AN1268: Authenticating Silicon Labs Devices Using Device Certificates

This application note describes how to authenticate a device as a genuine Silicon Labs product at any time during its life. The digital certificates for secure identity are stored in the device and the Silicon Labs Server.

This secure identity feature is only available on Secure Vault devices.

KEY POINTS

- Secure identity on Secure Vault devices
- Device certificate options
- Remote authentication process
- Examples for certificate chain verification and remote authentication

1. EFR32 Series 2 Device Security Features

Protecting IoT devices against security threats is central to a quality product. Silicon Labs offers several security options to help developers build secure devices, secure application software, and secure paths of communication to manage those devices. Silicon Labs' security offerings were significantly enhanced by the introduction of the EFR32 Series 2 products that included a Secure Element. The Secure Element is a tamper-resistant component used to securely store sensitive data, keys and to execute cryptographic functions and secure services.

The Secure Element is the foundation of two core security functions:

- Secure Boot: Process where the initial boot phase is executed from an immutable memory (such as ROM) and where code is authenticated before being authorized to be executed.
- Secure Debug access control: The ability to lock access to the debug ports for operational security, and to securely unlock them when access is required by an authorized entity.

Some EFR32 Series 2 products offer additional security options through Secure Vault. Secure Vault is a dedicated security CPU that isolates cryptographic functions and data from the host processor core. Devices with Secure Vault offer the following security features:

- Secure Key Storage: Protects cryptographic keys by "wrapping" or encrypting the keys using a root key known only to the Secure Vault.
- Anti-Tamper protection: A configurable module to protect the device against tamper attacks.
- Device authentication: Functionality that uses a secure device identity certificate along with digital signatures to verify the source or target of device communications.

A Secure Element Manager and other tools allow users to configure and control their devices both in house during testing and manufacturing, and after the device is in the field.

1.1 User Assistance

In support of these products Silicon Labs offers white papers, webinars, and documentation. The following table summarizes the key security documents:

Document	Summary	Applicability
AN1190: Series 2 Secure Debug	How to lock and unlock EFR32 Series 2 debug access, including background information about the Secure Element	EFR32 Series 2
AN1218: Series 2 Secure Boot with RTSL	Describes the secure boot process on EFR32 Series 2 devices using Secure Element	EFR32 Series 2
AN1247: Anti-Tamper Protection Configuration and Use	How to program, provision, and configure the anti-tamper module	EFR32 Series 2 with Secure Vault
AN1268: Authenticating Silicon Labs Devices using Device Certificates (this document)	**How to authenticate a device using secure device certificates and signatures, at any time during the life of the product**	**EFR32 Series 2 with Secure Vault**
AN1271: Secure Key Storage	How to securely "wrap" keys so they can be stored in non-volatile storage.	EFR32 Series 2 with Secure Vault
AN1222: Production Programming of Series 2 Devices	How to program, provision, and configure security information using Secure Element during device production	EFR32 Series 2

1.2 Key Reference

Public/Private keypairs along with other keys are used throughout Silicon Labs security implementations. Because terminology can sometimes be confusing, the following table lists the key names, their applicability, and the documentation where they are used.

Table 1.1.

Key Name	SE Manager ID	Customer Programmed	Purpose	Used in
Public Sign key (Sign Key Public)	SL_SE_KEY_SLOT_APPLICATION_SECURE_BOOT_KEY	Yes	Secure Boot binary authentication and/or OTA upgrade payload authentication	AN1218 (primary), AN1222
Public Command key (Command Key Public)	SL_SE_KEY_SLOT_APPLICATION_SECURE_DEBUG_KEY	Yes	Secure Debug Unlock or Disable Tamper command authentication	AN1190 (primary), AN1222, AN1247
OTA Decryption key (GBL Decryption key) aka AES-128 Key	SL_SE_KEY_SLOT_APPLICATION_AES_128_KEY	Yes	Decrypting GBL payloads used for firmware upgrades	AN1222 (primary), UG266
Attestation key aka Private Device Key	SL_SE_KEY_SLOT_APPLICATION_ATTESTATION_KEY	No	Device authentication for secure identity	AN1268

2. Device Compatibility

This application note supports Series 2 device families with Secure Vault, and some functionality is different depending on the device.

Wireless SoC Series 2 device familieswith Secure Vault consist of:

- EFR32BG21B
- EFR32MG21B

3. Introduction

One of the biggest challenges for connected devices is post-deployment authentication. Silicon Labs' factory trust provisioning service with optional secure programming provides a secure device identity certificate, analogous to a birth certificate, for each individual silicon die during integrated circuit (IC) manufacturing. This enables post-deployment security, authenticity and attestation-based health checks. The device certificate guarantees the authenticity of the device for its lifetime. When the certificate is checked, a digital signature confirms that the certificate received has not been tampered with.

Certificates can now be used to authenticate Internet of Things (IoT) devices as well as Internet servers, now that Silicon Labs' Secure Vault devices have both cryptographic acceleration in hardware and tamper-resistant storage to handle digital certificate operations.

The digital signature and certificates are major cryptographic tools to verify the device is authentic. These tools are described in the following sections.

3.1 Digital Signature

The digital signature is used to protect integrity and authenticity of an electronic message.

Digital Signature Example:

Alice wants to give data to Bob, and Bob wants to make sure that the data came from Alice and has not been tampered with. Alice has a private/public key pair, and has previously given Bob her public key.

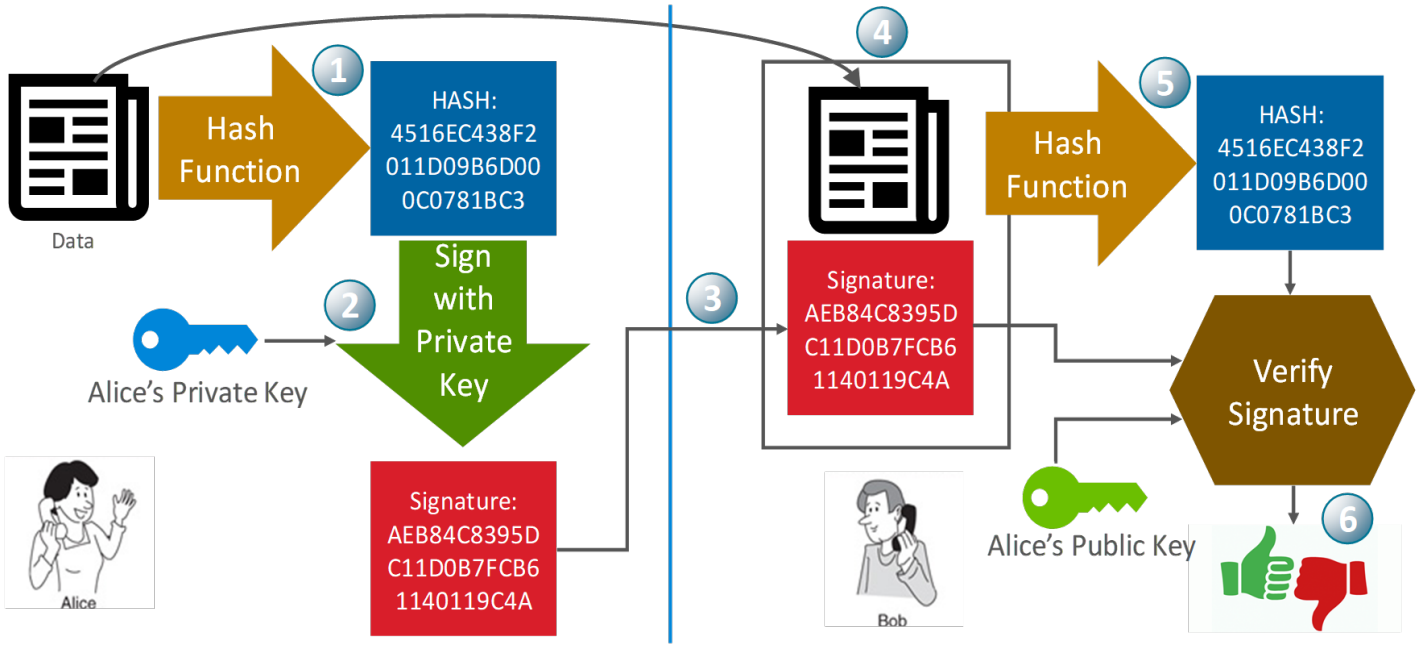


Figure 3.1. Digital Signature

1. Alice generates the hash (for example SHA256) of the data.
2. Alice's private key is used to sign the hash to create a signature. The hash is signed instead of the data itself because the signing operation is slow. Therefore it is more efficient to sign the hash instead of the arbitrarily large data.
3. The signature is attached to the end of the data.
4. The data and signature are given to Bob.
5. Bob independently generates the hash of the data.
6. The signature is verified with the hash and Alice's public key, which results in a true or false outcome indicating if the data is valid.

