TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

“TRUSTED” COMPUTING

DISTRIBUTED OPERATING SYSTEMS

HERMANN HÄRTIG, SUMMER 2018

Lecture Goals

Understand principles of:

- Authenticated booting, relation to (closed) secure booting
- Remote attestation
- Sealed memory
- Dynamic root of trust, late launch
- Protection of applications from the OS
- Point to implementation variants (TPM, iSGX, ARM-TZ)

Lecture NON-Goals

Non-Goal:

- Lots of TPM, TCG, Trustzone, SGX details
  → read the documents once needed

Some Terms

- Secure Booting
- Authenticated Booting
- (Remote) Attestation
- Sealed Memory
- Late Launch / dynamic root of trust
- Trusted Computing (Group) / Trusted Computing Base

- Beware of terminology chaos !

Trusted Computing (Base)

Trusted Computing Base (TCB)

- The set off all components,
  hardware, software, procedures,
  that must be relied upon to enforce a security policy.

Trusted Computing (TC)

- A particular technology comprised of authenticated booting, remote attestation and sealed memory.

TC Key Goals

- Can running certain Software be prevented?
- Which computer system do I communicate with ?
- Which stack of Software is running?
  - In front of me?
  - On my server somewhere?
- Restrict access to certain secrets (keys) to certain software?
- Protect an application against the OS

Usage Examples (1)

Digital Rights Management:

- Provider sells content
- Provider creates key, encrypts content
- Client downloads encrypted content, stores on disk
- Provider sends key, but needs to ensure that only specific SW can use it
- Has to work also when client is off line
- PROVIDER DOES NOT TRUST CUSTOMER

Usage Examples (2)

Virtual machine provided by cloud

- Client buys Cycles + Storage (Virtual machine)
- Client provides its own operating system
- Needs to ensure that provided OS runs
- Needs to ensure that provider cannot access data
- CUSTOMER DOES NOT TRUST PROVIDER

Usage Examples (3)

Industrial Plant Control (Uranium enrichment)

- Remote Operator sends commands, keys
- Local operator occasionally has to run test SW, update to new version, ...
- Local technicians are not Trusted

Usage Examples (4)

Anonymity Service

- Intended to provide anonymous communication over internet
- Legal system can request introduction of trap door (program change)
- Anonymity-service provider not trusted

Trusted Computing Terminology

Measuring

- "process of obtaining metrics of platform characteristics"
- example for metric: Hash- Codes of SW

Attestation

- "vouching for accuracy of information"

Sealed Memory

- binding information to a configuration

Notation

- H(M)
  Collision-Resistant Hash Function H
  applied to content M

- S^{pair}: S^{priv} S^{pub}
  Asymmetric key pair of entity S
  used to conceal or sign some content
  S^{pub} is published, S^{priv} must be kept secret

- S^{symm}
  symmetric key, must be kept secret ("secret key")

Notation

- "Digital Signature": { M } S^{priv}

  S^{pub} can be used to verify that S has signed M

  is short for: (M, encrypt(H(M), S^{priv}))

  S^{pub} is needed and sufficient to check signature

- "Concealed Message": { M } S^{pub}

  Message concealed for S

  S^{priv} is needed to unconceal M

Identification of Software

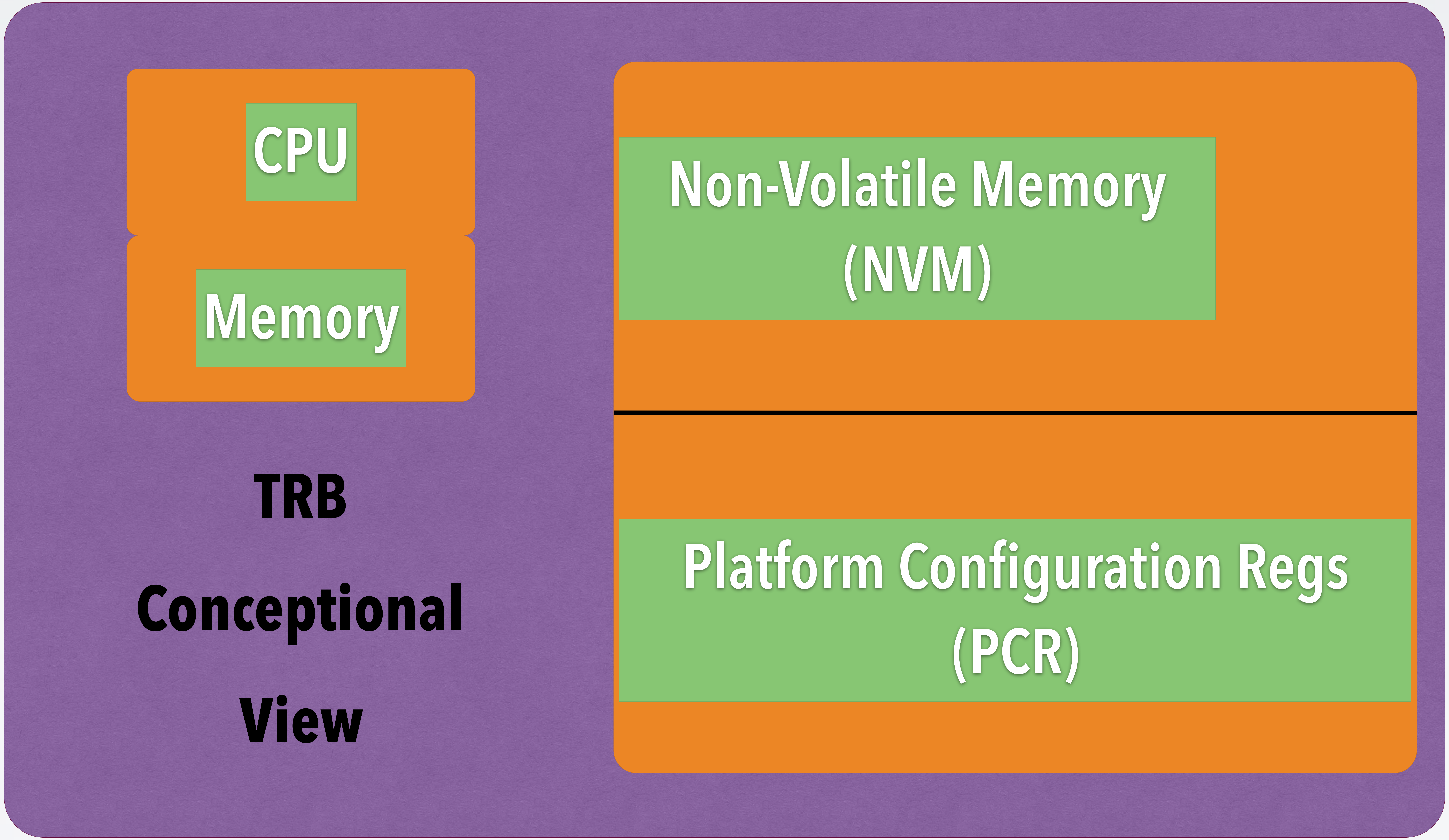
Program vendor: Foosoft FS

Two ways to identify Software: Hash / Public Key

- H(Program)
- {Program, ID- Program}FS^{priv}
  use FS^{pub} to check
  the signature must be made available,
  e.g. shipped with the Program

The „ID" of SW must be known.
FS^{pub} can serve as ID as well.

Tamperresistant Black Box(TRB)



Ways to "burn in" the OS or "Secure Booting"

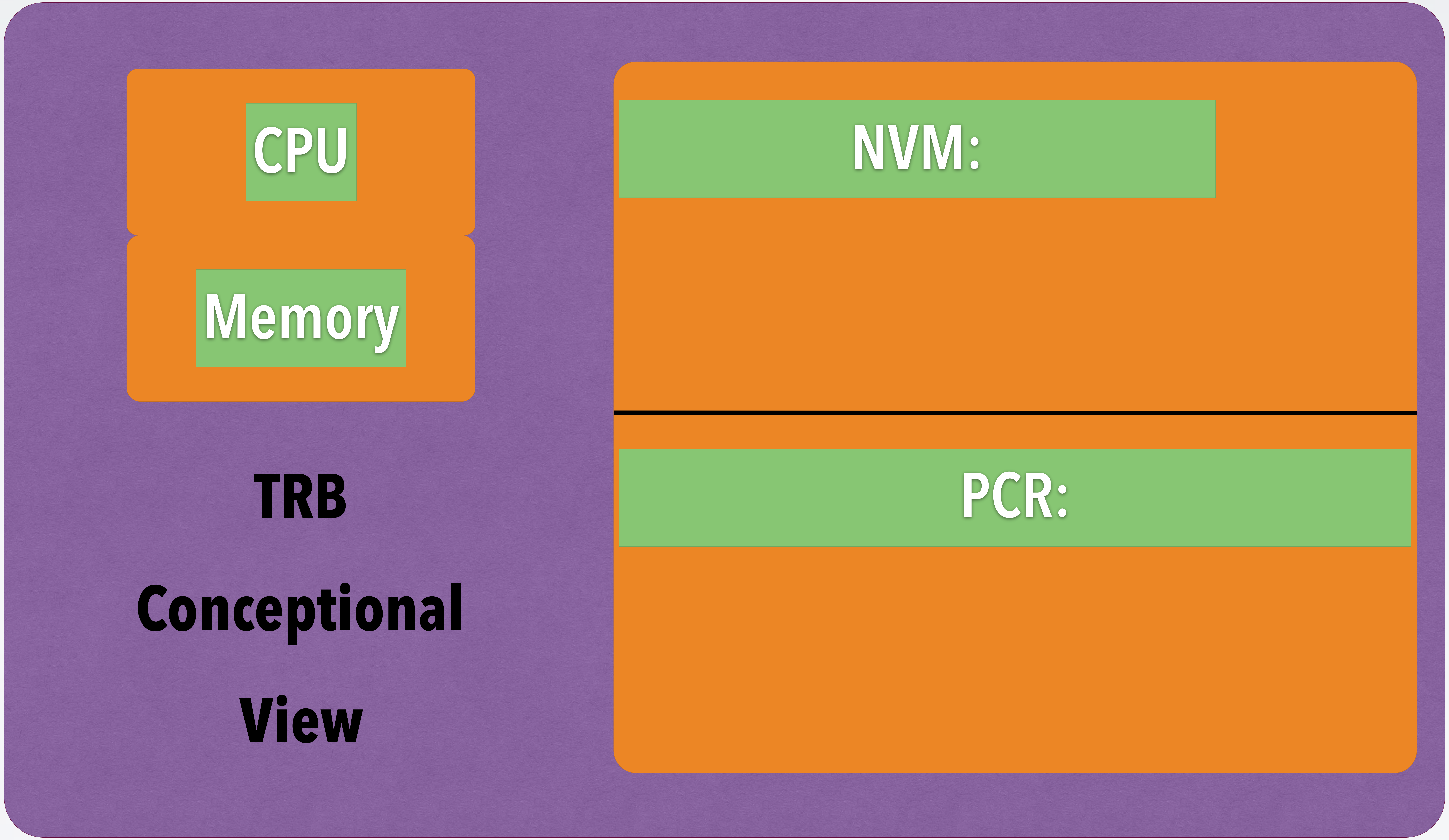
- Read-Only Memory (Flash)
- H(OS) in NVM preset by manufacturer
  - load OS- Code
  - compare H(loaded OS code) to preset H(OS)
  - abort if different
- FS^{pub} in NVM preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using FS^{pub}
  - abort if check fails

