Security in IoT

THREATS EVOLVE. SO SHOULD YOUR DEVICE SECURITY.

BRENT WILSON | PRODUCT SECURITY TEAM

APRIL-2020

silabs.com/security

Talk Talks LIVE Schedule

Topic	**Date**
Bluetooth AoX Solutions	Thursday, April 2
15.4 Mesh Networking Technologies	Tuesday, April 7
Bluetooth Mesh Solutions & Tools	Thursday, April 9
Device & Network Security for the IoT	Tuesday, April 14
Evolution of Bluetooth 5, 5.1, & 5.2	Thursday, April 16
Connected Home Over IP (CHIP) for Beginners	Tuesday, April 21

https://www.silabs.com/about-us/events/tech-talks

The Leader in IoT Wireless Connectivity

Bluetooth® Proprietary THREAD WiFi® zigbee Z-WAVE®

>35,000
Customers

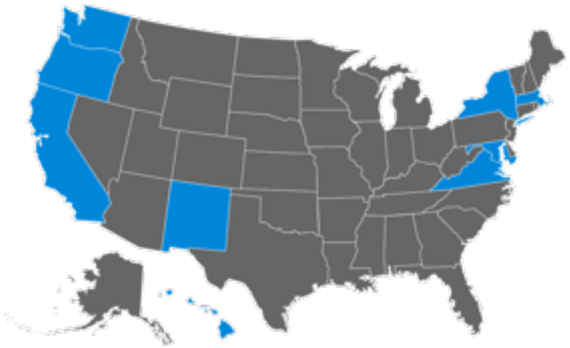
>3B
Products Shipped

#1
IoT Wireless Solutions

20-30%
Wireless Y-Y CAGR*

*Across 15.4, BLE, Wi-Fi, Proprietary

IoT Security Legislation is Happening



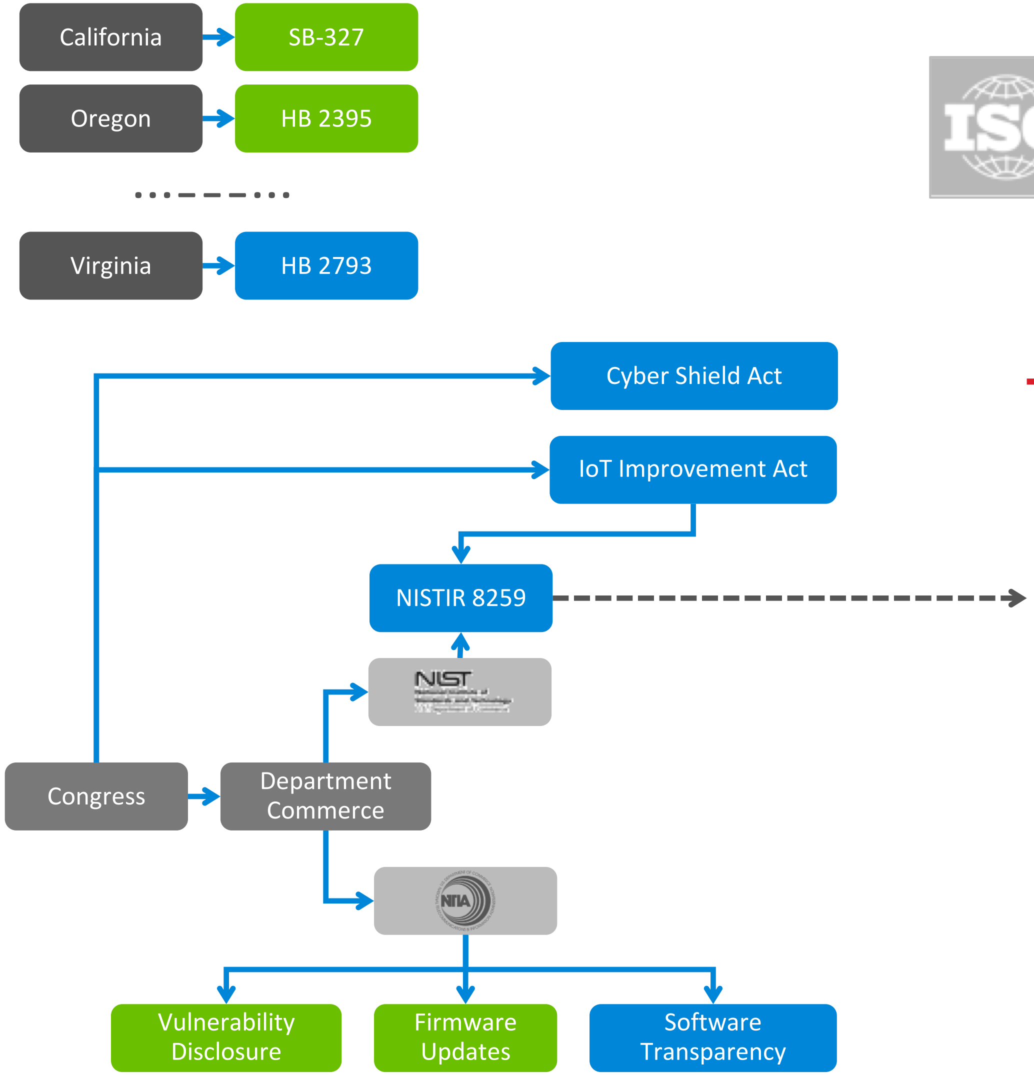
Multiple states have already introduced bills that resemble California's CCPA example

Virginia	(HB 2793)
Oregon	(HB 2395)
Hawaii	(SB 418)
Maryland	(SB 0613)
Massachusetts	(SD 341)
New Mexico	(SB 176)
New York	(S00224)
Rhode Island	(SB 234)
Washington	(SB 5376)

- California Consumer Privacy Act (§ SB-327)
  - Introduced Feb 13, 2017
  - Approved Sept 28, 2018
  - **Effective Jan 1, 2020 (<3yrs)**
- Requires **'reasonable security features'**
  - appropriate to the nature and function of the device
  - appropriate to the information it may collect, contain, or transmit
  - designed to protect the device and any information contained therein from unauthorized access, destruction, use, modification, or disclosure
  - Pre-programmed passwords are unique in each device manufactured

Already accounts for ~30% US population

Governmental Regulatory Landscape – United States



ISO International Organization for Standardization

Draft NISTIR 8259

Core Cybersecurity Feature Baseline for Securable IoT Devices:

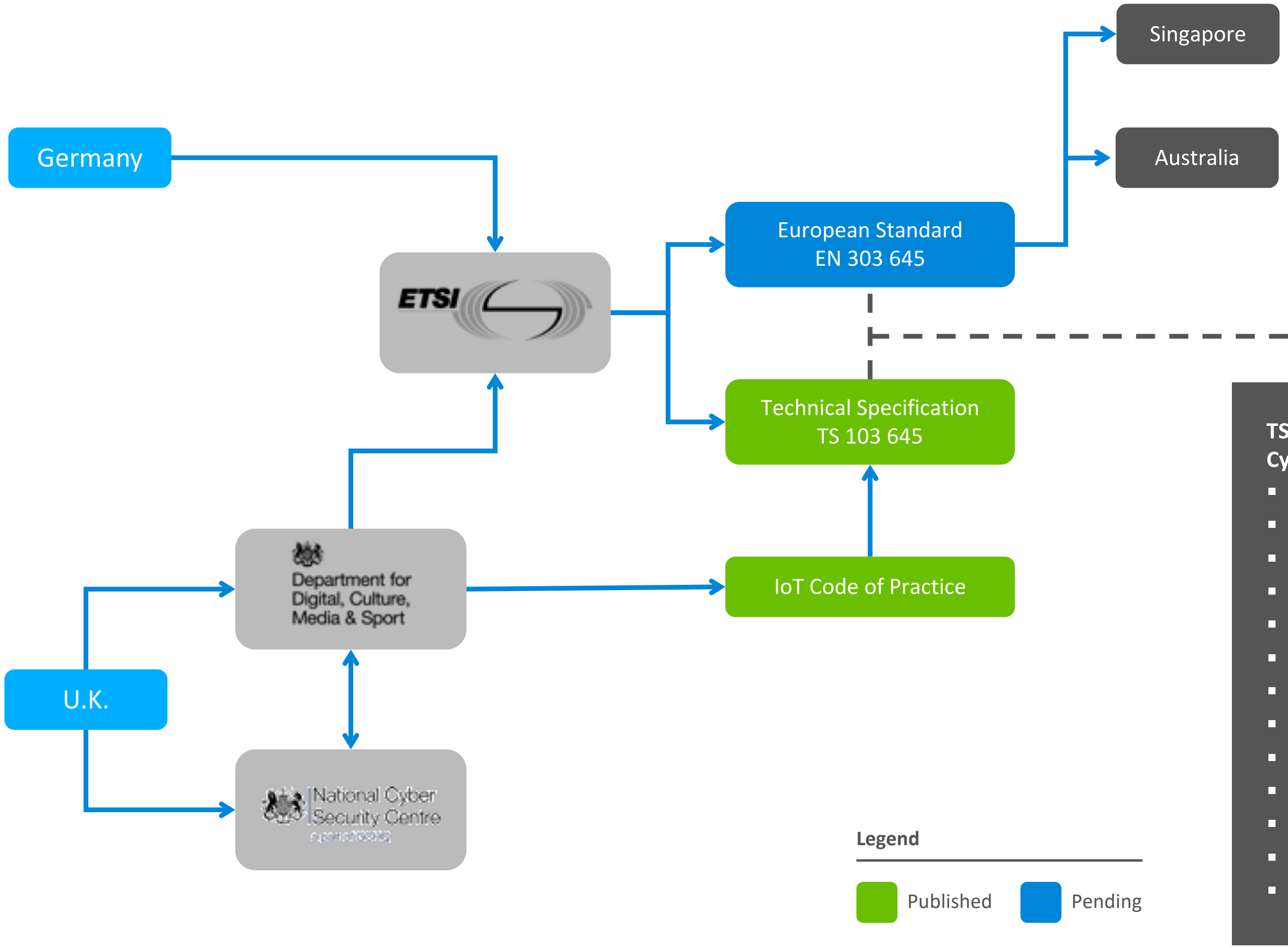
A Starting Point for IoT Device Manufacturers

Concern	Federal Requirement
Device Identification	The IoT device can be uniquely identified logically and physically.
Device Configuration	The IoT device's software and firmware configuration can be changed, and such changes can be performed by authorized entities only.
Data Protection	The IoT device can protect the data it stores and transmits from unauthorized access and modification.
Logical Access to Interfaces	The IoT device can limit logical access to its local and network interfaces to authorized entities only.
Software and Firmware Update	The IoT device's software and firmware can be updated by authorized entities only using a secure and configurable mechanism.
Cybersecurity Event Logging	The IoT device can log cybersecurity events and make the logs accessible to authorized entities only.

Legend

🟩 Published 🟦 Pending

Governmental Regulatory Landscape – Europe (& extended adoptees)


Germany
ETSI
U.K.
Department for Digital, Culture, Media & Sport
National Cyber Security Centre
European Standard EN 303 645
Technical Specification TS 103 645
IoT Code of Practice
Singapore
Australia
Legend
🟩 Published
🟦 Pending

TS103 645 / EN 303 645
Cyber Security for Consumer Internet of Things

- No universal default passwords
- Implement a means to manage reports of vulnerabilities
- Keep software updated
- Securely store credentials and security-sensitive data
- Communicate securely
- Minimize exposed attack surfaces
- Ensure software integrity
- Ensure that personal data is protected
- Make systems resilient to outages
- Examine system telemetry data
- Make it easy for consumers to delete personal data
- Make installation and maintenance of devices easy
- Validate input data

Mapping Security Requirements to Security Features

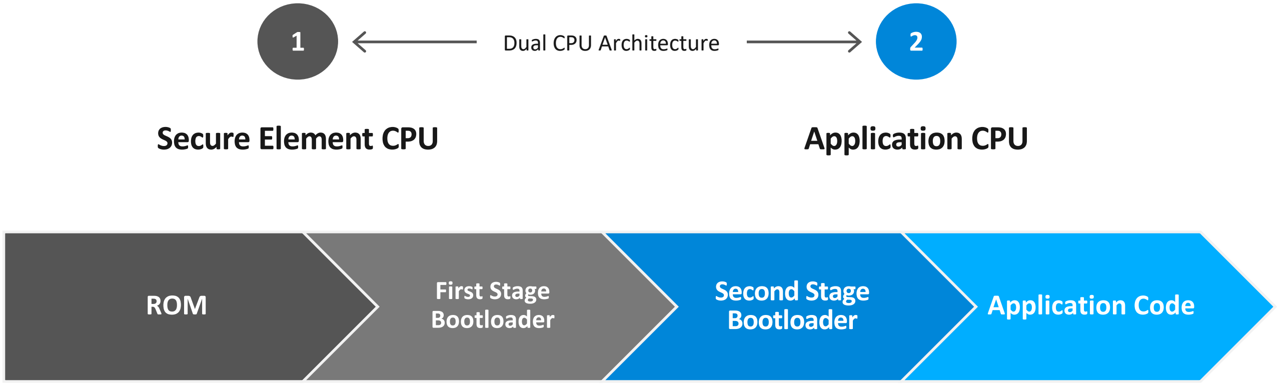
Security Requirement	Security Feature
The IoT device can be uniquely identified logically and physically	Secure Attestation
The IoT device’s software and firmware configuration can be changed, and such changes can be performed by authorized entities only	Secure Upgrade
The IoT device can protect the data it stores and transmits from unauthorized access and modification	Secure Key Storage
The IoT device can limit logical access to its local and network interfaces to authorized entities only	Secure Debug
The IoT device’s software and firmware can be updated by authorized entities only using a secure and configurable mechanism	Secure Upgrade
The IoT device can log cybersecurity events and make the logs accessible to authorized entities only	Anti-Tamper
Ensure software integrity	Secure Boot

