차세대 기밀 컴퓨팅

새로운 보안 요구와 기술적 도전

Rethinking Trust and Resource Sharing in Confidential Computing

2026 Sejong Cybersecurity Workshop

서울대학교 컴퓨터연구소

유 준 승

Contents

1 Confidential Computing

2 Trusted Execution Environments

3 Problem: Trust Boundary

4 Problem: Resource Sharing

Contents

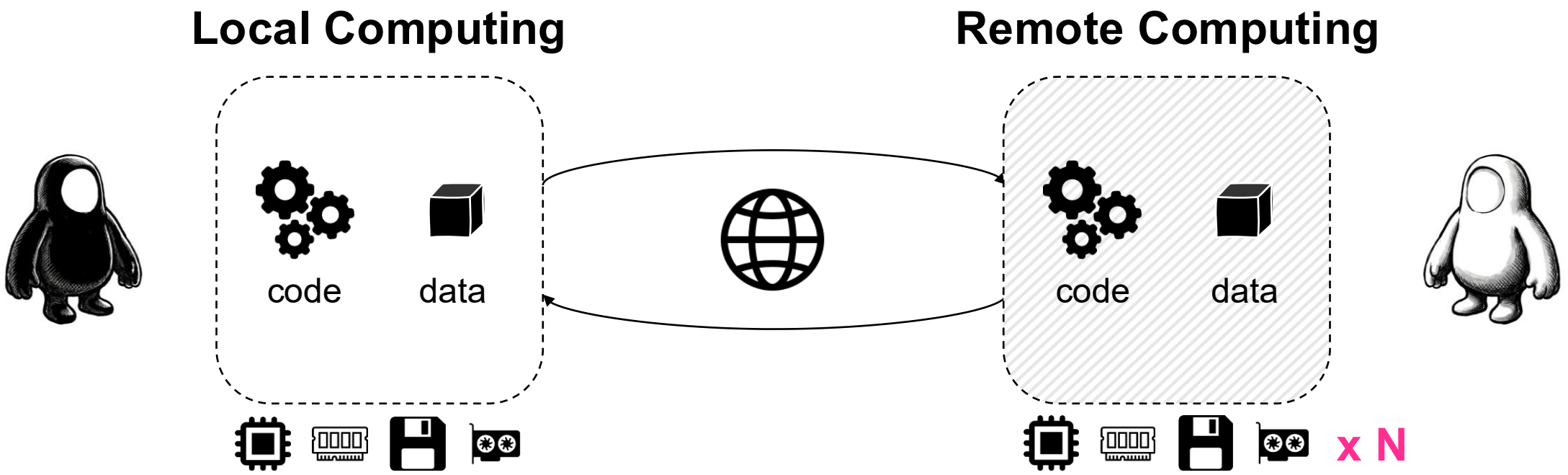
1 Confidential Computing

2 Trusted Execution Environments

3 Problem: Trust Boundary

4 Problem: Resource Sharing

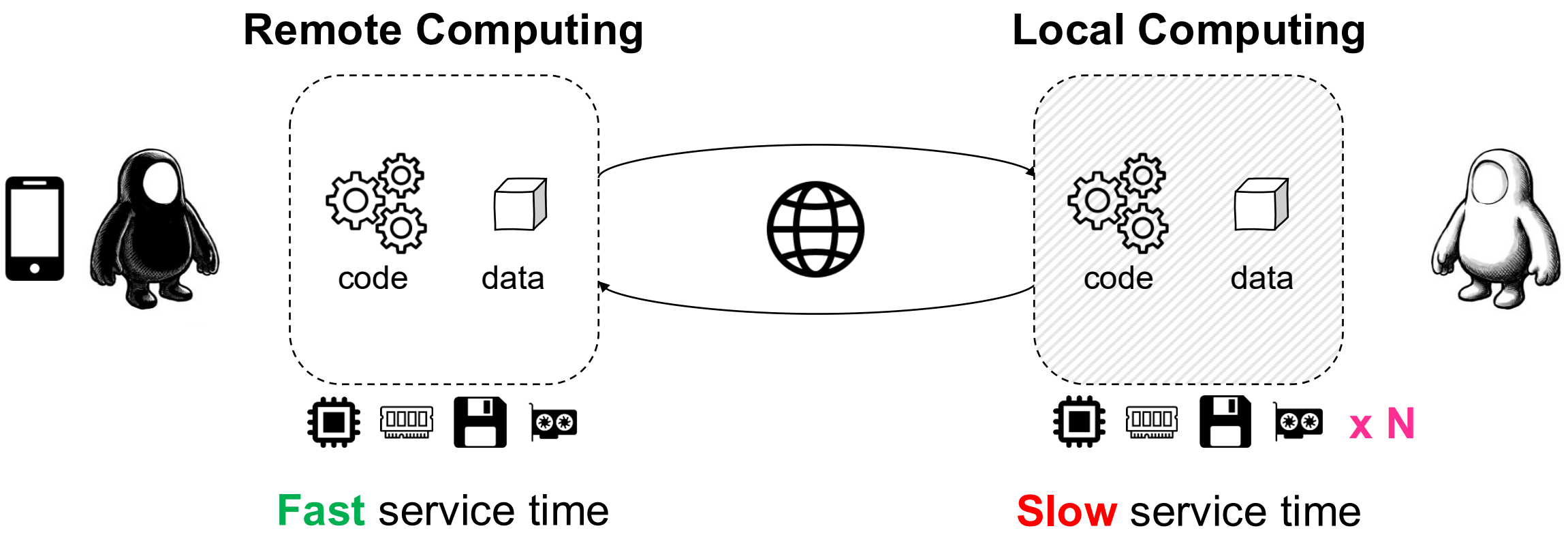
Remote Computing



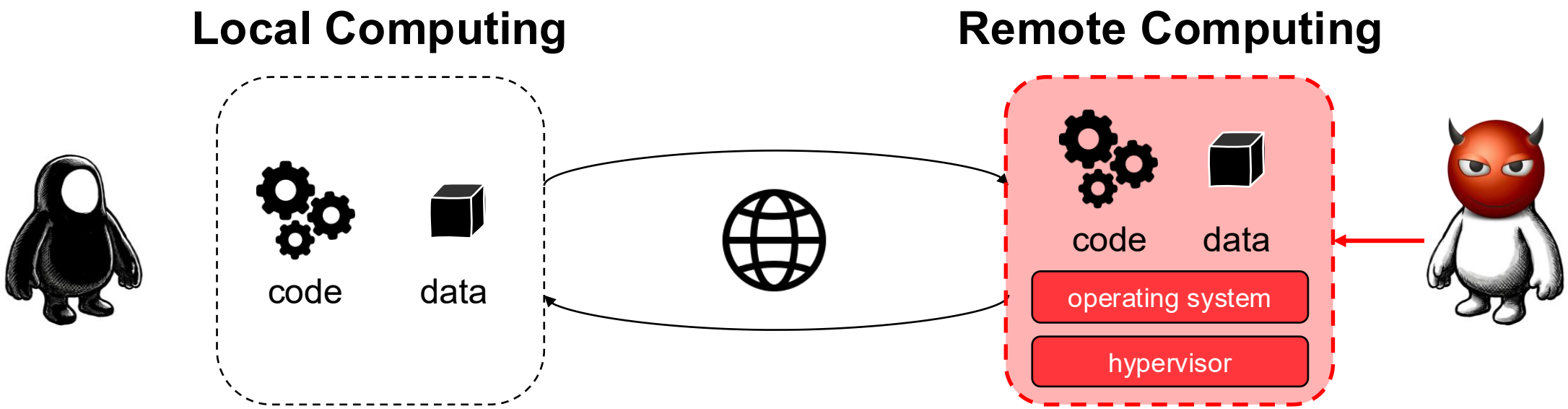
Limited resources
Slow computation

Sufficient resources
Fast computation

Remote Computing

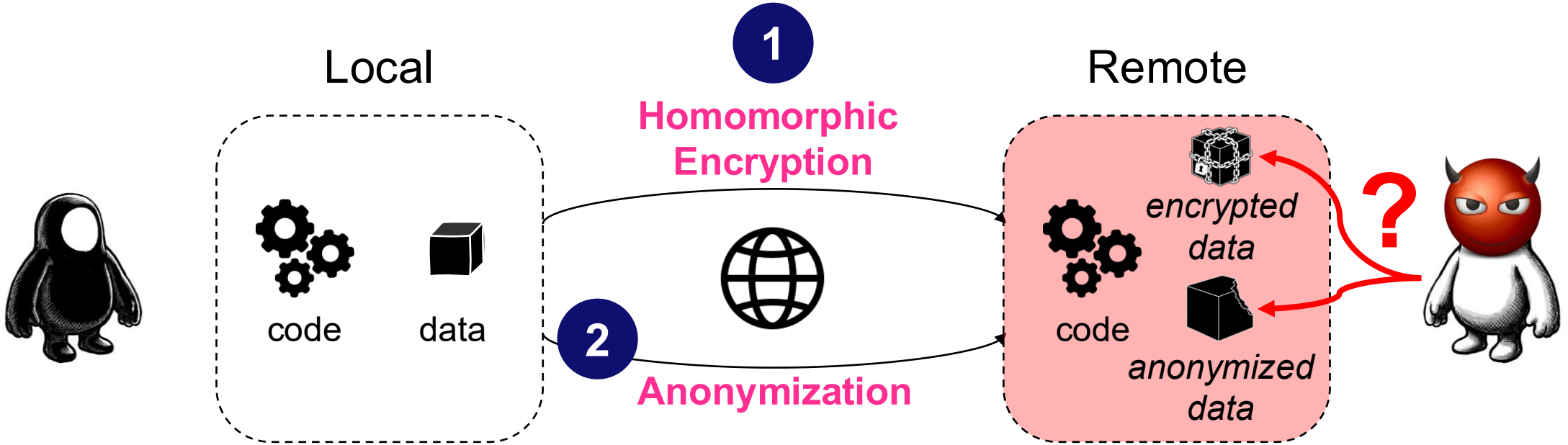


