MyTEE: Own the Trusted Execution Environment on Embedded Devices

Seungkyun Han and Jinsoo Jang

Chungnam National University, Korea

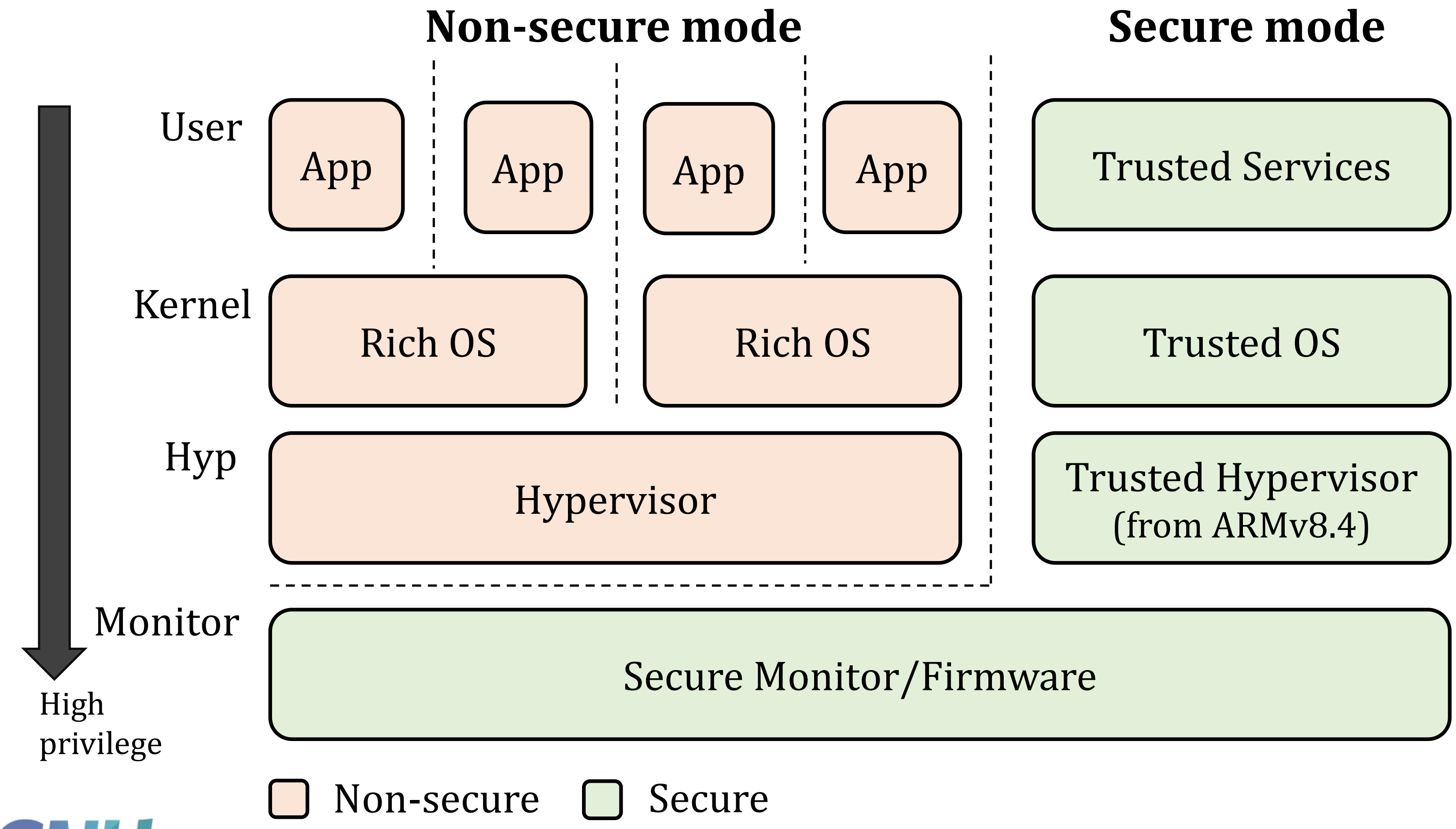
Network and Distributed System Security Symposium (NDSS '23)

Trusted Execution Environment

- Isolate and protect security-critical services
  - ✓ Mobile banking and payment
  - ✓ Digital rights management (DRM)
  - ✓ Private and confidential data
    - ➢ User credential
    - ➢ Crypto key
    - ➢ Medical information

Background

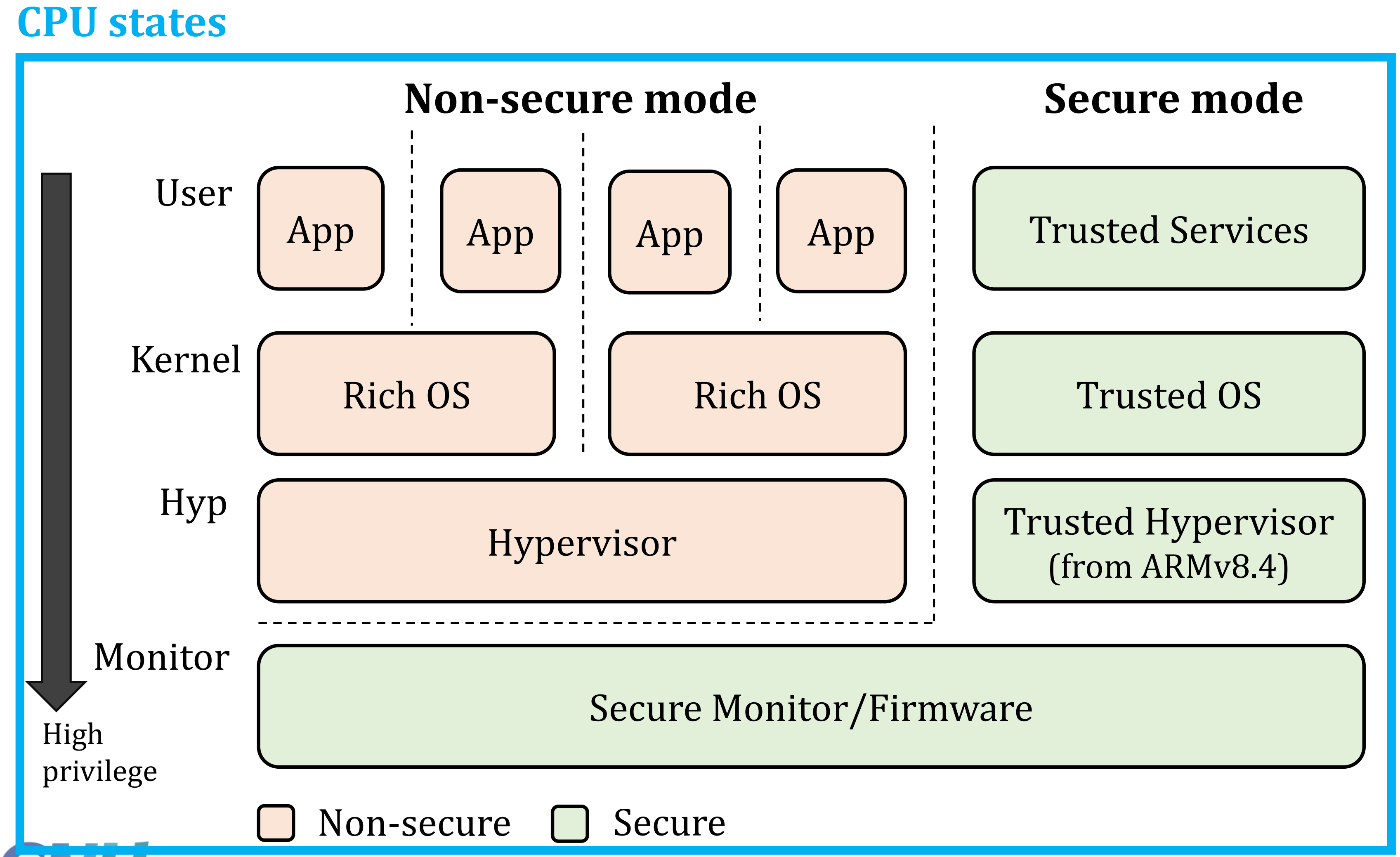
- ARMv8 Architecture and TrustZone

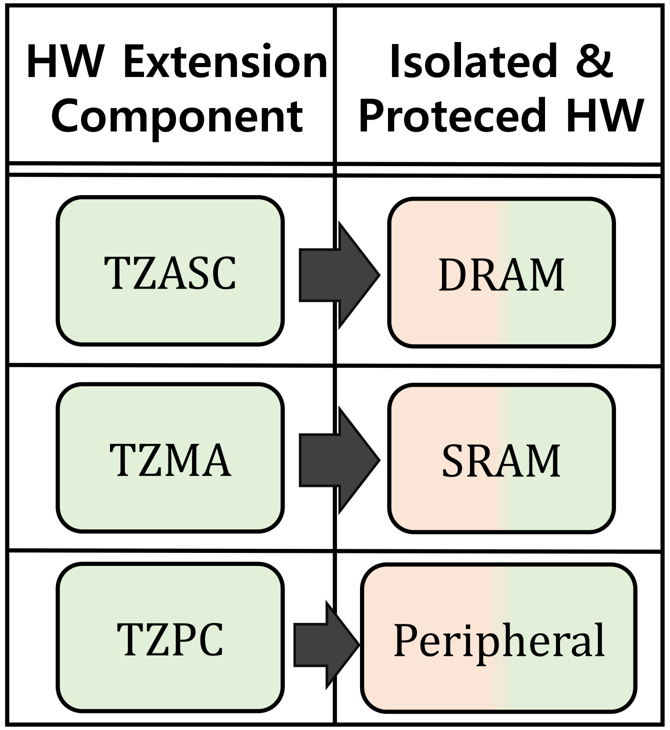


HW Extension Component	Isolated & Proteced HW
TZASC	DRAM
TZMA	SRAM
TZPC	Peripheral

Background

- ARMv8 Architecture and TrustZone

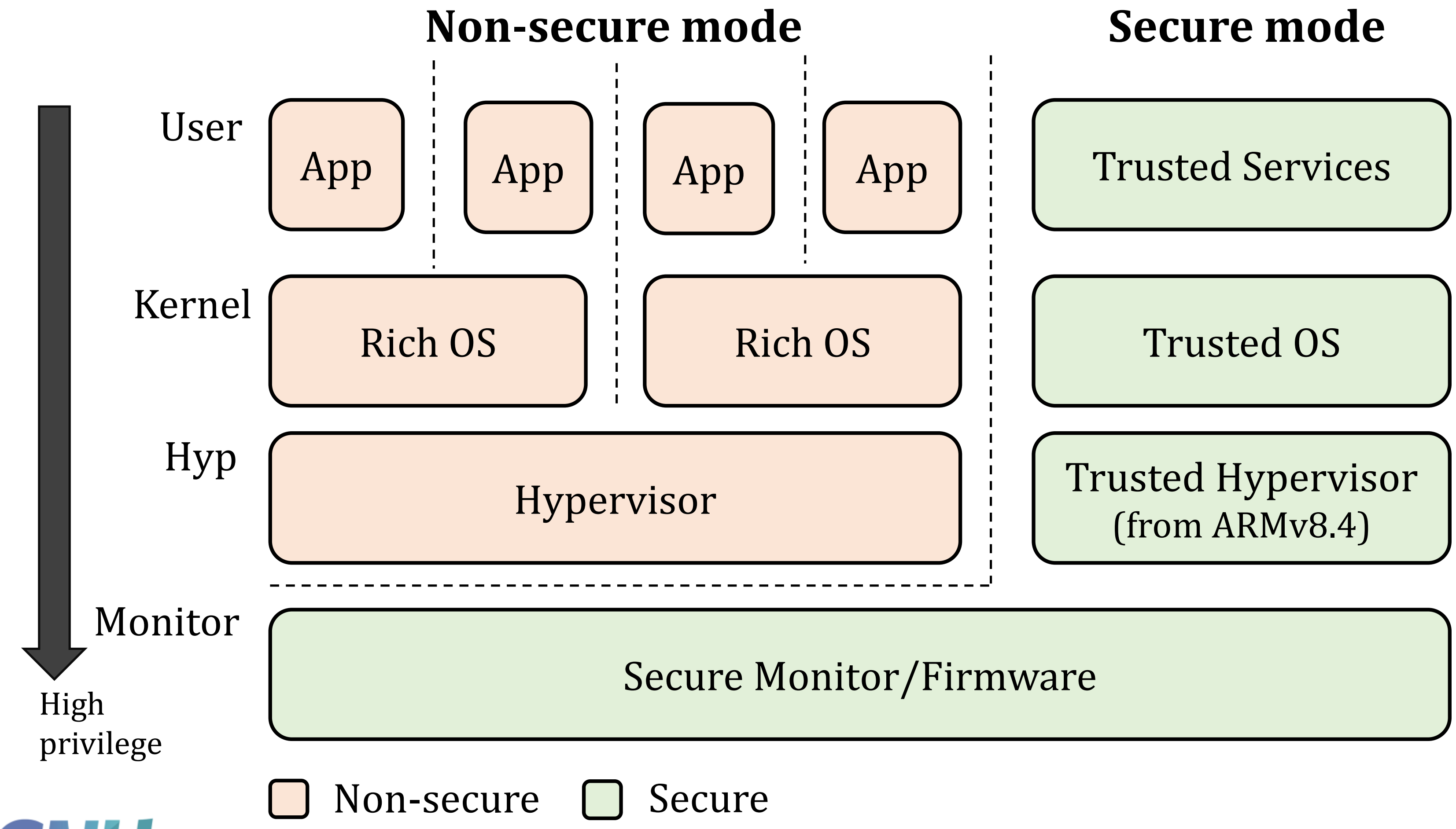




HW Extension Component	Isolated & Proteced HW
TZASC	DRAM
TZMA	SRAM
TZPC	Peripheral

Background

- ARMv8 Architecture and TrustZone

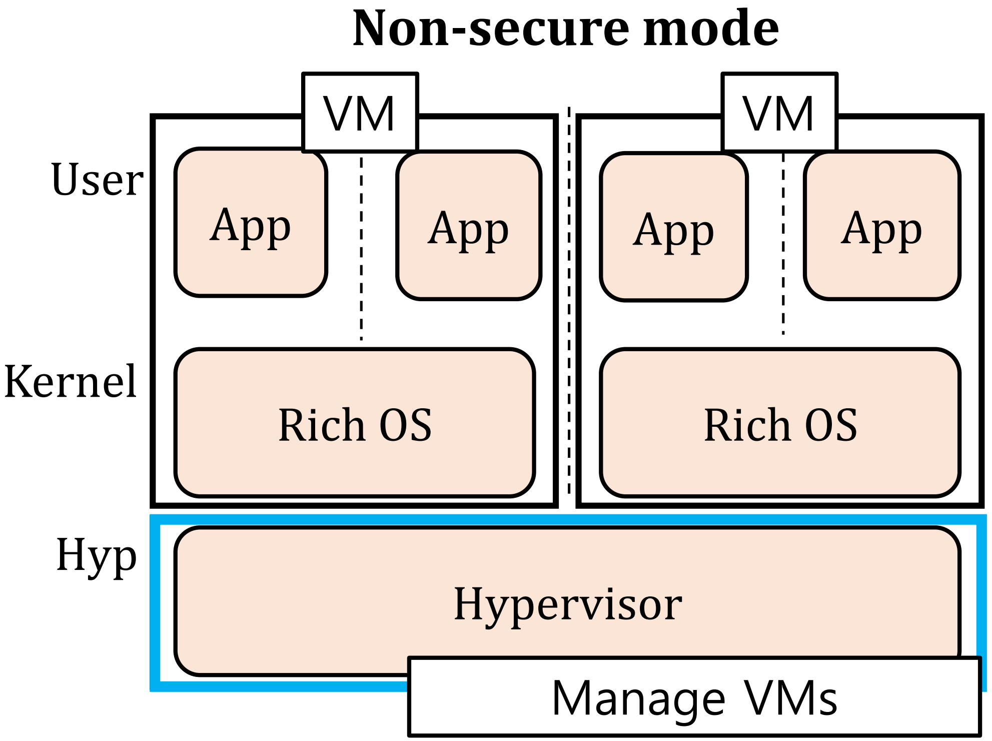


TrustZone hardware extensions

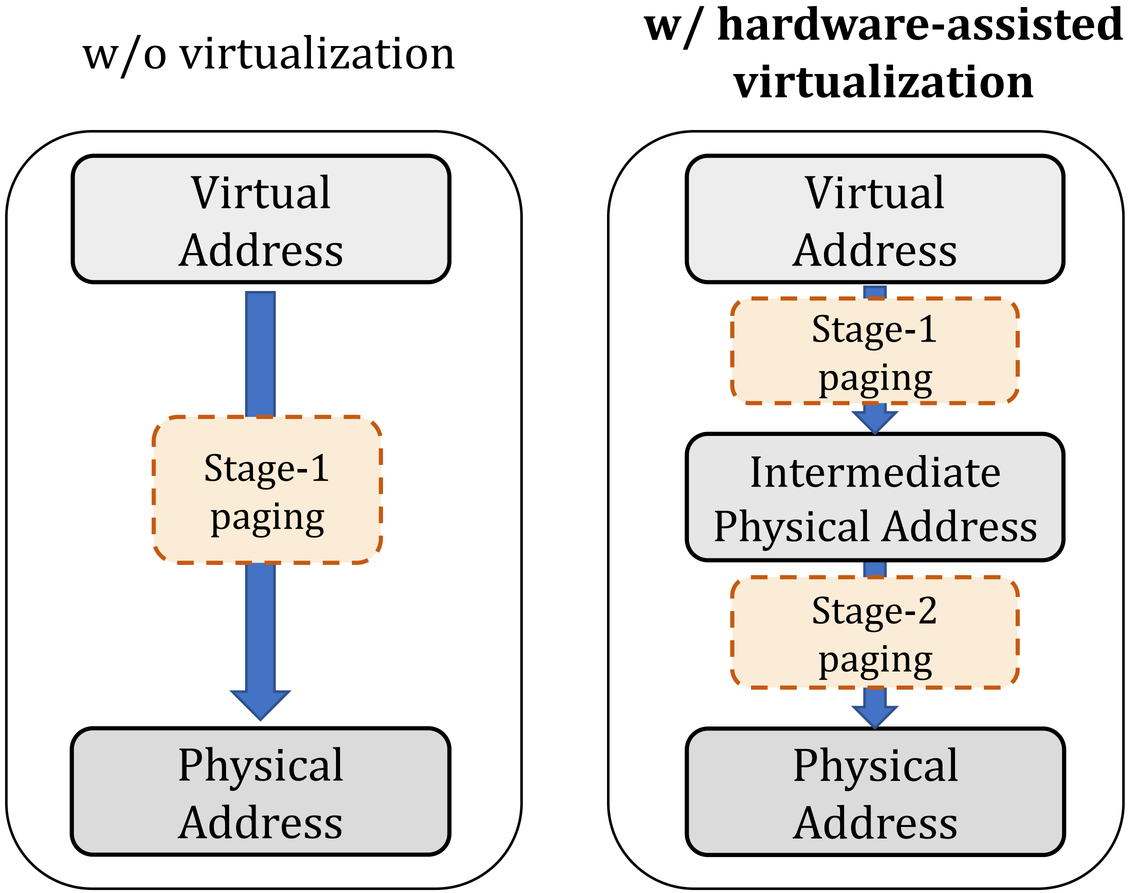
HW Extension Component	Isolated & Proteced HW
TZASC	DRAM
TZMA	SRAM
TZPC	Peripheral