Security Portfolio

Feature	Basic	+Root of Trust	+Secure Element	Secure Vault
TRNG with continuous health check	✓	✓	✓	✓
Crypto Engine	✓	✓	✓	✓
Secure Boot	✓	✓	✓	✓
Secure Boot with RTSL	-	✓	✓	✓
ARM® TrustZone®	-	✓	✓	✓
Debug Access Lock/Unlock	-	✓	✓	✓
DPA Countermeasures	-	-	✓	✓
Anti-Tamper	-	-	-	✓
Secure Attestation	-	-	-	✓
Secure Key Management	-	-	-	✓
Secure Key Storage	-	-	-	✓
Advanced Crypto	-	-	-	✓
	Series 1 – xG1x 90nm M4	**Series 2 – xG22 40nm M33**	**Series 2 – xG21A 40nm M33**	**Series 2 – xG21B 40 nm M33**

Secure Boot and Secure Updates

LOCAL & REMOTE ATTACK VECTOR



Immutable memory, check secure element bootloader code (SEB), can update SEB code

Check second stage bootloader code (SSB), can update SSB code

Check application code, can update application code

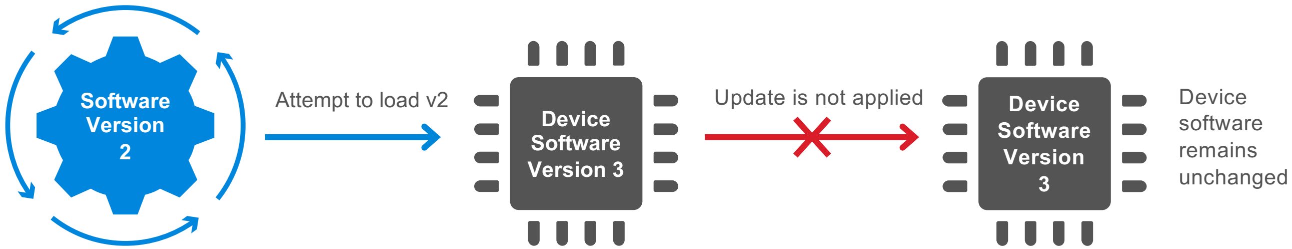
Execute trusted application code against immutable memory and through full chain of trust

- Vulnerabilities
  - Replacing code with 'look-alike code' makes a product appear normal. Hackers use it to copy/re-direct data to alternate servers.
- Secure Boot with RTSL
  (Root-of-Trust & Secure Loader)
  - Use and execute only trusted application code against immutable memory and through a full chain of trust
  - Authenticate firmware upgrades prior to applying the update

Anti-Rollback Protection

LOCAL & REMOTE ATTACK VECTOR

Failure



Software Version 2

Attempt to load v2

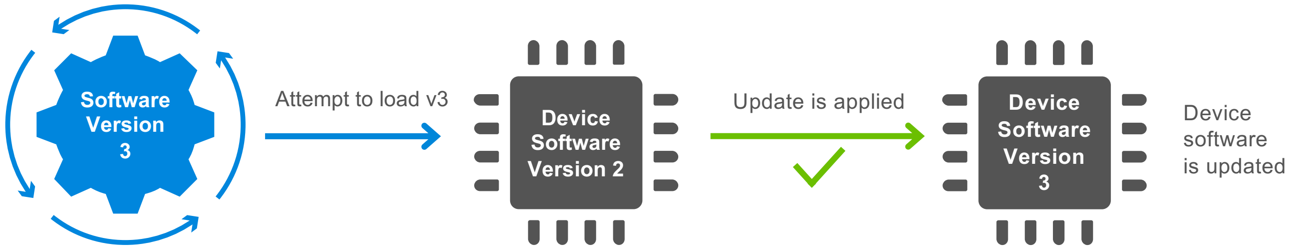
Device Software Version 3

Update is not applied

Device Software Version 3

Device software remains unchanged

Success



Software Version 3

Attempt to load v3

Device Software Version 2

Update is applied

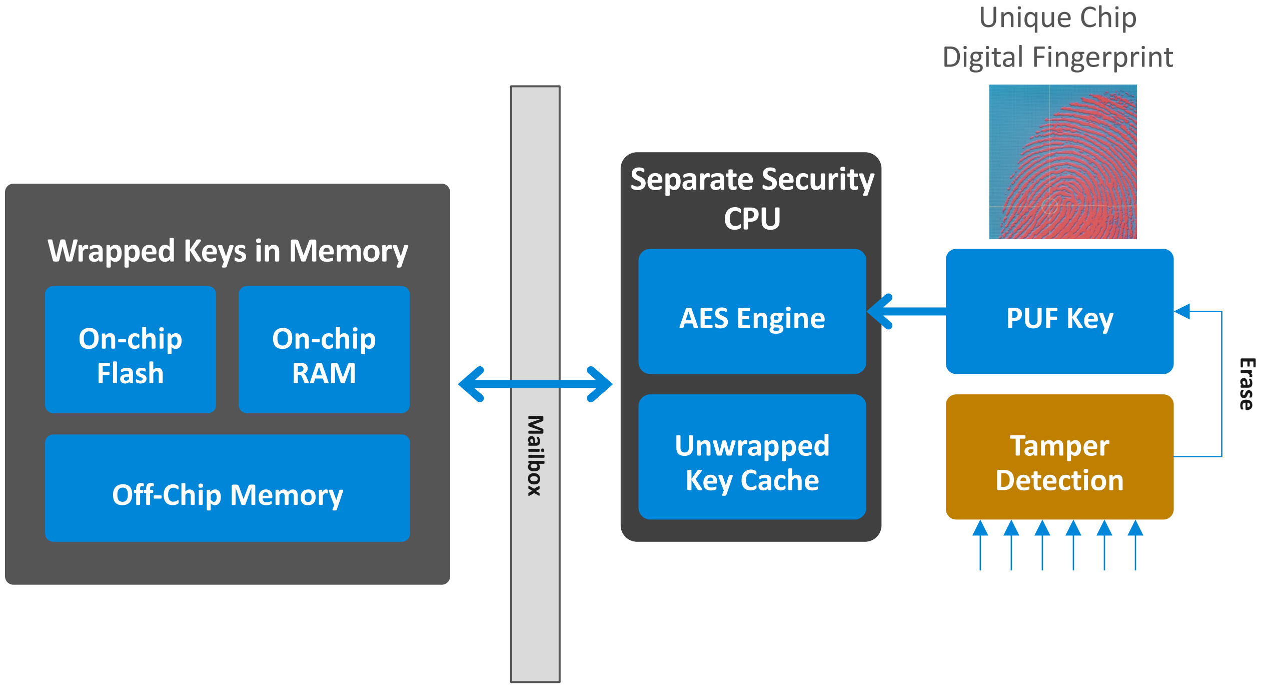
Device Software Version 3

Device software is updated

- Vulnerabilities
  - Adversaries may have knowledge of a security flaw present in older firmware
- Anti-Rollback Protection
  - Prevents older digitally signed firmware from being re-loaded into a device to re-expose patched flaws

Secure Key Storage

LOCAL & REMOTE ATTACK VECTOR



- Vulnerabilities
  - When an attacker learns how to extract keys or content from a device, they use the same attack vector to attack other devices
- Secure Key Storage
  - A Physically Unclonable Function creates a secret, random, & unique key, from individual device imperfections
  - The PUF-key encrypts all keys in the secure key storage. It is generated at startup and is not stored in flash

DPA Countermeasures

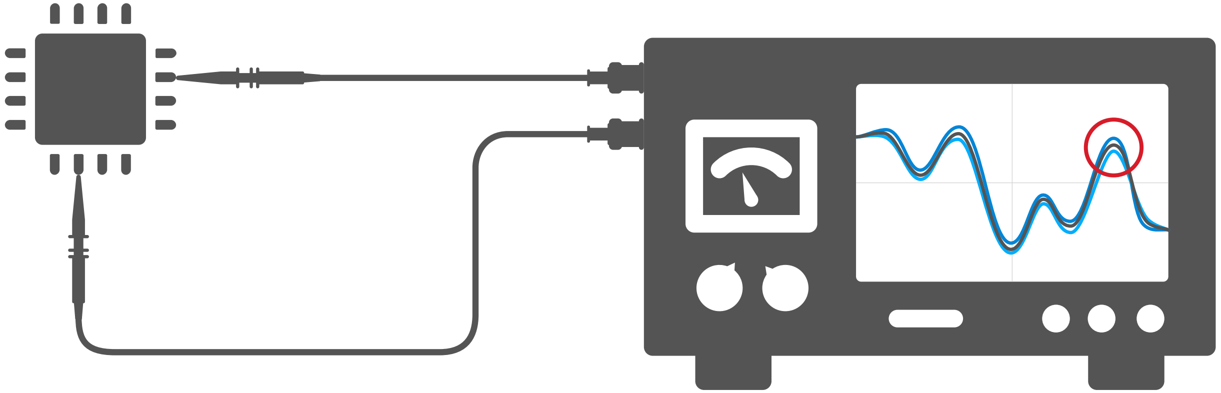
LOCAL ATTACK VECTOR

1

A Differential Power Analysis (DPA) attack requires hands-on access to the device.

2

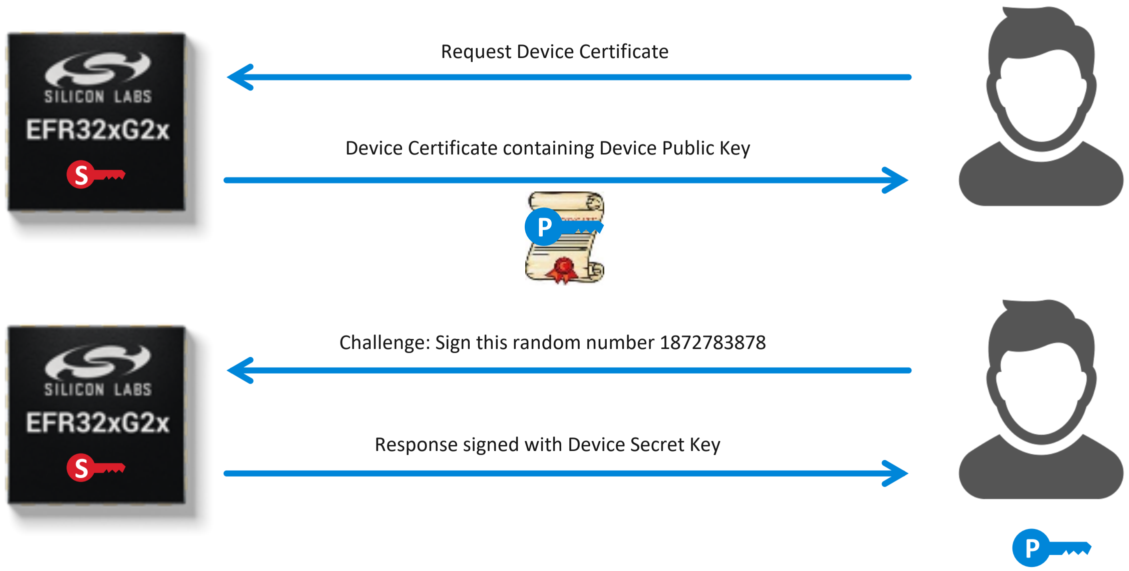
Monitoring electromagnetic radiation and fluctuations in power consumption during crypto operations may reveal security keys and other data.



- Vulnerabilities
  - Observing subtle signal differences during given internal operations can provide insight into cryptographic functions
- DPA Countermeasures
  - Countermeasures add masks and random timings to internal operations and distorts DPA snooping

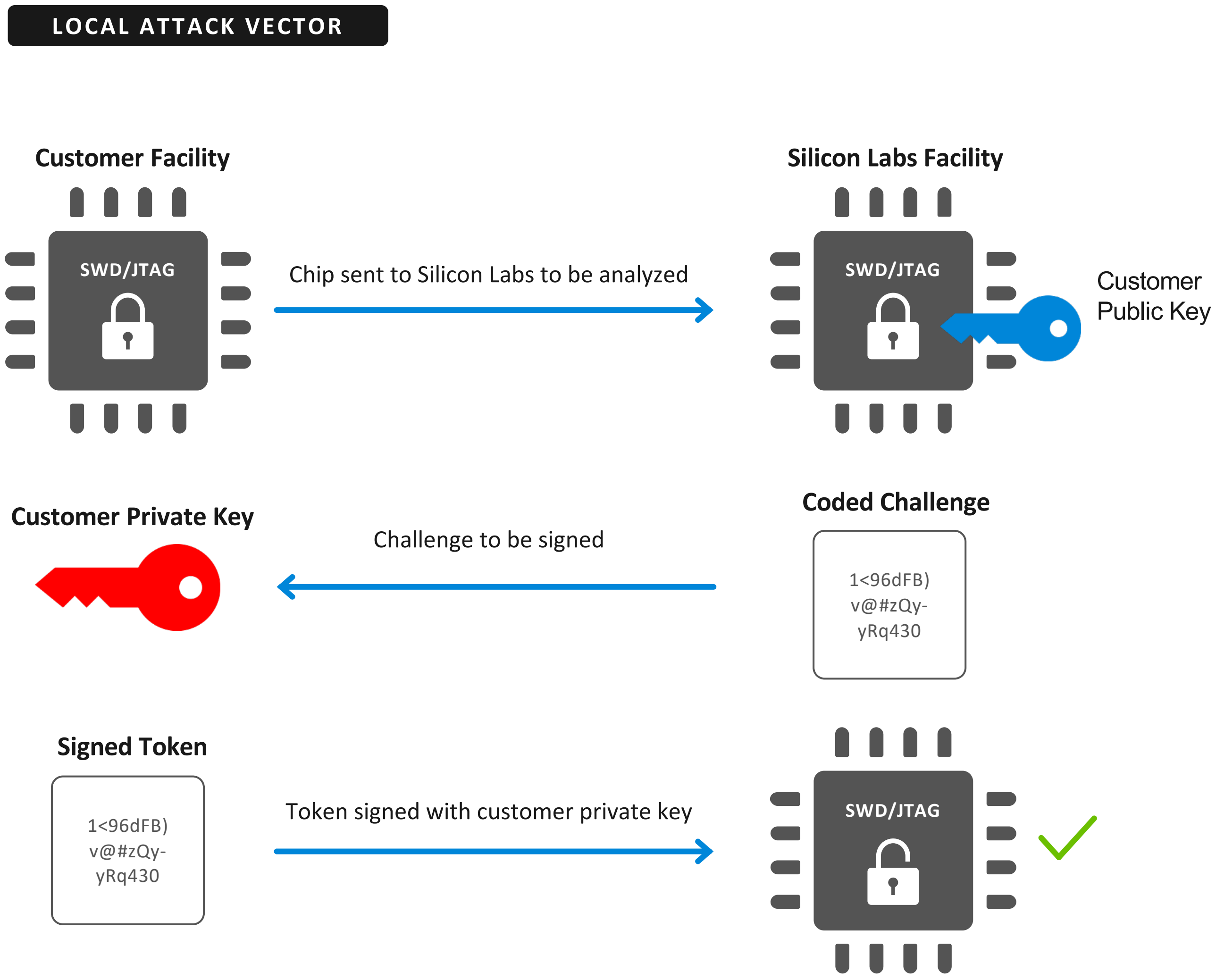
Secure Attestation

LOCAL ATTACK VECTOR


SILICON LABS
EFR32xG2x
S
Request Device Certificate
Device Certificate containing Device Public Key
P
SILICON LABS
EFR32xG2x
S
Challenge: Sign this random number 1872783878
Response signed with Device Secret Key
P