Authenticated Booting, using HASH

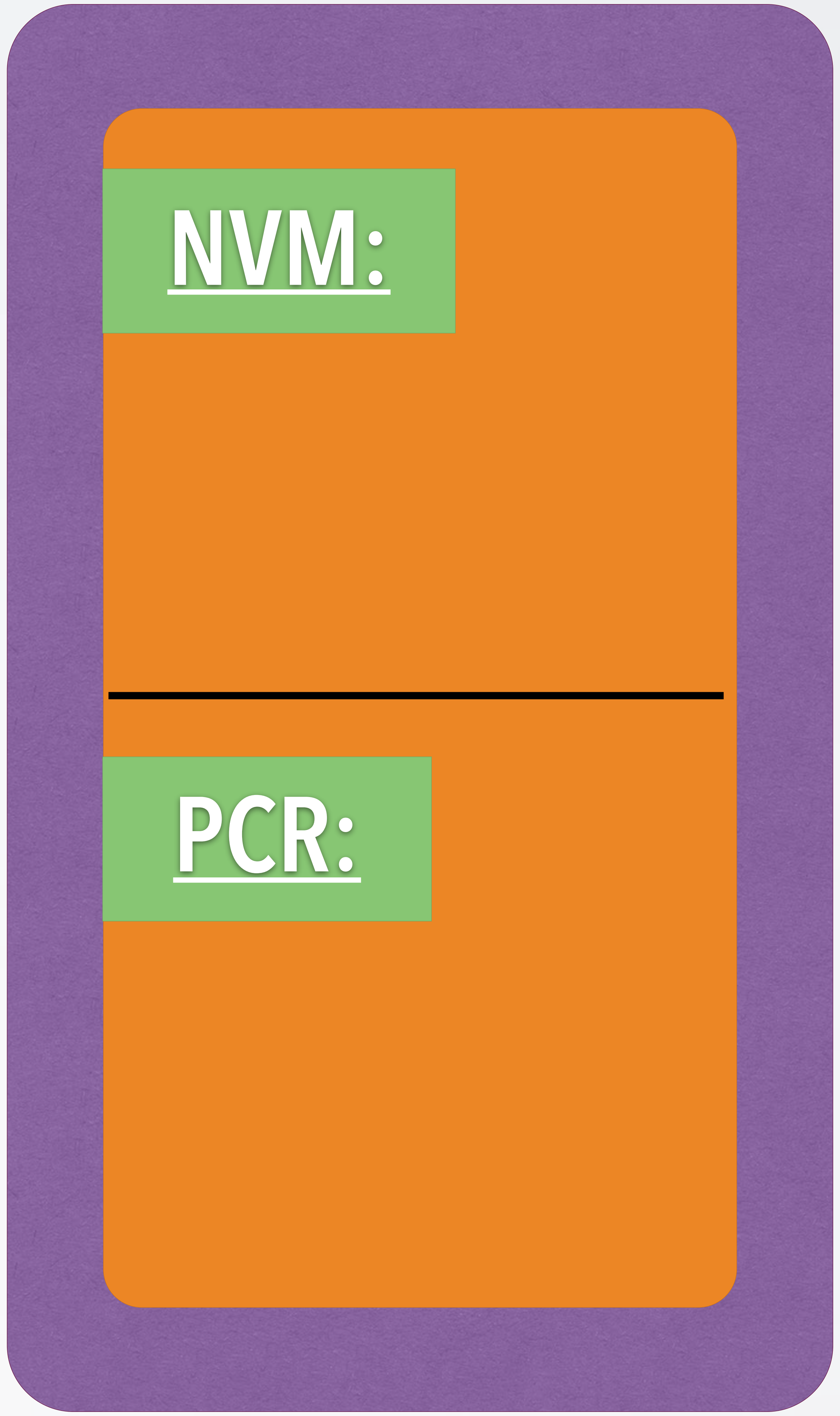
Steps:

A. Preparation by TRB and OS Vendors
B. Booting & "Measuring"
C. Remote attestation

Tamperresistant Black Box(TRB)



Tamperresistant Black Box(TRB)



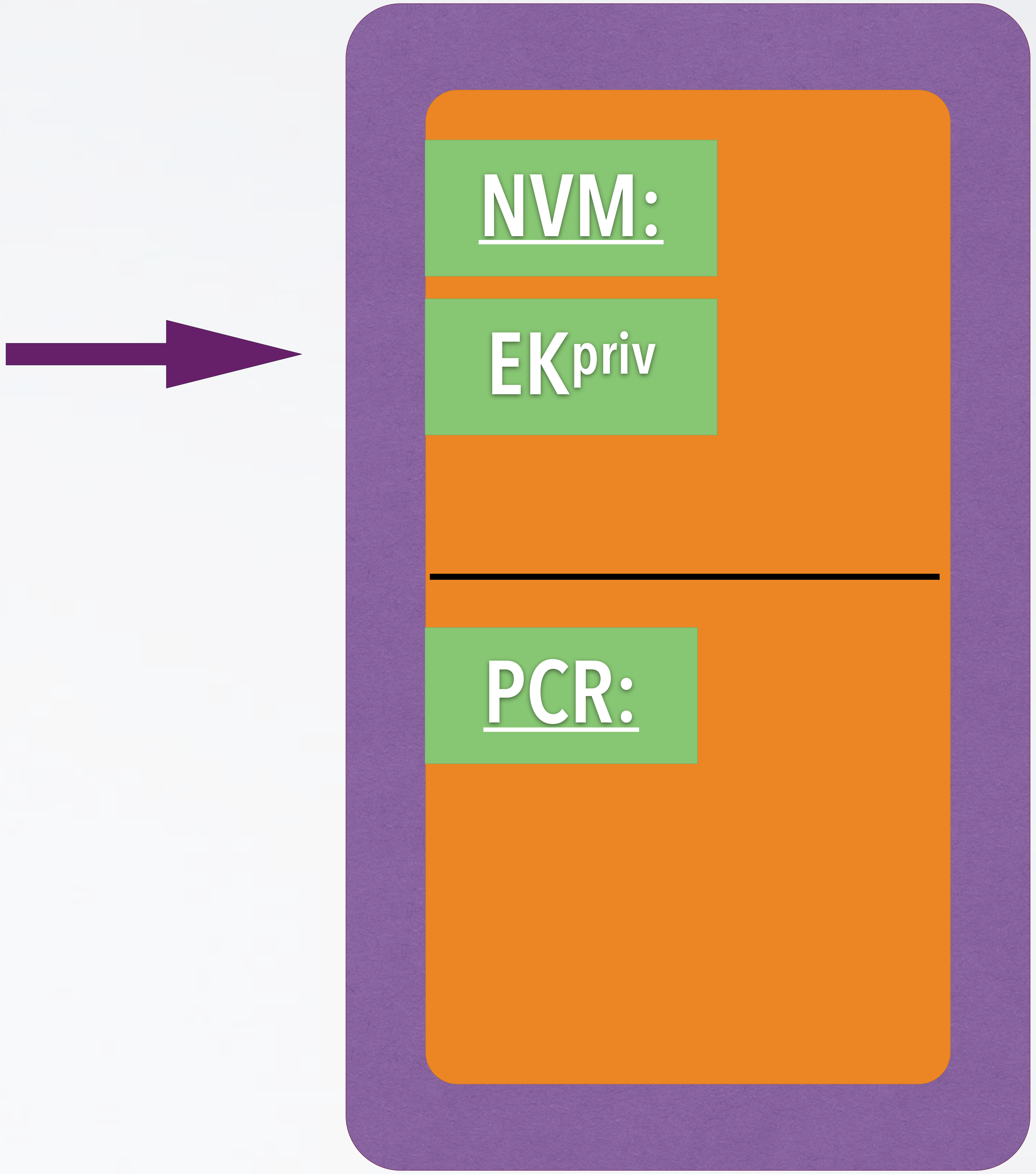
TRB Vendor

TRB generates key pair:

„Endorsement Key“ EK^{pair}

stores EK^{priv} in TRB NVM

publishes EK^{pub}



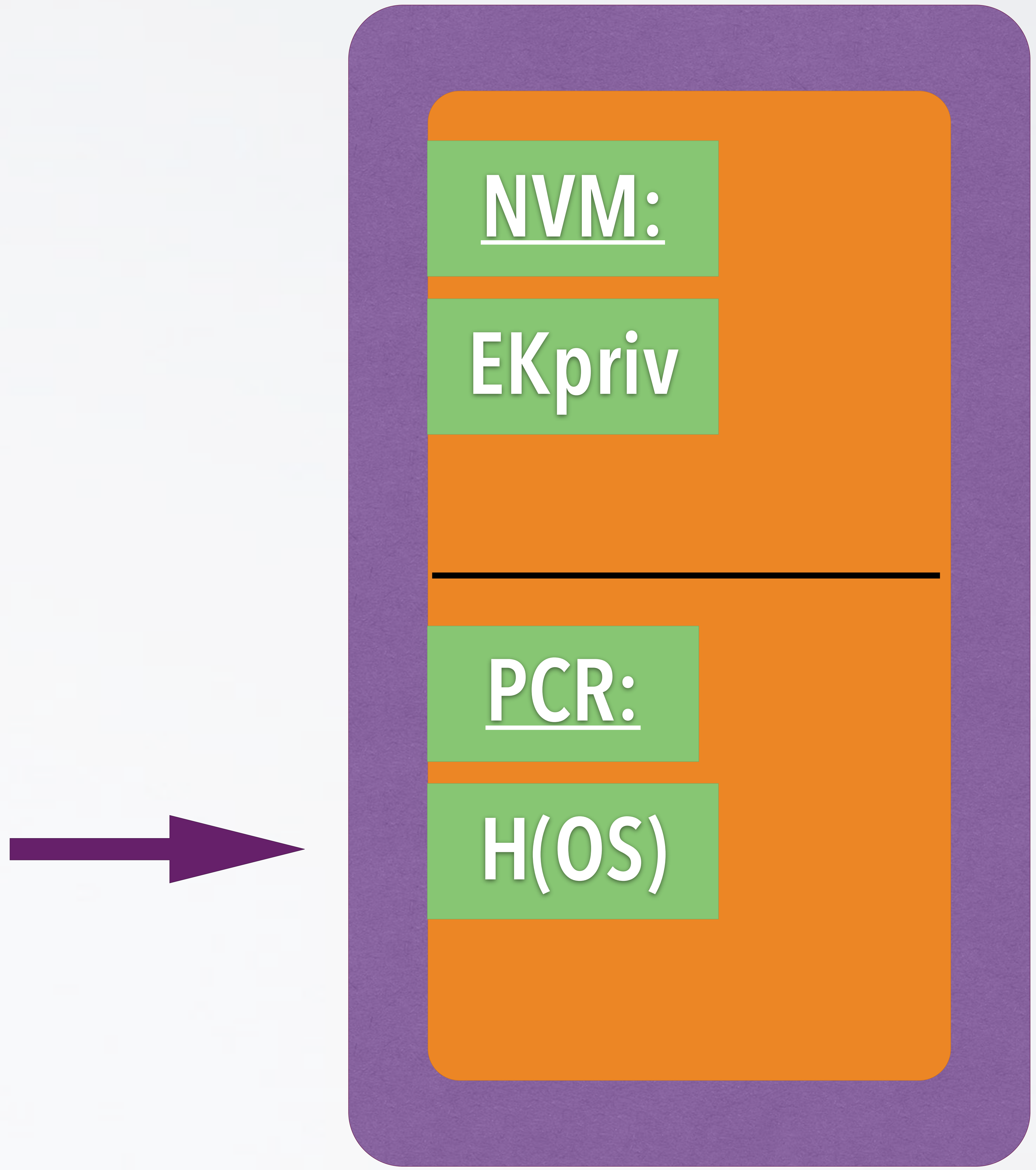
TRB and OS vendor

- TRB vendor certifies:
  {"a valid EK", EK^{pub}}TRB_Vendor^{priv}
- OS-Vendor certifies:
  {„a valid OS", H(OS)}OS_Vendor^{priv}
- serve as identifiers:
  EK^{pub} and H(OS)

Booting

TRB:

