USER'S GUIDE

Apollo4 and Apollo4 Blue OEM Provisioning Process and Tools

Ultra-Low Power Apollo SoC Family

A-SOCAP4-UGGA01EN v1.2

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	October 10, 2020	Initial Release.
1.1	October 13, 2020	Added Debug Certificate chain picture.
1.2	October 22, 2020	Added OEM Asset Generation tool information.

Reference Documents

Document ID	Description
A-SOCAP4-WPNA01EN	Apollo4 and Apollo4 Blue SoC Security Features White Paper
A-SOCAP4-UGGA02EN	Apollo4 and Apollo4 Blue Secure Update User's Guide
	Apollo4 and Apollo4 Blue Secure Bootloader Scripts User's Guide
	Apollo4 and Apollo4 Blue Getting Started Guide

Table of Contents

SECTION

1

Introduction

This document provides an overview of the OEM provisioning process for Apollo4 and Apollo4 Blue SoC and the details of the tools required including OEM certificate generation. OEM provisioning is the process of creating digital security assets in a form that is suitable for the customer's device production flow. Security assets include OEM symmetric keys, Root of Trust, and any security sensitive data provisioned at the time of device manufacturing.

1.1 System Requirements

The provisioning tools are designed to run on a Linux host machine with the following requirements:

- Linux – Kernel Version 4.15.0 – 107-generic (Ubuntu 16.04.2).
- OpenSSL – 1.0.2g
- Python – 3.5.2

1.2 Terminology

This section defines some of the terminologies used in this document.

Table 1-1: Terminology

Abbreviation	Definition
ICV	Integrated Circuit Vendor
DM LCS	Device Manufacturer Life Cycle State
OEM	Original Equipment Manufacturer
RoT	Root of Trust
RMA	Returned Merchandise Authorization

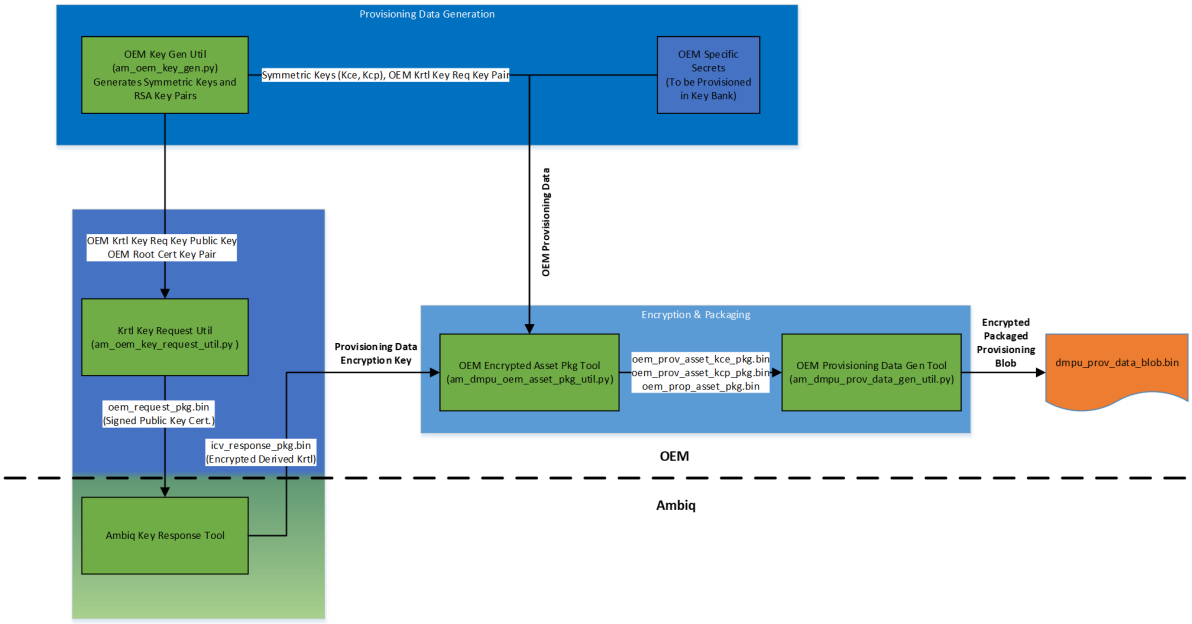
SECTION

2

OEM Provisioning Data Gen Flow

Figure 2-1 shows the general flow for OEM security asset generation for Apollo4 and Apollo4 Blue. The following sections outline the operation of the tools within each of blocks.

