EmSPARK™ Security Suite

Provisioning and Secure Boot with the EmSPARK™ Security Suite for i.MX8 (M, Mini, Nano, Plus) Device Family

Date September 22, 2021 | Version 1.0

TABLE OF CONTENTS

1. INTRODUCTION

The EmSPARK™ Secure Boot identifies, authenticates, and starts up the software on the device when the device is powered on. In addition, it provides boot resiliency and facilitates secure firmware update. Several stages of authentication, protection, and encryption/decryption protect the firmware images stored in the non-volatile memory (NVM) whether or not the device is powered on. Boot resiliency implements failover logic using two boot stacks and an i.MX8 hardware watchdog. Secure firmware update verifies the authenticity of update payloads.

Cryptographic keys used for boot authentication operations are setup during the device provisioning. Some of such keys and certificates are customer configured in the provisioned firmware image and some keys and certs are device-specific generated during provisioning. The EmSPARK Evaluation Kit has preconfigured the keys and certificates that a customer would configure when building the firmware. The associated private key files are provided to facilitate the execution of example applications.

To support Secure Boot, the EmSPARK Security Suite includes:

- Tools, applications and procedures to cryptographically protect the firmware images for installation on the device
  - Tools to configure keys and certificates that after installed on the device can be rotated during the device life cycle
  - Tools to generate encrypted firmware components
- Provisioning application
  - Provisioning application to install the firmware
- CoreLockr APIs to develop applications that run on the device rich OS and provide ability to
  - Manage during runtime selected keys used during boot
  - Conduct preliminary verification of update payloads

Devices provisioned with the EmSPARK Security Suite are also EmPOWER™ enabled. EmPOWER is a SaaS solution that provides the essential cloud services needed to secure, provision, update and manage intelligent edge devices. For information about EmPOWER please contact Sequitur Labs.

This document explains how the EmSPARK Security Suite instruments the Secure Boot capabilities. Because the cryptographic material on the device is initially installed or generated

during provisioning, this document includes a provisioning process overview. Additional documentation describing the tools, applications and procedures to cryptographically protect the firmware images for installation on the device is available upon request. For information about the CoreLockr APIs, see additional EmSPARK Evaluation Kit documentation.

Descriptions in this document occasionally mention tools included in the EmSPARK Development Kit which are not part of the EmSPARK Evaluation Kit. Please contact Sequitur Labs for additional information.

Secure Boot attests the firmware during the boot process. The boot chain comprises several individually protected components. Keys for the hardware Root of Trust (RoT) and unique device identification are configured or generated during provisioning. See **2 Secure Provisioning Overview** section for provisioning overview.

1.1. Acronyms

ATF	ARM Trusted Firmware
BRN	Boot Random Number
CAAM	Cryptographic Acceleration and Assurance Module.
CSR	Certificate Signing Request.
CST	NXP Code Signing Tool for code signing and for use with the HAB library.
HAB	High-Assurance Boot
NVM	Non Volatile Memory
OEM	Original Equipment Manufacturer
OTPMK	One-Time Programmable Master Key. The OTPMK fuse value is programmed by NXP during chip manufacture. It is unique per device key and used as a seed for key derivation.
ROM	Read Only Memory
SPL	Secondary Program Loader
SRK	Super Root Key
SRKH	Super Root Key Hash
TA	Trusted Applications

1.2. Terminology

BLOB	Cryptographic Binary Large Object. Blobs contain encrypted data and an encrypted version of the key used to decrypt the data. On a device, a Blob is encrypted and decrypted using device-specific keys derived from the OTMPK. See **2 Secure Provisioning Overview** section.
CoreTEE™	Sequitur's Trusted Execution Environment (TEE), or secure OS, enabled by ARM's TrustZone™ architecture.
Gold Blob	Blobs created in a Gold System. During provisioning, the provisioning application decrypts the Gold Blobs and re-encrypts the data with device-specific keys.
Gold System	A device flashed with a special purpose firmware with the objective of generating Gold Blobs. The board must have the SRKH fuses programmed and be closed. Gold Systems are capable of generating a BRN, which is used to help create the keys used in the encryption of Gold Blobs.

Personalization Manifest	Customer configured data including keys and certificates that are part of the components provisioned on the device.
Plex	A set of components in a boot stack. Two plexes are implemented (active and non-active) and used for failover logic and secure update.
Provisioning SPL	Application that executes secure provisioning, see **3 Secure Boot Overview** section.
SLIP	Encrypted component containing customer personalization data such as keys and certificates. Personalization data is configured in manifests, which after built and encrypted are called SLIPs. They are installed on device along with the device firmware.

