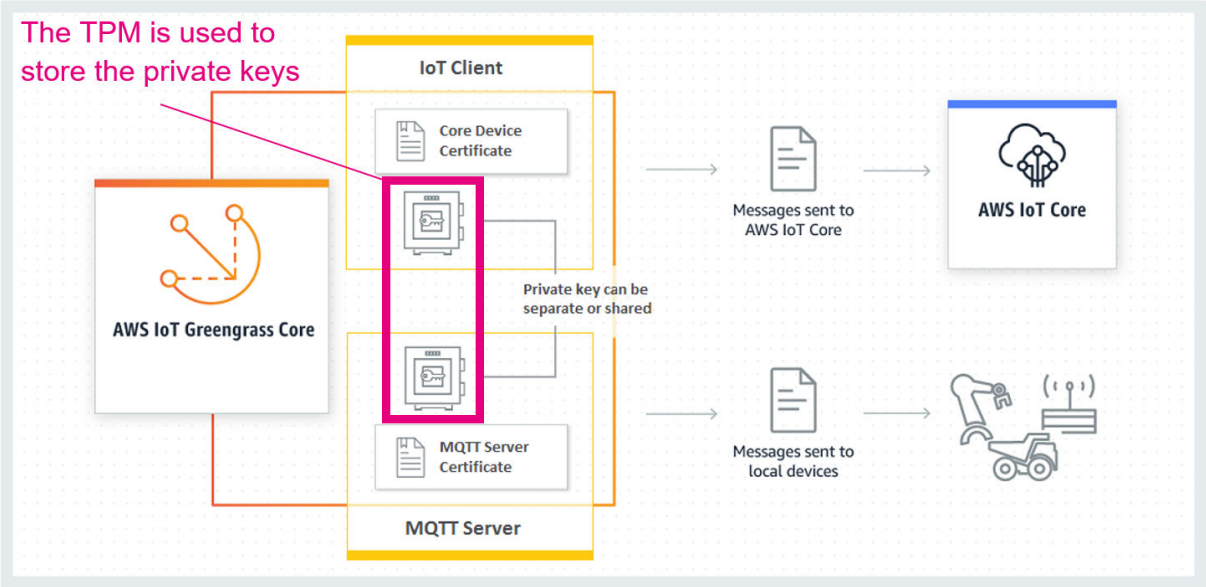
Securing cloud connections: key storage for AWS™ IoT Greengrass with the ST33TPHF2XSPI and ST33TPHF2XI2C TPM devices

Introduction

The purpose of this application note is to explain how to integrate Amazon Web Services™ (AWS™) IoT Greengrass on a Raspberry Pi® 3B+ with the ST33TPHF2XSPI or ST33TPHF2XI2C trusted platform module (TPM) 2.0 device.

Note that AWS IoT Greengrass could also be integrated on an STM32 board (see How to integrate AWS IoT Greengrass on top of openSTLinux_distribution for further details on the setup configuration).

Figure 1. Hardware security architecture for AWS IoT Greengrass



Note: Picture provided by courtesy of Amazon®.

This application note explains step by step how to:

- set up the Raspberry Pi as the AWS IoT Greengrass Core;
- download and install the TPM software stack (TSS);
- provision the TPM device;
- configure the target for a secure connection with the AWS cloud;
- execute individual AWS Greengrass™ device testing.

To perform these tasks, some knowledge of AWS IoT Greengrass, Raspberry Pi and Linux® kernel configuration and building is necessary.

To learn about AWS Greengrass™, refer to:

- the AWS Greengrass web page: see [Greengrass]
- the AWS Greengrass developer guide: see [DeveloperGuide]

To understand how a TPM device can increase the security of AWS cloud connections, refer to [HW_security].

The instructions provided in this application note have been tested with AWS IoT Greengrass Version 1, and the software versions listed below. Any other software version or functionality has not been tested.

Table 1. Tested software and functionality versions

TSS version	Tpm2-tools version	Tpm2-pkcs11 version	AWS Greengrass IDT version	AWS IoT Greengrass Core version
2.4.1	4.2.1	1.0.2	3.1.x	1.10.x
3.0.1	4.3.0	1.4.0	3.2.0	

Note: Amazon Web Services, AWS and AWS Greengrass are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

1 Reference links

The table below lists all the public resources that are needed in conjunction with this application note and are referenced therein. They can be links to open-source software resources, or to guides provided by third-party websites.

