TECHNISCHE UNIVERSITÄT DRESDEN

"TRUSTED" COMPUTING

DISTRIBUTED OPERATING SYSTEMS

Hermann Härtig

Summer 2017

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 occasionally changes

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

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

PROTECTION OF APPLICATION

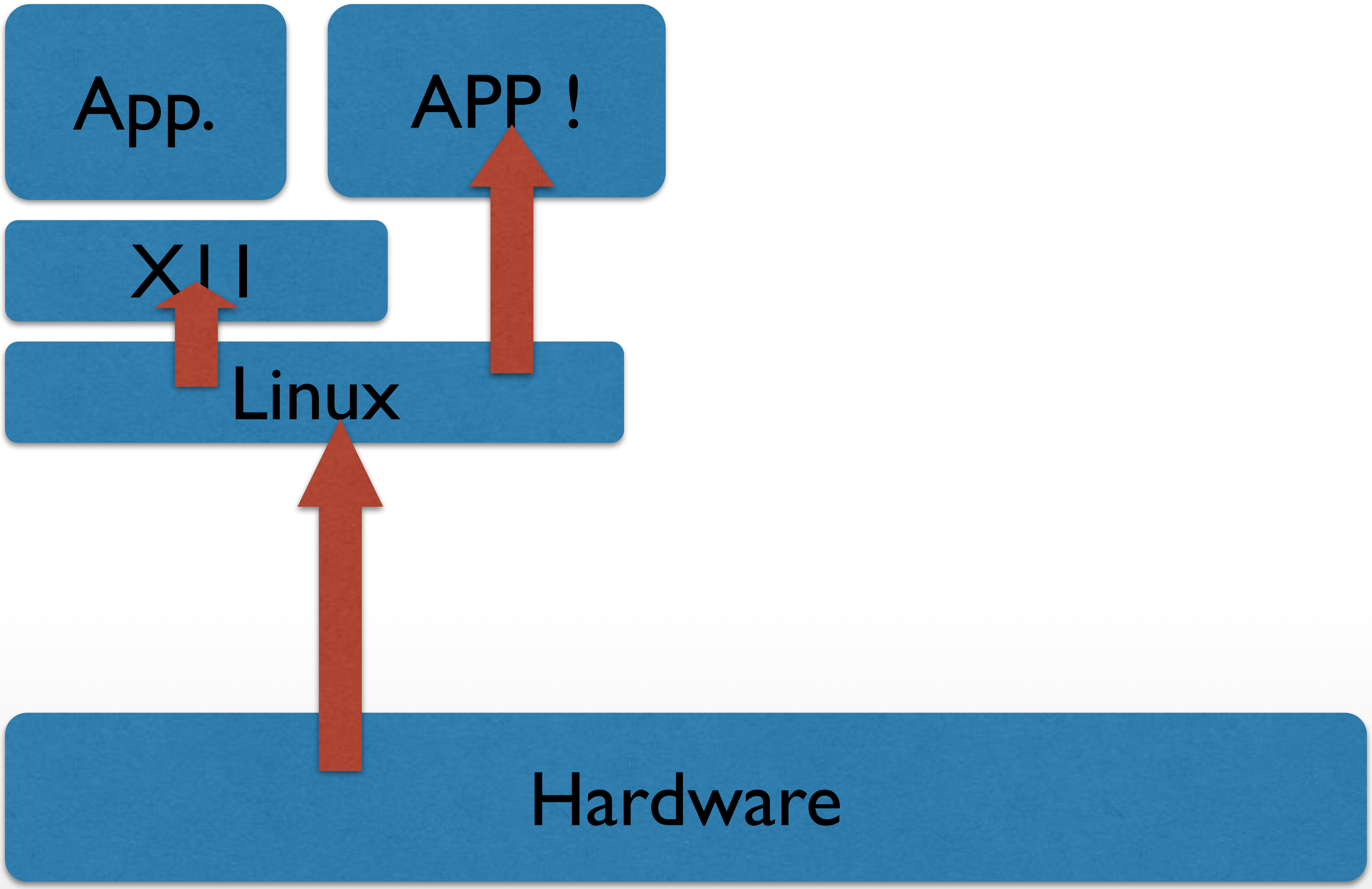
Principle Method:

separate critical Software

rely on small Trusted Computing Base

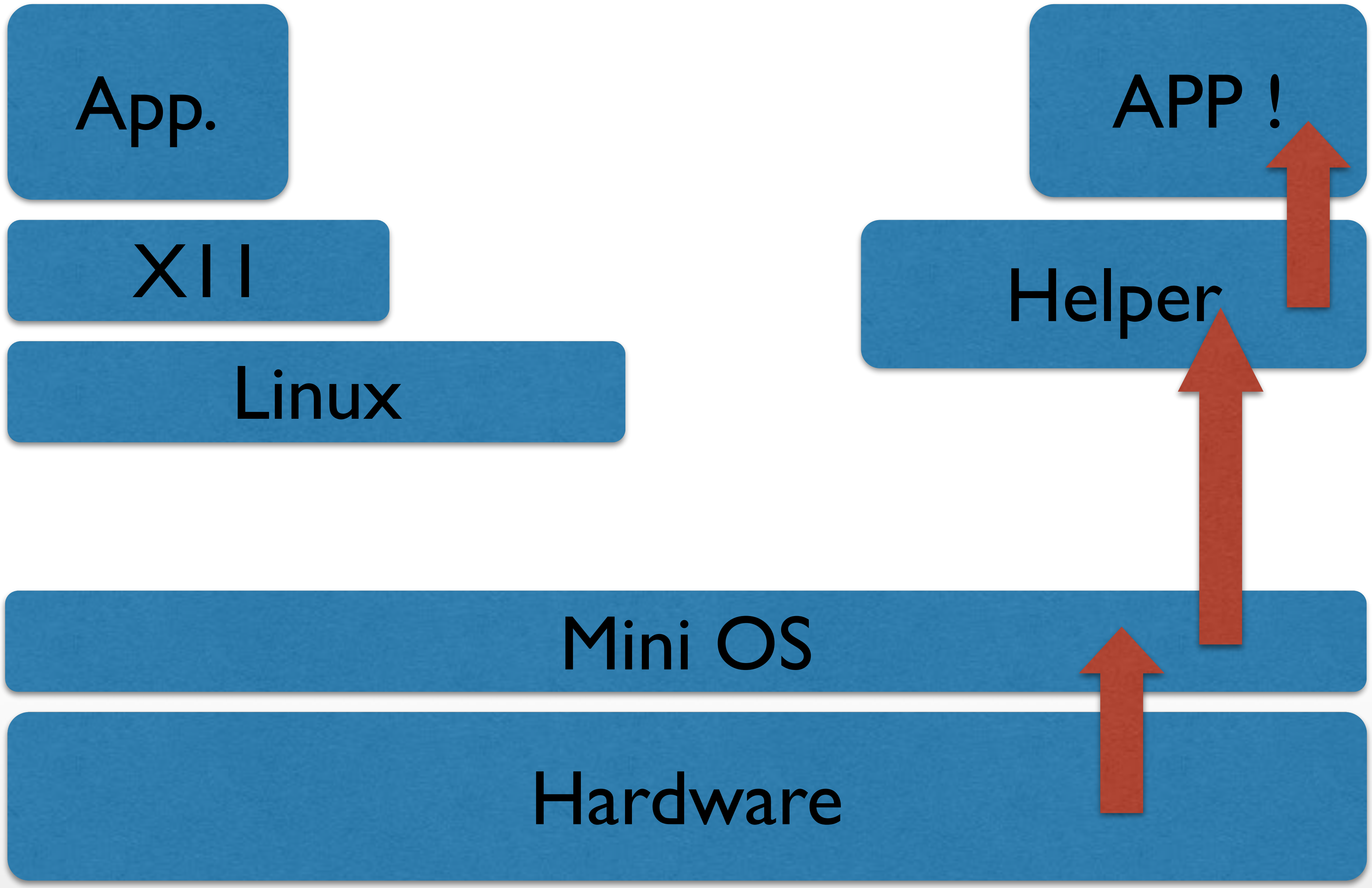
- Small OS kernels
  micro kernels, separation kernels, ….
- Hardware/Microcode

SMALL TRUSTED COMPUTING BASE

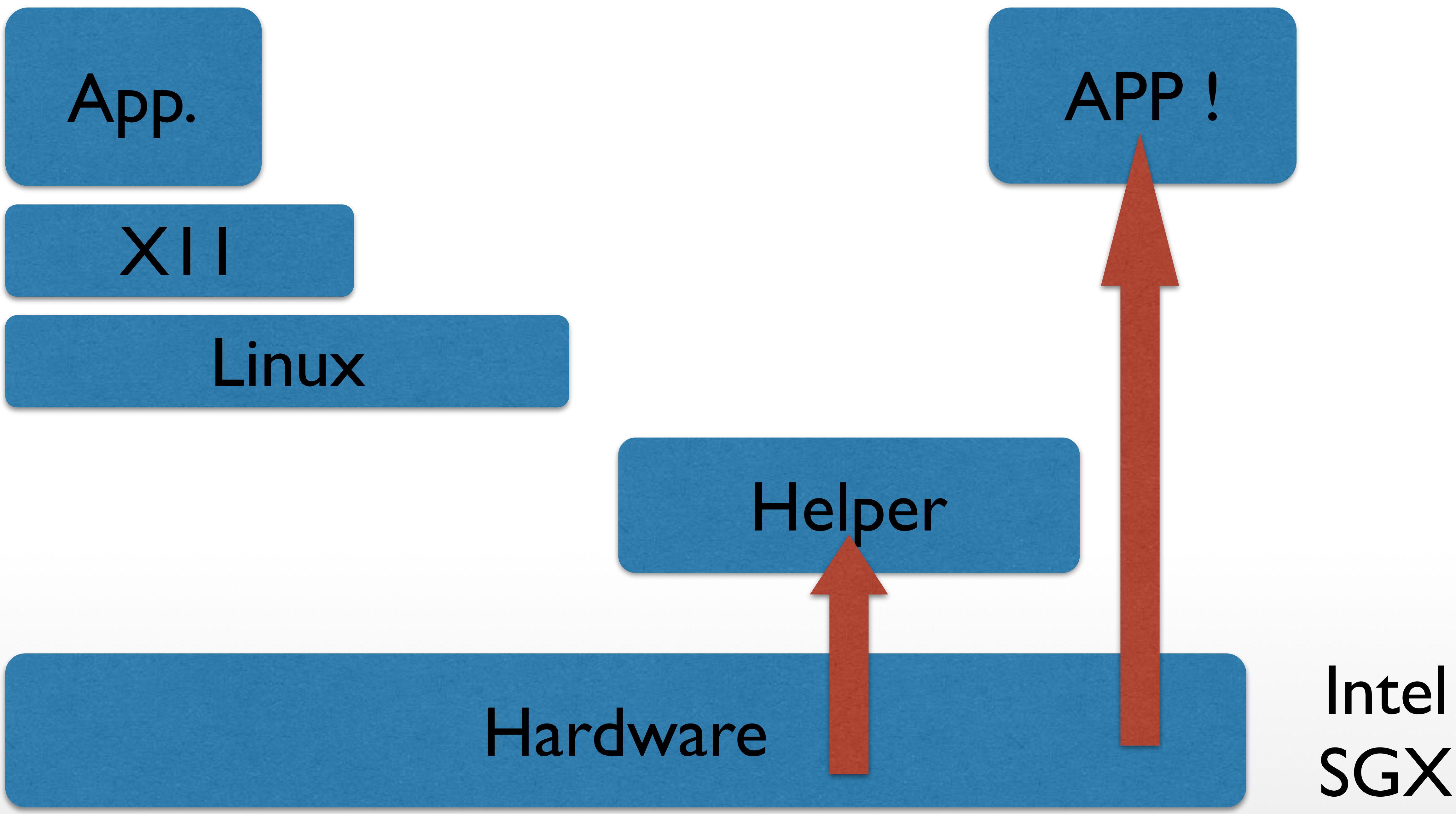


TECHNISCHE UNIVERSITÄT DRESDEN

SMALL TRUSTED COMPUTING BASE



SMALL TRUSTED COMPUTING BASE



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

S^{pair}:	S^{priv} S^{pub}	Asymmetric key pair of entity S
S^{symm}		Symmetric Key

