NXP Semiconductors
Application Note

Document Number: AN12263
Rev. 0, 10/2018

HABv4 RVT Guidelines and Recommendations

Contents

1. Introduction

The High Assurance Boot (HABv4) support available in the i.MX 50, i.MX 53, i.MX 6, i.MX 7, i.MX RT and Vybrid families and i.MX 8M, i.MX 8MM devices provides an option to extend the root of trust beyond the initial primary boot image. An Application Programming Interface (API) is provided by the on-chip ROM that allows the use of the HAB library to extend the root of trust and authenticate additional software images. This document describes what a system designer should consider when planning to make use of this security feature.

1.1. Audience

This document provide details on:

- The HAB API included in the ROM Vector Table (RVT)
- How to develop solutions to extend the root of trust beyond the initial primary boot image

It is assumed that the reader is familiar with the High Assurance Boot concepts explained in Secure Boot on i.MX 50, i.MX 53, i.MX 6 and i.MX 7 Series using HABv4 (AN4581) document and also with the High Assurance Boot Version 4 Application Programming Interface Reference Manual included in the i.MX Code Signing Tool (CST) package.

1.2. Definitions, Acronyms, and Abbreviations

Table 1 describes the definitions of terms and acronyms used in this document.

Table 1. Definition of terms and acronyms

Terms or acronyms	Definitions	Remarks
API	Applications Programming Interface	Set of methods used to communicate with other software environment.
CAAM	Cryptographic Acceleration and Assurance Module	Accelerator for encryption, stream cipher, and hashing algorithms, with a random number generator and runtime integrity checker.
CSF	Command Sequence File	Binary data structure interpreted by the HAB to guide authentication operations.
CST	Code Signing Tool	Application running on a build host to generate a CSF and associated digital signatures.
DCD	Device Configuration Data	Binary table used by the ROM code to configure the device at early boot stage.
HAB	High Assurance Boot	Software library executed in internal ROM on the NXP processor at boot time which, among other things, authenticates software in external memory by verifying digital signatures in accordance with a CSF. This document is strictly limited to processors running HABv4.
OCRAM	On-Chip Random-Access Memory	Internal On-Chip RAM available in all i.MX devices.
RVT	ROM Vector Table	Vector table located at the ROM memory map which contains the pointers to the HAB API functions.
SNVS	Secure Non-Volatile Storage	Security feature that includes a security state machine and security violation detection circuits that, together with High Assurance Boot software, determine whether the chip is currently in a secure state.

2. HAB Programming Interface

The HAB Application programming Interface (API) is exposed through the ROM Vector Table (RVT). The RVT table location is device dependent and can be found in the Reference Manual specific to the target device in the “Internal ROM/RAM memory map” section.

Table 2. HAB API RVT address

Device	Silicon Revision	HAB API RVT Address
i.MX 8MM [2]	All	0x00000900
i.MX 8MQ	All	0x00000880
i.MX 6DualPlus/6QuadPlus	All	0x00000098
i.MX 6Dual/6Quad	< v1.3	0x00000094
	>= v1.3 [1]	0x00000098
i.MX 6DualLite/6Solo	< v1.2	0x00000094
	>= v1.2	0x00000098
i.MX 6SoloLite	All	0x00000094
i.MX 6UltraLite	All	0x00000100
i.MX 6SoloX	All	
i.MX 6SLL	All	
i.MX 6ULL	All	
i.MX 6ULZ [2]	All	
i.MX 7Solo/7Dual	All	
i.MX 7ULP – Cortex A7 [2]	All	
i.MX 7ULP – Cortex M4 [2]	All	0x1C000500
i.MX RT1060	All	0x00200300
i.MX RT1050	All	
i.MX RT1020	All	0x002002C0
i.MX RT1015 [2]	All	
Vybrid F and R	All	0x00000054
i.MX 50 Family [3]	All	0x00000094
i.MX 53 Family [3]	All	0x00000094
i.MX 28 Family	All	0xFFFF8AF8

1. In i.MX6D/Q Rev v1.3 the USB_ANALOG_DIGPROG is 0x00630005, Software must handle it as described in the SoC Reference Manual.
2. Pre-production devices.
3. i.MX 50 and i.MX 53 family devices are no longer recommended for new designs.

