Tech Talks LIVE Schedule – Presentation will begin shortly

Silicon Labs LIVE:

Wireless Connectivity Tech Talks

Topic	Date
Building a Proper Mesh Test Environment: How This Was Solved in Boston	Thursday, July 2
Secure Your Bluetooth Design with BG21/BG22	**Thursday, July 23**
New Bluetooth Mesh Light & Sensor Models	Thursday, July 30
Simplicity Studio v5 Introduction	Thursday, August 6
Long-Range Connectivity Using Proprietary RF Solution	Thursday, August 13
Wake Bluetooth from Deep Sleep Using an RF SIgnal	Thursday, August 20
Implementing a Bluetooth Network Co-Processor	Thursday, August 27

Fill out the survey for a chance to win a BG22 Thunderboard!

Find Past Recorded Sessions at:
https://www.silabs.com/support/training

SILICON LABS

WELCOME

Silicon Labs LIVE:

Wireless Connectivity Tech Talks

Secure Your Bluetooth Design with BG21/BG22

JULY 23 2020

silabs.com/security

BG21: Optimized for Secure Mains Powered Devices

BG21 can be paired with EFP to reduce active TX/RX current consumption

Radio
Up to +20 dBm TX
Extremely good RX sensitivity
Bluetooth 5.1

Current Consumption
8.8 mA RX (1 Mbit/s GFSK)
10.5 mA TX @ 0 dBm
33.8 mA TX @ 10 dBm
4-8uA EM2

World Class Protocol Stacks
Bluetooth 5.1 and Bluetooth mesh
Apple HomeKit

Compact Size
4x4 QFN32 (20 GPIO)

ARM Cortex-M33 with TrustZone
80 MHz w/ FPU and DSP
Up to 92kB RAM and 1024kB flash
50.9 µA/MHz

Peripherals Fit for Purpose
3x USART, 2x I2C
1x 12-bit ADC, 2x ACMP
7x timers
Up to 20x GPIO

Security
True Random Number Generator
Hardware Accelerated Crypto Engine
Secure Boot with RTLS
Secure debug with lock/unlock
DPA Countermeasures

With Secure Vault™
Anti tamper
Secure attestation
Secure key management and storage
Advanced crypto

BG22: Optimized Battery Powered Bluetooth LE

Optimized

Secure Bluetooth 5.2 SoCs for High-Volume Products

Radio
Bluetooth 5.2
+6 dBm TX
-106.7 dBm RX (125Kbps)
AoA & AoD

Ultra-Low Power
3.5 mA TX (radio)
2.6 mA RX (radio)
1.4 µA EM2 with 32 kB RAM
0.5 µA w/ RTC in EM4

World Class Software
Bluetooth 5.2
Bluetooth mesh LPN
Direction Finding
Apple HomeKit

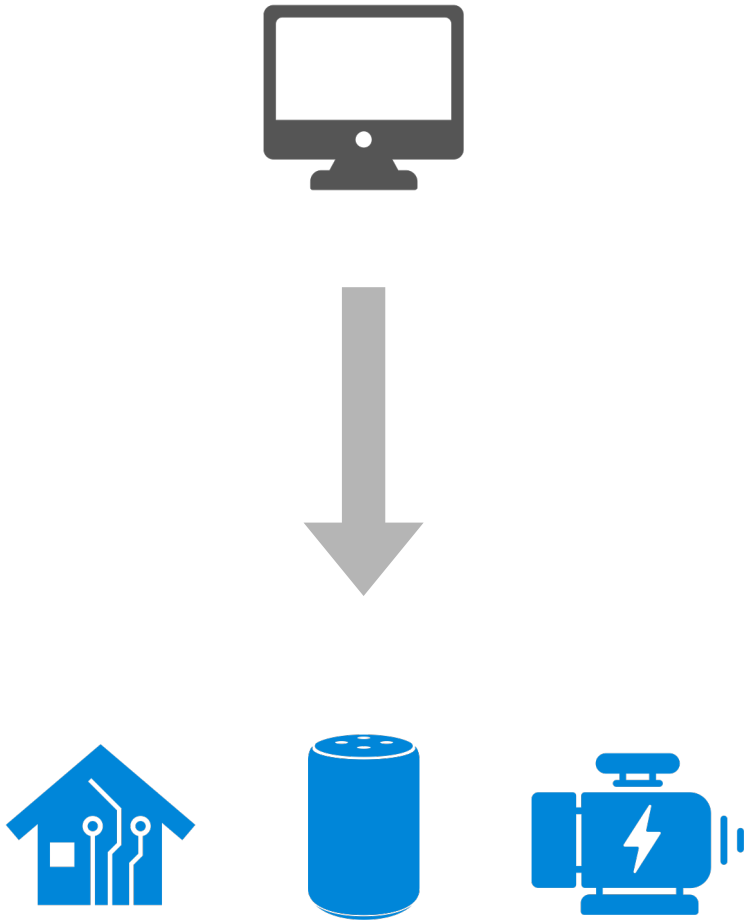
Compact Size
5x5 QFN40 (26 GPIO)
4x4 QFN32 (18 GPIO)
4x4 TQFN32 (18 GPIO)

ARM Cortex-M33 with TrustZone
76.8 MHz
FPU and DSP
352/512 kB of flash
32kB RAM

Peripherals Fit for Purpose
2x USART, 2x I2C, 2x PDM and GPIO
12-bit ADC (16 channels)
Built-in temperature sensor with +/- 1.5 °C
32 kHz, 500ppm PLFRCO eliminates crystal

Security
AES128/256,SHA-1, SHA-2 (256-bit)
ECC (up to 256-bit), ECDSA and ECDH
True Random Number Generator (TRNG)
Secure boot with RTSL
Secure debug with lock/unlock

Hacking Targets are moving from IT to IoT

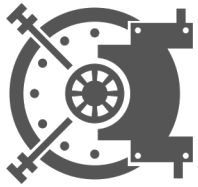


- Targeting end users is small reward
- Targeting big business has greater reward
  - Companies are the new ransomware targets, not individuals
  - Companies cannot afford the downtime
  - Companies have more money
  - Companies don't want negative press
- New laws require "All Connected Devices" have reasonable security

	Reward	Trend in OT	Comment	IoT Target
Denial of Service	$ $ $ $ $	Growing	Very simple to implement	**YES**
Spam Attacks	$	None	Little reward, IoT often headless	
Cryptocurrency Mining	$ $	Neutral	Limited, requires compute cycles not common in IoT	
Ransomware	$ $ $ $	Growing	Tends to be highly targeted	**YES**
Blackmail / Extortion	$ $	Neutral	Not easy to scale	
Pranks / Nuisance	$	None	Little reward, no professional crime incentive	
Information Theft	$ $ $	Neutral	Done because it is simple	**YES**
Click Fraud	$ $ $ $ $	Growing	High volumes of "Bots" to create 'click' revenue	**YES**
Premium Services	$ $ $ $	Down	Difficult to conduct	
Sniffing Network Traffic	$ $	Neutral	Difficult with SSL/TLS	
Pivot Attacks	$ $ $ $ $	Growing	Easy access point to fleet servers	**YES**
Proxy	$	Neutral	Not lucrative, but useful	

