TrustedIoT

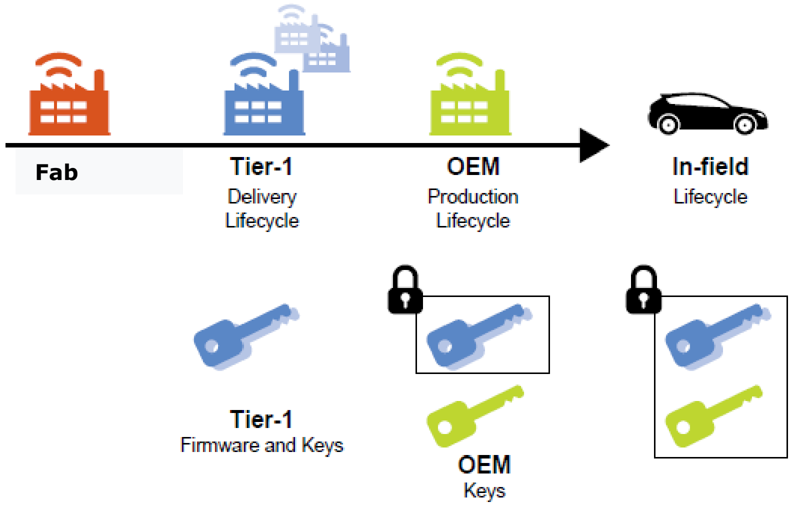
Introduction to Embedded Security

Bruno da Silva

EMBEDDED SECURITY

THE NEED

- Nearly every embedded system provides some level of security nowadays.
- Large demand from industry
  - Automotive market:
    - on-chip security subsystems
    - key usage policies
    - secure boot



  - Mobile phone market:
    - Android and iOS include support for hardware-backed key storage
    - Need for hardware-supported verified boot process as well as security services
  - Even in the Xbox One’s SoC!
    - featured a processing core, cryptographic engines, a random number generator and dedicated memories.

EMBEDDED SECURITY

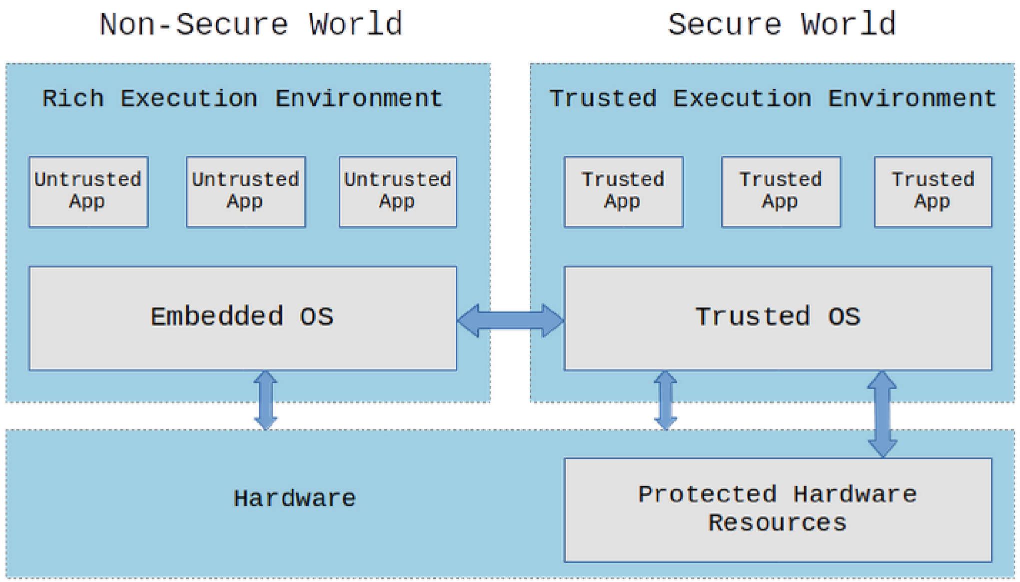
THE NEED

- Security is a big concern for embedded hardware, especially for connected devices such as IoT devices.
  - Distributed Denial of Service (DDoS) attacks
  - Unauthorized access to internal networks
  - …
- Modern attacks aim at compromising SW components
  - SW change at faster pace => hard to keep it validated and formally verified.
  - The SW part of the security system may also be a target for attacks
- There is a clear need for secure environments where compromised SW components do not compromise the system.