2.1. Authentication functions

The HAB API functions available in the HAB RVT are called from the runtime software to perform authentication of additional boot images. The recommended HAB API functions for authenticating images are described in Table 3.

Table 3. HAB API authentication functions offset

HAB API Function	Offset from HAB API RVT Address
hab_rvt.entry	0x04
hab_rvt.authenticate_image [1]	0x10
hab_rvt.authenticate_image_no_dcd [2]	0x2C
hab_rvt.exit	0x08

1. See item 4 in section 5 Security Considerations.
2. Only available in HAB version 4.2.0 and newer.

2.2. Status verification functions

The HAB API also provides the status functions to verify the processor security configuration and status. If a failure is detected the HAB events are obtained by using the report event function.

Table 4. HAB API status functions offset

HAB API Function	Offset from HAB API RVT Address
hab_rvt.report_event	0x20
hab_rvt.report_status	0x24

2.3. Other functions

Additional functions that are available to be utilized by post-ROM boot stage components are shown in Table 5.

Table 5. HAB API additional functions offset

HAB API Function	Offset from HAB API RVT Address
hab_rvt.check_target[1]	0x0C
hab_rvt.run_dcd[2]	0x14
hab_rvt.run_csf	0x18
hab_rvt.assert	0x1C
hab_rvt.failsafe	0x28
hab_rvt.get_version	0x30

1. Only available in HAB Version 4.2.5 and newer.
2. Please refer to item 4 in section 5 Security Considerations.

For more details on each API and its functionality please refer to the *"High Assurance Boot Version 4 Application Programming Interface Reference Manual"* included in the CST package.

3. HAB Persistent Data

During the boot process, the ROM and HAB make use of the internal On-Chip RAM (OCRAM). If a design intends to make use of HAB after the execution transitions to the user image, it is important to preserve the persistent data region initially setup by HAB. It contains data that HAB expects to be persistent, for example events, public keys and HAB process related information.

Table 6. HAB persistent memory region addresses and sizes

Device	Silicon Revision	Base Address	Size
i.MX 8M	v2.0	0x009061C0	0x0B80
	v2.1		0x1200
i.MX 8MM	v1.0	0x00908040	0x0B80
i.MX 7ULP – Cortex A7	v2.0	0x2F006840	
i.MX 7ULP – Cortex M4	v2.0	0x20008040	
i.MX 7Solo/Dual	All	0x009049C0	
i.MX 6 Family	All	0x00904000	
i.MX RT Family	All	0x20201000	
Vybrid F and R	All	0x3F077000	
i.MX 50 Family	All	0xF8004000	0x900
i.MX 53 Family	All	0x1FFE4000	
i.MX 28 Family	All	0x0001A800	0x600

4. Caching Considerations

The caches are used during the image authentication process and their management is necessary when working with HAB. Depending on the user specific application the following procedures should be considered.

When extending the root of trust using the `hab_rvt.authenticate_image()` API function the data caches are used, if enabled, leading to a faster image authentication. In case the end user application and system design does not require caches, they can be enabled before calling the HAB API and disabled afterwards.

NOTE

The caches must be properly flushed before disabling the Memory Management Unit (MMU) on the respective processor.

In the devices listed in Table 7 it is necessary to inform ROM code that caches are enabled prior to calling the `hab_rvt.authenticate_image()` API function. Otherwise, the ROM will not flush the caches and the authentication may fail. Writing '0x1' to the PU_IROM_MMU_EN_VAR register prior to calling `hab_rvt.authenticate_image()` will notify ROM of the cache state.