Note: This scheme requires distribution of Alice's public key.

3.2 Digital Certificates and Chain of Trust

In Figure 3.1 Digital Signature on page 5, Bob already had access to Alice's public key, which he trusted. However, it is not always feasible to pre-share a public key with everyone for secure identity verification and no mechanism to revoke or for revoking or inactivating the public key in case it gets stolen.

A digital certificate is simply a small, verifiable data file that contains identity credentials and a public key. That data is then signed either with the corresponding private key, or a different private key. The digital certificate can be used to prove the ownership of a public key.

- If it is signed using the corresponding private key, it is called a self-signed certificate.
- If it is signed by another private key, the owner of that private key is acting as a Certificate Authority (CA).
- A Certificate Authority (CA) is a trusted third party by both the owner and party relying on the certificate.

Concatenation of digital certificates builds a chain of trust.

- At the root of the chain is a self-signed certificate, called a root certificate or a CA certificate.
- The root or CA certificate can be used to sign another certificate.

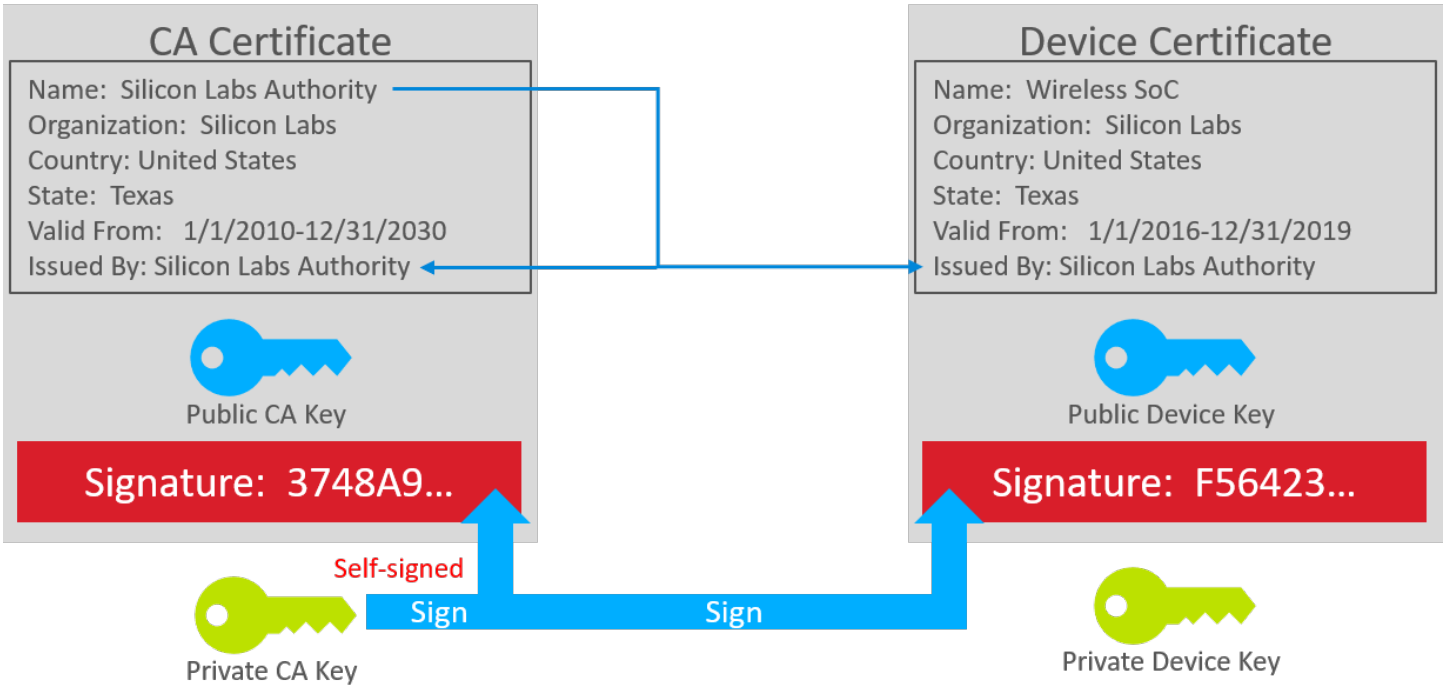


Figure 3.2. Digital Certificates and Chain of Trust

Note: The private key is never included as part of the certificate – it must be stored separately and kept private. The security of the scheme relies on protecting the private keys.

3.3 Digital Certificates Verification

This section illustrates the process shown in Figure 3.1 Digital Signature on page 5, but using digital certificates.

Digital Certificates Verification Example:

Alice wants to give data to Bob, signed with her private key. Alice has a digital certificate signed by a trusted third party (CA) in addition to her private key. Bob has a certificate from the trusted CA but nothing else is previously shared.

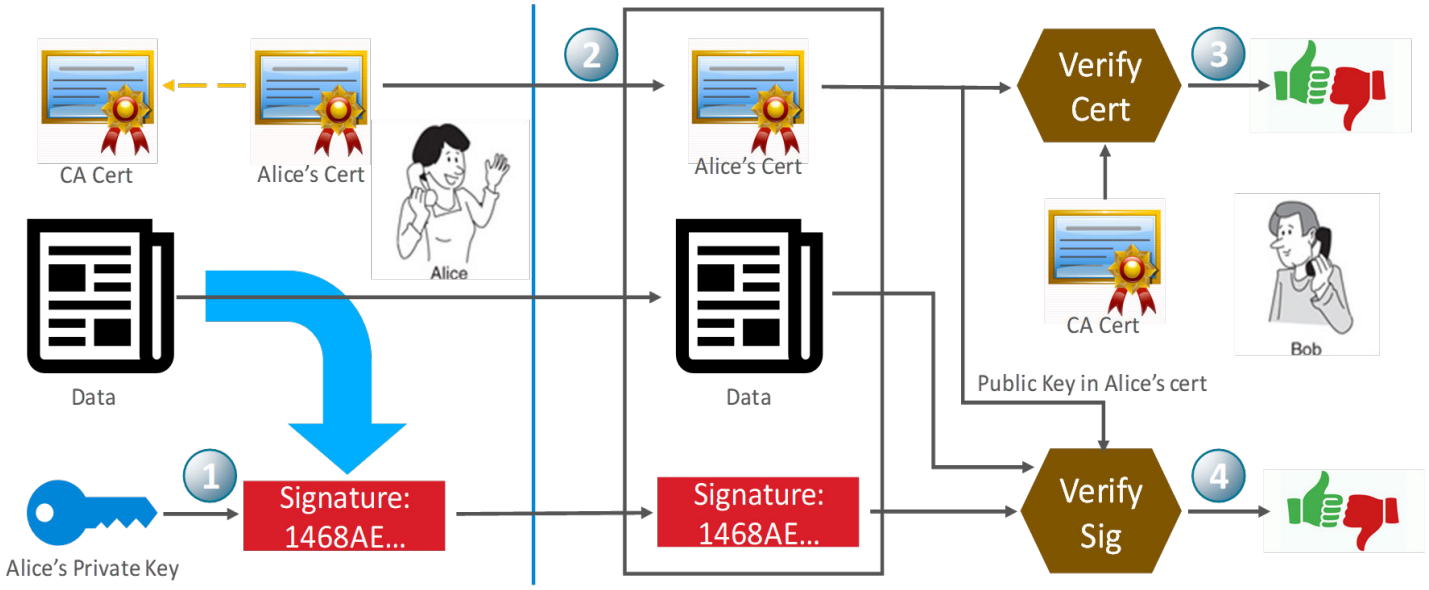


Figure 3.3. Digital Certificate Verification

1. Alice uses her private key to sign the data.
2. Alice gives the data, signature, and her certificate to Bob.
3. Bob first verifies that the Alice's certificate is valid, to prove Alice is the owner of the certificate's public key. This is done by verifying that Alice's certificate contains a valid signature created by the CA.
4. Bob then verifies the signature of the data using the public key in Alice's certificate.