- Vulnerabilities
  - Many systems use a UID to identify devices, but the UID is public (can be copied)
  - Developers are concerned with the authenticity of their devices
  - Most successful companies suffer counterfeit products and “ghost shifts”
- Secure Attestation
  - Secure Vault devices generate a unique device ECC keypair on-chip and securely store the private key
  - The device secret key never leaves the chip
  - During production, the test program reads the device public key, places it in the certificate signs the device certificate with an HSM secret key, and stores it back into the chip in OTP memory
  - An external service can now request the certificate chain from the device and our CA web server, retrieve the unique device public key.
  - The external service can then perform a “Challenge Response” to the chip at any time during the life of the product to Authentic the chip is genuine Silicon Labs silicon

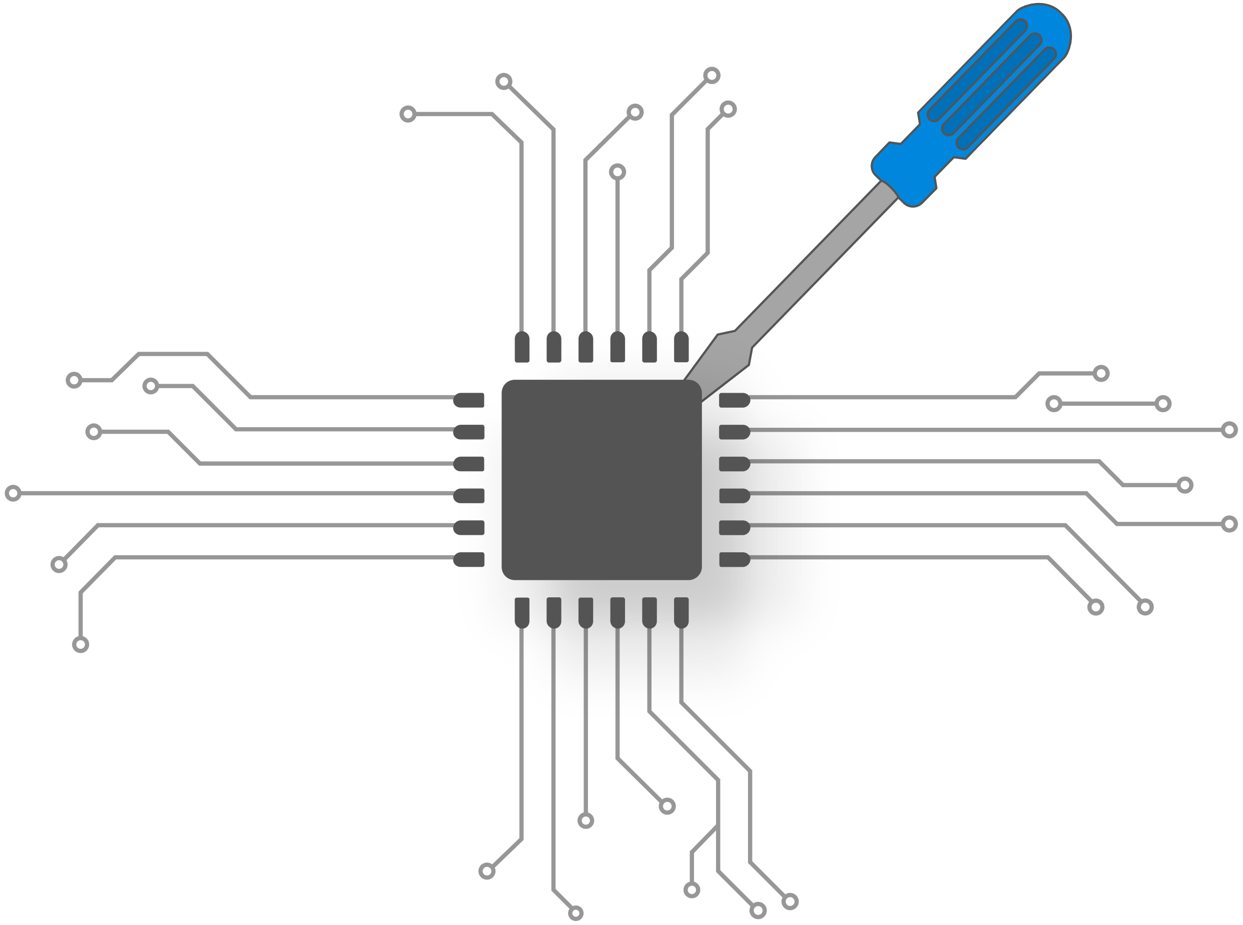
Secure Debug



- Vulnerabilities
  - Unlocked ports are a significant security vulnerability
  - 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

Anti-Tamper

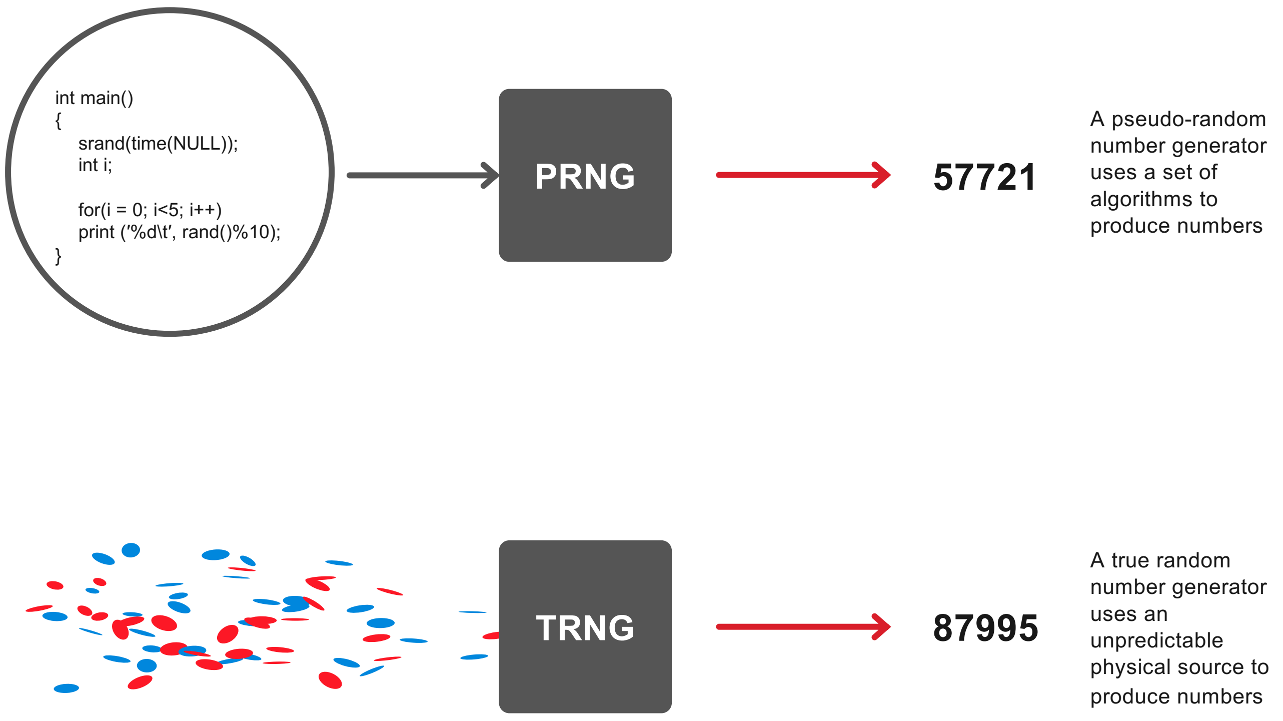
LOCAL ATTACK VECTOR



- Vulnerabilities
  - Tamper attacks come from single or multiple vectors.
  - Common attacks include voltage glitching, magnetic interference and forced temperature adjustment
- Tamper detection and rapid response
  - Anti-tamper requires both an attack detection and suitable rapid response which may include key deletion.

True Random Number Generator

LOCAL & REMOTE ATTACK VECTOR



```
int main()
{
    srand(time(NULL));
    int i;

    for(i = 0; i<5; i++)
    print ('%d\t', rand()%10);
}
```

A pseudo-random number generator uses a set of algorithms to produce numbers

A true random number generator uses an unpredictable physical source to produce numbers

- Vulnerabilities
  - If any bias in generating a number can be determined, hackers leverage that to reduce the time and effort they need to acquire secret keys
- True Random Numbers
  - True Random Number Generator that meets NIST SP 800-90 and AIS-31

Thank You!

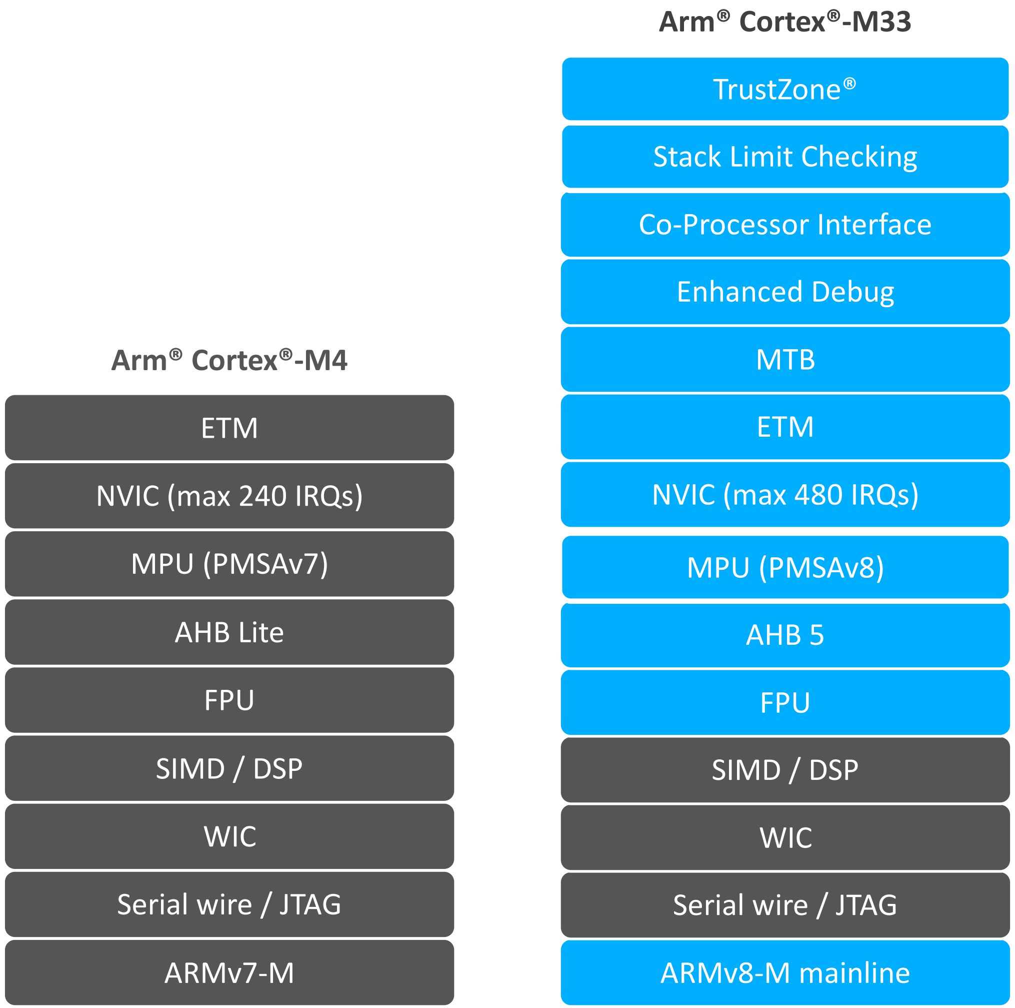
Q & A

Topic	Date
Bluetooth AoX Solutions	Thursday, April 2
15.4 Mesh Networking Technologies	Tuesday, April 7
Bluetooth Mesh Solutions & Tools	Thursday, April 9
Device & Network Security for the IoT	Tuesday, April 14
Evolution of Bluetooth 5, 5.1, & 5.2	Thursday, April 16
Connected Home Over IP (CHIP) for Beginners	Tuesday, April 21

https://www.silabs.com/about-us/events/tech-talks

Q & A

Cortex-M33 enhancements over Cortex-M4



Arm® Cortex®-M4

ETM

NVIC (max 240 IRQs)

