Achieving End-to-end Security Starting With Secure Boot

Donnie Garcia

Solutions Architect for Secure Transactions

May 2018 | AMF-ENT-T3145

SECURE CONNECTIONS FOR A SMARTER WORLD

NXP Solutions Realize the Power of 8.4B Connected Things

8.4B Connected Things

Machine Learning

Edge Compute

Connectivity & Security

Solving Complex Integration

Reducing Development Time

Providing Embedded Expertise

2000 products

200+ Product Families

15+ Product Lines

1 NXP Portfolio

NXP SOLUTIONS

Expertise Applied Within IoT Verticals

- Industry 4.0
- Smart Retail
- Home Control & Security
- Wireless Power

Agenda

- Essential Security Goals
- Secure Boot Architecture
- i.MX RT Secure Boot Technology
  - Hardware
  - Firmware
  - Tools and Infrastructure
- Key Management Table
- Lifecycle View
- How Secure Boot Enables Secure Transactions
- Conclusion

Start With a Security Model

Policies

The rules in place that identify the data that should be protected

For example

The management of firmware, secret keys, user and application data
Passwords, personal information, network credentials

Threat landscape

The definition of the attacks and attackers that the end device will face and protect against. Considers the access to the device, and cost of the attack

For example

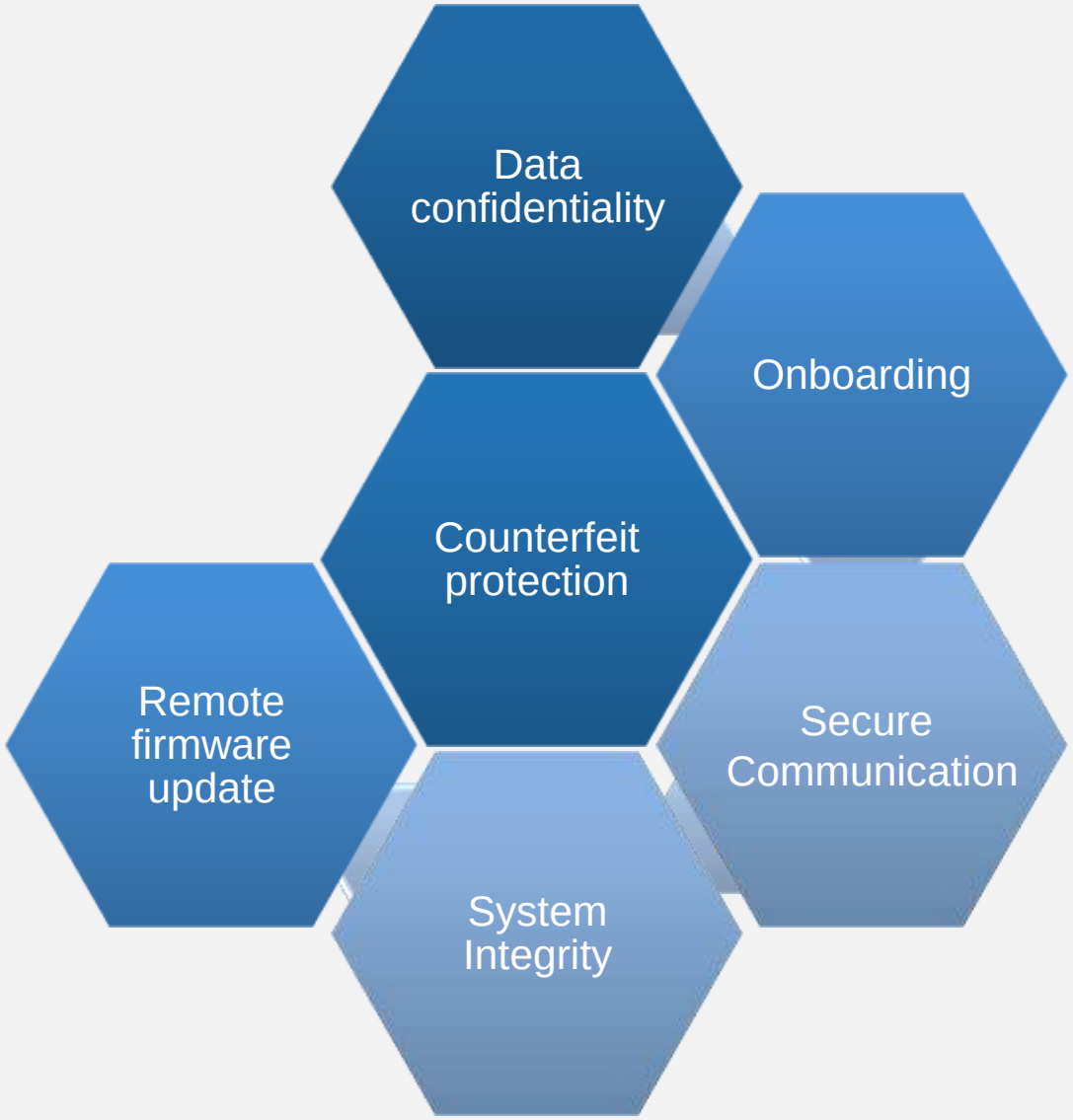
Expert attackers who will use off the shelf tools to gain access and insert malware

Methods

The means by which the policies for the device are enforced. Involves the application of security technology to achieve product goals

For example

Disabling debug access to restrict the availability of secret data on a processor



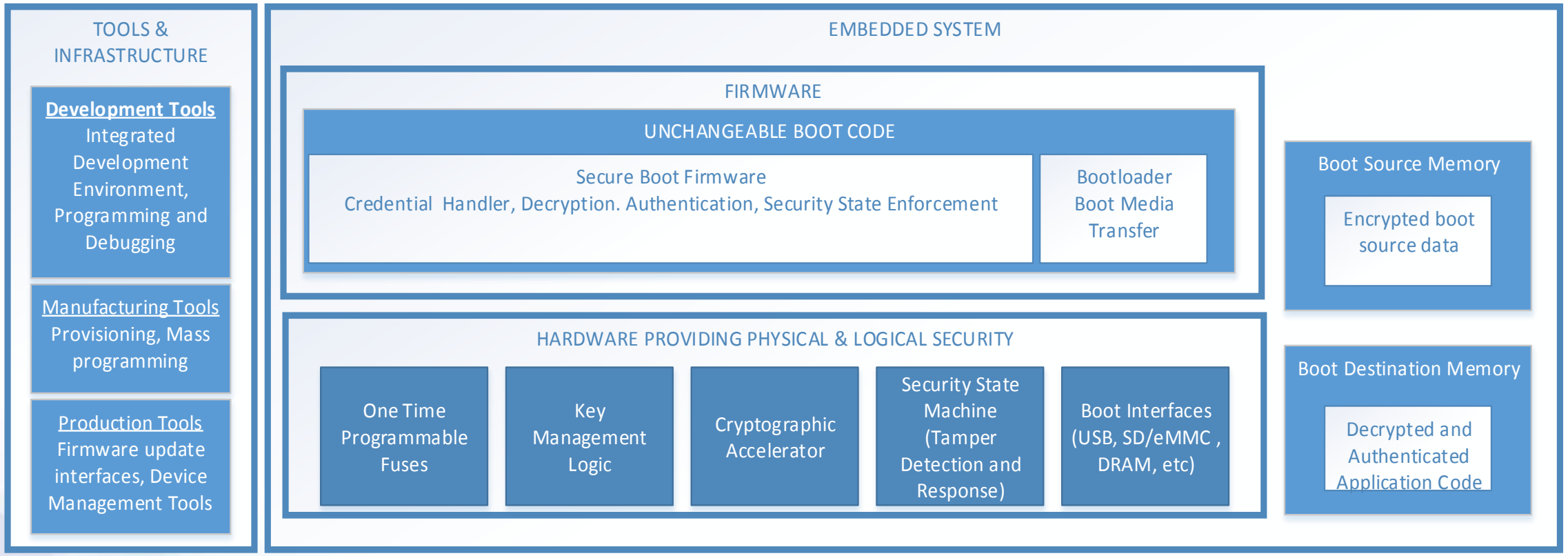
Essential Security Goals

- Counterfeit protection
  - Every device has a unique identity that can not be reproduced by an attacker
- Onboarding
  - Shared credentials between the end device and the back end system
- System integrity
  - Trust in the functionality provided by the end device
- Secure communication
  - Cryptography applied to the communications for the device
- Data confidentiality
  - Protection of data in the device
- Remote firmware update
  - Safe updating of the software on the end device

