TECHNISCHE
UNIVERSITÄT
DRESDEN

# "TRUSTED" COMPUTING

# DISTRIBUTED OPERATING SYSTEMS

**Hermann Härtig**

Summer 2016

# 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 variants of implementation in HW (TPM, SGX)

Beware of terminology changes !

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

- Attention: terminology has changed

# 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 PROBLEMS

- 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

## 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

# PROTECTION OF APPLICATION

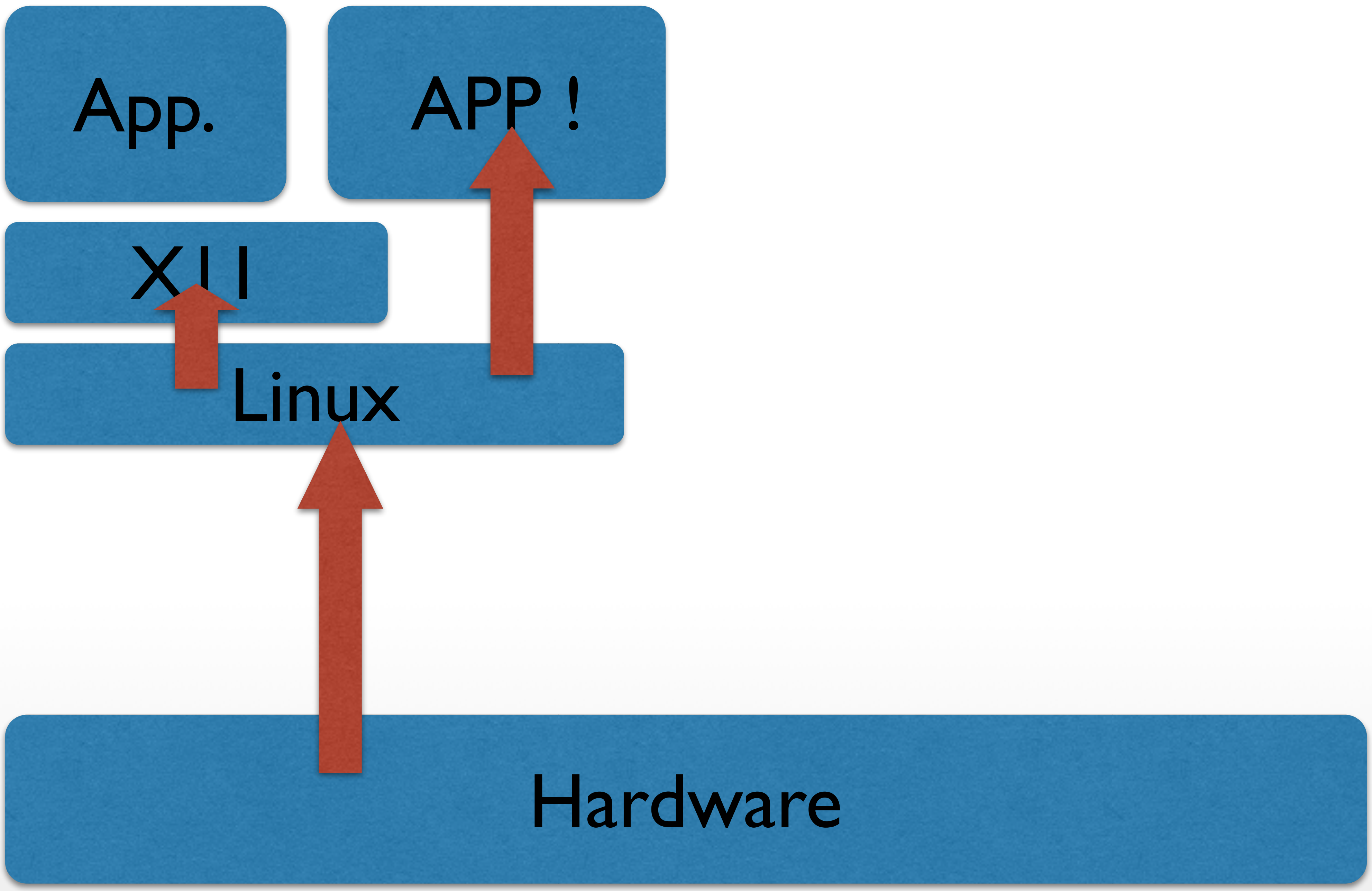
Principle Method:

separate critical Software

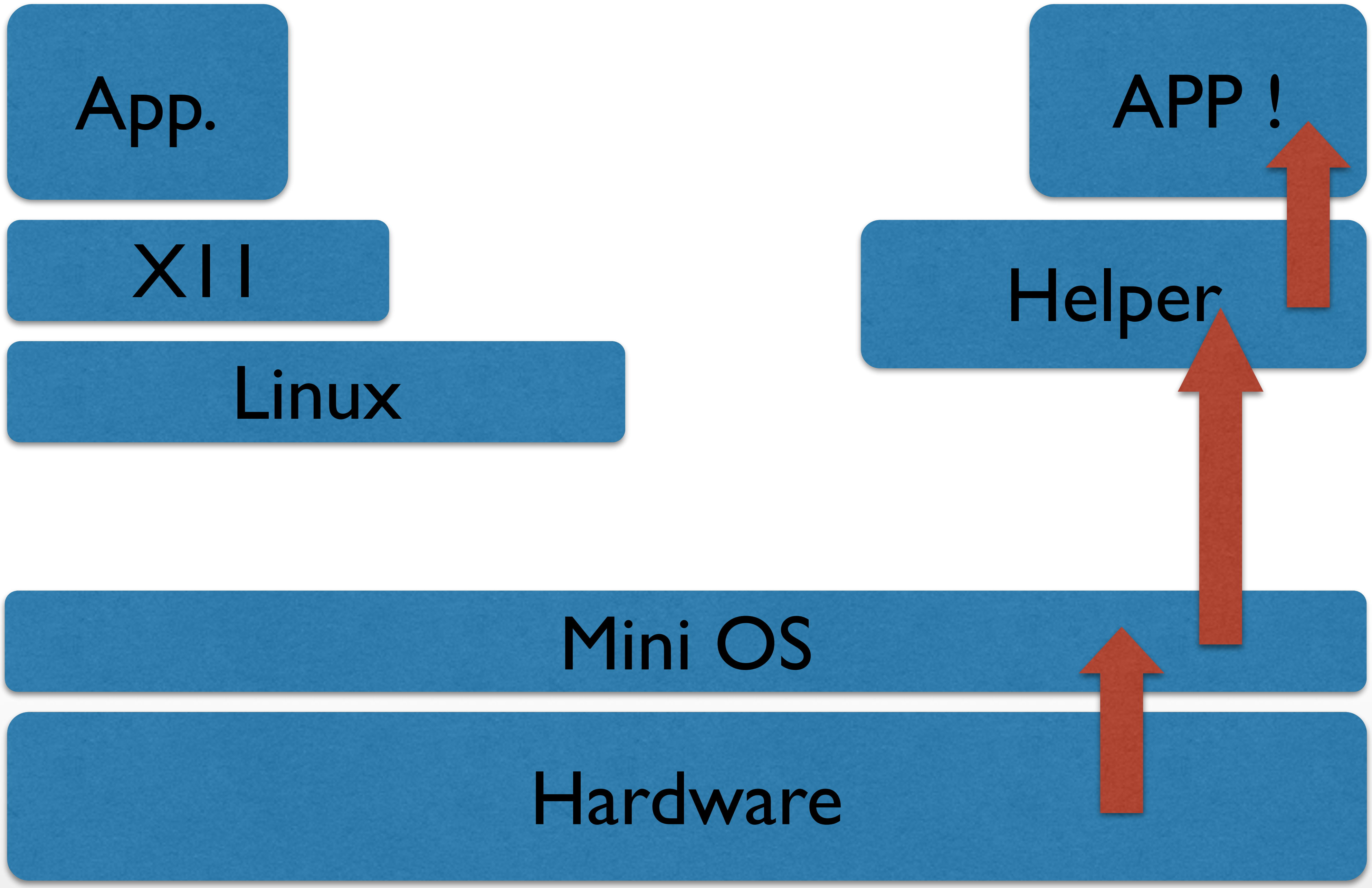
rely on small Trusted Computing Base

- Small OS kernels
  micro kernels, separation kernels, ….
- Hardware

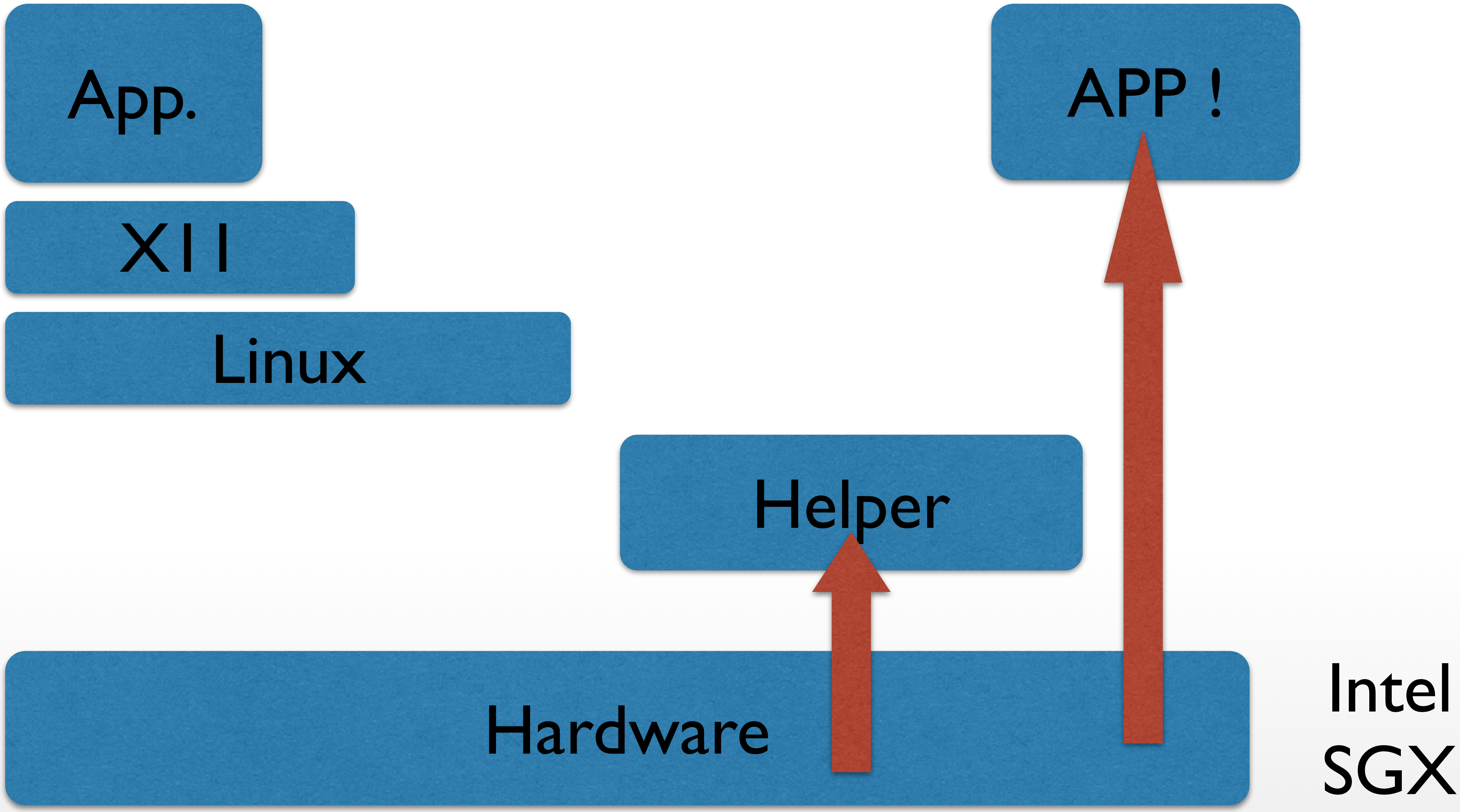
# SMALL TRUSTED COMPUTING BASE



# SMALL TRUSTED COMPUTING BASE



# SMALL TRUSTED COMPUTING BASE



# NOTATION

SKpriv SKpub Asymmetric key pair of some entity S

- { M }XKpriv Digital Signature for message M using the private key of signer X
- { M }YKpub Message encrypted using public concellation key of Y

