TEXAS INSTRUMENTS

Application Report

SWRA651 – April 2019

Secure Boot in SimpleLink™ CC13x2/CC26x2 Wireless MCUs

Bhargavi Nisarga and Jeffrey Stanton

ABSTRACT

Secure boot is a security enabler offered on embedded processors to verify the integrity and authenticity of the code to be executed on the processor at the time of device booting process. This application report will first discuss the need for secure boot in microcontrollers (MCUs) and the root of trust components essential for supporting this security feature. Next, the application report covers SimpleLink CC13x2/CC26x2 wireless MCU hardware security features and the software development kit (SDK) components, (including the boot image manager, BIM) that enables the secure boot process on these wireless MCUs. An overview of the SimpleLink CC13x2/CC26x2 secure boot features enabled by the BIM and the supported tools are also discussed here.

Contents

List of Figures

List of Tables

Trademarks

SimpleLink is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

In embedded systems, secure boot is a security process that is widely implemented in microprocessors and microcontrollers that do not have on-chip, execute in place (XiP) flash memory. These microprocessors require storing the application software in external persistent storage memory which is loaded onto the device volatile random access memory (RAM) or an external DDR during the device boot process. In order to ensure that the application software to be loaded and executed from the external memory is not compromised, the root of trust entities on these processors (typically a ROM bootcode) verify the authenticity of the external memory software before executing it on-chip.

MCUs that have integrated flash memory, on the other hand, typically store the application software in internal non-volatile memories with the central processing unit (CPU) executing the application software upon exit of the boot routine. The authentication of the on-chip application software image during the device boot routine and prior to the execution of the application software is referred to as “secure boot”.

Updating application software in internal flash on these microcontrollers requires either having the device JTAG access enabled , a firmware update mechanism in place (using a bootloader) or an application function that performs on-chip flash updates (data logging). These on-chip flash update mechanisms can be controlled by disabling JTAG access to the device, verifying new firmware image authenticity prior to updating application software on-chip and marking flash memories holding data as non-executable. However, despite these controlled flash update mechanisms used in MCUs with internal flash memory, secure boot is becoming increasingly important for the following reasons:

- This routine establishes a secure known starting point upon device boot
- Any persistent malware injected into the device is detected during boot routine and appropriate measures are taken (not starting the device or entering a safe/recovery mode)
- In cases where the MCUs allow any un-verified programming of on-chip memory (via a serial bootloader or application-driven functions) secure boot verifies integrity and validity of the code executed on-chip.

Also, in some applications, the MCU performing secure boot is eventually responsible for verifying authenticity of all other processors in the system as part of the system boot routine. The secure boot measurement can also be used to remotely attest device firmware/software, if needed.

When designing system security keeping defense-in-depth in mind, it is important to note that secure boot is one of the security layers. This should be used in conjunction with other security layers to enable a holistic security solution. Secure boot provides application software verification to establish a stable starting point during the device boot routine. However, secure boot is not intended to mitigate threats related to debug security or firmware updates or runtime security that can potentially enable adversaries to alter device’s runtime operation. The system designer should consider additional security defense mechanisms like enabling debug-lock, secure firmware updates, trusted execution environment and others, depending on the threat vectors that are pertinent to the system.

1.1 Cryptography Fundamentals

The secure boot process in MCUs is very similar to process in microprocessors, except that the software first being verified is stored in persistent, on-chip flash memory versus an off-chip memory. Secure boot uses cryptographic hashes and asymmetric cryptography to generate digital signatures which are used to determine the authenticity of the software. A cryptographic hash function is a mathematical algorithm that maps data of arbitrary sizes to a digest of a fixed bit size (known as a “hash” or “hash digest”).

Another characteristic of a cryptographic hash function is that it produces a one-way result, that is, a function which is infeasible to invert. Asymmetric cryptography involves a matched set of private-public key pairs. The public key can be shared with anyone and is not a secret. However, the public key itself needs to be maintained with integrity. The private key is the secret credential and needs to be kept confidential. The private key is used to sign data to generate a digital signature and the public key is used to verify the digital signature of the data during the authentication process. In the case of secure boot, a cryptographic hash of the firmware/software image is first generated. The fixed-size hash value is then signed/encrypted with the private key to generate a digital signature. This step itself happens at a secure environment. Figure 1 shows the digital signature generation flow.

