follow x509v3\_config -X509 V3 certificate extension configuration format. In this case, the file should contain:

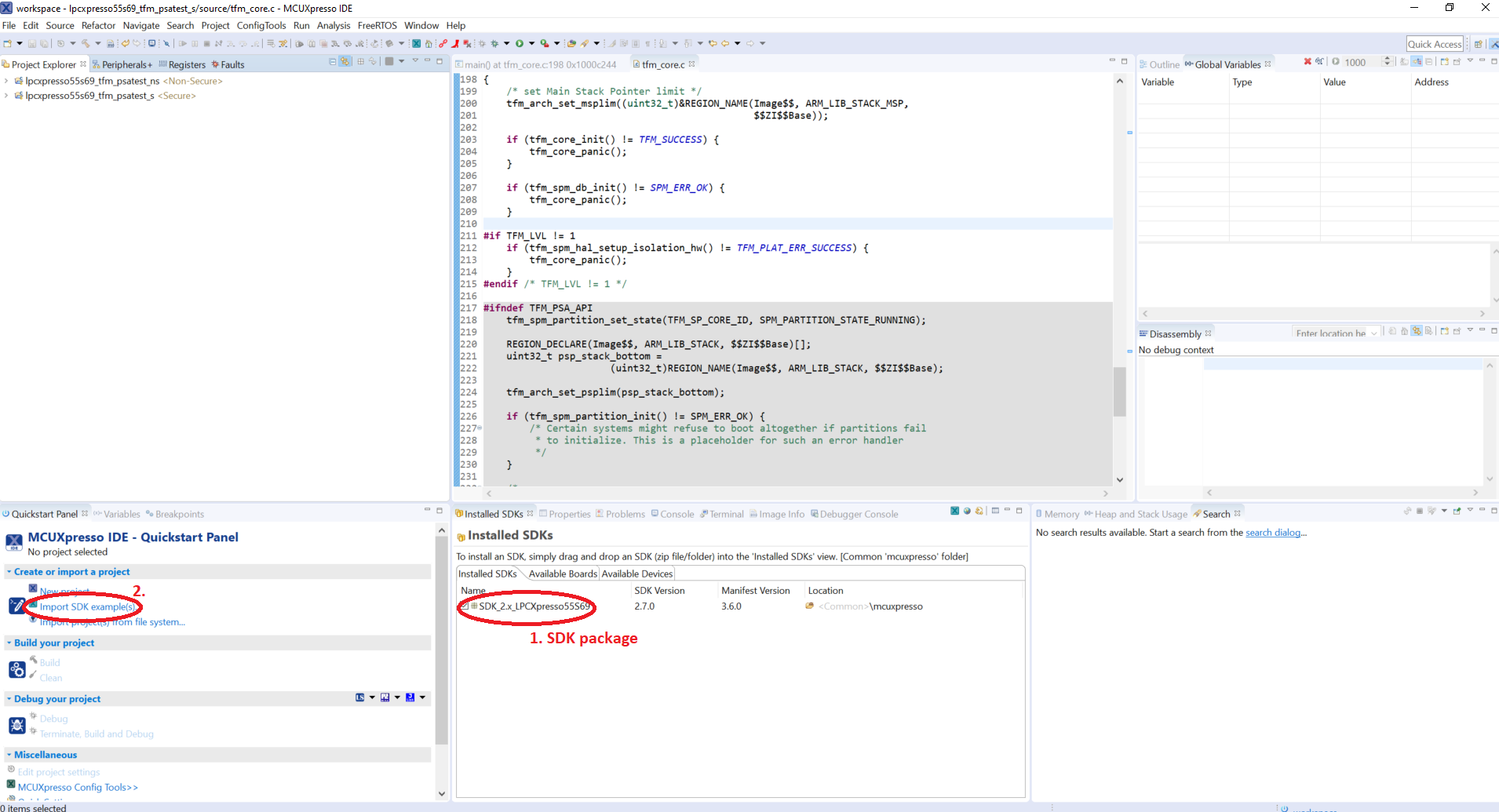
authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