- H(M) Collision-Resistant Hash Function

- Certificate by authority Ca:

  { ID, SKpub , other properties } CaKpriv

# NOTATION

Note:

- "{ M }SkprivDigital Signature"
  is short for: encrypt(H(M),Skpriv)

- "{ M }Skpub    Message concealed …"
  does not necessarily imply
  public key encryption for full M
  (rather a combination of
  symmetric and asymmetric methods)

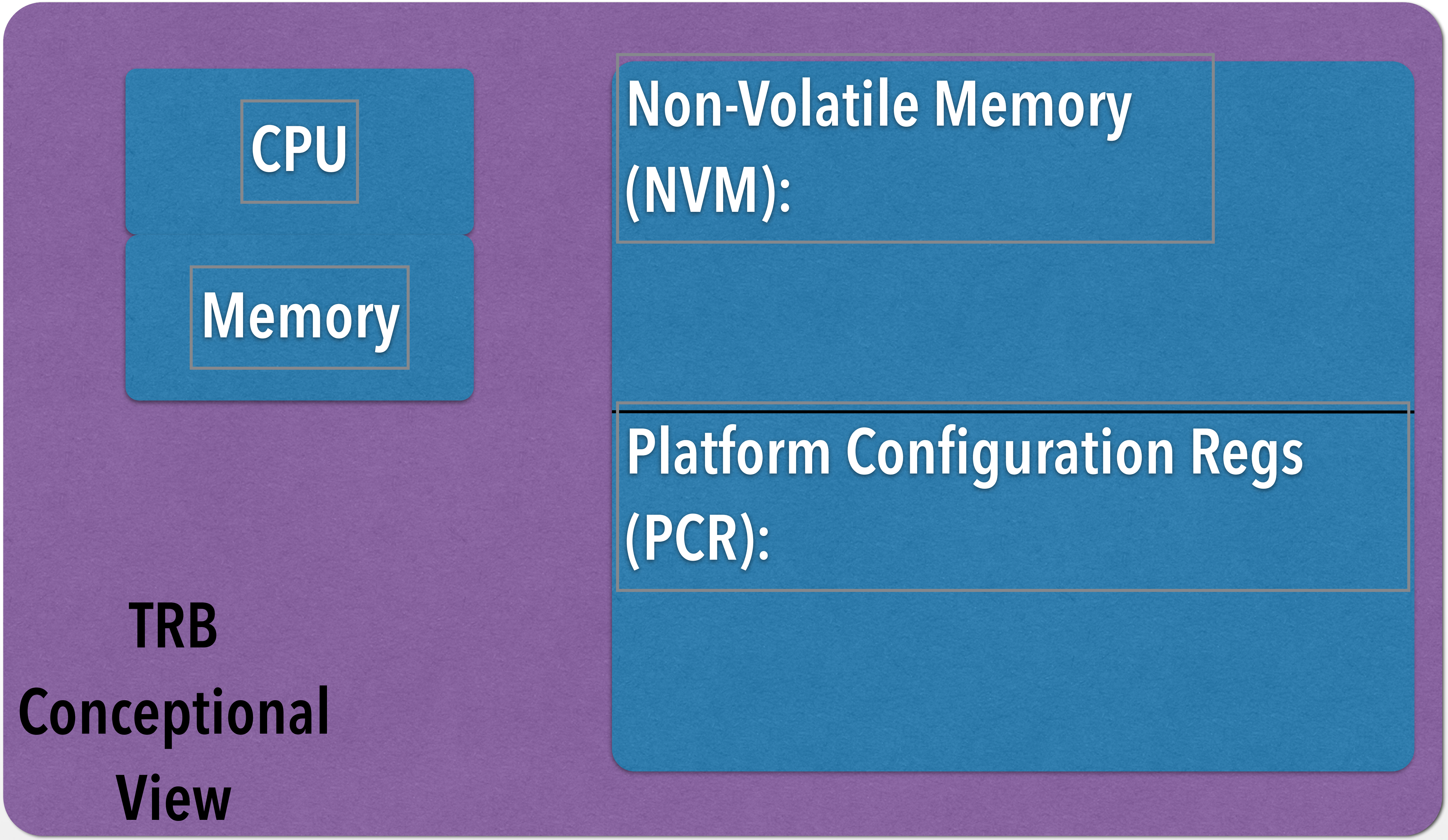
# IDENTIFICATION OF SOFTWARE

Program vendor: Foosoft FS

Two ways to identify Software: Hash / public key

- H(Program)
- {Program, ID- Program}FSKpriv
  use FSKpub to check
  the signature must be made available,
  e.g. shipped with the Program
- The „ID" of SW must be known.

# TAMPERRESISTANT BLACK BOX(TRB)



# WAYS TO "BURN IN" THE OS OR SECURE BOOTING

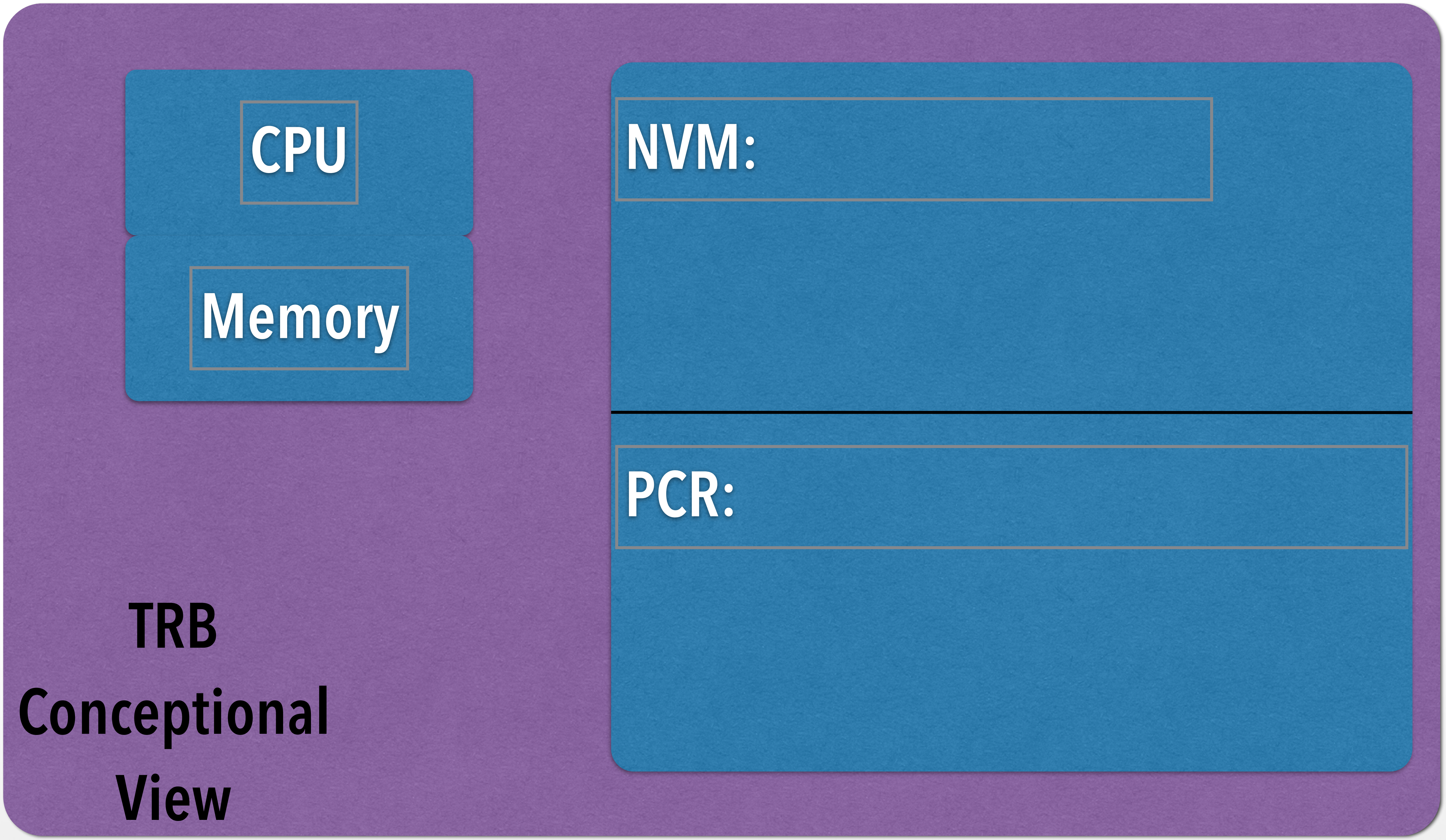
- Read-Only Memory
- H(OS) in NVM preset by manufacturer
  - load OS- Code
  - compare H(loaded OS code) to preset H(OS)
  - abort if different
- FSKpub in NVM preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using FSKpub
  - abort if check fails

# AUTHENTICATED BOOTING, USING HASH

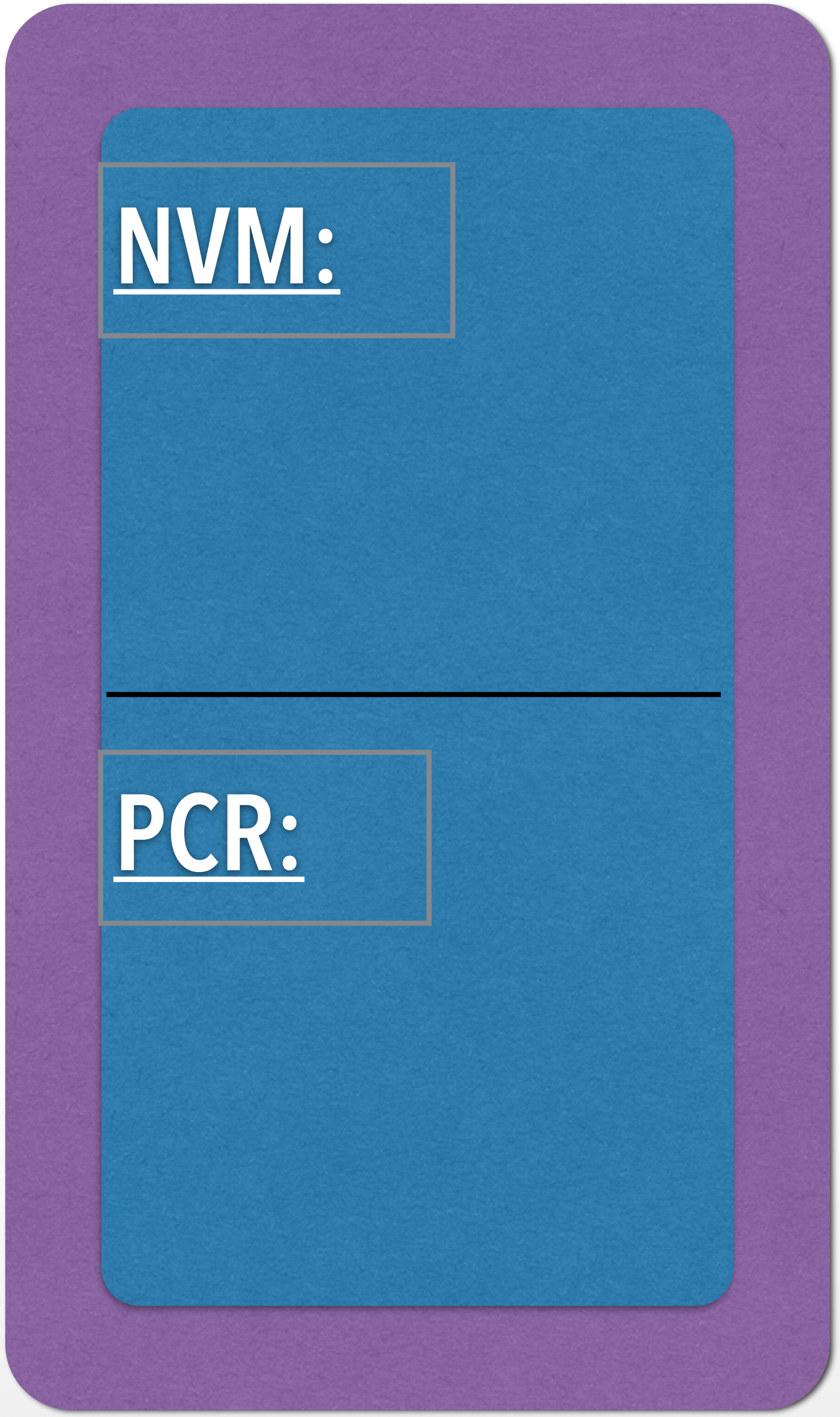
Steps:

1. Preparation by TRB and OS Vendors
2. Booting & "Measuring"
3. Remote attestation

# TAMPERRESISTANT BLACK BOX(TRB)

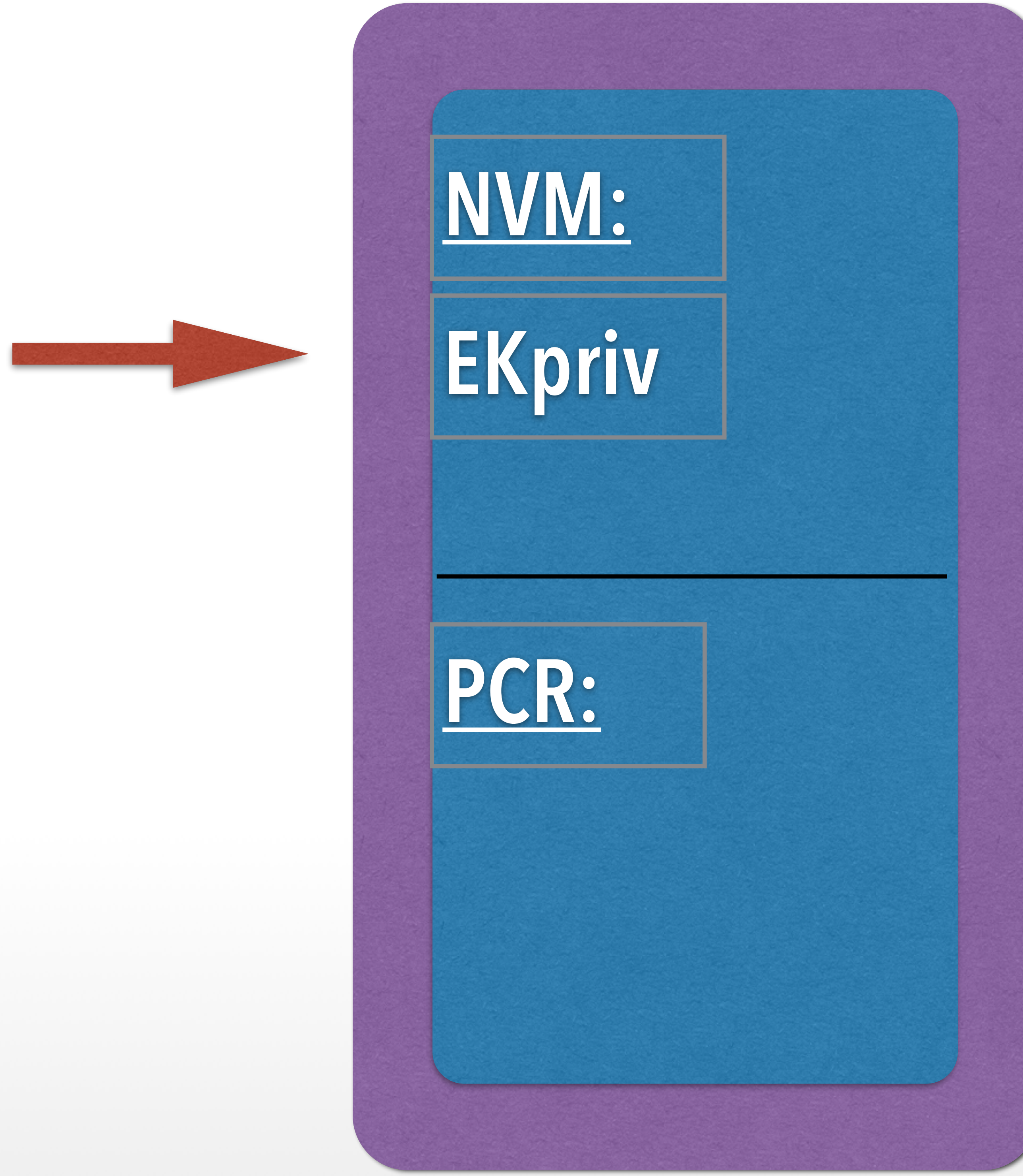


# TAMPERRESISTANT BLACK BOX(TRB)



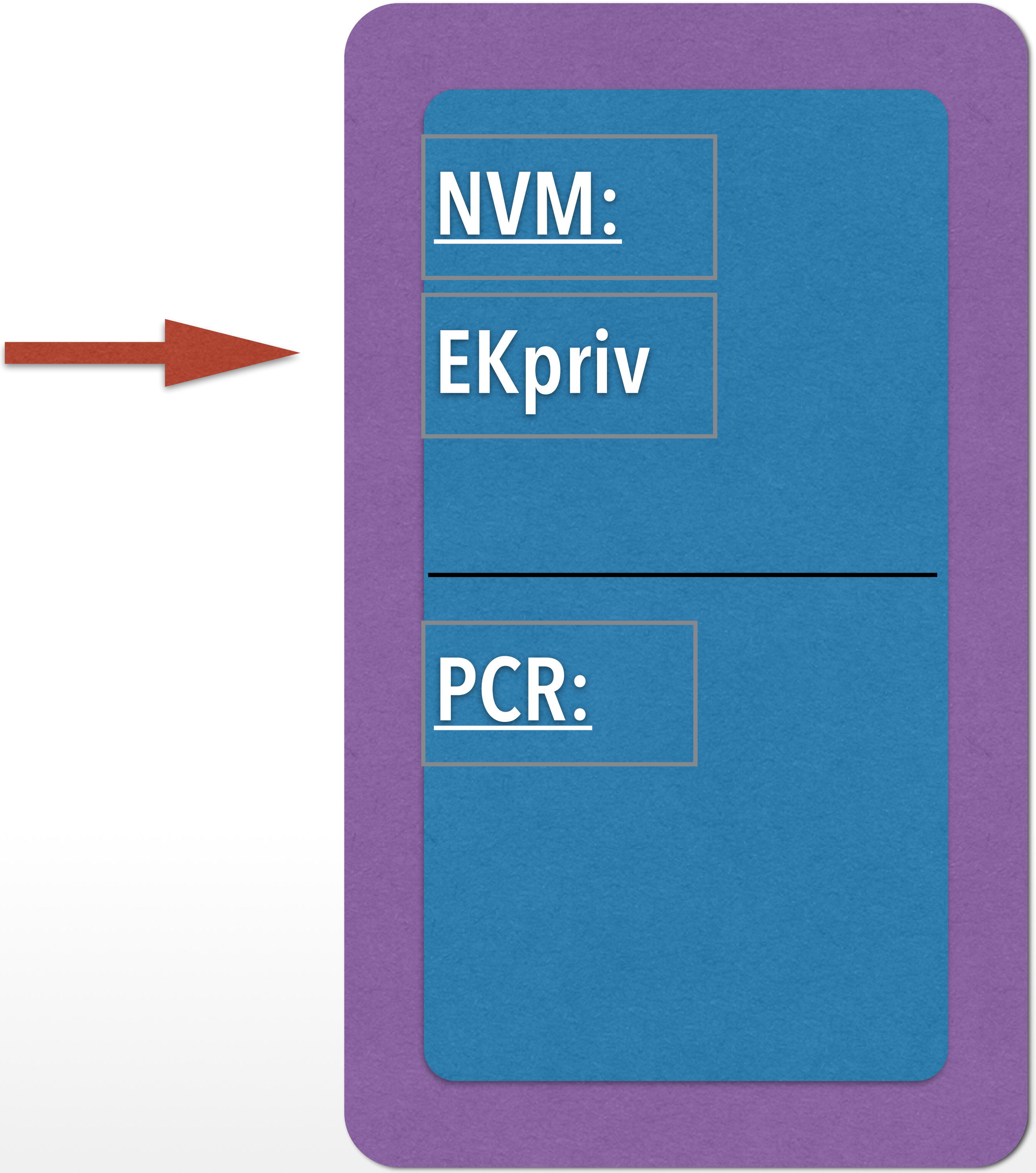
# TRB VENDOR

TRB generates key pair:
„Endorsement Key“ (EK)
stores in TRB NVM
emits EKpub



# OS VENDOR

TRB generates key pair:
„Endorsement Key“ (EK)
stores in TRB NVM
emits EKpub

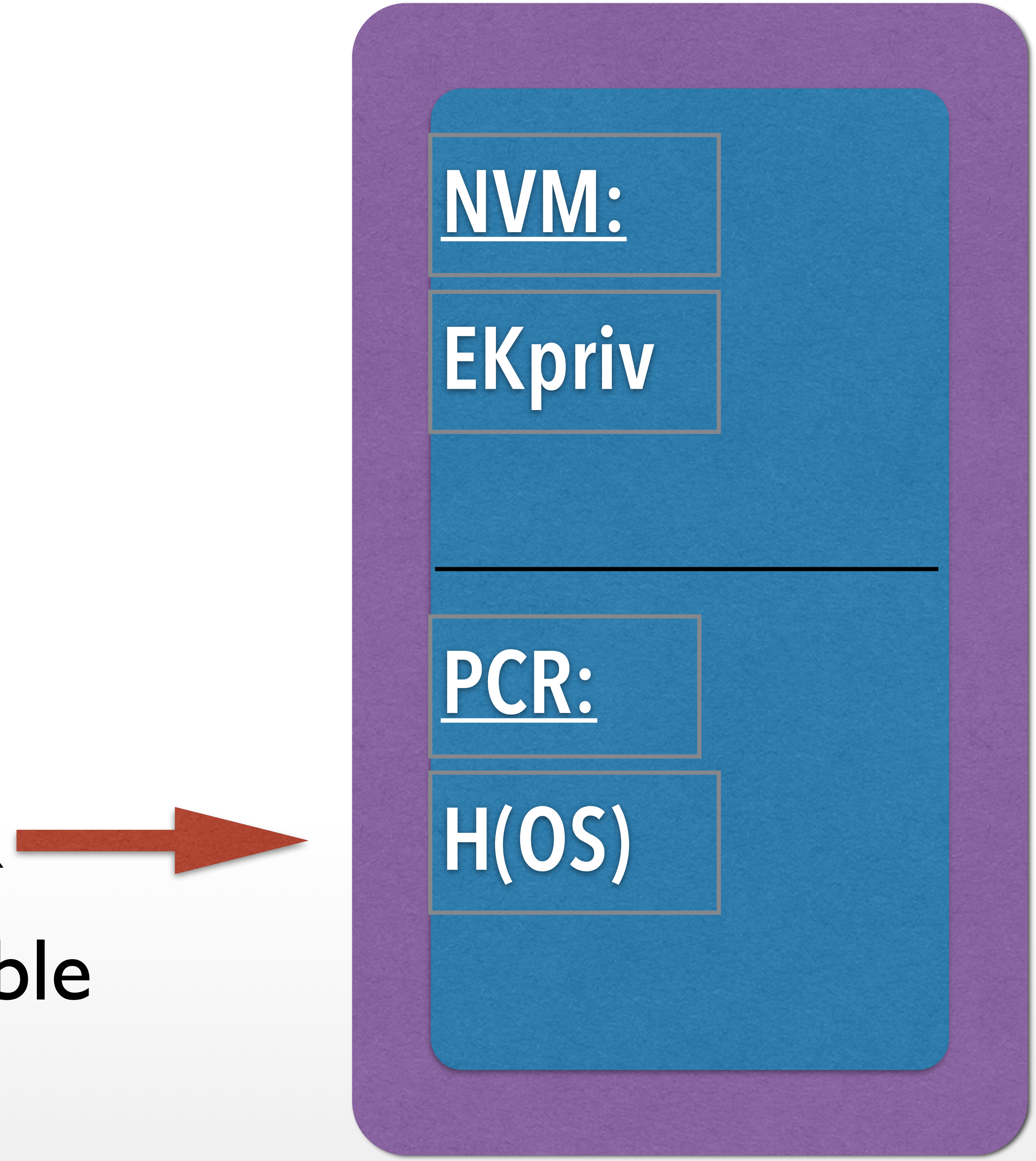


# TRB AND OS VEDNOR

- TRB vendor certifies:
  {"a valid EK", EKpub}TRB_VKpriv
- OS-Vendor certifies:
  {„a valid OS", H(OS)}OS_VKpriv
- serve as identifiers:
  EKpub and H(OS)

TRB:

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



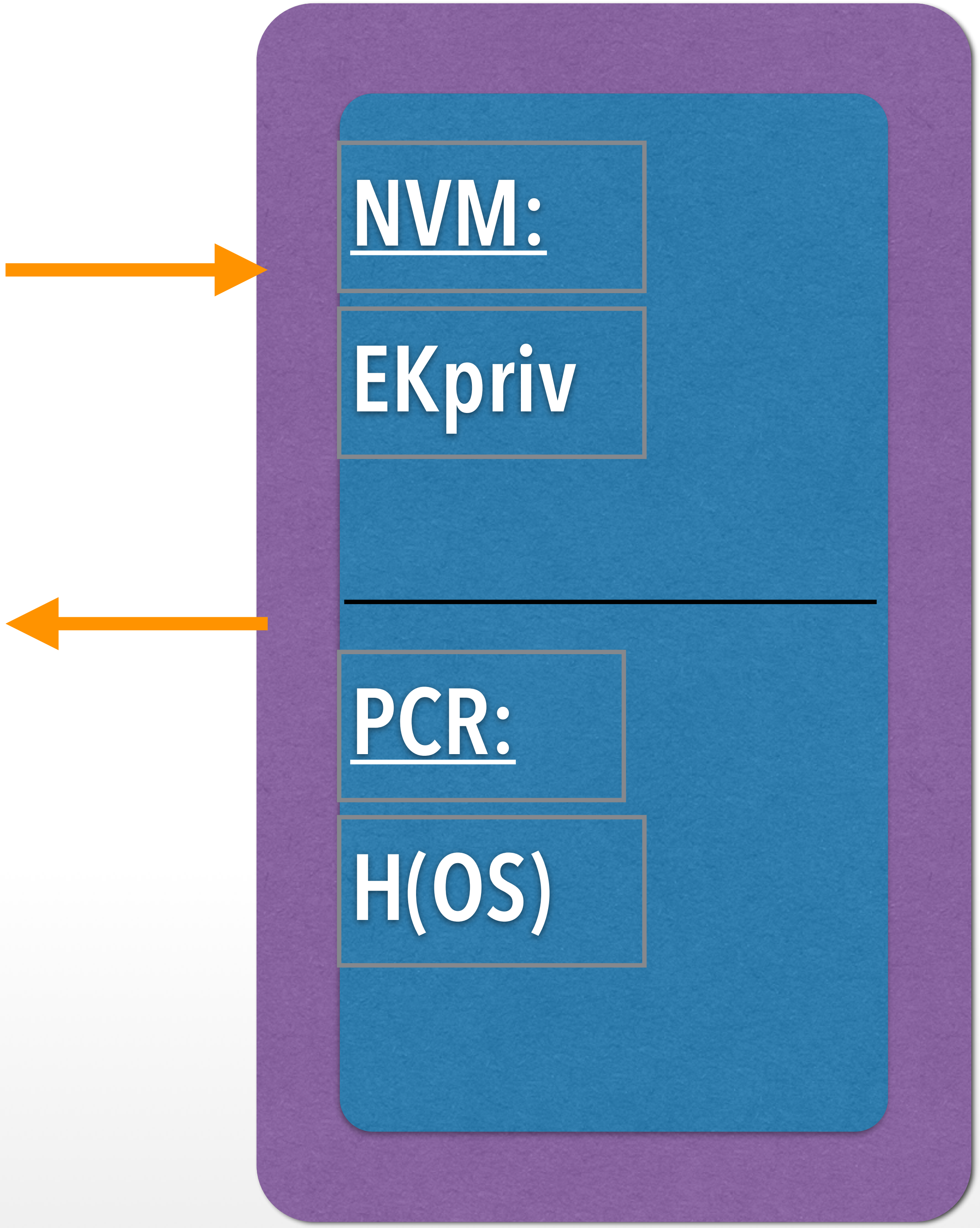

PCR not (directly) writable by OS

more later

# ATTESTATION (FIRST BASIC EXPLANATION)

Challenge:
send NONCE

Response:
{NONCE', PCR}EKpriv



# PROBLEM

- boot Linux

  → challenge

  ← response "Linux"