MPU (PMSAv7)

AHB Lite

FPU

SIMD / DSP

WIC

Serial wire / JTAG

ARMv7-M

Arm® Cortex®-M33

TrustZone®

Stack Limit Checking

Co-Processor Interface

Enhanced Debug

MTB

ETM

NVIC (max 480 IRQs)

MPU (PMSAv8)

AHB 5

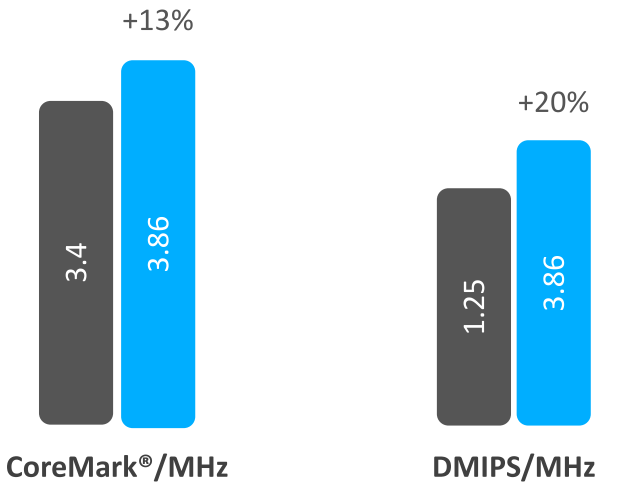
FPU

SIMD / DSP

WIC

Serial wire / JTAG

ARMv8-M mainline



🟦 New or Updated

TrustZone® for Arm v8-M

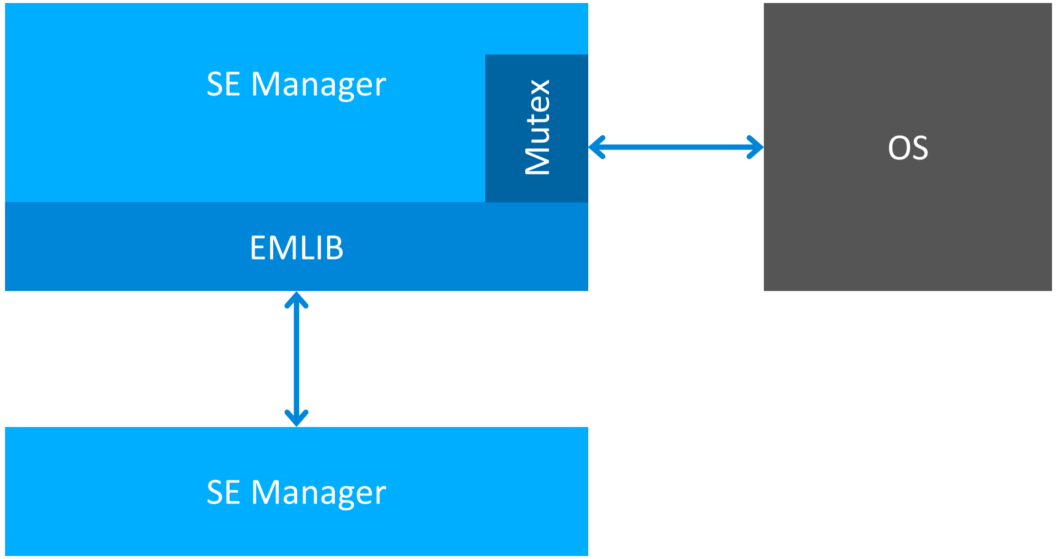
Why

- Added security in case of software bugs or malicious attacks
  - There is no such thing as perfect, bug-free software
  - Best practice prevent tasks from being able to interfere with other tasks
  - Firmware source may not be in our control
  - Most projects include 3rd party libraries and other untrusted code
- Give access to 3rd parties and “app”-stores without system wide access

Silicon Labs

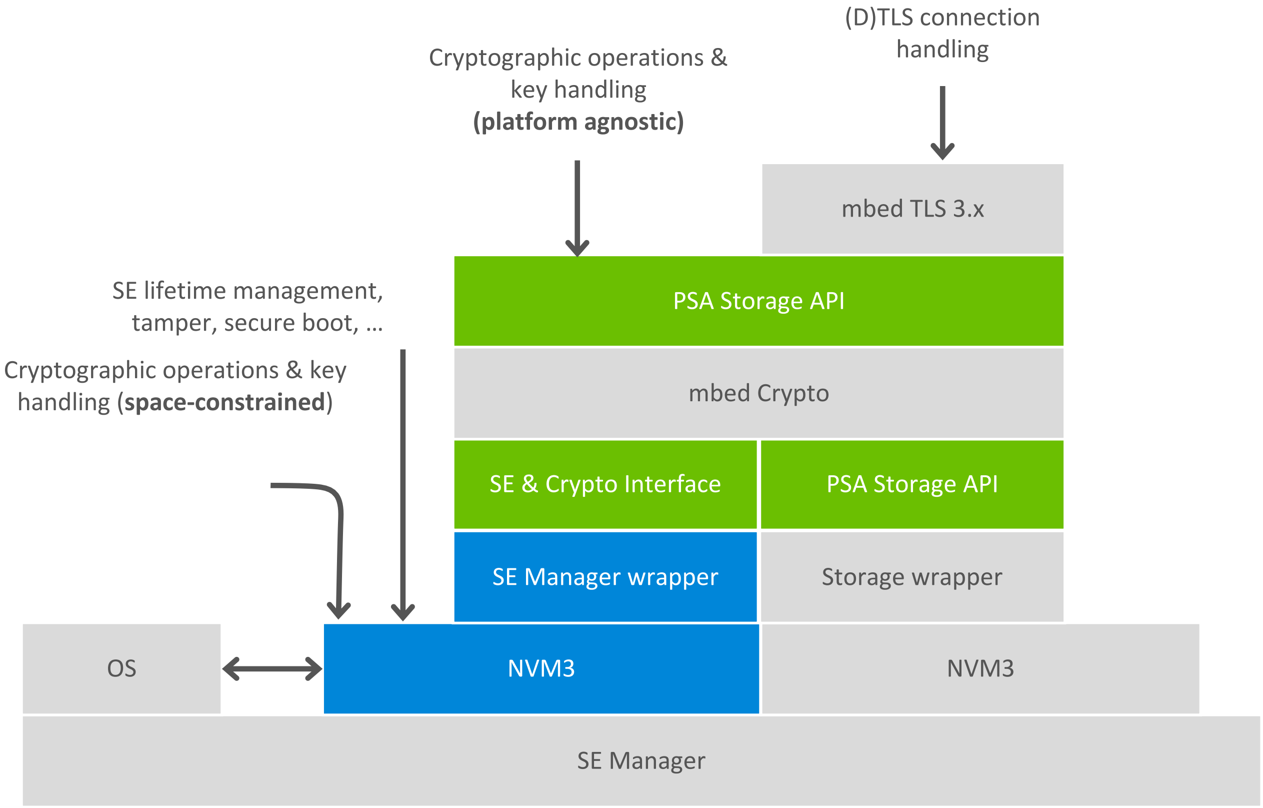
- Arm Cortex-M TrustZone provides software isolation to code, memory and I/O whilst retaining real-time deterministic response and minimal switching overhead
- Application code is separated into “Secure” and “non-Secure” code

SE Manager



- EMLIB provides an abstraction of the mailbox interface, allowing to construct messages and set up cryptoDMA transfers
- On top of EMLIB, SE Manager provides an abstraction of the SE's command set. This is e.g. doing cryptographic operations, writing OTP memory, setting up tamper, initializing the secure boot key, etc. The SE Manager also provides thread synchronization.
- The SE Manager provides these functions:
  - Initialization
  - Device status
  - Accessing user data
  - Debug access control
  - Secure boot setup
  - Internal SE lock API
  - Secure key storage
  - Cryptographic operations
  - Tamper set up

Secure Element Manager Integration with mBed



Cryptographic operations & key handling **(platform agnostic)**

(D)TLS connection handling

mbed TLS 3.x

PSA Storage API

SE lifetime management, tamper, secure boot, …

mbed Crypto

Cryptographic operations & key handling (**space-constrained**)

SE & Crypto Interface

PSA Storage API

SE Manager wrapper

Storage wrapper

OS

NVM3

NVM3

SE Manager

SE Manager gets wrapped by a lightweight translation layer, mapping the mbed Crypto 'Secure Element' interface (crypto_se_driver.h) and crypto acceleration calls to SE Manager calls.

🟦 SL Implementation

🟩 Standard API

Secure Boot Differences Between Series 1 and Series 2 Silicon

	Basic	+ Root of Trust	+ Secure Element
Chip (Example)	xG12	xG22	xG21
Security Implementation	SSB Integrity and immutability of the Public Key (Locked Bits area)	Root of Trust (ROM based)	Root of Trust (ROM based)
Boot from Immutable Memory	No	Yes	Yes
First Stage Bootloader Authentication	No	Yes (Virtual SE)	Yes (SE)
Second Stage Bootloader Authentication	No	Yes(1)	Yes(1)
Application Code Authentication	Optional	Yes(1)	Yes(1)
Naming	Gecko Bootloader Secure Boot	Secure Boot with RTSL	Secure Boot with RTSL

Note (1): with enabled Secure Boot bit

Series 1 without Secure Element

Cortex M4

Flash

Lock Bits Page (Special Area of Flash)

- Customer Supplied Application Public Key

First Stage Bootloader

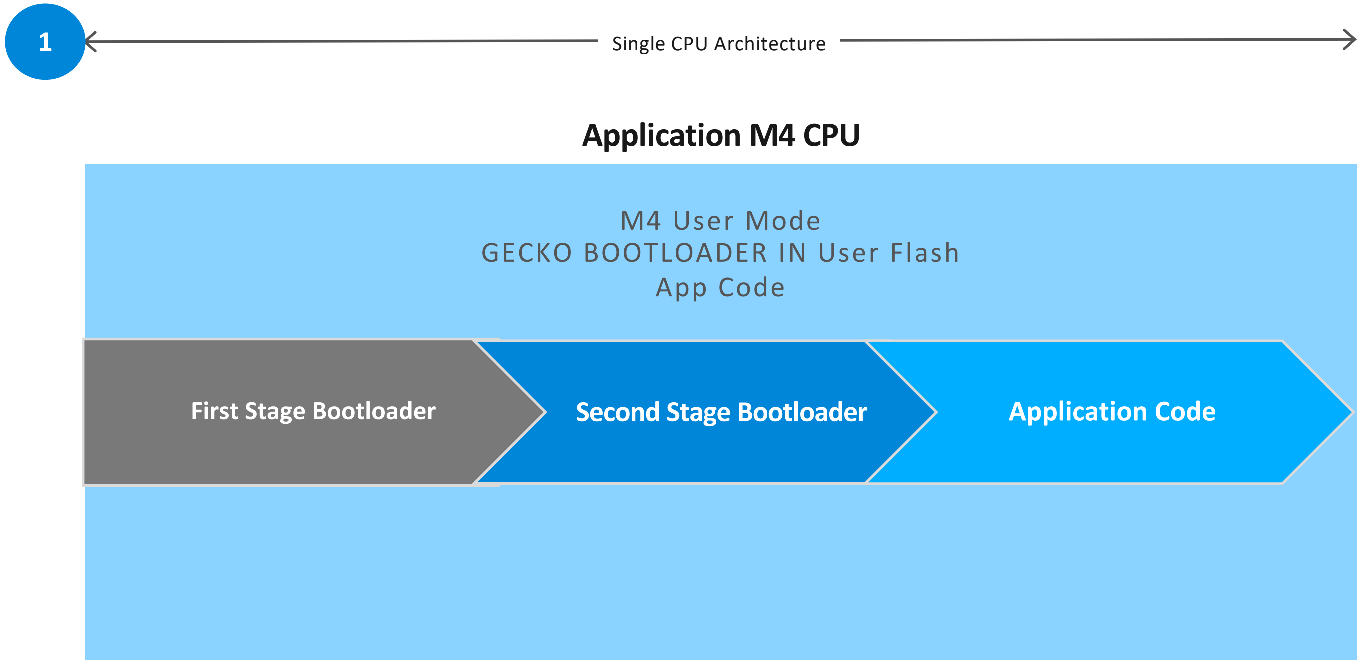
- No Signature check

Second Stage Bootloader

- No Signature check

Application Code

- ECDSA P-256 Signature

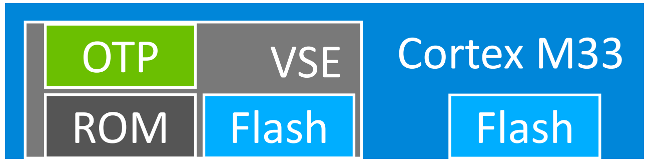


- Check for **Second Stage Bootloader** update and apply it if available

- Check for Application Code update and apply it if available
- Check Application Code Authenticity

- **Execute Code**

Series 2 without Secure Element (only xG22 BLE for now)