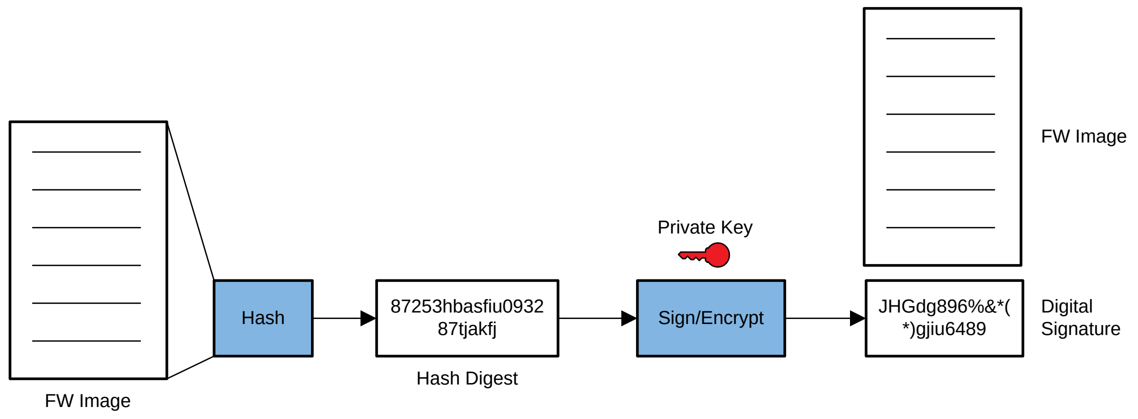


Figure 1. Generating Digital Signature of the Firmware

The digital signature is verified by the corresponding public key during the secure boot process. Figure 2 shows the digital signature verification flow.

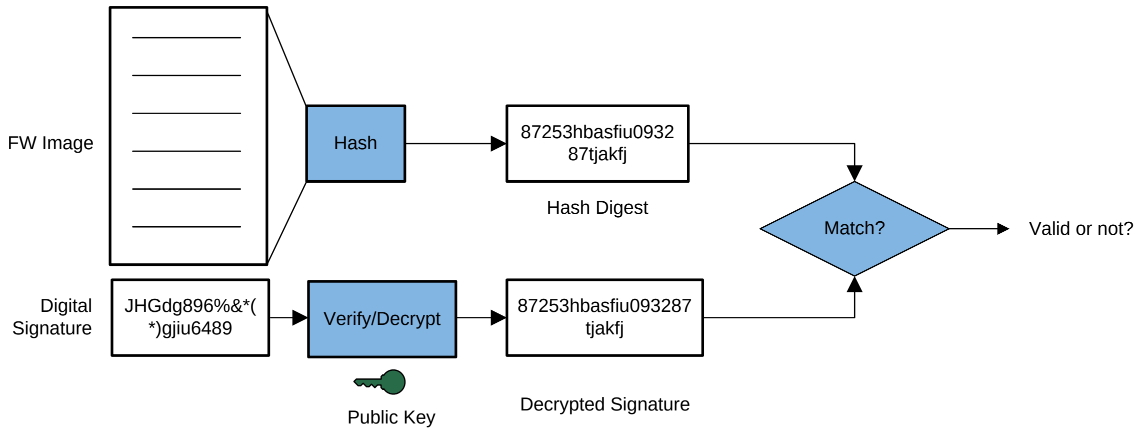
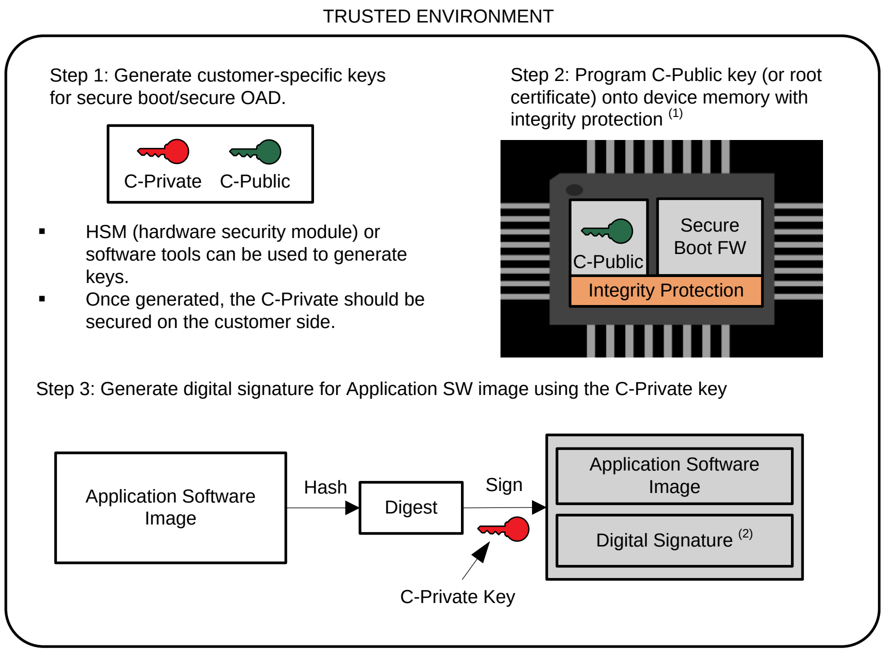


Figure 2. Verifying Digital Signature of the Firmware

1.2 Fundamentals of Secure Boot

Secure boot relies upon two fundamental concepts: key management and hardware root of trust.

