USER'S GUIDE

Apollo4 Family Security

Ultra-Low Power Apollo SoC Family

A-SOCAP4-UGGA05EN v1.3

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	November 19, 2021	Initial release
1.1	August 1, 2022	▪ Updated content in section 7 Run-Time Security Services ▪ Updated Table 9-9 NVM Access Protection Field Description
1.2	October 19, 2022	▪ Updated document template ▪ Updated Figure 6-1 Table for LCS Defaults ▪ Added note on section 8 Factory Default SoC Configuration
1.3	July 5, 2023	▪ Updated and added notes in section 2 Introduction ▪ Added new content in section 8 Factory Default SoC Configuration ▪ Updated references to "Apollo4 and Apollo4 Blue..." with "Apollo4 Family..."

Reference Documents

Document ID	Description
A-SOCAP4-UGGA01EN	Apollo4 Family Provisioning, Update, and Tools User's Guide
A-SOCAP4-UGGA02EN	Apollo4 Family Secure Update User's Guide

Table of Contents

List of Tables

List of Figures

SECTION

1

Terminology

This section defines the abbreviation and terminology used in this document.

Table 1-1: Terminology

Abbreviation	Definition	Meaning
AES	Advanced Encryption Standard	
	BootROM	This is a ROM program that originally boots on reset and provides the initial immutable RoT.
	Certificate Chain	A dependent sequence of certificates that together can be used validate content in the SoC memory. The certificate chain is composed of key certificates and a content certificate. The advantage of a chain is that some of the links can change without installing a completely new chain.
CM LCS	Chip Manufacturing LCS	Ambiq acting with the fabrication, packaging, and test house partners (e.g., ICV) is considered the Chip Manufacturer. In this LCS, the SoC is mostly open and ready for initial provisioning with Ambiq security assets.
	Content Certificate	A certificate used to load and validate software components.
DCU	Debug Control Unit	CryptoCell-312 mechanism to control the debug mechanism and feature enablement.
DM LCS	Device Manufacturing LCS	These are typically Ambiq's direct customers and consumers of the Apollo4 SoCs (e.g., OEM). In this LCS, the customer provisions their security policy and assets.
HBK	Hash of the Public Key	Ambiq uses the Dual scheme: ▪ Hbk0: A 128-bit truncated SHA-256 digest of the ICV PubKB0. ▪ Hbk1: A 128-bit truncated SHA-256 digest of the OEM PubKB1.
HMAC	Hashed Message Authentication Code	
HUK	Hardware Unique Key	This is a 256-bit SoC-unique AES key that is used as the root for deriving all SoC-specific keys.
ICV	Integrated Chip Vendor	ICV (e.g., CM) is Ambiq acting with the fabrication, packaging, and test house partners.

Table 1-1: Terminology *(Continued)*

Abbreviation	Definition	Meaning
INFO0	InfoSpace Region #0	This is an area of memory inside the SoC that is designed to be programmed by the OEM and hold customer visible attributes include security assets.
INFO1	InfoSpace Region #1	This is an area of memory inside the SoC that is design to be programmed by the ICV to hold SoC trims and security assets.
IPT	ICV Provisioning Tool	A tool used by Ambiq's internal teams (software, validation, production test) to provision the secure assets of the SoC.
Kce	OEM Code Encryption Key	128-bit AES key that is used to decrypt software images as part of the Secure Boot process.
Kceicv	ICV Code Encryption Key	128-bit AES key that is used to decrypt software images as part of the Secure Boot process.
Kcp	OEM Provisioning Master Key	128-bit AES key that is used for the OEM asset provisioning flow.
	Key Certificate	A certificate used to validate the next certificate in a chain.
Kpicv	ICV Provisioning Master Key	128-bit AES key that is used for the ICV asset provisioning flow.
Krtl	Platform key	An ICV key placed in the RTL. It is used for provisioning during production life-cycle states.
LCS	Life Cycle State	The state of the SoC that determines the level of security and access to features.
OEM	Original Equipment Manufacturer	These are typically Ambiq's direct customers and consumers of the Apollo4 SoCs (e.g., DM).
OPT	OEM Provisioning Tool	A set of example scripts designed to run on a Linux OS computer for the purpose of providing an example of the OEM provisioning flow. It is envisioned that these scripts will be adopted by the OEM into their production flow.
OTP	One Time Programmable memory	This is an area of memory in the Apollo4 that can be programmed only once during provisioning. The OPT burns OEM keys and assets to the SoC OTP in DM LCS.
PKI	Public Key Infrastructure	A set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.
RMA LCS	Return Merchandise Authorization	This is the process of returning a faulty SoC. The RMA LCS is a special state that does not allow recovery of the SoC. As the RMA LCS is entered, the security assets for the ICV and OEM are erased.
RoT	Root of Trust	This is the original source that can always be trusted within a cryptographic system.
RSA	Rivest–Shamir–Adleman	An algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone.
SBR	Secure BootROM	This is actually a misnomer. The SBR is code found in a locked down region of MRAM. It is the first stage of processing after the actual BootROM in the Apollo4. This architecture was done to reduce the risk on the Rev B silicon.

Table 1-1: Terminology *(Continued)*

Abbreviation	Definition	Meaning
SE LCS	Secure Enable LCS	This is the state when the SoC is fully secured and content is authenticated and updated according to the established OEM security policy. Debugging is disabled. This is the proper state when the SoC is "in the field".
Secure OTA	Secure Over the Air programming	This is the process of authenticating and installing a "blob" of data that has been delivered to the SBL. This blob could be received over any radio (air) or wired interface (I^2C, SPI, SWO, UART).
TEE	Trusted Execution Environment	
Wire Update		This is the process and protocol of receiving a sequence of packets over a wired interface (I^2C, SPI, or UART).

SECTION

2

Introduction

NOTES:

- This document assumes "public side crypto enabled devices".
- This document applies to the Apollo4 family of SoCs. Any reference to Apollo4 assumes any device unless explicitly stated otherwise.
- References to **/tools/apollo4b_scripts/** directory applies to **/tools/apollo4l_scripts/** directory unless explicitly stated otherwise.
- Apollo4 and Apollo4 Plus devices utilize the same **/tools/apollo4b_scripts/** directory. Apollo4 Lite and Blue Lite devices use the **/tools/apollo4l_scripts/** directory.
- References to register documentation within AmbiqSuite, such as **/mcu/apollo4b/regs/**, apply for Apollo4 Plus and Apollo4 Lite, and are found within their respective directories in **AmbiqSuite/mcu/**.

The security features of the Apollo4 system on chip (SoC) enable a trusted execution environment for resource constrained wearable and IoT devices. The Apollo4 supports the Arm Platform Security Architecture (PSA) and provides a robust system level security leveraging Ambiq's SecureSPOT™ technology. The Apollo4 security includes Arm Cryptocell hardware cryptographic acceleration, One-Time Programmable (OTP) memory, a hardened BootROM, and a secure boot firmware in MRAM to ensure a complete end-to-end secure environment for customer applications. The Apollo4 establishes and maintains a Root-of-Trust (RoT) throughout SoC boot, reset, and firmware upgrade flows.

The specific security features supported by the Apollo4 SoC include:

- Secure Boot using Root-of-Trust in OTP
- Secure Over-the-Air (OTA) Updates
- Secure Wired Updates over I^2C, SPI, or UART
- Secure Key Storage in OTP
- Support for Secure and Non-secure modes of operation

- Lifecycle States (LCS) management
- Secure Debug using Secure Debug Certificates which provide selective debug controls and/or transition to RMA LCS.
- Support for Certificate Chain based authentication for customer image(s) to run, which are verified using PKI rooted in on chip RoT.
- Hardware Crypto Acceleration and True Random Number Generator (TRNG)
- CRC32
- Tools for asset provisioning during manufacturing
- Support for customer specific secondary boot loader (to be executed after Ambiq Secure Bootloader)

This document provides an overview to new customers on how to work with the SecureSPOT features of the Apollo4 SoCs. This document assumes the customer will supplement the information provided here with the Reference Documents noted earlier. Information presented in those documents is key to understanding this document in depth.

The Apollo4 SecureSPOT includes cryptographic hardware acceleration, one-time programmable memory, customer trimming, Secure BootROM, Secure Bootloader, runtime security API, and provisioning tools designed to operate as a complete package to ensure protection of the reader's assets.

The first part of this document focuses on the Apollo4 Security infrastructure overview, and second part on configuring, provisioning and usage.

The document will cover the following important topics:

- Apollo4 Security infrastructure
  - Apollo4 Secure Boot overview
  - Security Life Cycle States (LCS)
  - Secure Boot Mode
  - Debug Support
  - Run Time Security Services
- Configuration, Provisioning & Usage
  - Default SoC Configuration
  - OTP Configuration
  - OEM InfoSpace (INFO0) Configuration
  - Configuring Secure Boot Mode
  - Transition to Secure LCS
  - Transition to RMA LCS
  - Image Updates

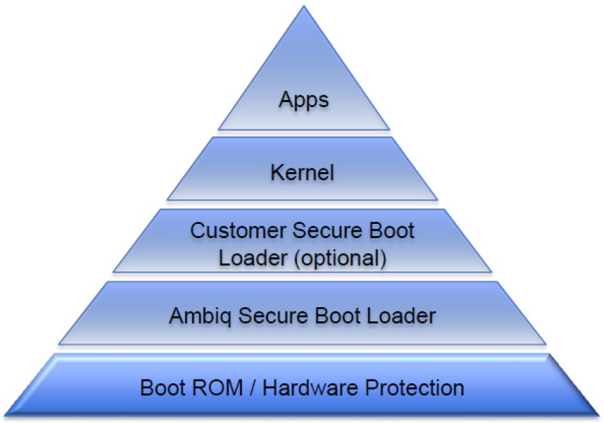
SECTION

3

Apollo4 Secure Boot Overview

At a high level, the Apollo4 Secure Boot architecture and features are modeled in Figure 3-1.

