Works With 2021

SEC-301: Hands-On with CPMS Security

Table of Contents

- Overview
- Setup
- References
- Part 1
 - CPMS
 - Application
 - Flash Programming
- Part 1.1
 - CPMS
 - Bootloader
 - Application
 - Image Preparation
 - Flash Programming
- Part 2
 - CPMS
 - Bootloader
 - Application
 - Image Preparation
 - Flash Programming
- Part 2.1
 - CPMS
- Part 3
 - CPMS
 - Bootloader
 - Application
 - Image Preparation
 - Flash Programming

Overview

As IoT devices have become more secure, device configuration has become more complex. The Silicon Labs Custom Part Manufacturing Service (CPMS) makes it easy to configure and order parts with custom security specifications. This lab will walk through the steps for using Simplicity Studio v5 along with CPMS to order custom parts.

Setup

1. *CPMS*: To access CPMS, go to https://cpms.silabs.com/login and log in using your Silicon Labs credentials. If you do not have access initially, just register your account.

2. *Simplicity Commander:* This lab uses the Windows Command Prompt to access the Simplicity Commander CLI. All of the commands line instructions in this lab manual are executed from the *WorksWith-SEC301* directory. If you wish to work out of a different directory, full paths to relevant files must be given.

The pre-work for this course includes instructions for how to add commander to your path. The command line instructions in this lab manual assume that Commander is in your path. If it is not, the full path to commander will need to be given: C:\SiliconLabs\SimplicityStudio\ <version>\developer\adapter_packs\commander\commander.exe.

References

More information about Silicon Labs security features can be found in the following App Notes:

- AN1190: Secure Debug
- AN1218: Secure Boot with RTSL
- AN1222: Production Programming
- AN1247: Secure Vault Tamper
- AN1268: Secure Identity

Part 1

A simple part with a pre-flashed user application

CPMS

- 1. In a browser, open CPMS at https://cpms.silabs.com/login
- 2. Log in using your www.silabs.com account credentials
- 3. Click Create a new Custom Part

☆ // Custom Part Manufacturing Servi	ice	
	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	

1. Part: Select "EFR32MG21B010F1024IM32-B"

Start Creating a new Custom Part	
To get started, select the part to base your custom programming on and give your new OPN a name. On the next screen you will be able to set your custom programming data and request samples be sent to you.	
Select a stock part for you to customize and give your product an alias name. This alias name is only used on this portal for you to remember a specific orde it has no relation to the configuration of your part in any way.	r;
Don't see a part you want? <u>Tell us</u> !	
Start typing to select a Part to custom program EFR32MG21B010F1024IM32-B	
Part Details Product: Wireless Group: ZigBee and Thread Family: EFR32MG21 Series 2 SoCs Flash size: 1024kB	

- 2. *Name:* Enter "*SEC301-Part1*". This name will be used within CPMS to help differentiate between custom devices.
- 3. Estimated Product Order Volume: Select < 1,000 units
- 4. Estimated First Volume Order Time: Select 1-3 months

\ friendly name for you to refer b	to on this portal. This name does not appear in the final chip.	
stimated Production Order Vo	ne? (just guess if you are not sure)	
o < 1,000 units		
🔘 1,000 - 9,999 units		
🔘 10,000 - 99,999 units		
O 100,000 - 999,999 units		
≥ 1,000,000 units		
stimated First Volume Order 1	e? (just guess if you are not sure)	
1-3 months		
4-6 months		
O 6+ months		

- 4. Click Customize. This takes you to the part customization page. Change the following configurations (configurations not listed can be left as the default):
 - 1. *Debug Lock*: This example is only going to set up the firmware, so you can set the debug lock to the minimum setting, 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 commany either from a debugger over DCI or through the mailbox interface. Three properties govern the behavior of the de reduces the general attack surface and prevents information leakage post Silicon Labs manufacturing.	ds to the Secure Element, bug lock. Locking the part

2. Configure Secure Boot, Flash Lock, and Tamper Settings: Off. This will be used in later examples in this lab.

Sonfigure 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 <u>secure boot with RTSL</u> and <u>production programming</u>

Application

- 1. Open Simplicity Studio
- 2. Click File > Import...

SEC301-LabManual.md

Si	5_work	kspace - F	PG12-ChipS	SHOUTER-E	Driver/src	/main.c -	Simp	licity Studio	D™	
File	Edit	Source	Refactor	Navigate	Search	Project	Run	Window	Help	
ଜ ଚ	New Open Open Open Other Recen	File File (Netv Recent Fi Network t Files	work Analy ile (Networ Analyzer A	zer Trace, E rk Analyzer Actions	nergy Pro Trace, En	ofiler, etc ergy Prof	.) ïler, et	A tc.)	lt+Shift+N	> > >
	Close Close	All						Cti	Ctrl+W rl+Shift+W	
	Save Save	As							Ctrl+S	
	Save A	AII t						C	tri+Snitt+S	
	Move. Renan	 ne							F2	
\$	Refres	sh							F5	
	Conve	ert Line D	elimiters To	c						>
£	Print								Ctrl+P	
2	Impor	t								
4	Expor	t								
	Switch	n Workspa	ace							
يك	Netwo	ork Analy	zer Import.							
	Prope	rties							Alt+Enter	
	Restar	t								
	Exit									

3. Click Browse... and navigate to the *WorksWith-SEC301* directory on your computer

4. You should see SEC301-app and SEC301-btl. For now, select SEC301-app.

ct a project to import:					
:\Users\bethorel\Documents\V	orksWith-SEC301	×	/	Brows	e
ected projects: Project Name	Project	Type			
SEC301-app	Simpli	city Studio (.sls)			
SEC301-btl	Simpli	city Studio (.sls)			

- 5. Click Next >, then Next >
- 6. Uncheck the Use default location option to choose a new location to import the project. Navigate to the *WorksWith-SEC301* directory and click Select Folder. This is done so that the project files are all in once place and close to the key files that will be used later in the lab.