- "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}))

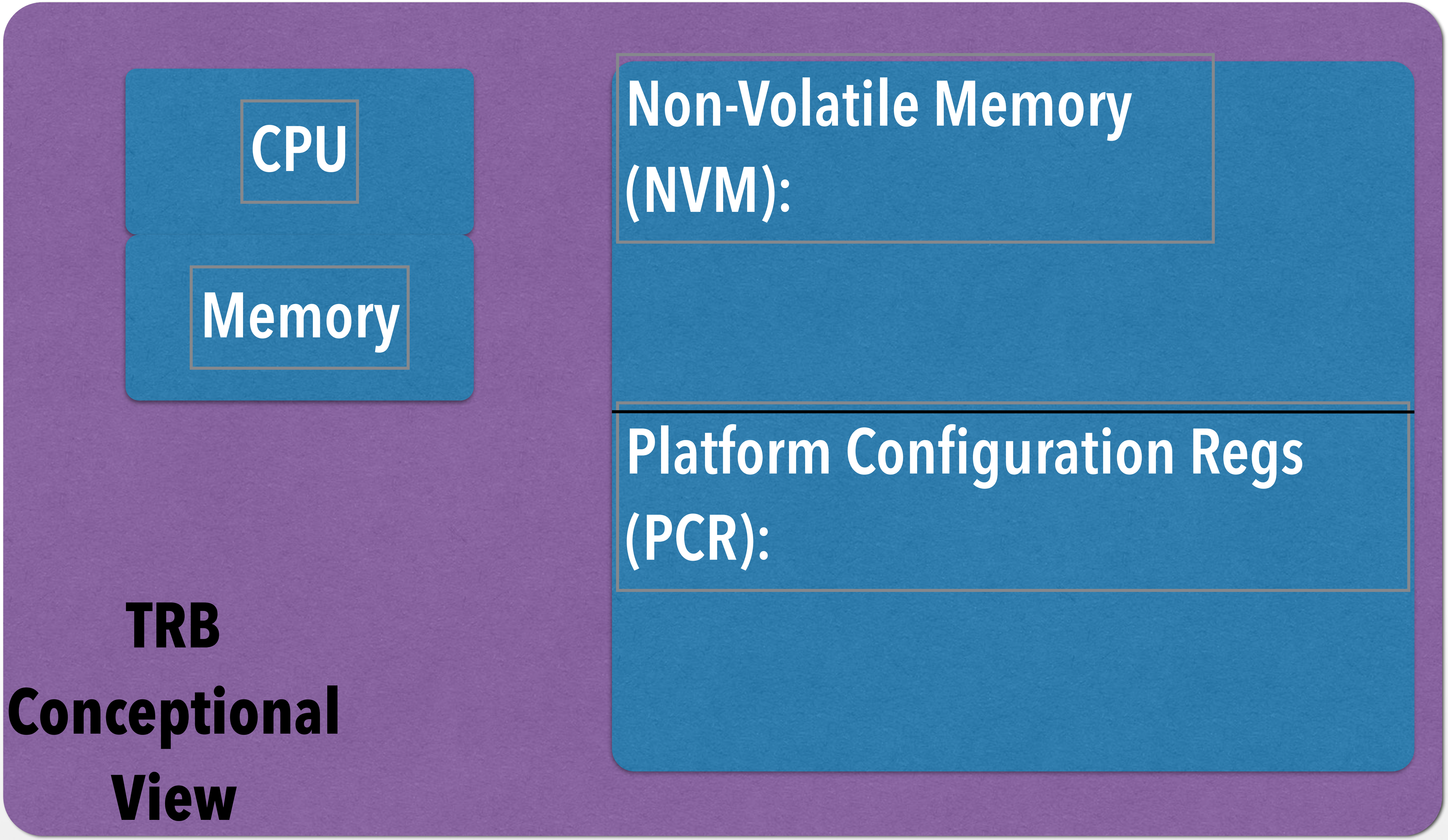
- "Concealed Message": { M } S^{pub}
  Message concealed for S
  S^{pub} is needed to unconceal M

NOTATION

- "Digital Signature": { M } S^{priv}
  S^{pub} is used to verify that S has signed M
  is short for: M, encrypt(H(M), S^{priv})

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

TAMPERRESISTANT BLACK BOX(TRB)



WAYS TO "BURN IN" THE OS OR SECURE BOOTING

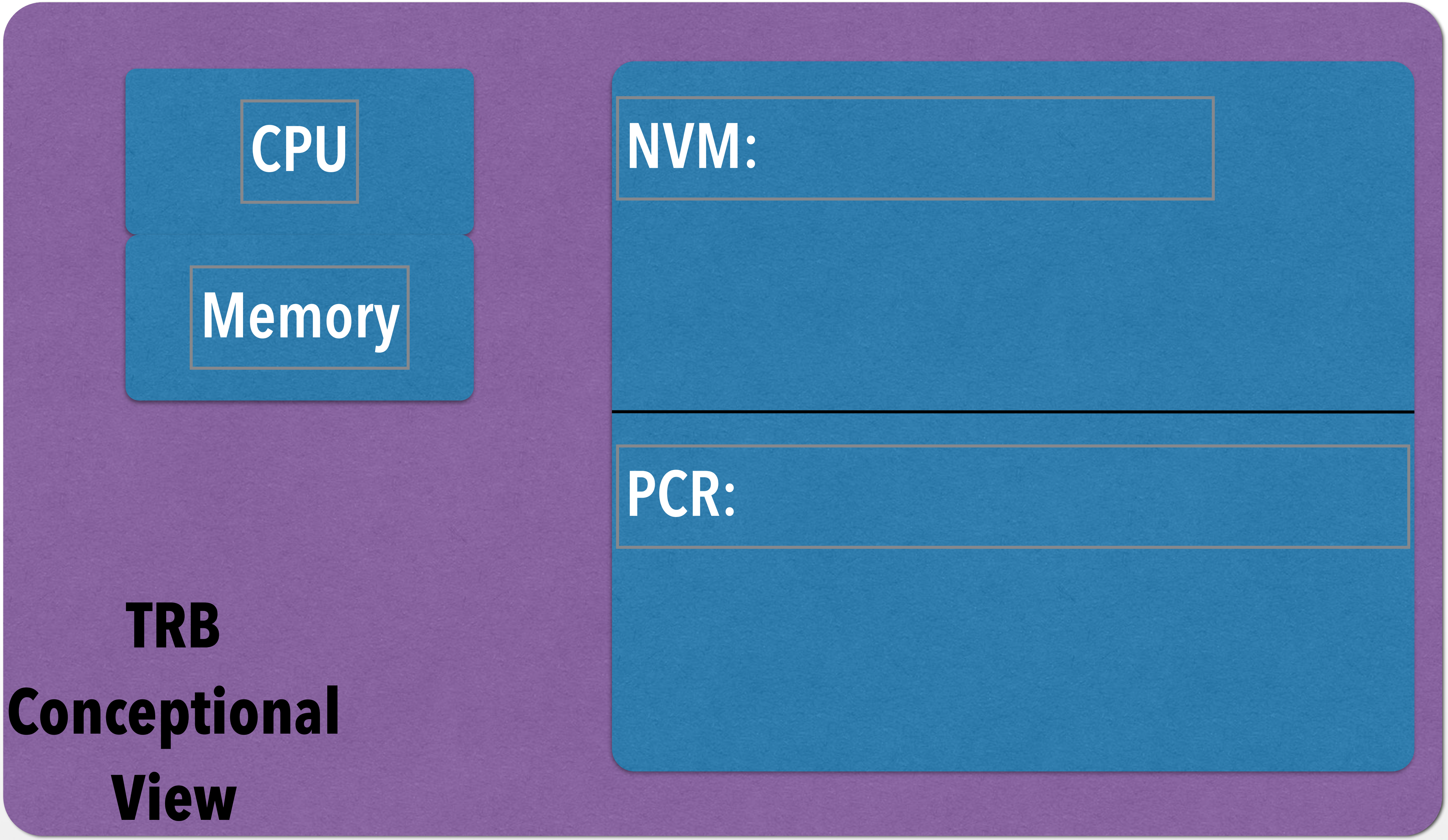
- Read-Only Memory
- H(OS) in NVM preset by manufacturer
  - load OS- Code
  - compare H(loaded OS code) to preset H(OS)
  - abort if different
- FSKpub in NVM preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using FSKpub
  - abort if check fails