Background

- Virtualization Extension

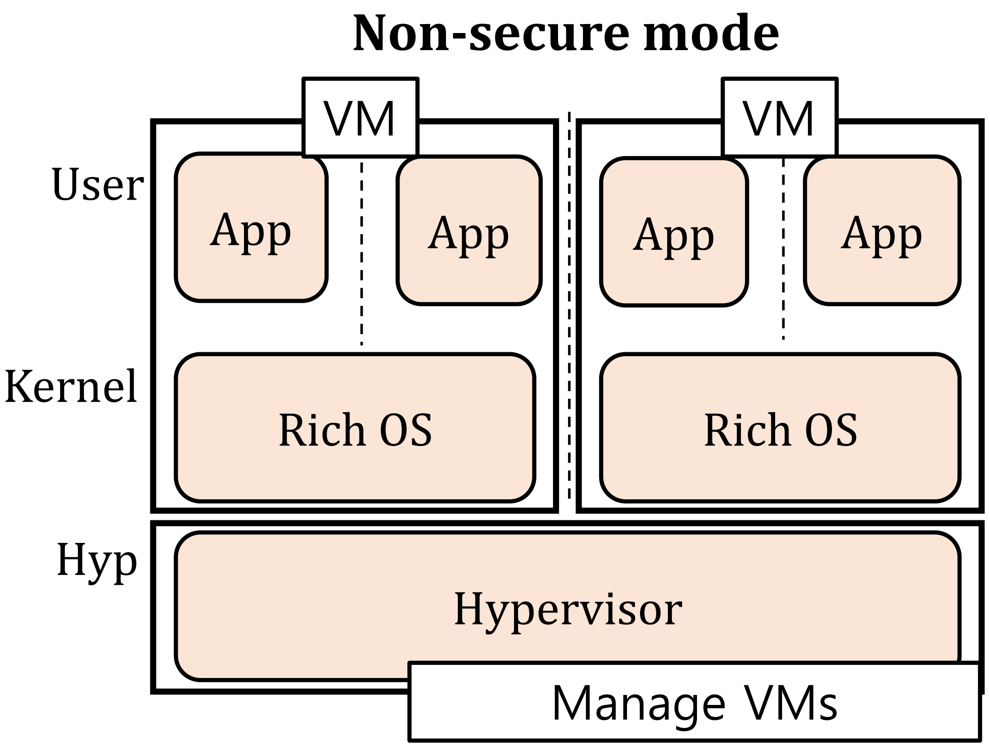


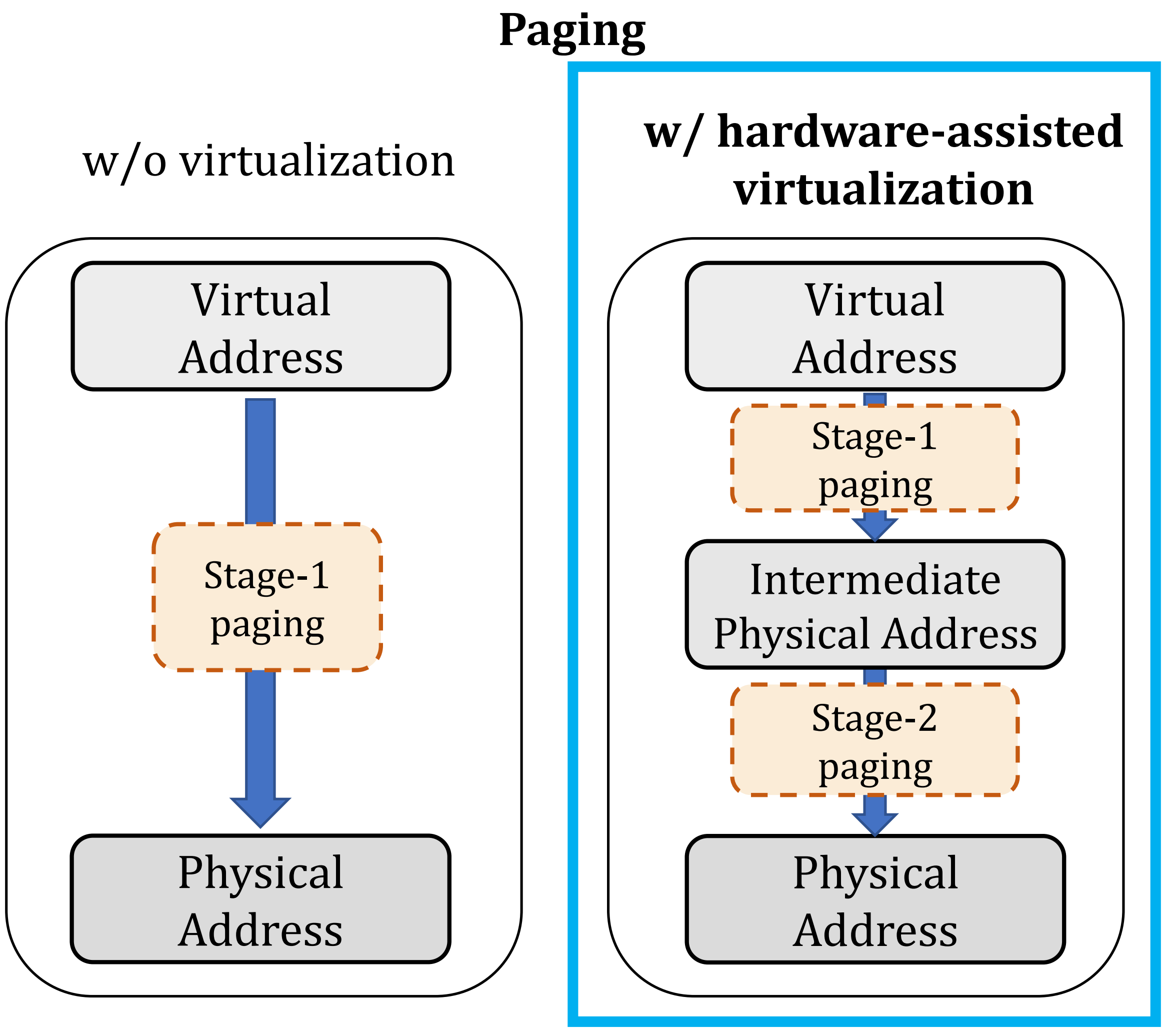
Paging



Background

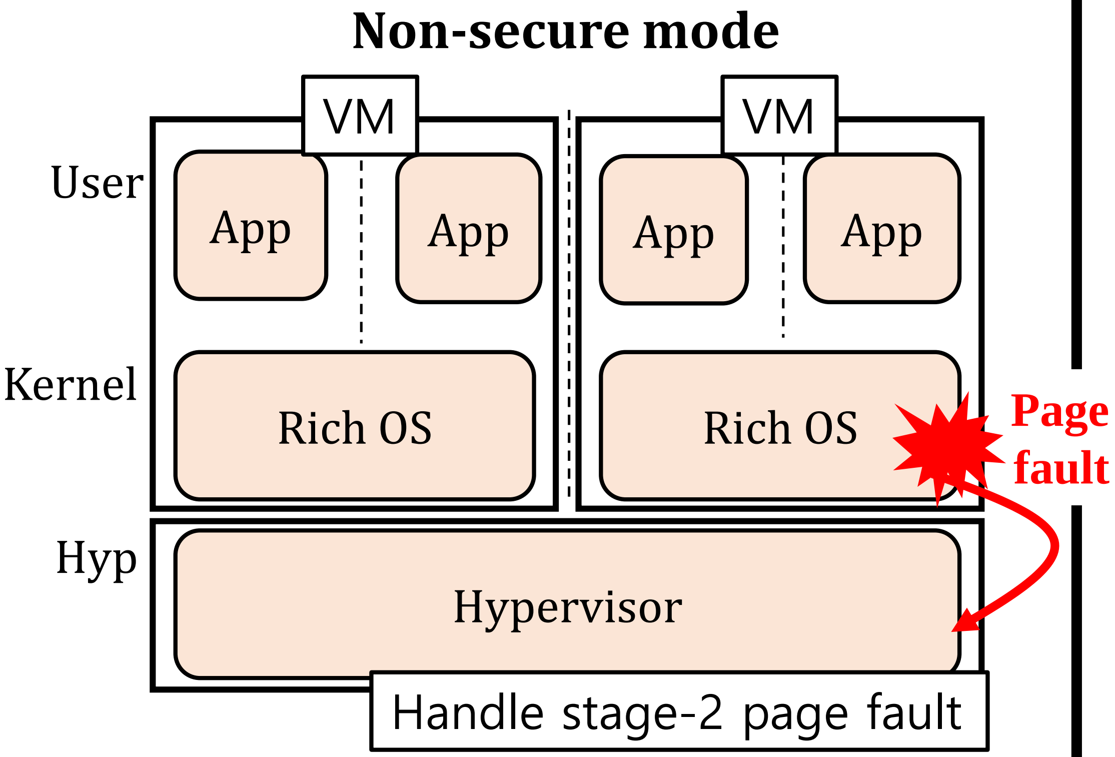
- Virtualization Extension

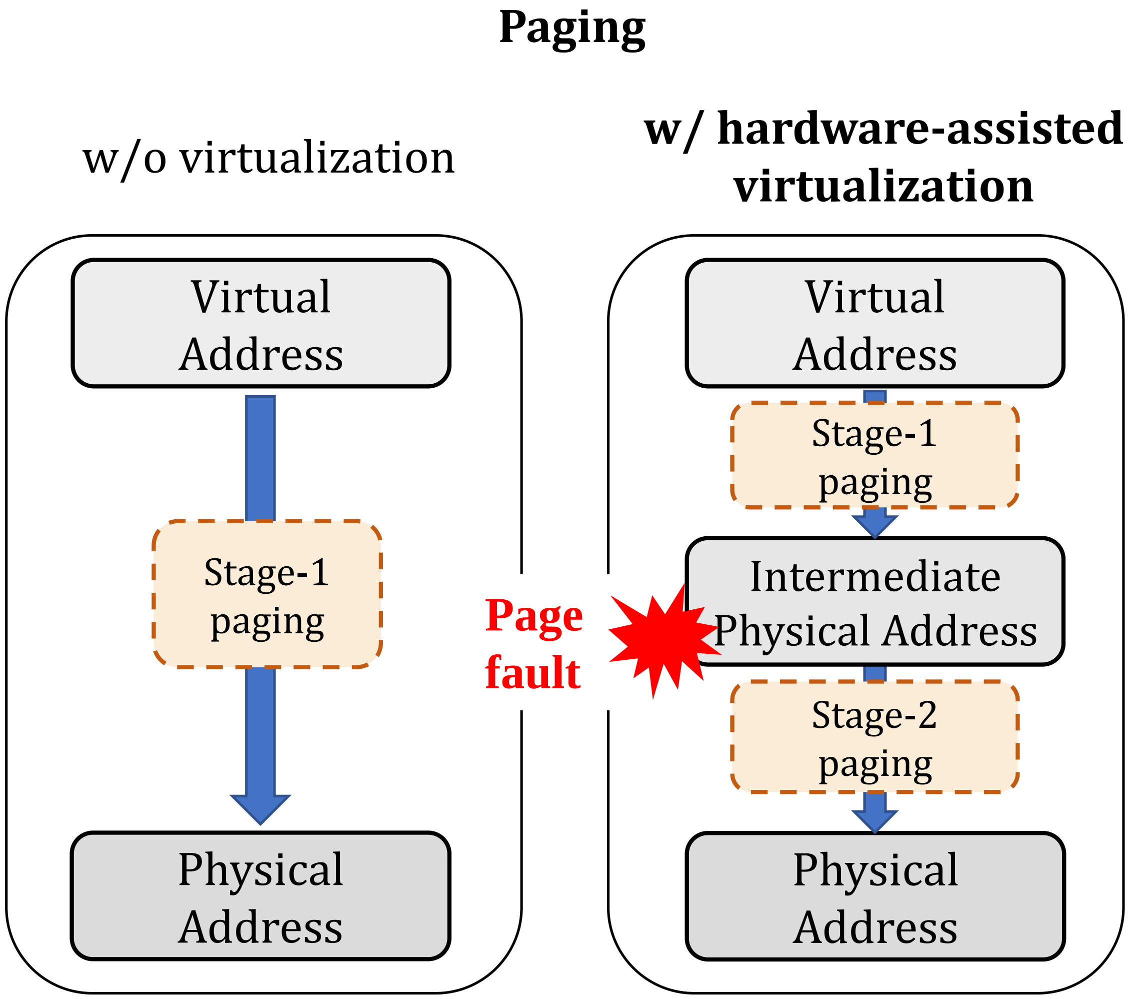




Background

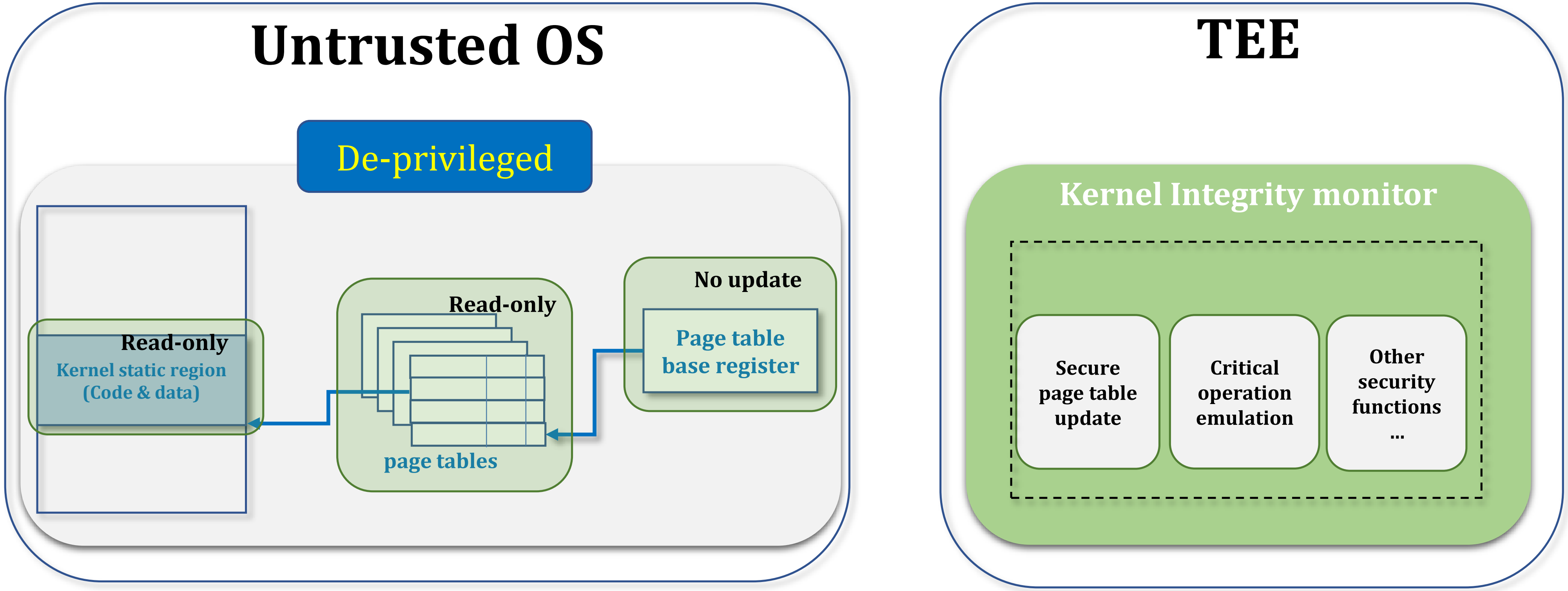
- Virtualization Extension





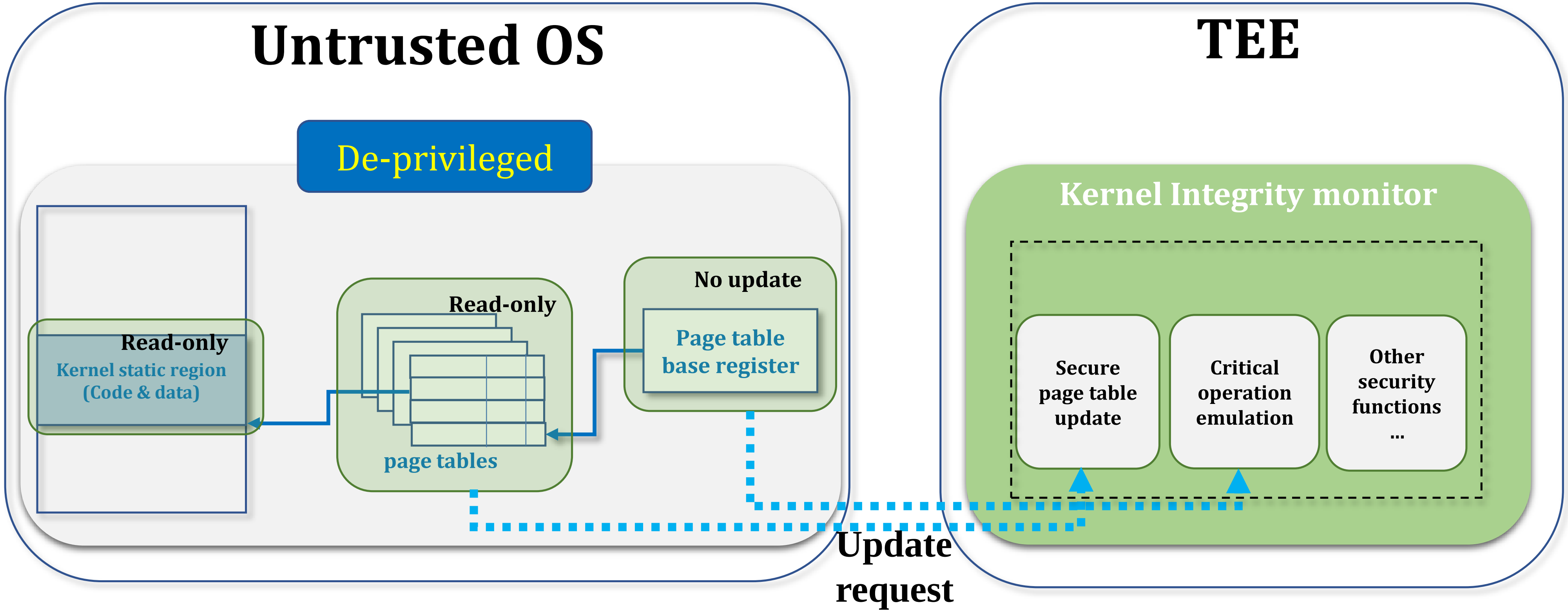
Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)



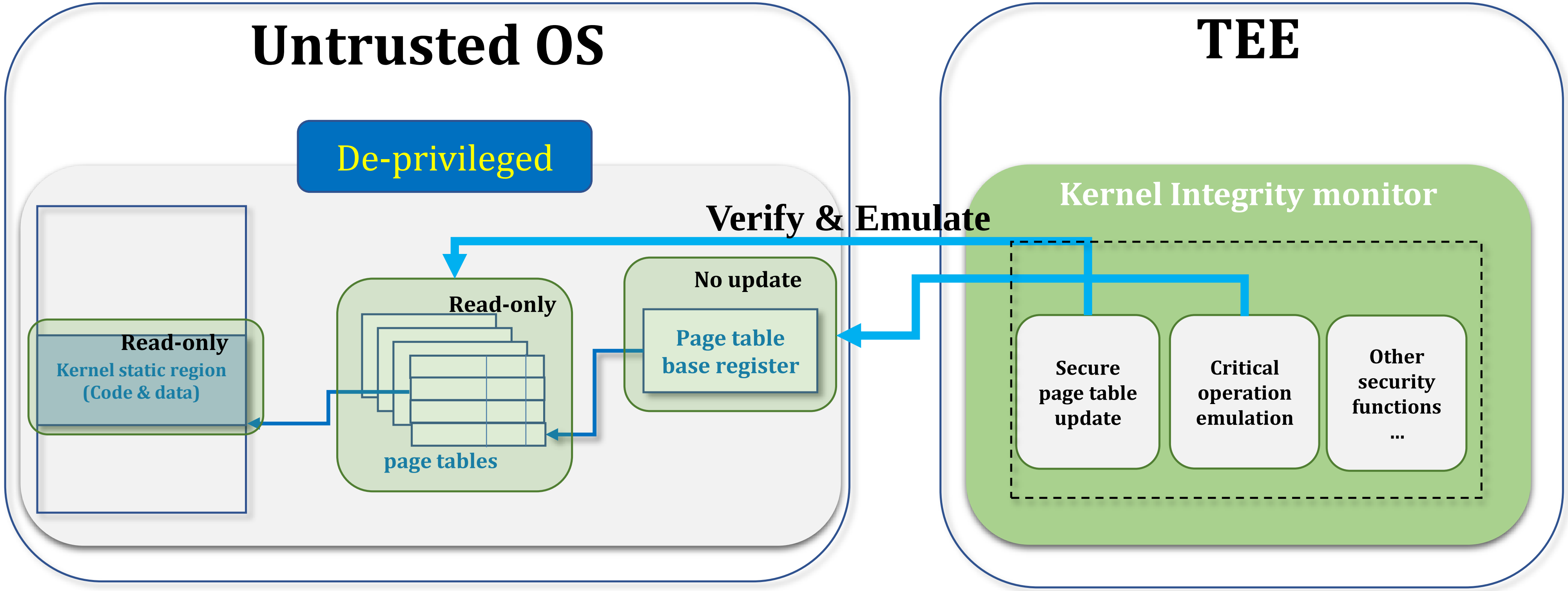
Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)



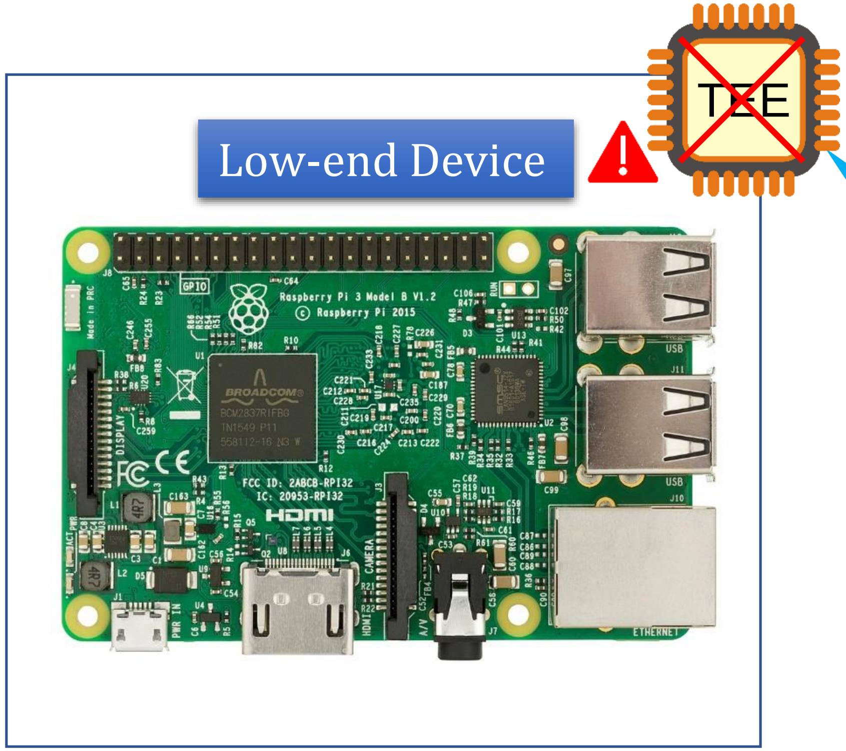
Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)



Lack of TrustZone Extensions

- Some SoCs do not support TrustZone hardware extentions



- No TZASC → No DRAM protection
- No TZMA → No SRAM protection
- No TZPC → No peripheral protection

Lack of TrustZone Extensions

- Example ARMv8-A based SoCs that lack TrustZone extensions

Vendor	SoC	Secure State	TZPC	TZASC	TZMA	ISA	Device
Bradcom	BCM2837	●	○	○	○	v8.0	I
Unisoc	SC9863A	●	○	○	○	v8.1	M, T
Amlogic	G12A	●	○	◐	○	v8.0	I
NXP	LS1012ASN	●	◐	○	◐	v8.0	I
MediaTek	MT6739, 6765	●	○	○	○	v8.0	M, T
Samsung	Exynos 7570, 7578	●	○	○	○	v8.0	M

● Supported, ◐ Presumably supported (not publicly opened), ○ Not supported,
M: Mobile phone, T: Tablet PC, I: IoT device

Lack of TrustZone Extensions

- Example ARMv8-A based SoCs that lack TrustZone extensions

Vendor	SoC	Secure State	TZPC	TZASC	TZMA	ISA	Device
Bradcom	BCM2837	●	○	○	○	v8.0	I
Unisoc	SC9863A	●	○	○	○	v8.1	M, T
	7578						

How to build the TEE without depending on the TrustZone hardware extensions?

