

Building Trust Despite Digital Personal Devices

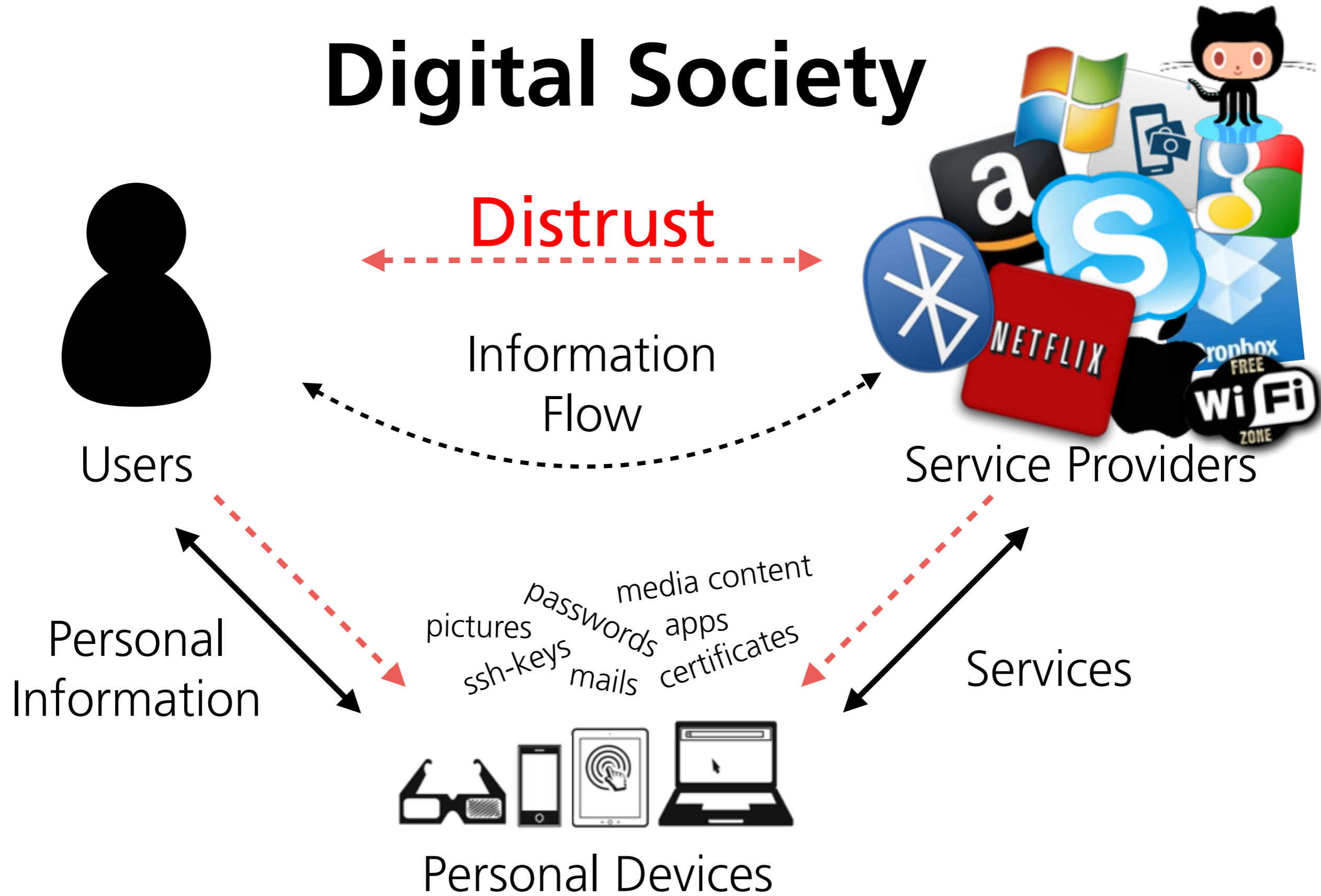
OpenIT - 07.03.2014

by

Javier González

Javier González - jgon@itu.dk
Philippe Bonnet - phbo@itu.dk

Digital Society



Digital Society

Privacy

Control


Privacy?



Personal Devices

define: privacy

pri·va·cy

/ˈprɪvəsi/ 

noun

noun: privacy

1. the state or condition of being free from being observed or disturbed by other people.
"she returned to the privacy of her own home"
synonyms: [seclusion](#), [solitude](#), [isolation](#), freedom from disturbance, freedom from interference [More](#)
- the state of being free from public attention.
"a law to restrict newspapers' freedom to invade people's privacy"

Translate privacy to

Use over time for: privacy



The social science viewpoint

What are the social norms around privacy and personal data processing? How do they evolve in time? How do they evolve with respect to IT evolution?

Getting beyond the lame meme

- "Privacy is dead – deal with it"
- "I've done nothing wrong, why should I care?"
- "In Denmark, people are very trusting, so privacy is not an issue"

Contextual Integrity

Helen Nissenbaum. *Privacy in Context*, 2010

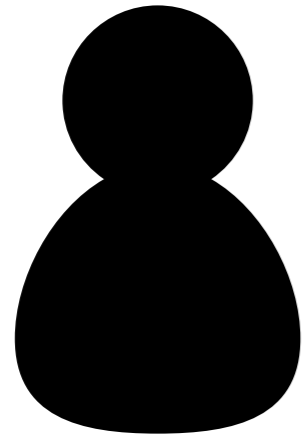
Exchange/sharing of personal data is at the core of any social interaction

- Privacy is not about “not sharing” personal data!

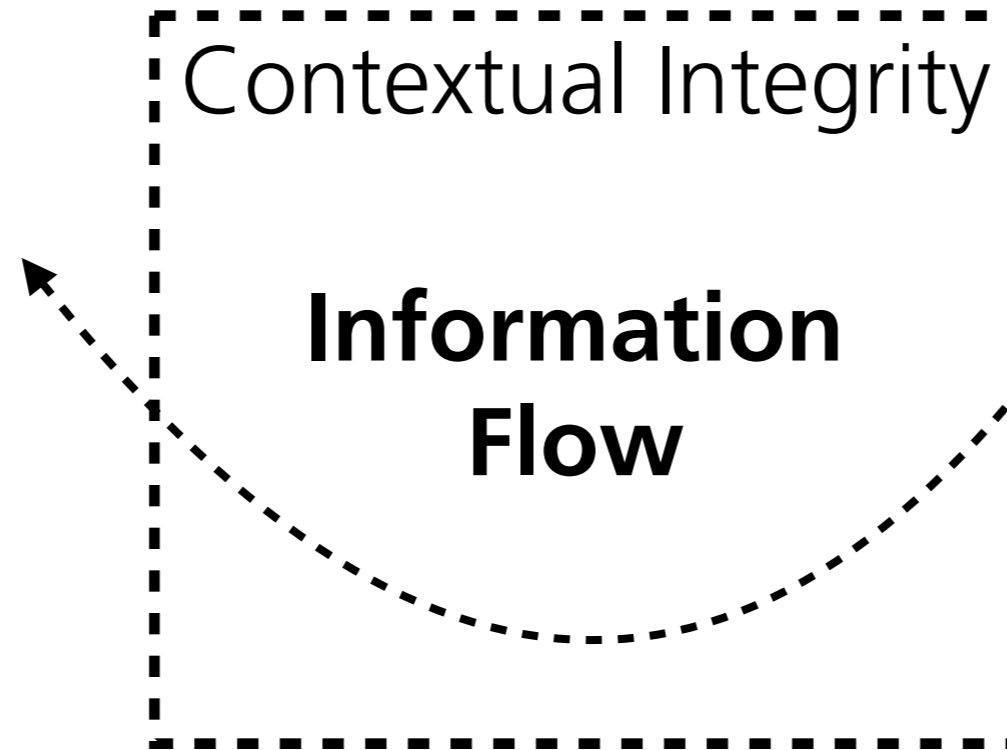
Any social context (work, education, health, ...) defines – more or less explicitly – a social norm, i.e., an appropriate behaviour that is to be expected.