Secure Boot Architecture

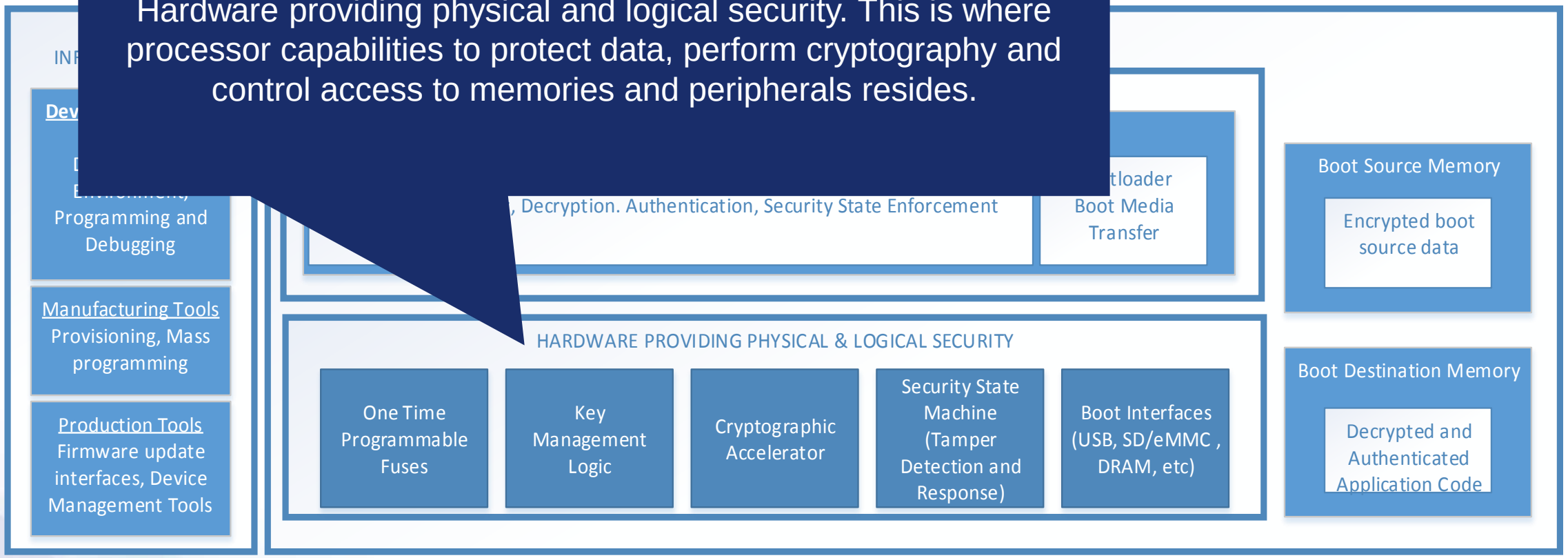
Components of a Secure Boot

Secure Boot Architecture Diagram

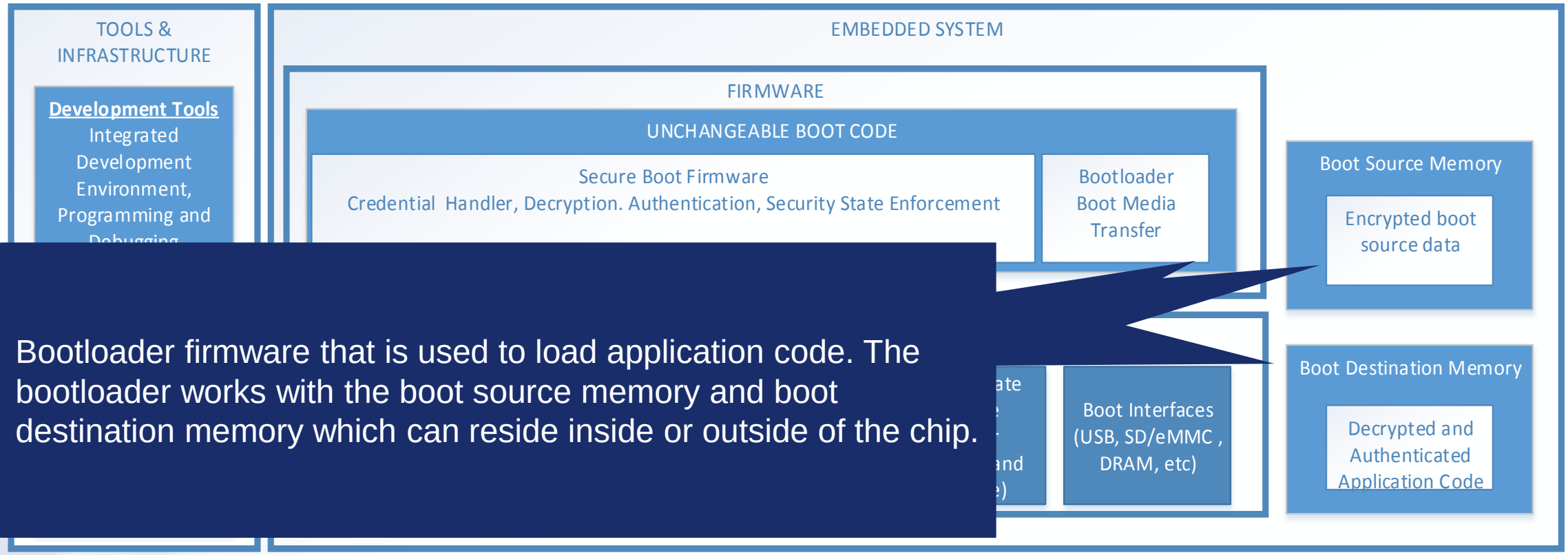


Se

Hardware providing physical and logical security. This is where processor capabilities to protect data, perform cryptography and control access to memories and peripherals resides.

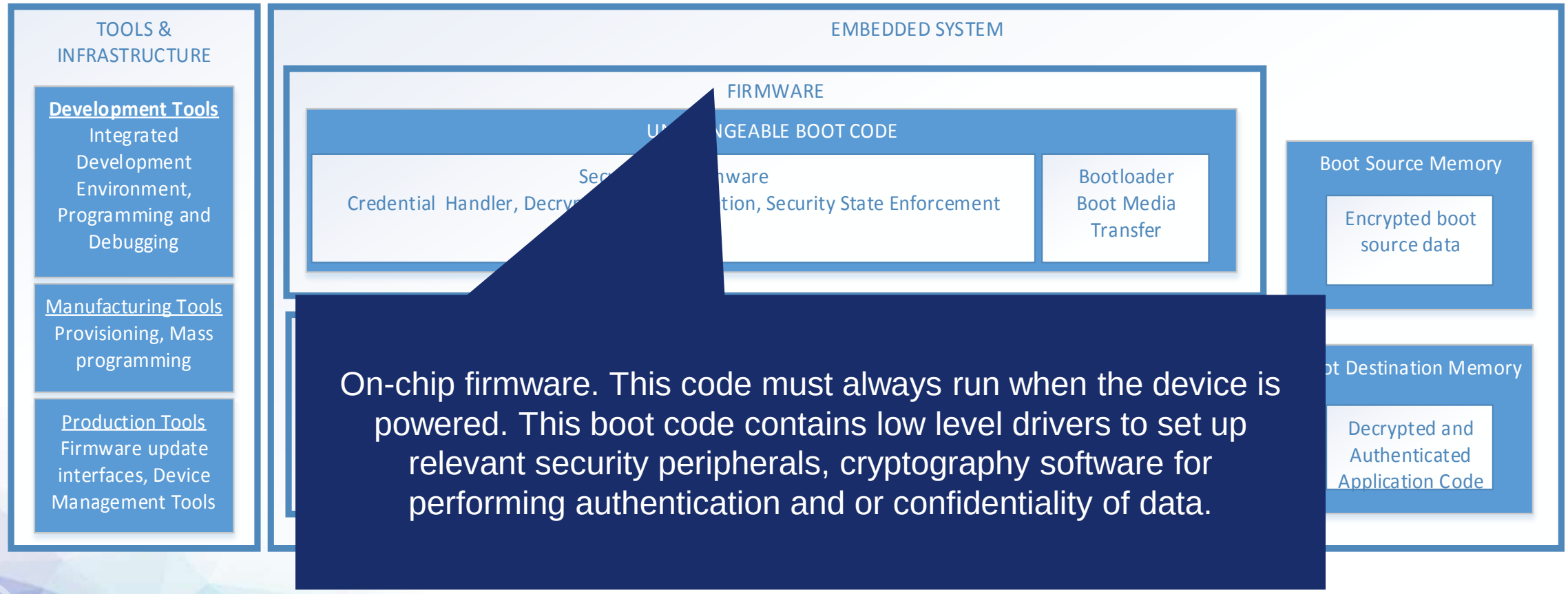


Secure Boot Architecture Diagram

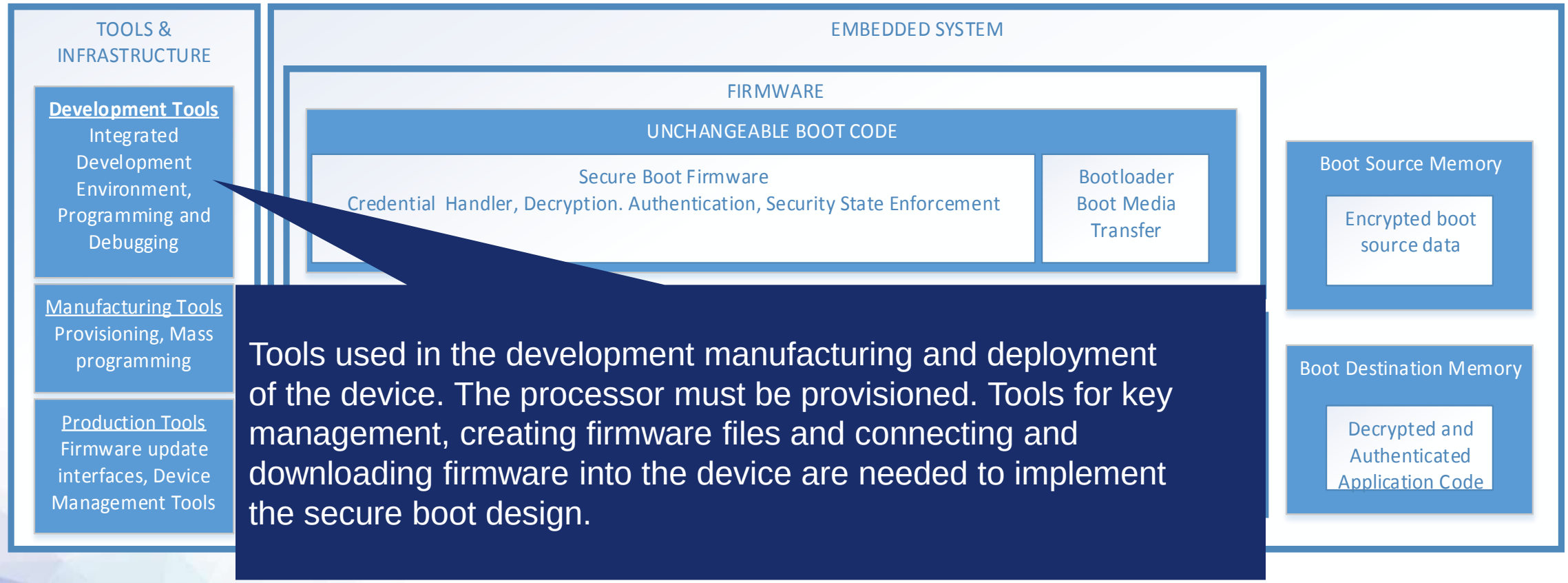


Bootloader firmware that is used to load application code. The bootloader works with the boot source memory and boot destination memory which can reside inside or outside of the chip.