Key management is an important aspect to enable security of the keys involved in secure boot. For the secure boot flow, the customer needs to securely generate a private-public key pair (referred to as C-Private and C-Public keys, where C stands for Customer). It is a common industry practice to use a hardware security module (HSM) or software tools (such as OpenSSL's key generation tools) to generate the secure boot public and private key pairs. The customer should always secure the private key on their end as this private key is used for signing the initial and all subsequent firmware images to be programmed onto the device (during production and in-field firmware updates). The signing of the firmware image should take place in a trusted environment. Once the digital signature is generated, the signature is appended to the firmware image programmed onto the device. The corresponding public key is programmed onto the MCU and should be stored with integrity protection on-chip. The immutable storage of both the public key and the secure boot verification code constitutes to enabling an on-chip hardware root of trust (RoT) that is essential for the secure boot process. Figure 3 illustrates the secure boot provisioning and signing flow.


TRUSTED ENVIRONMENT
Step 1: Generate customer-specific keys for secure boot/secure OAD.
C-Private
C-Public
▪ HSM (hardware security module) or software tools can be used to generate keys.
▪ Once generated, the C-Private should be secured on the customer side.
Step 2: Program C-Public key (or root certificate) onto device memory with integrity protection (1)
C-Public
Secure Boot FW
Integrity Protection
Step 3: Generate digital signature for Application SW image using the C-Private key
Application Software Image
Hash
Digest
Sign
C-Private Key
Application Software Image
Digital Signature (2)

(1) If not supported by the MCU boot routine, the secure boot image verification firmware should also be programmed onto the MCU with integrity protection on-chip for this firmware.

(2) The digital signature is typically stored as part of image metadata along with other details essential for performing the secure boot check (e.g. image memory range, type of crypto function used).

Figure 3. Secure Boot – Signing and Provisioning Flow

There are multiple ways of handling the public key for secure boot process. One way, is to store the public key as-is on the MCU with integrity protection of specific non-volatile flash memory regions; and this key is directly used to verify the digital signature of the firmware image during the secure boot process. This is the approach used in the secure boot firmware for CC13x2 / CC26x2 wireless MCUs (discussed further in Section 2).

Another way is to handle the public key in a digital certificate chain. This approach is more suitable for software images delivered by multiple entities that have delegated image signing keys. The digital certificate chain typically comprises of:

- The delegated secure boot public key, either signed by the customer's root private key or a root CA (certificate authority),
- The root certificate issuer's self-certificate (that is, the root public key signed by the root private key)
- Any intermediate certificates (if used).

In this case, only the root certificate issuer's self-certificate (or its hash value) is stored on-chip with integrity protection and is the hardware RoT element on-chip. The secure boot process first involves parsing the certificate/certificate chain, presented as part of firmware metadata and validating it against the root certificate stored as a RoT element on-chip. Then, the validated secure boot public key is used for verifying the firmware on-chip. This approach with digital certificate chain will not be discussed further in this application report.

In addition to storing the secure boot public key and the verification code with integrity protection on-chip, the secure boot process should be executed upon every boot. This is typically enabled as part of the device boot routine before the control is handed over to the user application.

2 SimpleLink CC13x2/CC26x2 Hardware Security Features Enabling Secure Boot

This section covers the SimpleLink CC13x2/CC26x2 hardware security features that facilitate enabling hardware root of trust elements for the secure boot solution.

2.1 On-Chip Flash Programming and Erase Options

The SimpleLink CC13x2/CC26x2 devices are wireless MCUs with integrated non-volatile flash memory. The secure boot routine is intended to verify the integrity and authenticity of the application software (including any protocol stack software) on the on-chip flash upon every device boot/reset.

The on-chip flash memory can only be programmed via:

- JTAG test and debug access ports (debugger or production programmer)
- CPU core:
  - ROM serial bootloader (invoked by external host processor)
  - User application software

The JTAG test and debug access ports (TAP, DAP) can be disabled / secured by selectively configuring the customer configuration (CCFG) parameters (discussed in Section 2.2). It is important to note that some features of JTAG WUC (wake-up controller) test access port is always enabled to send/receive subset of boot ROM commands (chip erase, MCU reset) even when all JTAG TAP/DAP user configurations are disabled. The chip erase functionality of the JTAG WUC can be further disabled as discussed in the following section.

The ROM serial bootloader on CC13x2/CC26x2 devices enables downloading flash images from external devices over the serial interfaces on the UART0 and SSI0 peripherals. Note: The CC13x2/CC26x2 ROM bootloader “does not” implement any bootloader interface security or FW image security (confidentiality, integrity, authenticity); however, the secure boot routine can still be enabled when application requirements dictate the need for keeping the ROM serial bootloader enabled in the field. With necessary device configuration in place for the secure boot process, once the ROM serial bootloader programs the on-chip memory, upon subsequent device boot, the secure boot validates the new image on-chip before starting application firmware execution.

