



TUESDAY, SEPTEMBER 23, 2025

ON THE ENFORCEMENT OF ATTESTABLE AVAILABILITY GUARANTEES FOR ARM-BASED INDUSTRIAL MULTI-TENANT REAL-TIME SYSTEMS

Edouard Michelin

Master's thesis

**ENGINEERED
TO OUTFIT**

Outline

1. Context
2. Background
3. Goals
4. Threat Model
5. Design
6. Implementation
7. Evaluation
8. Conclusion

Context

Industrial Control Systems



Context

Industrial Control Systems

Industry 4.0

Digitalization of OT

→ OT & IT convergence

- Automated control and protection
- Remote supervision and control

Context

Industrial Control Systems

Industry 4.0

Digitalization of OT

→ OT & IT convergence

- Automated control and protection
- Remote supervision and control

But:

- Increased attack surface
- Different requirements and objectives → security gaps

Context

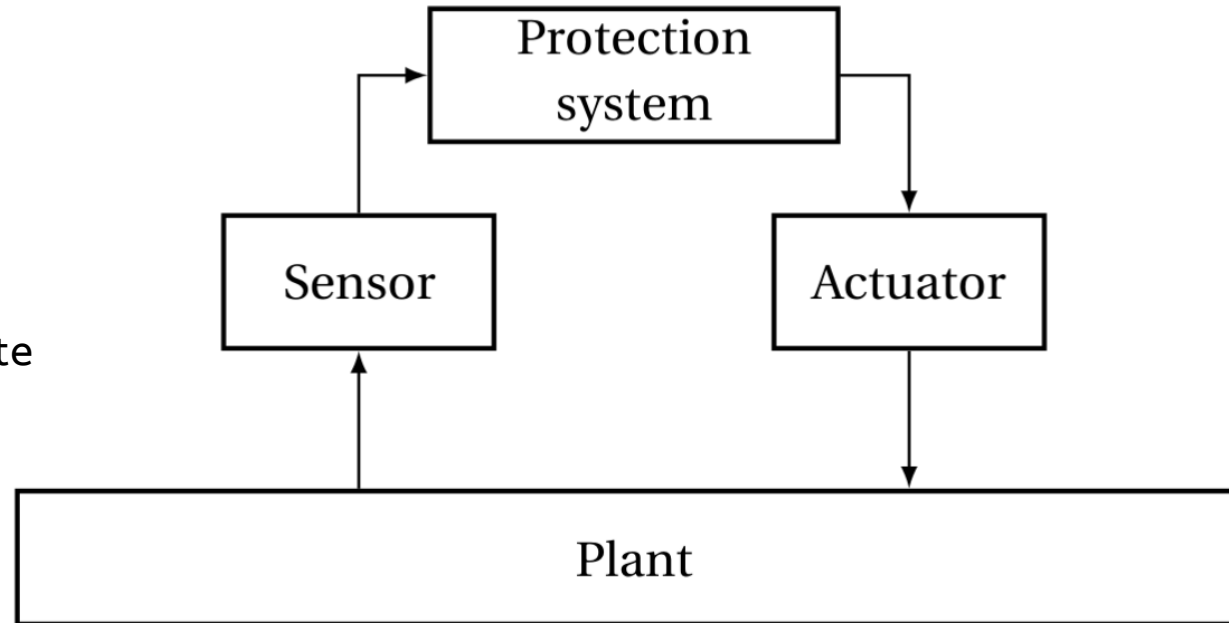
Industrial Control Systems

Control and Protection Systems

Control loop

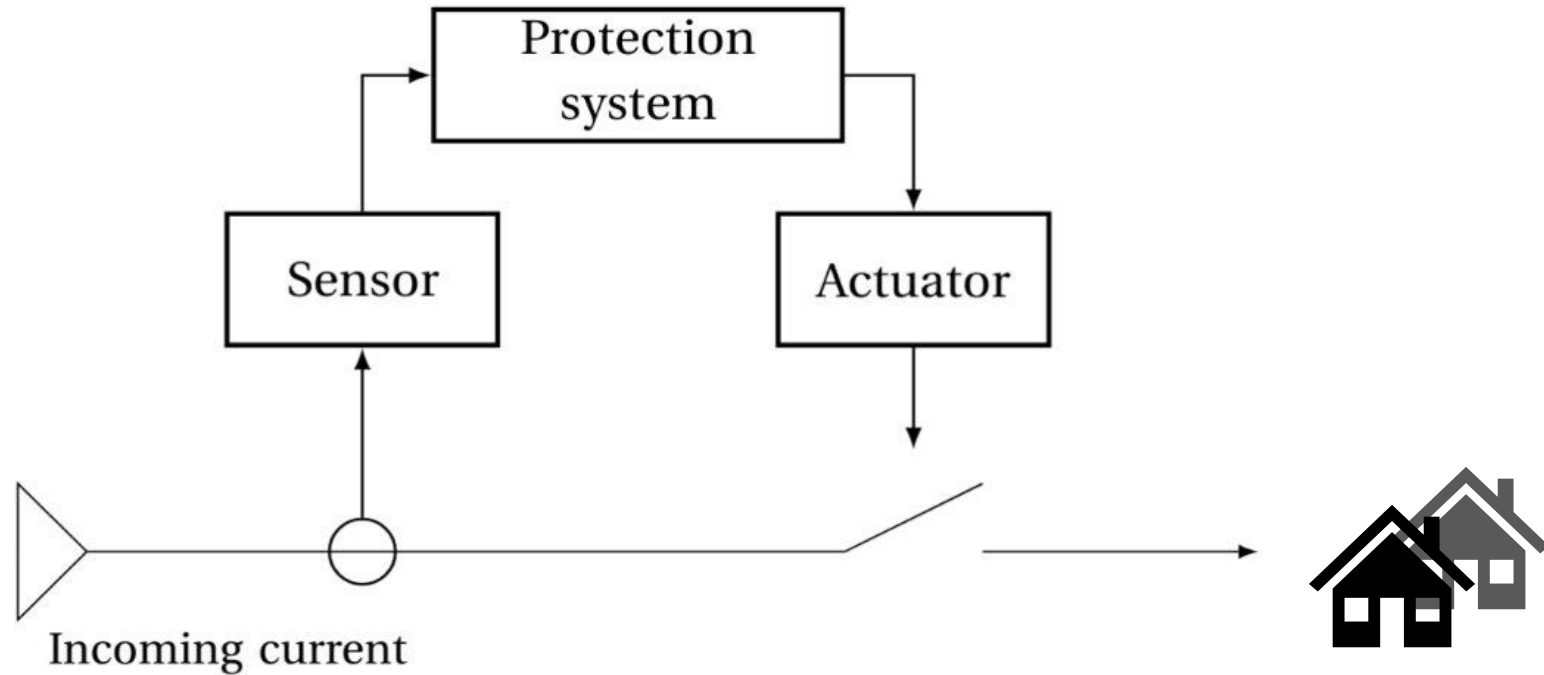
1. Sensing
2. Computing
3. Actuating

Every iteration must complete



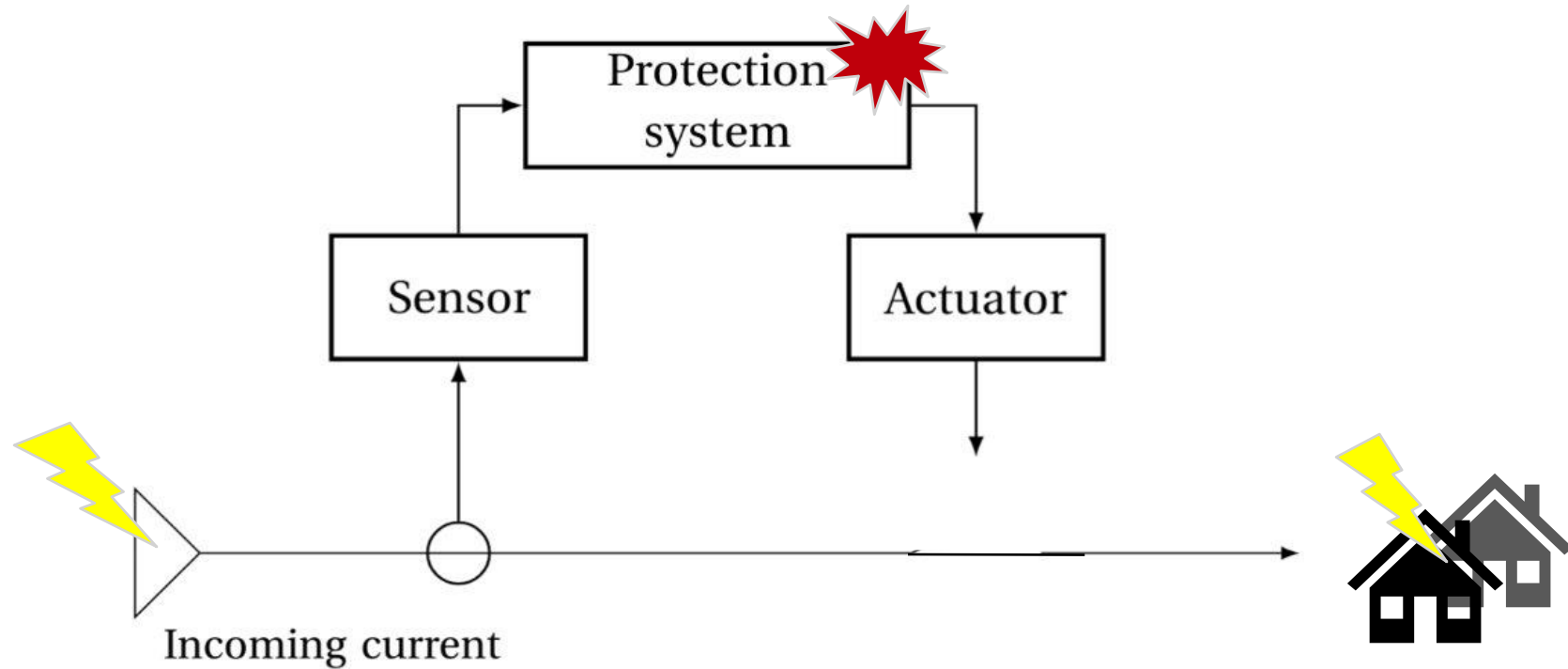
Context

Industrial Control Systems



Context

Industrial Control Systems

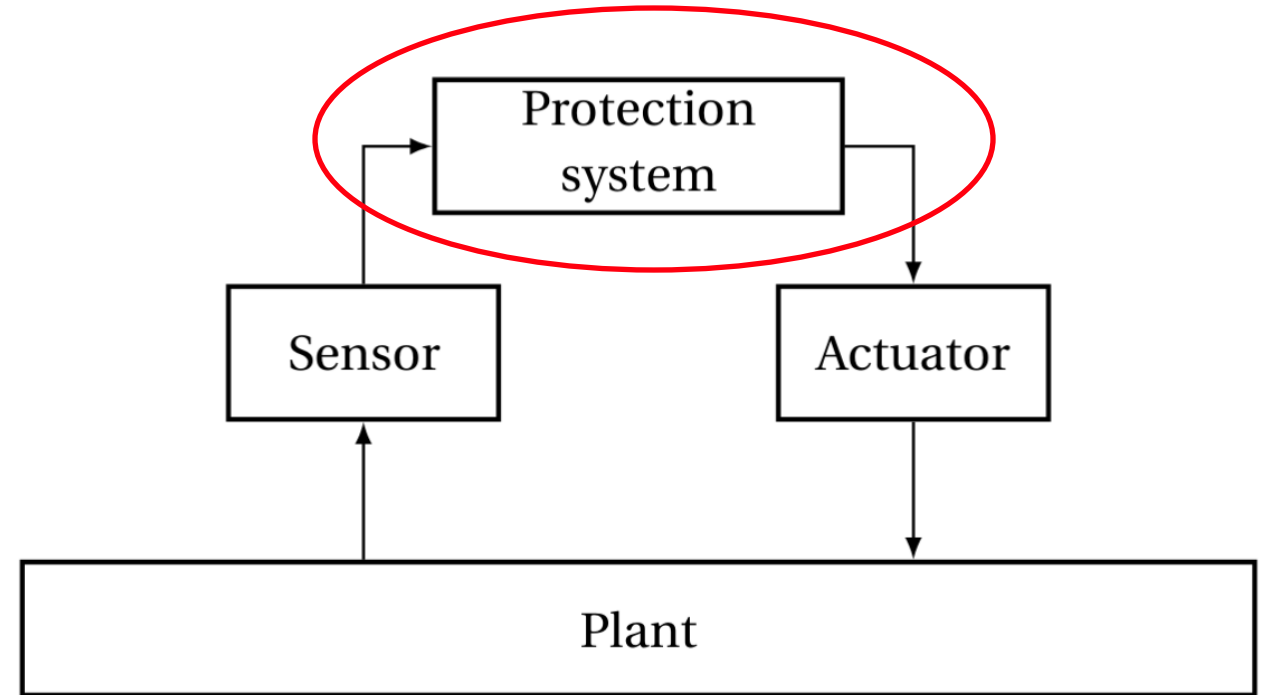


Context

Industrial Control Systems

Current Limitations

- Closed systems, one solution per device ✖
- Lack of portability across systems ✖



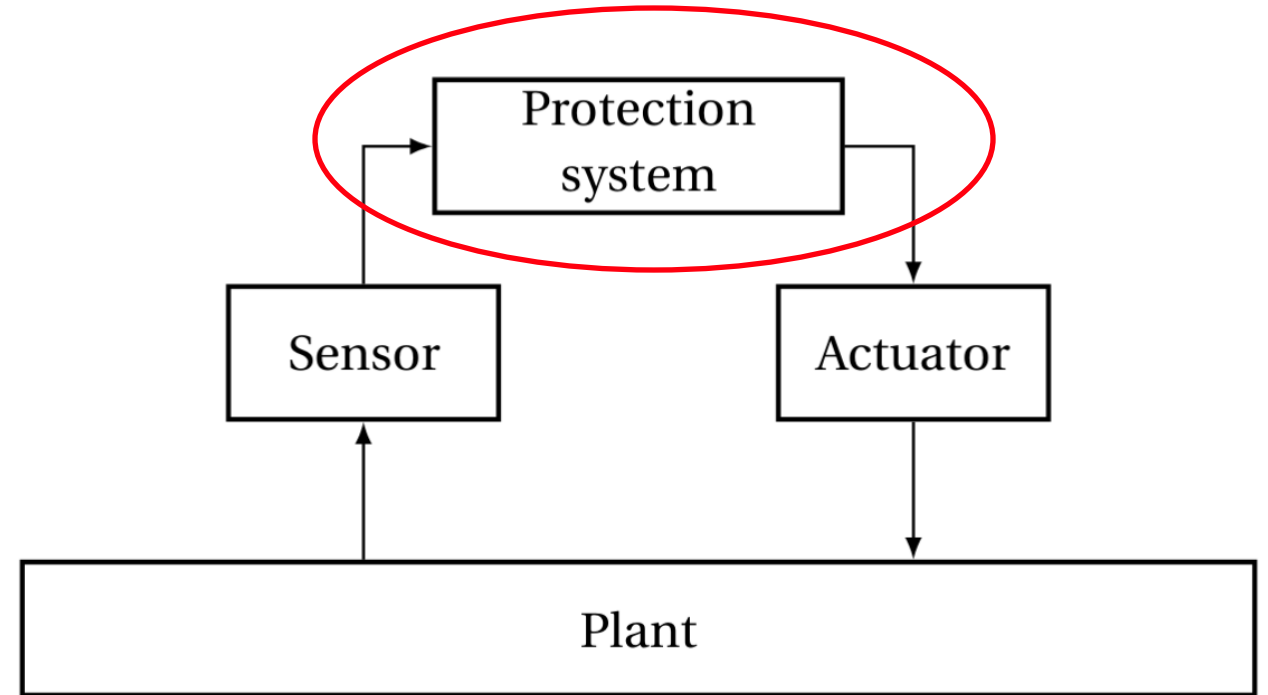
Context

Industrial Control Systems

Current Limitations

- Closed systems, one solution per device ✖
- Lack of portability across systems ✖

- Provider-centric approach
- All these solutions are costly
- Lack of customization
- Hard to maintain over time
- Custom hardware explosion



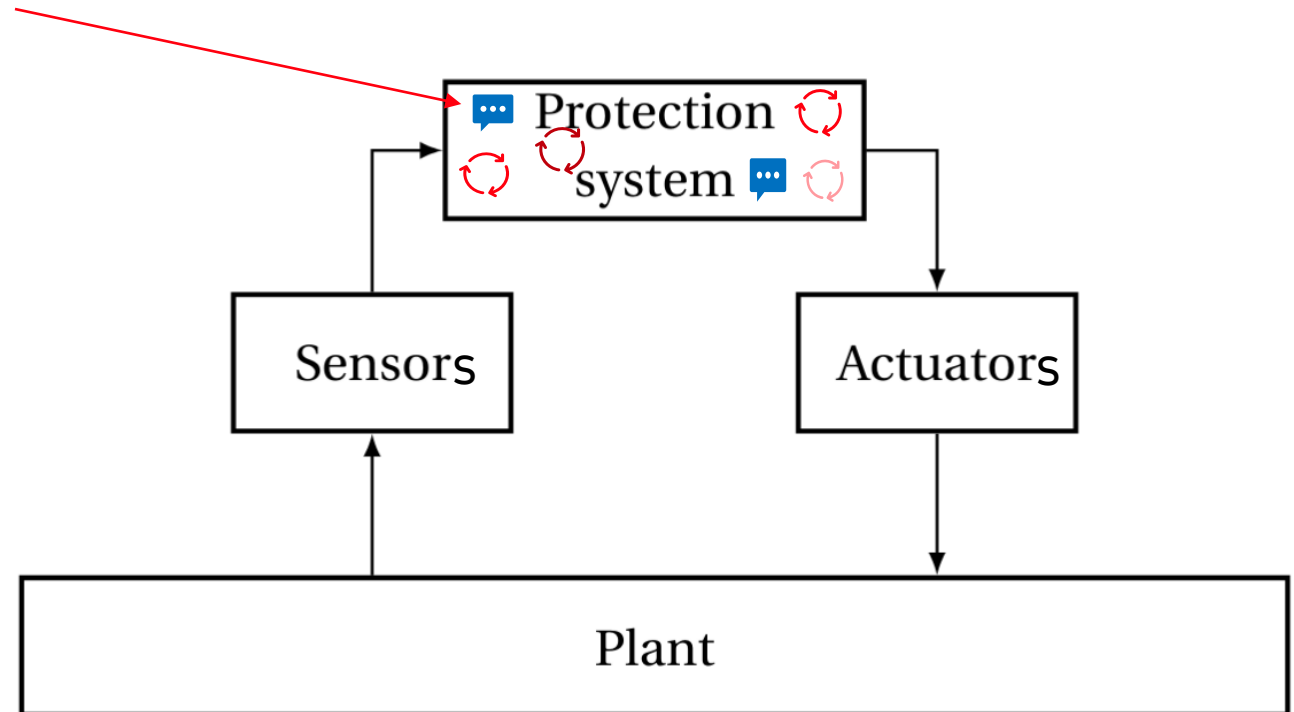
Goals

- Guaranteed attestable availability
 - System integrity and data confidentiality
 - Open, multi-tenant system
 - COTS hardware
 - Multi-processor
 - Separation of powers
- } General industry requirements + verifiable availability
- } Customer-centric, flexible, cost-saving

Background

Hard Real-Time Systems

- Tasks have:
 - Period
 - Deadline (= end of period)
 - Execution time
 - Priority
 - Resource requirements (CPU, devices)
- They compete for CPU
- May compete for devices
 - resource contention
 - scheduling problems
 - protocols exist!



