AN1247: Anti-Tamper Protection Configuration and Use

This version of AN1247 has been deprecated with the release of Simplicity SDK Suite 2025.6.1.

For the latest version, see docs.silabs.com.

****

This application note describes how to program, provision, and configure the Secure Engine anti-tamper module. Many aspects of the anti-tamper module, including disabling the anti-tamper response when needed, are discussed.

The anti-tamper module is only available on Secure Vault High devices. The external tamper detect module is available on some Secure Vault Mid devices (e.g. xG27) and Secure Vault High devices (e.g. xG25B).

KEY POINTS

- Tamper responses
- Tamper sources
- Tamper configuration
- Tamper disable
- Examples of provisioning and disabling the anti-tamper module

1. 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 Series 2 products that included a Secure Engine. The Secure Engine is a tamper-resistant component used to securely store sensitive data and keys and to execute cryptographic functions and secure services.

On Series 1 devices, the security features are implemented by the TRNG (if available) and CRYPTO peripherals.

On Series 2 devices, the security features are implemented by the Secure Engine and CRYPTOACC (if available). The Secure Engine may be hardware-based, or virtual (software-based). Throughout this document, the following abbreviations are used:

- HSE - Hardware Secure Engine
- VSE - Virtual Secure Engine
- SE - Secure Engine (either HSE or VSE)

Additional security features are provided by Secure Vault. Three levels of Secure Vault feature support are available, depending on the part and SE implementation, as reflected in the following table:

Level (1)	SE Support	Part (2)
Secure Vault High (SVH)	HSE only (HSE-SVH)	Refer to UG103.05 for details on supporting devices.
Secure Vault Mid (SVM)	HSE (HSE-SVM)	"
"	VSE (VSE-SVM)	"
Secure Vault Base (SVB)	N/A	"

Note:

1. The features of different Secure Vault levels can be found in https://www.silabs.com/security.
2. UG103.05.

Secure Vault Mid consists 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 for execution.
- 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.

Secure Vault High offers additional security options:

- Secure Key Storage: Protects cryptographic keys by "wrapping" or encrypting the keys using a root key known only to the HSE-SVH.
- 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 Engine 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 Series 2 debug access, including background information about the SE	Secure Vault Mid and High
AN1218: Series 2 Secure Boot with RTSL	Describes the secure boot process on Series 2 devices using SE	Secure Vault Mid and High
AN1222: Production Programming of Series 2 Devices	How to program, provision, and configure security information using SE during device production	Secure Vault Mid and High
AN1247: Anti-Tamper Protection Configuration and Use (this document)	How to program, provision, and configure the anti-tamper module	Secure Vault High
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	Secure Vault High
AN1271: Secure Key Storage	How to securely “wrap” keys so they can be stored in non-volatile storage.	Secure Vault High

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	Customer Programmed	Purpose	Used in
Public Sign key (Sign Key Public)	Yes	Secure Boot binary authentication and/or OTA upgrade payload authentication	AN1218 (primary), AN1222
Public Command key (Command Key Public)	Yes	Secure Debug Unlock or Disable Tamper command authentication	AN1190 (primary), AN1222, AN1247
OTA Decryption key (GBL Decryption key) aka AES-128 Key	Yes	Decrypting GBL payloads used for firmware upgrades	AN1222 (primary), UG266/UG489
Attestation key aka Private Device Key	No	Device authentication for secure identity	AN1268

1.3 SE Firmware

Silicon Labs strongly recommends installing the latest SE firmware on Series 2 devices to support the required security features. Refer to AN1222 for the procedure to upgrade the SE firmware and UG103.05 for the latest SE Firmware shipped with Series 2 devices and modules.

2. Introduction

The HSE-SVH Anti-Tamper module is used to hamper or prevent both reverse engineering and re-engineering of proprietary software systems or applications.

Tamper attacks come from one or more vectors. Common attacks include voltage glitching, magnetic interference, and forced temperature adjustment. The HSE-SVH Anti-Tamper module provides fast hardware detection of external tamper signals such as case opening, glitching, and logical attacks allowing analysis and escalation up to and including bricking the device.

The anti-tamper module connects a number of hardware and software-driven tamper signals to a set of configurable hardware and software responses. This can be used to program the device to automatically respond to external events that could signal that someone is trying to tamper with the device, and very rapidly remove secrets stored in the HSE.

The available tamper signals range from signals based on failed authentication and secure boot to specialized glitch detectors. When any of these signals fire, the tamper block can be configured to trigger several different responses, ranging from triggering an interrupt to erasing the One-Time-Programmable (OTP) memory, removing all HSE secrets and resulting in a permanently destroyed device.

Silicon Labs provides Custom Part Manufacturing Service (CPMS) to protect the users' privacy by configuring the most effective tamper detection features at the Silicon Labs factory. For more information about CPMS, see UG519: Custom Part Manufacturing Service User's Guide.

Some SVM devices (e.g. xG25A and xG27) and SVH devices (e.g. xG25B) feature an External Tamper Detect module which is used to detect signals such as case opening. The ETAMPDET signal on SVH devices is routed to the SE as an Anti-Tamper module tamper source, in addition to being a stand-alone module. For more information about ETAMPDET operation, refer to the device reference manual. Examples demonstrating how to use ETAMPDET can be found on the Silicon Labs peripheral_example github repository.

3. Secure Engine Manager

The Secure Engine Manager provides thread-safe APIs for the SE's mailbox interface. The SE Manager APIs related to tamper operations are listed in the following table.

For the SE's mailbox interface, see section "*Secure Engine Subsystem*" in AN1190: Series 2 Secure Debug.

Table 3.1. SE Manager API for Tamper Operations

SE Manager API	Usage
sl_se_init_otp	Initialize SE OTP configuration (including tamper configuration on HSE-SVH devices).
sl_se_read_otp	Read SE OTP configuration (including tamper configuration on HSE-SVH devices).
sl_se_init_otp_key	Used during device initialization to upload the Public Command Key.
sl_se_read_pubkey	Read the stored Public Command Key.
sl_se_get_serialnumber	Read out the serial number (16 bytes) of the HSE device.
sl_se_get_challenge	Read out the current challenge value (16 bytes) for tamper disable.
sl_se_roll_challenge	Used to roll the current challenge value (16 bytes) to invalidate the Disable Tamper Token.
sl_se_disable_tamper	Temporarily disable tamper configuration using the Disable Tamper Token.
sl_se_get_status	Read the current HSE status (including recorded tamper status on HSE-SVH devices).
sl_se_get_reset_cause	Read the EMU->RSTCAUSE register from HSE devices after a tamper reset.
sl_se_get_tamper_reset_cause	Read the cached value of the EMU->TAMPERRSTCAUSE register after a tamper reset.
sl_se_enter_active_mode	Force the SE to remain active to enable the detection of glitch tamper events on the host Cortex-M33 core (see fourth note below)
sl_se_exit_active_mode	Exit active mode and allow the SE to sleep when not performing operations. This will prevent the detection of glitch tamper events when the SE is sleeping. This API should only be used if active mode was entered by calling `sl_se_enter_active_mode`. If active mode is set through a DCI command, it can only be disabled through a DCI command. (see fourth note below)

Note:

- The `sl_se_get_reset_cause` is only available on EFR32xG21B devices. The `EMU->RSTCAUSE` register can be directly read on other HSE-SVH devices.
- The `sl_se_get_tamper_reset_cause` is unavailable on EFR32xG21B devices, and SE firmware ≥ v2.2.1 is required.
- The SE Manager API document can be found at https://docs.silabs.com/gecko-platform/latest/service/api/group-sl-se-manager.
- Does not apply to EFR32MG21B parts.

4. Tamper Responses

A tamper source can lead to a series of different autonomous responses from the HSE. These responses are listed in the following table.

Table 4.1. Tamper Responses

Level (1)	Response (2)	Description
0	Ignore	No action is taken
1	Interrupt	Triggers the SETAMPERHOST interrupt on the host
2	Filter	Increases a counter in the tamper filter
4	Reset	Resets the device
7	Erase OTP	Erases the device's OTP configuration

Note:

1. Level 3, 5, and 6 are reserved.
2. These responses are cumulative:
   - If a filter response is triggered, it will also trigger an interrupt.
   - If a reset response is triggered, it will supersede the interrupt. The filter counter and interrupt flag are clear at reset.
   - If an erase OTP response is triggered, it will erase the OTP and reset the device. The device will fail to boot and become unusable.

4.1 Interrupt

If a tamper source is configured to respond with the interrupt response or higher (≥ level 1), the `SETAMPERHOST` interrupt line to the host Cortex-M33 will be pulsed and make the NVIC trigger the corresponding interrupt handler (`SETAMPERHOST_IRQHandler`).

After the interrupt has been handled, the tamper status can be found by reading the HSE status (using `sl_se_get_status` in the SE Manager), which contains a list of all the tamper sources that have been triggered since the last time the status was read. Reading HSE status clears the registered tamper sources.

Note: Enabling the `SEMAILBOXHOST` clock for the tamper source is required to trigger the `SETAMPERHOST` interrupt in most HSE-SVH devices. EFR32xG21B does not require this.

4.2 Filter

The HSE has a filter to debounce spurious tamper events. The filter has a counter that is periodically reset. If a tamper source is configured to the filter response (level 2), when it is triggered, the counter is increased. If the counter value reaches a configurable threshold, the `Filter counter` tamper source (number 1) is triggered, which can configure to lead to any other responses (1, 4, or even 7).

Only a single shared filter counter is available, so the cumulative triggering of all tamper sources configured to the filter level will increase the same counter. The filter can be programmed to use one of the trigger thresholds and reset periods provided below. The filter counter is zero upon a tamper or normal reset.

Filter Trigger Threshold

- Value (n): 0 to 7
- Filter Trigger Threshold: $256/2^n$ (256 to 2)

Filter Reset Period

- Value (n): 0 to 31
- Filter Reset Period: 32 ms * 2^n (32 ms to ~795.4 days)

Example Filter Configuration

For example, consider a device with a Filter Trigger Threshold of 3 and Filter Reset Period of 5. If that device detects 32 ($256/2^3$) Filter response events in 1.024 seconds (32 ms * 2^5), the `Filter counter` tamper source (number 1) will trigger.

4.3 Reset

The reset response resets the HSE and Cortex-M33. After a tamper reset, the last reset cause can be directly read from `EMU->RSTCAUSE` register or using `sl_se_get_rstcause` in the SE Manager. In cases where the reset was caused by a tamper response, the source of the tamper can be determined by calling `sl_se_get_tamper_reset_cause` in the SE Manager. (Note that this API is not available for EFR32xG21B-based parts). See Table 5.2 for the list of tamper sources. Tamper reset occurs when the HSE sends a request to the Cortex-M33's EMU, which issues a hard reset.

If a tamper reset is triggered during boot, this can lead to a boot loop. To debug such a scenario, the HSE has a tamper reset counter and enters diagnostic mode if the counter reaches a programmable threshold. Users can issue a non-tamper reset to clear the tamper reset counter before the programmable threshold is reached.

In diagnostic mode, the Cortex-M33 is held in reset and only DCI commands are available. The device will remain in diagnostic mode until a power-on or pin reset occurs.

For more information on the SE's DCI, see section "*Secure Engine Subsystem*" in AN1190: Series 2 Secure Debug.

4.4 Erase OTP

The Erase OTP response is the strongest reaction the HSE can take, and it will make the device and all wrapped secrets unrecoverable. After this response, the device will no longer be able to boot or connect to a debugger.

This response should typically only be used in situations where the device believes that it is under an actual attack, for instance through the detection of several voltage or digital glitches in a short time window.

5. Tamper Sources

The following tables list the available tamper sources and the default level on the EFR32xG21B and other HSE-SVH devices. The tamper sources with the default level higher than 0 (`Ignore`) are always in effect even if the user does not initialize the tamper configuration in HSE OTP. Users can keep or escalate the default tamper responses (≥ 0 for `Ignore` and ≥ 4 for `Reset`) of any sources when initially configuring the part.

Table 5.1. Tamper Sources on the EFR32xG21B Devices

Type	Number	Name	Description	Default Level
SE Hardware	0	Reserved	—	—
"	1	Filter counter	Filter counter reached the configured threshold value	0 (Ignore)
"	2	SE watchdog	Internal SE watchdog expired	4 (Reset)
"	3	Reserved	—	—
"	4	SE RAM CRC	A 2-bit, non-correctable error in the SE RAM has occurred.	4 (Reset)
"	5	SE hard fault	The SE core has encountered a hard fault exception indicating that an invalid memory access was attempted.	4 (Reset)
"	6	Reserved	—	—
SE Software	7	SE software assertion	SE firmware has triggered an assertion, indicating that one of several sanity checks has failed and that normal operation cannot continue without a reset.	4 (Reset)
"	8	SE secure boot	Secure boot of SE firmware failed	4 (Reset)
"	9	User secure boot	Secure boot of host firmware failed	0 (Ignore)
"	10	Mailbox authorization	Unauthorized command received over the Mailbox interface. This can be triggered by either (1) an incorrectly signed debug unlock or tamper disable token or (2) attempting to export a non-exportable key.	0 (Ignore)
"	11	DCI authorization	Unauthorized command received over the DCI interface. This can be triggered by either (1) an incorrectly signed debug unlock or tamper disable token or (2) attempting to export a non-exportable key.	0 (Ignore)
"	12	OTP read	OTP or flash content could not be properly authenticated.	4 (Reset)
"	13	Reserved	—	—
"	14	Self test	A check of the integrity of the SE's internal storage failed during boot up.	4 (Reset)
"	15	TRNG monitor	The TRNG monitor performs a number of tests on the collected entropy data. If any of these tests fail, this tamper source is triggered.	0 (Ignore)
Hardware	16 - 23	PRS0 - 7 [1]	PRS consumer for SE Tamper 0 - 7 asserted	0 (Ignore)
"	24	Decouple BOD [1]	Decouple Brown-Out-Detector threshold alert	4 (Reset)
"	25	Temperature sensor [1]	SE temperature is monitored to be within 5 degrees C of the limits for the device. If the limit is exceeded, this tamper source will be triggered.	0 (Ignore)
"	26	Voltage glitch falling	Voltage glitch detector detected a falling glitch	0 (Ignore)
"	27	Voltage glitch rising	Voltage glitch detector detected a rising glitch	0 (Ignore)
"	28	Secure lock	This tamper source indicates that the guarding mechanism (comparing the locks with their logical complement) of the debug port locks has failed.	4 (Reset)

Type	Number	Name	Description	Default Level
"	29	SE debug	Debug access to SE	0 (Ignore)
"	30	Digital glitch	Digital glitch detector detected an event	0 (Ignore)
"	31	SE ICACHE	The SE's instruction cache uses a checksum to verify the integrity of the data. This tamper source is triggered if the checksum is invalid.	4 (Reset)