- reboot Windows

  → send data

add one step of indirection:

create keypairs at each reboot

# BOOTING (CONSIDERING REBOOT)

At booting, TRB :

- computes H(OS) and stores in PCR
- creates 2 keypairs for the booted, "active" OS :
  - ActiveOSAuthK /* for Authentication
  - ActiveOSConsK /* for Concellation
- certifies: { ActiveOSAuthKpub, ActiveOSConsKpub,H(OS)}EKpriv
- hands over ActiveOSKeys to booted OS

# ATTESTATION (CONSIDERING REBOOT)

Remote Attestation:

- Challenge: nonce
- Active OS generates response:
  { ActiveOSConsKpub, ActiveOSAuthKpub,
      H(OS)}EKpriv  /* see previous slide
  {nonce'} ActiveOSAuthKpriv

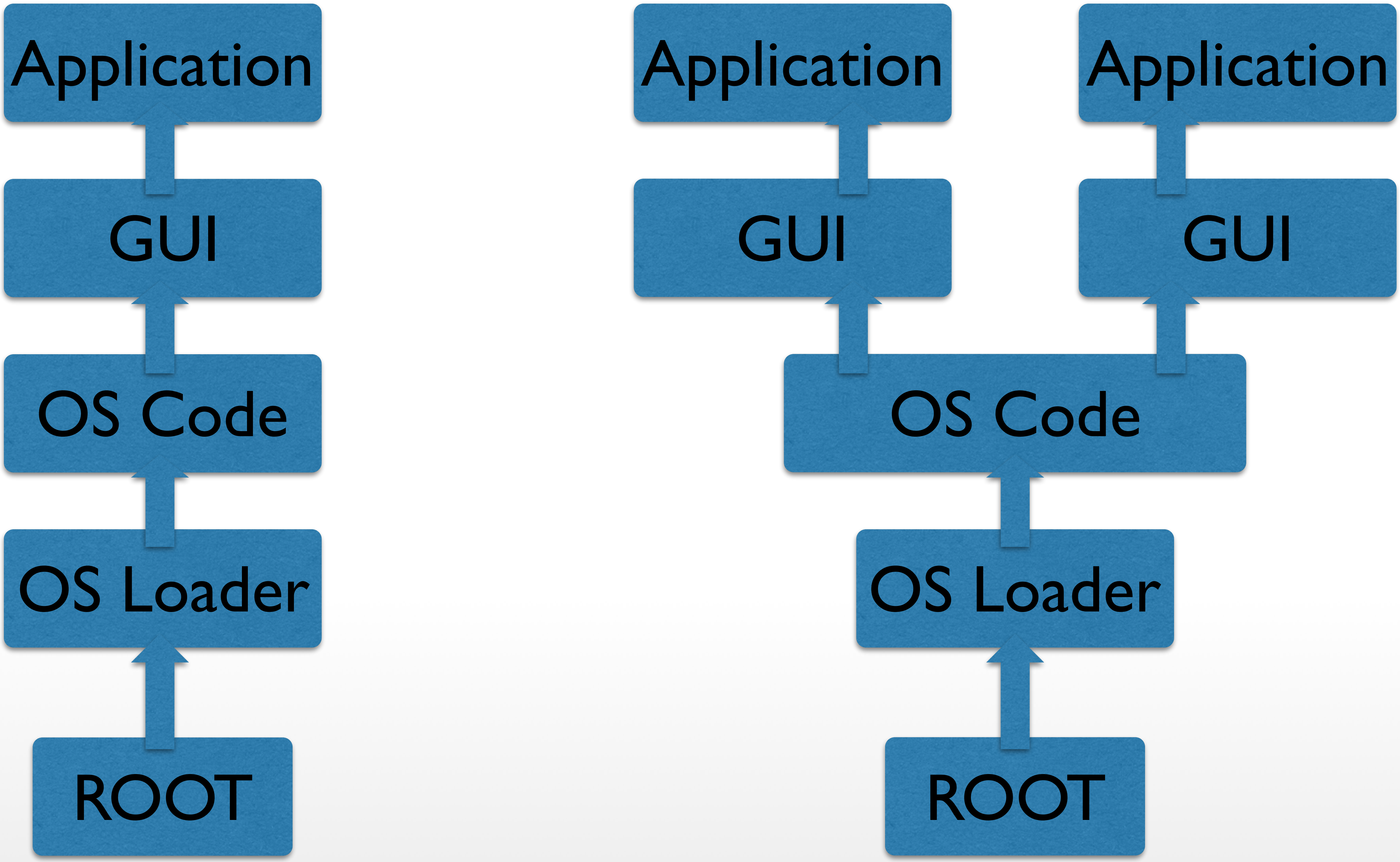
Secure channel:

{ message } ActiveOSConsKpub

# ASSUMPTIONS

- TRB can protect: EK, 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 Problems:

- Very large Trusted Computing Base for Booting (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 (unpublished ?)