Si Import Project	_	×

Project Configuration

Select the project name and location.

Project name: SEC30	I-app			
Use default location	on			
Location: C:\Users\b	ethorel\Documents\\	WorksWith-SEC30	1\SEC301-app	Browse
?	< Back	Next >	Finish	Cancel

7. Click Finish

8. Build the project. This will generate binaries for the project.

File	Edit	Source	Refactor	Navigate	Search	Project	Run	Window	Help
📬	- 8 (ē 🛞 🔻	- 🔨 - 🛪	r 🕶 🤮 🕶 🗄	a ₹ ₹	- 🏷 🔇		⇒ - 🛃	1
🔁 Pi	roject	Explorer	⊠ Build	'GNU ARM	v10.2.1 -	Default'	for pr	oject 'SEC3	01-app'

Flash Programming

1. In CPMS, return to the Flash Programming section

2. Firmware Type: Select App only

n Programming involv	s the addition of customer specific code to a standard product. Customer code in INTEL HEX format is required.
Firmware	
Fill Character 0x FF	
Manuill fill upped a	v upper particular addresses of the flagh with the bute you provide have
we will fill unused o	runspecified addresses of the hash with the byte you provide here.
Firmware Type	runspecified addresses of the hash with the byte you provide here.
Firmware Type	3ootloader only O App and Bootloader
Firmware Type	3ootloader only O App and Bootloader
Firmware Type	Bootloader only O App and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE
Firmware Type App only Intel HEX 	Bootloader only O App and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE
Firmware Type App only Intel HEX 	Bootloader only O App and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE

- 3. Click on CLICK HERE OR DRAG DROP TO UPLOAD A FILE
- 4. Navigate to the directory where you imported *SEC301-app*. If you chose to use the default directory, on Windows this will be in *C:/Users/<username>/SimplicityStudio/v5_workspace*.
- 5. In the *GNU ARM v10.2.1 Default* directory, select *SEC301-app.hex* and click Open. CPMS only accepts Intel Hex files for firmware images.

\vdash \rightarrow \checkmark \uparrow 📜 \Rightarrow This PC \Rightarrow D	ocuments > WorksWith-SEC301 > SEC30	1-app > GNU ARM v10.2.1 - Default	› ر	,	NU ARM v10.2.1 - D
Organize 🔻 New folder					• •
	Name	Date modified	Туре	Size	
	📕 autogen	8/18/2021 2:50 PM	File folder		
챯 Dropbox (Silicon Labs)	gecko_sdk_3.2.1	8/18/2021 2:50 PM	File folder		
lendrive	SEC301-app.hex	8/18/2021 2:50 PM	HEX File	26 KB	
This PC					
3D Objects					
Documents					
Downloads	_				
Music					
Pictures					
Videos					
🕒 OSDisk (C:)					
Mahuark					
Verwork .					
File name: SEC30	1-app.hex			 Custom Files 	~
					Connect

6. You should now be able to see the binary for the application in CPMS

Programming involves the addition of customer specific code to a standard p	product. Customer code in INTEL HEX format is required.
irmware	
Fill Character 0x FF	
	rovido horo
We will fill unused or unspecified addresses of the flash with the byte you p	iovide here.
We will fill unused or unspecified addresses of the flash with the byte you p	ovide here.
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader	lovide here.
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader CUCK HERE OR DRAC D	
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader CLICK HERE OR DRAG D SEC301-app.hex	ROP TO UPLOAD A FILE
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader CLICK HERE OR DRAG D SEC301-app.hex :10000000010020950300009103000091	ROP TO UPLOAD A FILE
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader CLICK HERE OR DRAG D SEC301-app.hex :100000000010002095030000910300009103000000 :1000100091030000910300009103000090 :1000200091030000910300009103000090	ROP TO UPLOAD A FILE
We will fill unused or unspecified addresses of the flash with the byte you p Firmware Type App only Bootloader only App and Bootloader CLICK HERE OR DRAG D SEC301-app.hex :1000000001000295030000910300009103000000 :10001000910300009103000090 :1000200091030000910300009103000090 :1000200091030000910300009103000090 :1000300091030000910300009103000090 :100030009103000091030000910300009103000090 :100030009103000091030000910300009103000090 :10003000910300009103000091030000910300009103000090 Comparison of the flash with the byte you p Click HERE OR DRAG D Click HERE D Click HERE OR DRA	ROP TO UPLOAD A FILE

7. Scroll to the top of the page, and click **PROCEED TO REVIEW**

Title Base Part	SEC301-Part1 EFR32MG21B010F102	4IM32-B			More •••
Select Part	Customize	Review	Processing	Shipping	💿 Done
Customizo Vou	- Dort				
Vour OPN programming d	Fail	wing for comple program	aming Valu can laava this pag	in and	
come back at any time to d	complete your order. Incomplete	e orders are retained 30 o	days from last access.	PROCEE	D TO REVIEW >

You can now review the pricing for your custom part and the security configurations you've entered.
 You will *NOT* actually be ordering parts for these examples, so do *NOT* enter any banking information for this lab.