Figure 3-1: Apollo4 Secure Boot Architecture



Immutable Root of Trust

- Secure Boot ROM
- Hardware enforced
- Initiates chain of trust firmware validation using secure boot certificates
- Hardware crypto backed
- Lifecycle State management
- Establishes debug lockouts to prevent debug/tracing/interrupts during secure boot critical section execution based on secure debug certificates
- Hardware OTP key storage and security policy configuration

The base of the security model is the Root-of-Trust in OTP memory which is provisioned during chip and device manufacturing. Upon a power-cycle, the BootROM validates SoC trims and security settings, and then authenticates the Secure BootROM (SBR) in MRAM. Control is then passed to the Secure BootROM. The Secure BootROM provides for CM and DM state provisioning of the security assets and policies in the OTP memory. The SBR also performs Certificate Chain based verification of the Secure Bootloader (SBL) and processing of Secure Debug certificates. The Secure Bootloader includes support for OTA firmware updates, self-upgrade, wired updates, certificate chain updates, key revocation, trim patch updates, and certificate chain-based authentication of optional customer secondary bootloaders or customer main images.

3.1 Secure Boot Functional Components

3.1.1 Secure BootROM (Immutable)

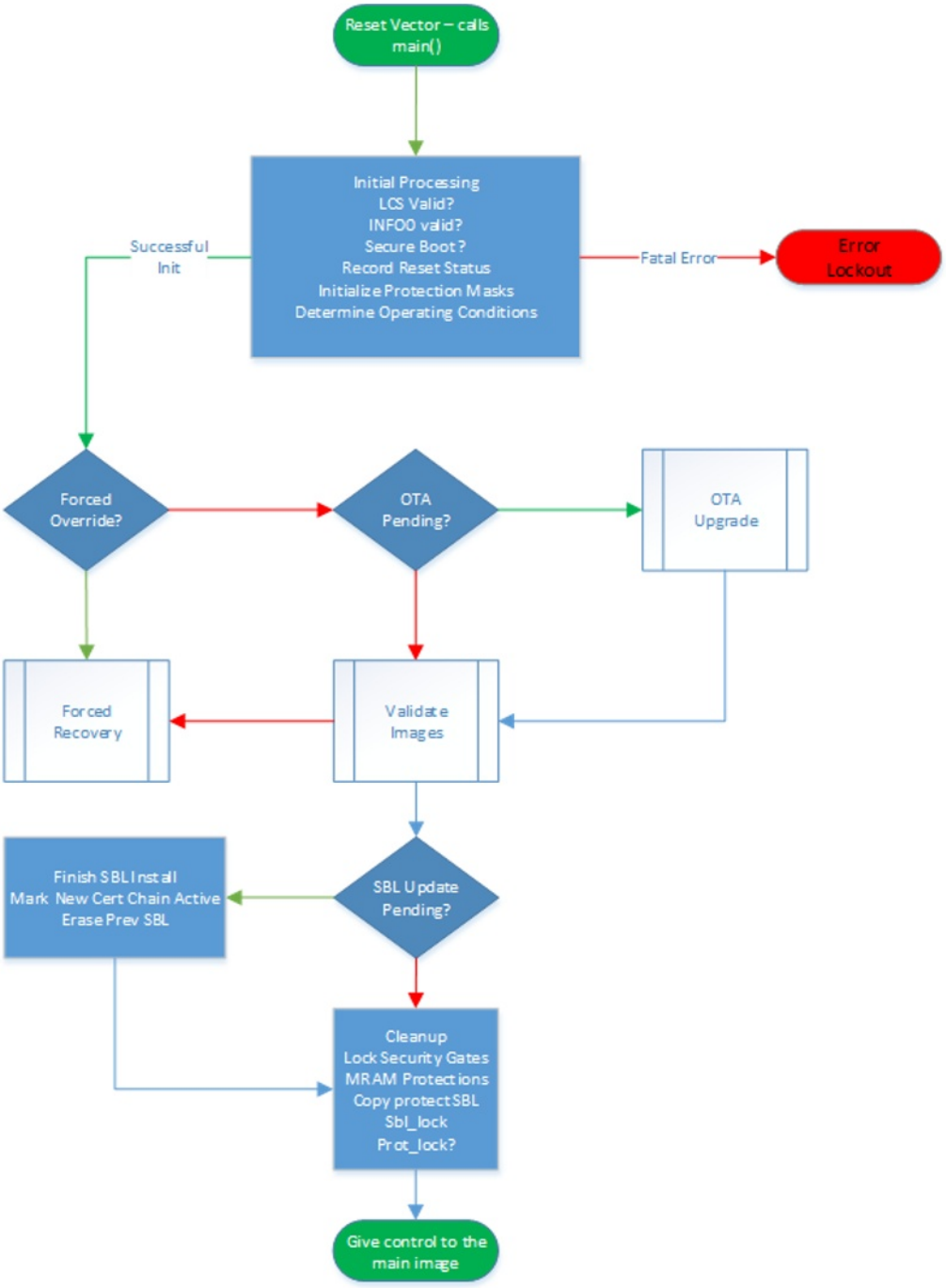
- Authentication of SBL using Secure Boot certificate to continue to boot flow
- ICV/OEM Provisioning support
- Life Cycle Management
- Secure Debug Certificate Processing

3.1.2 Secure Boot Loader (SBL) (Upgradeable)

- Authentication of Ambiq and Customer images
- Secure OTA
- Value Added Services
  - Support for Non-secure and Secure mode of operation
  - Enforcement of OTP security policies
  - Wired Updates
  - Trim Updates
- SBL Updates
- SBL can be in-field updated using secure OTA

The overall flow is shown in Figure 3-2 on page 15.

Figure 3-2: Secure Boot Loader (SBL) Flow



3.1.3 Customer Secure Bootloader

- Enhanced/Custom Services
- Proprietary Cryptographic Algorithms
- External Memory Device Drivers
- Custom Interface Drivers or Protocols

SECTION

4

Security Life Cycle States

4.1 Chip Manufacturer (CM) LCS

Ambiq acting with our fabrication, packaging and test house partners is considered the Chip Manufacturer. In CM LCS, the SoC is mostly open and ready for initial provisioning. In CM LCS, Ambiq provisions our secret assets (root keys), our Root of Trust and our security policy.

4.2 Device Manufacturer (DM) LCS

Ambiq's customers are considered Device Manufacturers. DM LCS is the initial state of the Apollo4 devices when received by customers. Ambiq provides tools that allow customers to provision their own secret assets, Root of Trust, and security policy. The customer can elect to operate the Apollo4 in non-secure mode and stay in DM LCS perpetually or they can operate in secure mode in Secure Enabled LCS.

4.3 Secure Enabled (SE) LCS

SE LCS is the state when the SoC is fully secured and content is authenticated and updated according to the established customer security policy. Debugging is disabled. This is the proper state of a fully secure device "in the field."

4.4 Return Merchandise Authorization (RMA) LCS

RMA LCS is activated when a faulty SoC needs to be returned to Ambiq for analysis and debug. RMA LCS is a special state and does not allow the recovery of the SoC. As RMA LCS is entered both Ambiq and customer security assets are erased.

SECTION

5

Secure Boot Mode

The Apollo4 SoC defaults as a "Secure" SKU, although it may be released as a "Non-Secure" SKU in the future. For the Secure SKU, there are two security "modes" which can be provisioned by the customer using OTP, as part of provisioning:

- Non-Secure Mode
  - Secure boot and some other secure services are not supported.
- Secure Mode
  - Secure boot and other secure services are supported.

An unprogrammed chip defaults to Non-secure mode.

The Apollo4 can be configured for Secure mode in both DM LCS, and Secure LCS. When configured for secure boot, SBL validates the boot images using pre-installed certificate chain, and any tampering or corruption of image would result in a boot failure.

> **NOTE:** When it comes to Ambiq installed assets (e.g., SBL), it is always verified using the same certificate chain approach, irrespective of whether Secure Mode is enabled or not.