SVE OTP

- Secure Boot Enabled bit
- Customer Supplied Application Public Key

ROM

- Silicon Labs FSB Public Key

First Stage Bootloader

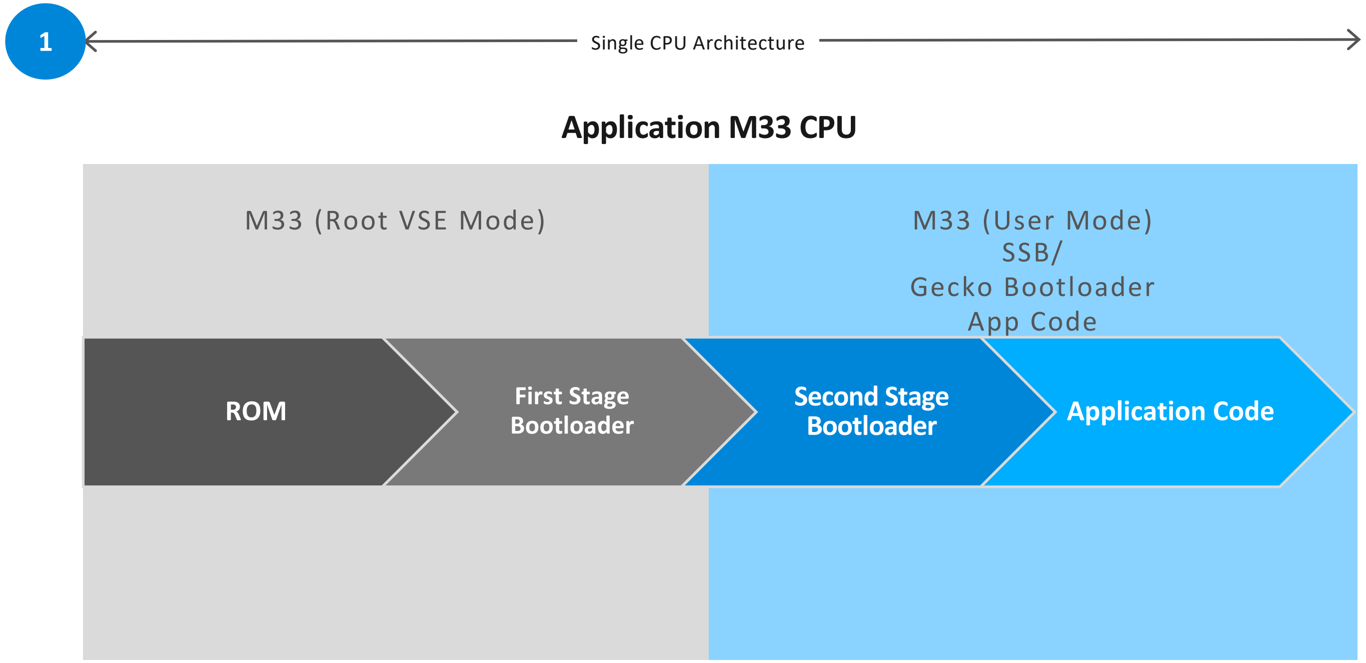
- VSE Code Signature

Second Stage Bootloader

- ECDSA P-256 Signature

Application Code

- ECDSA P-256 Signature



- M33 Reset to Root VSE Mode
- Checks for a staged **First Stage Bootloader** update and apply it if available
- Check for **First Stage Bootloader** Code Authenticity

- Check for **Second Stage Bootloader** update and apply it if available
- Check Secure Boot Enabled Bit
- Check **Second Stage Bootloa** Code Authenticity

- M33 Switches to User Mode
- Check for Application Code update and apply it if available
- Check Application Code Authenticity

- **Execute Code**

Series 2 implementation takes advantage of
Secure Boot with Root of Trust and Secure Loader

Secure Vault – A Deeper Dive

THREATS EVOLVE. SO SHOULD YOUR DEVICE SECURITY.

silabs.com/security

Secure Vault Features – Deeper Dive

**Threats evolve.
So should your
device security.
Introducing
Secure Vault.**

Secure Element

Provide hardware isolation between security functions and host processor

Secure Element Subsystem
Security isolation in hardware

True Random Number Generator
Generate keys for proper cryptography

Secure Boot with RTSL
Only boot authenticated firmware

Crypto Engine
Up to 512-bit ciphers and elliptic curves

Secure Debug with Lock / Unlock
Allow enhanced FAs

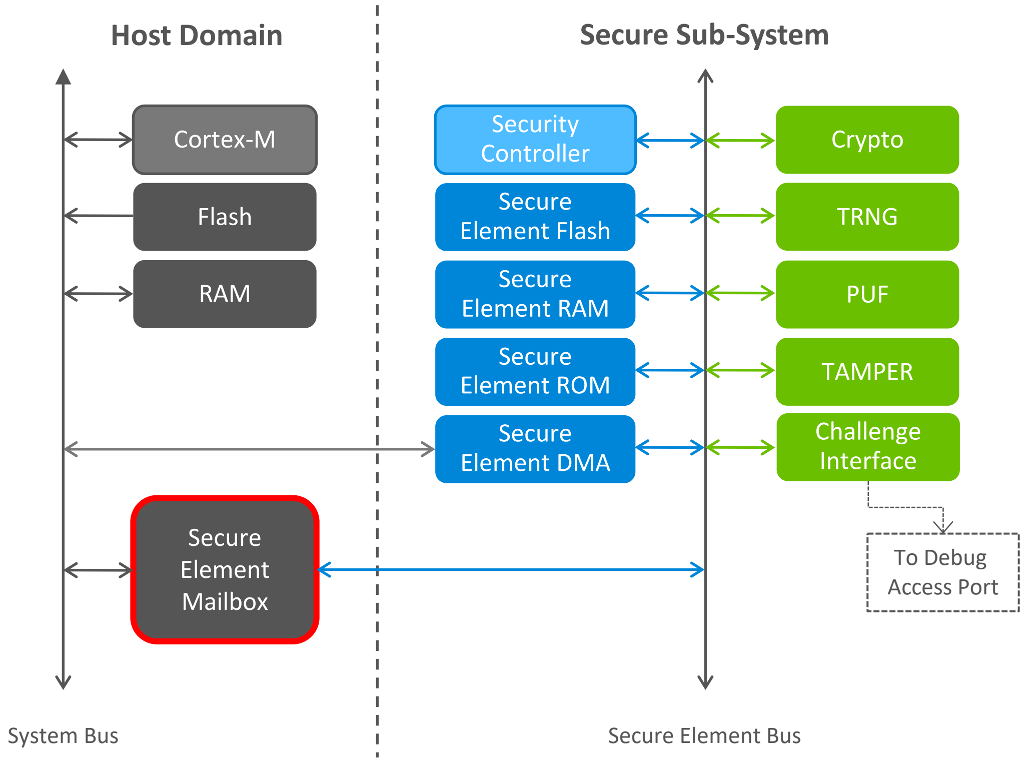
Secure Key Management
Isolate encrypted keys from application code

Secure Attestation
Ensure integrity and authenticity

Anti-tamper
Detect tamper and protect keys/data

DPA Countermeasures
Resist side channel attacks

Secure Element Subsystem



All cryptographic functions use a dedicated crypto-coprocessor

- Random number generation
- Symmetric encryption/decryption
- Hashing
- Keypair generation
- Key storage
- Signing / Verifying signatures

Limited accessibility to crypto-coprocessor

- Via a Host mailbox interface
- Debug pins (with Debug Challenge Interface, or DCI)

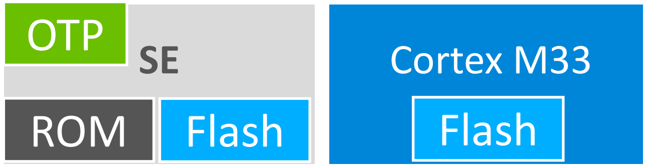
Crypto-coprocessor is not customer programmable

- (but can be securely updated)

Crypto-coprocessor benefits

- Increases security: access to crypto functions is tightly controlled, supports key isolation, supports Secure Boot
- Frees the Host Processor for other tasks

Secure Boot with Root of Trust and Secure Loader



SE OTP

- Secure Boot Enabled bit
- Customer Supplied Application Public Key

ROM

- Silicon Labs FSB Public Key

First Stage Bootloader

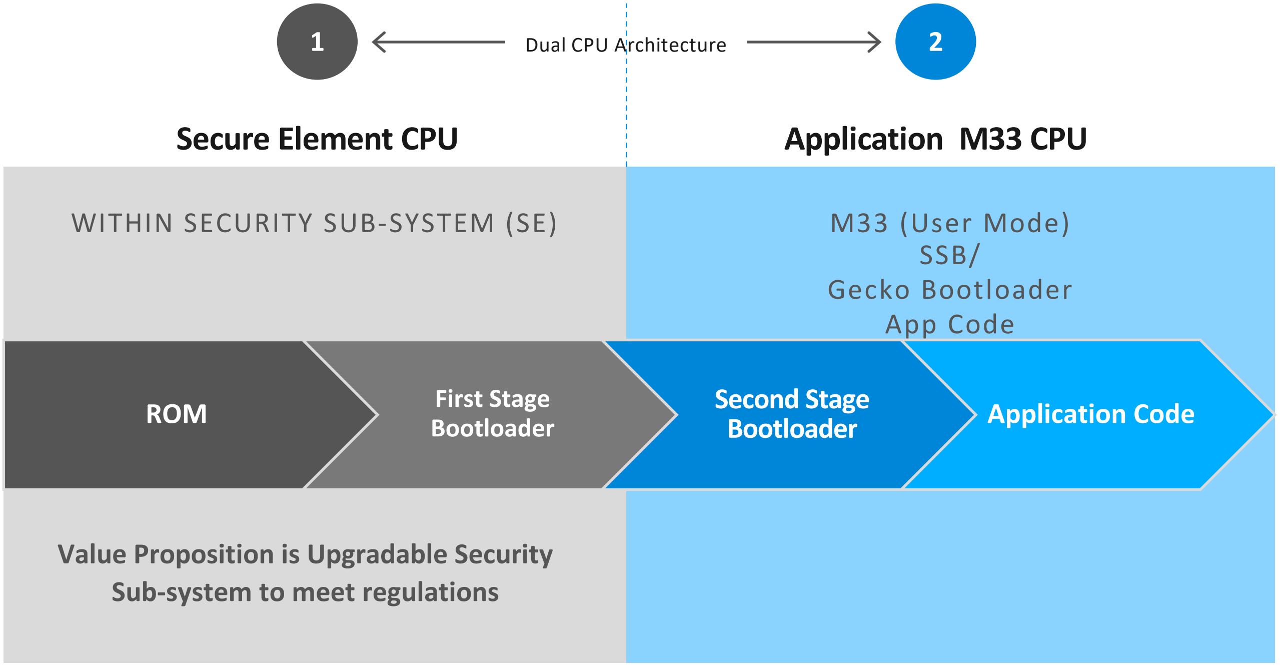
- SE Code Signature

Second Stage Bootloader

- ECDSA P-256 Signature

Application Code

- ECDSA P-256 Signature



Value Proposition is Upgradable Security Sub-system to meet regulations

- Checks for a staged **First Stage Bootloader** update and apply it if available
- Check **First Stage Bootloader** Code Authenticity
- M33 held in Reset

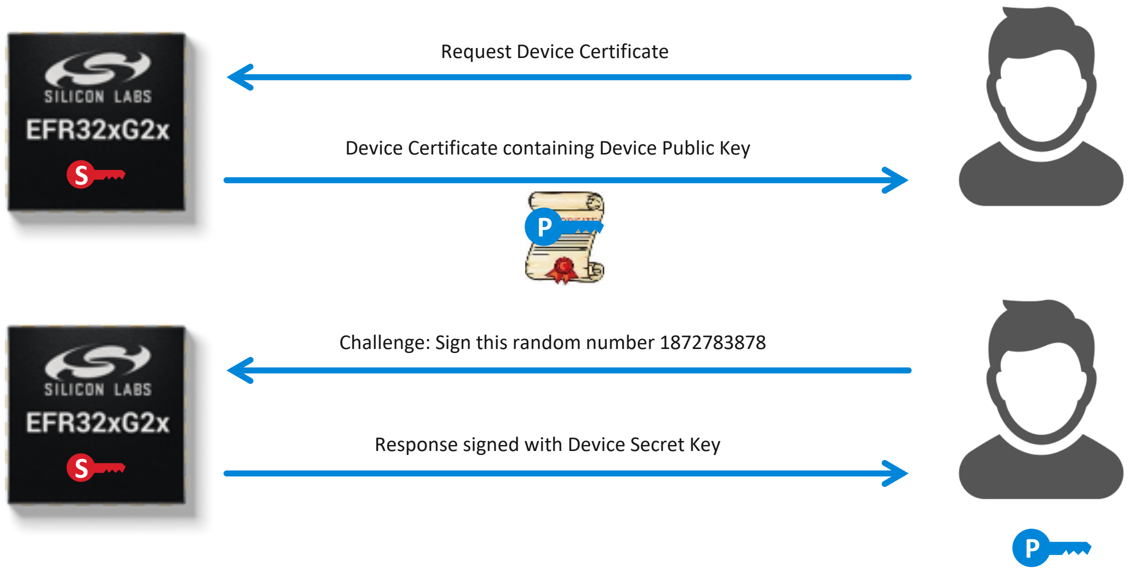
- Check for **Second Stage Bootloader** update and apply it if available
- Check Secure Boot Enabled Bit
- Check **Second Stage Bootloader** Code Authenticity

- SE Releases M33 from Reset
- Check for Application Code update and apply it if available
- Check Application Code Authenticity

- **Execute Code**

Series 2 implementation takes advantage of
Secure Boot with Root of Trust and Secure Loader

