OPTIGA™ TPM Application Note

EK-based Device Onboarding

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 used on a Raspberry Pi® to perform the Endorsement Key (EK) based device onboarding.

The Endorsement Key (EK) based device onboarding is a mechanism to register a device on a cloud service using EK as device identity without the need to provision TPM with additional key or certificate. Moreover, it is possible to perform secure transfer of secret keys (HMAC/RSA/ECC) from a server to a device's TPM. Depending on the application, these keys can be used for different purposes, e.g., second layer data encryption/decryption.

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 [1][2]. More information on TPM specification can be found on Trusted Computing Group (TCG) in reference [3].

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 device onboarding or key sharing for their target applications.

Table of contents

List of figures

List of tables

Acronyms and Abbreviations

Acronym	Definition
CA	Certificate Authority
EK	TPM's Endorsement Key
JWT	JSON Web Token (RFC 7519)
PKI	Public Key Infrastructure
TPM	Trusted Platform Module

1 Prepare Raspberry Pi®

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

1.1 Prerequisites

- Raspberry Pi® 4
- Flash the Raspberry Pi® OS image (2021-01-11 release from [4]) on a micro-SD card (≥8GB)
- OPTIGA™ TPM SLx 9670 TPM2.0 [1]

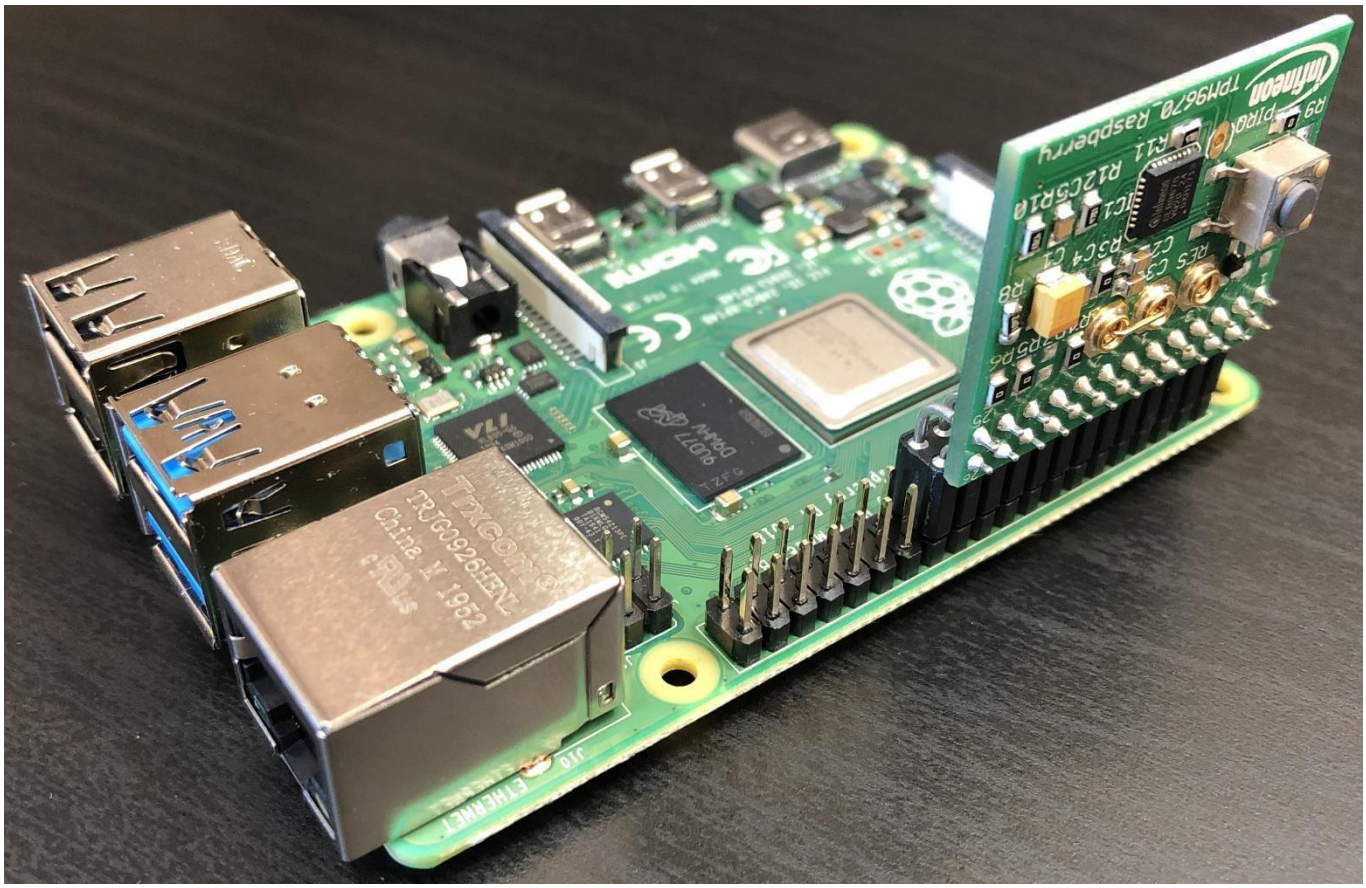

Figure 1 Infineon Iridium SLx 9670 TPM2.0 SPI Board on a Raspberry Pi® 4

1.2 Enable TPM

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

Open the configuration file in an editor:

Code Listing 1

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

Insert the following lines to enable SPI and TPM.

Code Listing 2

```
001        dtoverlay=tpm-slb9670
```

Save the file and exit the editor.

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

Code Listing 3

```
001        $ ls /dev | grep tpm
002        tpm0
003        tpmrm0
```

1.3 Install Dependencies

Install software dependencies on Raspberry Pi®.

Code Listing 4

```
001        $ sudo apt update
002        $ sudo apt install xxd jq maven openjdk-9-jre bc openssl
             autoconf-archive libcmocka0 libcmocka-dev procps iproute2
             build-essential git pkg-config gcc libtool automake libssl-
             dev uthash-dev autoconf doxygen libjson-c-dev libini-config-
             dev libcurl4-gnutls-dev uuid-dev pandoc
```

Install TPM software stack on Raspberry Pi®.

Code Listing 5

```
001        $ git clone https://github.com/tpm2-software/tpm2-tss.git
002        $ cd tpm2-tss
003        $ git checkout 3.0.3
004        $ ./bootstrap
005        $ ./configure
006        $ make -j$(nproc)
