ESPRESSIF DevCon24

Trusted Execution Environment (TEE) on ESP32-C6

Enhanced Security for RISC-V SoCs

Sachin Billore & Laukik Hase

CONTENTS

Speaker Intro

We are a part of the Security Team, and we work on building and maintaining the Security features supported in ESP-IDF and elsewhere.

Among other things, we work on support for cryptographic peripherals such as AES, RSA, Key Manager, flash encryption and secure boot.

Sachin Billore

Senior Software Engineer

Laukik Hase

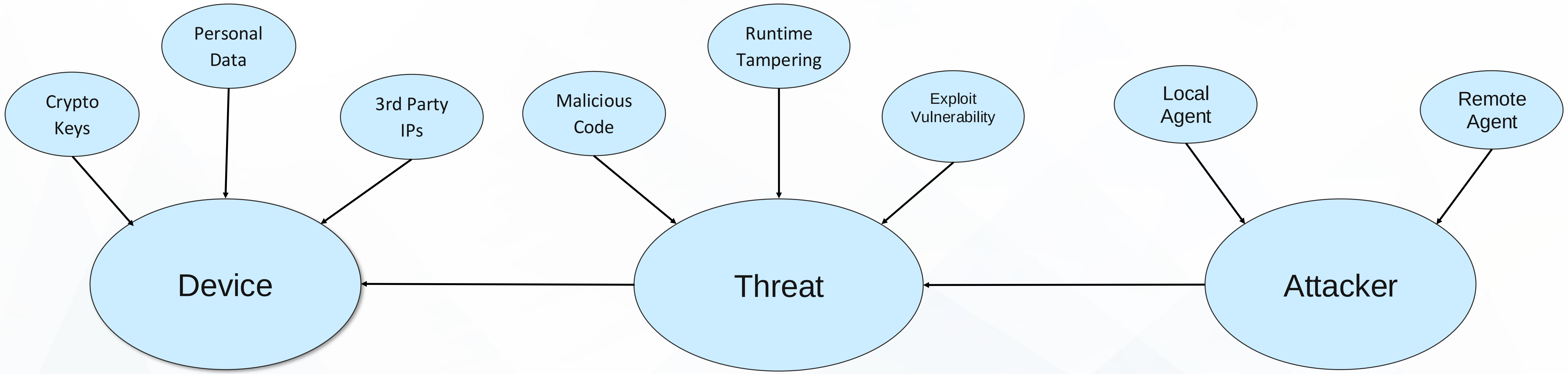
Embedded Software Engineer

01 Trusted Execution Environment (TEE)

Introduction: The Why and the What

Introduction: Why TEE?

A device's Security Model is derived from its Threat Model



Introduction: Why TEE?

ESPRESSIF DevCon24

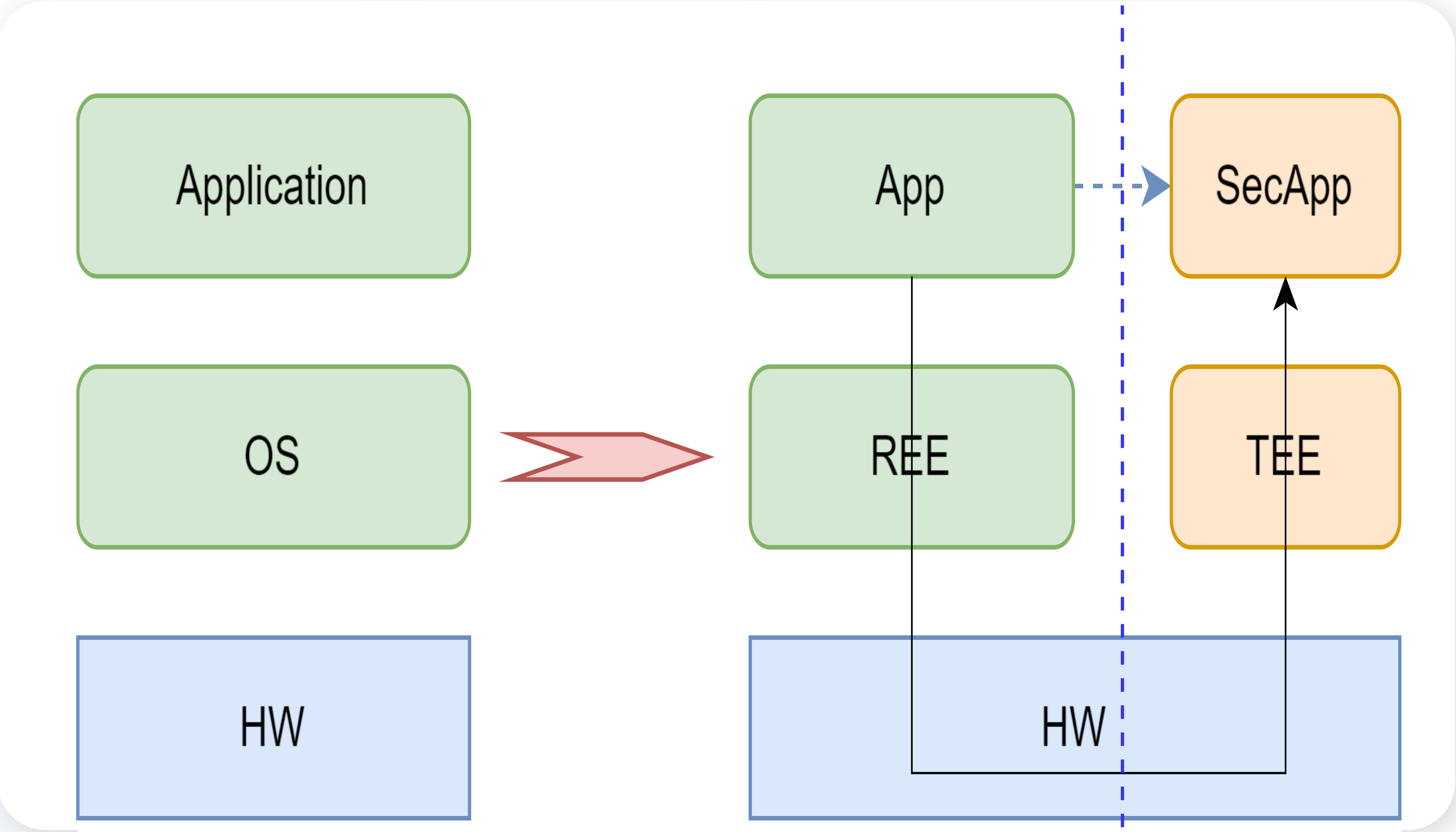
> A device's Security Model is derived from its Threat Model

- Threat Mitigation
  - Reduce the attack surface by isolating security-sensitive system assets from the rest of the system
  - Confidentiality of the data, Authenticity and Integrity of execution environment
  - No primary focus on invasive or non-invasive hardware attacks

Introduction - What is TEE

H/W-Defined, S/W-Implemented Isolation Mechanism

- Isolates security-sensitive hardware and software resources from the rest of the system, forming a parallel execution environment (TEE)
- TEE provides runtime security for these hardware and software resources, protecting them from the rest of the system
- Establishes a clear and well-defined interface between the two environments