TRUSTED EXECUTION ENVIRONMENT

TEE

- Nowadays many embedded devices make use of Trusted Execution Environment (TEE)
- A TEE is an environment where the executed code and the accessed data is:
  - Isolated
  - Protected (confidentiality)
  - Integrity

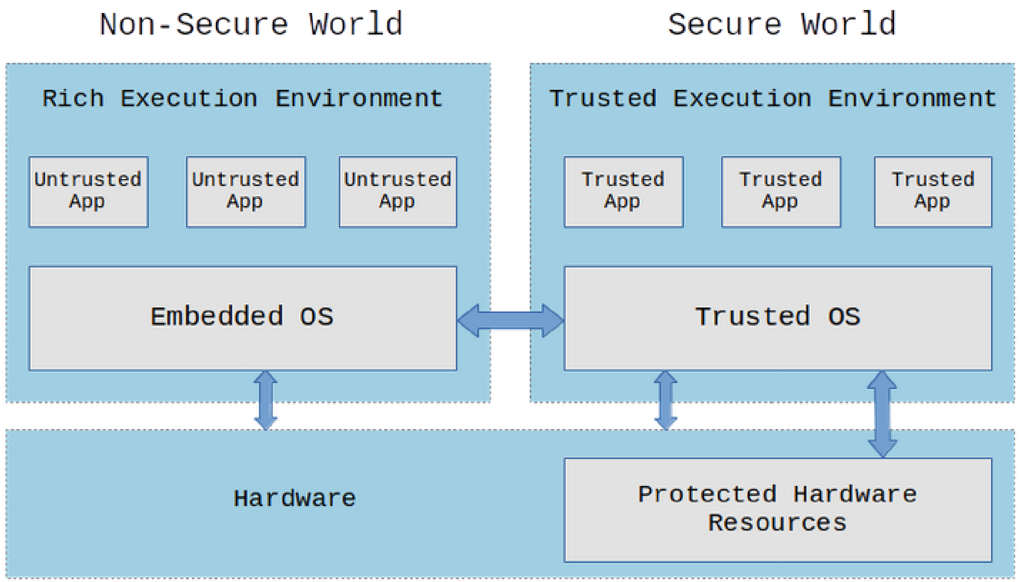


Source: https://embeddedbits.org/introduction-to-trusted-execution-environment-tee-arm-trustzone/

TRUSTED EXECUTION ENVIRONMENT

TEE

- Untrusted applications run on a Rich Execution Environment (REE) and trusted applications on a TEE.
- Trusted applications and associated data is completely isolated
  - From other trusted applications.
  - From untrusted OS and its applications.
- HW support is needed!



Source: https://embeddedbits.org/introduction-to-trusted-execution-environment-tee-arm-trustzone/