The Four Pillars of IoT Security

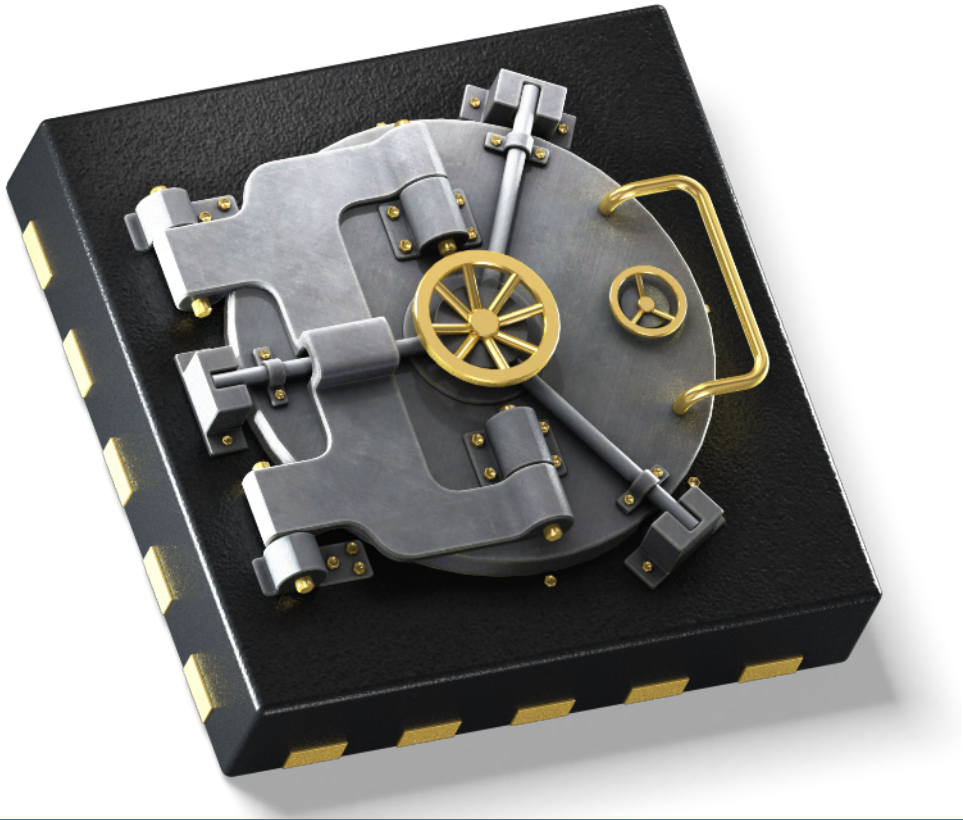
Security Portfolio



Feature	Basic	+Root of Trust	+Secure Element	Secure Vault
True Random Number Generator	✓	✓	✓	✓
Crypto Engine	✓	✓	✓	✓
Secure Boot	✓	✓	✓	✓
Secure Boot with RTSL	-	✓	✓	✓
ARM® TrustZone®	-	✓	✓	✓
Secure Debug with Lock/Unlock	-	✓	✓	✓
DPA Countermeasures	-	-	✓	✓
Anti-Tamper	-	-	-	✓
Secure Attestation	-	-	-	✓
Secure Key Management	-	-	-	✓
Advanced Crypto	-	-	-	✓
	Series 1 – xG1x M4	**Series 2 – xG22 M33**	**Series 2 – xG21A M33**	**Series 2 – xG21B M33**

silabs.com/security

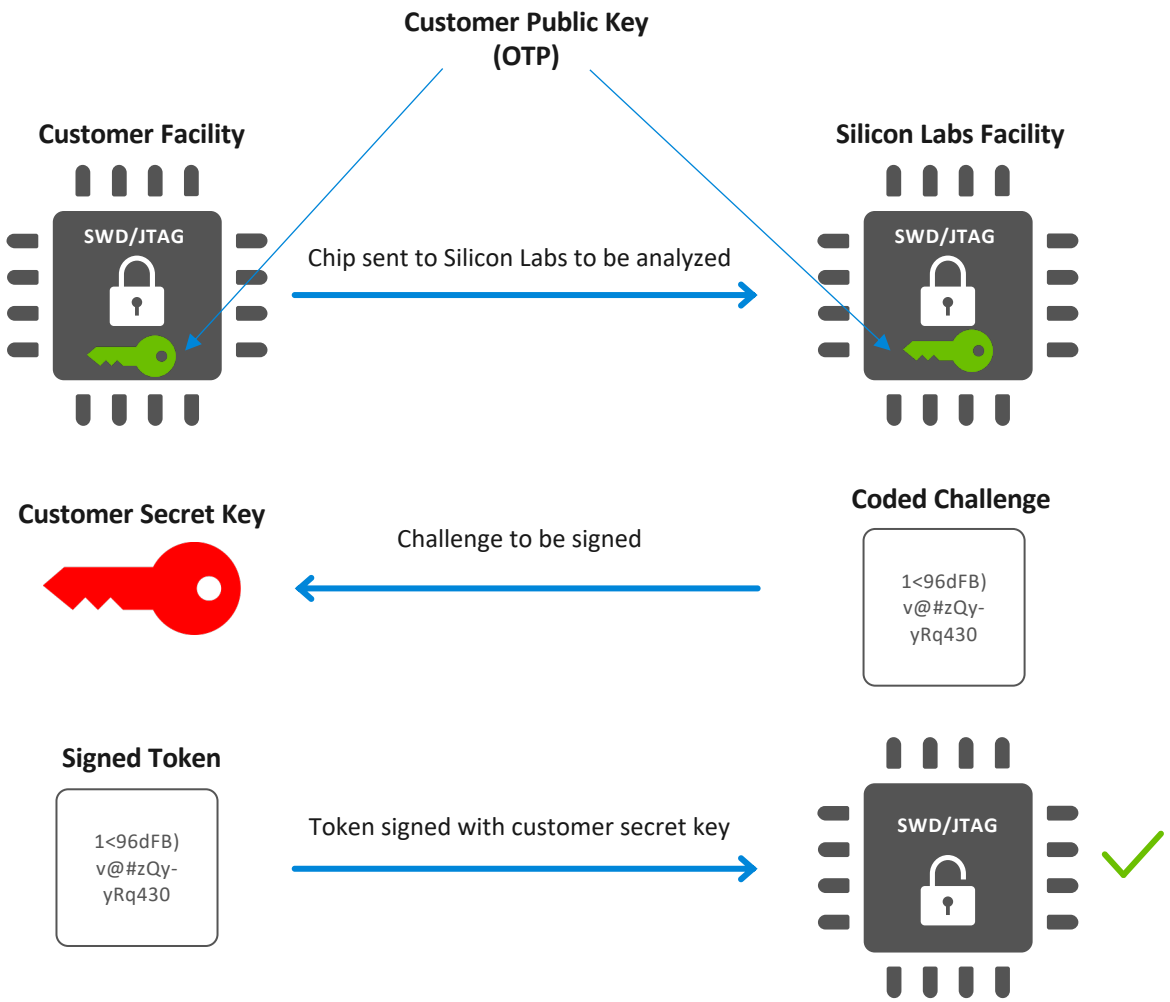
Securing Bluetooth Products with BG21A/BG22



www.silabs.com/security

- **ARM Cortex M33 Core with TrustZone**
  - Provides cost effective hardware isolation
- **Hardware Accelerated Crypto**
  - Faster, more energy efficient and secure than software
- **True Random Number Generator (TRNG)**
  - Compliant with NIST SP800-90 and AIS-31
- **Secure Debug with Lock/Unlock**
  - Allows authenticated access for enhanced Failure Analysis (FA)
- **Secure Boot with Root of Trust and Secure Loader (RTSL)**
  - Prevents malware injection and rollback
  - Ensures authentic firmware execution and OTA updates