The current NXP U-Boot implementation contains an example in the `authenticate_image()` function which checks if the MMU is enabled prior to writing in the PU_IROM_MMU_EN_VAR register.

Table 7. PU_IROM_MMU_EN_VAR addresses

Device	PU_IROM_MMU_EN_VAR address
i.MX 6Dual/6Quad	0x009024A8
i.MX 6Solo/6DualLite	0x00901DD0
i.MX 6SoloLite	0x00900A18

5. Security Considerations

The items below list all the recommended security considerations system designers should implement when developing their own solutions to extend the root of trust.

1. NXP recommends using the `hab_rvt.authenticate_image()` API function whenever possible. This ensures all the proper authentication steps are performed. For HAB versions 4.2.0 and newer it is highly recommended to use `authenticate_image_no_dcd()` API function instead.

2. Boot sequences may consist of several images with each launching the next image as well as alternative images, should one boot device or boot image be unavailable or unusable. The authentication of each image in a boot sequence must be enclosed with its own `hab_rvt.entry()` ... `hab_rvt.exit()` API function pairs in order to ensure that security state information gathered for one image cannot be misapplied to another image.

3. Steps must be taken to remove support in customer application software if the end product is not using HAB API at all. For instance, U-Boot users must ensure CONFIG_SECURE_BOOT is not being selected in their build environment.

4. For devices prior to HAB 4.3.7, it is highly recommended to ensure the DCD pointer is Null prior to calling HAB `authenticate_image()` or `run_dcd()` API functions. DCD should not be present in images that will be verified by software using HAB RVT authentication APIs. Older versions of HAB will run DCD commands if available, which could lead to an incorrect authentication boot flow. For more details, please contact your local NXP representative.

5. Write, Check, and Set MID commands are deprecated from the Code Signing Tool (CST) version 2.3.3 and beyond and are also not implemented in the newer versions of HAB. The inappropriate use of Write Data command may lead to an incorrect authentication boot flow. It is recommended to ensure the image does not contain WRITE DATA commands prior to calling the `authenticate_image()` API function. For more details, please contact your local NXP representative.

6. It is highly recommended to call the HAB API functions from a software running in the Arm TrustZone Secure World mode. In this case the ROM code must setup certain CAAM and SNVS registers which can be only programmed in a Secure World context.

7. When deploying an encrypted boot environment, the PRIBLOB setting should be advanced in the CAAM Security Configuration Register (SCFGR). The DEK blob must be initially generated with default setting PRIBLOB=01 and the runtime software should set PRIBLOB=11. This ensures cryptographic separation of private blob types avoiding any modification or replacement of DEK blobs. Please refer to the encrypted boot application note (AN12056) for more details.

8. In the i.MX 7ULP, i.MX 7D, i.MX 6SoloX and i.MX 6UL devices the CAAM registers may be reset in suspend and resume operation. If deploying an encrypted boot environment using these devices, the PRIBLOB settings should be restored to ensure that the cryptographic separation of private blob types is still valid. Please refer to the encrypted boot application note (AN12056) for more details.

6. Known Limitations

1. Starting from HAB Version 4.3.7, the Run DCD function and the Authenticate Image function called with a non-null DCD pointer, will return an error if called outside the boot ROM.

2. The HAB Version 4.2.7 does not support `get_version()` and `authenticate_image_no_dcd()` API functions.

7. Revision history

Table 8. Revision history

Revision number	Date	Substantive changes
0	10/2018	Initial release

8. References

- i.MX 6, 7, 8M and 8MM families Reference Manuals and Security Reference Manuals
- Secure Boot on i.MX 50, i.MX 53, i.MX 6 and i.MX 7 Series using HABv4 Application Note available on nxp.com (AN4581)
- Encrypted Boot on HABv4 and CAAM Enabled Devices Application Note (AN12056)
- High Assurance Boot Version 4 Application Programming Interface Reference Manual available in the Code Signing Tool package

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 that may be provided in NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number: AN12263
Rev. 0,
10/2018