Remote Computing



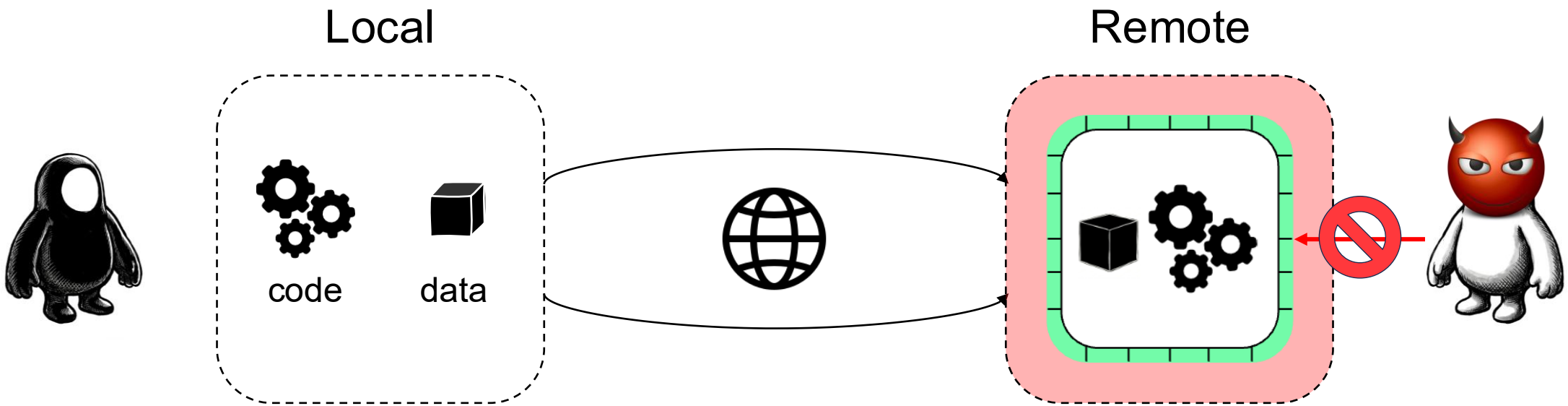
Proprietary asset send over to remote platform is vulnerable to disclosure

Possible Solutions



	Homomorphic Encryption	Anonymization	???
Performance	Low	Medium	High
Data Utilization	High	Low	High

Confidential Computing



The protection of data-in-use by performing computation in a hardware-based, attested ***Trusted Execution Environment***

- Confidential Computing Consortium (CCC) -

Contents

1 Confidential Computing

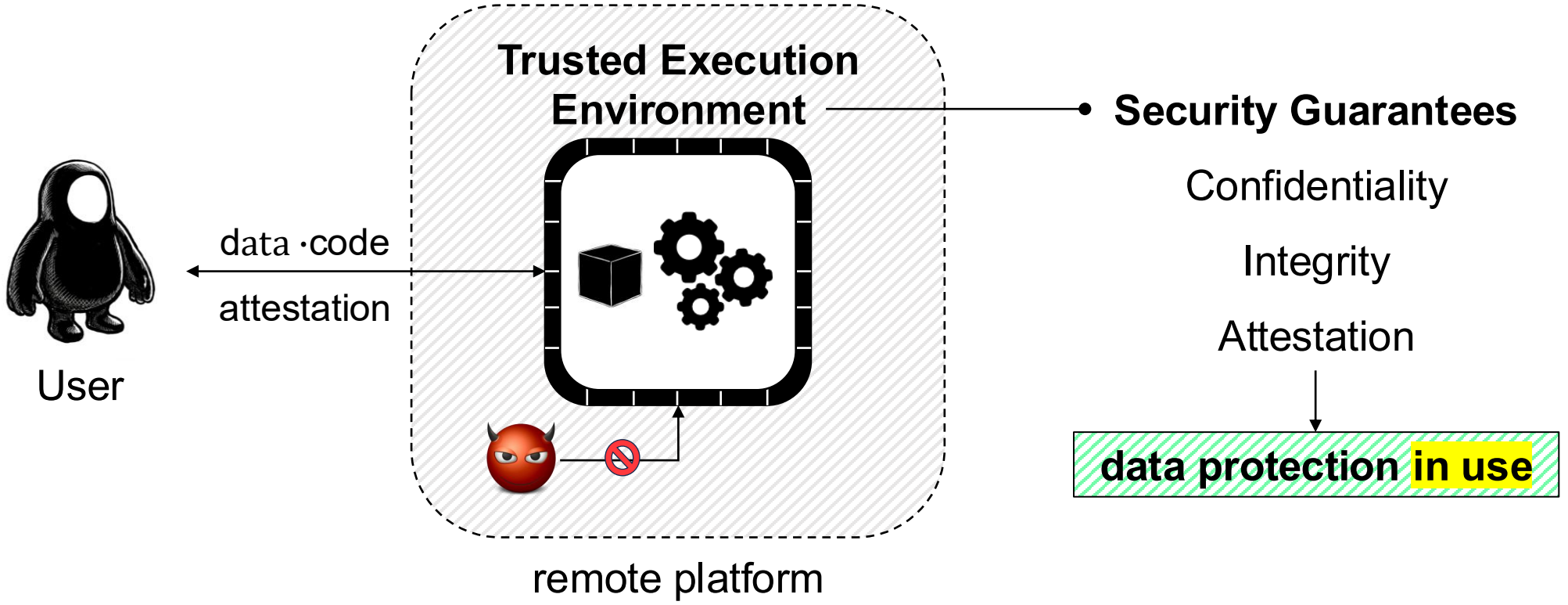
2 Trusted Execution Environments

3 Problem: Trust Boundary

4 Problem: Resource Sharing

Trusted Execution Environment (TEE)