Table 5.2. Tamper Sources on Other HSE-SVH Devices

Type	Number	Name	Description	Default Level
SE Hardware	0	Reserved	—	—
"	1	Filter counter	Filter counter reached the configured threshold value	0 (Ignore)
"	2	SE watchdog	Internal SE watchdog expired	4 (Reset)
"	3	Reserved	—	—
"	4	SE RAM ECC2	A 2-bit, non-correctable error in the SE RAM has occurred.	4 (Reset)
"	5	SE hard fault	The SE core has encountered a hard fault exception indicating that an invalid memory access was attempted.	4 (Reset)
"	6	Reserved	—	—
SE Software	7	SE software assertion	SE firmware has triggered an assertion, indicating that one of several sanity checks has failed and that normal operation cannot continue without a reset.	4 (Reset)
"	8	SE secure boot	Secure boot of SE firmware failed	4 (Reset)
"	9	User secure boot	Secure boot of host firmware failed	0 (Ignore)
"	10	Mailbox authorization	Unauthorized command received over the Mailbox interface. This can be triggered by either (1) an incorrectly signed debug unlock or tamper disable token or (2) attempting to export a non-exportable key.	0 (Ignore)
"	11	DCI authorization	Unauthorized command received over the DCI interface. This can be triggered by either (1) an incorrectly signed debug unlock or tamper disable token or (2) attempting to export a non-exportable key.	0 (Ignore)
"	12	OTP read	OTP or flash content could not be properly authenticated	4 (Reset)
"	13	Reserved	—	—
"	14	Self test	A check of the integrity of the SE's internal storage failed during boot up.	4 (Reset)
"	15	TRNG monitor	The TRNG monitor performs a number of tests on the collected entropy data. If any of these tests fail, this tamper source is triggered.	0 (Ignore)
Hardware	16	Secure lock	This tamper source indicates that the guarding mechanism (comparing the locks with their logical complement) of the debug port locks has failed.	4 (Reset)
"	17	Digital glitch	Digital Glitch detector detected an event	0 (Ignore)
"	18	Voltage glitch	Voltage Glitch Detector detected an event	0 (Ignore)
"	19	SE ICACHE	The SE's instruction cache uses a checksum to verify the integrity of the data. This tamper source is triggered if the checksum is invalid.	4 (Reset)

Type	Number	Name	Description	Default Level
"	20	SE RAM ECC1	SE RAM 1-bit ECC error occurred	0 (Ignore)
"	21	BOD [1]	Brown-Out-Detector threshold alert	4 (Reset)
"	22	Temperature sensor [1]	SE temperature is monitored to be within 5 degrees C of the limits for the device. If the limit is exceeded, this tamper source will be triggered.	0 (Ignore)
"	23	DPLL fall	DPLL lock failed low	0 (Ignore)
"	24	DPLL rise	DPLL lock failed high	0 (Ignore)
"	25	PRS0 or ETAMPDET	PRS consumer for SE Tamper 25 or ETAMPDET asserted	0 (Ignore)
"	26 - 31	PRS1 - 6 or PRS0 - 5 [1]	PRS consumer for SE Tamper 26 - 31 asserted	0 (Ignore)

Note:

- [1] These tamper sources are available down to EM2. Other sources are available in EM1 and above.
- In EFR32xG21B devices, hardware tamper sources 24 to 27 can operate down to Energy Mode 3 (EM3), whereas other hardware tamper sources (16 - 23 and 28 - 31) can be active down to Energy Mode 1 (EM1).
- In other HSE-SVH devices, tamper sources 25 to 31 are used for External Tamper Detect (ETAMPDET) if present and PRS consumers. Devices with ETAMPDET (e.g. EFR32xG25B) will have 6 PRS consumers (26 to 31) and devices without ETAMPDET will have 7 PRS consumers (25 to 31).
- The **ETAMPDET** source gets triggered when any of the ETAMPDET channels are asserted.
- User configuration or tamper disable cannot reduce the tamper response below the default Level.
- The **User secure boot** source gets triggered if secure boot is enabled and host image verification fails. It is likely to put the device in the boot loop if users escalate the tamper response of this source to 4 (Reset).
- The **Mailbox and DCI authorizations** get triggered whenever one of the following conditions has occurred. The mailbox returns `SE_RESPONSE_AUTHORIZATION_ERROR`, and DCI returns `AUTH_ERROR = 2`.
  1. A mailbox or DCI command tries to exercise a key that it is not allowed to use (e.g., trying to export a non-exportable key).
  2. A secure debug access or tamper disable request over the mailbox or DCI is invalidly signed.
  3. A malformed HSE firmware upgrade over the mailbox or DCI is attempted.
- The **OTP read** gets triggered if there is an issue when decrypting and authenticating settings in HSE OTP or flash.
- The HSE has redundancy built into the locking mechanism, and the **Secure lock** source is used to detect errors in that redundancy.
- PRS inputs can allow user applications to implement additional tamper sources and feed them into the tamper response mechanism. The **PRS** tamper sources are under the control of the user application and could be reconfigured or disabled if the user application is compromised.
- The **Temperature sensor** source is not completely accurate and is generally only suitable for systems that expect to stay well within the specified temperature range. Users requiring a tighter temperature limit can implement their temperature monitor and provide the results as a tamper source via PRS.
- On EFR32xG23B and later devices, the default behavior is to detect tamper events only when the SE core is active. To detect tamper events when the SE is not performing operations, call `sl_se_enter_active_mode()`. This prevents the SE from sleeping and will result in higher current draw.

6. Anti-Tamper Configuration

The user can provision the anti-tamper configuration in HSE OTP detailed in the following table through `sl_se_init_otp` in the SE Manager. Tamper configurations must be programmed with secure boot settings and are immutable once written.

For more information on enabling the OTP tamper configuration along with the secure boot settings, see section "*Enabling Secure Boot and Tamper Configuration*" in AN1222: Production Programming of Series 2 Devices.

Table 6.1. Anti-Tamper Configuration

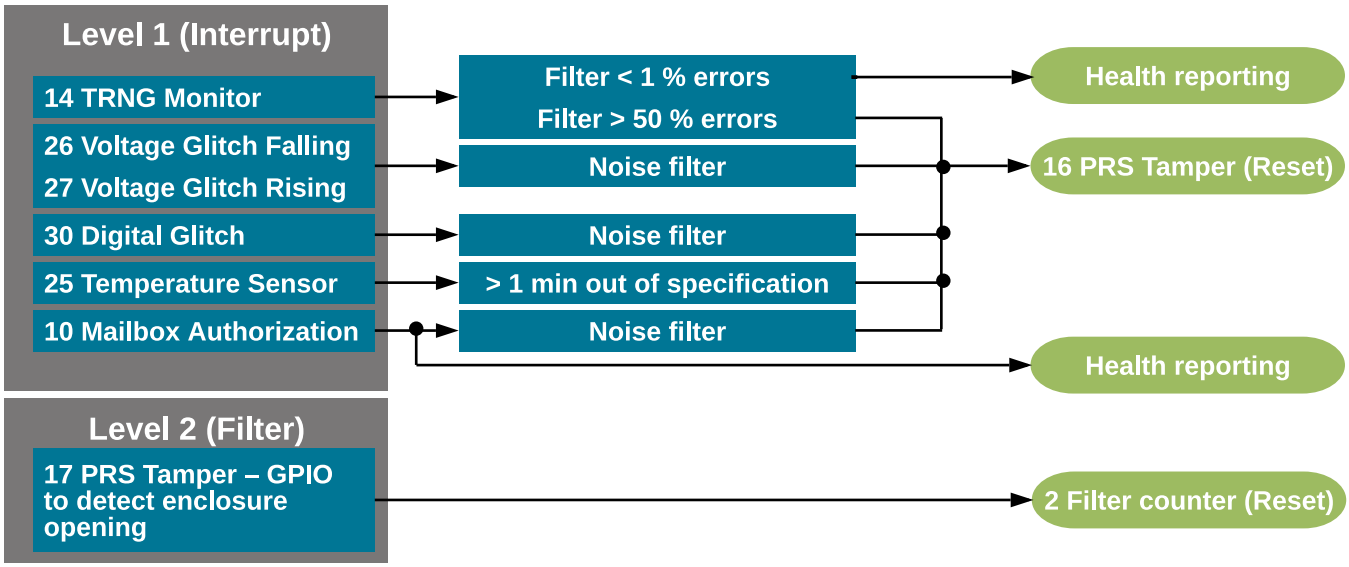
Setting	Description
Tamper response levels	A user response level for each tamper source (1)
Filter settings	The tamper filter counter has two settings: trigger threshold and reset period
Digital Glitch Detector Always On	Bit 1 of tamper flag: 0 — Digital glitch detector runs only when the HSE is executing a command; 1 — Digital glitch detector runs even when the HSE is not performing any operations (note that this leads to increased energy consumption)
Keep Tamper Alive During Sleep (2)	Bit 2 of tamper flag: 0 — The tamper module stops running in sleep mode; 1 — The tamper module keeps running in sleep mode (down to EM3)
Reset threshold	The number of consecutive tamper resets (up to 255) before the part enters diagnostic mode (3)

Note:

1. The effective response of a tamper source is the maximum value between the default level and user level (`Active level = MAX(default level, user level)`). If the user sets the response of a tamper source to a level lower than the default level, the setting will still be saved to HSE OTP but does not take any effect. The HSE returns the user levels instead of active levels of all tamper sources when reading back (sl_se_read_otp) the tamper configuration from the HSE OTP.
2. This flag is not available on EFR32xG21B devices.
3. If the threshold is set to 0, the part will never enter the diagnostic mode due to tamper reset.

7. Usage Example

Several of the available tamper sources report internal HSE errors. A number of these sources are configured to reset the device (level 4) by default. Custom handling of internal and external tamper sources (default level 0) can be configured to trigger an interrupt (level 1) on the Cortex-M33 or trigger an interrupt and increase a counter in the tamper filter (level 2) as in the following figure for EFR32xG21B devices.



Note: The actions for level 1 on the right side are implemented by the tamper interrupt handler.

Figure 7.1. Custom Handling of Tamper Sources (EFR32xG21B Devices)

Usage example highlights:

- The response of the TRNG monitor depends on the failure rate due to lack of entropy.
- The voltage and digital glitch detectors can see spurious activations. They should typically not be used to drive a high-level tamper response directly. Instead, they should feed their signals into a tamper interrupt, which activates a high-level action (e.g., Reset in this example) through PRS tamper if a certain number of detections (noise filter) occur in a short time window.
- The operating conditions decide the time out of the specification filter for the temperature sensor. For some systems, any time out of specification should trigger a reset.
- Mailbox authorization is handled similarly for voltage and digital glitch detectors.
- A PRS tamper implements a high-level response for a tamper interrupt, which issues a tamper reset (level 4) to prevent or slow further attacks.
- In extreme cases, if the system identifies an attack with high confidence, a PRS tamper can be configured as Erase OTP (level 7) to brick the part and prevent further attacks. This is recommended only when the destruction of parts is acceptable and where high confidence of an attack can be achieved.
- Another PRS tamper detects enclosure opening from GPIO. This source feeds into the tamper filter counter (level 2), which will trigger an interrupt (cumulative effect) and activate a `Filter counter` (number 1) response (Reset in this example) if the filter counter reaches the trigger threshold within the filter reset period. This filter counter response approach is less flexible than the interrupt response approach since the trigger threshold and filter reset period are one-time programmable.

8. Tamper Disable

For diagnostic purposes, it may be necessary to disable the tamper response. For example, if a user has configured the part to Erase OTP on external tamper detection, disabling the tamper response is required to open the unit and perform failure analysis or field service activities.

After the tamper configuration has been initialized, users can temporarily restore the tamper response to default for a set of tamper sources via a Disable Tamper Token authenticated against the Public Command Key in HSE OTP (similar to secure debug unlock). This is only possible if the Public Command Key has been provisioned in the device.

