AN1222: Production Programming of Series 2 Devices

This application note demonstrates how to properly program, provision, and configure Series 2 devices in a production environment.

Series 2 devices contain a Secure Element, which runs Secure Element Firmware. When a newer version of Secure Element Firmware is released, the firmware may be upgraded either in the production programming process for devices still in manufacturing or via a field update for devices already deployed in the field. Keys must be provisioned to the Secure Element's one-time-programmable memory to use the Secure Boot and Secure Debug features.

For more information about Secure Element, see section "Secure Element Subsystem" in application note AN1190: Series 2 Secure Debug.

KEY POINTS

- It is the customer's responsibility to ensure the Secure Element Firmware is up-to-date
- The Secure Element Firmware can be upgraded via the Serial Wire Debug (SWD) interface
- Secure Element Firmware is protected from downgrade
- Secure Element's one-time-programmable memory (OTP) prevents re-writing of:
  - GBL Decryption Key
  - Public Sign Key
  - Public Command Key
  - Secure Boot Enable flag

1. EFR32 Series 2 Device Security Features

Protecting IoT devices against security threats is central to a quality product. Silicon Labs offers several security options to help developers build secure devices, secure application software, and secure paths of communication to manage those devices. Silicon Labs' security offerings were significantly enhanced by the introduction of the EFR32 Series 2 products that included a Secure Element. The Secure Element is a tamper-resistant component used to securely store sensitive data, keys and to execute cryptographic functions and secure services.

The Secure Element is the foundation of two core security functions:

- Secure Boot: Process where the initial boot phase is executed from an immutable memory (such as ROM) and where code is authenticated before being authorized to be executed.
- Secure Debug access control: The ability to lock access to the debug ports for operational security, and to securely unlock them when access is required by an authorized entity.

Some EFR32 Series 2 products offer additional security options through Secure Vault. Secure Vault is a dedicated security CPU that isolates cryptographic functions and data from the host processor core. Devices with Secure Vault offer the following security features:

- Secure Key Storage: Protects cryptographic keys by "wrapping" or encrypting the keys using a root key known only to the Secure Vault.
- Anti-Tamper protection: A configurable module to protect the device against tamper attacks.
- Device authentication: Functionality that uses a secure device identity certificate along with digital signatures to verify the source or target of device communications.

A Secure Element Manager and other tools allow users to configure and control their devices both in house during testing and manufacturing, and after the device is in the field.

1.1 User Assistance

In support of these products Silicon Labs offers whitepapers, webinars, and documentation. The following table summarizes the key security documents:

Document	Summary	Applicability
AN1190: Series 2 Secure Debug	How to lock and unlock EFR32 Series 2 debug access, including background information about the Secure Element	EFR32 Series 2
AN1218: Series 2 Secure Boot with RTSL	Describes the secure boot process on EFR32 Series 2 devices using Secure Element	EFR32 Series 2
AN1247: Anti-Tamper Protection Configuration and Use	How to program, provision, and configure the anti-tamper module	EFR32 Series 2 with Secure Vault
AN1268: Authenticating Silicon Labs Devices using Device Certificates	How to authenticate a device using secure device certificates and signatures, at any time during the life of the product	EFR32 Series 2 with Secure Vault
AN1271: Secure Key Storage	How to securely "wrap" keys so they can be stored in non-volatile storage.	EFR32 Series 2 with Secure Vault
AN1222: Production Programming of Series 2 Devices (this document)	**How to program, provision, and configure security information using Secure Element during device production**	**EFR32 Series 2**

1.2 Key Reference

Public/Private keypairs along with other keys are used throughout Silicon Labs security implementations. Because terminology can sometimes be confusing, the following table lists the key names, their applicability, and the documentation where they are used.

Key Name	SE Manager ID	Customer Programmed	Purpose	Used in
Public Sign key (Sign Key Public)	SL_SE_KEY_SLOT_APPLICATION_SECURE_BOOT_KEY	Yes	Secure Boot binary authentication and/or OTA upgrade payload authentication	AN1218 (primary), AN1222
Public Command key (Command Key Public)	SL_SE_KEY_SLOT_APPLICATION_SECURE_DEBUG_KEY	Yes	Secure Debug Unlock or Disable Tamper command authentication	AN1190 (primary), AN1222, AN1247
OTA Decryption key (GBL Decryption key) aka AES-128 Key	SL_SE_KEY_SLOT_APPLICATION_AES_128_KEY	Yes	Decrypting GBL payloads used for firmware upgrades	AN1222 (primary), UG266
Attestation key aka Private Device Key	SL_SE_KEY_SLOT_APPLICATION_ATTESTATION_KEY	No	Device authentication for secure identity	AN1268

2. Device Compatibility

This application note supports Series 2 device families. Some functionality is different depending on the device.

Wireless SoC Series 2 families consist of:

- EFR32BG21A/EFR32BG21B/EFR32BG22
- EFR32FG22
- EFR32MG21A/EFR32MG21B/EFR32MG22

3. Overview

On Series 2 devices, the security features are implemented by the Secure Element. The Secure Element may be hardware-based, or virtual (software). If hardware-based, the implementation may be either with or without Secure Vault. Throughout this document, the following conventions will be used.

- SE - Hardware Secure Element, either with or without Secure Vault if not specified
- VSE - Virtual Secure Element
- Secure Element - Either SE or VSE

More steps are involved in the production programming process of Series 2 devices compared to Series 1 devices. The steps vary if the device is to have secure boot enabled or disabled. For more information about Secure Boot, see AN1218: Series 2 Secure Boot with RTSL. Enabling Secure Debug is an optional step in the process. For more information about Secure Debug, see AN1190: Series 2 Secure Debug.

A general overview of the production programming steps is described in the following sections. Although some steps can be performed using Simplicity Studio 5, this application note will focus more on using Simplicity Commander because it is more suitable for production environments.

3.1 Production Programming for Secure Boot-Disabled Device

The production programming flow for Secure Boot-disabled devices is illustrated in Figure 3.1 Series 2 High-Level Production Programming Flowchart for Secure Boot-Disabled Devices on page 6.

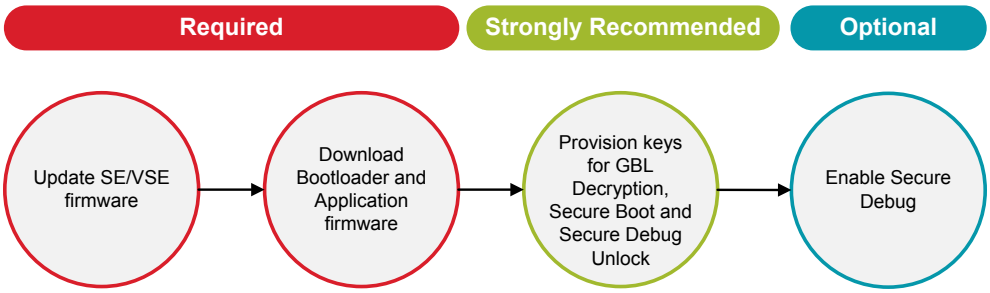

Figure 3.1. Series 2 High-Level Production Programming Flowchart for Secure Boot-Disabled Devices

Upgrading the Secure Element Firmware, and flashing the bootloader and application firmware are required steps in the production programming process. Provisioning the GBL Decryption Key for GBL payload decryption, Public Sign Key for Secure Boot, and Public Command Key for Secure Debug Unlock are strongly recommended. Enabling the Secure Debug feature is optional.