Contextual integrity gives a framework to reason about the norms that apply, in a given social context, to the flows of personal data (i.e., information norms)

Digital Society



Users

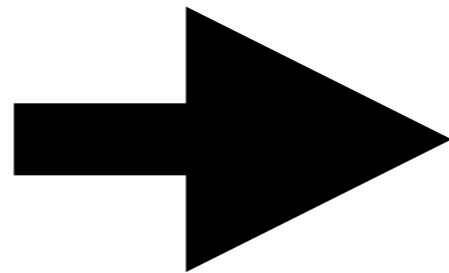


Service Providers



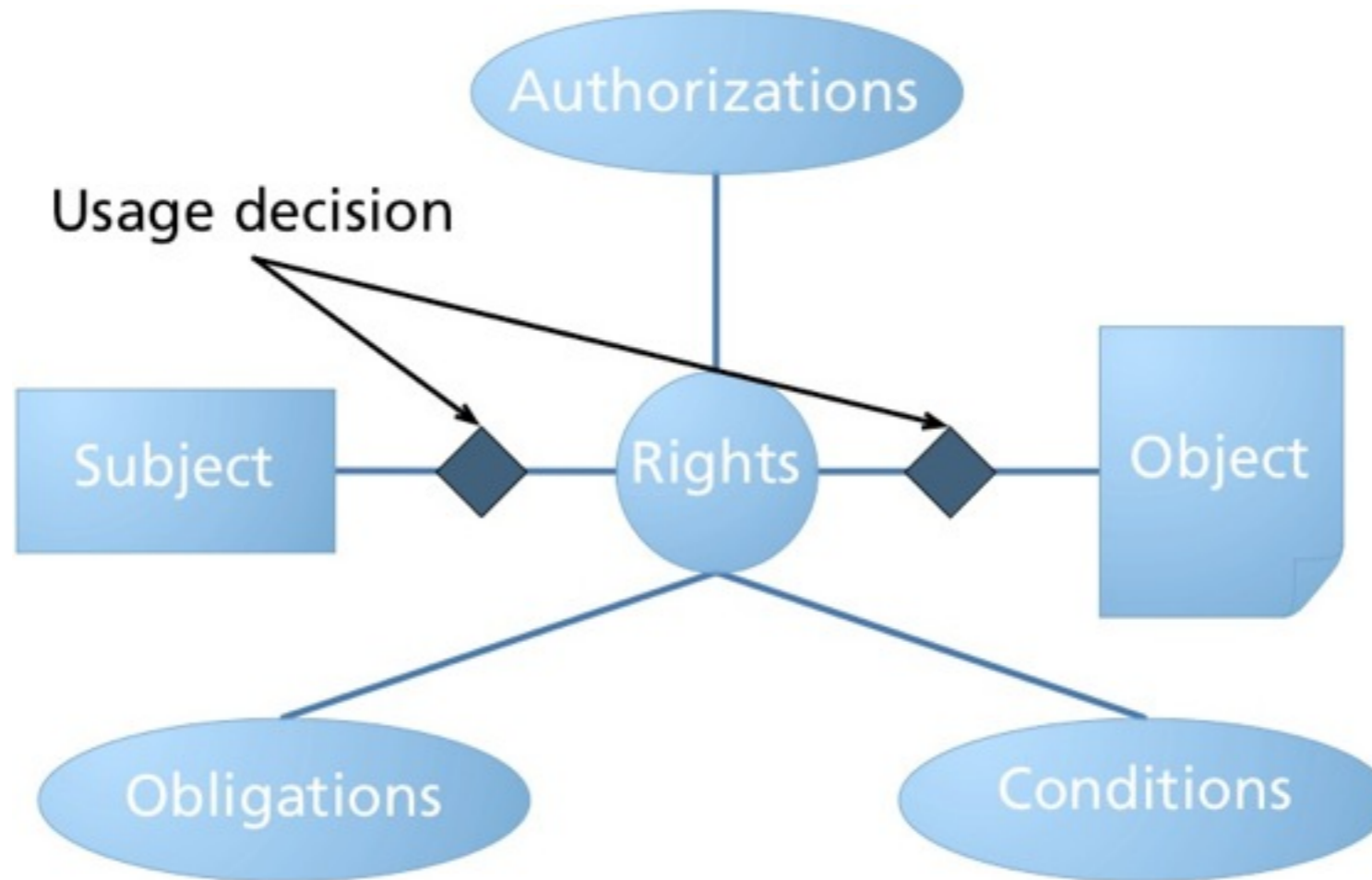
Personal Devices

**Social Science
Framework**

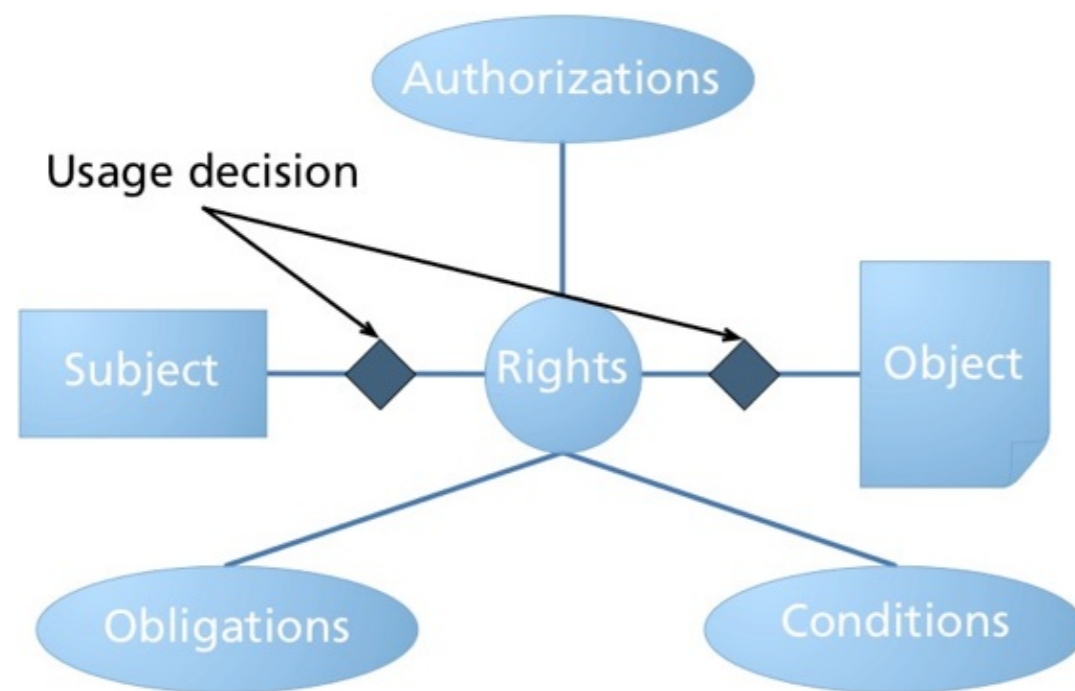


**Computer Science
Implementation**

UCON_{ABC}



UCON_{ABC}



Formal Model

Can model complex frameworks

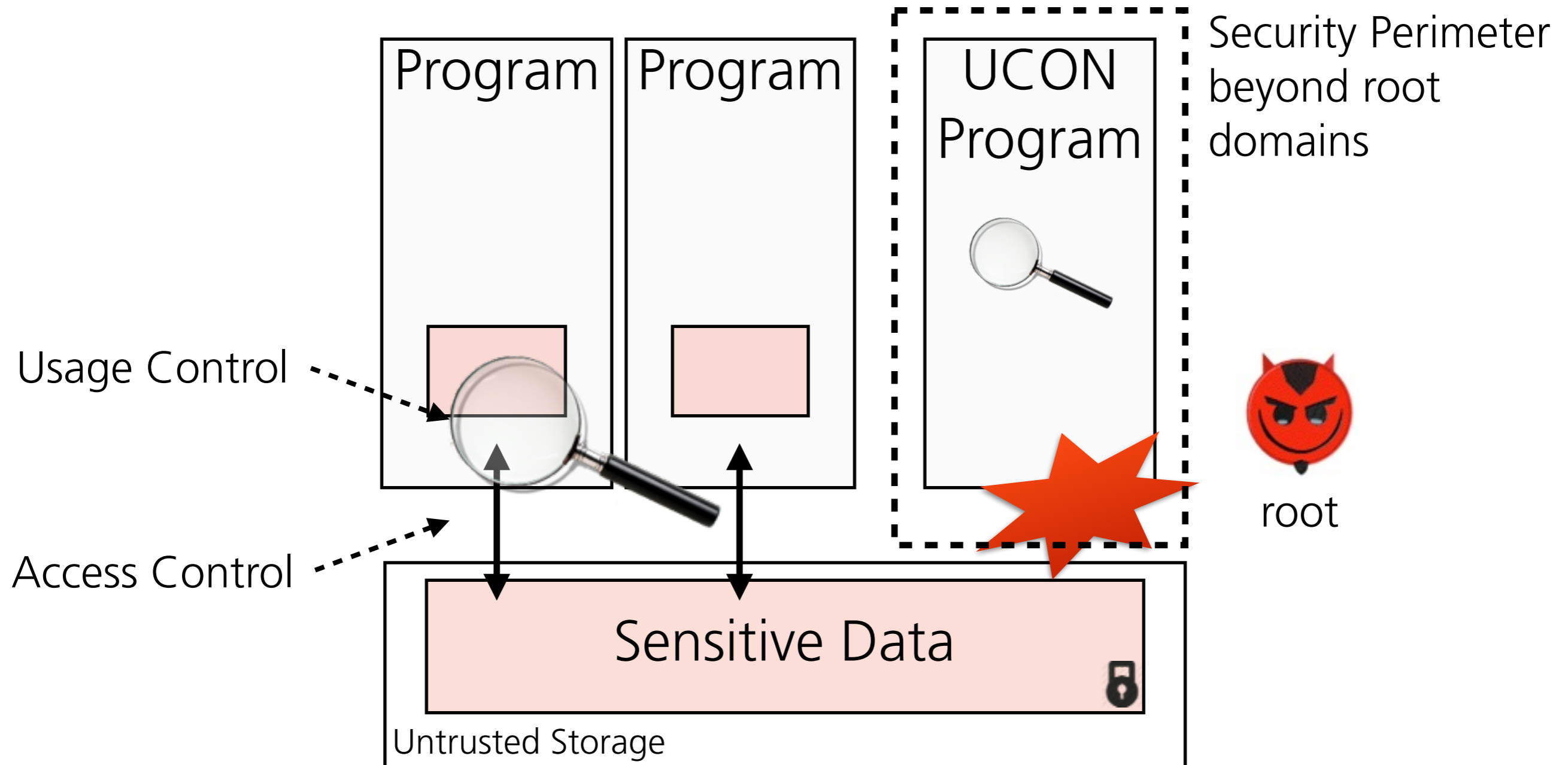
Strong security Assumptions

No implementation!