007        $ sudo make install
008        $ sudo ldconfig
```

Install TPM tools on Raspberry Pi®.

Code Listing 6

```
001        $ git clone https://github.com/tpm2-software/tpm2-tools.git
002        $ cd tpm2-tools
003        $ git checkout 5.0
004        $ ./bootstrap
005        $ ./configure
006        $ make -j$(nproc)
007        $ sudo make install
008        $ sudo ldconfig
```

2 Setup

This section describes all the steps necessary for setting up a server and a device on a Raspberry Pi®.

2.1 Server

The server is developed using the Spring framework [5]. Download the server source code on Raspberry Pi®.

Code Listing 7

```
001        $ git clone https://github.com/infineon/ek-based-onboarding-
             optiga-tpm
002        $ mv ek-based-onboarding-optiga-tpm local-server
003        $ cd local-server
004        $ git checkout server
```

Build the source.

Code Listing 8

```
001        $ mvn package
```

2.2 Device

Download the device scripts on Raspberry Pi®.

Code Listing 9

```
001        $ git clone https://github.com/infineon/ek-based-onboarding-
             optiga-tpm
002        $ mv ek-based-onboarding-optiga-tpm device-scripts
003        $ cd device-scripts
004        $ git checkout device
```

3 Operation

Table 1 shows the flow of the system.

Table 1 Operation flow

Operation flow	Description
1. Onboarding	This is performed once to onboard a device to a server in the following sequence. 1. The server verifies EK certificate is issued by Infineon's CA. 2. If whitelist feature is enabled in Figure 4, only devices with whitelisted EK public key can be onboarded. 3. 3 TPM key objects (HMAC, RSA, and ECC) will be generated and exported from the server to a device.
2. Authentication	The authentication step requires a device to produce a total of 3 signatures using the 3 imported keys (this is for demonstration purpose only, in normal circumstances, a single signature may be sufficed). After a successful authentication, device will receive a server signed JWT formatted token. The JWT is needed for all subsequent communication with the server as a proof of identity. The JWT is only valid for a finite period of time. The authentication step can be repeated to obtain a new JWT.
3. Key derivation & second layer encryption	This is to demonstrate the imported HMAC key can be used to introduce an additional layer of data encryption. A session-based encryption key can be derived using the HMAC function after exchanging seeds between server and device.

3.1 Start Server

Start the server on Raspberry Pi®.

Code Listing 10

```
001     $ cd local-server/server/target
002     $ sudo java -jar server-0.0.1-SNAPSHOT.jar
```

The server is ready for operation once you see the following message.