AUTHENTICATED BOOTING, USING HASH

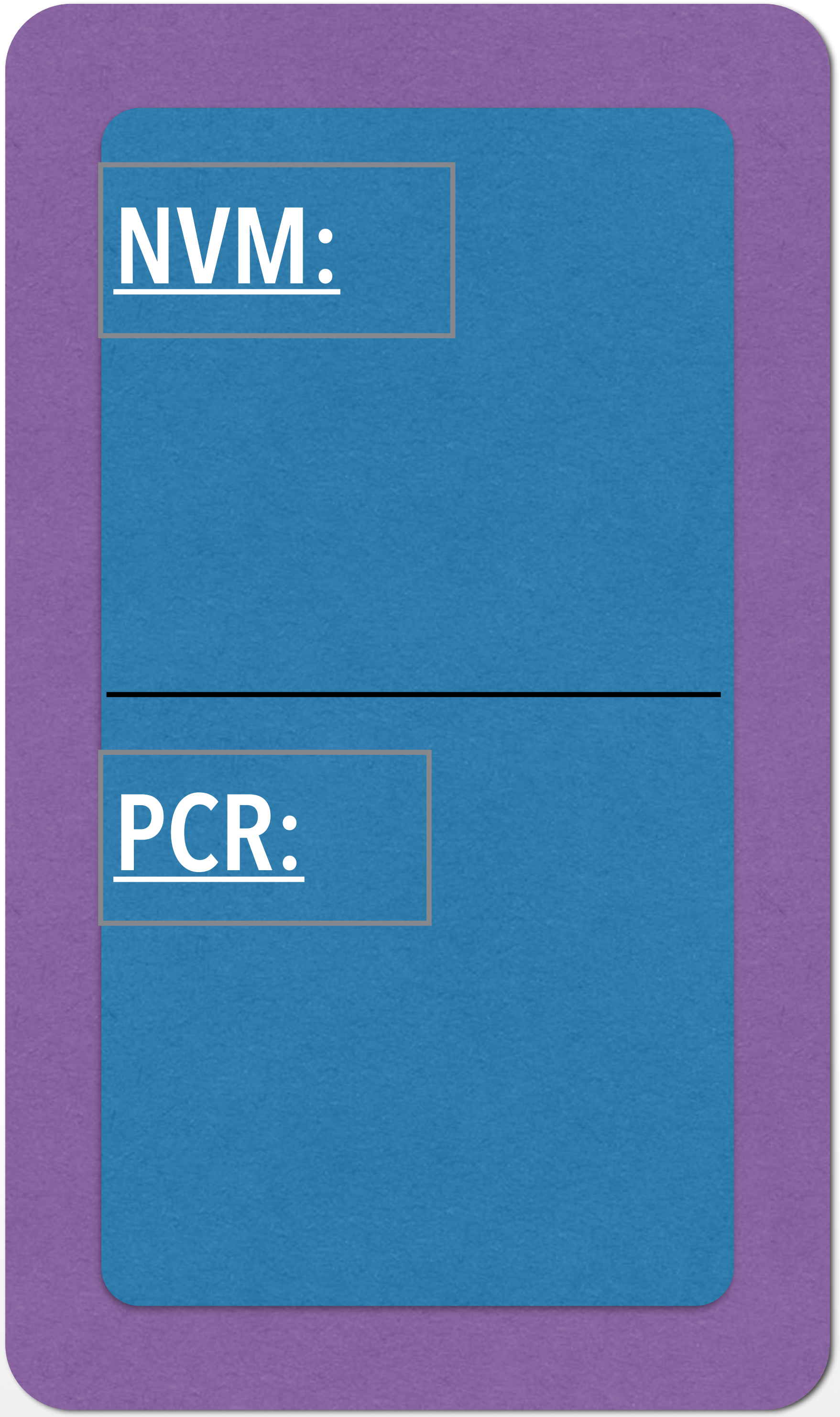
Steps:

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

TAMPERRESISTANT BLACK BOX(TRB)

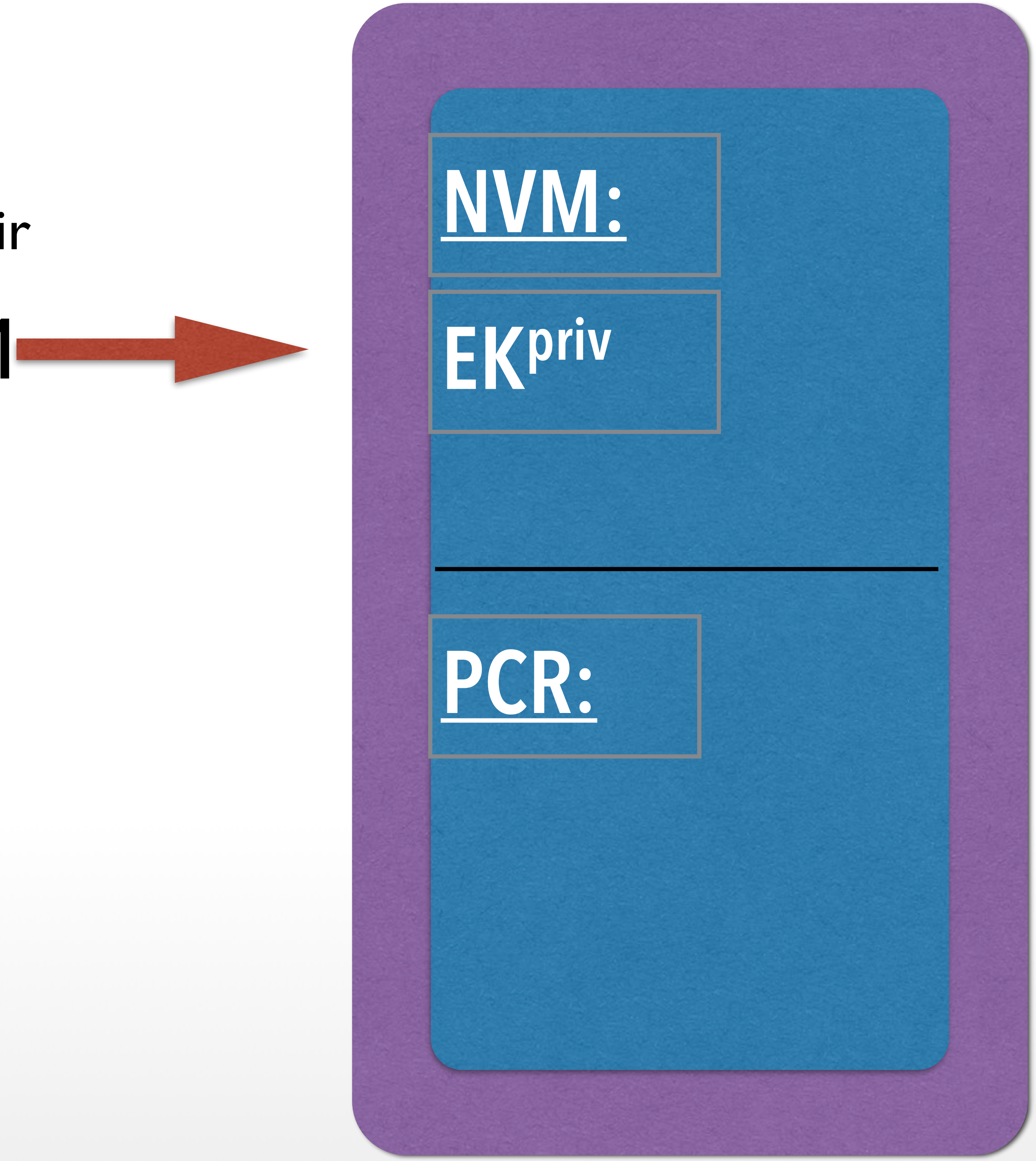


TAMPERRESISTANT BLACK BOX(TRB)



TRB VENDOR

TRB generates key pair:
„Endorsement Key“ EK^{pair}
stores EK^{priv} in TRB NVM
emits EK^{pub}