Enforcement: a priori control of usage rights

Audit: a posteriori control of how rights were used

Implementing UCON_{ABC}



Requirements UCON_{ABC}

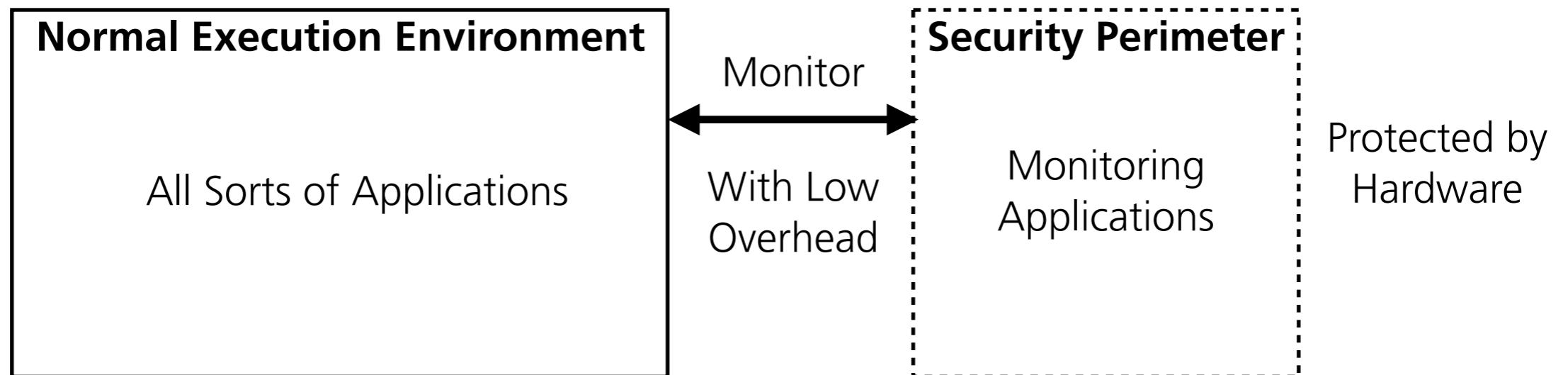
UCON needs to be implemented within a security perimeter protected by hardware so that attackers with root privileges cannot disable it using software.

From inside the security perimeter it should be possible to “monitor” programs outside the security perimeter

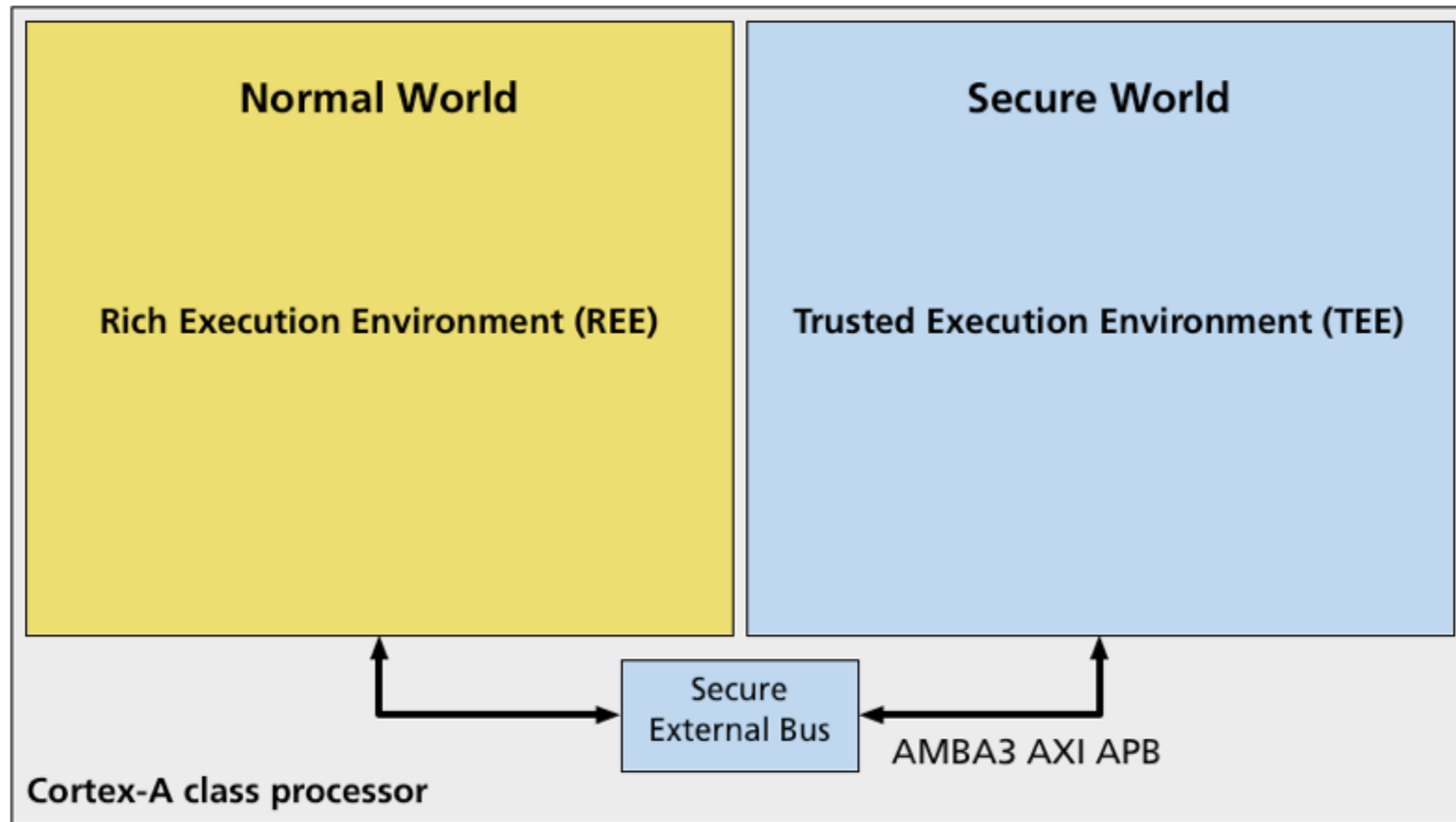
Communicating with programs in the security perimeter should entail a low overhead