TRUSTED EXECUTION ENVIRONMENT

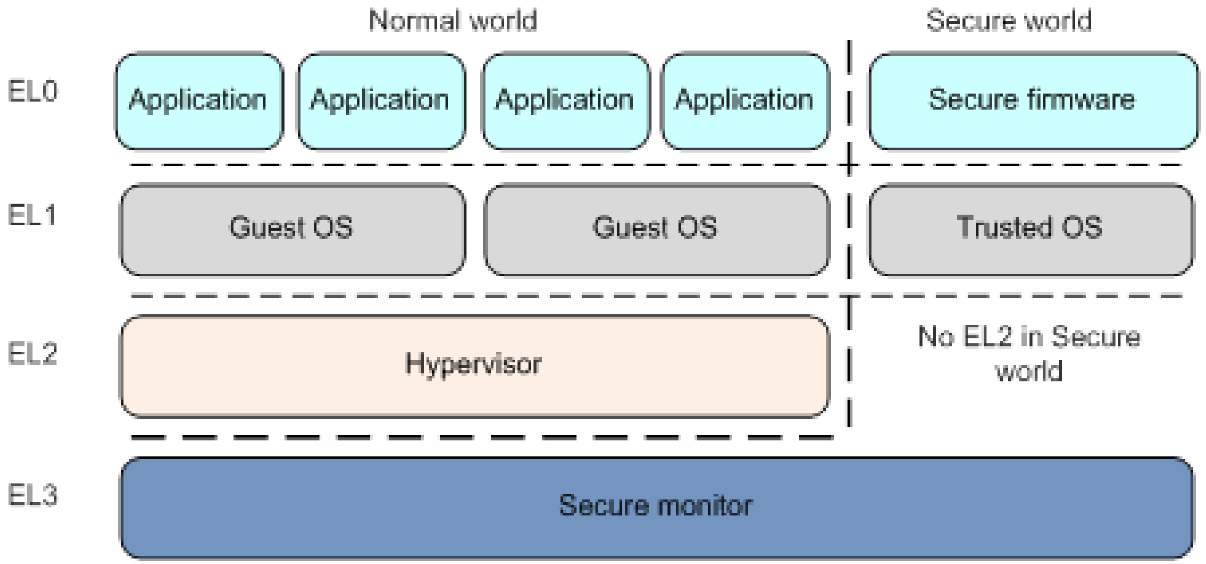
TEE

- Many processors nowadays provide support for TEE:
  - ARM TrustZone: specially used on embedded systems with ARM processors
    - Kinibi
    - Qualcomm Secure Execution Environment
    - iTRustee (Huawei)
  - RISC-V Multizone: a TEE for RISC-V processors
  - AMD Platform Security Processor (PSP)
  - Intel Software Guard Extensions (SGX)
  - Apple SEP
  - Google Titan M
  - …

TRUSTED EXECUTION ENVIRONMENT

ARM TRUSTZONE

- Differences between ARM architectures (Cortex-A or Cortex-M)
- Allow developers to use single core processors.
- Originally, 3 execution modes:
  - EL0 - User mode
  - EL1 - Kernel mode
  - EL2 - Hypervisor mode
- A new level of execution is inserted: EL3 - Secure Monitor



Sources:
https://developer.arm.com/documentation/102418/0101/TrustZone-in-the-processor?lang=en
PINTO, Sandro; SANTOS, Nuno. **Demystifying arm trustzone: A comprehensive survey**. ACM Computing Surveys (CSUR), 2019, vol. 51, no 6, p. 1-36.

HARDWARE

- There is a need of hardware support for:
  1. Hardware-based isolation
  2. A Root-of-Trust (RoT)
  3. A secure boot solution
  4. A secure bootloader

- Dedicated HW for securely store passwords, certificates, or encryption keys is needed by many secure systems
  - Trusted Platform Modules (TPMs)