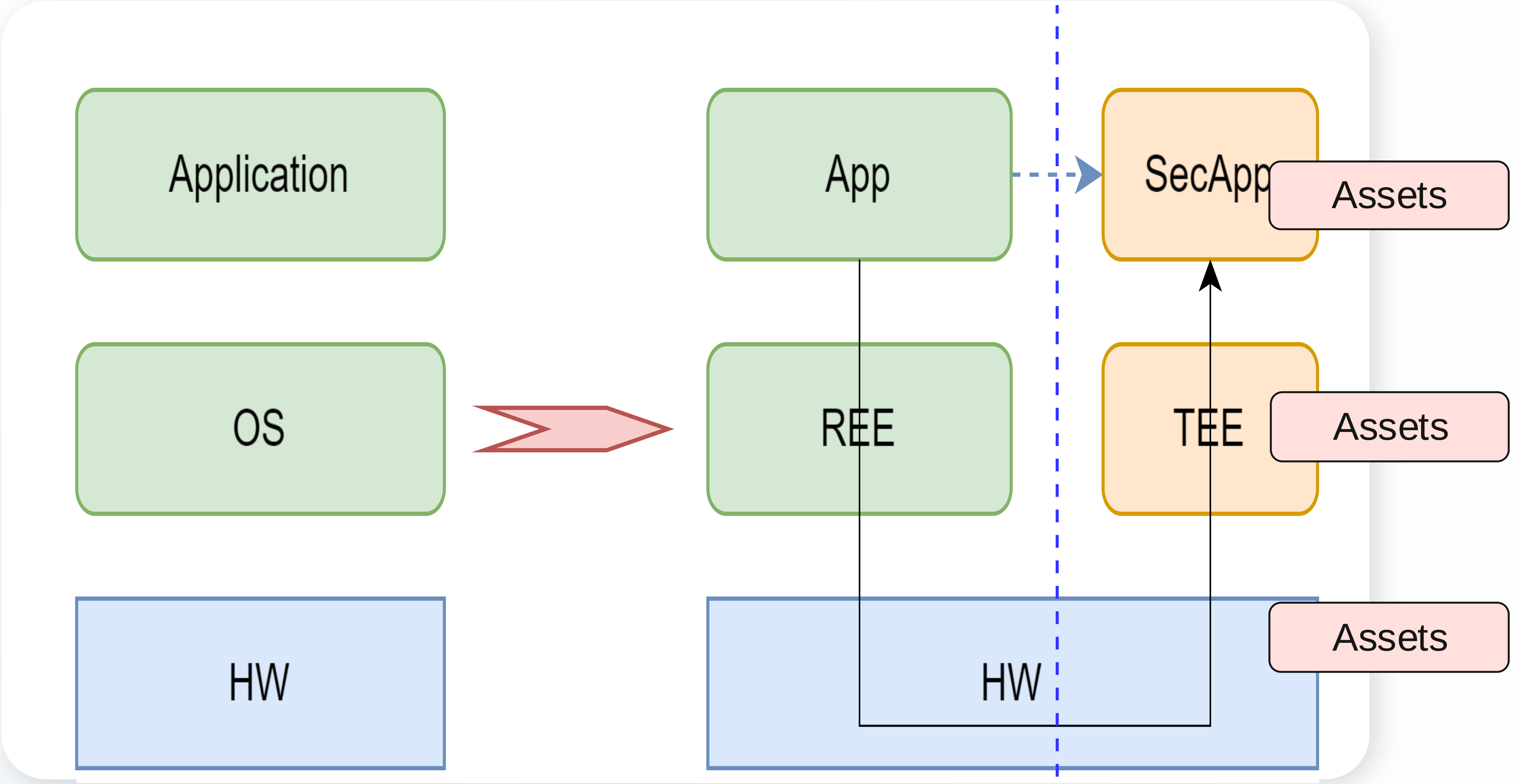


Introduction - What is TEE

Real world example

Assets : Cryptographic key, Secure Application, Crypto HW

- Crypto key generate or provision
- Use in challenge response protocol
- Secure communication

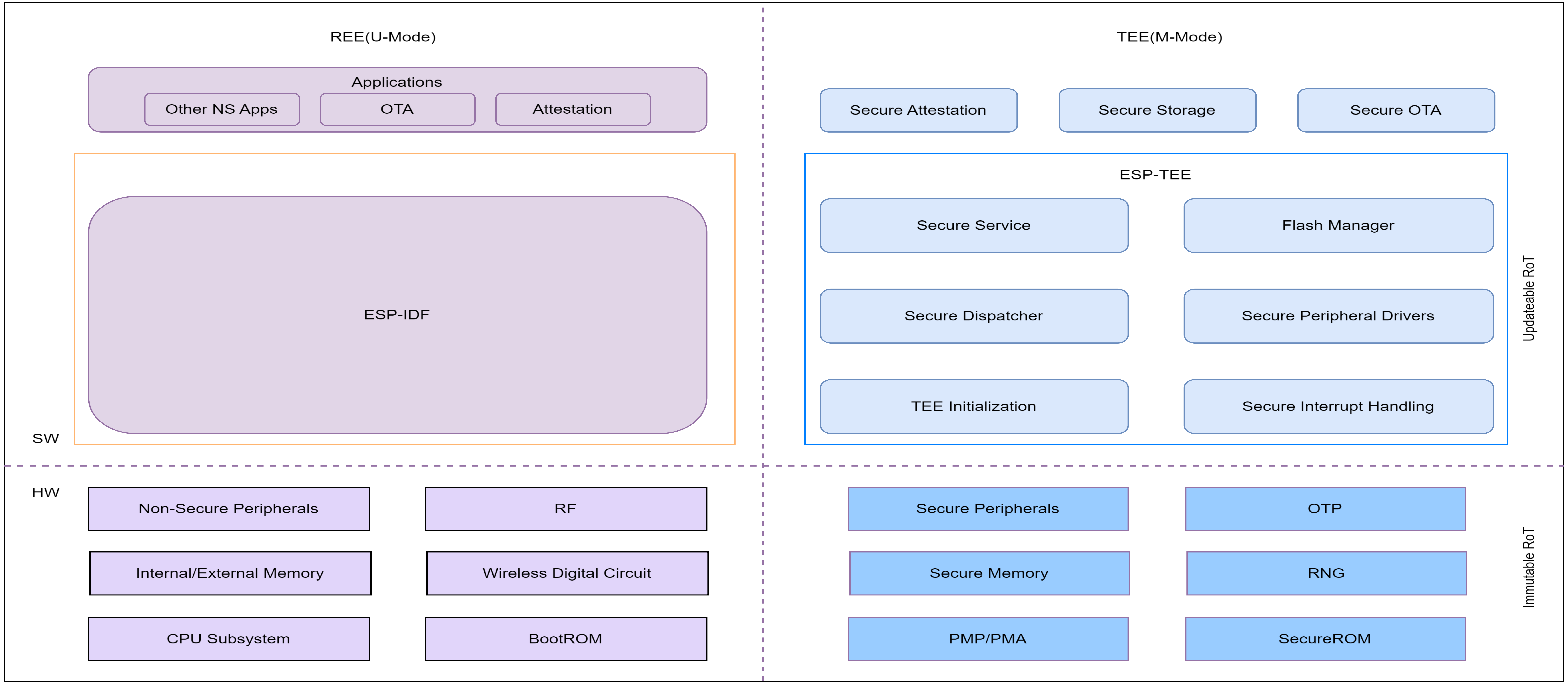


- Enhance system security, implementation completely defined by system requirement.
- Requires for various certification and regulation.

