



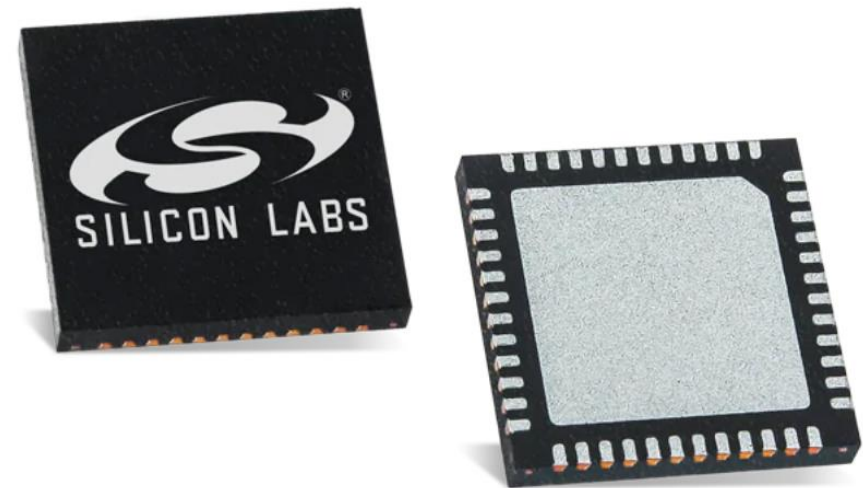
SEC-301: Hands on with CPMS Security

Benjamin Thorell

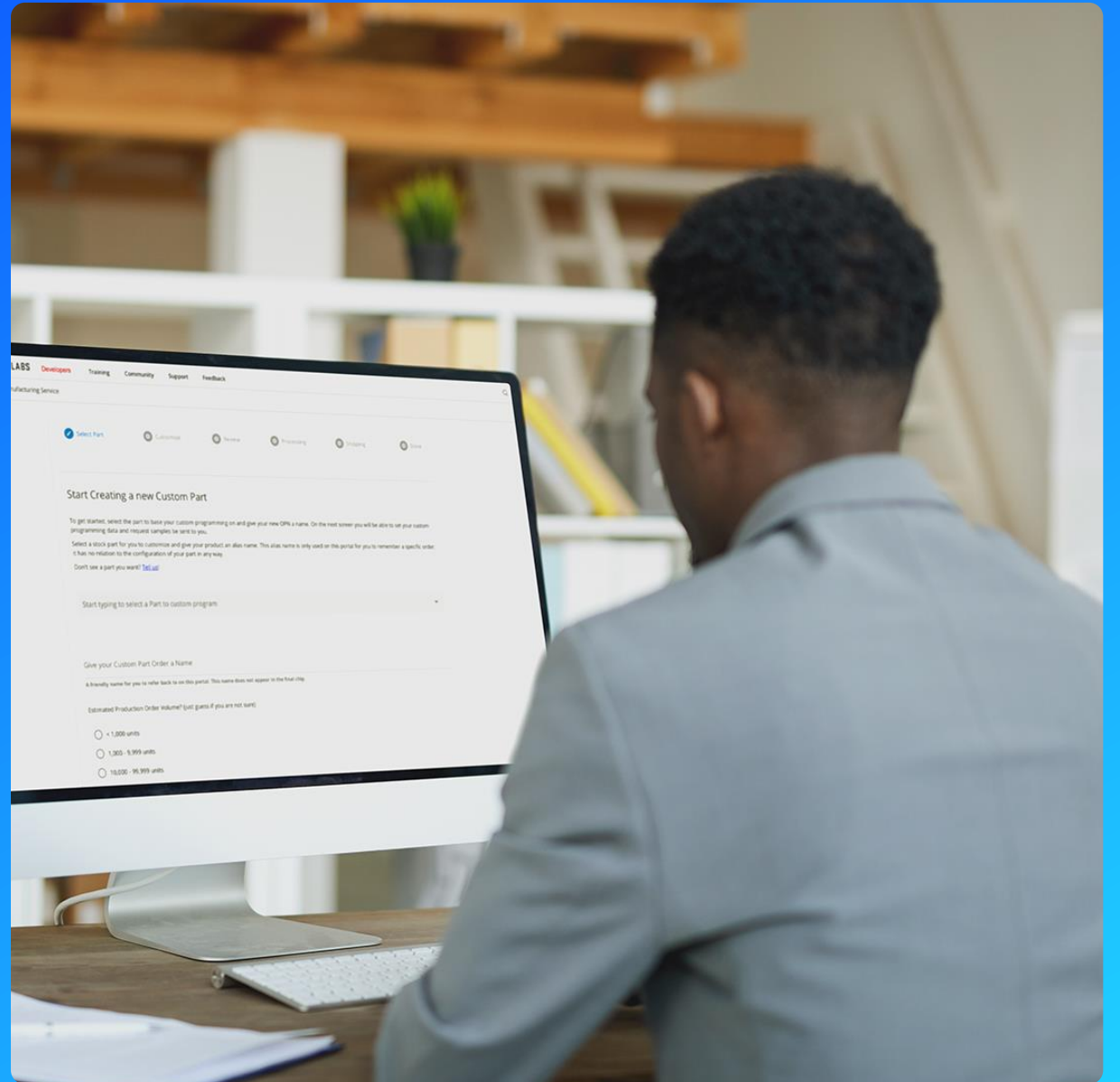


CPMS

- **What is CPMS?**
 - CPMS (Custom Part Manufacturing Service) is a service offered by Silicon Labs that allows you to order custom parts that have your firmware and security settings programmed into them before they are sent to the CM
- **Why is this important?**
 - IoT security is complex, and it's easy to accidentally leave a system vulnerable. CPMS provides a “checklist” of easily enabled security features
 - IoT devices are at their most vulnerable during production. CPMS allows you to secure your parts from the moment they're programmed
- **Where is it?**
 - <https://cpms.silabs.com/>



Customization Options



SE Version

- **CPMS allows you to select the Secure Element firmware version that is programmed into your custom parts**
 - We recommend using the latest SE version to ensure all patches are in place

SE Version v1.2.7 (latest)

SE Version
v1.2.7

We recommend using the latest SE version to ensure all patches are in place. We further recommend that you implement the ability to apply SE updates in your manufacturing line and over the air in the event new vulnerabilities are patched.

Debug Lock

- **CPMS allows you to select the state of the debug lock when the part is shipped to the CM**
- **Series 2 devices have 4 options for the debug lock:**
 - Permanent – the debug port is locked and cannot be unlocked
 - Standard – the debug port is locked, but it can be unlocked with a full flash erase
 - Secure – the debug port is locked, but it can be unlocked with a full flash erase or with a debug unlock token. The debug unlock token is verified with a public key stored in the device, and it only unlocks the debug port until the next reset
 - Unlocked – the debug port is unlocked

Debug Lock

Standard Secure Permanent Unlocked