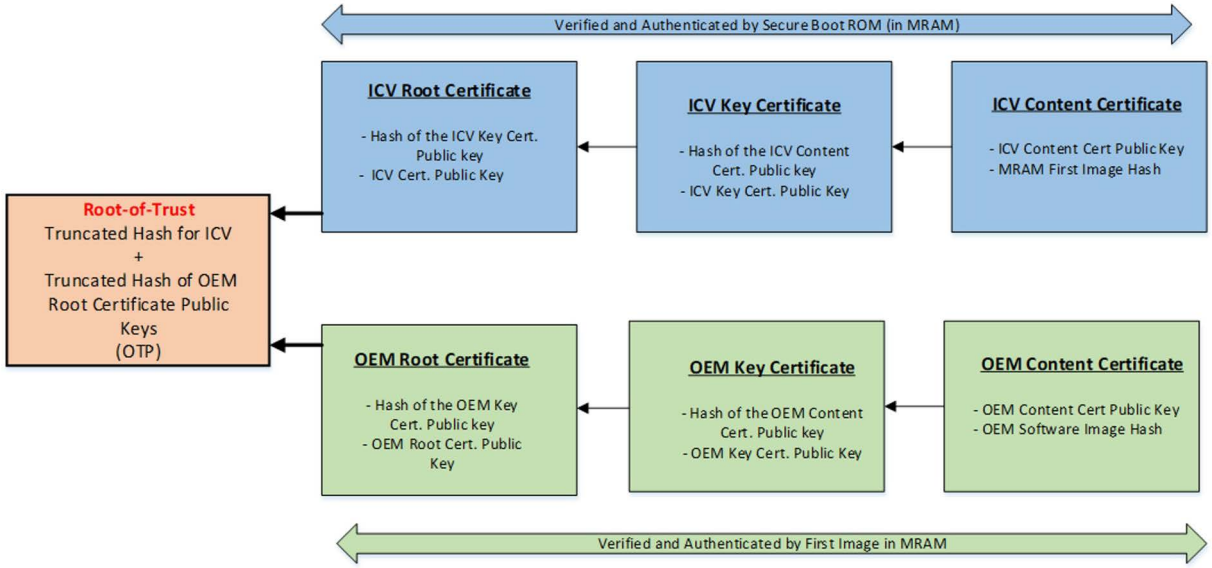
5.1 OEM Certificates and Certificate Chains

The Apollo4 security features support certificate chains for OEM content management as shown in Figure 5-1 on page 18. The Root-of-Trust for these chains is the same truncated public key hash in OTP mentioned above.

There are three certificates in the chain:

- OEM Root Certificate
- OEM Key Certificate
- OEM Content Certificate

Figure 5-1: Supported Certificate Chains for OEM Content Management



5.1.1 Verifying the Certificate Chain

Use the following procedure to verify the certificate chain:

1. Retrieve the public key from the certificate and calculate its hash.
   - For the OEM Root Certificate compare the hash to the HBK1.
   - For all other certificates compare the hash value to value stored from the previous certificate.
2. Verify the RSA signature:
   a. Calculate the hash of the certificate excluding the signature.
   b. Use the public key in the certificate to decrypt the signature, recreating the hash value.
   c. Compare the two hash values.
3. Check the software version information.

NOTE:
- The software version for the chain must be equal or larger than the software version stored in OTP.
- When in DM LCS, the verification step binding the Root Certificate to the RoT in OTP is skipped, as RoT is not programmed yet.

5.2 Certificate Formats

5.2.1 Key Certificates

The following structure is used for OEM Root and Key certificates.

Table 5-1: OEM Root and Key Certificate Structure

Structure	Field	Description
Certificate Header	magicNumber	Magic number to validate the certificate.
	certVersion	Certificate version to validate the certificate.
	certSize	Offset in words to the Certificate signature. And number of software components, if any exist.
	certFlags	Bit field interpreted according to the certificate type.
Certificate Public Key	N	3072-bit public key in big endian format
	Np	160-bit Barrett n' value
Certificate Body	swVer	Software version associated with certificate
	nextPubKeyHash	SHA256 hash result
Certificate Signature	sig	3072-bit RSA PSS signature

5.2.2 Content Certificate

The Content Certificate contains information about one or more images that need to be verified. The certificate contains an entry for each image, which has the address of the image in NVM, the size of the image, and the Hash corresponding to the image (for authentication). By convention, the content certificate Image[0] always corresponds to the main application image. Thus, upon successful verification, control is passed to this image.

Content Certificate can also contain optional protection configuration, instructing the SBL to apply either copy or write protection attributes to the images before handing off control to the main image.

Table 5-2: Content Certificate Structure

Structure	Field	Description
Certificate Header	magicNumber	Magic number to validate the certificate.
	certVersion	Certificate version to validate the certificate.
	certSize	Offset in words to the Certificate signature. And number of software components, if any exist
	certFlags	Bit field interpreted according to the certificate type.
Certificate Public Key	N	3072-bit public key in big endian format
	Np	160-bit Barrett n' value

Table 5-2: Content Certificate Structure *(Continued)*

Structure	Field	Description
Content	swVer	Software version associated with certificate
	nonce	pseudo-random arbitrary number
	imageRec[]	array of image records
	imageHash	SHA256 hash result
	loadAddr	MRAM address to load image
	imageMaxSize	Max Image size in bytes
	isAesCodeEncUsed	Image is encrypted (Not Supported)
	CopyProtect	Protect the MRAM sectors from Read
	WriteProtect	Protect the MRAM sectors from Write
Certificate Signature	sig	3072-bit RSA PSS signature

SECTION

6

Debug Support

The Apollo4 has hardware support for selective enabling of pre-identified debug features using Arm Cryptocell Debug Control Unit (DCU).

DCU controls are defined as 'qualified' bitmasks – with predefined significance for bits (see Global DCU Enabled in Figure 6-1). At the hardware level, each qualified bit is triple encoded for added security (raw DCU mask).

Hardware raw DCU bits Raw DCU bits (CRYPTO->HOSTDCU*2-3) are triple encoded: b'101 enables a feature, while b'010 disables it.

The following controls available for the raw DCU masks:

- Enable Bitmask (control individual debug feature accessibility)
  - Fixed Default Values based on LCS
- Lock Bitmask (once locked, the corresponding DCU bit cannot be modified till POI)
  - OTP configuration to lock the state of DCU (**`OTP_LOCKMASK*`**)

Figure 6-1: Table for LCS Defaults

	Enabled		Unlocked (Free to change)
	Disabled		Can be locked for further changes based on OEM controlled OTP settings

		Default				
	Debug Control	DM	Secure	RMA	Global DCU enable (Qualified)	Actual DCU bit with encoding
Runtime	CPU Debug - Invasive				1	[66:64]
	CPU Debug - Non-invasive				2	[69:67]
	CPU Tracing					
	ETM				3	[72:70]
	ITM				2	[69:67]
	TPIU/SWO				4	[75:73]
	DWT				4	[75:73]
	Perf Cnt / Energy Mon				5	[78:76]
	Cache Debug				6	[81:79]
	System Debug					
	SWD				11	[96:94]
	ETB				13	[102:100]
	TPIU/Trace Port Output				14	[105:103]

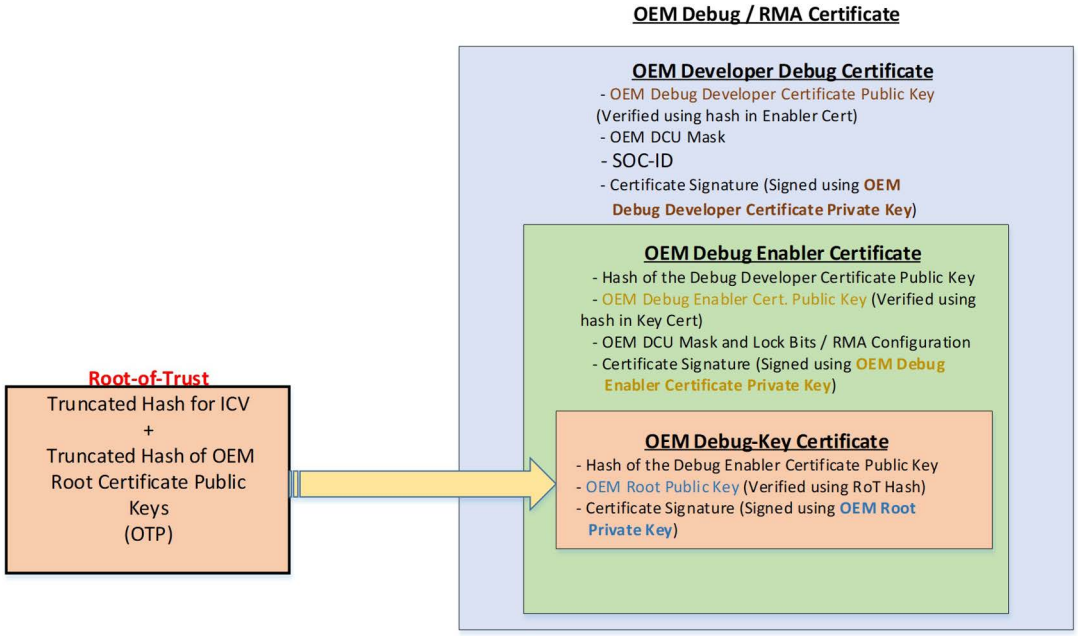
Enabling Debugging of the Apollo4 SoCs While in Secure LCS

In order to override default DCU values (based on LCS) and locking (based on OTP configuration), the customer must load a Secure Debug Certificate into NVM.

6.1 Secure Debug Certificates

Secure Debug Certificates are a set of two or three level certificates which contain public keys and authentication values. Figure 6-2 shows the relationship between the certificates.

Figure 6-2: Relationship Between Secure Debug Certificates



For the two-level SD certificate scheme, the chain order is:

- enabler debug certificate (signed using OEM Root Key) → developer debug certificate.

For the three-level SD certificate chain order is

- key certificate (signed using OEM Root Key) → enabler debug certificate → developer debug certificate.

The Key certificate is protected by the software certificate version (and hence can be revoked).

Ambiq recommends use of three level certificate chain as depicted here.