Secure Boot Architecture Diagram



Secure Boot Architecture Diagram



i.MX RT Security Technology

Achieving a Secure Boot With the Crossover Processor

About i.MX RT

High Performance
Real-Time Processing

- Cortex-M7 up to 600MHz (50% faster than current existing M7 products)
- 20ns interrupt latency
- Up to 512KB Tightly Couple Memory

High level of Integration

- High Security enabled by AES-128, HAB and On-the-fly QSPI Flash Decryption
- 2D graphics acceleration engine
- Parallel camera sensor interface
- LCD display controller up to WXGA (1366x768)
- Audio interface with three I2S for multichannel high performance audio

Low BOM Cost

- Competitive pricing starting @ $2.98 10k RSL
- Fully integrated PMIC with DC-DC
- Low cost package, 10x10 BGA, enabling 4 Layer PCB design
- SDRAM interface

Easy to Use

- MCU customers can leveraging their current toolchain (MCUXpresso, IAR, Keil)
- Rapid and easy prototyping and development with NXP FreeRTOS, SDK, ARM mbed and the global ARM ecosystem
- Single voltage input simplifies power circuit design
- Scalability to Kinetis & i.MX products

i.MX RT1050 Security Architecture

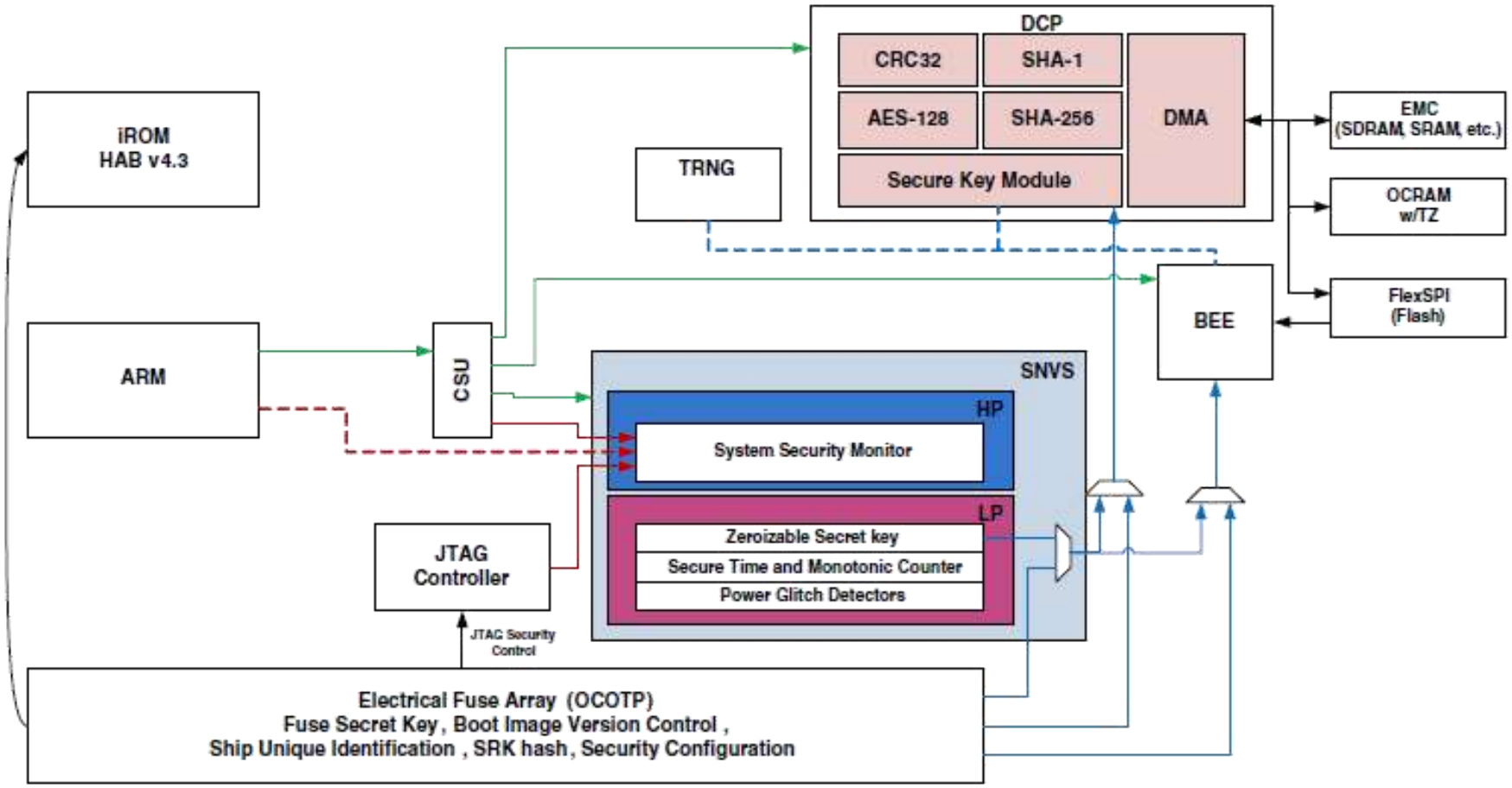
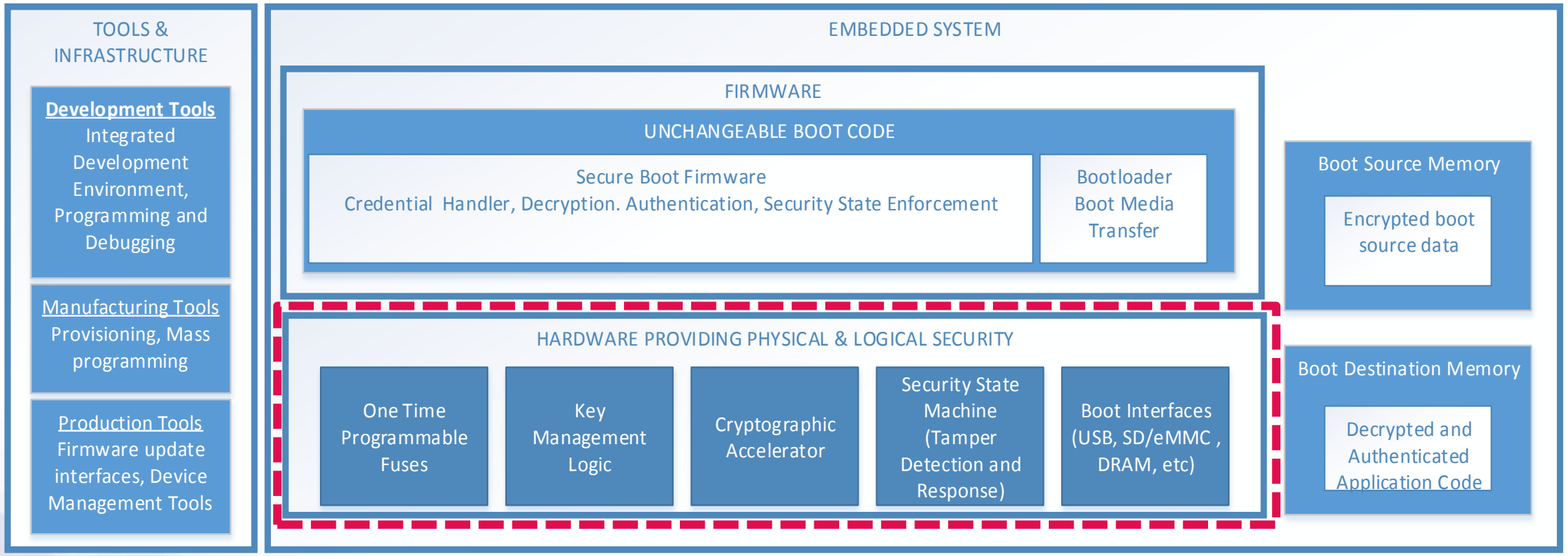


Figure 1-1. Security subsystem (simplified)

i.MX RT Security Technology

Hardware

Secure Boot Architecture Diagram

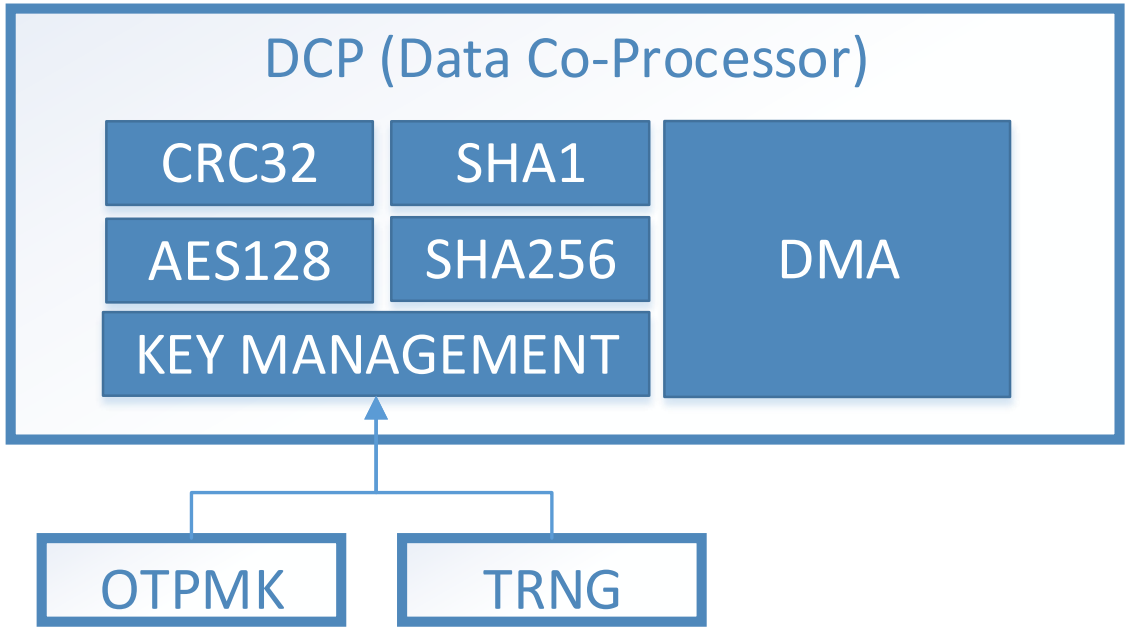


OTP Functions for Secure Boot

Function	Comment
Security Configuration	Set the chip level security setting to manage the lifecycle of the device. There are two settings, Open or Closed. A secure end design uses the Closed setting.
Field Return Configuration	Set the chip into a Field Return state to allow access to test functionality. This option can be disabled to restrict all access.
JTAG Security Mode	Controls the security mode of the JTAG debug Interface. The JTAG can be completely disabled.
Boot Configuration	Set the options for type of boot interfaces, speeds of I/O during boot, if there is a recovery boot image and boot timing.
Super Root Key Hash (SRKH)	A hash of the set of public keys that is used to check the integrity of the public keys that are part of the boot image. The SRKH ensures that the public key used to authenticate the boot image has not been modified from the expected values.
Super Root Key Revoke	Fuse settings to apply controls to the boot image. Used to achieve roll back protections.