- resets TRB !
- measures OS code H(OS)
- stores H(OS) in PCR



PCR not (directly) writable by OS

more later

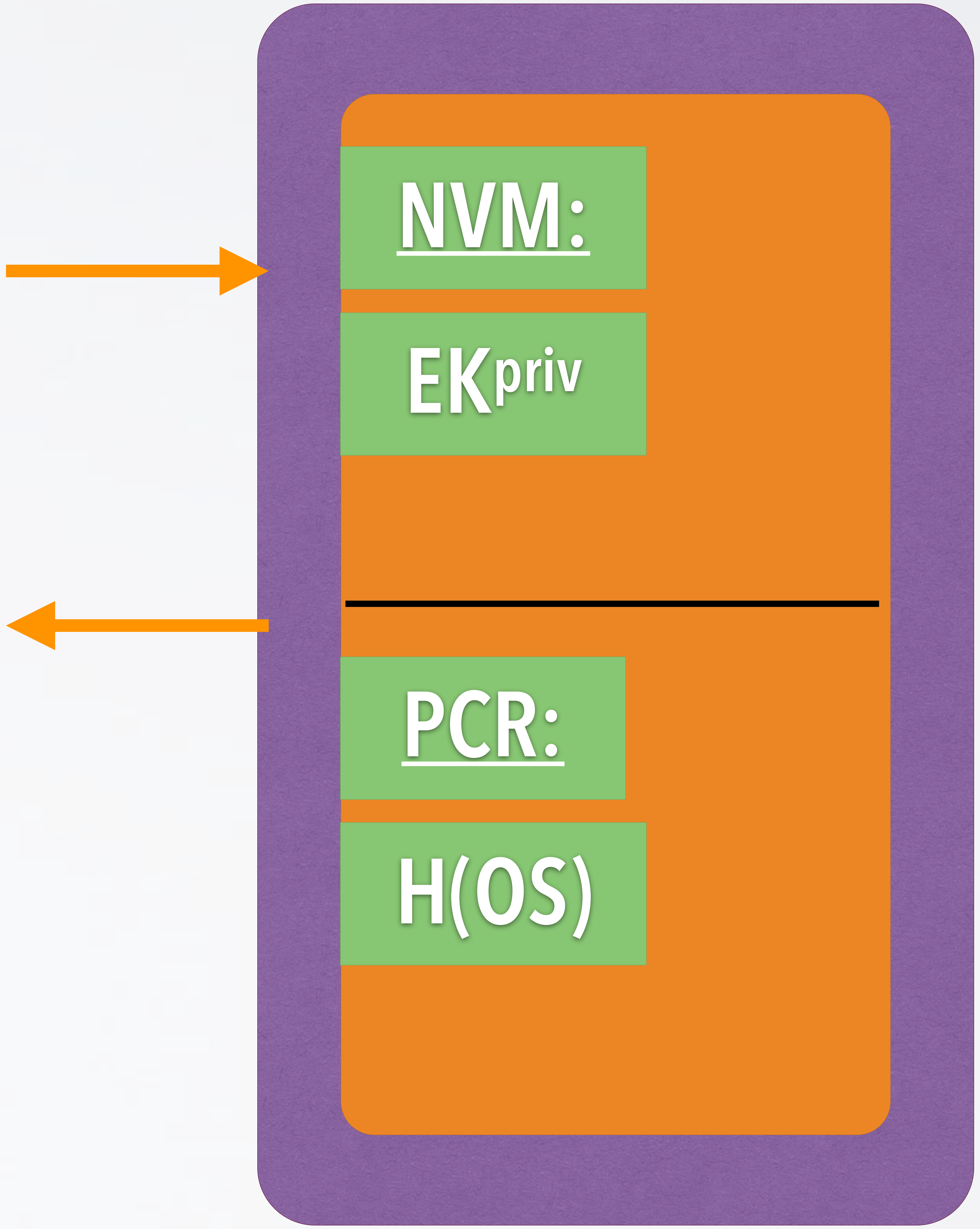
Attestation (first basic explanation)

Challenge:

send NONCE

Response:

{NONCE′, PCR}EK^{priv}



Problem

- boot Linux
  - → challenge
  - ← response "Linux"

- reboot Windows
  - → send data

<u>add one step of indirection:</u>

<u>create keypairs at each reboot</u>

Booting (Considering Reboot)

At booting, TRB :

- computes H(OS) and stores in PCR
- creates 2 keypairs for the booted, "active" OS (like "Session key"):
  - $ActiveOSAuth^{pair}$ /* for Authentication
  - $ActiveOSCons^{pair}$ /* for Concellation
- certifies:
  { $ActiveOSAuthK^{pub}$,ActiveOSConsKpub, H(OS)} EK^{priv}
- hands over ActiveOSKeys to booted OS

Attestation (Considering Reboot)

Remote Attestation:

- Challenge: nonce
- Active OS generates response:
  { $ActiveOSCons^{pub}$, $ActiveOSAuth^{pub}$, H(OS)}EK^{priv}
  /* see previous slide
  {nonce'} $ActiveOSAuth^{priv}$

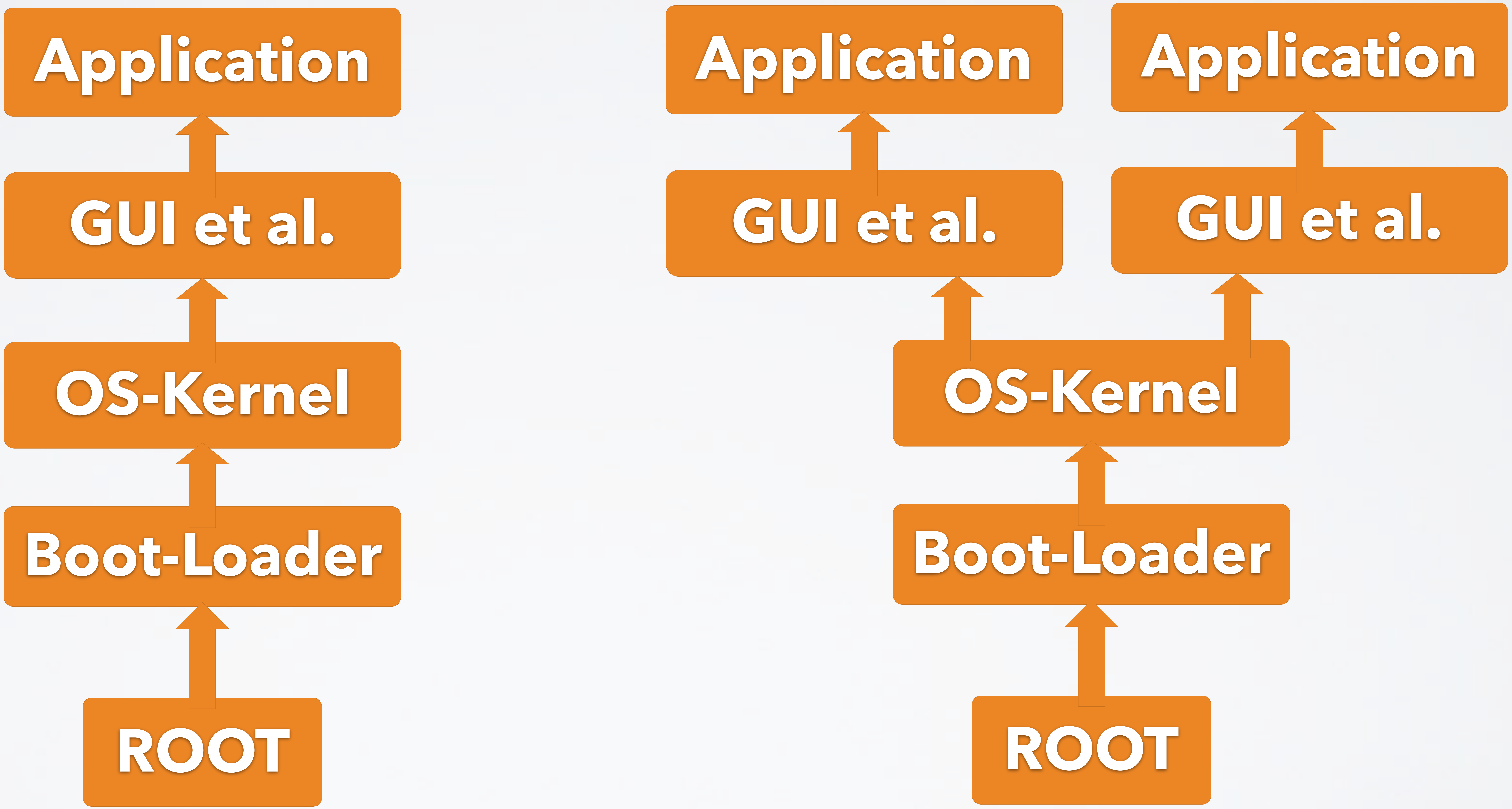
Secure channel:

{ message } $ActiveOSCons^{pub}$

Assumptions

- TRB can protect: EK^{priv}, PCR
  OS can protect: “Active OS keys”
- Rebooting destroys content of
  - PCR
  - Memory Holding “Active OS keys”

Software Stacks and Trees



Software Stacks and Trees

2 Concerns:

- Very large Trusted Computing Base for Booting (including Device Drivers etc)
- Remote attestation of one process (leaf in tree)

Software Stacks and Trees

"Extend" Operation:

- stack: PCRn = H(PCRn-1 || next-component)
- tree: difficult (hearsay, unpublished ?)

Software Stacks and Trees

Key pairs per step:

- OS controls applications → generate key pair per application
- OS certifies
  - { Application 1, App1Kpub } $ActiveOS^{priv}$
  - { Application 2, App2Kpub } $ActiveOS^{priv}$

Late Launch/Dynamic Root of Trust

Problem: huge Software to boot system !!!

- Use arbitrary SW to start system and load all SW
- provide specific instruction to enter "secure mode"
  - set HW in specific state (stop all processors, IO, …)
  - Measure "root of trust" SW and store in PCR

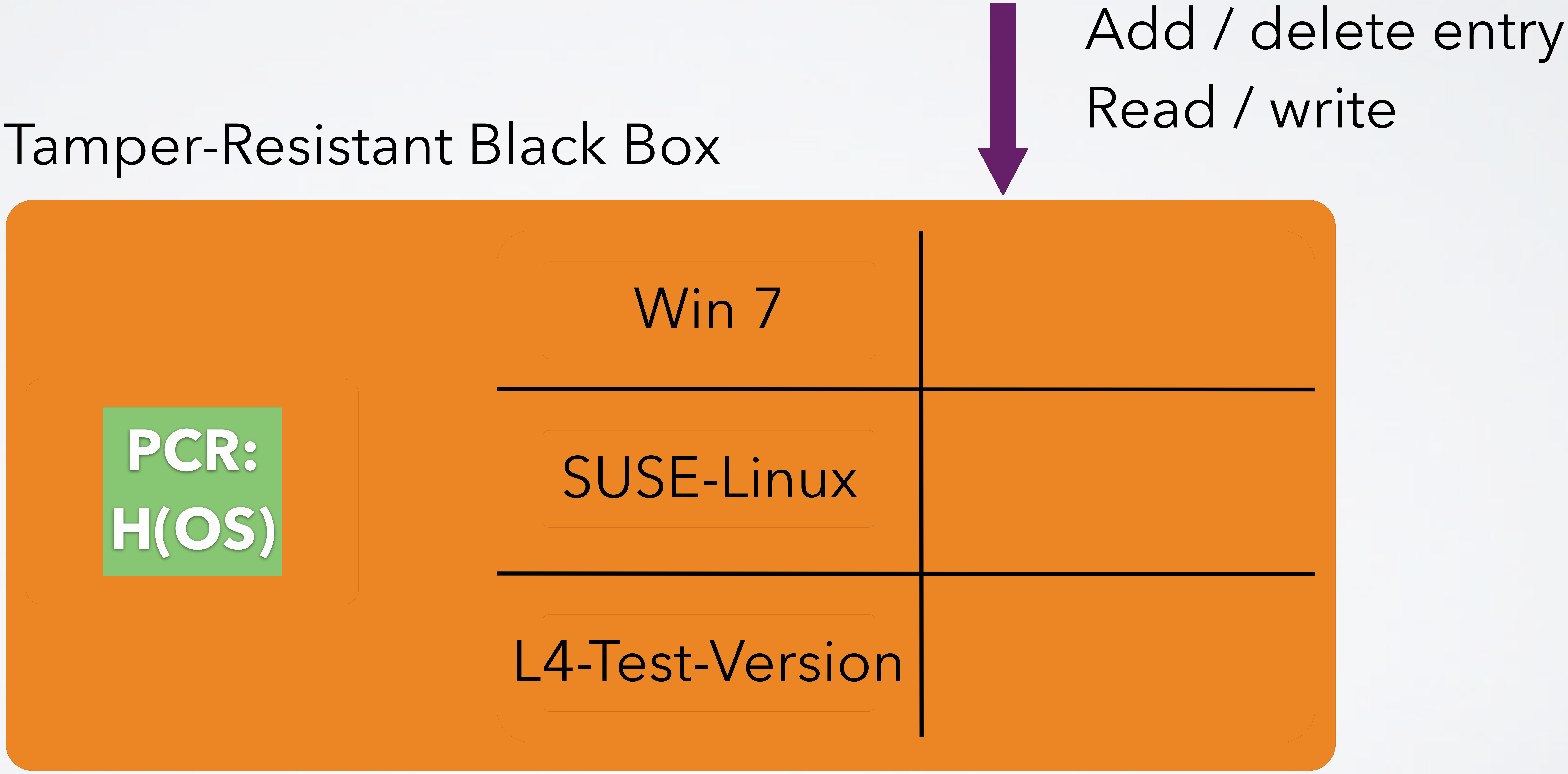
- AMD: "skinit" (Hash) arbitrary root of trust
- Intel: "senter" (must be signed by chip set manufacturer)