The function of each certificate in a three level Certificate is as follows:

- OEM Debug Key Certificate
  - Signed by OEM's Root Key (Hash Provisioned in OTP)
  - Contains the Hash of Public Key used to verify the Enabler Debug Cert
- OEM Debug Enabler Certificate
  - Signed by Enabled Debug Cert Key (Hash in Key Cert)
  - Contains the Hash of Public Key used to verify the Developer Debug Cert
  - Defines the subset of DCU bits open for editing by Developer
  - Overrides the DCU lock mask (supersedes the OTP configuration)
  - Applicable only to the LCS for which it is created
  - Also used for transition to RMA LCS (see next section)
- OEM Developer Debug Certificate
  - Signed by Developer Debug Cert Key
  - Defines the actual DCU overrides
  - Specific to a chip (SOCID) when in Secure LCS

Refer to the *Apollo4 Family Provisioning, Update, and Tools User's Guide A-SOCAP4-UGGA01EN* for details on how to generate Secure Debug or RMA Certificates.

6.2 Secure Debug Certificate Handling

As mentioned before, the customer must provide a mechanism to download the Secure Debug Certificate (SD Cert) on the device in production application. If enabled, Ambiq Secure Bootloader provides a means to download an SD Cert using wired interface, but OEM Security Policies are enforced for the download.

The location for the SD Certs are specified using INFO0 configuration. A subsequent Reset after installing the SD Cert will make the Debug overrides effective. Once installed, the SD Certs remain effective, until they are removed. The Debug overrides are in place till POI (even if the SD Certs is removed before next reset).

SD Certs are chip specific, when in Secure LCS and the 256b SOC-ID must be accessible to read through HAL API.

SECTION

7

Run-Time Security Services

AmbiqSuite relies on Arm CryptoCell engine to provide hardware acceleration for various security services to the main application.

Access to these services is provided through Run-time Software Library – maintained as part of open source effort at TrustZone CryptoCell-312 runtime library from Arm, which should be ported to respective OEM infrastructure. It supports Arm MBed™ TLS APIs to access the underlying hardware acceleration.

- Supported Algorithms: This is a non-exhaustive list of algorithms supported. (Refer to Arm® TrustZone® CryptoCell-312 documentation for reference)
  - Symmetric Key
    - AES 128/192/25
    - DES/TDES 64/128/192
    - AES MAC 128/192/256
    - AES-CCM 128/192/256
  - Asymmetric Key
    - RSA PKCS#1 2048/3072
    - ECC
    - DH
  - Hash / HMAC
    - SHA1, SHA2 (SHA224, SHA256, SHA384, SHA512)
    - MD5
- TRNG

> **NOTE:** Weak cryptographic algorithms including DES, TDES, SHA1, and MD5 are supported for legacy purposes only. Ambiq recommends against their use.

SECTION

8

Factory Default SoC Configuration

Ambiq ships the SoC in a factory default configuration as follows:

- Device Manufacturing (DM) LCS
- The customer's OTP space is not provisioned and set to 0's
- Device boots in non-secure mode
- Main image assumed at default (0x18000) address
- Debugger is open after Ambiq bootloader exits
- Debugger not allowed during Ambiq bootloader
- Default wired connections for Apollo4, Apollo4 Plus devices both Blue and non-Blue:
  - GPIO60 for TX
  - GPIO47 for RX
  - For Apollo4 and Apollo4 Plus (Blue and non-Blue), by default Wired Host connection is enabled with UART0, GPIO47 as Rx, and GPIO60 as Tx, no CTS/RTS, and a Baudrate of 115200. In addition, Secure Bootloader (SBL) will timeout after 500 msec
- Default wired connections for Apollo4 Blue Lite:
  - GPIO13 for TX
  - GPIO11 for RX
  - UART Module is Module 2
  - For Apollo4 Blue Lite, by default, the Wired Host connection is enabled with UART2, GPIO11 as RX, and GPIO13 as TX, no CTS/RTS, and a Baudrate of 115200. In addition, Secure Bootloader (SBL) will timeout after 500 msec.
- Default wired connections for Apollo4 Lite:
  - GPIO54 for TX
  - GPIO11 for RX
  - UART Module is Module 2
  - For Apollo4 Lite, by default, the Wired Host connection is enabled with UART2, GPIO11 as RX, and GPIO54 as TX, no CTS/RTS, and a Baudrate of 115200. In addition, Secure Bootloader (SBL) will timeout after 500 msec.
- INFO0 can be programmed with tools provided in the AmbiqSuite SDK to generate (create_info0.py) and program INFO0 (jlink-prog-info0.txt)

Default Factory parts boot in non-secure mode without any further configuration. The main image (or secondary bootloader) can be programmed and run from default (0x18000) address in non-volatile memory. Device can be used for development and non-secure testing without any further provisioning, thus INFO0 can be programmed – but is not necessary to get started.

The INFO0 and OTP programming overrides many of the default settings to allow the customization of security policy and assets.

NOTE: GPIO60 is unavailable on the Apollo4 Blue Plus KXR package, part number AMA4B2KP-KXR. INFO0 and OTP must be programmed via the SWD debugger interface before using UART wired updates. Reference *Section 4 of A-SOCAP4-UGGA01EN, Apollo4 Family OEM Provisioning, Update, and Tools User's Guide*, for more details.

SECTION

9

SecureSPOT Configuration

There are two memory areas that are open to the customer for configuration of the SecureSPOT features: One Time Programmable (OTP) and INFO0.

9.1 One Time Programmable (OTP) Memory

OTP is just like it sounds. OTP memory bit cells can be programmed from 0 to 1 while in DM LCS, but 1 to 0 is not possible. It can only be configured once and then permanently retains the configuration for the life of the part (or until RMA LCS when some of it is erased). This is done as a security measure to protect against tampering and other types of intrusions.

OTP is typically configured in a batch as part of provisioning at OEM device manufacturing.

As an alternative, it is also possible to manually and partially program the OTP, while still in DM LCS. All OTP area fields that get programmed as part of provisioning, are writable in DM LCS and thus can be provisioned programmatically (through OTP HAL API) if desired[1]. Transitioning OTP area bits from 0 to 1 is a one way operation.

Note that in all cases provisioning of RoT also results in DM to SE LCS transition and most of the fields are no longer writable in SE LCS.

The most up to date documentation of the OTP fields is provided along with the AmbiqSuite SDK here: **/mcu/apollo4b/regs/pages/am_mcu_apollo4b_otp.html**

In general, OTP contains the following:

- Root of Trust (RoT)

[1] As long as the partial provisioning configuration is subset of full provisioning data, even a partially provisioned part can accept DM Provisioning step for Secure LCS transition.

- Hardware Keys
  - OEM Symmetric Key used to authenticate the provisioning tools (KCP)
  - OEM Symmetric Key used for encryption/description (KCE)
- OEM Security Policy Selections (OTP_SECURITY and OTP_SEC_POL)
- Debug Control Unit (DCU) Overrides (OTP_DCU_DISABLEOVERRIDE)
- Wired Update Configuration (OTP_BOOT_OVERRIDE and OTP_WIRED_CONFIG)
- OEM Key Assets

Further documentation of the meaning of these fields is contained in *Section 14 Appendix on page 46*.

9.2 OEM Keys/Secure Assets

9.2.1 Asymmetric Root of Trust

The Apollo4 SecureSPOT requires the OEM to generate a key pair which in turn is used to generate the OEM portion (128-bits) of the SoC Root-of-Trust. This key pair is only known to the customer, not Ambiq and ensures that the customer has complete and final control over their own SoC security. Ambiq provides tools which accept the public key from the key pair and generate a truncated hash which is stored in OTP.

Table 9-1: Asymmetric Root of Trust Field Descriptions

Field Name	Description
OTP_ROT4-7	This field is the 128 bit OEM component of Root of Trust Words (HBK1)

9.2.2 Symmetric Keys

The Apollo4 SecureSPOT requires two symmetric keys to be provisioned into OTP. The first key KCP is a 128-bit root key designed to authenticate the OEM provisioning tool to the Apollo4 SoC. The second symmetric key KCE is another 128-bit root key designed to be used for OEM firmware image decryption. These keys cannot be read after the OEM provisions the device in DM LCS. They can be loaded into AES hardware engine directly for symmetric key crypto operations.

Table 9-2: Symmetric Keys Field Descriptions

Field Name	Description
OTP_KCP0-3	128 bit AES Key (referred to as Provisioning Key – KCP)
OTP_KCE0-3	128 bit AES Key (referred to as Code Encryption Key – KCE)

9.2.3 Custom Key Storage

In addition to the required secure assets, the Apollo4 provides additional key storage area (1KBytes) that can be used to store any OEM specific secrets. The areas are divided into four quadrants. The access policy for each quadrant is individually controlled by the customer. The customer can manage access to each quadrant by Ambiq's Secure Bootloader (SBL). The customer can also manage access through a designated Keybank Access Key, which is programmed during provisioning. Quadrants 0 and 1 are reserved for static keys programmed as part of DM provisioning. Quadrants 2 and 3 either be used for static keys or may be optionally used for one-time programmable (OTP) run time generated keys.

Table 9-3: Custom Key Storage Field Descriptions

Field Name	Description
OTP_KEYBANK_CUST_QUADRANT0_KEY0-63	Customer Keybank Quadrant 0
OTP_KEYBANK_CUST_QUADRANT1_KEY0-63	Customer Keybank Quadrant 1
OTP_KEYBANK_CUST_QUADRANT2_KEY0-63	Customer Keybank Quadrant 2
OTP_KEYBANK_CUST_QUADRANT3_KEY0-63	Customer Keybank Quadrant 3

See *Section 14 Appendix on page 46* for more information about the OEM Key Storage area.