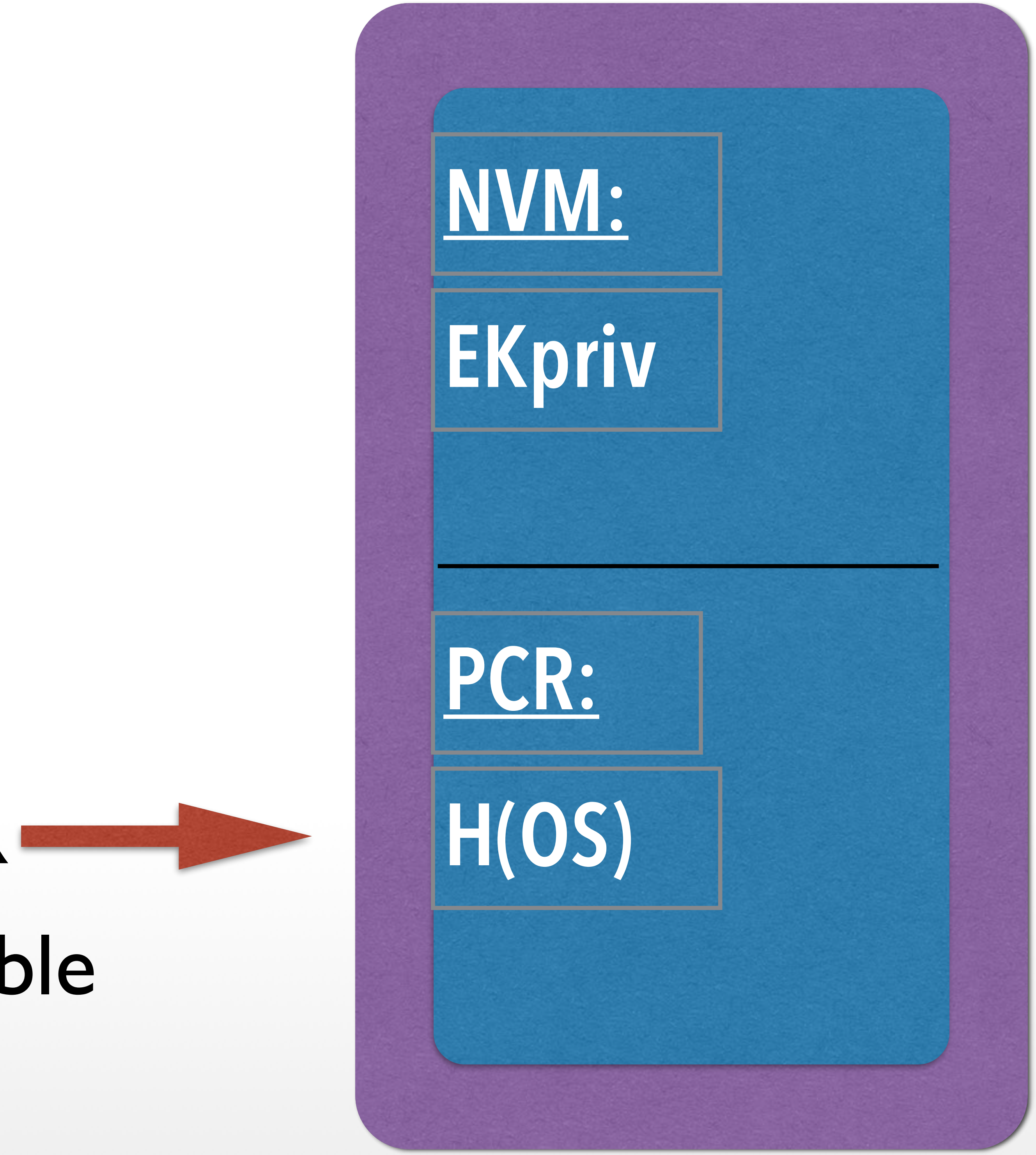
TRB AND OS VENDOR

- 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



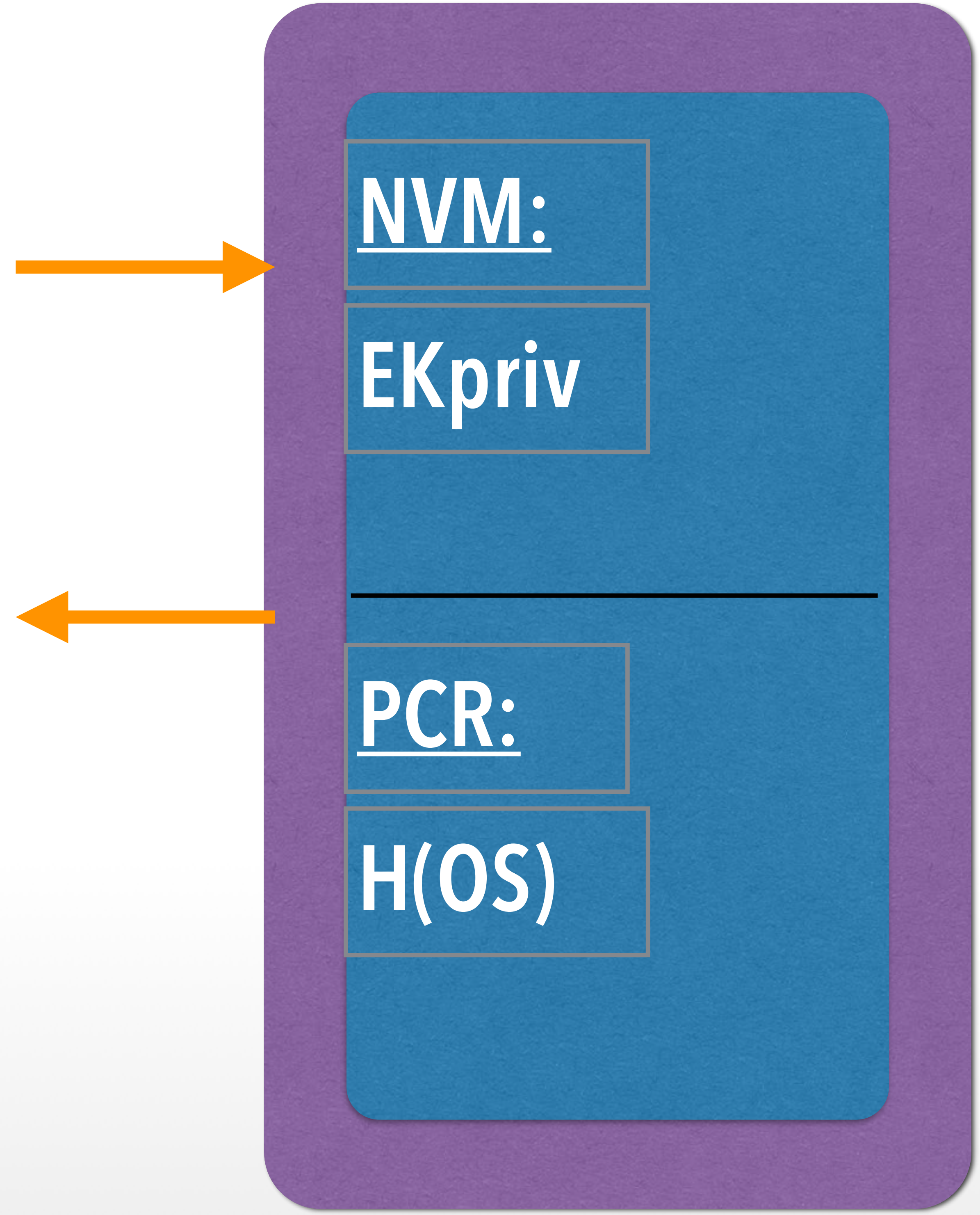
PCR not (directly) writable by OS

more later

ATTESTATION (FIRST BASIC EXPLANATION)

Challenge:
send NONCE

Response:
{NONCE', PCR}EK^{priv}



PROBLEM

- boot Linux

  → challenge

  ← response "Linux"

- reboot Windows

  → send data

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 :
  - $ActiveOSAuth^{pair}$ /* 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'} $ActiveOSAuth^{priv}$

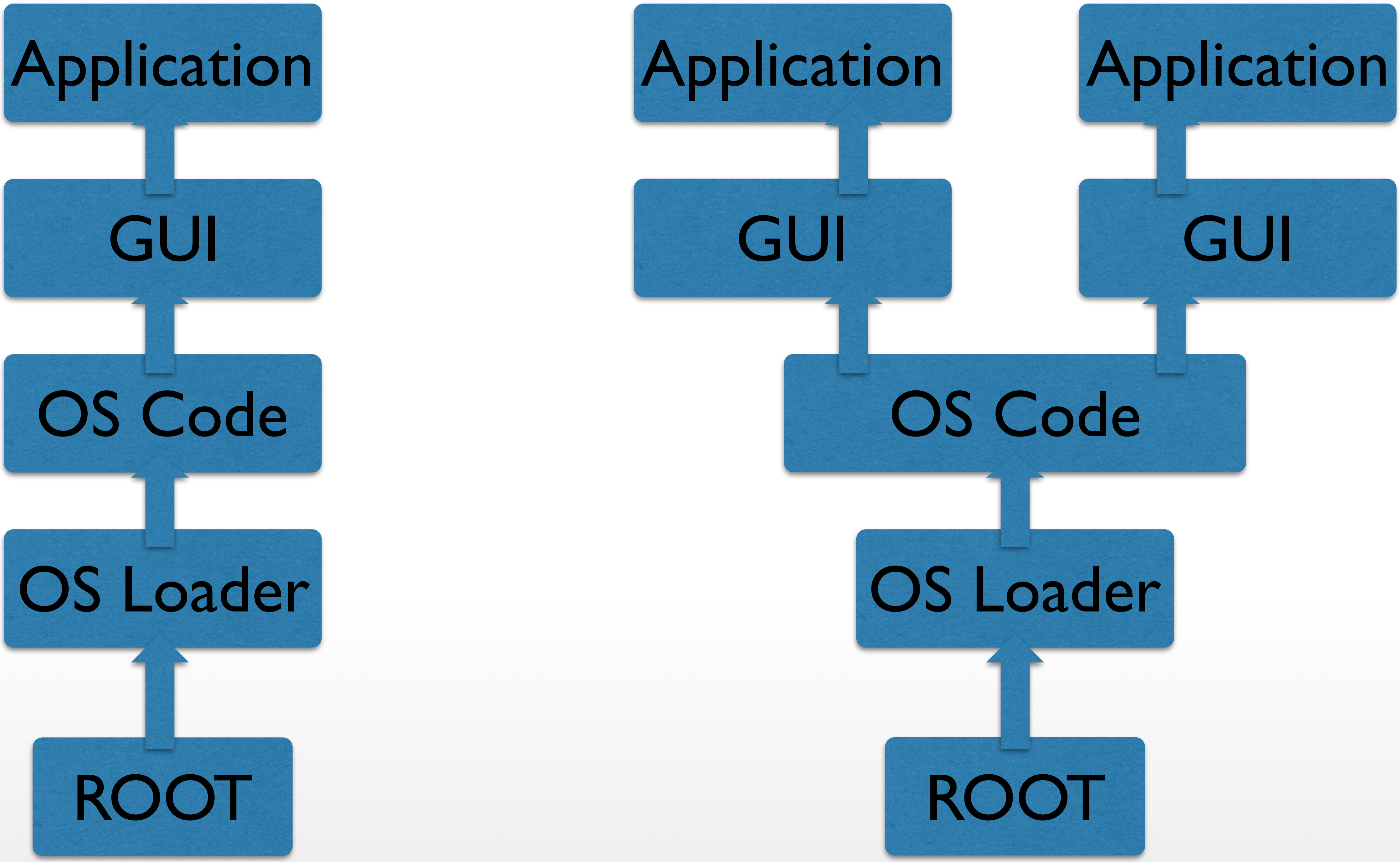
Secure channel:

{ 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 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 } $ActiveOS^{priv}$
  - { Application 2, App2Kpub } $ActiveOS^{priv}$

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

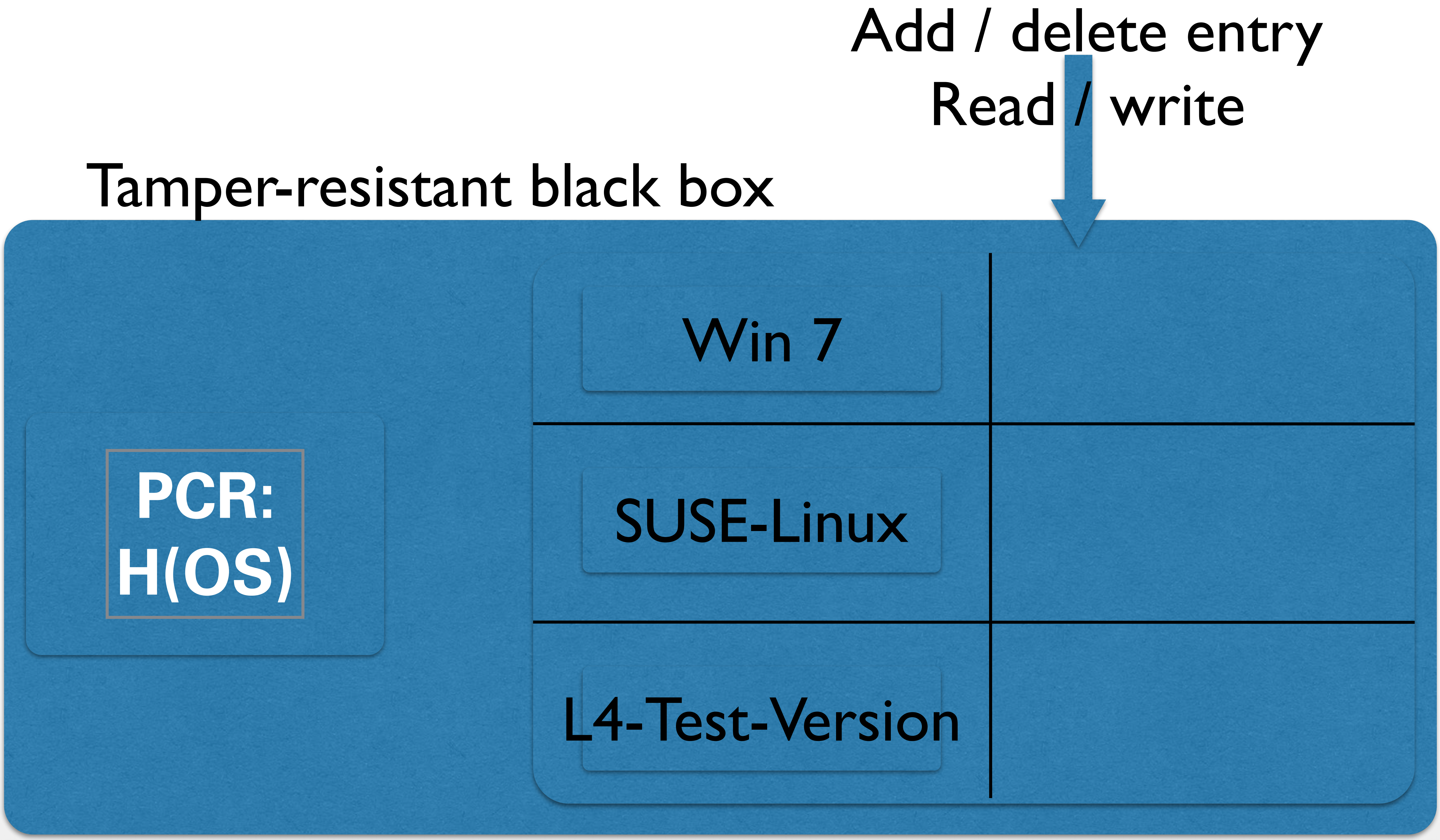
- AMD: "skinit" (Hash) arbitrary root of trust
- Intel: "senter" (must be signed by chip set manufacturer)