Part 1.1

Improving security using standard lock

CPMS

1. Click on BACK TO CUSTOMIZATION to go back to the Customization page

Review and Order Your Custom Part



Please review your custom configurations and estimated pricing below. You can modify your customization by clicking Back to Customization. If you have been issued a Coupon, enter it here to redeem, otherwise click Proceed to Payment below to pay with your Credit Card. Samples of your custom part will ship within 4 weeks of payment. In the unlikely event that the custom part cannot be instantiated as defined a refund of the payment will be issued, and if the generated samples do not match the definition, new corrected samples will be generated.

2. Select Standard for Debug Lock. You will see a warning appear indicating that you need to flash a bootloader in order to apply the lock.

ase update your customization to ensure these conditions are met:	SAVED
I Flash a bootloader for locking the part	
ecurity Options	
SE Version v1.2.7 (latest) 🖍	
We recommend using the latest SE version to ensure all patches are in place. We further recommend that you implement the ability to apply updates in your manufacturing line and over the air in the event new vulnerabilities are patched.	SE
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.	
reades the Beneral attack surface and prevents information reakage post sincon cass manufacearing.	

Bootloader

- 1. In Simplicity Studio, import *SEC301-btl* (refer to Part 1 Application for instructions on how to import an SLS project)
- 2. Open SEC301-btl.isc

B main.c 🛛 & SEC301-btLisc 🖾	° 0
Gecko Bootloader, version:1.12.0	► Generate <pre>≪ Preview</pre>
🛦 General 🗸 🕸 Plugins 🗇 Storage 🛱 Callbacks 🗼 Other	
Application configuration Generation directory: Relative to ISC file (CL\bers\bethorefLDocuments\WorksWith-SEC301\SEC301-btl) Select architecture for this application: Boarce EFR32AC218 2.4 of bt. 10 dBm Radio Board (BRD4181C) Part: EFR32MC218010F10/allM32 Toolchair GRU ARM v10.2.1 Edit Architecture	Hardware Configurator Interface Manage Integration between your project and Hardware Configurator inside Simplicity Studio. Configurator Configurator Configurator Reset Hardware Configurator Custom Hardware Configurator Hardware Configurator Integration
Device name: SEC301-bit	
Description	
This sample configuration of the Gecko bootloader configures the bootloader to use the internal main flash to store firmware update images. A The storage configuration is set up to store a single firmware update image at a time, in a single storage stor. The storage store flow store on the storage of the storage store of the storage store of the storage store of the store of the storage store of the store	

3. At the top right, click on Generate

4. Now that the files have been generated, **Build** the project (if the build button is greyed out, you may need to click on the project in the Project Explorer)

Application

- 1. You also need to tell the application to accommodate a bootloader:
- 2. In the SEC301-app project, open SEC301-app.slcp

EC301-app OVERVIEW SOF	TWARE COMPONENTS CONFIGURATION TOOLS	
Target and SDK Selection	Project Details	Project Generators
Wireless Gecko	SEC301-app This example project shows how to blink an LED in a bare-metal configuration. Category Example/Platform Preferred SDK Gecko SDK Suite: Amazon, Bluetooth 3.2.1, Bluetooth Mesh 2.1.1, EmberZNet 6.10.1.0, Flex 3.2.1.0, Hornekit 1.0.1.0, MCU 6.1.1.0, Micrium OS Kernel, OpenThread 1.2.1.0 (GitHub-48b129e74), Platform 3.2.1.0	Simplicity IDE Project A Simplicity IDE project supporting builds for MCUs using C/C+ and assembly files.
EFR32MG21B010F1024IM32 EFR32xG21B 2.4 GHz 10 dBm Radio Board (BRD4181C) Wireless Starter Kit Mainboard (BRD4001A Rev A01)	Import Mode Link sdk and copy project sources	
Change Target/SDK	Force Generation	Edit

- 3. Click on the SOFTWARE COMPONENTS tab
- 4. In the Search bar, search for "bootloader"
- 5. Click on *Platform > Bootloader > Bootloader Application Interface*, and click Install

SEC301-app OVERVIEW	SOFTWARE COMPONENTS C	CONFIGURATION TOOLS	
▼ Filter : Configurable Components □	Installed Components Compo	nents Installed by You	Search keywords, component's name bootloader 🗞
OTA Broadcast Bootloader Client OTA Broadcast Bootloader Server OTA Unicast Bootloader Client	0 ^ 0 0	Bootloader Application Interface	Install Add component to project
OTA Unicast Bootloader Server	•	Description	
 ▼ Test OTA Bootloader Test Common OTA Broadcast Bootloader Test OTA Unicast Bootloader Test 		This component must be added to a project in order to use the Gecko Bootloa project a part of flash memory will be reserved for Bootloader usage in the component also provides a bootloader interface for interacting with the Gec Quality PRODUCTION	der. When this component is part of a application linker file. This ko Bootloader.
OTA Bootloader Interface		Open in Browser	
▼ Platform		Application Interface	
▼ Bootloader			
Bootloader Application Interface		Description	
▼ Services			
Legacy Bootloader Application Interf	ace		View Dependencies

6. Now that the configuration is set, Build the project

Image Preparation

1. CPMS requires the firmware image to be in one file, so you need to merge application and bootloader hex files. This can be done with various tools. This lab uses the Simplicity Commander command line

interface.