Note: The hash process in Figure 3.1 Digital Signature on page 5 is skipped in this example.

4. Secure Identification on Secure Vault Devices

The goal of secure identification is to prove the ownership of a device's unique public key to an external service. It enables the external service to identify the device as legitimate and to authenticate device-generated data or messages.

4.1 Chain of Trust

The chain of trust on Secure Vault devices is illustrated in Figure 4.1 Chain of Trust on page 8.

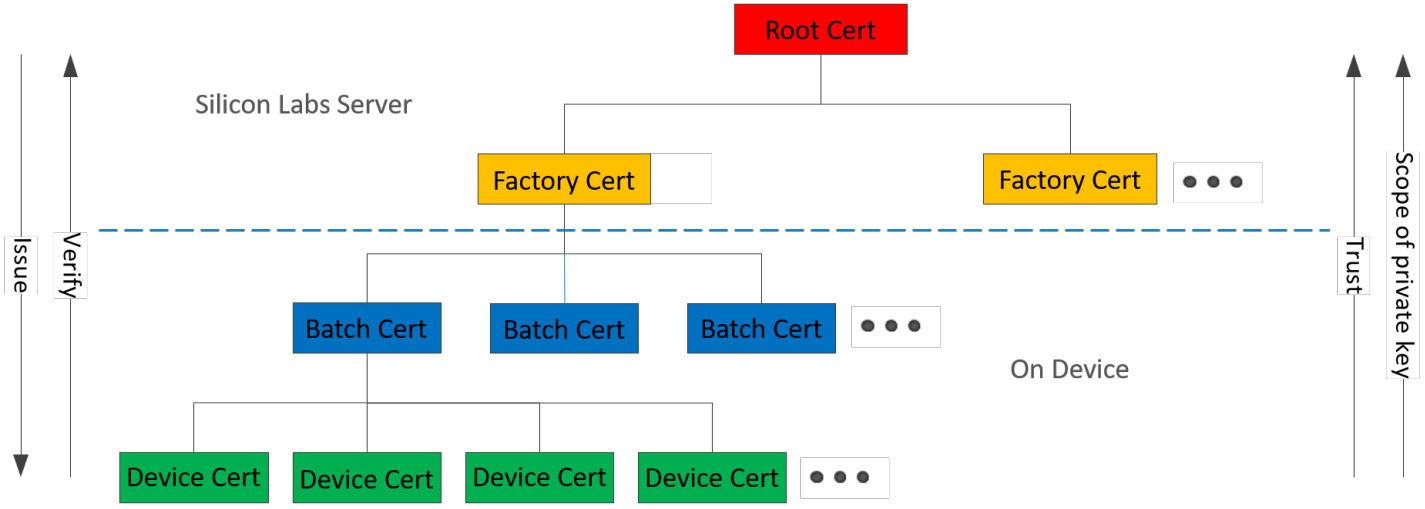


Figure 4.1. Chain of Trust

- Silicon Labs is a Certificate Authority (CA).
- The root certificate and factory certificate are stored in the Silicon Labs Server.
- The factory certificate is static per factory.
- The batch certificate and device certificate are stored on the device.
- The batch certificate is rolled per production batch.
- The device certificate is a unique cryptographic identity.
- All certificates are X.509 standard format.
  - TLS-compliant: Standard endpoint authentication methods are used in internet communications
  - Signature algorithm: ECDSA-prime256v1 with SHA256
- Each certificate in the chain is signed by the certificate above it (Figure 4.3 Signing for Certificates on page 10).

Note: A certificate can be revoked if needed, for instance if security issues arise. The certificate revocation lists are stored in the Silicon Labs Server.

4.2 Device Certificate

The device certificate example is described in Figure 4.2 Device Certificate Example on page 9.

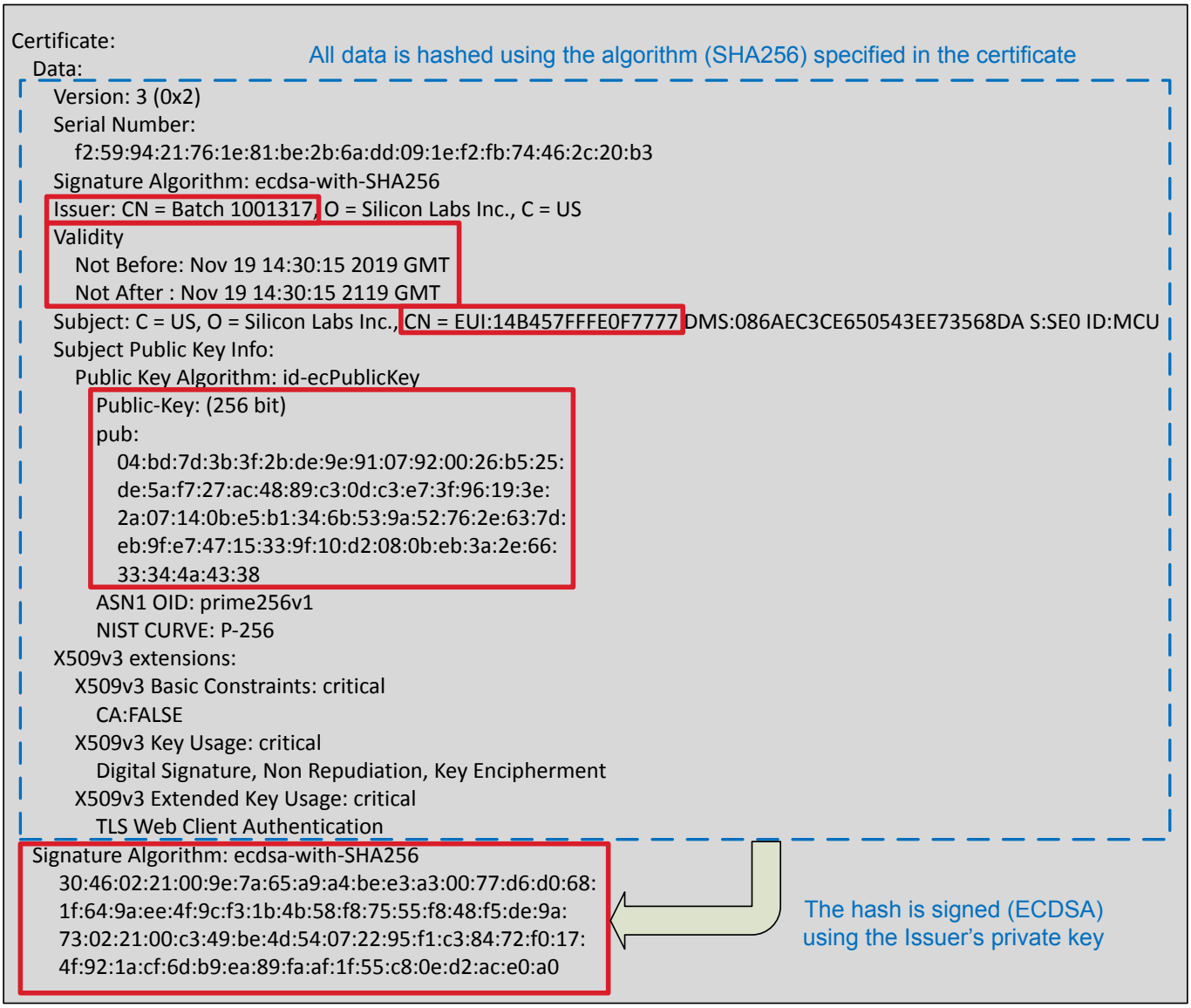


Figure 4.2. Device Certificate Example

- The device certificate is in X.509 DER format (~0.5 kB).
- The device certificate is stored in Secure Element (SE) one-time programmable memory (OTP). It cannot be modified once programmed.
- The batch number (`Issuer: CN = Batch` field) identifies the factory and batch in which the device was produced.
- The validity period is 100 years from device manufacture date.
- The device 64-bit hard-coded unique ID (EUI) is encoded in the `Subject: CN` field, which blinds this certificate to the device.
- The device-specific public key is embedded in the device certificate and the corresponding private key is securely stored in the Secure Key Storage on the chip.
- The Issuer's private key is used to sign the hash of the certificate data to create a device certificate signature.