Code Listing 11

```
001     ...
002     2020-06-10 22:37:51.856  INFO 12828 --- [           main]
        o.s.m.s.b.SimpleBrokerMessageHandler     : Started.
003     2020-06-10 22:37:52.414  INFO 12828 --- [           main]
        o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on
        port(s): 443 (https) 80 (http) with context path ''
004     2020-06-10 22:37:52.418  INFO 12828 --- [           main]
        com.ifx.server.ServerApplication         : Started
        ServerApplication in 91.269 seconds (JVM running for 98.966)
```

View the webpage (https://localhost) using Raspberry Pi® OS built-in web browser. A warning message may appear because the server is using a self-signed certificate. Bypass the warning and proceed as usual.

On the upper menu bar, click on "Start" to enter the sign in page (Figure 2). Sign in using the credential from Table 2 to enter the dashboard page.

Table 2 Default login credential

Username	infineon
Password	noenifni

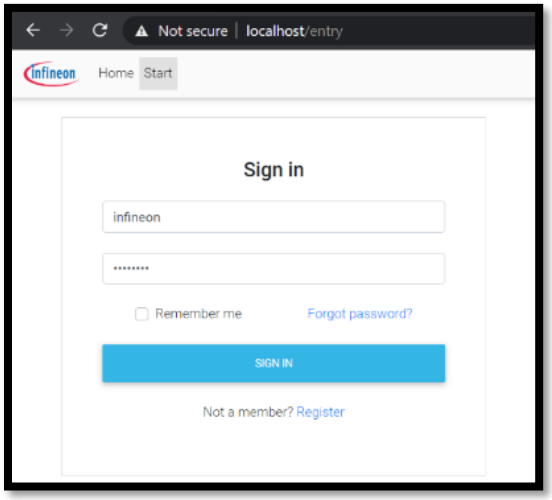

Figure 2 Server sign in page

The dashboard page (Figure 3, Figure 4) has the following elements.

Table 3 Server dashboard elements

Element	Description
A	The table shows a list of onboarded devices and a button to unpair devices.
B	Server log.
C	Upload a CSV formatted file containing a list of whitelisted EK public keys.
D	Enable or disable the effect of whitelisting. If the option is disabled, a device can be onboarded only if: • TPM is issued by Infineon If the option is enabled, a device can be onboarded only if: • TPM is issued by Infineon, and • EK public key is whitelisted
E	The table shows a list of whitelisted devices and a button to blacklist devices.