2. Open a terminal in the WorksWith-SEC301 directory.

3. Run:

```
commander convert "SEC301-app\GNU ARM v10.2.1 - Default\SEC301-app.hex"
   "SEC301-btl\GNU ARM v10.2.1 - Default\SEC301-btl.hex" -o SEC301-merged.hex
Parsing file SEC301-app\GNU ARM v10.2.1 - Default\SEC301-app.hex...
Parsing file SEC301-btl\GNU ARM v10.2.1 - Default\SEC301-btl.hex...
Writing to SEC301-merged.hex...
DONE
```

This will create SEC301-merged.hex in the WorksWith-SEC301 directory

Flash Programming

- 1. In CPMS, scroll down to Flash Programming and for Firmware Type select App and Bootloader
- 2. Upload the merged hex file, SEC301-merged.hex

ganize • New folder	^			
Quick access	A			• •
Quick access	Name	Date modified	Туре	Size
	📜 keys	8/18/2021 2:44 PM	File folder	
Dropbox (Silicon Labs)	SEC301-app	8/18/2021 2:50 PM	File folder	
OneDrive	📜 SEC301-btl	8/18/2021 3:06 PM	File folder	
	SEC301-merged.hex	8/18/2021 3:10 PM	HEX File	52 KB
This PC				
3D Objects				
E. Desktop				
Documents				
Downloads				
Music				
E Pictures				
Videos				
SDisk (C:)				
Network				
File name: SEC201	merged bey			Custom Files

Programming involv	es the addition of customer specific code to a standard product. Customer code in INTEL HEX format is required	Ι.
Firmware		
Fill Character 0x FF		
Firmware Type		
App only	Bootloader only App and Bootloader 	
App only	Bootloader only App and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE	

Part 2

A part secured against untrusted CMs

CPMS

1. In CPMS, click on the Home icon at the top of the page to go to the CPMS home page



- 2. Create a new custom part. Besides a name of "SEC301-Part2", all the other device ordering information are the same as in Part 1 CPMS (the part is "EFR32MG21B010F1024IM32-B")
- 3. Customize (as before, configurations not listed can be left as the default):
 - 1. Debug lock: Standard
 - 2. Configure Secure Boot, Flash Lock, and Tamper Settings: Yes
 - Enable Secure Boot: yes
 - Require Verify Certificate: no
 - 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 <u>secure boot with RTSL</u> and <u>production programming</u>



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.

4. *Standard Security Keys:* Because Secure Boot is enabled, you need to supply a Secure Boot Key. Ideally, this key would be generated and managed by an HSM. This lab will use commander.

All the keys for this lab have already been generated and are been provided in the *WorksWith-SEC301/keys* directory. For reference, the keys were generated with the command:

commander security genkey --type ecc-p256 --privkey keys\SEC301-<type>priv.pem --pubkey keys\SEC301-<name>-pub.pem

To provision the Secure Boot Key in CPMS, click on the blue upload button in the Secure Boot Key field, then select the *WorksWith-SEC301/keys/SEC301-sign-pub.pem* file.

Secure Boot Key					$\mathbf{\bullet}$
This key is used for binary part of the key you used to .pem or .der file)	authentication and/or OTA upgrade pay o sign your bootloader or application ima	load authentication. If you er age here. (eg. 0x04123456789	nabled secure boot, y 9ABCEDF, total 65 b	ou must provide ytes. You can als	the public o upload a
oen → ✓ ↑ 🖡 > This PC > D	ocuments > WorksWith-SEC301 > kevs		v ق	Search	kevs
anize New folder			-)))))
	Name	Date modified	Туре	Size	
Quick access	0.050201	0/2/2021 10:27 AM	DCDdaala Daawaaat	1 1/10	
Dropbox (Silicon Labs)	SEC301-appcert-priv	8/3/2021 10:27 AM	PGPdesk Document		
	SEC301-appcen-pub	8/3/2021 10.27 AW	PGPdesk Document	1 KB	
OneDrive	SEC301-bt/cert-pub	8/3/2021 10:21 AM	PGPdesk Document	1 KB	
This PC	SEC301-cmd-priv	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
3D Objects	SEC301-cmd-pub	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
Desktop	SEC301-sign-priv	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
Documents	SEC301-sign-pub	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
Downloads					
Music					
Pictures					
Videos					
OSDisk (C:)					
Network					

5. You should now be able to see the public boot key in CPMS

Standard Security Keys
Secure Boot Key 0x 049f1152674e81833acd9548ddfc943dec670cab6e82ee45fec81044f8ef28140c980f828d229a5dd34d1f4b118697efb5adc621a7faa5f5
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. 0x04123456789ABCEDF, total 65 bytes. You can also upload a .pem or .der file)

6. Flash Programming: Because Secure Boot is enabled, the images need to be signed.

Bootloader

- 1. Prepare the bootloader to use a signature:
- 2. In Simplicity Studio, open SEC301-btl.isc

- 3. Click on the Plugins tab, then select Bootloader Core, provides API: core
- 4. Click Enable Secure Boot