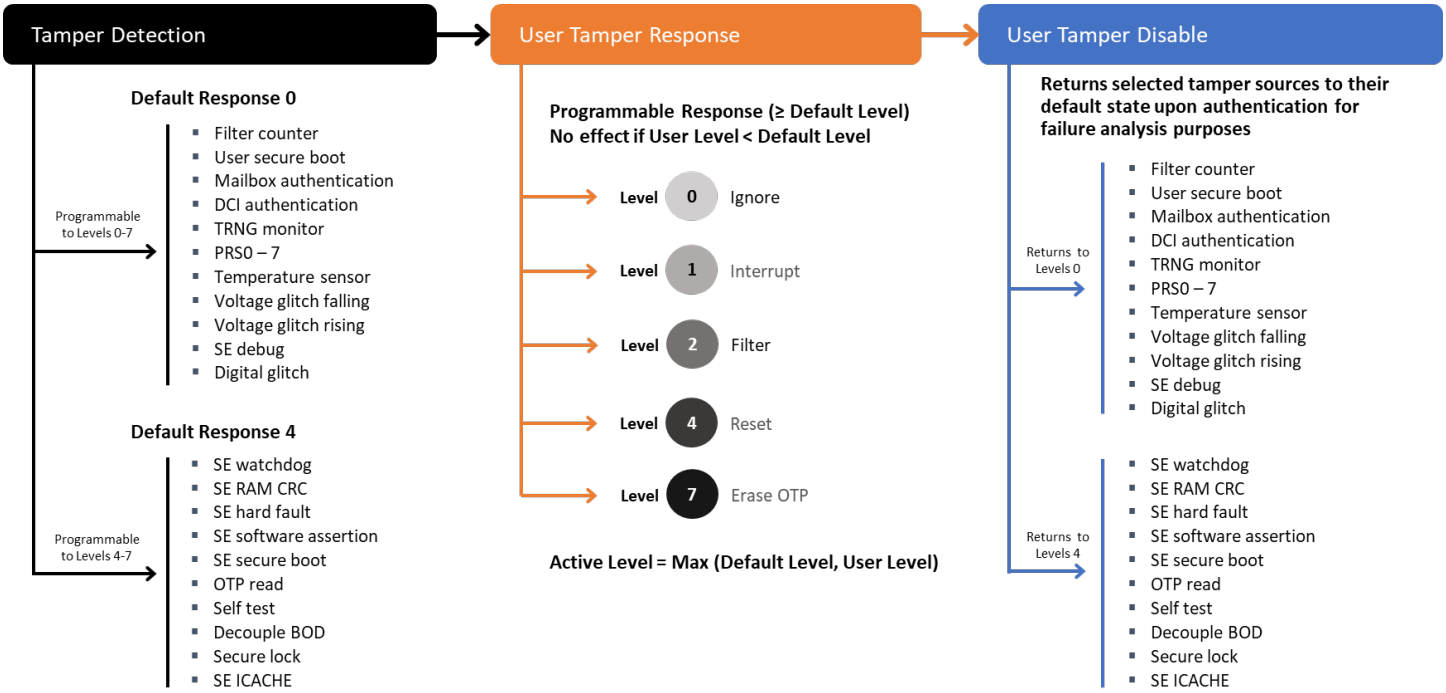


Figure 8.1. Tamper Disable on the EFR32xG21B Devices

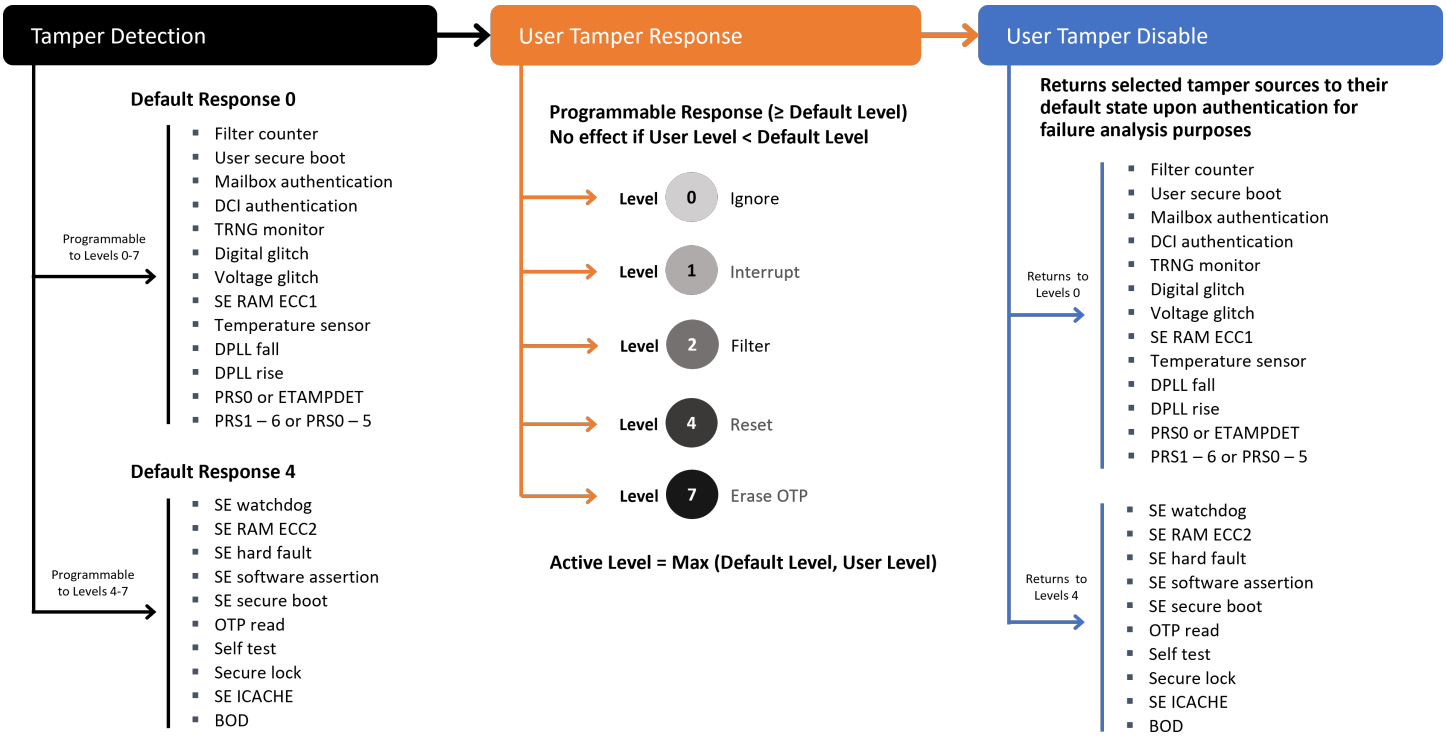


Figure 8.2. Tamper Disable on Other HSE-SVH Devices

8.1 Disable Tamper Token

The elements of the Disable Tamper Token are described in the following figures and table.

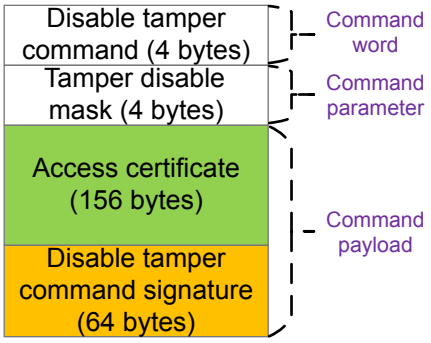


Figure 8.3. Disable Tamper Token

Table 8.1. Elements of Disable Tamper Token

Element	Value	Description
Disable tamper command	0xfd020001	The command word of the Disable Tamper Token.
Tamper disable mask	Device-dependent	The command parameter of the Disable Tamper Token.
Access certificate (1)	Device-dependent	See section Access Certificate.
Disable tamper command signature (1)	Device-dependent	See section Challenge Response.

Note:

1. The disable tamper command payload consists of an access certificate and a disable tamper command signature.

Tamper Disable Mask																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Tamper source 31	Tamper source 30	Tamper source 29	Tamper source 28	Tamper source 27	Tamper source 26	Tamper source 25	Tamper source 24	Tamper source 23	Tamper source 22	Tamper source 21	Tamper source 20	Tamper source 19	Tamper source 18	Tamper source 17	Tamper source 16	Tamper source 15	Tamper source 14	Tamper source 13	Tamper source 12	Tamper source 11	Tamper source 10	Tamper source 9	Tamper source 8	Tamper source 7	Tamper source 6	Tamper source 5	Tamper source 4	Tamper source 3	Tamper source 2	Tamper source 1	Tamper source 0

Figure 8.4. Tamper Disable Mask

Note: Set bit to restore the default response of the corresponding tamper source.

The Disable Tamper Token temporarily reverts all masked tamper sources in the figure above to the hard-coded configuration (Figure 8.1 Tamper Disable on the EFR32xG21B Devices on page 13 and Figure 8.2 Tamper Disable on Other HSE-SVH Devices on page 13).

The Disable Tamper Token can only restore the escalated user-level configuration to default. It cannot degrade the default level of a tamper source.

8.2 Access Certificate

The elements of the access certificate are described in the following figure and table.

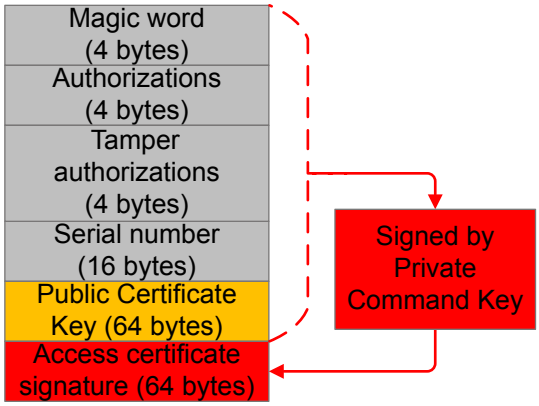


Figure 8.5. Access Certificate

Table 8.2. Elements of the Access Certificate

Element	Value	Description
Magic word	0xe5ecce01	A constant value used to identify the access certificate.
Authorizations	0x0000003e (1)	A value used to authorize which bit in the debug mode request can be enabled for secure debug.
Tamper Authorizations	0xffffffb6 (2)	A value used to authorize which bit in the tamper disable mask can be set to disable the tamper response.
Serial number	Device-dependent	A number used to compare against the on-chip serial number for secure debug or tamper disable.
Public Certificate Key (3)	Device-dependent	The public key corresponding to the Private Certificate Key (3) used to generate the signature (ECDSA-P256-SHA256) in a challenge response.
Access certificate signature	Device-dependent	All the content above is signed (ECDSA-P256-SHA256) by the Private Command Key corresponding to the Public Command Key in the HSE OTP.

Note:

1. The value allows all debug options to be reset for secure debug. Note that the commands for debug unlock and tamper disable are separate, so the secure debug lock will not be disabled when issuing a tamper disable command.
2. Value that sets available bits in the tamper disable mask for tamper disable.
3. The Private/Public Certificate Key is a randomly generated key pair. It can be ephemeral or retainable.

The Private Certificate Key can be used repeatedly to generate the signature in a challenge response on one device until the Private/Public Certificate Key pair is discarded. This can reduce the frequency of access to the Private Command Key, allowing more restrictive access control on that key.

For more information about secure debug, see AN1190: Series 2 Secure Debug.

Tamper Authorizations																															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Tamper disable mask 31	Tamper disable mask 30	Tamper disable mask 29	Tamper disable mask 28	Tamper disable mask 27	Tamper disable mask 26	Tamper disable mask 25	Tamper disable mask 24	Tamper disable mask 23	Tamper disable mask 22	Tamper disable mask 21	Tamper disable mask 20	Tamper disable mask 19	Tamper disable mask 18	Tamper disable mask 17	Tamper disable mask 16	Tamper disable mask 15	Tamper disable mask 14	Tamper disable mask 13	Tamper disable mask 12	Tamper disable mask 11	Tamper disable mask 10	Tamper disable mask 9	Tamper disable mask 8	Tamper disable mask 7	Tamper disable mask 6	Tamper disable mask 5	Tamper disable mask 4	Tamper disable mask 3	Tamper disable mask 2	Tamper disable mask 1	Tamper disable mask 0

Figure 8.6. Tamper Authorizations

Note:

- Set the bit to enable the corresponding bit in the tamper disable mask.
- The Disable Tamper Token will restore the default response of the corresponding tamper source if the same bit is set in the tamper disable mask and tamper authorizations.

8.3 Challenge Response

The elements of the challenge response are described in the following figure and table.

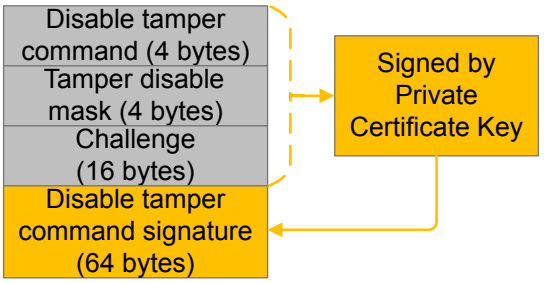


Figure 8.7. Challenge Response

Table 8.3. Elements of the Challenge Response

Element	Value	Description
Disable tamper command	0xfd020001	The command word of Disable Tamper Token.
Tamper disable mask	Device-dependent	The command parameter of Disable Tamper Token.
Challenge	Device-dependent (1)	A random value generated by the HSE.
Disable tamper command signature	Device-dependent (2)	All the content above is signed (ECDSA-P256-SHA256) by the Private Certificate Key corresponding to the Public Certificate Key in the access certificate.

Note:

1. The challenge remains unchanged until it is updated to a new random value by rolling the challenge. The Private Certificate Key can be reused for signing when device challenge is refreshed.
2. This signature is the final argument of the Disable Tamper Token.

8.4 Tamper Disable Flow

The tamper disable flow is described in the following figure.

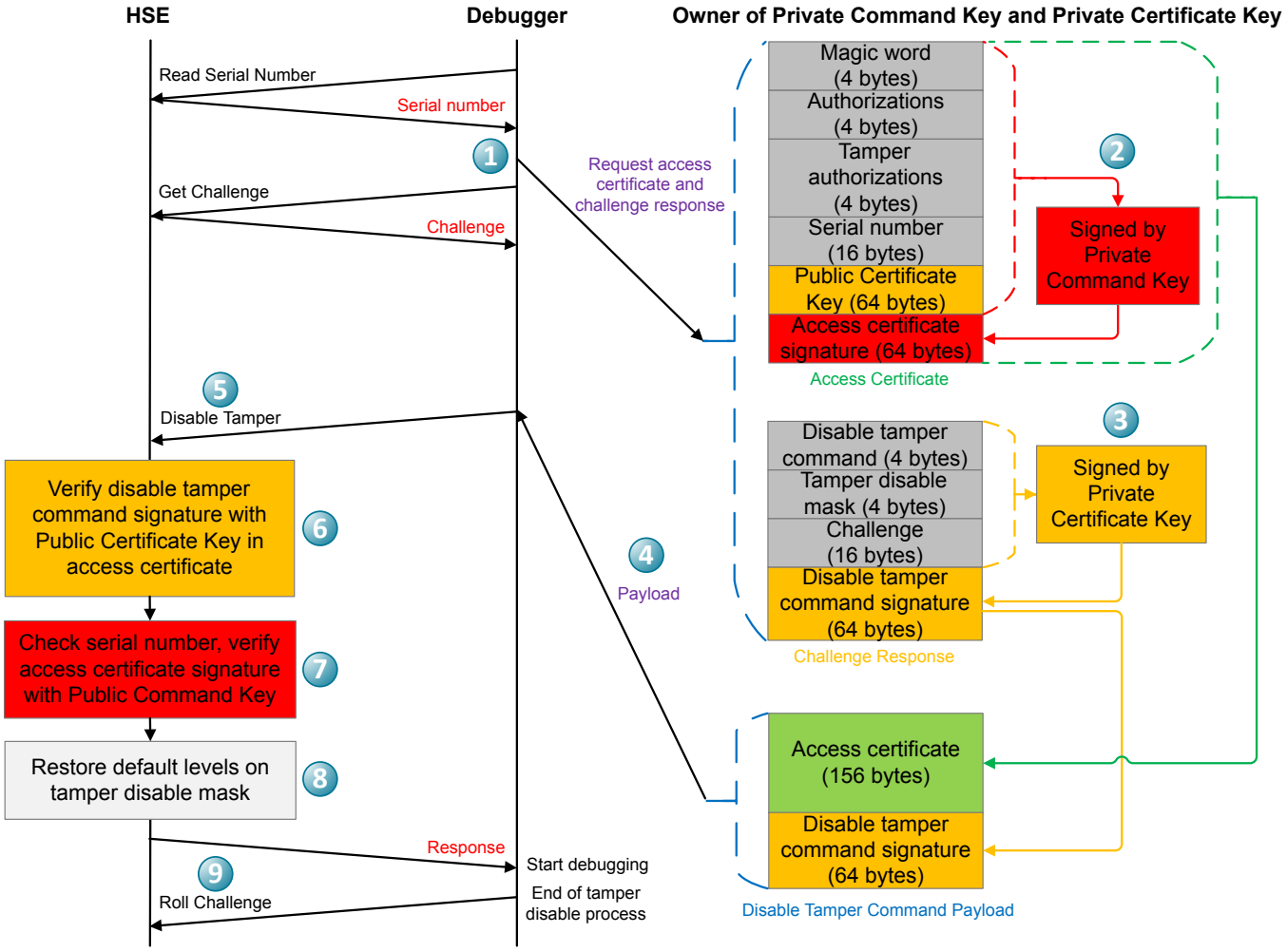


Figure 8.8. Tamper Disable Flow

1. Get the serial number and challenge from the HSE.
2. Generate the access certificate with device serial number.
3. Generate the challenge response with device challenge.
4. Generate the disable tamper command payload with access certificate and disable tamper command signature.
5. Send the Disable Tamper Token to the HSE.
6. Verify the disable tamper command signature using the Public Certificate Key in the access certificate.
7. Verify the serial number and the access certificate signature using the on-chip serial number and Public Command Key in the HSE OTP.
8. Restore default levels on tamper disable mask until the next power-on or pin reset.
9. Roll the challenge to invalidate the current Disable Tamper Token.

Note: Refer to the Simplicity Commander example for details on how to follow this flow using Simplicity Commander.

9. Examples

9.1 Overview

The examples for HSE-SVH Anti-Tamper module are described in the following table.

Table 9.1. Tamper Examples

Example	Device (Radio Board)	HSE Firmware	Tool
Provision Tamper configuration	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	SE Manager
Provision Public Command Key & Tamper configuration	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	Simplicity Commander
"	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	Simplicity Studio 5
Tamper disable and Roll challenge	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	SE Manager
"	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	Simplicity Commander
Roll challenge	EFR32MG21B010F1024IM32 (BRD4181C)	Version 1.2.9	Simplicity Studio 5

Note: Unless specified in the example, these examples can be applied to other HSE-SVH devices.

9.1.1 Using a Platform Example

Simplicity Studio 5 includes the SE Manager platform example for tamper. This application note uses platform examples of GSDK v4.1.0. The console output may be different on the other version of GSDK.

Refer to the corresponding `readme` file for details about each SE Manager platform example. This file also includes the procedures to create the project and run the example.