A more detailed version of the Series 2 production programming flowchart for a Secure Boot-disabled device is illustrated in Figure 3.2 Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Disabled Device on page 6.

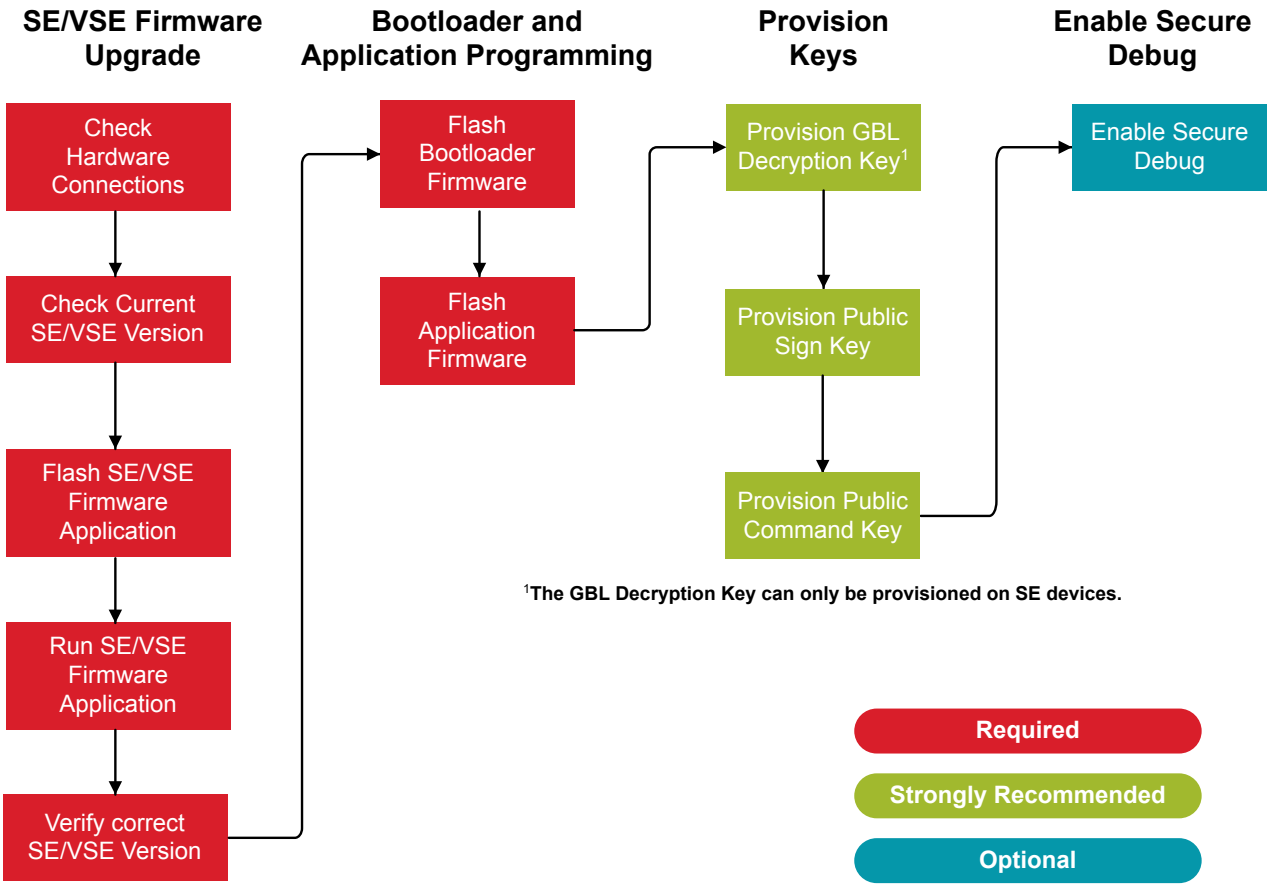

Figure 3.2. Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Disabled Device

3.2 Production Programming for Secure Boot-Enabled Devices

The production programming flow for Secure Boot-enabled devices is illustrated in Figure 3.3 Series 2 High-Level Production Programming Flowchart for Secure Boot-Enabled Devices on page 7.

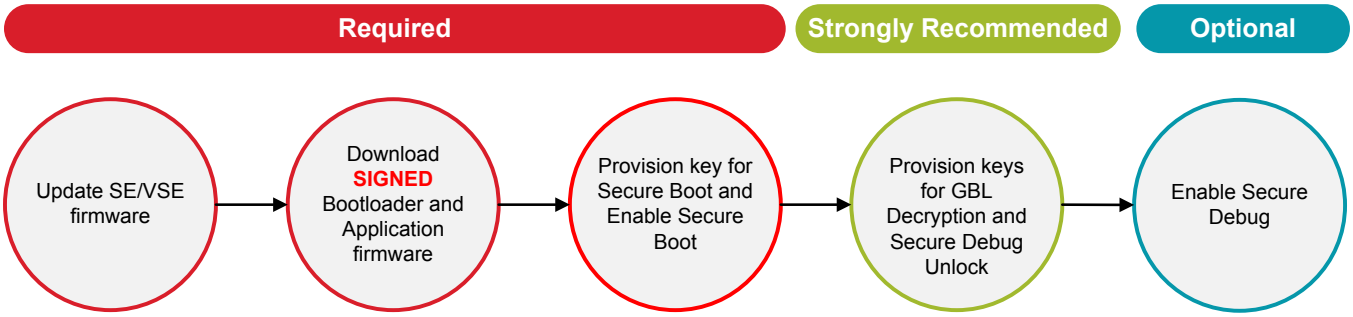

Figure 3.3. Series 2 High-Level Production Programming Flowchart for Secure Boot-Enabled Devices

Upgrading the Secure Element Firmware, and flashing the **SIGNED** bootloader and application firmware are required steps in the production programming process. Provisioning the Public Sign Key and enabling Secure Boot are also required in the production programming process to enable the Secure Boot option. Provisioning the GBL Decryption Key for GBL payload decryption and the Public Command Key for Secure Debug Unlock are strongly recommended. Enabling the Secure Debug feature is optional.

A more detailed version of the Series 2 production programming flowchart for a Secure Boot-enabled device is illustrated in Figure 3.4 Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Enabled Device on page 7.

