Getting Started with Secure Software

Class 3: Architecting a Secure Solution

April 22, 2020

Jacob Beningo

DesignNews

Presented by:

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

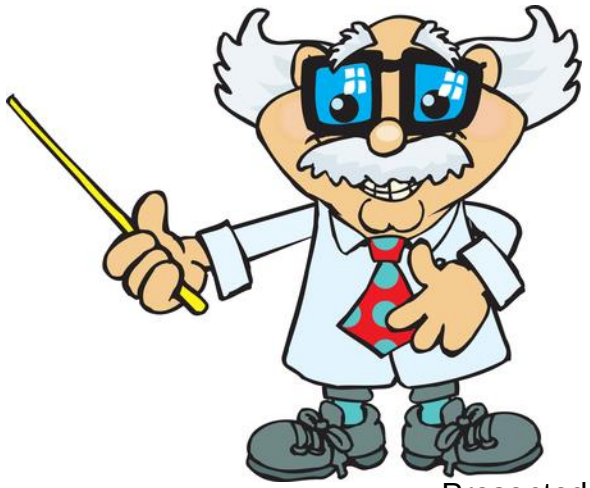
Course Overview

Topics:

- Introduction to Platform Security Architecture (PSA)
- Performing a Security Threats Analysis
- **Architecting a Secure Solution**
- Secure Boot and the Root-of-Trust
- Secure Frameworks and Ecosystems

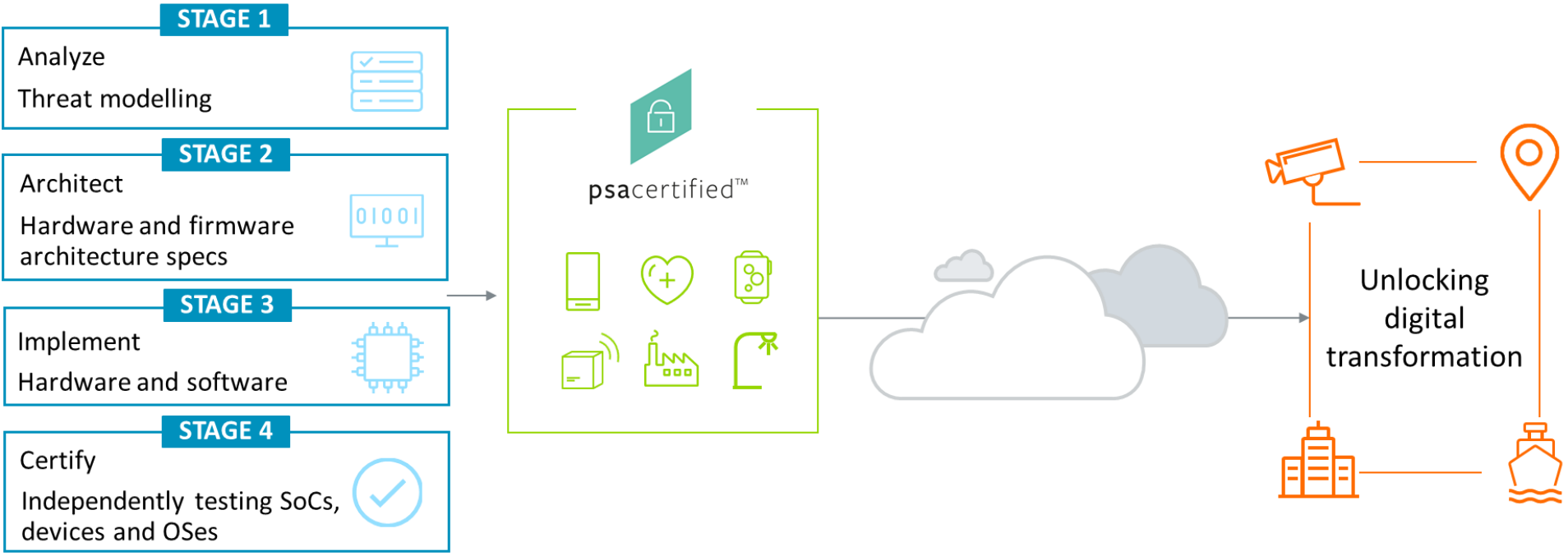
Session Overview

- PSA Review
- Security through Isolation – Option 1
- Security through Isolation – Option2