9.1.2 Using Simplicity Commander

1. This application note uses Simplicity Commander v1.14.6. The procedures and console output may be different on other versions of Simplicity Commander. The latest version of Simplicity Commander can be downloaded from https://www.silabs.com/developers/mcu-programming-options.

```
commander --version
```

```
Simplicity Commander 1v14p6b1289

JLink DLL version: 7.70d
Qt 5.12.10 Copyright (C) 2017 The Qt Company Ltd.
EMDLL Version: 0v18p9b677
mbed TLS version: 2.16.6


DONE
```

2. The Simplicity Commander's Command Line Interface (CLI) is invoked by `commander.exe` in the Simplicity Commander folder. The location for Simplicity Studio 5 in Windows is `C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\commander`. For ease of use, it is highly recommended to add the path of `commander.exe` to the system `PATH` in Windows.
3. If more than one Wireless Starter Kit (WSTK) is connected via USB, the target WSTK must be specified using the `--serialno <J-Link serial number>` option.
4. If the WSTK is in debug mode OUT, the target device must be specified using the `--device <device name>` option.

For more information about Simplicity Commander, see UG162: Simplicity Commander Reference Guide.

9.1.3 Using Simplicity Studio

The security operations are performed in the Security Settings of Simplicity Studio. This application note uses Simplicity Studio v5.4.0.0. The procedures and pictures may be different on the other version of Simplicity Studio 5.

1. Right-click the selected debug adapter **RB (ID:J-Link serial number)** to display the context menu.

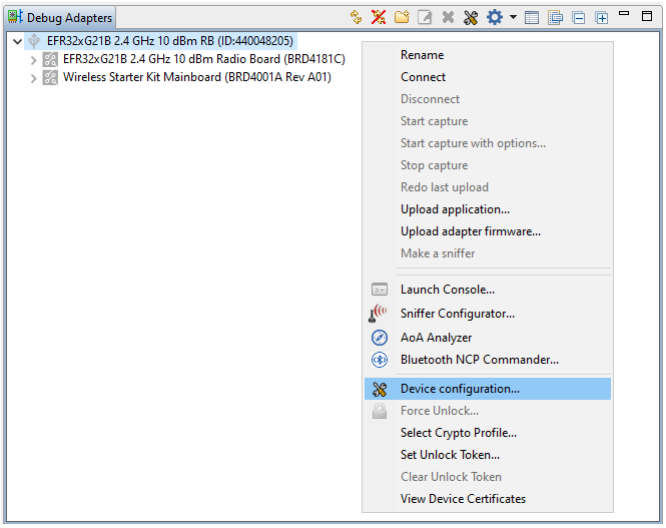


Figure 9.1. Debug Adapters Context Menu

2. Click **Device configuration...** to open the **Configuration of device: J-Link Silicon Labs (serial number)** dialog box. Click the **Security Settings** tab to get the selected device configuration.

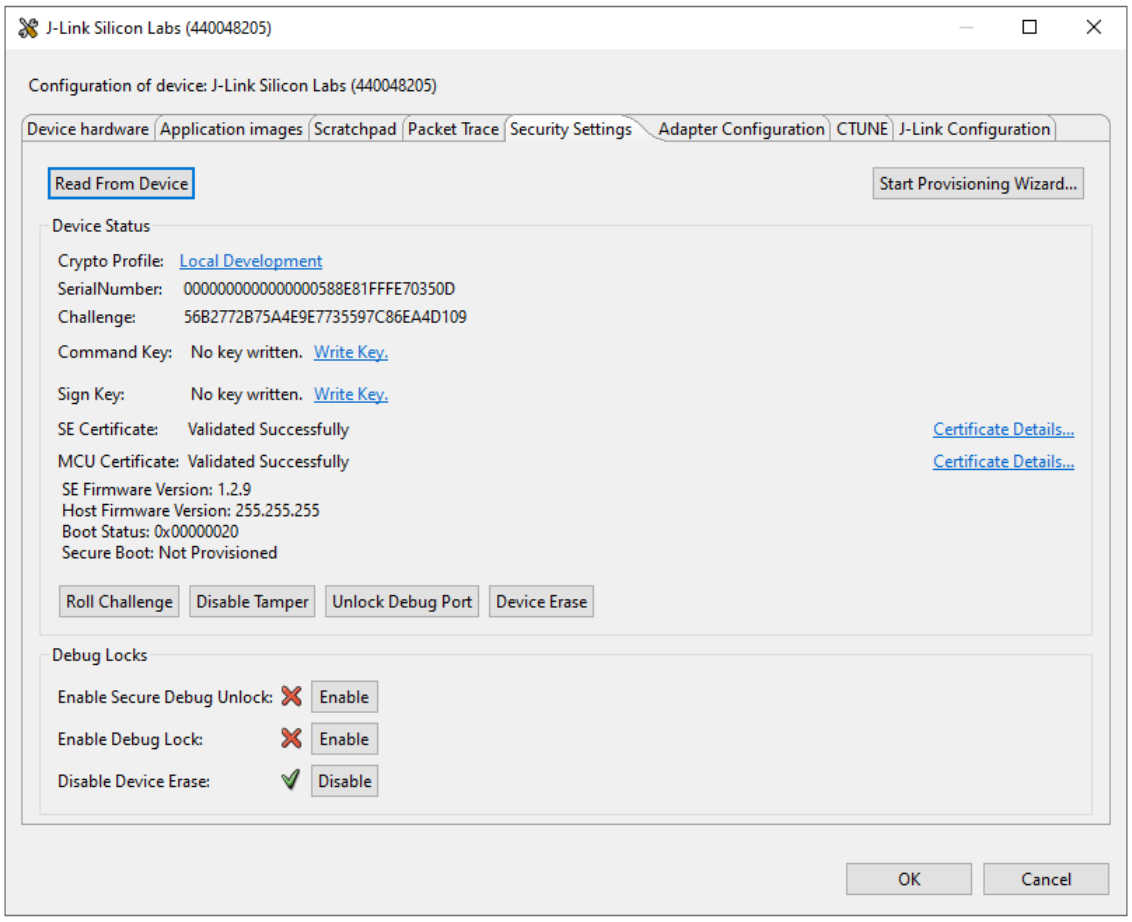


Figure 9.2. Configuration on Selected Device

9.1.4 Using an External Tool

The tamper disable example uses **OpenSSL** to sign the access certificate and challenge response. The Windows version of OpenSSL can be downloaded from https://slproweb.com/products/Win32OpenSSL.html. This application note uses OpenSSL Version 1.1.1h (Win64).

```
openssl version
```

```
OpenSSL 1.1.1h  22 Sep 2020
```

The OpenSSL's Command Line Interface (CLI) is invoked by `openssl.exe` in the OpenSSL folder. The location in Windows (Win64) is `C:\Program Files\OpenSSL-Win64\bin`. For ease of use, it is highly recommended to add the path of `openssl.exe` to the system `PATH` in Windows.

9.2 Provision Public Command Key and Tamper Configuration

The Public Command Key pair can be generated from the “unsafe” private key delivered with Simplicity Studio, by Simplicity Commander, or by a Hardware Security Module (HSM). Using an HSM is recommended for production systems.

Generated from “Unsafe” Key

External tools such as openssl can be used to generate a public key from the reference private key provided in Simplicity Studio. Note that this private key is well known and should not be used in production devices.

Run the `openssl ec` command to generate the Public Command Key from the Private Command Key.

```
openssl ec -in /c/SiliconLabs/SimplicityStudio/v5/developer/adapter_packs/secmgr/scripts/offline/cmd-unsafe-
privkey.pem -pubout -out cmd-unsafe-pubkey.pem
```

Generated Using Simplicity Commander

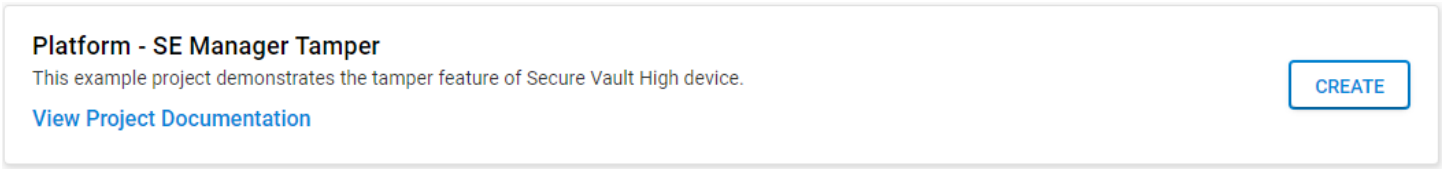
Run the `util genkey` command to generate the Public Command Key pair (`command_key.pem` and `command_pubkey.pem`) and Public Command Key token file (`command_pubkey.txt`).

```
commander util genkey --type ecc-p256 --privkey command_key.pem --pubkey command_pubkey.pem
--tokenfile command_pubkey.txt
```

```
Generating ECC P256 key pair...
Writing private key file in PEM format to command_key.pem
Writing public key file in PEM format to command_pubkey.pem
Writing EC tokens to command_pubkey.txt...
DONE
```

9.2.1 SE Manager - Tamper Platform Example

Click the `View Project Documentation` link to open the readme file for instructions on creating the project and running the example.



1. Press `ENTER` two times to program the secure boot and tamper configuration to the HSE OTP of an uninitialized device.

```
SE Manager Tamper Example - Core running at 38000 kHz.
  . SE manager initialization... SL_STATUS_OK (cycles: 7 time: 0 us)

  . Read EMU RSTCAUSE register... SL_STATUS_OK (cycles: 3728 time: 98 us)
  + The EMU RSTCAUSE register (MSB..LSB): 00000043

  . Read SE OTP configuration... SL_STATUS_NOT_INITIALIZED (cycles: 7487 time: 197 us)
  + Cannot read SE OTP configuration.
  + Press ENTER to initialize SE OTP for tamper configuration or press SPACE to abort.
  + Warning: The OTP configuration cannot be changed once written!
  + Press ENTER to confirm or press SPACE to abort if you are not sure.
  + Initialize SE OTP for tamper configuration... SL_STATUS_OK (cycles: 267256 time: 7033 us)
  + Issue a power-on or pin reset to activate the new tamper configuration.

  . SE manager deinitialization... SL_STATUS_OK (cycles: 9 time: 0 us)
```

Note: This example does not enable the secure boot.

2. Press the **RESET** button on the WSTK to restart the program. It will display the current tamper configuration of the device.

```
SE Manager Tamper Example - Core running at 38000 kHz.
  . SE manager initialization... SL_STATUS_OK (cycles: 10 time: 0 us)

  . Read EMU RSTCAUSE register... SL_STATUS_OK (cycles: 3736 time: 98 us)
  + The EMU RSTCAUSE register (MSB..LSB): 00000043

  . Read SE OTP configuration... SL_STATUS_OK (cycles: 7174 time: 188 us)
  + Secure boot: Disabled
  + Tamper source level
    Filter counter         :  1
    SE watchdog            :  4
    SE RAM CRC             :  4
    SE hard fault          :  4
    SE software assertion  :  4
    SE secure boot         :  4
    User secure boot       :  0
    Mailbox authorization  :  1
    DCI authorization      :  0
    OTP read               :  4
    Self test              :  4
    TRNG monitor           :  1
    PRS0                   :  1
    PRS1                   :  1
    PRS2                   :  2
    PRS3                   :  2
    PRS4                   :  4
    PRS5                   :  4
    PRS6                   :  7
    PRS7                   :  7
    Decouple BOD           :  4
    Temperature sensor     :  2
    Voltage glitch falling :  2
    Voltage glitch rising  :  2
    Secure lock            :  4
    SE debug               :  0
    Digital glitch         :  2
    SE ICACHE              :  4
  + Reset period for the tamper filter counter: ~32 ms x 1024
  + Activation threshold for the tamper filter: 4
  + Digital glitch detector always on: Disabled
  + Tamper reset threshold: 5

  . Current tamper test is NORMAL.
  + Press SPACE to select NORMAL or TAMPER DISABLE, press ENTER to run.
```

9.2.2 Simplicity Commander

1. Run the `security writekey` command to provision the Public Command Key (e.g., `command_pubkey.pem`).

```
commander security writekey --command command_pubkey.pem --device EFR32MG21B010F1024 --serialno 440030580
```

```
Device has serial number 00000000000000014b457fffe0f77ce

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

Note: The Public Command Key cannot be changed once written.

2. Run the `security readkey` command to read the Public Command Key from the HSE OTP for verification with the key in step 1.

```
commander security readkey --command --device EFR32MG21B010F1024 --serialno 440030580
```

```
B1BC6F6FA56640ED522B2EE0F5B3CF7E5D48F60BE8148F0DC08440F0A4E1DCA4
7C04119ED6A1BE31B7707E5F9D001A659A051003E95E1B936F05C37EA793AD63
DONE
```

3. Run the `security genconfig` command to generate a default `user_configuration.json` file for secure boot and tamper configuration.

```
commander security genconfig --nostore -o user_configuration.json --device EFR32MG21B010F1024 --serialno
440030580
```

```
Configuration file written to user_configuration.json
DONE
```

Note: Simplicity Commander Version 1.14.6 or above is required to support tamper configuration for all HSE-SVH devices.

4. Use a text editor to modify the default tamper responses in `user_configuration.json` to the desired configuration as below.

```
{
    "OPN": "EFR32MG21B010F1024",
    "VERSION": "1.0.0",
    "mcu_flags": {
        "SECURE_BOOT_ANTI_ROLLBACK": false,
        "SECURE_BOOT_ENABLE": false,
        "SECURE_BOOT_PAGE_LOCK_FULL": false,
        "SECURE_BOOT_PAGE_LOCK_NARROW": false,
        "SECURE_BOOT_VERIFY_CERTIFICATE": false
    },
    "tamper_filter": {
        "FILTER_PERIOD": 10,
        "FILTER_THRESHOLD": 6,
        "RESET_THRESHOLD": 5
    },
    "tamper_flags": {
        "DGLITCH_ALWAYS_ON": false
    },
    "tamper_levels": {
        "DCI_AUTH": 0,
        "DECOUPLE_BOD": 4,
        "DGLITCH": 2,
        "FILTER_COUNTER": 1,
        "MAILBOX_AUTH": 1,
        "OTP_READ": 4,
        "PRS0": 1,
        "PRS1": 1,
        "PRS2": 2,
        "PRS3": 2,
        "PRS4": 4,
        "PRS5": 4,
        "PRS6": 7,
        "PRS7": 7,
        "SECURE_LOCK": 4,
        "SELF_TEST": 4,
        "SE_CODE_AUTH": 4,
        "SE_DEBUG": 0,
        "SE_HARDFAULT": 4,
        "SE_ICACHE": 4,
        "SE_RAM_CRC": 4,
        "SOFTWARE_ASSERTION": 4,
        "TEMP_SENSOR": 2,
        "TRNG_MONITOR": 1,
        "USER_CODE_AUTH": 0,
        "VGLITCH_FALLING": 2,
        "VGLITCH_RISING": 2,
        "WATCHDOG": 4
    }
}
```

Note: This example does not enable the secure boot.

5. Run the `security writeconfig` command to program the secure boot and tamper configuration to the HSE OTP. This command can be executed once per device.

```
commander security writeconfig --configfile user_configuration.json --device EFR32MG21B010F1024 --serialno
440030580
```

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

6. Run the `security readconfig` command to check the secure boot and tamper configuration of the device.