Source: Guidelines on Hardware-Rooted Security in Mobile Devices

SECURE EMBEDDED SYSTEMS

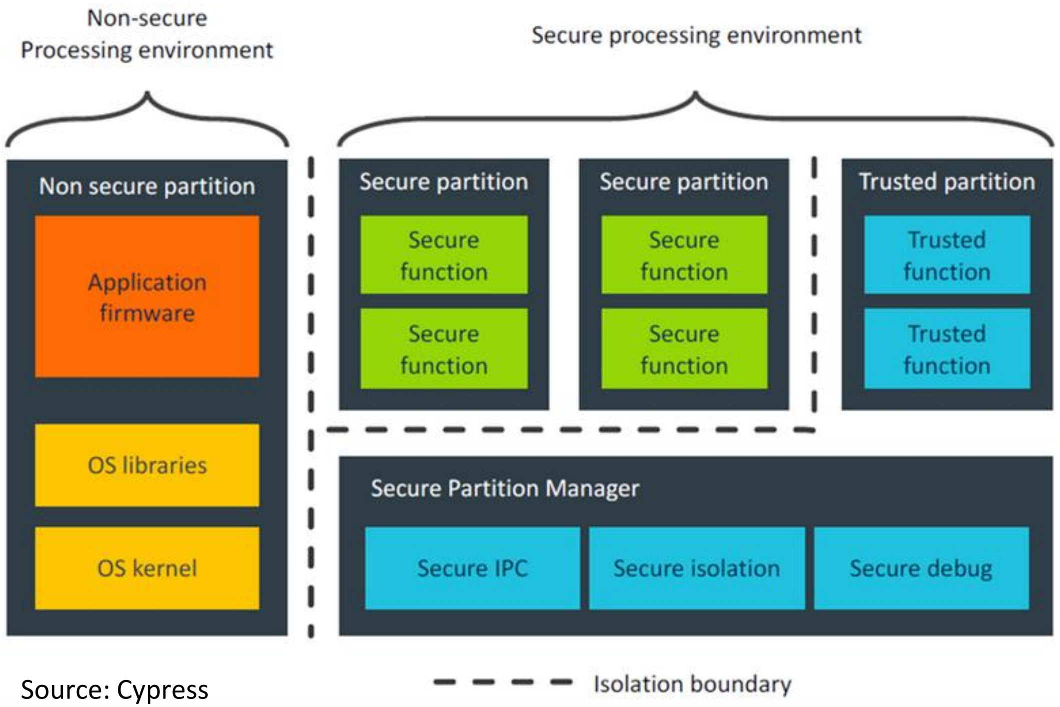
1.- HARDWARE-BASED ISOLATION

- Security begins with isolation
- Developers need to isolate their applications into different domains
  - Each with their own privileges
  - Access granted to only specific areas of memory
  - If a hacker is able to gain access to one area in memory, they won't be able to access other areas of memory.
- Problem on embedded: many applications allow software modules or components to access the entire memory map.
  - Too easy for hackers!
- Isolation in an embedded systems is done through **hardware isolation**

SECURE EMBEDDED SYSTEMS

1.- HARDWARE-BASED ISOLATION

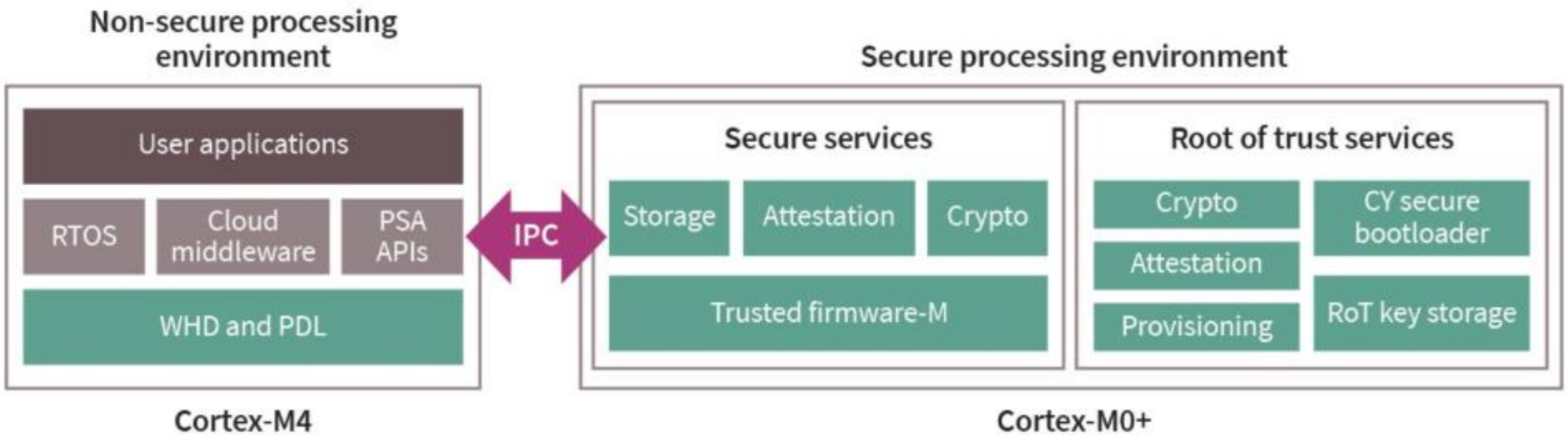
- Some MCUs support **hardware-based isolation**
- The application is executed on:
  - Secure Processing Environments (SPE)
    - Peripherals
    - Memory
    - Functions
  - Non-Secure Processing Environments (NSPE)
- Hardware-based isolation can be implemented on:
  - Processor level
  - Memory
  - Shared memory
  - Peripherals



Source: Cypress

1.- HARDWARE-BASED ISOLATION: PROCESSOR

- Example of a multicore approach:
  - One microcontroller core dedicated to the secure processing environment
  - One to the non-secure processing environment.