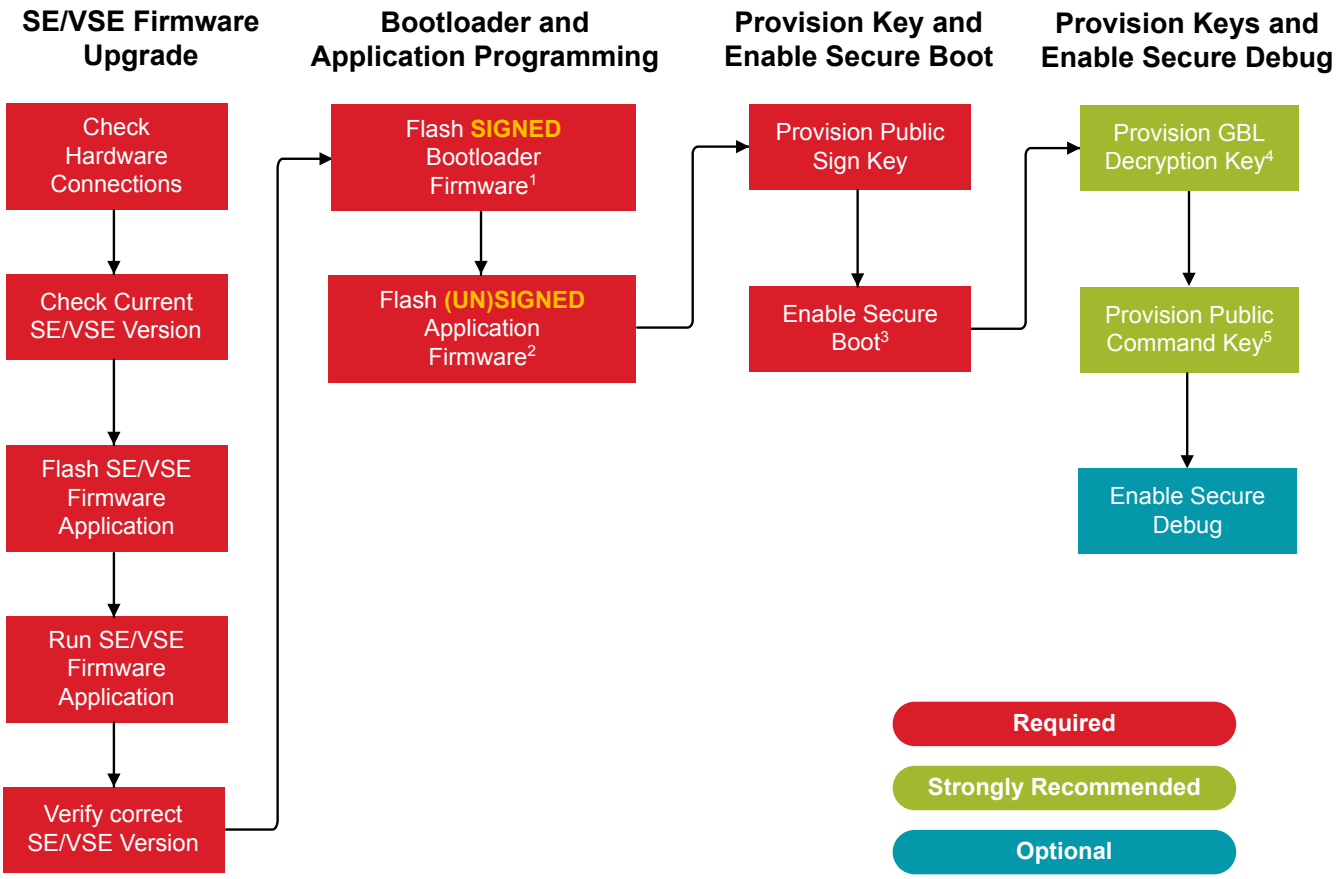

Figure 3.4. Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Enabled Device

Note:

1. The device will enter the Secure Boot failed state if the bootloader firmware is either unsigned or incorrectly signed (see 7. Bootloader Firmware Programming).
2. If the Secure Boot option is enabled in the bootloader, the application firmware must be signed (see 8. Application Firmware Programming).
3. On SE with Secure Vault devices, the anti-tamper protection configuration is provisioned with Secure Boot settings (see 10. Enabling Secure Boot).
4. The GBL Decryption Key can only be provisioned on SE devices.
5. The Public Command Key can also be used to temporarily disable anti-tamper protection on SE with Secure Vault devices (see AN1247: Anti-Tamper Protection Configuration and Use).

4. Using Simplicity Commander

1. Simplicity Commander's Command Line Interface (CLI) is invoked by `commander.exe` in the Simplicity Commander folder. The location in Windows is `C:\SiliconLabs\SimplicityStudio\<version>\developer\adapter_packs\commander`.
2. Simplicity Commander Version 1.9.2 is used in this application note.

```
commander --version
```

```
Simplicity Commander 1v9p2b791

JLink DLL version: 6.70a
Qt 5.5.1 Copyright (C) 2017 The Qt Company Ltd.
EMDLL Version: 0v17p12b535
mbed TLS version: 2.6.1

Emulator found with SN=440068705 USBAddr=0

DONE
```

3. The target Wireless Starter Kit (WSTK) must be specified using the `--serialno <J-Link serial number>` option if more than one WSTK is connected via USB.
4. The target device must be specified using the `--device <device name>` option if the WSTK is in debug mode OUT.
5. For more information about Simplicity Commander, see UG162: Simplicity Commander Reference Guide.

5. Secure Element Firmware Programming

5.1 Overview

Production programming of Series 2 devices is identical to production programming of Series 1 devices, with the addition of the Secure Element Firmware in the production programming process. Consistent with best practices for Internet of Things (IoT) security, the Secure Element Firmware provided with Series 2 devices supports secure firmware updates. Silicon Labs will periodically release new versions of the Secure Element Firmware to fix bugs and patch vulnerabilities, which may require updates to devices on the manufacturing line or to devices already in the field.

Silicon Labs operates under a "Security as a Shared Responsibility Model." This model provides flexibility to system integrators to manage Secure Element Firmware security updates on their own timetable based on their product's use case, risk assessment, agility of their manufacturing flow, and the agility of their field firmware deployment flow.

Series 2 devices will rarely ship with the latest Secure Element Firmware installed, meaning system integrators must add Secure Element Firmware programming to their production programming flow.

In all cases, Silicon Labs recommends that system integrators:

- Subscribe to security notifications by managing their notification settings in the Silicon Labs Support Portal. This is the easiest method to be notified of Secure Element Firmware updates and discovered vulnerabilities.
- Ensure they are installing the latest Secure Element Firmware release in their manufacturing line.
- Be prepared to deploy security-related field updates to devices in the field.

5.2 How to Check the Secure Element Firmware Version on a Device

The Secure Element Firmware version of the device can be found in two ways.

- Simplicity Studio
- Simplicity Commander

At the time of this writing, the latest Secure Element Firmware shipped with Series 2 devices and modules (if available) are listed in Table 5.1 Latest Secure Element Firmware Version on Shipped Series 2 Devices and Modules on page 10.