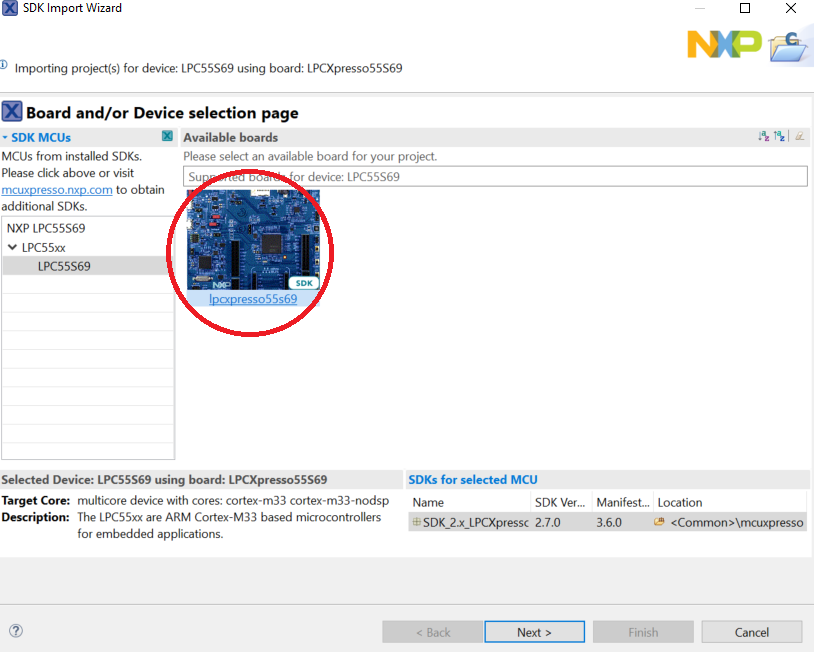
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

## Prepare Binary Files

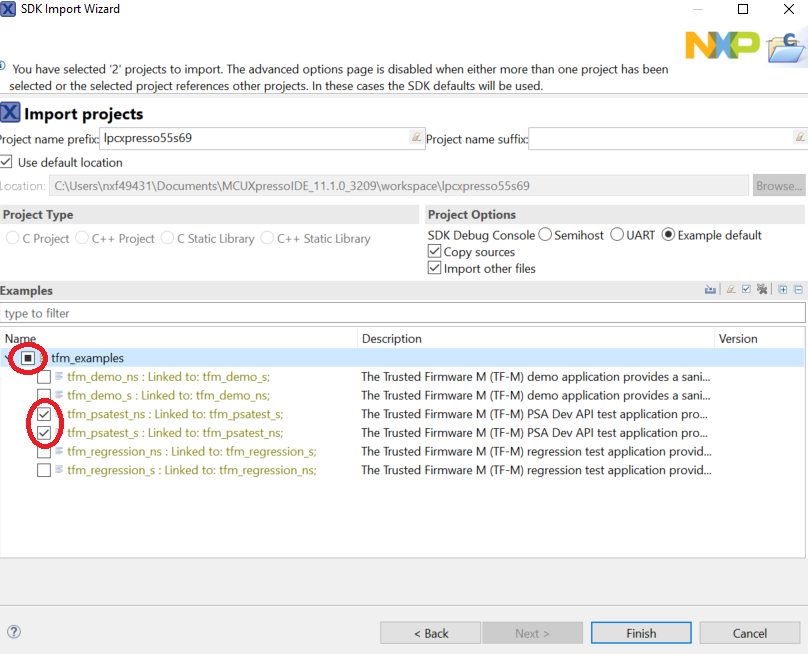
* Open MCUxpresso IDE
* Move generated SDK package to the Installed SDKs list (See figure 1)
* Click on Quickstart Panel -> Create or import a project -> Import SDK example(s) (See figure 1)
* Choose lpcxpresso55s69 board and click on next (See figure 2)
* Choose examples tfm\_examples/tfm\_psatest\_s and tfm\_examples/tfm\_psatest\_ns and click on finish button (See figure 3)
* In the project explorer are located two projects
* Right click on lpcxpresso55s69\_tfm\_psate\_ns and choose build project (both projects will be built)
* Right click on lpcxpresso55s69\_tfm\_psate\_ns.axf in lpcxpresso55s69\_tfm\_psate\_ns/Debug and choose Binary Utilities/Create binary (do the same for secure project) and binary files will be created (See figure 4)