Presented by:

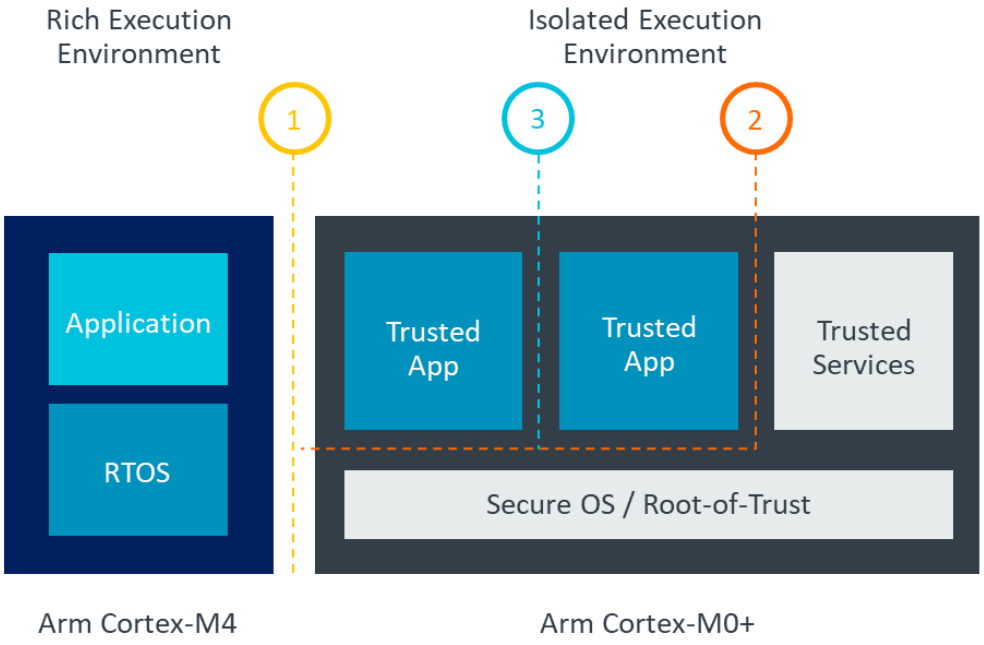
Platform Security Architecture (PSA)



PSA: enabling right-sized device security

Security Through Isolation – Option #1

Multi-core Processing



Hardware-based Isolation within PSoC 64 Secure MCUs

Hardware based isolation within PSoC 64 Secure MCUs enables secure element functionality and reduces the attack surface

Three levels of isolation

1. Secure execution environment (SEE) isolated from rich execution environment
2. Root-of-trust and trusted services isolation within SEE
3. Application isolation within SEE

Security Through Isolation – Option #1

Execution Environments

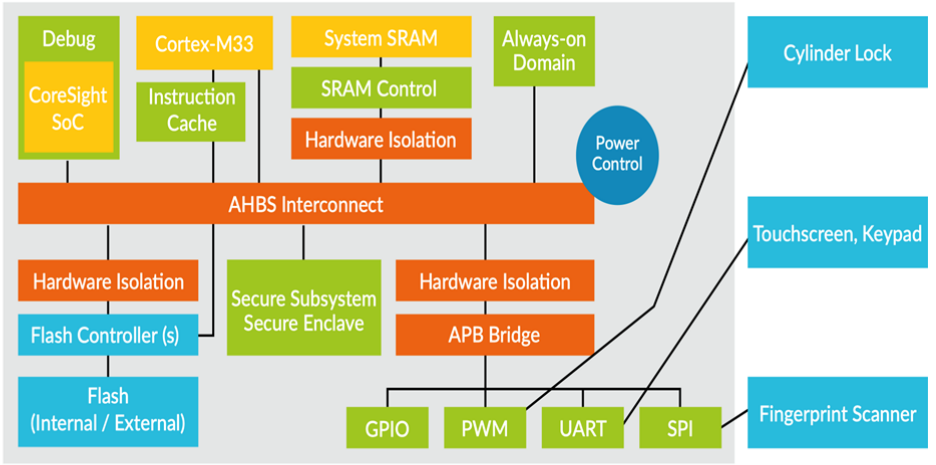
Secure Processing Environment (SPE) is for the sensitive assets and the code that manages them

Non-secure Processing Environment (NSPE) is where the main application and communication firmware executes.