Background

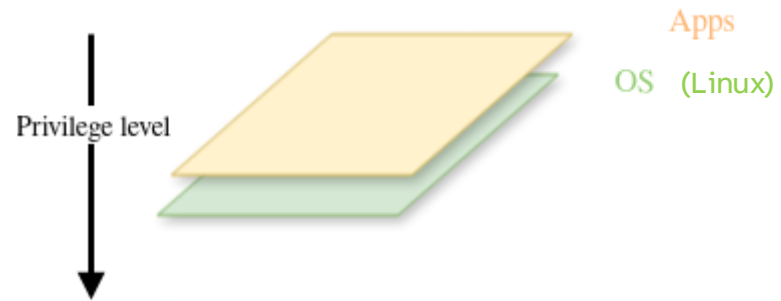
Trusted Computing on Arm



Before TEEs

Background

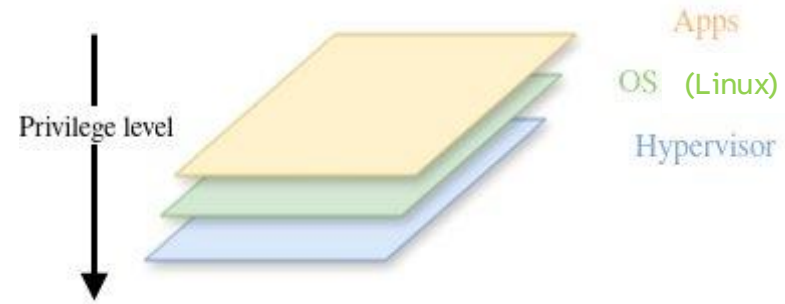
Trusted Computing on Arm



Before TEEs

Background

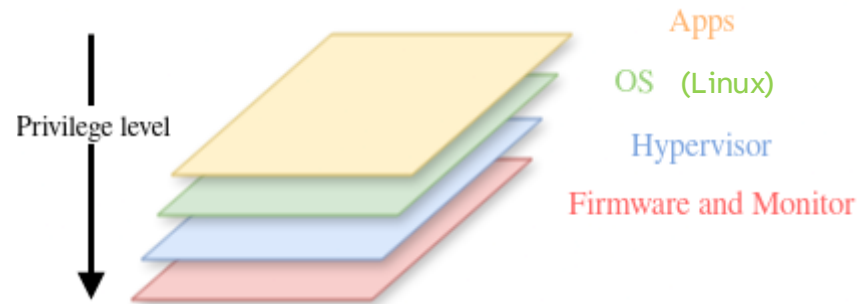
Trusted Computing on Arm



Before TEEs

Background

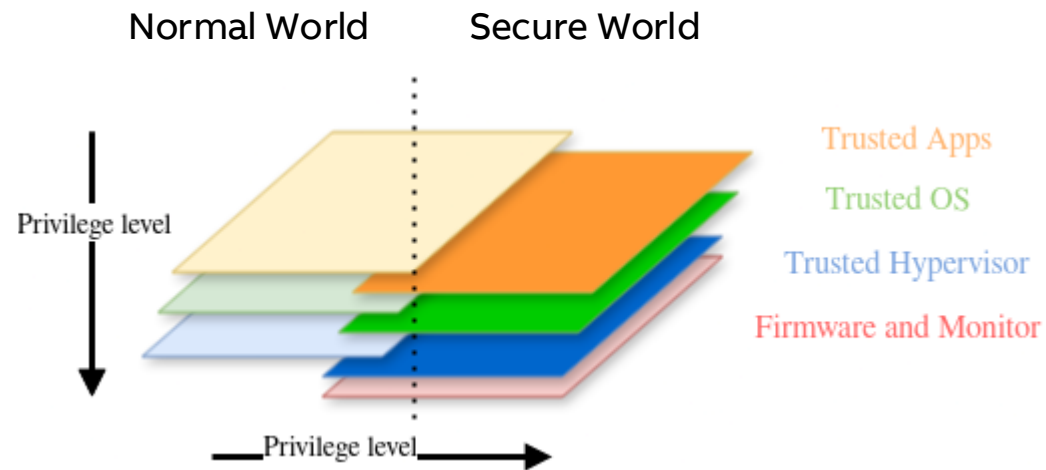
Trusted Computing on Arm



Before TEEs

Background

Trusted Computing on Arm



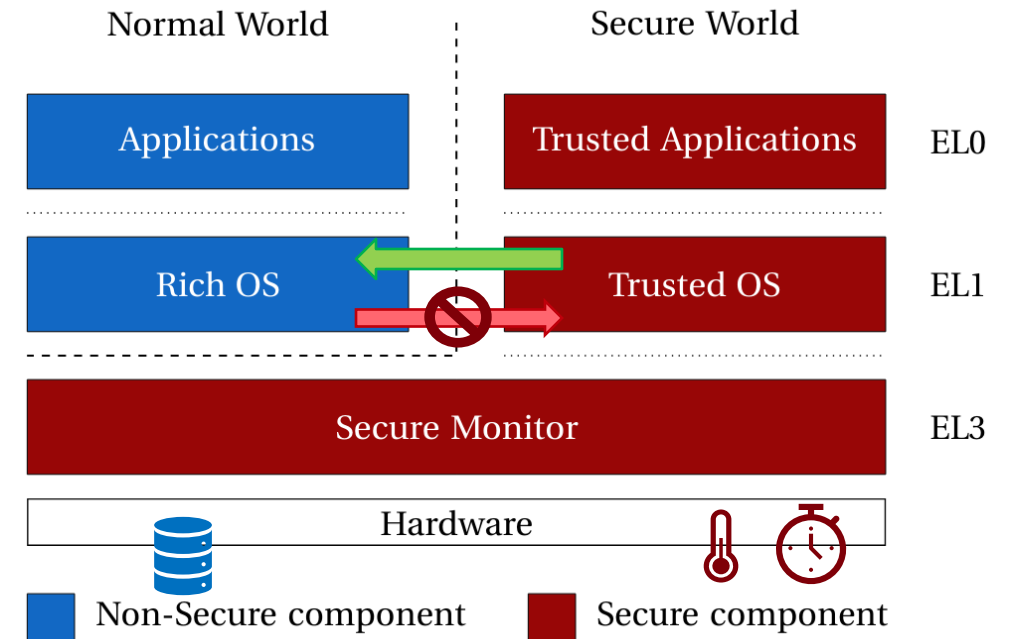
TrustZone TEE

Background

Trusted Computing on Arm

TrustZone TEE

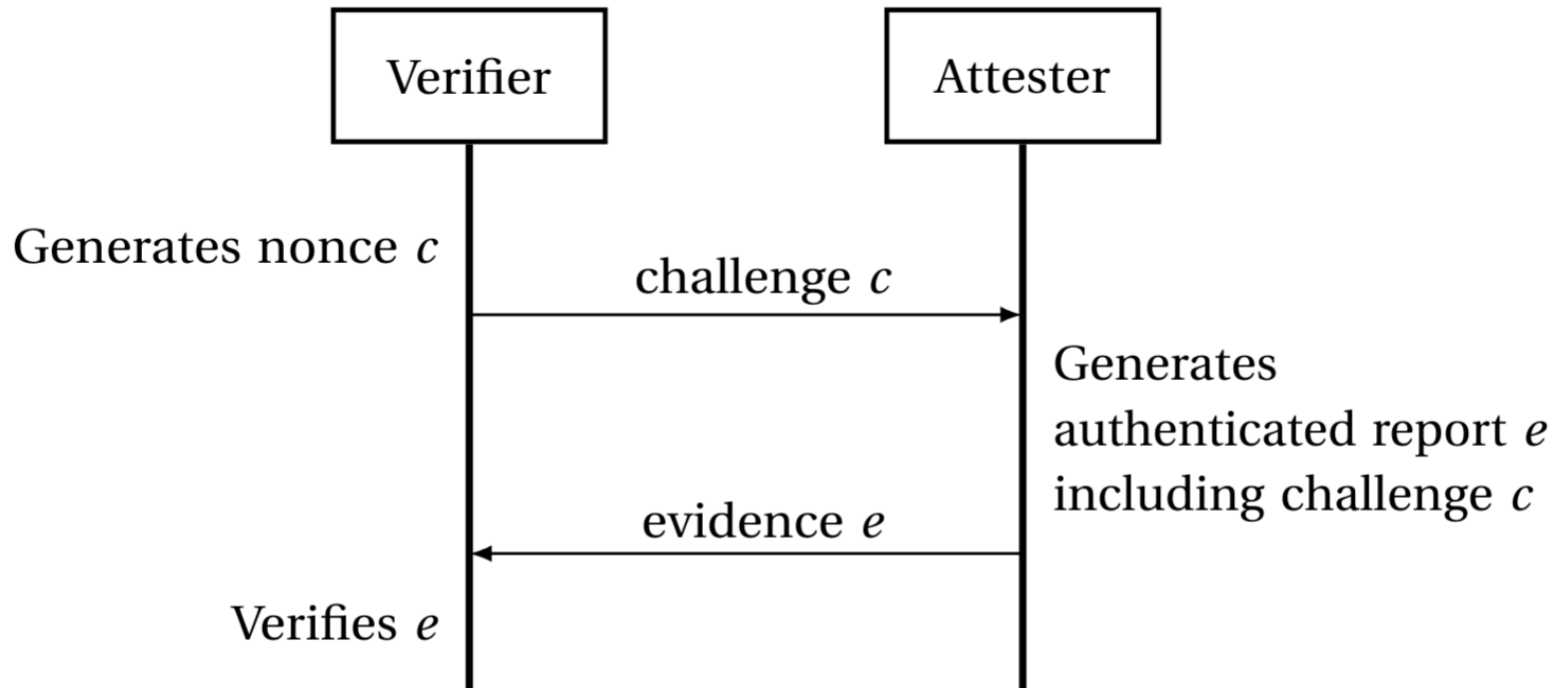
- Secure world isolated from Non-Secure world
- Partitioning of memory and peripherals
- Communication between worlds via SMC



Background

Remote Attestation

Procedure overview



Background

Remote Attestation

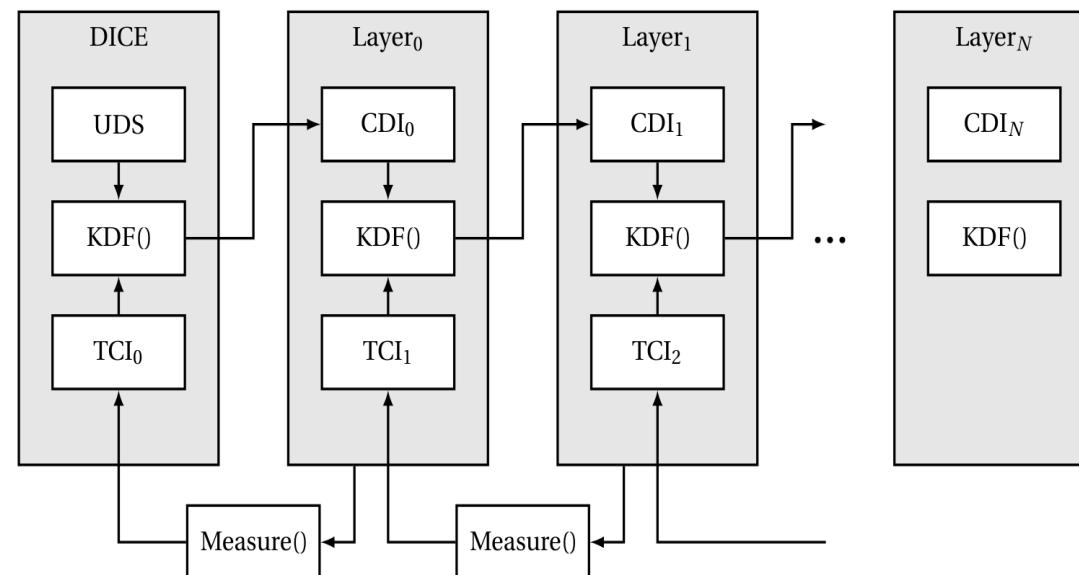
Root-of-Trust and Chain-of-Trust

- Attested device needs a trust anchor (DICE RoT)
- Need to extend trust to the attesting environment (AE)
- A trusted layer receives a valid secret (CDI)
- The AE needs a valid CDI to produce valid evidences