- Personal **safe** in remote environment



Types of Trusted Execution Environments

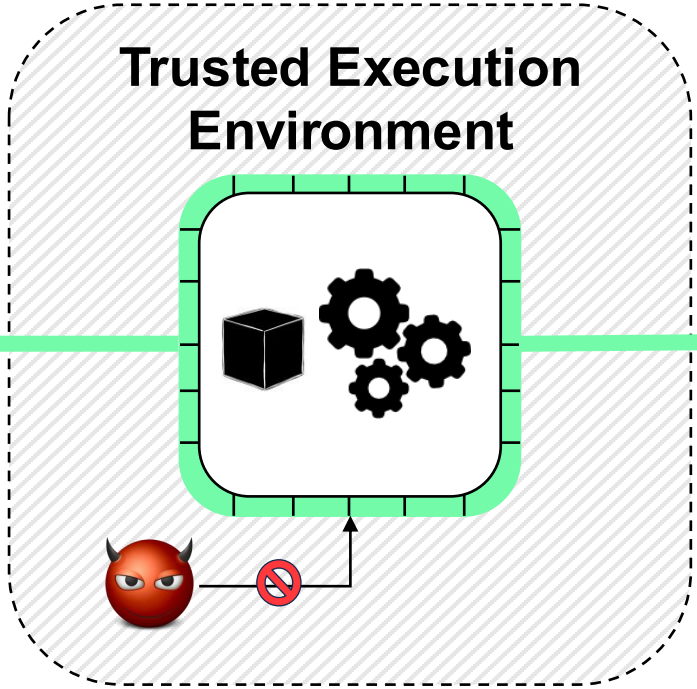
SW based TEEs

operating system, hypervisor

trust system SW with high privilege

no hardware dependency

large trusted computing base (TCB)



remote platform

HW based TEEs

hardware, processor(CPU)

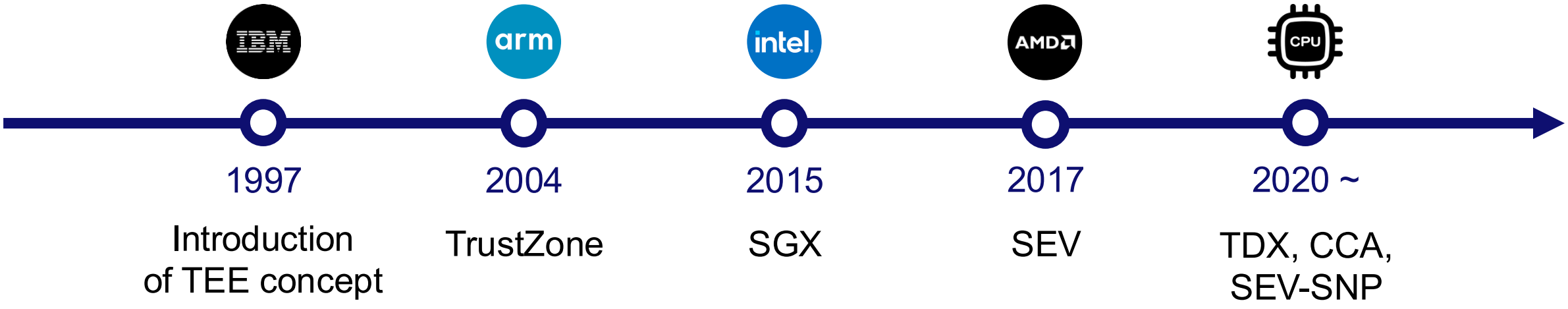
trust *only* the hardware

requires support from CPU vendor(s)

small TCB, **near-native** performance

"Keep the trusted base as simple and small as possible"

Evolution of HW-based TEEs



Evolution Trends

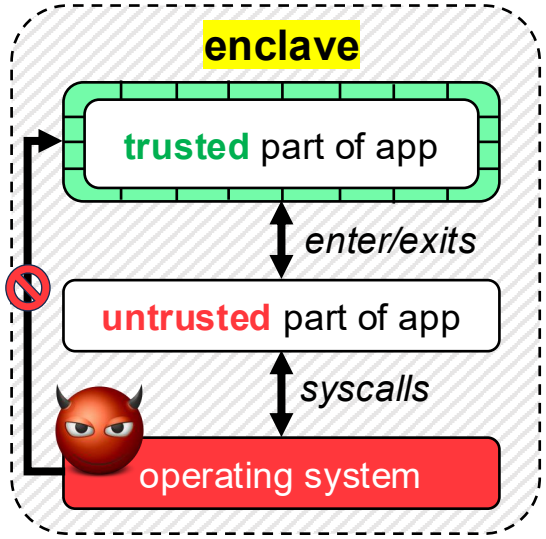
Mainstream Support – TEE extension and support from major CPU architectures / ISAs

Availability – TEE-based instances supported by major cloud platforms (e.g., AWS, Google)

Granularity of Protection – Larger protection scope from application-level to VM-level

Protection Boundary Expansion of TEEs

Application-level TEEs



remote platform

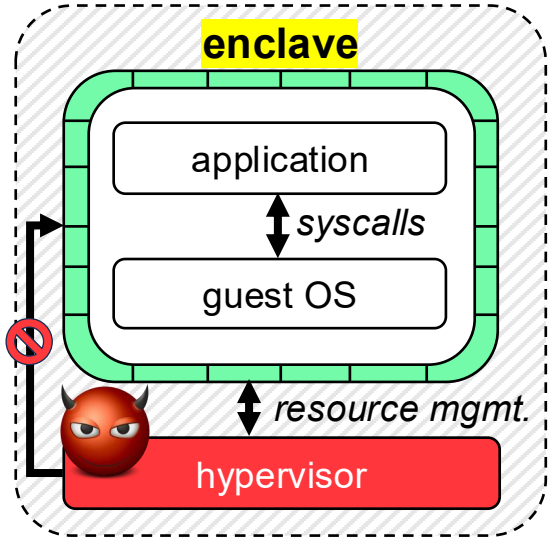
Intel SGX

designed to protect small, sensitive part of the app

small size (128MB), **high** overhead (~ 100x)