🖻 main.c 🛛 & *SEC301-btl.isc 😂 🚢 SEC301-app.slcp				- 6
Gecko Bootloader, version:1.12.0			Generate	« Preview
🎄 General 🗇 Plugins 🛛 🧶 Storage 🛤 Callbacks 🛔 Other				
Plugin configuration				
Use this section to select or unselect the plugins that you want to use in your application				
		Γ		^
٩	≯	Plugin: I Bootloader Core		
✓ □ \$ Communication	^	Quality: 🚱 Unknown plugin quality		
🗌 🗇 BGAPI UART DFU		Description:		
EZSP-SPI		Core library for bootloader		^
A VART XMODEM				
🗌 🗇 XMODEM Parser, provides API: xmodemParser				
✓ ■ \$\$ Core				
Application upgrade version check				~
🖂 🕸 Bootloader Core, provides API: core				
GBL Compression (LZ4)		Options:	Reset to def	faults
GBL Compression (LZMA)		Require signed firmware upgrade files		
✓ Image Parser, provides API: imageParser		Require encrypted firmware upgrade files		
Image Parser with legacy EBL support, provides API: imageParser		Use symmetric key stored in Secure Element storage		
Image Parser without encryption support, provides API: imageParser				
		Enable certificate support		
Delay, provides API: delayDriver		Allow use of public key from manufacturing taken storage		
SPI Master, provides API: spiDriver		Allow use of public key non-manufacturing token storage		
□ 🗣 SPI Slave, provides API: spiSlaveDriver		keject direct signed images		
□ 🗣 UART, provides API: uartDriver		L Enable application rollback protection		
✓ ■ \$ Storage		Minimum application version allowed: 0		
Common Storage, provides API: storageCommon		Prevent write/erase of verified application		
🗹 🇇 Common Storage (single storage slot only), provides API: storageCommon		Prevent bootloader write/erase		
🖂 🕸 Internal Storage, provides API: storage		Skip verification of application on EM4 reset		
SPI Flash Storage, provides API: storage		Pres address of bostoade upgrade image 27769		
👻 🔳 🍣 Utils		base audress of bootroauer upgrade image. 22700		
🗹 🕸 Crypto, provides API: aes, sha, ecdsa		Details (double-click on files to show content):		
Cyclic Redundancy Check, provides API: crc		Located at: C:\SiliconLabs\SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\v3.2\platform\bootloader\core		
Debug	~			`

5. Click Generate

6. Build the project

Application

- 1. Prepare the application to use a signature:
- 2. Copy the *application_properties.c* file (found in the *WorksWith-SEC301* directory) to the *SEC301-app* directory. This prepares space in the application for a signature.

copy application_properties.c SEC301-app

3. Build the project

Image Preparation

- 1. Prepare the hex files in commander:
- 2. Sign the bootloader

```
commander convert "SEC301-btl\GNU ARM v10.2.1 - Default\SEC301-btl.hex" --
secureboot --keyfile keys\SEC301-sign-priv.pem --outfile SEC301-btl-
signed.hex
```

This will create the SEC301-btl-signed.hex signed image file in your WorksWith-SEC301 directory.

Parsing file SEC301-btl\GNU ARM v10.2.1 - Default\SEC301-btl.hex... Found Application Properties at 0x000024a8 Writing Application Properties signature pointer to point to 0x000025e0 Setting signature type in Application Properties: 0x00000001 Image SHA256: ca36debc860cdb720aabe9fdd37dc730172fe34571aedc452b52f9ef5a824264 R = 746AF8EB33BF0432286B2D60E23C827B5CFBF1ED5BB078C3C19F30E36988EA04 S = CE64F9A71A4C69B1C759FEFF6D2F6CA6F1A0D9CC151F7B447D31B8EE0E94770D Writing to SEC301-btl-signed.hex... DONE

3. Sign the application in commander:

```
commander convert "SEC301-app\GNU ARM v10.2.1 - Default\SEC301-app.hex" --
secureboot --keyfile keys\SEC301-sign-priv.pem --outfile SEC301-app-
signed.hex
```

This will create the SEC301-app-signed.hex signed image file in your WorksWith-SEC301 directory.



4. Merge the signed hex files:

```
commander convert SEC301-app-signed.hex SEC301-btl-signed.hex -o SEC301-
merged-signed.hex
```

Parsing file SEC301-app-signed.hex... Parsing file SEC301-btl-signed.hex... Writing to SEC301-merged-signed.hex... DONE