Table 2. Reference links

Reference	Open-source software resource
[Greengrass]	The AWS Greengrass web page: https://aws.amazon.com/greengrass(1)
[DeveloperGuide]	The AWS Greengrass developer guide available from https://docs.aws.amazon.com/greengrass(1)
[HW_security]	*Hardware security integration* section of the developer guide available at: https://docs.aws.amazon.com/greengrass/latest/developerguide/hardware-security.html(1)
[Signup]	AWS account creation page: https://portal.aws.amazon.com/billing/signup#/start(1)
[RPi_setup]	*Setting up a Raspberry Pi* section of the developer guide available at: https://docs.aws.amazon.com/greengrass/latest/developerguide/setup-filter.rpi.html(1)
[LinuxKernel]	Linux kernel sources for a Raspberry Pi, available at: https://github.com/raspberrypi/linux(1)
[TCG-TPM-I2C-DRV]	STMicroelectronics patch for I²C TPM devices: https://github.com/STMicroelectronics/TCG-TPM-I2C-DRV
[TPM2-TSS_repository]	The TPM2 software stack repository: https://github.com/tpm2-software(1)
[TSS]	Latest TPM software stack available at https://github.com/tpm2-software/tpm2-tss/releases(1)
[TPM2-tools]	Latest TPM2-tools version available at: https://github.com/tpm2-software/tpm2-tools/releases(1)
[PKCS#11]	Latest release of PKCS#11, available at https://github.com/tpm2-software/tpm2-pkcs11/releases(1)
[GreengrassGroup&Core]	*Configure AWS IoT Greengrass on AWS IoT* section of the AWS IoT Greengrass developer guide: https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-config.html(1)
[GreengrassCore_SW]	*Start AWS IoT Greengrass on the core device* section of the AWS IoT Greengrass developer guide: https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-device-start.html(1)
[AWS-IoT-Console_login]	URL to log in to the AWS IoT Console: https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/dashboard(1)
[IAM-policy&user]	*Prerequisites for running the AWS IoT Greengrass qualification suite* section of the AWS IoT Greengrass developer guide: https://docs.aws.amazon.com/greengrass/latest/developerguide/dev-tst-prereqs.html(1)
[IDT]	AWS Greengrass IDT, available at: https://docs.aws.amazon.com/greengrass/latest/developerguide/dev-test-versions.html(1)
[Results-logs]	*Understanding results and logs* section of the AWS IoT Greengrass developer guide: https://docs.aws.amazon.com/greengrass/latest/developerguide/results-logs.html(1)
[Docker]	Docker software available at: https://get.docker.com(1)
[Dependency-checker-script]	Dependency checker script available at: https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-dependency-checker-GGCv1.11.x.zip(1)

1. *This URL belongs to a third party. It is active at document publication, however, STMicroelectronics shall not be liable for any change, move or inactivation of the URL or the referenced material.*

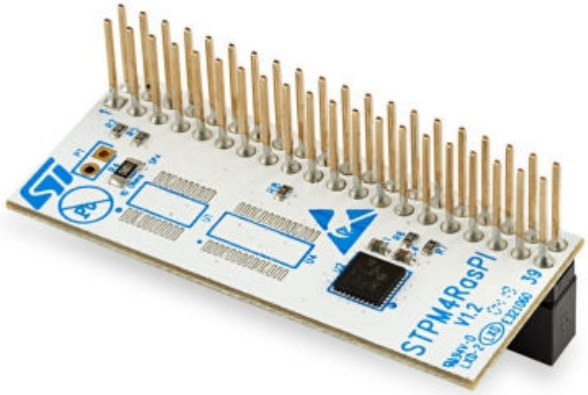
2 Hardware setup

The STPM4RasPI extension board features a soldered TPM device and a Raspberry Pi connector. It is used to connect the ST33TPHF2XSPI or ST33TPHF2XI2C TPM device to the Raspberry Pi board.

The ST33TPHF2XSPI and ST33TPHF2XI2C devices are based on Arm® cores.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Figure 2. STPM4RasPI extension board



The soldered TPM device follows the Trusted Computing Group® (TCG) standards for the trusted platform module defined in the TCG Trusted Platform Module Library Specifications version 2.0 Level 0 Revision 138 [TPM 2.0 r138] and errata version 1.4 [TPM 2.0 rev138 Err].