4.3 Signing and Verification

Signing and verification for certificates on Secure Vault devices are described in Figure 4.3 Signing for Certificates on page 10 and Figure 4.4 Verification for Certificates on page 10.

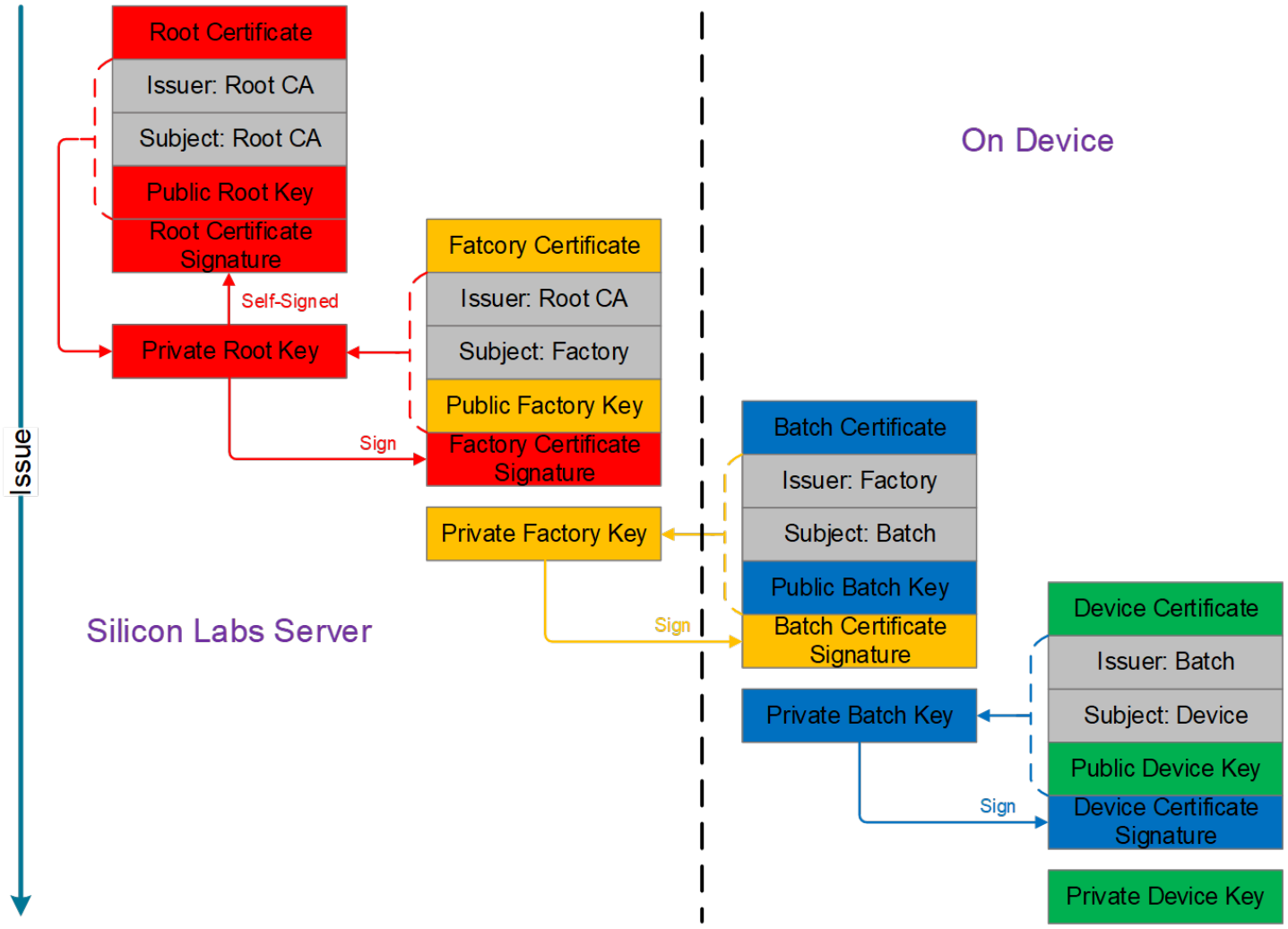


Figure 4.3. Signing for Certificates

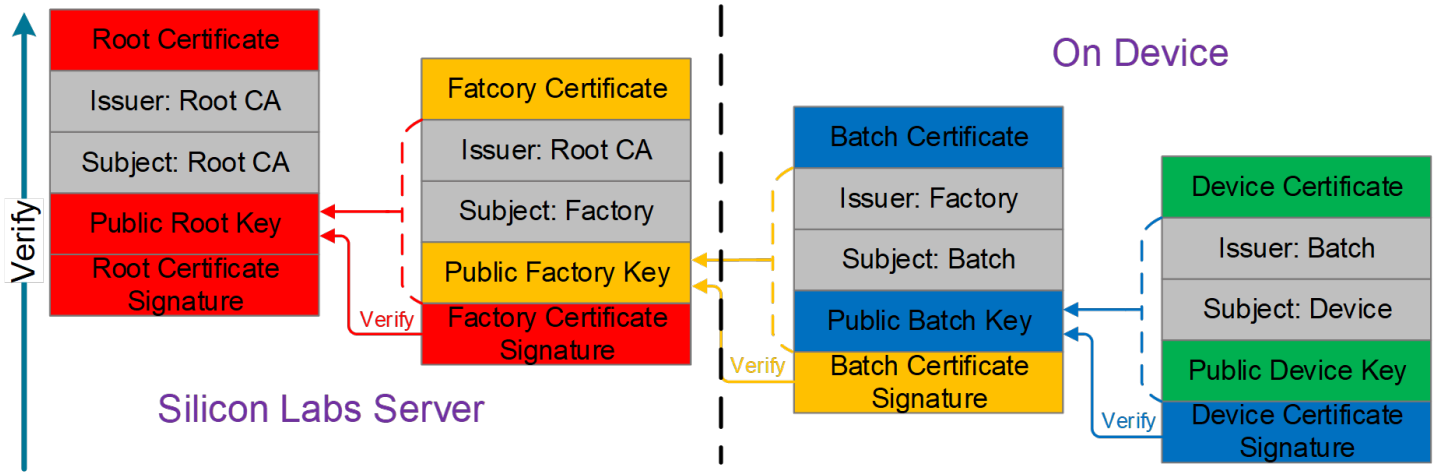


Figure 4.4. Verification for Certificates

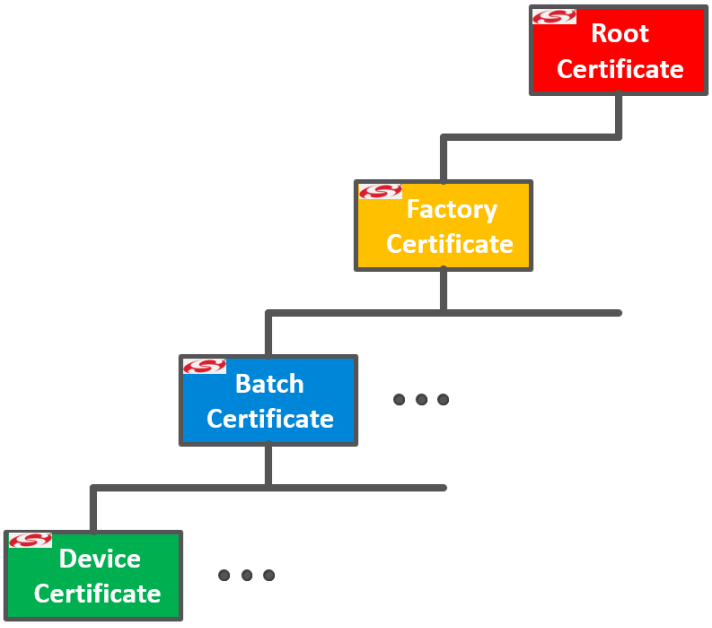
5. Device Certificate Options

The Secure Vault devices are each programmed with a device certificate during IC production. The device certificate is signed with a Public Device Key, using a Private Batch Key that can be validated against a Silicon Labs certificate chain (Figure 4.4 Verification for Certificates on page 10 and section 7.2 Certificate Chain Verification). The device private key never leaves the Secure Key Storage on the chip. Customers can create their own device certificates during their production.

Three device certificate options (standard, modified, and external) are provided to meet different requirements. The modified and external options are subject to an additional cost. For these options, contact your local sales representative (http://www.silabs.com/buysample/pages/contact-sales.aspx?view=map).

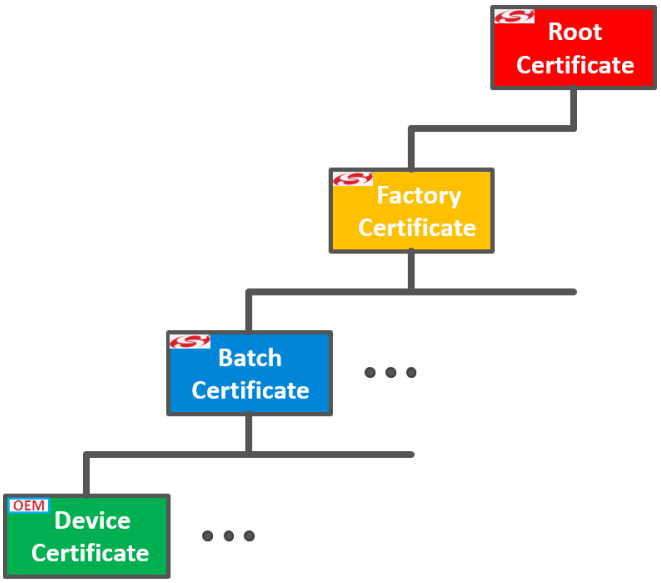
Standard Device Certificate:

- Comes standard with Secure Vault devices.
- Cryptographically proves the device is an authentic Silicon Labs device.
- Does not protect against overproduction or counterfeit products that are built with authentic Silicon Labs devices.
- Signed to a Silicon Labs Certificate Authority (CA).
- The device can prove that it possesses the private key associated with the public key in its certificate by signing the response to a given challenge (Figure 6.1 Remote Authentication Process on page 13 and section 7.3 Remote Authentication).



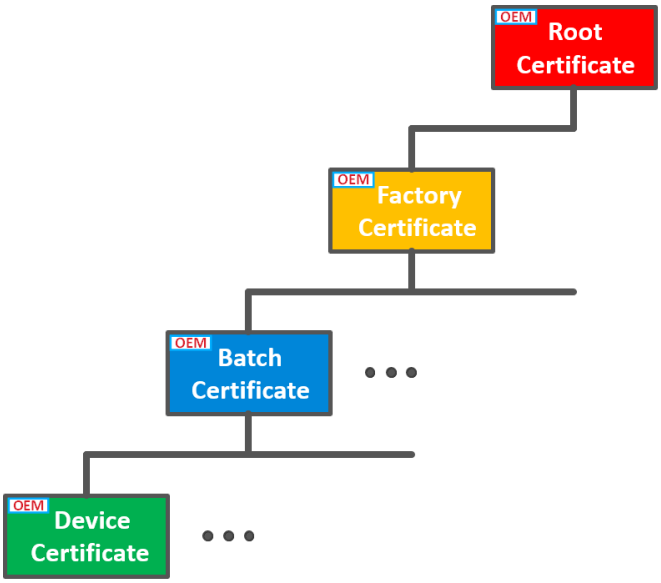
Modified Device Certificate:

- Available as a customization service on Secure Vault devices (OEM custom part number).
- Cryptographically proves the device is an authentic Silicon Labs device that was produced for a specific OEM.
- Protects against overproduction by Contract Manufacturer (CM).
- Device Certificate X.509 fields can be specified, with restrictions.
- Signed to a Silicon Labs Certificate Authority (CA).



External Device Certificate:

- Available as a customization service on Secure Vault devices (OEM custom part number).
- Cryptographically proves the device is an authentic Silicon Labs device that was produced for a specific OEM.
- Protects against overproduction by Contract Manufacturer (CM).
- Factory Certificate is custom for each OEM.
- Device Certificate and Factory Certificate X.509 fields can be specified, with restrictions.
- Signed to a OEM Certificate Authority (CA).
- Root Certificate Authority is OEM-specified and is optional.
- Electronic delivery of all batch and device certificates signed under this OEM factory certificate is supported.



6. Remote Authentication Process

Remote authentication is used to manage attestation by requesting that the device sign a challenge based on its secure identity.

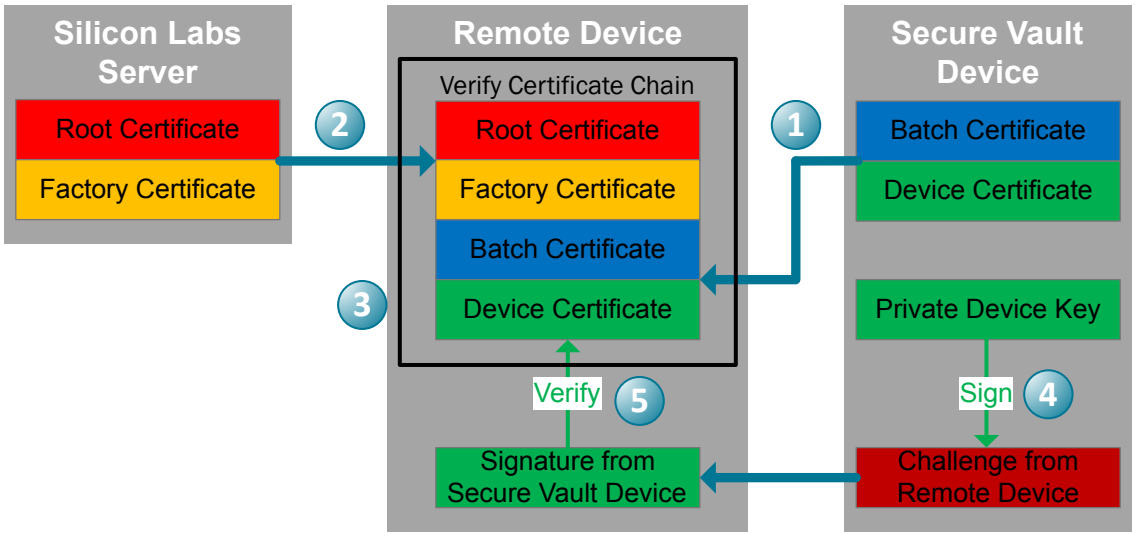


Figure 6.1. Remote Authentication Process

1. The remote device requests the device certificate and batch certificate from the Secure Vault device.
2. The remote device looks up the factory certificate and root certificate from the Silicon Labs Server.
3. The remote device validates each certificate in the chain using the public key of each Issuer (Figure 4.4 Verification for Certificates on page 10).
4. The remote device then sends an attestation challenge (random number) to the Secure Vault device. The Secure Vault device uses the Private Device Key in the Secure Key Storage on the chip to sign the challenge and sends the signature to the remote device.
5. The remote device validates the signature using the Public Device Key in the device certificate.