Sealed Memory

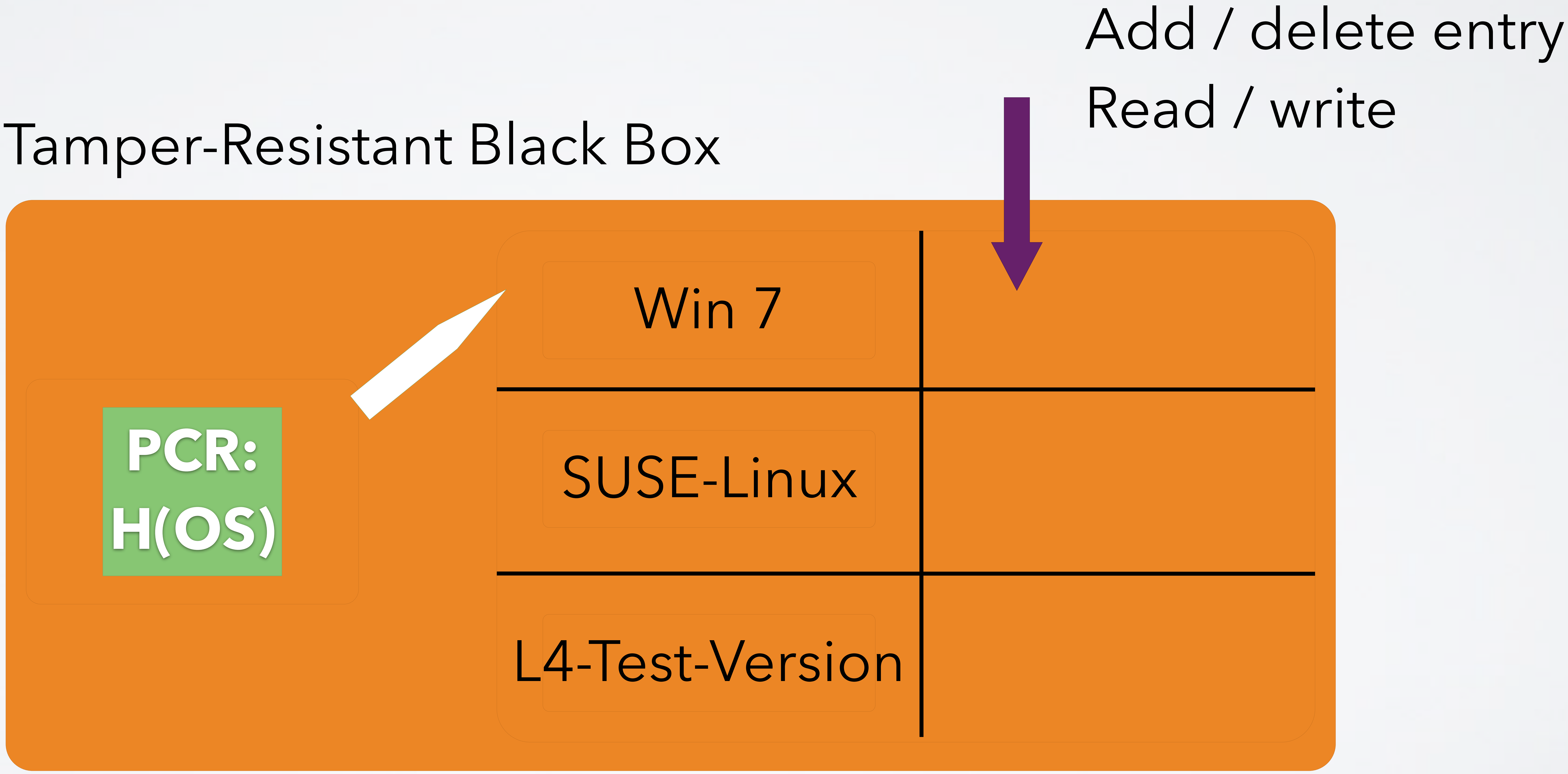
Goal:

- Send information using secure channels
- Bind that information to Software configuration
- Work offline:
  How to store information in the absence of communication channels?
- For example DRM:
  bind encryption keys to specific machine, specific OS