Two different TPM models exist:

- The ST33TPHF2XI2C, a TPM2.0 device with an I²C interface. Visit the st.com website for more detailed product information.
- The ST33TPHF2XSPI, a TPM2.0 device with an SPI interface. Visit the st.com website for more detailed product information.

3 Raspberry Pi configuration

The Raspberry Pi configuration takes three steps, detailed in the following subsections.

3.1 Step 1: Creating an AWS account

To create an AWS account, click on the link provided in [Signup].

To register, you will need an e-mail address, a password, and an account name. You will also need to provide a valid credit card number.

3.2 Step 2: Raspberry Pi setup

AWS provide a guide to set up a Raspberry Pi for AWS Greengrass. Follow the steps described in this guide (see [RPi_setup].

3.3 Step 3: TPM enablement preparation

The third step consists in configuring the Raspberry Pi to enable the TPM device, and installing all necessary software dependencies.

3.3.1 SPI and I²C buses

To configure the Raspberry Pi to enable the TPM device, navigate to **[Menu]>[Preferences]>[Raspberry PI Configuration]** and activate **I²C and/or SPI** in the **Interfaces** item. This corresponds to the communication buses that one can use to communicate with the ST33TPHF2XSPI and ST33TPHF2XI2C TPM devices.

3.3.2 Software dependencies

Certain software packages are required in the frame of this application note. To install all the necessary dependencies, run the following commands on the Raspberry Pi:

```
sudo apt-get install bc libssl-dev flex bison autoconf libtool autoconf-archive libgcrypt20-dev doxygen libdbus-1-dev libglib2.0-dev libcmocka-dev libcurl4-gnutls-dev pandoc libjson-c-dev libsqlite3-dev libyaml-dev gnutls-bin libp11-kit-dev libp11-3 libp11-dev libengine-pkcs11-openssl openjdk-8-jdk

sudo pip install pyyaml
sudo pip install pyasn1-modules
```

4 TPM enablement in the Linux kernel

To allow communication with the TPM device, enable the TPM driver at kernel level. To do so, download the Linux kernel sources and recompile them with some modifications.

The following sections explain how to download, reconfigure and recompile the Linux kernel.

4.1 Downloading the Linux kernel sources for Raspberry Pi

Just run the following code:

```
cd /home/pi/Downloads
git clone -b rpi-5.4.y --depth=1 https://github.com/raspberrypi/linux(1)
```

4.2 Device tree configuration

To enable the TPM device on the SPI or I²C bus, modify the device tree by editing the `arch/arm/boot/dts/bcm2710-rpi-3-b-plus.dts` file as described in the tables below.

Table 3. Editing the dts file for the ST33TPHF2XSPI TPM device

Replace	By

`spidev0: spidev@0{`
`  compatible = "spidev";`
`  reg=<0>; /* CE0 */`
`  #address-cells = <1>;`
`  #size-cells = <0>;`
`  spi-max-frequency = <500000>;`
`};`
	`st33htpm0: st33htpm@0{`
	`  compatible = "st,st33htpm-spi";`
	`  reg = <0>;`
	`  #address-cells = <1>;`
	`  #size-cells = <0>;`
	`  spi-max-frequency = <33000000>;`
	`  status = "okay";`
	
	`};`

Table 4. Editing the dts file for the ST33TPHF2XI2C TPM device

Replace	By

`&i2c1 {`
`  pinctrl-names = "default";`
`  pinctrl-0 = <&i2c1_pins>;`
`  clock-frequency = <100000>;`
`};`
	`&i2c1 {`
	`  pinctrl-names = "default";`
	`  pinctrl-0 = <&i2c1_pins>;`
	`  clock-frequency = <400000>;`
	`  st33htpi: st33htpi@0{`
	`  compatible = "st,st33htpm-i2c";`
	`  reg = <0x2E>;`
	`  status = "okay";`
	`  };`
	`};`

4.3 Kernel option configuration

Open the `linux/arch/arm/configs/bcm2709_defconfig` file, and add the lines described hereunder.