Secure Platforms

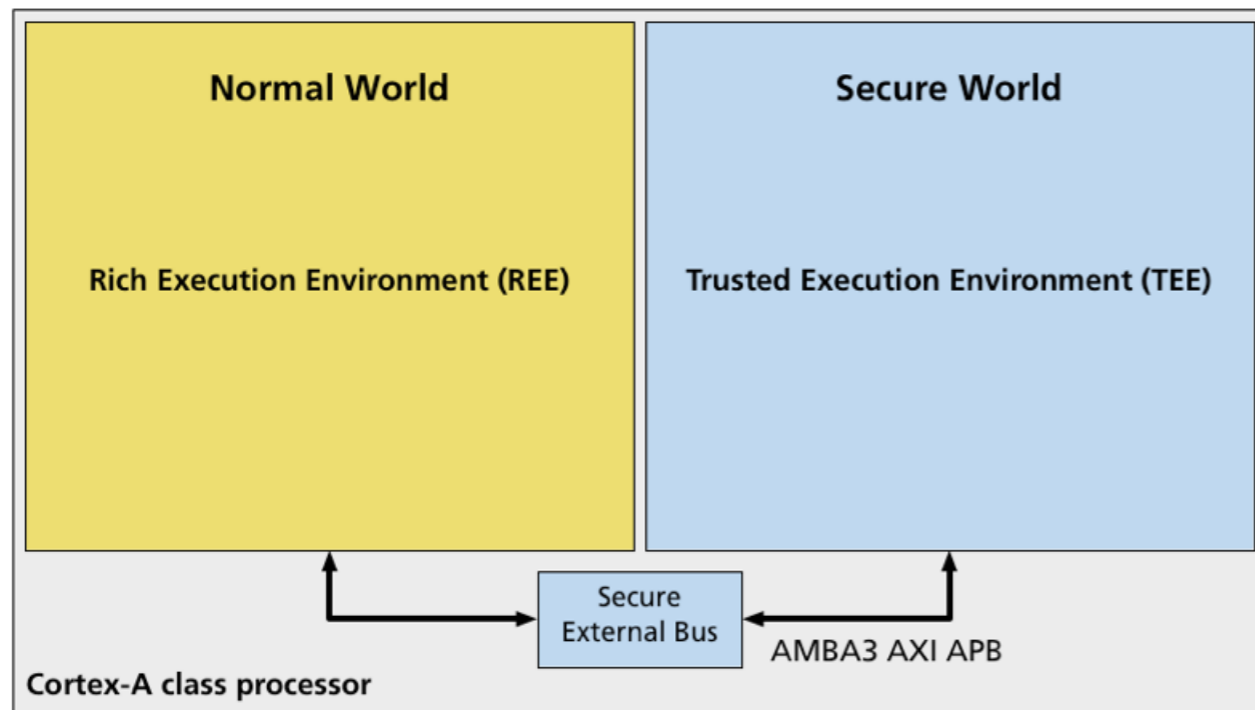
	Level of Protection	Communication	Execution Environment	Examples
Only Software	Low	Messages, Sockets, SM	Rich OS	Android, Linux, IOS, Windows
Software and Hardware	Medium/High	Shared Memory	Rich OS + TEE	ARM TrustZone
SW and Tamper resistant HW	Very High	Narrow Interfaces	Secure Ad-hoc OS	IBM CryptoCards, TPM, Secure Token



ARM TrustZone



ARM TrustZone



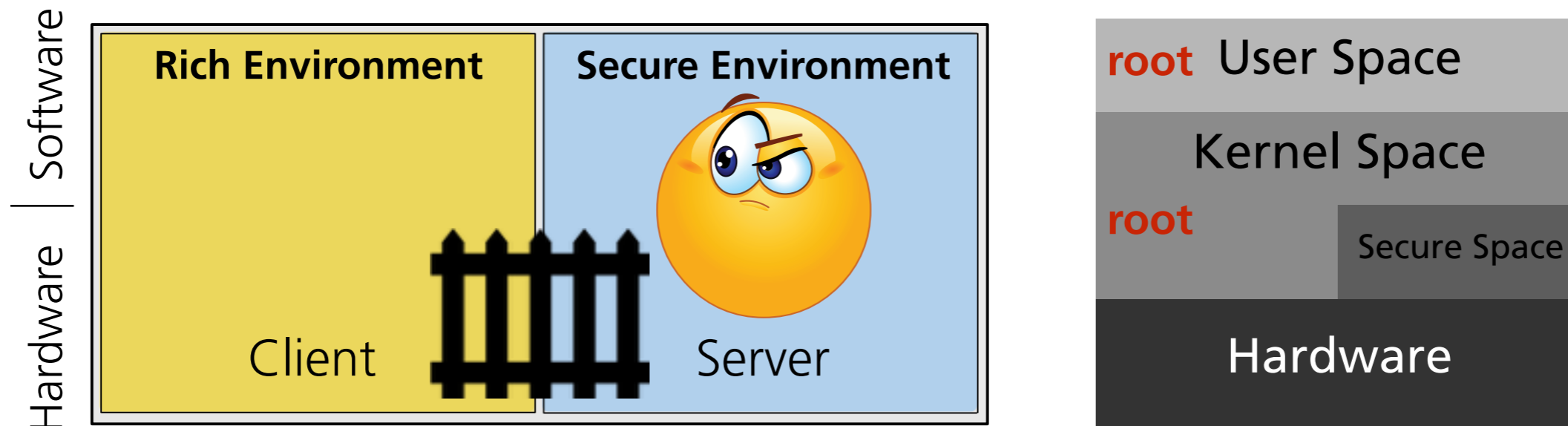
Secure memory, rich memory and shared memory

Secure acquisition/release of peripherals in runtime (e.g., ethernet, screen, flash)

Gatekeeper - Context switch controlled by hardware (AMBA3)

Shared processor

Rich/Secure Abstraction



Principle 1: Self preservation first. Under the suspicion of a threat, the secure environment isolates itself logically and gives up availability in order to protect data integrity, confidentiality and durability.

Principle 2: Lead all communications. The secure environment defines all parameters that define this communication: protocol, certificates, encryption keys, etc.

Principle 3: Secure all interactions. The secure area has priority to obtain exclusive access to secure peripherals.