Virtual Machine (VM)-level TEEs

Intel TDX
AMD SEV
ARM CCA



remote platform

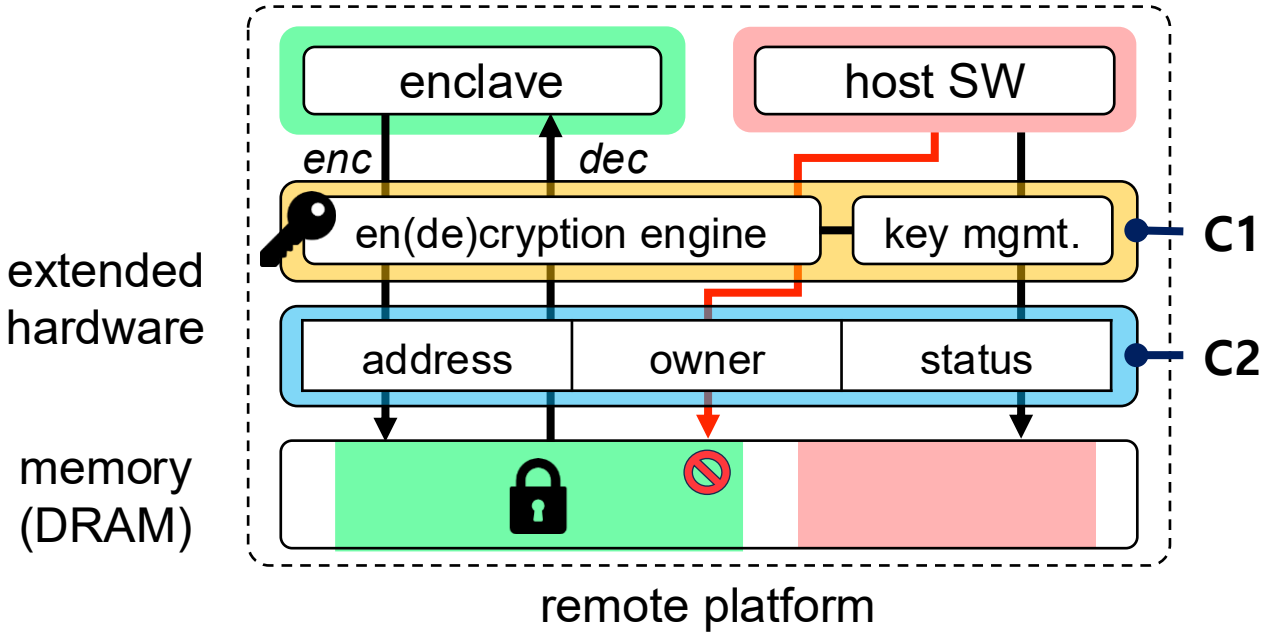
designed to protect the app at VM granularity

large size, **low** overhead (< 10%)

Hardware-enforced Protection of TEEs

① Hardware conducts **en(de)cryption** during memory access

② Hardware conducts **permission check** during memory access



Component 1 – Memory Encryption

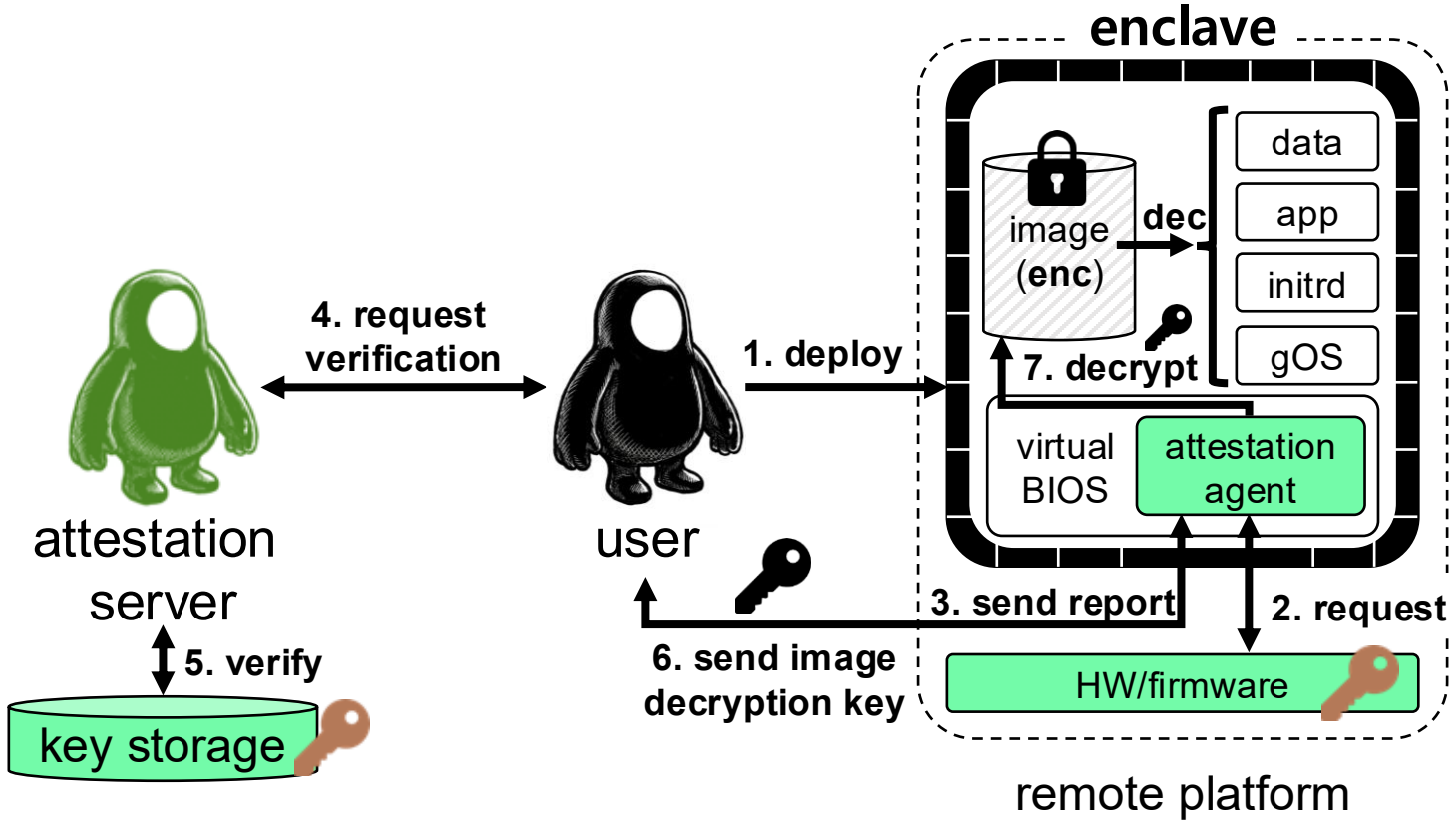
Data from memory is decrypted during load and encrypted during store (AMD SME, Intel MKTME)

Component 2 – Ownership Tracking

Each physical memory page is assigned to a single owner. Ownership is checked in the occurrence of memory access (AMD RMP, ARM GPT)

TEE Provisioning

- Hardware serves as **root-of-trust**



We want to **verify**

1. **Identity** of the application
2. **Validity** of the platform

- launch/boot measurements
- security policy/configuration
- platform TCB version
- signature/MAC