The debug access port connected to the Series 2 device's Cortex-M33 processor can be closed by issuing commands to the Secure Element, either from a debugger over DCI or through the mailbox interface. Three properties govern the behavior of the debug lock. Locking the part reduces the general attack surface and prevents information leakage post Silicon Labs manufacturing.

Initialize OTP Settings

- **CPMS allows you to configure OTP security settings. Since these settings are One Time Programmable, once set, they cannot be cleared**
 - *Enable Secure Boot* requires that any code on the device must have a valid signature or certificate in order to run. This ensures that only approved code runs on the device.
 - *Require Verify Certificate before secure boot* requires that certificates be used in the Secure Boot chain, rather than direct signing. This reduces the need to access the private key corresponding to the signing public key on the device.
 - *Enable Anti Rollback* prevents applications from “updating” to older (potentially vulnerable) versions of the firmware
 - *Flash Page Locking* prevents applications from writing to certain flash pages

Configure Secure Boot, Flash Lock, and Tamper Settings

These configurations can only be made at one time and are irreversible once they are made.

Read more about [secure boot with RTSL](#) and [production programming](#)

Enable Secure Boot with RTSL

If set, authenticates the first code image in flash memory, which is typically the second stage bootloader, before allowing that code to run. Enabling secure boot will ensure that the device will only boot code that has been properly signed by you.

Require Verify Certificate before secure boot

The Verify intermediate certificate before secure boot option provisions the Public Sign Key to enable certificate-based Secure Boot. Enabling this reduces the need to access the OTP signing key allowing more stringent access restrictions. It also provides the ability to roll the intermediate key in the event it is compromised.