The on-chip flash memory can be write-protected in 8kB flash sectors to provide integrity protection for the designated flash sector(s). This enables protecting the flash sector(s) from being both programmed and erased. The flash sector write protection is enabled by configuring the CCFG parameters.

Additionally, on-chip flash erase functions supported by the device hardware enable erasing on-chip flash, which is useful during device development and recovery phases. The on-chip flash can be erased (entire or parts of it) by:

- Chip-erase - function
- Bank-erase - ROM serial bootloader command
- TI Failure Analysis (FA) access

Chip erase is a Boot ROM function that can be requested via the always enabled JTAG TAP interface (that is, via a pin sequence possible only with local access to the device) and a successful chip erase operation will force the content of the flash main bank (all flash sectors) back to the state it was when delivered by TI.

Bank erase is a ROM serial bootloader command that is requested via the serial bootloader interface (e.g. UART, SPI). A successful bank erase operation will erase all main bank sectors that are not protected by write protect configuration bits in CCFG page.

TI failure analysis (FA) access enables TI to unlock the devices to perform quality or failure analysis. Unlocking device for FA access performs a full chip erase (including on-chip flash and SRAM) and places the device in test mode (as if the device was at TI final test phase).

All of flash erase commands and options discussed above can be disabled by the configuring the corresponding CCFG parameters.

2.2 Customer Configuration (CCFG)

The CC13x2/CC26x2 CCFG parameters enable device configuration customization. The CCFG must typically be set during the production programming process and contains configuration parameters for the Boot ROM code, device hardware, and device firmware. In the context of device security configuration used to implement secure boot, CCFG parameters include:

- JTAG TAP (test access port) and DAP (debug access port) configuration
- Flash sectors write protection configuration
- ROM Bootloader configuration
- Flash erase configuration (chip-erase and mass erase)
- TI Failure analysis (FA) access configuration
- Image valid address: reset vector configuration

The CCFG parameters are located at the end of the last flash sector. The CCFG parameters are set at compile time in the ccfg.c file included with the CC13x2/CC26x2 software development kit (SDK) or can be configured by the application via supplied flash programming application program interfaces (APIs). Therefore, during the device production programming and as part of CCFG configuration, it is recommended to enable write protection of the flash page containing the CCFG along with the chip-erase disable configuration in order to protect the integrity of the customer configurations.

The various CCFG parameters offer flash programming and erase protection options to the customer application. Additionally, the image valid address CCFG parameter enables configuring start of the flash vector table in order to enable the boot sequence in ROM to transfer control to a flash image.

For more information, see the Device Configuration chapter in the *CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual*.

2.3 Hardware RoT for Secure Boot

Immutable storage of customer public key and the secure boot firmware plus the assurance of secure boot process being executed upon every boot constitute to the hardware RoT essentials for the secure boot process. This section discusses how the hardware security features discussed in the previous sections are used to enable these hardware RoT essentials to implement secure boot on CC13x2/CC26x2 wireless MCUs.

The Boot Image Manager (BIM) is a software executable that hosts the secure boot and secondary bootloader (for secure over the air download (OAD)) routines. For more information about the BIM, see Section 3.