* UDS = Unique Device Secret

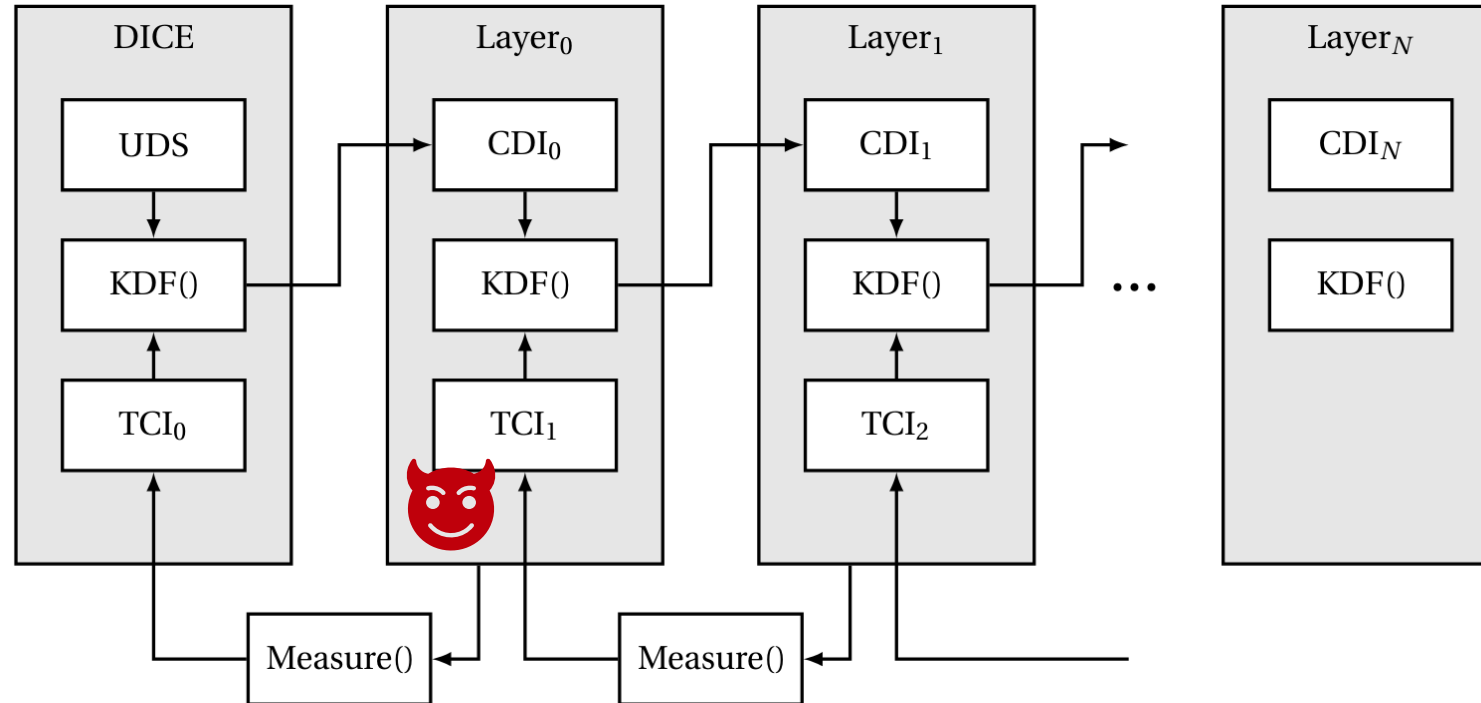
* CDI = Compound Device Identifier

* TCI = TCB Component Identifier



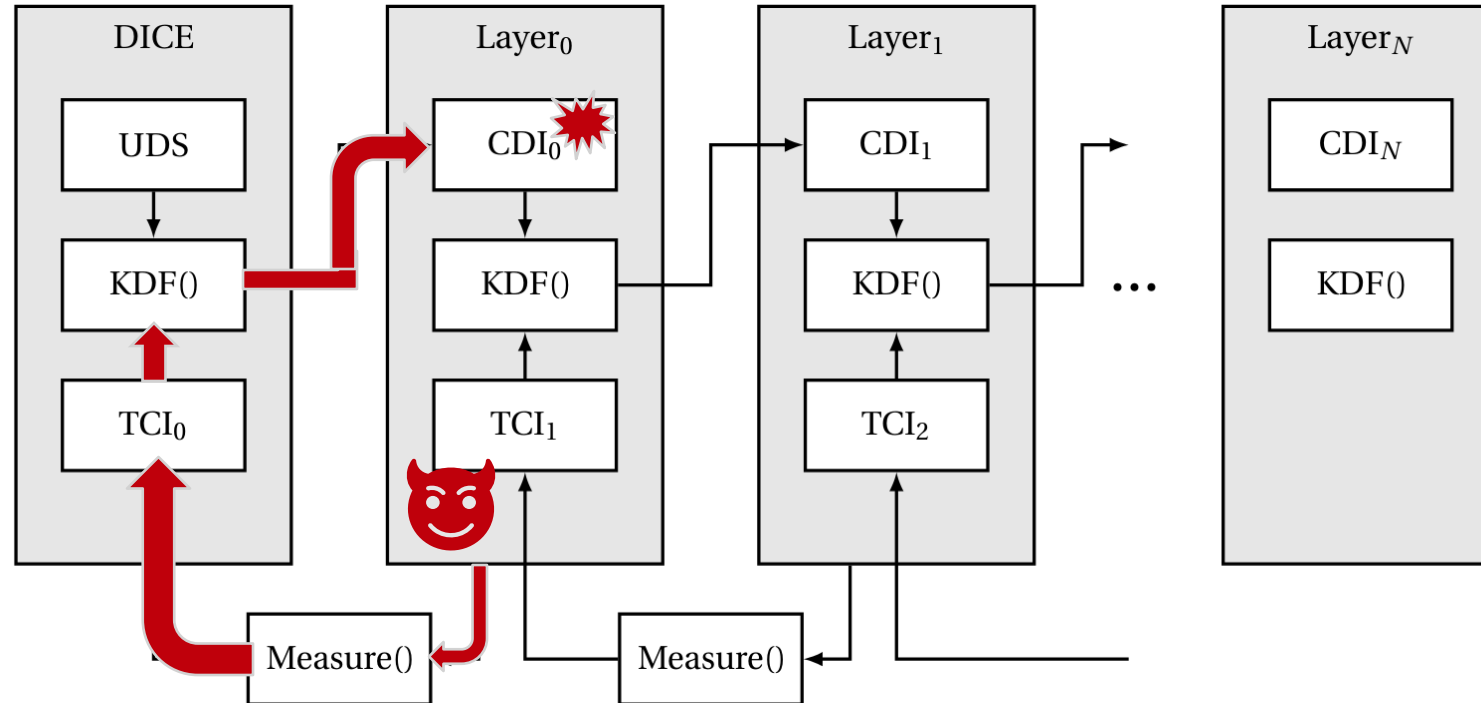
Background

Remote Attestation



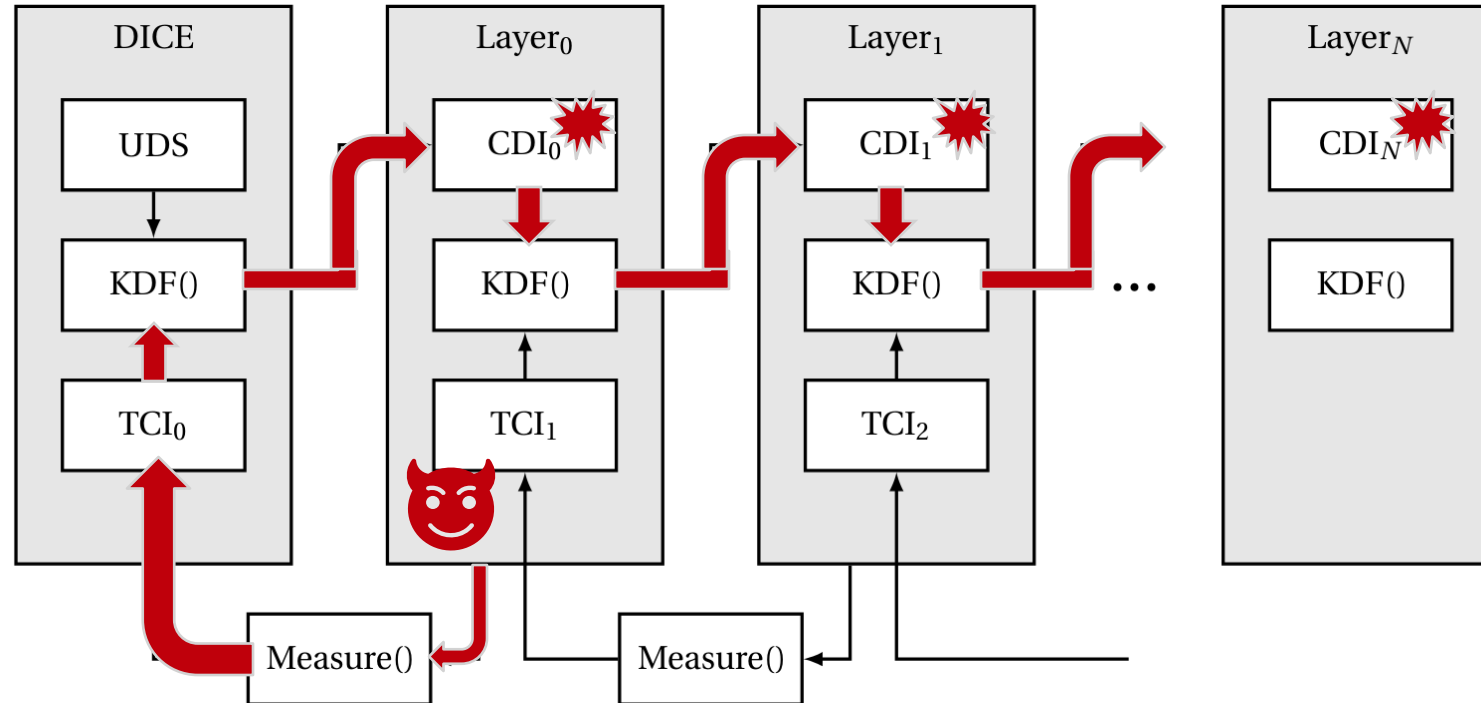
Background

Remote Attestation



Background

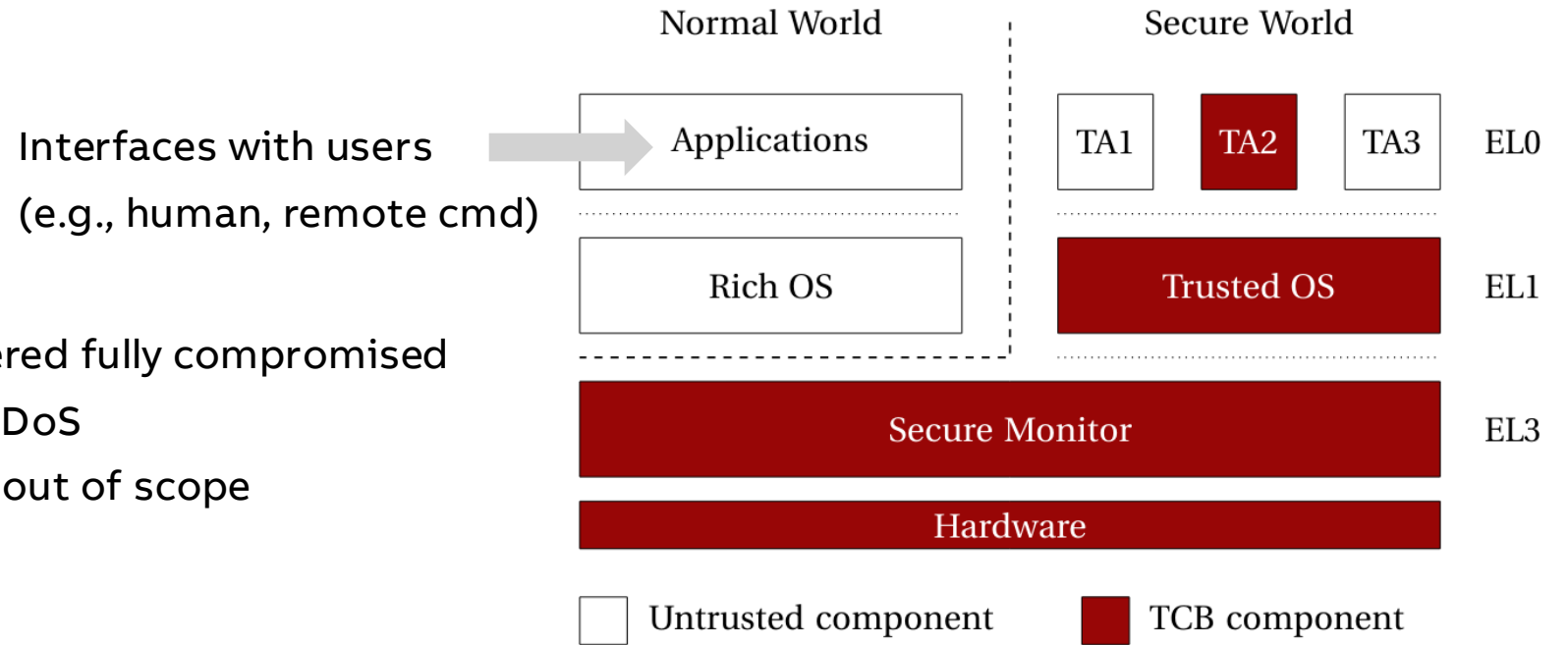
Remote Attestation



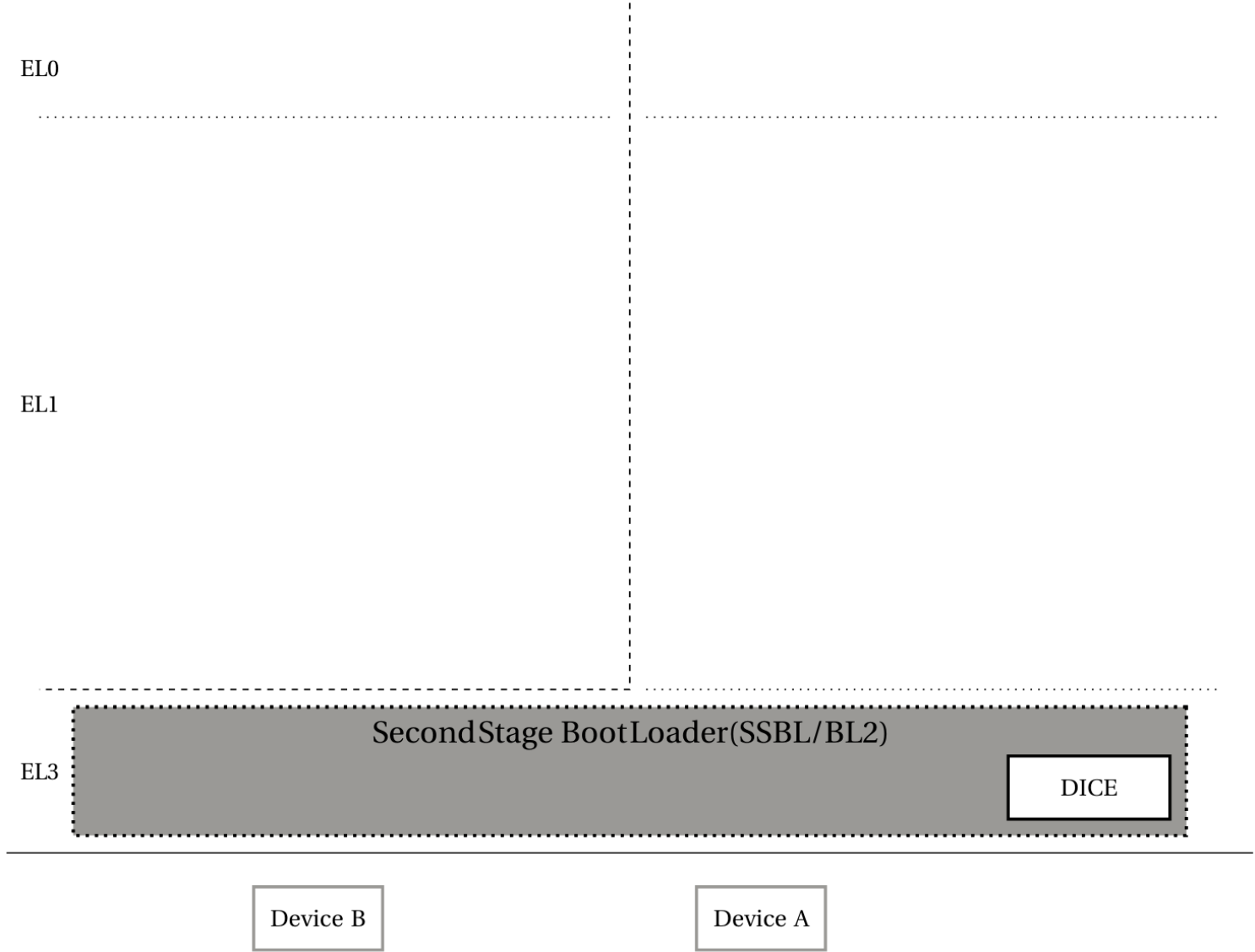
Threat Model

From the perspective of TA2

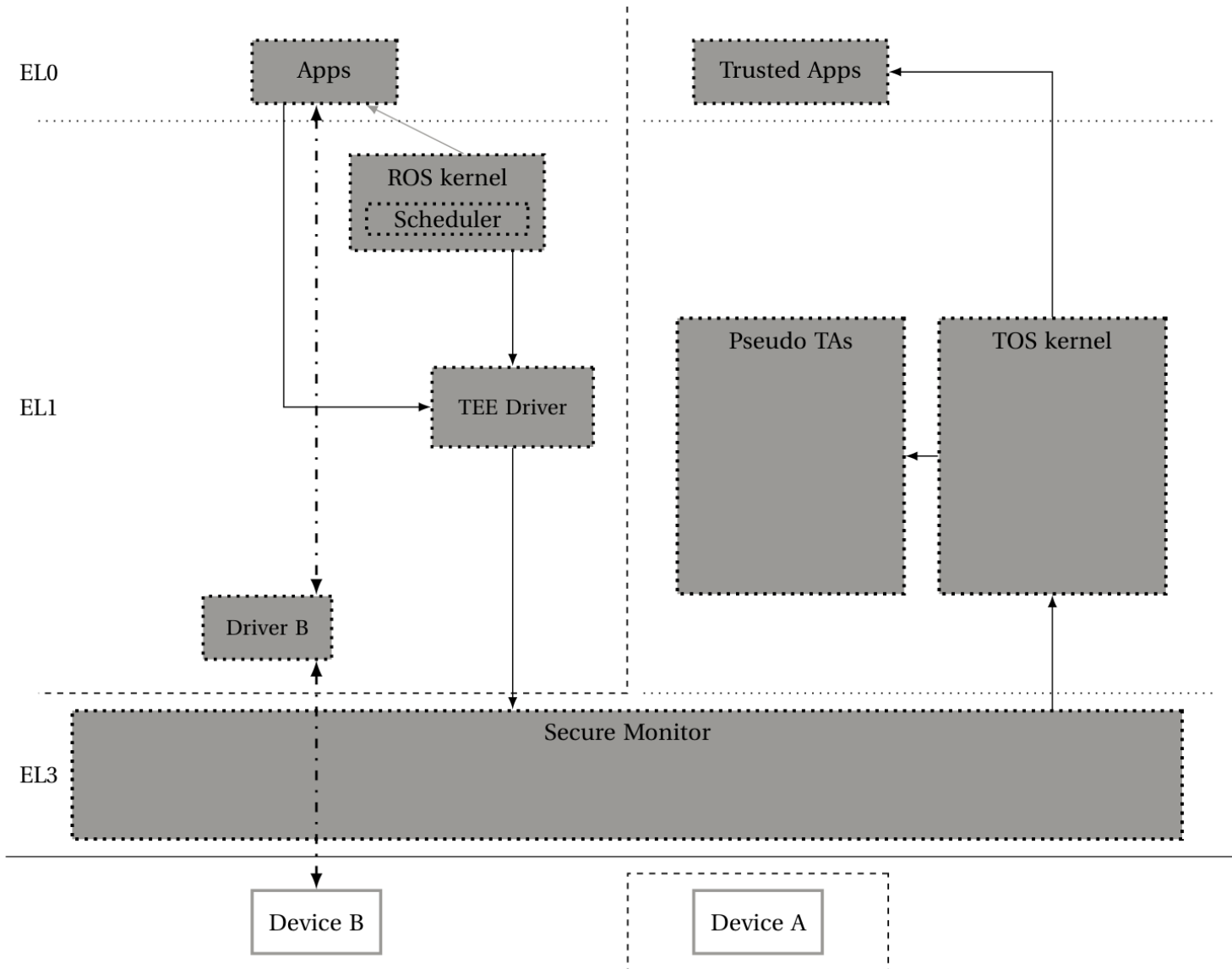
- Normal world (Linux, apps) considered fully compromised
 - o e.g., controlled by attacker, or DoS
- Physical attacks and side-channels out of scope