```
commander security readconfig --serialno 440030580
```

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

Tamper Levels
FILTER_COUNTER     : 1
WATCHDOG           : 4
SE_RAM_CRC         : 4
SE_HARDFAULT       : 4
SOFTWARE_ASSERTION : 4
SE_CODE_AUTH       : 4
USER_CODE_AUTH     : 0
MAILBOX_AUTH       : 1
DCI_AUTH           : 0
OTP_READ           : 4
SELF_TEST          : 4
TRNG_MONITOR       : 1
PRS0               : 1
PRS1               : 1
PRS2               : 2
PRS3               : 2
PRS4               : 4
PRS5               : 4
PRS6               : 7
PRS7               : 7
DECOUPLE_BOD       : 4
TEMP_SENSOR        : 2
VGLITCH_FALLING    : 2
VGLITCH_RISING     : 2
SECURE_LOCK        : 4
SE_DEBUG           : 0
DGLITCH            : 2
SE_ICACHE          : 4

Tamper Filter
Filter Period   : 10
Filter Threshold : 6
Reset Threshold  : 5

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

9.2.3 Simplicity Studio

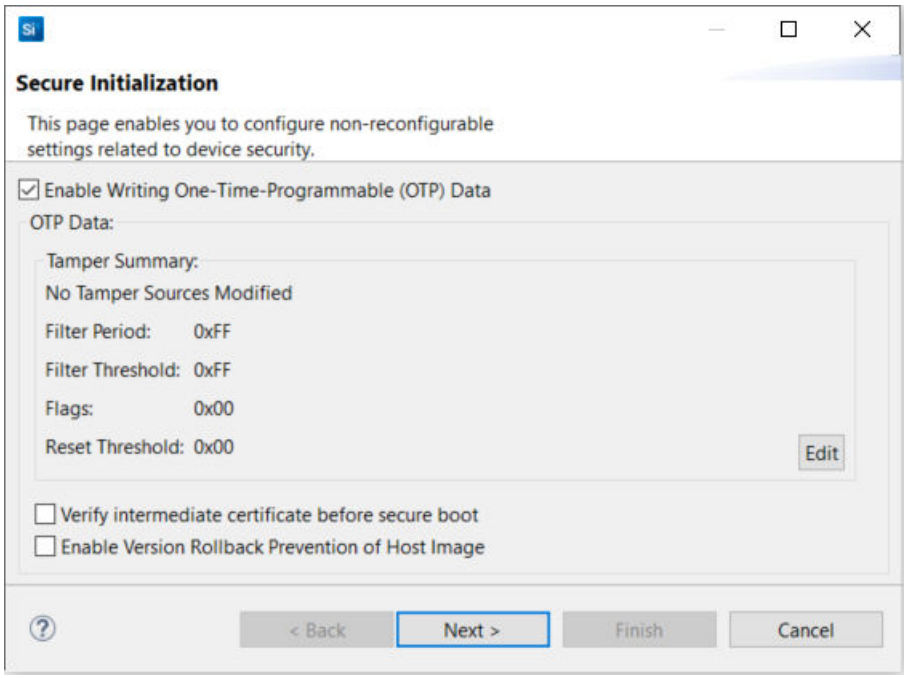
This example focuses on provisioning the Public Command Key and tamper configuration. It skips the procedures for provisioning of the Public Sign Key and Secure Boot Enabling.

1. Run the `util keytotoken` command to convert the Public Command Key file (PEM format) into a text file (`command_pubkey.txt`).

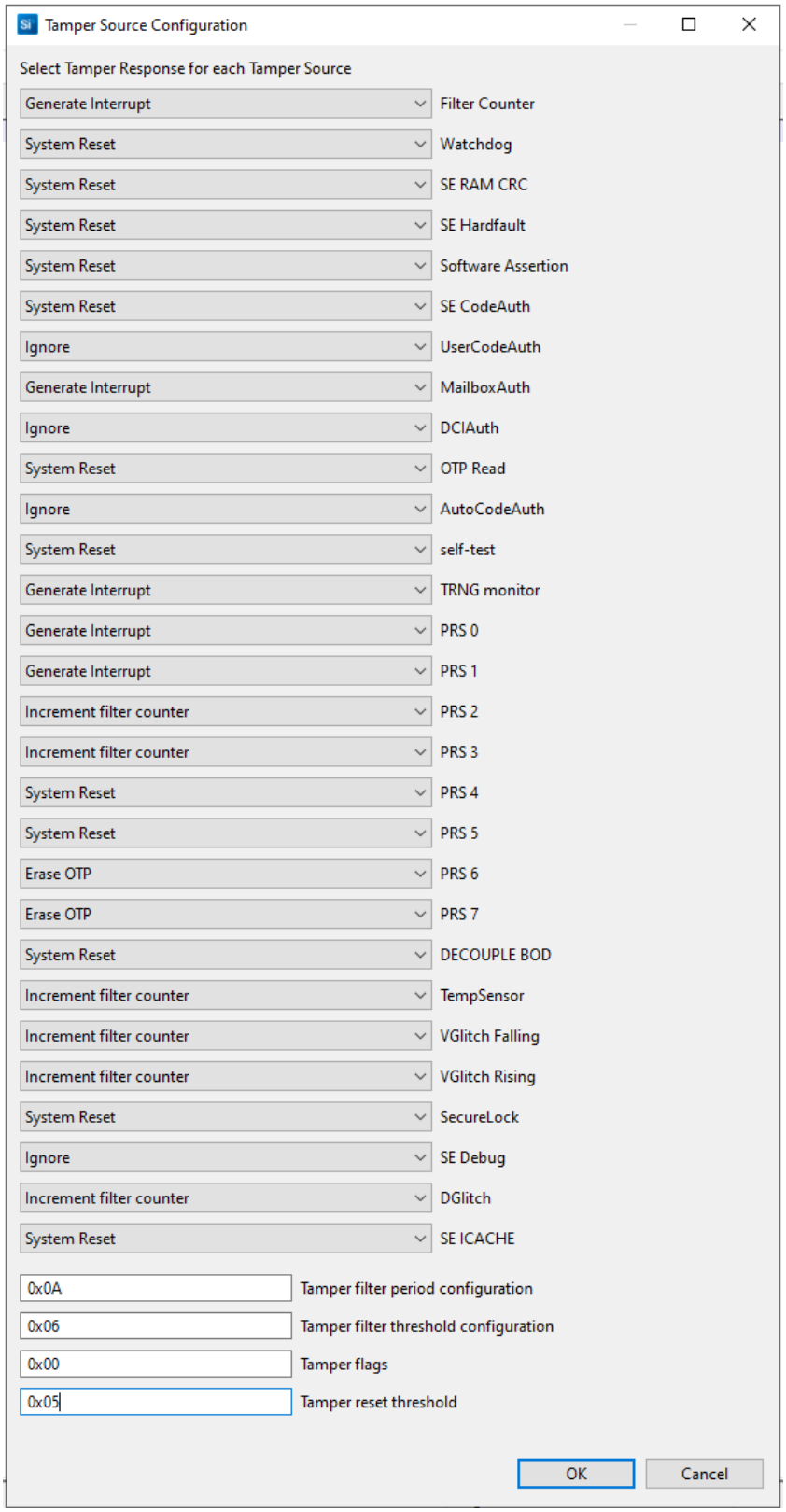
```
commander util keytotoken command_pubkey.pem --outfile command_pubkey.txt
```