● Supported, ◐ Presumably supported (not publicly opened), ○ Not supported,
M: Mobile phone, T: Tablet PC, I: IoT device

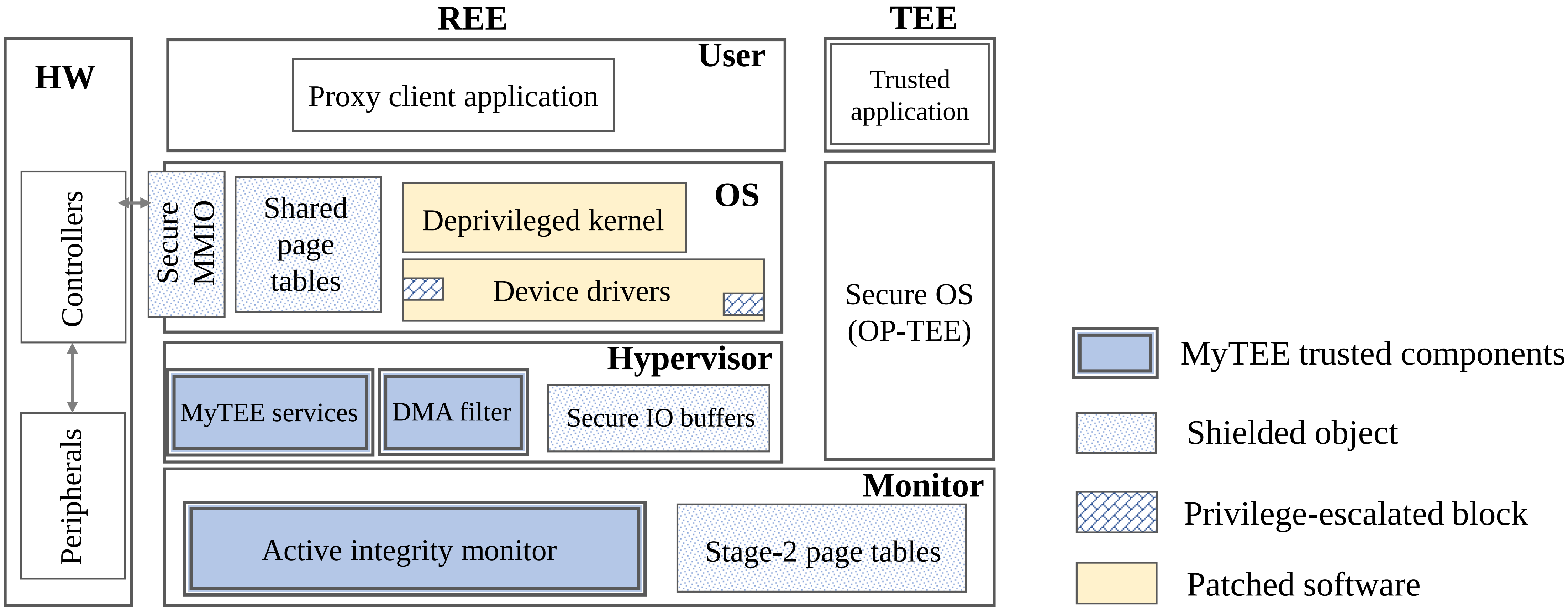
Our Goal & Assumption

- Build the TEE without the support of mandatory TrustZone extensions
  - ✓ TEE memory protection
  - ✓ Secure IO without bloating the TEE

- Assumption
  - ✓ Kernel text and data are immutable by RKP
  - ✓ Host and peripheral hardware are physically isolated and not malicious
  - ✓ Secure boot
  - ✓ Side channel attacks are not considered

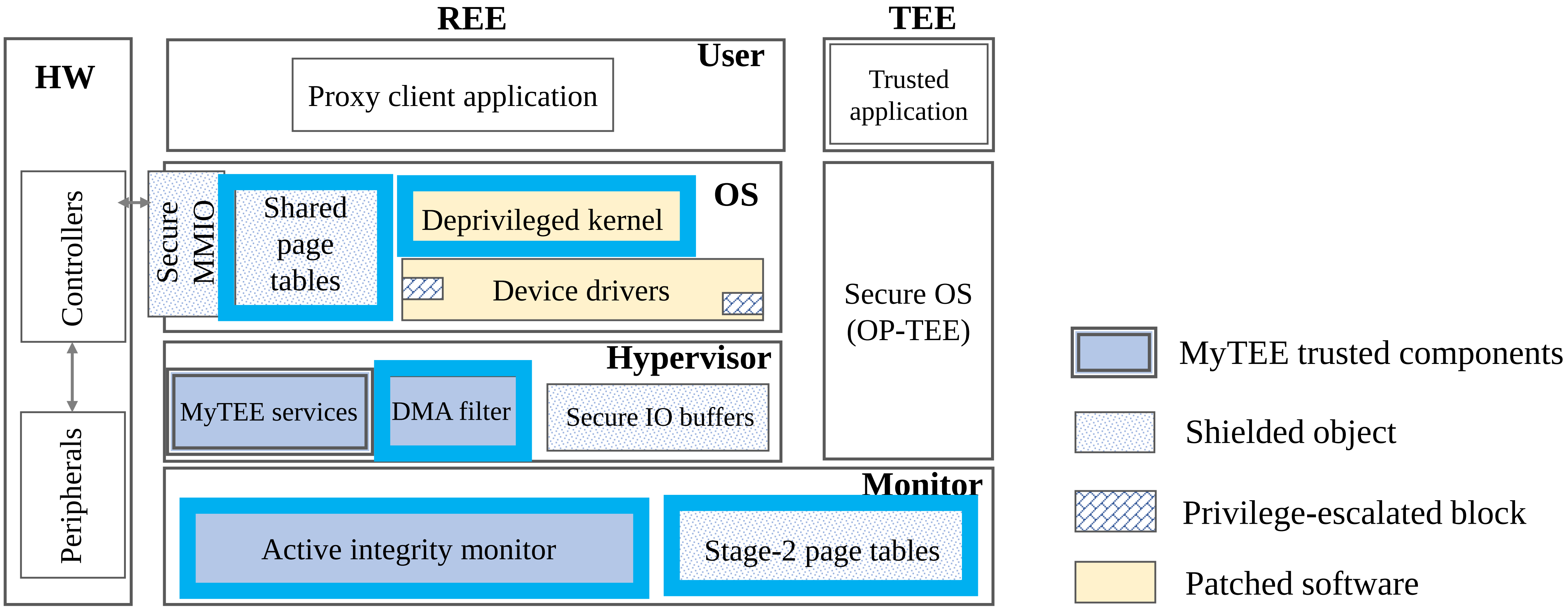
MyTEE – Overview

- Framework to provide memory protection and secure IO



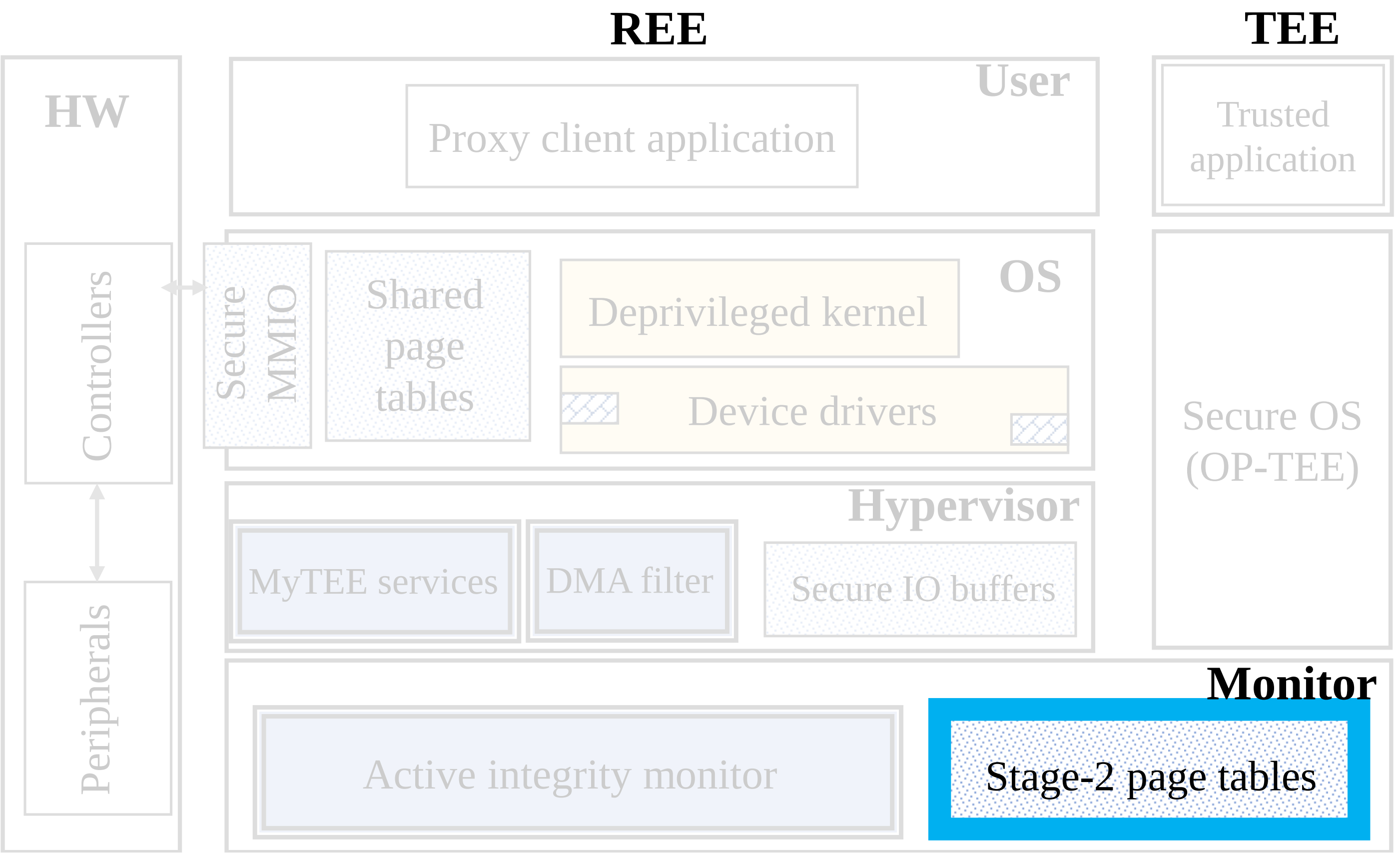
System Design: Memory Protection

- Memory protection components



System Design: Memory Protection cont’d

- Leverage stage-2 paging to protect the TEE and the hypervisor



Isolate the TEE and the hypervisor from the untrusted OS

▭ MyTEE trusted components

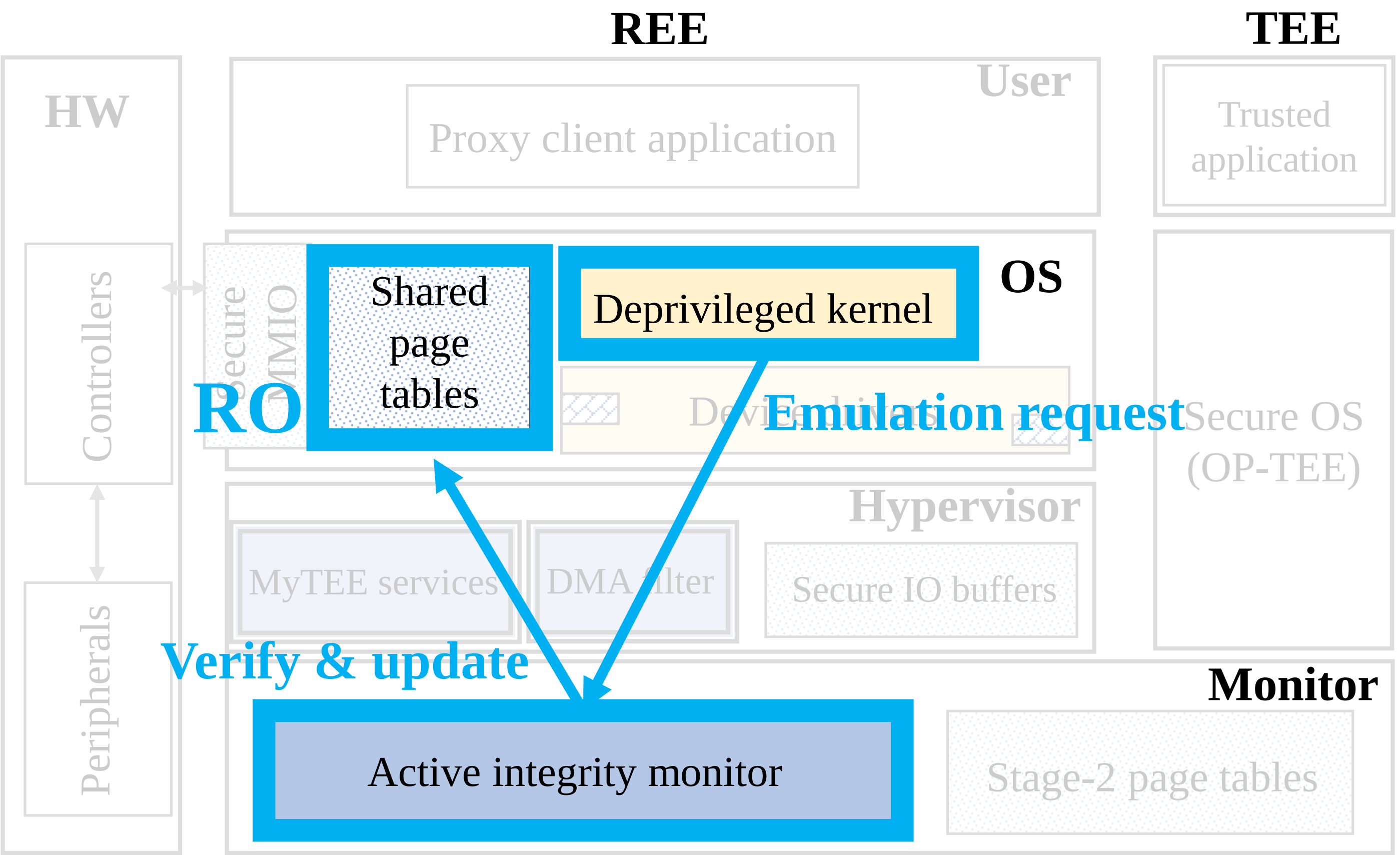
░ Shielded object

▨ Privilege-escalated block

🟨 Patched software

System Design: Memory Protection cont'd

- Deprivileging the OS to emulate the security critical operations



- Set page tables to read-only

- Verify and emulate the page table update

▭ MyTEE trusted components

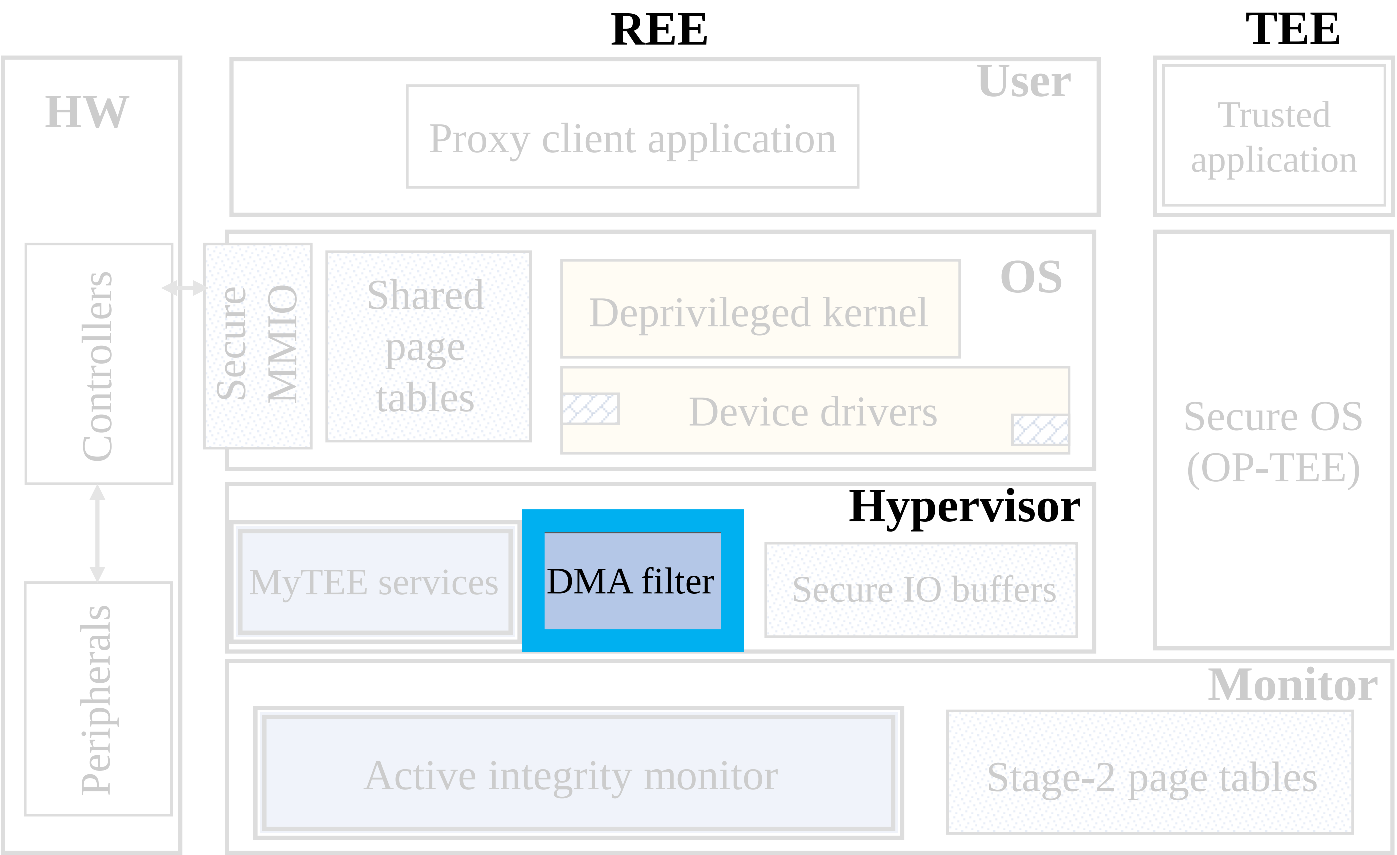
░ Shielded object

▨ Privilege-escalated block

▭ Patched software

System Design: Memory Protection cont'd

- DMA filter for preventing DMA attacks



REE

TEE

HW

User

Proxy client application

Trusted application

Controllers

Secure MMIO

Shared page tables

Deprivileged kernel

OS

Device drivers

Secure OS (OP-TEE)

Peripherals

Hypervisor

MyTEE services

DMA filter

Secure IO buffers

Monitor

Active integrity monitor

Stage-2 page tables

Prevent the DMA to the protected memory regions

▣ MyTEE trusted components

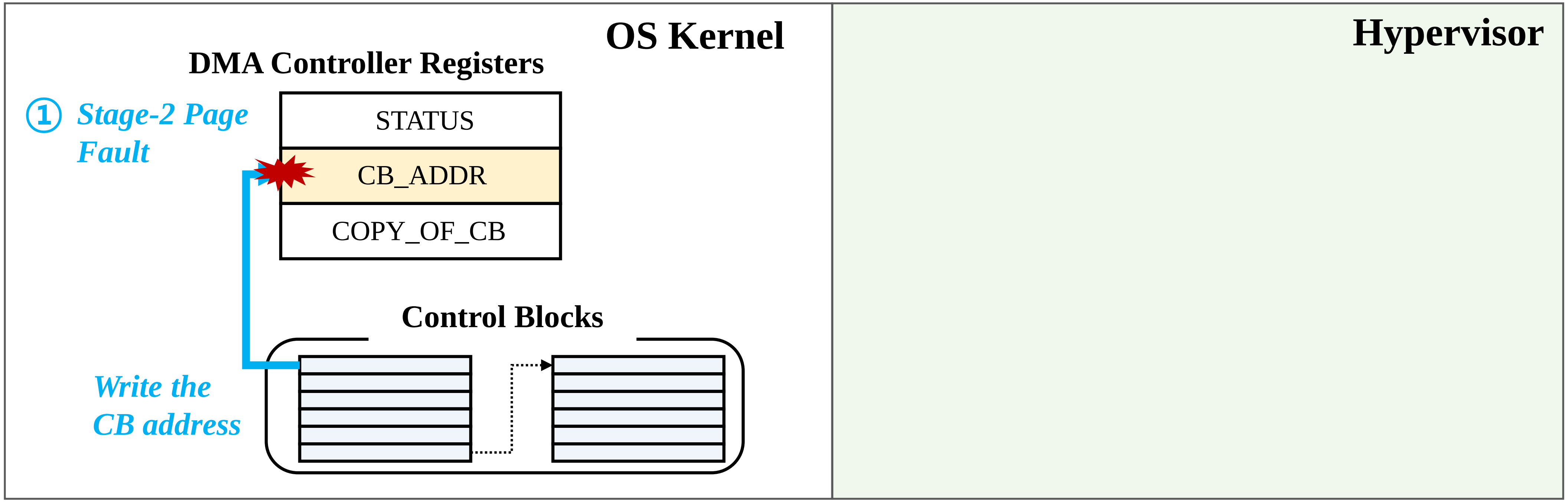
░ Shielded object

▨ Privilege-escalated block

□ Patched software

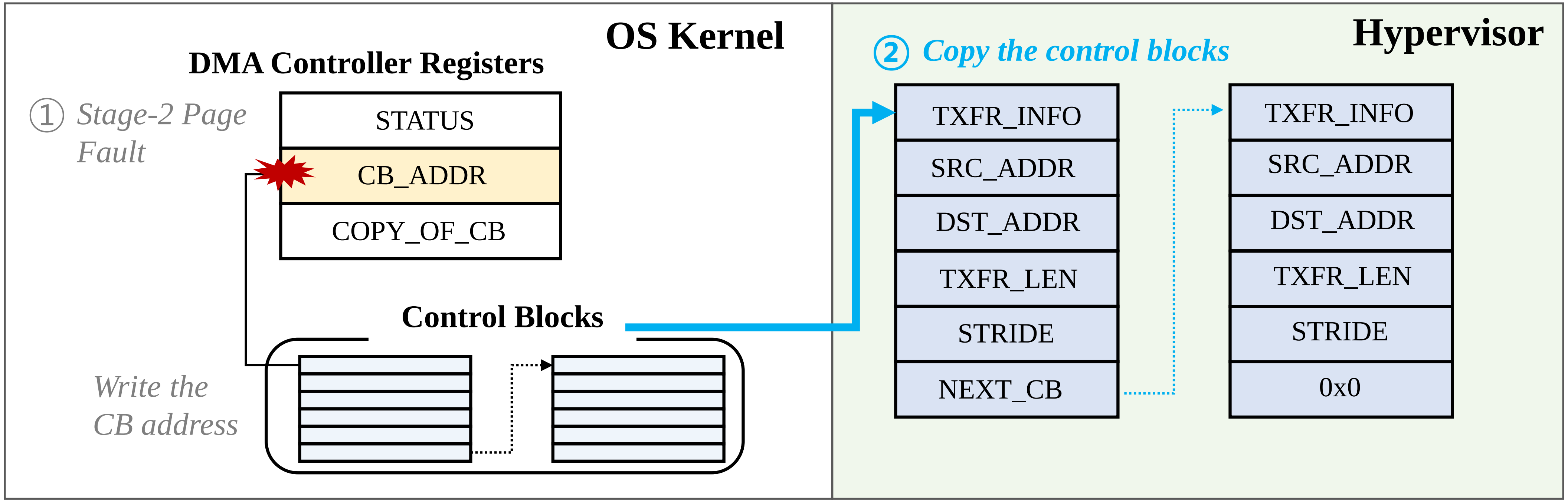
System Design: Memory Protection cont’d

- DMA filter for preventing DMA attacks (cont’d)
  - MMIO for DMA controller is protected by using stage-2 paging
  - Verify and emulate every DMA request



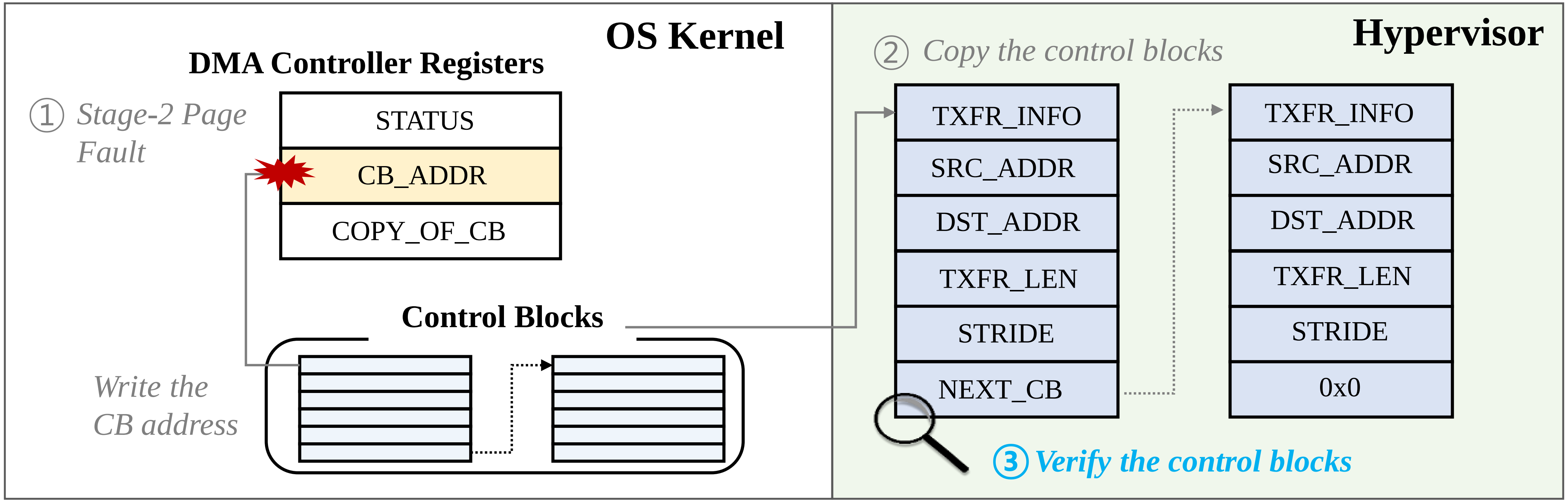
System Design: Memory Protection cont’d

- DMA filter for preventing DMA attacks (cont’d)
  - MMIO for DMA controller is protected by using stage-2 paging
  - Verify and emulate every DMA request