The following hardware mechanisms can be used to implement the PSA isolation:

- Memory Protection Unit (MPU) based isolation.
- TrustZone-based isolation.
- Dual Micro Processor Units (MPUs) or Multiple CPUs.
- Trusted Subsystem (integrated/off-chip).
- Other isolation solutions, such as Custom Logic.

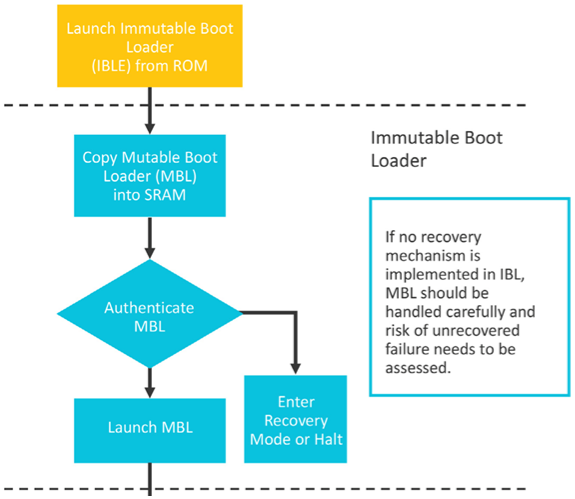
Chip Level Isolation



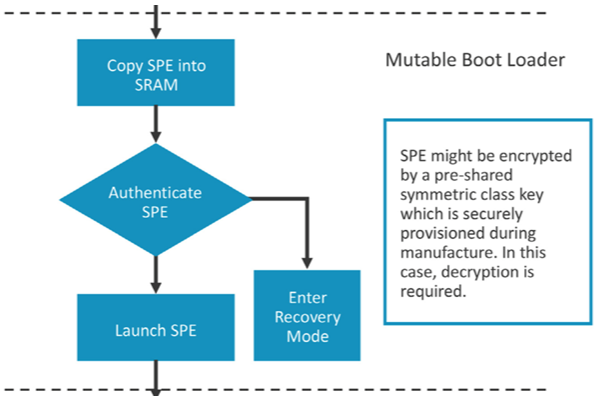
Security Through Isolation – Option #1

Smart Door Lock – Trusted Boot Design

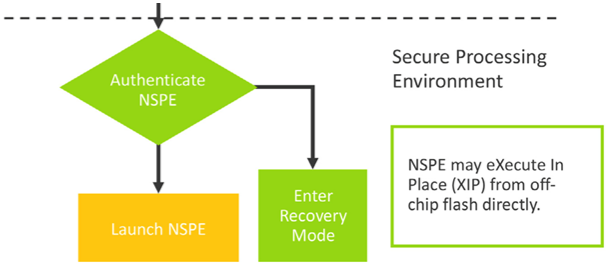
Stage 1 – Immutable BL



Stage 2 – Mutable BL



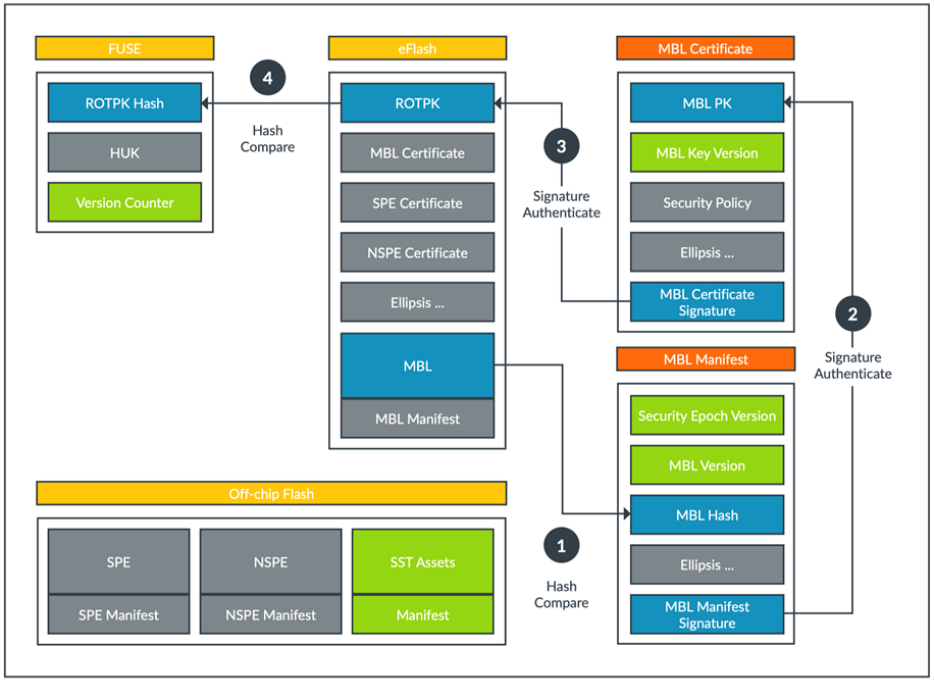
Stage 3 – SPE



Security Through Isolation – Option #1

Smart Door Lock – Authentication Chain

- The Mutable Boot Loader (MBL) hash is calculated in step 1, then the MBL hash is compared with the hash embedded in its manifest.
- The manifest is signed by the MBL private key offline and it can be validated in step 2, by the corresponding MBL Public Key contained in the MBL certificate provisioned on the device.
- The MBL certificate is upgradable and needs to be further validated by the ROTPK from the key pair used to sign the MBL certificate. This validation is done in step 3.
- The ROTPK is not stored in the OTP to reduce the cost. In step 4, the hash of the ROTPK is calculated and compared with the value stored in OTP for tamper detection.