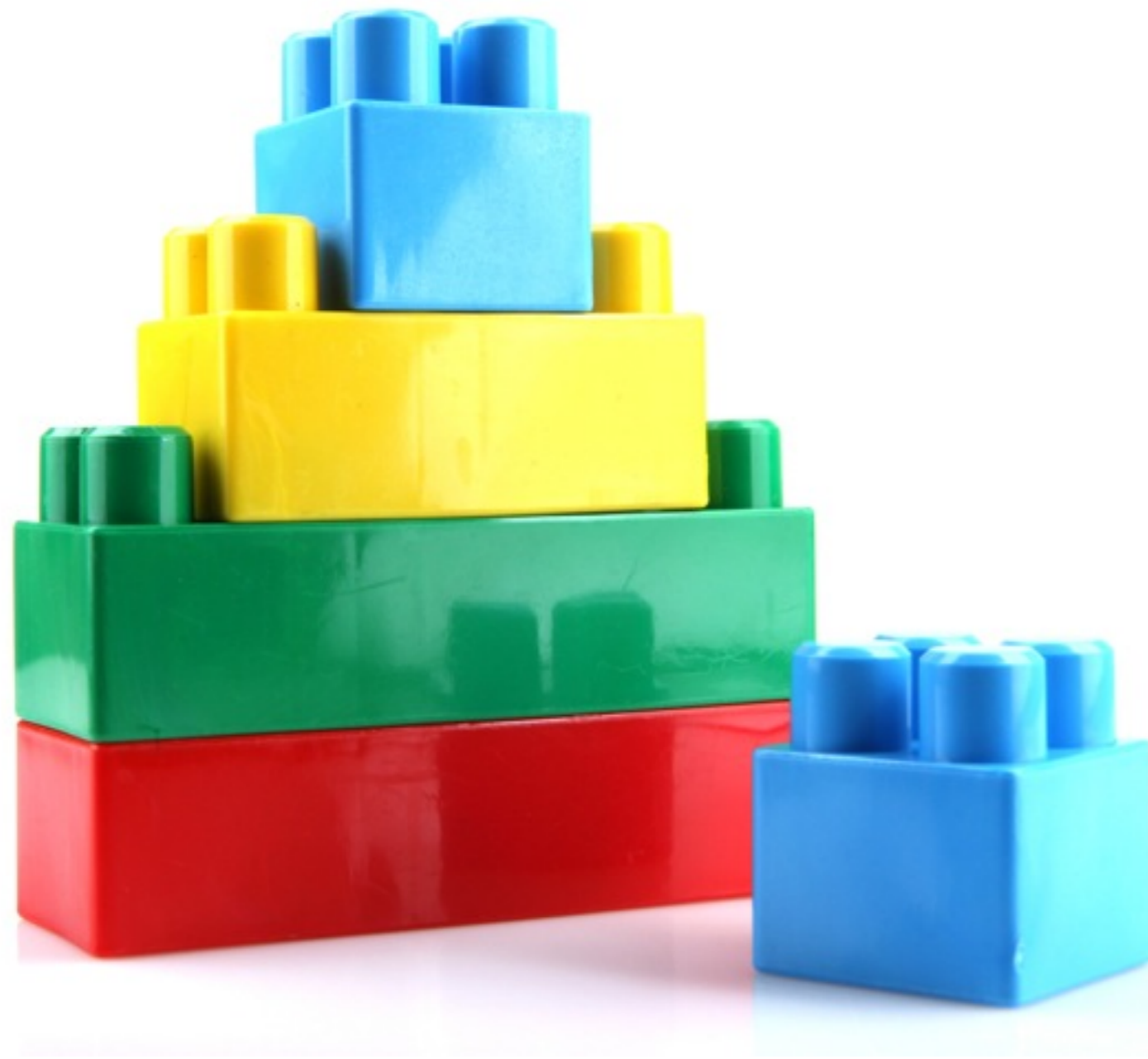
Building Trust

Users have the certainty that their sensitive information will not be misused by the software running in their devices.

Service providers have the certainty that the devices interacting with their systems are not compromised

Both should have the freedom to choose who they trust, and the technology should aid giving certainty, not enforcing

Building Trust



Support different digital contexts

Enforce usage policies

Monitor the integrity of the system

Protect data in secondary storage

Trusted Storage Module (TSM)

Background Information

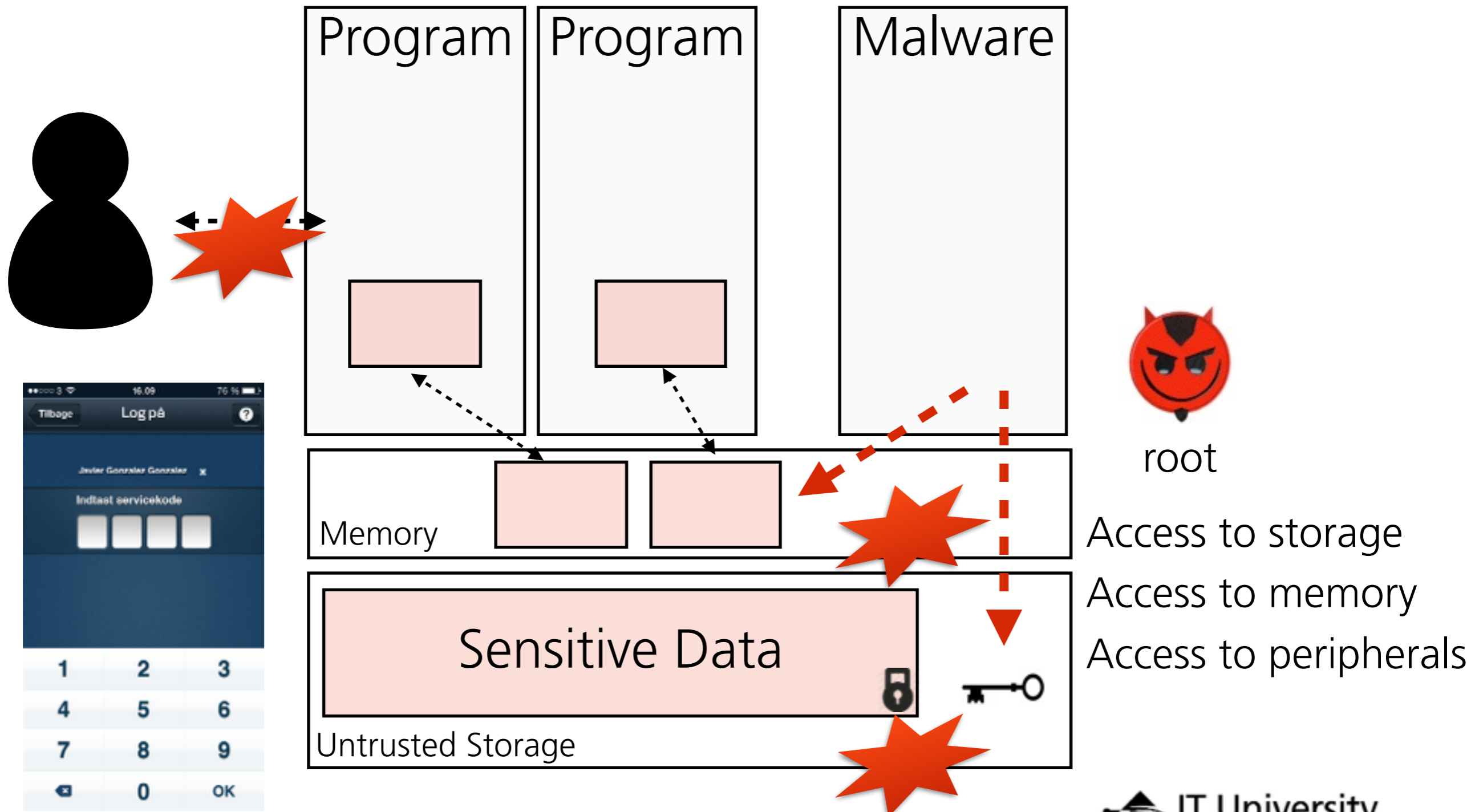
A system providing trusted storage should guarantee data confidentiality, integrity, availability, and durability

Today's security policies rely on data encryption to support confidentiality and integrity. However, if encryption keys can be compromised or stolen on the client computer, then there is not much protection left.