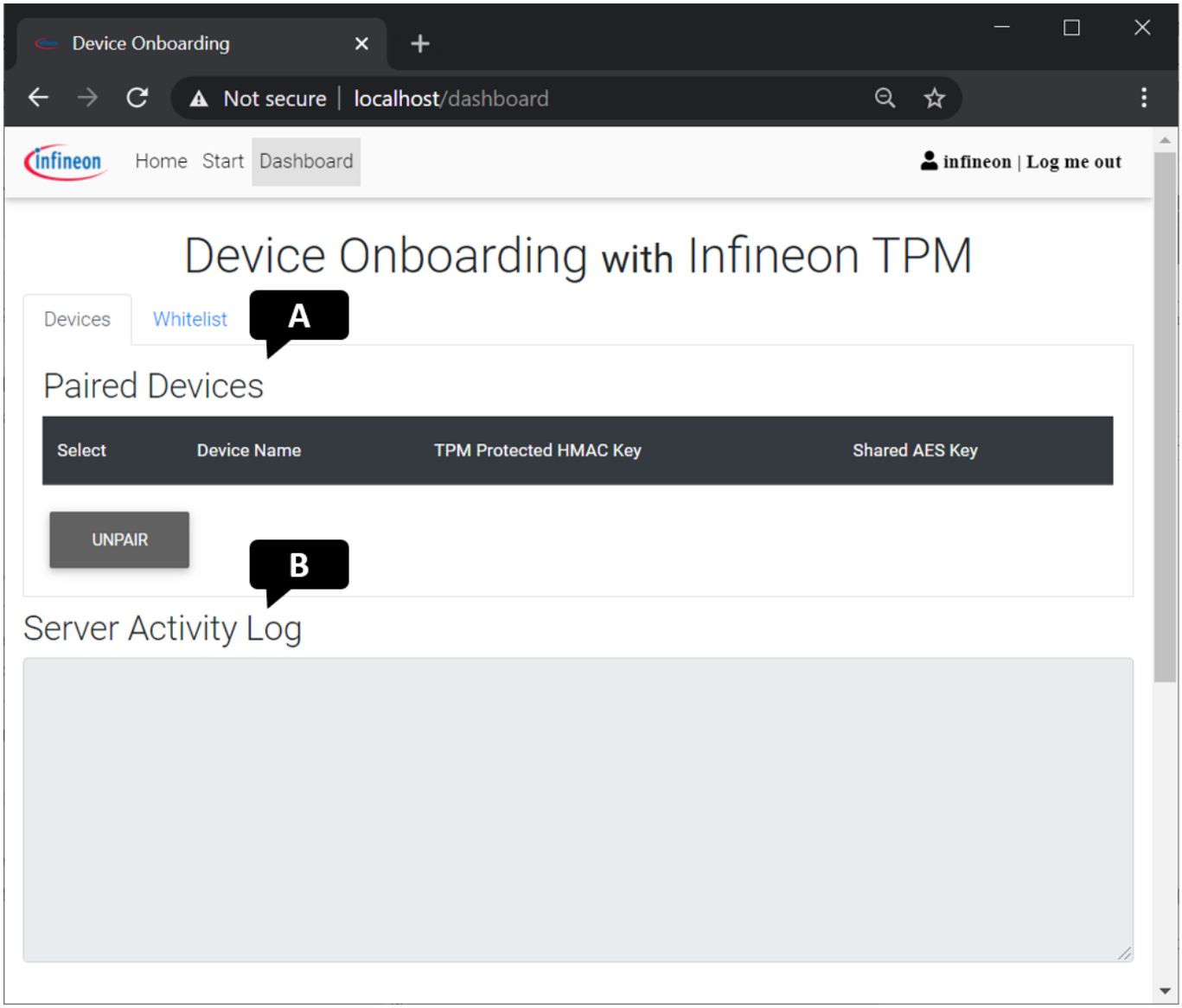

Figure 3 Server dashboard device page

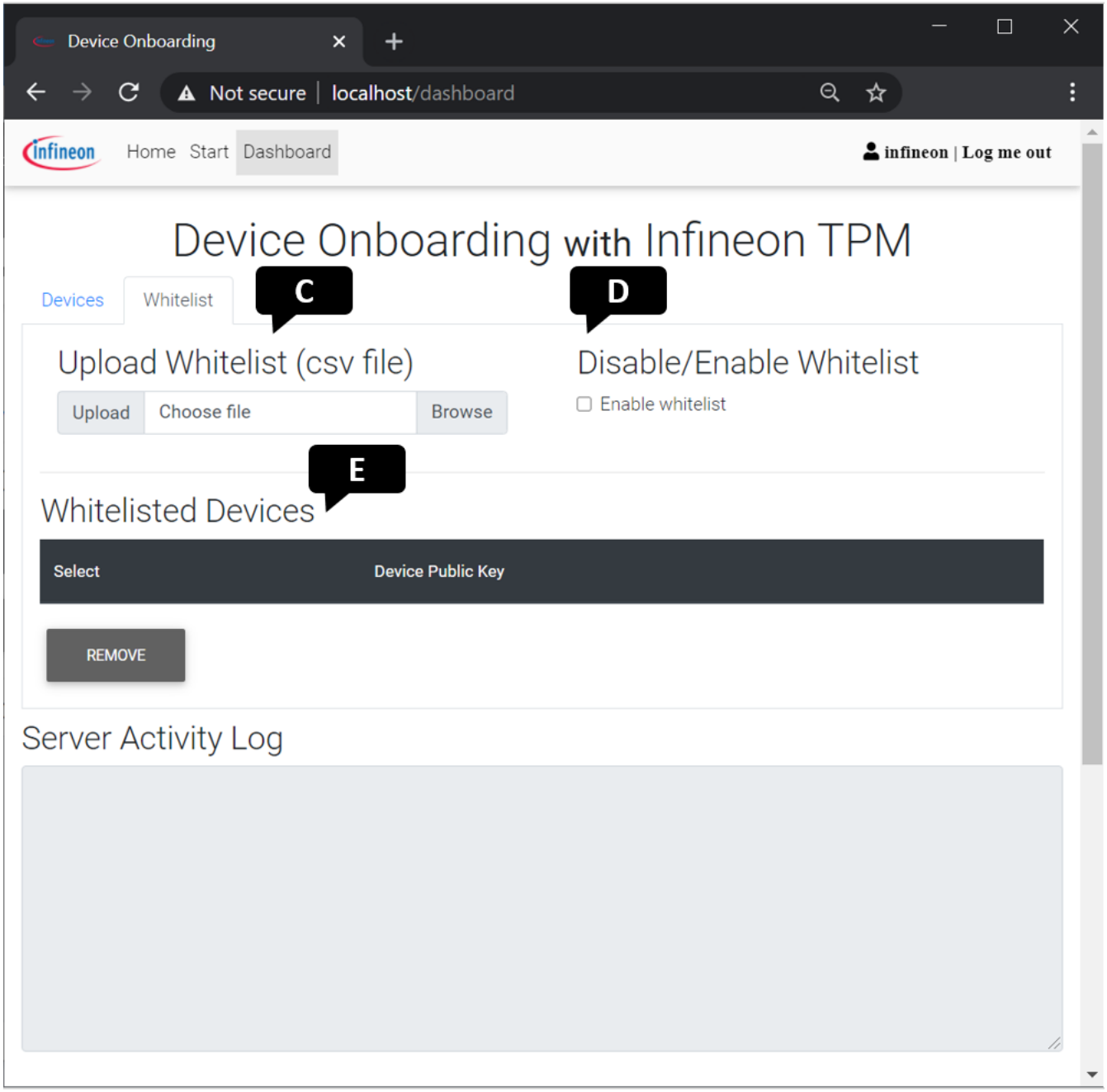

Figure 4 Server dashboard whitelist page

3.2 Run Device Scripts

With the server running, execute the following scripts in Table 4 sequentially.

Table 4 Device scripts

	Script	Description
Onboarding	**0_prep.sh**	Authorize non-privileged access to the TPM device nodes.
	1_clean.sh	Clean up the environment, erase non-essential files.
	2_tpm-clear-init.sh	Initialize the TPM in the following sequence: 1. Clear TPM. 2. Create RSA EK. 3. Read out RSA EK certificate. 4. Create an RSA-based parent key. The parent key is used as a wrapping key to securely import keys from a server. Create a csv formatted whitelist containing a single EK public key. If the whitelist feature is enabled, remember to import the csv file via the dashboard page (Figure 4) before proceeding.
	3_onboard-req.sh	Send an onboarding request to a server. The request is composed of: • Username (“infineon”) to pair device with • EK certificate • Parent public key Expected response from the server: • Credential blob containing a decryption key (Ƙ) • Encrypted device id, secured by Ƙ • Encrypted HMAC-SHA256 key, secured by parent key and inner wrap key (Ƙ) • Encrypted RSA-2048 key, secured by parent key and Ƙ • Encrypted ECC NIST P-256 key, secured by parent key and Ƙ • Challenge If the whitelist feature is enabled, only whitelisted devices can be onboarded.
	4_tpm-key-import.sh	Process the server response in the following sequence: 1. Perform TPM activate credential on the credential blob to recover Ƙ. 2. Decrypt device id using Ƙ. 3. Import HMAC, RSA, and ECC keys into TPM under the RSA-based parent key and using Ƙ as inner wrap key. Sign the challenge using the 3 imported keys.