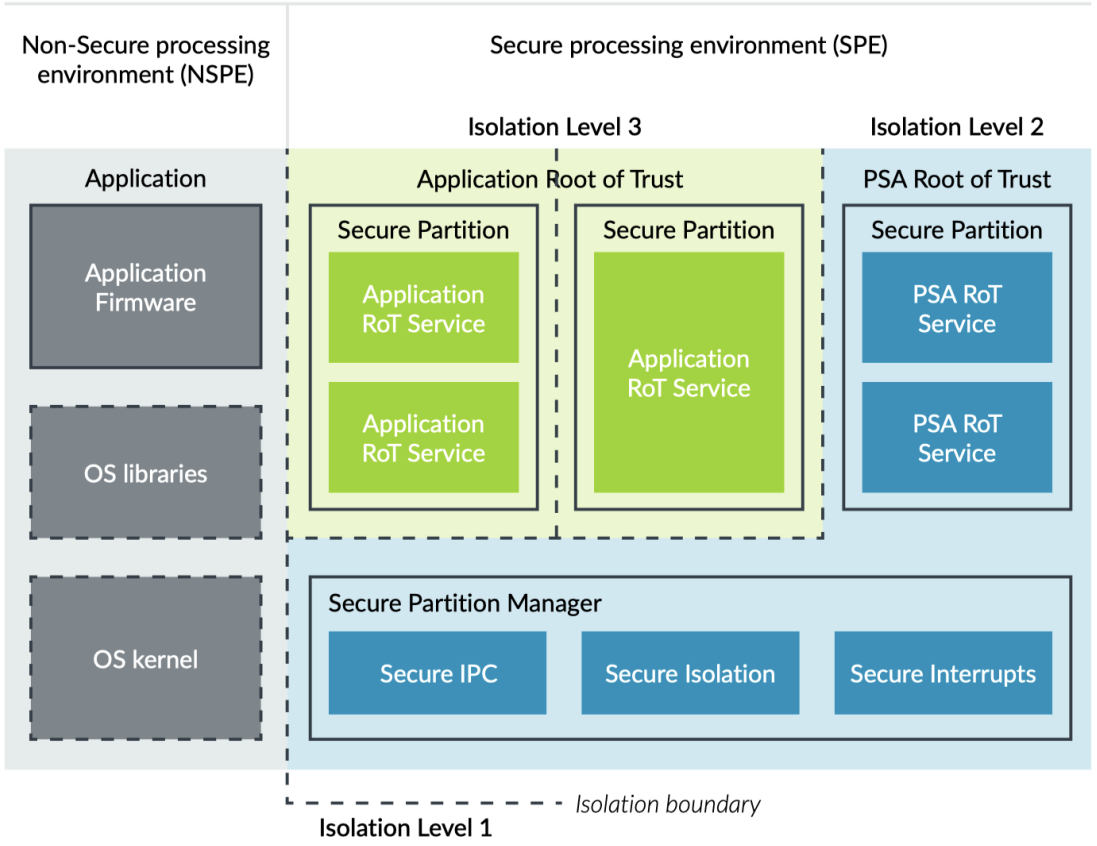


Security Through Isolation – Option #1

Smart Door Lock – Isolation Levels