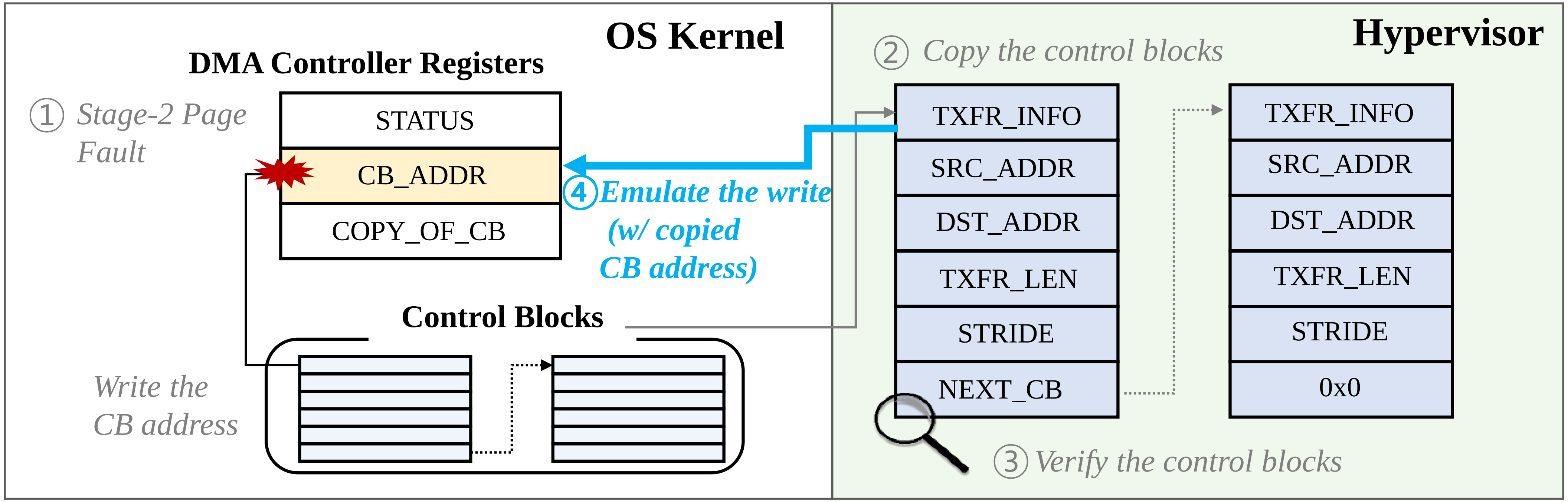
System Design: Memory Protection cont’d

- DMA filter for preventing DMA attacks (cont’d)
  - MMIO for DMA controller is protected by using stage-2 paging
  - Verify and emulate every DMA request



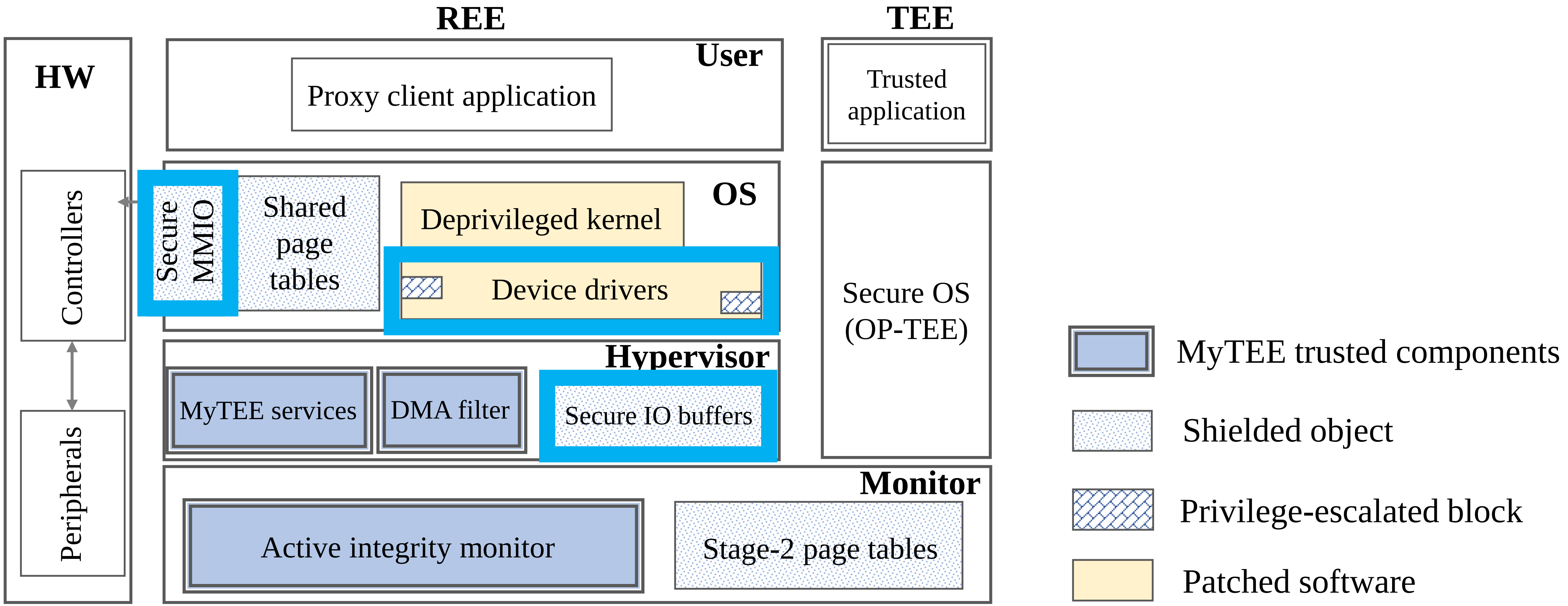
System Design: Memory Protection cont'd

- DMA filter for preventing DMA attacks (cont'd)
  - MMIO for DMA controller is protected by using stage-2 paging
  - Verify and emulate every DMA request



System Design: Secure IO

- Secure IO components
- Do not bloat the TEE by leveraging the kernel device driver



REE

TEE

HW

Controllers

Peripherals

User

Proxy client application

Trusted application

OS

Secure MMIO

Shared page tables

Deprivileged kernel

Device drivers

Secure OS (OP-TEE)

Hypervisor

MyTEE services

DMA filter

Secure IO buffers

Monitor

Active integrity monitor

Stage-2 page tables

MyTEE trusted components

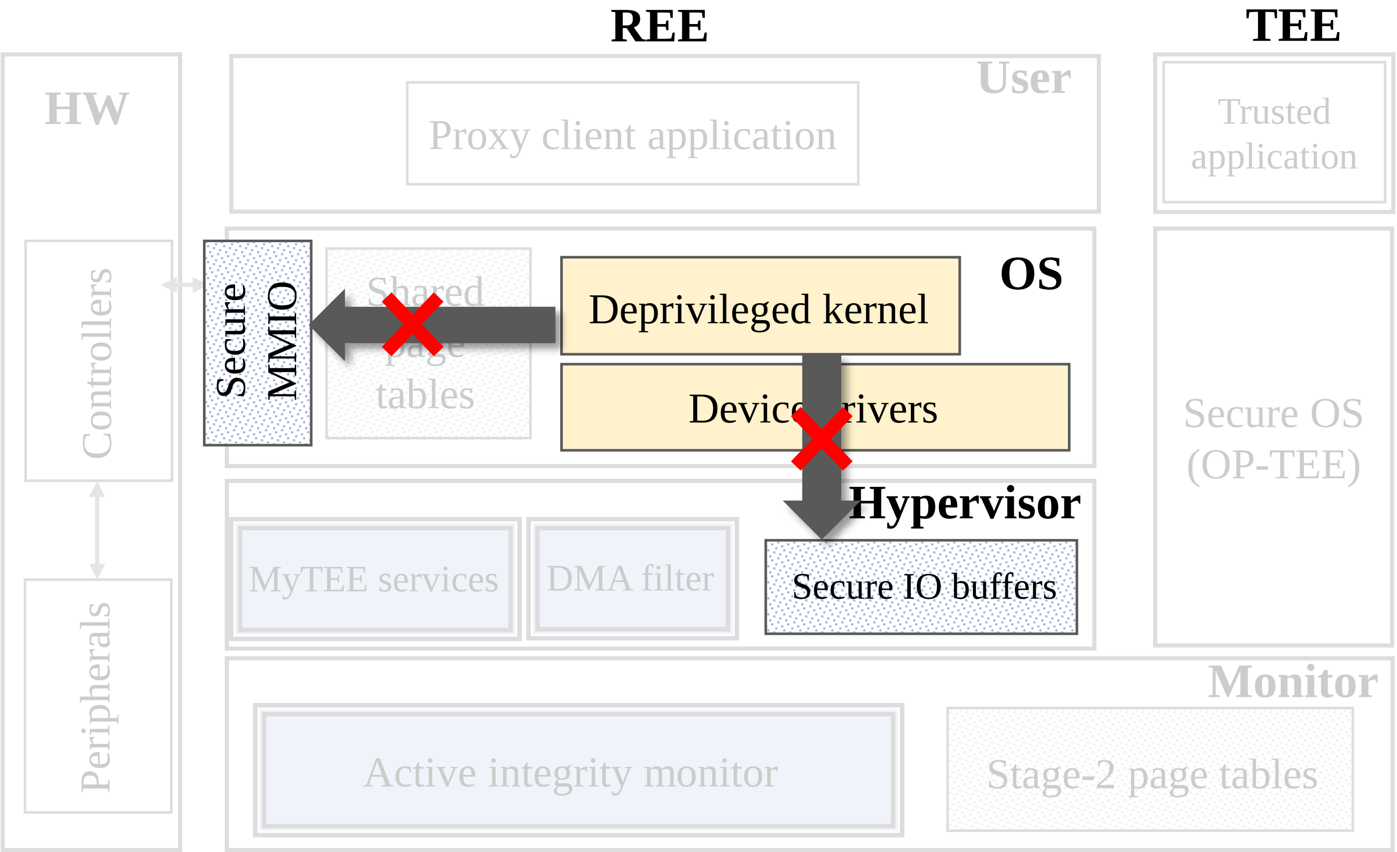
Shielded object

Privilege-escalated block

Patched software

System Design: Secure IO cont'd

- Stage-2 paging protects the MMIO and buffers for IO



OS and kernel drivers cannot access the protected MMIO region and IO buffers

▭ MyTEE trusted components

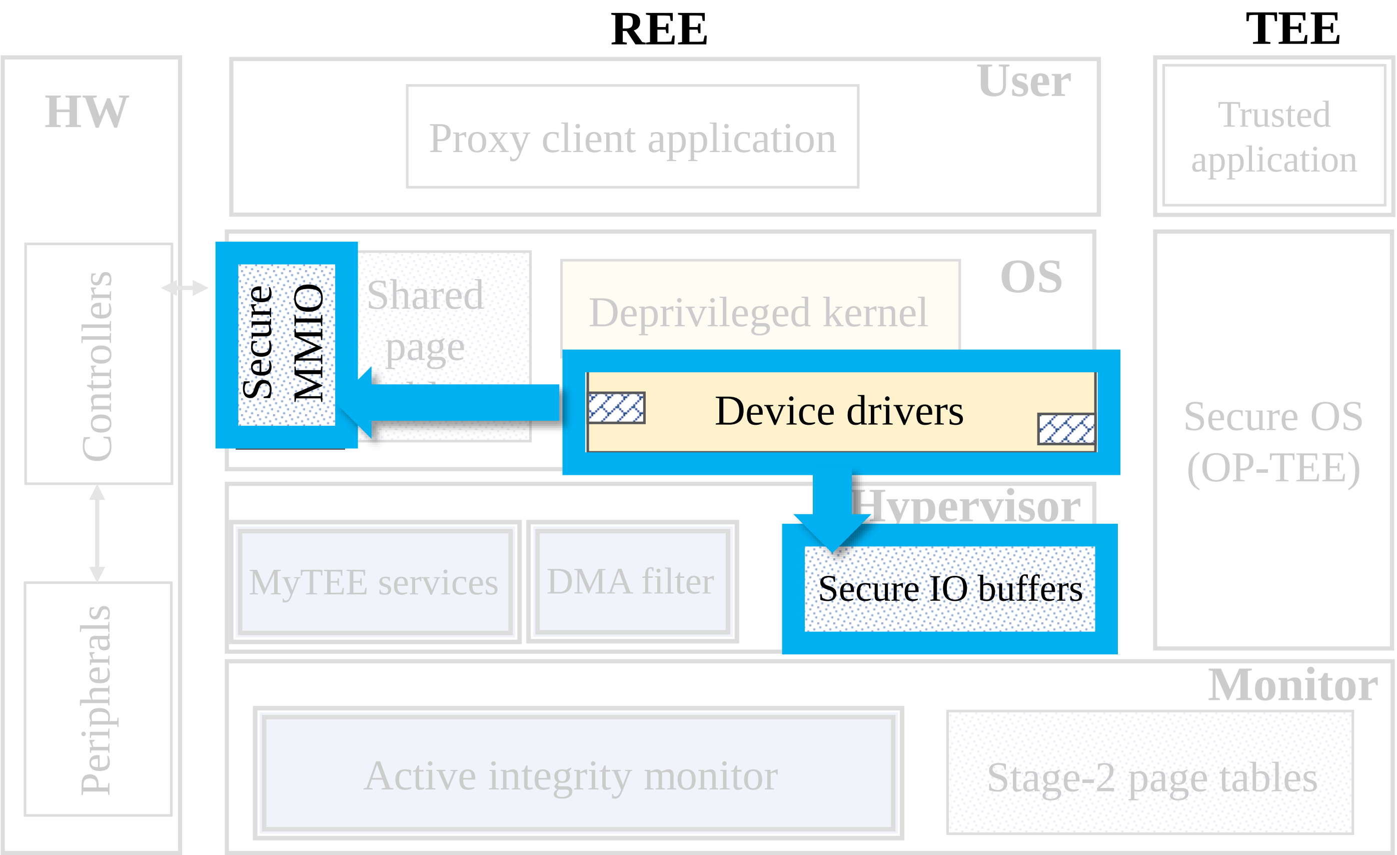
▨ Shielded object

▨ Privilege-escalated block

▭ Patched software

System Design: Secure IO cont'd

- Part of device driver text is given higher (hypervisor) privilege

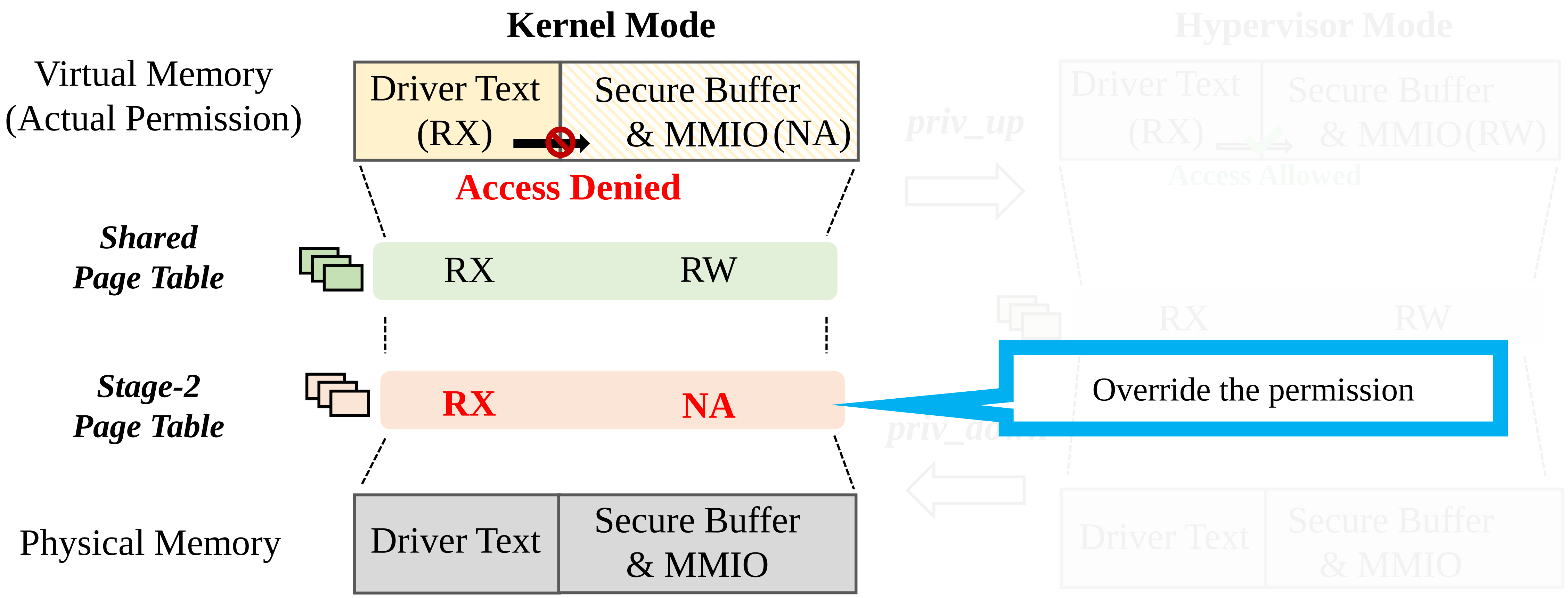


Privilege-escalated kernel driver can access the protected objects

▭	MyTEE trusted components
░	Shielded object
▨	Privilege-escalated block
🟨	Patched software

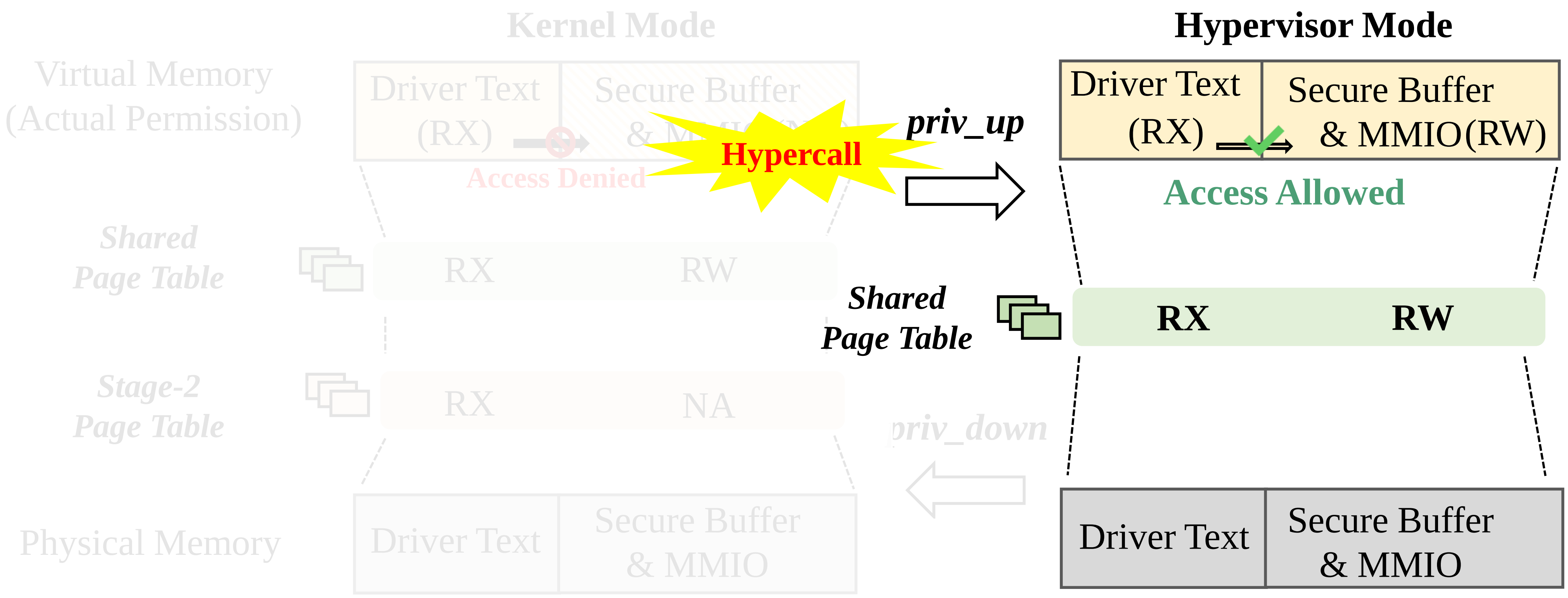
System Design: Secure IO cont'd

- Temporal privilege escalation



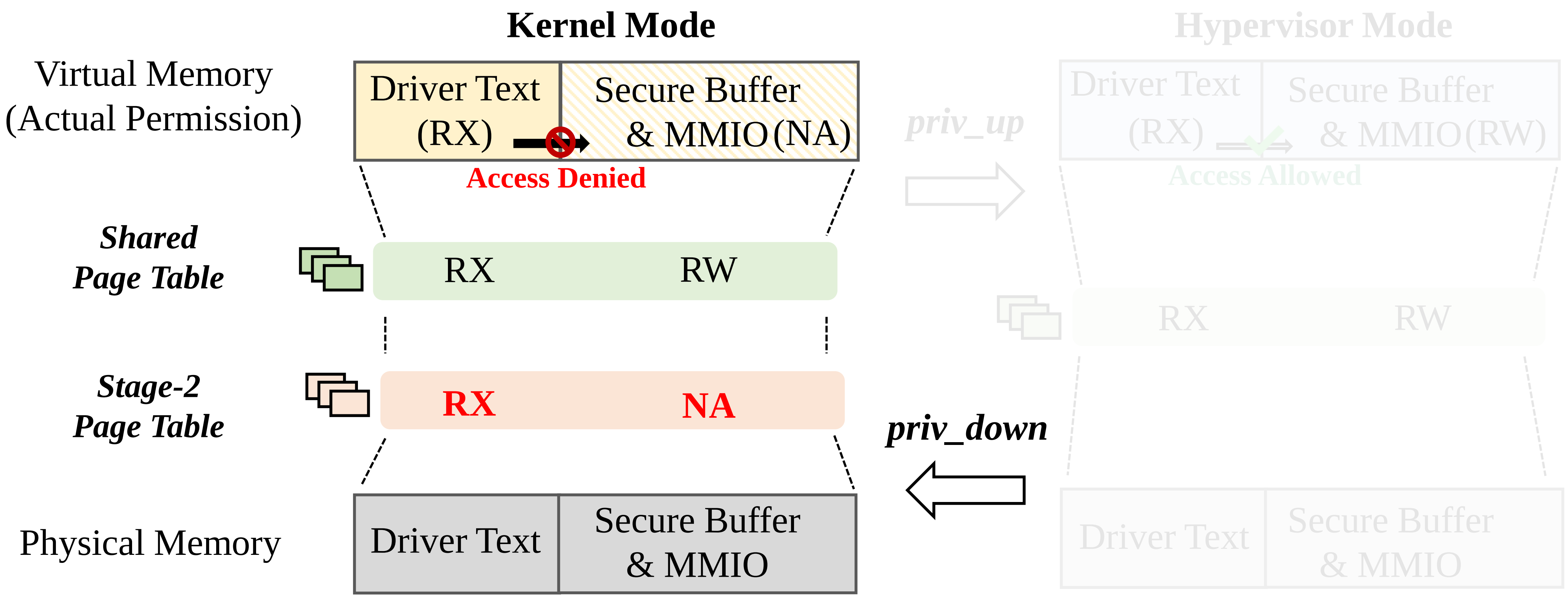
System Design: Secure IO cont’d

- Temporal privilege escalation (cont’d)