Figure 2-1: General Flow for OEM Security Asset Generation



2.1 Key Gen Utility

The Key Gen Utility is used to generate all the keys required for OEM secrets provisioning at the device manufacturing. The following python utility can be used to generate the keys.

```
./oem_tools_pkg/am_oem_key_gen_util/am_oem_key_gen_util.py
```

2.1.1 Input Parameters

The inputs to the Key Gen Utility are provided through the configuration file as a command-line parameter. The inputs configured in the configuration file is described in the example config file itself at the following location:

```
./oem_tools_pkg/am_oem_key_gen_util/oemKeyGenConfig.cfg
```

2.1.2 Key Gen Command

The following are the Key Gen Command:

```
./oem_tools_pkg/am_oem_key_gen_util/$
python am_oem_key_gen_util.py ./oemKeyGenConfig.cfg
```

At successful execution, the Key Gen Utility should create the following two folders containing all the required symmetric and asymmetric keys for provisioning. It also generates the asymmetric keys for OEM certificate chain and debug/RMA certificates.

```
./oem_tools_pkg/am_oem_key_gen_util/oemAesKeys
./oem_tools_pkg/am_oem_key_gen_util/oemRsaKeys
```

2.2 HBK Gen Utility

The HBK Gen Utility generates the Root Of Trust for the OEM (HBK 1).

```
./oem_tools_pkg/cert_utils/am_hbk_gen/am_hbk_gen_util.py
```

2.2.1 Input Parameters

The parameters to the HBK Gen Util are command-line parameters as shown below.

```
python am_hbk_gen_util.py -key <path to the OEM Root Public key> -endian <B |
L> -hash_format <SHA256 | SHA256_TRUNC >
```

2.2.2 HBK Gen Command

The command below will generate the truncated hash of the OEM Root Certificate Public key.

```
./oem_tools_pkg/cert_utils/am_hbk_gen/$
```

```
python am_hbk_gen_util.py -key ../../am_oem_key_gen_util/oemRSAKeys/
oemRootCertPublicKey.pem -endian L -hash_format SHA256_TRUNC
```

The output files will be generated at the following output folder:

```
./oem_tools_pkg/cert_utils/am_hbk_gen/hbk_gen_util_outPut
```

2.3 OEM Key Request Utility

The OEM Key Request Utility is used to generate the derived-Krtl-key-request-certificate to be processed at ICV secure labs. The output of this utility is sent to ICV which contains the key request public key. This is used by the ICV response utility to encrypt the derived Krtl key used for encrypting the OEM assets.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_key_request/
am_oem_key_request_util.py
```

2.3.1 Input Parameters

The inputs to the OEM Key Request Utility is provided through the configuration file as a command line parameter. The details of the config file parameters are described in the example config file itself.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_key_request/am_config/
am_dmpu_oem_key_request.cfg
```

2.3.2 OEM Key Request Command

The following command will generate the key request binary data:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_key_request/$
python am_oem_key_request_util.py ./am_config/ am_dmpu_oem_key_request.cfg
```

The output file containing the key request data is generated as follows:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_key_request/oem_request_pkg.bin
```

The key request should be delivered to Ambiq which will process the request and return the **icv_response_pkg.bin** file.