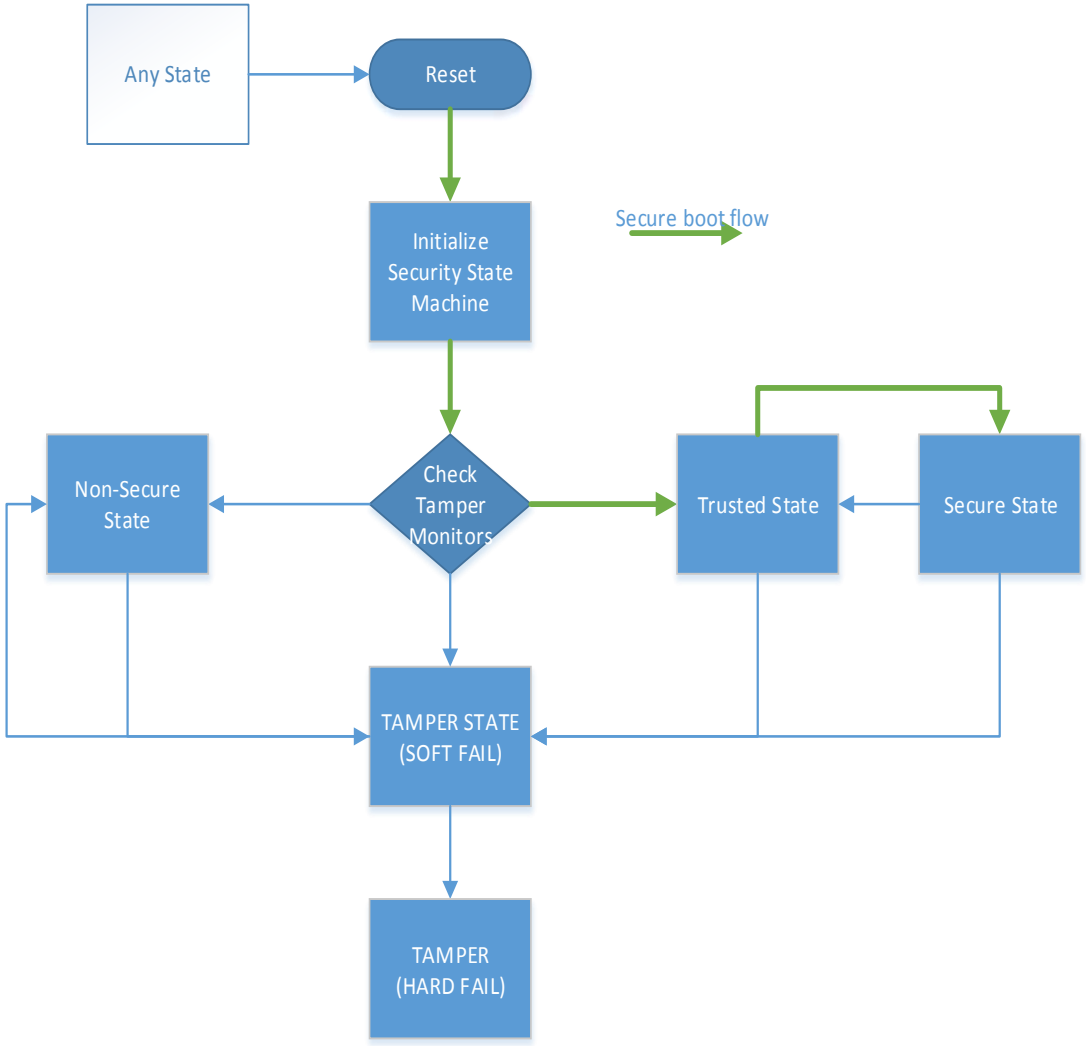
Cryptographic Accelerator and Key Management Logic

- DCP can accelerate the SHA-256 HASH function and AES128
- Key Management Logic allows the DCP to have access to a protected key which is provisioned during the i.MX RT manufacturing process
- This key is called the One-Time Programmable Master Key (OTPMK)

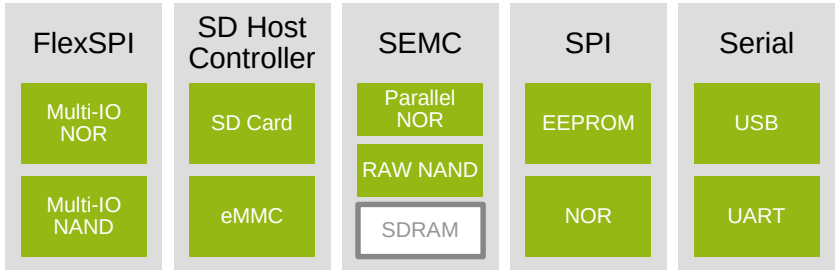


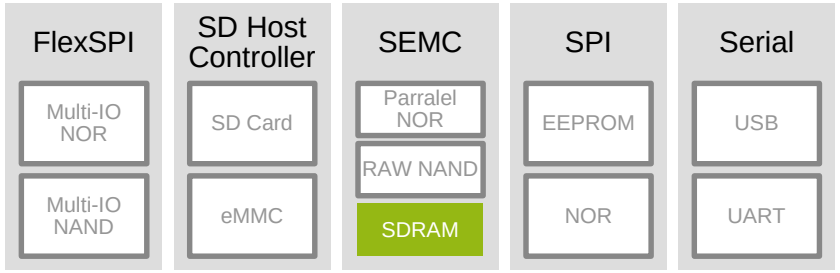
Security State Machine

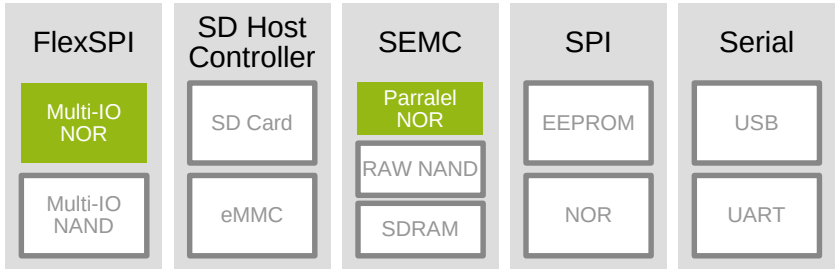
- Monitoring the process of booting and enforcing security protections is a block in the i.MX RT named the Secure Non-Volatile Storage (SNVS).
- Security state machine separated into an independent power domain on the chip.
- Power domain isolation allows tamper monitoring to be extended into a device state where a backup battery, such as a coin cell, is used for protection.
- SNVS serves as the SOC's central reporting point for security-relevant events such as the success or failure of boot software validation and the detection of security threat events.



Boot Interfaces and Memory Types





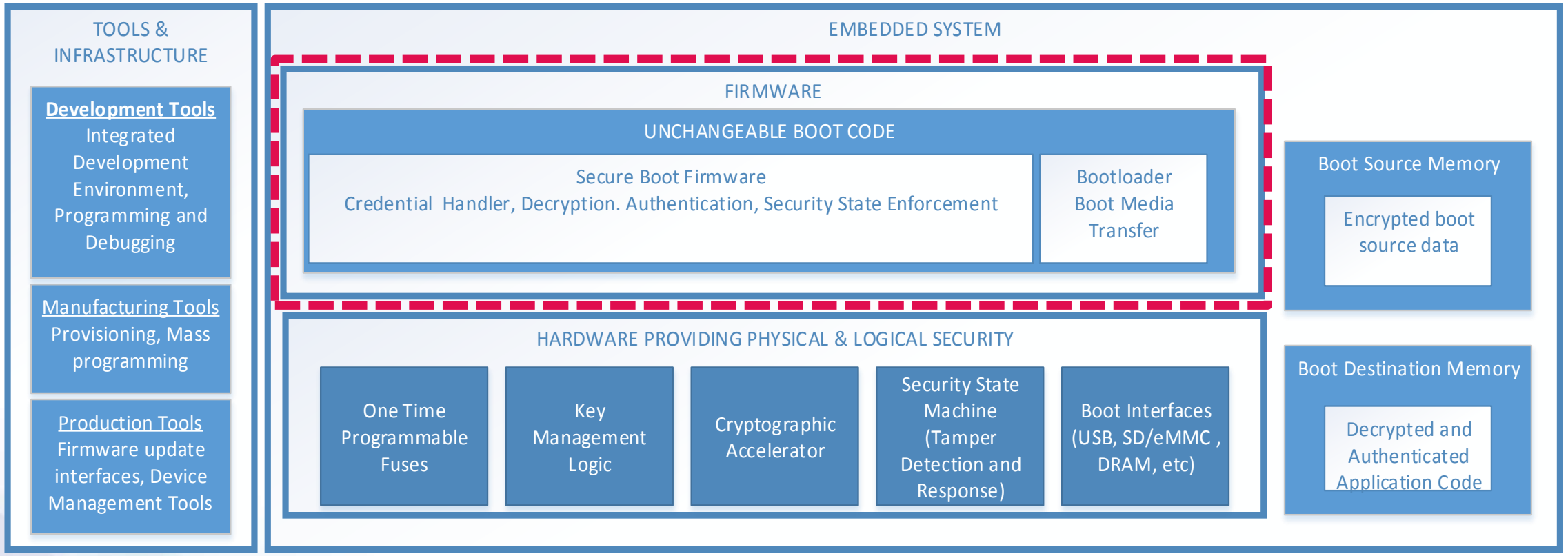


- Boot source
  - FlexSPI NOR
  - FlexSPI NAND
  - SD/eMMC
  - Parrallel NOR/Raw NAND
  - Serial (USB/UART)
  - SPI NOR/EEPROM
- Boot destination
  - System RAM (ITCM/OCRAM)
  - SDRAM
- Boot directly to
  - FlexSPI NOR
  - SEMC Parallel NOR