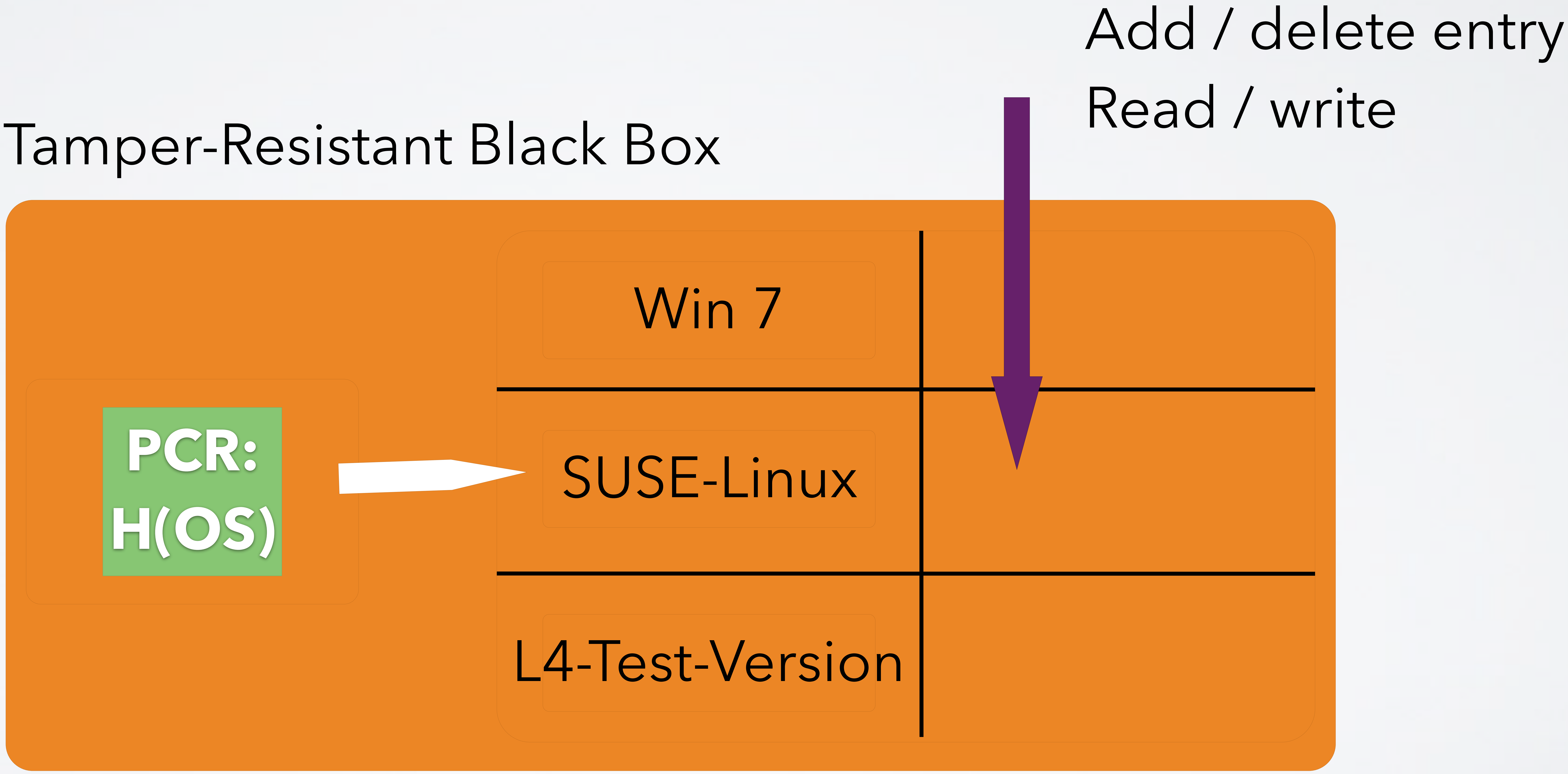
Sealed Memory Principle



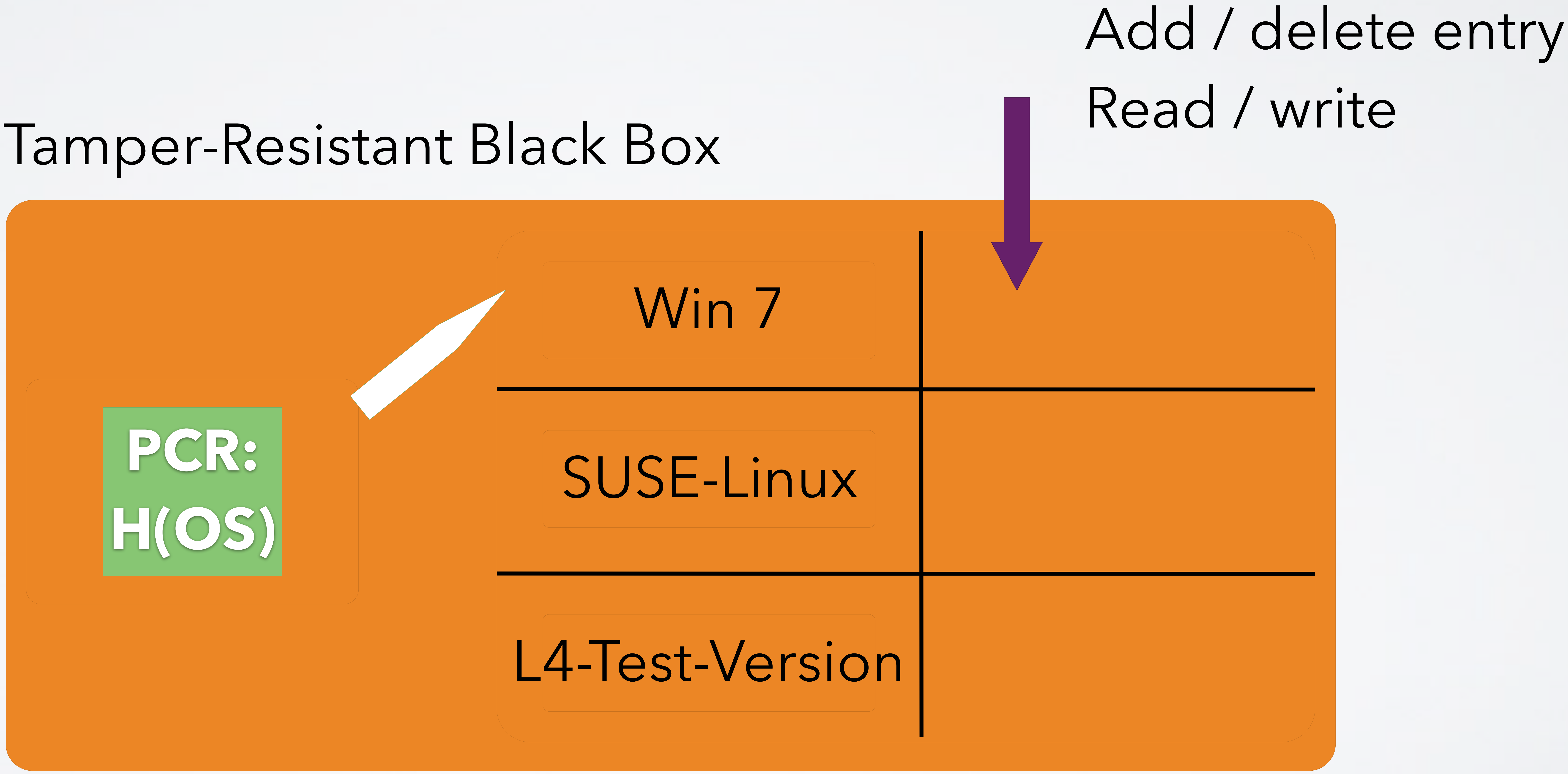
Sealed Memory Principle



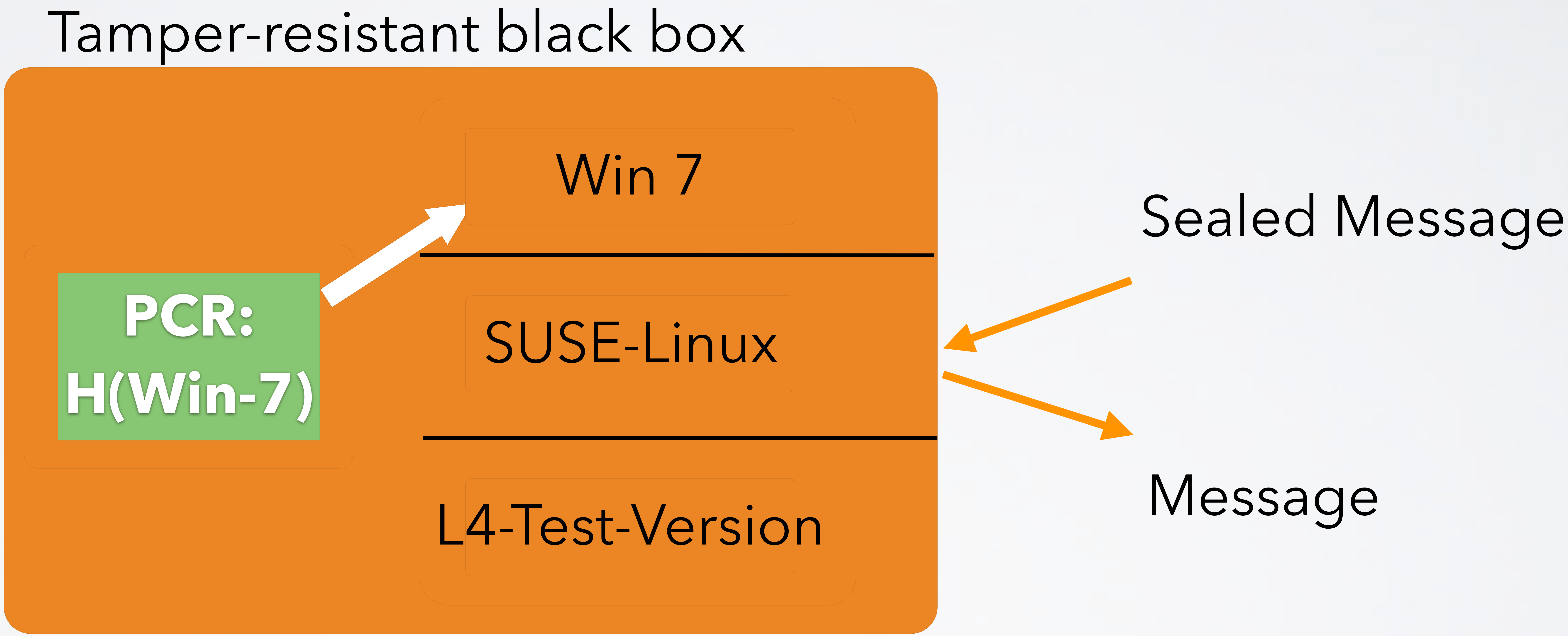
Sealed Memory Principle



Sealed Memory Principle



Sealed Memory: Seal Operation

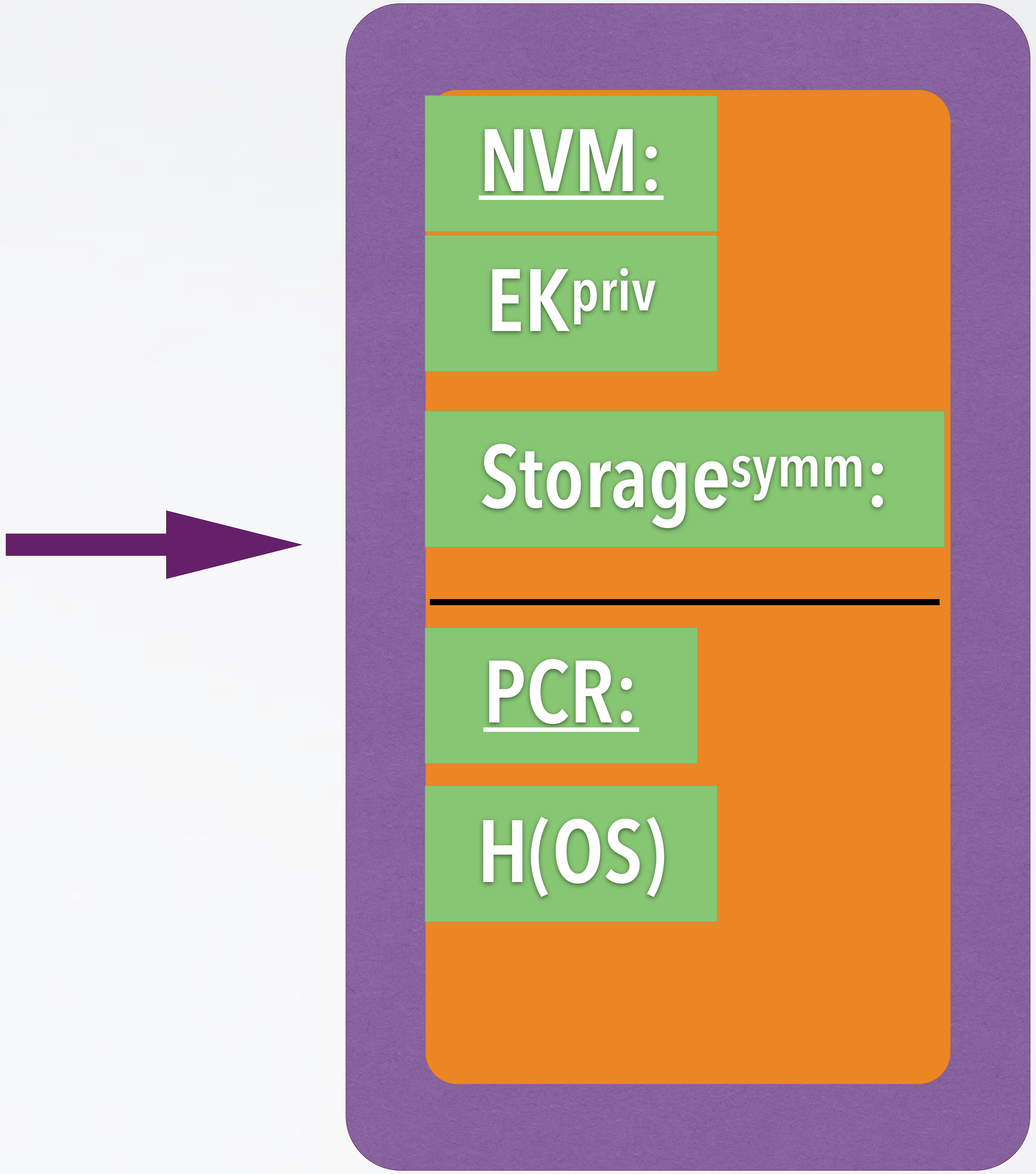


Implementation

TRB generates

symmetric Storage Key

never leaves chip



Sealed Memory

Seal(message):

encrypt("PCR, message", S) → "sealed_message";

emit sealed_message

Unseal(sealed_message):

decrypt(sealed_message, S) → "SealTime_PCR,message";

If SealTime_PCR == PCR

then emit message

else abort

Sealed Memory for future configuration

Seal(message, FUTURE_Config):

encrypt("FUTURE_Config, message", S) → "sealed_message";

emit sealed_message

"seals" information such that it can be unsealed by a future configuration

(for example: future OS version)

Example

- Win8: Seal („SonyOS, Sony-Secret“)
  → SealedMessage (store it on disk)
- L4: Unseal (SealedMessage)
  → SonyOS, Sony-Secret
  → PCR#SonyOS
  → abort
- SonyOS: Unseal(SealedMessage
  → SonyOS, Sony-Secret
  → PCR==SonyOS

Tamper Resistant Box ?

Ideally, includes CPU, Memory, …

Current practice

- Additional physical protection, for example IBM 4758 … look it up in Wikipedia
- HW support:
  - TPM:
    separate "Trusted Platform Modules" (replacing BIOS breaks TRB)
  - Add a new privilege mode: ARM TrustZone
  - raise to user processes: Intel SGX

Protection of Application

Principle Method:

separate critical Software

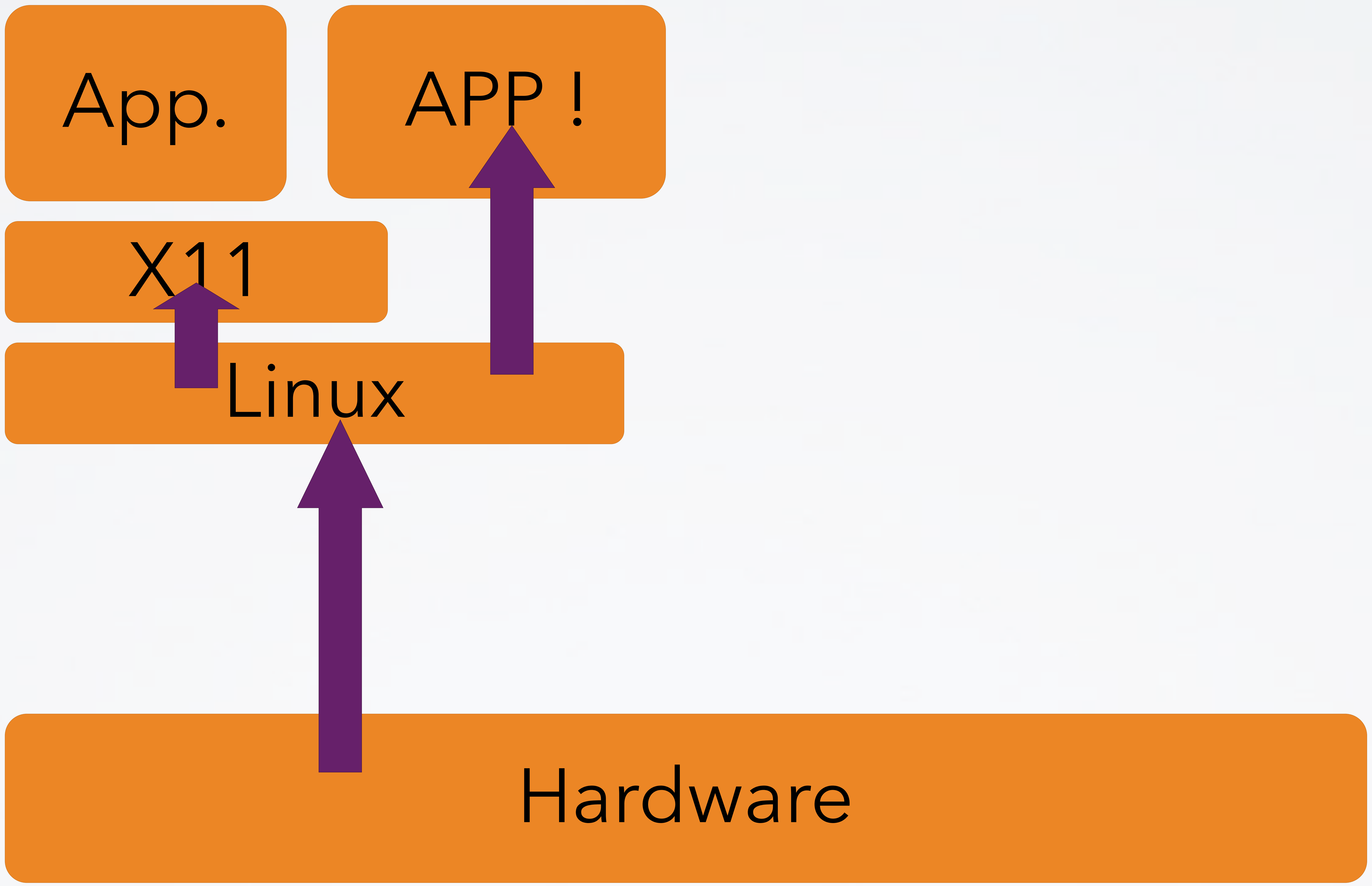
rely on small Trusted Computing Base

- Small OS kernels

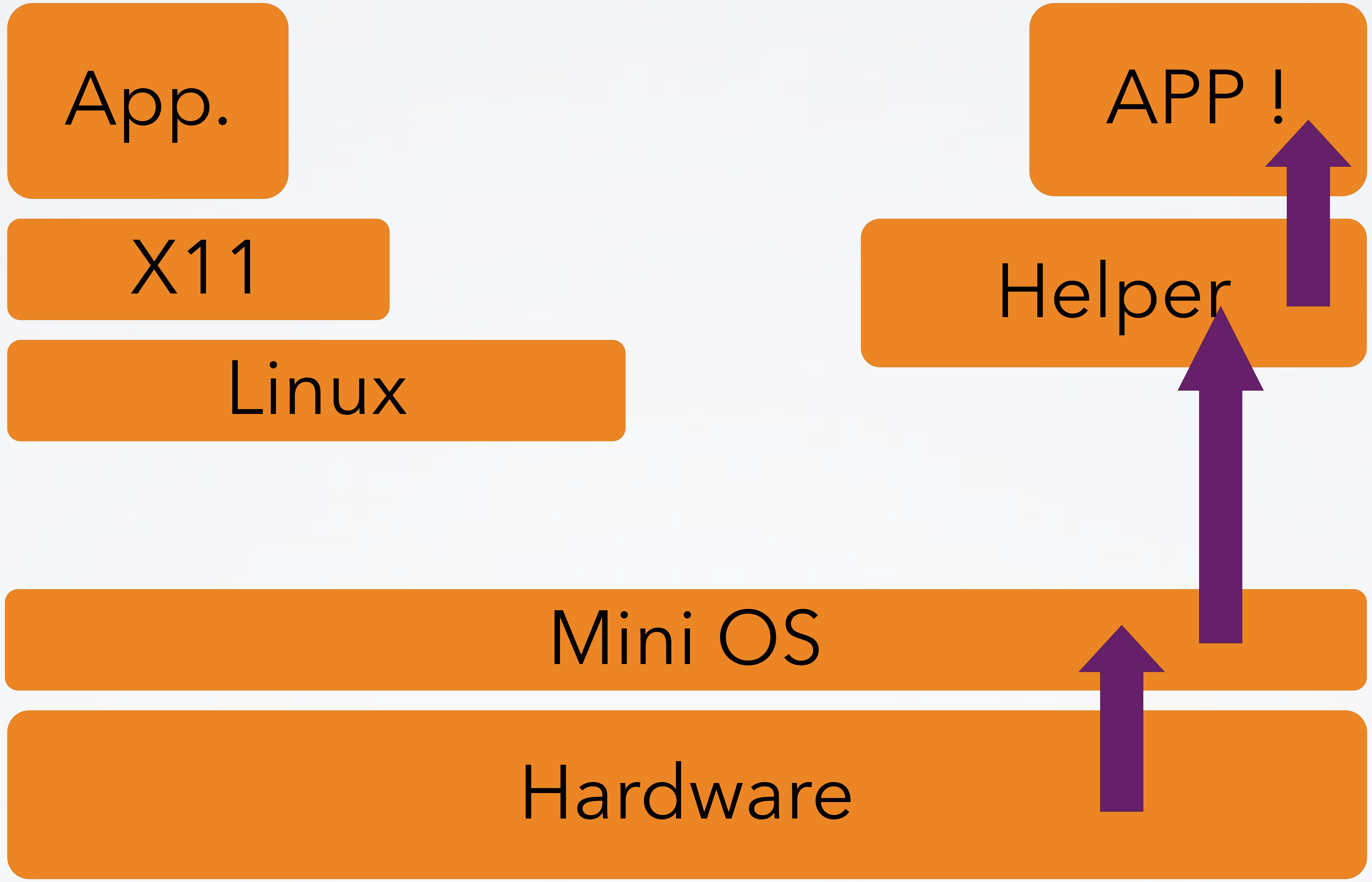
  micro kernels, separation kernels, ….