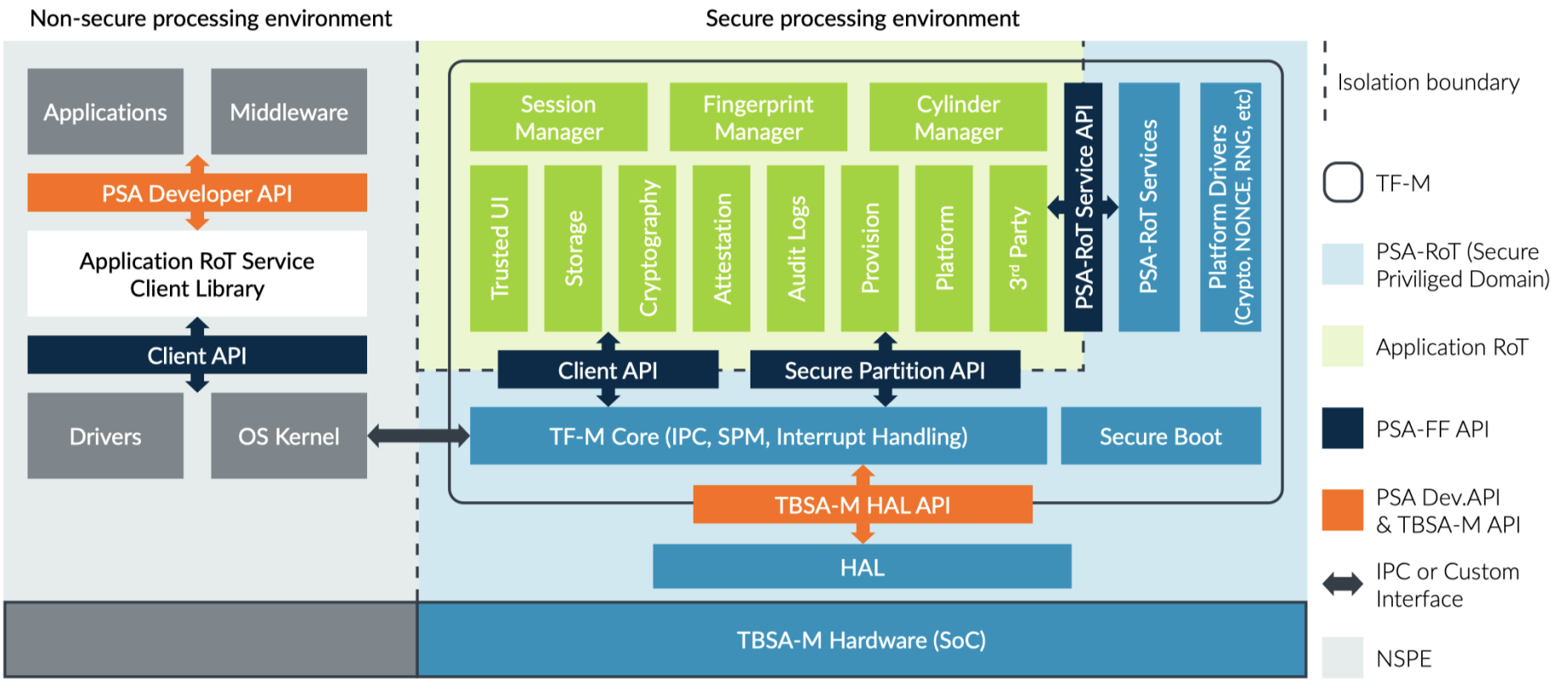
Asset Partitioning

- Confidential Code
- Critical Process
- Secure Peripheral
- Shared Peripheral
- NSPE code