02 ESP-TEE: Architecture

TEE on the ESP32-C6 SoCs

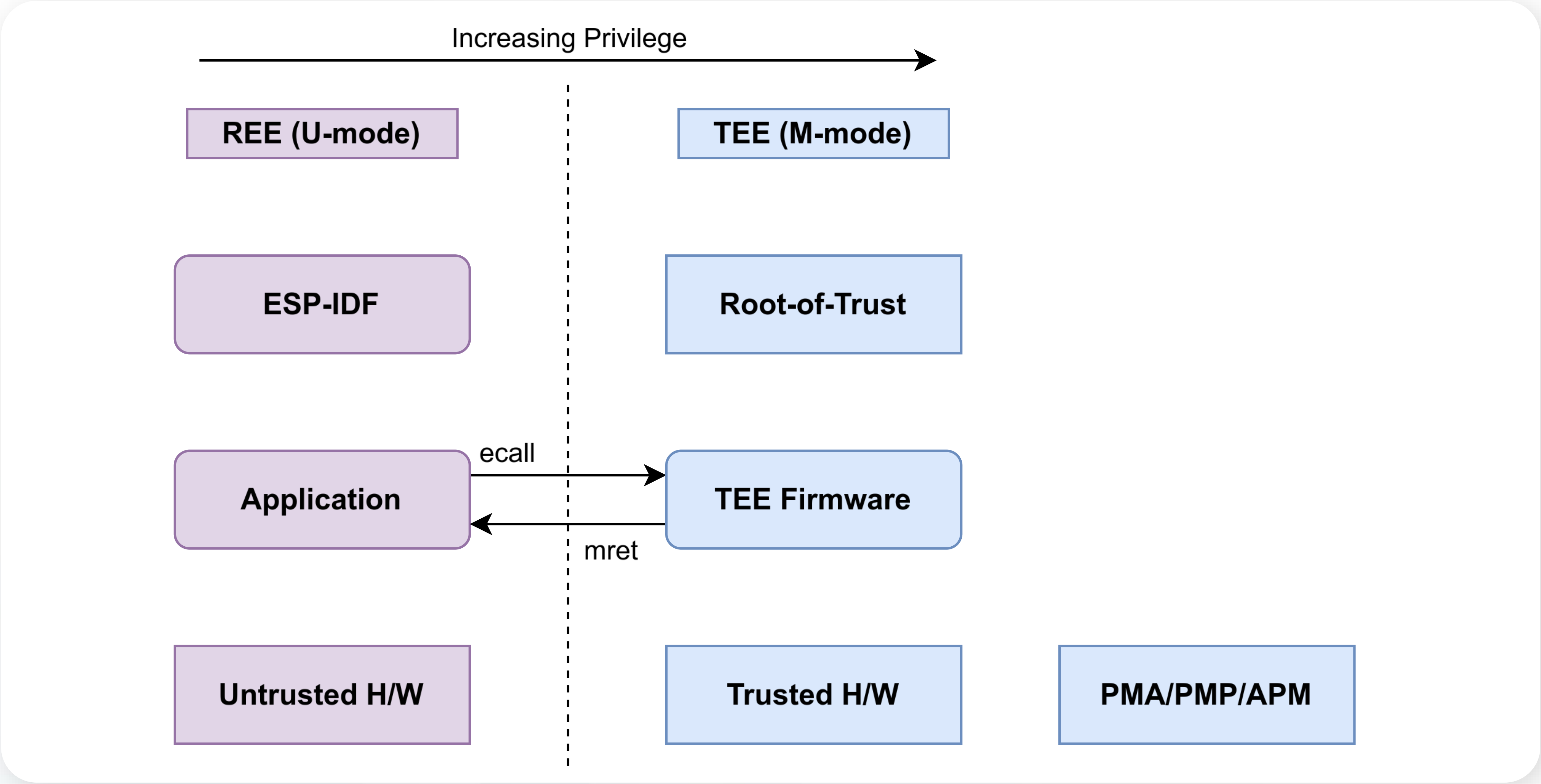
TEE on ESP SoCs: ESP-TEE



ESP-TEE: Architecture

Hardware Components (Immutable Root-of-Trust)

- ESP-TEE uses RISC-V ISA security features to achieve TEE/REE isolation
- Requires RISC-V M and U Privilege modes
  - M-Mode: Higher privilege mode host TEE firmware
  - U-Mode: Lower privilege mode host ESP-IDF and untrusted HW
- PMP/PMA to manage physical memory access
  - Default M –Mode have full access and U-Mode none
  - M-Mode configure PMP and PMA for selective memory access by U-Mode
- PMS: APM+TEE(hardware peripheral)
  - Complement PMP/PMA to manage the physical memory
- Root Of Trust: ROM, eFuse, Crypto HW, Isolation HW (APM), Secure memory