Contents

1 Confidential Computing

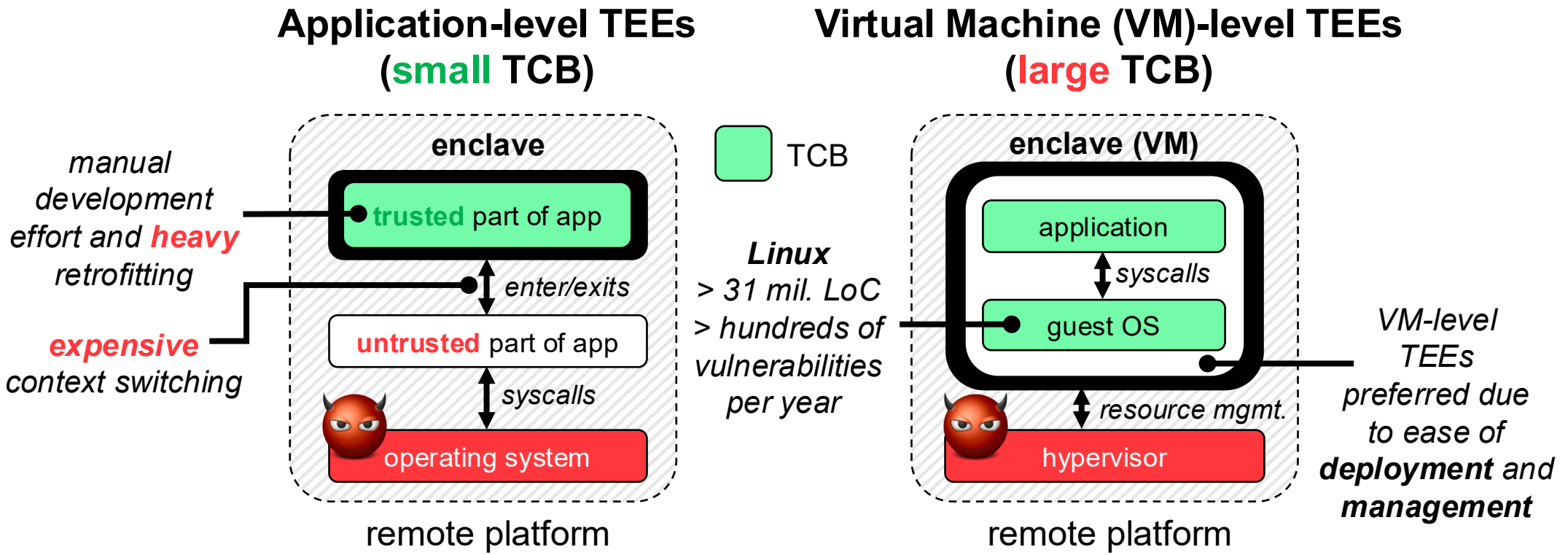
2 Trusted Execution Environments

3 **Problem: Trust Boundary**

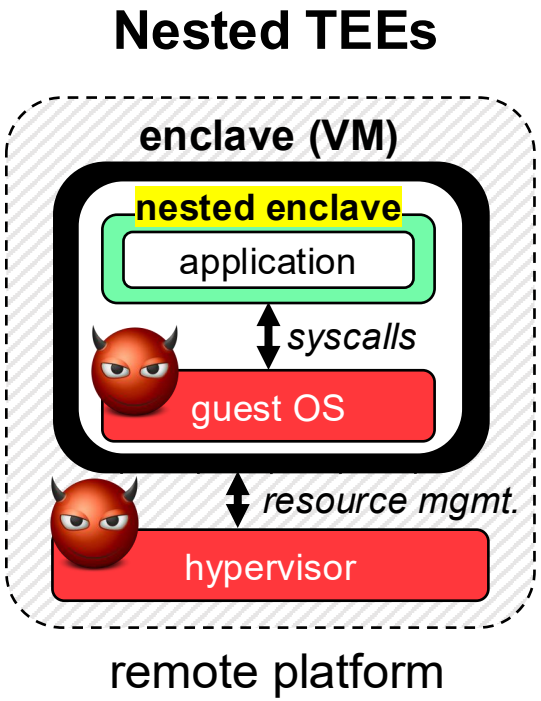
4 Problem: Resource Sharing

Trusted Computing Base of TEEs

- Guest OS remains as a large attack surface



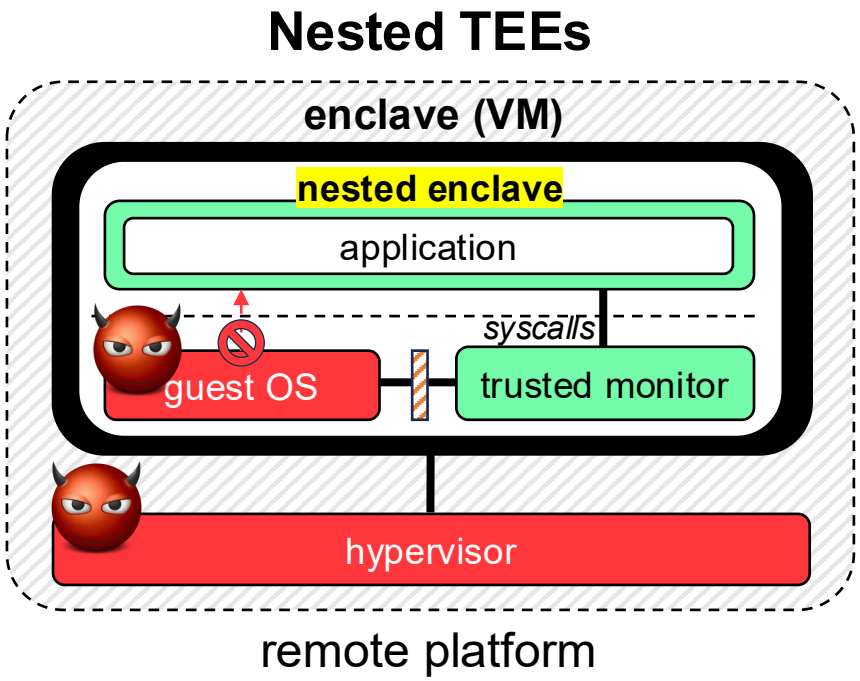
Remediation: Nested TEE(s)