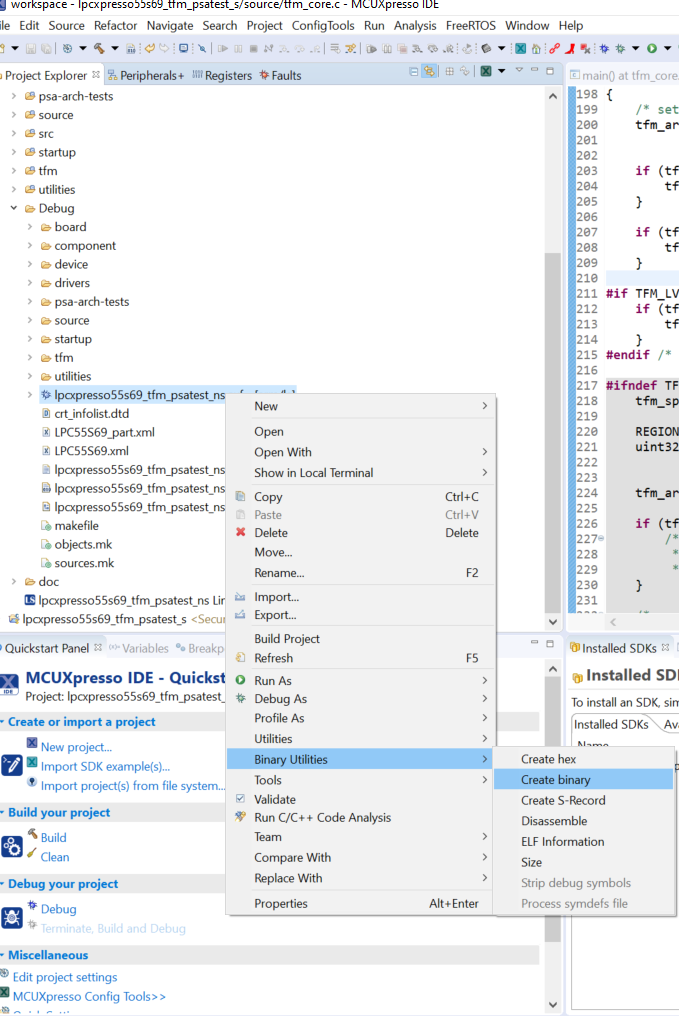
1. MCUxpresso IDE



1. SDK Import Wizard – step 1



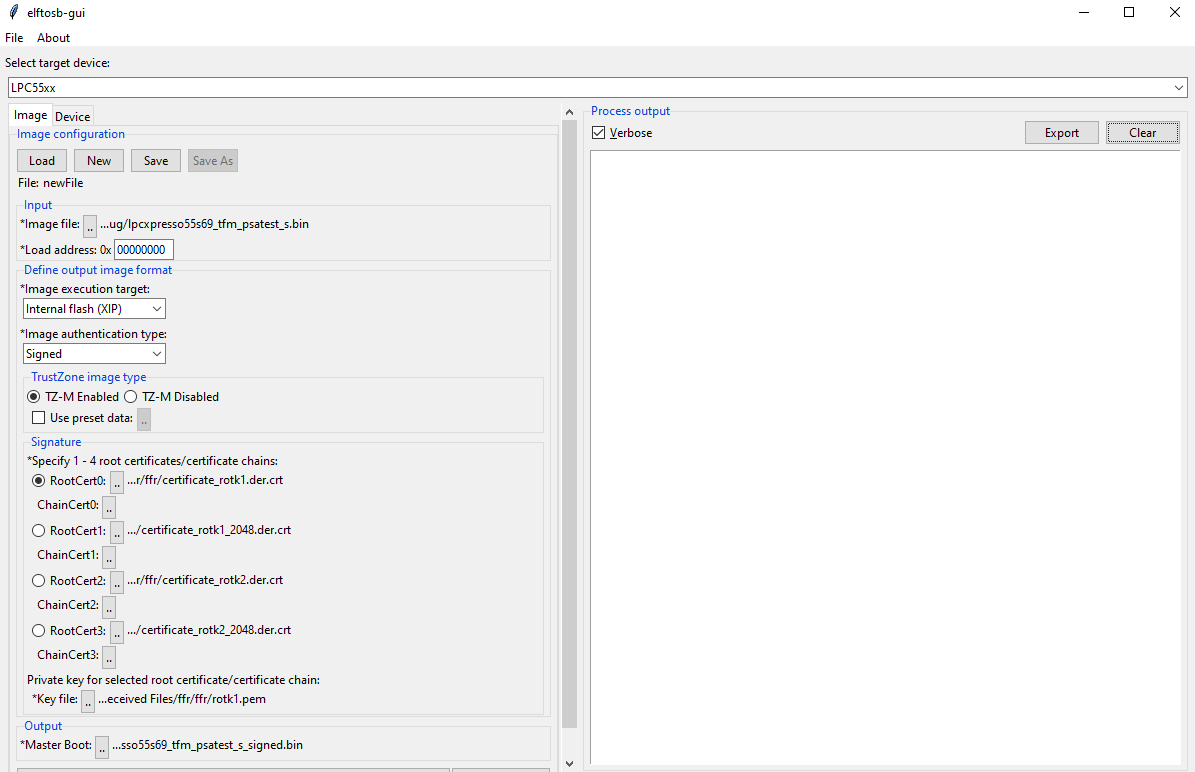
1. SDK Import Wizard – step 2



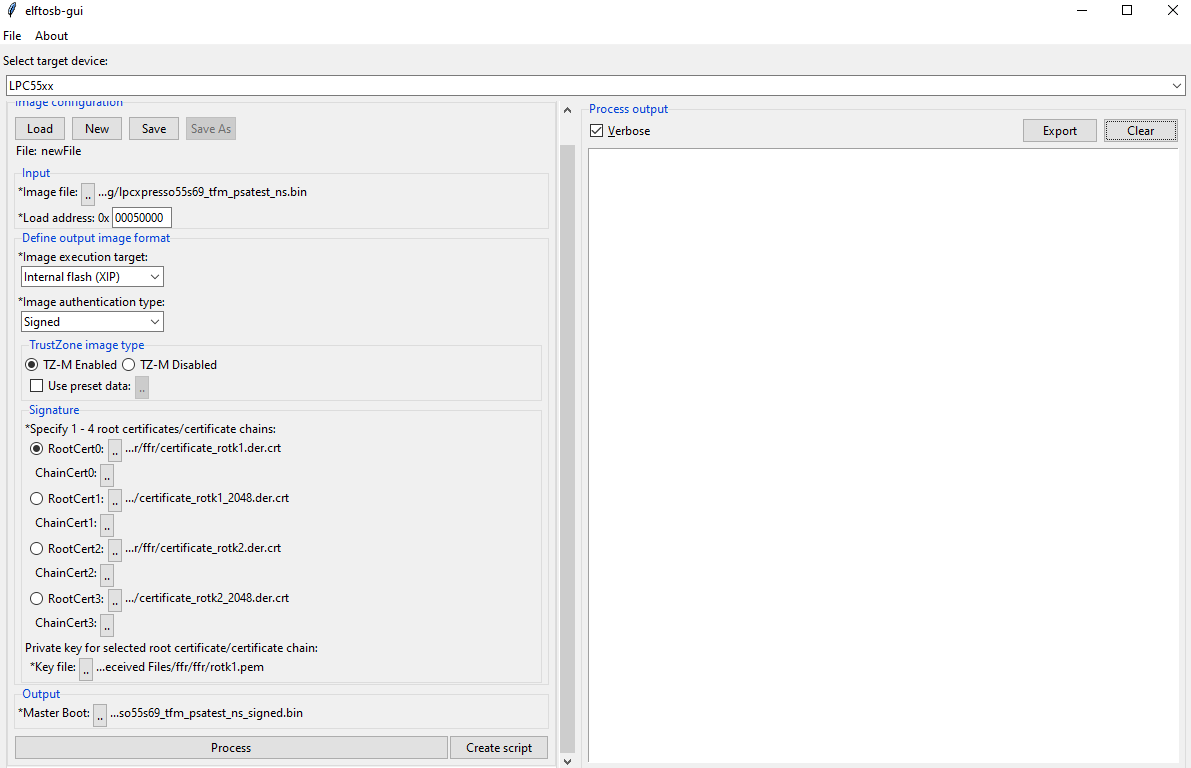
1. Create binary file

## Signed Images Preparation

* Select LPC55Sxx device
* Set all settings according to figures 5 and 6.



1. Elftosb-gui setup for signed image (secure)



1. Elftosb-gui setup for signed image (nonsecure)

## Erasing a Device

Before signed images are loaded to the device, the internal flash memory should be erased:

blhost -p COMxx flash-erase-region 0x00000 0x9ddfc

## CFPA Page Preparation

By default, the CFPA is cleared. There are registers related to secure boot which is set up.