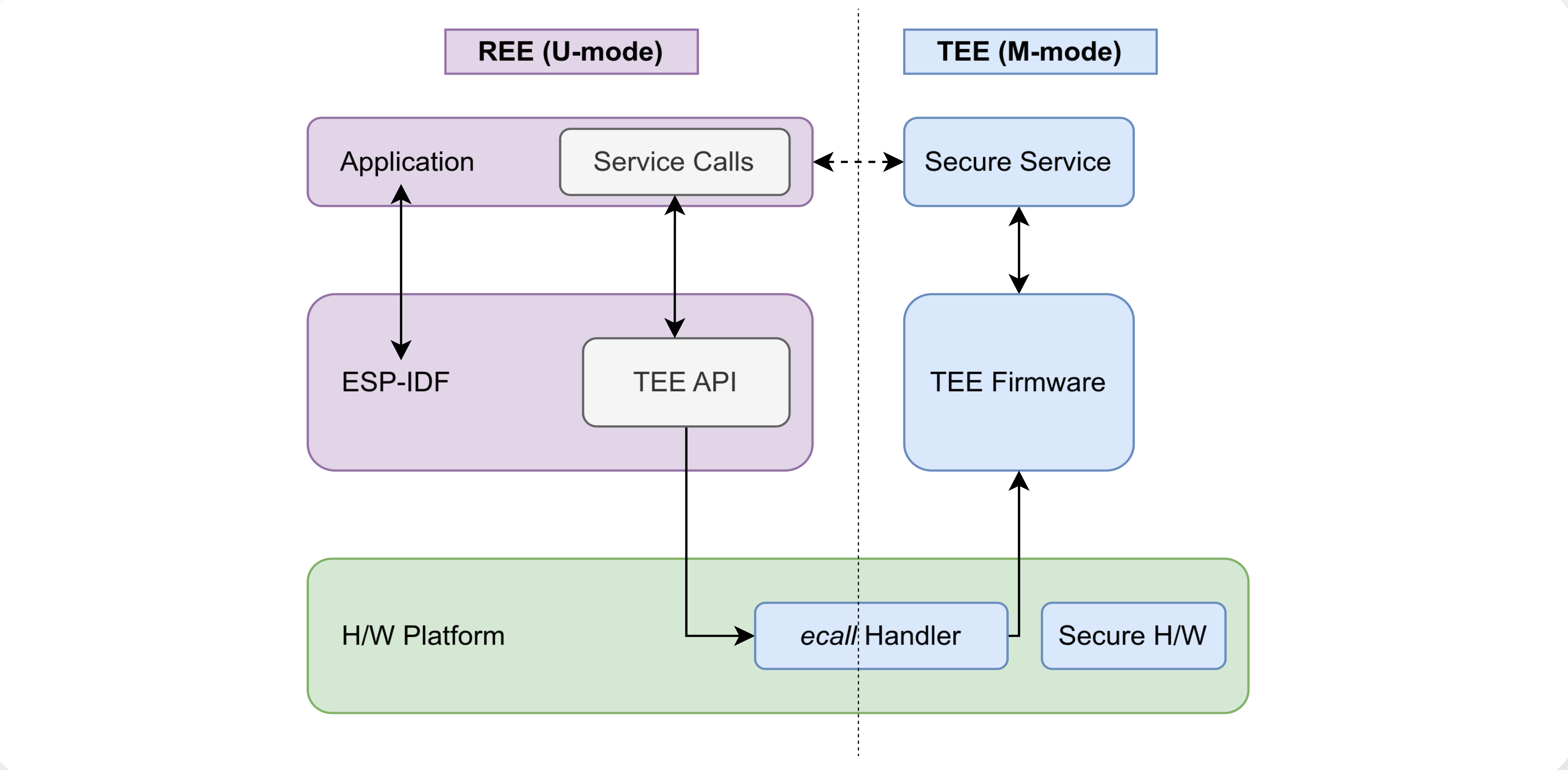


ESP-TEE: Architecture

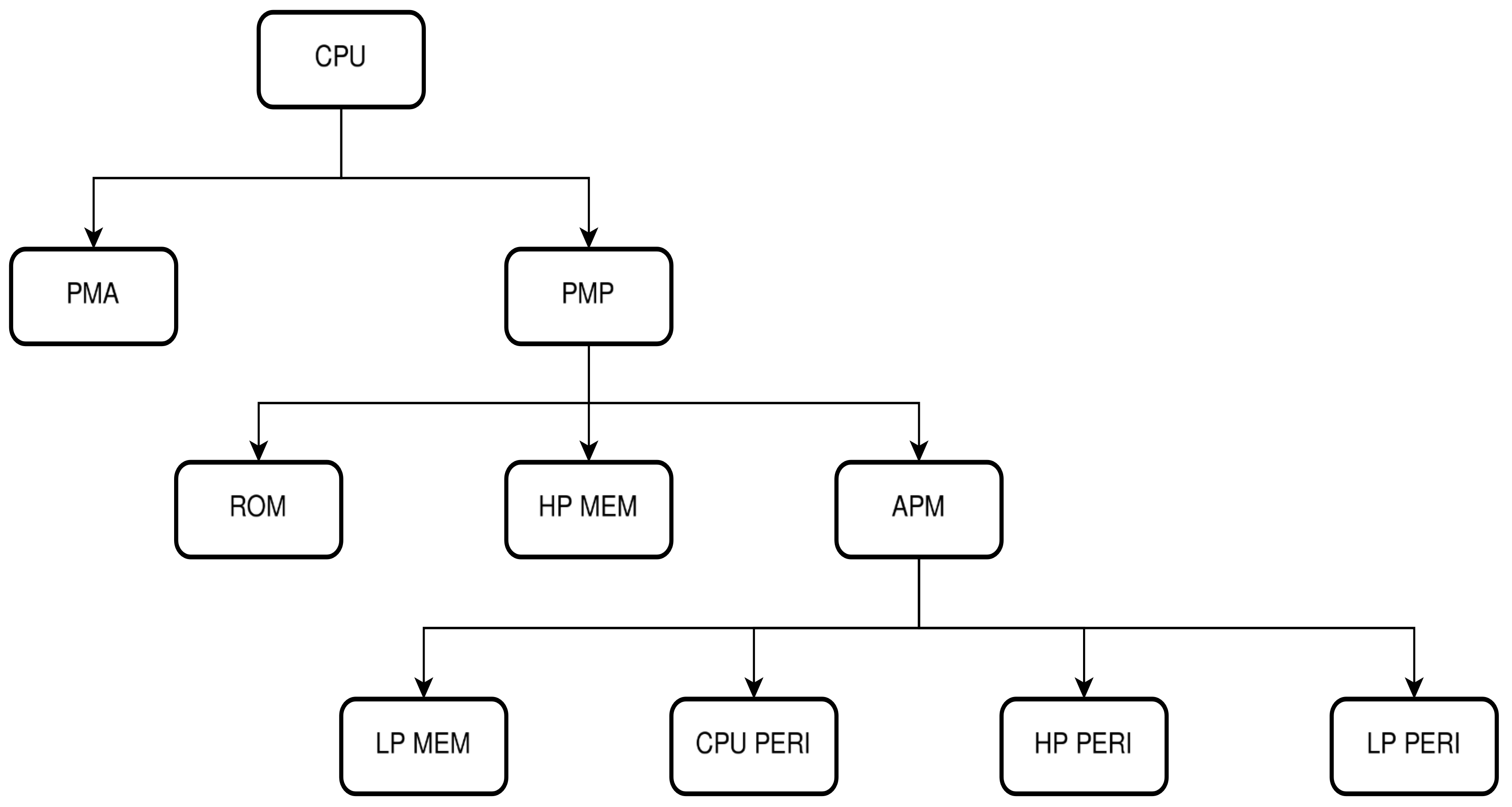
ESPRESSIF DevCon24

Software Components (Updatable Root-of-Trust)

- TEE firmware include various internal components responsible for -
  - Secure configuration to achieve isolation
  - TEE initialization for secure services, interrupt and exception handling
- Includes Basic Secure Services, Crypto API and other secure HW abstraction which are used to implements use case based Secure Services (Attestation, OTA, Secure Storage etc.)
- Provide TEE interface API to REE to invoke secure service into TEE
- Uses RISC-V Environment call (ecall) to transition between TEE and REE



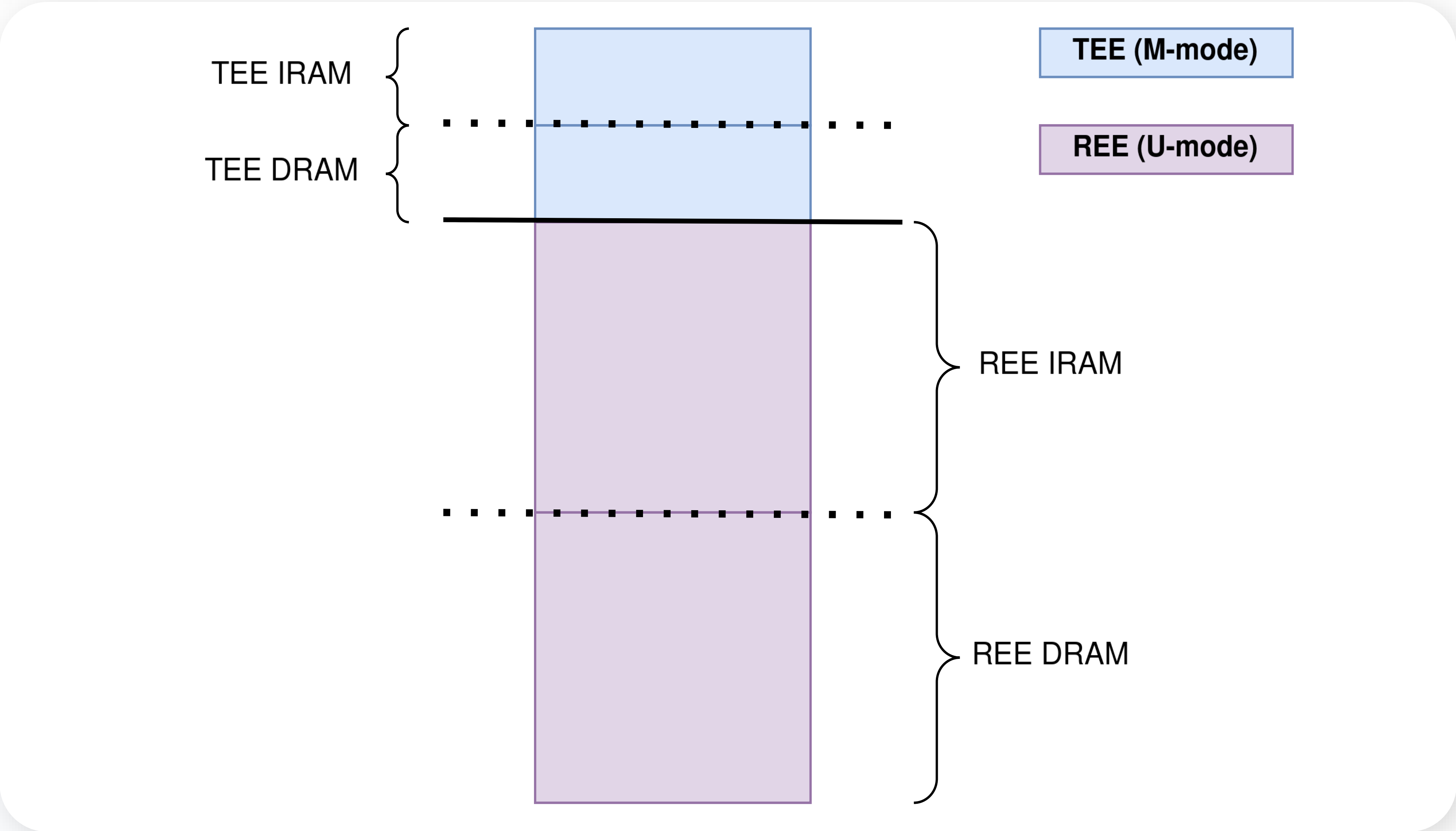
ESP-TEE: Architecture



Resource Isolation

- Isolation between TEE and REE is established during TEE initialization.
- RISC-V primitives like PMP, PMA, and proprietary hardware peripheral APM are used for this isolation.
- PMP, PMA configuration and the APM peripheral (protected by PMP) are modifiable only in M-Mode.
- All configurations are locked and remain so until the next reset.