2. SECURE PROVISIONING OVERVIEW

The EmSPARK suite tools and processes facilitate the generation of cryptographically protected firmware components for provisioning on the device. This section is an overview of the provisioning preparation steps, provisioning process, and resulting behavior and cryptographic material on the device after provisioning.

1. Preparation Steps

The preparation steps include:

- Configure and build provisioning SPL: corresponds to the provisioning application
- Configure and build production SPL: the primary SPL on production devices
- Sign provisioning SPL: signing with NXP's Code Signing Tool
- Sign production SPL: signing with NXP's Code Signing Tool
- Build additional secure components: U-Boot, FDT, Kernel, ATF
- Configure and build personalization data manifests: contain customer keys and certs
- Place CoreTEE binary in build environment: Sequitur provides CoreTEE
- Encrypt components: Gold Blobs generated in a Gold System
- Prepare additional components for the Rich OS filesystem
- Produce provisioning image including provisioning manifest: using tools included in the EmSPARK Development Kit

The Gold System is provided to customers along with the EmSPARK Development Kit. The Gold System produces the Gold Blobs used to create the firmware provisioning image. For integration details, please see the Gold System Guide document provided with the EmSPARK Development Kit.

2. Provisioning Process

The provisioning process uses the NXP manufacturing protection capabilities to decrypt the Gold Blobs and re-encrypts the components using the One Time Programmable Master Key (OTPMK). The hardware RoT supports a fully encrypted boot chain. The RoT is the first key in the root process to secure the device, the i.MX8 the process uses the OTPMK feature during the early boot stages. The provisioning process generates selected keys and certificates, and installs additional keys and certificates contained in the customer personalization data manifests.

For i.MX8, the provisioning application is the provisioning SPL. The provisioning application generates random keys for the root of trust (RoT), generates unique device identification, decrypts the firmware components and re-encrypts them with device-specific keys. The following is an outline of the provisioning application steps:

1. Program fuses
    - SRK Hash fuses, SRK hash value used by High-Assurance Boot (HAB)
    - Lock fuses, configure the boot security setting to “Closed” so that HAB functions are executed, the security hardware is initialized and the production SPL is authenticated by the HAB before its execution.
2. Initialize the CAAM and device drivers
3. Load from NVM the provisioning manifest
4. Load from NVM the components in the component index, encrypted as Gold Blobs
5. Install production SPL
    - Load component
    - Decrypt the SPL using the manufacturing protection feature available on the i.MX8 processor
    - Write the signed production SPL to NVM
6. Install additional components in the component index, for each component
    - Load component
    - Decrypt the component using manufacturing protection
    - Re-encrypt the component with device-specific keys using the OTPMK
    - Write the component to NVM
    - Setup the following components for two boot stacks, primary and non-primary to support failover logic
        - U-Boot
        - ATF
        - CoreTEE
        - Kernel
        - FDT
7. Install personalization data contained in manifests and generate unique device identification
    - Load manifests containing keys and certs
    - Decrypt manifests
    - Generate device keys and certs
        - Generate OEM Device Key
        - Generate EmPOWER Device Key
        - Generate OEM Device CSR and OEM Device Certificate
        - Generate EmPOWER Device CSR and EmPOWER Device Certificate
    - Generate AES keys to encrypt manifests
    - Re-encrypt manifests with device specific keys
        - Encrypt manifest containing keys and certs
        - Encrypt manifest containing component index for both boot stacks
    - Write manifests to NVM
8. Finish provisioning

3. Provisioned Device

After the provisioning process completes:

- The fuses are programmed setting up the hardware root of trust on the device
- The hardware root of trust supports a fully encrypted boot chain with device-specific keys
- The firmware is protected in non-volatile storage by encryption while the device is at rest
- The production SPL supporting failover and firmware update logic is installed
- The secure components are installed in two boot stacks
- Keys and certificates originated from personalization manifests or generated during provisioning are stored encrypted in NVM
- The firmware is EmPOWER enabled

The following keys and certificates exist on the device after provisioning:

- Fuse credentials
  - OTPMK, the fuse value is programmed by NXP during chip manufacture
  - SRKH
- AES keys generated to encrypt the personalization data. AES keys are available to secure components only (SPL and CoreTEE). They are not exposed to the Rich OS. When a user modifies the manifests throughout the device life cycle, the manifests are re-encrypted by CoreTEE using the AES key for that manifest. The encrypted manifest is then saved to NVM at its original location for access on the next boot.
- Personalization credentials available during runtime:
  - For use in customer designed scenarios
    - **OEM Device Key**, key pair unique per device, immutable
    - **OEM Device Public Key**, pair of the private key, customer decides usage scenarios including production of encrypted payloads such as opaque keys and opaque objects tailored to the specific device
    - **OEM Device Certificate**, customer decides usage scenarios, including TLS mutual authentication
    - **OEM Public Key**, used for OEM signature verification, customer decides usage scenarios including opaque keys and opaque objects
    - **OEM Root Certificate**, contains the OEM Public Key, customer decides usage including OEM key rotation
    - **OEM Cloud Certificate**, customer decides usage scenarios, TLS mutual authentication
    - **OEM Cloud Public Key**, customer decides usage scenarios, including Cloud signature verification
    - **OEM Payload Public Key**, used to verify update payloads
    - **OEM Command Public Key**, used to verify commands modifying the Certificate Store and the Key Store which are managed in the TEE

  - For use with EmPOWER services
    - **EmPOWER Device Key**, key pair unique per device, immutable
    - **EmPOWER Device Public Key**, pair of the private key
    - **EmPOWER Device Certificate**, used for TLS authentication with EmPOWER cloud service
    - **EmPOWER Root Certificate**, used for EmPOWER cloud connectivity
    - **EmPOWER Public Key**, public key contained in EmPOWER Root Certificate

- **EmPOWER Cloud Cert**, used for TLS mutual authentication with EmPOWER cloud service
- **EmPOWER Cloud Public Key**, public key contained in the EmPOWER Cloud Cert

Please see the CoreLockr™ API documentation for additional key and certificate description. Information about configuration and usage is available on the KEYS and CERTIFICATES document provided with the EmSPARK Development KIT.

3. SECURE BOOT OVERVIEW

Secure boot supports secure update and failover functionality that implements two boot stacks, primary and non-primary. The trusted boot chain starts on reset and completes when the application is running on the device. The i.MX8 provides a robust secure boot mechanism through NXP's HAB hardware. This allows verification of the initial payload against a fuse pattern (SRKH) to authenticate that the loaded payload is authorized to run on the device. The i.MX8 boot payload includes a header containing all the public keys used to create the SRKH and specifies which one is used with this payload. However, this step is only the first link in the boot chain. For more specifics on this process, please refer to the NXP documentation on the HAB.

EmSPARK provides a reference based on the NXP BSP to securely boot the rest of the software. The ROM verifies the production SPL image. In the secure RAM, SPL will set up the Secure Environment. SPL configures the device and loads manifests containing security credentials stored in NVM during provisioning, CoreTEE, U-Boot, the Linux Kernel and DTB into RAM. All components are encrypted using the NXP CAAM Blob function and will be decrypted during the loading process. SPL transfers control to ATF then ATF to CoreTEE.

CoreTEE initializes during this phase. The initialization includes loading all the credentials that are made available during runtime, enabling the TrustZone address space controller and separating memory into secure and non-secure regions.

At the completion of this initialization, CoreTEE returns to ATF and ATF exits with exception to U-Boot in the non-secure world. U-Boot now completes the boot process, including booting Linux. Linux boots using its normal process. **Figure 1** depicts the process.

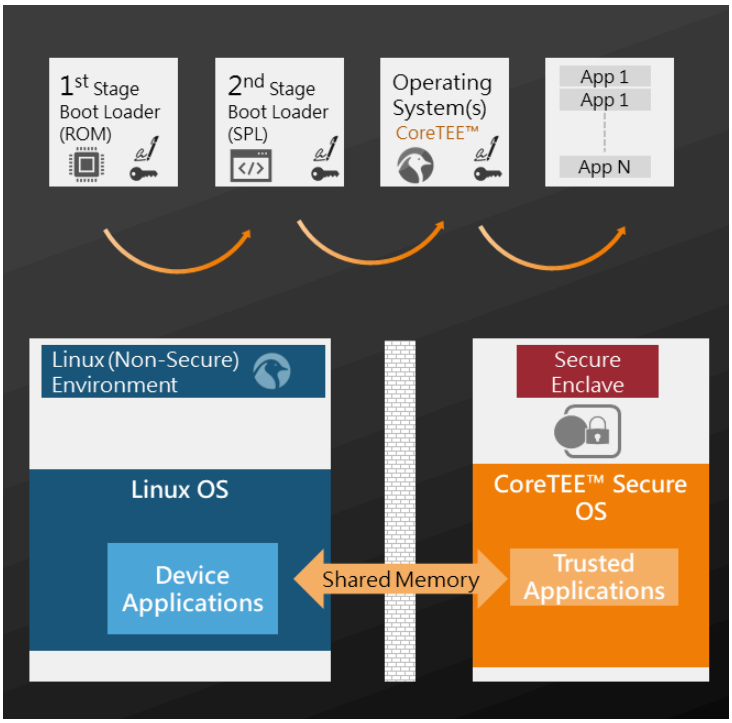

Figure 1 Secure Boot

The following sections are an overview of the secure boot steps and illustrate sample output printed on the device console during boot.