Flash Programming

1. In CPMS, for *Firmware Type* select App and Bootloader and upload the merged signed hex.

	ocuments > WorksWith-SEC301 >		~	Ö	, ○ Search W	orksWith-SEC
anize New folder						• •
	Name	Date modified	Туре	5	Size	
Quick access	kevs	8/18/2021 2·44 PM	File folder			
Dropbox (Silicon Labs)	SEC301-app	8/18/2021 3:26 PM	File folder			
OneDrive	SEC301-btl	8/18/2021 3:23 PM	File folder			
Olieblive	SEC301-app-signed.hex	8/18/2021 3:29 PM	HEX File		26 KB	
This PC	SEC301-btl-signed.hex	8/18/2021 3:25 PM	HEX File		27 KB	
🔰 3D Objects	SEC301-merged.hex	8/18/2021 3:10 PM	HEX File		52 KB	
Desktop	SEC301-merged-signed.hex	8/18/2021 3:30 PM	HEX File		53 KB	
Documents						
Downloads						
Music						
Pictures						
Videos						
🕒 OSDisk (C:)						
Network						
Network						
File name: SEC20	1-merged-signed bex					
nie name, joecou	T mergea signeaties			\sim	Custom Files	
The name. SECSU				~	Custom Files	Canc
Flash Programming				~	Open	Canc
Flash Programming Flash Programming involves th Firmware Fill Character Ox FF We will fill unused or un Firmware Type	ne addition of customer specific code to a sta	andard product. Customer co	ode in INTEL HEX	format is	s required.	Canc
Flash Programming Flash Programming involves th Firmware Fill Character Ox FF We will fill unused or un Firmware Type App only O Boc	ne addition of customer specific code to a stand nspecified addresses of the flash with the byt otloader only App and Bootloader 	andard product. Customer co	ode in INTEL HEX	format is	s required.	Canc
Flash Programming Flash Programming involves th Firmware Fill Character Ox FF We will fill unused or un Firmware Type O App only O Boo	ne addition of customer specific code to a stand nspecified addresses of the flash with the by potloader only () App and Bootloader	andard product. Customer co te you provide here. DRAG DROP TO UPLOAD A F	ode in INTEL HEX 1	format is	s required.	Canc

Part 2.1

Improving security using Secure lock

CPMS

- 1. In CPMS, click on BACK TO CUSTOMIZATION to go back to the Customization page
- 2. Scroll up to *Debug Lock* and select Secure. You will see a warning appear indicating that you need to provide a command key in order to apply the lock.

Please update your customization to ensure these conditions are met:	
Provide a command key for secure debug lock	
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.	

3. Providing a command key works just like the secure boot key. Scroll to the *Standard MCU Keys* and upload the *keys/SEC301-cmd-pub.pem* file into the *Command Key* slot.

rganize 👻 New folder					*==- *
	Name	Date modified	Туре	Size	
Cuick access	SEC301-appcert-priv	8/3/2021 10:27 AM	PGPdesk Document	1 KB	
🗦 Dropbox (Silicon Labs)	SEC301-appcert-pub	8/3/2021 10:27 AM	PGPdesk Document	1 KB	
	SEC301-btlcert-priv	8/3/2021 10:21 AM	PGPdesk Document	1 KB	
Oleblive	SEC301-btlcert-pub	8/3/2021 10:21 AM	PGPdesk Document	1 KB	
🞐 This PC	🔒 SEC301-cmd-priv	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
🧊 3D Objects	🔒 SEC301-cmd-pub	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
E. Desktop	🔒 SEC301-sign-priv	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
Documents	🔒 SEC301-sign-pub	8/3/2021 10:05 AM	PGPdesk Document	1 KB	
Downloads					
Music					
Pictures					
🖶 Videos					
SDisk (C:)					
Network					
File name: SEC20	1 cmd pub			Custom Filor	
				Custom mes	1
				Open	Cancel
Command Kay					
	1f7704905029256ab5096c2a297221af54f9	dd3f3_8005_598662d7c69b50)12/176b03c2e3/fef373	d0b7728f5676d0	

Now you have a part that only boots signed images, doesn't allow debug access except to those with the private key, and comes shipped with your own firmware.

Part 3

A secure part using certificate chains

CPMS

- 1. In CPMS, click on the Home icon at the top of the screen to go to the CPMS home page
- 2. Create a new custom part. Besides a name of "SEC301-Part3", all the other device ordering information are the same as in Part 1 CPMS (the part is "EFR32MG21B010F1024IM32-B")
- 3. Customize:
 - 1. Debug lock: Secure
 - 2. Configure Secure Boot, Flash Lock, and Tamper Settings: Yes
 - Secure Boot: Yes
 - Require Verify Certificate: Yes. Enabling this feature will prevent the device from booting any images that are not signed with a valid certificate. To use this feature, you need to update your signature configurations to use certificates.
 - Enable Anti Rollback: Yes. Enabling this feature will prevent the device from "upgrading" to firmware with a firmware version number that is lower or equal to the current firmware version. This prevents attackers from exploiting patched vulnerabilities and allows for the revocation of compromised certificates.
 - Flash Page Locking: Full. This setting will lock all pages including the second stage bootloader and application. If the image and signature does not end on a page boundary and your system is tight on memory, you can select Narrow to leave the last (partially unused) page unlocked for future use.
 - 3. Standard Security Keys:
 - 1. Import the secure boot key (*keys/SEC301-sign-pub*) as in Part 2 CPMS
 - 2. Import the command key (keys/SEC301-cmd-pub) as in Part 2.1 CPMS
 - 3. OTA Decryption Key: If you plan to encrypt future firmware upgrades, you should provision an OTA key and set appropriate settings in the btl configuration (highly recommended, but not covered by this lab)

Bootloader

- 1. Prepare the Bootloader to use certificates:
- 2. In Simplicity Studio, open SEC301-btl.isc
- 3. Click on the Plugins tab, then select Bootloader Core, provides API: core
- 4. Click Enable certificate support