Enable Anti Rollback

We recommend enabling anti-rollback. If set, the first stage bootloader will compare the version of the first image in flash memory, which is typically the second stage bootloader, to the version of the image that has been staged for upgrade. If the staged image has a version that is **greater than** the current image, the upgrade will succeed. Otherwise the upgrade operation will be ignored.

Flash Page Locking

None Full Narrow

This feature write/erase locks flash pages starting at 0 that have been validated by the first stage bootloader signature check. This will prevent flash modification of the locked pages by any means other than through the hardware secure engine (write/erase attempts from the CPU or from the debug port will be ignored).

"Full" - locks from page 0 up to and including the page containing the signature. This may lock flash bytes that are located after the end of the signature if the signature does not terminate at a flash page boundary.

"Narrow" - locks from page 0 up to the flash page immediately before the signature if the signature does not terminate at a flash page boundary. This may leave some of the end bytes of the image or the signature unprotected by write/erase lock if the signature does not terminate at a flash page boundary.

Note: if the signature terminates at a flash page boundary, the behavior of the "Full" setting and the "Narrow" setting are identical.

Tamper Response Configuration

- **CPMS allows you to configure responses for 27 tamper sources**
- **When a tamper source is triggered, the device can choose to either:**
 - Ignore it
 - Generate an Interrupt
 - Increment the Filter Counter
 - Trigger a System Reset
 - Erase the OTP memory (note that this will make the device and all wrapped secrets unrecoverable. After this response, the device will no longer be able to boot.)

Tamper Response Configuration

Custom tamper configuration is an advanced feature. Default configurations are usually sufficient for most cases. Note: Custom configuration or tamper disable cannot reduce the tamper response below the default Level.

[Read more about secure vault tamper](#)

SE watchdog
Internal SE watchdog expires

Ignore Generate Interrupt Increment Filter Counter **System Reset** Erase OTP

SE RAM CRC

Tamper Response Configuration – Filter Counter

- Every tamper source has the option to increment the “Filter Counter”
- The Counter resets to 0 at a pre-defined period
- Once the Counter reaches a pre-determined Trigger Threshold, the Filter Counter tamper source is triggered
- Both the Reset Period and the Trigger Threshold can be configured in CPMS

Filter Counter
Filter counter reaches configured threshold value

Ignore Generate Interrupt Increment Filter Counter System Reset Erase OTP

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 configured to use one of the trigger thresholds and reset periods given in the dropdowns below. The filter counter is reset upon a tamper reset.

Filter Reset Period
32 ms

Filter Trigger Threshold
256

Tamper Response Configuration – Other

- **CPMS also lets you configure a “Reset Threshold”**
 - If the device undergoes that many consecutive tamper resets, it will enter a safe diagnostic state
- **Lastly, CPMS allows you to configure whether the digital glitch detector runs continually or only when the SE is performing operations**

Reset Threshold
10

Number of consecutive tamper resets before entering safe diagnostic state. (0-255)