ROTKH\_REVOKE field at CFPA page address 0x9DE18 has to be setup to accept signed images with created certificates. See Protected Flash Region v1.0.xlsx in attachments in UM11126\_LPC55x\_v.1.7.pdf reference manual.

Enter UART ISP mode

* Press and hold the ISP button (Switch S1) on the LPC55Sxx development board while pressing the RESET button (Switch S4) to enter ISP mode

Test communication through BLHOST tool

blhost -p COMxx get-property 1

\*if ping failure show it has got wrong connection

Prepare CFPA page in .bin file (example with RoT key 0-3 enabled and Version set to 1 is attached – InFieldROMAPIScratch\_0x9de00.bin)

Write images into flash memory

blhost -p COMxx write-memory 0x9DE00 <path to the CFPA(.bin)>

*NOTE: Never restart the device before both pages (CFPA and CMPA) are loaded, otherwise you will increase a probability of lock the device.*

## CMPA Page Preparation

The CMPA page is configured to boot the signed image.

The Root Key Table Hash (RKTH) is counted by elftosb-gui (see figure 7). This RTKH field at CMPA page address 0x9e450 has to be setup. See Protected Flash Region v1.0.xlsx in attachments in UM11126\_LPC55x\_v.1.7.pdf reference manual.

Enter UART ISP mode

* Press and hold the ISP button (Switch S1) on the LPC55Sxx development board while pressing the RESET button (Switch S4) to enter ISP mode