Secure Debug Unlock



- Vulnerabilities
  - Unlocked ports are a significant security vulnerability (local attack vector)
  - Unlocking debug ports typically wipes the memory to protect IP but this limits device failure analysis capabilities
- Secure Debug
  - Lock the emulation port and use optional cryptographic tokens to unlock it allowing memory to remain intact

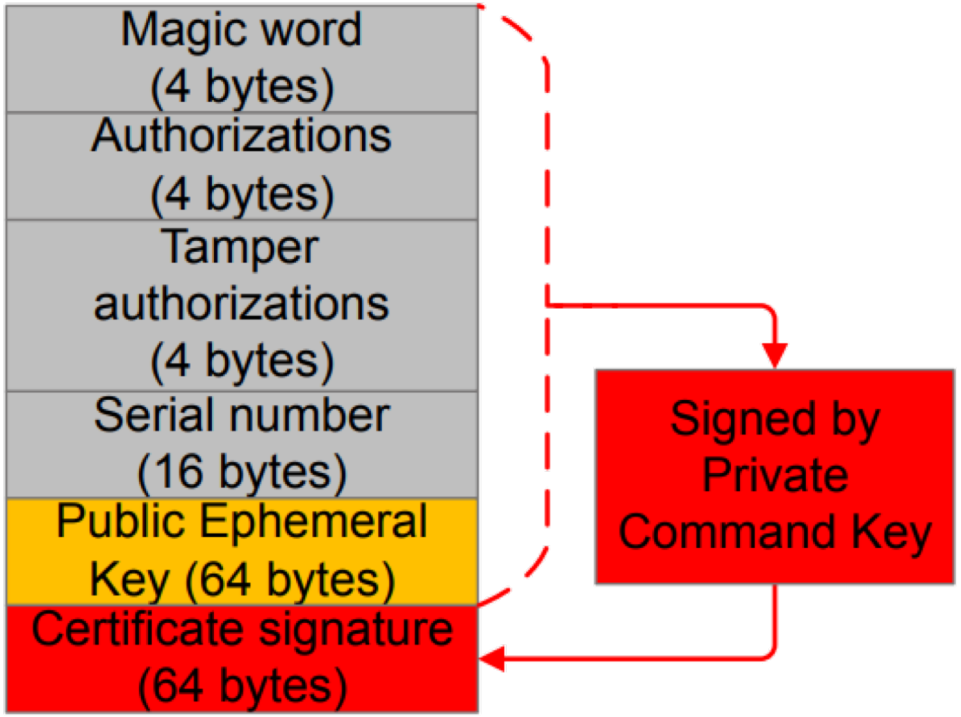
Secure Debug Unlock: How to Provision

- Generate public/private “command key” pair (ECDSA P-256)
- Write Public Command Key into device at manufacturing and enable debug access options
- NOTE: public key and debug access options are stored in OTP (can’t be changed again after writing)

Property	Description if Value==True	Default Value
Debug Lock	The debug port is kept locked on boot	False (Disabled)
Device Erase	The Device Erase command is available (will unlock debug if executed)	True (Enabled)
Secure Debug	Secure debug unlock is available	False (Disabled)

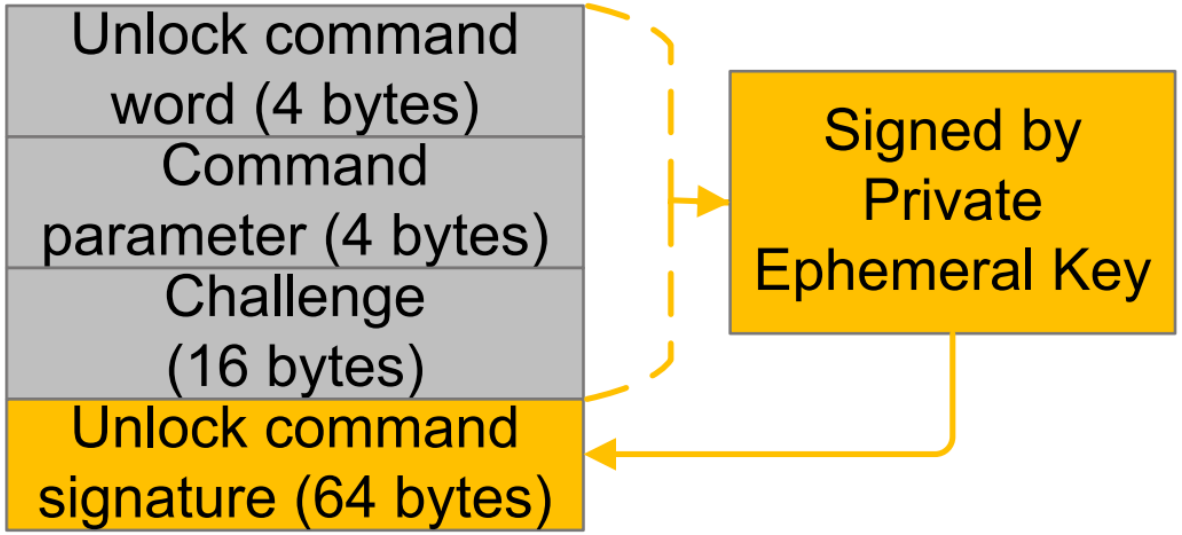
Secure Debug Unlock: How to Use

- To securely unlock the debug port
  - Create an ephemeral key pair (ECDSA NIST256p)
  - Get the serial number from the target device
  - Use the private command key to sign a certificate containing the Public Ephemeral key