9.3 OEM Security Policy

This section described the various settings that customers can use to establish a customized security policy that meets their product needs. Some of the settings are in INFO0 space while others are in OTP. Some of the fields are triple bit encoded to make electrical attacks more difficult. See *Section 14 Appendix on page 46* for more information about these fields.

9.3.1 Configuration of Secure Boot

Table 9-4: Secure Boot Field Descriptions

Field Name	Description
OTP_SECURITY : CUST_SECBOOT	This field determines if Secure Boot mode is enabled. Note that this is distinct from the SE LCS state, which is a lifecycle state. Devices in DM as well as SE LCS could be configured for non-secure boot, or secureboot based on this configuration. Secure-boot enabled devices go through a certificate chain based verification for the image by Ambiq SBL.
OTP_SECURITY : CUST_SECBOOTONRST	This field should only be used if Secure Boot (CUST_SECBOOT) is enabled. This field determines is the Secure Bootloader will execute the entire secure boot flow on every reset event. Selection of this options necessarily causes additional latency on resets.

9.3.2 Secondary Bootloader Configuration

Table 9-5: Secondary Bootloader Field Description

Field Name	Description
OTP_SECURITY : PLONEXIT	If configured to not lock it (value 0), the Secure Bootloader (SBL) keeps PROTLOCK open for customer secondary bootloader. This allows access to customer keybank area, if configured. In addition, PROTLOCK is used to deny/allow writing to the "Flash" (NVM) protection registers. OTP_SECURITY:PLONEXIT should be set, if no customer secondary bootloader present. This ensures that the main images do not have the ability to change NVM protection. Moreover, it also locks the OTP keybank from general access once SBL exits. When a secondary bootloader is present (OTP_SECURITY:PLONEXIT is 0), the secondary Bootloader should assert the PROTLOCK upon exit: MCUCTRL->BOOTLOADER_b.PROTLOCK (write 1 to clear) It is recommended to be done by Secondary Bootloader on exit, but may be optionally kept open if the entire customer application is trusted.

9.3.3 INFO Space Protections

Table 9-6: INFO Space Protections Field Description

Field Name	Description
OTP_SECURITY : DIS_CUST_INFO_PROG	This field allows the customer to selectively disable write access to any or all of the four quadrants of INFO0 space.

9.3.4 Update Configuration

Table 9-7: Update Configuration Field Description

Field Name	Description
OTP_SEC_POL : AUTH_ENF_ECC	This field forces authentication of customer updates even if there is an Embedded Content Certificate (ECC) included with the update.
OTP_SEC_POL : ENC_ENFORCE	This field determines whether decryption is required for all image updates.
OTP_SEC_POL : AUTH_ENFORCE	This field determines whether authentication is required for all image updates.

9.3.5 Wired Interface Configuration

Table 9-8: Wired Interface Configuration Field Description

Field Name	Description
OTP_BOOT_OVERRIDE : GPIO	This field sets the GPIO pin to allow an external device to override the secure boot flow and force processing of a wired update.
OTP_BOOT_OVERRIDE : POL	This field sets the polarity of the GPIO override pin as active high or low.
OTP_BOOT_OVERRIDE : ENABLE	This field enables the GPIO override feature. By default it is disabled.
OTP_WIRED_CONFIG : UART OTP_WIRED_CONFIG : SPI OTP_WIRED_CONFIG : I2C	These fields enable UART, SPI, or I^2C interfaces for wired update operation. The customer may enable one or more of these interfaces as needed, but typically will choose only one.
OTP_WIRED_CONFIG : SLAVEINTPIN	This field selects the GPIO to be used by an external device to interrupt the IOS if SPI or I^2C modes are enabled.
OTP_WIRED_CONFIG : I2CADDR	This field selects the I^2C address to be used by the IOS if I^2C mode is enabled.
OTP_WIRED_CONFIG : UARTMODULE	This field selects the UART instance/module (0-3) to be used for the wired interface if UART mode is enabled. Specific UART configurations are controlled through INFO0.

9.3.6 NVM Access Protection

The NVM space for the Apollo4 is 2MB. For each of the protection fields in OTP there are four (4) 32-bit words or 128 bits total assigned to protect regions/sections of NVM. This means that the customer may protect memory blocks of 16K each. Bit 0 is the first 16K block (addresses 0x00000000 – 0x00003FFF) and Bit 127 is the last 16K block (addresses 0x001FC000 to 0x001FFFFF).

Table 9-9: NVM Access Protection Field Description

Field Name	Description
OTP_CUST_WPROT OTP_CUST_RPROT	These fields are controlled by the customer during provisioning. They provide permanent write and read protection. WPROT provide write protection to the specific NVM 16K blocks (e.g., "Chunks"). RPROT provide "copy" or read protection. See *Section 14 Appendix on page 46* for more information. Note that protection for lower 96K of the NVM memory must never be enabled (e.g., do not set lower 6 bits (bit0-bit5) of OTP_CUST_WPROT0 or OTP_CUST_RPROT0) as it would interfere with the functioning of the pre-installed bootloader.
OTP_SBL_WPROT OTP_SBL_RPROT	Similar to CUST_*PROT. However, while protected from general access, write/access to these NVM blocks is possible through SBL via a secure update.

9.3.7 Symmetric Keybank Protections

Table 9-10: Symmetric Keybank Protection Field Description

Field Name	Description
OTP_CUSTOTP_PROGLOCK OTP_CUSTOTP_RDLOCK	These values determine the read/program access to the four quadrants of OEM/customer key banks in OTP. Quadrants #0 and #1 cannot be programmed after exiting the DM LCS until returning to the RMA LCS (and hence used for static keys), Quadrants #2 and #3 can be programmed from the main application or locked down. These can be optionally used to store run-time generated keys. ■ RDLOCK[3:0] locks down read access to the OEM/customer key banks when running boot operations (e.g., SBR, SBL, and Secondary Bootloader). ■ RDLOCK[7:4] locks down read access to the OEM/customer READ key banks after boot. ■ If not locked, the key banks could be accessed by an authorized software, as long as it has access to a 128-bit unlock key, as programmed in OTP_CUSTOTP_READ_KEY[1] ■ PROGLOCK[1:0] lock down access to programming OEM/customer key banks (Quadrant #2 and #3) when running the boot operations. ■ PROGLOCK[3:2] lock down access to programming OEM/customer key banks (Quadrant #2 and #3) after boot. ■ If not locked, these key banks can be programmed by an authorized software, as long as it has access to a 128-bit program key, as programmed in OTP_CUSTOTP_PROG_KEY[1]

[1] Implies multiple registers (OTP_CUSTOTP_PROG_KEY0 to OTP_CUSTOTP_PROG_KEY3).

Table 9-11: LCS Conditions and Results

LCS	Condition	Result
CM, DM, SE	(OTP_CUST_RDLOCK[0] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #0 Key Bank can be read during boot.
CM, DM, SE	(OTP_CUST_RDLOCK[4] == 0) AND (OTP_CUSTOTP_READ_KEY0 != 0)	Quadrant #0 Key Bank can be read after boot using the master read key.
DM, RMA	N/A	Quadrant #0 Key Bank can be programmed.

Table 9-11: LCS Conditions and Results *(Continued)*

LCS	Condition	Result
CM, DM, SE	(OTP_CUST_RDLOCK[1] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #1 Key Bank can be read during boot.
CM, DM, SE	(OTP_CUST_RDLOCK[5] == 0) AND (OTP_CUSTOTP_READ_KEY0 != 0)	Quadrant #1 Key Bank can be read after boot using the master read key.
DM, RMA	N/A	Quadrant #1 Key Bank can be programmed
CM, DM, SE	(OTP_CUST_RDLOCK[2] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #2 Key Bank can be read during boot.
CM, DM, SE	(OTP_CUST_RDLOCK[6] == 0) AND (OTP_CUSTOTP_READ_KEY0 != 0)	Quadrant #2 Key Bank can be read after boot using the master read key.
DM, RMA	(OTP_CUST_PROGLOCK[0] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #2 Key Bank can be programmed during boot.
DM, RMA	(OTP_CUST_PROGLOCK[2] == 0) AND (OTP_CUSTOTP_PROG_KEY !=0)	Quadrant #2 Key Bank can be programmed after boot using the master programming key.
CM, DM, SE	(OTP_CUST_RDLOCK[3] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #3 Key Bank can be read during boot.
CM, DM, SE	(OTP_CUST_RDLOCK[7] == 0) AND (OTP_CUSTOTP_READ_KEY0 != 0)	Quadrant #3 Key Bank can be read after boot using the master read key.
DM, RMA	(OTP_CUST_PROGLOCK[1] == 0) AND ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1))	Quadrant #3 Key Bank can be programmed during boot.
DM, RMA	(OTP_CUST_PROGLOCK[3] == 0) AND (OTP_CUSTOTP_PROG_KEY !=0)	Quadrant #3 Key Bank can be programmed after boot using the master programming key.

NOTE: The expression ((SBRLOCK == 1) OR (SBLLOCK == 1) OR (PROTLOCK == 1) implies that the code is running in the Bootloader and not yet reached the main application.

9.3.8 Debug Control

The Apollo4 SoC supports fine granularity debug control through a hardware Debug Control Unit (DCU). There are distinct debug features which are controlled by individual DCU bits. As can be seen in *Section 14 Appendix on page 46*, debug control defaults differ based on the LCS.

Hardware raw DCU bits are triple encoded: b'101 enables a feature, while b'010 disables it.