Key pairs per step:

- OS controls applications → generate key pair per application
- OS certifies
  - { Application 1, App1Kpub } ActiveOSKpriv
  - { Application 2, App2Kpub } ActiveOSKpriv

# LATE LAUNCH/DYN 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
  - store measurement in PCR

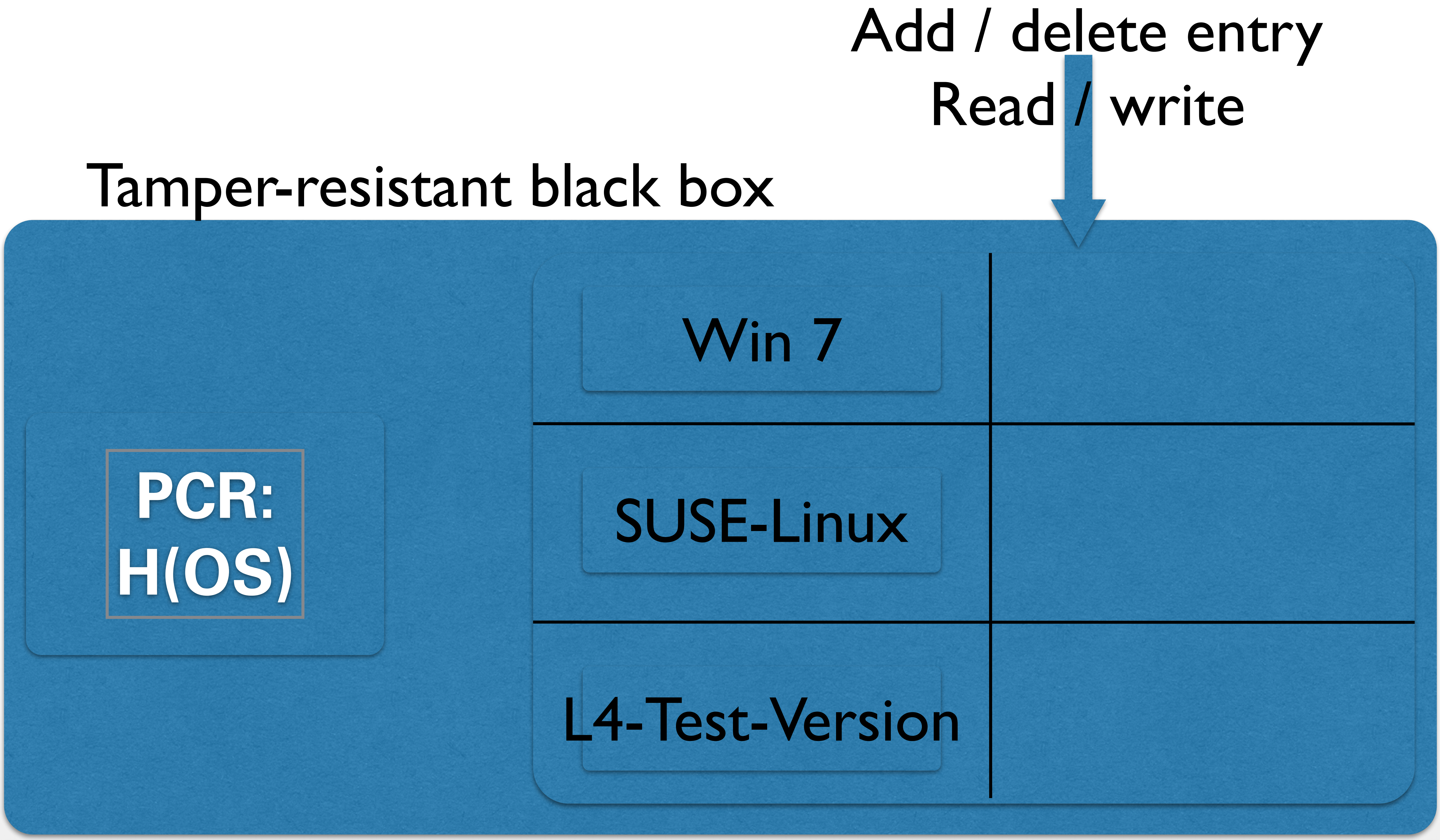
- AMD: “skinit” (Hash) arbitrary root of trust
- Intel: “senter” (must be signed by chip set manufacturer)

# SEALED MEMORY

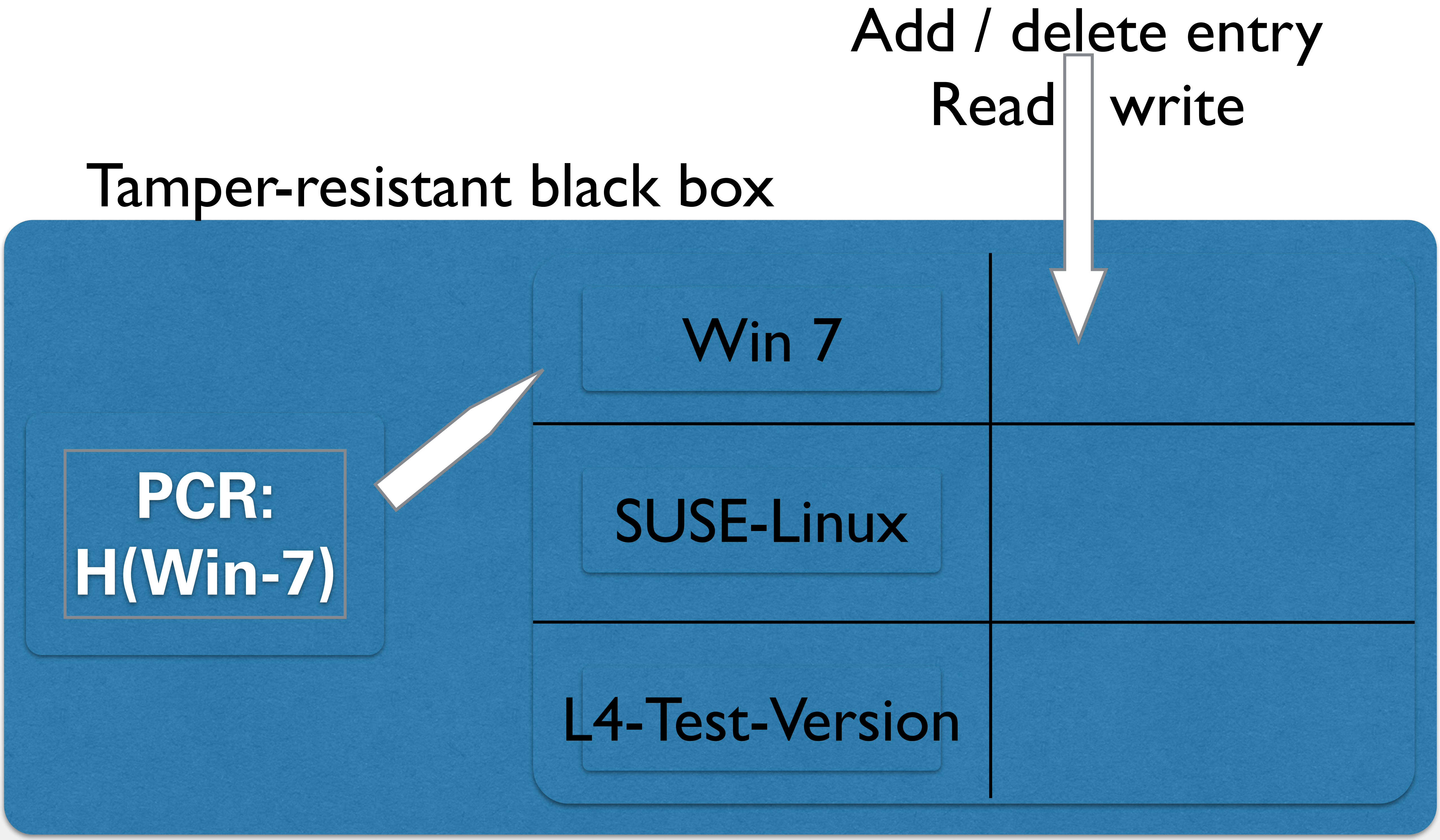
Problem:

- 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

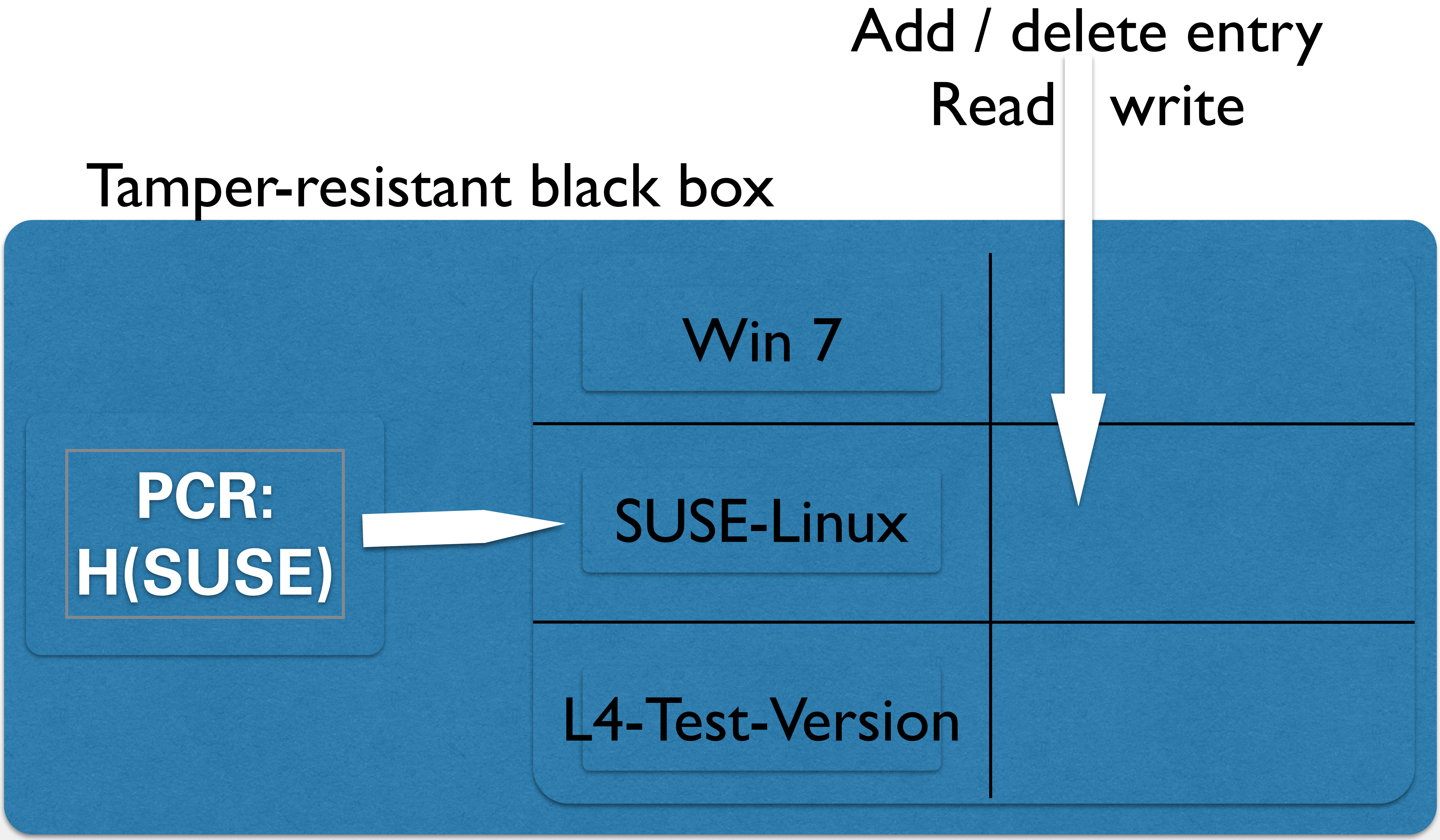
# SEALED MEMORY



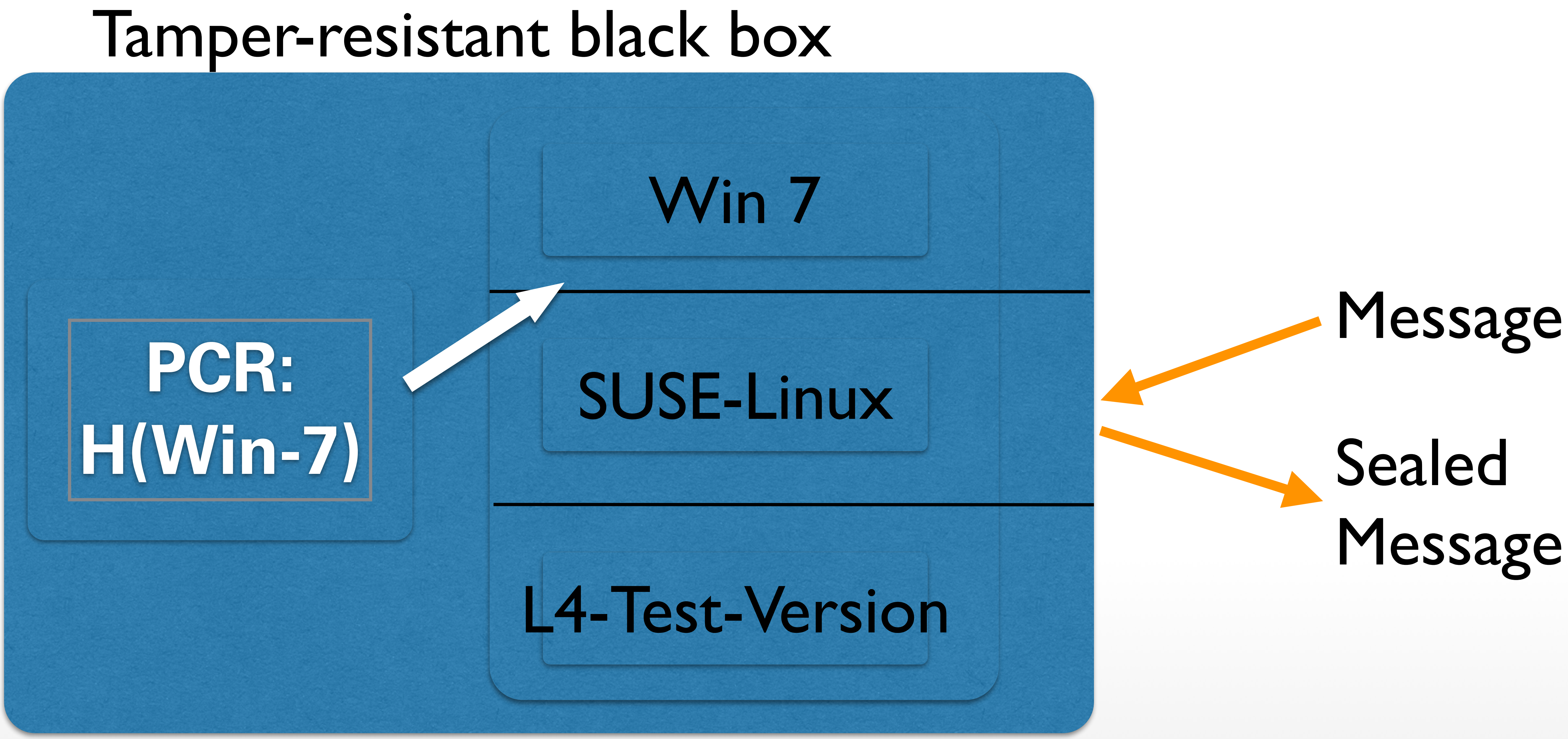
# SEALED MEMORY



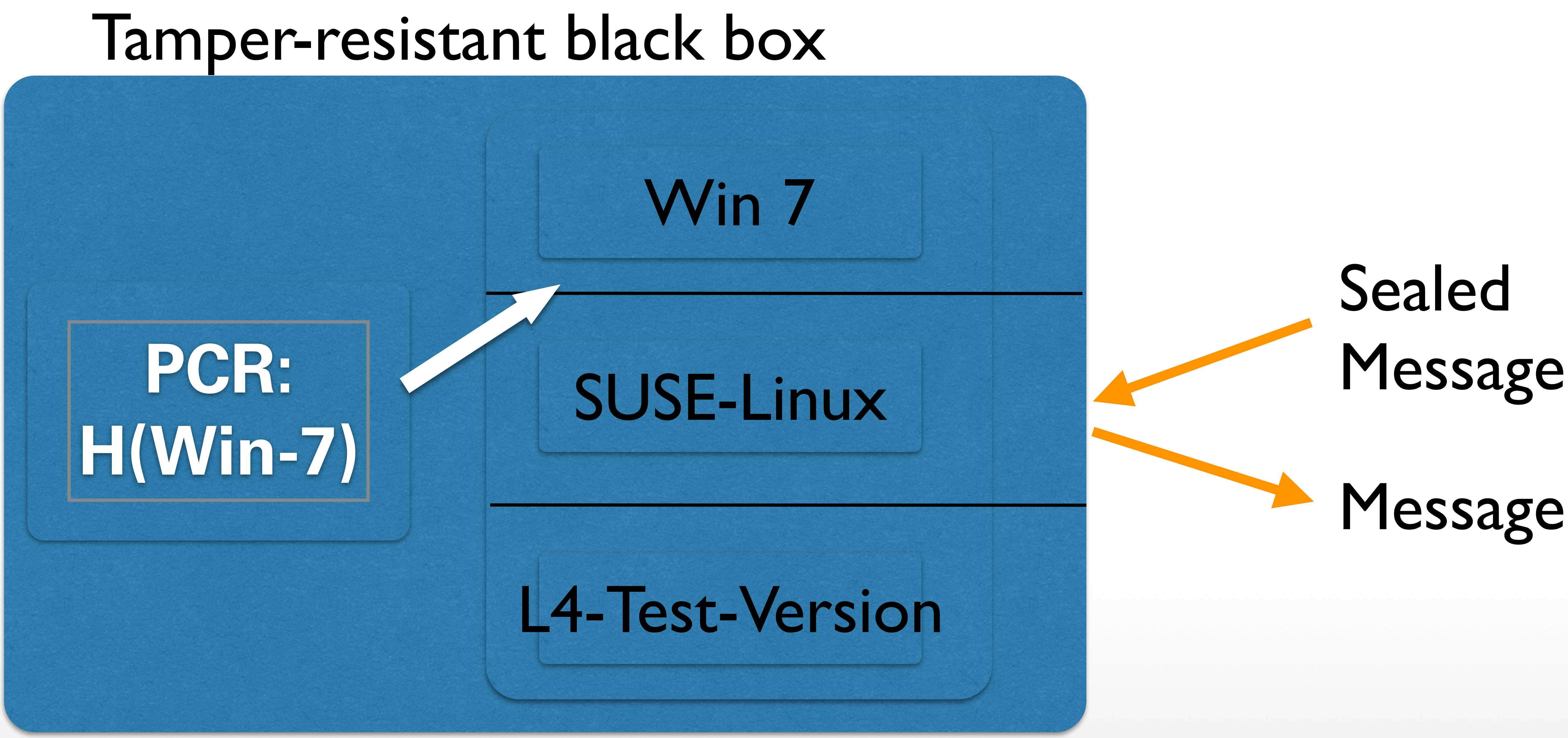
# SEALED MEMORY



# SEALED MEMORY: SEAL OPERATION

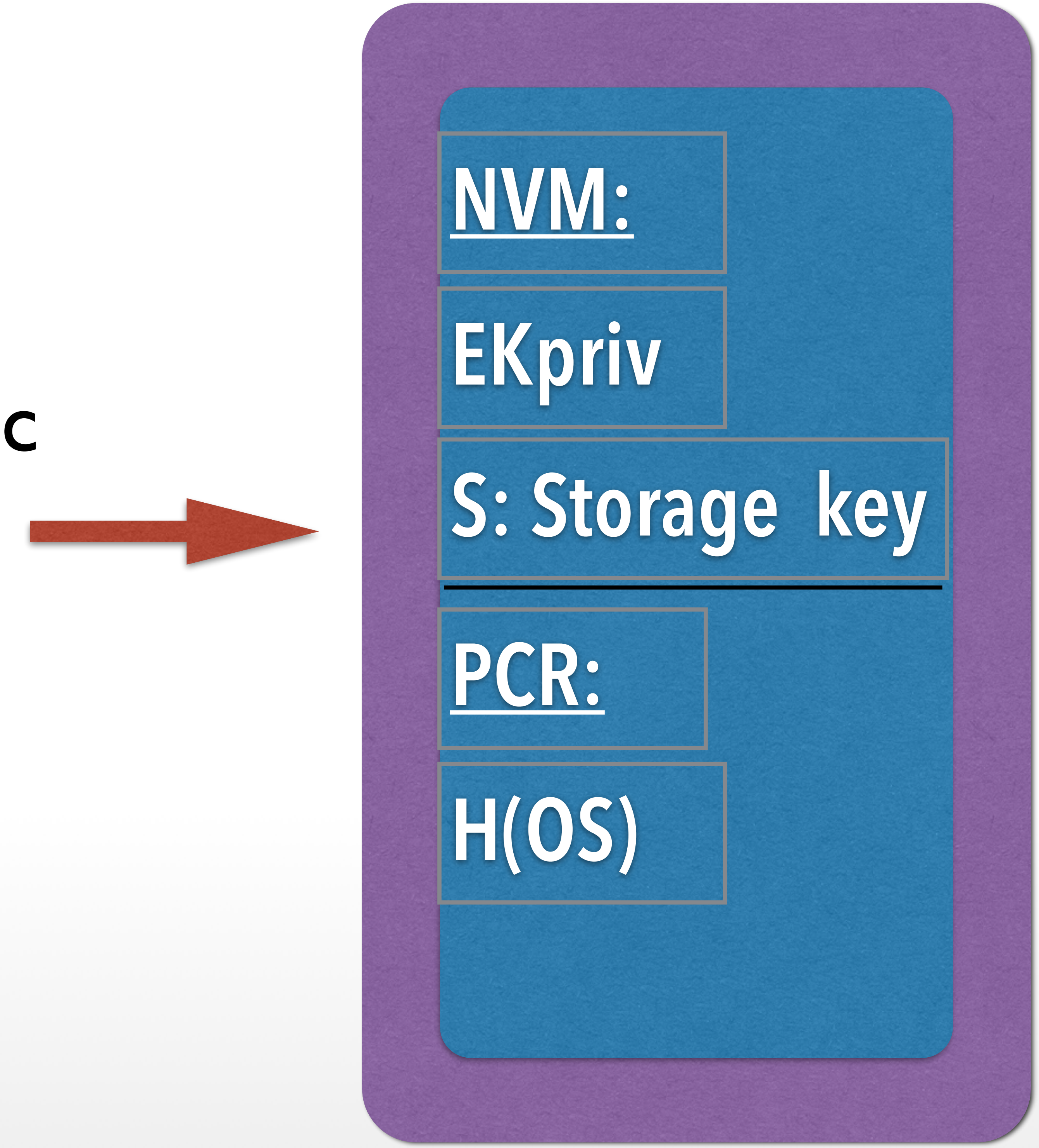


# SEALED MEMORY: UNSEAL OPERATION



# IMPLEMENTATION

TRB generates symmetric Storage Key (S) 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
  → emit SonySecret