System Design: Secure IO cont'd

- Temporal privilege escalation (cont'd)



CNU

SECRET

System Design: Secure IO cont'd

- Instrumentation example with MyTEE APIs for enabling the secure IO

```
static int bcm2835_send_data(struct
     mbox_chan *link, void *data) {
    ...
    writel(msg, mbox->regs + MAIL1_WRT);
    ...
}
```

System Design: Secure IO cont’d

- Instrumentation example with MyTEE APIs for enabling the secure IO

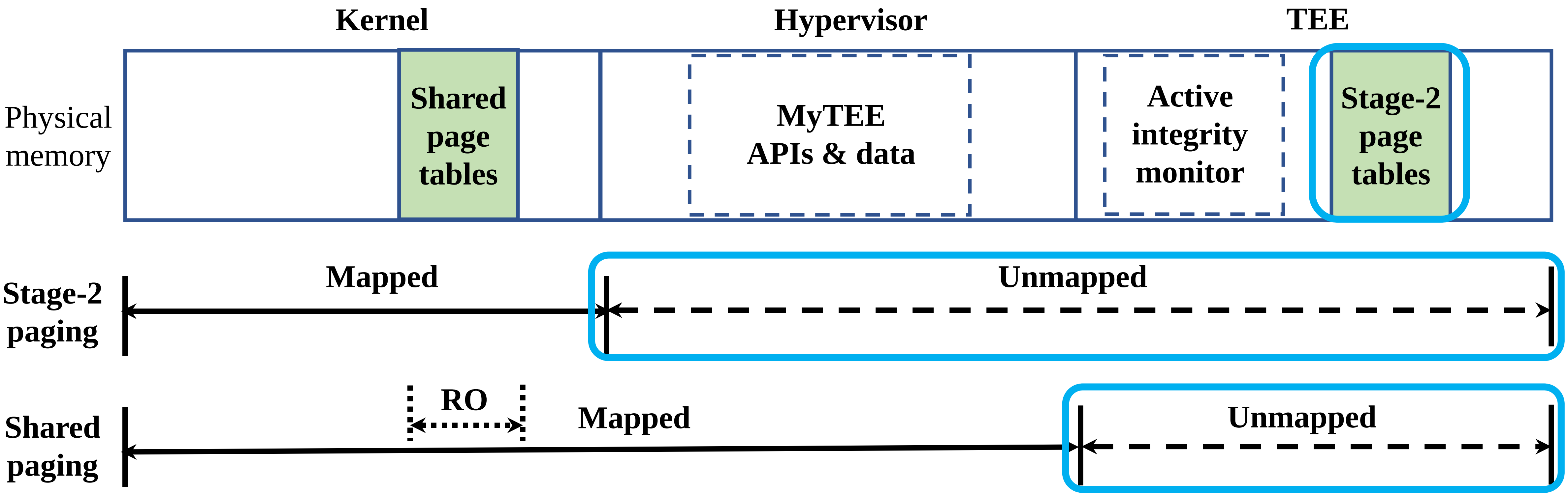
- Instrumented driver for the secure IO
- Hypervisor change is minimized

```
static int bcm2835_send_data(struct
     mbox_chan *link, void *data) {
    ...
    writel(msg, mbox->regs + MAIL1_WRT);
    ...
}
```

```
mytee_wrapper_writel(u32 msg, u32 mmio_addr){
    int ret;
    ret = mytee_verify_memopr(MAILBOX_WRT, mmio_addr, \
    sizeof(u32));
    if(!ret){
        mytee_log_txn(MAILBOX_WRT, msg);
        writel(msg, mmio_addr);
}}
static int bcm2835_send_data(struct
     mbox_chan *link, void *data) {
    ...
    mytee_priv_up();
    mytee_wrapper_writel(msg, mbox->regs + MAIL1_WRT);
    mytee_priv_down();
... }
```

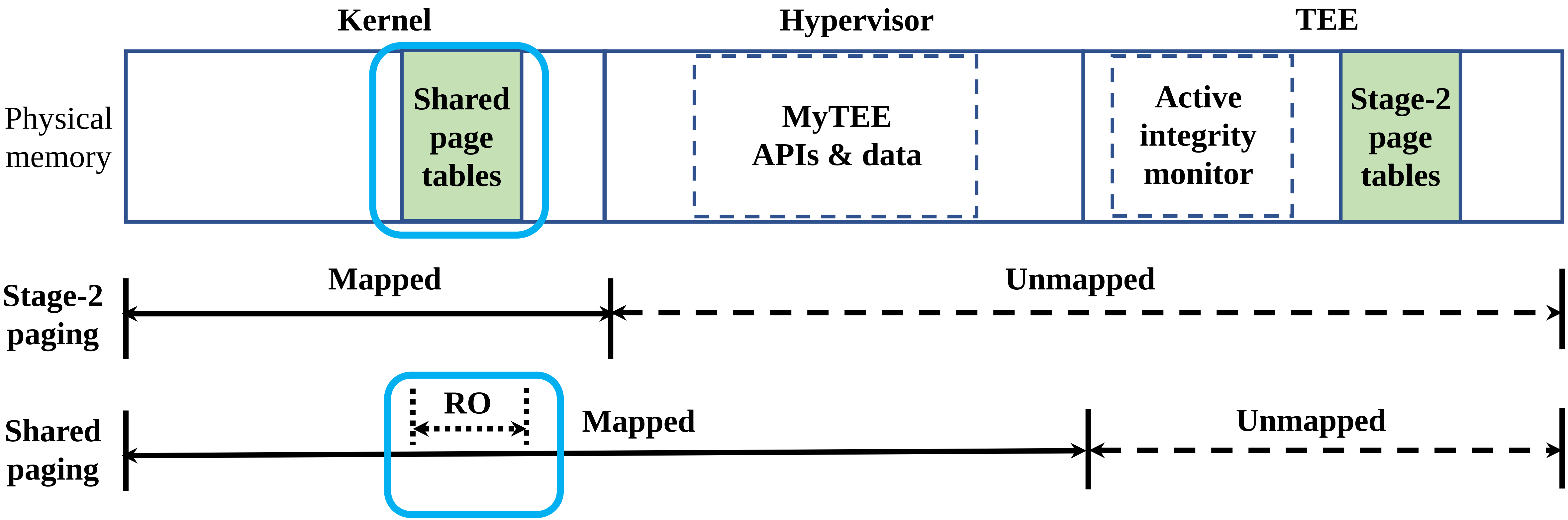
System Design: Secure IO cont'd

- Page tables are secured from the malicious privileged code
  - **Stage-2 page tables** are placed in the **unmapped region from the OS and hypervisor**
  - Shared page tables are set to RO so even the hypervisor cannot manipulated it



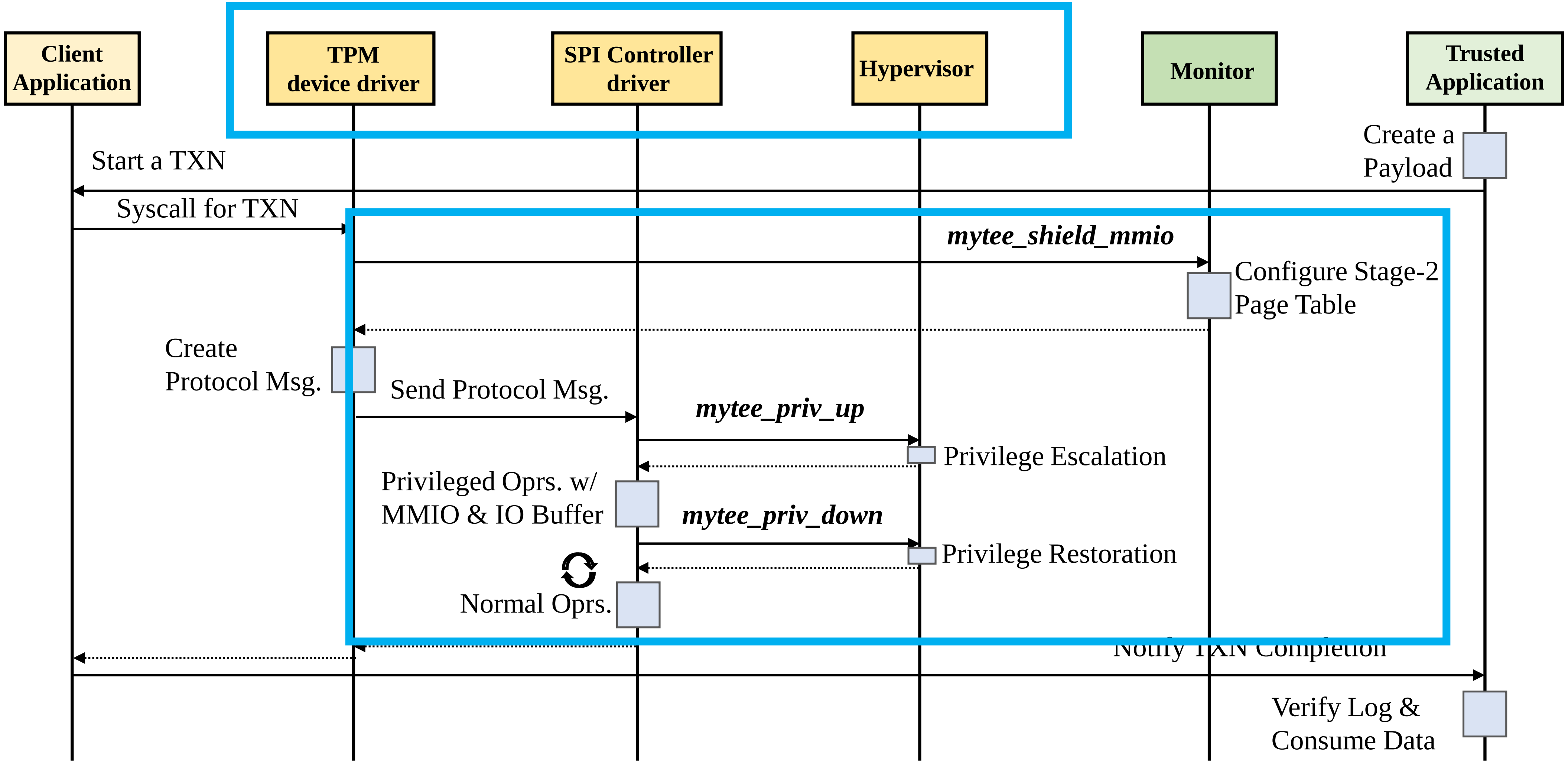
System Design: Secure IO cont'd

- Page tables are secured from the malicious privileged code
  - Stage-2 page tables are placed in the unmapped region from the OS and hypervisor
  - **Shared page tables** are **set to RO** so even the hypervisor cannot manipulated it