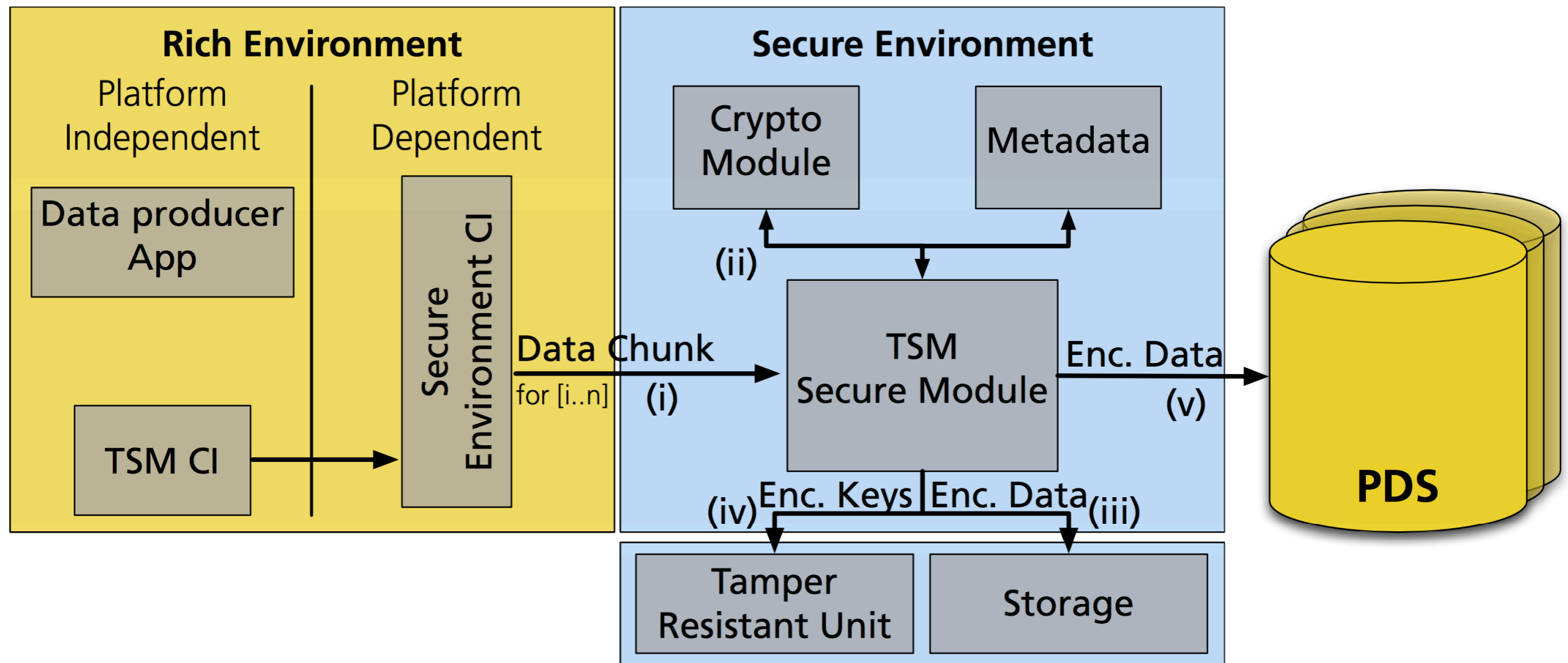
Approaches using tamper-resistant hardware: low functionality, physical separated, and narrow interfaces

Trusted Storage Module (TSM)

Thread Model

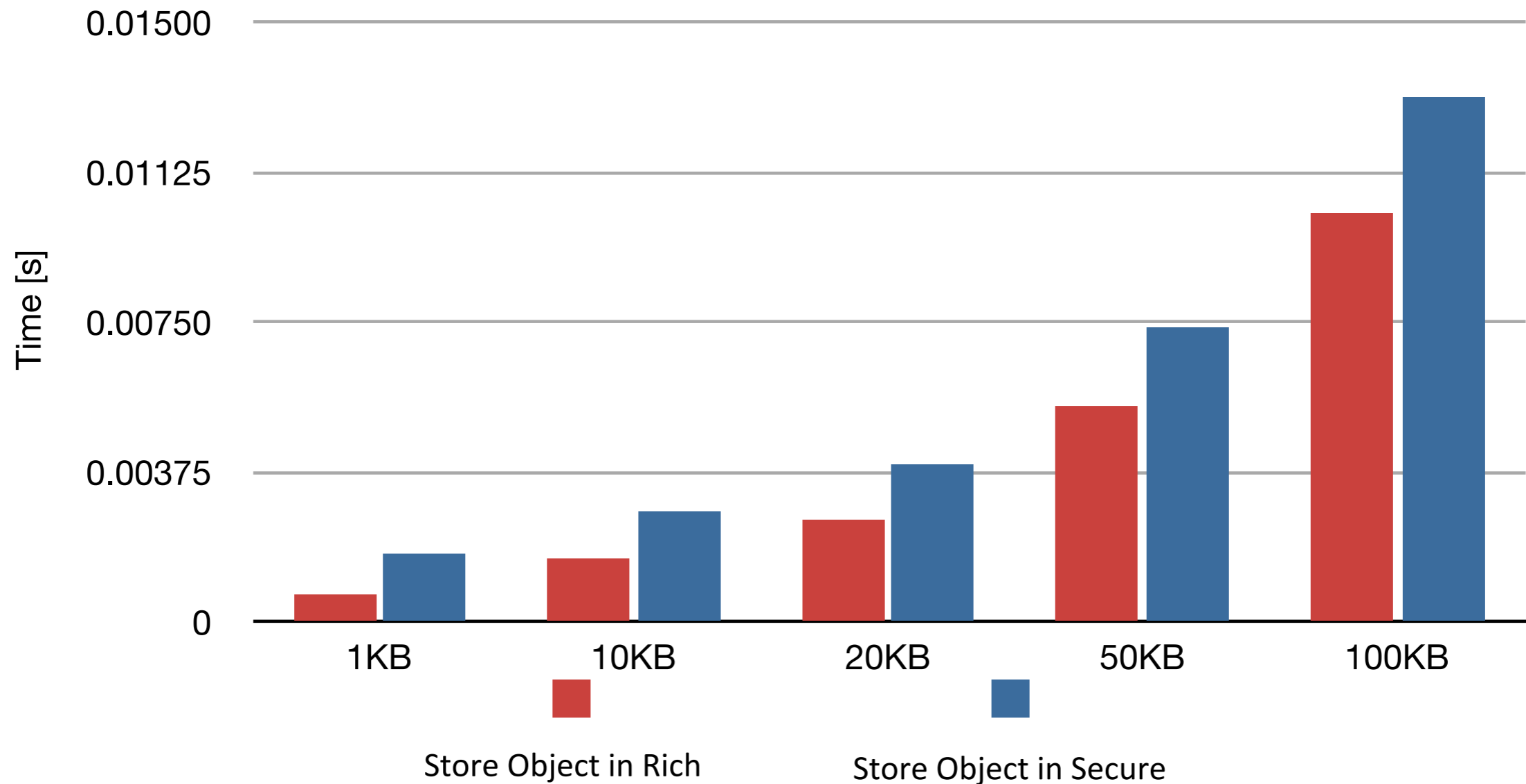


Trusted Storage Module (TSM) Architecture



Trusted Storage Module (TSM)

Overhead



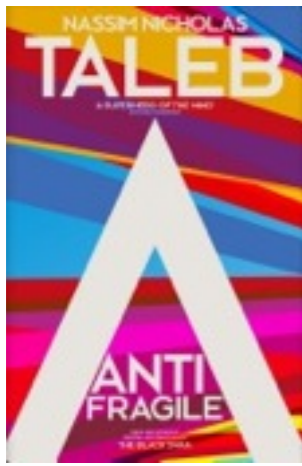
Trusted Storage Module (TSM)

Contributions

Provide a mechanism for rich apps to store securely objects containing sensitive information introducing low overhead

Provide a mechanism to enforce access control to the files storing those objects, i.e., encryption, secure memory and secure peripherals

Use untrusted storage as a cheap and “unlimited” source of secondary storage



Antifragile Storage

Can we learn from successful attacks and improve, being better prepared for future attacks?