ESP-TEE: Architecture

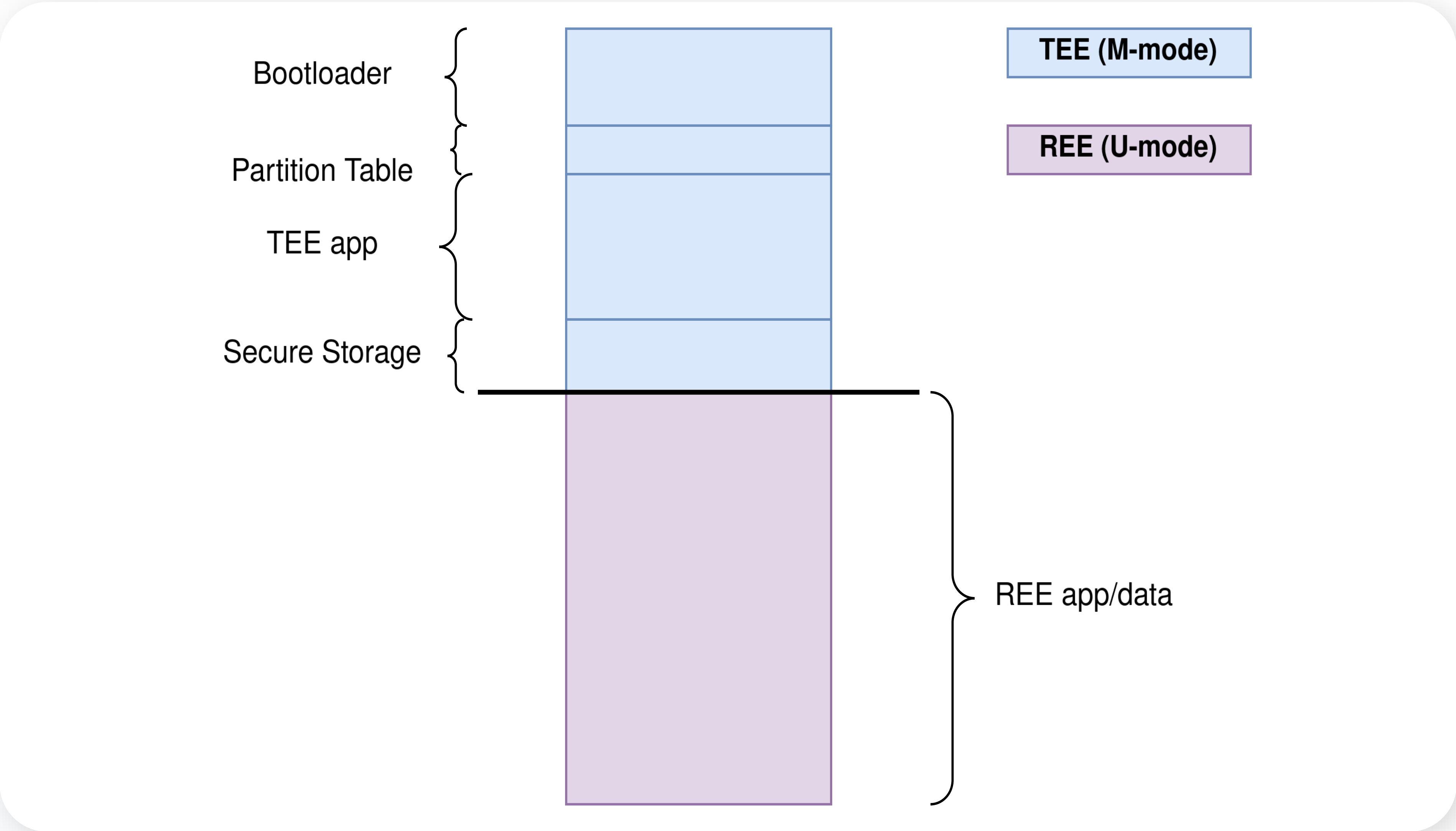


Internal SRAM

Internal Memory (SRAM)

- A memory region is reserved for the TEE starting from the beginning of the HP SRAM
- Permission management is done by PMP for all regions (TEE I/DRAM and REE I/DRAM)
- TEE region is further divided into IRAM (text – RX) and DRAM (data – RW) sections using PMA

ESP-TEE: Architecture



External Flash

External Memory (Flash)

- Designated partitions in the external flash are reserved for the TEE – XIP execution, secure storage, and OTA data
- Access through cache protected using PMP
- Physical Access protected using APM

Peripherals

- AES, SHA, eFuse and Interrupt Controller are protected from REE access using APM

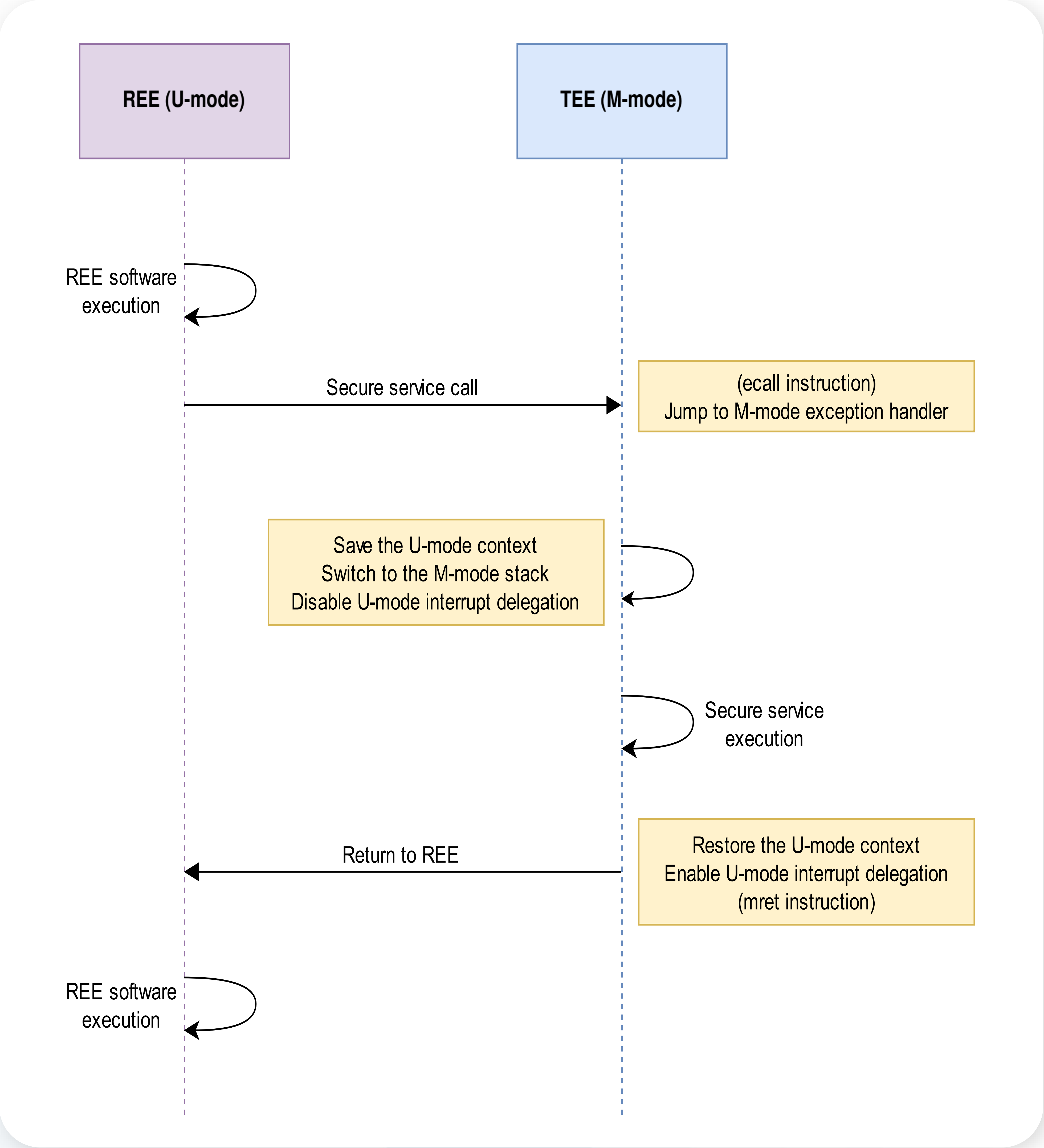
03

ESP-TEE: Internals

TEE Firmware and Secure Services

ESP-TEE: Secure Services

- Secure Service Call Table
  - Indexed list of all secure services provided by TEE – shared with the REE
- Service Dispatcher
  - Entry point to the TEE for any secure service call from REE
  - Parses the input arguments and passes execution control to the appropriate service