```
Writing EC tokens to command_pubkey.txt...
DONE
```

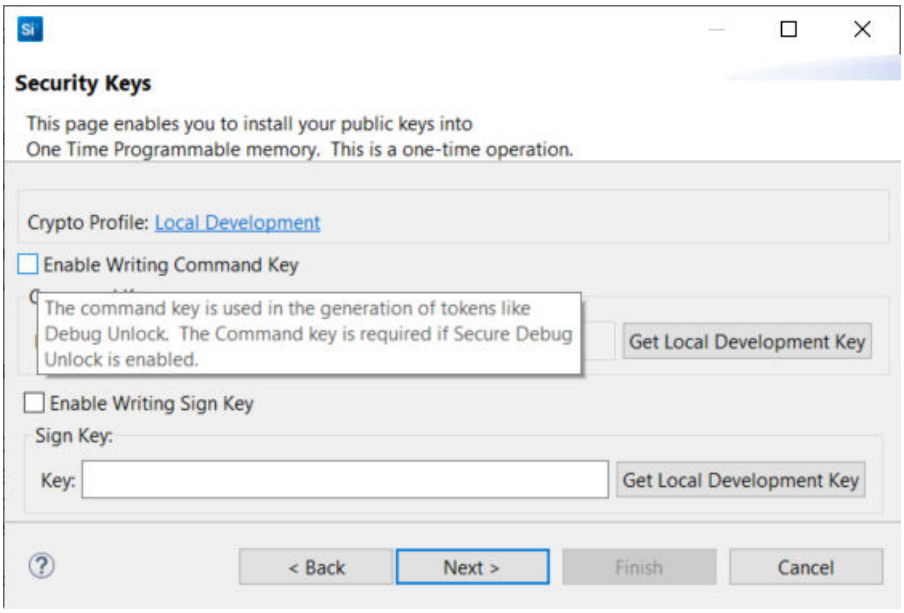
2. Open the **Security Settings** of the selected device as described in 9.1.3 Using Simplicity Studio.
3. Click [**Start Provisioning Wizard...**] in the upper right corner to display the **Secure Initialization** dialog box.



4. Click [**Edit**] to open the **Tamper Source Configuration** dialog box. Use the dropdown menus to modify the default tamper responses to the desired configuration. Click [**OK**] to exit.



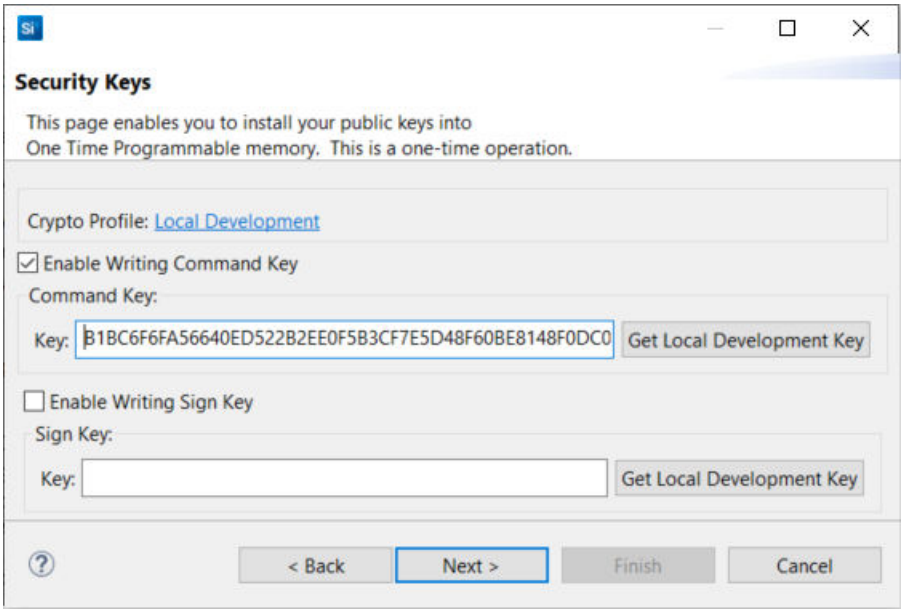
5. Click [**Next >**]. The **Security Keys** dialog box is displayed.



6. Using a text editor, open the `command_pubkey.txt` file generated in step 1.

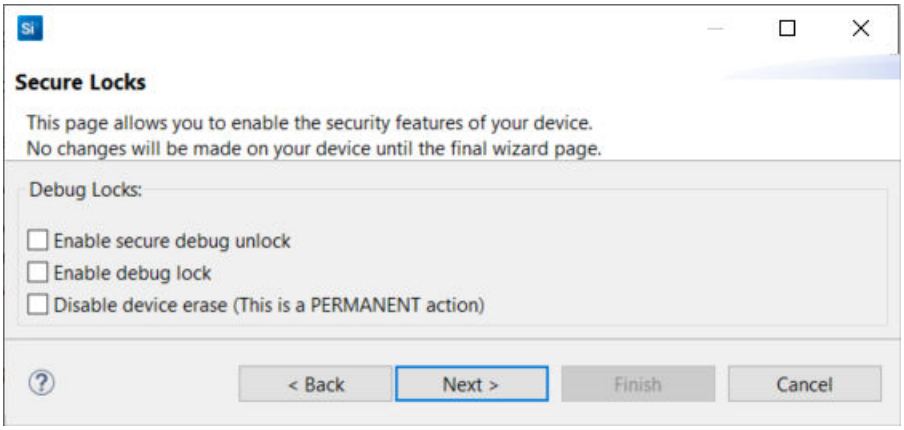
```
MFG_SIGNED_BOOTLOADER_KEY_X : B1BC6F6FA56640ED522B2EE0F5B3CF7E5D48F60BE8148F0DC08440F0A4E1DCA4
MFG_SIGNED_BOOTLOADER_KEY_Y : 7C04119ED6A1BE31B7707E5F9D001A659A051003E95E1B936F05C37EA793AD63
```

7. Check **Enable Writing Command Key**. Copy the Public Command Key (X-point `B1BC...` first, then Y-point `7C04...`) to the **Key:** box under **Command Key:**.

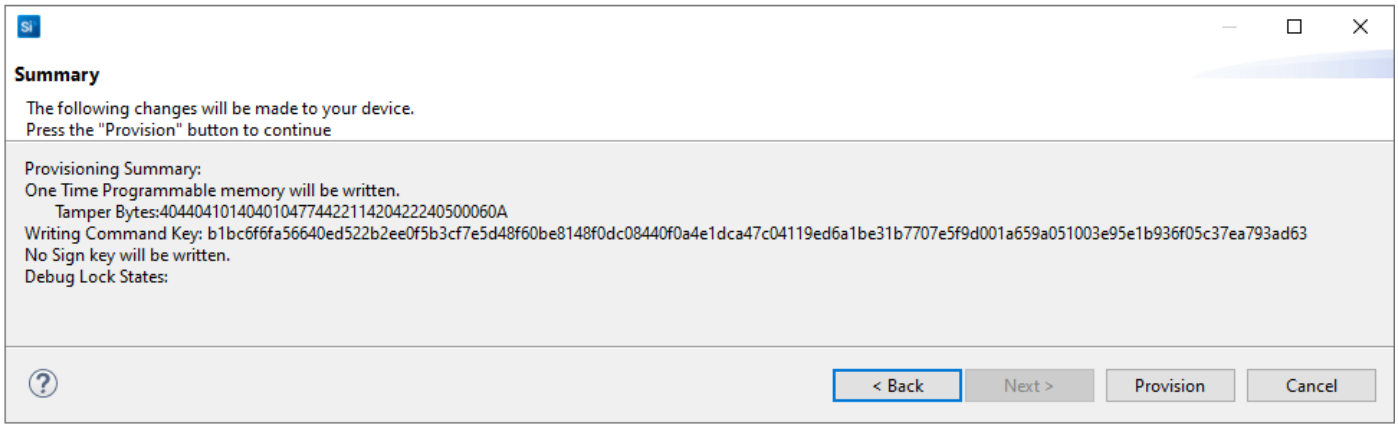


Note: This example does not enable the secure boot (not checking **Enable Writing Sign Key** option).

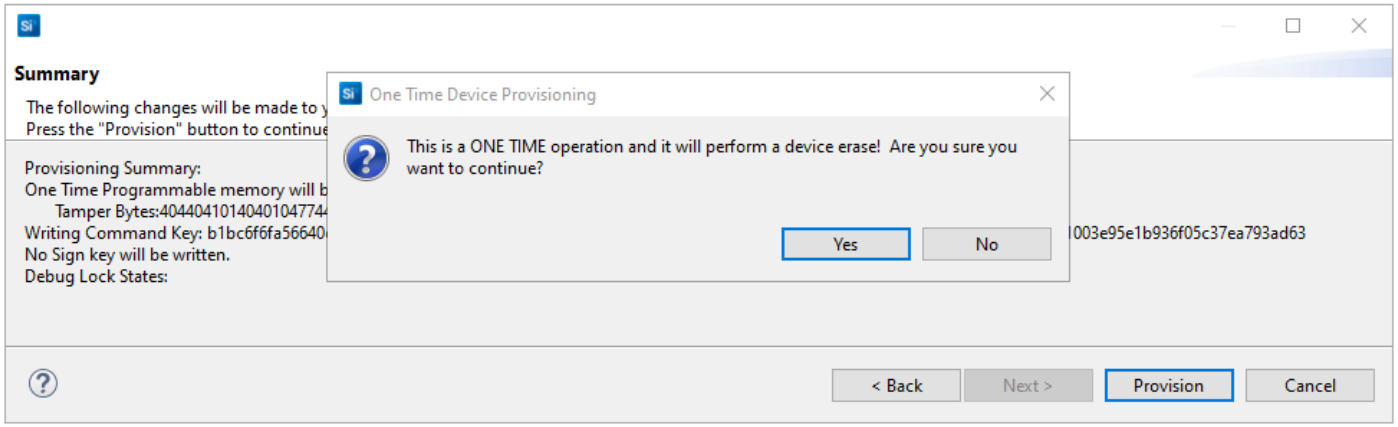
8. Click [**Next >**]. The **Secure Locks** dialog box is displayed. The **Debug locks** are set by default. Uncheck **Enable secure debug unlock** and **Enable debug lock**.



9. Click [**Next >**] to display the **Summary** dialog box. Verify the tamper configuration and Public Command Key in the **Provisioning Summary** are correct.

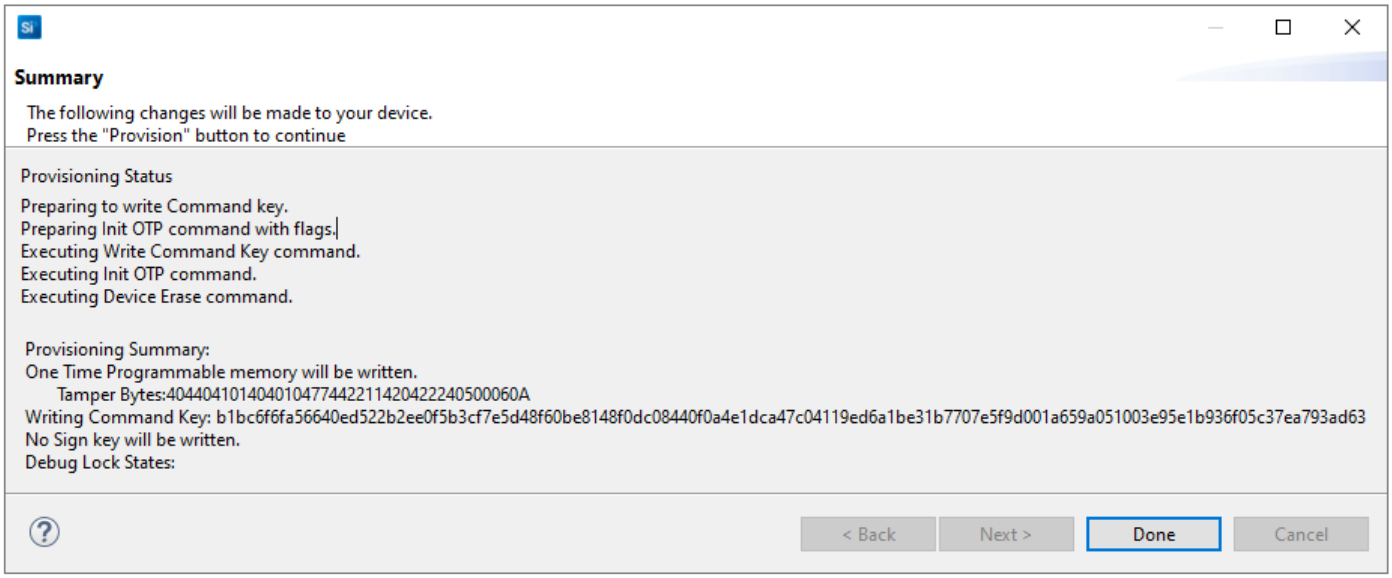


10. If the information displayed is correct, click [**Provision**]. Click [**Yes**] to confirm.

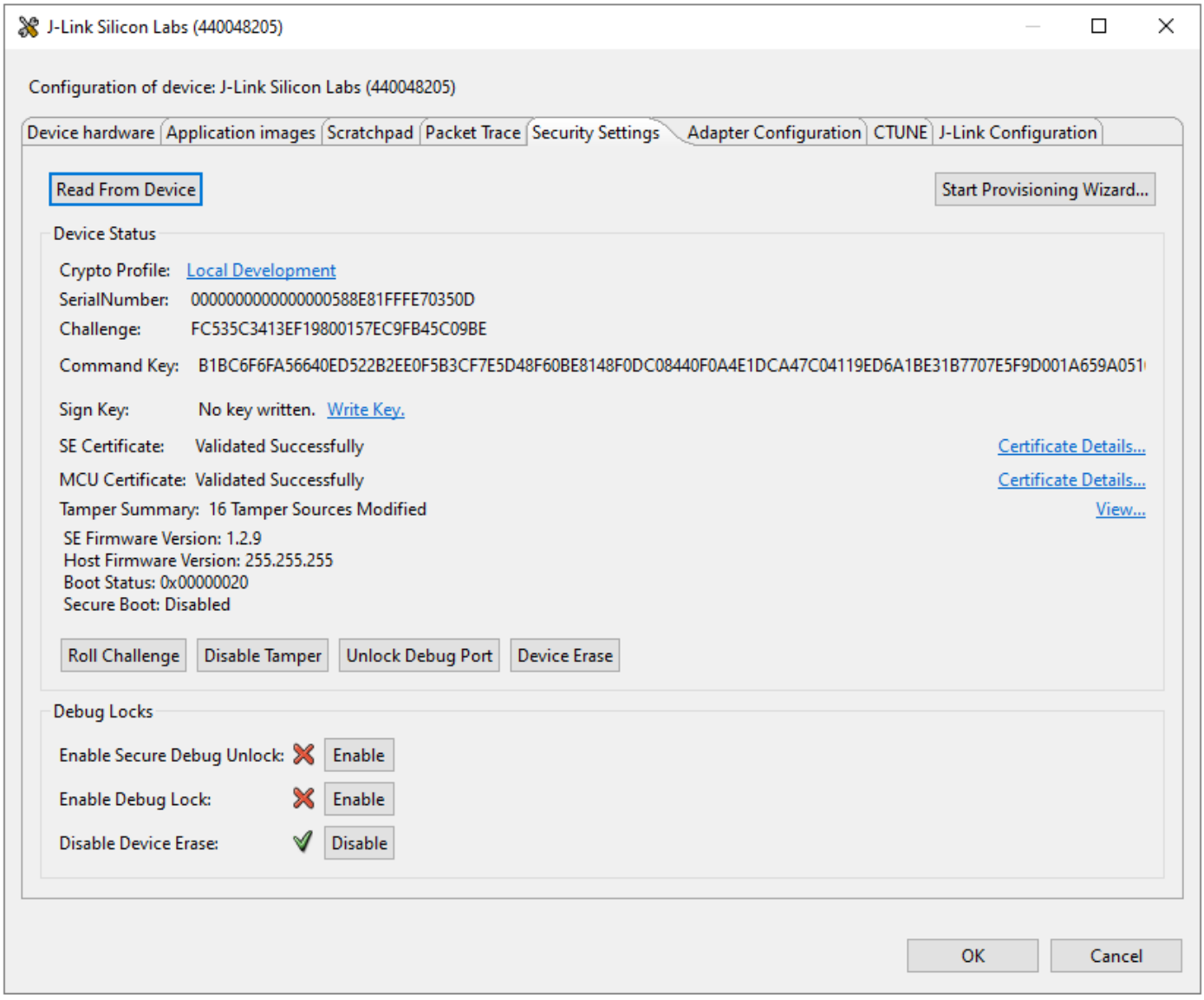


Note: The Public Command Key and tamper configuration cannot be changed once written.

11. The **Provisioning Status** is displayed in the **Summary** dialog box.



12. Click [**Done**] to exit the provisioning process. The device configuration is updated.



13. Click the **View...** link to check the tamper configuration or click [**OK**] to exit.

9.3 Tamper Disable and Roll Challenge

PRS Tamper Sources

The tamper configuration in the SE Manager Tamper platform example is used to demonstrate the tamper disable on HSE-SVH devices. The following tables list the PRS tamper source usage on EFR32xG21B and other HSE-SVH devices on this example. The push buttons PB0 and PB1 are on the Wireless Starter Kit (WSTK) Mainboard.

Table 9.2. PRS Tamper Source Usage on EFR32xG21B Devices

Source (Bit)	Default Level (Response)	User Level (Response)	PRS Producer	Tamper Disable Mask (1)
PRS0 (16)	0 (Ignore)	1 (Interrupt)	Push button PB0	0
PRS1 (17)	0 (Ignore)	1 (Interrupt)	—	1
PRS2 (18)	0 (Ignore)	2 (Filter)	Push button PB0	0
PRS3 (19)	0 (Ignore)	2 (Filter)	—	1
PRS4 (20)	0 (Ignore)	4 (Reset)	Push button PB1	1
PRS5 (21)	0 (Ignore)	4 (Reset)	Software (2)	1
PRS6 (22)	0 (Ignore)	7 (Erase OTP)	—	1
PRS7 (23)	0 (Ignore)	7 (Erase OTP)	—	1

Note:

1. The tamper disable mask is `0x00fa0000` to restore the tamper sources PRS1, PRS3, PRS4, PRS5, PRS6, and PRS7 to default response (Ignore).
2. The Software PRS triggers the tamper source PRS5 to reset the device if the filter counter reaches the trigger threshold (4) within the filter reset period (~32 ms x 1024).

Table 9.3. PRS Tamper Source Usage on Other HSE-SVH Devices

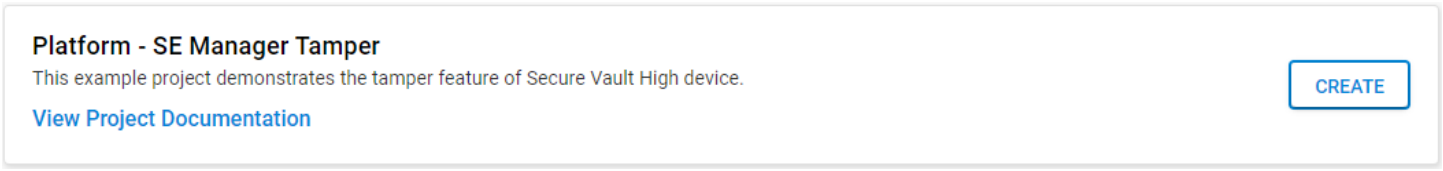
Source (Bit) (1)	Default Level (Response)	User Level (Response)	PRS Producer	Tamper Disable Mask (2)
PRS0 (25 or 26)	0 (Ignore)	1 (Interrupt)	—	1
PRS1 (26 or 27)	0 (Ignore)	1 (Interrupt)	Push button PB0	0
PRS2 (27 or 28)	0 (Ignore)	2 (Filter)	Push button PB0	0
PRS3 (28 or 29)	0 (Ignore)	2 (Filter)	—	1
PRS4 (29 or 30)	0 (Ignore)	4 (Reset)	Push button PB1	1
PRS5 (30 or 31)	0 (Ignore)	4 (Reset)	Software (3)	1
PRS6 (31 or —)	0 (Ignore)	7 (Erase OTP)	—	1

Note:

1. The HSE-SVH devices with ETAMPDET peripheral only have PRS0 (bit 26) to PRS5 (bit 31).
2. The tamper disable mask depends on whether the HSE-SVH device has an ETAMPDET peripheral.
   a. Without ETAMPDET peripheral, the tamper disable mask is `0xf2000000` to restore the tamper sources PRS0, PRS3, PRS4, PRS5, and PRS6 to default response (Ignore).
   b. With ETAMPDET peripheral, the tamper disable mask is `0xe4000000` to restore the tamper sources PRS0, PRS3, PRS4, and PRS5 to default response (Ignore).
3. The Software PRS triggers the tamper source PRS5 to reset the device if the filter counter reaches the trigger threshold (4) within the filter reset period (~32 ms x 1024).

9.3.1 SE Manager - Tamper Platform Example

Click the `View Project Documentation` link to open the `readme` file for instructions on creating the project and running the example.



Follow the procedures in 9.2.1 SE Manager - Tamper Platform Example if the HSE OTP is uninitialized. The following sections describe an initialized device that runs in Normal and Tamper Disable modes.

9.3.1.1 Normal

1. Press `ENTER` to run the `NORMAL` tamper demo. Follow the instructions to go through the example.

```
. Current tamper test is NORMAL.
+ Press SPACE to select NORMAL or TAMPER DISABLE, press ENTER to run.

. Normal tamper test instructions:
+ Press PB0 to increase filter counter and tamper status is displayed.
+ PRS will issue a tamper reset if filter counter reaches 4 within ~32 ms x 1024.
+ Press PB1 to issue a tamper reset.
+ Device will enter diagnostic mode if tamper reset reaches 5.
```

2. Press PB0 to trigger PRS0 (Interrupt) and PRS2 (Filter) to issue an interrupt. The active tamper sources (`0x00050000`) of the EFR32xG21B device are PRS0 (`bit 16`) and PRS2 (`bit 18`).

```
. Get tamper status... SL_STATUS_OK (cycles: 11937 time: 314 us)
+ Recorded tamper status (MSB..LSB): 00050001
+ Currently active tamper sources (MSB..LSB): 00050000
```

3. Press PB0 (Filter on PRS2) 4 times within ~32 ms x 1024 to trigger an interrupt when reaching the filer counter threshold. The program will use software PRS to issue a tamper reset through the PRS5 tamper source. The active tamper sources (`0x00050002`) of the EFR32xG21B device are Filter (`bit 2`), PRS0 (`bit 16`), and PRS2 (`bit 18`).

```
. Get tamper status... SL_STATUS_OK (cycles: 11725 time: 308 us)
+ Recorded tamper status (MSB..LSB): 00050002
+ Currently active tamper sources (MSB..LSB): 00050002
+ Tamper filter threshold is reached, issue a reset through PRS
```

4. Press PB1 to trigger PRS4 (Reset) to issue a tamper reset.
5. After a tamper reset, the `SETAMPER` (bit 13) in `EMU->RSTCAUSE` register is set. Note that bit 1 indicates a pin reset and will also be set.

```
. Read EMU RSTCAUSE register... SL_STATUS_OK (cycles: 4071 time: 107 us)
+ The EMU RSTCAUSE register (MSB..LSB): 00002002
+ The tamper reset is observed
```

6. After five consecutive tamper resets (reset threshold in this example), the device will enter diagnostic mode until a power-on or pin reset.

9.3.1.2 Tamper Disable

This example uses the tamper disable mask (`0x00fa0000`) to restore the tamper sources PRS1, PRS3, PRS4, PRS5, PRS6, and PRS7 of EFR32xG21B device to default response (Ignore).

1. Press `SPACE` to select `TAMPER DISABLE`, press `ENTER` to run.