Security Through Isolation – Option #1

Smart Door Lock – Isolation Levels



Security Through Isolation – Option #2

arm TRUSTZONE

Normal environment (Non-Secure)

Application Examples
- User applications
- RTOS
- Device drivers
- Protocol stacks

Normal Resources
- General peripherals

Handler Mode

Thread Mode

Protected environment (Secure)

Handler Mode

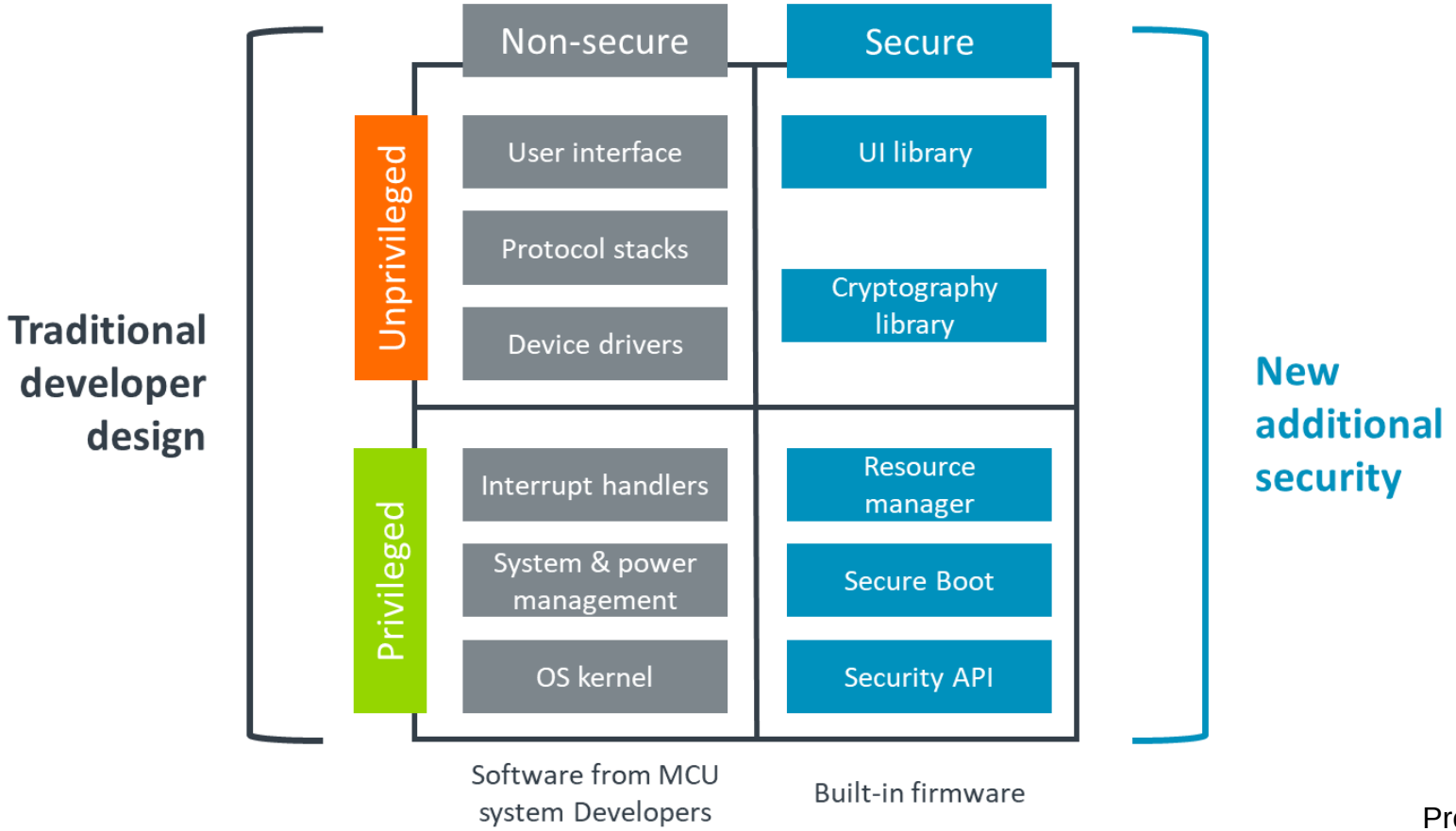
Thread Mode

Secure Software Examples
- Secure Boot
- Cryptography libraries
- Authentication
- RTOS support APIs / RTOS