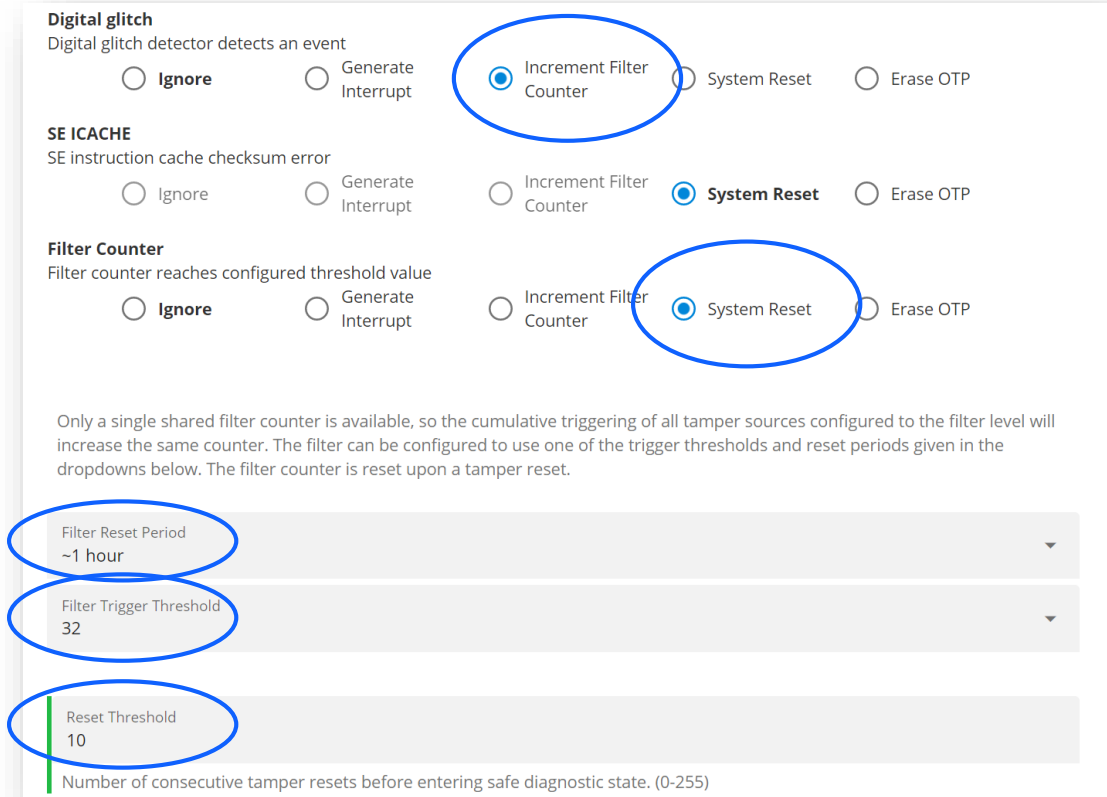
Though the default tamper configuration is not expected to generate tamper resets, we recommend that the device be configured to enter a safe state if 10 consecutive tamper resets are seen. This will ensure that if resets are seen for some reason the device will not enter an infinite reset loop which is beneficial from both a security and quality point of view. If the threshold is set to 0, the part will never enter the diagnostic mode due to tamper reset.

Digital Glitch Detector Always On

If checked, then the digital glitch detector runs continually even when the SE is not performing any operations; otherwise, the detector only runs when the SE is executing a command.

Filter Counter Example

- In the example on the right, the Digital glitch tamper source has been configured to increment the Filter Counter when it triggers
- Every ~1 hour, the Counter resets to 0
- Once the Counter reaches 32, the Filter Counter tamper source will trigger, resulting in a System Reset
- After 10 System Resets (triggered by the Filter Counter or any other tamper source), the device enters a safe diagnostic state



Digital glitch
Digital glitch detector detects an event

Ignore Generate Interrupt Increment Filter Counter System Reset Erase OTP

SE ICACHE
SE instruction cache checksum error

Ignore Generate Interrupt Increment Filter Counter System Reset Erase OTP

Filter Counter
Filter counter reaches configured threshold value

Ignore Generate Interrupt Increment Filter Counter System Reset Erase OTP

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 configured to use one of the trigger thresholds and reset periods given in the dropdowns below. The filter counter is reset upon a tamper reset.

Filter Reset Period
~1 hour

Filter Trigger Threshold
32


Reset Threshold
10

Number of consecutive tamper resets before entering safe diagnostic state. (0-255)


Standard Security Keys

- **CPMS allows you to provision standard security keys into the device**
 - The *Secure Boot Key* is a public key used as the root of trust during the secure boot process to authenticate the firmware
 - The *Command Key* is a public key used to validate Secure Debug tokens
 - The *OTA Decryption Key* is a symmetric key used for decrypting GBL firmware upgrades

Standard Security Keys

Secure Boot Key 

This key is used for binary authentication and/or OTA upgrade payload authentication. If you enabled secure boot, you must provide the public part of the key you used to sign your bootloader or application image here. (eg. 0x04123456789...ABCEDF, total 65 bytes. You can also upload a .pem or .der file)