Secure Attestation



Request Device Certificate

Device Certificate containing Device Public Key

Challenge: Sign this random number 1872783878

Response signed with Device Secret Key

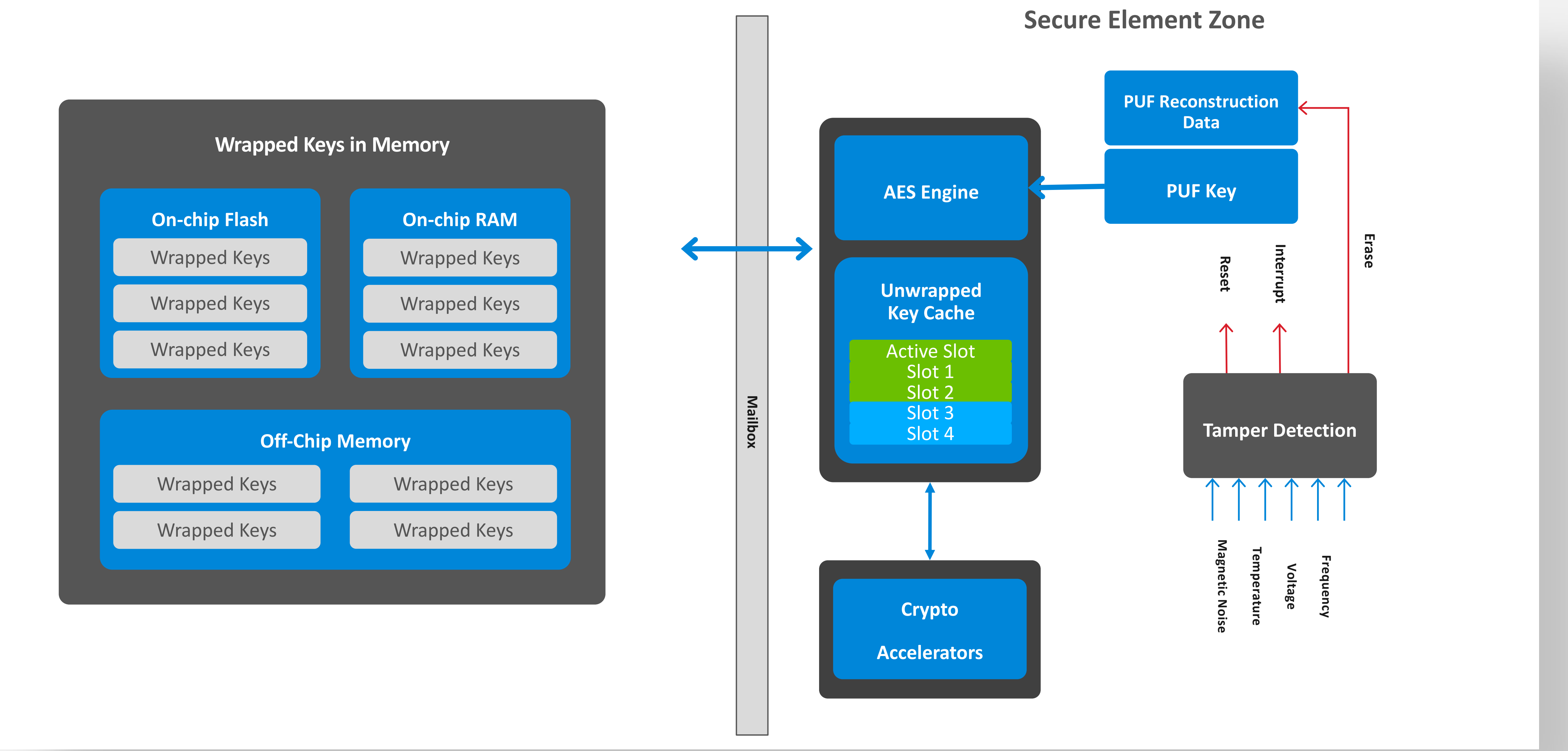
Why

- Many systems use a UID to identify devices, but the UID is public (can be copied)
- Developers are concerned with the authenticity of their devices
- Most successful companies suffer counterfeit products and “ghost shifts”

Silicon Labs

- Secure Vault devices generate a unique device ECC keypair on-chip and securely store the private key
- The device secret key never leaves the chip
- During production, the test program reads the device public key, places it in the certificate signs the device certificate with an HSM secret key, and stores it back into the chip in OTP memory
- An external service can now request the certificate chain from the device and our CA web server, retrieve the unique device public key.
- The external service can then perform a “Challenge Response” to the chip at any time during the life of the product to Authentic the chip is genuine Silicon Labs silicon

Secure Key Management



Secure Element Zone

Wrapped Keys in Memory

On-chip Flash

Wrapped Keys

Wrapped Keys

Wrapped Keys

On-chip RAM

Wrapped Keys

Wrapped Keys

Wrapped Keys

Off-Chip Memory

Wrapped Keys

Wrapped Keys

Wrapped Keys

Wrapped Keys

Mailbox

AES Engine

Unwrapped Key Cache

Active Slot

Slot 1

Slot 2

Slot 3

Slot 4

Crypto

Accelerators

PUF Reconstruction Data

PUF Key

Reset

Interrupt

Erase

Tamper Detection

Magnetic Noise

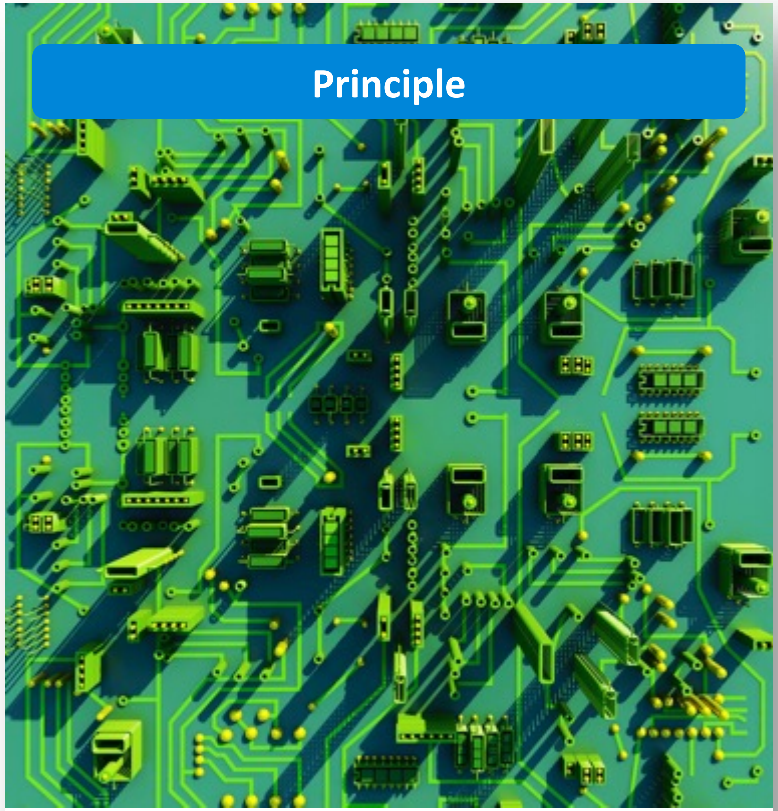
Temperature

Voltage

Frequency

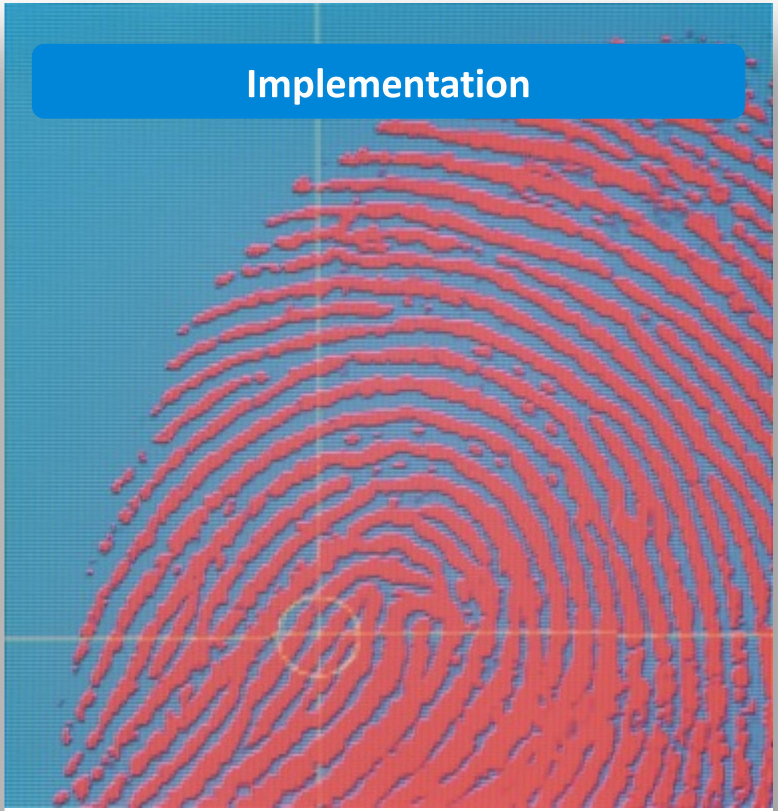
SRAM-PUF TECHNOLOGY (Physically Unclonable Function)

Principle



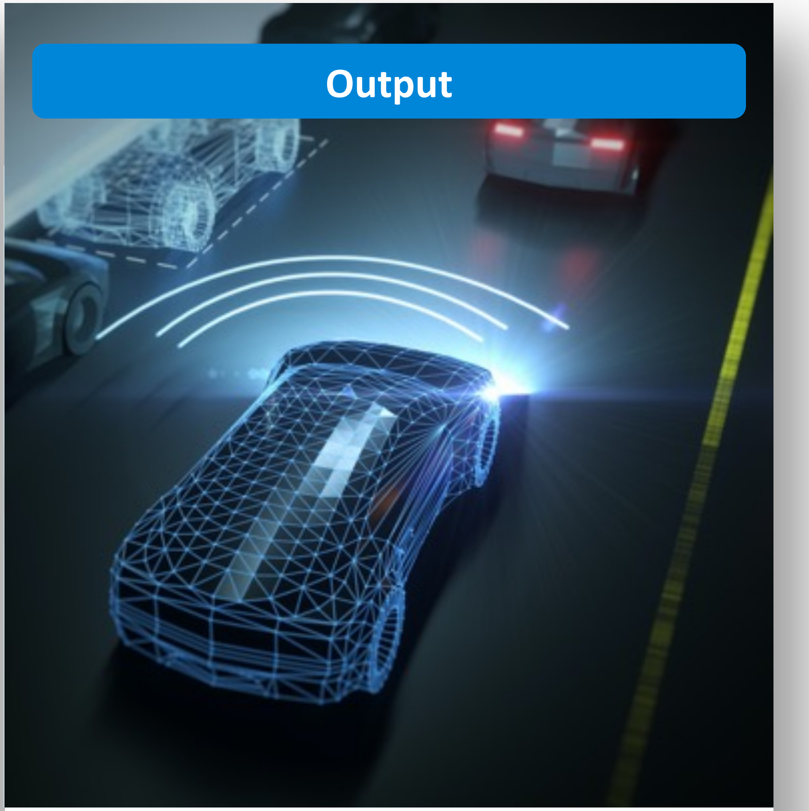
- Based on process disparities during manufacturing
- Create small variations of transistor properties
- Every chip is unique and unclonable

Implementation



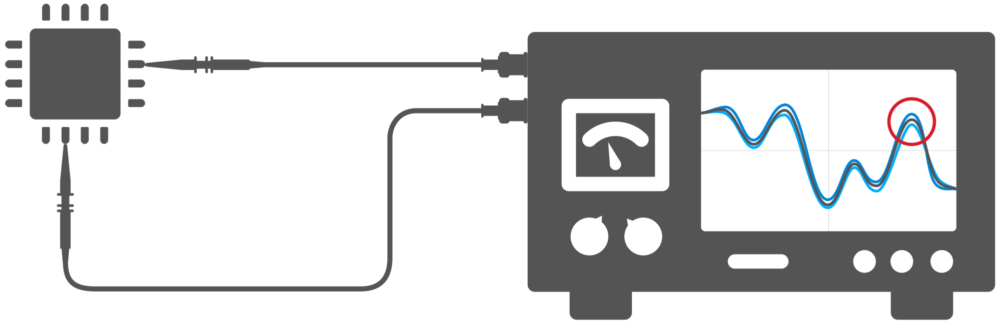
- SRAM startup values are random
- An array creates a unique fingerprint