i.MX RT Security Technology

Firmware

Secure Boot Architecture Diagram



Boot Data Components

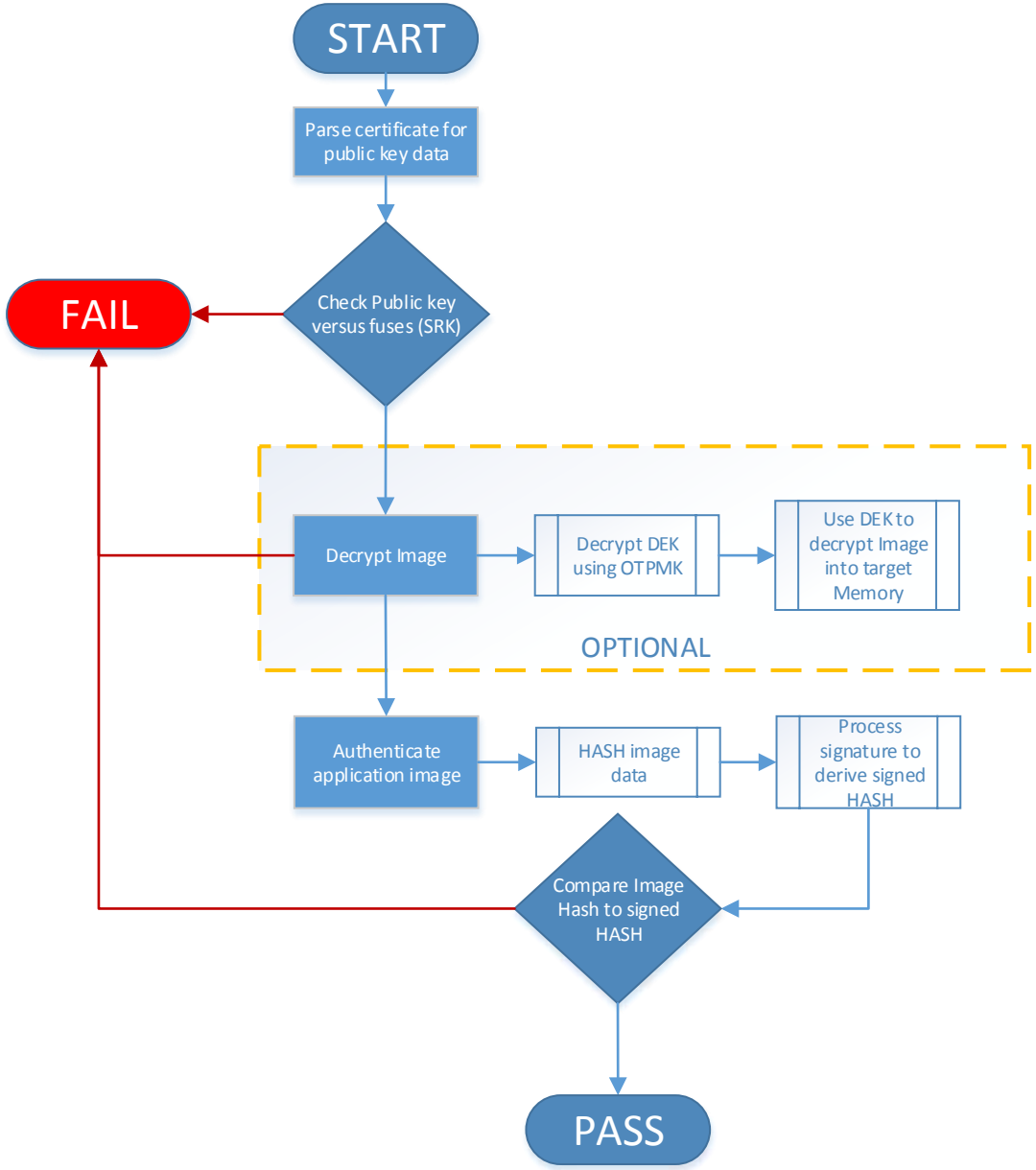
Component	Description	Use Scenario / Example
Image Vector Table (IVT)	List of addresses which details the location of other boot data components.	Every boot requires IVT. IVT contains the start address of the device configuration (DCD) block.
Boot Data	A structure that details where to load the image and the image size.	Every boot requires Boot Data.
Device Configuration Data (DCD)	Data that can be used to initialize interfaces for customization to specific hardware in the embedded system.	Hardware dependent if a DCD is needed. For example, when booting from FlexSPI only, DCD is not needed. When booting to SDRAM, the DCD is used to configure the SEMC memory controller to work with the specific memory device.
Image	This is the application image which is being loaded.	Every boot requires Image Data.
HAB Data	This is a group of components required to perform a secure boot. It includes a command sequence file, a certificate and a signature.	Only secure boot flows require the HAB data.
Data Encryption Key Blob (DEK Blob)	This is an encrypted key for the case of handling encrypted images.	Only secure boot flows which enable authenticated and encrypted boot require the DEK Blob.

High Assurance Boot (HAB)

- Operating on the data handled by the bootloader is the HAB (High Assurance Boot). This secure code library has its roots in processors dating back to the very first versions of i.MX. Over time, this firmware has been maintained and updated to address the latest in cryptographic algorithms, rollback protection and security vulnerabilities. The version of HAB for i.MX RT is 4.3.
- After controls enforced by hardware, and in conjunction with the handling of the boot data by the bootloader, control is passed over to the HAB component to perform the cryptographic operations.
- The HAB follows the commands provided by the command sequence file. The below diagram describes the runtime operations performed by the HAB component. There are options for supporting an authenticated boot, or an encrypted and authenticated boot as shown in the figure below.

HAB Runtime Operation

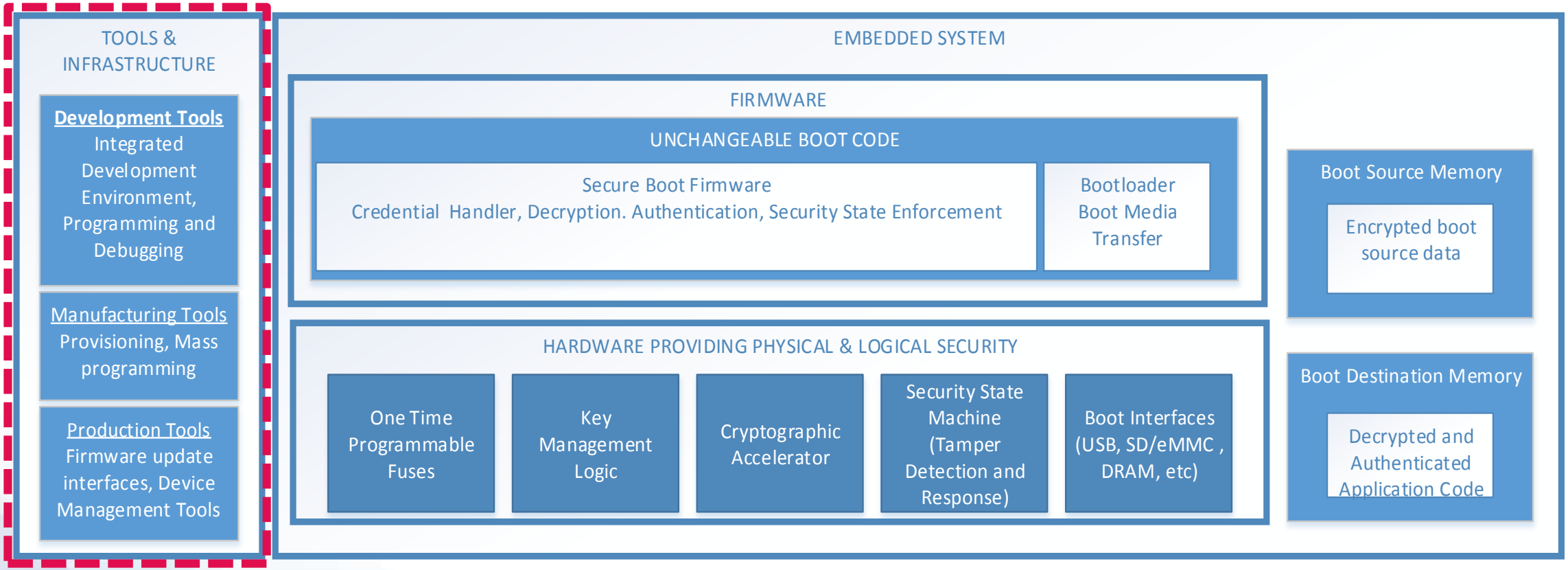
- Option for Authenticated and encrypted handling of boot data
- Authentication always occurs first on the data
- SRK hash checks integrity
- Hardware (SNVS/Security state machine) dictates what operations are performed



i.MX RT Security Technology

Tools and Infrastructure

Secure Boot Architecture Diagram



Tools: Flashloader Package

Elftosb

- Creates secure binary (SB) files
- Operates with Input file (BD)
- Automatically formats the boot data components

Blhost

- The blhost tool is a host utility that provides command line access for an i.MX RT device which is running the flashloader
- This tool facilitates the provisioning process. For example, it can be used to command the target processor to create encrypted blocks of data which are necessary for the authenticated and secure boot process.