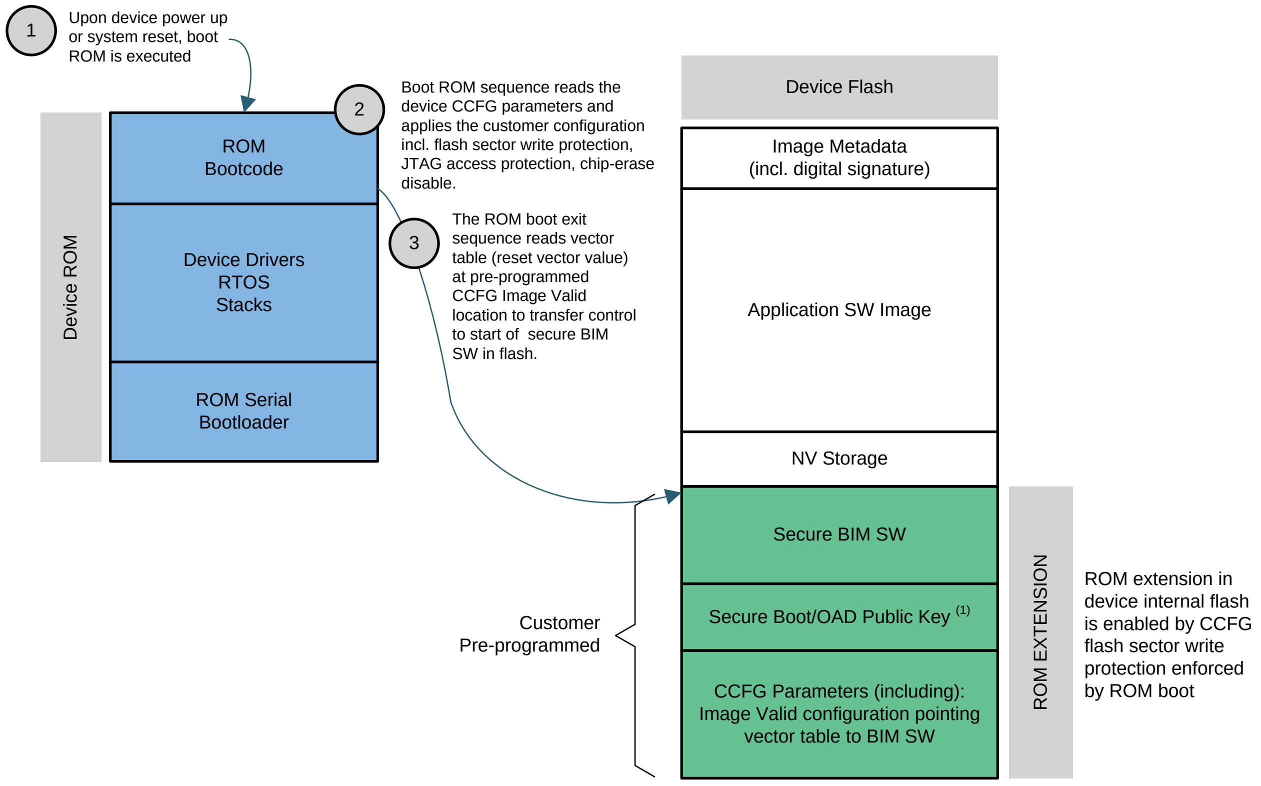
The BIM software, including the C-Public key used for verification of the application software, should be programmed along with the device CCFG parameters onto the last sector of flash. To enable integrity protection for this last flash sector that holds the BIM, public key and the CCFG parameters, the following configurations should be considered:

- Flash sector write protection configuration
- JTAG TAP, DAP access disable configuration
- Chip-erase disable configuration

Additionally, the CCFG “Image Valid” parameter should be configured with a value corresponding to the address of the flash vector table within the BIM region. And, the vector table within the BIM region should be programmed with appropriate initial stack pointer and reset vector values such that the device Boot ROM sequence transfers control to execute the BIM software upon any device system or power-on reset event. Once the above recommended CCFG parameters are enabled, the last sector of flash containing the CCFG, BIM and public key will no longer be writable or erasable even if the ROM serial bootloader is enabled.

The CC13x2/CC26x2 device hardware features, combined with trusted programming of the BIM software containing the customer public key and the CCFG parameters, enables the hardware RoT needed for the secure boot process.

As discussed in Section 1.2, the application software image should be signed using the customer private key (C-Private) in a trusted environment. The corresponding public key (C-Public) should be programmed along with the BIM software and other CCFG configuration during device programming in a trusted environment. The image signature is stored as part of the image metadata in the internal flash. The image metadata also has additional details about the application software image including valid address range of the application software image to verify, type of algorithms used, and so forth (see Figure 4).


1
Upon device power up or system reset, boot ROM is executed
2
Boot ROM sequence reads the device CCFG parameters and applies the customer configuration incl. flash sector write protection, JTAG access protection, chip-erase disable.
3
The ROM boot exit sequence reads vector table (reset vector value) at pre-programmed CCFG Image Valid location to transfer control to start of secure BIM SW in flash.
Device ROM
ROM Bootcode
Device Drivers RTOS Stacks
ROM Serial Bootloader
Device Flash
Image Metadata (incl. digital signature)
Application SW Image
NV Storage
Secure BIM SW
Secure Boot/OAD Public Key (1)
CCFG Parameters (including): Image Valid configuration pointing vector table to BIM SW
Customer Pre-programmed
ROM EXTENSION
ROM extension in device internal flash is enabled by CCFG flash sector write protection enforced by ROM boot

(1) The BIM software for CC13x2/CC26x2 devices only supports storing ECDSA public key as is (not as a certificate).

Figure 4. Hardware RoT for Secure Boot

Upon device reset or power-up, the boot ROM is executed. The boot ROM sequence reads the device CCFG parameters in the last flash sector and applies the customer configuration, including JTAG access protection, chip-erase disable and flash sector write protection on a per-sector basis. The boot ROM then transfers control to the BIM software that verifies authenticity of the application software image. The BIM first computes the hash of the application software image using the SHA-256 accelerator and then, verifies the hash against the decrypted image signature value (decrypted using the secure boot public key). If the verification passes, the BIM transfers control to the application software image by jumping to a fixed location in application memory space. If the verification fails, then, BIM offers custom options to handle this case. For more information, see Section 3.4.