Table 5.1. Latest Secure Element Firmware Version on Shipped Series 2 Devices and Modules

Wireless SoC Series 2	Secure Element	Shipped Secure Element Firmware Version (Device and Module)
EFR32xG21A	SE without Secure Vault	Version 1.2.1
EFR32xG21B	SE with Secure Vault	Version 1.2.1
EFR32xG22	VSE	Version 1.2.1

5.2.1 Check the Secure Element Firmware Version Using Simplicity Studio 5

1. Change the Wireless Starter Kit (WSTK) **Debug Mode:** to **OUT**.
2. Right-click the selected debug adapter **Custom Board (ID:J-Link serial number)** to display the context menu.

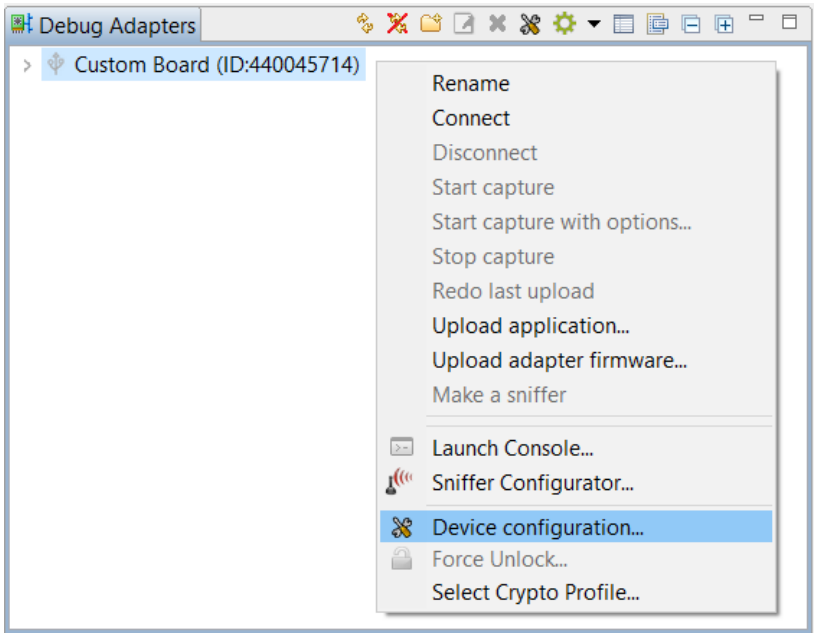

Figure 5.1. Context Menu of Debug Adapters

3. Click **Device configuration...** to open the **Configuration of device: J-Link Silicon Labs (serial number)** dialog box. Click the **Device hardware** tab to enter the part number in the **Target part:** box.