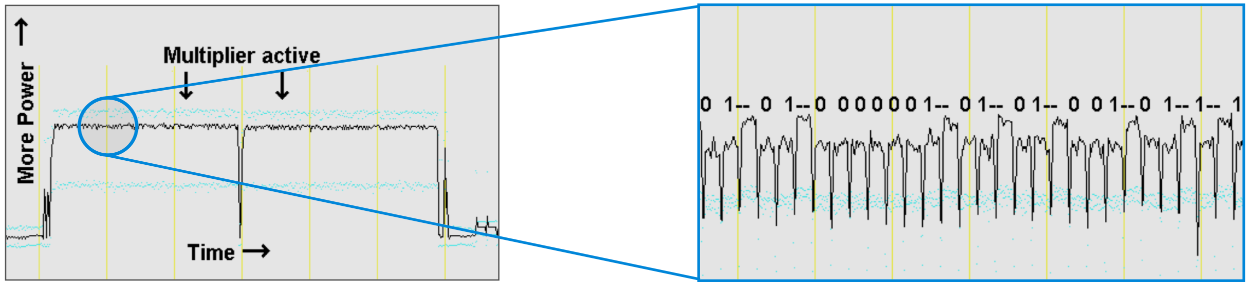
Output



- Fingerprint is converted into a unique root key per chip

Differential Power Analysis Countermeasures





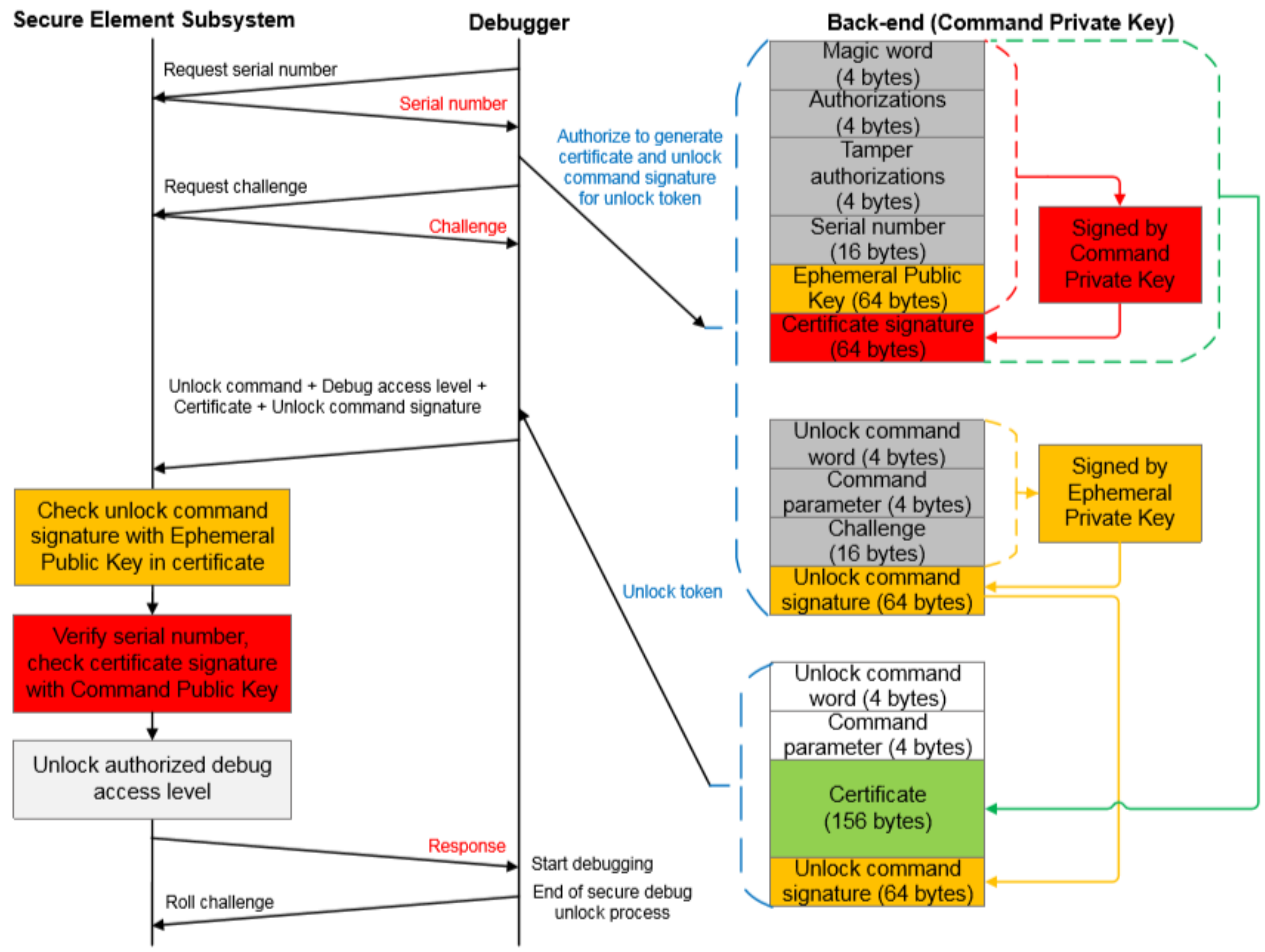
Why

- Cryptographic systems can expose secrets via the power trace
- This is called a Differential Power Analysis (DPA) attack
- Local ‘Hands-On’ accessibility is typically normal with IoT devices
- Low power engines are intrinsically hardened against such attacks
- DPA equipment has recently become easily available for little cost

Silicon Labs

- Devices contain patented countermeasures that randomize power consumption to further hide the secrets

Secure Debug with Lock/Unlock



Secure Element Subsystem
Debugger
Back-end (Command Private Key)
Request serial number
Serial number
Authorize to generate certificate and unlock command signature for unlock token
Request challenge
Challenge
Magic word (4 bytes)
Authorizations (4 bytes)
Tamper authorizations (4 bytes)
Serial number (16 bytes)
Ephemeral Public Key (64 bytes)
Certificate signature (64 bytes)
Signed by Command Private Key
Unlock command + Debug access level + Certificate + Unlock command signature
Unlock command word (4 bytes)
Command parameter (4 bytes)
Challenge (16 bytes)
Unlock command signature (64 bytes)
Signed by Ephemeral Private Key
Check unlock command signature with Ephemeral Public Key in certificate
Verify serial number, check certificate signature with Command Public Key
Unlock token
Unlock command word (4 bytes)
Command parameter (4 bytes)
Certificate (156 bytes)
Unlock command signature (64 bytes)
Unlock authorized debug access level
Response
Start debugging
End of secure debug unlock process
Roll challenge

Create Unlock Certificate

- Request serial number of device over the Debug Challenge Interface (DCI) and check to make sure the correct device is being debugged
- Debug Access Level defines whether the Secure or Non-secure part of the CM33 is to be debugged and at what privilege level
- Create the Unlock Certificate containing the Ephemeral Public Key and sign with the Private Command Key.

Request a Challenge and build Challenge Response

- Request a challenge from the device which will return a nonce (random number)
- Sign the Unlock Command, Debug Access Level, and the Challenge with the Ephemeral Private Key

Build and send the Unlock Token

- Build a packet with the Unlock Command, Debug Access Level, Unlock Certificate, and Ephemeral Signature and send to the device.

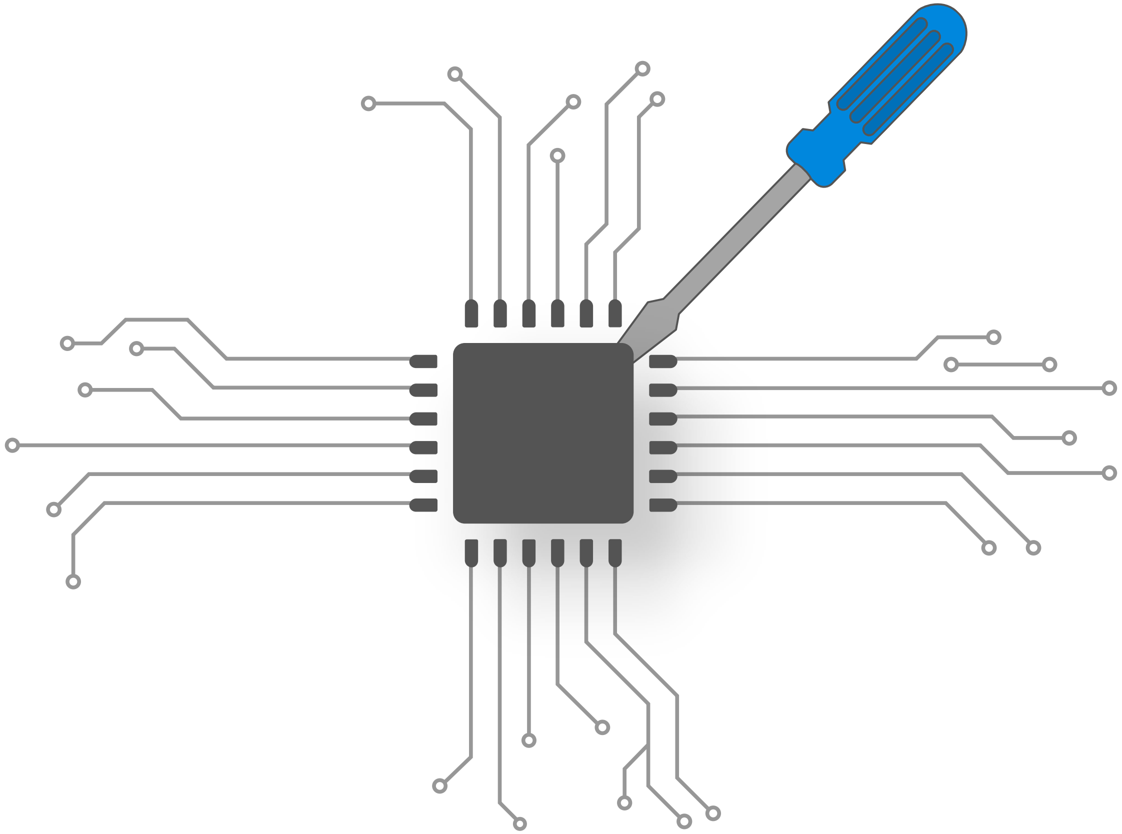
Verify the Unlock Token and open the debug port

- Check the Unlock Certificate Signature with Command Public key stored in OTP
- Check the Ephemeral Signature with the Ephemeral Key that was delivered in the Unlock Certificate

Close the Debug Session

- Issue a command to the device to roll the Challenge. This will obsolete the Unlock Token as Ephemeral signature will no longer match

Anti-Tamper (1/2)



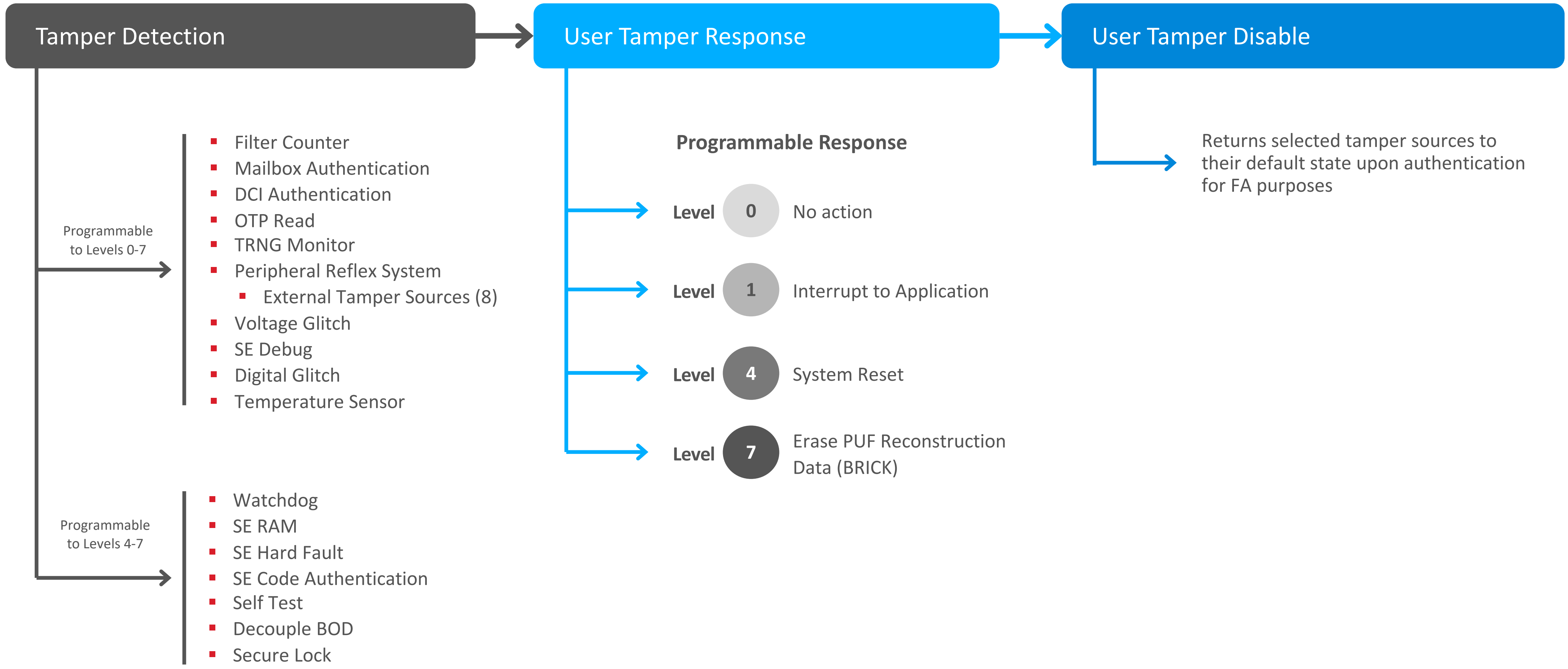
Why

- Many attacks force a device outside its standard operating range(s)
  - temperature, voltage, clock-inputs, magnetic noise
  - Debuggers running at a high rate, reboots at a high rate
- Cost of these attacks is now low enough for both large scale and hobbyists

Silicon Labs

- Implemented an ability to detect when these attacks happen
  - Voltage, clock, temperature and magnetic tamper detectors in our devices
  - Secure boot, secure debug use counters to flag abnormal behavior
  - External triggers from broken enclosures via buttons and traces
- Implemented an ability to respond to these attacks
  - Programmable tamper response
    - Includes an ability to perform rapid deletion of Secure Key Storage (forced bricking)

Anti-Tamper $^{(2/2)}$



Tamper Detection

User Tamper Response

User Tamper Disable

Programmable to Levels 0-7

- Filter Counter
- Mailbox Authentication
- DCI Authentication
- OTP Read
- TRNG Monitor
- Peripheral Reflex System
  - External Tamper Sources (8)
- Voltage Glitch
- SE Debug
- Digital Glitch
- Temperature Sensor

Programmable to Levels 4-7

- Watchdog
- SE RAM
- SE Hard Fault
- SE Code Authentication
- Self Test
- Decouple BOD
- Secure Lock

Programmable Response

Level 0	No action
Level 1	Interrupt to Application
Level 4	System Reset
Level 7	Erase PUF Reconstruction Data (BRICK)

Returns selected tamper sources to their default state upon authentication for FA purposes

Cryptography Engine $^{(1/4)}$

What & Why

- Cryptography is be used to encrypt, decrypt and sign data
- Any secure system requires:
  - integrity (signatures)
  - authenticity (signatures)
  - confidentiality (encryption)
- Cryptography should be common-place
  - Firmware updates,
  - On sensitive data
  - In communications with other devices or the cloud
    - Eg. Bluetooth, ZigBee, TLS (IP-connections) etc.