3 SimpleLink CC13x2/CC26x2 SDK Components Enabling Secure Boot

This section covers the SimpleLink CC13x2/CC26x2 SDK components and its features that enable the secure boot solution on these devices.

3.1 BIM Overview

The boot image manager (BIM) software component in the internal flash memory and is a non-RTOS micro-application available as part of the SimpleLink SDK that handles boot-time authentication of on-chip image prior to transferring execution to the main application firmware image. In this context, the main application firmware image is the software image that defines the primary behavior of the device, and typically includes the wireless application, protocol stack and RTOS that resides in and executes from internal flash memory. The BIM uses secure hash (SHA-256) and asymmetric cryptographic algorithms (ECDSA P-256) to implement the secure boot signature verification process on the wireless MCU. The flash memory region write protection can be used to maintain the integrity of the BIM software and the associated public key used for firmware image verification; thus enabling integrity protection for the secure boot verification code and associated public key stored in on-chip flash. For further details about BIM and the security functions offered by BIM, see TI Resources pages:

- Boot Image Manager (BIM)
- Secure BIM

The BIM also handles secure OAD (over the air download) with authentication of new application firmware images to be programmed on-chip. Although the security basics in BIM for secure boot and secure OAD are similar, this application note will mostly be discussing BIM in the context of secure boot.

3.2 BIM Configurations

The BIM is provided in two source code project configurations as described below based on the method of how firmware updates occur (if supported). Regardless of the BIM configuration, the same image authentication signature checks are performed upon each boot.

Table 1. BIM OAD Project Configurations

BIM OAD Project Configuration	Intended Application
On chip	On-chip OAD stores the new downloaded image to internal flash.
	• On chip firmware updates with persistent and user application. Persistent application is not update-able.
	• Standalone / no firmware update operation supported
	• Firmware update from external source, such as ROM serial bootloader
Off chip	Off-chip OAD stores the new downloaded image to external flash.
	• Firmware image is updated from off chip memory

BIM also supports two methods of security: Unsecure and Secure "OAD". Unsecure OAD does not authenticate/verify the new image before programming or executing it on the device. Secure OAD authenticates new images using an Elliptic Curve Digital Signature Algorithm (ECDSA) before it is installed and executed on-chip. Be sure to select the “Release” build configuration within the BIM project to enable the secure boot authentication. Build configurations with “unsecure” or without the SECURITY predefined symbol defined do not enable image authentication.

For further details about the BIM configuration options, see TI Resources page: OAD Storage & Security.

3.3 BIM Entry and Exit

An essential property of any secure boot implementation is to ensure that the main application image is always authenticated prior to execution of that image. Bypassing this authentication step would amount to product vulnerability as an attacker could execute an untrusted firmware image on the device. When properly configured, the BIM is the first software entity to execute from the system flash following a reset, which can include power-on, system, watchdog or external reset sources (pin or brown out). Upon each reset, the device boot ROM is executed before handing execution to a defined entry address in the internal flash memory. This controlled entry to flash memory from the device boot ROM to the BIM is defined by setting the IMAGE_VALID_CONF field within the CCFG block to the base address of the flash page containing the BIM (typically last page of Flash memory).

Table 2. IMAGE_VALID_CONF Register Field Description

Bit	Field	Type	Reset	Description
31-0	IMAGE_VALID	R	FFFFFFFFh	This field must have the address value of the start of the flash vector table in order to enable the boot FW in ROM to transfer control to a flash image. Any illegal vector table start address value will force the boot FW in ROM to transfer control to the serial bootloader in ROM.

With the IMAGE_VALID_CONF setting configured to the BIM's base address and associated flash protection fields enabled in the CCFG, the BIM will:

- Reside in write-protected flash
- Be the first software entity to execute following a boot ROM exit upon device reset
- Perform a signature validation check on the main application firmware image prior to jumping the program counter (PC) to the main application firmware image. The flash protection using CCFG is discussed in Section 2.