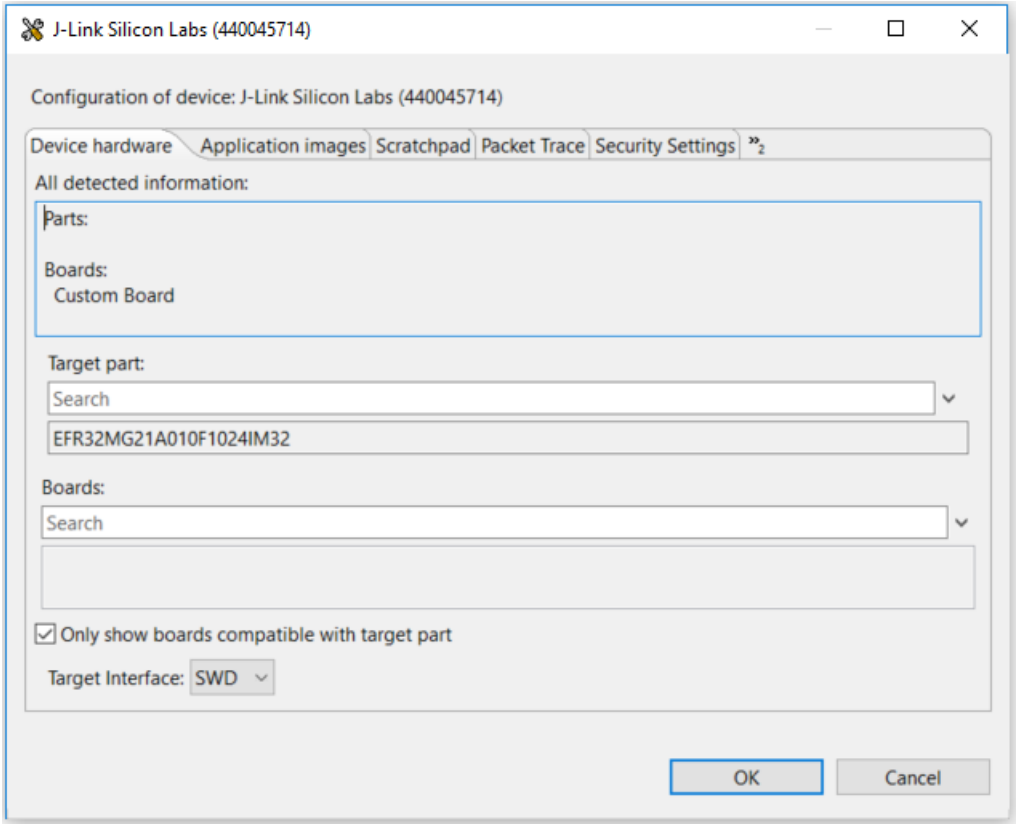

Figure 5.2. Configuration of Device

4. Click [**OK**] to exit.

5. Connect the device to the WSTK. Select the device in the **Debug Adapters** view.

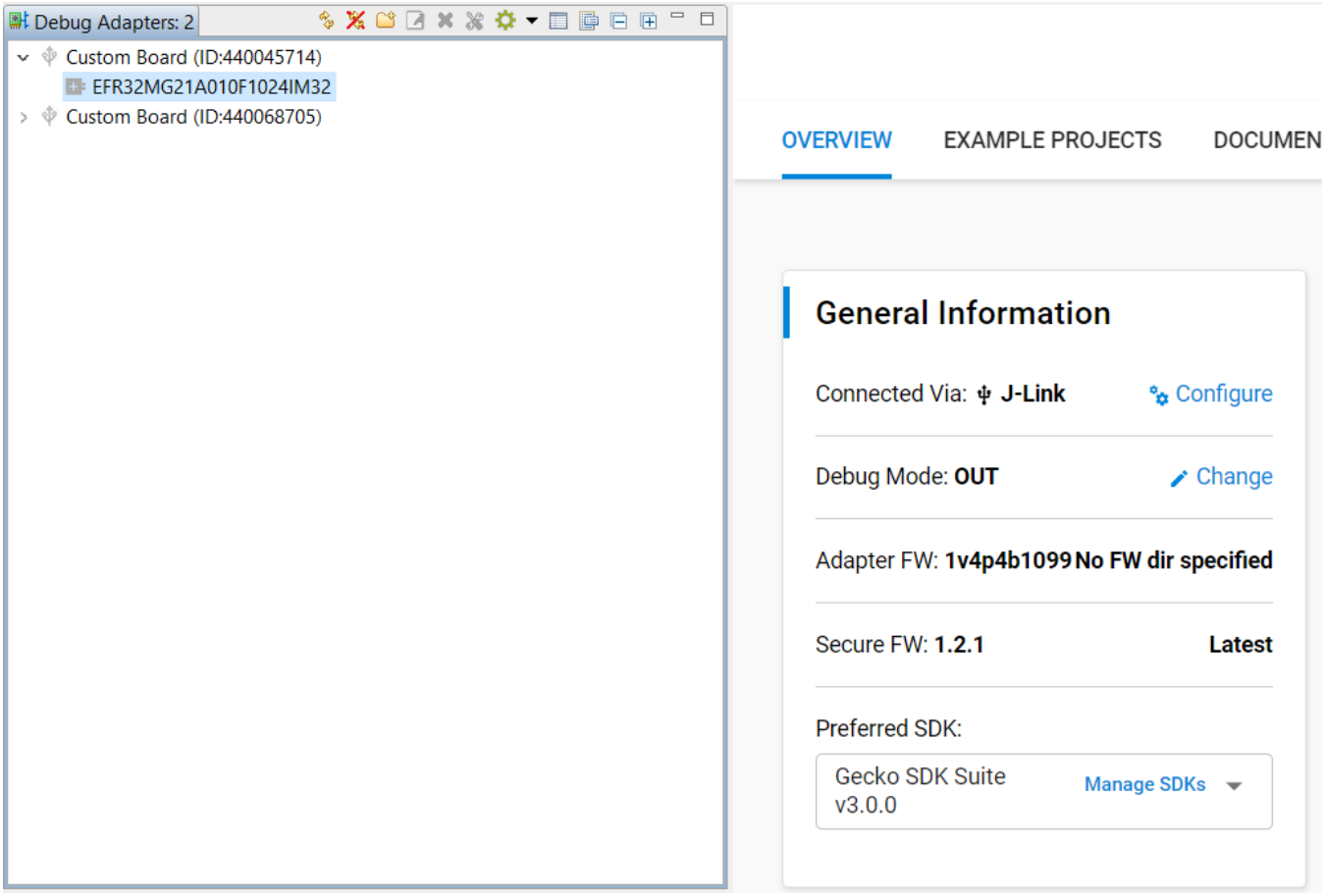

Figure 5.3. Secure Element Firmware Version in Simplicity Studio

6. The Secure Element Firmware version will appear in the **Secure FW:** line. In this example, the SE firmware version on the EFR32MG21A is 1.2.1.

5.2.2 Check the Secure Element Firmware Version Using Simplicity Commander

To check the Secure Element Firmware version on the device, you must issue the Simplicity Commander security command `security status`.

```
commander security status --device EFR32MG22C224F512 --serialno 440068705
```

```
SE Firmware version : 1.2.1
Serial number       : 000000000000000014b457fffed50d1e
Debug lock          : Disabled
Device erase        : Enabled
Secure debug unlock : Disabled
Secure boot         : Disabled
Boot status         : 0x20 - OK
DONE
```

In this example, the VSE firmware version on the EFR32MG22 is 1.2.1.

5.3 How to Find the Latest Secure Element Firmware

Instructions on retrieving the latest version of the Secure Element Firmware can be found in section "Gecko Bootloader Operation - Secure Element Upgrade" in UG266: Silicon Labs Gecko Bootloader User's Guide.

5.4 Serial Wire Debug (SWD)

The Secure Element Firmware cannot be directly programmed to the Secure Element using the SWD interface. Instead, a loader application that contains and will install the Secure Element Firmware is flashed onto the host MCU. The Secure Element Firmware is encrypted, versioned, and signed.

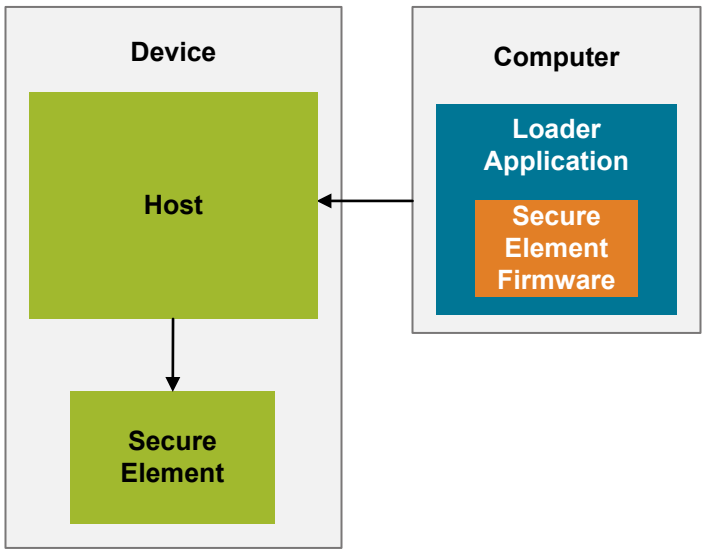

Figure 5.4. SWD Secure Element Firmware Upgrade Block Diagram

Using the SWD interface, the user flashes the loader application onto the host. The host then runs the loader application, which checks the signature and version of the Secure Element Firmware. If the signature check passes and the upgrade's version number is higher than the device's Secure Element Firmware version, the firmware is applied to the Secure Element. The upgrade will not be applied if the signature check fails or if the upgrade's version number is less than or equal to the device's Secure Element Firmware version. Trying to apply a lower Secure Element Firmware version to the device does no harm, but the upgrade will be ignored. This also means the device's Secure Element Firmware cannot be downgraded. After the Secure Element Firmware has been upgraded, the loader application can be deleted and the application firmware can be flashed via the SWD interface.

As detailed in Figure 3.2 Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Disabled Device on page 6 or Figure 3.4 Series 2 Step-by-Step Production Programming Flowchart for a Secure Boot-Enabled Device on page 7, the steps to upgrade the Secure Element Firmware are:

1. Connect Hardware: Connect the device's SWD interface with the WSTK and ensure proper connections
2. Check Version: Check the Secure Element Firmware version already on the device
3. Flash Secure Element Firmware: Flash the loader application onto the host processor.
4. Run: Allow the loader application to run and install the Secure Element Firmware.
5. Re-Check Version: Ensure the update succeeded

Each of these steps is described in more detail in the next sections.

5.4.1 Connect Hardware

After connecting the device's SWD interface to the WSTK, try to read the device information using Simplicity Commander, to verify that proper connections were established to the device.

```
commander device info --device EFR32MG21A010F1024 --serialno 440068705
```

```
Part Number    : EFR32MG21A010F1024
Die Revision   : A1
Production Ver : 0
Flash Size     : 1024 kB
SRAM Size      : 96 kB
Unique ID      : 000d6ffffead3d94
DONE
```

5.4.2 Check Version

To check the Secure Element Firmware version on the device, issue the Simplicity Commander security command `security status`.

```
commander security status --device EFR32MG21A010F1024 --serialno 440068705
```

```
SE Firmware version : 1.2.0
Serial number       : 000000000000000000d6ffffead3d94
Debug lock          : Disabled
Device erase        : Enabled
Secure debug unlock : Disabled
Tamper status       : OK
Secure boot         : Disabled
Boot status         : 0x20 - OK
DONE
```

Note: The `Tamper status` item is device-dependent.