Successful attacks (hardware and software) that do not entail the collapse of the system are good and welcome (if detectable), and even intentionally provoked

The level of hardware tamper resistance is the upper boundary to which the system can benefit from being harmed

How do we detect these attacks (e.g., traps, human intervention)? How to learn from them (e.g., AI, machine learning)?

* Nassim Nicholas Taleb

Trusted Integrity Module (TIM)

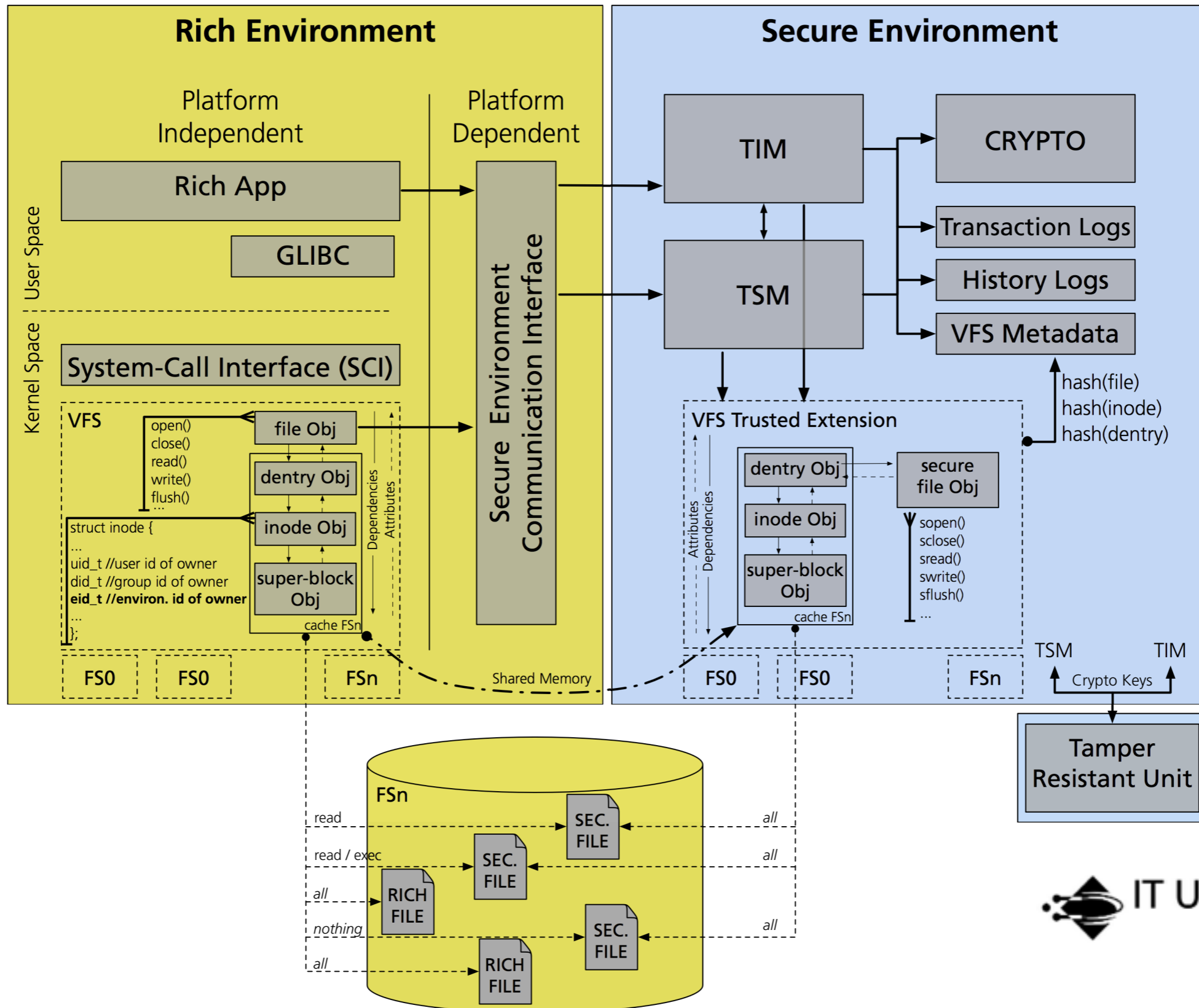
Background Information

Secondary storage is normally attacked. To survive a reboot or hide from system administrators, attackers make modifications to system files, line commands and system libraries.

Running processed produce logs, which are accessible from users space (applications), and therefore can be tampered with.

Storage-based integrity checks need to build on top of trusted storage (e.g., TSM)

Trusted Integrity Module (TIM) Architecture



Trusted Integrity Module (TIM)

Contribution

Provide an architecture to guarantee the integrity of system files adding a low overhead to file system primitives

Provide an method to log actions involving system files in trusted storage, preventing attackers to clean after themselves

TIM is in itself a storage-based Intrusion Detection System (IDS) without much less assumptions than current approaches

Next Steps

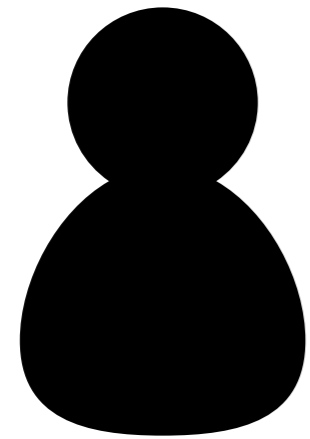
Implement UCON on top of TIM to add complex usage control policies to secure file operations (i.e., enforcement, embedded behaviour, and audit)

Extend TIM with a machine learning algorithm to learn from past attacks - antifragility

Exploring how to monitor running processes without introducing a big overhead. Ideas: Use of resources, peripherals, etc.

Example:
**Supporting different
digital “social” contexts**

Security today



Users

Imposition



Service Providers*

Distrust Lock up

bypass

?

DRM-like

Cyberactivism:

- Software Freedom
- Privacy
- Anti Copyright



Secure Personal Devices

* some - normally those involved in media content.