Test communication through BLHOST tool

blhost -p COMxx get-property 1

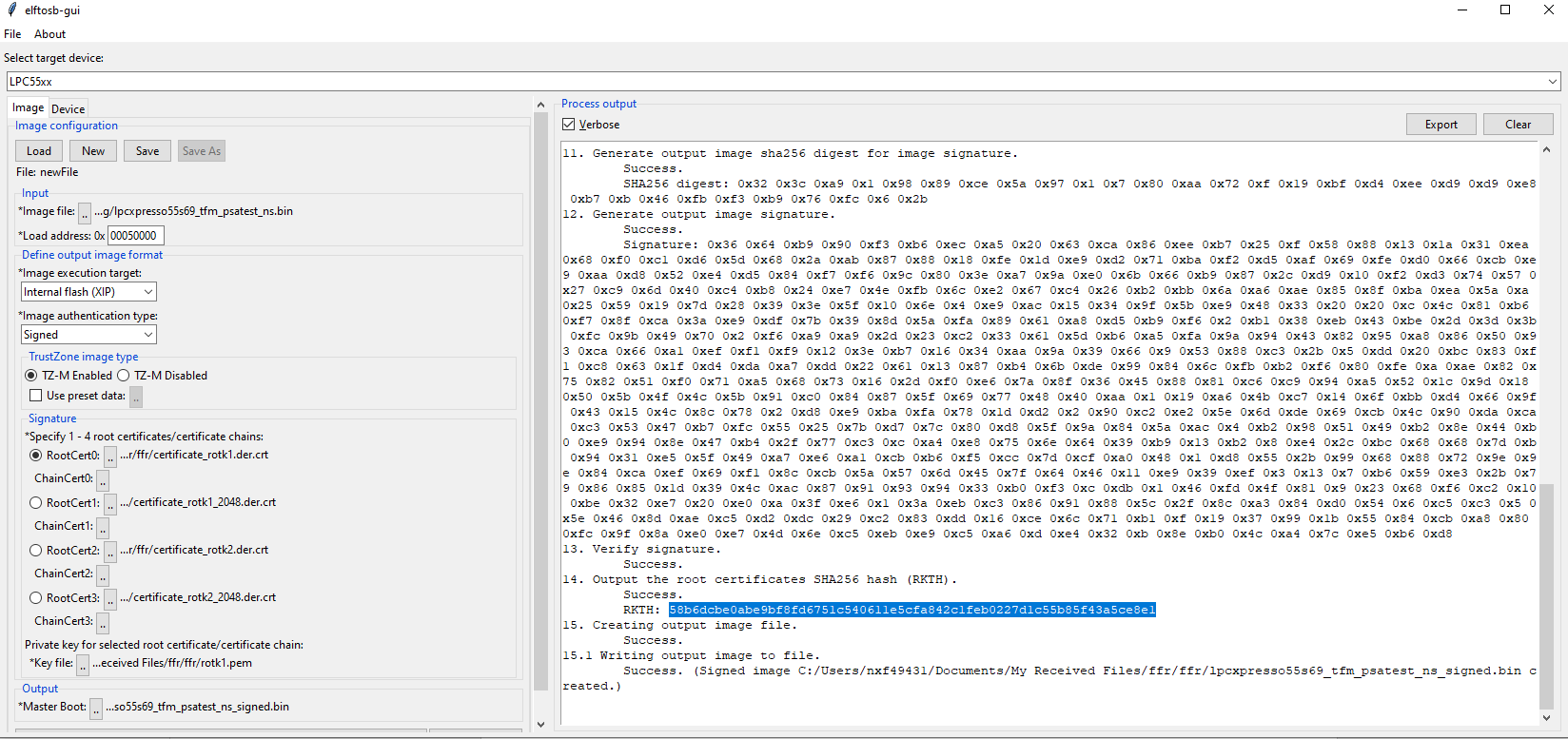
\*if ping failure show it has got wrong connection

Prepare CMPA page in .bin file (example is attached CustomerManufacturingFactoryConfigurationArea\_0x9e400.bin – when the attached file is used, the RKTH has to be updated according to hash value counted by elftosb-gui (figure 7))

Write images into flash memory

blhost -p COMxx write-memory 0x9E400 <path to the CMPA(.bin)>

*NOTE: Never restart the device before both pages (CFPA and CMPA) are loaded, otherwise you will increase a probability of lock the device.*



1. Root of Trust Keys Table Hash

## Loading Signed Images without Flash Encryption

### Loading Signed Images

The signed images are programmed into the device by means of BLHOST utility in ISP mode. For more information, see the BLHOST manual.

Enter UART ISP mode

* Press and hold the ISP button (Switch S1) on the LPC55Sxx development board while pressing the RESET button (Switch S4) to enter ISP mode

Test communication through BLHOST tool

blhost -p COMxx get-property 1

\*if ping failure show it has got wrong connection

Write images into flash memory

blhost -p COMxx write-memory 0 <path to the signed secure image(.bin)>

blhost -p COMxx write-memory Y <path to the signed nonsecure image(.bin)>

\*Y The value is taken from map file (0x00050000 for this example)

After these steps, the signed images are loaded in the flash.

## Loading Signed Images with Flash Encryption Using PRINCE

### PRINCE related PUF key store setup

The keys used for PRINCE encryption/decryption are generated in the device using on-chip SRAM PUF and they are delivered to the PRINCE engine through internal hardware bus.