	5_onboard-ack.sh	Send an acknowledgement packet to the server containing the 3 signatures.
Authentication	**6_auth.sh**	Perform authentication in the following sequence: 1. Request a challenge from the server. 2. Sign the challenge using the 3 imported keys. 3. Upon receiving valid signatures, the server generates a token (JWT) and returns it to the device. The JWT comprises a subject (device id), token validity period, and a server generated signature. It is mandatory to attach the JWT to all subsequence requests make to the server.

Key derivation	**7_seed-exchange.sh**	Perform seed exchange.
	8_derive-aes-key.sh	Device derives a session-based AES key (AES-256/CBC/PKCS5Padding) by performing HMAC operation taking seeds and the HMAC key as input.
Second layer encryption	**9_secured-download.sh**	Device downloads encrypted media from server and decrypts it using the AES key. The expected media are: • A 3gp (decrypted.3gp) formatted video, can be played using Raspberry built-in VLC Media Player • A text file (decrypted.txt) with message “You have successfully decrypted this message.” • A 4 bytes binary file (decrypted.hex) with hexadecimal 0xdeadbeef

References

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

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

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

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

[5] https://spring.io/

Revision history

Reference	Description
Revision 1.0, 2021-02-22	
all	Initial version

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

Edition 2021-02-22

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:
csscustomerservice@infineon.com