Secure Resources
- Secure storage
- Crypto accelerators

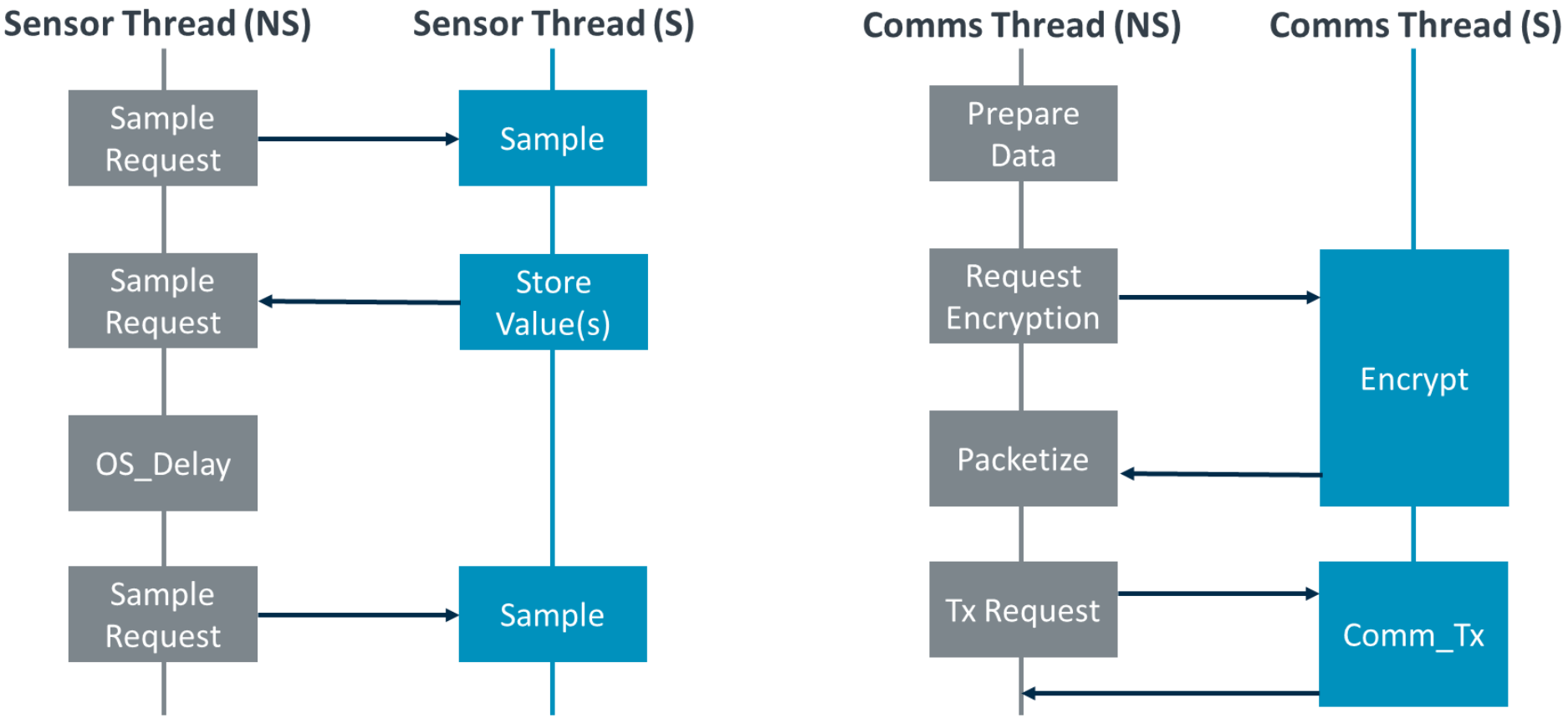
Security Through Isolation – Option #2

Software Component Organization



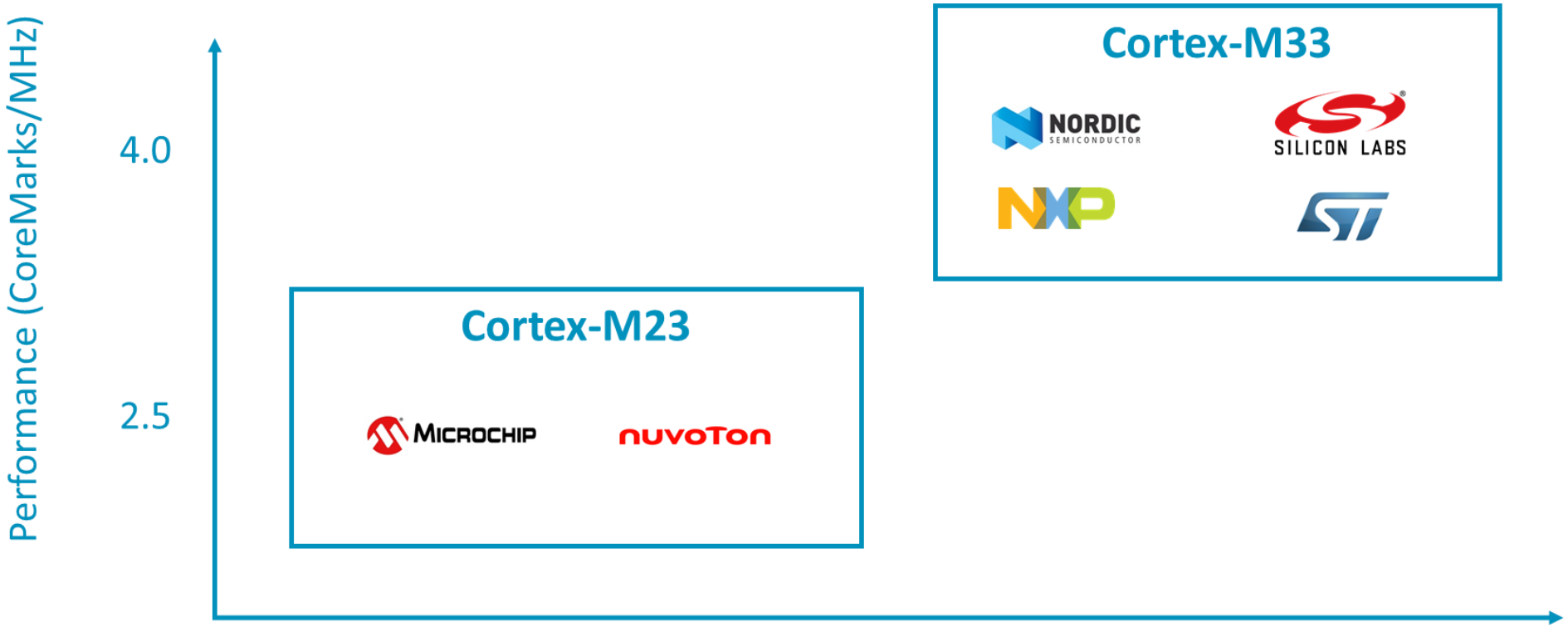
Security Through Isolation – Option #2

Thread design based on TrustZone hardware-enforced isolation



Security Through Isolation – Option #2

Microcontrollers with TrustZone



Additional Resources

- Beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm
- Platform Security Architecture:
  - www.arm.com/psa
- Threat-based analysis method:
  - www.cypress.com/psoc6security

From www.beningo.com under

- Blog > CEC – Getting Started with Secure Software