7. Examples

7.1 Overview

The secure device authentication examples are described in Table 7.1 Secure Device Authentication Examples on page 14.

Table 7.1. Secure Device Authentication Examples

Example	Device	Radio Board	SE Firmware	Tool
Certificate chain verification	EFR32MG21B010F1024IM32	BRD4181C	Version 1.2.4	Simplicity Commander and OpenSSL
	EFR32MG21B010F1024IM32	BRD4181C	Version 1.2.4	Simplicity Commander
Remote authentication	EFR32MG21B010F1024IM32	BRD4181C	Version 1.2.4	Secure Element Manager and mbed TLS

Note: Assume the root certificate (`root.pem`) and factory certificate (`factory.pem`) have already been retrieved from the Silicon Labs Server.

7.1.1 Using Simplicity Commander

1. Simplicity Commander's Command Line Interface (CLI) is invoked by `commander.exe` in the Simplicity Commander folder. The location in Windows is `C:\SiliconLabs\SimplicityStudio\<version>\developer\adapter_packs\commander`.
2. Simplicity Commander Version 1.10.1 is used in this application note.

```
commander --version
```

```
Simplicity Commander 1v10p1b839

JLink DLL version: 6.70a
Qt 5.12.1 Copyright (C) 2017 The Qt Company Ltd.
EMDLL Version: 0v17p14b549
mbed TLS version: 2.6.1

Emulator found with SN=440068705 USBAddr=0

DONE
```

3. If more than one WSTK is connected via USB, the target Wireless Starter Kit (WSTK) must be specified using the `--serialno <J-Link serial number>` option.
4. If the WSTK is in debug mode OUT, the target device must be specified using the `--device <device name>` option

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

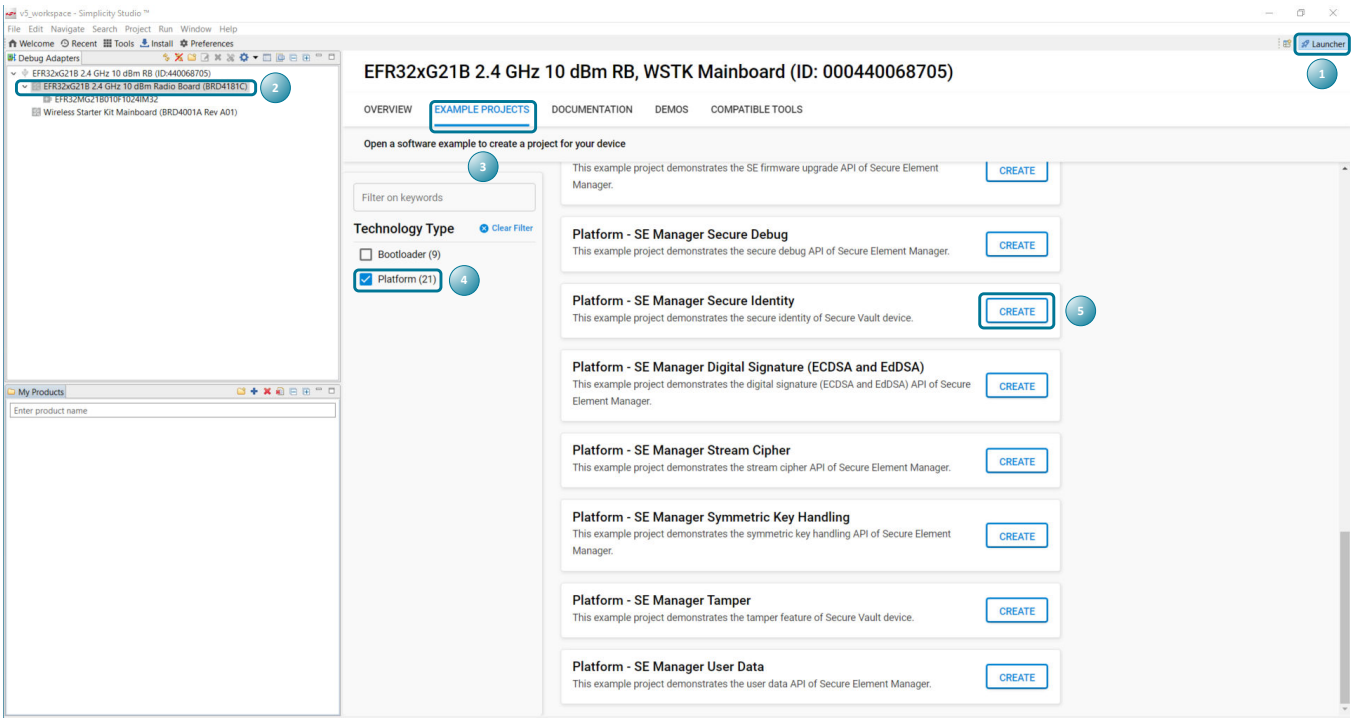
7.1.2 Using an External Tool

OpenSSL is used in the certificate chain verification example to validate the certificate chain. The Windows version of OpenSSL can be downloaded from here — https://slproweb.com/products/Win32OpenSSL.html.

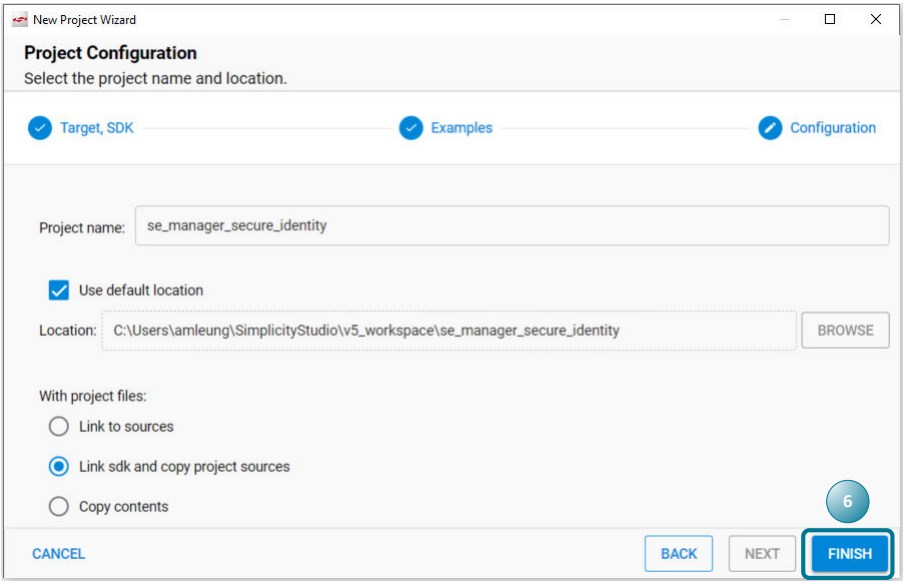
7.1.3 Using the Platform Example

This section describes how to use Simplicity Studio 5 to build the secure identity platform example and program it to the Wireless Starter Kit (WSTK).