```
. Current tamper test is NORMAL.
+ Press SPACE to select NORMAL or TAMPER DISABLE, press ENTER to run.
+ Current tamper test is TAMPER DISABLE.
```

2. This example will prompt to program the default Public Command Key in flash to the HSE OTP if this key does not exist. Press `ENTER` two times to confirm and `ENTER` again to restore the default tamper level. Follow the instructions shown in step 3 to go through the example (steps 4 to 6).

```
. Verify the device public command key in SE OTP.
+ Exporting a public command key from a hard-coded private command key... SL_STATUS_OK (cycles: 210999 time: 5552 us)
+ Reading the public command key from SE OTP... SL_STATUS_NOT_INITIALIZED (cycles: 7763 time: 204 us)
+ Press ENTER to program public command key in SE OTP or press SPACE to abort.
+ Warning: The public command key in SE OTP cannot be changed once written!
+ Press ENTER to confirm or press SPACE to skip if you are not sure.
+ Programming a public command key to SE OTP... SL_STATUS_OK (cycles: 79656 time: 2096 us)
+ Press ENTER to disable tamper signals or press SPACE to exit.
```

3. Press `ENTER` to restore the default tamper level if the default Public Command Key in flash matches with the key in the HSE OTP. Follow the instructions to go through the example (steps 4 to 6).

```
. Verify the device public command key in SE OTP.
+ Exporting a public command key from a hard-coded private command key... SL_STATUS_OK (cycles: 200804 time: 5284 us)
+ Reading the public command key from SE OTP... SL_STATUS_OK (cycles: 7134 time: 187 us)
+ Comparing exported public command key with SE OTP public command key... OK
+ Press ENTER to disable tamper signals or press SPACE to exit.

. Start the tamper disable processes.
+ Creating a private certificate key in a buffer... SL_STATUS_OK (cycles: 214059 time: 5633 us)
+ Exporting a public certificate key from a private certificate key... SL_STATUS_OK (cycles: 206545 time: 5435 us)
+ Read the serial number of the SE and save it to access certificate... SL_STATUS_OK (cycles: 7930 time: 208 us)
+ Signing the access certificate with private command key... SL_STATUS_OK (cycles: 222650 time: 5859 us)
+ Request challenge from the SE and save it to challenge response... SL_STATUS_OK (cycles: 4208 time: 110 us)
+ Signing the challenge response with private certificate key... SL_STATUS_OK (cycles: 223559 time: 5883 us)
+ Creating a tamper disable token to disable tamper signals... SL_STATUS_OK (cycles: 946431 time: 24906 us)
+ Success to disable the tamper signals!

. Tamper disable test instructions:
+ Press PB0 to increase filter counter and tamper status is displayed.
+ PRS will NOT issue a tamper reset even filter counter reaches 4 within ~32 ms x 1024.
+ Press PB1 will NOT issue a tamper reset.
+ Issue a power-on or pin reset to re-enable the tamper signals.
+ Press ENTER to roll the challenge to invalidate the current tamper disable token or press SPACE to exit.
```

4. Press PB0 to verify tamper sources PRS0 (Interrupt) and PRS2 (Filter) of EFR32xG21B device can still issue an interrupt.

```
. Get tamper status... SL_STATUS_OK (cycles: 11259 time: 296 us)
+ Recorded tamper status (MSB..LSB): 00050001
+ Currently active tamper sources (MSB..LSB): 00050000
```

5. The PRS5 tamper source (configured as Reset) was restored to the default (Ignore), so it cannot issue a tamper reset even if users press PB0 (Filter on PRS2) 4 times within ~32 ms x 1024.
6. The PRS4 tamper source (configured as Reset) was restored to the default (Ignore), so it cannot issue a tamper reset even if users press PB1.

7. Issue a power-on or pin reset to exit the tamper disable state or press ENTER to roll the challenge.

```
. Check and roll the challenge.
+ Request current challenge from the SE... SL_STATUS_OK (cycles: 0 time: 0 us)
+ The current challenge (16 bytes):
  AA C1 79 FC FC C5 78 8E A0 3F 91 AB 5D A9 C5 04
+ Rolling the challenge... SL_STATUS_OK (cycles: 0 time: 0 us)
+ Request rolled challenge from the SE... SL_STATUS_OK (cycles: 0 time: 0 us)
+ The rolled challenge (16 bytes):
  0F 63 9C 44 46 E4 7C B2 C9 CA 66 13 34 34 92 8E
+ Issue a power-on or pin reset to activate the rolled challenge.

. SE manager deinitialization... SL_STATUS_OK (cycles: 0 time: 0 us)
```

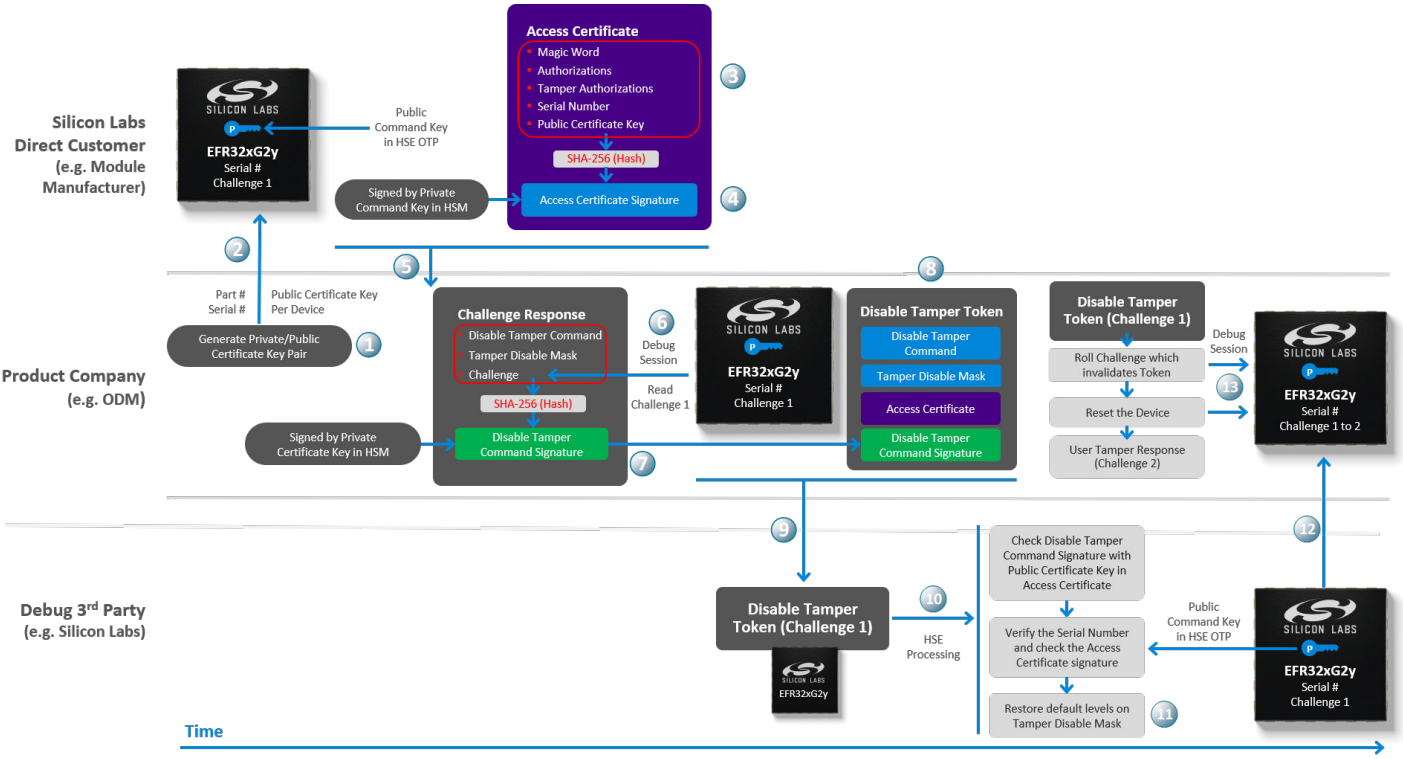
9.3.2 Simplicity Commander

The tamper disable was designed with three organizations in mind:

1. The Direct Customer to whom Silicon Labs sells the chip. This chip has the Public Command Key installed in the SE OTP.
2. The Product Company is a customer of the Direct Customer. This is applicable if the Direct Customer is creating a white-labeled product for another company or a sub-component that goes into another company's product.
3. The Debug 3rd Party could be anyone, internal or external, that the Product Company decides is qualified to debug the device.

Because the Public Command Key is installed into the SE OTP of a large number of devices and cannot be changed, the corresponding Private Command Key must be guarded by a very stringent process. If this Private Command Key is ever leaked, all the devices programmed with the corresponding Public Command Key will be compromised.

A tamper disable use case is described in the following figure, and the signing process is performed by a Hardware Security Module (HSM).



The tamper disable flow moving across the time axis from left to right is explained below:

1. The Product Company creates a Private/Public Certificate Key pair for each device. Because the key pair is assigned only to a single device, the company may not need to protect the Private Certificate Key as securely as the Private Command Key by the Direct Customer.

   In this example, the Private/Public Certificate Key pair (`cert_key.pem` and `cert_pubkey.pem`) is generated by running the `util genkey` command.

```
commander util genkey --type ecc-p256 --privkey cert_key.pem --pubkey cert_pubkey.pem
```

```
Generating ECC P256 key pair...
Writing private key file in PEM format to cert_key.pem
Writing public key file in PEM format to cert_pubkey.pem
DONE
```

2. The Public Certificate Key (`cert_pubkey.pem`) for each device is passed to the Silicon Labs Direct Customer. The part number and serial number are also required if Direct Customer cannot access the device.

   Run the `security status` command to get the device serial number. The `--serialno` option is for the J-Link serial number of the WSTK.

```
commander security status --device EFR32MG21B010F1024 --serialno 440030580
```

```
SE Firmware version   : 1.2.9
Serial number         : 000000000000000014b457fffe0f77ce
Debug lock            : Disabled
Device erase          : Enabled
Secure debug unlock   : Disabled
Tamper status         : OK
Secure boot           : Disabled
Boot status           : 0x20 - OK
Command key installed : True
Sign key installed    : False
DONE
```

3. The Direct Customer then places that Public Certificate Key in the access certificate. The access certificate is per device because it contains the unique device serial number. This certificate is generated once upon creation of the device, and thereafter, is generally only modified when the Private/Public Certificate Key pair is changed by the Product Company.

   The following two steps are **OPTIONAL** for customization of Authorizations and Tamper Authorizations.

   a. (**Optional**) Run the `security genauth` command to generate the default certificate authorization file (`certificate_authorization.json`).

```
commander security genauth -o certificate_authorizations.json --nostore --serialno 440030580
```

```
DONE
```

   b. (**Optional**) Use a text editor to modify the default Authorizations and Tamper Authorizations in the `json` file.

   Run the `security gencert` command with the following parameters from the Product Company to generate an unsigned access certificate (`access_certificate.extsign`) in Security Store:

   - Device part number
   - Device serial number
   - Public Certificate Key

```
commander security gencert --device EFR32MG21B010F1024 --deviceserialno 000000000000000014b457fffe0f77ce
--cert-pubkey cert_pubkey.pem --extsign
```

```
Authorization file written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/device_000000000000000014b457fffe0f77ce/
certificate_authorizations.json
Cert key written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/device_000000000000000014b457fffe0f77ce/
cert_pubkey.pem
Created an unsigned certificate in Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/device_000000000000000014b457fffe0f77ce/
access_certificate.extsign
DONE
```

   Note:

   - The `--extsign` option to create an unsigned access certificate is only available in Simplicity Commander Version 1.11.2 or above.
   - The unsigned access certificate is generated with the default certificate authorization file (`certificate_authorization.json`) which uses `0x0000003e` for Authorizations and `0xffffffb6` (HSE-SVH device) for Tamper Authorizations (Table 8.2 Elements of the Access Certificate on page 15).
   - **(Optional)** Use `--authorization` option if the customized `json` file generated in the above optional steps (a) and (b) is used.

```
commander security gencert --device EFR32MG21B010F1024 --authorization certificate_authorizations.json
--deviceserialno 000000000000000014b457fffe0f77ce --cert-pubkey cert_pubkey.pem --extsign
```

4. The signing of the access certificate can be done by passing an unsigned access certificate to a Hardware Security Module (HSM) containing the Private Command Key.

   In this example, the OpenSSL is used to sign the access certificate (`access_certificate.extsign`) in Security Store with the Private Command Key (`command_key.pem`). The access certificate signature is in the `cert_signature.bin` file.

```
openssl dgst -sha256 -binary -sign command_key.pem -out cert_signature.bin access_certificate.extsign
```

   Run the `util signcert` command with the following parameters to verify the signature and generate the signed access certificate (`access_certificate.bin`):
   - Unsigned access certificate
   - Access certificate signature
   - Public Command Key

```
commander util signcert access_certificate.extsign --cert-type access --signature cert_signature.bin
--verify command_pubkey.pem --outfile access_certificate.bin
```

```
R = 76CDC5BA18E5248FDA5418002F250F149B449829A005D6F0726268016CC53ED4
S = E4B8ABA2CF742B0E6CC5BA2C1023D76BEEF3C4A11DA97CC4D23459F32237A206
Successfully verified signature
Successfully signed certificate
DONE
```

   Note:
   - Put the required files in the same folder to run the command.
   - The `util signcert` command for access certificate is only available in Simplicity Commander Version 1.11.2 or above.
   - The access certificate signature can be in a Raw or Distinguished Encoding Rules (DER) format.

5. The access certificate is passed to the Product Company. The purpose of the access certificate is to grant overall debug access capabilities to the Product Company and authorize them to allow third parties to debug the device. The Product Company can now use the access certificate to generate the Disable Tamper Token. The same access certificate can be used to generate as many Disable Tamper Tokens as necessary without having to ever go back to the Direct Customer.
6. To create the Disable Tamper Token, a debug session must be started with the device and the challenge value (which is a random number `Challenge 1` in this example) should be read out to generate the challenge response.

   Run the `security gencommand` command to generate the challenge response without disable tamper command signature and store it in a file (`command_unsign.bin`).

```
commander security gencommand --action disable-tamper --disable-param 0x00fa0000 -o command_unsign.bin
--nostore --device EFR32MG21B010F1024 --serialno 440030580
```

```
Unsigned command file written to:
command_unsign.bin
DONE
```

   Note:
   - The tamper disable mask (`0x00fa0000`) is based on the Tamper platform example on EFR32xG21B devices (Table 9.2 PRS Tamper Source Usage on EFR32xG21B Devices on page 31).
   - If the `--disable-param` option is not provided, it will restore all tamper sources (`0xfffffb6`) by default.

7. The challenge response is then cryptographically hashed (SHA-256) to create a digest. The digest is then signed by the Private Certificate Key to generate the disable tamper command signature.

   The signing of the challenge response can be done by passing an unsigned challenge response to a Hardware Security Module (HSM) containing the Private Certificate Key.

   In this example, the OpenSSL is used to sign the challenge response (`command_unsign.bin`) with the Private Certificate Key (`cert_key.pem`). The disable tamper command signature is in the `command_signature.bin` file.