2.4 OEM Asset Packaging Utility

The OEM Asset Packaging Utility is used to encrypt the OEM assets before sending it to the device manufacturing to provision the encrypted assets securely.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/
am_dmpu_oem_asset_pkg_util.py
```

This utility mainly encrypts three plain text assets (Kcp, Kce, and OEM secrets) separately and generate three encrypted assets blobs.

The plain text OEM-secrets and **icv_response_pkg.bin** received from ICV can be placed at the following location as described in the example config file:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/inputData/
```

2.4.1 Input Parameters

The three config files for each asset are as follows.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/am_config/
oem_asset_enc.cfg
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/am_config/
am_asset_oem_ce.cfg
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/am_config/
am_asset_oem_cp.cfg
```

2.4.2 Command(s) For Asset(s) Generation

The following command(s) are used to generate the encrypted OEM assets:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/$
python am_dmpu_oem_asset_pkg_util.py ./am_config/oem_asset_enc.cfg
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/$
python am_dmpu_oem_asset_pkg_util.py ./am_config/am_asset_oem_ce.cfg
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/$
python am_dmpu_oem_asset_pkg_util.py ./am_config/ am_asset_oem_cp.cfg
```

The output files containing the encrypted assets should be generated as follows.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/oem_asset_pkg.bin
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/
oem_prov_asset_kce_pkg.bin
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/
oem_prov_asset_kcp_pkg.bin
```

2.5 OEM Asset Gen Util

The OEM Asset Gen Utility generates a binary file used to initialize OEM OTP security settings. These setting are available to set in the file named **oem_asset_gen.cfg**.

The tool creates a 2k binary data file that is an input to the OEM Provisioning Data Gen Utility discussed in the next section.

2.5.1 Input Parameters

OEM OTP security setting can be modified via the configuration file named **oem_asset_-gen.cfg**:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/oem_asset_gen_util.py
```

The following config file is used to generate the OEM provision blob.

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/am_config/
oem_asset_gen.cfg
```

2.5.2 Tool Execution

The following command will be used to generate the encrypted OEM assets blob:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/$
python oem_asset_gen_util.py ./am_config/oem_asset_gen_cfg
```

The output of the tool generates as follows:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/oem_asset_test_data.bin
```

2.6 OEM Provisioning Data Gen Util

The OEM Provisioning Dara Gen Utility generates the final encrypted OEM provisioning blob which is decrypted by the OPT tool on the device before provisioning the data.

The tool mainly joins the three encrypted OEM assets generated by the OEM Asset Packaging Utility. It also adds default DCU, initial software version and more to the final blob which is provided through the config file.

The utility is available at the following path.

2.6.1 Input Parameters

The following is the input parameter:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/
am_dmpu_prov_data_gen_util.py
```

The following config file is used to generate the OEM provision blob:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/am_config/
am_dmpu_data_gen.cfg
```

2.6.2 Tool Execution

The following command will be used to generate the encrypted OEM assets blob:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/$
python am_dmpu_prov_data_gen_util.py ./am_config/am_dmpu_data_gen_cfg
```

The output of the tool generates as follows:

```
./oem_tools_pkg/oem_asset_prov_utils/oem_asset_package/dmpu_prov_data_blob.bin
```

2.7 OEM Provisioning Tool (OPT)

The OPT is a Ambiq signed tool which is downloaded and executed on the chip during device production in the OEM manufacturing facility.

The tool is downloaded in SRAM at address 0x10030000, and the OEM encrypted assets generated as per *Section 2.6.2 Tool Execution on page 12*, at the SRAM address 0x10037000. After downloading these 2 blobs, the device needs to be reset. At the following boot, the OEM assets get provisioned if both the blobs are authenticated successfully. After a successful OEM provisioning, the device will reset, and is transitioned to secure LCS.

Note: If the device provisioning fails for some reason, it will remain in DM LCS.