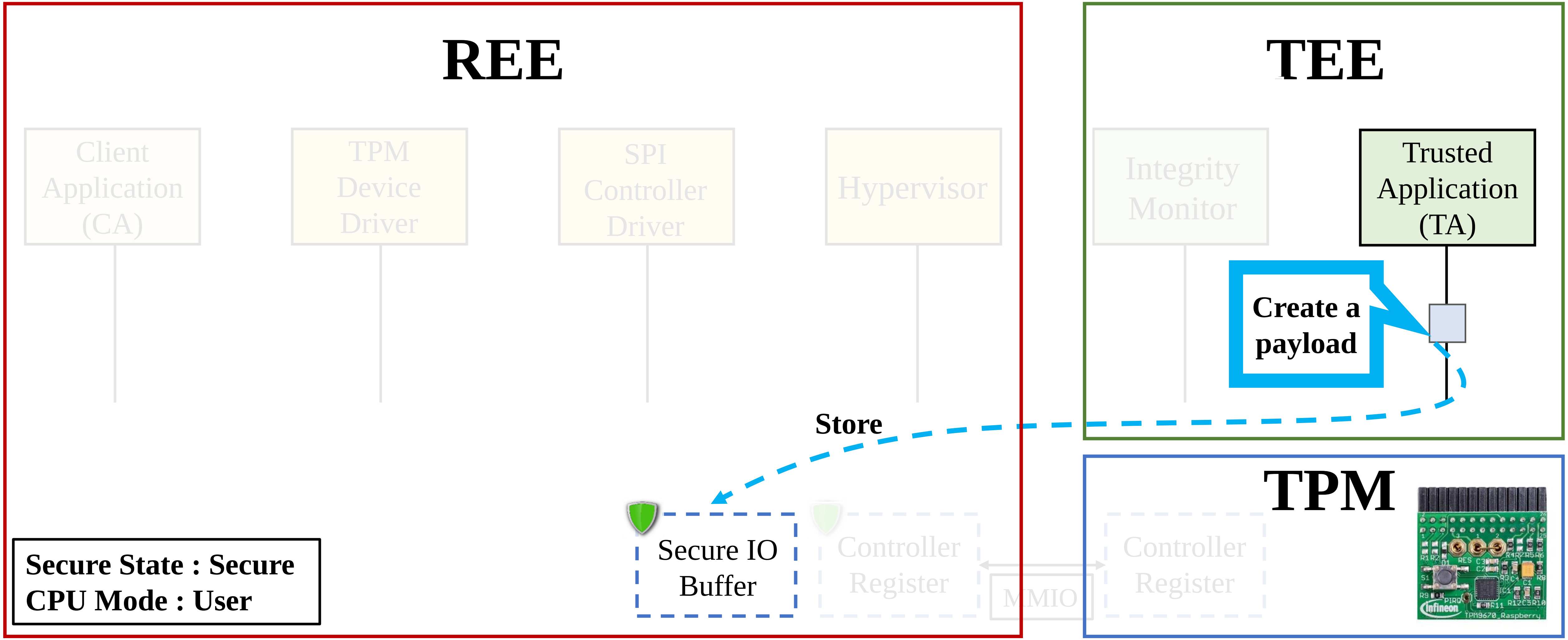
Secure IO Example: Overview

- Trusted TPM



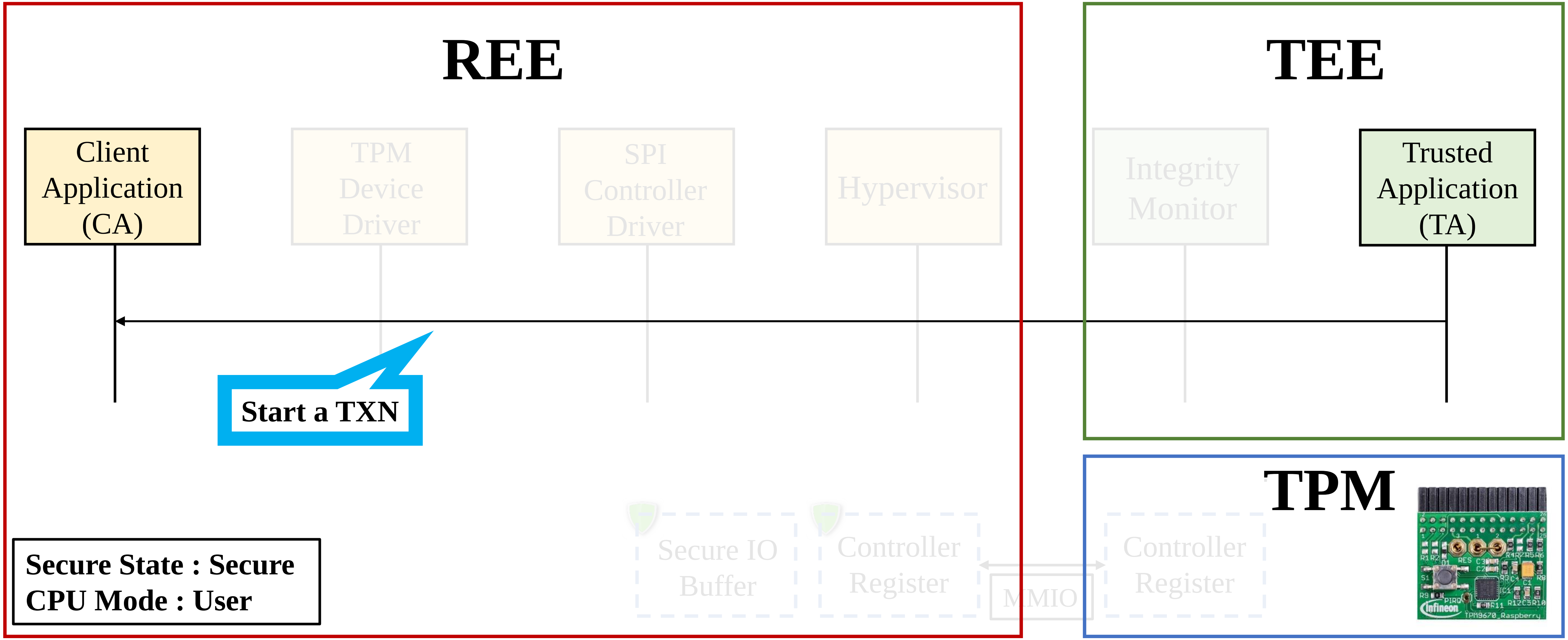
Secure IO Example

- Trusted TPM



Secure IO Example

- Trusted TPM

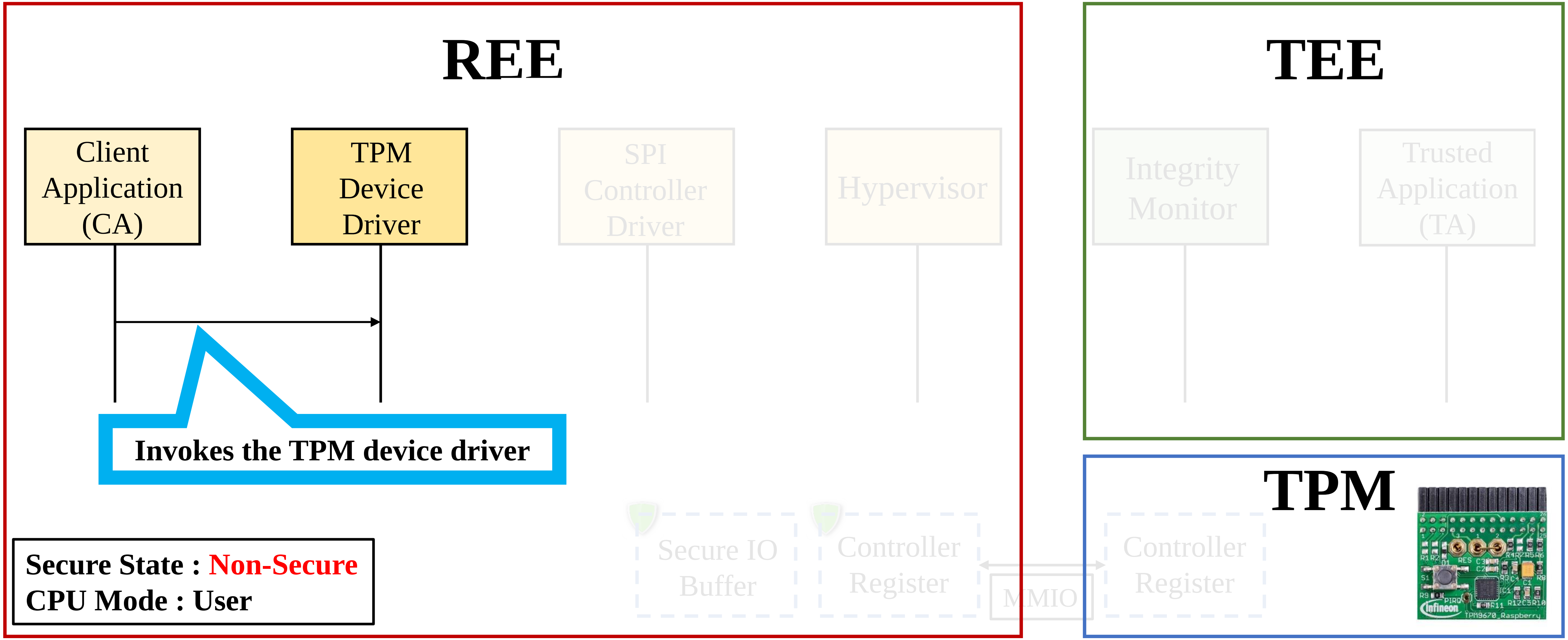


Start a TXN

Secure State : Secure
CPU Mode : User

Secure IO Example

- Trusted TPM



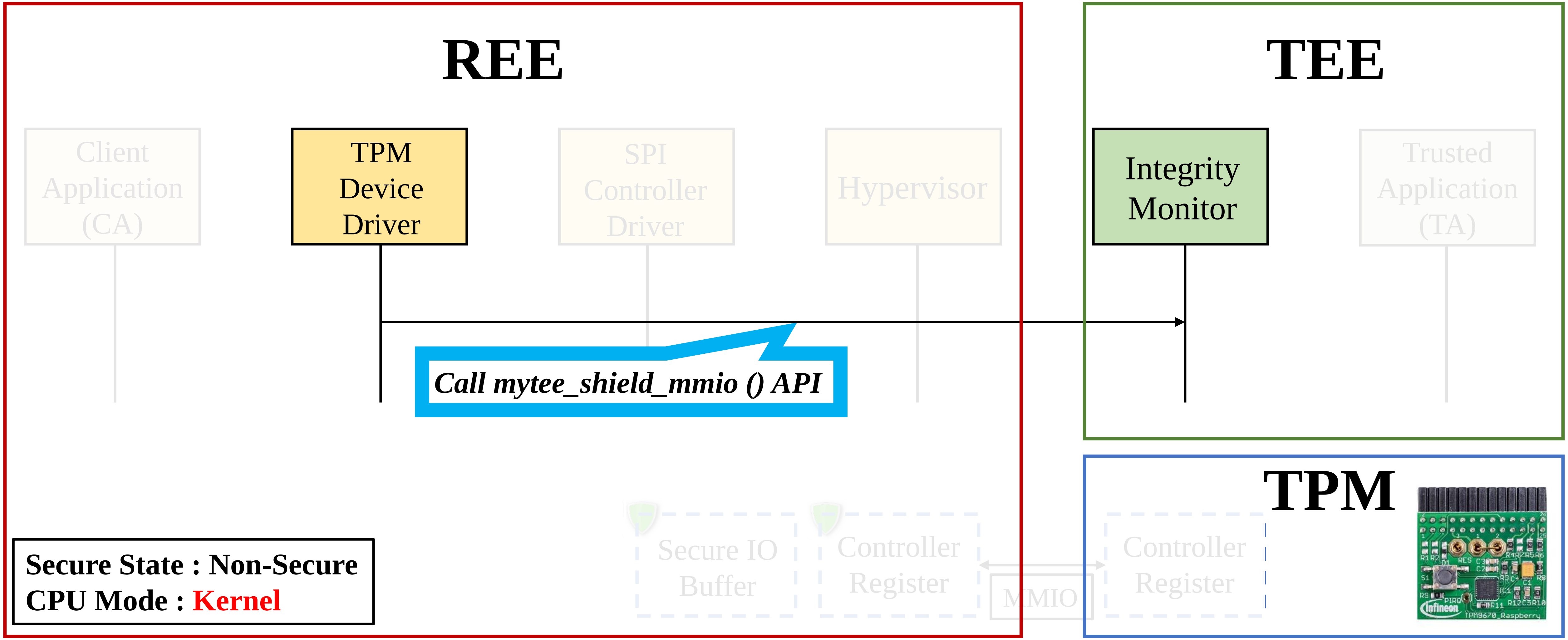
Invokes the TPM device driver

Secure State : Non-Secure

CPU Mode : User

Secure IO Example

- Trusted TPM

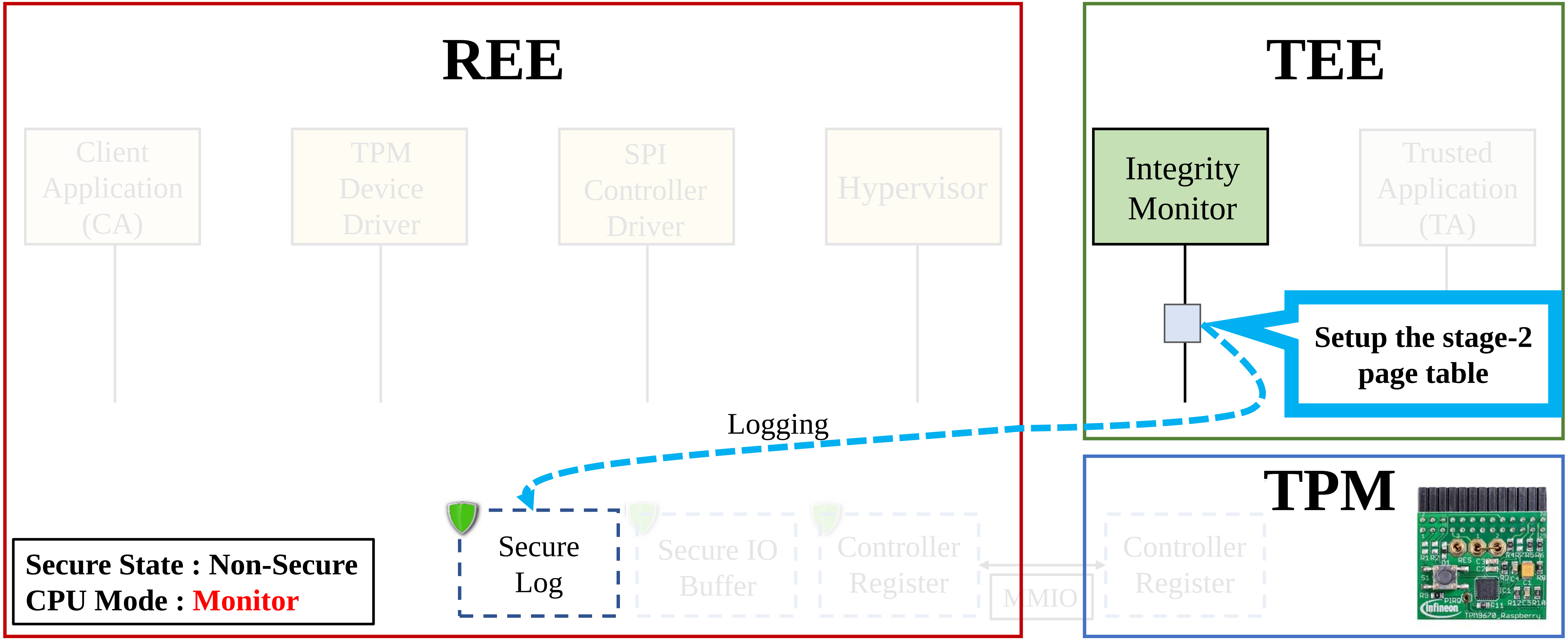


Call mytee_shield_mmio () API

Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

- Trusted TPM

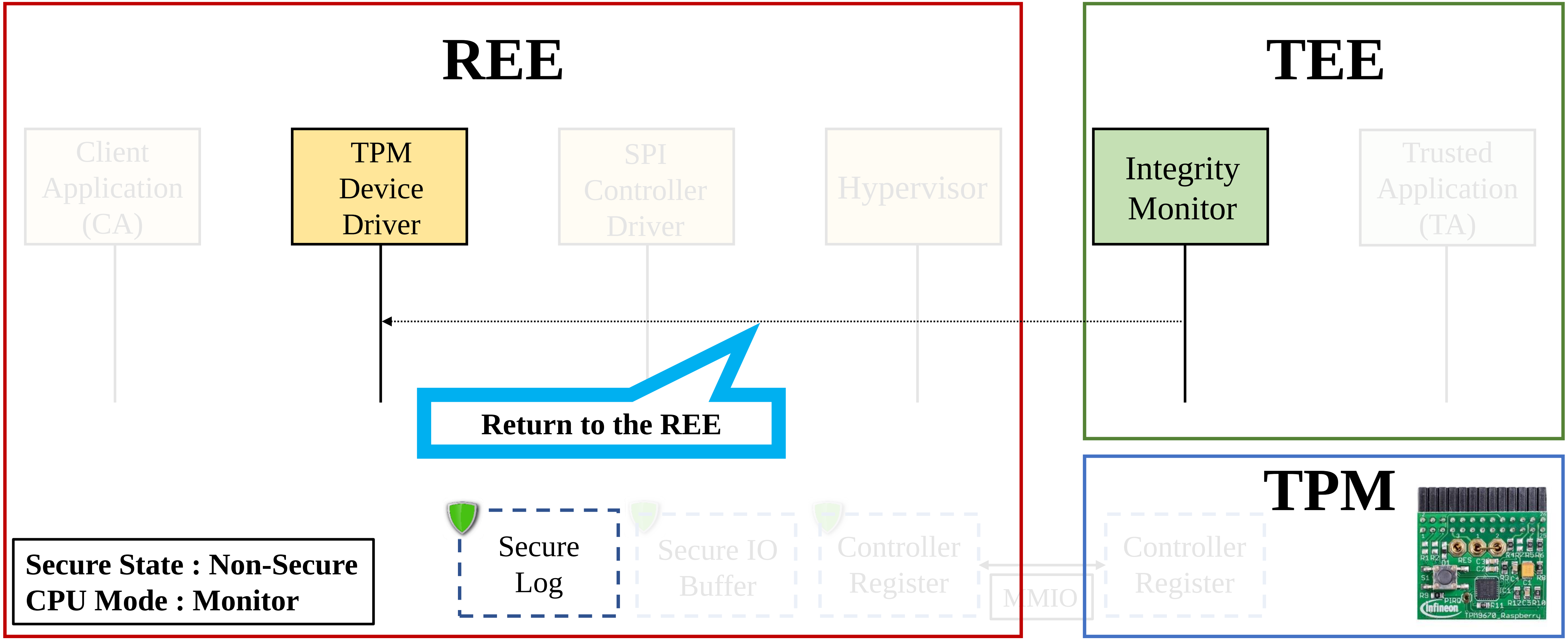


Setup the stage-2 page table

Secure State : Non-Secure
CPU Mode : Monitor

Secure IO Example

- Trusted TPM



REE

Client Application (CA)

TPM Device Driver

SPI Controller Driver

Hypervisor

Return to the REE

Secure Log

Secure IO Buffer

Controller Register

MMIO

Secure State : Non-Secure
CPU Mode : Monitor

TEE

Integrity Monitor

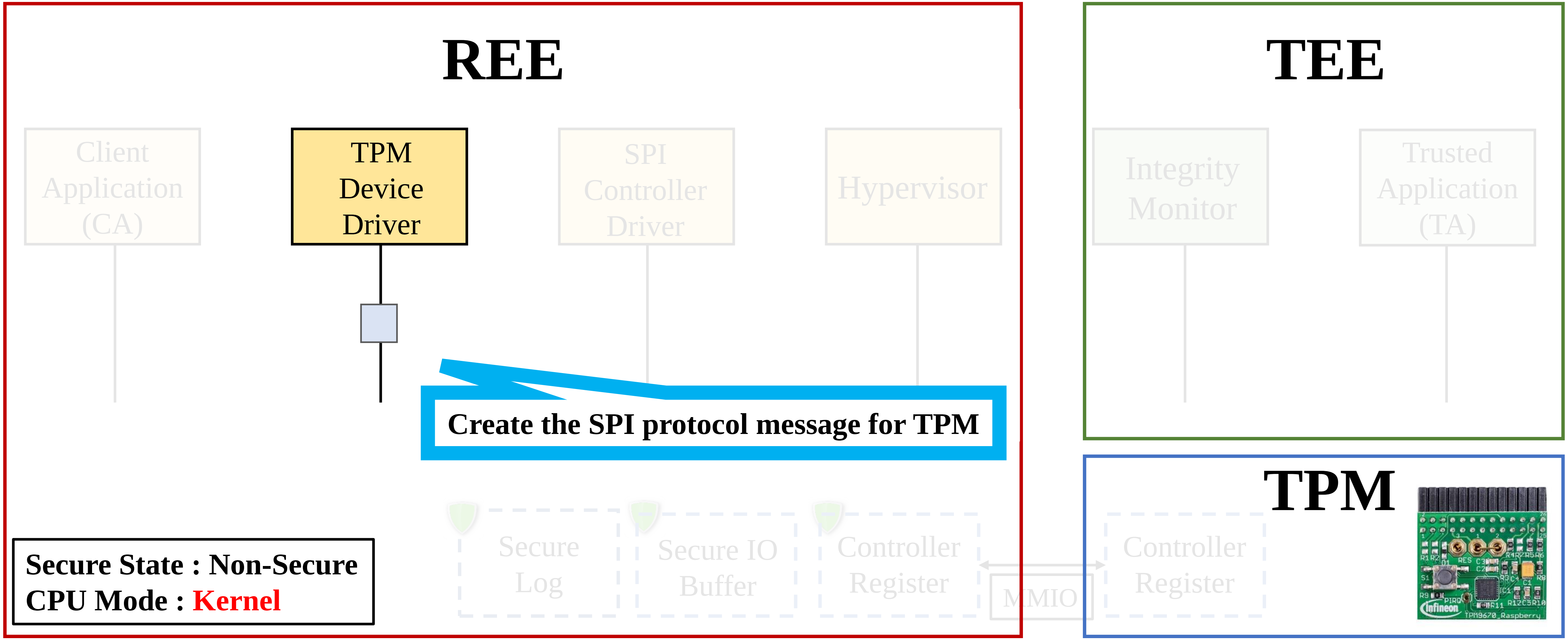
Trusted Application (TA)

TPM

Controller Register

Secure IO Example

- Trusted TPM

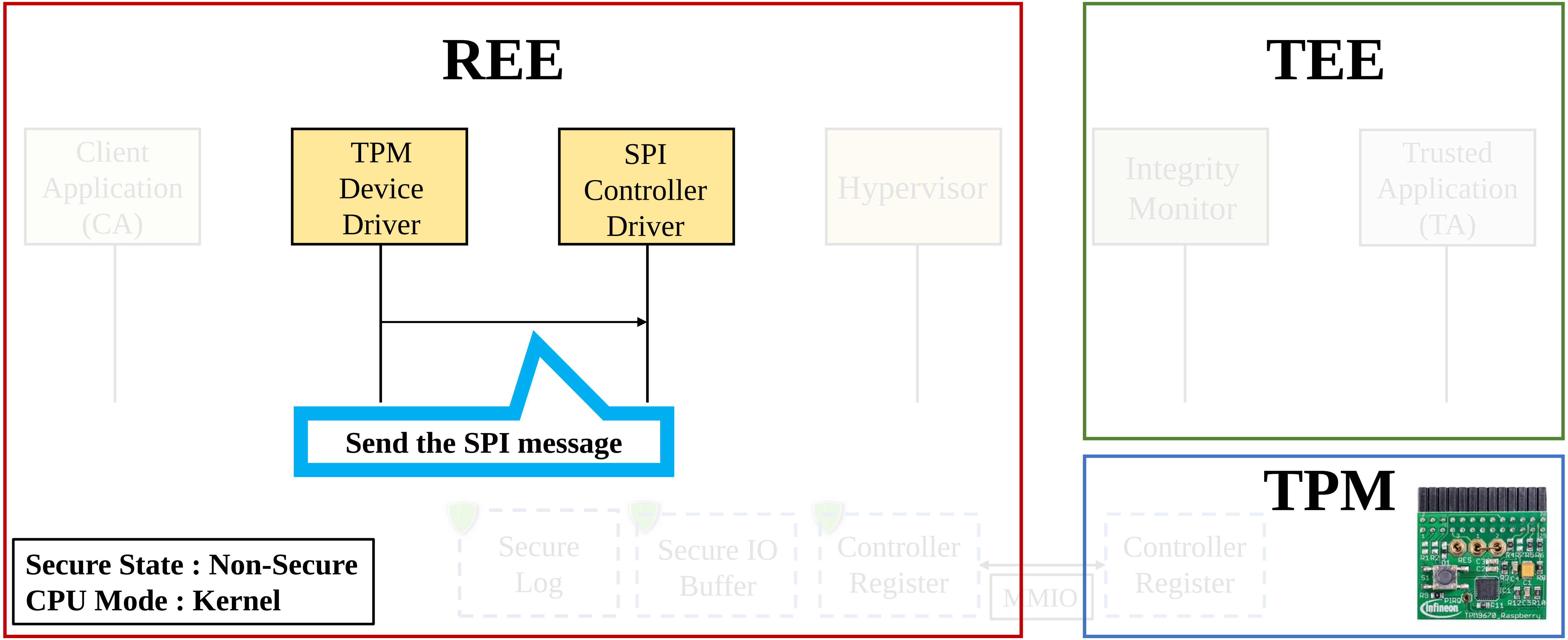


Create the SPI protocol message for TPM

Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

- Trusted TPM

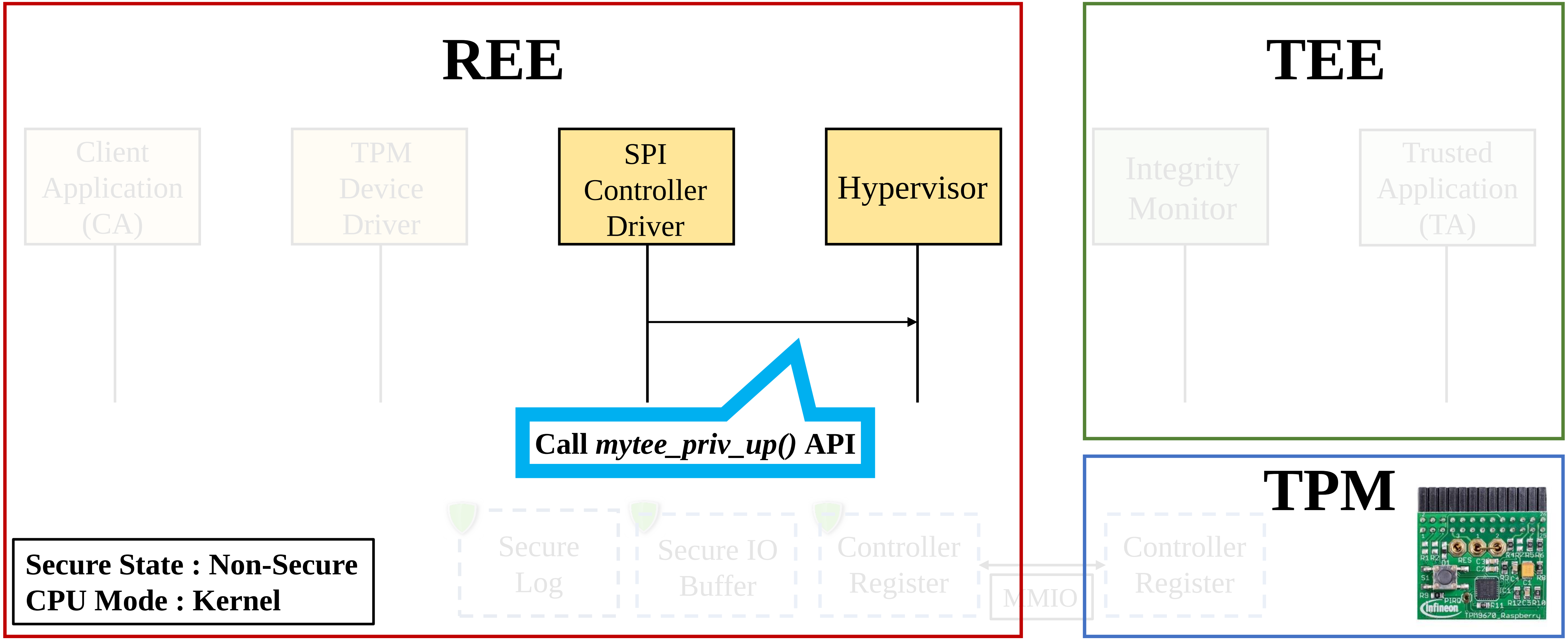


Send the SPI message

Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

- Trusted TPM



REE

Client Application (CA)

TPM Device Driver

SPI Controller Driver

Hypervisor

Call *mytee_priv_up()* API

Secure Log

Secure IO Buffer

Controller Register

MMIO

TEE

Integrity Monitor

Trusted Application (TA)

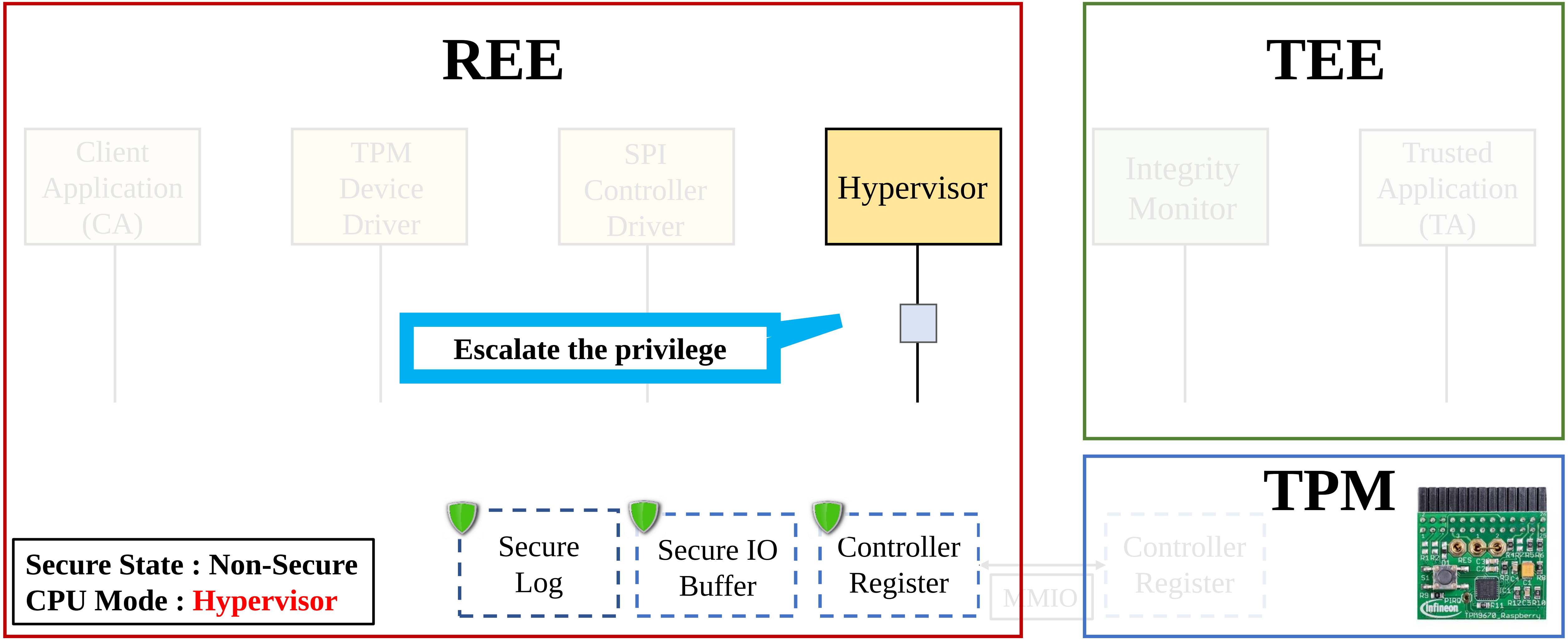
TPM

Controller Register

Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

- Trusted TPM



REE

Client Application (CA)

TPM Device Driver

SPI Controller Driver

Hypervisor

Escalate the privilege

Secure State : Non-Secure
CPU Mode : Hypervisor

Secure Log

Secure IO Buffer

Controller Register

MMIO

TEE

Integrity Monitor

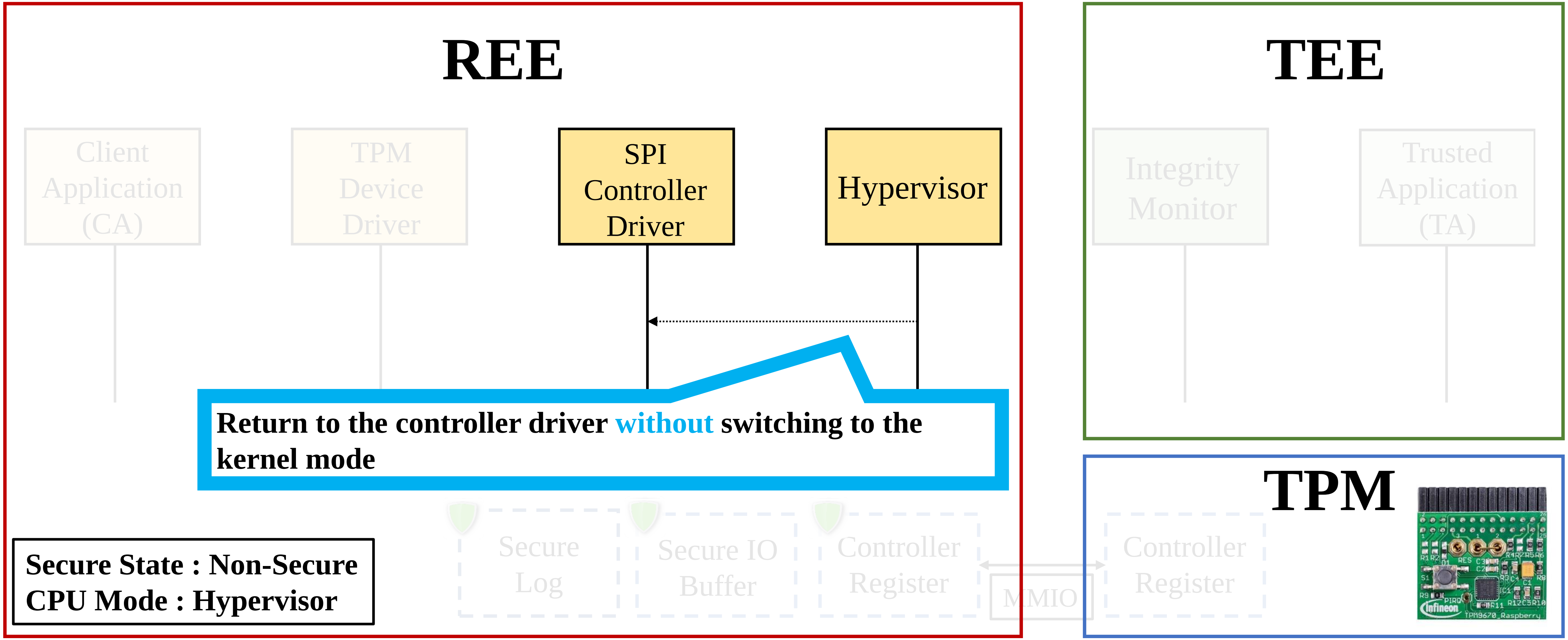
Trusted Application (TA)