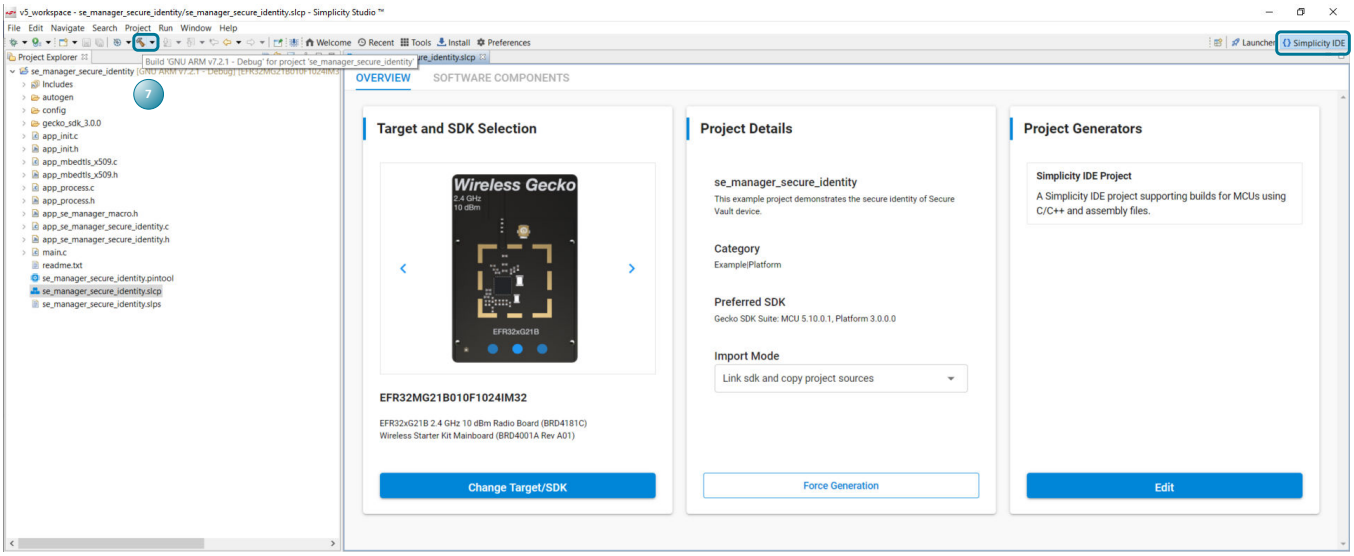
1. The connected WSTK is displayed in the Debug Adapters view in the [**Launcher**] perspective.Click the target radio board (**BRD4181C** in this example) This will automatically configure the task bars for use with your device.
2. From the [**Launcher**] perspective, click [**EXAMPLE PROJECTS**].
3. On the [**EXAMPLE PROJECTS**] tab, check the **Platform (n)** under **Technology Type**.
4. Find the **Platform - SE Manager Secure Identity** example and click [**CREATE**].



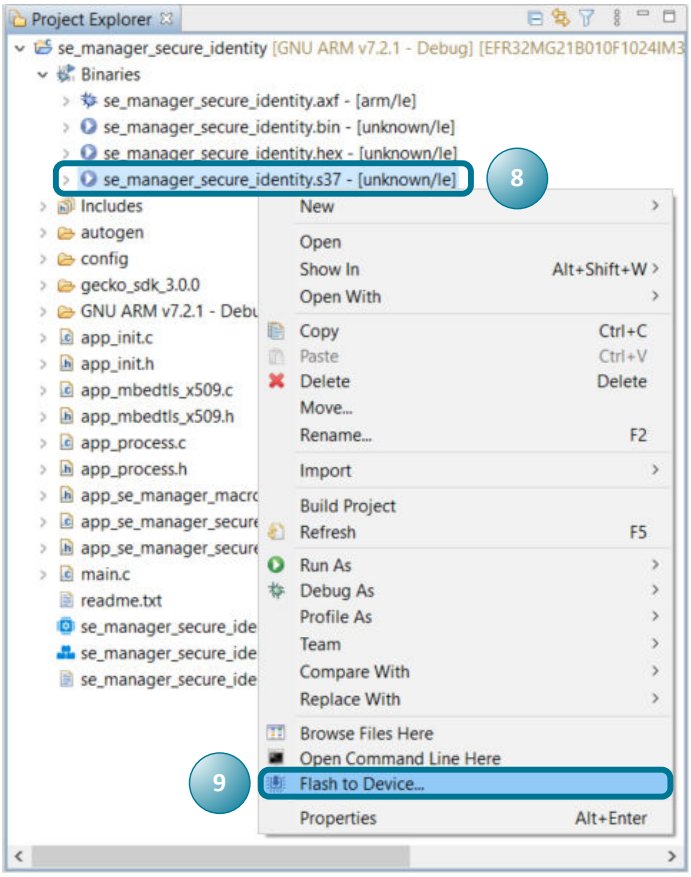
5. In the [**Project Configuration**] dialog, optionally name your project and select a different project location. Click [**FINISH**].

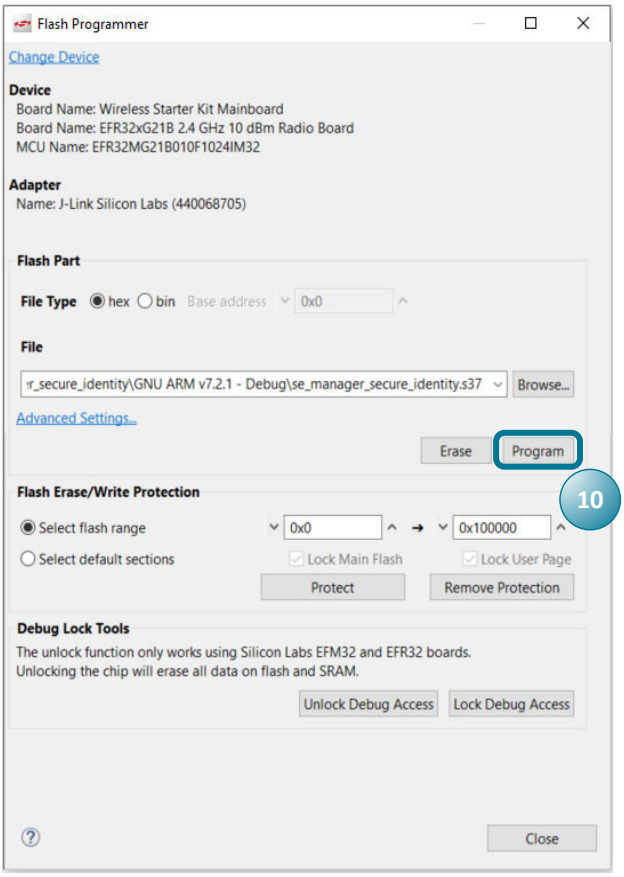


6. In the [**Simplicity IDE**] perspective, click the [**Build**] icon ( ).



7. In the Project Explorer view, right-click the `se_manager_secure_identity.s37` file under **Binaries** in Project Explorer.
8. Click [**Flash to Device...**] in the context menu to open the **Flash Programmer**.
9. Click [**Program**] to flash the `se_manager_secure_identity.s37` file to the radio board.





7.2 Certificate Chain Verification

Certificate chain verification is the process of making sure a given certificate chain is well-formed, valid, properly signed, and trustworthy. The certificate signature is verified using the public key in the issuer certificate (Figure 4.4 Verification for Certificates on page 10).

7.2.1 Simplicity Commander and OpenSSL

1. Run the `security readcert` command to save the batch certificate in PEM format.

```
commander security readcert batch -o batch.pem --serialno 440068705
```

```
DONE
```

2. Run the `security readcert` command to save the device certificate in PEM format.

```
commander security readcert mcu -o device.pem --serialno 440068705
```

```
DONE
```

3. Run the DOS `copy` command to merge the root certificate (`root.pem`) and factory certificate (`factory.pem`) into a certificate chain file (`chain.pem`).

```
copy root.pem+factory.pem chain.pem
```

```
root.pem
factory.pem
        1 file(s) copied.
```

4. The certificate information (e.g. `device.pem`) can be displayed by OpenSSL.

```
openssl x509 -in device.pem -text -noout
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            f2:59:94:21:76:1e:81:be:2b:6a:dd:09:1e:f2:fb:74:46:2c:20:b3
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: CN = Batch 1001317, O = Silicon Labs Inc., C = US
        Validity
            Not Before: Nov 19 14:30:15 2019 GMT
            Not After : Nov 19 14:30:15 2119 GMT
        Subject: C = US, O = Silicon Labs Inc., CN = EUI:14B457FFFE0F7777 DMS:086AEC3CE650543EE73568DA S:SE0 ID:MCU
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:bd:7d:3b:3f:2b:de:9e:91:07:92:00:26:b5:25:
                    de:5a:f7:27:ac:48:89:c3:0d:c3:e7:3f:96:19:3e:
                    2a:07:14:0b:e5:b1:34:6b:53:9a:52:76:2e:63:7d:
                    eb:9f:e7:47:15:33:9f:10:d2:08:0b:eb:3a:2e:66:
                    33:34:4a:43:38
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage: critical
                TLS Web Client Authentication
    Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:9e:7a:65:a9:a4:be:e3:a3:00:77:d6:d0:68:
         1f:64:9a:ee:4f:9c:f3:1b:4b:58:f8:75:55:f8:48:f5:de:9a:
         73:02:21:00:c3:49:be:4d:54:07:22:95:f1:c3:84:72:f0:17:
         4f:92:1a:cf:6d:b9:ea:89:fa:af:1f:55:c8:0e:d2:ac:e0:a0
```