Lockdown, Freedom, and Certainty

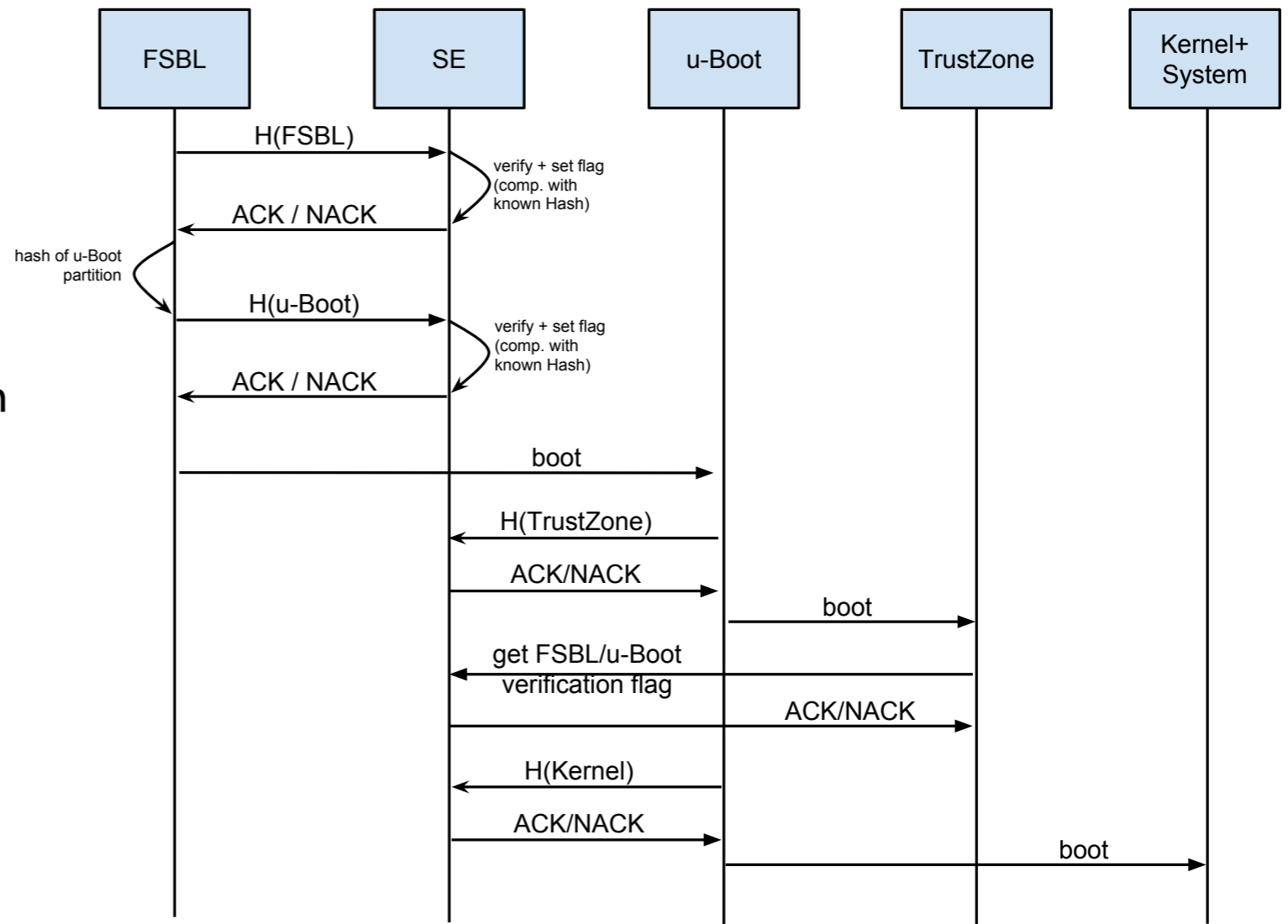
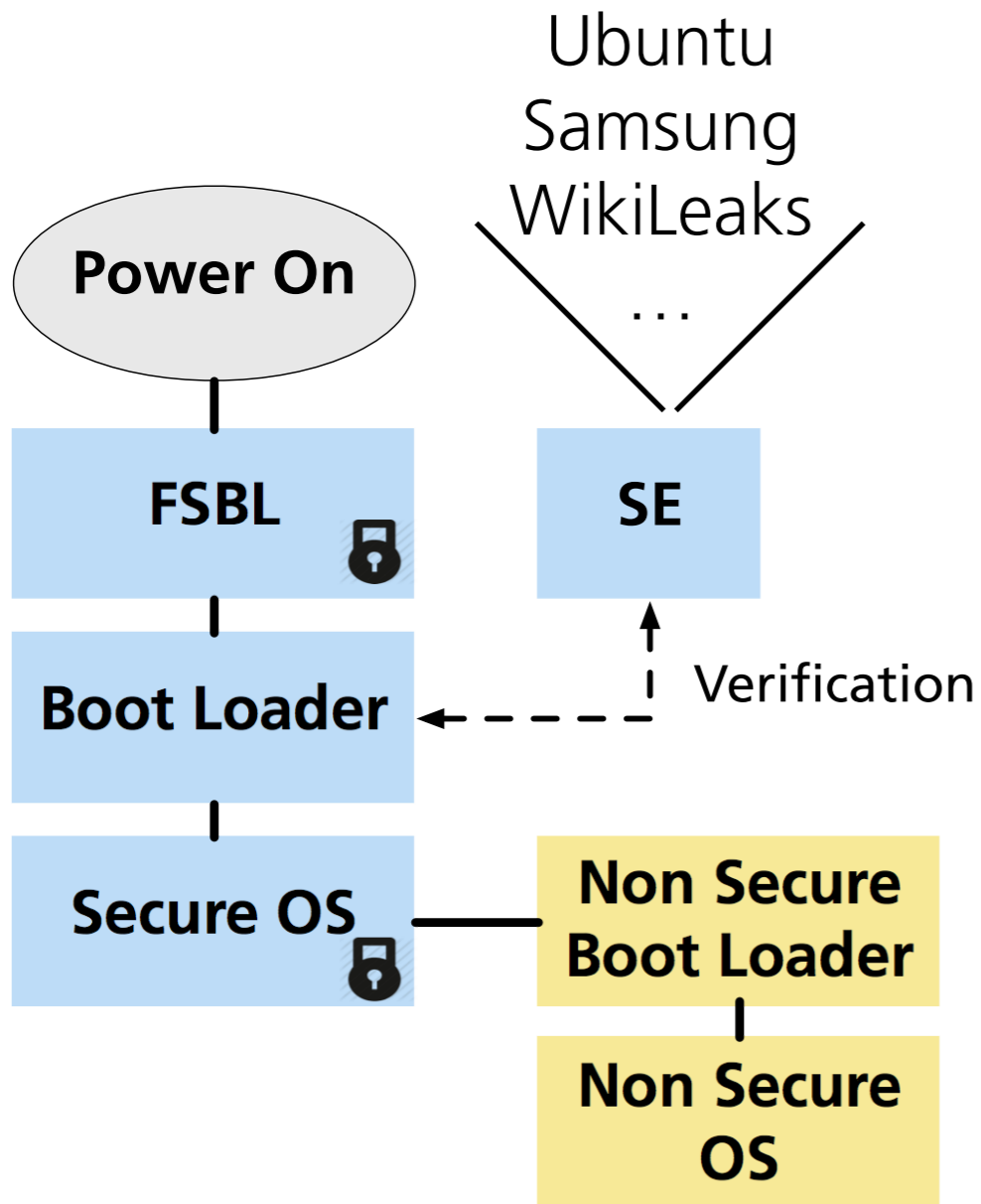
Cory Doctorow. The Coming Civil War
over General Purpose Computing

Lockdown: *"Your TPM comes with a set of signing keys it trusts, and unless your bootloader is signed by a TPM-trusted party, you can't run it. Moreover, since the bootloader determines which OS launches, you don't get to control the software in your machine."*

Freedom: *"Android lets you tick a box to run any code you want."*

Certainty: *"You tell your TPM which signing keys you trust - say, Ubuntu, EFF, ACLU and Wikileaks - and it tells you whether the bootloaders it can find on your disk have been signed by any of those parties. It can faithfully report the signature on any other bootloaders it finds, and it lets you make up your own damn mind about whether you want to trust any or all of the above."*

Certainty Boot Architecture



Certainty Boot

Contributions

Provide a *Trusted Boot* using a Secure Element (SE) and TrustZone. The boot sequence is stored in the SE, which can only be accessed from the secure environment. (2-phase verification)

New signing keys can be added to the SE by the user

Applications can check the boot sequence trace through the secure environment to verify that they trust the running software

Contributions

We have a framework that can implement a usage control model defining privacy policies (contextual integrity) in order to build trust in digital interactions

Bottom - up: We have a prototype where storage-based usage control policies based on trusted storage and its integrity can easily be implemented: TSM and TIM

Top - down: We have one way to give users the freedom to choose their software while giving certainty to both users and service providers

Building Trust Despite Digital Personal Devices

OpenIT - 07.03.2014

by
Javier González

Javier González - jgon@itu.dk
Philippe Bonnet - phbo@itu.dk