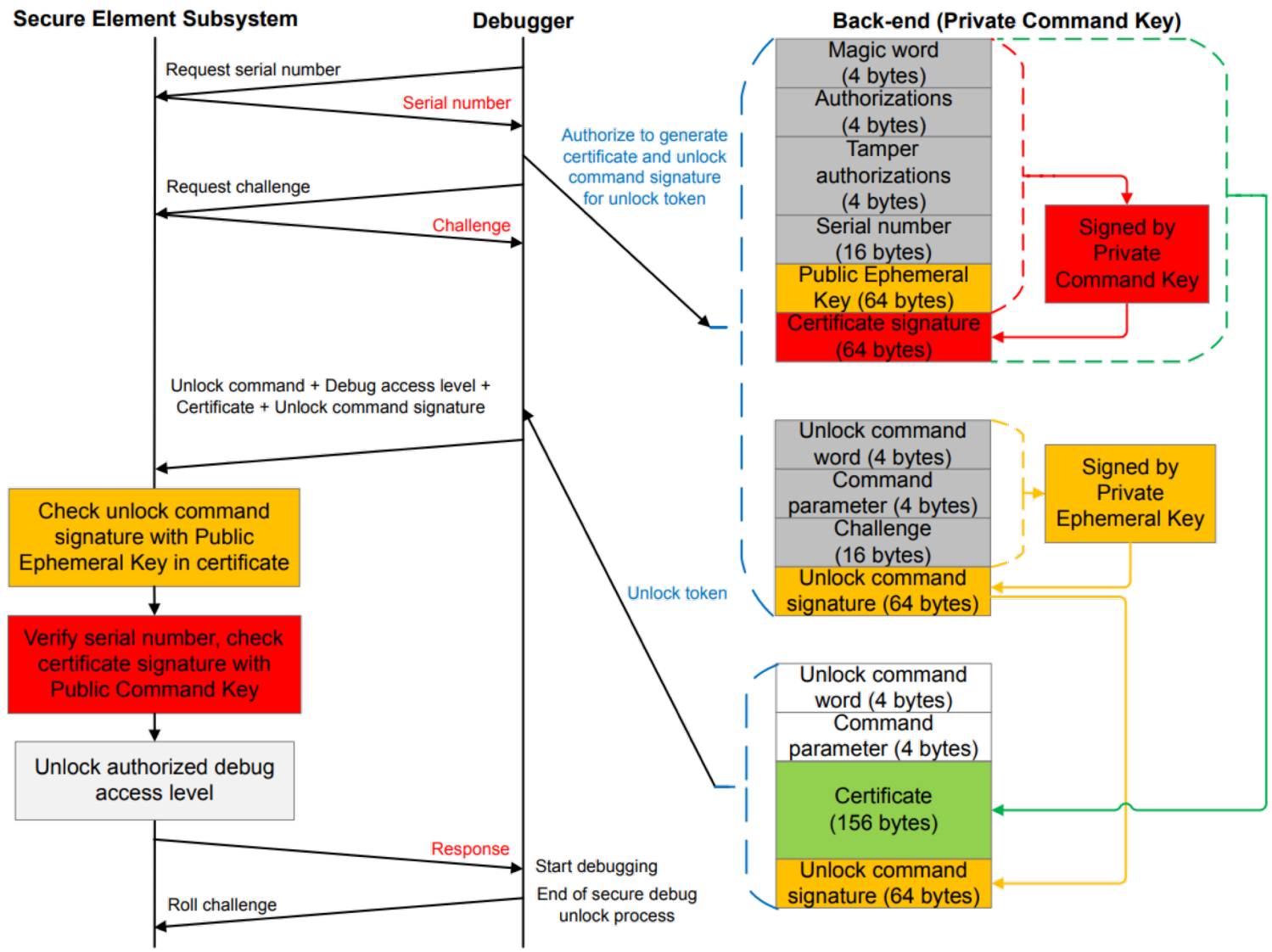


Secure Debug Unlock: How to Use

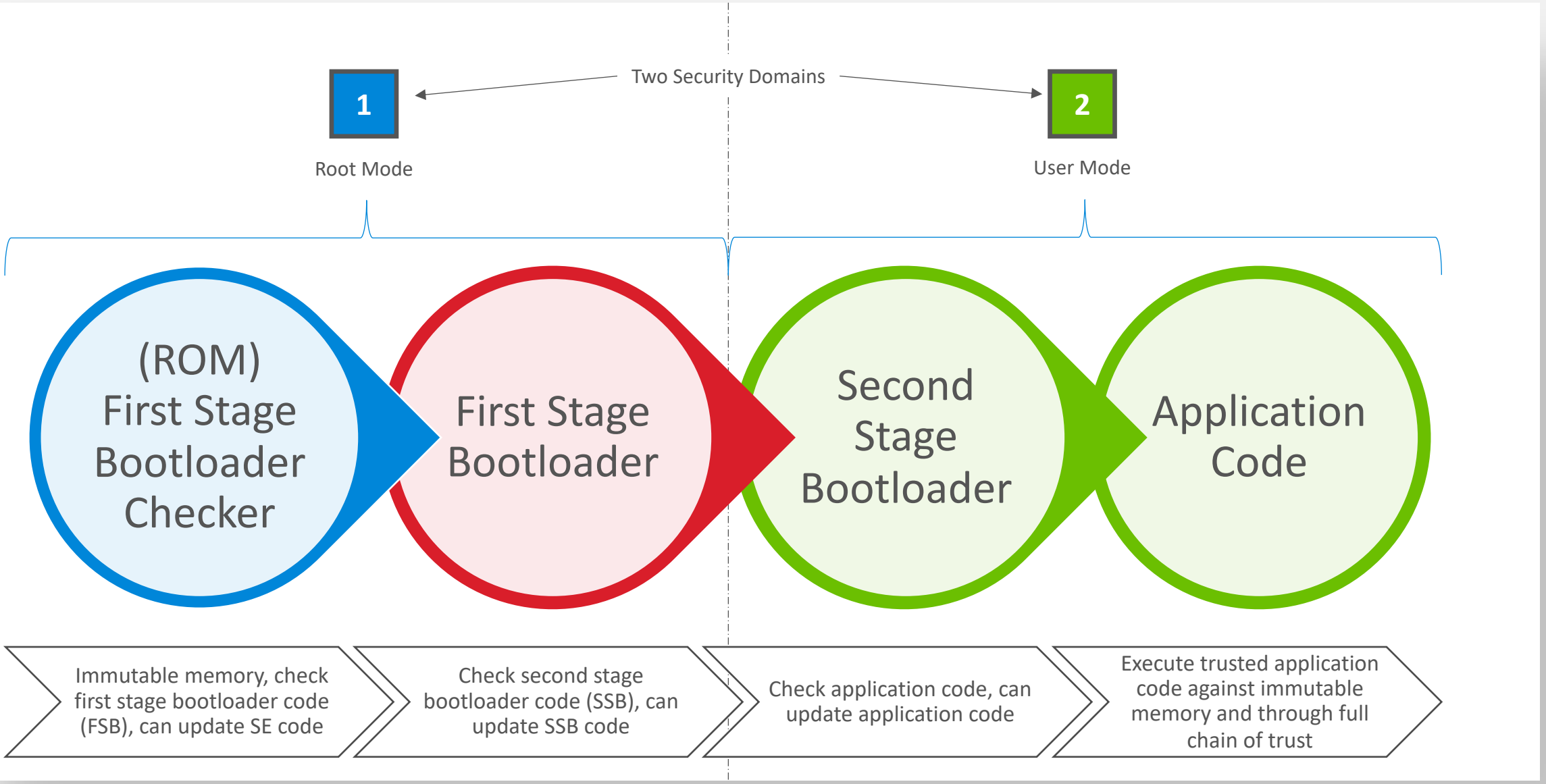
- Get the challenge from the device
- Sign the challenge with the Private Ephemeral Key