For the ST33TPHF2XSPI

Add:

```
CONFIG_TCG_TIS_CORE=y
CONFIG_MEMCG=y (this line is not for the TPM, but to enable the Memory Control Group,
necessary for AWS Greengrass Core software execution)
```

Make sure that the following lines are also present:

```
CONFIG_TCG_TPM=y
CONFIG_TCG_TIS_SPI=m
```

For the ST33TPHF2XI2C

Add:

```
CONFIG_CRC_CCITT=y
CONFIG_TCG_TIS_CORE=y
CONFIG_TCG_TIS_I2C=m
CONFIG_MEMCG=y (this line is not for the TPM, but to enable the Memory Control Group,
necessary for AWS Greengrass Core software execution)
```

Make sure that the following line is also present:

```
CONFIG_TCG_TPM=y
```

4.4 I²C driver creation

STMicroelectronics provide a patch to create a generic I²C driver for the ST33TPHF2XI2C TPM device. For the web link to download this patch, refer to [TCG-TPM-I2C-DRV].

Download the patch and copy it into the `linux/drivers/char` directory. Then apply the patch with the following commands:

```
cd /home/pi/Downloads/linux/drivers/char
patch -b -p0 < patchTPMv_5_4_2.patch
```

4.5 Compiling the kernel

Once the aforedescribed modifications have been made to the Linux kernel, compile and install it via the following commands:

```
cd /home/pi/Downloads/linux
make bcm2709_defconfig
make -j4 zImage modules dtbs
sudo make modules_install
sudo cp arch/arm/boot/dts/*.dtb /boot/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
sudo cp arch/arm/boot/zImage /boot/kernel7.img
```

After completion, reboot the Raspberry Pi board.

TPM driver enablement is correct if the `/dev/tpm0` and `/dev/tpmrm0` files are present.

5 Installing the TPM software stack and tools

The TPM2 software stack ([TPM2-TSS_repository]) is an open-source project that aims to implement the TCG TSS specification [TPM TSS 2.0].

5.1 Changing authorizations for the TPM driver

The TPM driver file must be writable. Run the following command to authorize writing to the device:

```
sudo chmod 766 /dev/tpm0
sudo chmod 766 /dev/tpmrm0
```

To run these commands automatically at each boot, add them to the `/etc/rc.local` file:

```
sudo nano /etc/rc.local
```

Add the following lines before “exit 0”:

```
./bin/chmod 766 /dev/tpm0
./bin/chmod 766 /dev/tpmrm0
```

5.2 TSS installation

Download the latest stable release of the TPM software stack. Refer to [TSS] for the URL.
Extract to `/home/pi/Downloads/tpm2-tss`, then run the following commands:

```
cd /home/pi/Downloads/tpm2-tss
./bootstrap
./configure --with-device=/dev/tpmrm0
make
sudo make install
```

5.3 TPM2-tools installation

Download the latest release of tpm2-tools. Refer to [TPM2-tools] for the URL.
Extract to `/home/pi/Downloads/tpm2-tools`:

```
cd tpm2-tools
./bootstrap
./configure
make -j4
sudo make install
sudo ldconfig
```

5.4 TPM2-PKCS#11 installation

Create a directory for the PKCS#11 store:

```
sudo mkdir /opt/tpm2-pkcs11
sudo chmod 777 /opt/tpm2-pkcs11
```

Then download the latest release of PKCS#11. Refer to [PKCS#11] for the URL.
Extract to `/home/pi/Downloads/tpm2-pkcs11`, then run the following commands:

```
cd /home/pi/Downloads/tpm2-pkcs11
./bootstrap
./configure --with-storedir=/opt/tpm2-pkcs11
make
sudo make install
```

The location of the PKCS#11 store must be exported as an environment variable. To do so, run the following command:

```
sudo leafpad /home/pi/.bashrc
```

Add the following line to the end of the file:

```
export TPM2_PKCS11_STORE=/opt/tpm2-pkcs11
```

You can check that the TSS installation is correct by running a simple command:

```
tpm2_getrandom --hex 20
```

This command should output 20 random bytes to the terminal.

6 Setting up the Greengrass Group and its Core

AWS provide a guide to create a first AWS Greengrass Group and a Core for this group, and to download the core software to the Raspberry Pi: refer to [GreengrassGroup&Core].

Make sure to download the **core security resources** as they will be used later.