ESP-TEE: Secure Services

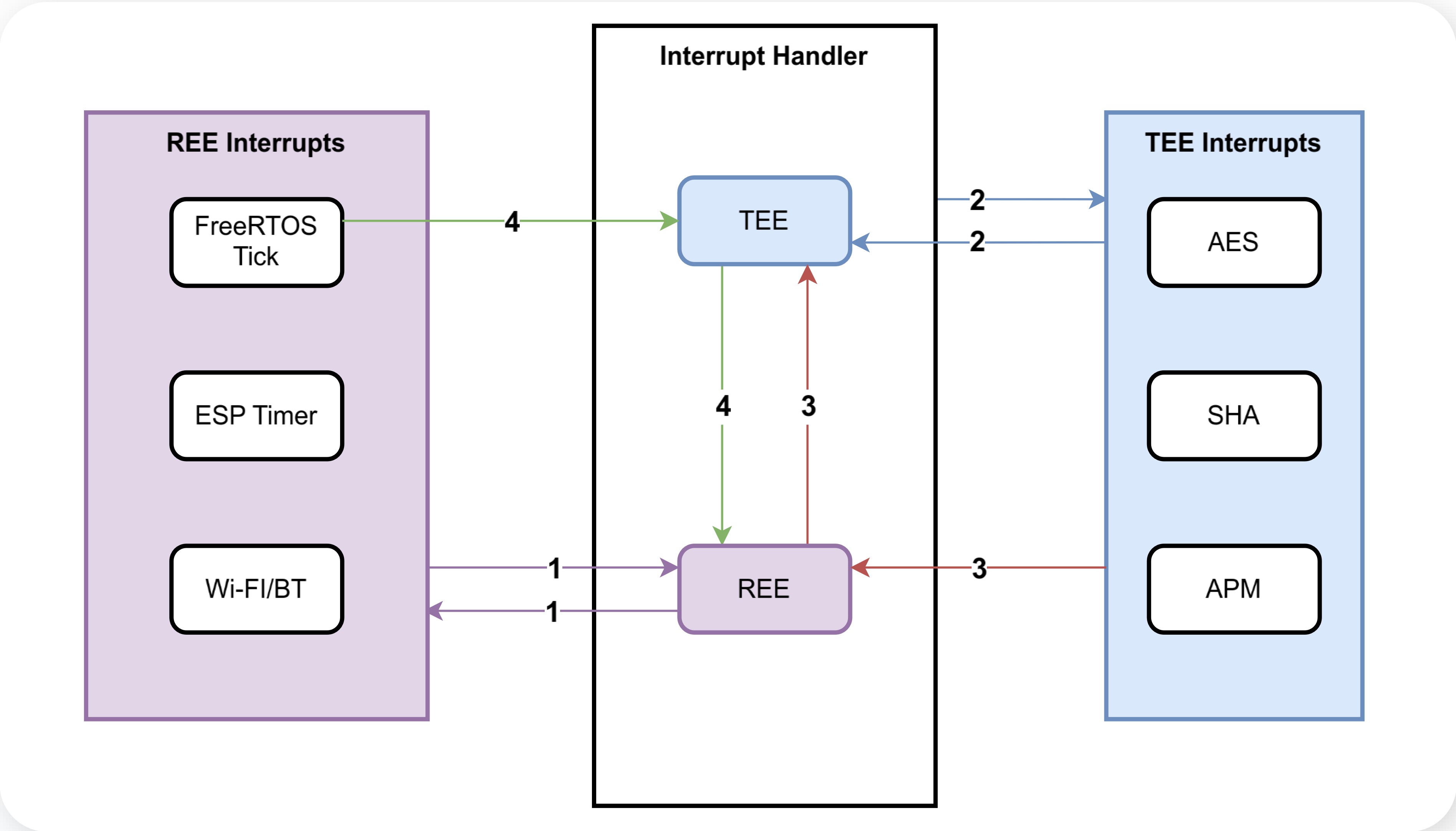
ESP-IDF Secure Services

- Basic secure services included in the TEE firmware, providing routine functionalities to the REE
- E.g., CryptoAPI services (AES/SHA), interrupt matrix and eFuse access

Custom Secure Services

- Optional secure services included as configurable TEE features
- Users can also define their own services as required
- E.g., Secure Storage, Secure OTA, and Attestation (included by default with ESP-TEE)

ESP-TEE: Interrupts

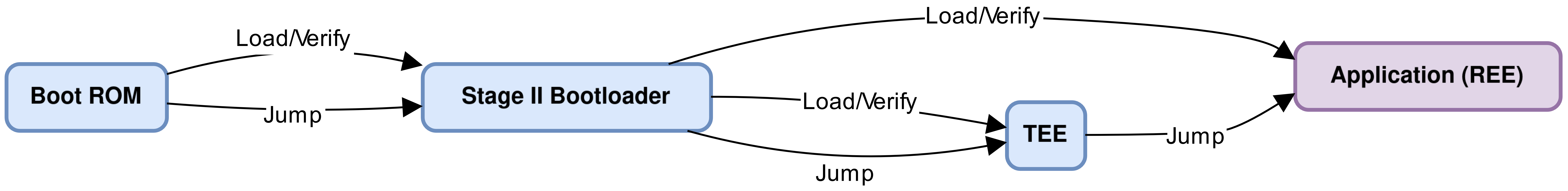


Interrupt Handling

- Separate vector tables for TEE (`mtvec`) and REE (`utvec`)
- Secure interrupt sources are protected from REE
- REE interrupts are delegated to REE when CPU is in REE (`mideleg`)
- Four interrupt handling scenarios
    1. REE interrupts in REE
    2. TEE interrupts in TEE
    3. TEE interrupts in REE
    4. REE interrupts in TEE

ESP-TEE: Boot Sequence

- After a reset, the Boot ROM loads the stage II bootloader, which then loads the TEE and application images from the boot device.
- Control is passed to the TEE, which sets up memory, interrupts, and peripheral access, before switching to the REE in U-mode to run the application.