Best of both ends…

- **Deployment** at VM-level TEE
- **Protection** at application-level TEE

Strawman Solution for Nested TEEs

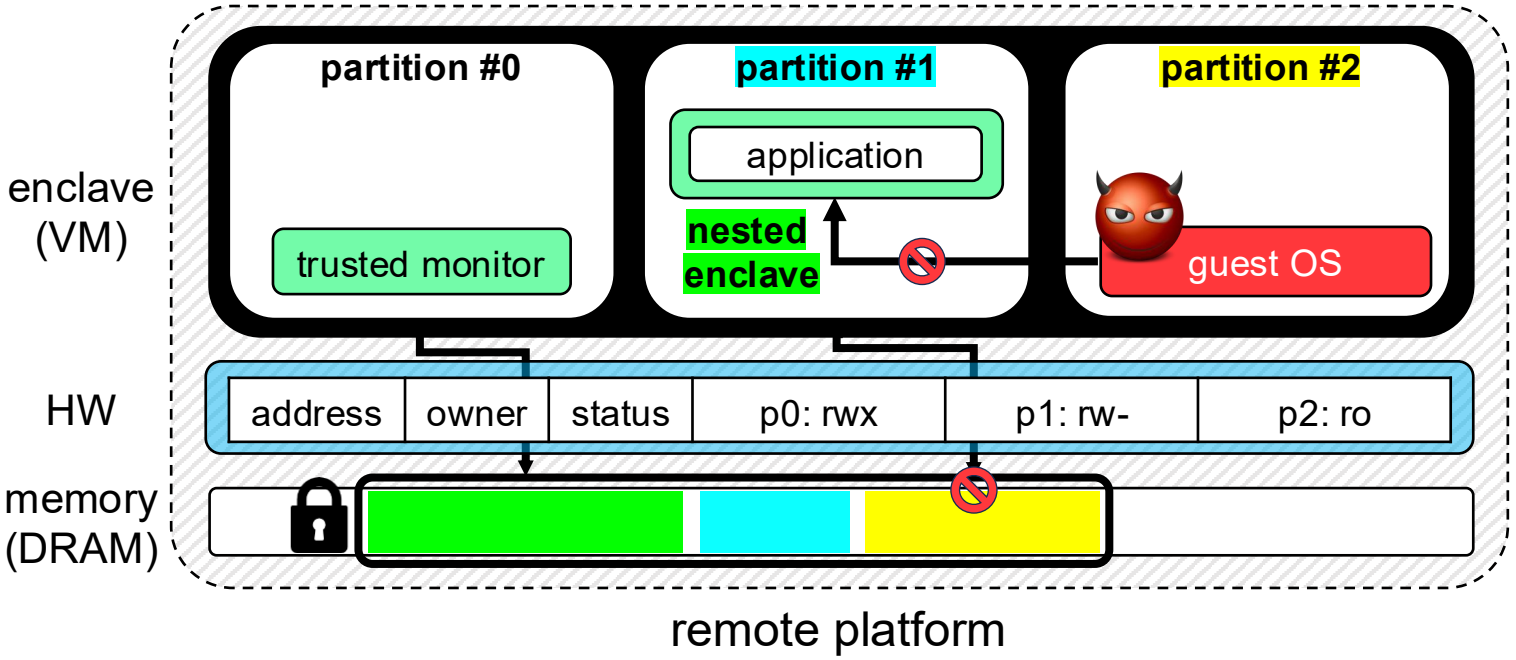


Intra-kernel Privilege Separation

- Decompose the kernel into trusted and untrusted part
- **Trusted monitor** enforces isolation and communication between the guest OS and the application
- Requires **custom kernel** with security hardening
  - Full control-flow integrity (forward + backward)
  - Sensitive/privileged instruction instrumentation

Privilege Layering for Nested TEEs

- Hardware-assisted intra-TEE privilege separation
  - Partitions the address space with separate access permissions from different privilege levels
  - Partition with the highest privilege manages the access permissions of other partitions

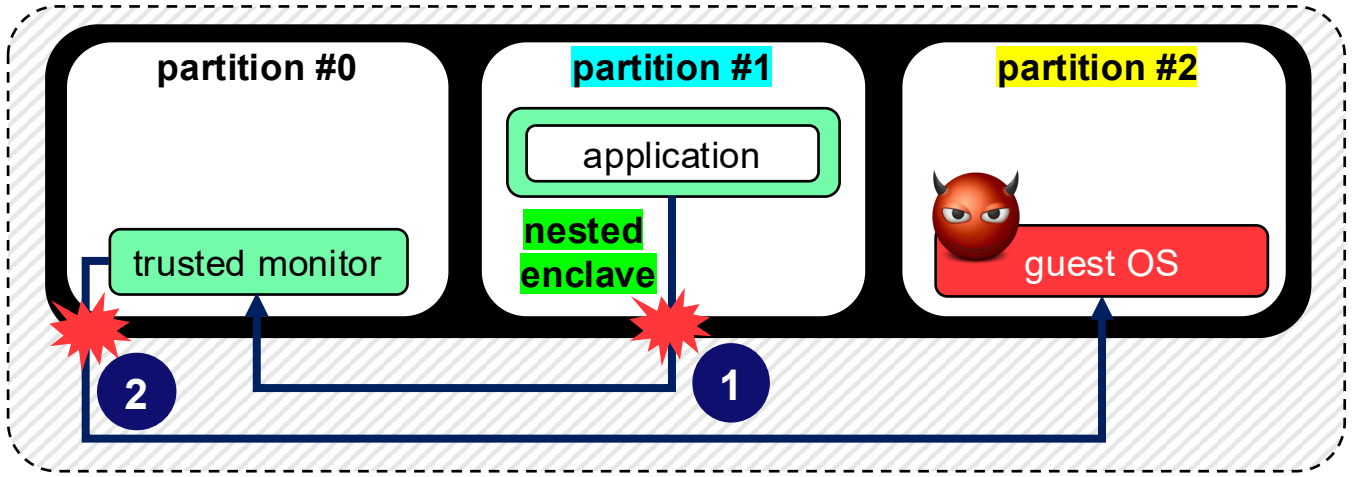


Supported by…

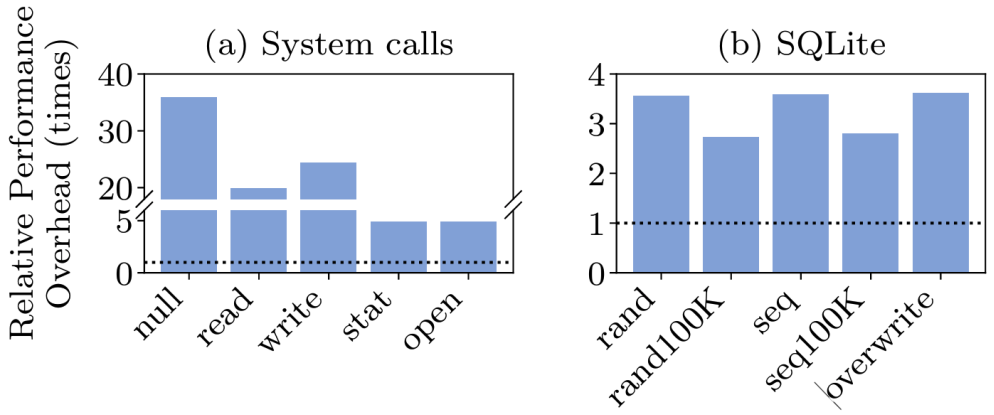
- AMD SEV VMPLs
- ARM CCA Planes
- Intel TDX TD Partitioning

Performance of Nested TEEs

- Application still requires guest OS for execution
  - Placing app and guest OS in separate partitions induces `vmexit`(s) per system call
  - High performance degradation on real-world applications



How can we realize nested TEEs both securely and efficiently?