1. **Initiation, Pre-boot**
   - When the device is powered on, as a result of the fuses programmed during provisioning when the internal boot mode was set to "Closed" security configuration, HAB functions are executed and the security hardware is initialized.

2. **Software Authentication**
   - ROM loads the signed production SPL
   - ROM verifies the SPL signature against the SRKH written in fuses during provisioning
     - If the signature verification is successful, then ROM loads SPL to RAM
   - SPL sets the watchdog time.

```
The SPL has passed verification against SRKH in fuses.

Running Sequitur Labs Secure Boot Steps.
Setting watchdog time to 60s
```

3. **Memory Isolation, Trusted Execution Environment (TEE) Establishment and Application Authentication**
   - SPL starts up in the secure environment. It will handle the components.

```
-----------------------------------------------------------
Setting Up Secure Environment
Loading into Secure RAM and decrypting blobbed components
-----------------------------------------------------------
```

   - SPL checks the boot state

    - If “update” then apply the update logic, see **4 Update** section
    - Else, continue normal boot with primary boot stack
- SPL loads the Gold Blobs into the RAM secure area. Each Gold Blob is authenticated and un-encrypted
    - SPL loads personalization data manifests containing keys, certs and license keys stored on NVM during provisioning. Each manifest file is called a SLIP. For each manifest:
        - Read encrypted manifest
        - Decrypt manifests using AES keys generated on device during provisioning

```
Processing blobbed slip [layout]
Loading slip - pd_seq - from NVM
Processing blobbed slip [pd_seq]
Loading slip - pd_oem - from NVM
Processing blobbed slip [pd_oem]
Loading slip - certs - from NVM
Processing encrypted slip [certs]
Loading slip - empower - from NVM
Processing encrypted slip [empower]
```

    - SPL sets up components in the primary boot stack, for each component:
        - Load encrypted component
        - Decrypt and authenticate component using device-specific keys derived from OTPMK, note that 0x00 means no error

```
State: 0x1c
Setting BLC to: 5
SPL in process of updating [0]
Error value for: SPL  is  0x00
Error value for: CT  is  0x00
Error value for: UB  is  0x00
Error value for: LX  is  0x00
Error value for: FS  is  0x00
Setting BLC to: 4
[check_boot_state] - Stateval: 0x1c

Loading firmware version [eval] for Plex A
Calling component_setup
Loading KERNEL from GOLD BLOB
Loading u-boot from GOLD BLOB
DTB loaded from U-Boot
Loading CoreTEE from GOLD BLOB
Loading ATF from GOLD BLOB
```

        - If unable to set up any component, proceed with failover logic, Secure Boot follows the failover boot path to activate the non-primary boot stack
- SPL passes control to ATF and CoreTEE is initialized

```
Calling into ATF...
```

- CoreTEE sets up the secure operating environment including the loading of provisioned credentials to be available during the device operation
- CoreTEE returns control to ATF, up to this point the process has been running in secure memory. ATF exits with EL3 exception level to U-Boot

```
-----------------------------------------------------------
Separating RAM into Secure and Non-Secure
-----------------------------------------------------------
```

4. **Memory Isolation, Non-Secure Environment Set Up**
   - U-Boot runs in non-secure memory

```
U-Boot ...
U-Boot is running NON-SECURE
```

   - U-Boot initializes and sets up the Rich OS and applications in the non-secure environment
     - Load Linux which has a CoreTEE driver
     - Load DTB
   - The device boots to Linux and the process completes

```
NXP i.MX Release Distro 5.4-zeus imx8mp-var-dart ttymxc1
imx8mp-var-dart login:
```

4. UPDATE

During boot, SPL checks the boot state for updates. If boot state is “update”, the following sequence takes place:

1. Load the signed update payload containing encrypted components and update manifest
2. Verify the update payload signature using the OEM Payload Verification Public Key stored on the device during provisioning
3. If SPL component is updated, apply SPL update sequence
4. If other components are updated, proceed to update the non-primary boot stack
5. For each component in the update payload
   - Decrypt the Gold Blob component using manufacturing protection
   - Re-encrypt the component with device-specific keys using OTPMK
   - Write the component to NVM
6. Update and write to NVM component index manifest
7. Upon successful update completion
   - Clear update SPL flag
   - Activate non-primary boot stack to boot the device with updated firmware
   - Update boot state
8. Continue normal boot