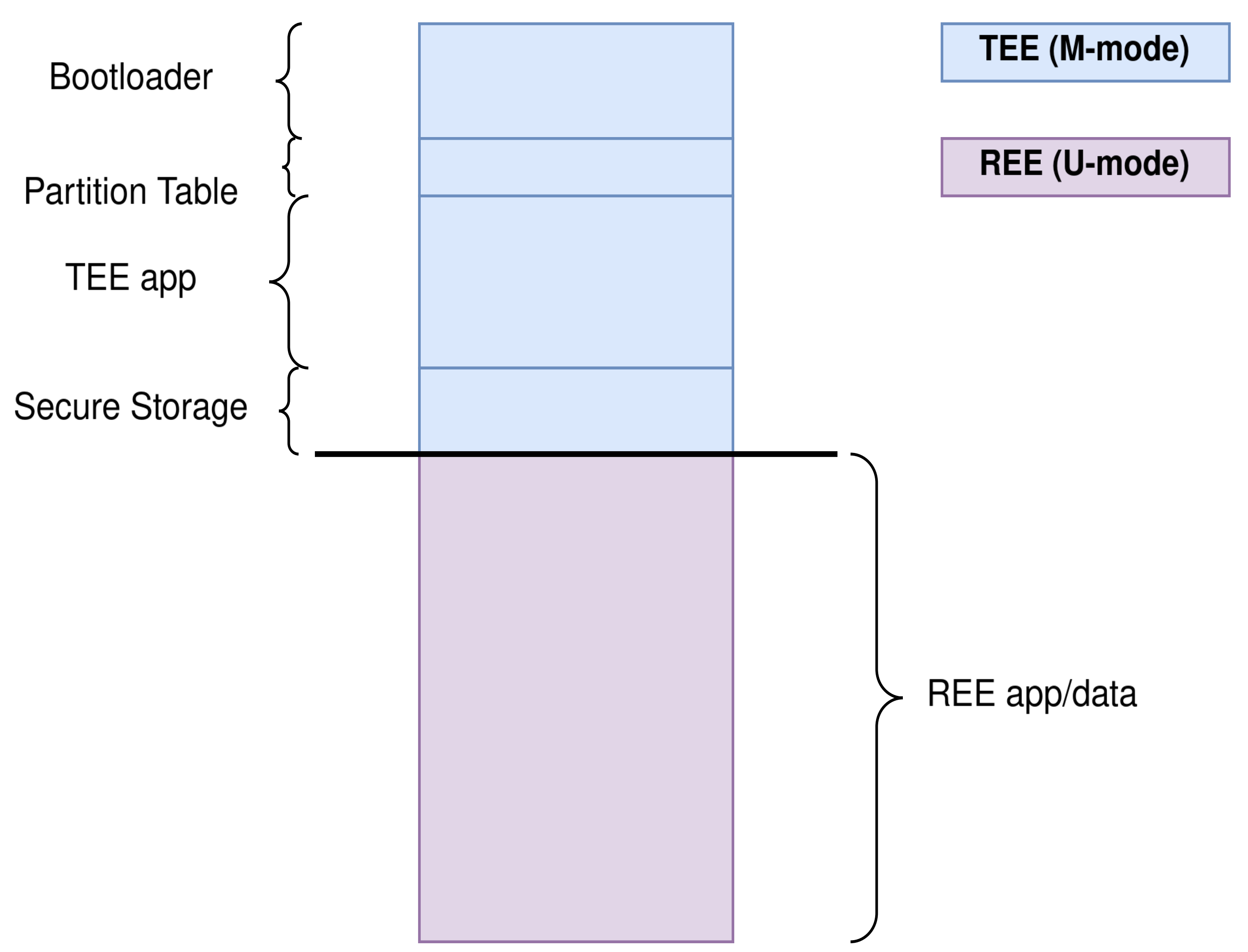


04

ESP-TEE: Use Cases

Practical applications of TEE

ESP-TEE: Secure Storage



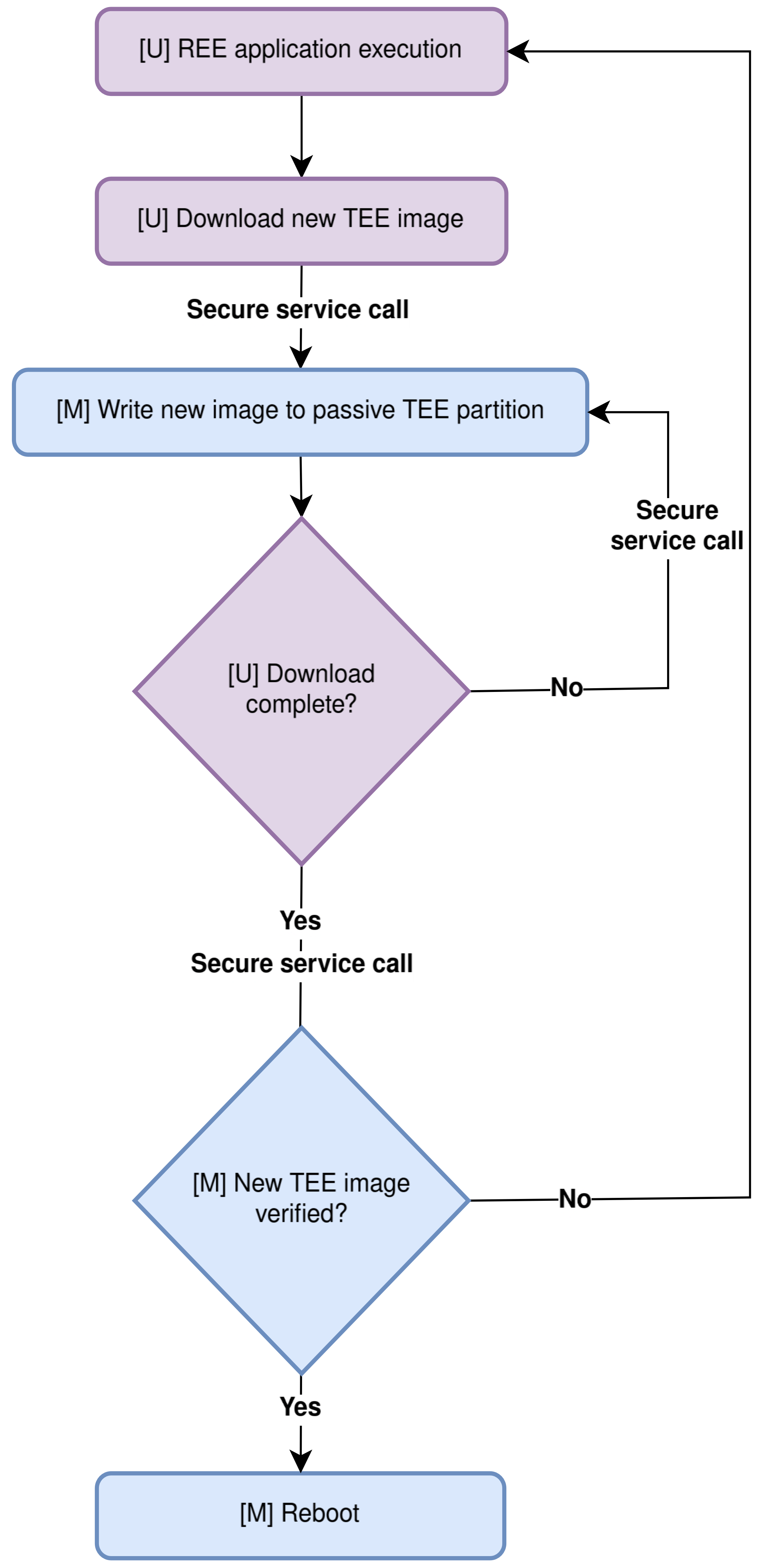
Persistent Storage for Sensitive Information

- Utilizes a dedicated external flash partition
- Encrypted with AES-256-GCM scheme with device-specific key (eFuse)
- Data and encryption key are inaccessible to the REE

Key Management

- Supports storage of ECDSA secp256r1 key pairs (14 slots)
- Interface to the REE for securely signing messages in the TEE using the above key material