Secure Debug Unlock: How to Use

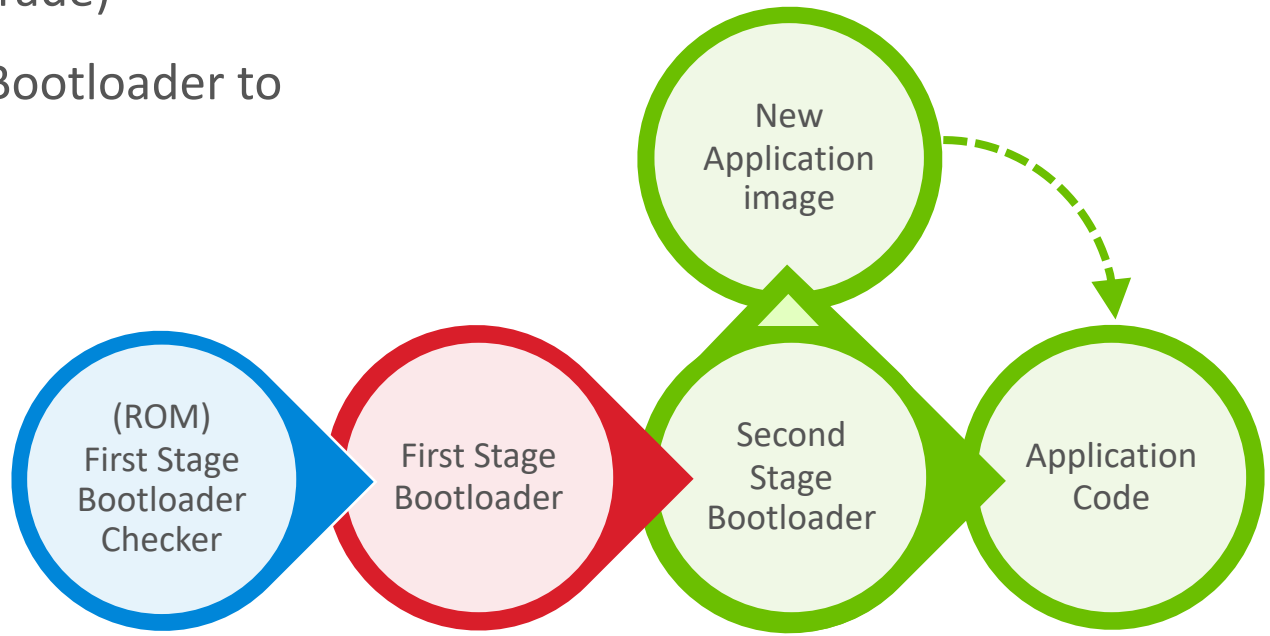


Series 2 Secure Boot with RTSL and Gecko Bootloader



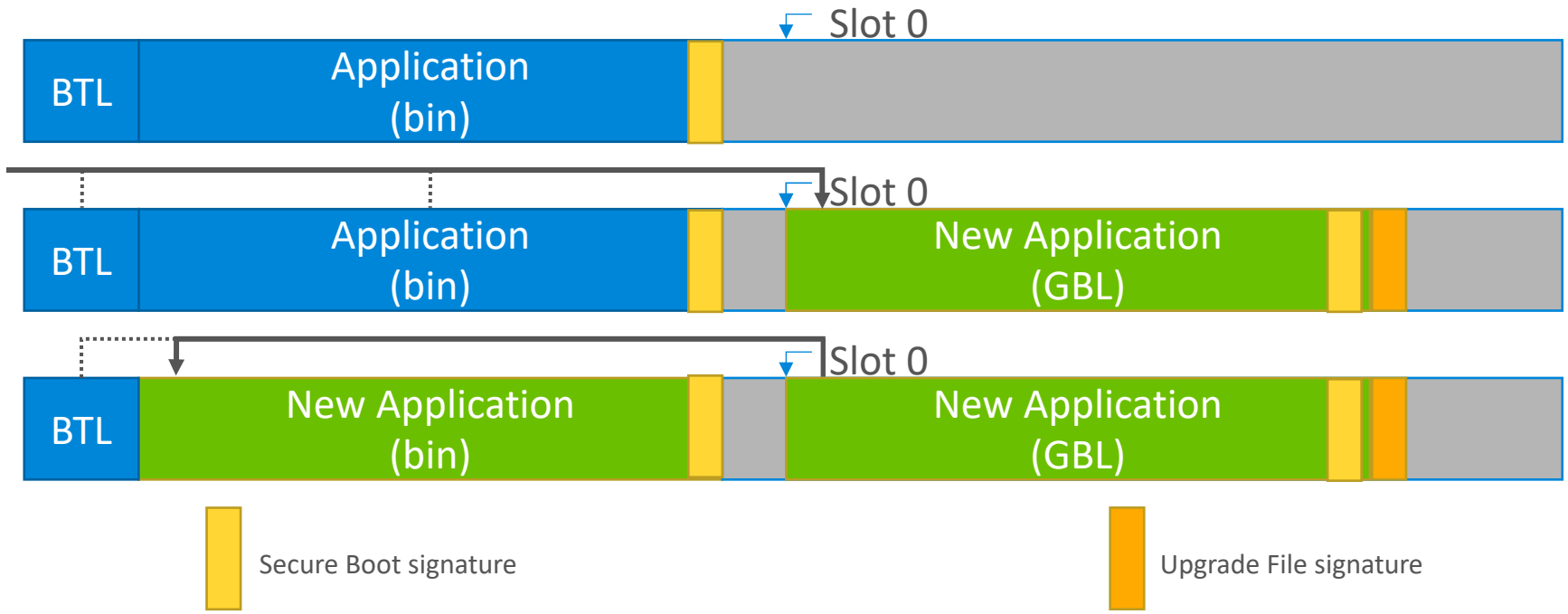
Application Driven Firmware Upgrade

- Second Stage Bootloader (SSB) checks the signature of the Application directly on every boot (secure boot)
- When a new image is available, the Application requests the Second Stage Bootloader to check the signature of the new image (secure upgrade)
- Application requests the Second Stage Bootloader to install the new Application image



Application Driven Firmware Upgrade

1. Download the new image to a designated download area first with the main application
2. Check the signature of the downloaded image, and if it is correct, then
3. Apply the downloaded image (i.e. let the 2nd stage bootloader copy it over the old application)



Application Driven Firmware Update

- Application driven firmware update works great if there is enough flash space for two separate copies of the application
- The application can continue to run during the update
- The implementation is provided as example code, which is customizable to fit different use cases

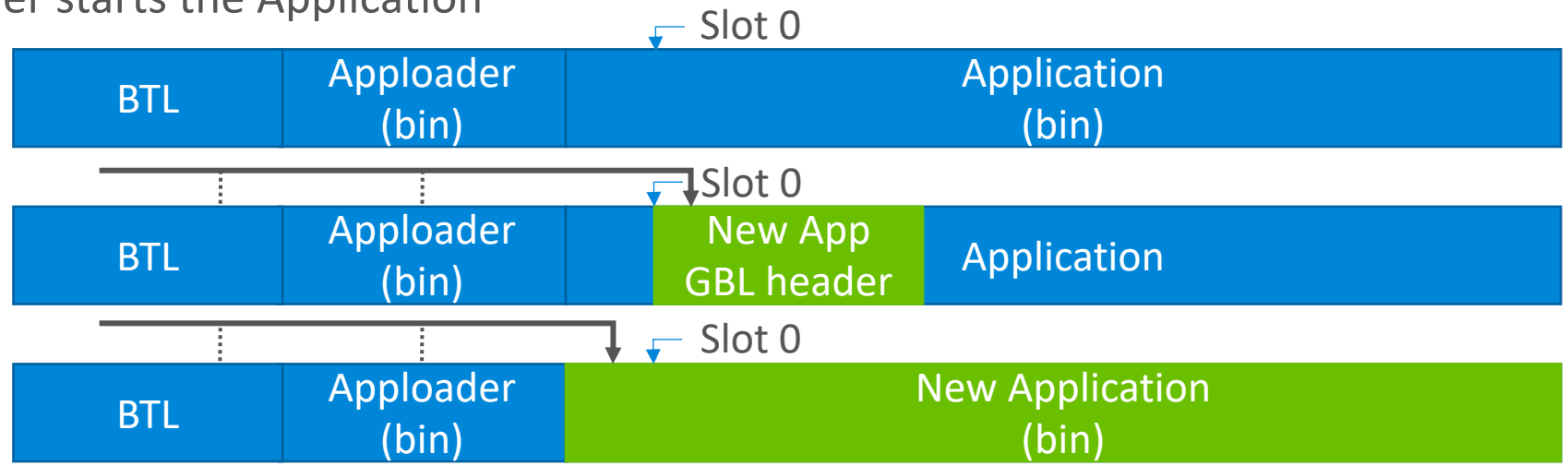
But what if there's not enough storage space for two separate copies of the application image?

What is the Apploader and why is it needed?