Table 9-12: Debug Control Field Description

Field Name	Description
OTP_LOCKMASK[0-3]	The DCU values can be locked and prevented from further modifications by configuring and specifying the triple bits corresponding to the feature to be locked.
OTP_DCU_DISABLEOVERRIDE	This field provides the ability to disable a feature upon SBL exit, even if the DCU settings allow for the feature. Setting a bit in the 21-bit field forces the desired debug feature to be disabled. Unlike the DCU locking, this override can be reversed by programming MCUCTRL->DEBUGGER register.

9.4 OEM Infospace (INFO0) Configuration

INFO0 provides MRAM based NVM storage which is separate from the main MRAM. Some of the fields in INFO0 are predefined for certain configurations and settings to further modulate the security settings in the OTP. It also contains fields which allow customers to override the defaults for certain attributes of the boot flow.

Rest of the INFO0 (undefined) is free for customers to use to store any other device information.

INFO0 is split into four quadrants, which can be individually write protected through OTP security settings.

9.4.1 Signature

Table 9-13: Signature Field Description

Field Name	Description
INFO0_SIGNATURE	This field uses four (4) 32-bit words to form a 128-bit signature. The signature is initialized to an invalid value during Ambiq's manufacturing. INFO0 provisioning will set the signature to indicate a valid customer INFO0 configuration. Unprogrammed INFO0 has a predefined fixed behavior for various settings.

9.4.2 Customer Trims

Table 9-14: Customer Trims Field Description

Field Name	Description
INFO0_CUSTOMER_TRIM	This field must be set to '0' due to an Errata (ERR079).

9.4.3 Wired UART Config

Table 9-15: Wired UART Config Field Description

Field Name	Description
INFO0_SECURITY_WIRED_IFC_CFG0	This word contains the UART module configuration. It must be consistent with the OTP_WIRED_CONFIG values set.
INFO0_SECURITY_WIRED_IFC_CFG1	This word contains the configuration for up to four GPIOs that may be used for UART RX, TX, CTS, RTS pins. Since the configurations below are "raw" there is no implied order to the pins.
INFO0_SECURITY_WIRED_IFC_CFG2 INFO0_SECURITY_WIRED_IFC_CFG3 INFO0_SECURITY_WIRED_IFC_CFG4 INFO0_SECURITY_WIRED_IFC_CFG5	These words contain the GPIO configuration to be used for PIN0, 1, 2, and 3 respectively.
INFO0_WIRED_TIMEOUT	Because the UART interface is asynchronous, the Secure Bootloader will wait for a configured amount of time before is exits the wired update process waiting for a connection from external host.

9.5 Security Version

Table 9-16: Security Version Field Description

Field Name	Description
INFO0_SECURITY_VERSION	This field contains the Version ID for INFO0 and is just for tracking purpose.

9.5.1 Memory Reservation

Table 9-17: Memory Reservation Field Description

Field Name	Description
INFO0_SECURITY_SRAM_RESV	This word determines the amount of CPU SRAM (DTCM) to keep reserved for application scratch space. This reserves the specified memory at the top end of CPU SRAM memory address range (ending at 0x1005FFFF). This memory is not disturbed by the Secure Boot Loader during any of the boot operations. The starting point of the reserved memory is 0x1006000 – SRAM_RESV.

9.5.2 Enable RMA Override

This field determine if Ambiq has permission to download an Ambiq RMA certificate after the customer has performed their part of the RMA process. Refer to *Section 12 Transition to RMA LCS on page 42* for RMA transition process.

9.5.3 Secure Debug Certificate Locations

Table 9-18: Secure Debug Certificate Locations Field Description

Field Name	Description
INFO0_SBR_SDCERT_ADDR	This word is the location of the of the Secure Debug Certificate in memory. The certificate may be located in user accessible MRAM, TCM or SRAM.

9.5.4 Main Application Location (Non-Secure Boot)

Table 9-19: Main Application Location (Non-Secure Boot) Field Description

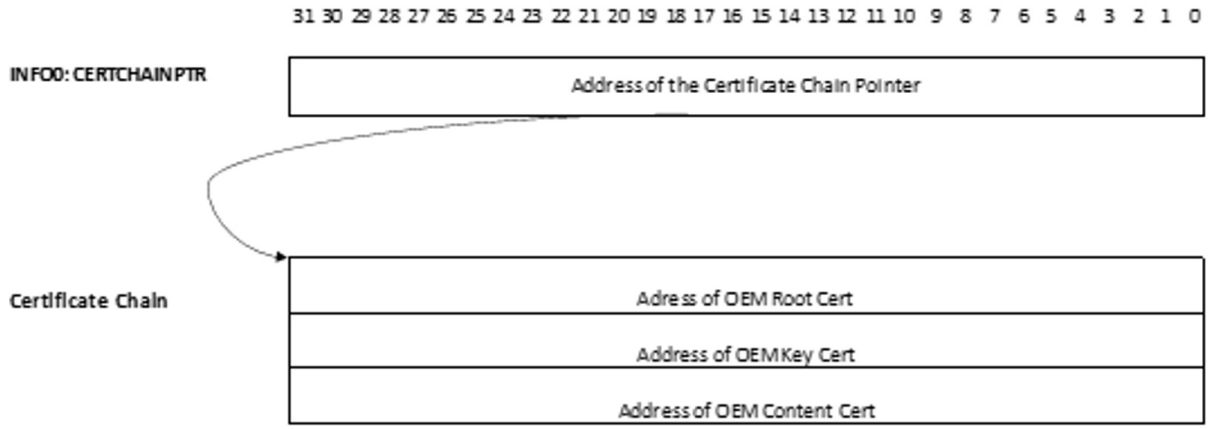
Field Name	Description
INFO0_MAINPTR	This word is the location of the main application image when non-secure boot flow is operational.

9.5.5 Certificate Chain Location (Secure Boot)

Table 9-20: Certificate Chain Location (Secure Boot) Field Description

Field Name	Description
INFO0_CERCHAINPTR	The Apollo4 SoC use a "chain" of certificates to authenticate images/content during the secure boot flow. This word is the address of the Certificate Chain pointers, which are 3 contiguous words in memory. These point to the OEM Root, OEM Key, and OEM Content certificates as shown in Figure 9-1.

Figure 9-1: OEM Certificate Chain Pointers



SECTION

10

Configuring Secure Boot Mode

The following steps outline the process of configuring the Apollo4 SoC into Secure Boot Mode:

1. Generate the Certificates using the tools provided.
2. Program the Certificate Chain Pointer in INFO0.
3. Program the NVM at the address in the INFO0_CERCHAINPTR with the three certificate addresses.
4. Program all three certificates in NVM at the designated locations in memory.
5. Program the images corresponding to the Content Certificate Image[] list at the designated locations in memory.
6. Configure OTP_SECURITY : CUST_SECBOOT to 0x2.
7. Configure OTP_SECURITY : CUST_SECBOOTONRST to 0x2 (enable) or 0x5 (disable)
8. Reset the SoC.

SECTION

11

Transition to Secure LCS

Secure LCS is a very restrictive security state for the Apollo4 SoC. The transition to this state will necessarily lock down resources and restrict many of the normal development pathways (e.g., the debugger). When in Secure LCS, default DCU settings are different. All the debug access is disabled by default, so a customer cannot talk to the device natively through debugger (Jlink etc). If a subset of debug functionality is desired the customer must install a Secure Debug Certificate with corresponding configuration prior to the transition to Secure LCS. Most of the faults and errors detected by SecureSPOT functionality in the Secure Boot flow will result in Apollo4 SoC being "locked up" by design.

Therefore, Ambiq's recommendation is that the customer always keep a path open to enable 'downloading' by either enabling wired update in the Secure Bootloader or by providing a custom implementation in their main application to update the images in the device. This would be the path to load a Secure Debug Certificate in order to open up debugging on a device after it has been fielded. In addition, selective debug functions could also be enabled in the main application also, using the DCU HAL, but this will only work if the corresponding DCU flags are not already locked through OTP configuration.

In general, it is expected that the customers will do one step provisioning on their production line to set up OTP – be it keys, configurations or the RoT itself. Provisioning of RoT in turn also triggers advancement to Secure LCS upon reset.

Subsections below describe the provisioning process in detail. Alternate means using OTP HAL are possible but requires full understanding of the OTP structure and details of individual fields to be programmed.