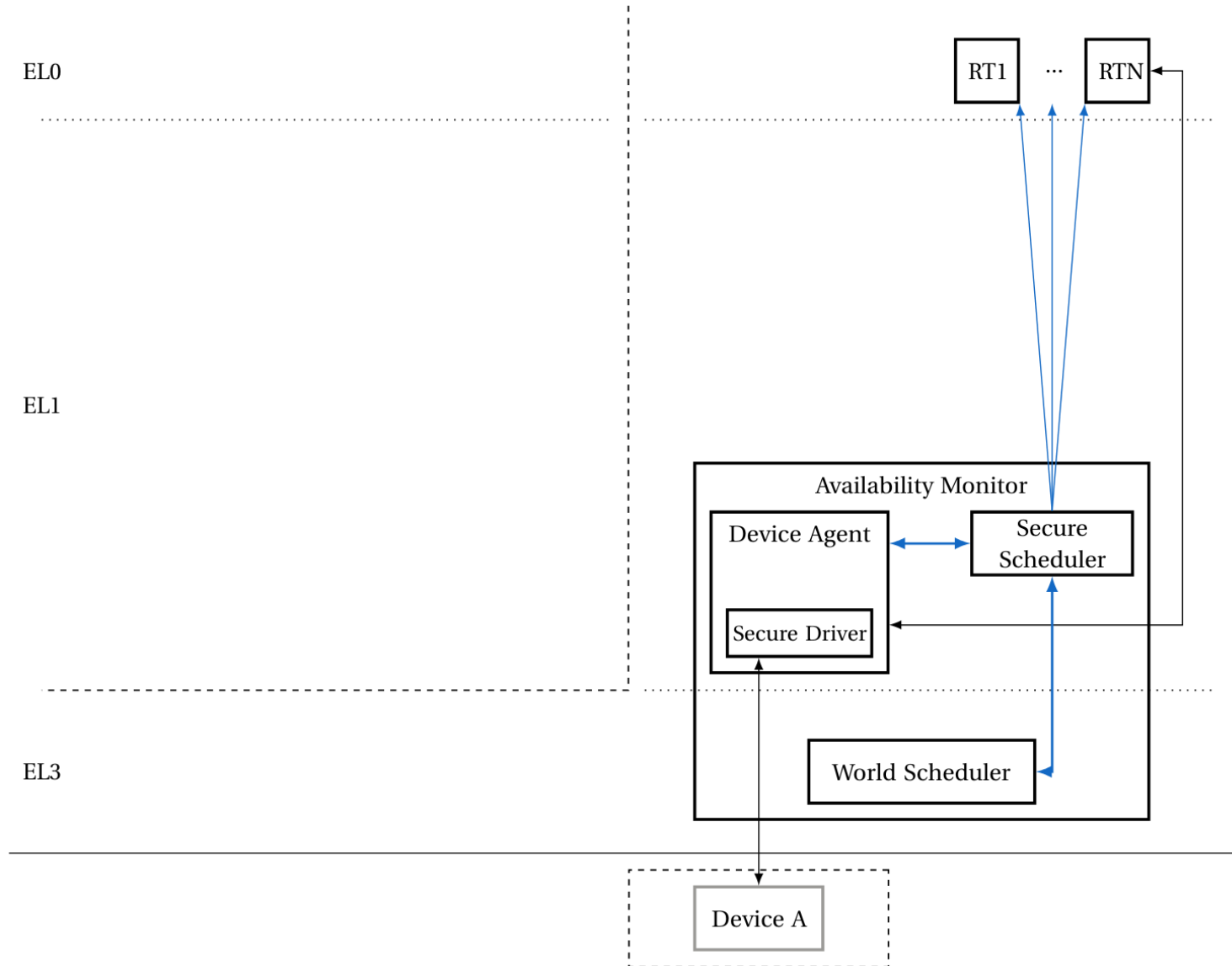
Design Overview



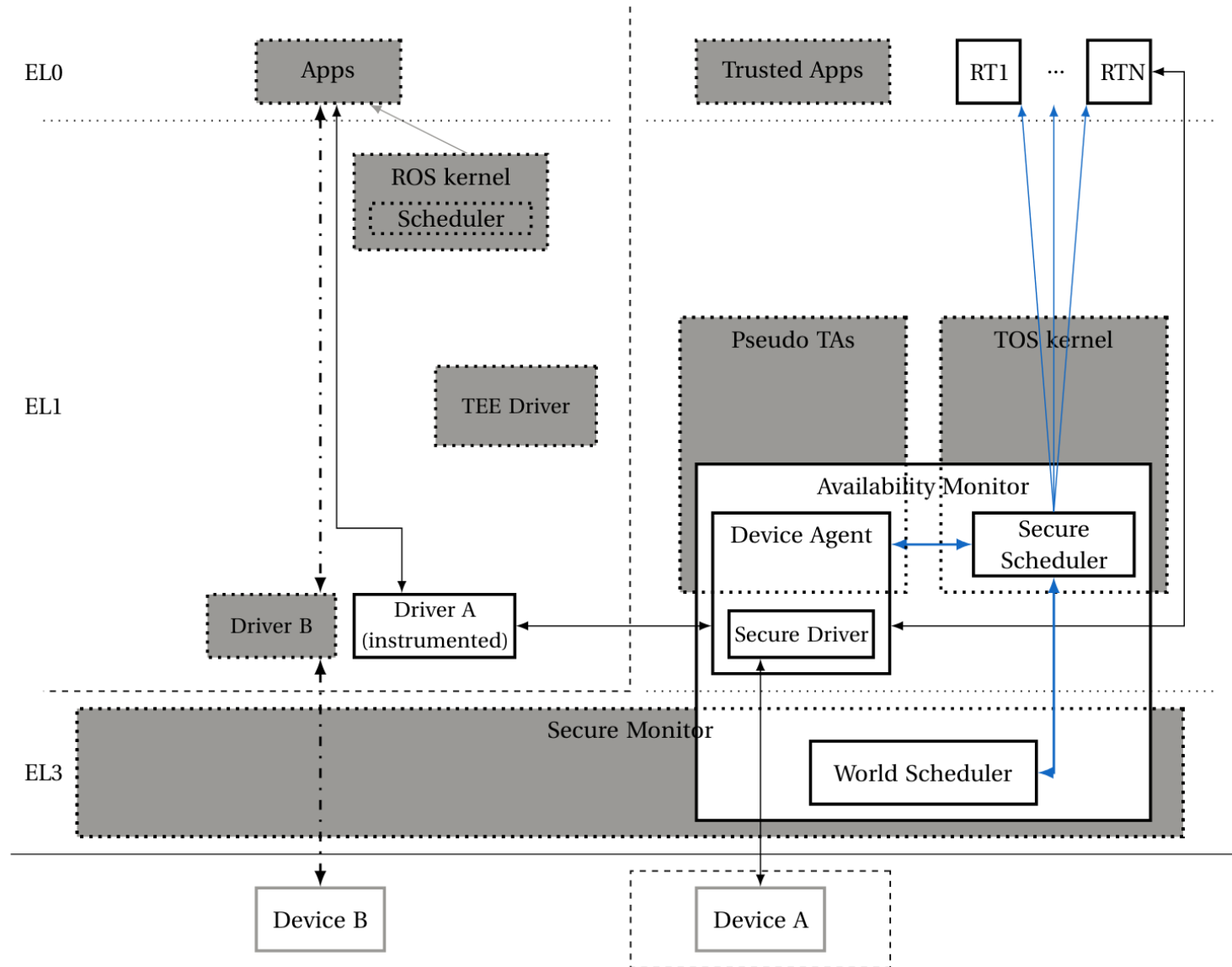
Design Overview



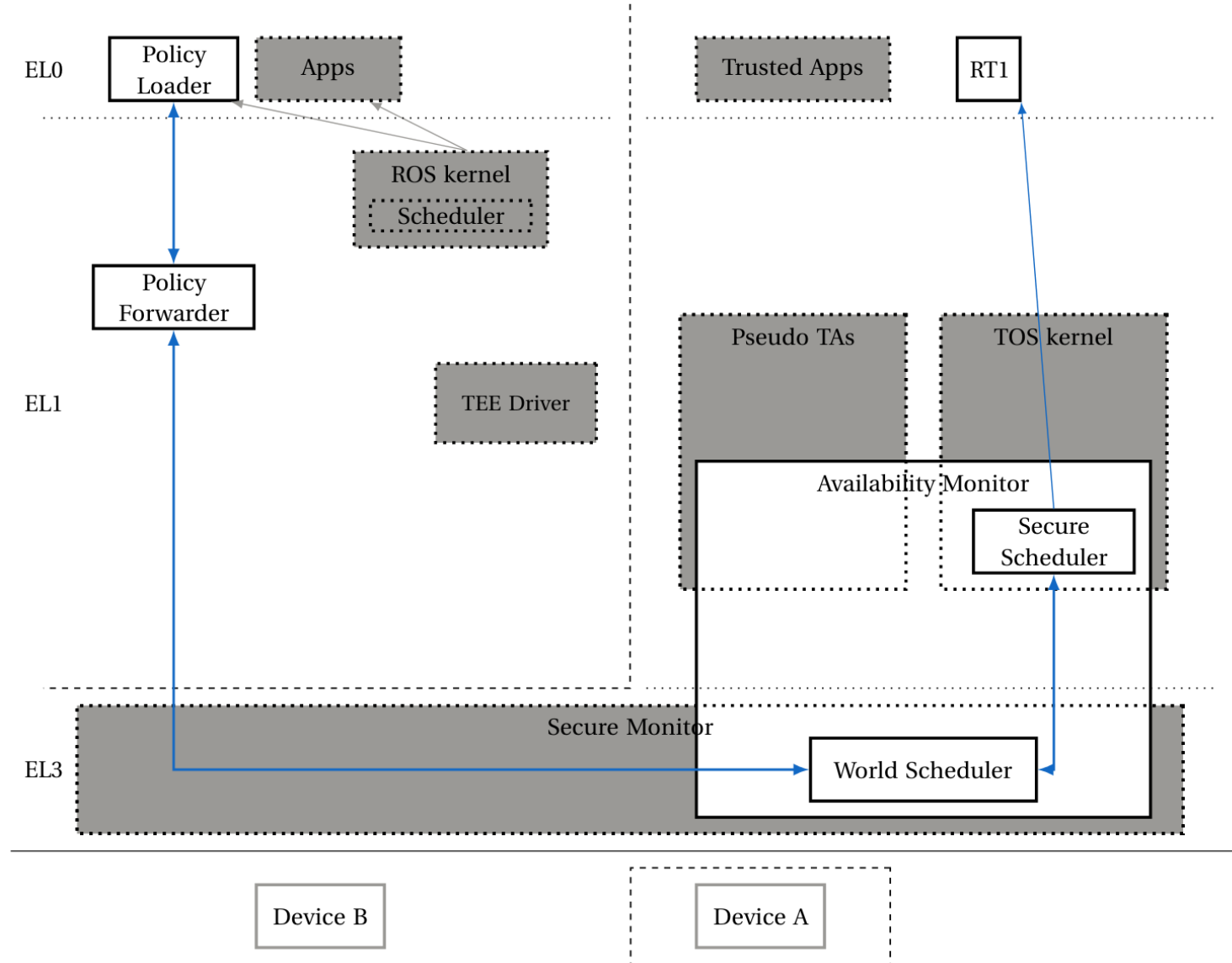
Design Overview



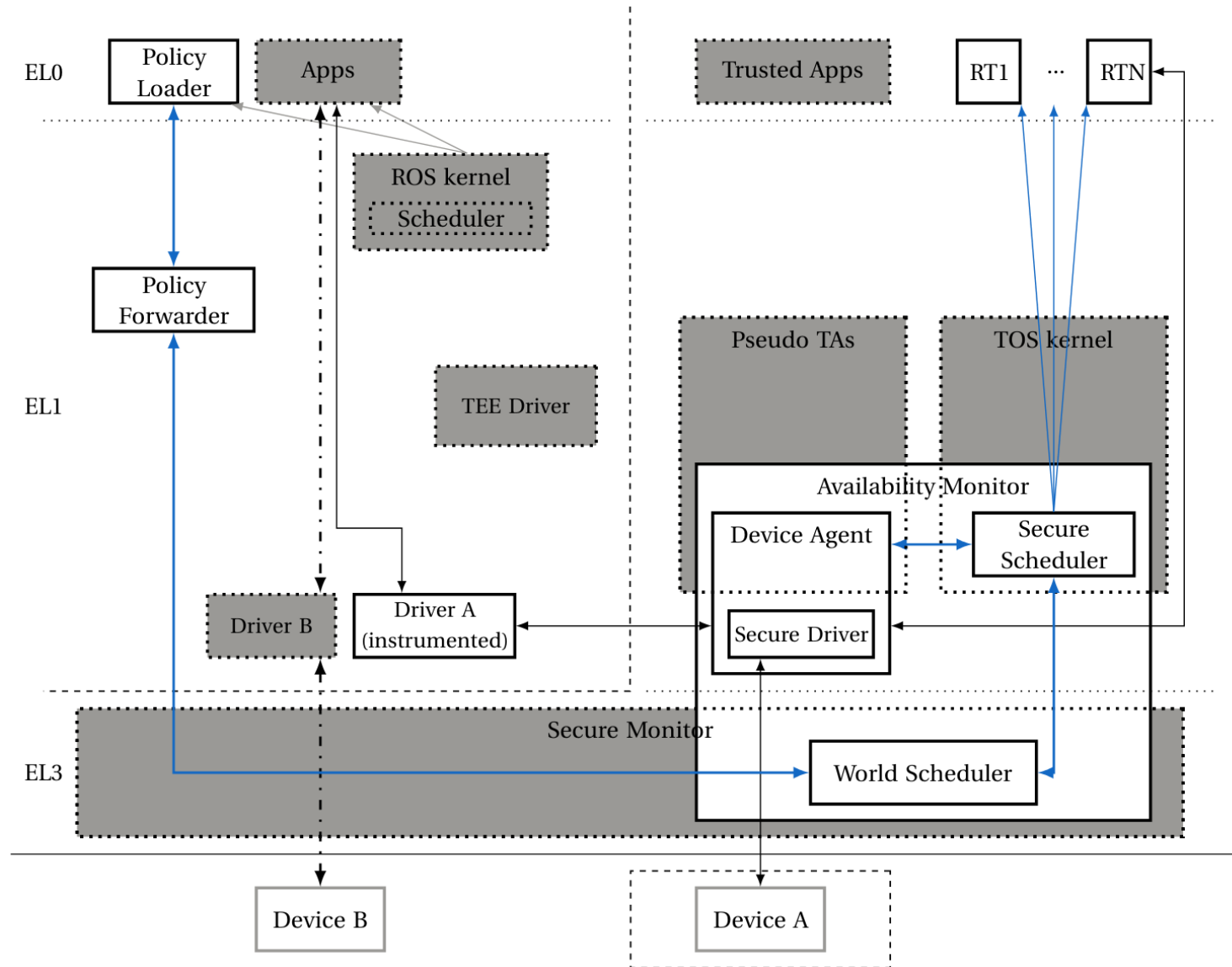
Design Overview



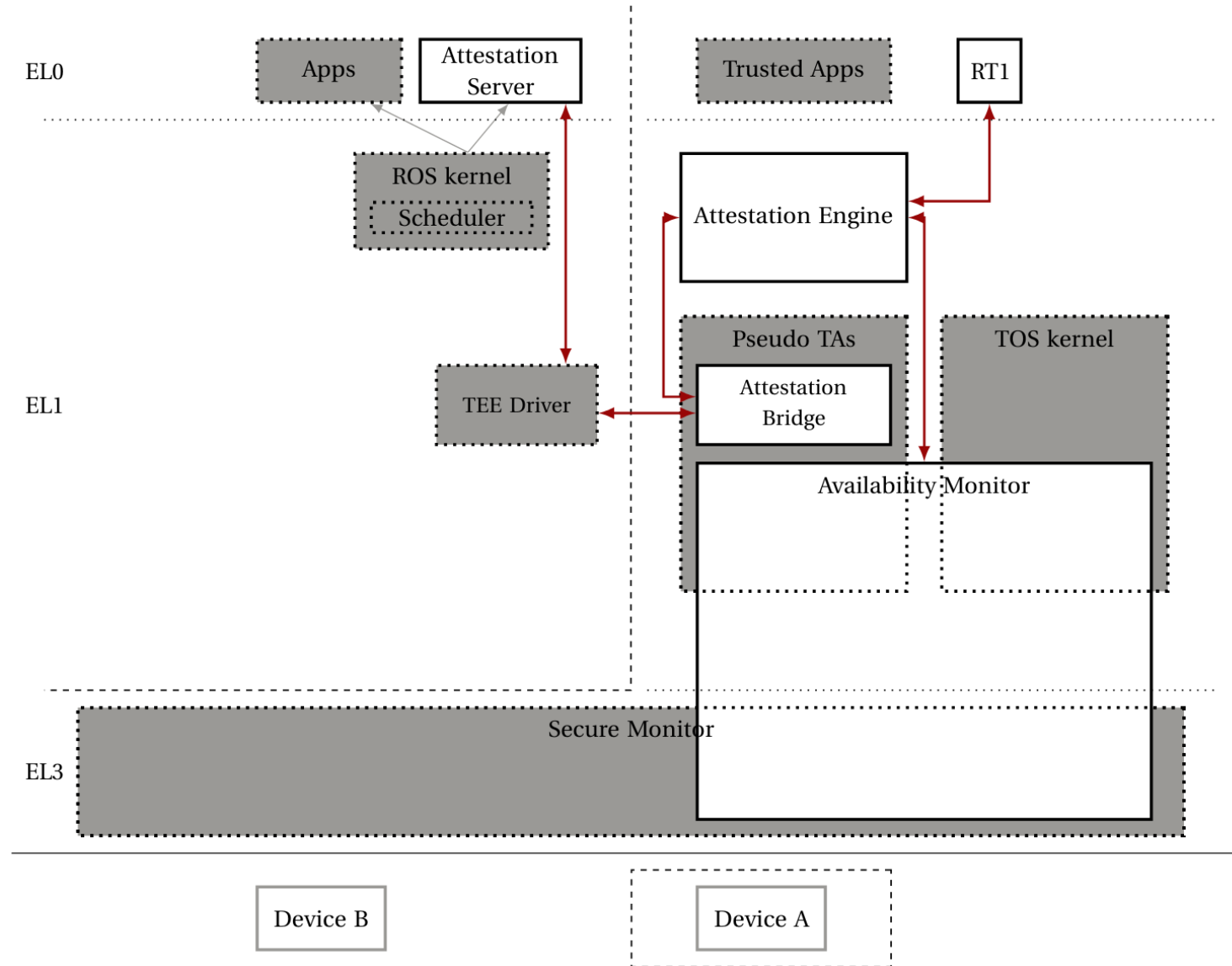
Design Overview



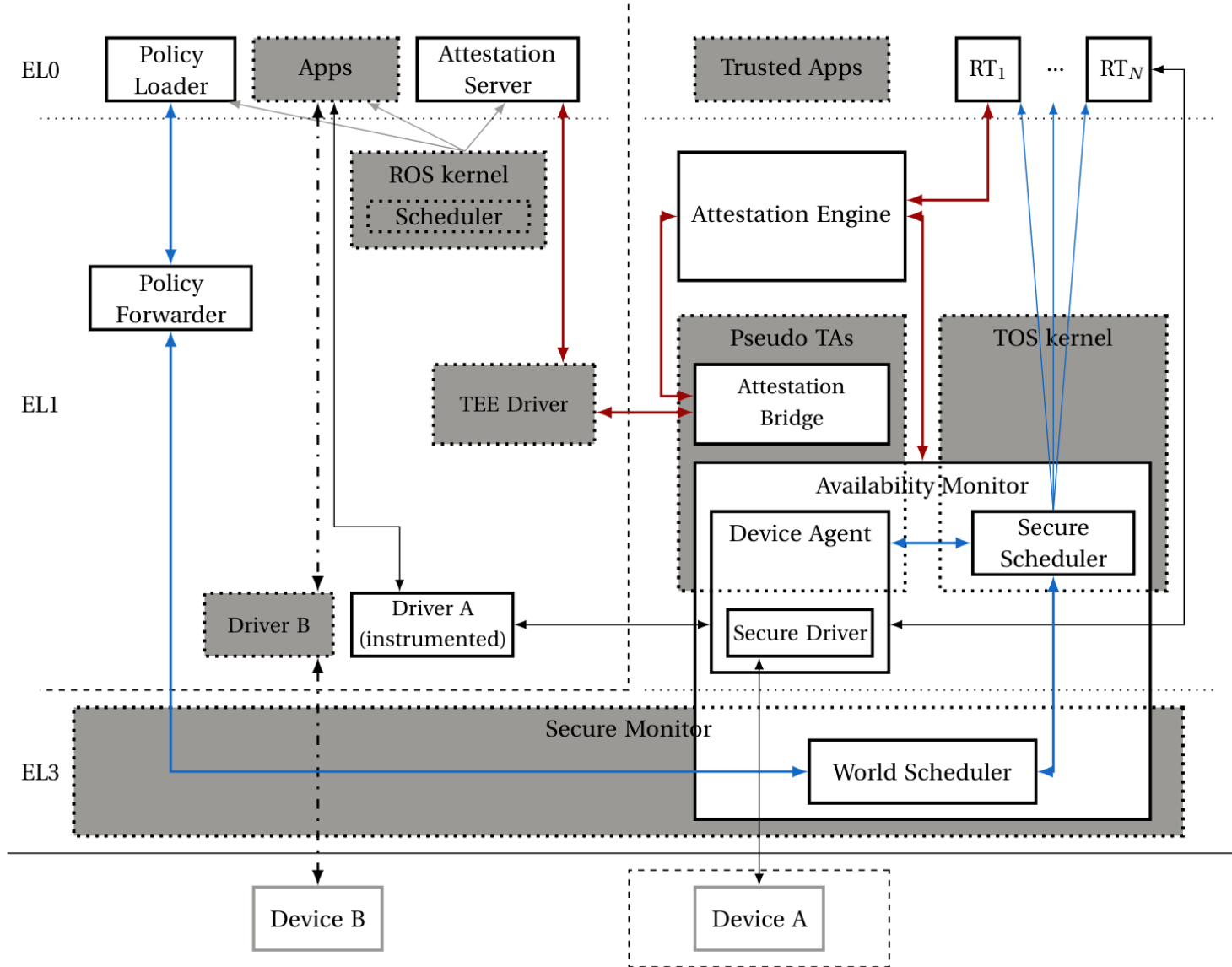
Design Overview



Design Overview



Design Overview

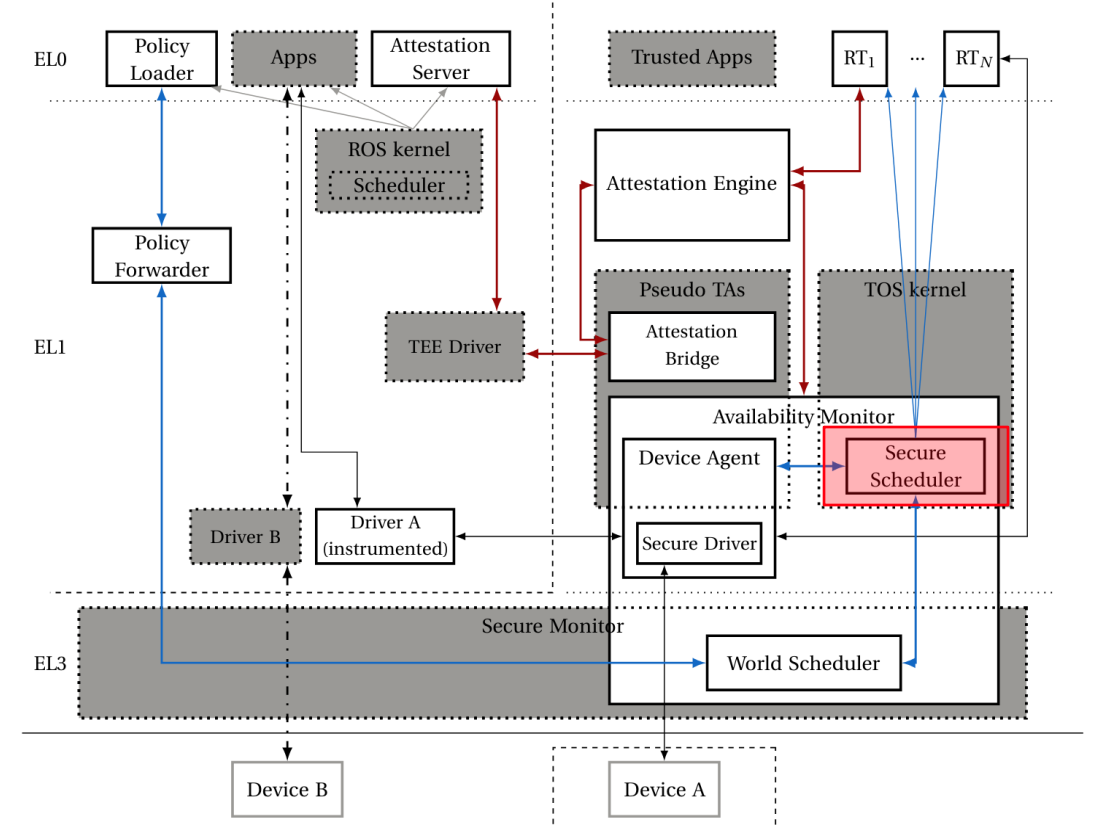


Design

The Availability Monitor

The Secure Scheduler

Responsibility: ensure availability of CPU resources



Design

The Availability Monitor

The Secure Scheduler

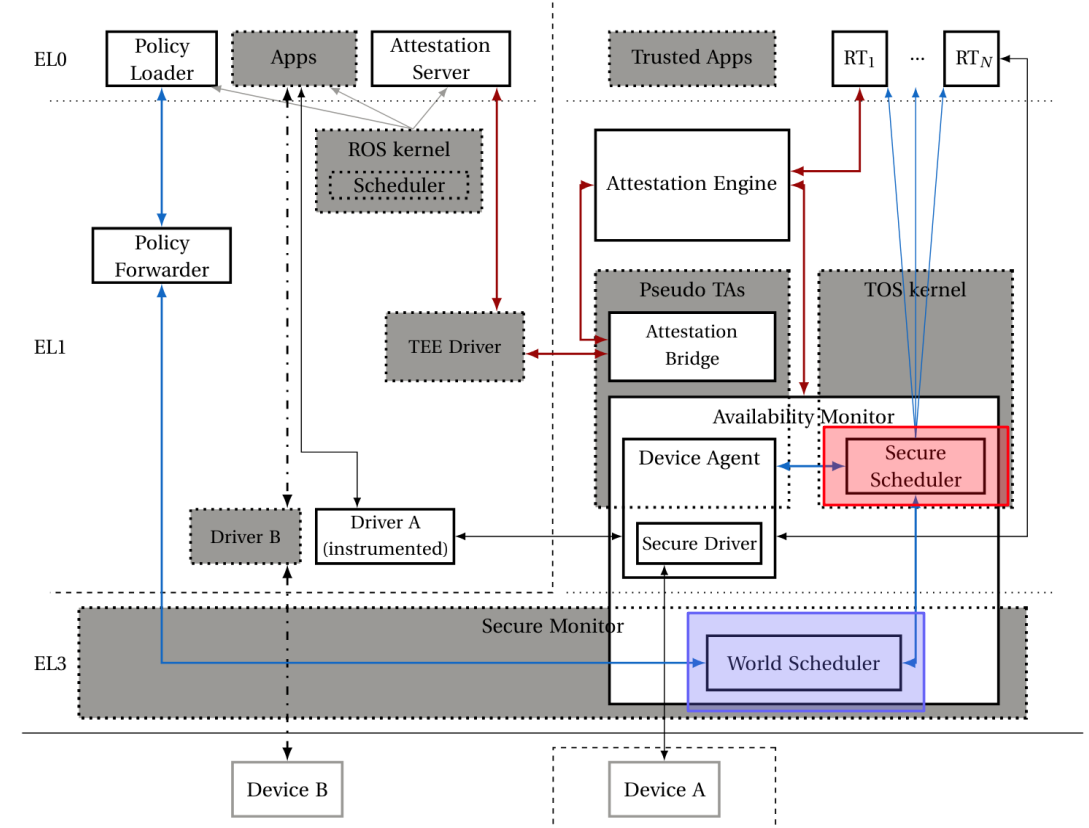
Responsibility: ensure availability of CPU resources

Share CPU resources:

- Between critical tasks
- Between S and NS workload, via the **World Scheduler**

Timer to return control to TCB

Works together with Device Agents

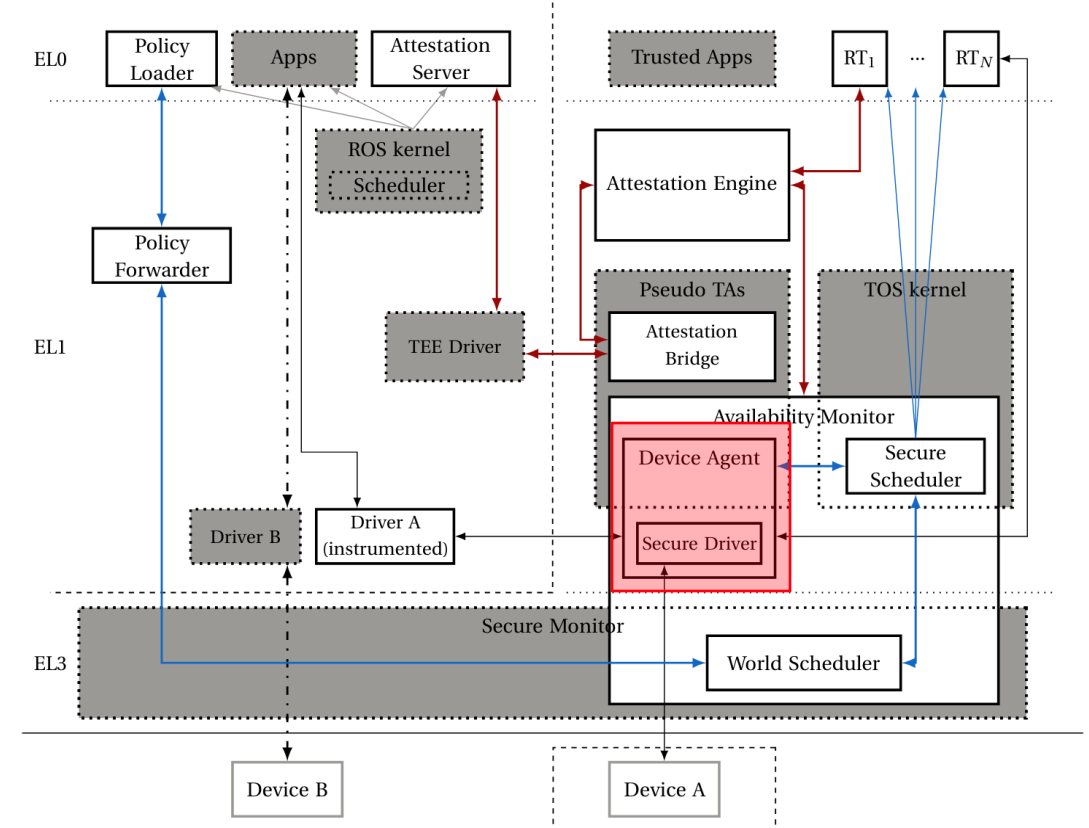


Design

The Availability Monitor

The Device Agents

Responsibility: ensure availability of devices



Design

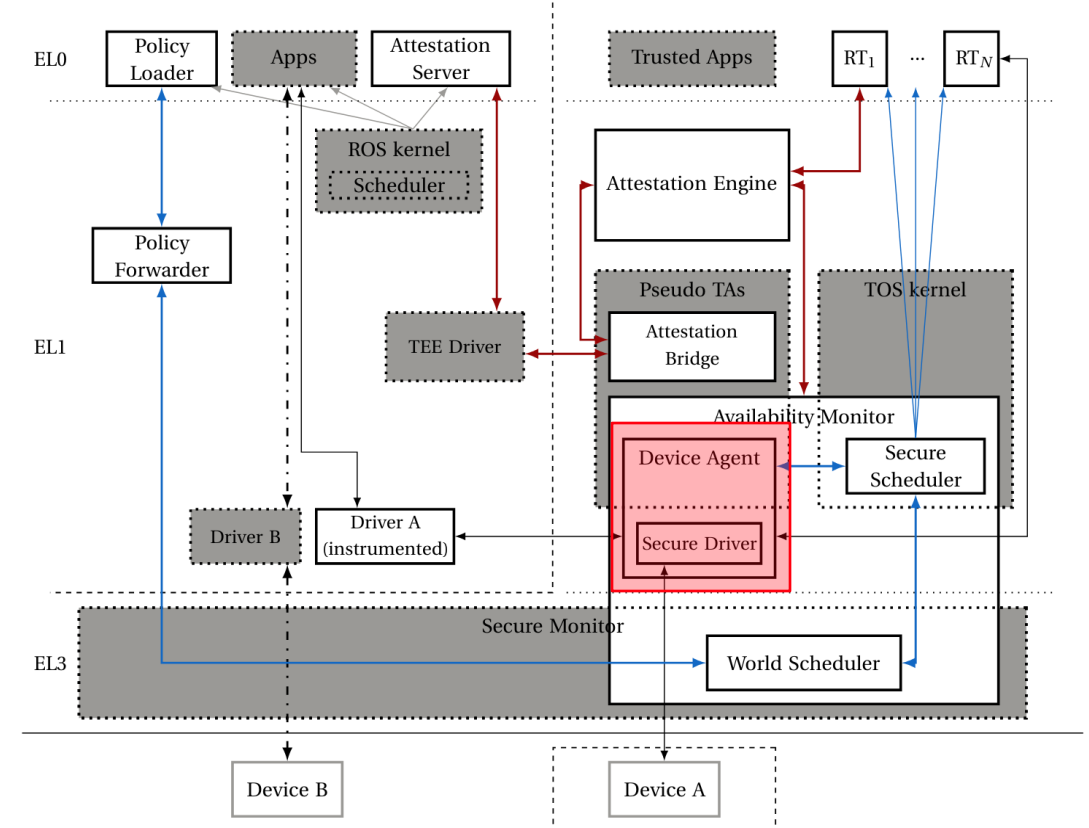
The Availability Monitor

The Device Agents

Responsibility: ensure availability of devices