```
openssl dgst -sha256 -binary -sign cert_key.pem -out command_signature.bin command_unsign.bin
```

8. Run the `security disabletamper` command with the access certificate (`access_certificate.bin`) from Direct Customer and disable tamper command signature (`command_signature.bin`) in step 7 to generate the Disable Tamper Token.

```
commander security disabletamper --disable-param  0x00fa0000 --cert access_certificate.bin
--command-signature command_signature.bin EFR32MG21B010F1024 --serialno 440030580
```

```
Certificate written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_000000000000000014b457fffe0f77ce/access_certificate.bin
R = A70834D97640A92510D151765F0EED6C6A05CB8BE81E06E905C230ED24E71659
S = 9B69C113C2B7DEE60BF0BC7D72719F7F9465840D68EADBBB4F9BCE7A1267B936
Command signature is valid
Tamper successfully disabled.
Command disable tamper payload was stored in Security Store
DONE
```

Note:

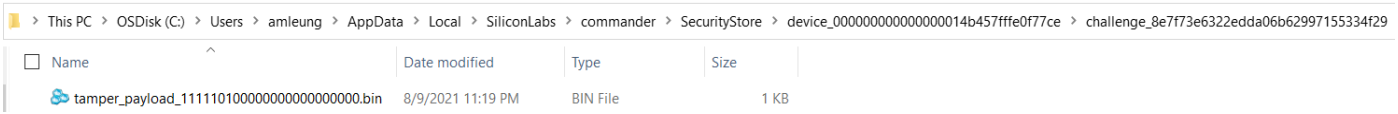
- Put the required files in the same folder to run the command.
- The disable tamper command signature can be in a Raw or Distinguished Encoding Rules (DER) format.
- Simplicity Commander Version 1.11.2 or above is required to support signature in DER format.

9. **(Alternative)** Key protection is not required if the Private Certificate Key is ephemeral. Steps 6 to 8 can be implemented by running the `security disabletamper` command with the access certificate (`access_certificate.bin`) from the Direct Customer and Private Certificate Key (`cert_key.pem`) to generate the Disable Tamper Token.

```
commander security disabletamper --disable-param 0x00fa0000 --cert access_certificate.bin --cert-privkey
cert_key.pem --device EFR32MG21B010F1024 --serialno 440030580
```

```
Certificate written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_000000000000000014b457fffe0f77ce/access_certificate.bin
Cert key written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_000000000000000014b457fffe0f77ce/cert_pubkey.pem
Created unsigned disable tamper command
Signed disable tamper command using
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_000000000000000014b457fffe0f77ce/cert_key.pem
Tamper successfully disabled.
Command disable tamper payload was stored in Security Store
DONE
```

10. The Disable Tamper Token (aka `Command disable tamper payload`) file (`tamper_payload_11111010000000000000000.bin`, where `11111010000000000000000` is `0x00fa0000` for tamper disable mask) is stored in the Security Store. The location in Windows is `C:\Users\<PC user name>\AppData\Local\SiliconLabs\commander\SecurityStore\device_<Serial number>\challenge_<Challenge value>`.



Users can also use the `security getpath` command to get the path of the Security Store or a specified device.

```
commander security getpath
```

```
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore
DONE
```

```
commander security getpath --deviceserialno 000000000000000588e81fffe70350d
```

```
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_000000000000000588e81fffe70350d
DONE
```

11. The Disable Tamper Token and the device are now delivered to the Debug 3rd Party.

    Run the `security gencommand` command to create the Security Store to place the Disable Tamper Token file.

```
commander security gencommand --action disable-tamper --disable-param 0x00fa0000
--device EFR32MG21B010F1024 --serialno 440030580
```

```
Unsigned command file written to Security Store:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_00000000000000014b457fffe0f77ce/challenge_8e7f73e6322edda06b62997155334f29/
disable_tamper_command_to_be_signed09_08_2021.bin
DONE
```

    Copy the Disable Tamper Token file (`tamper_payload_11111010000000000000000.bin`) from Product Company to the Windows Security Store `challenge_<Challenge value>` folder located in `C:\Users\<PC user name>\AppData\Local\SiliconLabs\commander\SecurityStore\device_<Serial number>\challenge_<Challenge value>`.

12. The device compares the Disable Tamper Token contents with its internal serial number, challenge value, and Public Command Key to determine the token's authenticity. If authentic, it will execute the disable tamper command to restore the default levels on the tamper disable mask (`0xfa000000`); otherwise, it will ignore the command.

    Run the `security disabletamper` command to disable the tamper.

```
commander security disabletamper --disable-param 0x00fa0000 --device EFR32MG21B010F1024
--serialno 440030580
```

```
Disabling tamper with tamper payload:
C:/Users/<username>/AppData/Local/SiliconLabs/commander/SecurityStore/
device_00000000000000014b457fffe0f77ce/challenge_8e7f73e6322edda06b62997155334f29/
tamper_payload_11111010000000000000000.bin
Tamper successfully disabled.
DONE
```

    Note: Users can verify the Disable Tamper Token by following steps 4 to 6 in 9.3.1.2 Tamper Disable if the EFR32xG21B device is running in the Normal mode of the SE Manager Tamper platform example.

13. The Debug 3rd Party can now use this same Disable Tamper Token to disable the tamper (step 12), over and over again after each power-on or pin reset, until they have finished debugging the device.
14. Once the Debug 3rd Party has finished debugging, they will send the device back to the Product Company.

15. Once the Product Company receives the device, they will immediately start a debug session to roll the challenge (from `Challenge 1` to `Challenge 2` in this example). Rolling the challenge will effectively invalidate any Disable Tamper Token that has been previously given to any third party.

    Run the `security rollchallenge` command and reset the device to invalidate the current Disable Tamper Token. The challenge cannot be rolled before it has been used at least once — that is, by running the `security disabletamper` or `security unlock` command.

```
commander security rollchallenge --device EFR32MG21B010F1024 --serialno 440030580
```

```
Challenge was rolled successfully.
DONE
```

    The unlock token is invalidated after rolling the challenge because any previously issued Disable Tamper Token now contains a different challenge value (`Challenge 1`) than the challenge value currently in the device (`Challenge 2`).

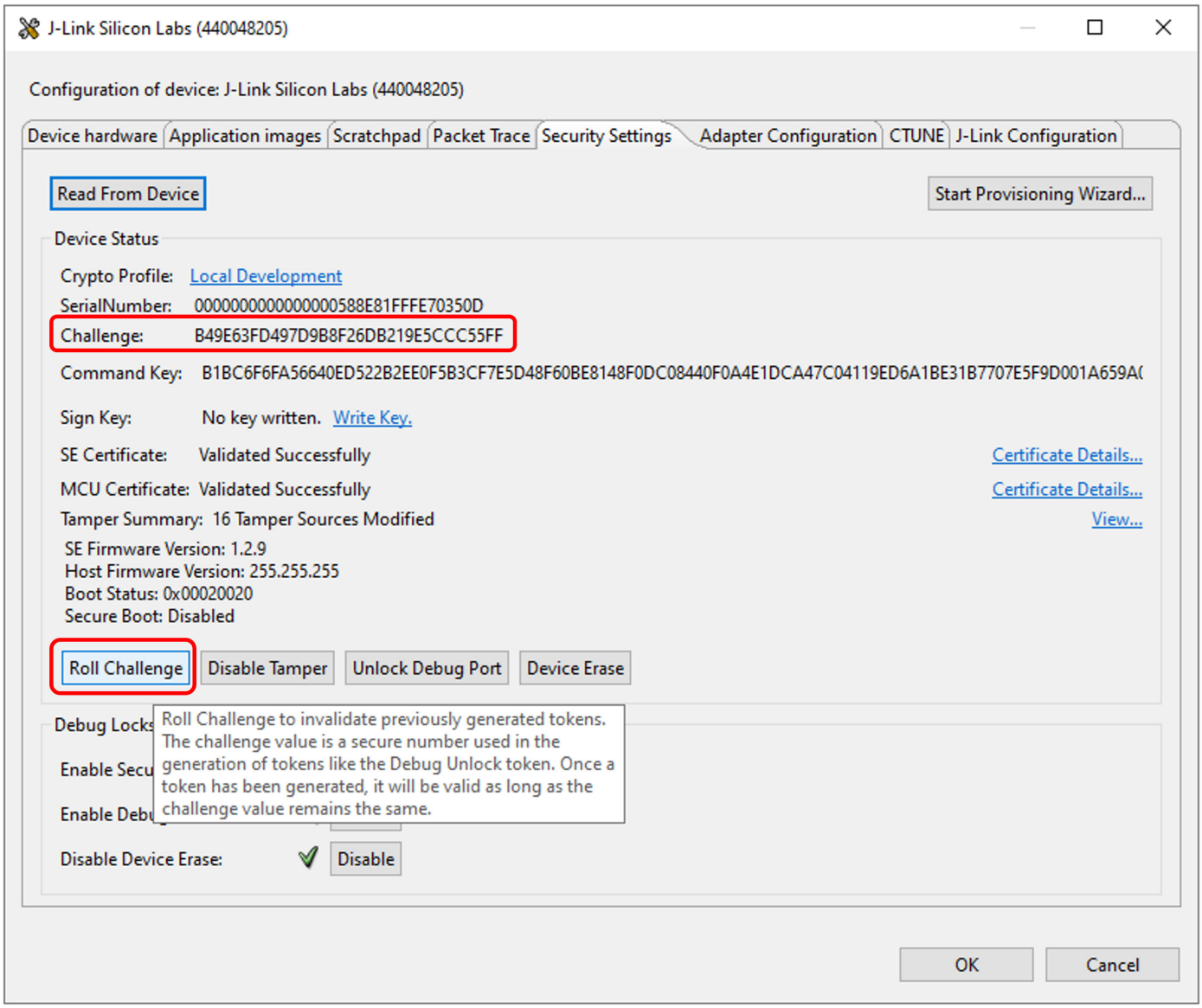
    The validation process of any previously issued Disable Tamper Token will always fail until a new Disable Tamper Token is issued with a current matching challenge value (`Challenge 2`).

Note: Direct Customer can directly use the Private Command Key on the connected chip to generate the Disable Tamper Token in Security Store. But it has a high risk (cannot use HSM) to leak the Private Command Key to a 3rd party when using this approach.

```
commander security disabletamper --disable-param 0x00fa0000 --command-key command_key.pem
--device EFR32MG21B010F1024 --serialno 440030580
```

9.3.3 Simplicity Studio

1. Open **Security Settings** of the selected device as described in 9.1.3 Using Simplicity Studio.
2. Click [**Roll Challenge**] to generate a new challenge value to invalidate the Disable Tamper Token for tamper disable. Click [**OK**] to exit.



10. Revision History

Revision 0.7

February, 2024

- Improved the descriptions of the tamper sources in Table 5.1 Tamper Sources on the EFR32xG21B Devices on page 8 and Table 5.2 Tamper Sources on Other HSE-SVH Devices on page 9.
- Updated Table 5.2 Tamper Sources on Other HSE-SVH Devices on page 9 to indicate tamper sources available in EM2.
- Added a note after Table 5.2 Tamper Sources on Other HSE-SVH Devices on page 9 to clarify the behavior of EFx32xG23B and later devices.
- Added the descriptions of sl_se_enter_active_mode and sl_se_exit_active_mode APIs to Table 3.1 SE Manager API for Tamper Operations on page 5.

Revision 0.6

June 2023

- Added ETAMPDET information to first page.
- Added ETAMPDET information to Introduction.
- Added `sl_se_get_tamper_reset_cause` to Table 3.1 SE Manager API for Tamper Operations on page 5.
- Updated note in 4. Tamper Responses.
- Added example filter tamper configuration in Filter.
- Updated Table 5.2 Tamper Sources on Other HSE-SVH Devices on page 9 for HSE-SVH devices with an ETAMPDET peripheral.
- Updated Table 6.1 Anti-Tamper Configuration on page 11 and note to clarify the user-defined level.
- Updated 7. Usage Example.
- Updated figures Figure 8.1 Tamper Disable on the EFR32xG21B Devices on page 13 and Figure 8.2 Tamper Disable on Other HSE-SVH Devices on page 13.
- Added a note to Tamper Disable Flow.
- Reordered Overview sections.
- Updated 9.1.2 Using Simplicity Commander to v1.14.6.
- Updated 9.2 Provision Public Command Key and Tamper Configuration.
- Moved program the default Public Command Key steps in 9.2.1 SE Manager - Tamper Platform Example to 9.3.1.2 Tamper Disable.
- Updated console outputs in 9.2.2 Simplicity Commander steps 4 and 6.
- Added a note to 9.2.2 Simplicity Commander step 4 and 9.2.3 Simplicity Studio step 7.
- Updated Table 9.3 PRS Tamper Source Usage on Other HSE-SVH Devices on page 31 for HSE-SVH devices with an ETAMPDET peripheral.
- Move Normal and Tamper Disable sections in 9.3 Tamper Disable and Roll Challenge to 9.3.1 SE Manager - Tamper Platform Example.
- Updated 9.3.2 Simplicity Commander.

Revision 0.5

August 2022

- Updated table and note in 1. Series 2 Device Security Features.
- Replaced Device Compatibility with 1.3 SE Firmware in 1. Series 2 Device Security Features.
- Updated figures in 9. Examples.
- Updated to GSDK v4.1.0 in 9.1.1 Using a Platform Example.
- Updated 9.2.3 Simplicity Studio in 9.2 Provision Public Command Key and Tamper Configuration.
- Added `security getpath` command to 9.3.2 Simplicity Commander.

Revision 0.4

March 2022

- Added digit 4 to Note 3 in 1. Series 2 Device Security Features.
- Updated Device Compatibility and moved it under 1. Series 2 Device Security Features.

Revision 0.3

January 2022

- Added UG489 to the table in 1.2 Key Reference.
- Added CPMS information to 2. Introduction.
- Updated note in 5. Tamper Sources.
- Added Keep Tamper Alive During Sleep flag to Table 6.1 Anti-Tamper Configuration on page 11.
- Updated to GSDK v4.0.1 in 9.1.1 Using a Platform Example.
- Updated note for Simplicity Commander support of tamper configuration on HSE-SVH devices in 9.2.2 Simplicity Commander.
- Replaced `gbl keyconvert` with `util keytotoken` in 9.2.3 Simplicity Studio.
- Moved Push button PB0 from PRS0 (25) to PRS1 (26) in Table 9.3 PRS Tamper Source Usage on Other HSE-SVH Devices on page 31 and updated note 1.
- Updated note in step 10 of 9.3.2 Simplicity Commander.

Revision 0.2

September 2021

- Formatting updates for source compatibility.
- Added revised terminology to 1. Series 2 Device Security Features and use this terminology throughout the document.
- Updated Device Compatibility.
- Updated 3. Secure Engine Manager for EMU->RSTCAUSE register on other HSE-SVH devices.
- Updated 4.1 Interrupt for the SEMAILBOXHOST clock on other HSE-SVH devices.
- Added Table 5.2 Tamper Sources on Other HSE-SVH Devices on page 9.
- Added Figure 8.2 Tamper Disable on Other HSE-SVH Devices on page 13.
- Replaced Disable Tamper Command with 8.1 Disable Tamper Token.
- Revised 9. Examples to the latest Simplicity Studio and Simplicity Commander version, updated the content.

Revision 0.1

September 2020

- 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

www.silabs.com