Then follow the [GreengrassCore_SW] guide to start the Greengrass Core software on your Raspberry Pi.

7 Preparing the PKCS#11 key store

This section describes how to provision the TPM device with one or multiple keys using a PKCS#11 interface. Two key pairs are used to secure AWS Greengrass communications:

- The “core private key” authenticates the Greengrass Core device to the Greengrass IoT server.
- The local MQTT server uses the MQTT key to secure communications between the Greengrass devices and the Greengrass Core.

These two keys can be different, or a shared key can be used for both.

The keys used to secure communications can be:

- An RSA key of size 2048 or larger.
- An EC key following a NIST-256 or NIST-384 curve.

Configure the PKCS#11 store with the following commands:

```
cd /home/pi/Downloads/tpm2-pkcs11/tools
./tpm2_ptool init --path=/opt/tpm2-pkcs11
```

The command returns the following output:

```
action: Created
id: 1
```

`id` indicates the number of the slot where the store has been initialized. It may be larger than 1 if other key stores have previously been initialized. The `id` number does not affect the content of this application note. For the following command, `id` is assumed to be 1:

```
./tpm2_ptool addtoken --pid=1 --sopin=adminPin --userpin=userPin --label=greengrass --
path=/opt/tpm2-pkcs11
```

Once the slot and the token have been created, one or two keys can be created to secure communications. Each different key must have a different key-label.

To create an RSA 2048 key, use the following command line:

```
./tpm2_ptool addkey --algorithm=rsa2048 --userpin=userPin --label=greengrass --key-
label=greenkey --path=/opt/tpm2-pkcs11
```

To create an ECC 256 key, use the following command line::

```
./tpm2_ptool addkey --algorithm=ecc256 --userpin=userPin --label=greengrass --key-
label=greenkey --path=/opt/tpm2-pkcs11
```

To create an ECC 384 key, use the following command line::

```
./tpm2_ptool addkey --algorithm=ecc384 --userpin=userPin --label=greengrass --key-
label=greenkey --path=/opt/tpm2-pkcs11
```

For the rest of this application note, it is assumed that a single shared key with the label `“greenkey”` is used to secure communications. If multiple keys are used, each key needs an associated certificate.

To get a list of the tokens in the PKCS#11 store, run the following command:

```
p11tool --list-token-urls
```

The key URL can be obtained from the token URL, by adding the key label (`“object”` attribute) and the private `“type”` attribute:

```
pkcs11:token=greengrass;object=greenkey;type=private
```

Each key should have a unique key URL, associated with the unique key-label specified at key creation time.

8 Associating a certificate with the key

To ensure device authentication, each key must have an associated certificate. For this purpose, one can use *OpenSSL*® to generate a certificate signing request (CSR) for each key, with the following command:

```
openssl req -engine pkcs11 -new -key
"pkcs11:token=greengrass;object=greenkey;type=private" -keyform engine -out /home/pi/
certificate_request.csr
```

During the command execution, one has to give answers to prompted questions. The command uses these answers to generate the CSR file.

With this CSR file, to request a certificate for the key from the AWS Certificate Authority, do as follows:

Step 1. Log in to the AWS IoT Console. Refer to [AWS-IoT-Console_login] for the URL.

Step 2. On the left pane, select **Secure>Certificates**, then click on the **Create** button.

Step 3. Select the **Create with CSR** option.

Step 4. Browse for the `certificate_request.csr` file created earlier.

Step 5. Download the created certificate (…`-certificate.pem.crt`).

Step 6. Attach the policy created at the beginning of this application note to the certificate.

Step 7. Make sure that the certificate is activated.

Step 8. Download the root certificate from the Amazon server, if not already done at the "Setting up the Greengrass Group and its Core" stage,

Step 9. Copy the downloaded certificates to `/greengrass/certs` using the following commands:

```
cp /home/pi/Downloads/…-certificate.pem.crt /greengrass/certs/
cp /home/pi/Downloads/root.ca.crt /greengrass/certs/root.ca.pem
```

9 Configuring the AWS Greengrass Core software

The `/greengrass/config/config.json` file contains all the user-configurable options for the Greengrass Core daemon. This file needs to be modified, to use the keys that have been generated, and the PKCS#11 libraries.

