zymbit

Getting Started Guide for AWS IoT Greengrass

ZYMKEY4

Table of Contents

1 Document information

1.1 Document revision history

DATE	VERSION	COMMENTS
7/7/2023	1.0	Initial release
7/12/2023	1.1	Incorporated AWS feedback

1.2 Applicable operating systems for this guide

This guide will show you how to setup AWS IoT Greengrass v2 HSI features secured by a Zymbit ZYMKEY4 on a Raspberry PI4, running Raspberry PI OS Bullseye 64bit. The instructions include how to run an example script accessing the Zymbit Python API. Also included are the setup steps to test with AWS IoT Device Tester for AWS IoT Greengrass V2.

2 Overview

The Zymbit ZYMKEY4 provides private keys for X.509 certificate maintained in hardware through a PKCS#11 key store. This feature enables you to securely store the device's private key and certificate so that they aren't exposed or duplicated in software.

The AWS IoT Greengrass Core software uses a private key and X.509 certificate to authenticate connections to the AWS IoT and AWS IoT Greengrass services. The ZYMKEY4 can be configured to use either of the supported AWS IoT Greengrass signature schemes, RSA-2048 or ECC keys.

3 Hardware description

3.1 Datasheet

ZYMKEY4 Datasheet

3.2 Standard kit contents

The ZYMKEY4 ships individually with no other accessories. An optional cable is available providing an interface to tamper event circuitry.

For more details, please visit: ZYMKEY4 Getting Started

3.3 User provided items

A Raspberry Pi4 with an SD-CARD and USB-C 5VDC Power Supply. The power supply should be of good quality and provide at least 2.5 Amp. Ethernet cable. Optionally HDMI display, keyboard, and mouse.

3.4 3rd party purchasable items

The ZYMKEY4 is supported on the Raspberry Pi 4 Model B.

4 Set up your development environment

4.1 Tools installation (IDEs, Toolchains, SDKs)

The ZYMKEY4 software environment is provided as Debian packages that are installed from a script. The script will install all the necessary packages, services, and libraries.

```
curl -G https://s3.amazonaws.com/zk-sw-repo/install_zk_sw.sh | sudo bash
```

An API interface is available for use via Python, C, or C++. There are no modules or libraries that require compiling.

The ZYMKEY4 does not require an IDE or SDK per se. Any toolchains required would be standard Raspberry PI / Debian tools for your application.

4.2 Additional software references

For additional information on the Raspberry Pi Operating System itself, see Raspberry Pi OS
See docs.zymbit.com for additional information and examples.

5 Set up device hardware

The first step is to set up your PI itself. There are many tutorials and videos online to assist you. The Raspberry PI Foundation provides a tutorial to help you here.

The ZYMKEY occupies the first 10 pins of the Raspberry PI GPIO header. The instructions for setting up a Raspberry PI with a ZYMKEY can be found here: ZYMKEY4 Getting Started

6 About AWS IoT Greengrass

To learn more about AWS IoT Greengrass, see How AWS IoT Greengrass works and What's new in AWS IoT Greengrass Version 2.

7 Greengrass Hardware Security Integration

The AWS IoT Greengrass Core software uses a private key and X.509 certificate to authenticate connections to the AWS IoT and AWS IoT Greengrass services. AWS IoT Greengrass Core software can use a hardware security module (HSM) such as the Zymbit ZYMKEY4 through the PKCS#11 interface to protect that private key. The private key used to generate the device certificate is kept on the Zymbit module and never exposed.

PKCS#11 support for the Zymbit products is available through the Zymbit package called *zkpkcs11*. The zkpkcs11 package is installed along with the rest of the Zymbit software with the curl command referenced above.