Spatial isolation:

- Isolate their peripheral at boot time (MMIO, interrupts)
- Ensure device integrity and state confidentiality (monitor every access)



Design

The Availability Monitor

The Device Agents

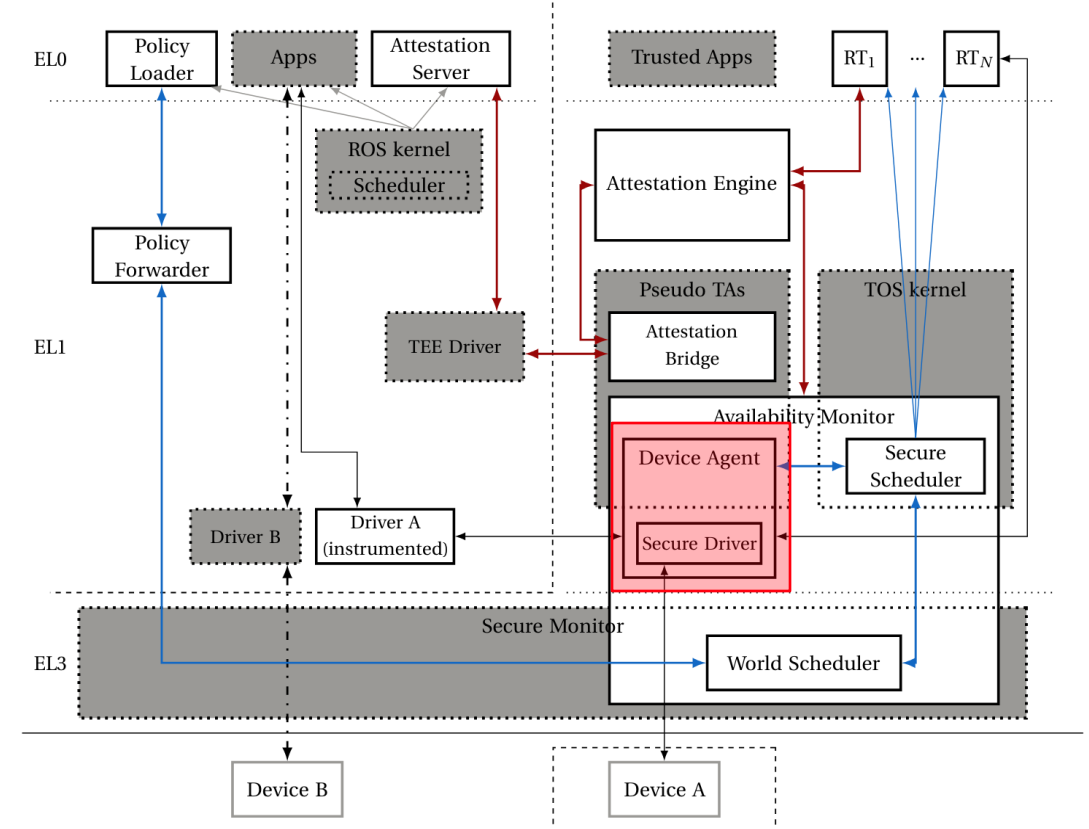
Responsibility: ensure availability of devices

Spatial isolation:

- Isolate their peripheral at boot time (MMIO, interrupts)
- Ensure device integrity and state confidentiality (monitor every access)

Temporal isolation:

- Ensure bounded access to their resource



Design

The Availability Monitor

The Device Agents

Responsibility: ensure availability of devices

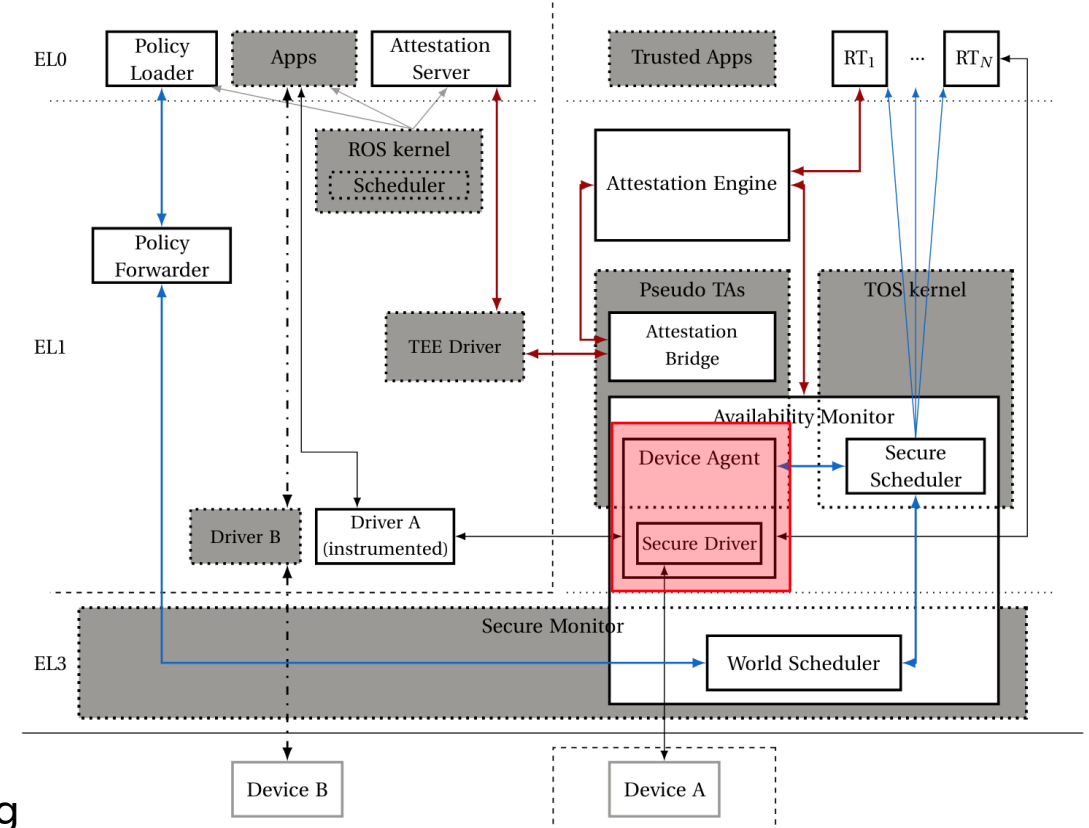
Spatial isolation:

- Isolate their peripheral at boot time (MMIO, interrupts)
- Ensure device integrity and state confidentiality (monitor every access)

Temporal isolation:

- Ensure bounded access to their resource

Prevents data leakage and corrupted configuration when switching owner



Design

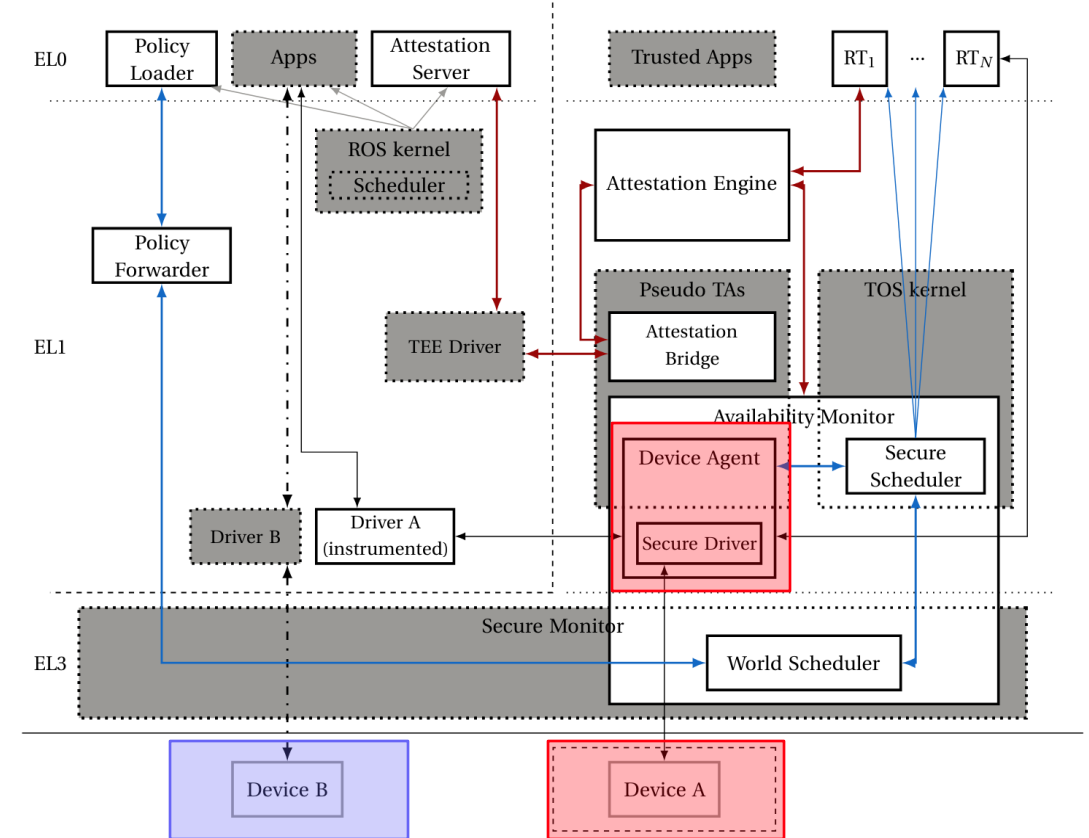
The Availability Monitor

The Device Agents

Responsibility: ensure availability of devices

Sharing between worlds:

- Devices used by secure tasks are mapped to S, otherwise mapped to NS



Design

The Real-Time Tasks

The Tasks

AM ensures availability of resources, but we also provide tasks with:

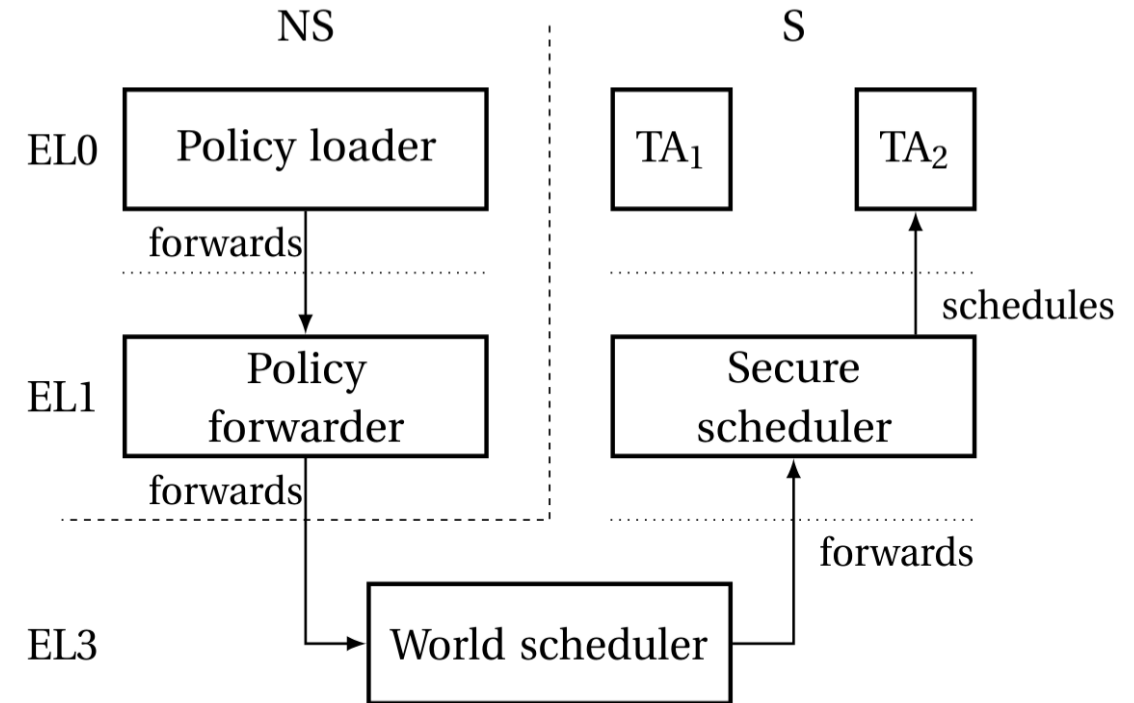
- Confidentiality of code and data at rest and in use (Confidentiality)
- Launch and runtime integrity (Integrity)
- Guaranteed loading (Availability)

Design

The Real-Time Tasks

The Policies

- Defined and provided by user (separation of powers)
- Collectively define scheduling policy of the system
→ **potential attack vector!!**

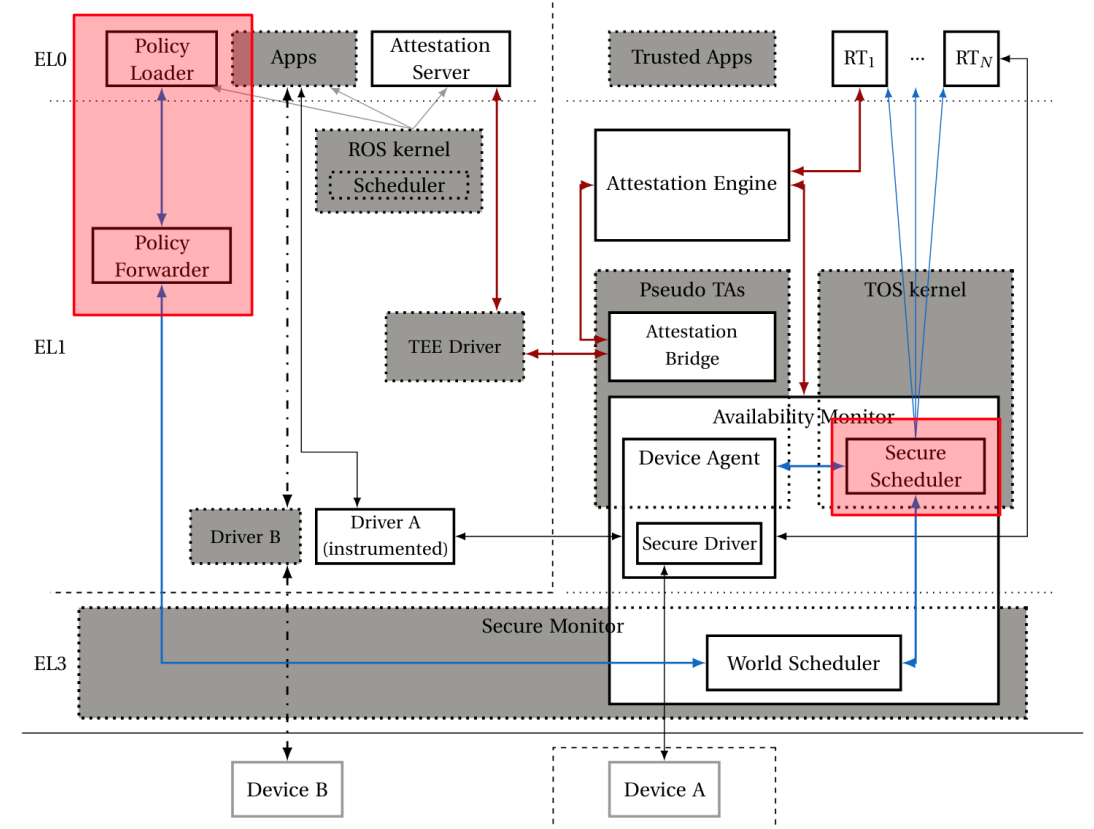


Design

The Real-Time Tasks

The Policies

- Authenticated policies
 - o Integrity
 - o Authorization
- Replay protection
- Rollback detection
- Denied forwarding mitigation



Design

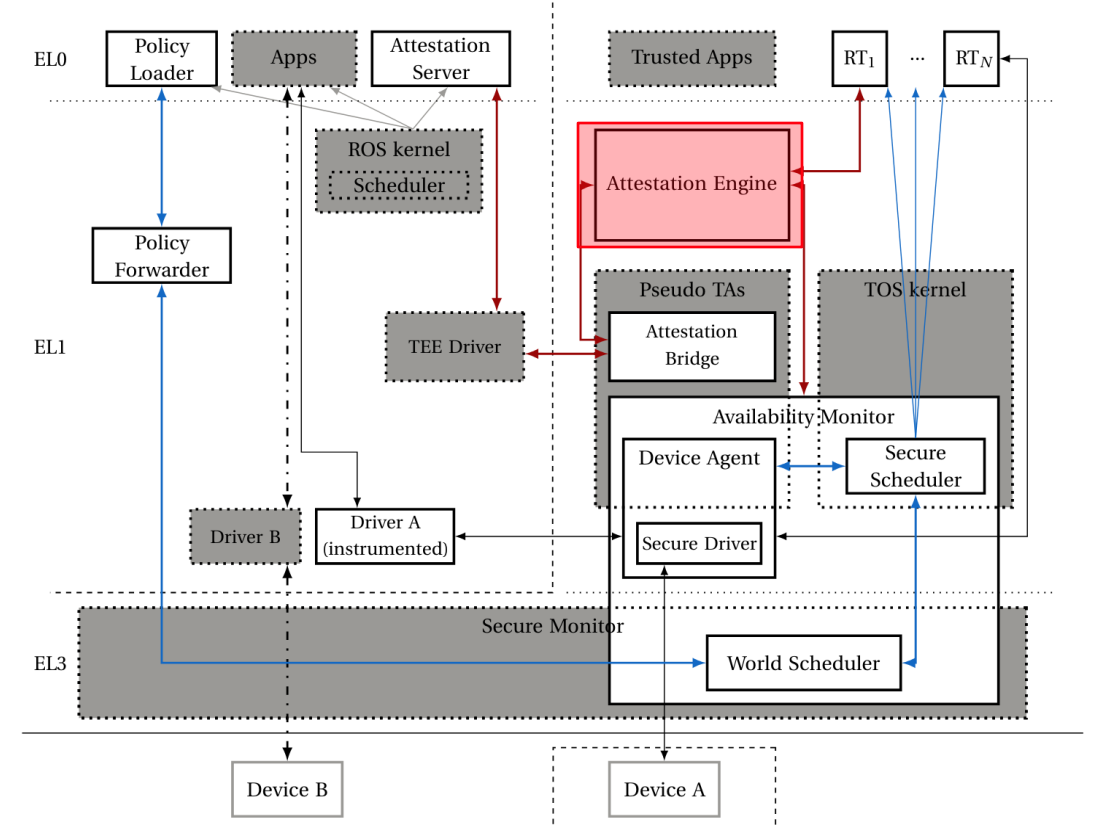
The Attestation Engine

Responsibility:

- verify AM enforces policies, and
- verify policies allow tasks to run expectedly
- verify correct policy and task are running

→ Real-Time Proof-of-eXecution [1] (RT-PoX)

→ Attestation of availability

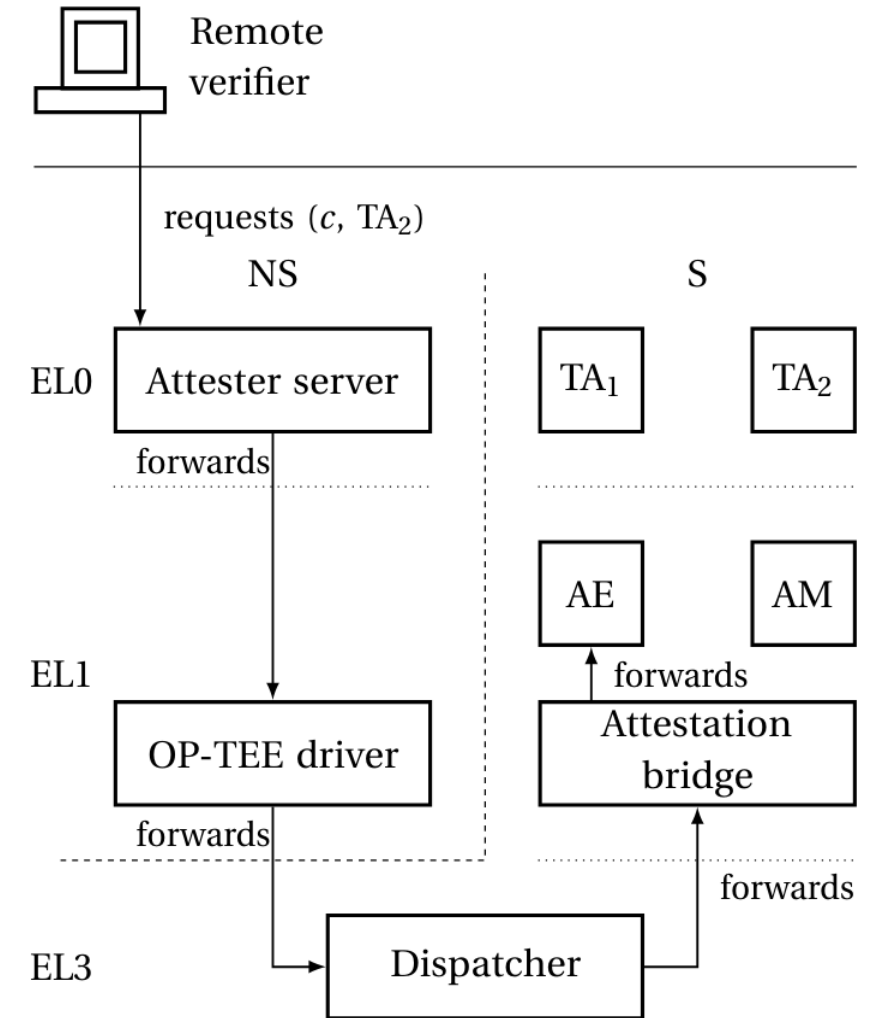


Design

The Attestation Engine

Attestation Process

1. Remote verifier sends challenge and task ID
2. Request forwarded by NS server
3. Attestation bridge in S forwards to the AE



Design

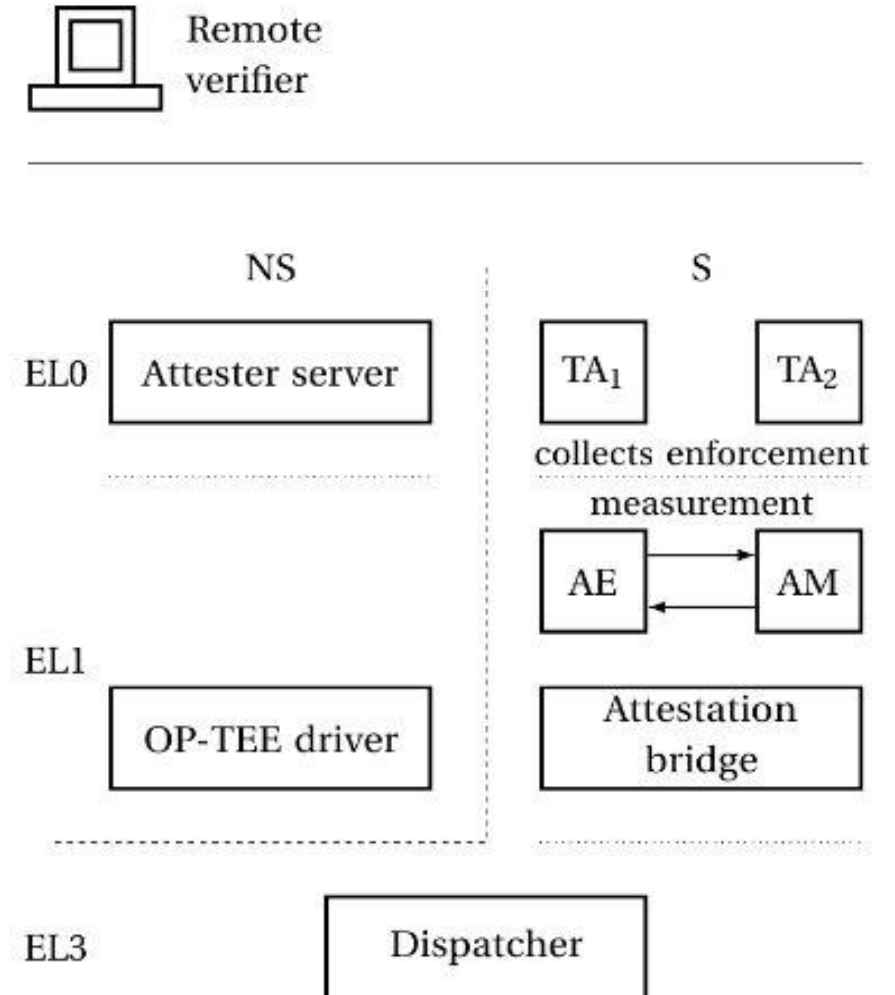
The Attestation Engine

Attestation Process

1. Remote verifier sends challenge and task ID
2. Request forwarded by NS server
3. Attestation bridge in S forwards to the AE
4. AE collects claims:
 - (1) reads AM memory to produce fingerprint of task

FINGERPRINT

- UUID
- properties (p, e, c, R)
- # missed deadlines
- current state's validity
- is allocated a valid thread?
- has recently run?