OEM provisioning tool consist of two parts:

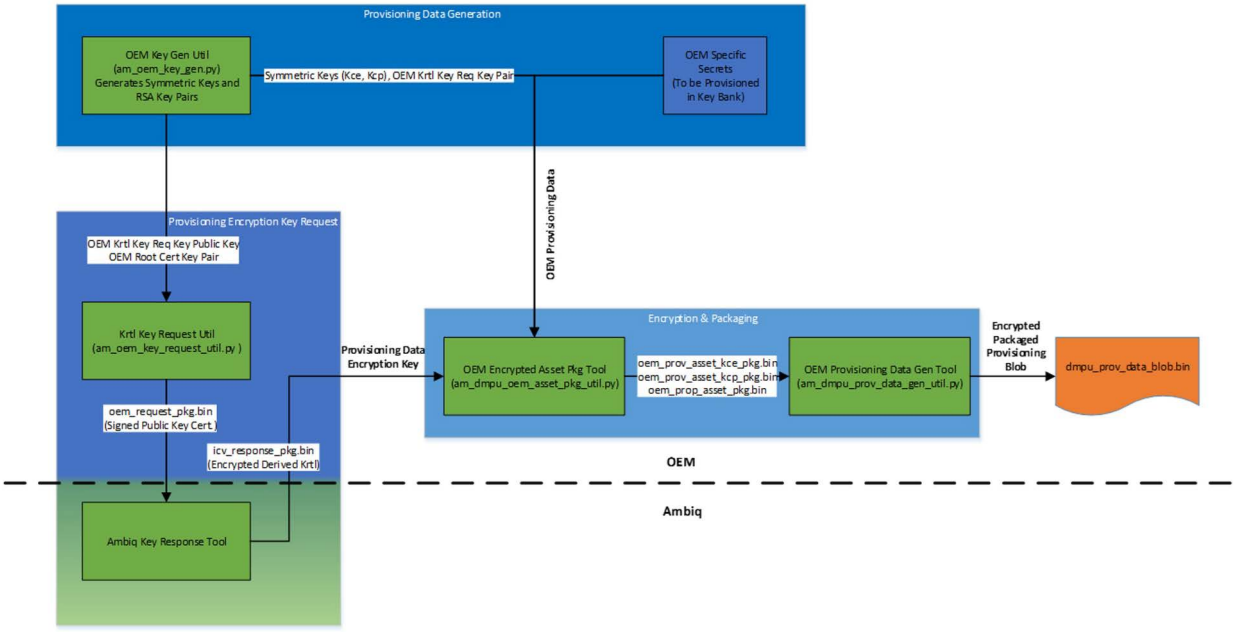
- Device Provisioning Tool (opt_image_pkg.bin)
  - Used to Provision OEM encrypted assets securely
  - Provided by Ambiq (Signed with Ambiq Private key)
- PC Side Tool Chain Example Scripts
  - Tools to generate OEM encrypted assets
  - Tools to package encrypted assets to single blob

11.1 Generating the Provisioning Blob

Figure 11-1 is a high-level view of the generation of the OEM Provisioning Blob while in DM LCS state. Ambiq provides the tools for each of these steps in the **/ tools/apollo4b_scripts/oem_tools_pkg** directory.

Refer to *Apollo4 Family Provisioning, Update, and Tools User's Guide A-SOCAP4-UGGA01EN* for details on how to use various tools provided with AmbiqSuite SDK to generate various provisioning assets.

Figure 11-1: OEM Provisioning Blob Generation



Use the following procedure to generate the provisioning blob:

1. Generate Asymmetric and Symmetric Keys
   This step could be replaced by OEM's own established key server
   am_oem_key_gen_util/am_oem_key_gen_util.py
2. Generating RoT (Needed to program in OTP)
   cert_utils/am_hbk_gen/am_hbk_gen_util.py
3. Make an Encryption Key Request to Ambiq.
   This essentially protects the provisioning information from prying eyes even in the case of untrusted ODM. The Response from Ambiq is required to encrypt the final provisioning asset blob.
   oem_asset_prov_utils/oem_key_request/am_oem_key_request_util.py
4. Key Asset Encryption which encrypts the OEM's sensitive asset data so that only they can access it.
   oem_asset_prov_utils/oem_asset_package/am_dmpu_oem_asset_pkg_util.py

5. Asset Generation which packages the OEM OTP configurations
   oem_asset_prov_utils/oem_asset_package/oem_asset_gen_util.py
6. Provisioning Data Generation which packages all the pieces together to generate final encrypted provisioning blob
   oem_asset_prov_utils/oem_asset_package/am_dmpu_prov_data_gen_util.py

The output of the last step is the final "Encrypted Provisioning Data Blob".

11.2 Provisioning the Device

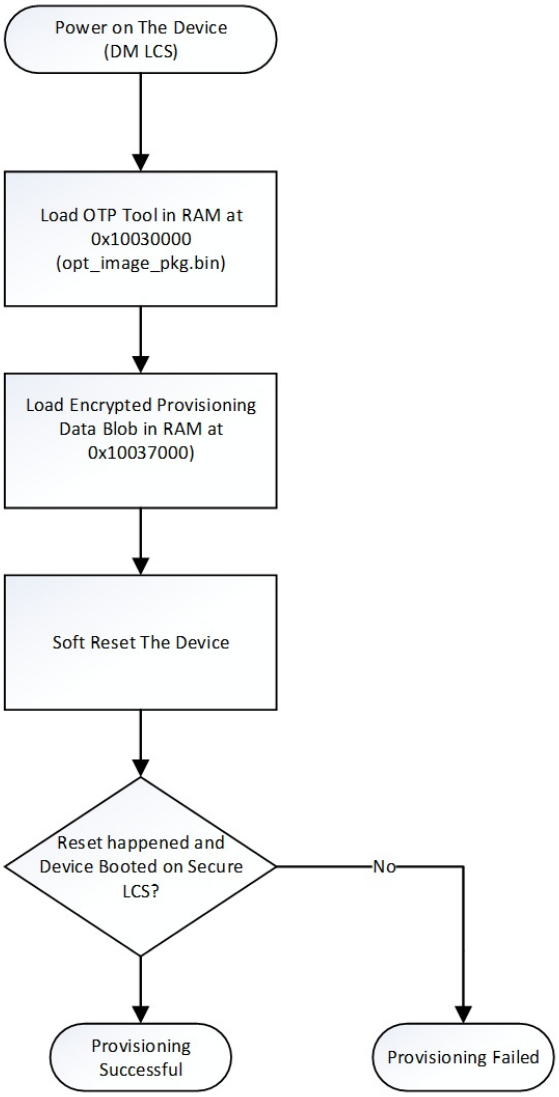
Ambiq provides an OEM Provisioning Tool (OPT) which is a binary that can be loaded on to the Apollo4 SoC during customer manufacturing. In addition to this binary, the Encrypted Provisioning Data Blob from the previous section must also be loaded into memory.

Use the following procedure to provision the device:

1. Load the OEM Provisioning Tool (OPT) @ 0x10030000
   oem_prov_tool/opt_image_pkg.bin
2. Load the OEM Provision Data Blob @ 0x10037000.
3. Perform a Soft Reset will accomplish the provisioning and transition the device to Secure LCS.

Figure 11-2 on page 41 shows the flowchart for OEM Provisioning Process.

Figure 11-2: OEM Provisioning Blob Generation



SECTION

12

Transition to RMA LCS

In some situations, it may be necessary to transition the Apollo4 SoC into the RMA LCS. The RMA LCS is a terminal State and is only possible to enter this state at the Ambiq Secure Labs which process the RMA. All Ambiq and OEM secrets are invalidated permanently during the transition to RMA LCS, including:

- KCE, KCP invalidated
- OTP Keybank area & access keys invalidated
- NVM Permanent Copy Protected sectors erased

In order to analyze issues with an RMA parts, the Ambiq/customer may be required to create special application images.

The chip cannot be returned to customer or recommissioned.

The transition to RMA LCS uses the same SD Cert infrastructure as previously described. RMA transition is only possible after the Apollo4 SoC processes specially created chip specific RMA Certificates (RMA Certs).

AmbiqSuite SDK provides reference scripts and configurations for RMA specific SD Certs, to assist customer generating OEM RMA Certs.

Successful RMA transition is a two-step process. It requires both a customer signed RMA Cert, as well as an Ambiq Signed RMA Cert for successful transition to RMA LCS.

After installation and processing of OEM RMA Cert, the customer has two options to enable Ambiq to complete the RMA transition:

- Enable SWD debug access to Ambiq (Either enabled my main image, or an SD Cert installed for the same after OEM RMA processing), to install the Ambiq RMA Cert
- INFO0->RMAOVERRIDE enabled, allows Ambiq use SBL wired download once the device has already gone through the OEM's RMA processing, to install Ambiq RMA Cert.

Use the following procedure to transition to RMA LCS are:

1. Install and process the OEM RMA Cert
2. Provide the Apollo4 SoC to Ambiq, enabling one of the options above to allow Ambiq install its RMA Cert independently
3. Provided following additional information along with the chip to Ambiq:
   - SOCID (if Secure LCS)
   - Info0 configurations:
     - SD Cert location
     - Main image (nonsecureboot), or Main CertChain Pointer (secureboot) location
   - Cert version counters – Content of registers at 0x400C2088 to 0x400C2098, from OTP

SECTION

13

Image Updates

The Apollo4 SecureSPOT supports image updates via customer applications (Firmware Over-the-Air) or via the wired update interface. Additionally, reference JLink scripts are provided with the SDK to assist updates in debug/production-line settings when debugger is enabled.

The following updates are supported:

- Firmware
  - Nonsecure
  - Secure
- INFO0
- Certificate Chain (Primarily used to update the version#, for revocation)
- Trim Patches
- Secure Bootloader Updates
- Key revocation (To revoke OTP Keybank keys)

Additionally, Wired Download through SBL wired update interface is also supported – which supports both raw downloads, as well as downloading an OTA image blob, and in turn initiating regular OTA (update) cycle.

Details of the Blob formats and processing are provided in the Apollo4 Soc Secure Update document.

Details on how to use the AmbiqSuite tools to generate and update the assets are provided in *Apollo4 Family Provisioning, Update, and Tools User's Guide A-SOCAP4-UGGA01EN*.

13.1 General Update (OTA) Process

In general, the authentication of the update image blob uses a Public Key based on the boot certificate chain) if required by the security policy. In addition, in-place decryption may also be required by the security policy.