The OPT tool is available at the following location:

```
./oem_tools_pkg/oem_prov_tool/opt_image_pkg.bin
```

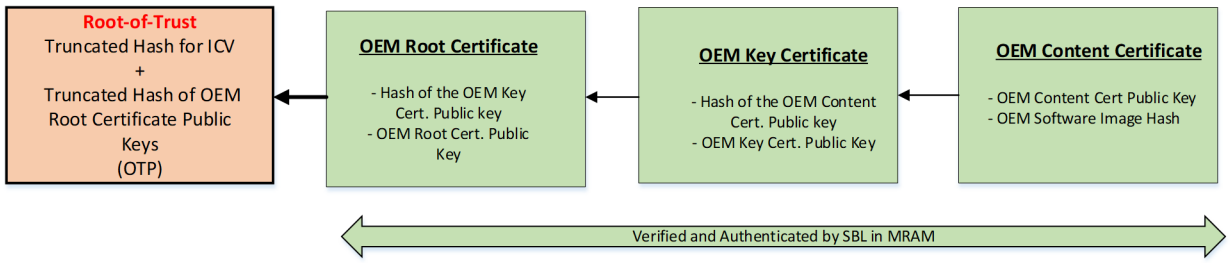
SECTION

3

OEM Certificate Generation

Provisioning of Apollo4 and Apollo4 Blue in the DM LCS depends upon a "chain" of public key certificates as shown in Figure 3-1. This method provides flexibility and additional security in case that the content changes or a certificate in the chain is compromised.

Figure 3-1: OEM Certificate Generation



3.1 Prerequisite

The OEM certificate chain generation process assumes that OEM RSA keys have already been generated using the KeyGen Utility during the provisioning data generation process. The OEM Certificate chain should use these same OEM RSA keys.

While processing the Certificate Chain authentication on the device, it is also assumed that the device is already provisioned with the Root of Trust (HBK1 in the OTP), as the OEM Root Certificate needs to be authenticated using the RoT.

3.2 Note on the Tool Output Files

Many of the steps below will generate both ***.txt** files that are appropriate to cut and paste into source code files for testing. However, the tools also generate ***.bin** files that provide the assets for later steps to produce a provisioning blob.

3.3 OEM Root Certificate Gen

The OEM Root Certificate is used to validate the Public key provided by the OEM. It also authenticates the Public key embedded in the OEM key certificate, which is next in the chain.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util.py
```

3.3.1 Input Parameters

The inputs to the OEM Root Certificate Gen is provided through the configuration file as a command-line parameter. The details of the configuration file parameters are described in the following with example configuration values.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_config/am_oem_root_cert_hbk1.cfg
```

3.3.2 OEM Root Cert Gen Command

The following command generates the OEM Root Certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_key_util.py ./am_config/am_oem_root_cert_hbk1.cfg
```

The output file containing the OEM Root Certificate is generated in binary and text formats:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/
oem_root_cert_hbk1.bin
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/
oem_root_cert_hbk1_Cert.txt
```

3.4 OEM Key Certificate Gen

The OEM Key Certificate Utility generates the OEM Key Certificate to validate the Public key in the Content Certificate which is next in the OEM Certificate Chain:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util.py
```

3.4.1 Input Parameters

The input to the OEM Key Certificate Gen is provided through the configuration file as a command-line parameter. The details of the configuration file parameters are described in the following with example configuration values.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_config/am_oem_key_cert.cfg
```

3.0.1 OEM Key Certificate Gen Command

The following command generates the OEM Key Certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_key_util.py ./am_config/am_oem_key_cert.cfg
```

The output file containing the OEM Key Certificate is generated in binary and text formats:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/
oem_key_cert.bin
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/oem_key_cert_Cert.txt
```

3.5 OEM Content Certificate Gen

The OEM Content Certificate is used to authenticate OEM software images on the device. It contains a list of software images, along with the start addresses and their sizes.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_content_util.py
```

3.5.1 Input Parameters

The inputs to the OEM Content Certificate Gen Tool is provided through the configuration file as a command-line parameter. The details of the configuration file parameters are described in the following with example configuration values.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_config/am_oem_cnt_cert.cfg
```