TPM

Controller Register

Secure IO Example

- Trusted TPM

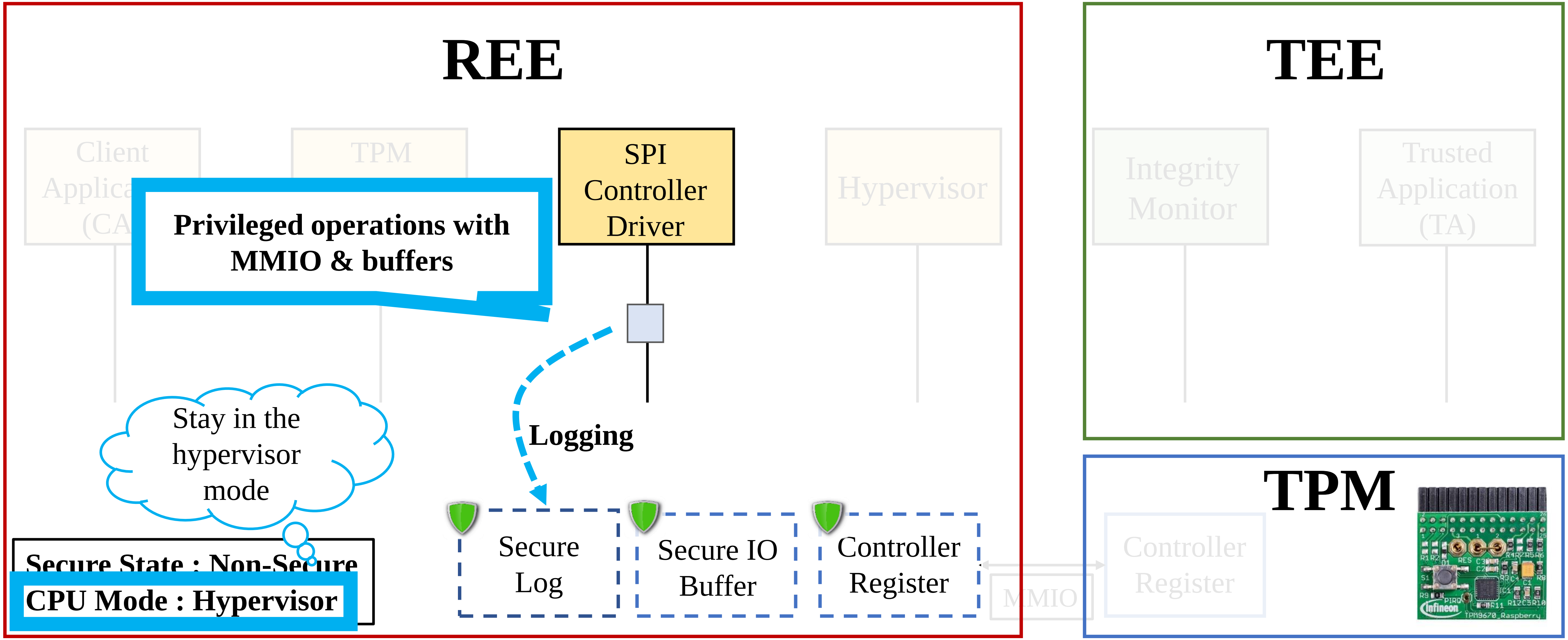


Return to the controller driver without switching to the kernel mode

Secure State : Non-Secure
CPU Mode : Hypervisor

Secure IO Example

- Trusted TPM



REE

Client Applica (CA

TPM

SPI Controller Driver

Hypervisor

Privileged operations with MMIO & buffers

Stay in the hypervisor mode

Logging

Secure Log

Secure IO Buffer

Controller Register

MMIO

Secure State : Non-Secure

CPU Mode : Hypervisor

TEE

Integrity Monitor

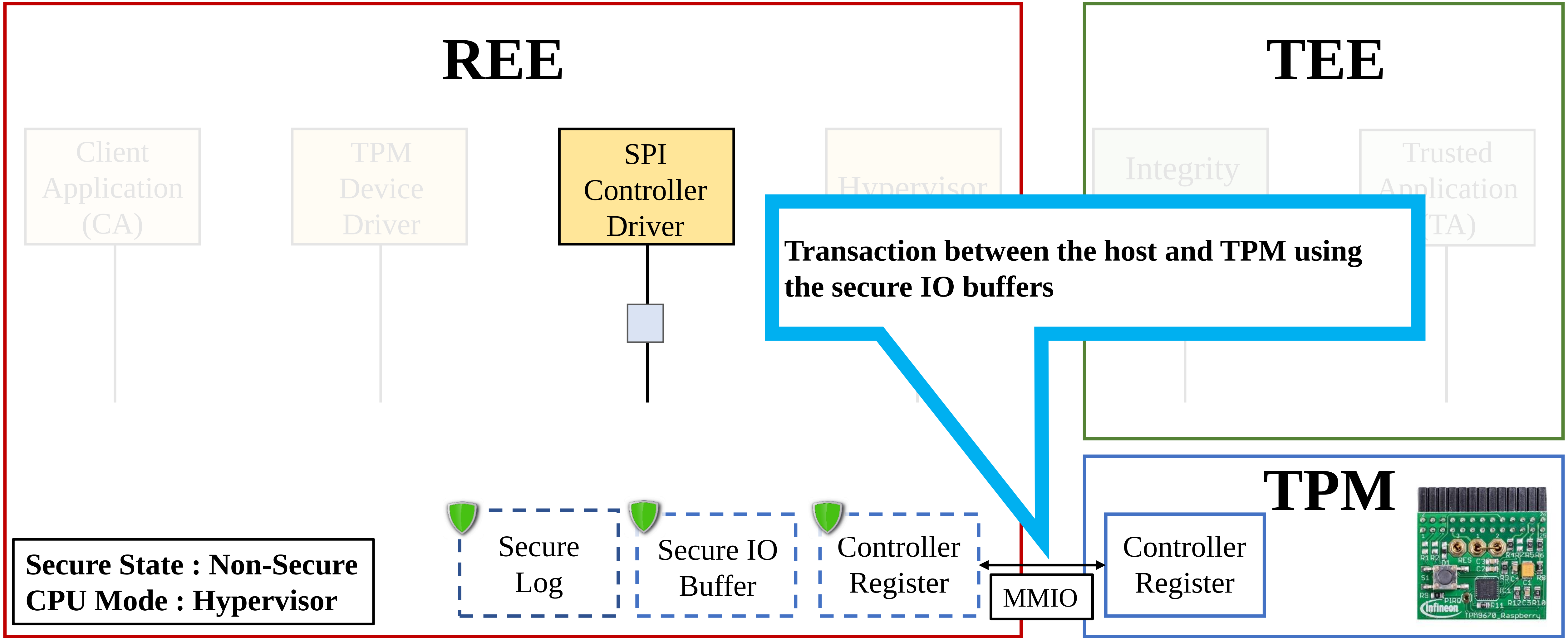
Trusted Application (TA)

TPM

Controller Register

Secure IO Example

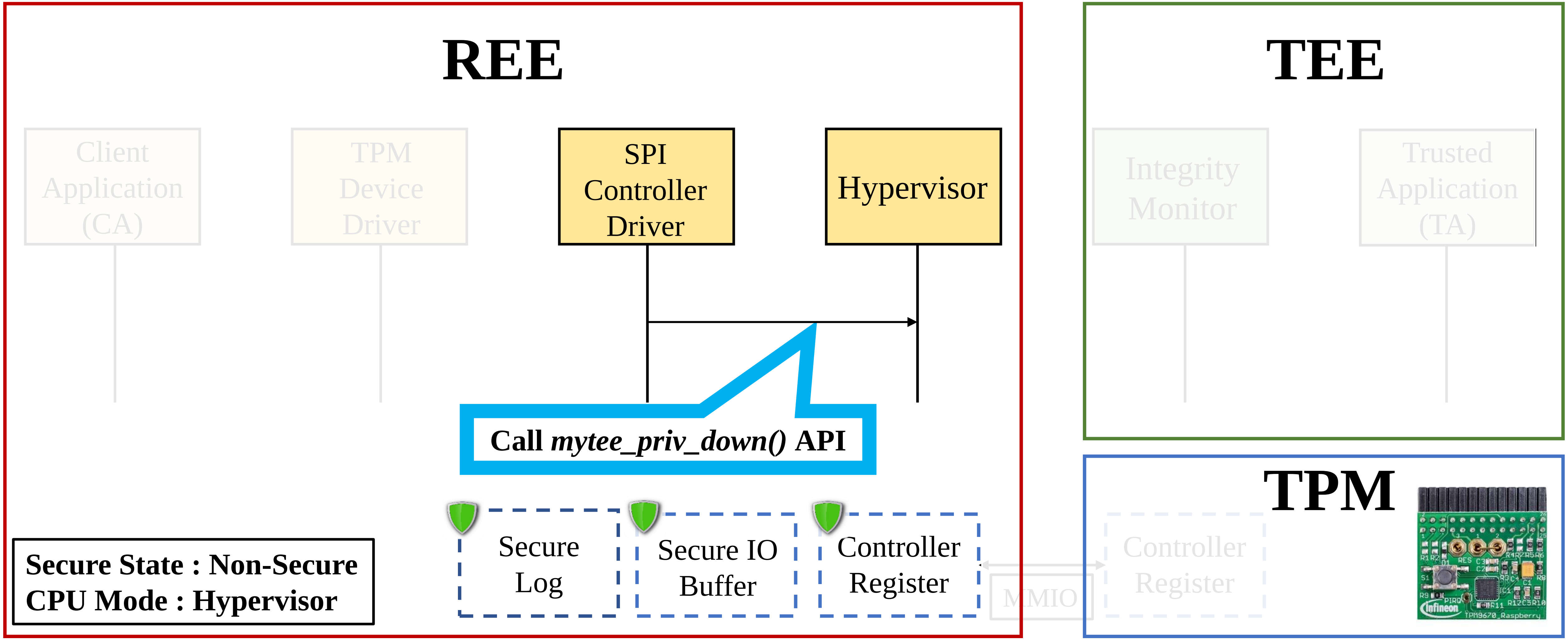
- Trusted TPM



Transaction between the host and TPM using the secure IO buffers

Secure IO Example

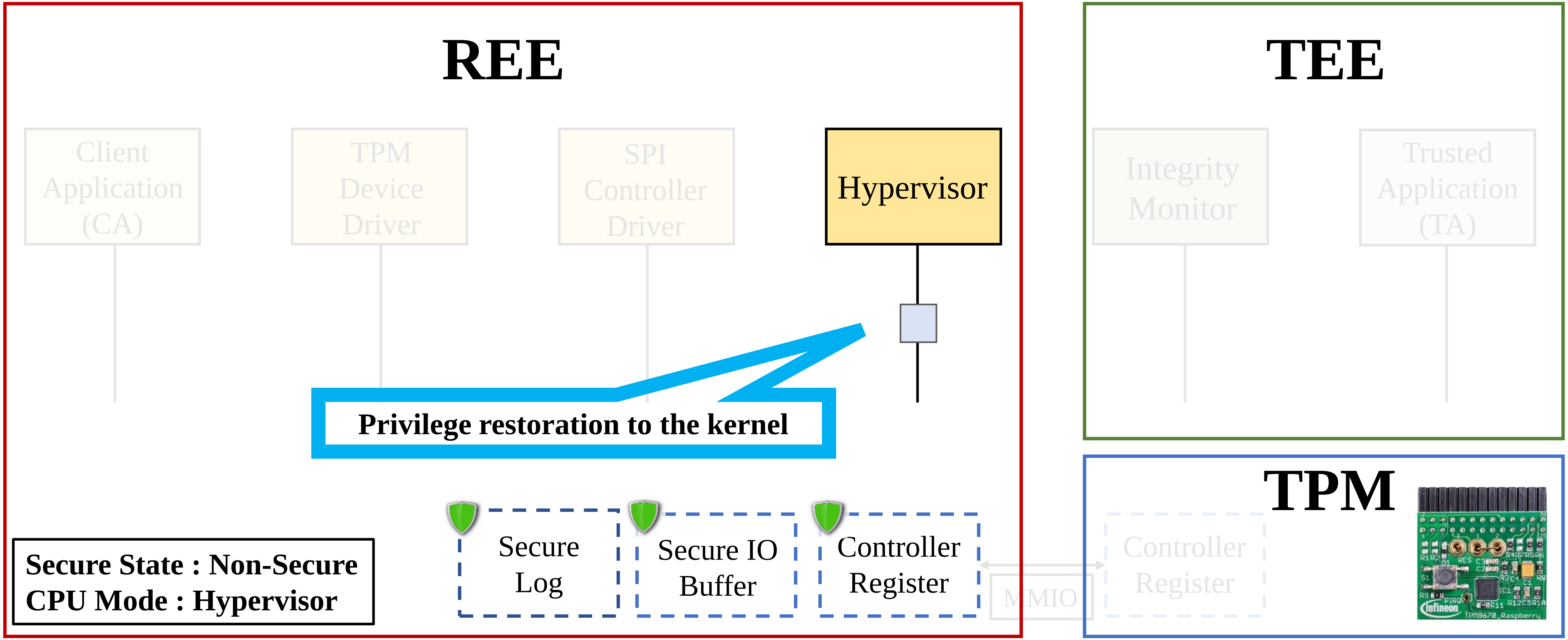
- Trusted TPM



Secure State : Non-Secure
CPU Mode : Hypervisor

Secure IO Example

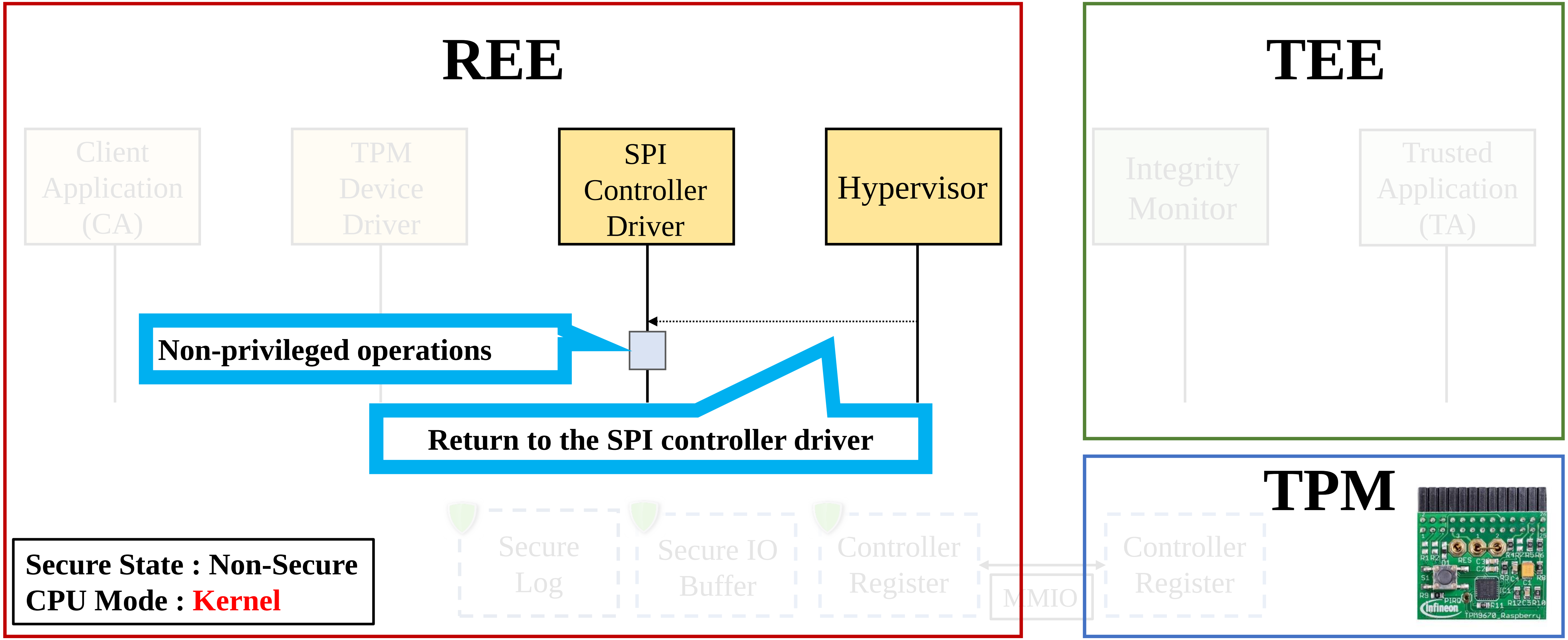
- Trusted TPM



Secure State : Non-Secure
CPU Mode : Hypervisor

Secure IO Example

- Trusted TPM



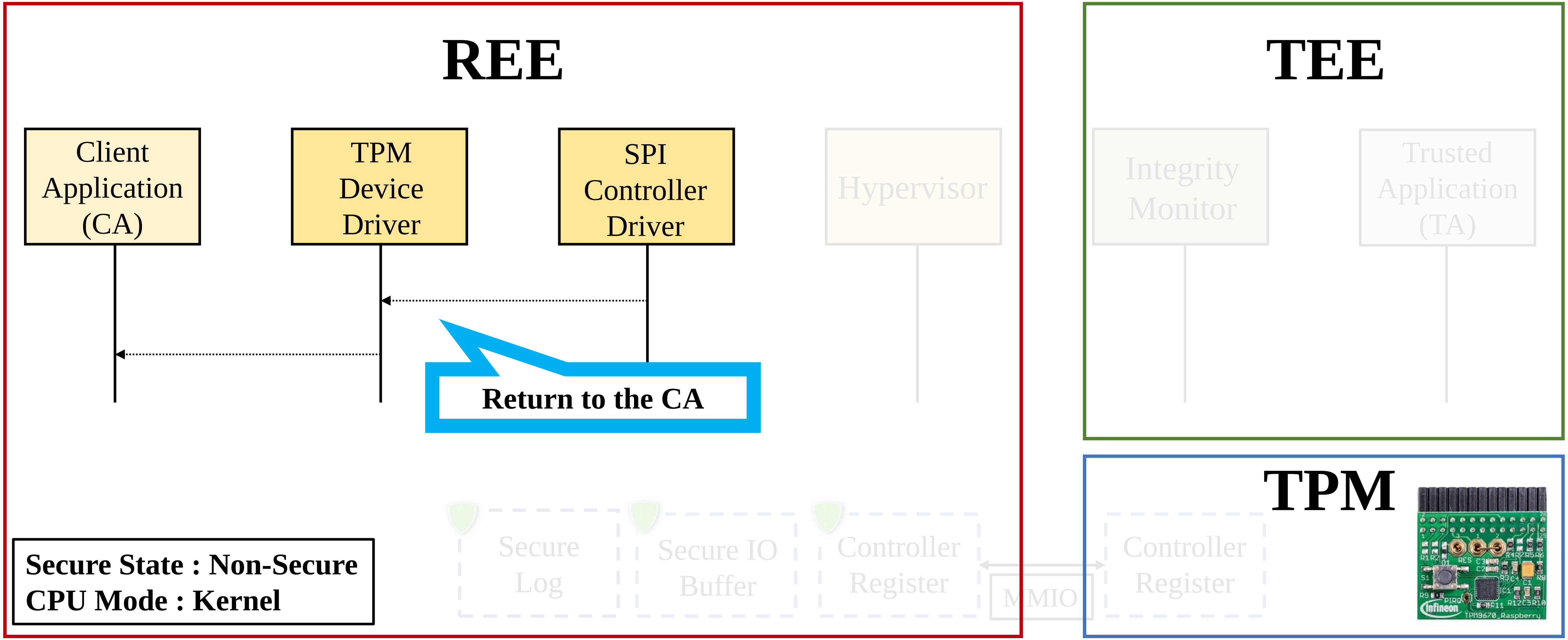
Non-privileged operations

Return to the SPI controller driver

Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

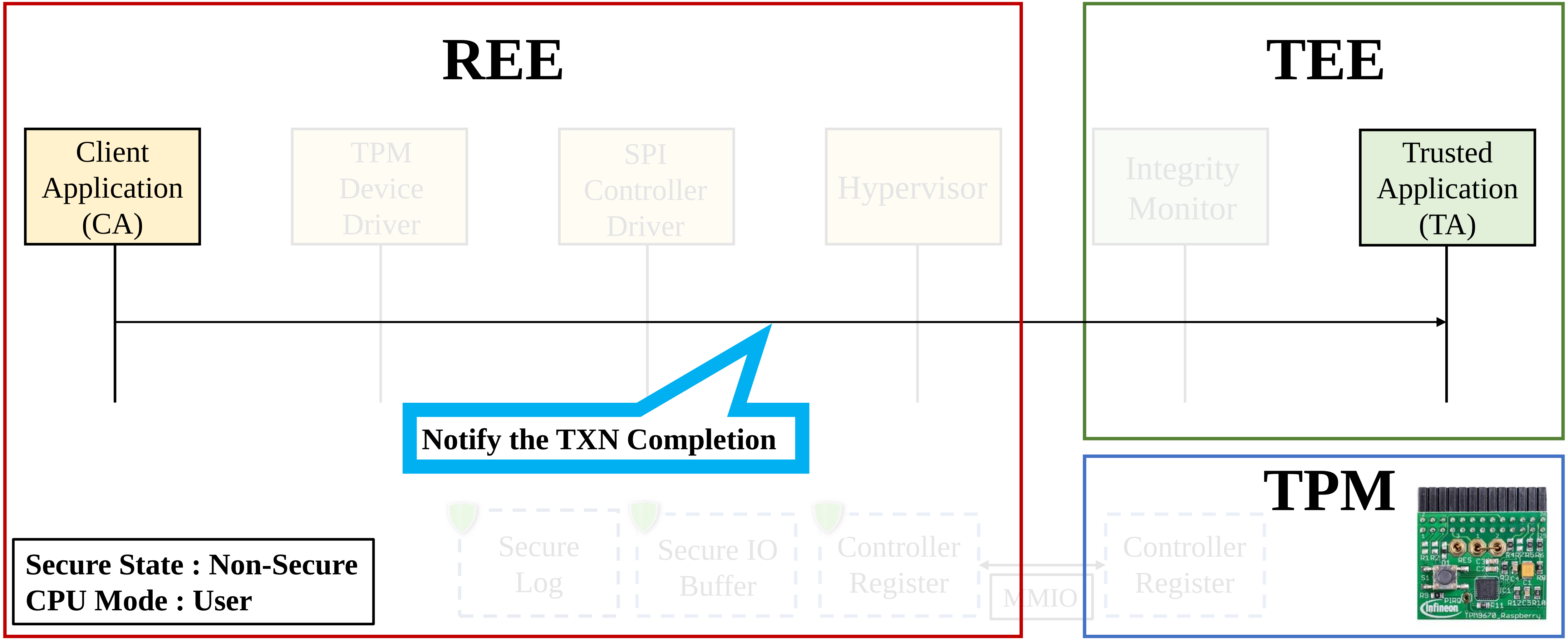
- Trusted TPM



Secure State : Non-Secure
CPU Mode : Kernel

Secure IO Example

- Trusted TPM

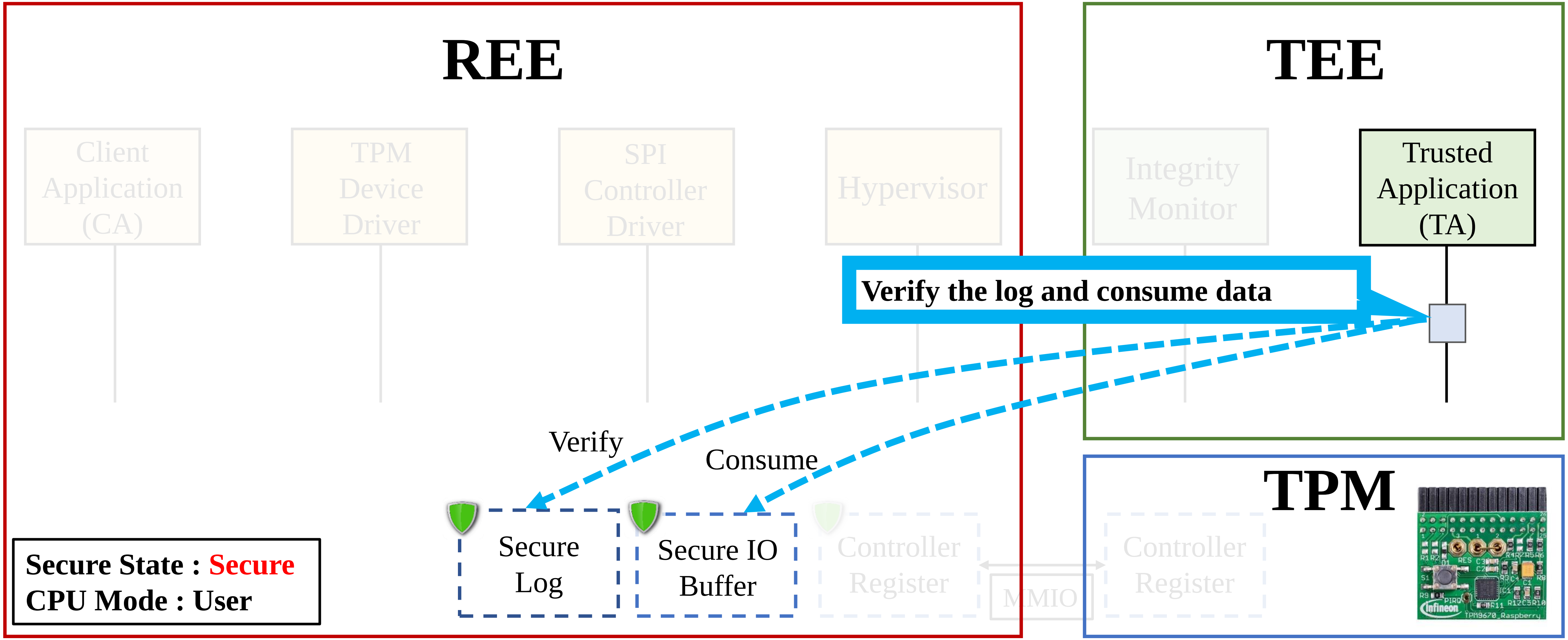


Notify the TXN Completion

Secure State : Non-Secure
CPU Mode : User

Secure IO Example

- Trusted TPM



Secure State : **Secure**
CPU Mode : User

Security Analysis

- Attack surfaces and defense mechanisms

Attack surface	Defense mechanism
TOCTOU attack against secure IO	Logging and verifying the payload
Abusing MyTEE APIs	Check the provenance of MyTEE API call
Malicious DMA	DMA Filter
Abusing the privilege-escalated memory operations	Verification of the memory operation range