3.4 Fault Handling

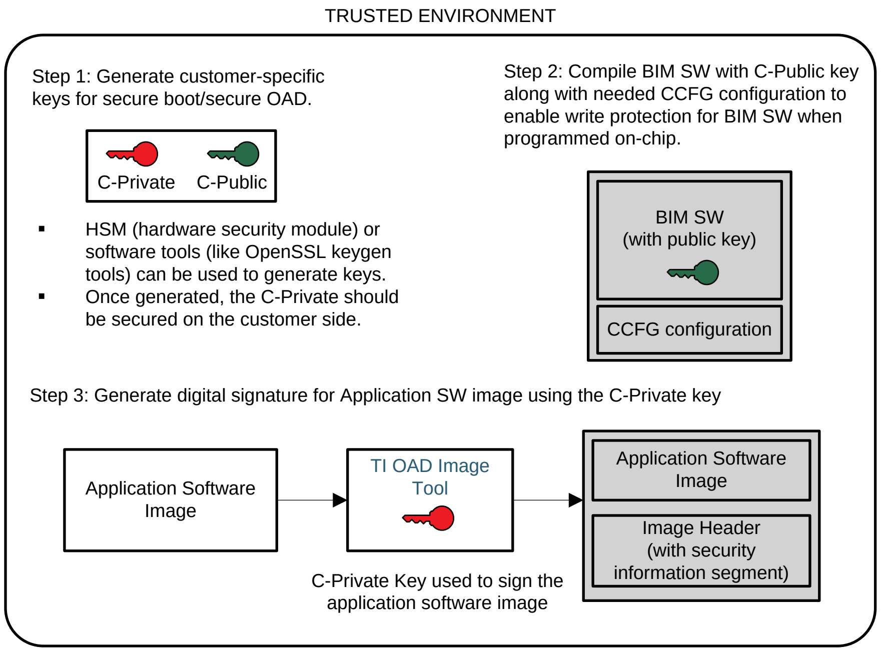
If a signature validation fault is detected by the BIM during the image authentication process, execution of the main firmware image will be inhibited by the BIM. The system developer can implement custom fault handling logic provided the custom extensions to the BIM fit within the flash page allocated to the BIM. Such extended fault handling logic can include, for example, indefinitely entering a low power state (TI default implementation), erasing the non-valid firmware image and/or illuminating a status LED on the board.

4 Secure Boot: Programming Flow and Tools Support

This section covers initial programming of the BIM software and initial/in-field programming of the application image topics. It also discusses the tools provided by TI to aid these processes.

4.1 Initial Programming

To enable secure boot and secure OAD, the customer/OEM should program the devices with the BIM software along with required CCFG configurations in a trusted environment, (see Figure 5).


TRUSTED ENVIRONMENT
Step 1: Generate customer-specific keys for secure boot/secure OAD.
C-Private
C-Public
▪ HSM (hardware security module) or software tools (like OpenSSL keygen tools) can be used to generate keys.
▪ Once generated, the C-Private should be secured on the customer side.
Step 2: Compile BIM SW with C-Public key along with needed CCFG configuration to enable write protection for BIM SW when programmed on-chip.
BIM SW (with public key)
CCFG configuration
Step 3: Generate digital signature for Application SW image using the C-Private key
Application Software Image
TI OAD Image Tool
C-Private Key used to sign the application software image
Application Software Image
Image Header (with security information segment)

(1) C-Private and C-Public keys refer to customer-specific keys and “C” stands for “Customer”.

Figure 5. Initial Programming Flow With TI Provided Tools to Facilitate Secure Boot on SimpleLink CC13x2/CC26x2 Devices

As shown in Figure 5:

Step 1: The customer should generate private-public key pair to be used for secure boot/secure OAD. An HSM (hardware security module) as part of the customer’s production infrastructure can be used to generate these keys. Another option is to use open source software tools (also located in SDK OAD tool folder). Note: TI default private-public key pair is provided for enabling the out-of-the-box demo. However, for production, it is strongly recommended that the customer generate customer-specific keys for this security solution. For more details, see TI Resource Explorer: Generating new security keys (Embedded).

Step 2: Once the customer-specific security keys are generated, the public key should replace the TI default public key used by the BIM software. Instructions to do this can be found at TI Resource Explorer: Generating new security keys (Embedded). Next, the BIM software compiled with C-public key should be programmed onto the CC13x2/CC26x2 device along with the appropriate CCFG parameters (as discussed in Section 2.3) onto the last flash sector."

Step 3: The application software should be signed with the customer-specific private key. TI provided OAD Image tool is intended to process the compiler output in the form of a hexfile and prepare the image for initial programming or OAD transfer. With the secure mode enabled and the customer private key provided to the tool, the OAD Image tool generates the image header file with the security information segment that includes the image signature.

See TI Resource Explorer for further details about:

- OAD Image Tool
- OAD Image Header segments/Security Information Segment

Initial programming of the BIM software with C-public key and the CCFG parameters (that locks the device JTAG access and sets the appropriate flash memory write/erase protection for the BIM software and CCFG parameters) should always be done in a trusted environment. This allows for the following initial programming flows:

- **Flow 1:** The BIM software image with CCFG configuration and the application software image with the image headers are programmed at the same time in a trusted programming environment. The BIM and application software images can either be into an unified image or programmed in a sequence.
- **Flow 2:** In a trusted environment, only the BIM software with appropriate CCFG configuration is programmed first. The CCFG configuration locks the device JTAG access and sets the appropriate flash memory write/erase protection for the BIM software and CCFG parameters on-chip. Next, the application software image is programmed via the ROM bootloader serial interface (UART, I2C,..) later in a potentially untrusted environment (contract manufacturing (CM) site or in-field). Note:
  - The ROM bootloader operation itself does not have security measures; however, the secure boot enabled by the secure BIM shall verify validity of the firmware programmed via the ROM bootloader upon subsequent reset.
  - The bank-erase ROM serial bootloader command may need to be enabled for handling faults during ROM bootloader operation. With this command enabled, a bank erase operation shall erase all main bank sectors that are not protected by write protect configuration bits in CCFG page.