The list of software images is listed in a text file that is used by the config file as mentioned above. The example list file is placed at the following location:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/inputData/images_table.tbl
```

The format of the image table is as follows:

- **ImageName** - Path to the image file.
- **RAMloadAdd** - DTCM/Shared SRAM address where an encrypted image needs to be decrypted and executed. 0xFFFFFFFF if the image is not encrypted and is to be executed from MRAM.
  Note: The *load-verify-scheme* in **am_oem_cnt_cert.cfg** must be set to '1'.
- **flashStoreAdd** – Address in MRAM to store and execute the image.
- **Enc-Scheme** - 0 - plain text, 1 encrypted.
  Note: Due to RAM limitations on Apollo4 and Apollo4 Blue, the encrypted option cannot be supported and should not be selected.

3.5.2 OEM Content Certificate Gen Command

The following command will be used to generate the content certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_content_util.py ./am_config/am_oem_cnt_cert.cfg
```

The output file containing the Ambiq assets is generated as follows:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_content_util_output/
content_cert.bin
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_content_util_output/
content_cert_Cert.txt
```

SECTION

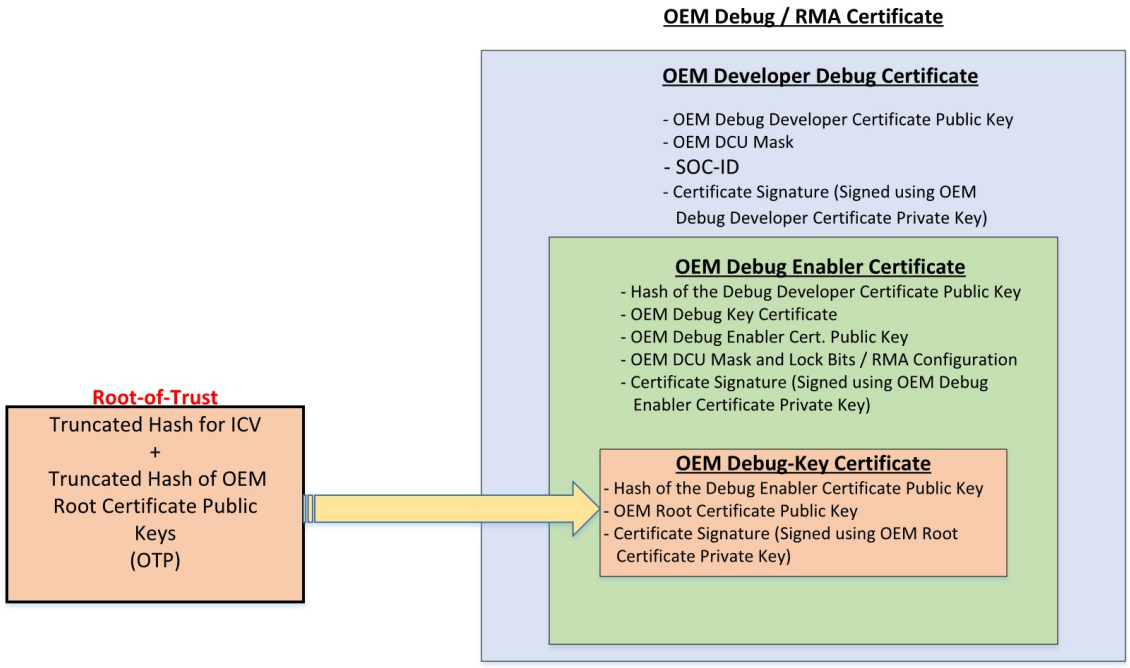
4

OEM Debug Certificate Gen

The debug certificate(s) is used to enable the debugging or changing the device RMA LCS for device analysis. The Debug certificates can be processed in DM and Secure LCS.

The debug certificate consists of three certificates debug-key certificate, Debug Enabler certificate, and Debug Developer certificate as shown in Figure 4-1. All these certificates need to be generated in the same sequence. The debug-key certificate is added to the debug enabler certificate using the Debug enabler certificate gen config file. Similarly, the debug enabler certificate is added to the debug developer certificate using the debug developer certificate gen config file. Finally, the debug developer certificate is downloaded on the device as a single entity to be processed by the SBR as three combined certificates.

Figure 4-1: OEM Debug Certificates



4.1 OEM Debug-Key Certificate Gen

The OEM Debug-Key Certificate is used to validate the Public key provided by the OEM. It also authenticates the Public key embedded in the OEM Enabler Certificate, which is next in the debug certificate chain.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util.py
```