SEALED MEMORY

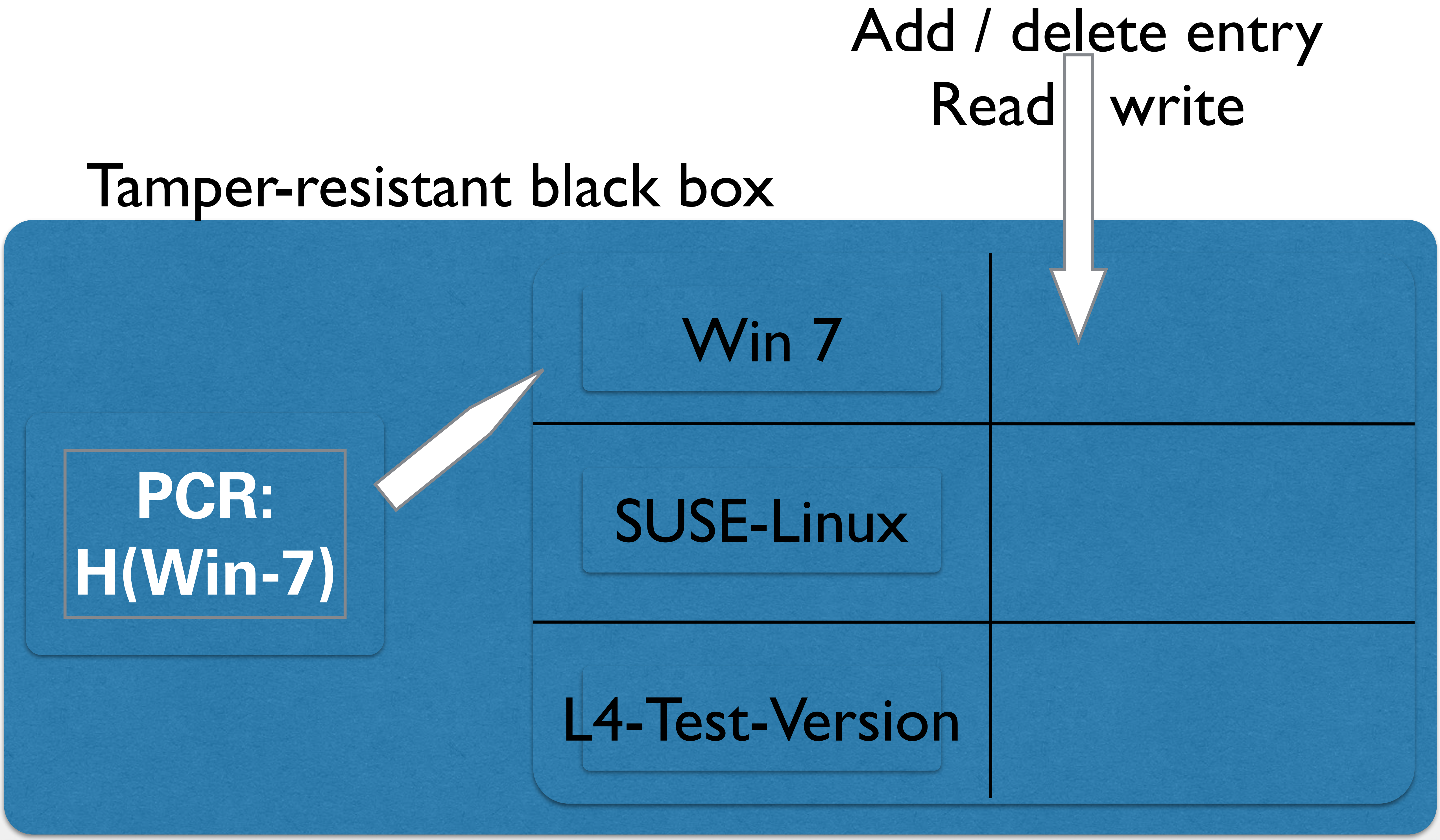
Problem:

- 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

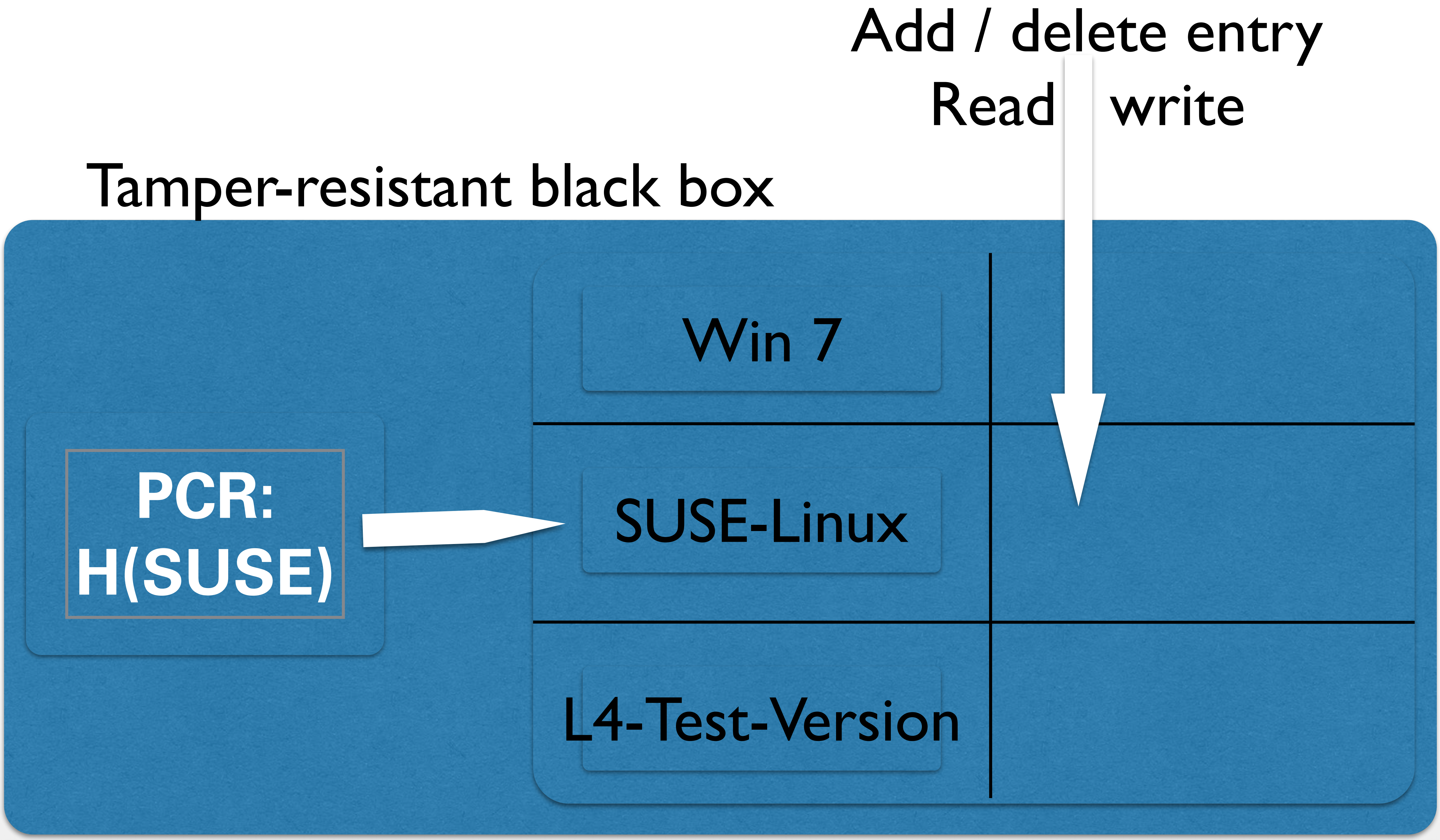
SEALED MEMORY



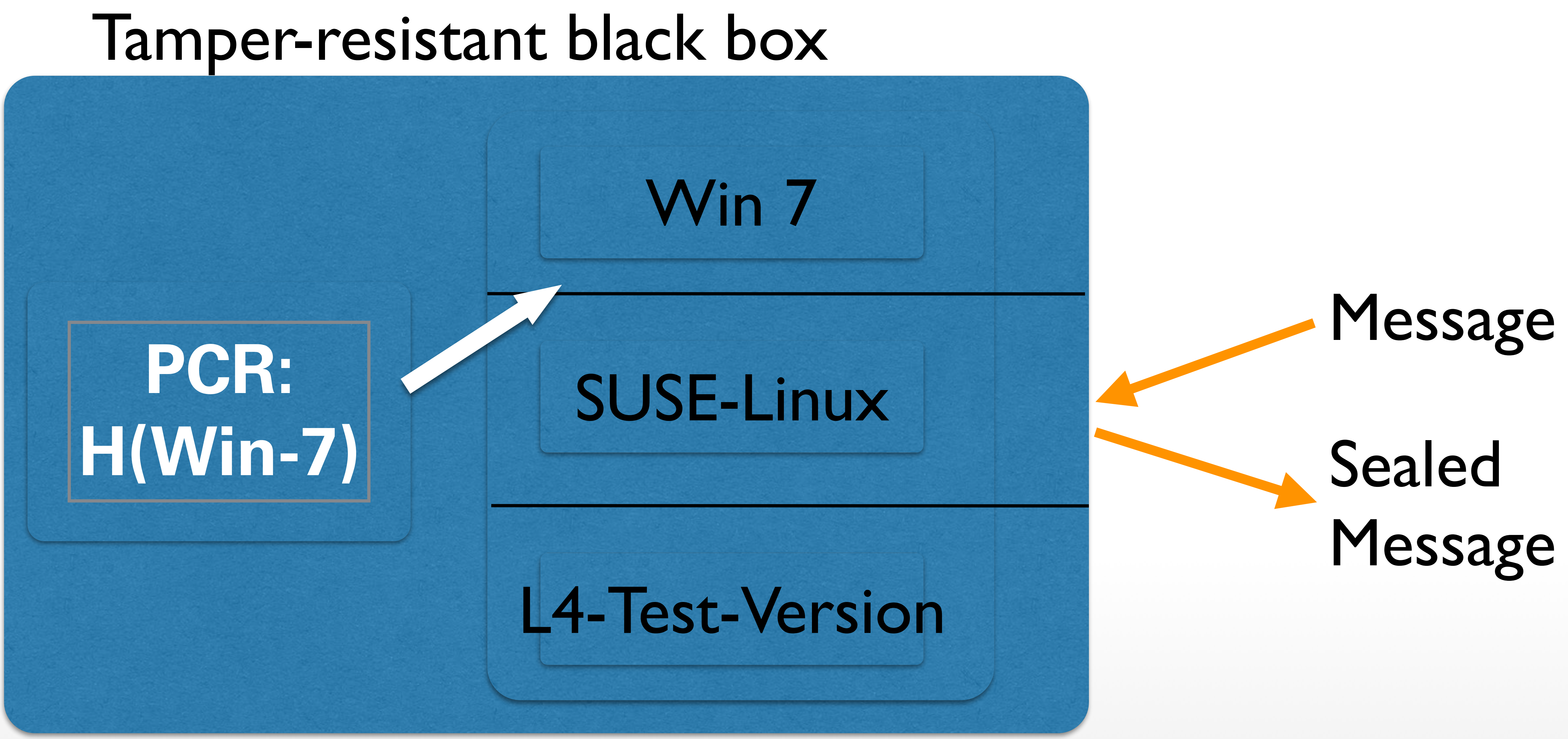
SEALED MEMORY



SEALED MEMORY

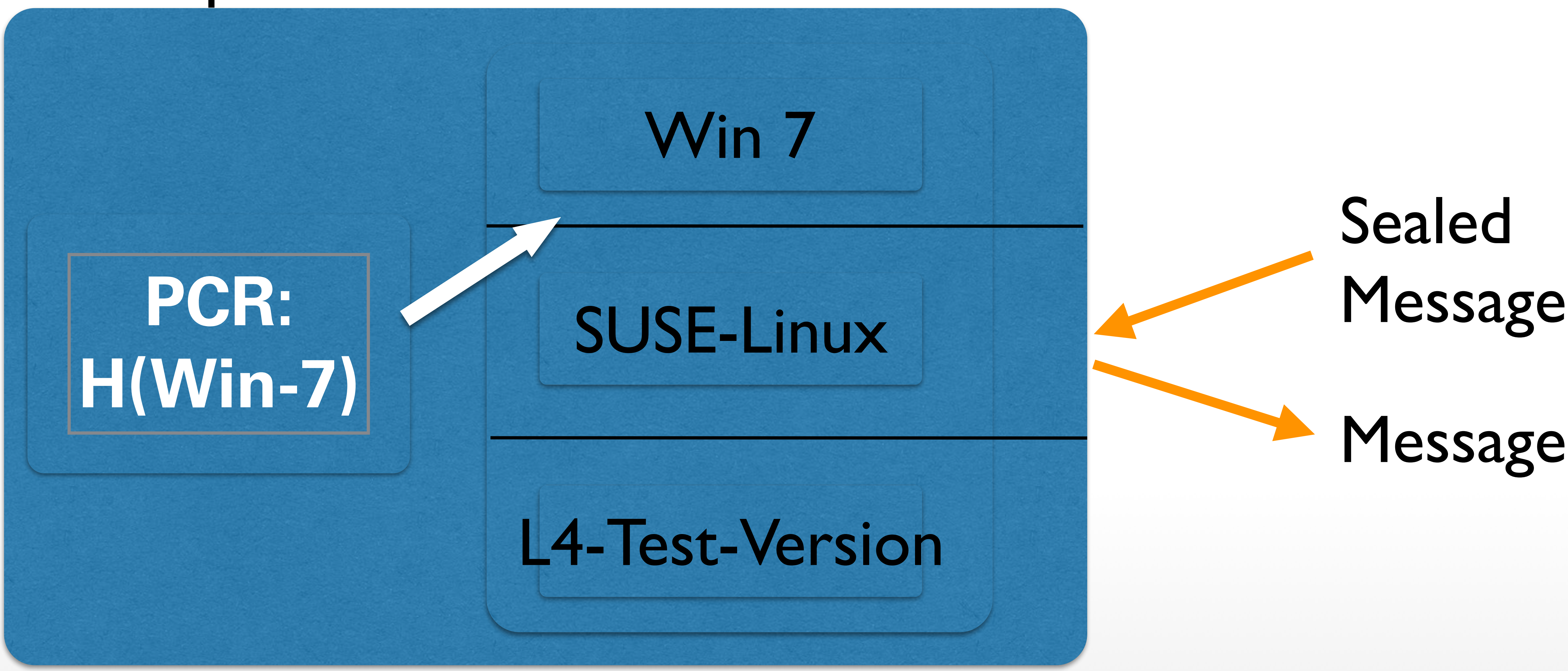


SEALED MEMORY: SEAL OPERATION



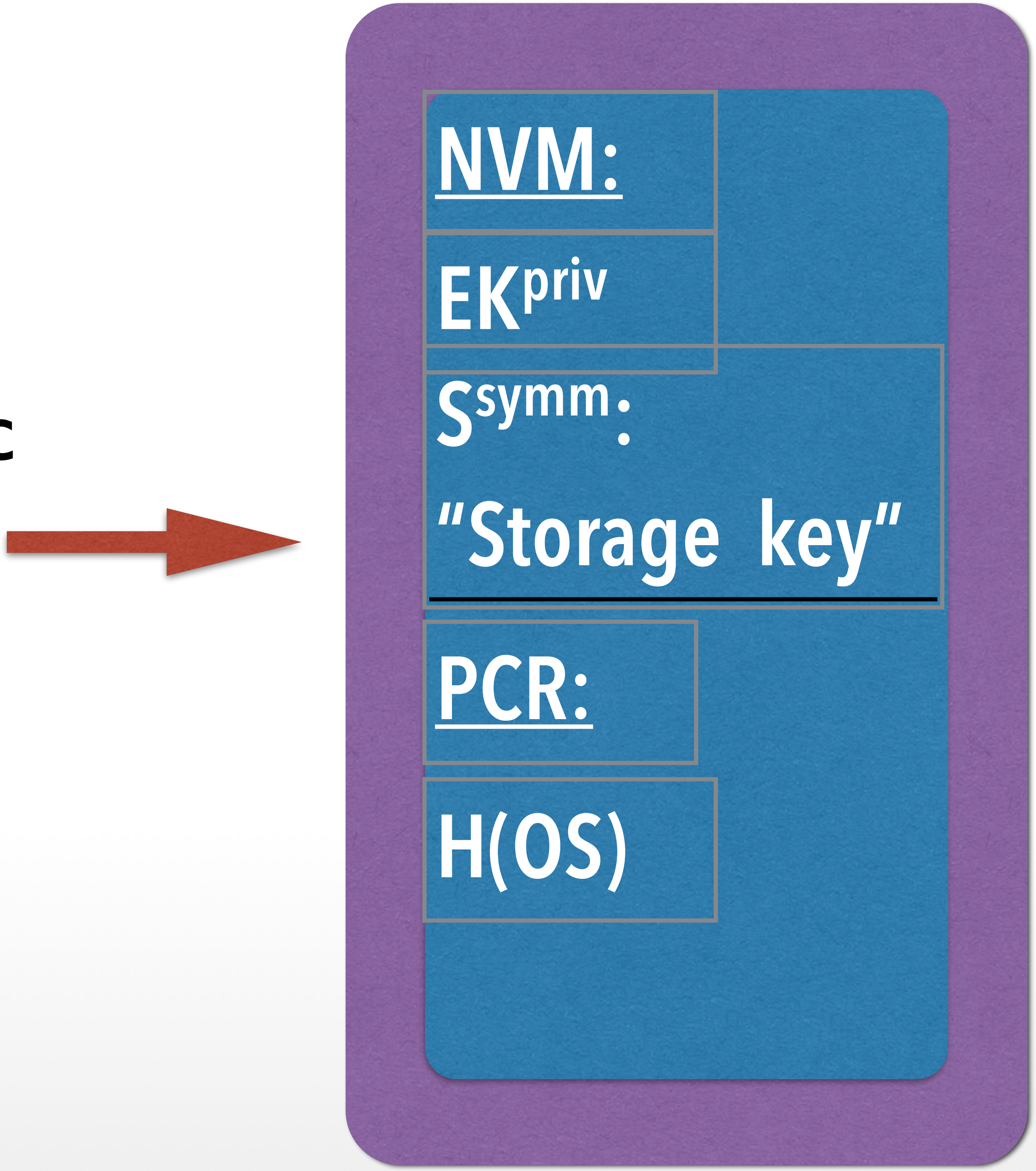
SEALED MEMORY: UNSEAL OPERATION

Tamper-resistant black box



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

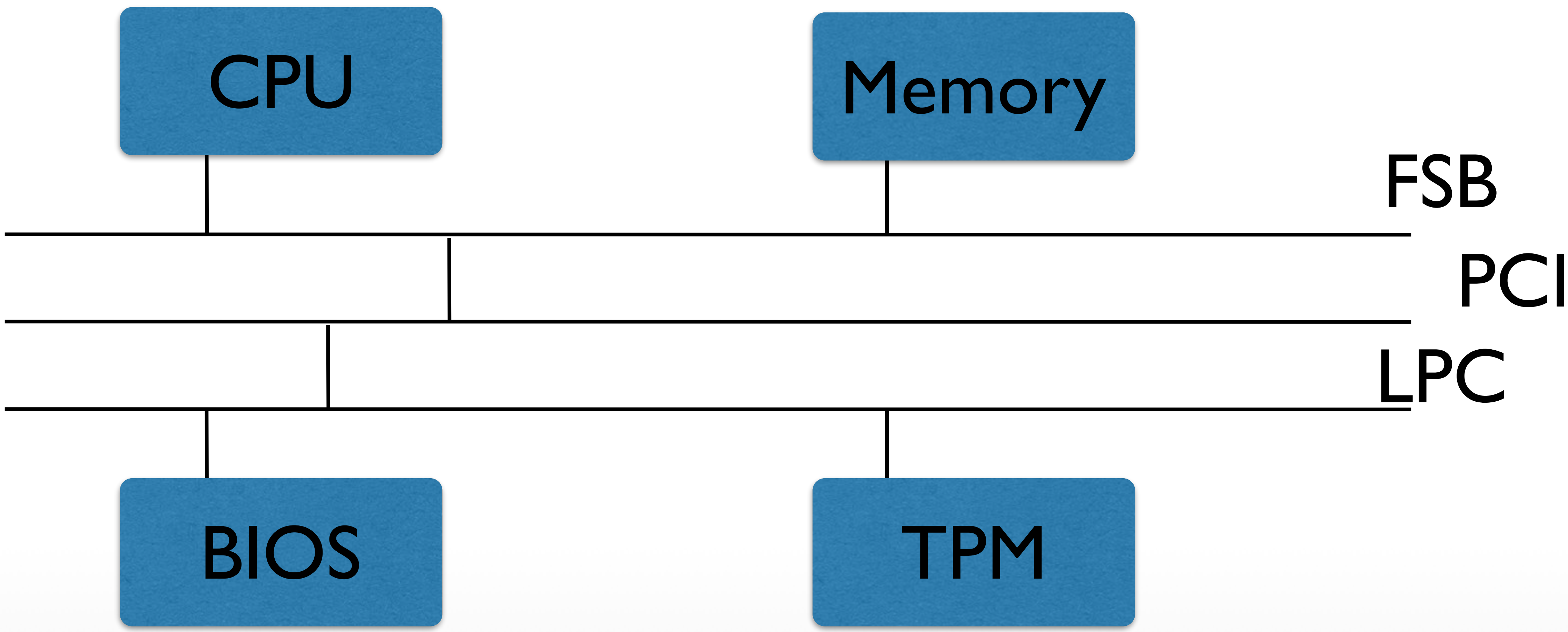
TAMPER RESISTANT BOX ?