5.4.3 Flash Secure Element Firmware

To flash the Secure Element Firmware upgrade application, run

```
commander flash --masserase s2c1_se_fw_upgrade_app_1v2p1.hex --device EFR32MG21A010F1024 --serialno 440068705
```

where `s2c1_se_fw_upgrade_app_1v2p1.hex` is replaced with the name of the Secure Element Firmware upgrade application file.

```
Parsing file s2c1_se_fw_upgrade_app_1v2p1.hex...
Erasing chip...
Flash was erased successfully
Writing 49152 bytes starting at address 0x00000000
Comparing range 0x00000000 - 0x0000BFFF (48 KB)
Programming range 0x00000000 - 0x00001FFF (8 KB)
Programming range 0x00002000 - 0x00003FFF (8 KB)
Programming range 0x00004000 - 0x00005FFF (8 KB)
Programming range 0x00006000 - 0x00007FFF (8 KB)
Programming range 0x00008000 - 0x00009FFF (8 KB)
Programming range 0x0000A000 - 0x0000BFFF (8 KB)
Verifying range 0x00000000 - 0x0000BFFF (48 KB)
DONE
```

5.4.4 Run

Allow the Secure Element Firmware upgrade application to run for at least two seconds. After two seconds, the Secure Element Firmware should have been upgraded.

5.4.5 Re-Check Version

Run the `security status` command again to check the upgraded Secure Element Firmware version.

```
commander security status --device EFR32MG21A010F1024 --serialno 440068705
```

```
SE Firmware version : 1.2.1
Serial number       : 000000000000000000d6ffffead3d94
Debug lock          : Disabled
Device erase        : Enabled
Secure debug unlock : Disabled
Tamper status       : OK
Secure boot         : Disabled
Boot status         : 0x20 - OK
DONE
```

The version is now upgraded to 1.2.1.

6. Field Upgrade the Secure Element Firmware on a Secure Boot-Enabled Device

If debug access was locked (standard or secure), the device must be unlocked before the Secure Element Firmware upgrade.

Instructions on how to unlock the device can be found in sections "Standard Debug Lock and Unlock" and "Secure Debug Unlock" of AN1190: Series 2 Secure Debug.

If Secure Boot was enabled, a **SIGNED** version of the Secure Element Firmware (`.seu` file) must be used for upgrade. Additional firmware signing information can be found in sections "ECDSA-P256-SHA256 Secure Boot" and "Certificate-Based Secure Boot" of AN1218: Series 2 Secure Boot with RTSL.

Details on Secure Element Firmware upgrade can be found in section "Gecko Bootloader Operation - Secure Element Upgrade" in UG266: Silicon Labs Gecko Bootloader User's Guide.

The Gecko Bootloader can parse the Secure Element Firmware GBL upgrade file and flash its content to the device.

ECDSA-P256-SHA256 Secure Boot Device:

To generate the Secure Element Firmware GBL upgrade file, run

```
commander gbl create seupgrade.gbl --seupgrade s2c1_se_fw_upgrade_1v2p1.seu --sign signing-key
```

where `s2c1_se_fw_upgrade_1v2p1.seu` is the Secure Element Firmware upgrade file and `signing-key` is the Private Sign Key for Secure Boot.

```
Initializing GBL file...
Adding Secure Element upgrade image to GBL...
Signing GBL...
Image SHA256: d09e4c9c9c2929f48e006b1ad1c7eadb7a3b60e724e873b837d2ccf7073cfff4
R = AD2F4563A015ABE8BD5EDA78BBBEB4AD94D6FA750892867BB3800E4049A63F21
S = FE905B92BEE2C84670ABD16B08623441DBE5ED9B6FD1BEB07858B944AE06EE90
Writing GBL file seupgrade.gbl...
DONE
```

Note: A copy of the Public Sign Key must be stored in the top page of the main flash (manufacturing token) for VSE firmware upgrade on an ECDSA-P256-SHA256 Secure Boot Device.

Certificate-based Secure Boot Device:

To generate the Secure Element Firmware GBL upgrade file, run

```
commander gbl create seupgrade.gbl --seupgrade s2c1_se_fw_upgrade_1v2p1.seu --sign bloadercert_key.pem
```

where `s2c1_se_fw_upgrade_1v2p1.seu` is the Secure Element Firmware upgrade file and `bloadercert_key.pem` is the Private Bootloader Key to sign the bootloader certificate.

```
Initializing GBL file...
Adding Secure Element upgrade image to GBL...
Signing GBL...
Image SHA256: d09e4c9c9c2929f48e006b1ad1c7eadb7a3b60e724e873b837d2ccf7073cfff4
R = 6B88E107E264AE702636C0DB7500BCB69ECE387E963C8C000D3E4CD758ECE4A4
S = 321FEF0E28447A03725590E468E44C3CCED0985E88AB84463A4FC94FF4FA0494
Writing GBL file seupgrade.gbl...
DONE
```

7. Bootloader Firmware Programming

If Secure Boot is enabled, a **SIGNED** version of the bootloader firmware must be programmed to the flash.

Instructions on how to sign the bootloader firmware can be found in sections "ECDSA-P256-SHA256 Secure Boot" and "Certificate-Based Secure Boot" of AN1218: Series 2 Secure Boot with RTSL.

For Series 2 devices, the bootloader is placed at address `0x00000000`. Silicon Labs recommends reserving 16 kB for the bootloader for SE devices and 24 kB for VSE devices.

Flashing the bootloader firmware using Simplicity Commander is similar to flashing the Secure Element Firmware upgrade application.

```
commander flash --masserase <bootloader file> --device <device name> --serialno <J-Link serial number>
```

where `<bootloader file>` is the name of the bootloader firmware file.

To check the Boot status of the device, run the `security status` command.

```
commander security status --device EFR32MG21A010F1024 --serialno 440068705
```

```
SE Firmware version : 1.2.1
Serial number       : 000000000000000000d6ffffead3d94
Debug lock          : Disabled
Device erase        : Enabled
Secure debug unlock : Disabled
Tamper status       : OK
Secure boot         : Enabled
Boot status         : 0x20 - OK
DONE
```

The Secure Boot process fails if the Boot status is not `0x20` (for example, `0x15` - Failed: Error while validating host application properties structure). It means the bootloader firmware is either unsigned or incorrectly signed and the only way to recover is to flash a correctly-signed image (see section "Recover Devices when Secure Boot Fails" in AN1218: Series 2 Secure Boot with RTSL).

8. Application Firmware Programming

The Secure Element Firmware upgrade via the SWD interface or a field update overwrites application firmware previously written to the flash. The application firmware must be programmed after the Secure Element Firmware upgrade.

If the Secure Boot option is enabled in the bootloader, a **SIGNED** version of the application firmware must be programmed to the flash.