Subsequent Processing depends on type of image:

- Firmware Updates will install the new firmware at designated location in NVM.
- Secure Firmware Updates will verify the certificate chain to ensure once installed it does not fail to boot, then install the images, and also install a new content certificate, if needed
- Info0 Updates will update INFO0 based on provided information
- Trim Updates will update Device trims by executing an embedded trim patching function
- Cert Chain Update will update the installed certificates used for secureboot verification.
- Key revocation update is used to update either the Ambiq or OEM keybank to revoke certain compromised symmetric keys

AmbiqSuite SDK provides reference JLink scripts for general OTA updates in **tools/apollo4b_scripts/jlink-ota.txt**

13.2 Secure Bootloader Image Updates

Installation of a replacement Secure Bootloader (SBL) requires special processing to avoid "locking up" the SoC. There are two slots (at addresses 0x00008000 & 0x00010000) which are reserved for Secure Bootloader images and their corresponding Certificate Chains. Only one image is in use at a time. The Secure Boot-ROM (SBR) determines the active chain based on a state flag, which is updated only after a successful upgrade. The inactive slot is available to use as a scratch space and also used as "Staging Area" for SBL upgrades.

The SBL Image Update consists of two blobs

- SBL Metadata (sbl_ota.bin) which can be stored anywhere and is discarded after the update
- Encrypted SBL Image (encrypted_sbl0.bin) which needs to be stored at the "Staging Area"

The SBL image update is initiated similar to other images, and consists of downloading the Encrypted SBL image to staging area, and then downloading the SBL metadata just like any other OTA blob and initiating the OTA.

AmbiqSuite SDK provides reference JLink scripts for SBL updates in **tools/apollo4b_scripts/jlink-prog-sbl[01].txt**:

- jlink-prog-sbl1.txt to be used if current SBL running from 0x8000
- jlink-prog-sbl0.txt to be used if current SBL running from 0x10000

SECTION

14

Appendix

14.1 OTP Configuration

OTP Region Start Address: 0x400C200.

Table 14-1: OTP Region Start Address: 0x400C2000

OTP Field	Purpose	OTP (word) Offset	MSB	LSB	Num Bits	Read Permissions	Write Permissions
RoT (HBK1)	128b Truncated hash of OEM Public Key	0x15	31	0	32	ALL	LCS=CM or LCS=DM
		0x16	31	0	32	ALL	LCS=CM or LCS=DM
		0x17	31	0	32	ALL	LCS=CM or LCS=DM
		0x18	31	0	32	ALL	LCS=CM or LCS=DM
KCP	128b OEM Symmetric Key used for Provisioning	0x19	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x1A	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x1B	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x1C	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
KCE	128b OEM Symmetric Key used for Encryption	0x1D	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x1E	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x1F	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
		0x20	31	0	32	LCS=CM or LCS=DM	LCS=DM or LCS=RMA (CM = 0's)
OEM-programmed flags (Key Attributes)	Number of "0" bits in HBK1	0x21	7	0	8	ALL	LCS=DM or LCS=RMA
	Number of "0" bits in KCP	0x21	14	8	7	LCS=CM or LCS=DM	LCS=DM or LCS=RMA
	KCP not in use	0x21	15	15	1	ALL	LCS=DM or LCS=RMA
	Number of "0" bits in KCE	0x21	22	16	7	LCS=CM or LCS=DM	LCS=DM or LCS=RMA
	KCE not in use	0x21	23	23	1	ALL	LCS=DM or LCS=RMA
	Reserved	0x21	29	24	6	ALL	ALL

Table 14-1: OTP Region Start Address: 0x400C2000

OTP Field	Purpose	OTP (word) Offset	MSB	LSB	Num Bits	Read Permissions	Write Permissions
HBK1_MINVER	NV counter for Certificate Version (0-95)	0x24	31	0	32	ALL	ALL
		0x25	31	0	32	ALL	ALL
		0x26	31	0	32	NONE	ALL
SECURITY	Secure Boot Enable	0x27	10	8	3	Bootloader	DM | RMA
	Secondary Bootloader Enable	0x27	11	11	1	Bootloader	DM | RMA
	Disable INFO0 Programming	0x27	15	12	4	Bootloader	DM | RMA
	Should not be used (Keep at 0)	0x27	16	16	1	Bootloader	DM | RMA
		0x27	18	18	1	Bootloader	DM | RMA
		0x27	19	19	1	Bootloader	DM | RMA
	Applicable only if Secureboot enabled	0x27	30	28	3	Bootloader	DM | RMA
	Enable Secureboot for Warm Reset Should not be used (Keep at 0)	0x27	31	31	1	Bootloader	DM | RMA
LOCKMASK0	Debug Control Lock Enable	0x2A	31	0	32	ALL	LCS=CM or LCS=DM
		0x2B	31	0	32	ALL	LCS=CM or LCS=DM
SBL_WPROT	Write Protect MRAM sectors (16K granularity) - Upgradable by Ambiq SBL	0x600	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x601	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x602	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x603	31	0	32	Bootloader	LCS=DM or LCS=RMA
SBL_RPROT	Copy Protect MRAM sectors (16K granularity) - Upgradable by Ambiq SBL	0x604	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x605	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x606	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x607	31	0	32	Bootloader	LCS=DM or LCS=RMA
WPROT	Permanent Write Protect MRAM sectors (16K granularity) - Pre-installed one time programmed content	0x608	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x609	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x60A	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x60B	31	0	32	Bootloader	LCS=DM or LCS=RMA
RPROT	Permanent Copy Protect MRAM sectors (16K granularity)	0x60C	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x60D	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x60E	31	0	32	Bootloader	LCS=DM or LCS=RMA
		0x60F	31	0	32	Bootloader	LCS=DM or LCS=RMA

Table 14-1: OTP Region Start Address: 0x400C2000

OTP Field	Purpose	OTP (word) Offset	MSB	LSB	Num Bits	Read Permissions	Write Permissions
SEC_POL	Enforce Authentication for Updates with CC	0x613	31	29	3	Bootloader	LCS=DM or LCS=RMA
	Enforce Encryption for Updates	0x613	28	26	3	Bootloader	LCS=DM or LCS=RMA
	Enforce Authentication for Updates	0x613	25	23	3	Bootloader	LCS=DM or LCS=RMA
	Wrap mode for Key Bank	0x613	22	19	4	Bootloader	LCS=DM or LCS=RMA
	Enable Bootloader Console	0x613	18	18	1	Bootloader	LCS=DM or LCS=RMA
	Reserved	0x613	17	0	18	Bootloader	LCS=DM or LCS=RMA
BOOT_OVERRIDE	SecureBoot Override Config	0x614	31	0	32	Bootloader	LCS=DM or LCS=RMA
WIRED_CONFIG	Wired Interface Config	0x615	31	0	32	Bootloader	LCS=DM or LCS=RMA
RSV		0x616-0x61F	31	0	32	Bootloader	LCS=DM or LCS=RMA
RSV	OEM general purpose space	0x620-0x67F	31	0	32	Bootloader	LCS=DM or LCS=RMA
CUSTOTP_READ_KEY	128b Secret Key used to unlock the Keybank for reading	0x680	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x681	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x682	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x683	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
CUSTOTP_PROG_KEY	128b Secret Key used to unlock the 2nd half of Keybank for Writing	0x684	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x685	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x686	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
		0x687	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
CUSTOTP_PROGLOCK	Controls Write Access to 2nd Half of Keybank	0x688	3	0	4	Ambiq SBR	LCS=DM or LCS=RMA
CUSTOTP_RDLOCK	Controls Read Access to Keybank	0x689	7	0	8	Ambiq SBR	LCS=DM or LCS=RMA
RSV		0x68A-0x6FF	31	0	32	Ambiq SBR	LCS=DM or LCS=RMA
KEYBANK_MFG	Keybank Q0	0x700-0x73F	31	0	32	((~CUSTOTP_RDLOCK[0] & (Bootloader)) | (~CUSTOTP_RDLOCK[4] & CUST_KEYBANK_KEY)) & LCS!=RMA	LCS=DM or LCS=RMA

Table 14-1: OTP Region Start Address: 0x400C2000

OTP Field	Purpose	OTP (word) Offset	MSB	LSB	Num Bits	Read Permissions	Write Permissions
KEYBANK_MFG	Keybank Q1	0x740-0x77F	31	0	32	((~CUSTOTP_RDLOCK[1] & (Bootloader)) | (~CUSTOTP_RDLOCK[5] & CUST_KEYBANK_KEY)) & LCS!=RMA	LCS=DM or LCS=RMA
KEYBANK_RT	Keybank Q2 (Can be programmed runtime)	0x780-0x7BF	31	0	32	((~CUSTOTP_RDLOCK[2] & (Bootloader)) | (~CUSTOTP_RDLOCK[6] & CUST_KEYBANK_KEY)) & LCS!=RMA	LCS=DM or LCS=RMA or ((~CUSTOTP_PROGLOCK[0] & (Bootloader)) | (~CUSTOTP_PROGLOCK[2] & CUSTOTP_PROG_KEY))
KEYBANK_RT	Keybank Q3 (Can be programmed runtime)	0x7C0-0x7FF	31	0	32	((~CUSTOTP_RDLOCK[3] & (Bootloader)) | (~CUSTOTP_RDLOCK[7] & CUST_KEYBANK_KEY)) & LCS!=RMA	LCS=DM or LCS=RMA or ((~CUSTOTP_PROGLOCK[1] & (Bootloader)) | (~CUSTOTP_PROGLOCK[3] & CUSTOTP_PROG_KEY))

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

A-SOCAP4-UGGA05EN v1.3
July 2023