The *zkpkcs11* package is based on the SoftHSM2 source code. Zymbit added two important extra features:

1. Zymbit private keys can be used for signing by specifying `--use-zkslot` when creating a new key object with `zk_pkcs11-util`. This only applies to ECC NIST-P256 (secp256r1).
2. Even though SoftHSM2 does key wrapping to protect its key objects, Zymbit goes a step further and protects all key material in its private object store with its data lock/unlock feature, even for slots that Zymbit products do not support, such as RSA. For example, if you wanted to setup a zkpkcs11 slot that was RSA, you could do that as well and, even though all actions would be done by OpenSSL in software on the host computer rather than the Zymbit module, the Zymbit module would still use its lock/unlock feature to protect the generated RSA private key.

For this Getting Started example, the first method will be used to sign with the Zymbit module based ECC 256 keys. We've made available scripts to help bootstrap the process.

8 Greengrass prerequisites

Refer to the online documentation detailing the prerequisites needed for AWS IoT Greengrass. Follow the instructions in the following sections:

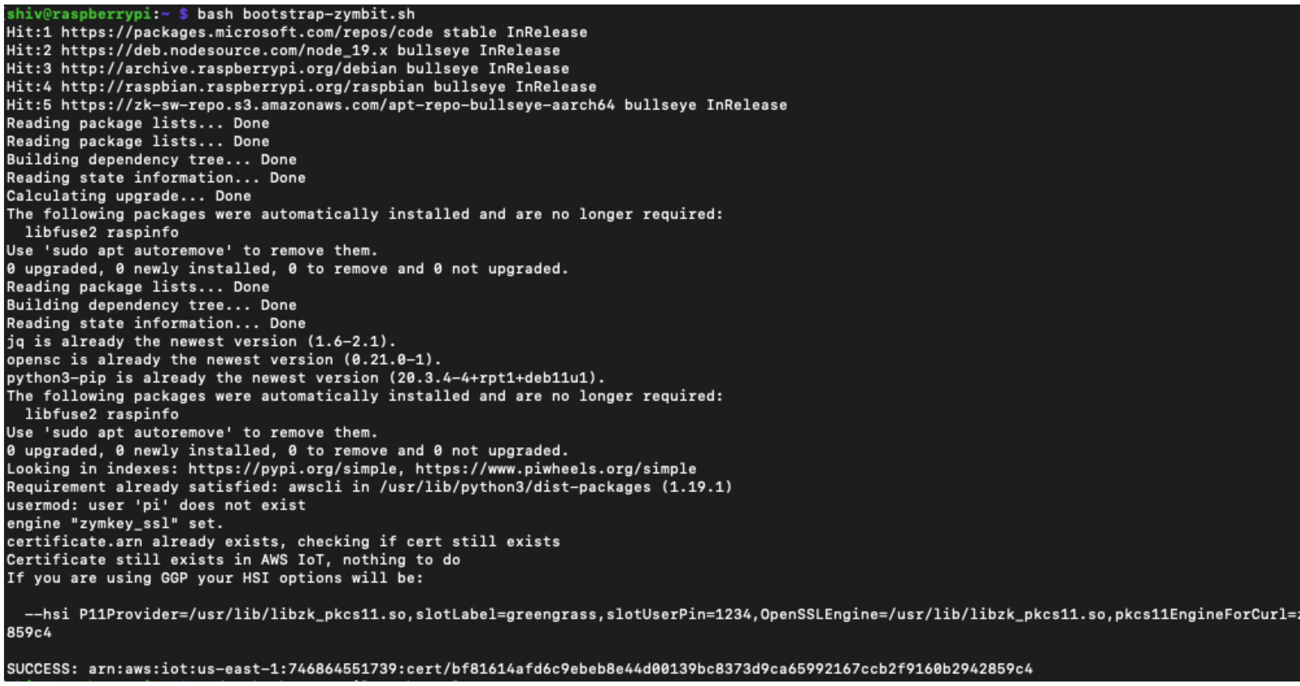
Step 1: Set up an AWS account
Step 2: Set up your environment

9 Install AWS IoT Greengrass

First, install and configure the AWS CLI on your IoT device. Then, follow the online guide to *Install with manual provisioning*. Refer to the instructions in the following steps:

- *Retrieve AWS IoT Endpoints*
- *Create an AWS IoT Thing*
- *Create the thing certificate* (Create the thing certificate from a private key in an HSM)

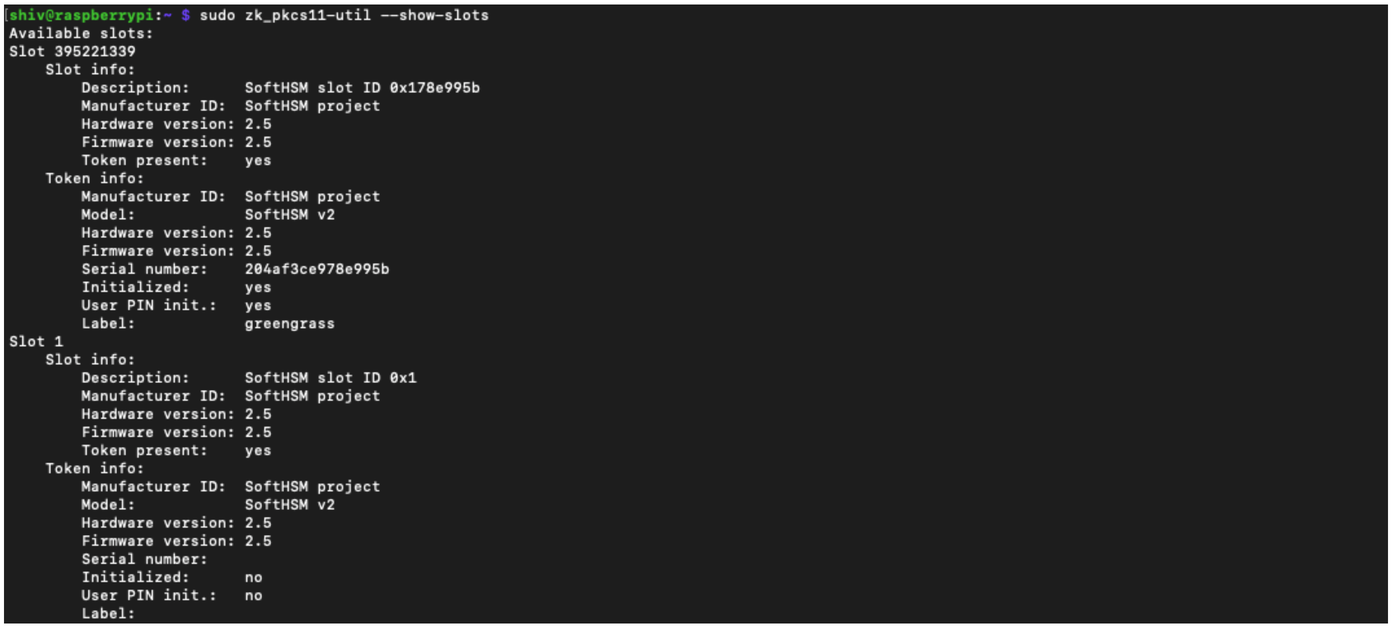
  Download the scripts from *this repository* onto your IoT device to automate this step
  Make sure your user has permission to execute them.
  If you want a more verbose output, you can uncomment the **set -x** command at the beginning of both scripts
  Make sure that the slot, pin, and id in bootstrap-zymbit.sh are what you want them to be based on your setup of PKCS11 on the HSM.
  Run bootstrap-zymbit.sh to generate the CSR. If done correctly, the script will automatically invoke bootstrap-common.sh and create the certificate using the HSM for AWS.
  Here is the example output on success:



```
shiv@raspberrypi:~ $ bash bootstrap-zymbit.sh
Hit:1 https://packages.microsoft.com/repos/code stable InRelease
Hit:2 https://deb.nodesource.com/node_19.x bullseye InRelease
Hit:3 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:4 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Hit:5 https://zk-sw-repo.s3.amazonaws.com/apt-repo-bullseye-aarch64 bullseye InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libfuse2 raspinfo
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
jq is already the newest version (1.6-2.1).
opensc is already the newest version (0.21.0-1).
python3-pip is already the newest version (20.3.4-4+rpt1+deb11u1).
The following packages were automatically installed and are no longer required:
  libfuse2 raspinfo
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: awscli in /usr/lib/python3/dist-packages (1.19.1)
usermod: user 'pi' does not exist
engine "zymkey_ssl" set.
certificate.arn already exists, checking if cert still exists
Certificate still exists in AWS IoT, nothing to do
If you are using GGP your HSI options will be:

  --hsi P11Provider=/usr/lib/libzk_pkcs11.so,slotLabel=greengrass,slotUserPin=1234,OpenSSLEngine=/usr/lib/libzk_pkcs11.so,pkcs11EngineForCurl=
859c4

SUCCESS: arn:aws:iot:us-east-1:746864551739:cert/bf81614afd6c9ebeb8e44d00139bc8373d9ca65992167ccb2f9160b2942859c4
```



```
shiv@raspberrypi:~ $ sudo zk_pkcs11-util --show-slots
Available slots:
Slot 395221339
    Slot info:
        Description:      SoftHSM slot ID 0x178e995b
        Manufacturer ID:  SoftHSM project
        Hardware version: 2.5
        Firmware version: 2.5
        Token present:    yes
    Token info:
        Manufacturer ID:  SoftHSM project
        Model:            SoftHSM v2
        Hardware version: 2.5
        Firmware version: 2.5
        Serial number:    204af3ce978e995b
        Initialized:      yes
        User PIN init.:   yes
        Label:            greengrass
Slot 1
    Slot info:
        Description:      SoftHSM slot ID 0x1
        Manufacturer ID:  SoftHSM project
        Hardware version: 2.5
        Firmware version: 2.5
        Token present:    yes
    Token info:
        Manufacturer ID:  SoftHSM project
        Model:            SoftHSM v2
        Hardware version: 2.5
        Firmware version: 2.5
        Serial number:
        Initialized:      no
        User PIN init.:   no
        Label:
```