ESP-TEE: OTA Updates



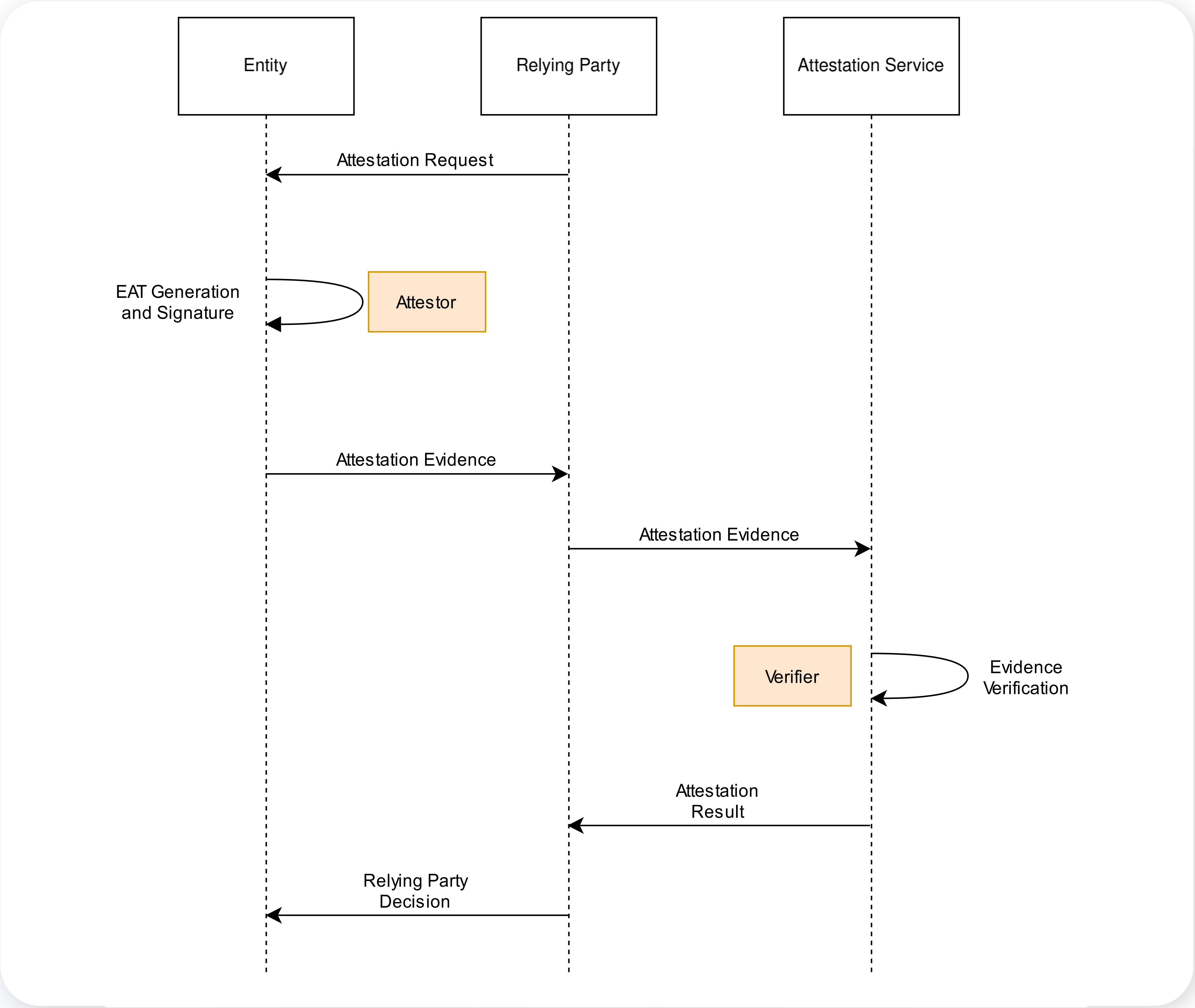
Securely manage TEE OTA updates

- Two TEE OTA app and one TEE OTA data partition entries are required
- Support for TEE app rollback
- Compatible with Secure Boot

ESP-TEE: Attestation

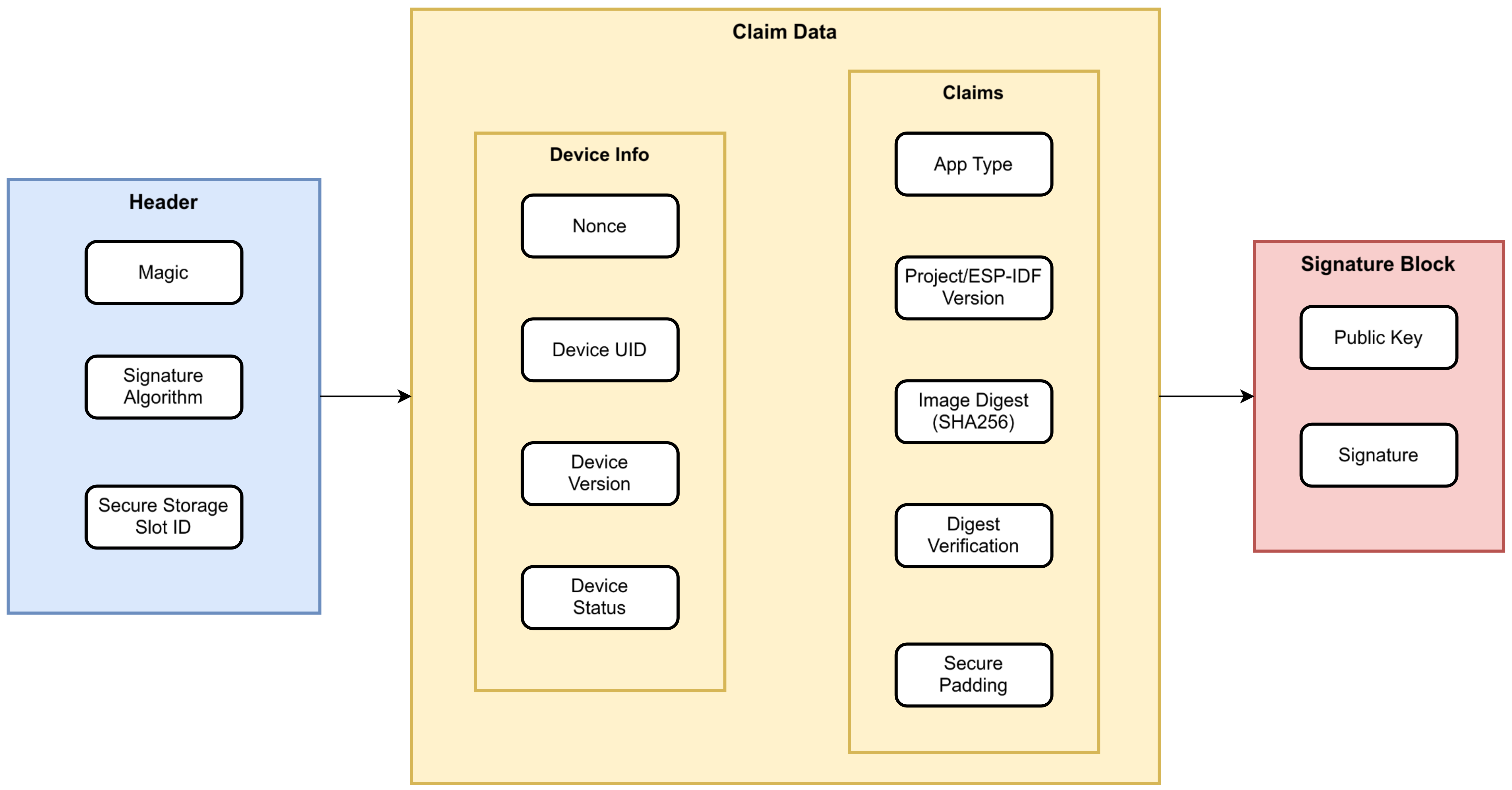
- Mechanism enabling a device to make claims about its identity and security status
- Useful for manufacturers and cloud service providers for device assurance and trust

- Entity Attestation Token (EAT)
  - Cryptographically signed small data blob containing device claims, defined in JSON format
  - Claims: Device ID, Software version, Hardware version, etc.



ESP-TEE: Attestation

- REE interface to securely generate and return an EAT from TEE
- EAT signed using an ECDSA secp256r1 key-pair from the given TEE secure storage slot
- Supported Claims for all firmware images (Bootloader, active TEE and REE app)
  - Project and ESP-IDF version
  - Image Digest (SHA256)
  - Public key corresponding to the signing private key (from secure storage)
  - ECDSA signature of the EAT



ESP-TEE: How to Use It

ESPRESSIF DevCon24

Migration of Existing Projects

- Enable the TEE Kconfig option
- Partition Table needs to be updated
- TEE firmware generation handled by the build system
- Reference examples will be provided for all the current use cases

Performance and Resource Impact

01

64 KB SRAM reserved for TEE (configurable)

02

~280KB external flash for TEE (use case dependent)

03

CPU cycles overhead: Service Call

Entry: 778 (4.86 µs @ 160 MHz)

Exit: 365 (2.28 µs @ 160 MHz)

Future Work

01

ESP-TEE Beta version will release soon

02

Support for other RISC-V SoCs in plans

03

Certification and Regulatory Compliance (E.g., PSA L2)

04

Optimize performance and resource impact

ESPRESSIF DevCon24

Thanks for watching !