Command Key 

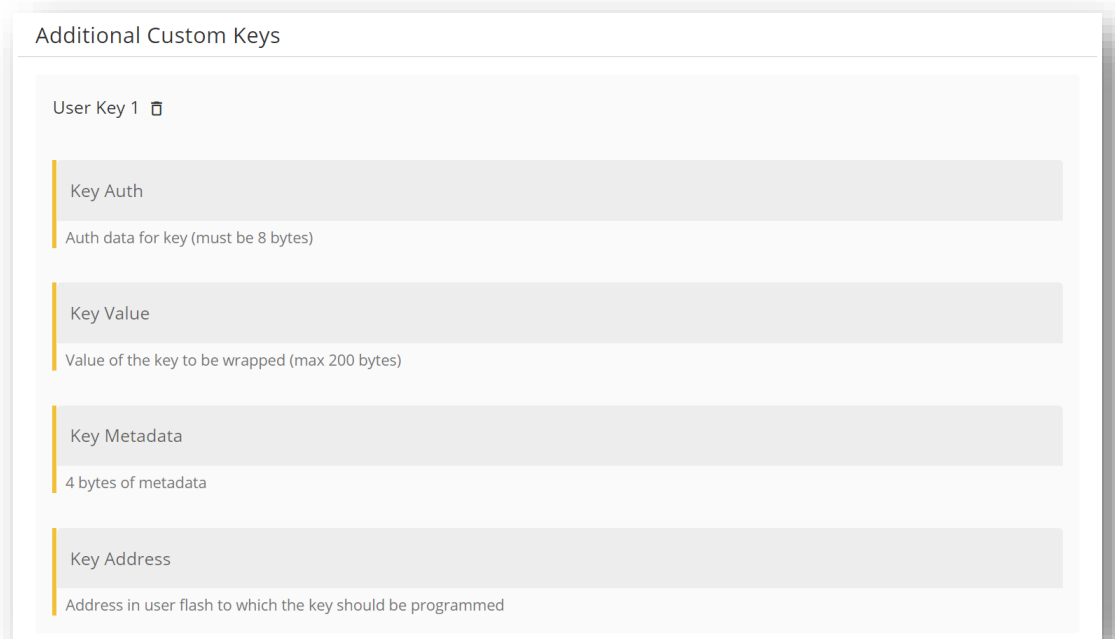
This key is used for Secure Debug Unlock or Disable Tamper command authentication. If you chose secure debug lock, you must provide the public part of your command key here. (eg. 0x04123456789...ABCEDF total 65 bytes. You can also upload a .pem or .der file)

OTA Decryption Key

This key is used for decrypting GBL payloads used for firmware upgrades. (eg. 0x0123456789...ABCEDF total 16 bytes.)

Custom Keys

- In addition to the standard Security Keys, CPMS allows you to provision custom keys
- These custom keys will be wrapped by the Secure Element, then stored at a specified address in user flash
- To provision a custom key, you must provide:
 - *Key Value* – the value of the key to be wrapped
 - *Key Address* – the address where the wrapped key will be stored
 - *Key Metadata* – a 32-bit key specification used by the SE (this value can be generated from a key descriptor using `sli_se_key_to_keyspec`)
 - *Key Auth* – an 8-byte password used to allow access to the wrapped key



Additional Custom Keys

User Key 1

Key Auth
Auth data for key (must be 8 bytes)

Key Value
Value of the key to be wrapped (max 200 bytes)

Key Metadata
4 bytes of metadata

Key Address
Address in user flash to which the key should be programmed

Custom Identity

- CPMS allows you to specify how to incorporate your own certificate chains into the Silicon Labs cert chain
- Cert chain implementations vary by use case, so certificate field details should be provided in the “Special Instructions” section

Custom Identity

Custom Identity allows customers to extend the default Silicon Labs certificate identity cert chain to provide your own. This is an advanced feature which requires additional charges. Please contact a Silicon Labs sales representative for details.

[Read more about secure identity](#)

Scope of Customization

Device certificate only The certificate chain

Special Instructions

Tell us how you would like to customize the identity of this part. (2000/2000 remaining)

Flash Programming

- CPMS allows you to program your application and/or bootloader into the device before it is sent to the CM
- The Fill character can be specified to aid in detecting memory corruption

Firmware

Fill Character
0x FF

We will fill unused or unspecified addresses of the flash with the byte you provide here.