5. Use OpenSSL to verify the certificate chain from step 1 to 3.

```
openssl verify -show_chain -CAfile chain.pem -untrusted batch.pem device.pem
```

```
device.pem: OK
Chain:
depth=0: C = US, O = Silicon Labs Inc., CN = EUI:14B457FFFE0F7777 DMS:086AEC3CE650543EE73568DA S:SE0 ID:MCU (untrusted)
depth=1: CN = Batch 1001317, O = Silicon Labs Inc., C = US (untrusted)
depth=2: CN = Factory, O = Silicon Labs Inc., C = US
depth=3: CN = Device Root CA, O = Silicon Labs Inc., C = US
```

7.2.2 Simplicity Commander

Run the `security readcert` command to display the key information about the on-chip certificates (e.g. `mcu`).

```
commander security readcert mcu --serialno 440068705
```

```
Version            : 3
Subject            : C=US  O=Silicon Labs Inc.  CN=EUI:14B457FFFE0F7777 DMS:086AEC3CE650543EE73568DA S:SE0 ID:MCU
Issuer             : CN=Batch 1001317  O=Silicon Labs Inc.  C=US
Valid From         : November 19 2019
Valid To           : November 19 2119
Signature algorithm: SHA256
Public Key Type    : ECDSA
Public key         : bd7d3b3f2bde9e9107920026b525de5af727ac4889c30dc3e73f96193e2a0714
                     0be5b1346b539a52762e637deb9fe74715339f10d2080beb3a2e6633344a4338
DONE
```

Run the `security attestation` command to verify the on-chip batch and device certificates with root and factory certificates.

```
commander security attestation --serialno 440068705
```

```
Certificate chain successfully validated up to Silicon Labs device root certificate.
```

7.3 Remote Authentication

See section 7.1.3 Using the Platform Example for more information on programming the security identity platform example to the WSTK. The `se_manager_identity` example uses APIs in Secure Element Manager and mbed TLS to emulate the processes in Figure 6.1 Remote Authentication Process on page 13.

To eliminate the communications between different parties in this example, the operations in the remote device are simulated by the Secure Vault device, and the factory certificate and root certificate are hard-coded in `app_mbedtls_x509.c`.

This example uses the CRYPTO engine in the Secure Element to accelerate the X.509 API functions of mbed TLS. The Private Device Key in the Secure Key Storage on the chip is used to sign the challenge from the remote device so this example can only run on a Standard Device Certificate.

The example redirects standard I/O to the virtual serial port (VCOM) of the WSTK. Open a terminal program (for example, Tera Term) and access the WSTK VCOM port (default setting is 115200 bps 8-N-1).

Step 1 in the Remote Authentication Process:

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

  . Secure Vault device:
  + Read size of on-chip certificates... SL_STATUS_OK (cycles: 4013 time: 105 us)
  + Read on-chip device certificate... SL_STATUS_OK (cycles: 4443 time: 116 us)
  + Parse the device certificate (DER format)... SL_STATUS_OK (cycles: 51168 time: 1346 us)
  + Get the public device key in device certificate... OK
  + Read on-chip batch certificate... SL_STATUS_OK (cycles: 4430 time: 116 us)
  + Parse the batch certificate (DER format)... SL_STATUS_OK (cycles: 62936 time: 1656 us)
```

Steps 2 and 3 in the Remote Authentication Process (certificate chain printout is disabled):

```
. Remote device:
+ Parse the factory certificate (PEM format)... SL_STATUS_OK (cycles: 177168 time: 4662 us)
+ Parse the root certificate (PEM format)... SL_STATUS_OK (cycles: 160914 time: 4234 us)
+ Verify the certificate chain with root certificate... SL_STATUS_OK (cycles: 964681 time: 25386 us)
```

Steps 2 and 3 in the Remote Authentication Process (certificate chain printout is enabled):

```
. Remote device:
+ Parse the factory certificate (PEM format)... SL_STATUS_OK (cycles: 177170 time: 4662 us)
+ Parse the root certificate (PEM format)... SL_STATUS_OK (cycles: 160935 time: 4235 us)
+ Verify requested for (Depth 3) ... OK
    cert. version     : 3
    serial number     : 12:E6:A2:A5:9C:AA:27:F9
    issuer name       : CN=Device Root CA, O=Silicon Labs Inc., C=US
    subject name      : CN=Device Root CA, O=Silicon Labs Inc., C=US
    issued  on        : 2018-10-10 17:32:00
    expires on        : 2118-09-16 17:32:00
    signed using      : ECDSA with SHA256
    EC key size       : 256 bits
    basic constraints : CA=true, max_pathlen=2
    key usage         : Digital Signature, Key Cert Sign, CRL Sign
+ Verify requested for (Depth 2) ... OK
    cert. version     : 3
    serial number     : 24:DC:7B:40:0C:32:9C:0A
    issuer name       : CN=Device Root CA, O=Silicon Labs Inc., C=US
    subject name      : CN=Factory, O=Silicon Labs Inc., C=US
    issued  on        : 2018-10-10 17:33:00
    expires on        : 2118-09-16 17:32:00
    signed using      : ECDSA with SHA256
    EC key size       : 256 bits
    basic constraints : CA=true, max_pathlen=1
    key usage         : Digital Signature, Key Cert Sign, CRL Sign
+ Verify requested for (Depth 1) ... OK
    cert. version     : 3
    serial number     : 23:09:DA:39:B4:78:05:AA
    issuer name       : CN=Factory, O=Silicon Labs Inc., C=US
    subject name      : CN=Batch 1001317, O=Silicon Labs Inc., C=US
    issued  on        : 2019-10-17 21:20:20
    expires on        : 2118-09-16 17:32:00
    signed using      : ECDSA with SHA256
    EC key size       : 256 bits
    basic constraints : CA=true, max_pathlen=0
    key usage         : Digital Signature, Key Cert Sign
+ Verify requested for (Depth 0) ... OK
    cert. version     : 3
    serial number     : F2:59:94:21:76:1E:81:BE:2B:6A:DD:09:1E:F2:FB:74:46:2C:20:B3
    issuer name       : CN=Batch 1001317, O=Silicon Labs Inc., C=US
    subject name      : C=US, O=Silicon Labs Inc., CN=EUI:14B457FFFE0F7777 DMS:086AEC3CE650543EE73568DA S:SE0 ID:MCU
    issued  on        : 2019-11-19 14:30:15
    expires on        : 2119-11-19 14:30:15
    signed using      : ECDSA with SHA256
    EC key size       : 256 bits
    basic constraints : CA=false
    key usage         : Digital Signature, Non Repudiation, Key Encipherment
    ext key usage     : TLS Web Client Authentication
+ Verify the certificate chain with root certificate... SL_STATUS_OK (cycles: 8920041 time: 234 ms)
```

Note: The longer processing time (234 ms) is due to the certificate chain printout.

Steps 4 and 5 in the Remote Authentication Process:

```
. Remote authentication:
+ Create 16 bytes challenge (random number) in remote device for signing... SL_STATUS_OK (cycles: 3724 time: 98 us)
+ Sign challenge with private device key in Secure Vault device... SL_STATUS_OK (cycles: 222346 time: 5851 us)
+ Verify signature with public device key in remote device... SL_STATUS_OK (cycles: 236712 time: 6229 us)

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

8. Revision History

Revision 0.1

September 2020

- Initial Revision.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com