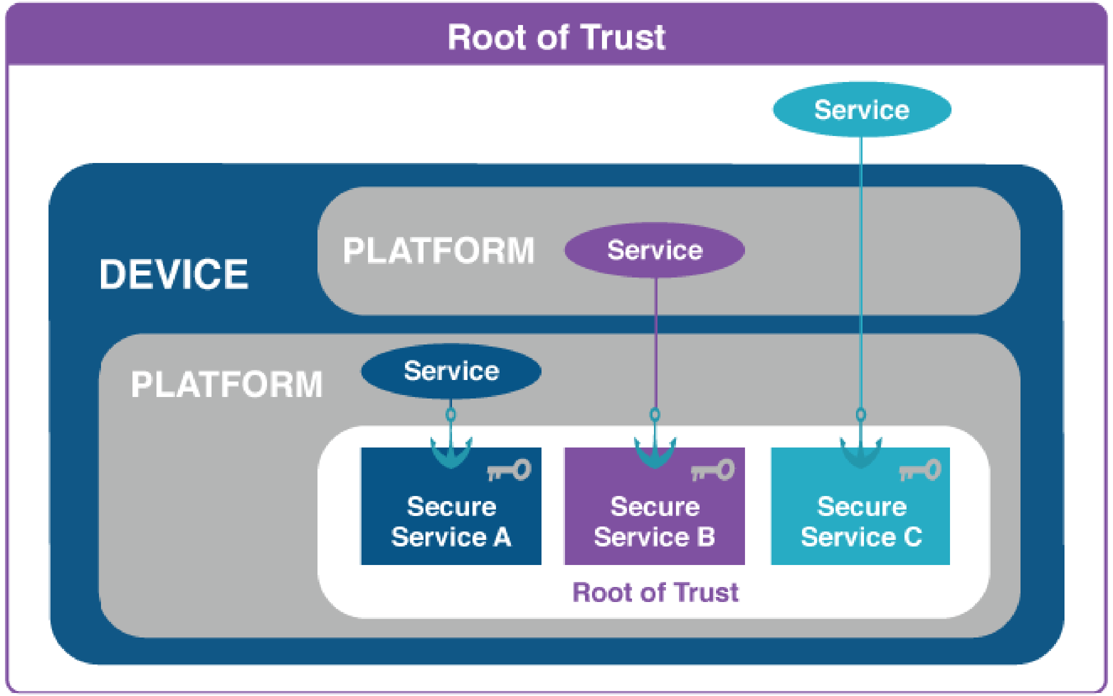
Isolated processing environments in PSOC64

Source: https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/psoc-64/

SECURE EMBEDDED SYSTEMS

2.- ROOT-OF-TRUST (ROT)

- The boot process is a critical moment
  - How to be sure there is no malware first running?
- A **Root-of-Trust** is an immutable, unclonable process or identity used as the first entity in a trust chain:
  - can successfully authenticate itself
  - facilitate secure operations on the system
    - security services (e.g. secure boot)
    - secure provisioning
    - attestation



- It means that:
  - MCU enable to embed significant information used by the Root-of-Trust that cannot change.
  - The system can be attested, using for instance its private key to sign operations.