4.1.1 Input Parameters

The inputs to the OEM Debug-Key Certificate Gen is provided through the configuration file as a command-line parameter. The details of the configuration file parameters are described in the following with example configuration values.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_config /am_oem_dbg_key_cert.cfg
```

4.1.2 OEM Debug-Key Gen Command

The following command generates the OEM Debug-Key Certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_key_util.py ./am_config/am_oem_dbg_key_cert.cfg
```

The output file containing the ICV Debug-Key Certificate is generated in binary and text formats as follows.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/
debug_oem_key_cert.bin
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_key_util_output/
debug_oem_key_cert_Cert.txt
```

4.2 OEM Debug Enabler Certificate Gen

The OEM Debug Enabler Certificate utility is used to generate the debug enabler certificate which contains the OEM DCU debug masks, and lock masks, or RMA information (in case of RMA certificate for OEM). It also authenticates the Public key embedded in the OEM debug developer certificate, which is next in the debug certificate chain.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_cert_dbg_enabler_util.py
```

4.0.1 Input Parameters

The inputs to the OEM Debug Enabler Certificate Gen is provided through the configuration file as a command-line parameter. The details of the configuration file parameters are described in the following with example configuration values.

```
./oem_tools_pkg/cert_utils/cert_gen_utils/am_config/
am_oem_dbg_enabler_cert.cfg
```

4.2.1 OEM Debug Enabler Certificate Gen Command

The following command generates the OEM Debug Enabler Certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_dbg_enabler_util.py ./am_config/am_oem_dbg_enabler_cert.cfg
```

The output file containing the OEM Debug Enabler Certificate is generated in binary format:

```
./oem_tools_pkg /cert_utils/cert_gen_utils/ am_debug_cert_output/
oem_dbg_cert_enabler_pkg.bin
```

4.3 OEM Debug Developer Certificate Gen

The OEM Debug Developer Certificate is used to generate the final debug certificate which contains OEM debug-key Certificate, OEM Debug Enabler Certificate, and SOC-ID (Chip specific Identification).

```
./oem_tools_pkg /cert_utils/cert_gen_utils/am_cert_dbg_developer_util.py
```

4.3.1 Input Parameters

The inputs to the OEM Debug Developer Certificate Gen is provided through the configuration file as a command-line parameter. The inputs configured in the configuration file is described in the example configuration file itself at the following location:

```
./oem_tools_pkg /cert_utils/cert_gen_utils/am_config/
am_oem_dbg_developer_cert.cfg
```

4.3.2 OEM Debug Developer Certificate Gen Command

The following command generates the OEM Developer Certificate:

```
./oem_tools_pkg/cert_utils/cert_gen_utils/$
python am_cert_dbg_developer_util.py ./am_config/am_oem_dbg_developer_cert.cfg
```

The output file containing the OEM Debug Developer Certificate is generated in binary and text ('c' Header file) formats as follows:

./oem_tools_pkg/cert_utils/cert_gen_utils/am_debug_cert_output/oem_dbg_cert_developer_pkg.bin

./oem_tools_pkg/cert_utils/cert_gen_utils/am_debug_cert_output/oemDeveloperCert.h

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 (512) 879-2850

A-SOCAP4-UGGA01EN v1.2
December 2020