The `config.json` file should look like this:

```
{
    "crypto": {
        "caPath": "file:///greengrass/certs/root.ca.pem",
        "PKCS11": {
            "P11Provider": "/usr/lib/arm-linux-gnueabihf/pkcs11/libtpm2_pkcs11.so",
            "SlotLabel": "greengrass",
            "SlotUserPin": "userPin"
        },
        "principals": {
            "IoTCertificate": {
                "certificatePath": "file:///greengrass/certs/xxxxxxxx-certificate.pem.crt",
                "privateKeyPath": "pkcs11: token=greengrass;object=greenkey;type=private"
            },
            "MQTTServerCertificate": {
                "certificatePath": "file:///greengrass/certs/xxxxxxxx-certificate.pem.crt",
                "privateKeyPath": "pkcs11: token=greengrass;object=greenkey;type=private"
            }
        }
    },
    "coreThing" : {
        "thingArn" : "arn:aws:iot:us-east-2:xxxxxxxxxxxx:thing/MyRPIGroup_Core",
        "iotHost" : "xxxxxxxxxxxxx.iot. us-east-2.amazonaws.com",
        "ggHost" : "greengrass.iot.us-east-2.amazonaws.com",
        "keepAlive" : 600
    },
    "runtime" : {
        "cgroup" : {
            "useSystemd" : "yes"
        }
    },
    "managedRespawn" : false
}
```

10 Running the IDT test suite

The AWS Partner Device Catalogue lists all the qualified hardware that works with AWS IoT Greengrass. To enter the Partner Device Catalogue, AWS provide a series of tests that the device running for qualification must pass. This test suite is provided through a software named the AWS Greengrass IDT (AWS IoT Device Tester). The IDT software is executed from a host computer running on Windows®, Mac®, or Linux®, connected to the board to be tested via Secure Shell (SSH).

10.1 Downloading and setting up the IDT

The Greengrass IDT (AWS IoT Device Tester) needs certain authorizations to run the tests. These can be given through an IAM (AWS Identity and Access Management) policy associated with an IAM user.

To create an IAM policy and an associated IAM user, follow step 2 of the [IAM-policy&user] guide.

Then, download the AWS Greengrass IDT (refer to [IDT] for the link) and extract it onto the host PC:

The IDT needs knowledge of the security resources downloaded in the `.csv` file for creating the IAM user. This information can be transferred through environment variables. To configure these variables, open `/home/$username/.bashrc` with a text editor, add the following lines to the end, then save the file:

```
export AWS_ACCESS_KEY_ID={your access key}
export AWS_SECRET_ACCESS_KEY={your secret access key}
```

10.2 Configuring the access to the device under test

In the extracted IDT folder, navigate to `configs`, then open the `config.json` file. Make sure that the region is set correctly. If you are unsure, set the region to `us-east-2`.

Make sure that the credentials are configured through environment variables:

```
"auth": {
    "method": "file",
    "credentials": {
        "profile": "environment"
    }
}
```

Then, edit the `device.json` file. It should look like this (where id "`RPIPool`" and SKU (stock keeping unit) name "`RPI3BwithHD8TPM`" are examples):

```
[
    {
        "id": "RPIPool",
        "sku": "RPI3BwithHD8TPM",
        "features": [
            {
                "name": "os",
                "value": "linux"
            },
            {
                "name": "arch",
                "value": "armv7l"
            },
            {
                "name": "container",
                "value": "yes"
            },
            {
                "name": "docker",
                "value": "yes"
            },
            {
                "name": "streamManagement",
                "value": "yes"
            },
            {
                "name": "hsi",
                "value": "yes"
            },
            {
                "name": "ml",
                "value": "no"
            }
        ],
        "hsm": {
            "OpenSSLEngine":
            "/usr/lib/arm-linux-gnueabihf/engines-1.1/pkcs11.so",
            "p11Provider": "/usr/lib/arm-linux-gnueabihf/pkcs11/libtpm2_pkcs11.so",
            "slotLabel": "greengrass",
            "slotUserPin": "userPin",
            "privateKeyLabel": "greenkey"
        },
        "kernelConfigLocation": "",
        "greengrassLocation": "/greengrass",
        "devices": [
            {
                "id": "deviceId",
                "connectivity": {
                    "protocol": "ssh",
                    "ip": "{Raspberry Pi IP address}",
                    "port": 22,
                    "auth": {
                        "method": "password",
                        "credentials": {
                            "user": "pi",
                            "password": "{Raspberry Pi password}"
                        }
                    }
                }
            }
        ]
    }
]
```