Security Analysis

- Attack surfaces and defense mechanisms

Attack surface	Defense mechanism
TOCTOU attack against secure IO	Logging and verifying the payload
Abusing MyTEE APIs	Check the provenance of MyTEE API call
Malicious DMA	DMA Filter
Abusing the privilege-escalated memory operations	Verification of the memory operation range

Security Analysis

- Attack surfaces and defense mechanisms

Attack surface	Defense mechanism
TOCTOU attack against secure IO	Logging and verifying the payload
Abusing MyTEE APIs	Check the provenance of MyTEE API call
Malicious DMA	DMA Filter
Abusing the privilege-escalated memory operations	Verification of the memory operation range

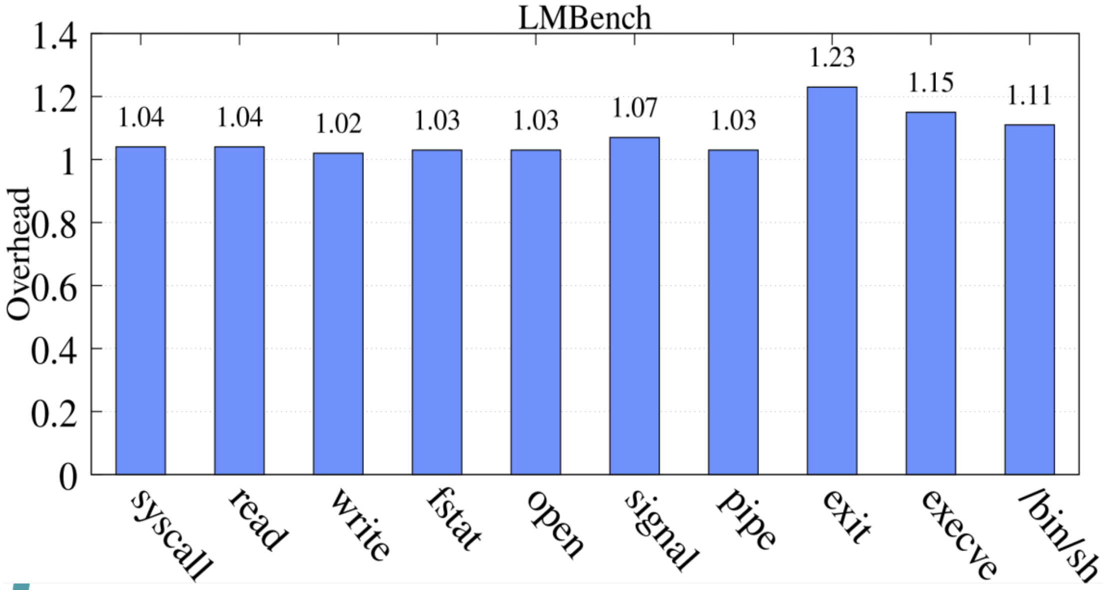
Security Analysis

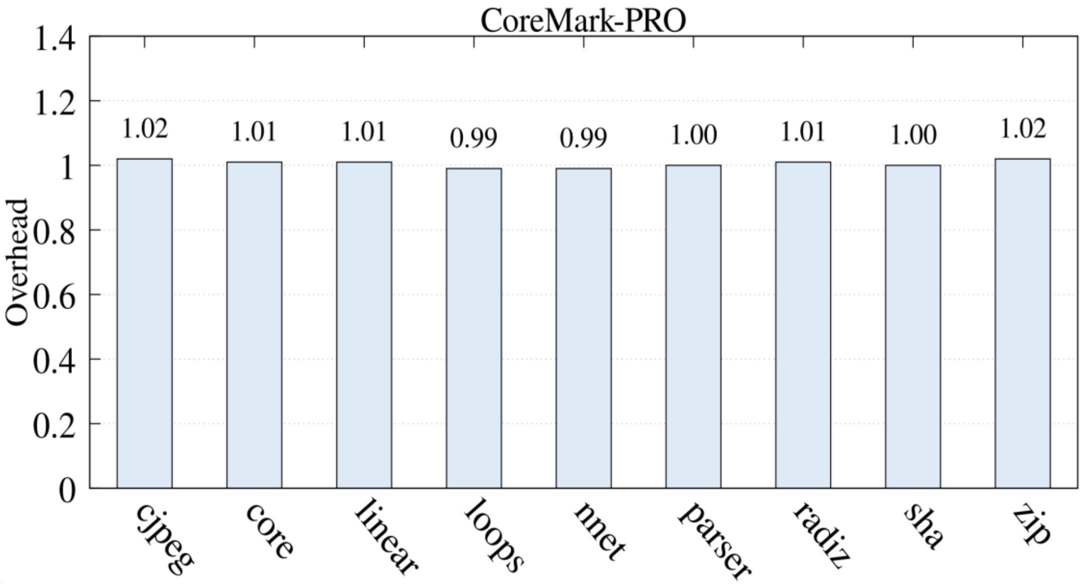
- Attack surfaces and defense mechanisms

Attack surface	Defense mechanism
TOCTOU attack against secure IO	Logging and verifying the payload
Abusing MyTEE APIs	Check the provenance of MyTEE API call
Malicious DMA	DMA Filter
Abusing the privilege-escalated memory operations	Verification of the memory operation range

Performance Evaluation

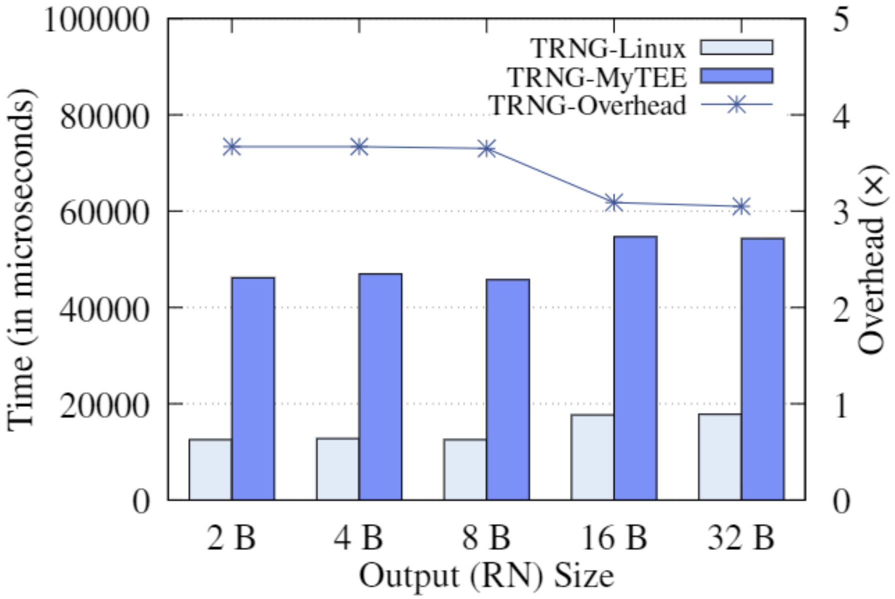
- LMBench
  - ✓ Measure the perf. of OS primitive operations
  - ✓ Maximum overhead of 23%
- CoreMark-PRO
  - ▪ CPU and memory benchmark
  - ✓ Negligible (1~2%)



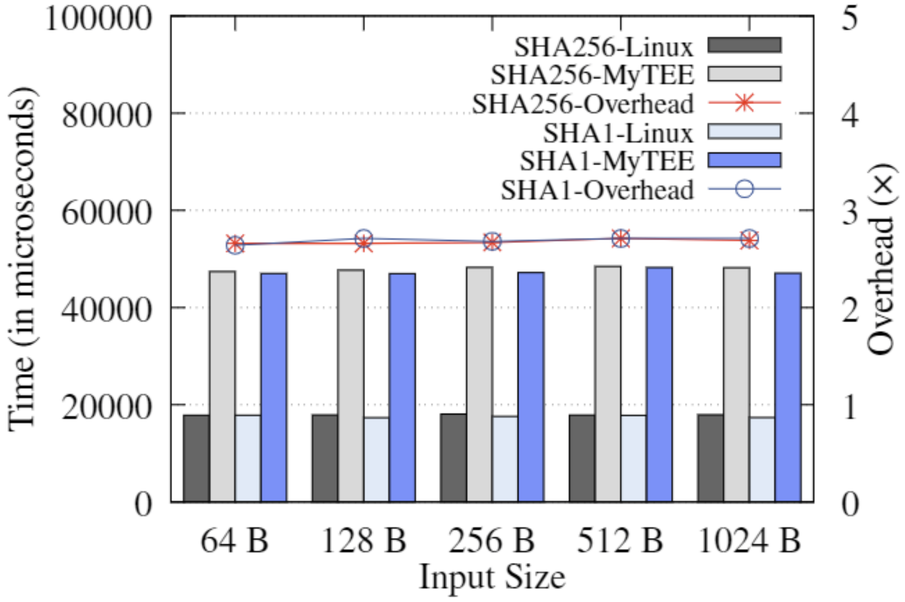


Performance Evaluation

- Trusted Applications – Secure TPM
  - ✓ Performance of random number generation and hashing



(a) TRNG



(b) Hashing

Conclusion

- MyTEE enables to build the TEE without depending on the TrustZone hardware extensions
  - ✓ Memory protection
  - ✓ Secure IO
- PoC implementation on Raspberry Pi 3
  - ✓ Three secure IO applications for the trusted keyboard, TPM, and framebuffer

Q & A

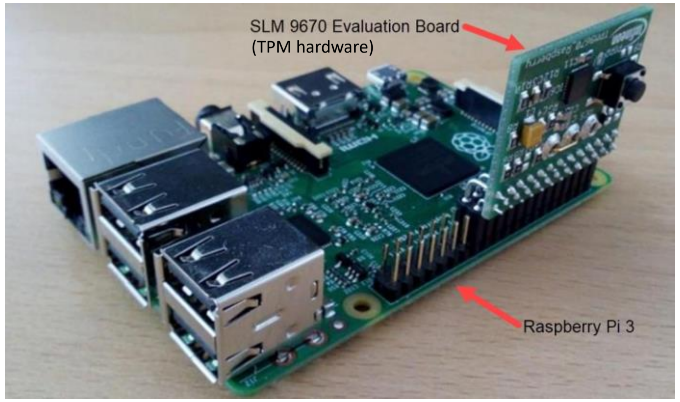
Thank you!

Q & A

Thank you!

Implementation

- Raspberry Pi 3+
  - ✓ Equipped with Broadcom BCM2837 SoC
- REE components
  - ✓ Linux 4.14
    - ➢ Deprivileged kernel and instrumented device drivers
    - ➢ Tiny-hypervisor with the DMA filter and MyTEE services
- TEE components
  - ✓ OP-TEE
    - ➢ Secure IO Applications: TPM, keyboard and frame Buffer
  - ✓ Trusted Firmware
    - ➢ Active integrity monitor in the monitor mode (TZ-RKP)

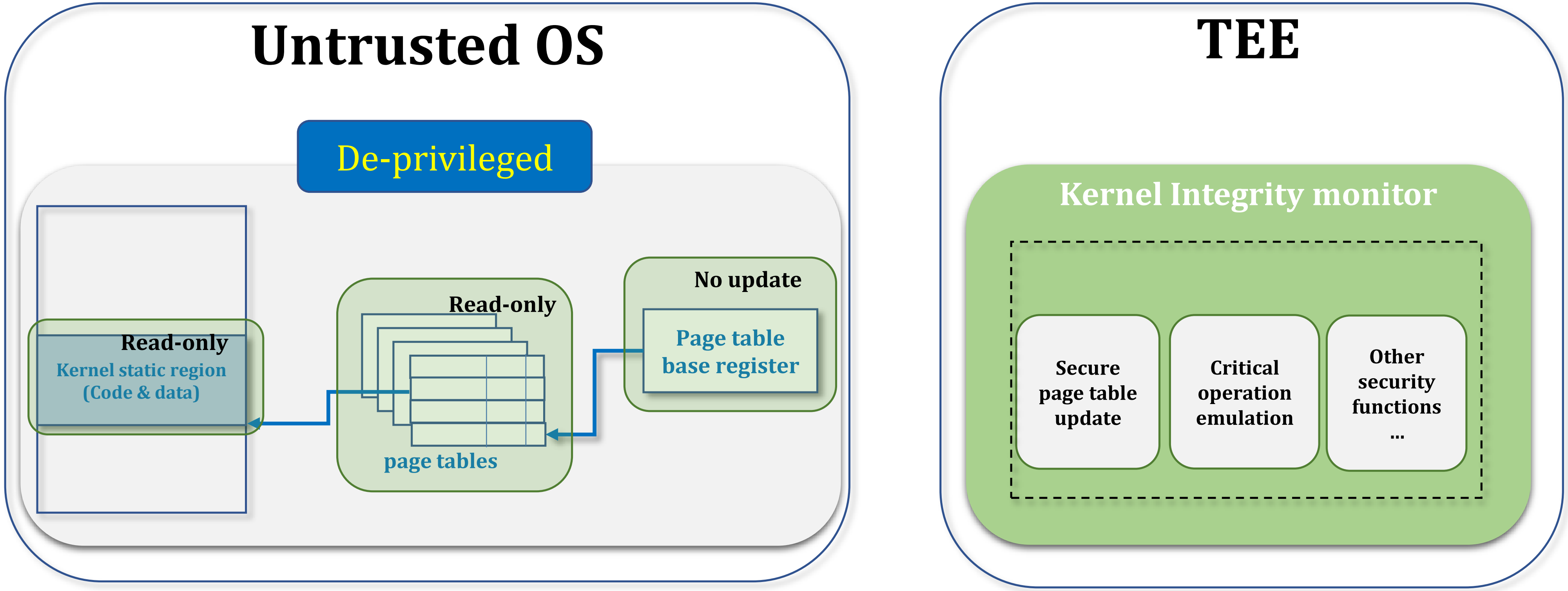


https://www.mouser.kr/new/infineon/infineon-slm9670-eval-board/

OP-TEE

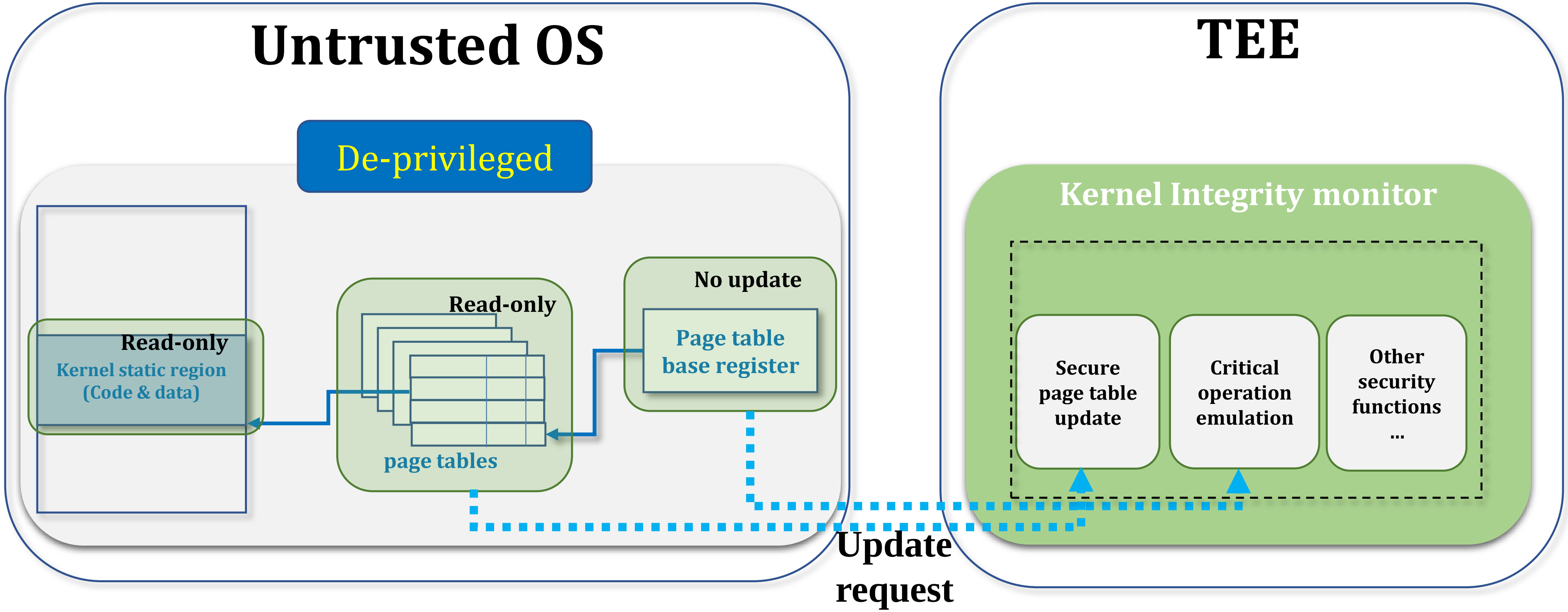
Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)



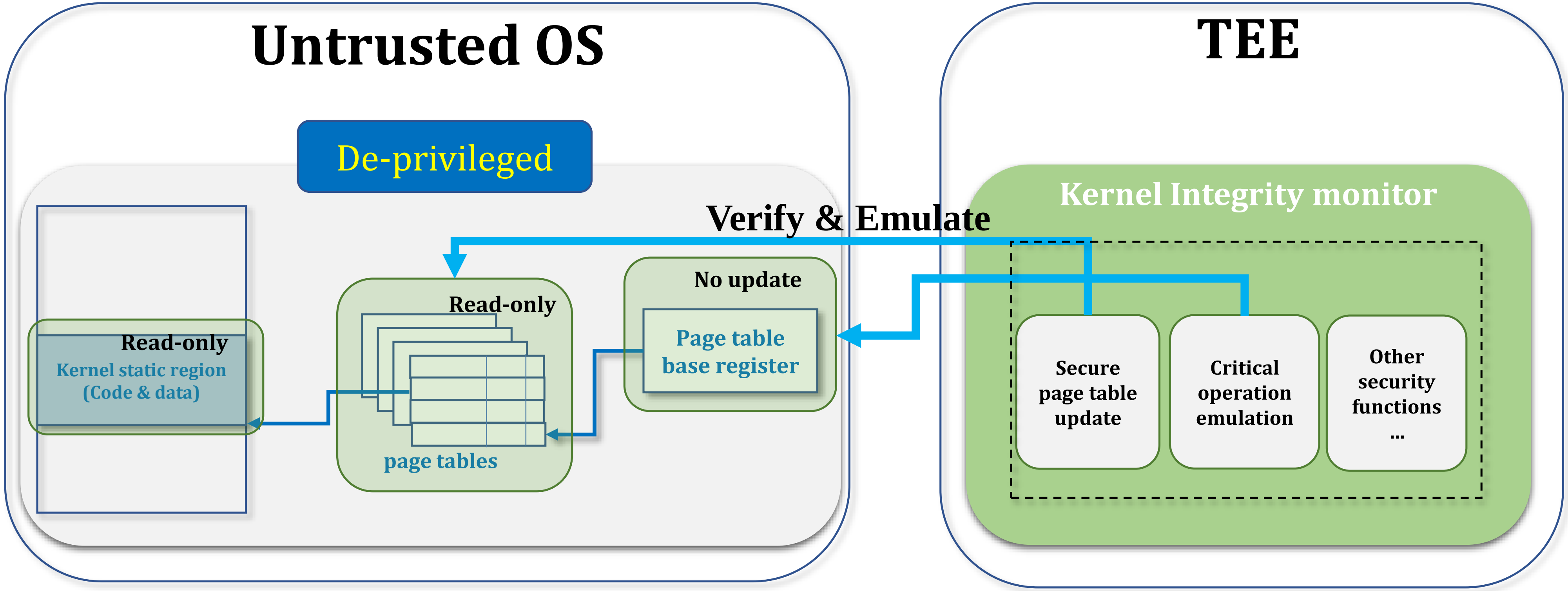
Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)



Background

- RKP (Real-time kernel protection)
  - ✓ Deprivilege the untrusted OS
  - ✓ Verify and emulate security critical operations (e.g., page table update)


Untrusted OS
TEE
De-privileged
Verify & Emulate
Kernel Integrity monitor
Read-only
Kernel static region
(Code & data)
Read-only
page tables
No update
Page table
base register
Secure
page table
update
Critical
operation
emulation
Other
security
functions
…