Instructions on how to sign the application firmware can be found in sections "ECDSA-P256-SHA256 Secure Boot" and "Certificate-Based Secure Boot" of AN1218: Series 2 Secure Boot with RTSL.

For Series 2 devices, the application firmware is usually placed at address `0x00004000` for SE devices and `0x00006000` for VSE devices.

Flashing the application firmware using Simplicity Commander is similar to flashing the Secure Element Firmware upgrade application.

```
commander flash <application file> --device <device name> --serialno <J-Link serial number>
```

where `<application file>` is the name of the application firmware file.

Note: Do not use the --masserase option to flash the application firmware as it will erase the bootloader at `0x00000000`.

9. Key Provisioning

9.1 Overview

The symmetric GBL Decryption Key is used to decrypt GBL files. All encrypted images on this device must be encrypted with the same 128-bit AES key.

If the Secure Boot feature is to be used, the Public Sign Key must first be provisioned to the device.

If the Secure Debug feature is to be used, the Public Command Key must first be provisioned to the device.

The GBL Decryption Key, Public Sign Key and the Public Command Key are written to one-time-programmable (OTP) memory. Once written, they cannot be changed.

Silicon Labs strongly recommends provisioning these keys even if the Secure Boot and/or Secure Debug features are not used, for future-proofing.

Note: The GBL Decryption Key can only be provisioned on SE devices.

9.2 Provisioning the GBL Decryption Key in Simplicity Commander

To write the GBL Decryption Key to the device, run the command

```
commander security writekey --decrypt aes-key.txt --device EFR32MG21A010F1024 --serialno 440068705
```

where `aes-key.txt` is the GBL Decryption Key. This command can be executed only once per device.

```
Device has serial number 000000000000000000d6ffffead3d94

================================================================================
Please look through any warnings before proceeding.
THIS IS A ONE-TIME command, all code to be run on the device must be signed by this key.
Type 'continue' and hit enter to proceed or Ctrl-C to abort:
================================================================================
continue
DONE
```

Note: The GBL Decryption Key cannot be read back from the device.

9.3 Provisioning the Public Sign Key in Simplicity Commander

To write the Public Sign Key to the device, run the command

```
commander security writekey --sign signing-key.pub --device EFR32MG21A010F1024 --serialno 440068705
```

where `signing-key.pub` is the Public Sign Key. This command can be executed only once per device.

```
Device has serial number 000000000000000000d6ffffead3d94

================================================================================
Please look through any warnings before proceeding.
THIS IS A ONE-TIME command, all code to be run on the device must be signed by this key.
Type 'continue' and hit enter to proceed or Ctrl-C to abort:
================================================================================
continue
DONE
```

To read the Public Sign Key on the device, run the command

```
commander security readkey --sign --device EFR32MG21A010F1024 --serialno 440068705
```

```
C4AF4AC69AAB9512DB50F7A26AE5B4801183D85417E729A56DA974F4E08A562C
DE6019DEA9411332DC1A743372D170B436238A34597C410EA177024DE20FC819
DONE
```

9.4 Provisioning the Public Command Key in Simplicity Commander

To write the Public Command Key to the device, run the command

```
commander security writekey --command command-key.pub --device EFR32MG21A010F1024 --serialno 440068705
```

where `command-key.pub` is the Public Command Key. This command can be executed only once per device.

```
Device has serial number 000000000000000000d6ffffead3d94

================================================================================
Please look through any warnings before proceeding.
THIS IS A ONE-TIME command which permanently ties debug and tamper access to certificates signed by this key.
Type 'continue' and hit enter to proceed or Ctrl-C to abort:
================================================================================
continue
DONE
```

To read the Public Command Key on the device, run the command

```
commander security readkey --command --device EFR32MG21A010F1024 --serialno 440068705
```

```
B1BC6F6FA56640ED522B2EE0F5B3CF7E5D48F60BE8148F0DC08440F0A4E1DCA4
7C04119ED6A1BE31B7707E5F9D001A659A051003E95E1B936F05C37EA793AD63
DONE
```

10. Enabling Secure Boot

The Secure Boot feature verifies the integrity and authenticity of the host application before allowing it to execute. Enabling this feature is **IRREVERSIBLE**, which means once enabled, Secure Boot can no longer be disabled throughout the life of the device. The Secure Boot settings are written to a one-time-programmable (OTP) memory. Once written, they cannot be changed.

On SE with Secure Vault devices, the anti-tamper configuration is provisioned with Secure Boot settings. The anti-tamper configuration determines the response from the SE with Secure Vault if a tamper event occurs.

Note:

- All tamper-related information in the following sections is only valid on SE with Secure Vault devices.
- For more information about anti-tamper configuration, see AN1247: Anti-Tamper Protection Configuration and Use.

The following command writes the Secure Boot settings and anti-tamper configuration to the device.

```
commander security writeconfig --configfile user_configuration.json --device EFR32MG21A010F1024 --serialno
440068705
```

```
================================================================================

THIS IS A ONE-TIME configuration: Please inspect file before confirming:
user_configuration.json
Type 'continue' and hit enter to proceed or Ctrl-C to abort:
================================================================================
continue
DONE
```

This command can be executed only once per device. The `user_configuration.json` is a JSON file that contains the desired Secure Boot settings and anti-tamper configuration. You can generate a default configuration file by using the following command.

```
commander security genconfig -o user_configuration.json --serialno 440068705
```

```
Configuration file stored in:
C:/Users/amleung/AppData/Local/SiliconLabs/commander/SecurityStore/device_000000000000000000d6ffffead3d94/
user_configuration.json
DONE
```

To check the device's Secure Boot settings and anti-tamper configuration, run the `security readconfig` command.

```
commander security readconfig --serialno 440068705
```

```
MCU Flags
Secure Boot                    : Enabled
Secure Boot Verify Certificate : Disabled
Secure Boot Anti Rollback      : Enabled
Secure Boot Page Lock Narrow   : Disabled
Secure Boot Page Lock Full     : Enabled

Tamper Levels
FILTER_COUNTER     : 0
WATCHDOG           : 4
SE_RAM_CRC         : 4
SE_HARDFAULT       : 4
SOFTWARE_ASSERTION : 4
SE_CODE_AUTH       : 4
USER_CODE_AUTH     : 4
MAILBOX_AUTH       : 0
DCI_AUTH           : 0
OTP_READ           : 0
AUTO_CODE_AUTH     : 0
SELF_TEST          : 4
TRNG_MONITOR       : 0
PRS0               : 0
PRS1               : 0
PRS2               : 0
PRS3               : 0
PRS4               : 0
PRS5               : 0
PRS6               : 0
PRS7               : 0
DECOUPLE_BOD       : 4
TEMP_SENSOR        : 1
VGLITCH_FALLING    : 0
VGLITCH_RISING     : 0
SECURE_LOCK        : 4
SE_DEBUG           : 0
DGLITCH            : 0
SE_ICACHE          : 4

Tamper Filter
Filter Period   : 0
Filter Treshold : 0
Reset Treshold  : 0

Tamper Flags
Digital Glitch Detector Always On: Disabled
DONE
```