Ideally, includes CPU, Memory, …

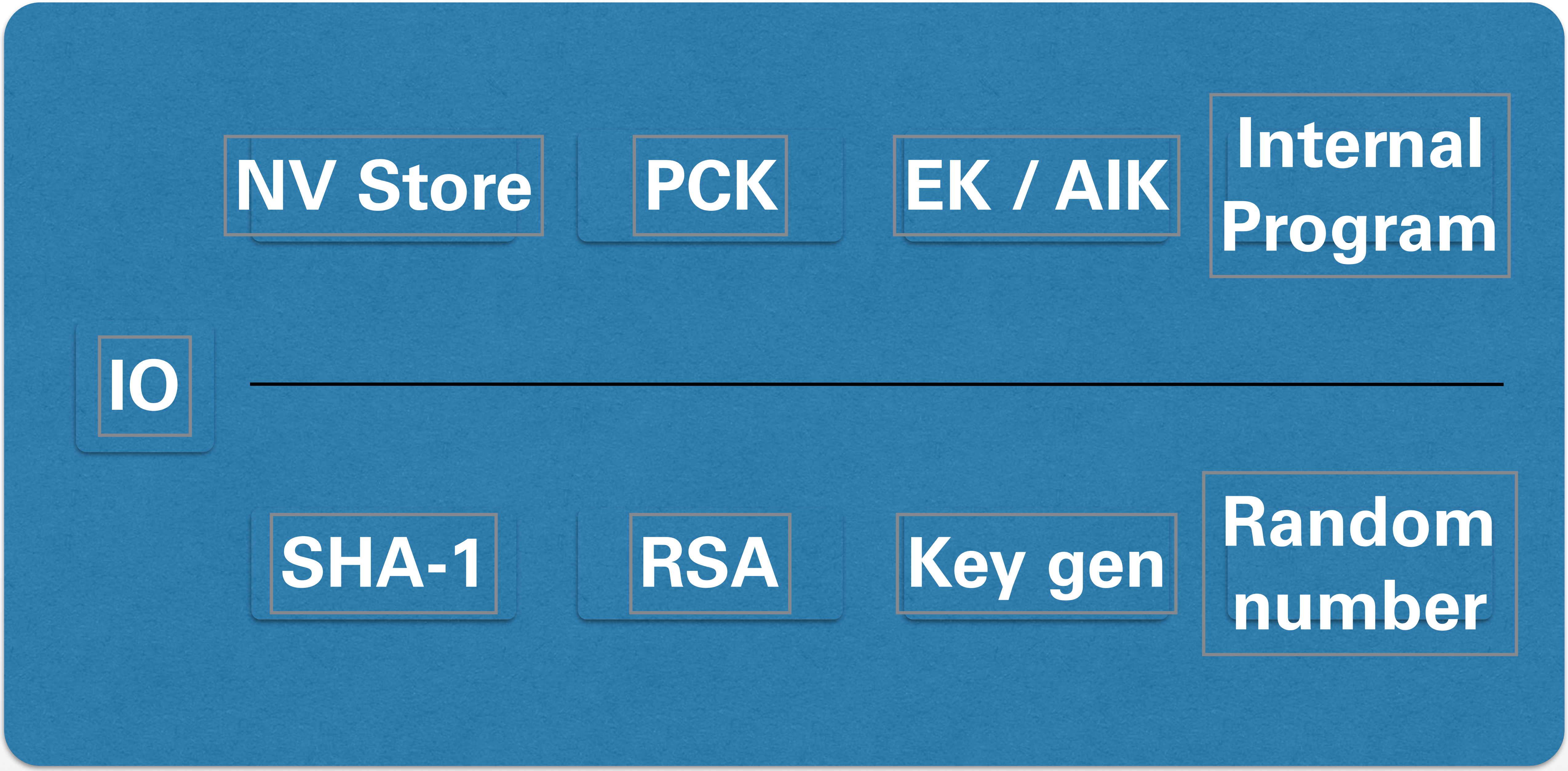
Current practice

- 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 processes: Intel SGX

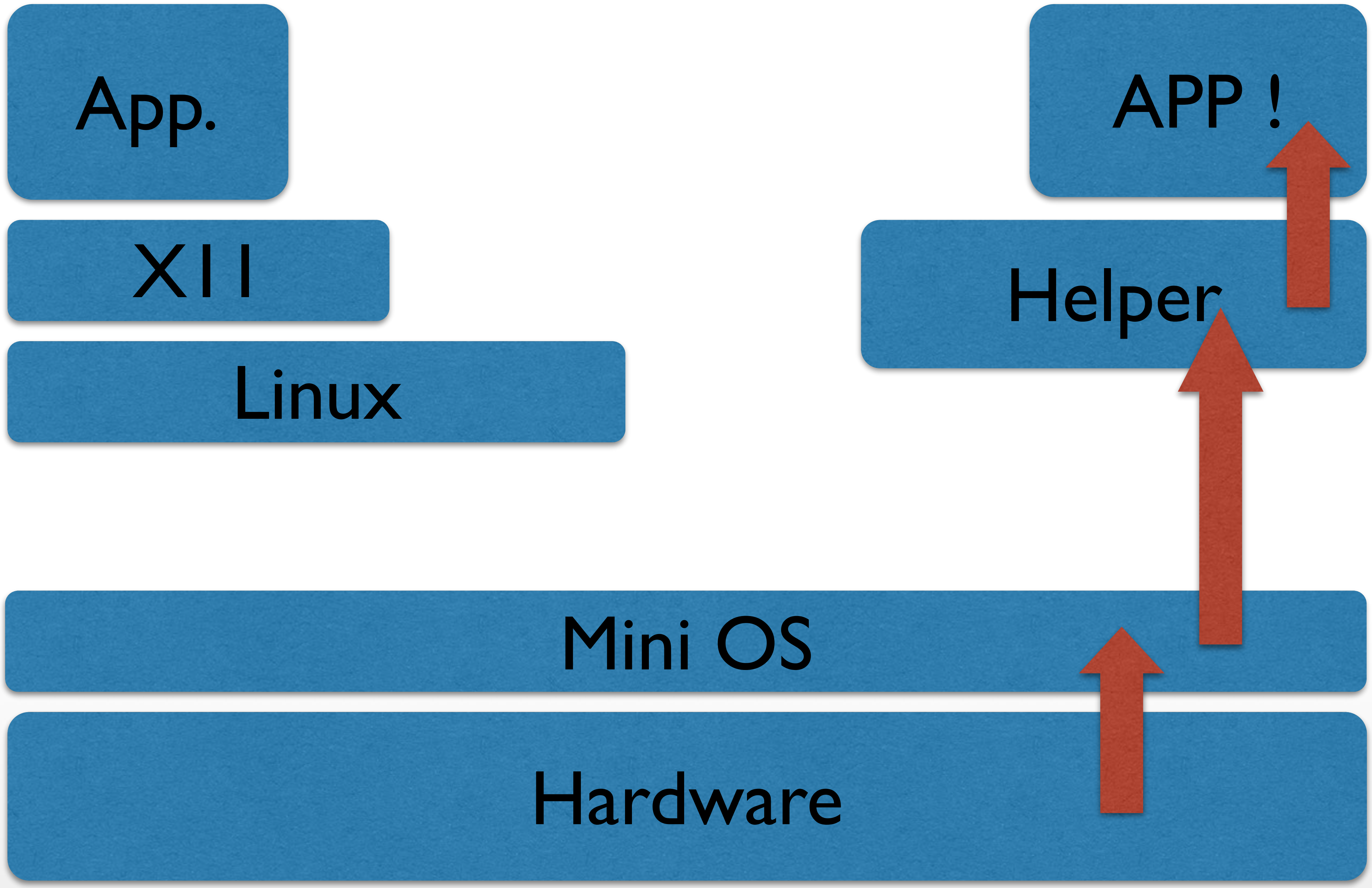
TCG PC PLATFORMS: "TRUSTED PLATFORM MODULE" (TPM)