Relative performance overhead on representative system calls and SQLite

Contents

1 Confidential Computing

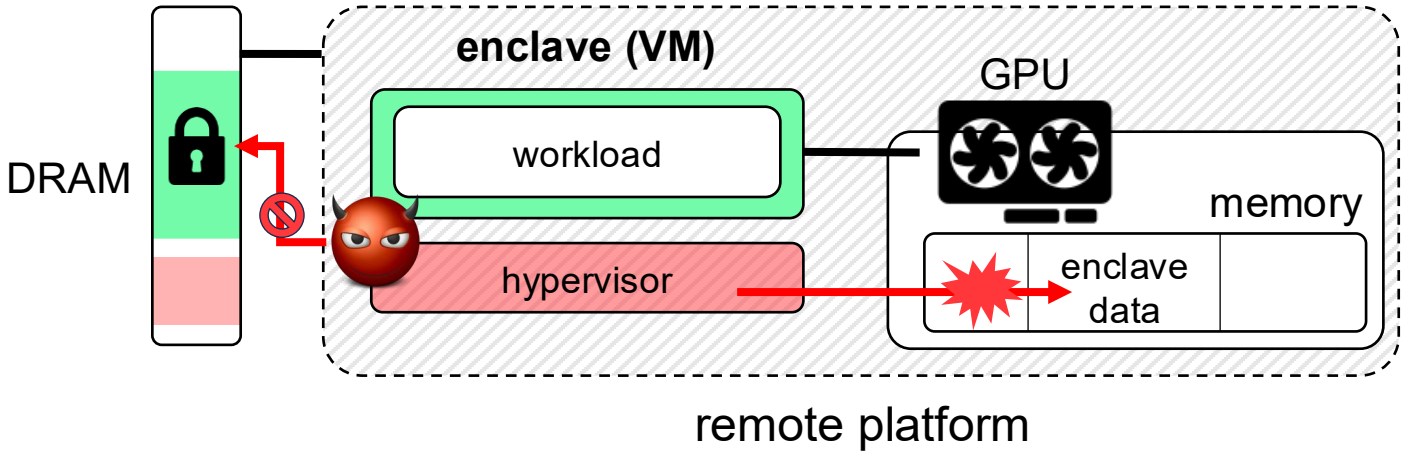
2 Trusted Execution Environments

3 Problem: Trust Boundary

4 **Problem: Resource Sharing**

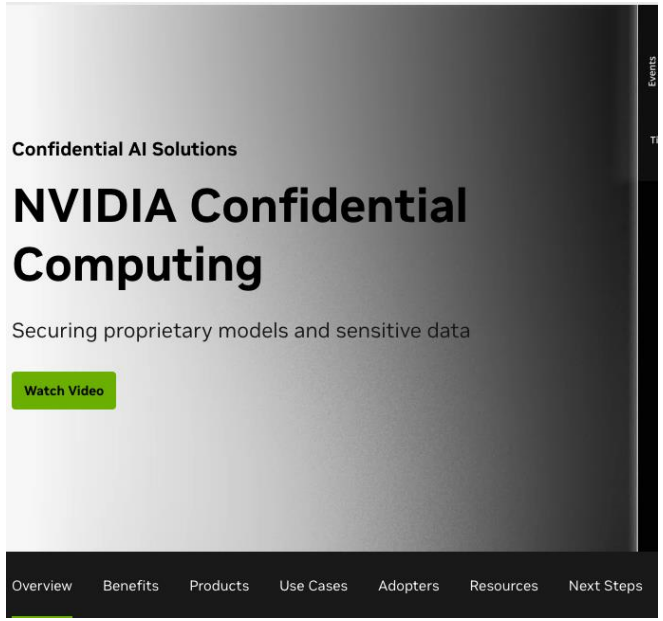
Protection Boundary of TEEs

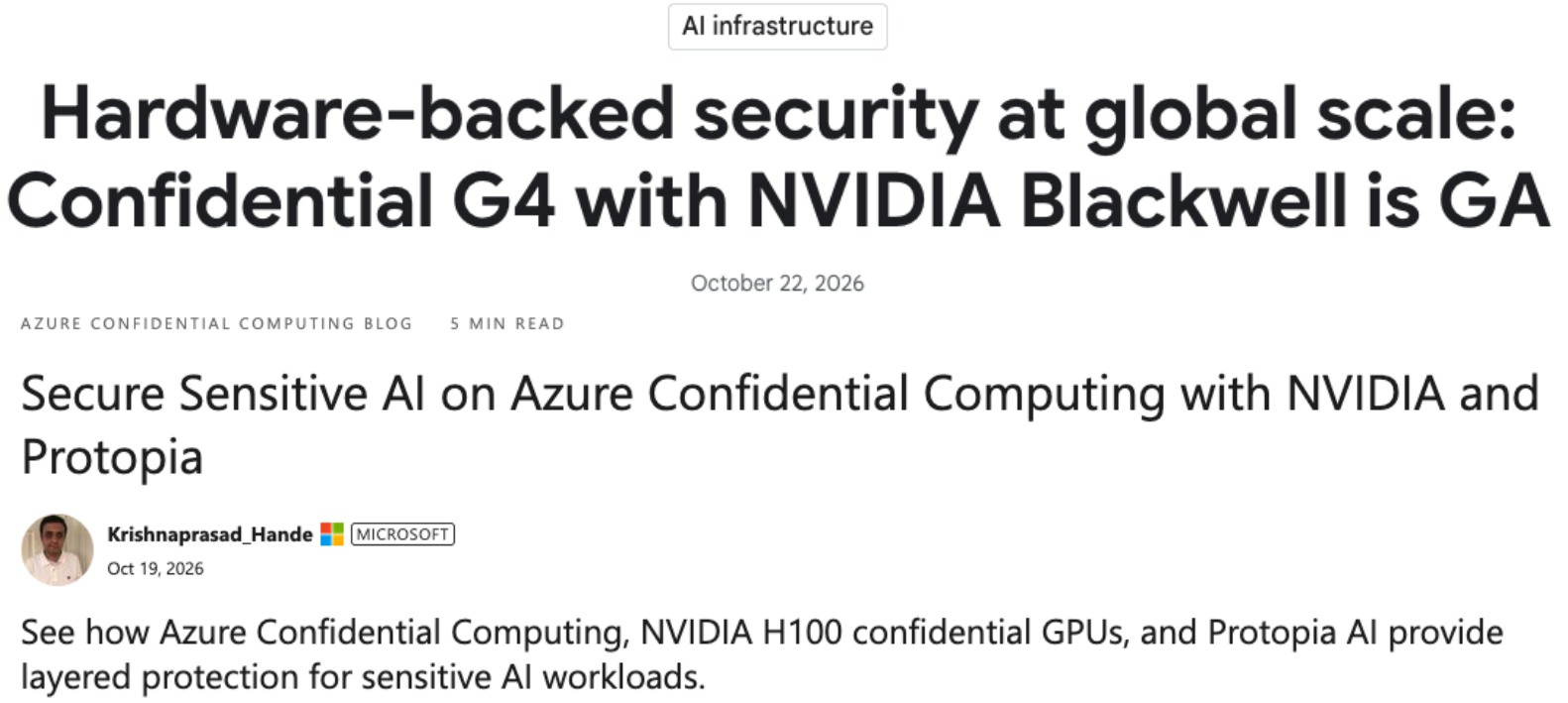
- CPU-Confidential Computing (CC) : secures computation within CPU package
- Does not protect device memory (e.g., GPUs for AI workloads)



NVIDIA Confidential Computing

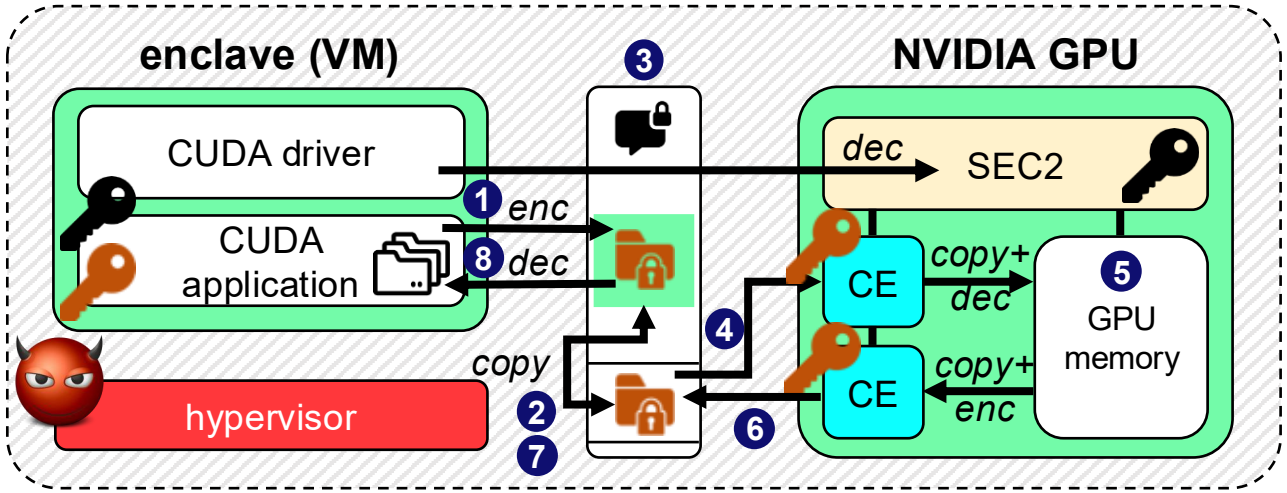
- GPU-CC : extend the trust boundary from CPUs to GPUs
  - Starting with NVIDIA Hopper architecture (2023, H100/200/300, B100/200, …)
  - Sharp increase in industrial/academic interest on utilization of TEEs for securing AI pipelines