- Continue the *Install with manual provisioning* guide at *Configure the thing certificate* until the end.

10 Create a Greengrass component

10.1 Create the component on your edge device

Below is an example component that you can use as the HelloWorld component for this section. This component will showcase some of Zymbit's API functions running as a Greengrass component in addition to AWS's basic HelloWorld example.

```
import zymkey
import sys

message = "Hello, %s!" % sys.argv[1]

# Print the message to stdout, which Greengrass saves in a log file.
print(message)

# Get timestamp from the Zymkey's RTC
time = zymkey.client.get_time()
print("GMT Time from Zymkey's RTC: ", time)

# Get the CPU's Temperature
temp = zymkey.client.get_cpu_temp()
print("CPU Temperature: ", temp)

# Get Model Number
model_num = zymkey.client.get_model_number()
print("Zymkey Model Number: ", model_num)

# Get Firmware Version
firmware_version = zymkey.client.get_firmware_version()
print("Zymkey Firmware Version: ", firmware_version)
```

Follow the instructions online under the section Develop and test a component on your device to create a simple component on your device.

10.2 Upload the Greengrass component

Follow the instructions online at Create your component in the AWS IoT Greengrass service to upload your component to the cloud, where it can be deployed to other devices as needed.

10.3 Deploy your component

Follow the instructions online at Deploy your component to deploy and verify that your component is running.

11 Troubleshooting

A troubleshooting section for the ZYMKEY4 can be found on our doc server:

General Troubleshooting
ZYMKEY4 Troubleshooting

The ZYMBIT Community pages can also be helpful

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’
```

If you need additional support, please contact
For more information, refer to the online documentation *Troubleshooting Greengrass v2*.