10.3 Running the test suite

If the AWS Greengrass daemon is not running on the Raspberry Pi, use the following command lines

```
cd /greengrass/ggc/core
sudo ./greengrassd start
```

If the AWS Greengrass daemon is running on the Raspberry Pi, it must be restarted to consider all the changes. Run:

```
cd /greengrass/ggc/core
sudo ./greengrassd stop
sudo ./greengrassd start
```

On the host PC, open a terminal, navigate to the extracted IDT's `bin` folder, and run the test suite with the following command:

```
devicetester_linux_x84-64 run-suite --pool-id RPIPool
```

DO NOT run the test suite as `sudo`, as this would cause tests to fail.

The test suite typically takes 30 minutes to run. A PASSED or FAILED result will appear for each test group in the terminal at the end of execution.

Refer to the logs folder to find more detailed results of the tests. To better understand these logs, refer to [Results-logs].

11 Optional: satisfying Docker® dependencies

To pass the optional Docker® tests in the AWS Greengrass test suite, one must install Docker Engine and Docker Compose.

Use the following convenience script to install Docker Engine:

```
sudo apt-get remove docker docker-engine docker.io
curl -fsSL https://get.docker.com(1) -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker ggc_user
sudo apt-get install -y libffi-dev python3 python3-pip
sudo apt-get remove python-configparser
sudo pip3 -v install docker-compose
```

To test Docker Engine installation, run the following command:

```
sudo docker run hello-world
```

To test Docker Compose installation, run the following command:

```
docker-compose --version
```

12 Troubleshooting

`/dev/tpm0` is not present.

Make sure that the device tree, driver and kernel configurations were all edited correctly, and that the STPM4RasPI board is correctly plugged in.

The Greengrass Core daemon is not starting.

AWS provide a dependency checker script for Greengrass setup. Download it and run it with the following commands:

```
cd /home/pi/Downloads
mkdir greengrass-dependency-checker-GGCv1.11.x
cd greengrass-dependency-checker-GGCv1.11.x
wget https://github.com/aws-samples/aws-greengrass-samples/raw/master/greengrass-dependency-checker-GGCv1.11.x.zip(1)

unzip greengrass-dependency-checker-GGCv1.11.x.zip
cd greengrass-dependency-checker-GGCv1.11.x
sudo modprobe configs
sudo ./check_ggc_dependencies | more
```

In addition, make sure that the PKCS#11 key store is configured correctly.

The following command allows one to display a list of tokens:

```
p11tool --list-token-urls
```

The following command allows one to display a list of keys associated with a token:

```
p11tool --login --list-keys
        'pkcs11:token=greengrass'
```

Appendix A Referenced documents

Table 5. Document references

Document reference	Document title
[TPM 2.0 r138]	TPM Library, Part 1, Part 2, Part 3, Part 4, Family 2.0, rev 1.38, TCG
[TPM 2.0 rev138 Err]	TPM Library, Family 2.0, rev 1.38, Errata, TCG
[TPM TSS 2.0]	TCG TSS 2.0 Overview and Common Structures Specification, version 0.90, revision 03, October 2 2019, TCG

Appendix B Glossary

Table 6. List of acronyms

Acronym	Meaning
AWS™	Amazon Web Services™
CSR	Certificate signing request
EC	Elliptic curve
ECC	Elliptic curve cryptography
I²C	Inter-integrated circuit
IAM	AWS Identity and Access Management
IDT	AWS IoT Device Tester
IoT	Internet of Things
MQTT	Message queuing telemetry transport
NIST	National Institute of Standards and Technology
PC	Personal computer
PKCS	Public key cryptographic standards
RSA	Public-key signature algorithm (Ron Rivest, Adi Shamir and Leonard Adleman)
SKU	Stock keeping unit
SPI	Serial peripheral interface
SSH	Secure Shell
TCG	Trusted Computing Group®
TPM	Trusted platform module
TSS	TPM software stack
URL	Uniform resource locator

Revision history

Table 7. **Document revision history**

Date	Revision	Changes
16-Sep-2021	1	Initial release.

Contents

List of tables