Firmware Type
 App only Bootloader only App and Bootloader

[CLICK HERE OR DRAG DROP TO UPLOAD A FILE](#)

Intel HEX

Custom Marking

Custom Marking

Custom marking involves the modification of the marking on integrated circuits from the standard marking. All marking requests are subject to Silicon Labs approval. Additional customer specifications may also be submitted as an attachment. **Custom marking changes are limited to alpha-numeric characters and not any existing pre-marked Silicon Laboratories logo.**

Upon acceptance of this request, Silicon Labs will create and email a custom factory marking specification for customer approval. If custom marking is requested in addition to any custom programming / serialization, the First Article Samples will have the standard marking with the custom programming/serialization. Full production can begin only after this process is complete.

Custom marking is not available for First Article Samples. All custom marking is subject to special terms & conditions for any orders accepted. **Minimum order quantities will be 4500 pieces per order line item for all custom marked parts.** Delivery lead-times will be longer.

Marking Line 1

- Default (Per Mark Spec)
- Custom

Other Customizations

- If there are any other customizations that you would like added to the device, you can use the “Other Customizations” option

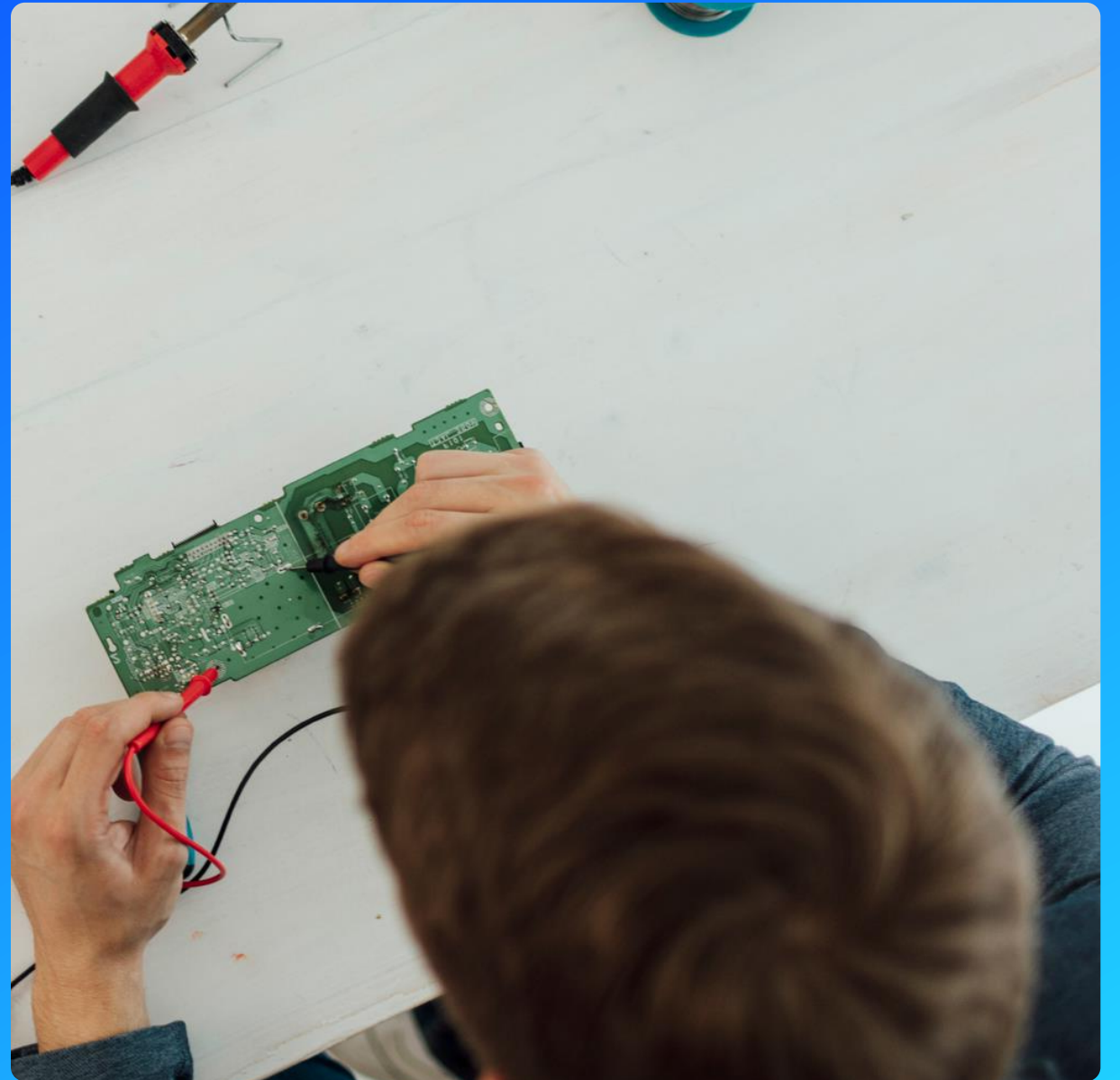
Other Customizations