🖻 main.c 🛛 🚓 *SEC301-btl.isc 😂 🚢 SEC301-app.slcp 🛛 🕮 brd4181c_efr32mg21b010f1024im32.hwconf		-
O Gecko Bootloader, version:1.12.0		Generate « Preview
🎄 General 🗇 Plugins 🛛 🚸 Storage 🛤 Callbacks 🛔 Other		
Plugin configuration Use this section to select or unselect the plugins that you want to use in your application	Dusin & Rationder Con	^
	Plugin: ◆ Bodtoader Core Quality: € Unknown plugin quality Description: Core library for bootloader	Reset to defaults
♀ Cypto, provides APE ase, sha, ecida ♀ Cyclic Redundancy Check, provides APE trc ♀ Debug ♀ EMLIB ● PEOP Contraction ♀ GPIO activation ♀ GPIO activation ♀ GPIO activation ♀ GPIO activation ♀ FMID Set Manager ♀ Token Management, provides APE tokenManagement ♀ mode TLS	Details (double-click on files to show content):	v

- 5. Click Generate
- 6. Build the project

Application

- 1. Prepare the App to use certificates:
- 2. In Simplicity Studio, open application_properties.c
 - SEC301-app [GNU ARM v10.2.1 Default] [EFR32MG21B010F1024
 - > 🗱 Binaries
 - Includes
 - 🗧 🗁 autogen
 - > 🗁 config
 - > > b gecko_sdk_3.2.1
 - > B GNU ARM v10.2.1 Default
 - > 🖻 app.c
 - > h app.h
 - > application_properties.c
 - > 🖻 blink.c
 - > 脑 blink.h
 - > 🖻 main.c
 - imported_project_report.html
 - SEC301-app.pintool
 - SEC301-app.slcp
 - SEC301-app.slps
- 3. Update *application_properties.c* by adding the following lines after

#define APP_PROPERTIES_VERSION (0UL)

and before

// Application properties for secure boot

const ApplicationProperties_t sl_app_properties = {

```
// Application properties for secure boot
const ApplicationCertificate_t sl_app_certificate = {
.structVersion = APPLICATION_CERTIFICATE_VERSION,
.flags = { 0U },
.key = { 0U },
.version = 0,
.signature = { 0U },
};
```

4. Update the ApplicationProperties_t struct (also in *application_properties.c*) to point to the new certificate:

.cert = (ApplicationCertificate_t *)&sl_app_certificate,

🖸 r	nain.c	🚜 SEC301-btl.isc	🖴 SEC301-app.slcp	is brd4181c_efr32mg21b010f1024im32.hwconf	■ application_properties.c ≅
1					
2	#inc]	.ude <stddef.h></stddef.h>			
3	#inc]	ude "application	n_properties.h"		
4	Ļ				
5	5 // Ap	plication version	on number (uint32_	t) for anti- <u>rollback</u>	
6	#defi	.ne APP_PROPERTI	ES_VERSION (0UL)		
7	′ // Ap	plication prope	rties for secure b	oot	
8	const	ApplicationCer	tificate_t sl_app_	_certificate = {	
9	.stru	ctVersion = APP	LICATION_CERTIFICA	TE_VERSION,	
10) .†lag	s = { 00 },			
11	key	= { 00 },			
12	.vers	100 = 0,			
	s .sigr	$ature = \{ 00 \},$			
14	· }; : // ^-	alication propo	ntios for socuro h	aat	
16		Application prope	nontion t cl ann n	nonontion - (
17	magi	$c = \Delta DDI TCATTON$		oper cies = 1	
18	stri	$c = A \Gamma LICATION$	I TCATION PROPERTIE	S VERSTON	
19	sign	atureType = APP	I TCATTON STGNATURE	NONE.	
20	.sigr	atureLocation =	0.		
21	app	= {	-,		
22	.type	e = APPLICATION	TYPE_MCU,		
23	.vers	$ion = APP_PROPE$	RTIES_VERSION,		
24	.capa	bilities = 0UL,			
25	.prod	$ uctId = \{ 0U \},$			
26	;},				
27	.cert	= (Application	Certificate_t *)&s	l_app_certificate,	
28	long.	TokenSectionAdd	ress = NULL,		
29	};				
30)				

5. Build the project

Key Note

1. To use certificates, more keys are needed. This lab will use 3 key pairs:

- SEC301-sign: The public key is provisioned on the device and is used to verify bootloader certificates. The private key is held safe in an HSM (although this lab uses commander) and is used to sign bootloader certificates.
- 3. *SEC301-btlcert*: This public key is inserted into the bootloader certificate and is used to verify application certificates. The private key is used to sign bootloader images and application certificates.
- 4. *SEC301-appcert*: This public key is inserted into application certificates. The private key is used to sign application images.

These 2 new key pairs (*SEC301-btlcert* and *SEC301-appcert*) are provided in the *WorksWith_SEC301* directory, and were generated with the commander instruction indicated in Part 2 - CPMS.