The proper PRINCE enabled Key Store needs to be generate via blhost application.

* generate device activation code and store it into key store structure

blhost -p COMxx key-provisioning enroll

* generate random PRINCE region 0, key type = 7

blhost -p COMxx key-provisioning set\_key 7 16

* generate random PRINCE region 1, key type = 8

blhost -p COMxx key-provisioning set\_key 8 16

* generate random PRINCE region 2, key type = 9

blhost -p COMxx key-provisioning set\_key 9 16

* save the key store into PFR page of Flash memory

blhost -p COMxx key-provisioning write\_key\_nonvolatile 0

### PRINCE region configuration with blhost

For PRINCE encryption and decryption the regions and sub-regions for the crypto operation needs to be configured. This can be done with ISP command “configure-memory”. This command have to be called with following data structure.

| Table 1. Structure of configure-memory command | | |
| --- | --- | --- |
| Offset | Size | Description |
| 0 | 4 | PRINCE Configuration |
| 4 | 8 | PRINCE Region Info |

Table 2. PRINCE configuration register for configure-memory command

|  |  |
| --- | --- |
| Bit | Symbol |
| 1:0 | 0x00 – PRINCE Region 0  0x01 – PRINCE Region 1  0x10 – PRINCE Region 2 |
| 25:2 | Reserved |
| 31:8 | 0x50 (‘P’) – Configure PRINCE |

Table 3. PRINCE region info register for configure-memory command

|  |  |
| --- | --- |
| Bit | Symbol |
| 31:0 | PRINCE Region X Start |
| 63:32 | PRINCE Region X Size |

Load structure into RAM memory and call “configure-memory” command with this structure:

1. Region selection (Region 0)

blhost -p COMxx fill-memory 0x20034000 4 0x50000000

1. Start address of encrypted area

blhost -p COMxx fill-memory 0x20034004 4 0

1. Length of the encrypted area

blhost -p COMxx fill-memory 0x20034008 4 0x40000

1. Call configure-memory with prepared structure in RAM

blhost -p COMxx configure-memory 0 0x20034000

1. Region selection (Region 1)

blhost -p COMxx fill-memory 0x2003400c 4 0x50000001

1. Start address of encrypted area

blhost -p COMxx fill-memory 0x20034010 4 0x40000

1. Length of the encrypted area

blhost -p COMxx fill-memory 0x20034014 4 0x40000

1. Call configure-memory with prepared structure in RAM

blhost -p COMxx configure-memory 0 0x2003400c

1. Region selection (Region 2)

blhost -p COMxx fill-memory 0x20034018 4 0x50000002

1. Start address of encrypted area

blhost -p COMxx fill-memory 0x2003401C 4 0x80000

1. Length of the encrypted area

blhost -p COMxx fill-memory 0x20034020 4 0x1ddfc

1. Call configure-memory with prepared structure in RAM

blhost -p COMxx configure-memory 0 0x20034018

### Loading Signed Images

The signed images are programmed into the device by means of BLHOST utility in ISP mode. For more information, see the BLHOST manual.

Enter UART ISP mode

* Press and hold the ISP button (Switch S1) on the LPC55Sxx development board while pressing the RESET button (Switch S4) to enter ISP mode

Test communication through BLHOST tool

blhost -p COMxx get-property 1

\*if ping failure show it has got wrong connection

Write images into flash memory

blhost -p COMxx write-memory 0 <path to the signed secure image(.bin)>

blhost -p COMxx write-memory Y <path to the signed nonsecure image(.bin)>

\*Y The value is taken from map file (0x00050000 for this example)

After these steps, the signed images are loaded in the flash.

## Download CFPA and CMPA pages

To check that CFPA and CMPA pages were setup correctly are used the following commands:

blhost -p COMxx read-memory 0x9de00 512 <path to the CFPA output(.bin)>

blhost -p COMxx read-memory 0x9e400 512 <path to the CMPA output(.bin)>

# Debug Authentication

# Provisioning