This is an advanced feature that requires engineer vetting and will incur extra charges. Examples include: custom limits, custom OUIs, and other kinds of custom testing.

Special Instructions

Tell us how you would like to customize the testing of this part. (1000/1000 characters remaining)

Hands-On Portion



Lab Overview

- **Part 1 – A simple part with a pre-flashed user application**
 - Debug Lock – Unlocked
 - No OTP configuration
 - Flash application
- **Part 1.2 – Improving security using Standard Lock**
 - Debug Lock – Standard
 - Add bootloader
- **Part 2 – A part secured against untrusted CMs**
 - Debug Lock – Standard
 - Secure Boot enabled
 - Flash “direct signed” application and bootloader
- **Part 2.2 – Improving security using Secure Lock**
 - Debug Lock – Secure
- **Part 3 – A secure part using certificate chains**
 - Debug Lock – Secure
 - All (non-tamper) OTP configurations set
 - Flash “certificate signed” application and bootloader

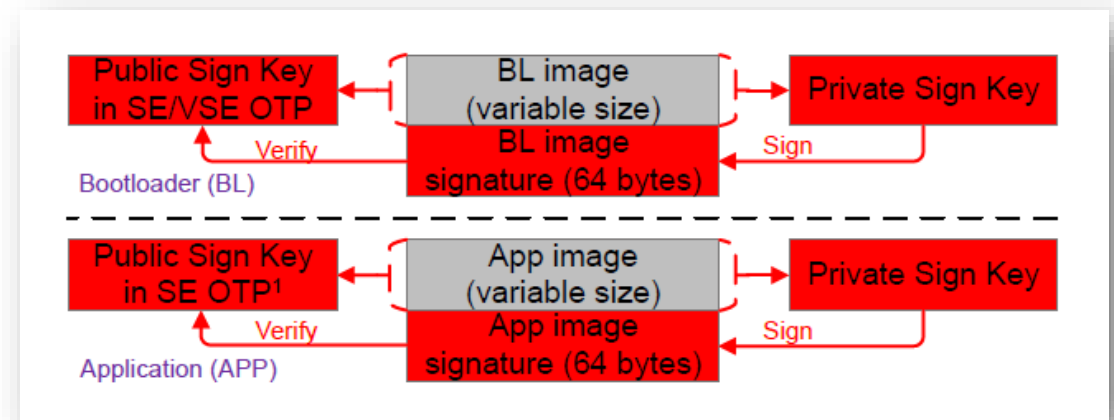
Start Creating a new Custom Part

Silicon Labs Custom Part Manufacturing Service (CPMS) lets you configure your own custom parts. As part of the customization process, we will send you samples for approval, and once approved, you will receive a unique Orderable Part Number (OPN) that you can use to order commercial quantities of your part from your Silicon Labs sales representative or authorized distributor.

[Create a new Custom Part](#)