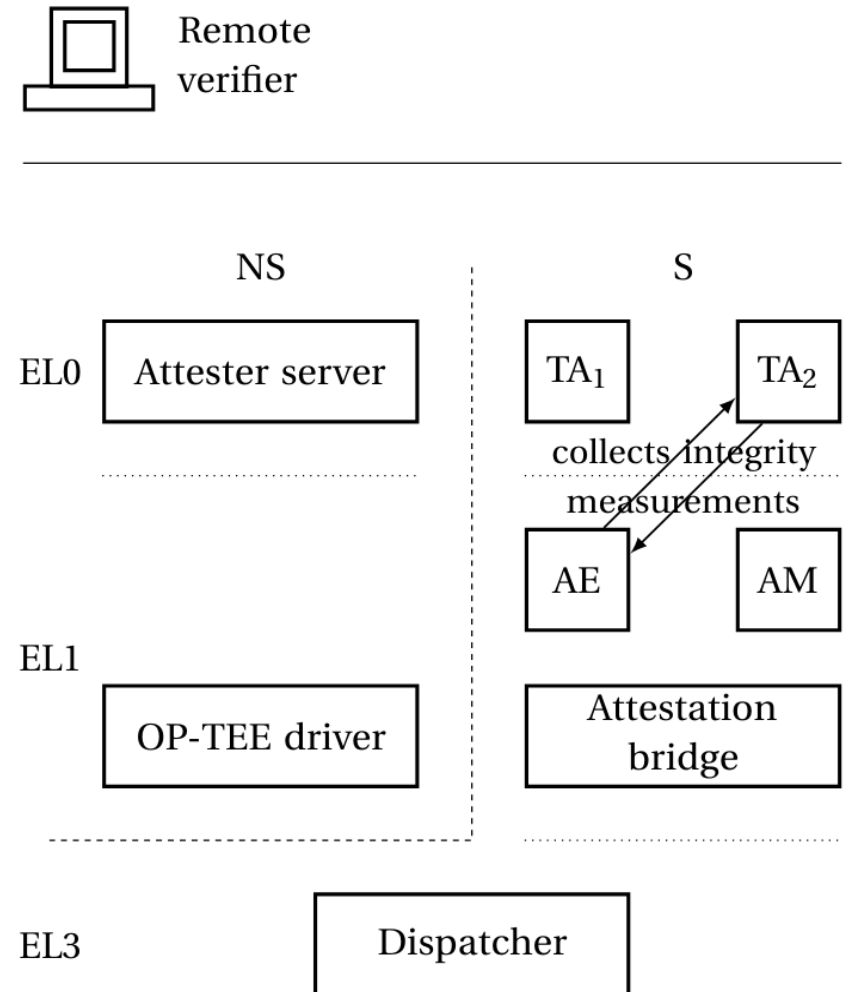


Design

The Attestation Engine

Attestation Process

1. Remote verifier sends challenge and task ID
2. Request forwarded by NS server
3. Attestation bridge in S forwards to the AE
4. AE collects claims:
 - (1) reads AM memory to produce fingerprint of task
 - (2) measures read-only sections in task's address space

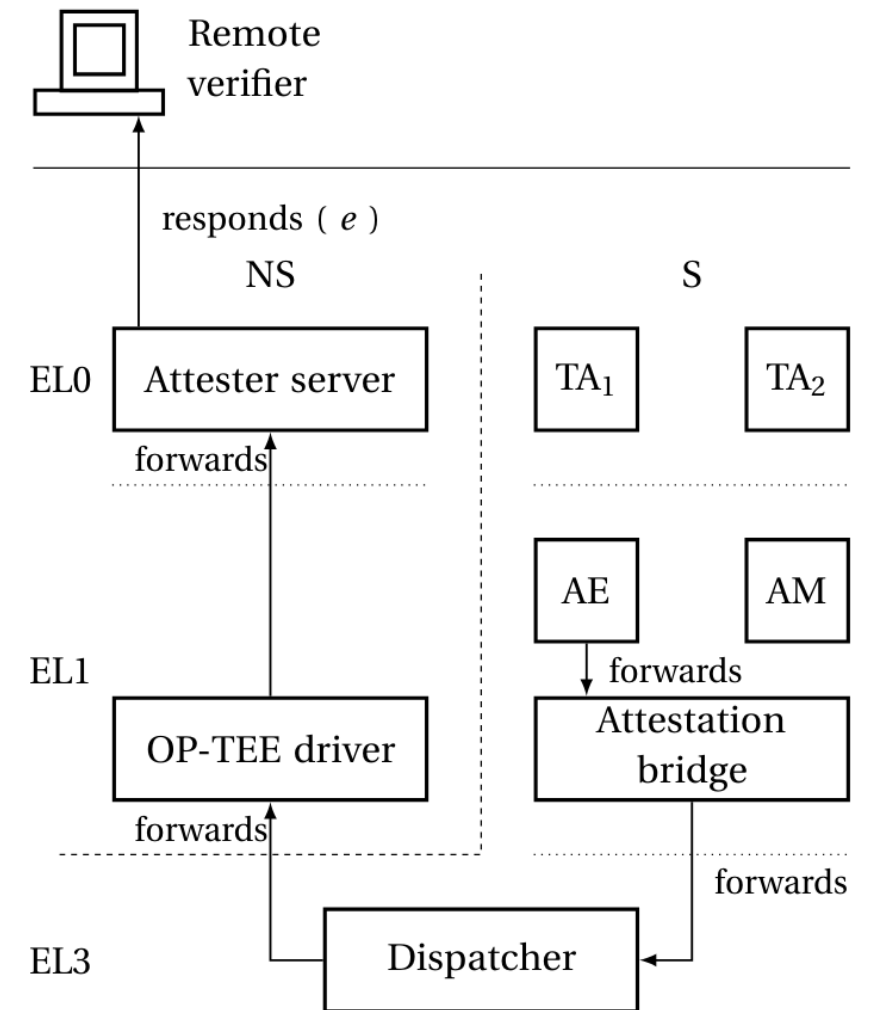


Design

The Attestation Engine

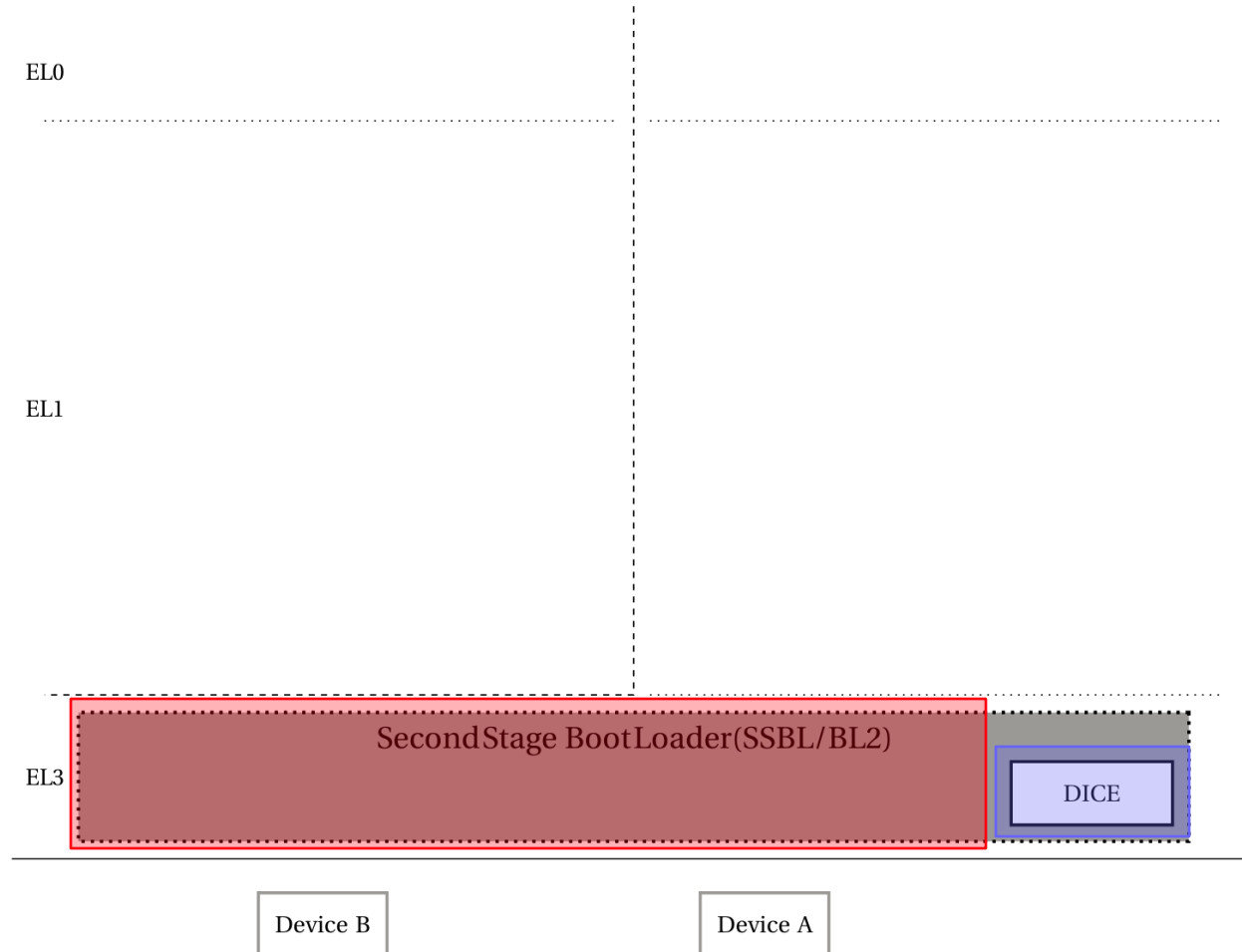
Attestation Process

1. Remote verifier sends challenge and task ID
2. Request forwarded by NS server
3. Attestation bridge in S forwards to the AE
4. AE collects claims:
 - (1) reads AM memory to produce fingerprint of task
 - (2) measures read-only sections in task's address space
5. AE signs (claims, challenge), and produces evidence
6. AE sends the evidence to the remote verifier



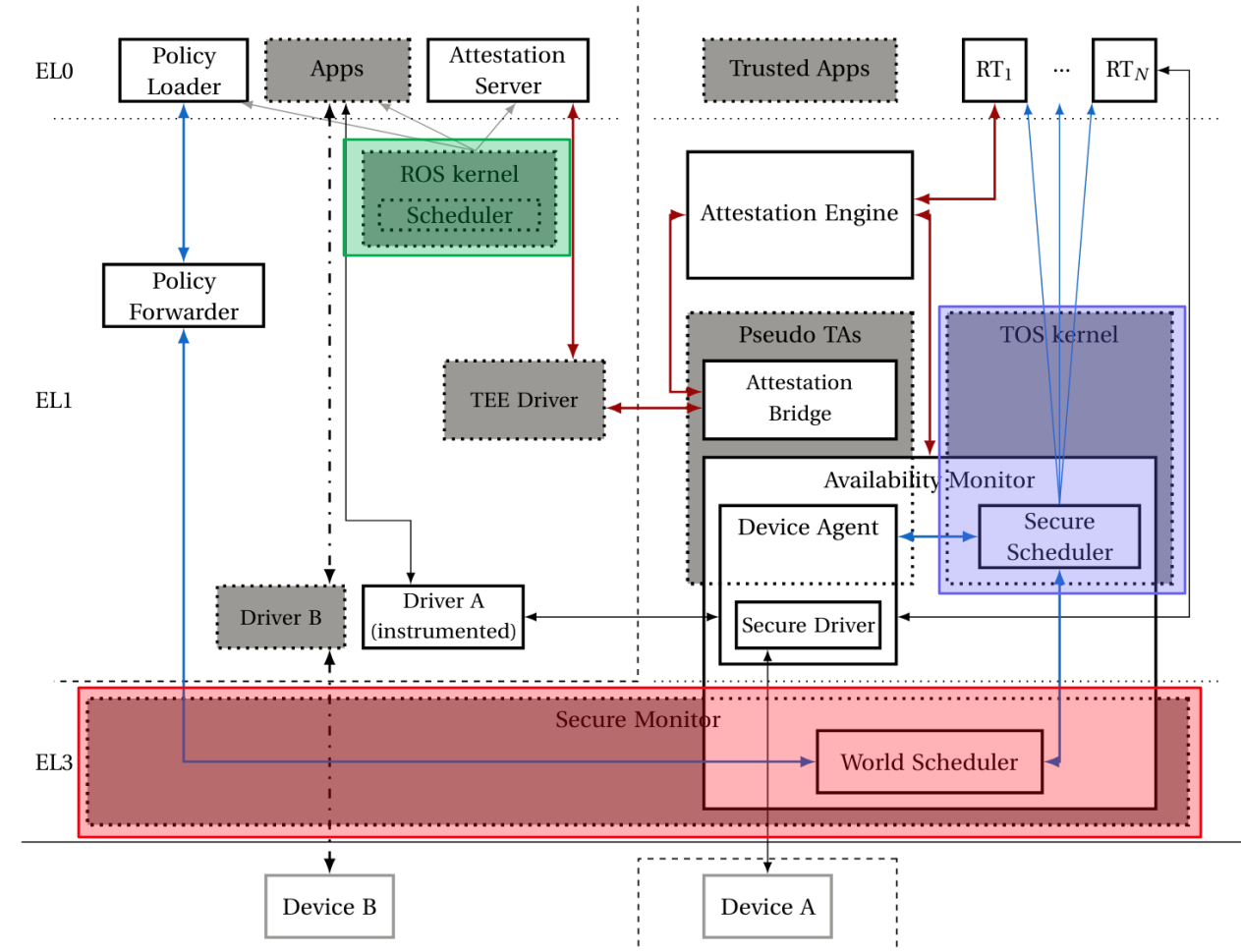
Implementation Setup

- Functional prototype on **TI TMDS64** evaluation kit
- DICE base layer in **U-Boot SPL 2025.01** (SSBL/BL2)
- **Google's OpenDICE** for reference DICE implementation
- **Arm Trusted Firmware-A v2.12.0** for EL3 Secure Monitor
- **OPTEE-OS v4.5** for S-EL1 kernel
- **Linux v6.12.17** with `PREEMPT_RT` for NS-EL1 kernel



Implementation Setup

- Functional prototype on **TI TMDS64** evaluation kit
- DICE base layer in **U-Boot SPL 2025.01** (SSBL/BL2)
- **Google's OpenDICE** for reference DICE implementation
- **Arm Trusted Firmware-A v2.12.0** for EL3 Secure Monitor
- **OPTEE-OS v4.5** for S-EL1 kernel
- **Linux v6.12.17** with `PREEMPT_RT` for NS-EL1 kernel



Implementation

Some Features

Critical RT Tasks

- Critical RT tasks implemented as OP-TEE TAs
- TA flag to protect main entry point
- System call to signal job completion

```
1 void ta_main_1_job(void)
2 {
3     // control loop with 1 job
4     for (;;) {
5         int measurement = sensor();
6         int result = compute(measurement);
7         actuate(result);
8
9         TEE_exit_job(0);
10    }
11 }
12
```

Evaluation

Performance, Security, Use cases

Evaluation on TI TMDS64 evaluation kit

- DICE base layer in **U-Boot SPL 2025.01** (SSBL/BL2)
- **Google's OpenDICE** for reference DICE implementation
- **Arm Trusted Firmware-A v2.12.0** for EL3 Secure Monitor
- **OPTEE-OS v4.5** for S-EL1 kernel
- **Linux v6.12.17** with `PREEMPT_RT` for NS-EL1 kernel
- 2x application cores (Cortex A53) running at 1.0GHz
- Ethernet ports for network connectivity

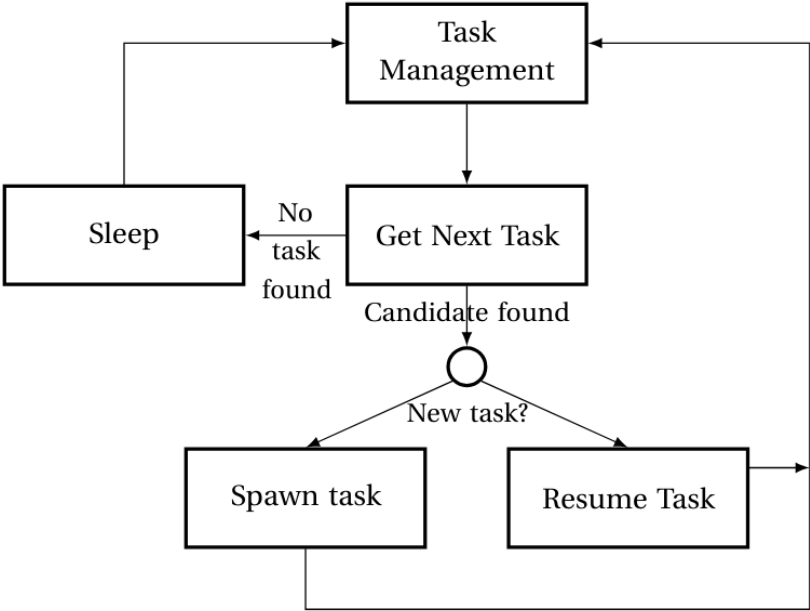


Evaluation

Performance

Secure Scheduler breakdown

Operation	Time (μs)
Task mgt <i>from idle/sleep</i>	0.3
Task mgt <i>from preempted task</i>	3.9
Get next task	0.2
Schedule task	0.4

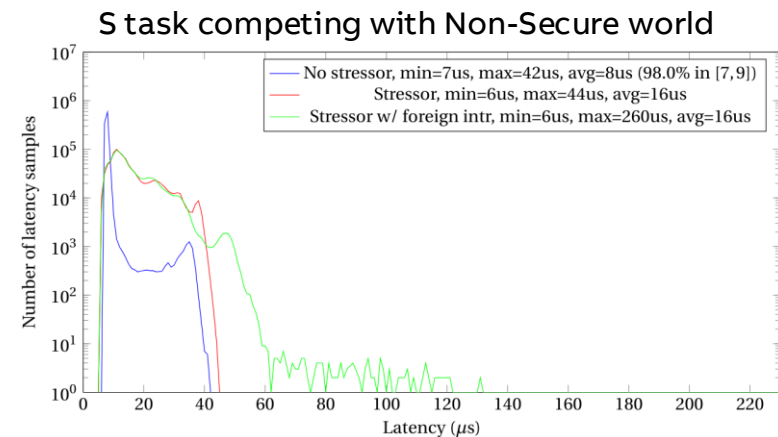
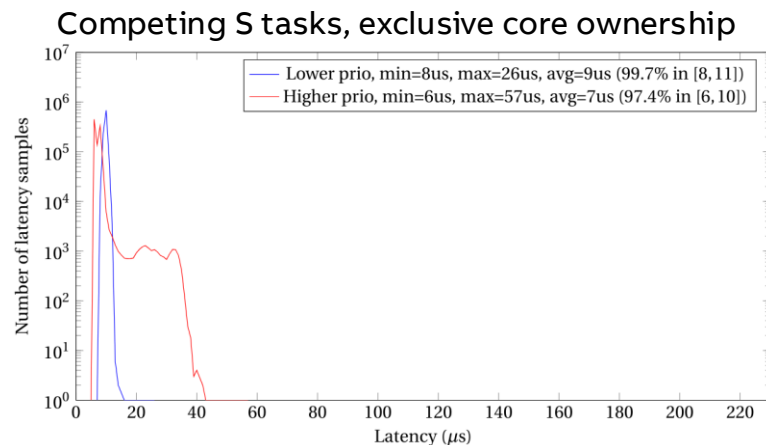
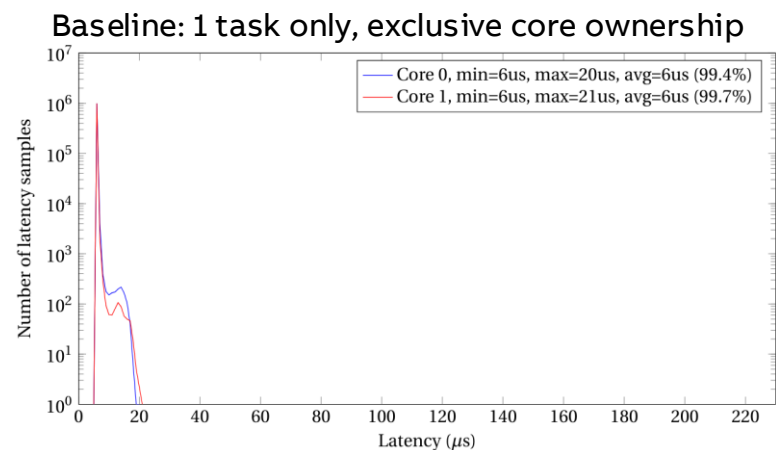


200 samples collected

Evaluation

Performance

Secure Scheduler Latency (custom *cyclictest*)