- Hardware/Microcode Support

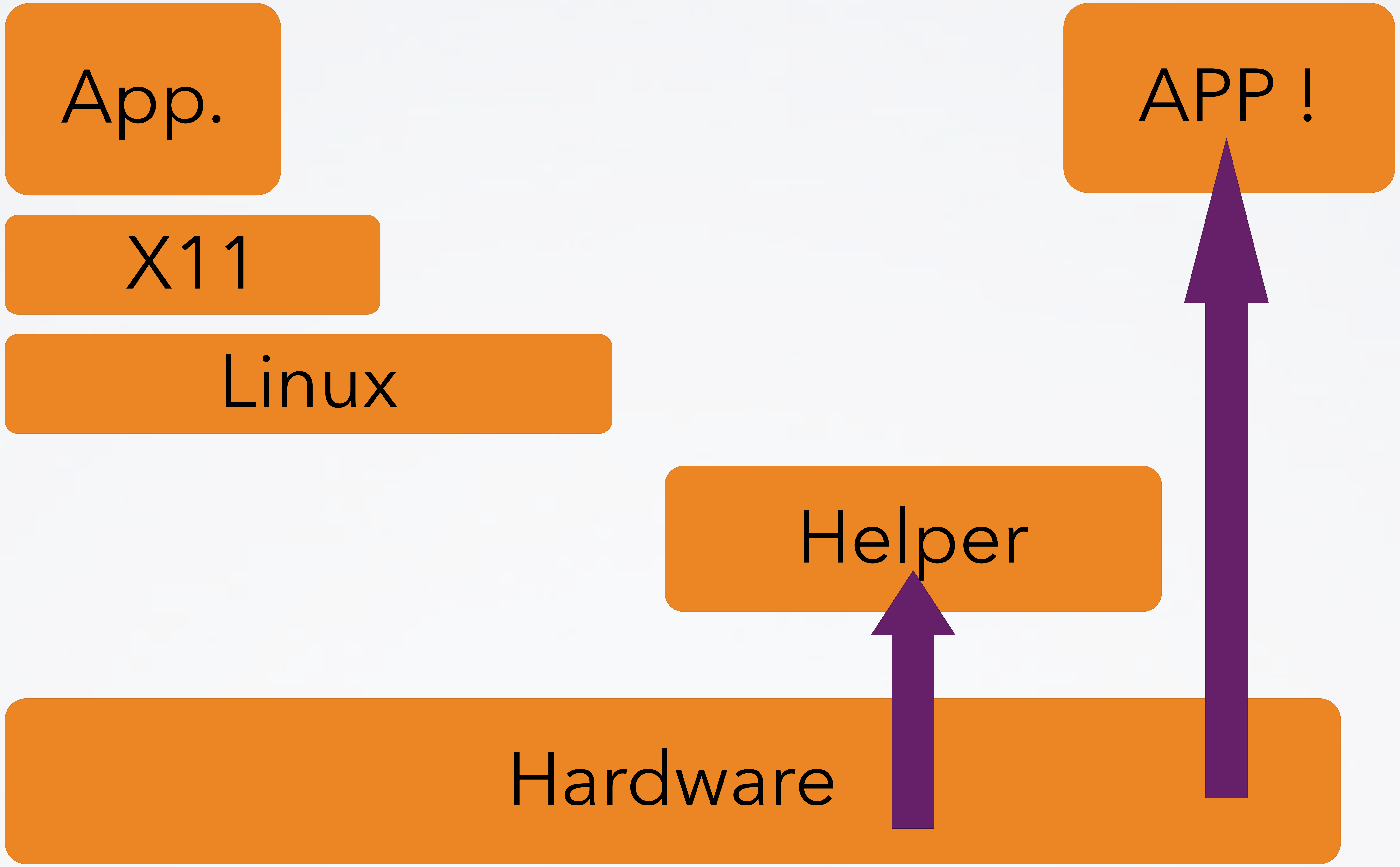
Small Trusted Computing Base



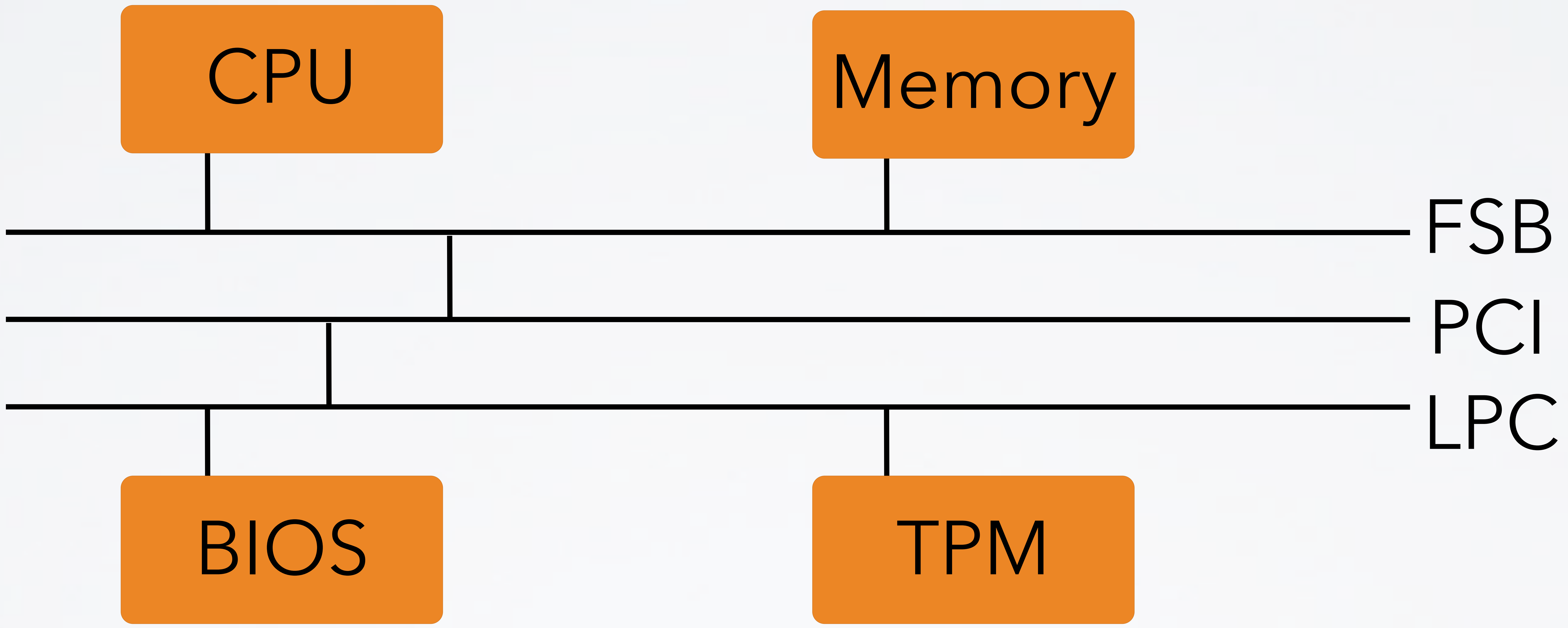
Small Trusted Computing Base



Small Trusted Computing Base



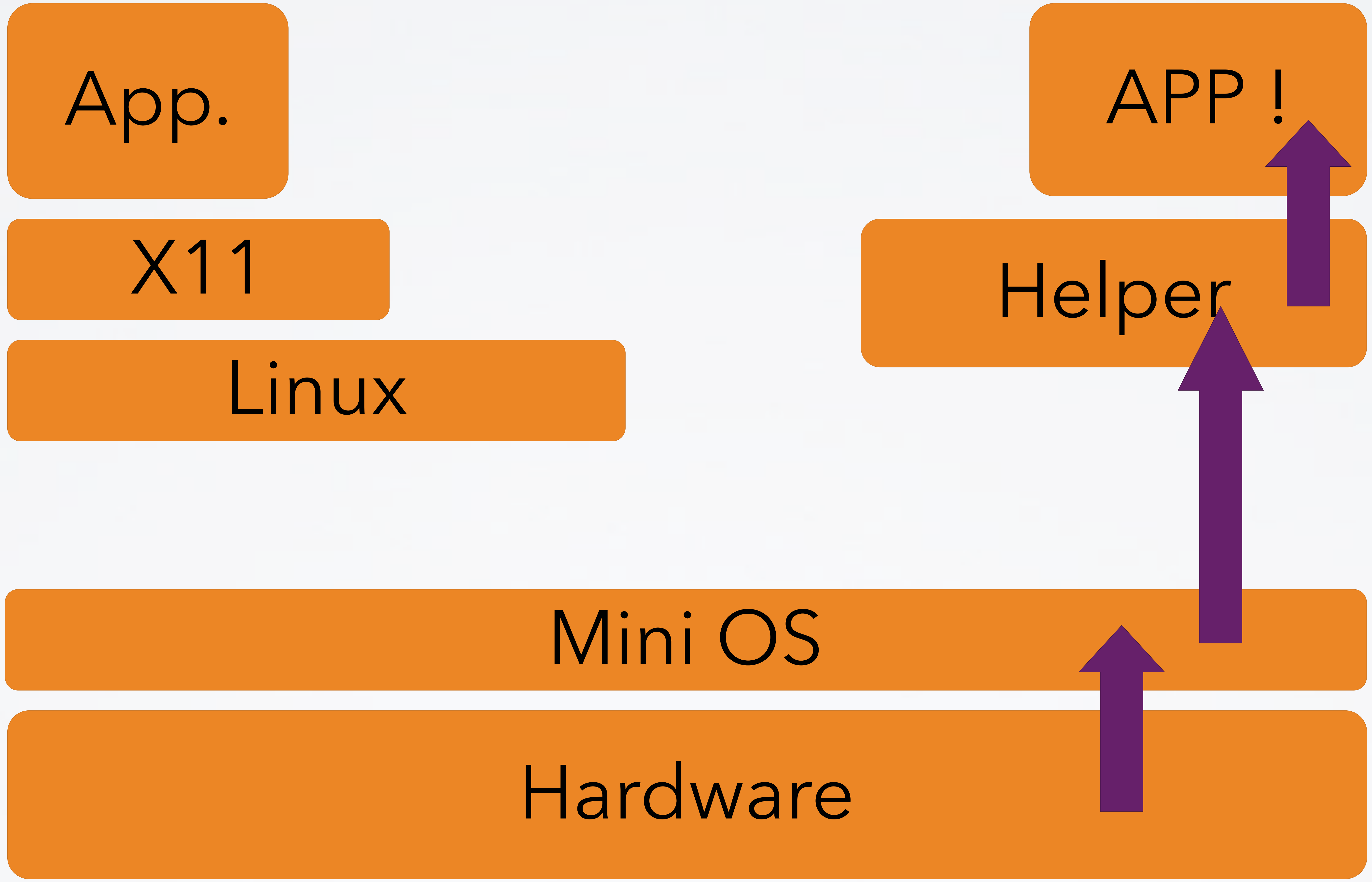
TCG PC Platforms: “Trusted Platform Module” (TPM)