- Second Stage Bootloader can load firmware images from
  - UART
  - SPI
  - Internal flash
  - External flash
- The second stage bootloader has no wireless capabilities
- The Apploader is a self-contained miniature version of the Bluetooth stack that is exclusively used for firmware updates
- With the Apploader, the application can be overwritten in place with an image received via Bluetooth, requiring much less storage

OTA DFU with Apploader

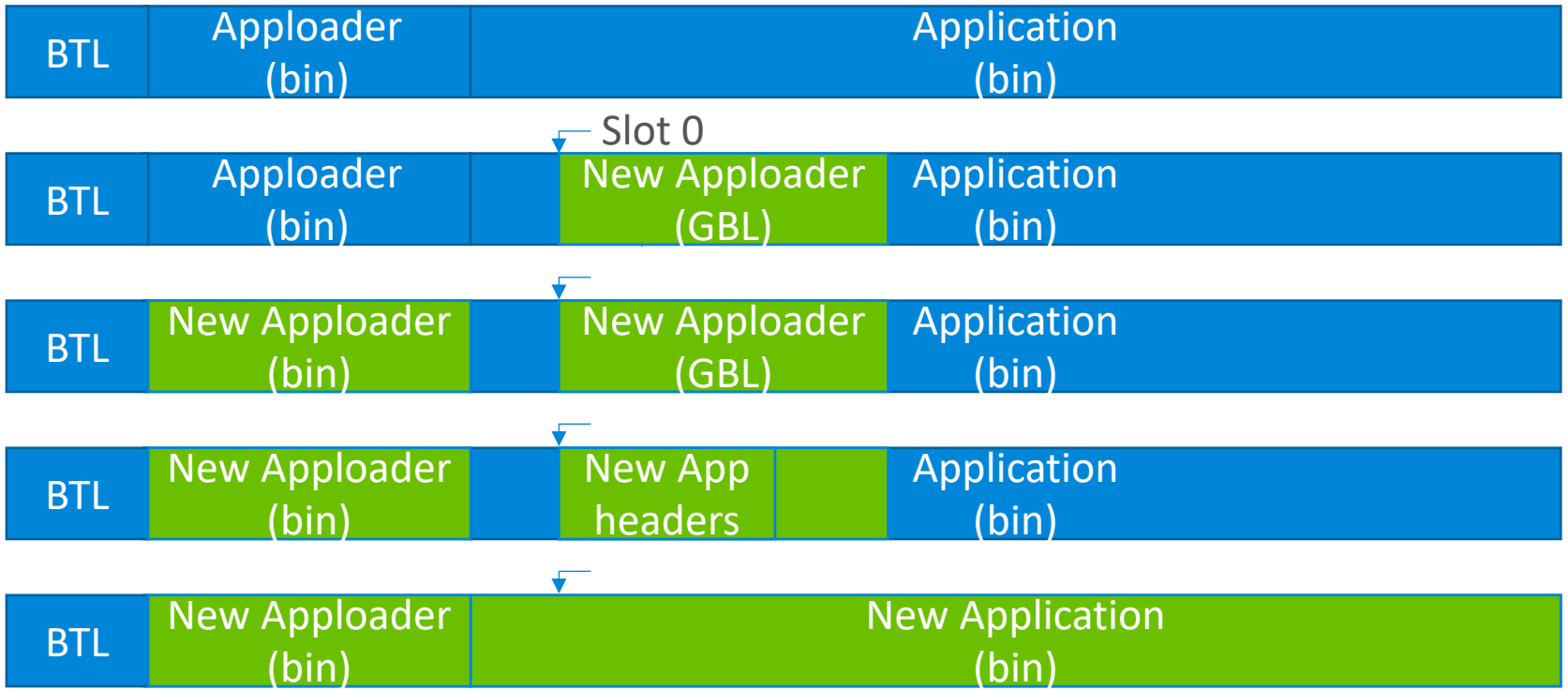
1. Application restarts the device in OTA mode
2. Apploader starts advertising instead of starting the user application
3. Apploader receives the new firmware image via Bluetooth connection
   a) The GBL header is copied to slot 0 for parsing
   b) The rest of the image file is parsed on the fly – The old application is overwritten with the new image
4. Apploader checks the signature of the Application
5. Apploader starts the Application



Upgrading the Apploader

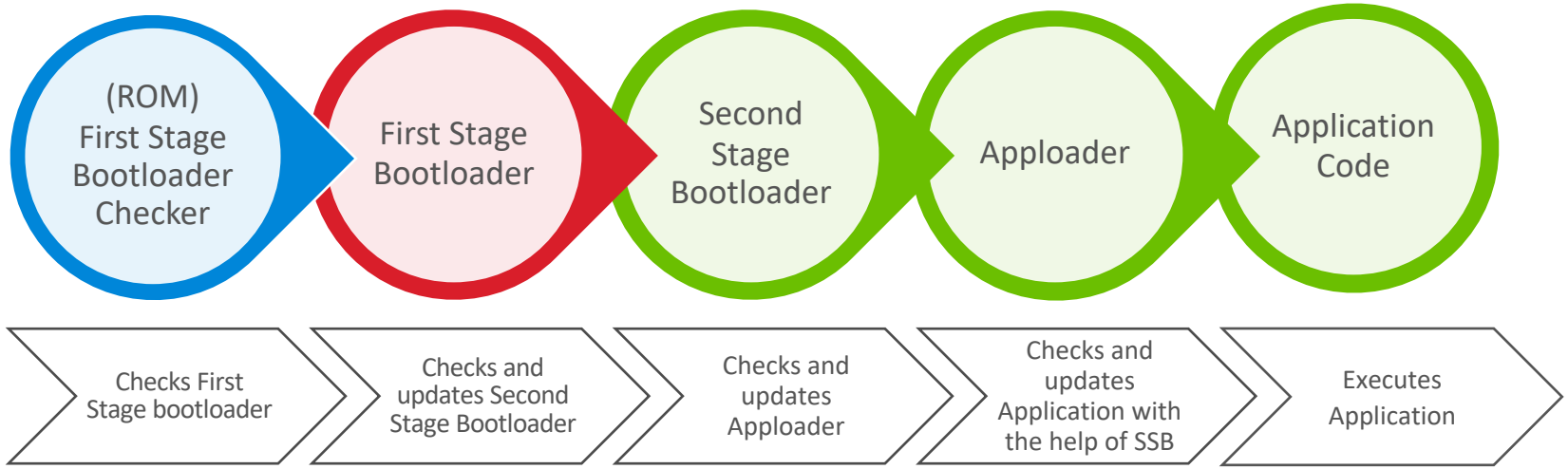
Apploader cannot overwrite itself, hence upgrading the Apploader involves more steps

1. Apploader downloads the new Apploader image to Slot 0 (making the Application image corrupt)
2. Second Stage Bootloader copies the new Apploader from Slot 0
3. Apploader downloads a new Application



Security with Apploader

- Apploader becomes part of the **Chain of Trust**
- Second Stage Bootloader (SSB) checks the signature of the Apploader
- Apploader checks the signature of the Application



Apploader pros and cons

Pros

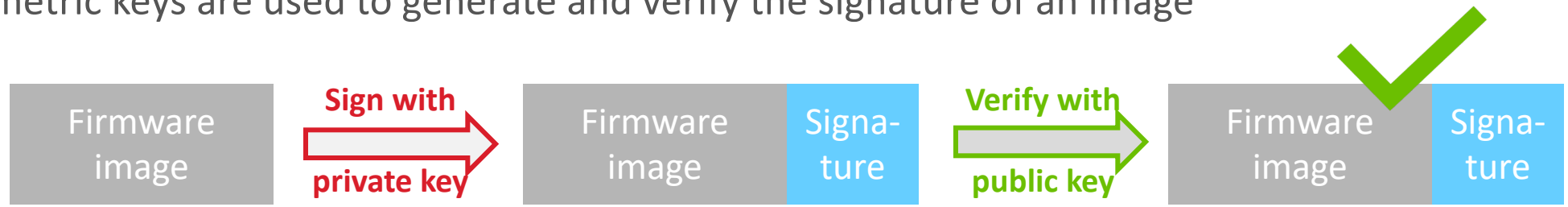
- Self-contained, fully tested OTA solution
- Can be used on any EFR32 device, even with small flash sizes