4.2 Software Updates In-Field

When the application software image needs to be updated in-field (that is, beyond the initial programming phase), you can generate a new software image. To generate the appropriate security image header, perform step #3 from Section 4.1.

The new software image along with the updated image headers is transferred over the air to the device which is configured to receive the image in either OAD on-chip or off-chip configuration. Once the new software image is received (either on-chip or off-chip), then, the BIM software is executed to validate the new software image. Once the software update is complete and the device is reset, as part of secure boot process, the BIM again validates the software image on-chip before transferring execution control to the application software image.

For BLE stacks specifically, the SDK supports an OAD distributor and OAD target application set-up that enables testing the software updates from an OAD distributor to an OAD target. Demo using CC2640R2 LaunchPads is available. In this demo set-up, the OAD distributor connects to the BLE target device (which has OAD functionality enabled) and is responsible for fragmenting an OAD enabled image into chunks of OAD blocks and sends each block over the to the OAD Target device as they are requested. The OAD distributor is a combination of BTool running on the PC that is connected to a host-test application running on a CC2640R2 LaunchPad. For further details about performing BLE OAD and OAD using Btool, see the following TI Resource Explorer links:

- Performing BLE OAD
- Setting up the BLE OAD Environment
  - OAD using Btool

5 Frequently Asked Questions

1. Once the BIM software is programmed on-chip with flash memory write/erase protection, how can it be updated or re-programmed?

   The device Flash memory protection is enabled by the CCFG. Locking the CCFG parameters in the last flash sector along with BIM is desirable to enable integrity protection of the BIM software and the secure boot public key. If it is required to reprogram the BIM or CCFG after the flash protection is enabled, the chip-erase boot ROM function should be used. This function can only be invoked with physical access to the device. This may require recalling products to perform the BIM software update and may not be practicable in many cases. A successful chip-erase operation will force the content of the flash main bank back to the state it was when delivered by TI (also referred as a blank device). For more information, see Section 2.1.

   With a blank device, any software can be programmed on-chip. Therefore, it is important that the customer evaluates the risk of their product being exposed to physical threats (that enables physical access to device to perform chip-erase function) vs. BIM software needing to be updated to decide if the chip-erase command should be kept enabled (via CCFG) or not. Table 3 lists combinations of CCFG parameter configurations which enable the last flash vector with CCFG and BIM software (incl. secure boot public key) to be reprogrammed or erased.

Table 3. Re-Programming/Erase Access of BIM Software in the Last Flash Sector of CC13x2/CC26x2 Devices

CCFG Parameters for On-Chip Flash Security			
Debug and Test Access Port Disabled (CCFG_TAP_DAP)	**Last Flash Sector Program and Erase Protection Enabled (CCFG_PROT)**	**Chip Erase Disabled (ERASE_CONF)**	**Last Flash Sector With CCFG and BIM Software (including secure boot public key) Accessible for Reprogramming or Erase?**
Yes	No	No	Can be re-programmed, modified or erased by application software or externally via serial bootloader (if enabled)
Yes	Yes	No	Can be erased with Chip erase function and subsequently reprogrammed. [1]
Yes	Yes	Yes	Not accessible for reprogramming or erase

[1] Chip erase function can only be invoked with physical access to the device.

NOTE: For more information, see the *Device Configuration* chapter in the *CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual* for parameter values needed to achieve above configurations

2. **What happens if customer private key used for image signing for secure OAD secure boot is compromised or lost?**

   The customer private key used for secure OAD/secure boot should be highly secured on the customer end. Since this private key is used for image signing, anyone with access to this private key can sign images and be able to update main application image on-chip. With the private key compromised or lost, the customer should generate new private-public key pair and program the new secure boot public key onto the devices in field.

   When using the BIM software for secure boot operation, it is recommended that the customer-specific secure boot public key is stored in flash protection sector along with the BIM software and device CCFG parameters. Updating this secure boot public key would require performing a chip-erase boot ROM function to completely erase the device and the, program the BIM with new public key (see Table 3). This may require recalling products to perform the public key update and may not be practicable in many cases. Therefore, customer private key security should be secured with utmost priority and security on customer side with appropriate key management infrastructure.

6 References

The latest secure OAD and BIM documentation can be found at: TI Resource Explorer - under Documents/BLE5-Stack/BLE5-Stack User's Guide/Over-the-Air Download (OAD).

- Texas Instruments: *CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual*
- SimpleLink™ CC13x2 and CC26x2 SDK (including BIM) Download