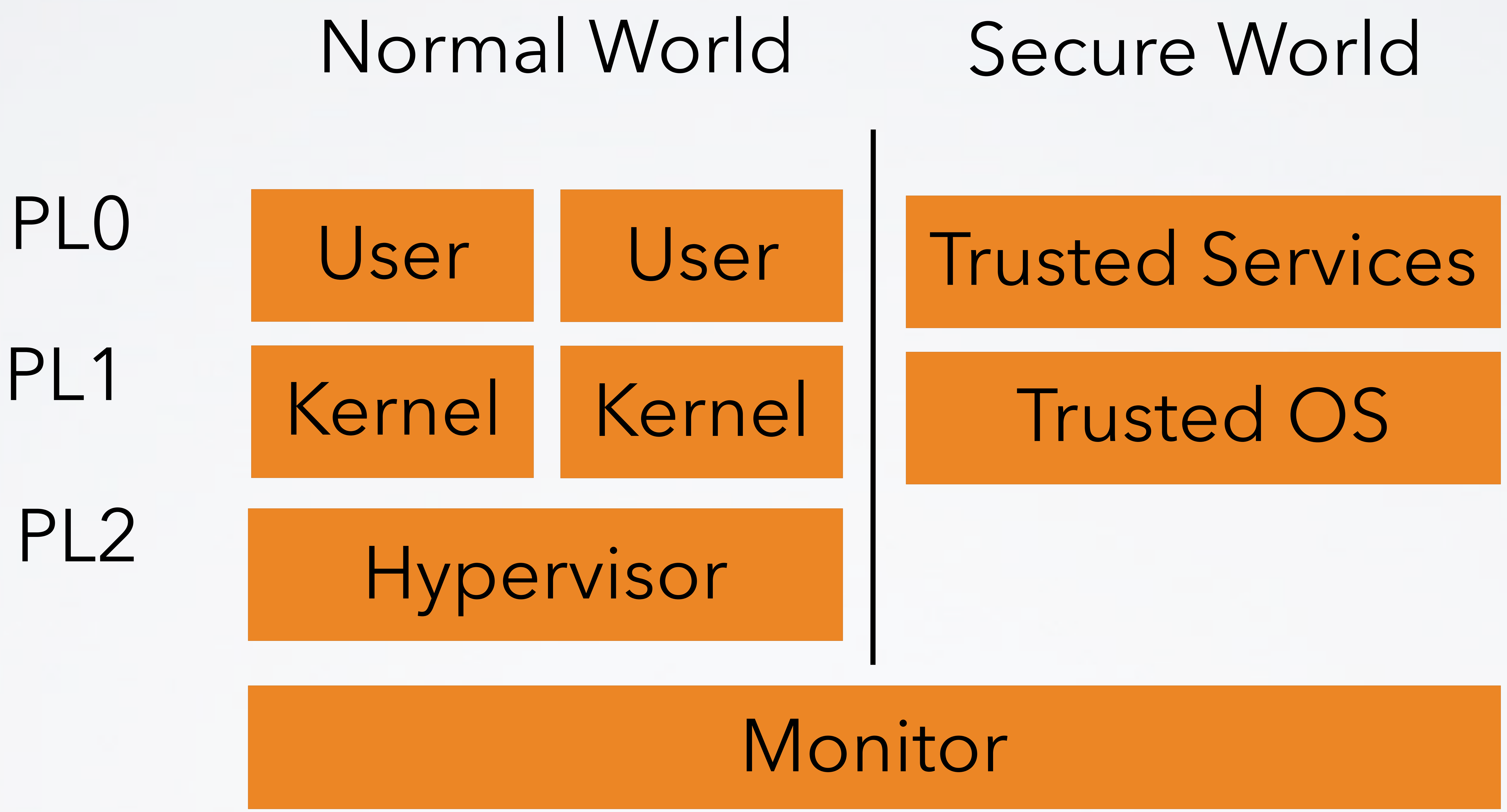
TPM

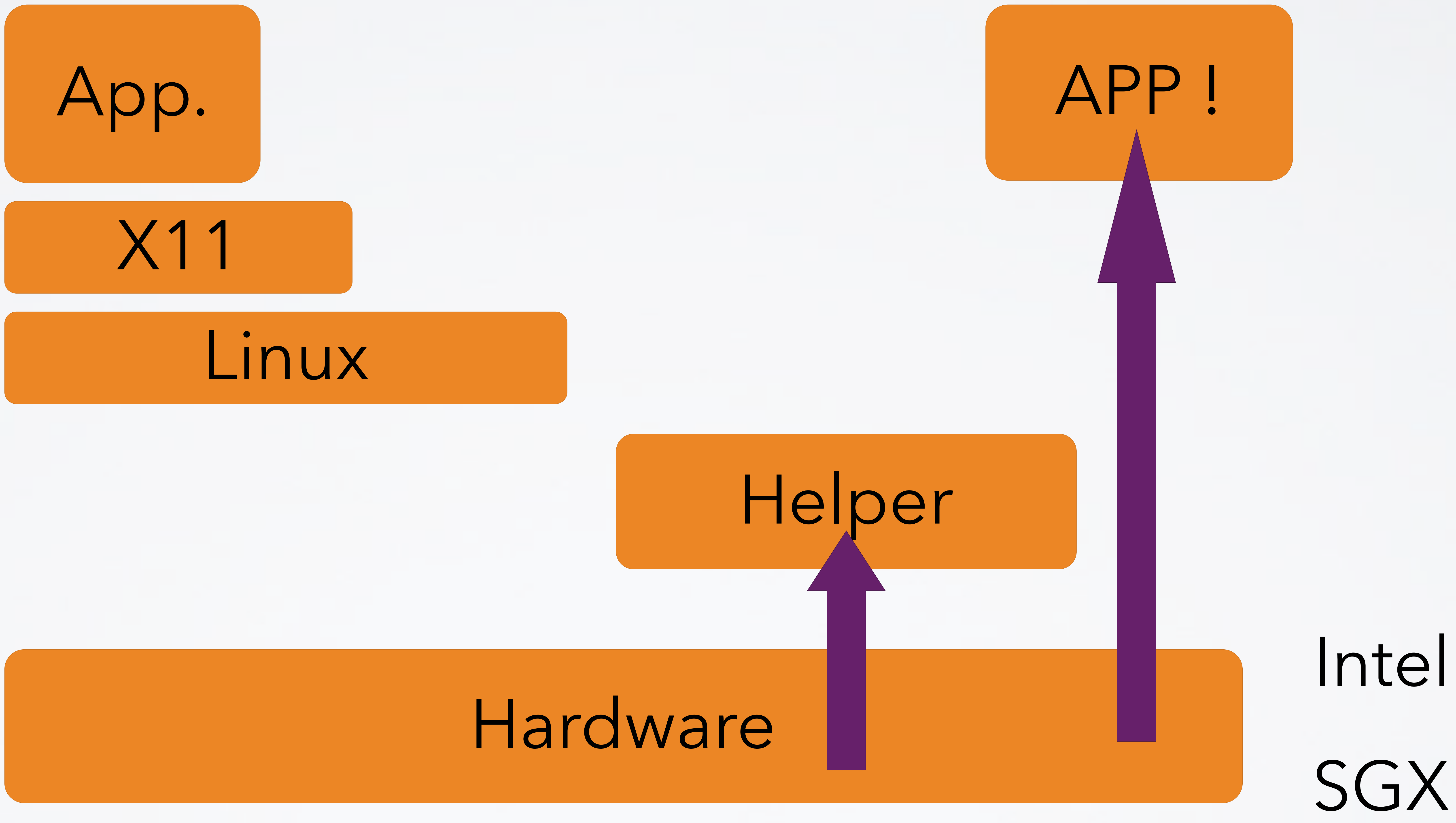

NV Store
PCK
EK / AIK
Internal Program
IO
SHA-1
RSA
Key gen
Random number

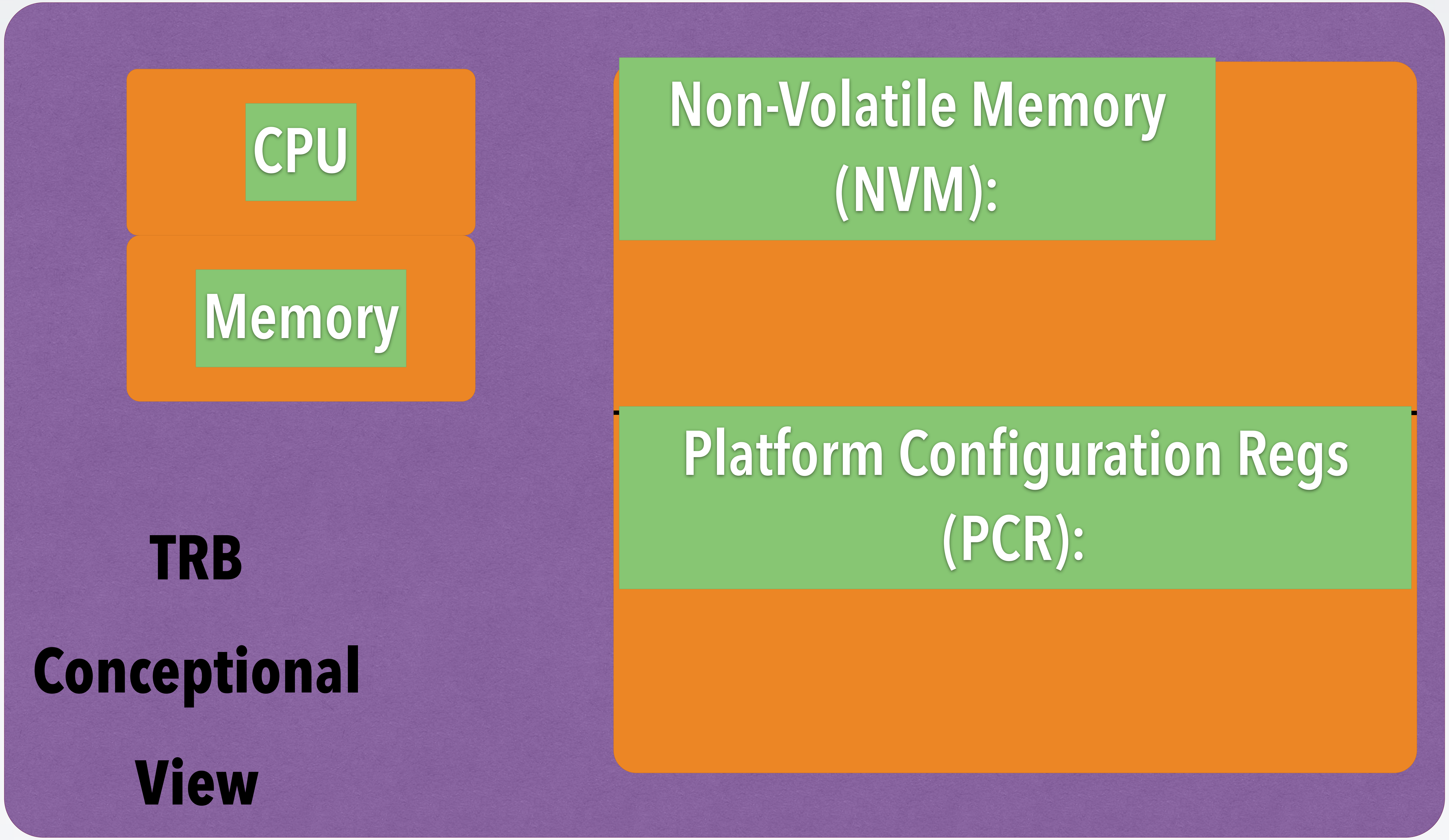
Small Trusted Computing Base

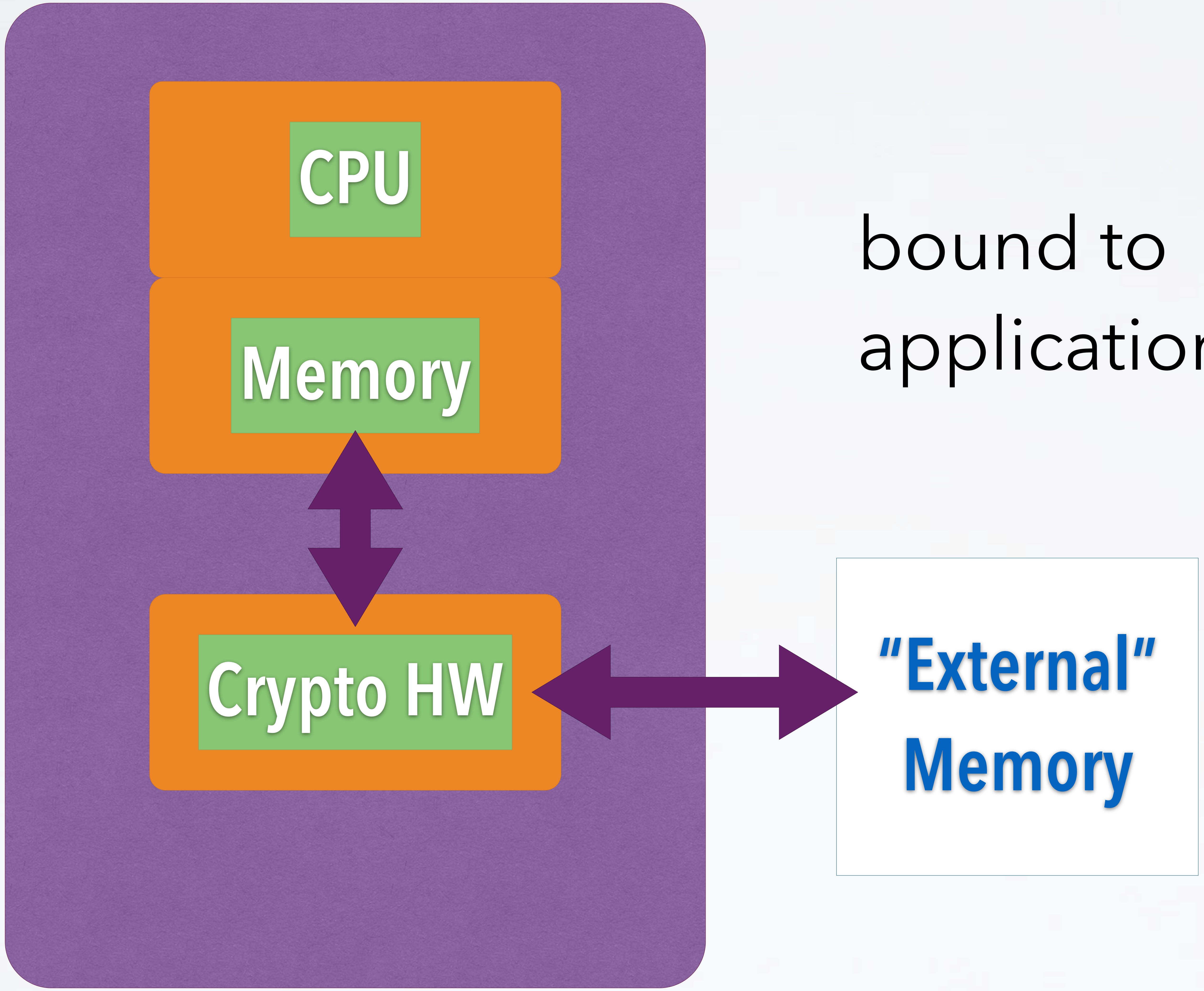


ARM TrustZone




App.
X11
Linux
APP !
Helper
Hardware
Intel
SGX


CPU
Memory
TRB
Conceptional
View
Non-Volatile Memory
(NVM):
Platform Configuration Regs
(PCR):

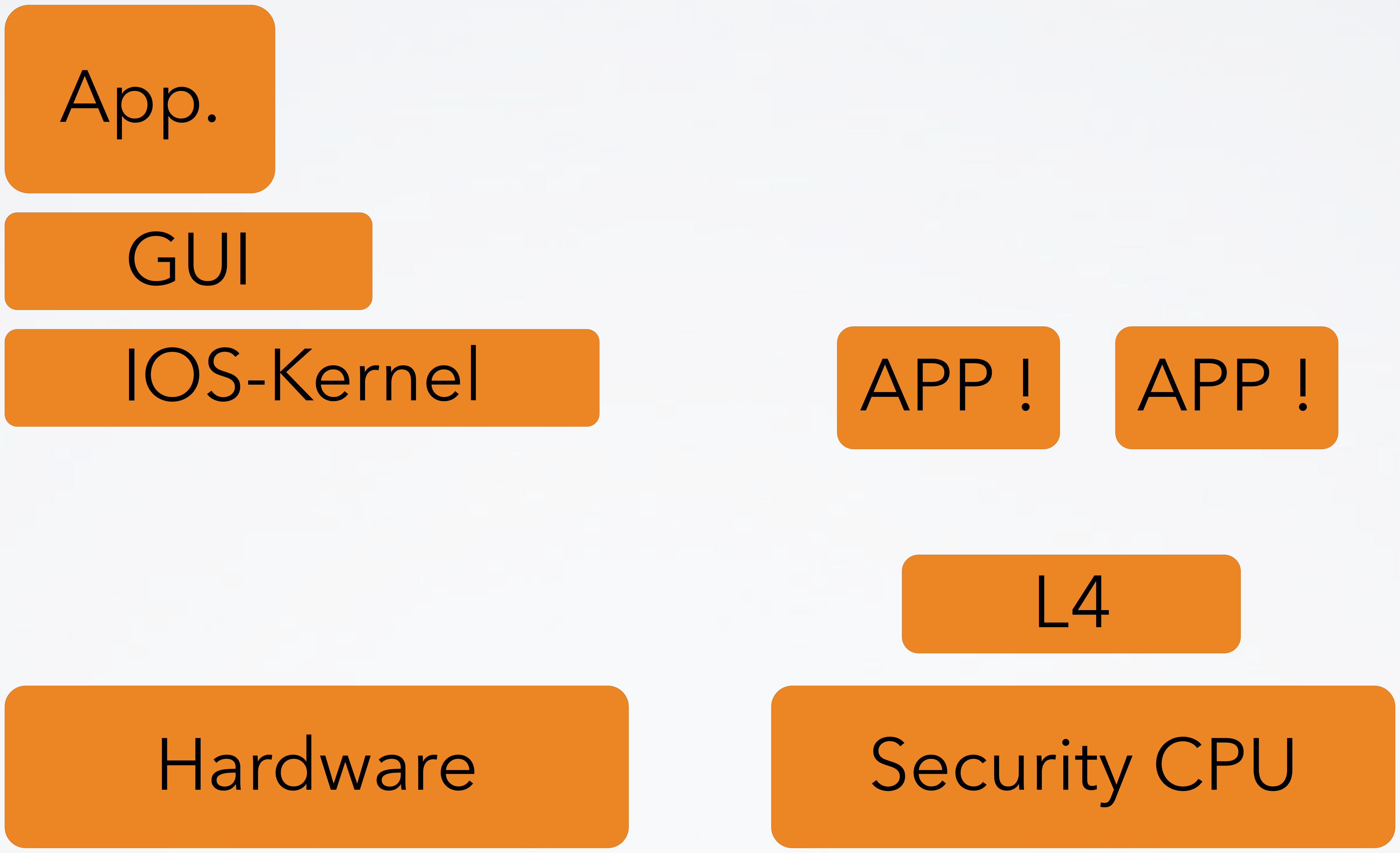


bound to
application "enclaves"

"Enclaves" for Applications:

- established per special new instruction
- measured by HW
- provide controlled entry points
- resource management via untrusted OS

iPhone



References

Important Foundational Paper:

Authentication in distributed systems: theory and practice
Butler Lampson, Martin Abadi, Michael Burrows, Edward Wobber
ACM Transactions on Computer Systems (TOCS)

More References

- TCG Specifications:https://www.trustedcomputinggroup.org/groups/TCG_1_3_Architecture_Overview.pdf
- ARM Trustzone & Intel SGX vendor sources