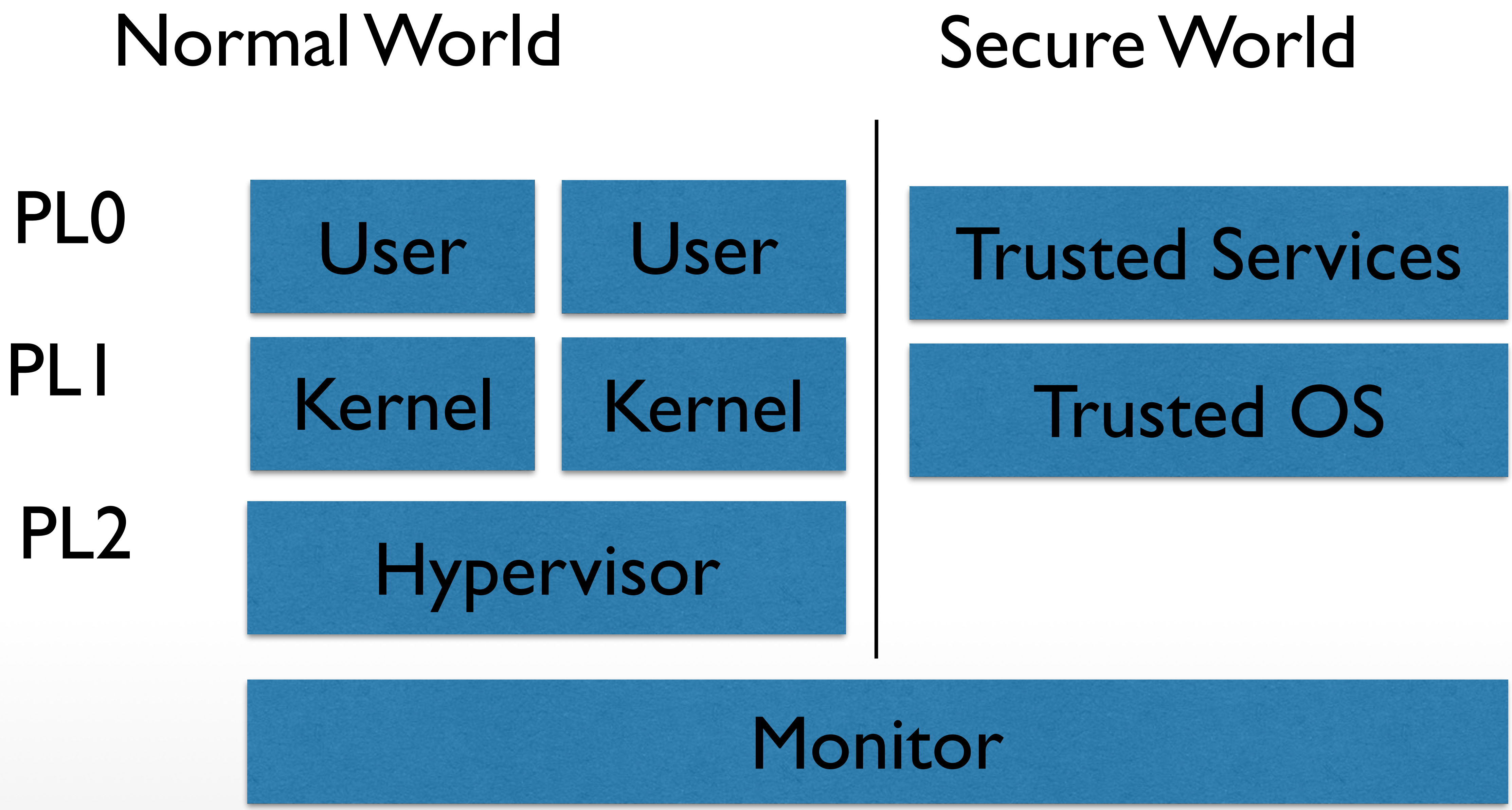
TPM


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

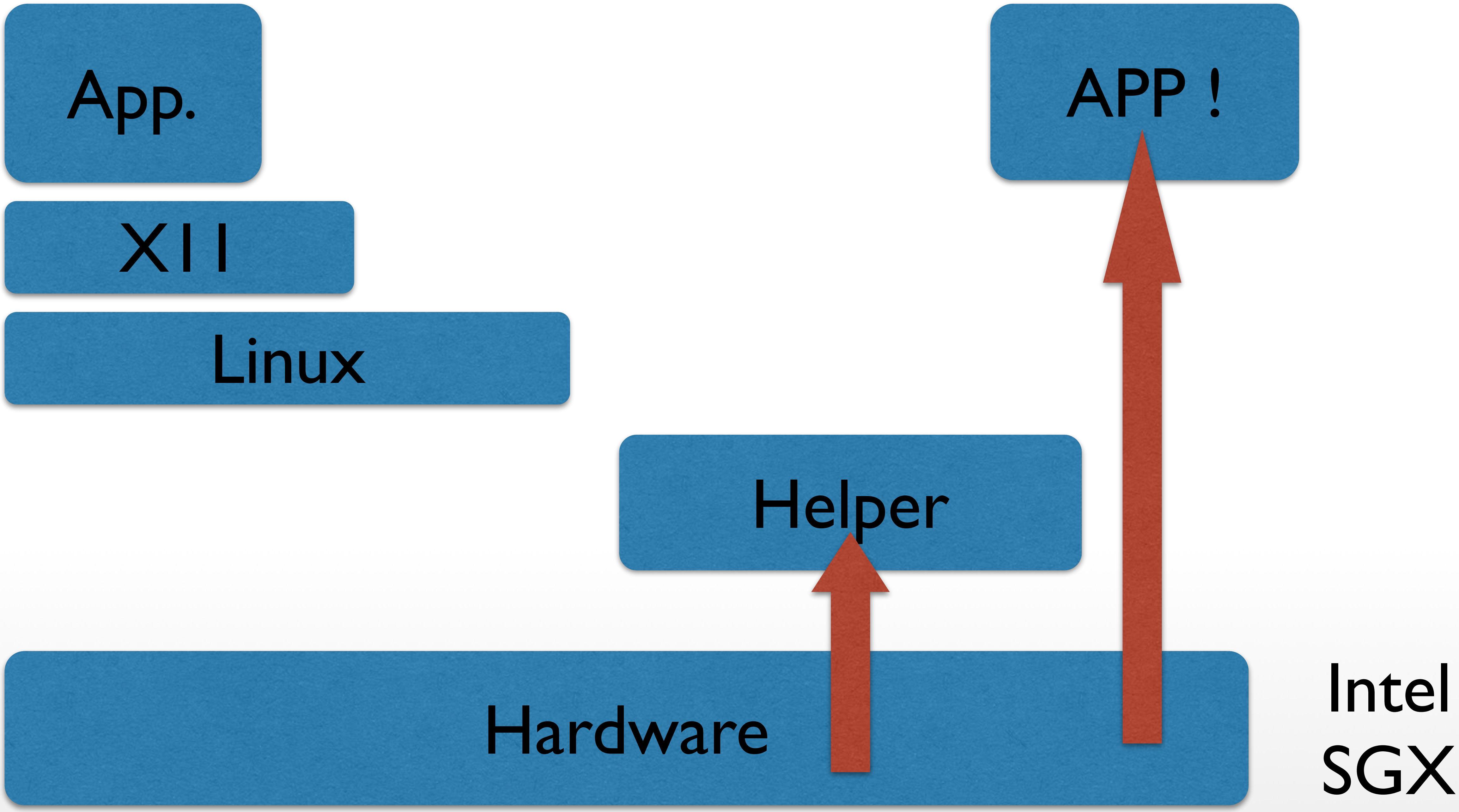
SMALL TRUSTED COMPUTING BASE



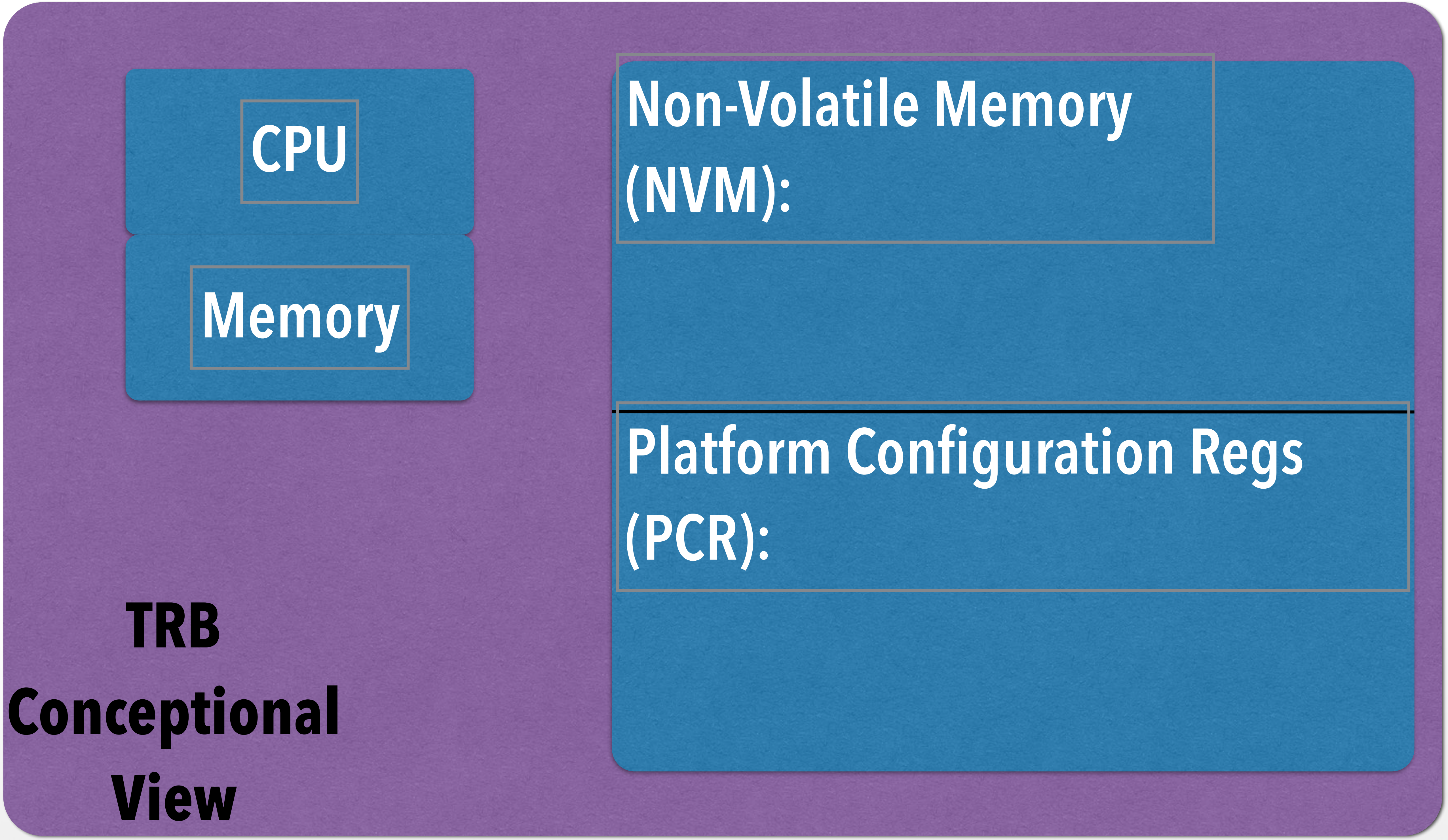
ARM TRUSTZONE



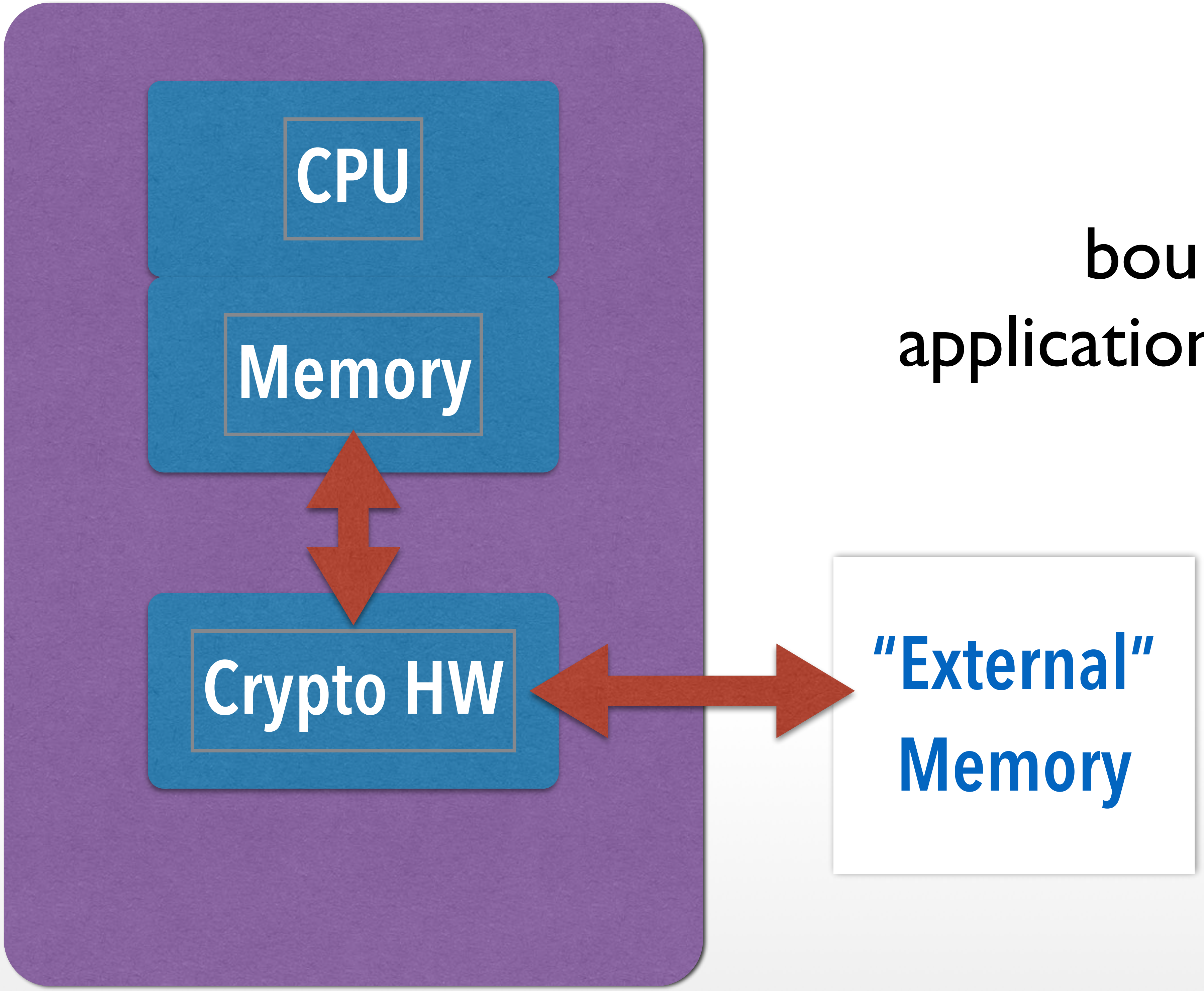
SMALL TRUSTED COMPUTING BASE



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 vendor sources