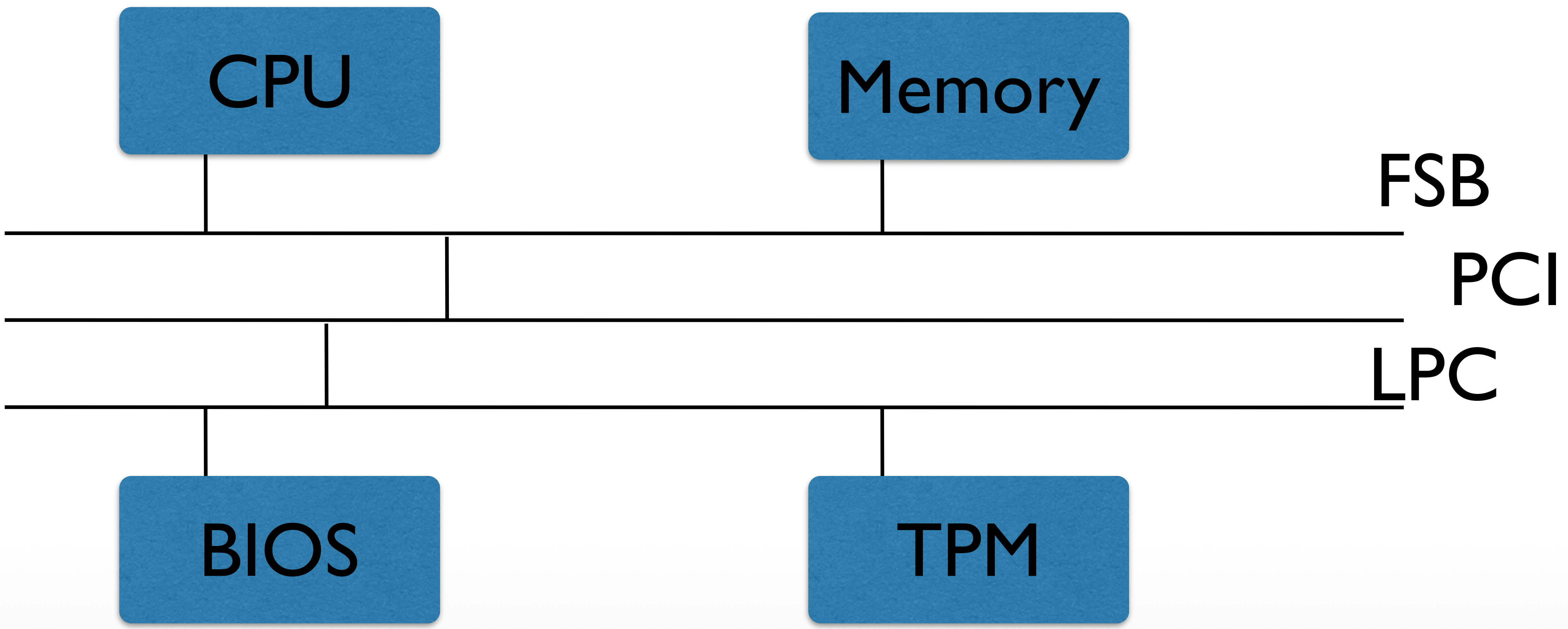
# TAMPER RESISTANT BOX ?

Ideally, includes CPU, Memory, …

Current practice

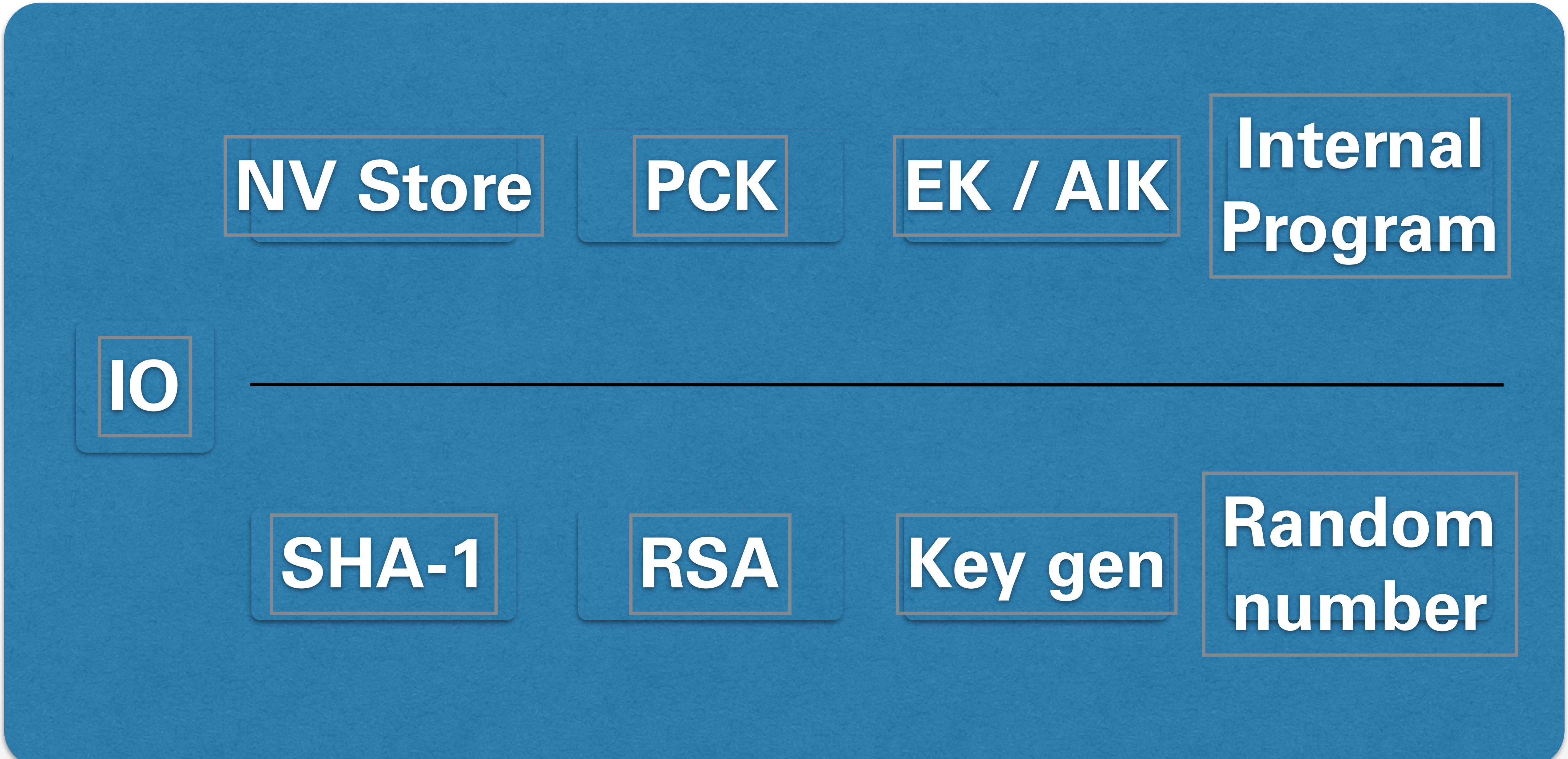
- Additional physical protection, for example IBM 4758 …
  look it up in Wikipedia
- HW versions
  - TPM:
    separate "Trusted Platform Modules"
    (replacing BIOS breaks TRB)
  - Add a new privilege mode: ARM TrustZone
  - raise to user: Intel SGX