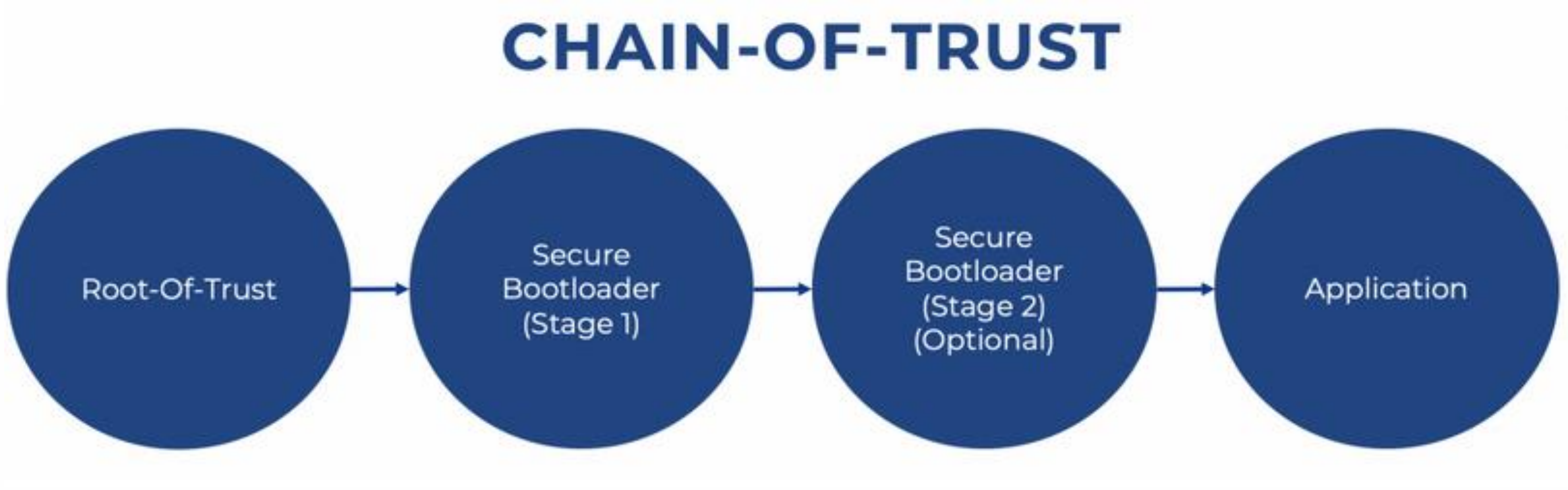
Source: Guidelines on Hardware-Rooted Security in Mobile Devices

Source: Secure Device Management for IoT

https://globalplatform.wpengine.com/wp-content/uploads/2018/05/GlobalPlatform-Chains-of-Trust-16May2018.pdf

2.- ROOT-OF-TRUST (ROT)

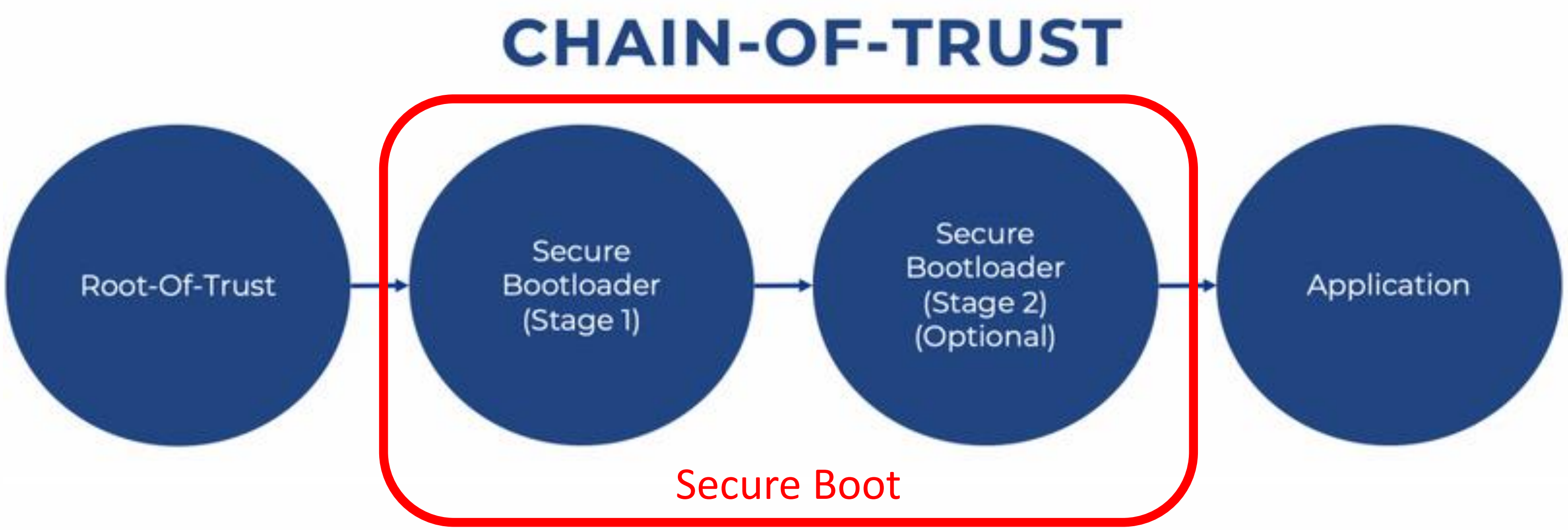
- Root-of-Trust serves as **the base trusted** software that then authenticates and validates the next piece of software loaded, which establishes a chain of trust:



- Root-of-Trust can also protect against activities such as:
  - Device cloning
  - Loading unauthorized firmware
  - Loading malware

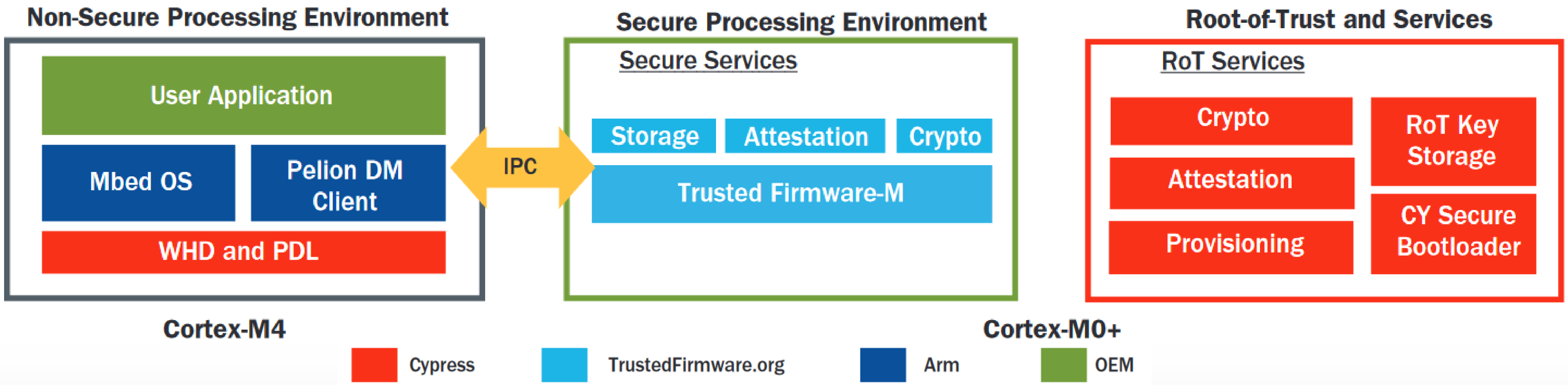
3.- SECURE BOOT

- The system needs to securely boot and validate all the code that it will be loading and executing.
- The Root-of-Trust and hardware-based isolation can be used for that purpose.



3.- SECURE BOOT

- Example: A multicore MCU
  - one core is dedicated to security features and code execution
  - other core is allocated to application-rich code. In this setup, interprocessor communication (IPC) is used to communicate between the cores.



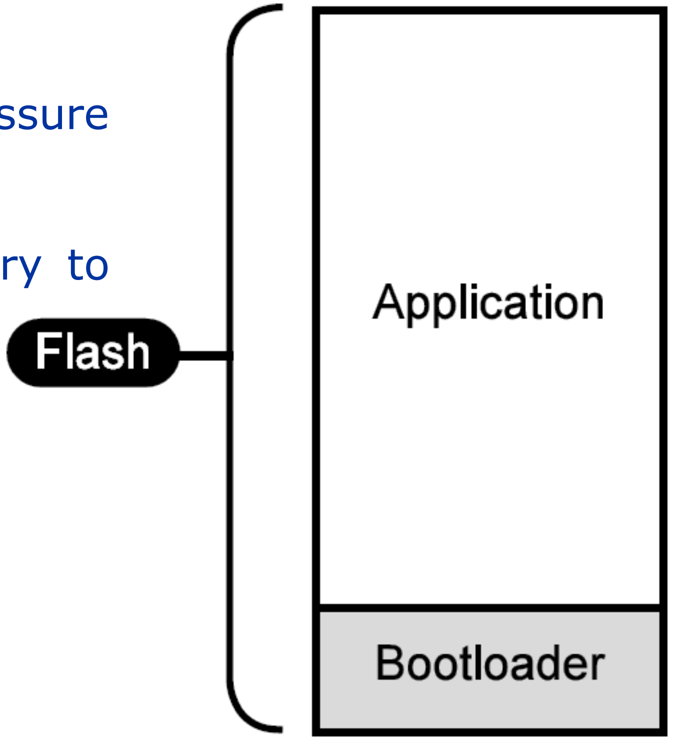
Source: [Secure Device Management for IoT]