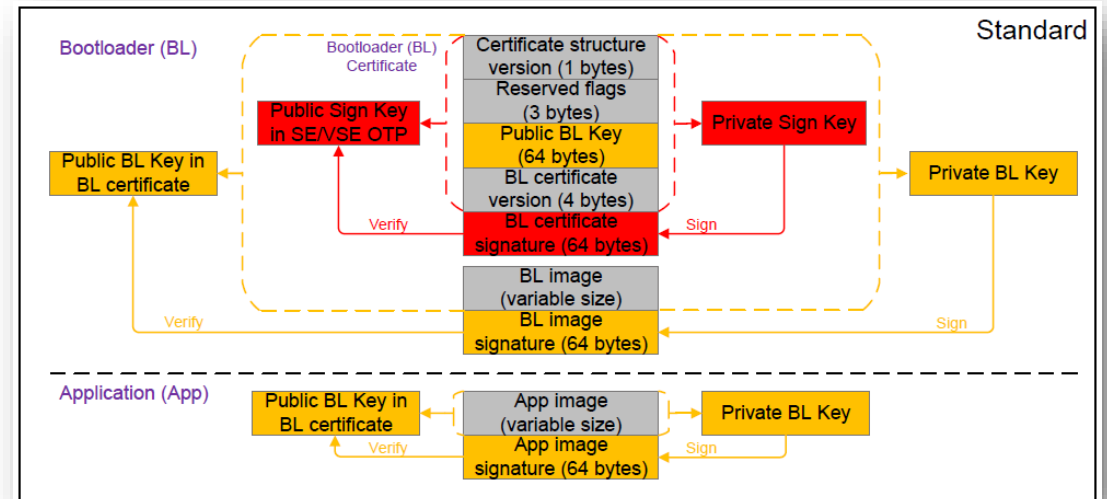
Direct Signing

- With direct signing, every bootloader and every application is signed with the Private Sign Key
- The Public Sign Key is provisioned in OTP memory on the device
- For 1000 applications, this would require using the Private Sign Key 1000 times



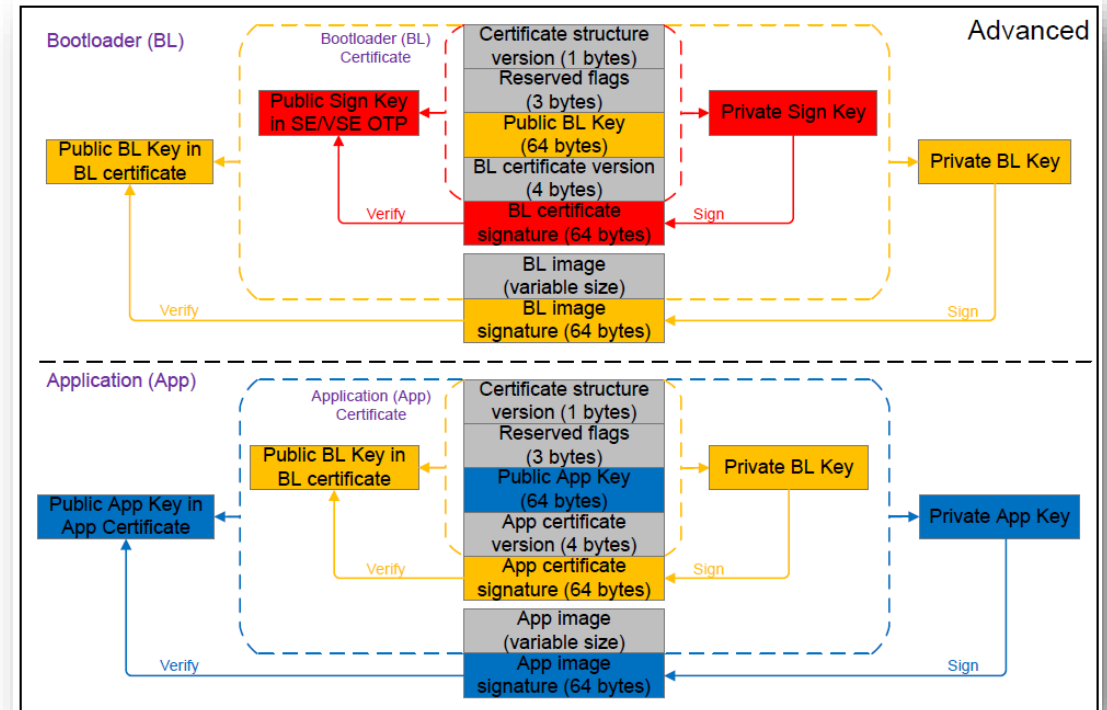
Standard Certificate Signing

- With certificates, the Private Sign Key is used to sign the bootloader certificate
- The bootloader and app are signed using a second key, the Private BL Key
- The bootloader certificate contains the Public BL Key, which is used to verify the bootloader and app



Advanced Certificate Signing

- With advanced certificates, the application also receives a certificate
- The Private BL Key is used to sign the App certificate, which contains the Public App Key
- The Private App Key is used to sign the App image
- For 1000 applications, this would require each private key be used ~10 times



On to the lab...