NVIDIA CC Demystified

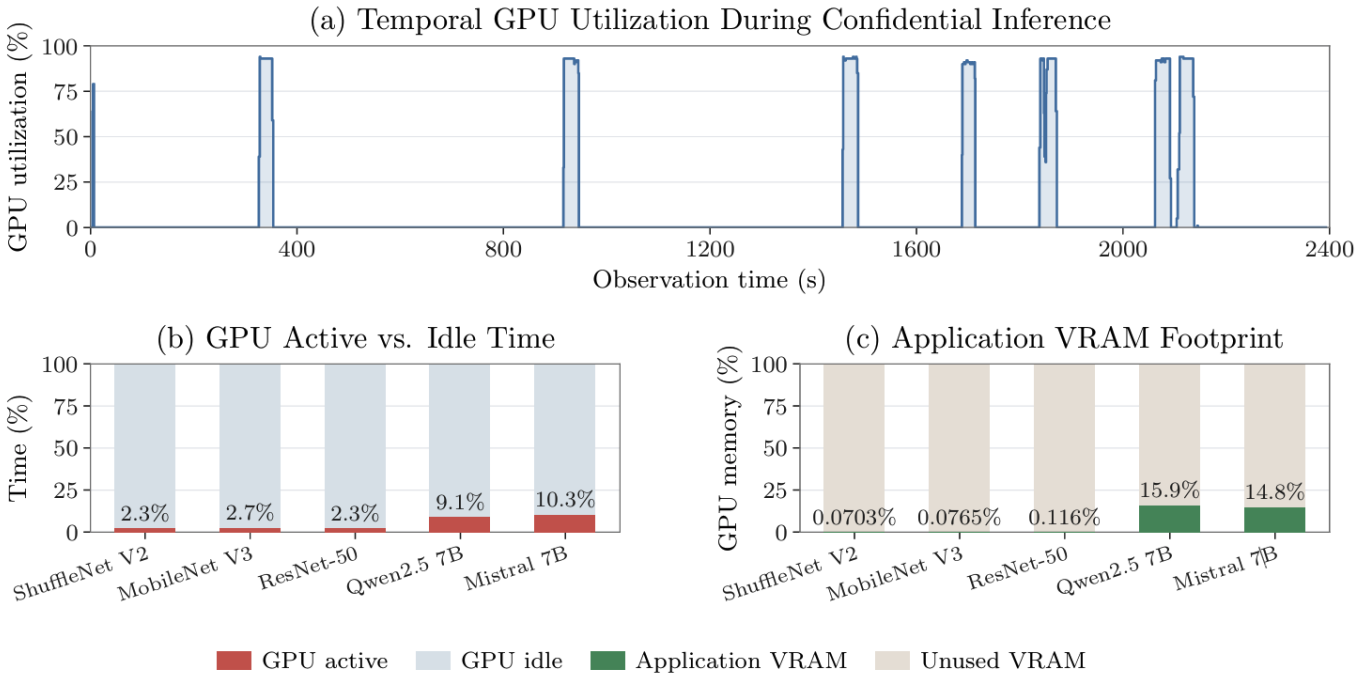
- Set of hardware and firmware components for CC
  - Secure Processor (SEC2): Secure control message transmission
  - Copy Engine (CE): Secure data transmission



remote platform

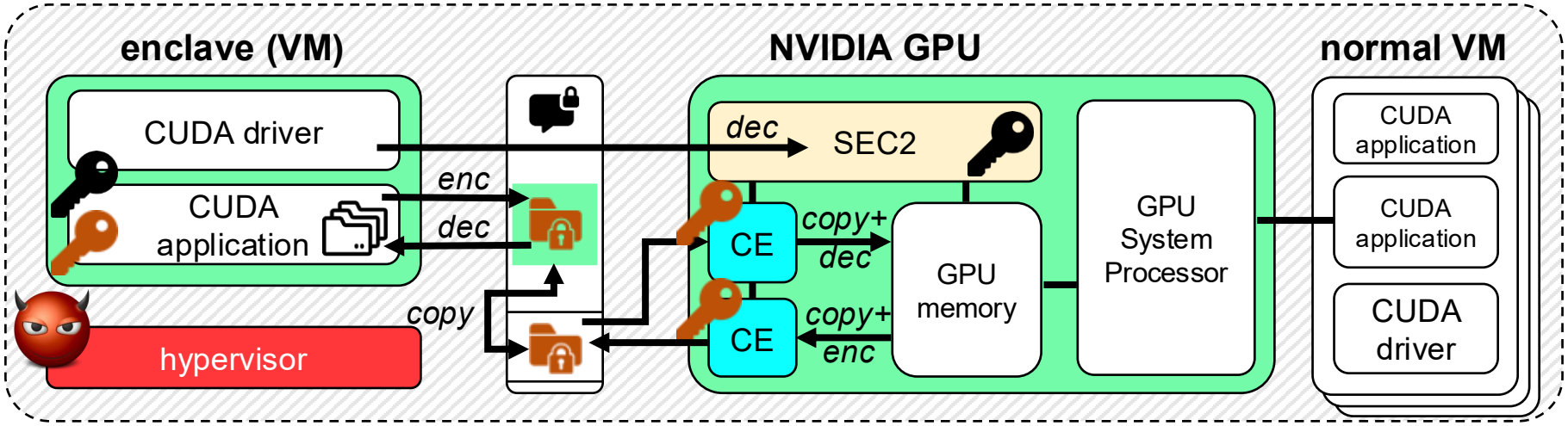
Granularity of NVIDIA CC

- Assigns a whole GPU to a single TEE (CVM)
  - Multiple CVMs nor VMs can share a confidential GPU once assigned to a specific CVM
  - Production model-serving demand is highly variable and bursty



Future Work : CC-enabled GPU Sharing

- Sharing CC-enabled GPU would greatly increase GPU utilization while protecting the sensitive workload(s)
  - NVIDIA GPUs support Multi-Instance GPUs (MIGs) ➔ **disabled** for CC-enabled GPUs
  - Existing GPU partitioning/sharing schemes fails to protect sensitive GPU workloads



remote platform

Summary

1. **Confidential Computing**
2. **Trusted Execution Environments**
3. **Problem: Trust Boundary**
4. **Problem: Resource Sharing**

Promising technology to protect sensitive data in **remote** environments

On-going research concerns how to utilize TEEs for CC both **securely** and **efficiently**

Thank you

Any questions?