Silicon Labs

- Lightning fast hardware engines that supports all major IoT ciphers
- An optimized mbed TLS library to our hardware engines
- SW libraries (eg. Wireless stacks, & secure bootloaders) use hardware crypto engines

Hardware Cryptography is typically faster, more energy efficient and more secure than SW implementations

Cryptography Engine $^{(2/4)}$

Protocol Usage & Support

Series 1

		Wireless							TCP/IP		
	Cipher	ZigbeePRO	Zigbee IP	Thread	Z-Wave	Bluetooth	Homekit	WMBus	SSL 3.0	TLS 1.2	TLS 1.3
Symmetric Encryption	Triple-DES							🟧	🟧		
	AES <=256	🟦	🟦	🟦	🟦	🟦		🟦		🟦	🟦
	CHACHA20						🟧				🟧
Asymmetric Encryption	RSA								🟧	🟧	
	ECC NIST <=256	🔷	🔷	🔷		🔷				🔷	🔷
	ECC NIST >256	🟧					🟧			🟧	🟧
	ECC Curve25519				🟧		🟧			🟧	🟧
Hash Function	SHA-1	🟦			🟦				🟦		
	SHA-2 <=256		🟦	🟦		🟦				🟦	🟦
	SHA-2 >256						🟧			🟧	🟧
	POLY1305						🟧				🟧

Series 2

		Wireless							TCP/IP		
	Cipher	ZigbeePRO	Zigbee IP	Thread	Z-Wave	Bluetooth	Homekit	WMBus	SSL 3.0	TLS 1.2	TLS 1.3
Symmetric Encryption	Triple-DES							🟧	🟧		
	AES <=256	🟦	🟦	🟦	🟦	🟦		🟦		🟦	🟦
	CHACHA20						🟦				🟦
Asymmetric Encryption	RSA								🟧	🟧	
	ECC NIST <=256	🟦	🟦	🟦		🟦				🟦	🟦
	ECC NIST >256	🟦					🟦			🟦	🟦
	ECC Curve25519						🟦			🟦	🟦
Hash Function	SHA-1	🟦			🟦				🟦		
	SHA-2 <=256		🟦	🟦		🟦				🟦	🟦
	SHA-2 >256						🟦			🟦	🟦
	POLY1305						🟦				🟦

🟧	Software Only	OK
🔷	Hardware + SW	Better
🟦	Hardware Only	Best

Cryptography Engine (3/4)

Protocol Usage & Support

- Not all Series 2 devices have the full hardware support to implement all ciphers in hardware
- xG22 is designed to focus on cost targets
  - Some additional security ciphers (over Series 1) in hardware
  - All ciphers still implemented whether in software or hardware

Series 1

		Wireless							TCP/IP		
	Cipher	ZigbeePRO	Zigbee IP	Thread	Z-Wave	Bluetooth	Homekit	WMBus	SSL 3.0	TLS 1.2	TLS 1.3
Symmetric Encryption	**Triple-DES**							🟧	🟧		
	AES	🟦	🟦	🟦	🟦	🟦		🟦		🟦	🟦
	CHACHA20						🟧				🟧
Asymmetric Encryption	**RSA**								🟧	🟧	
	ECC NIST <=256	🔷	🔷	🔷		🔷				🔷	🔷
	ECC NIST >256	🟧					🟧			🟧	🟧
	ECC Curve25519				🟧		🟧			🟧	🟧
Hash Function	**SHA-1**	🟦			🟦				🟦		
	SHA-2 <=256		🟦	🟦		🟦				🟦	🟦
	SHA-2 >256						🟧			🟧	🟧
	POLY1305						🟧				🟧

Series 2

		Wireless							TCP/IP		
	Cipher	ZigbeePRO	Zigbee IP	Thread	Z-Wave	Bluetooth	Homekit	WMBus	SSL 3.0	TLS 1.2	TLS 1.3
Symmetric Encryption	**Triple-DES**							🟧	🟧		
	AES	🟦	🟦	🟦	🟦	🟦		🟦		🟦	🟦
	CHACHA20						🟦				🟦
Asymmetric Encryption	**RSA**								🟧	🟧	
	ECC NIST <=256	🟦	🟦	🟦		🟦				🟦	🟦
	ECC NIST >256	🟦					🟦			🟦	🟦
	ECC Curve25519						🟦			🟦	🟦
Hash Function	**SHA-1**	🟦			🟦				🟦		
	SHA-2 <=256		🟦	🟦		🟦				🟦	🟦
	SHA-2 >256						🟦			🟦	🟦
	POLY1305						🟦				🟦

xG22

		Wireless							TCP/IP		
	Cipher	ZigbeePRO	Zigbee IP	Thread	Z-Wave	Bluetooth	Homekit	WMBus	SSL 3.0	TLS 1.2	TLS 1.3
Symmetric Encryption	**Triple-DES**							🟧	🟧		
	AES <=256	🟦	🟦	🟦	🟦	🟦		🟦		🟦	🟦
	CHACHA20						🟧				🟧
Asymmetric Encryption	**RSA**								🟧	🟧	
	ECC NIST <=256	🟦	🟦	🟦		🟦				🟦	🟦
	ECC NIST >256	🟧					🟧			🟧	🟧
	ECC Curve25519				🟧		🟧			🟧	🟧
Hash Function	**SHA-1**	🟦			🟦				🟦		
	SHA-2 <=256		🟦	🟦		🟦				🟦	🟦
	SHA-2 >256						🟧			🟧	🟧
	POLY1305						🟧				🟧

🟧	Software Only	OK
🔷	Hardware + SW	Better
🟦	Hardware Only	Best

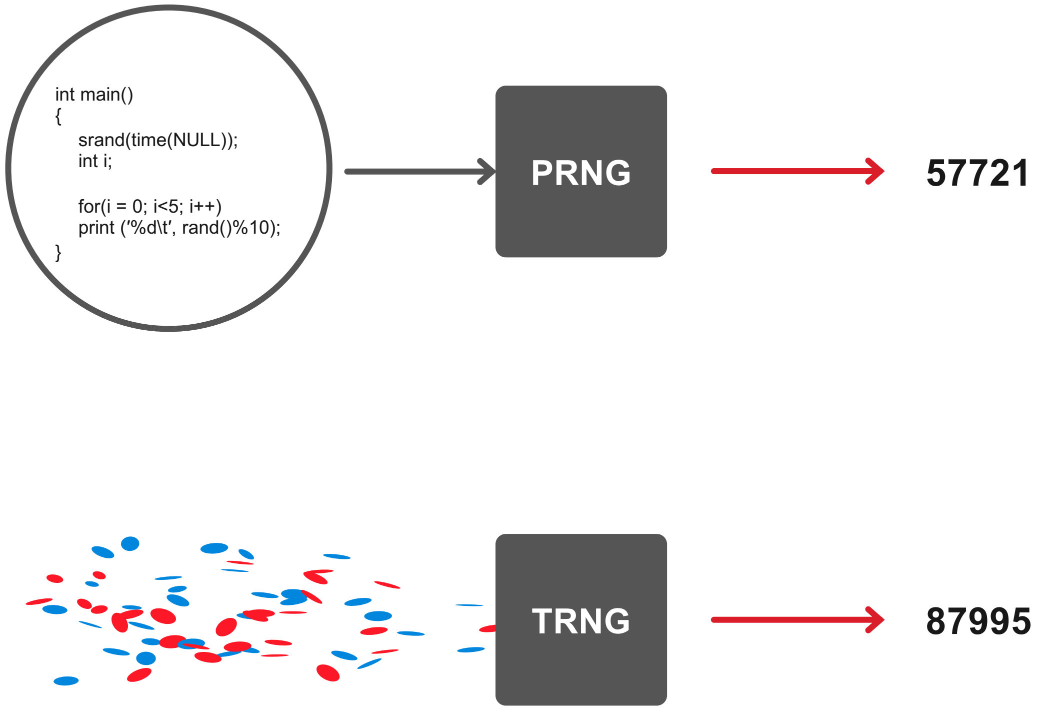
Cryptography Engine (4/4)

Cryptography Comparison Across Products*

		Series 1	Series 2	Notes
AES	Engine speed (128/256-bits)	54/75 cycles	22/30 cycles	2x faster on Series 2
PKI	Engine speed (P-256 sign)	~2500k cycles	~350k cycles	7x faster on Series 2
	Autonomous	No	Yes	Less CPU demand
	Cipher support (bits)	P≤256	P≤256, P≤521, Curve25519	Zigbee PRO, HomeKit, Z-Wave S2
Hash	Digest size	SHA≤256	SHA≤256 SHA≤512	HomeKit
	Engine speed (SHA-256)	66 cycles 512 bit	66 cycles 512 bit	
AEAD	ChaCha20-Poly1305		Yes	HomeKit / TLS 1.3
Key Protection	DPA countermeasures		Yes (AES and ECC)	Protection from side channel attacks
	Key Isolation		Yes	
	Secure key storage		Yes	

**Subject to Hardware support*
In Development

True Random Number Generators



```
int main()
{
    srand(time(NULL));
    int i;

    for(i = 0; i<5; i++)
    print ('%d\t', rand()%10);
}
```

A pseudo-random number generator uses a set of algorithms to produce numbers

A true random number generator uses an unpredictable physical source to produce numbers

Why

- Random numbers are
  - often used as secret keys
  - must be unpredictable to the adversary to be effective
  - necessary for many cryptographic algorithms and communication
    protocols to work
- True randomness is hard

Silicon Labs

- Our devices contain a True Random Number Generator (TRNG) peripheral that generates secret, high entropy data
- Both conditioning and health tests are performed in hardware
- Compliant with NIST SP 800-90A/B/C and AIS-31

Security – Cryptography Basics

THREATS EVOLVE. SO SHOULD YOUR DEVICE SECURITY.

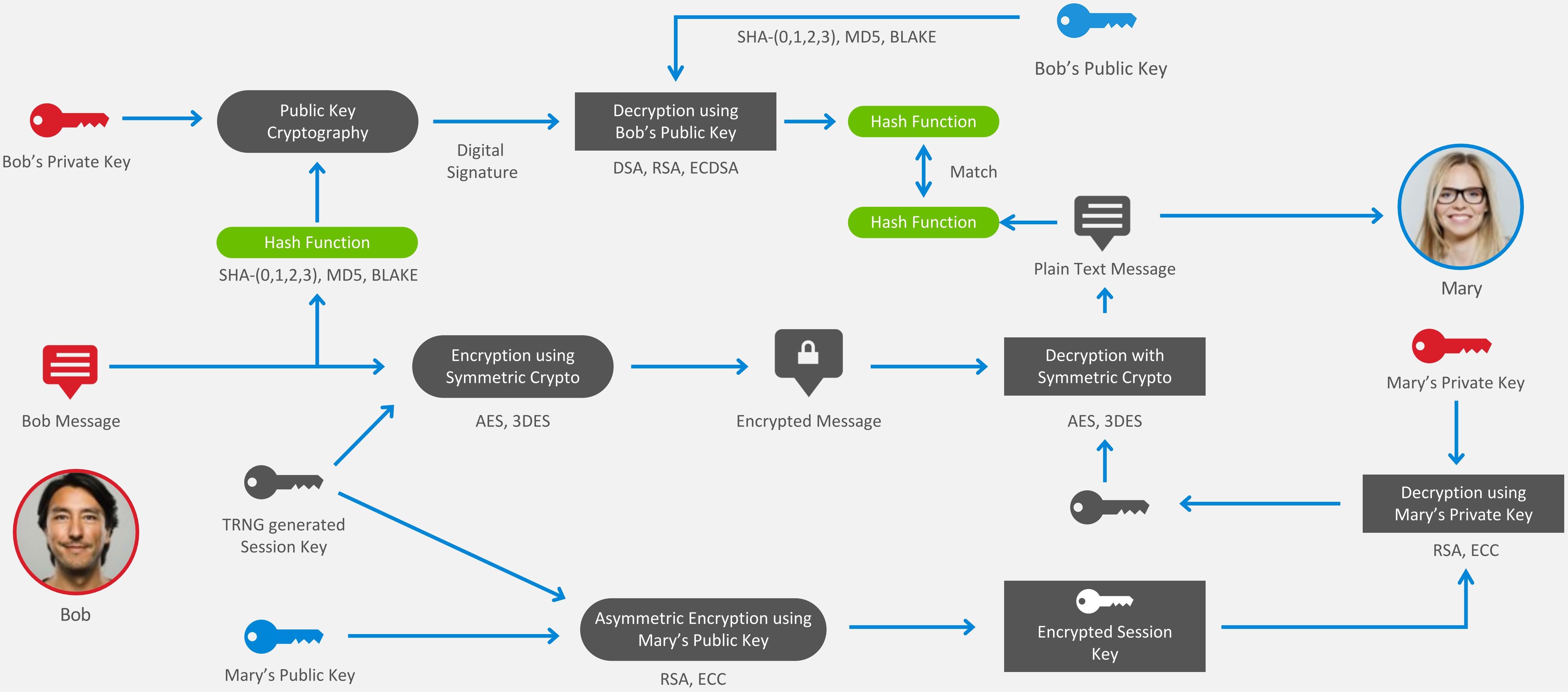
silabs.com/security

Basics of Modern Cryptography

But how are we sure that Bob

But how did Mary get the key?

has not been tampered with?



Bob's Private Key

Public Key Cryptography

Digital Signature

Decryption using Bob's Public Key

DSA, RSA, ECDSA

SHA-(0,1,2,3), MD5, BLAKE

Bob's Public Key

Hash Function

Match

Hash Function

Plain Text Message

Mary

Hash Function

SHA-(0,1,2,3), MD5, BLAKE

Bob Message

Encryption using Symmetric Crypto

AES, 3DES

Encrypted Message

Decryption with Symmetric Crypto

AES, 3DES

Mary's Private Key

TRNG generated Session Key

Decryption using Mary's Private Key

RSA, ECC

Bob

Mary's Public Key

Asymmetric Encryption using Mary's Public Key

RSA, ECC

Encrypted Session Key

Thank you!