# TCG PC PLATFORMS: "TRUSTED PLATFORM MODULE" (TPM)

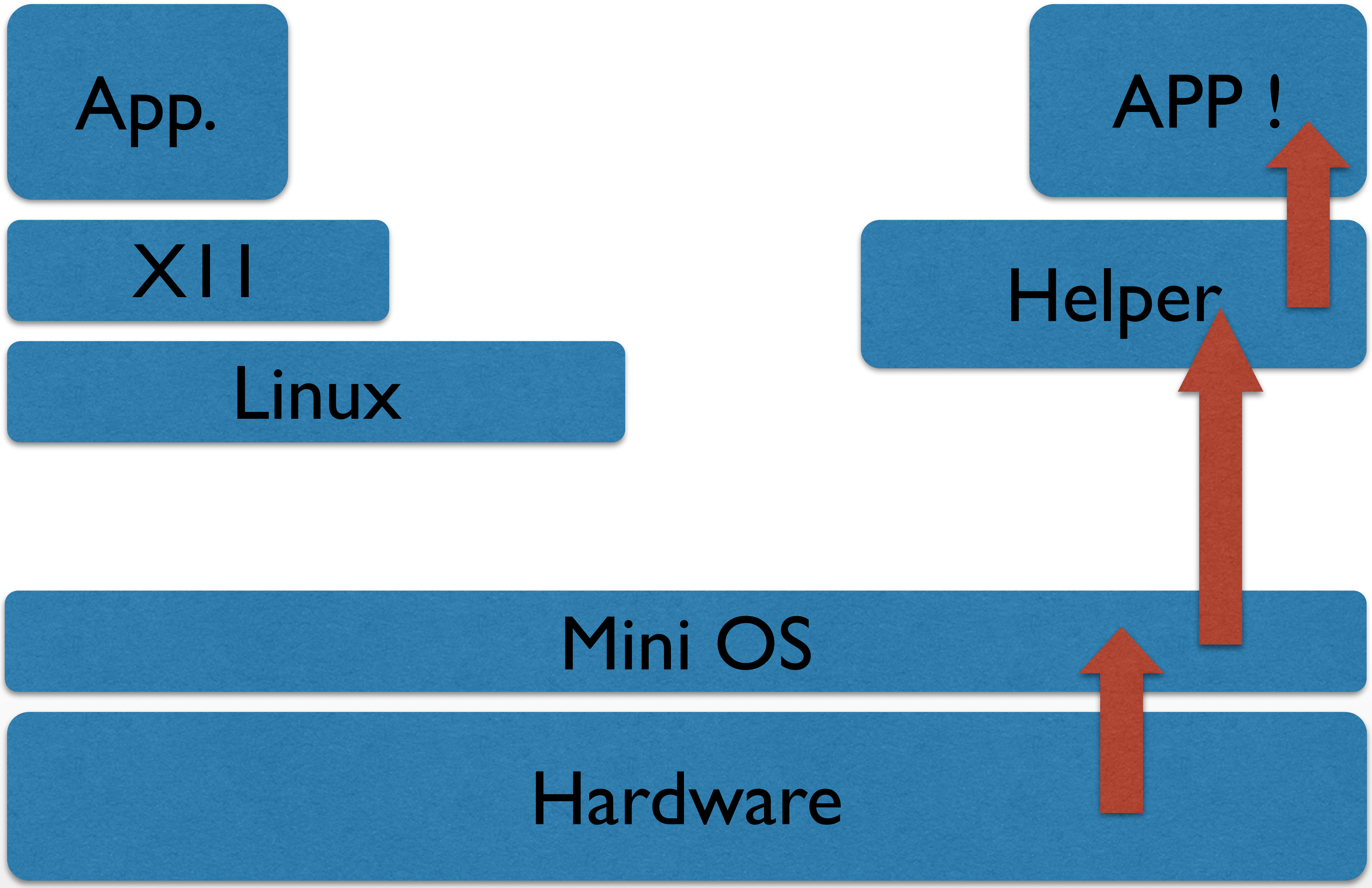


TECHNISCHE UNIVERSITÄT DRESDEN

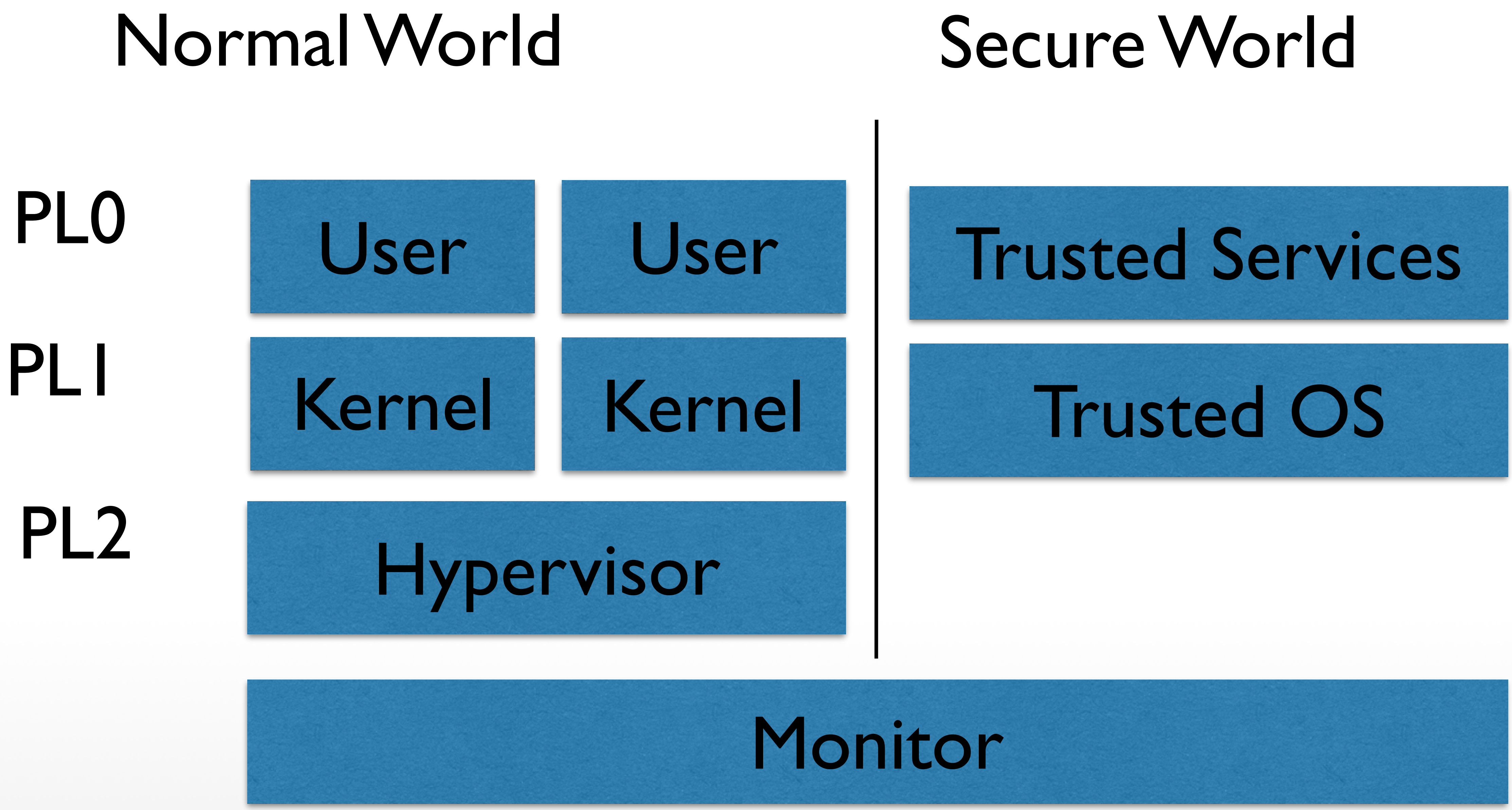
# TPM



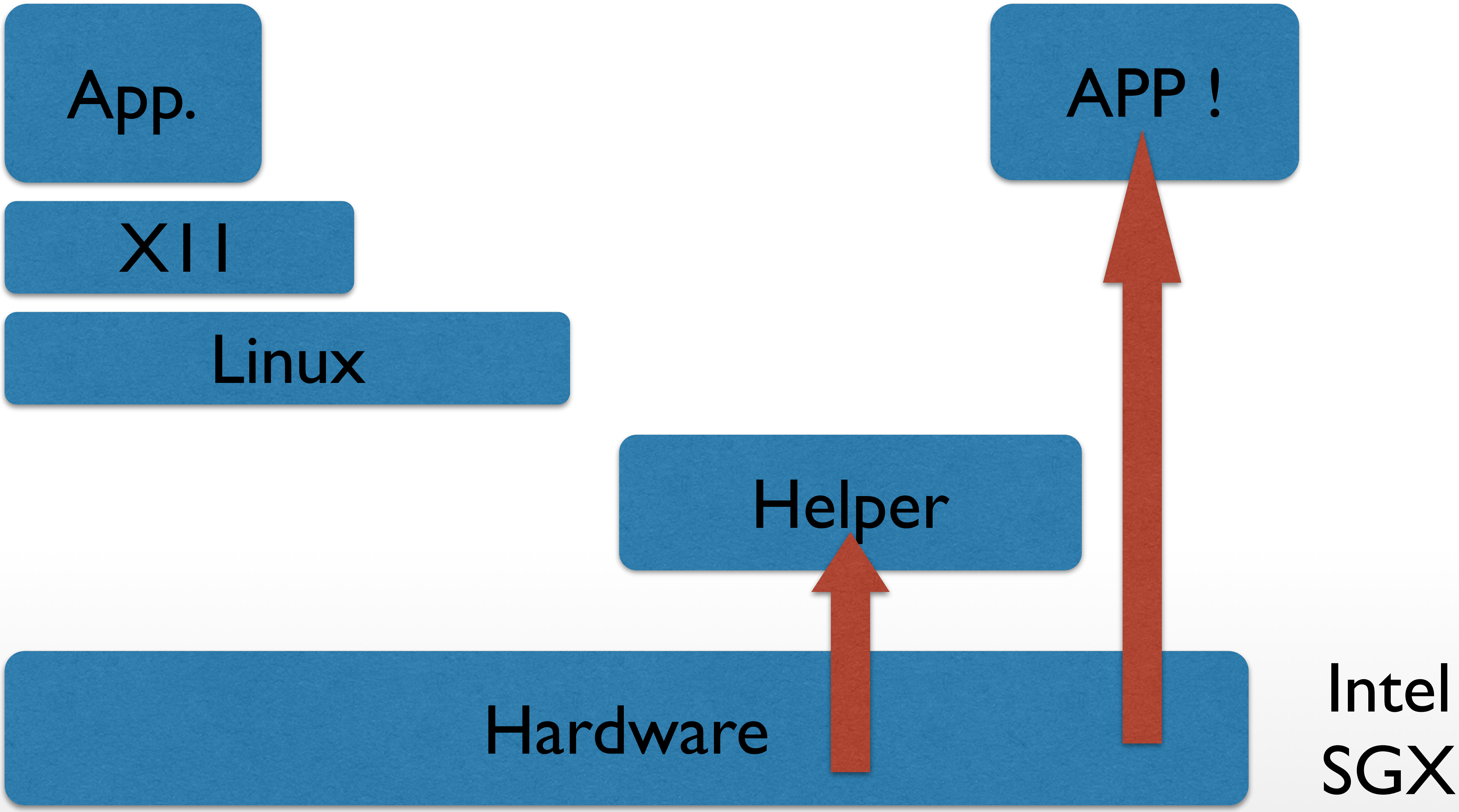
# SMALL TRUSTED COMPUTING BASE



# ARM TRUSTZONE



# SMALL TRUSTED COMPUTING BASE



# INTEL SGX

CPU

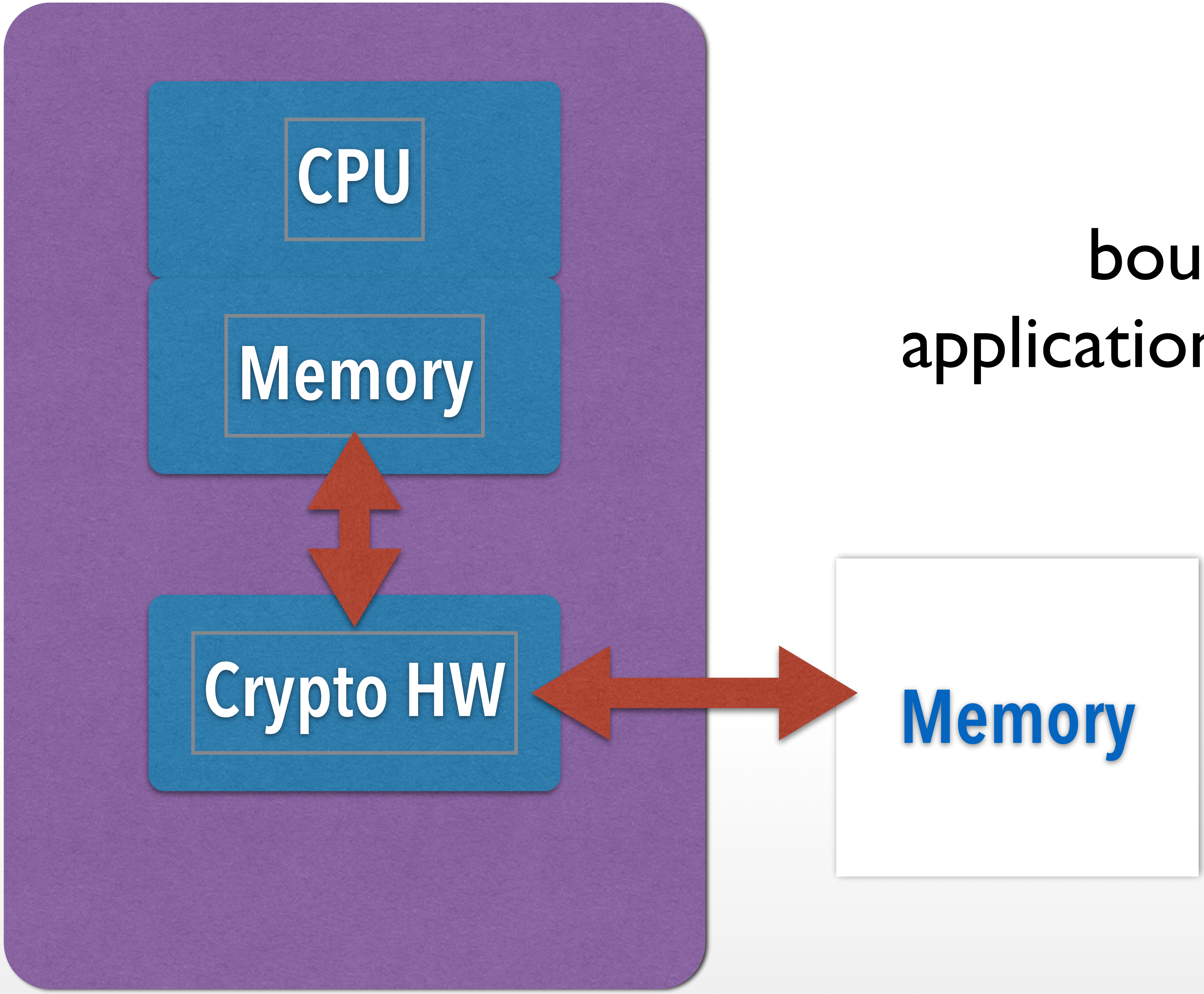
Memory

Non-Volatile Memory (NVM):

Platform Configuration Regs (PCR):

TRB Conceptional View

# INTEL SGX




bound to
application "enclaves"

# SGX ENCLAVES

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

# 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
  many sources