Cons

- Additional element in the Chain of Trust (has to be signed separately from the Application)
- Not customizable
- Cannot handle secure Bluetooth connections
- Upgrading the Apploader is a two step process

Keys

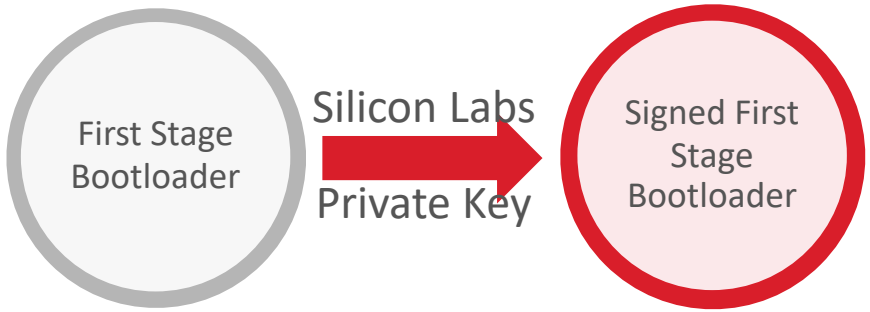
- Asymmetric keys are used to generate and verify the signature of an image

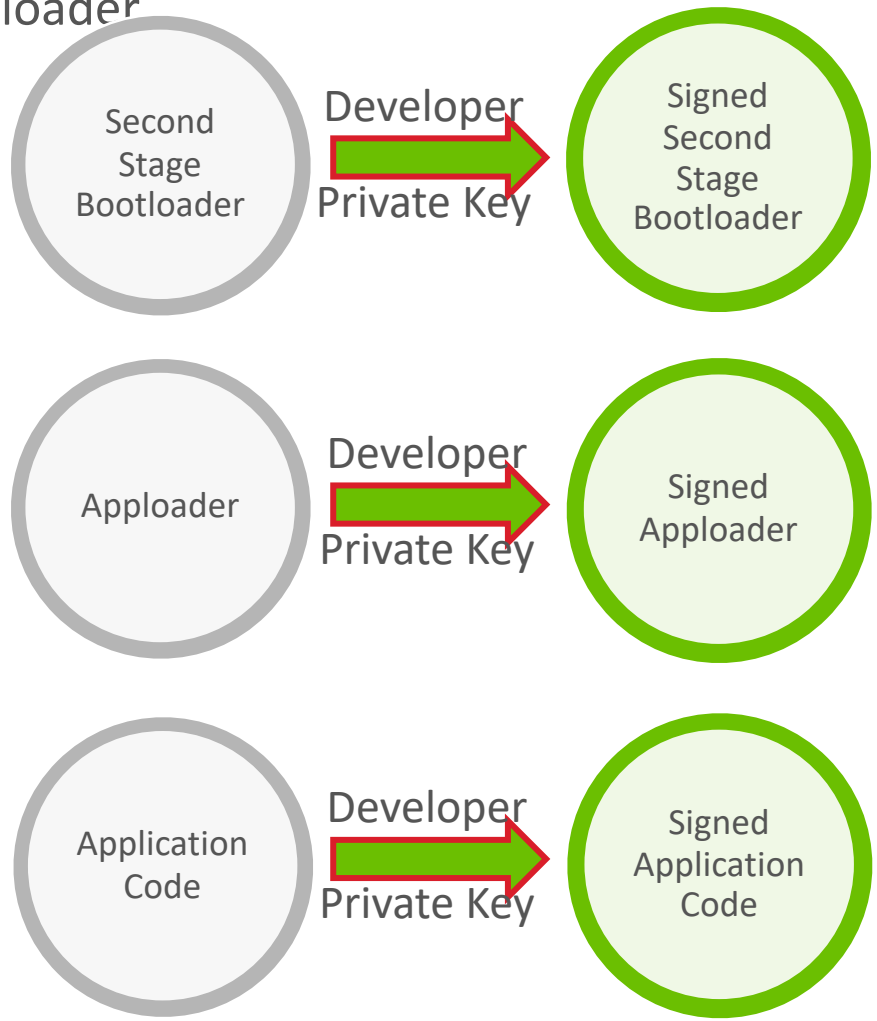


- Private keys are kept secret and only used once to sign the firmware upgrade image
- Public keys are burned into the EFR so that uploaded images can be verified during upgrade and optionally at every reboot
- On BG2x two type of keys are used
  - **Silicon Labs keys** (private key is stored securely at Silicon Labs, and public key is stored in ROM)
  - **Developer keys** (private key is stored securely by the developer, and public key is burned into the device by developer)
- The developer has to create their own developer keys

Using the Private Keys

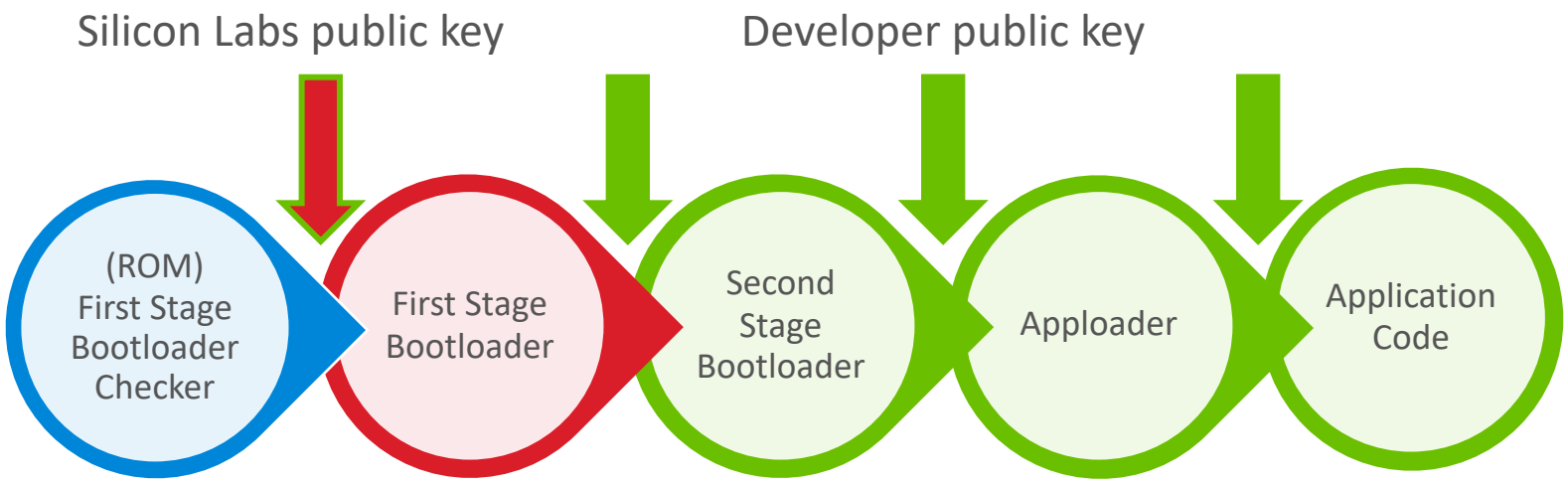
- Silicon Labs private key is used to sign the First Stage Bootloader
- Developer private key is used to sign
  - The Second Stage Bootloader
  - The Apploader
  - The Application