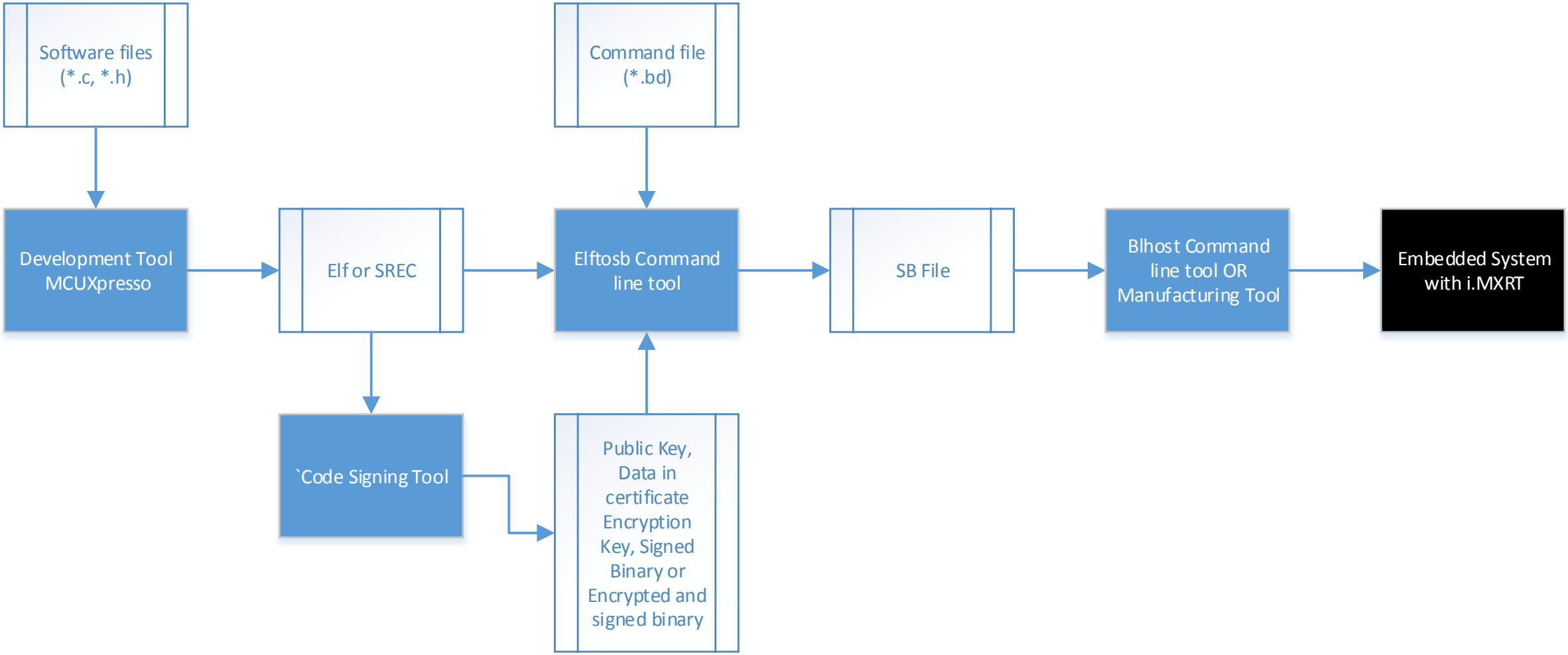
Manufacturing tool

- Manufacturing tool is an abstraction of the blhost tool. It runs on Windows machines and presents a graphical user interface to allow connection to a target and download of SB files that contain the bootable image.

Code Signing Tool

The code signing tool is a command line host tool that can be used to generate keys (symmetric and asymmetric), sign images and encrypt images for use in the secure boot.

Example Tools Flow



Key Management

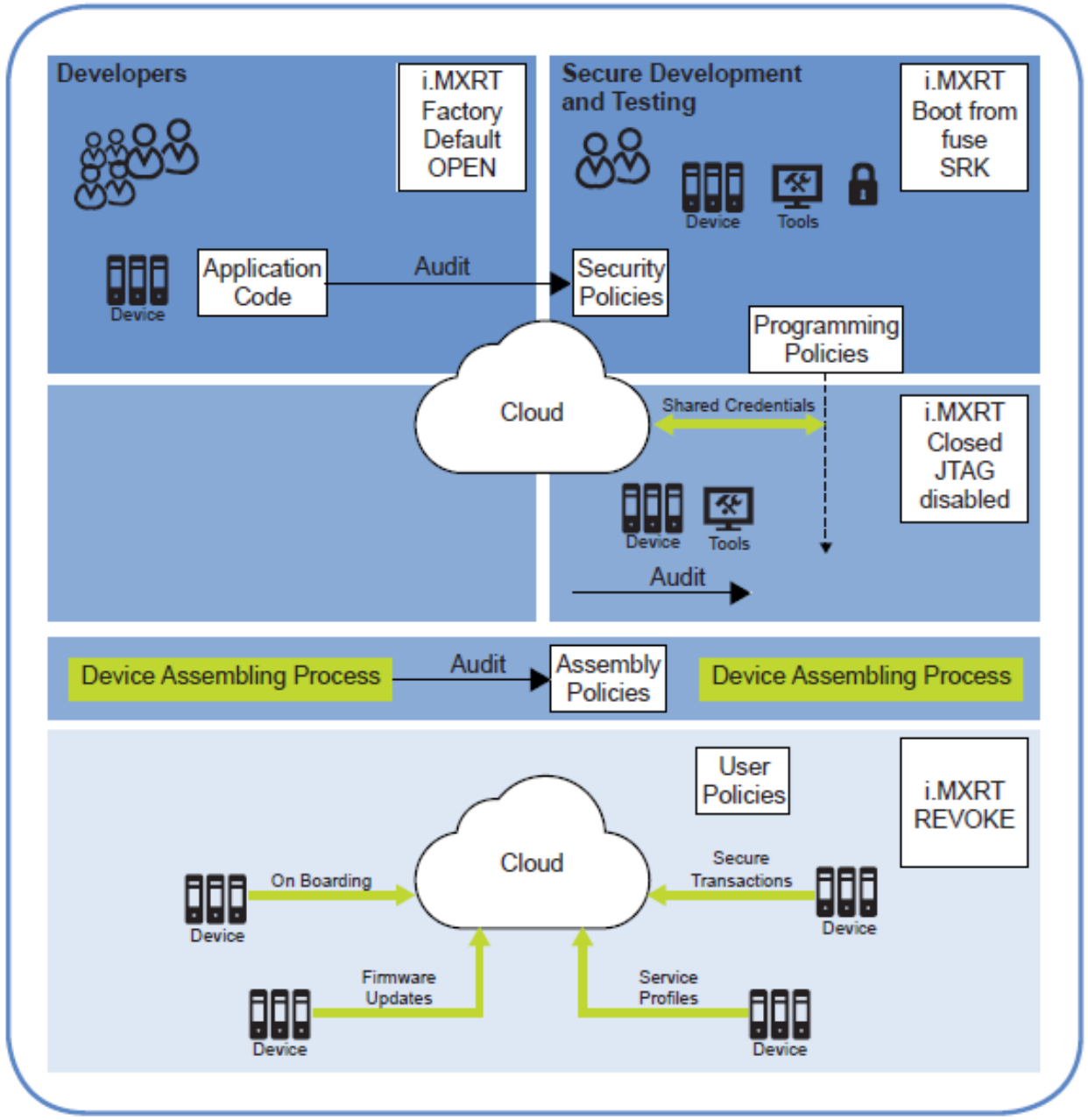
Key Management Table

Key Name	Description	Owner	Key Generation	Key Storage
Boot Private Key (1-4)	RSA Private key used for creating signatures of application binaries	OEM	Generated by the code signing tool	Trusted OEM machine
Boot Public Key (1-4)	Associated RSA public key for authenticating boot code. This key is inserted into a certificate which becomes part of the boot data.	OEM	Generated by the code signing tool	Not a secret key, checked for integrity by SRK Hash stored on chip OTP
Data Encryption Key (DEK)	AES128 bit symmetric key which is used to encrypt (AES-CCM) application code and data.	OEM	Generated by the code signing tool	Trusted OEM Machine
One-time programmable master key (OTPMK)	AES128bit symmetric key which is used to protect the DEK.	i.MX RT Device	Installed during NXP Manufacturing by using the on chip TRNG	On chip fuses. Checked for integrity by SNVS (Security state machine)

Lifecycle View

Example Lifecycle

Starting with the development phase, in both *Less trust environments* and *Secure Environments*, software development can be done on the factory default settings of the chip. This will have the Security Configuration fuses set to OPEN. As shown in the diagram, to maintain security, software must still be audited to follow device security policies. For example, ensuring the device does not prompt users to enter personal bank information, or restrict the dictionary of words a device can speak.