Image Preparation

- 1. Prepare the hex files in commander:
- 2. To generate the bootloader signature, run:

```
commander util gencert --cert-type secureboot --cert-version 0 --cert-
pubkey keys\SEC301-btlcert-pub.pem --sign keys/SEC301-sign-priv.pem --
outfile SEC301-btlcert.bin
```

3. To sign the bootloader, run:

```
commander convert "SEC301-btl\GNU ARM v10.2.1 - Default\SEC301-btl.hex" --
secureboot --certificate SEC301-btlcert.bin --keyfile keys\SEC301-btlcert-
priv.pem --outfile SEC301-btl-certsigned.hex
```

4. You can verify that this signed image is valid using:

```
commander util verifysign SEC301-btl-certsigned.hex --verify keys\SEC301-
sign-pub.pem
```

The output should look like:

```
Parsing file SEC301-btl-certsigned.hex...
Found application properties at 0x00002704
Found certificate at 0x0000267c
Successfully verified certificate signature with verification key.
Using certificate key to verify application signature.
Successfully verified application signature.
DONE
```

5. To generate the signature for the application, run:

commander util gencert --cert-type secureboot --cert-version 0 --certpubkey keys\SEC301-appcert-pub.pem --sign keys\SEC301-btlcert-priv.pem -outfile SEC301-appcert.bin

6. To sign the application, run:

```
commander convert "SEC301-app\GNU ARM v10.2.1 - Default\SEC301-app.hex" --
secureboot --certificate SEC301-appcert.bin --keyfile keys\SEC301-appcert-
priv.pem --outfile SEC301-app-certsigned.hex
```

7. You can verify that this signed image is valid using:

```
commander util verifysign SEC301-app-certsigned.hex --verify keys\SEC301-
btlcert-pub.pem
```

The output should look like:

```
Parsing file SEC301-app-certsigned.hex...
Found application properties at 0x00006220
Found certificate at 0x00006198
Successfully verified certificate signature with verification key.
Using certificate key to verify application signature.
Successfully verified application signature.
DONE
```

8. Lastly, combine the signed and certified images:

commander convert SEC301-app-certsigned.hex SEC301-btl-certsigned.hex -o
SEC301-merged-certsigned.hex

Flash Programming

- 1. In CPMS, scroll to Flash Programming and select App and Bootloader for Firmware Type
- 2. Upload SEC301-merged-certsigned.hex

pp tl pp-certsigned.hex pp-signed.hex tl-certsigned.hex tl-certsigned.hex tl-signed.hex herged.ecrtsigned.hex herged-signed.hex	Date modified 8/18/2021 2:44 PM 8/18/2021 3:26 PM 8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:25 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:30 PM	Type File folder File folder File folder HEX File	Size 27 KB 26 KB 29 KB 27 KB 52 KB 53 KB 53 KB	
pp tl pp-certsigned.hex pp-signed.hex tl-certsigned.hex tl-signed.hex herged.hex herged.certsigned.hex herged-certsigned.hex	Date modified 8/18/2021 2:44 PM 8/18/2021 3:26 PM 8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	Type File folder File folder HEX File HEX File HEX File HEX File HEX File	Size 27 KB 26 KB 29 KB 27 KB 52 KB 53 KB 53 KB	
pp til pp-certsigned.hex pp-signed.hex tt-certsigned.hex tt-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	Date modified 8/18/2021 2:44 PM 8/18/2021 3:26 PM 8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:30 PM	Type File folder File folder HEX File HEX File HEX File HEX File HEX File HEX File	Size 27 KB 26 KB 29 KB 27 KB 52 KB 53 KB 53 KB	
pp tl pp-certsigned.hex pp-signed.hex tl-certsigned.hex tl-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 2:44 PM 8/18/2021 3:26 PM 8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:30 PM	File folder File folder HEX File HEX File HEX File HEX File HEX File HEX File	27 KB 26 KB 29 KB 27 KB 52 KB 53 KB	
pp tl pp-certsigned.hex pp-signed.hex tl-certsigned.hex tt-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:26 PM 8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	File folder File folder HEX File HEX File HEX File HEX File HEX File HEX File	27 KB 26 KB 29 KB 27 KB 52 KB 53 KB	
tl pp-certsigned.hex pp-signed.hex tl-certsigned.hex tt-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:23 PM 8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	File folder HEX File HEX File HEX File HEX File HEX File HEX File	27 KB 26 KB 29 KB 27 KB 52 KB 53 KB	
pp-certsigned.hex pp-signed.hex ttl-certsigned.hex ttl-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:45 PM 8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File HEX File HEX File HEX File HEX File	27 KB 26 KB 29 KB 27 KB 52 KB 53 KB	
pp-signed.hex tl-certsigned.hex tl-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:29 PM 8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File HEX File HEX File HEX File	26 KB 29 KB 27 KB 52 KB 53 KB	
tl-certsigned.hex tl-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:47 PM 8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File HEX File HEX File	29 KB 27 KB 52 KB 53 KB	
tl-signed.hex herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:25 PM 8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File HEX File HEX File	27 KB 52 KB 55 KB 53 KB	
herged.hex herged-certsigned.hex herged-signed.hex	8/18/2021 3:10 PM 8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File HEX File	52 KB 55 KB 53 KB	
herged-certsigned.hex	8/18/2021 3:47 PM 8/18/2021 3:30 PM	HEX File HEX File	55 KB 53 KB	
herged-signed.hex	8/18/2021 3:30 PM	HEX File	53 KB	
iex				
nex				
			✓ Custom File	25
			Open	Cancel
p and Bootloader				
CLICK HERE OR D	DRAG DROP TO UPLOAD A	A FILE		
01000066 00000006C 00000097 000000BE 4800E0D4				Î
	p and Bootloader CLICK HERE OR D 01000066 0000006C 0000008E 4800E0D4	p and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD / CLICK HERE OR DRAG DROP TO UPLOAD /	p and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE D1000066 00000097 0000000BE 4800E0D4 0000034	p and Bootloader CLICK HERE OR DRAG DROP TO UPLOAD A FILE D1000066 00000006C 00000097 0000008E 4800E0D4 0000034