The `security genconfig` command generates a generic configuration file consisting of the properties listed in Table 10.1 Configuration File Description on page 23. A text editor can be used to modify the default settings shown below to the desired configuration.

```
{
   "mcu_flags": {
       "SECURE_BOOT_ENABLE": true,
       "SECURE_BOOT_VERIFY_CERTIFICATE": false,
       "SECURE_BOOT_ANTI_ROLLBACK": true,
       "SECURE_BOOT_PAGE_LOCK_NARROW": false,
       "SECURE_BOOT_PAGE_LOCK_FULL": true
   },
   "tamper_levels": {
        "FILTER_COUNTER": 0,
        "WATCHDOG": 4,
        "SE_RAM_CRC": 4,
        "SE_HARDFAULT": 4,
        "SOFTWARE_ASSERTION": 4,
        "SE_CODE_AUTH": 4,
        "USER_CODE_AUTH": 4,
        "MAILBOX_AUTH": 0,
        "DCI_AUTH": 0,
        "OTP_READ": 0,
        "AUTO_CODE_AUTH": 0,
        "SELF_TEST": 4,
        "TRNG_MONITOR": 0,
        "PRS0": 0,
        "PRS1": 0,
        "PRS2": 0,
        "PRS3": 0,
        "PRS4": 0,
        "PRS5": 0,
        "PRS6": 0,
        "PRS7": 0,
        "DECOUPLE_BOD": 4,
        "TEMP_SENSOR": 1,
        "VGLITCH_FALLING": 0,
        "VGLITCH_RISING": 0,
        "SECURE_LOCK": 4,
        "SE_DEBUG": 0,
        "DGLITCH": 0,
        "SE_ICACHE": 4
   },
   "tamper_filter" : {
        "FILTER_PERIOD": 0,
        "FILTER_THRESHOLD": 0,
        "RESET_THRESHOLD": 0
   },
   "tamper_flags": {
        "DGLITCH_ALWAYS_ON": false
   }
}
```

Table 10.1. Configuration File Description

Name	Description
For SE and VSE devices (mcu_flags):	
SECURE_BOOT_ENABLE	If set, verifies the image on the Cortex-M33 before releasing the Cortex-M33 from reset.
SECURE_BOOT_VERIFY_CERTIFICATE	If set, requires certificate-based signing of the host application.
SECURE_BOOT_ANTI_ROLLBACK	If set, prevents secure upgrading to a host image with a lower version than the image that is currently stored in flash.
SECURE_BOOT_PAGE_LOCK_NARROW	If set, locks flash pages that have been validated by the Secure Boot process to prevent re-flashing by other means than through the Secure Element. Write/erase locks pages from 0 through the page where the Secure Boot signature of the application is located, **not including** the last page if the signature is not on a page boundary.
SECURE_BOOT_PAGE_LOCK_FULL	If set, locks flash pages that have been validated by the Secure Boot process to prevent re-flashing by other means than through the Secure Element. Write/erase locks pages from 0 through the page where the Secure Boot signature of the application is located, **including** the last page if the signature is not on a page boundary.
For SE with Secure Vault devices:	
tamper_levels	The tamper levels of different tamper sources.
tamper_filter	The settings for tamper filters.
tamper_flags	The settings for tamper flags.

11. Enabling Secure Debug

The Secure Debug feature is enabled through the `Secure Debug Unlock` command. After locking the device, the `Secure Debug Unlock` command securely unlocks the device for debugging until the next device reset without erasing flash and RAM contents.

The following command enables the secure debug unlock.

```
commander security lockconfig --secure-debug-unlock enable --device EFR32MG22C224F512 --serialno 440068705
```

```
Secure debug unlock was enabled
DONE
```

After enabling the Secure Debug feature, you can lock the device by using the following command.

```
commander security lock --device EFR32MG22C224F512 --serialno 440068705
```

```
Device is now locked.
DONE
```

After locking the device, you can disable the device erase by using the following command.

```
commander security disabledeviceerase --device EFR32MG22C224F512 --serialno 440068705
```

```
================================================================================
THIS IS A ONE-TIME command which Permanently disables device erase.
If secure debug lock has not been set, there is no way to regain debug access to this device.
Type 'continue' and hit enter to proceed or Ctrl-C to abort:
================================================================================
continue
Disabled device erase successfully
DONE
```

Note: This is an **IRREVERSIBLE** action and should be the last step in production.

To check the debug lock status of the device, run the `security status` command

```
commander security status --device EFR32MG22C224F512 --serialno 440068705
```

```
SE Firmware version : 1.2.1
Serial number       : 000000000000000014b457fffed50d1e
Debug lock          : Enabled
Device erase        : Disabled
Secure debug unlock : Enabled
Secure boot         : Disabled
Boot status         : 0x20 - OK
DONE
```

12. Related Documents

- AN136: Silicon Labs Production Programming Options
- AN958: Debugging and Programming Interfaces for Custom Designs
- UG162: Simplicity Commander Reference Guide
- UG266: Silicon Labs Gecko Bootloader User's Guide
- AN1190: Series 2 Secure Debug
- AN1218: Series 2 Secure Boot with RTSL
- AN1247: Anti-Tamper Protection Configuration and Use
- AN1271: Secure Key Storage

13. Revision History

Revision 0.4

October 2020

- Removed a duplicate paragraph from 1.1 User Assistance

Revision 0.3

September 2020

- Added EFR32BG21B and EFR32MG21B to 2. Device Compatibility.
- Added Secure Element conventions to 3. Overview, updated the figures and content.
- Updated Simplicity Commander version to 1.9.2 in 4. Using Simplicity Commander.
- Added EFRxG21B to 5.2 How to Check the Secure Element Firmware Version on a Device.
- Updated the figures in 5.2.1 Check the Secure Element Firmware Version Using Simplicity Studio 5 to Simplicity Studio v5.
- Renamed Field Upgrade to 6. Field Upgrade the Secure Element Firmware on a Secure Boot-Enabled Device, updated the content and moved up two levels.
- Removed Over-The-Air (OTA) section.
- Added 7. Bootloader Firmware Programming.
- Updated 8. Application Firmware Programming.
- Updated 10. Enabling Secure Boot for SE with Secure Vault devices.
- Added 9.2 Provisioning the GBL Decryption Key in Simplicity Commander.
- Added AN1247 and AN1271 to 12. Related Documents.

Revision 0.2

March 2020

- Added EFR32xG22 devices to Device Compatibility section.
- Added Simplicity Commander section.
- Updated figures in Check Secure Element Firmware Version Using Simplicity Studio
- Modified Check Secure Element Firmware Version Using Simplicity Studio section.
- Added Note to Check Version section.
- Added Field Upgrade to Serial Wire Debug (SWD) section.
- Added disable device erase procedure to Secure Debug Enabling section.
- Added UG162 to Related Documents section.
- Changed all Simplicity Commander outputs to text, easy to update in the future.

Revision 0.1

December 2019

- Initial Revision.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com