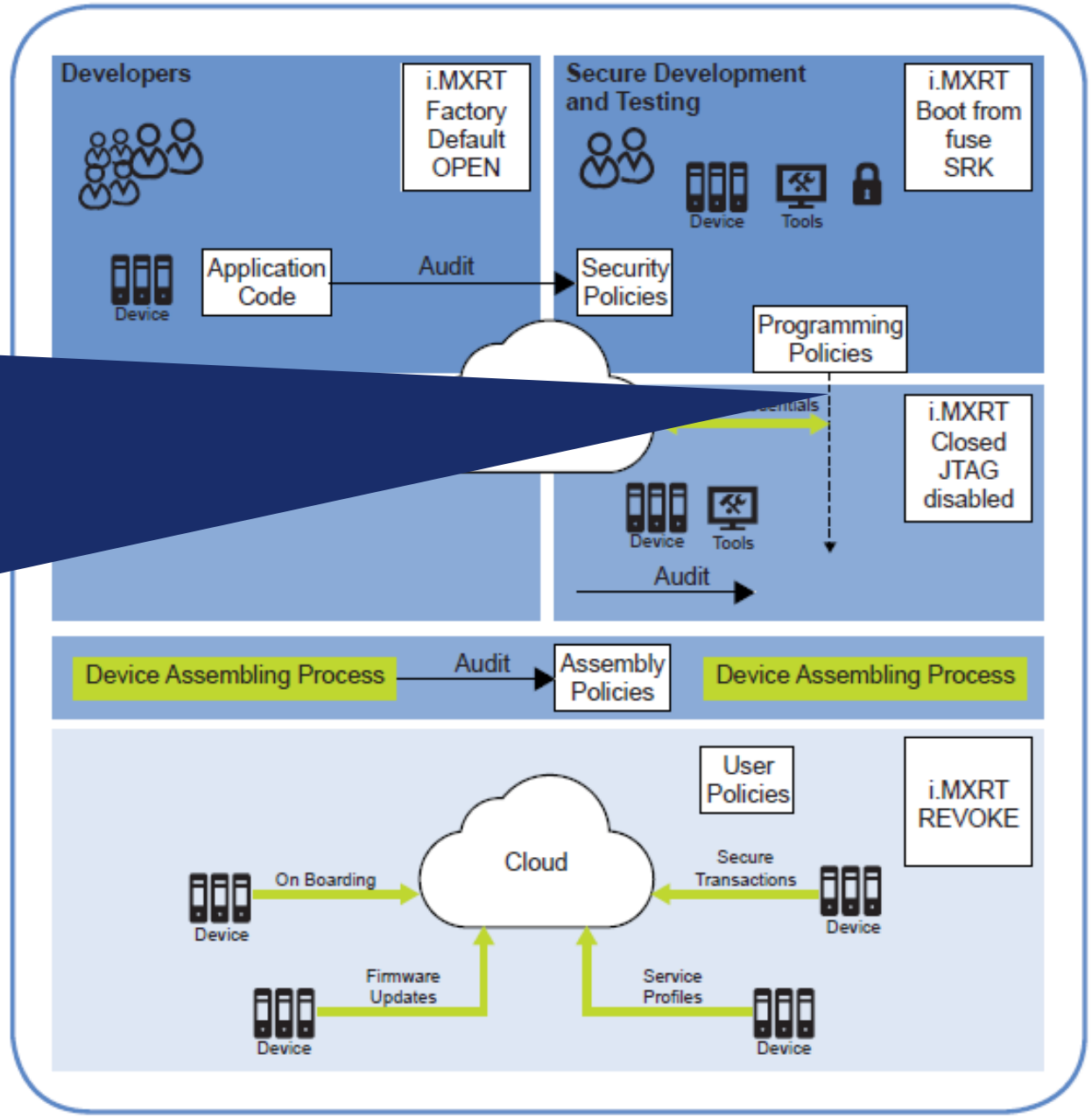
Example Lifecycle

In the development phase there is testing of the secure boot process. It will be necessary for **secure developers** to create the specific implementation needed for the hardware configuration of the end device.

i.MX RT will be configured to boot from fuses and with the intended interfaces for the target hardware.

Code Signing Tool is used to create the test Boot Private Key and Public Key pairs. In this phase the detailed steps needed to provision the device for the secure boot are created.

Programming Policies are needed to protect the process of installing the chip fuses and establish the root of trust for the chip.



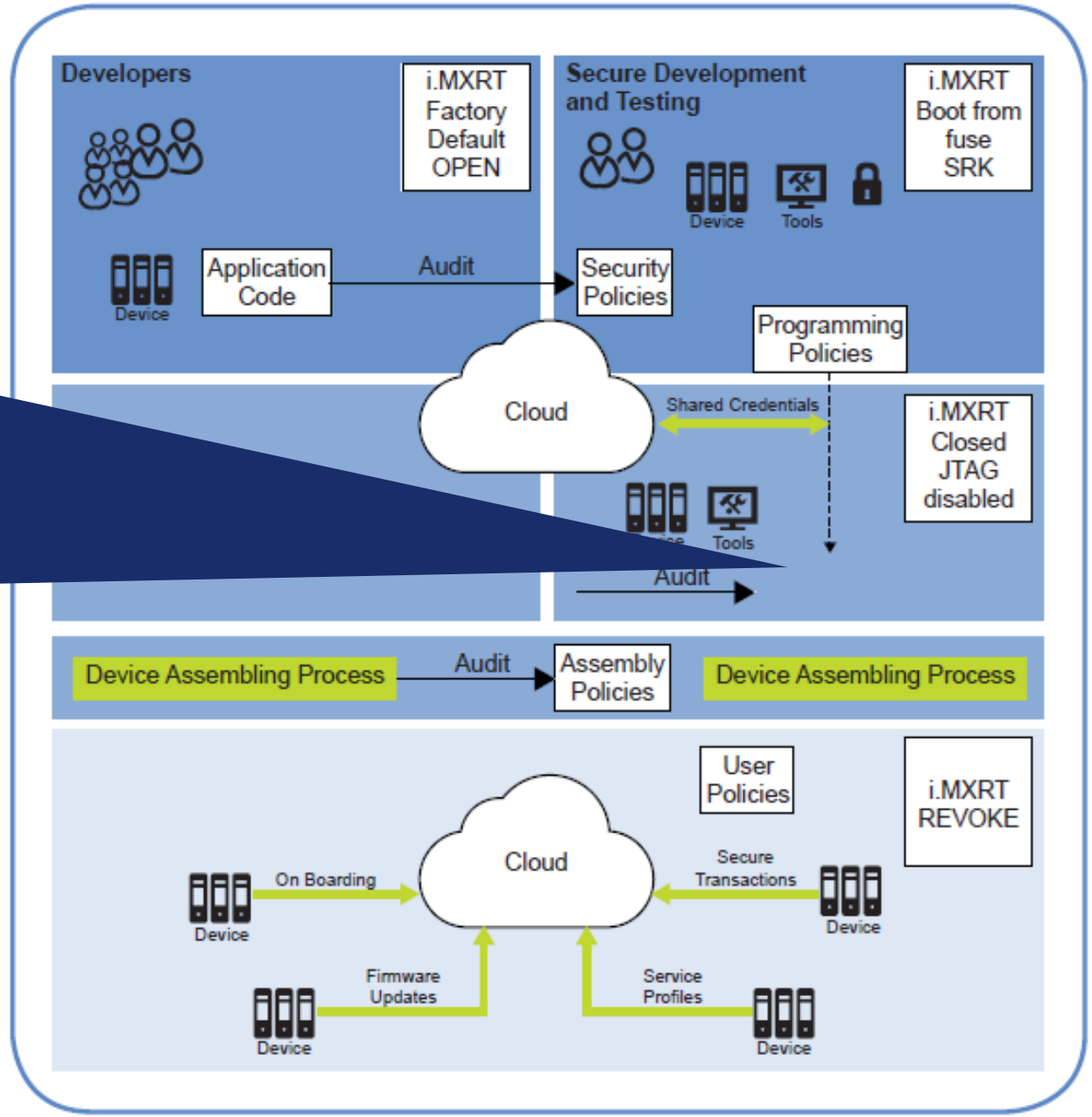
Example Lifecycle

With the *Programming Policies* in place, the Manufacturing Stage uses the tools as represented in the example tools flow to provision each device.

Each i.MX RT receives its public keys and application data which is encrypted with a key that is only accessible with the One Time Programmable Master Key (OTPMK) of the chip.

Part of the data which is passed to the device in this phase includes the shared credentials between the Cloud and the device. This enable use cases shown in the deployment phase.

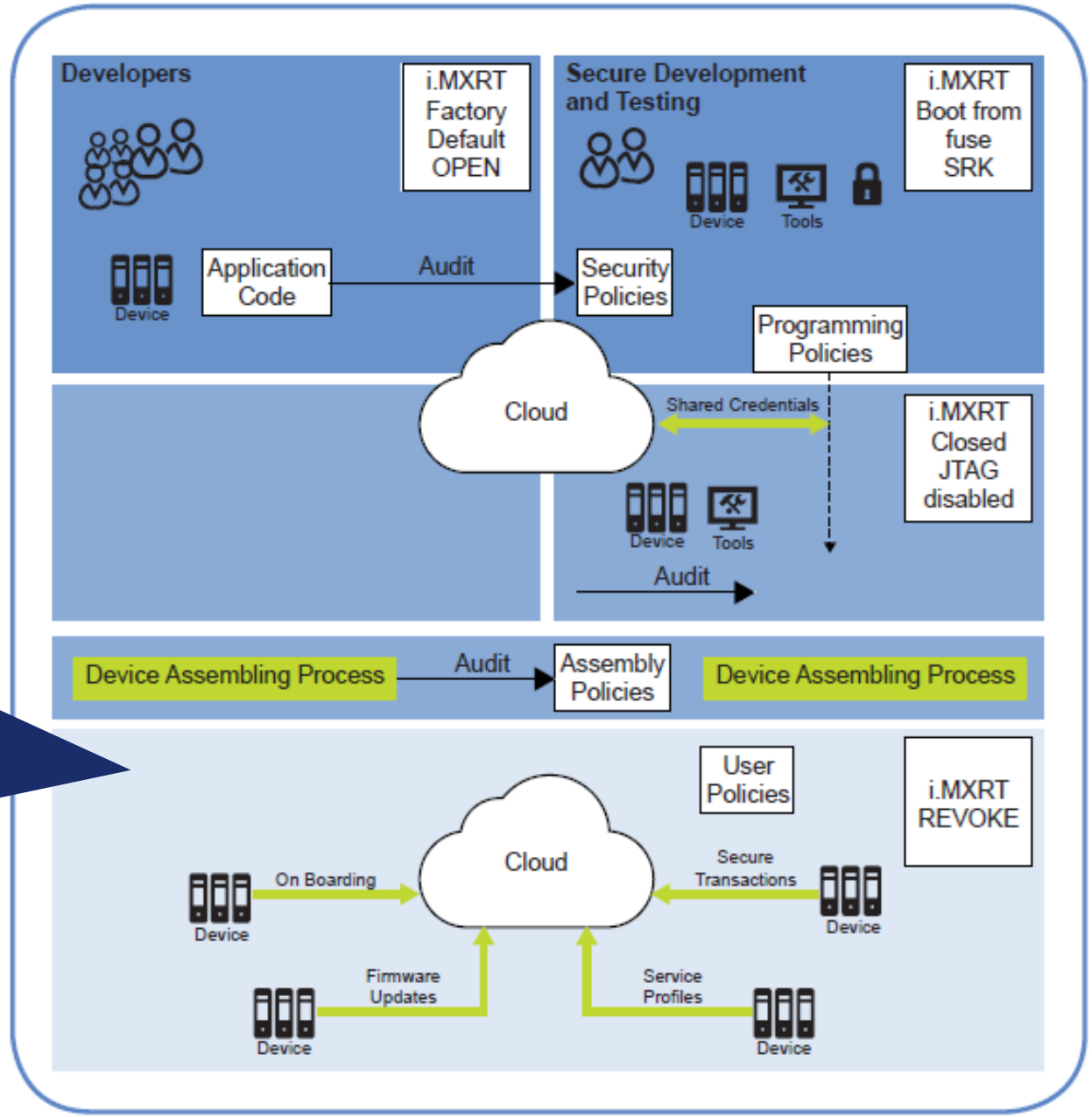
In this phase the i.MX RT fuses are set for Security Configuration Closed and the debug interfaces are locked.