1,000,000 samples each time

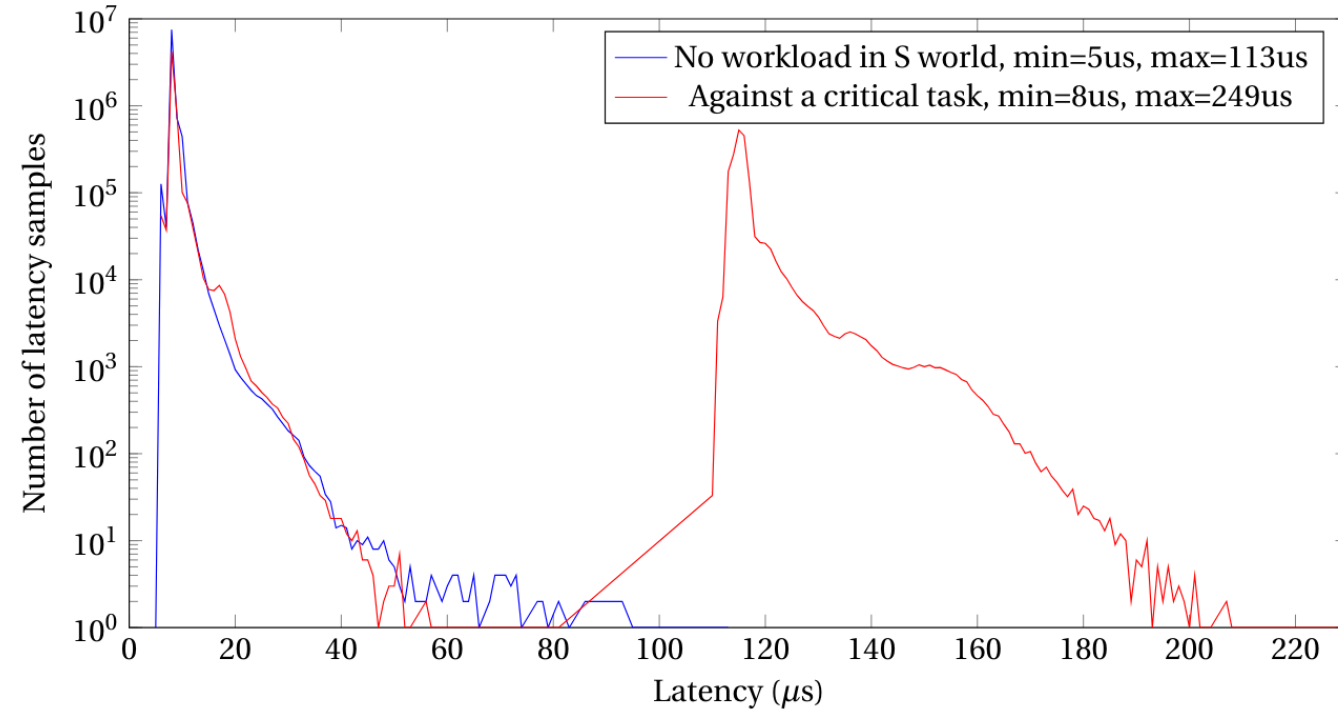
Stressor simulates DoS

Scheduling latency acceptable in all cases (below 60μs)

Evaluation

Performance

Non-Secure scheduler latency (NS and S tasks competing)



Critical task $T = (500, 100, C_0, \{\})$

8,999,999 and 7,199,984 samples collected

Evaluation

Performance

Real-Time tasks lifecycle breakdown

Only once,
When first loaded

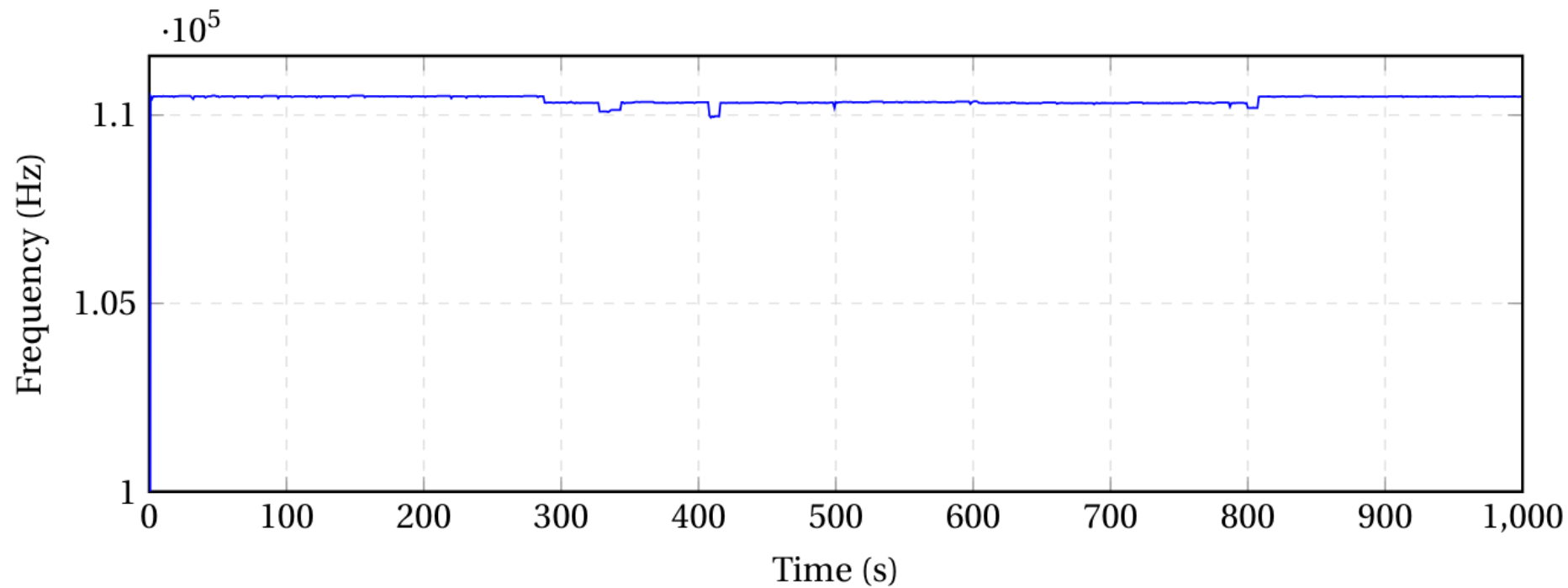
Operation	Description	Time (µs)
Allocation	Retrieve the task binary from memory and load it	15115
First entry → exit	First entry is done via a specialized TEE_InvokeCommand()	50
Re-entries → exit	Subsequent re-entries are done by restoring context	4

10 samples collected

Evaluation

Performance

Analysis of max. frequency for real-time tasks

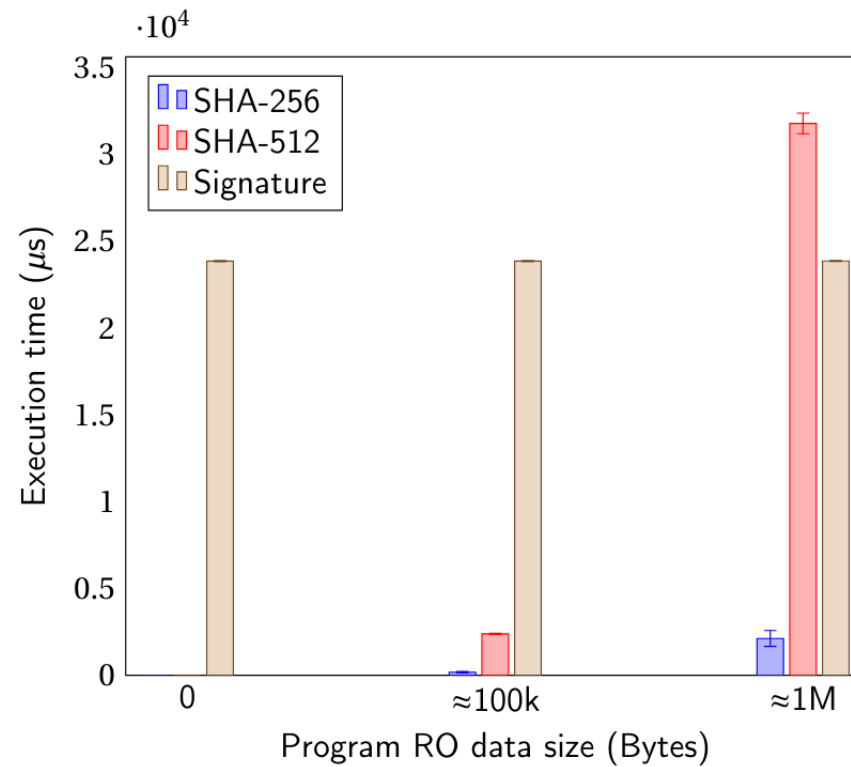


~110kHz $\approx 9\mu\text{s}$

Evaluation

Performance

Performance of the attestation for different tasks size and algorithms



Evaluation

Security

Number of added lines of code in the TCB

Description	LoC	
U-Boot SPL	773	}
DICE Engine	572	
DICE Integration	124	
Firewall Driver	77	
Trusted Firmware-A	614	}
World Scheduler	470	
DICE Service	123	
OP-TEE SPD	21	
OP-TEE	2069	
Scheduler	953	
Device Agents	322	
Secure Drivers	117	
Attestation Engine	325	
Memory Management	5	
Thread Management	155	
TA Management	24	
OP-TEE Entry	130	
Crypto Library	38	

Discarded when entering TFA (BL2 → BL3)

Evaluation

Security

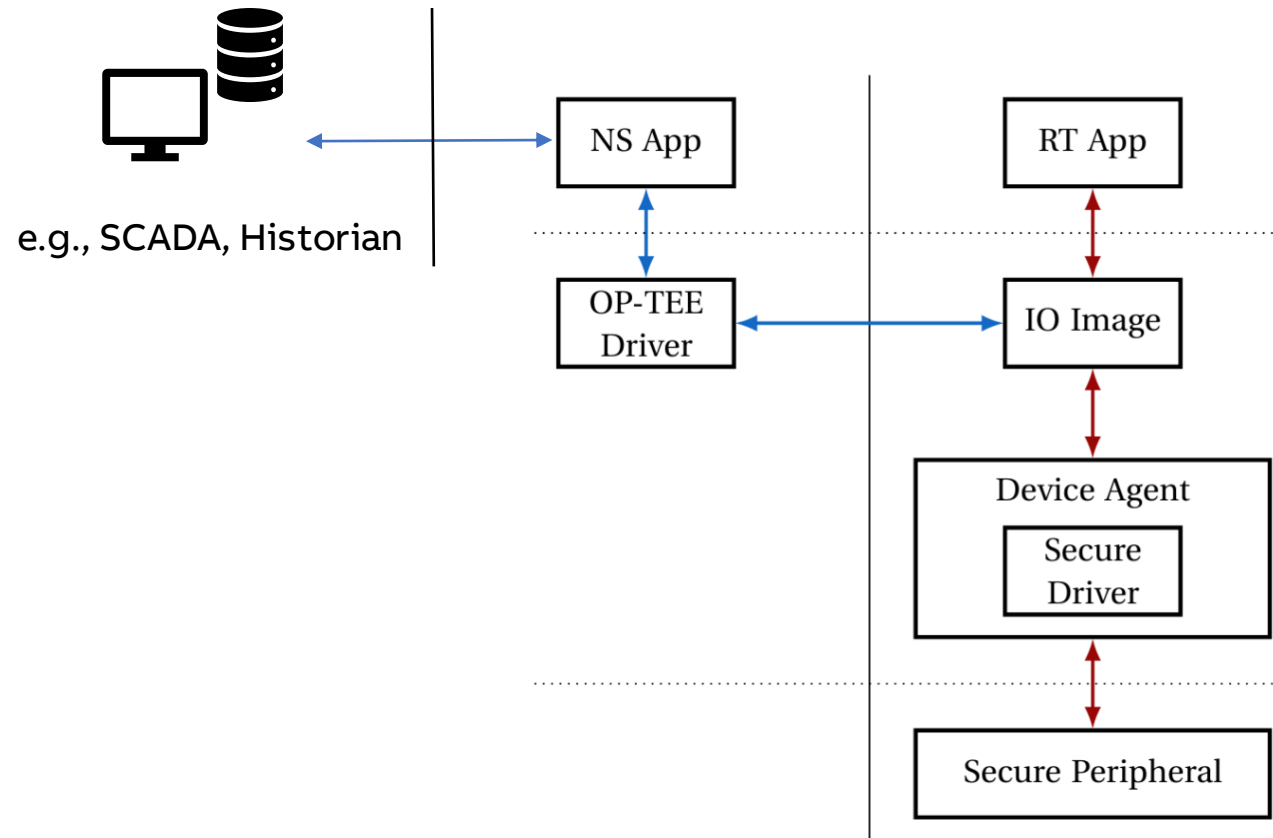
Mitigation of real-world attacks by technique (from MITRE ATT&CK Matrix for ICS)

- Hardcoded credentials leakage
- Illegal program modifications
- Denial of Service (DoS)
- Loss of protection (protection functions illegally disabled)
- Rootkit
- Alter device configuration (e.g., via IO Image)

Evaluation

Use Cases

Sample industrial setup



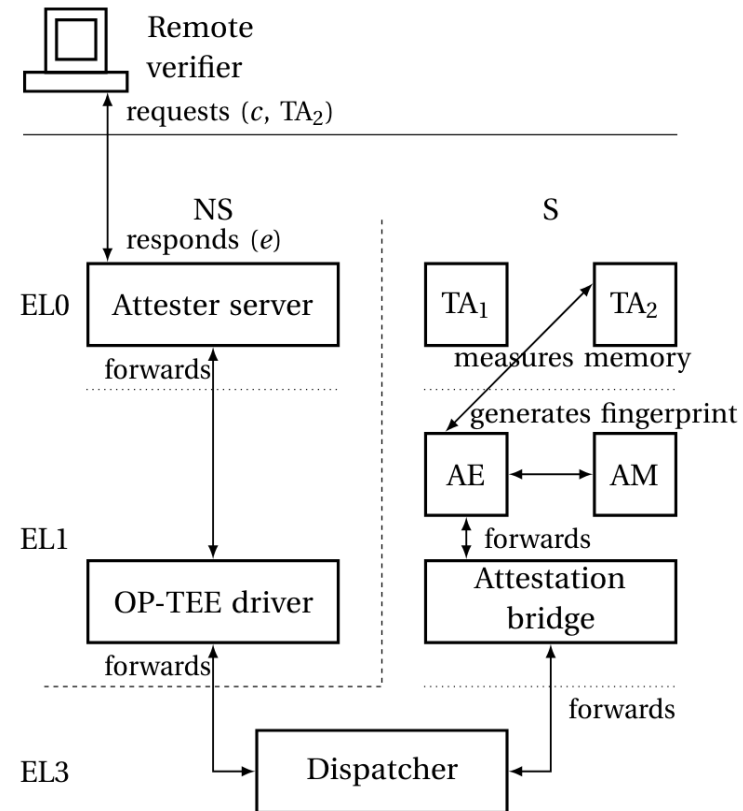
Evaluation

Use Cases

Remote verifier has:

- AE's public key
- Golden measurements

End-to-end remote attestation



Conclusion

Summary of contributions

Novelty: ensured and verifiable availability (on open, multiprocessor, off-the-shelf system).

- **Availability Monitor** (World & Secure schedulers, Device Agents, split-driver design)
- **Attestation Engine** (availability and integrity measurements)
- **Secure provisioning of policies by user**
- **Confidentiality and integrity of IP**
- **Prototype implementation**
- **Evaluation of the architecture** (Performance, Security, Use cases)

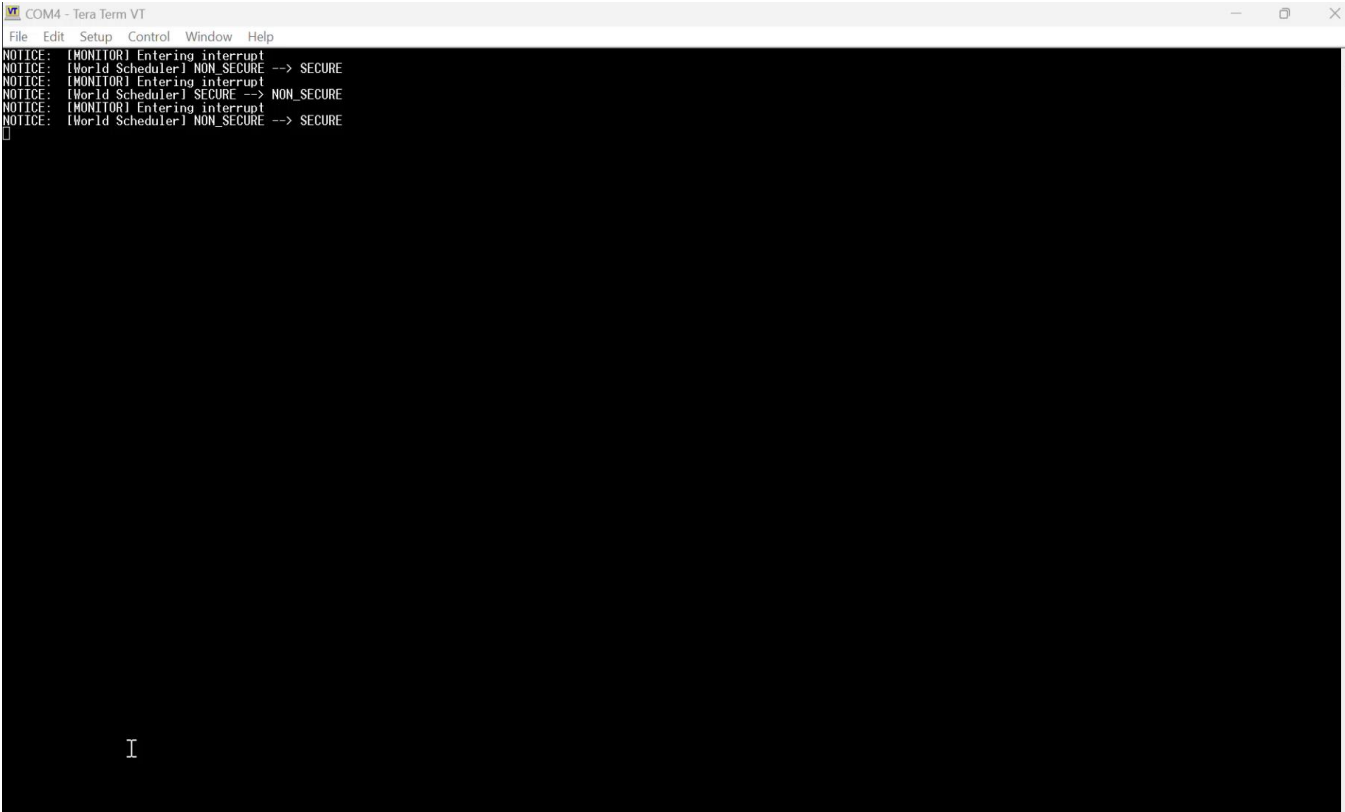
Not shown here

- Survey of (~20) fitting development boards
- Guide to building critical tasks for our scheduler
- Comprehensive background
- Analysis of related works

Evaluation

Demo

DICE and scheduling of worlds



The screenshot shows a Tera Term VT window titled "COM4 - Tera Term VT". The window contains a log of system events related to DICE and world scheduling. The log entries are as follows:

```
NOTICE: [MONITOR] Entering interrupt  
NOTICE: [World Scheduler] NON_SECURE --> SECURE  
NOTICE: [MONITOR] Entering interrupt  
NOTICE: [World Scheduler] SECURE --> NON_SECURE  
NOTICE: [MONITOR] Entering interrupt  
NOTICE: [World Scheduler] NON_SECURE --> SECURE
```

The rest of the window is black, with a single cursor character "I" visible at the bottom left.

ABB