3.- SECURE BOOT

- Secure boot requires a developer to carefully think through the boot sequence and develop a Chain-of-Trust that originates at the Root-of-Trust.
- As the system boots, each flash image and code are verified, and only afterward is it allowed to execute on the system.
- The system can either halt the boot sequence or maybe even revert to an earlier known working version of code in case a problem is detected.
- In order to revert the code, or update it to a new version, that requires the system to have a secure bootloader.

4.- SECURE BOOTLOADERS

- From time to time a firmware upgrade is needed.
- The bootloader is a small piece of code which provides the ability to download updates and replace old firmware.
- It is the task of the bootloader to verify the new firmware and to assure a secure boot.
- A must be as small as possible since it always resides in memory to make it possible for the device to be upgraded at anytime.

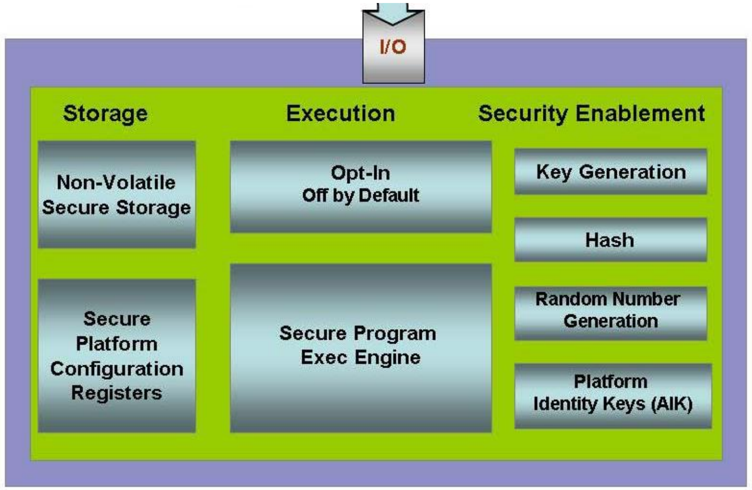


NxP Secure Bootloader Implementation: https://www.nxp.com/docs/en/application-note/AN4605.pdf
Application Note Atmel, Atmel AT02333: Safe and Secure Bootloader Implementation for SAM3/4

SECURE EMBEDDED SYSTEMS

TRUSTED PLATFORM MODULE

- Trusted Platform Module (TPM) is a MCU that can securely store passwords, certificates, or encryption keys.*



- Example: Infineon Optiga (a standalone security coprocessor)
  - Secure TPM 2.0 compliant microcontroller for automotive applications
    - Advanced cryptographic algorithms (RSA-2048, ECC-256, SHA-256)
    - enhanced security features (shielding, security sensors…) implemented in hardware
    - Can be integrated as a co-processor: it offers a SPI interface with TPM support.

Source:
- Infineon OPTIGA
- Infineon TPM

*https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/

SECURE EMBEDDED SYSTEMS

SHORTCOMINGS

- ARM TrustZone
  - Absence of secure storage: TrustZone in itself does not provide any way to store secret data.
  - ARM peripheral bus and other system buses but it fails to guarantee the secure transmission of data on its peripheral.
  - It does not specify secure entropy source for cryptography.
  - Security is provided through virtualization (virtual cores)
    - TrustZone architecture is **only** software based
    - It does not contain the security advantages of a dedicated hardware TPM chip
  - Overall, ARM TrustZone does not comply with the standards of NIST and MTM.

- Any flaw in the Root of Trust and the whole system is compromised

- Everyday security flaws are exploited:
  - Breaking Secure Bootloaders
  - Microarchitectural attacks to TEE

Source:

- ASHRAF, Naveeda, et al. "*Analytical study of hardware-rooted security standards and their implementation techniques in mobile.* "Telecommunication Systems, 2020, vol. 74, no 3, p. 379-403.
- https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/a-technical-report-on-tee-and-arm-trustzone

Questions? ? ?