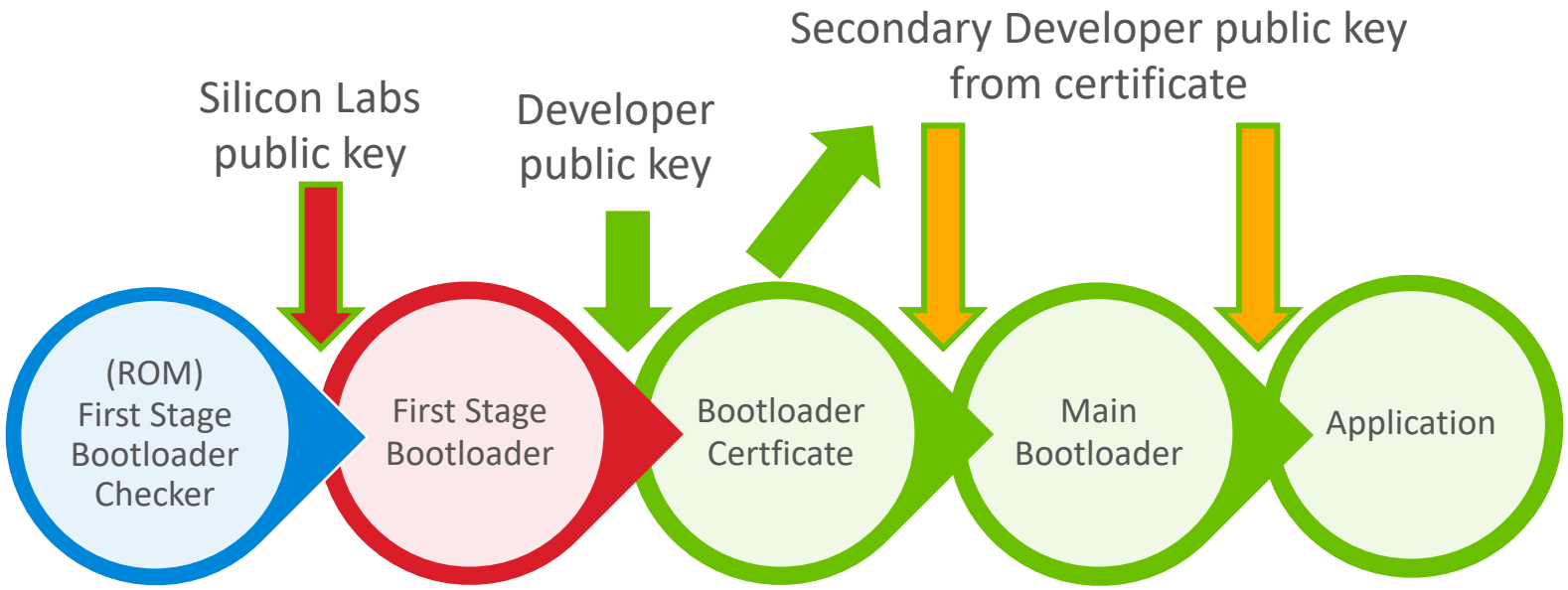
Using the Public Keys

- Silicon Labs public key is used by the First Stage Bootloader Checker to verify the First Stage Bootloader
- Developer public key is used
  - By the First Stage Bootloader to verify the Second Stage Bootloader
  - By the Second Stage Bootloader to verify the Apploader (or the Application when there's no Apploader)
  - By the Apploader to verify the Application



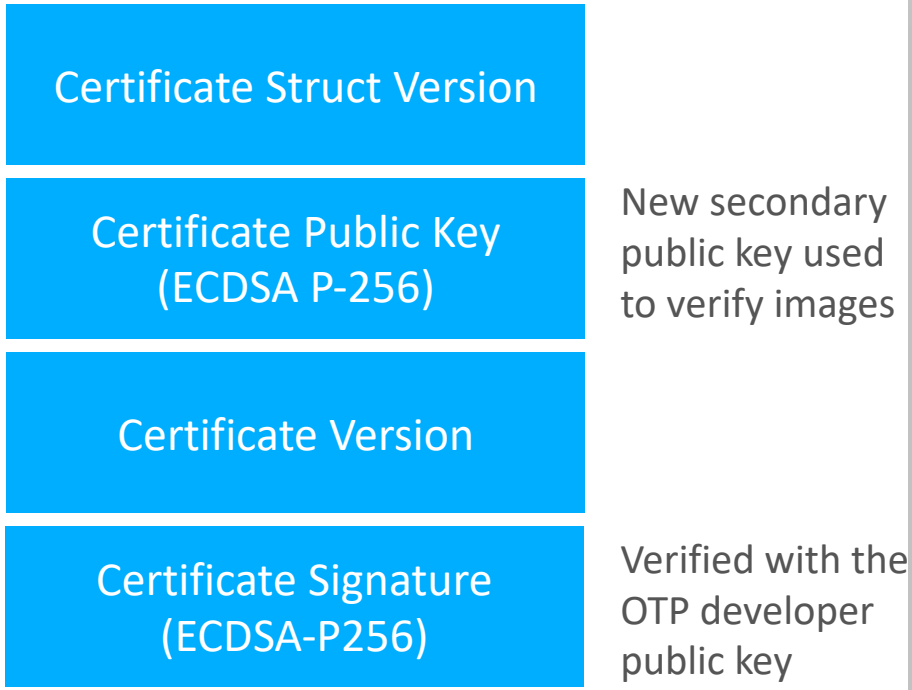
Certificate Based Secure Boot

- BG21/BG22 devices also support a certificate based secure boot
- Allows you to sign images with a secondary private key



Certificate Based Secure Boot

- Use of certificates allows delegation of keys to third parties
  - Example: A hardware manufacturer can control the root key and sign certificates for their customers, who can then sign their own firmware images
- Use of certificates allows the key used to sign the firmware images to be changed/revoked
  - Example: Using a separate key for prototypes and production
  - Root key is in OTP and thus cannot be revoked
- Use of certificates minimizes the use of the root developer key, which minimizes the risk of its exposure
  - Root key used only once to sign the certificate, then the private key associated with the certificate is used to sign the firmware images



Protecting Your Private Keys

- The security of any cryptographic system relies on protecting your private keys
- Simplicity Commander supports creating keys and signing images and certificates
  - Simplicity Commander has no method of securing the private keys it uses to sign images, so it is not recommended to be used to sign your production images
- For maximum security, Silicon Labs recommends using Hardware Security Modules (HSMs) or cloud-based signing services (which generally also use these modules)
  - The private keys remain embedded in the HSM hardware and are never exposed!

BG21/BG22 Secure Bluetooth Manufacturing Checklist

- Write command public key (OTP)
- Write sign public key (OTP)
- Enable secure boot (OTP)
- Configure debug lock options (OTP)
- Flash signed images (bootloader, application, optional apploader)

Useful References

- AN1218: Series 2 Secure Boot with RTSL
- AN1190: Series 2 Secure Debug
- UG162: Simplicity Commander Reference Guide
- UG266: Silicon Labs Gecko Bootloader User's Guide
- UG103.05: IoT Endpoint Security Fundamentals
- Code Example: Implementing OTA Firmware Update in User Application https://docs.silabs.com/bluetooth/latest/code-examples/stack-features/firmware-upgrade/implementing-ota-firmware-update-in-user-application

works with

BY SILICON LABS

SEPTEMBER 9–10, 2020 | VIRTUAL

workswith.silabs.com

TWO DAYS OF TECHNICAL TRAINING
FROM BEGINNER TO ADVANCED