Example Lifecycle

Once the chip is provisioned, it can be assembled in either Less Trust Environments or Secure Environments as represented in the figure.

For both cases, there should be *Assembly Policies*. Assembly policies ensure that only the approved components are assembled within the device. It provides guidance for inspection, such as details on the chip markings and pictures of the final assembled PCB board.

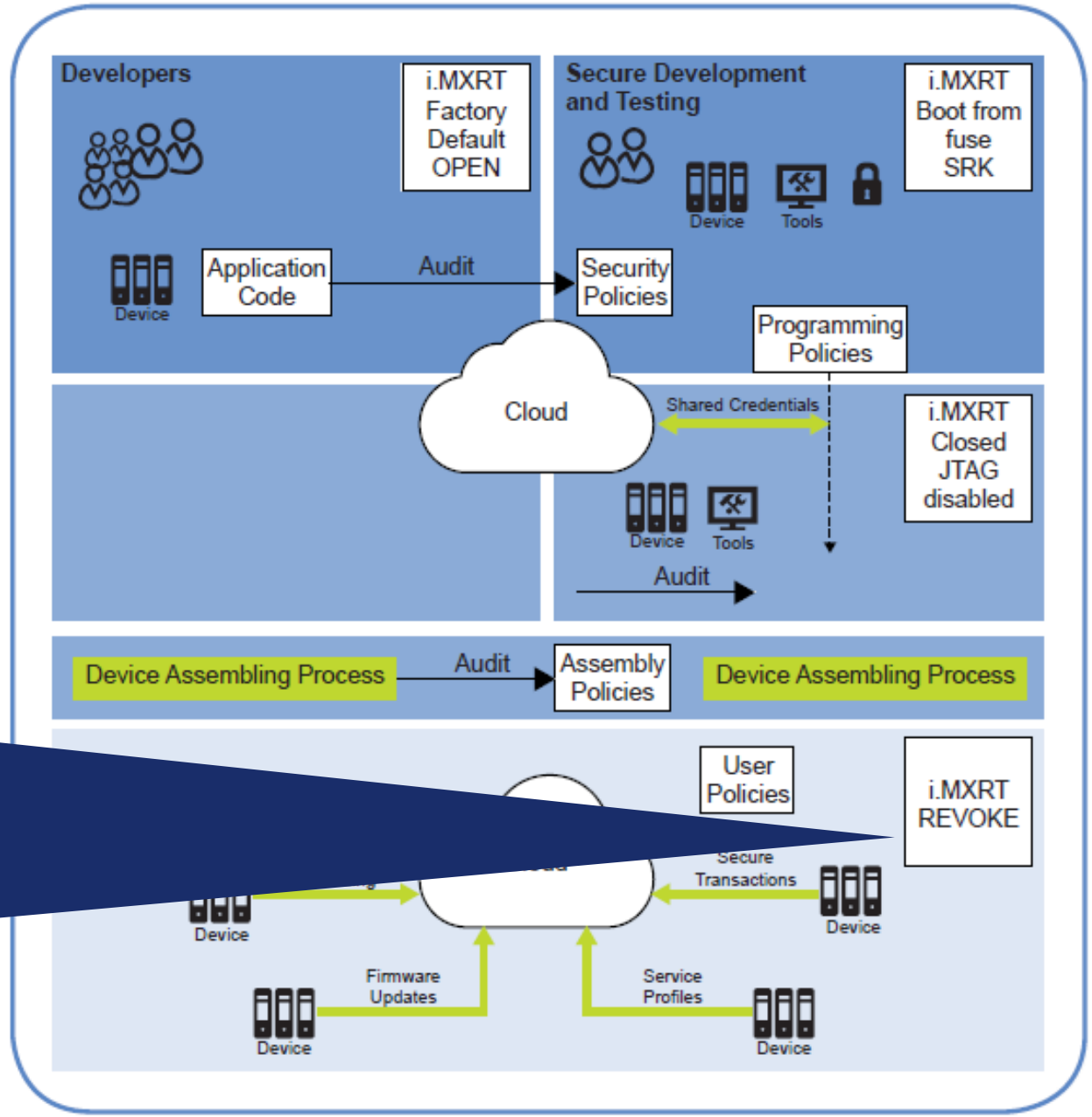


Example Lifecycle

In the deployed phase, the lifecycle of the device is maintained using the SRK Revoke fuses available in the chip.

When an OEM decides that previously signed firmware should no longer be allowed for the end device, they can use the SRK Revoke to move to the next set of Boot Private and Boot Public keys by blowing a fuse.

Because the hardware security state machine enforces the allowable operations, if older versions of application code are loaded, they will be rejected by the secure boot process.



Enabling Secure Transactions

Alignment to Security Goals

Counterfeit protections

With the authenticated and encrypted boot, security is enforced by a unique secret available only to the individual chip (OTPMK)

As the application code must be linked to the chip for it to be used, this establishes a link between known devices and the application functions to protect against clones

Onboarding

Chip specific unique and protected keys along with secure boot flow protect OEM installed cloud credentials

During manufacturing cloud credentials are encrypted with chip specific unique & protected keys

Cloud credentials become part of the secure boot image that is protected for integrity and confidentiality

System Integrity

Secure boot functions upon every reset and is the foundation for establishing trust in the device operation

Chip hardwre and ROM provides an immutable secure boot flow to support recovery from system run away scenarios once the device is rebooted.

Alignment to Security Goals (Continued)

Secure Communication

Authenticated application code includes TLS Stacks (WolfSSL or Arm MbedTLS)

Option for AES engine to use OTP or application generated keys

Hardware acceleration for AES and SHA-2 (SHA-256) with DCP key protection

Data Confidentiality

Based on device policies, data stored in system is protected by hardware managed keys

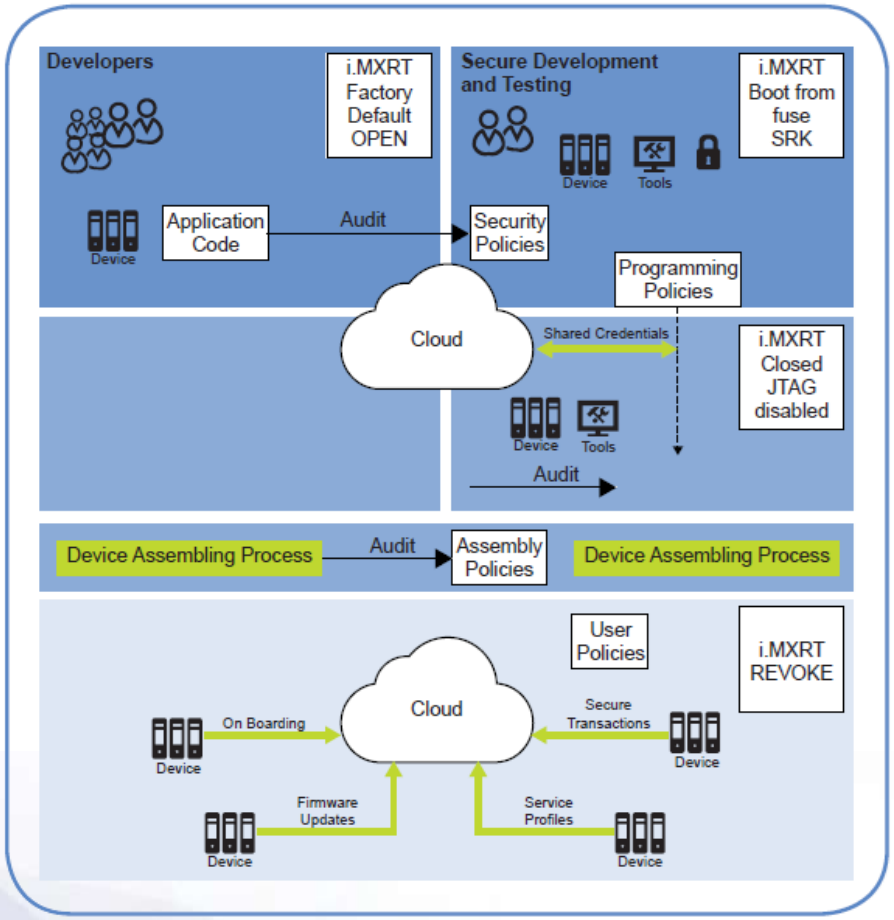
Option for AES engine to use OTP or Applicaiton generated keys

Hardware acceleration for AES and SHA-2 (SHA-256

Secure firmware update

New firmware applied to the system must pass the secure boot flow

ROM support for up to 3 revocations using SRK Revoke



Hands On

- Installation of tools for supporting code signing and provisioning for i.MX RT
- Generating Key Pairs
- Using elftosb to program fuses for setting Super Root Key Hash
- Using elftosb to sign application firmware
- Using Manufacturing tool to program signed firmware

Conclusions and Resources

Conclusion

- With the right hardware, software, tools and methodologies, the system designer can ensure that their creations support secure transactions over the lifecycle of the device.
- Realizing today’s security requirements is more achievable than ever with the latest class of crossover processors such as the i.MX RT.
- Secure boot design depends on the OEM configuration of the device. The OEM must have people and processes in place.
- The following table provides the resources needed to further investigate the essential secure boot design.

Resources

Processor summary page	The i.MX RT1050 family summary page provides links to chip documents (Data Sheet and Reference Manual)
Security Reference Manual	The i.MX RT security reference Manual. The security reference manual can be requested and is available for NDA customers
Security Application Note	Application note for i.MX RT security features
Hardware evaluation kit	The i.MX RT EVK provides a platform for embedded development. Multiple boot interfaces are supported
Software SDK	The MCUXpresso SDK is the software enablement which provides drivers and middleware for the i.MX RT
Flashloader tools package	Grouping of the manufacturing tools needed for provisioning the secure boot. This includes flashloader, elftosb, blhost and the manufacturing tool gui
Code Signing Tool	NXP Tool for creating keys and signing application code and data

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com