Distributed OS

Hermann Härtig

Authenticated Booting, Remote Attestation, Sealed Memory aka „Trusted Computing“

TECHNISCHE UNIVERSITÄT DRESDEN

Goals

Understand principles of:

- Authenticated booting, difference to (closed) secure booting
- Remote attestation
- Sealed memory
- Dynamic root of trust
- Protection of applications from the OS
- Some variants of implementations (HW)

Non-Goal:

- Lots of TPM, TCG-Spec 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 compromised 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?
- Can I restrict access to certain secrets (keys) to certain software?
- Can I protect an application against the OS

1) End User Example

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 CLIENT

2) Cloud Example

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
- CLIENT DOES NOT TRUST PROVIDER

3) Industrial Plant Example

(Uranium Enrichment) Plant Control

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

4) Anonymizer example

Anonymity Service

- Intended to provide anonymous communication over internet
- Legal system can request introduction of trap door (program change)
- 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

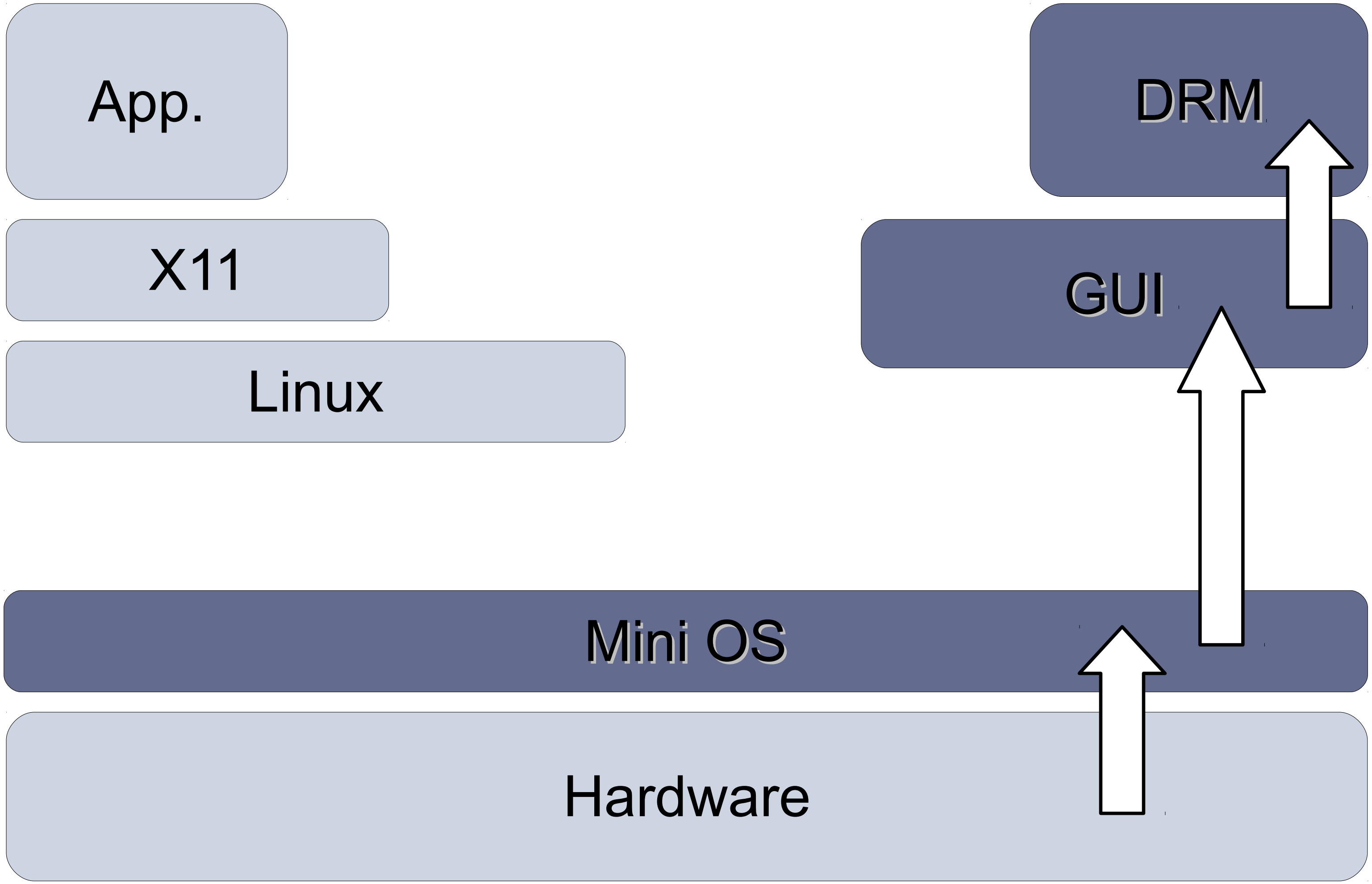
An example application: DRM

- „Digital Content“ is encrypted using symmetric key

- Smart-Card
  - contains key
  - authenticates device
  - delivers key only after successful authentication

- Assumptions
  - Smart Card can protect the key
  - „allowed“ OS can protect the key
  - OS cannot be exchanged

Small Trusted Computing Base



Protection of Application

Principle Method:

separate critical Software

rely on small Trusted Computing Base

- Small OS kernels

  micro kernels, separation kernels, ….

- Hardware

Notation

- **SK^{priv} SK^{pub}** Asymmetric key pair of some entity S

  { M }XK^{priv} Digital Signature for message M using the private key of signer X

  { M }YK^{pub} Message encrypted using public concellation key of Y

- **H(M)** Collision-Resistant Hash Function

- **Certificate** by authority Ca:

  { ID, SK^{pub} , other properties } CaK^{priv}

Notation

Note:

- "{ M }Sk^{priv} Digital Signature"

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

- "{ M }Sk^{pub} Message concealed ..."

  does not necessarily imply public key encryption

  for full M

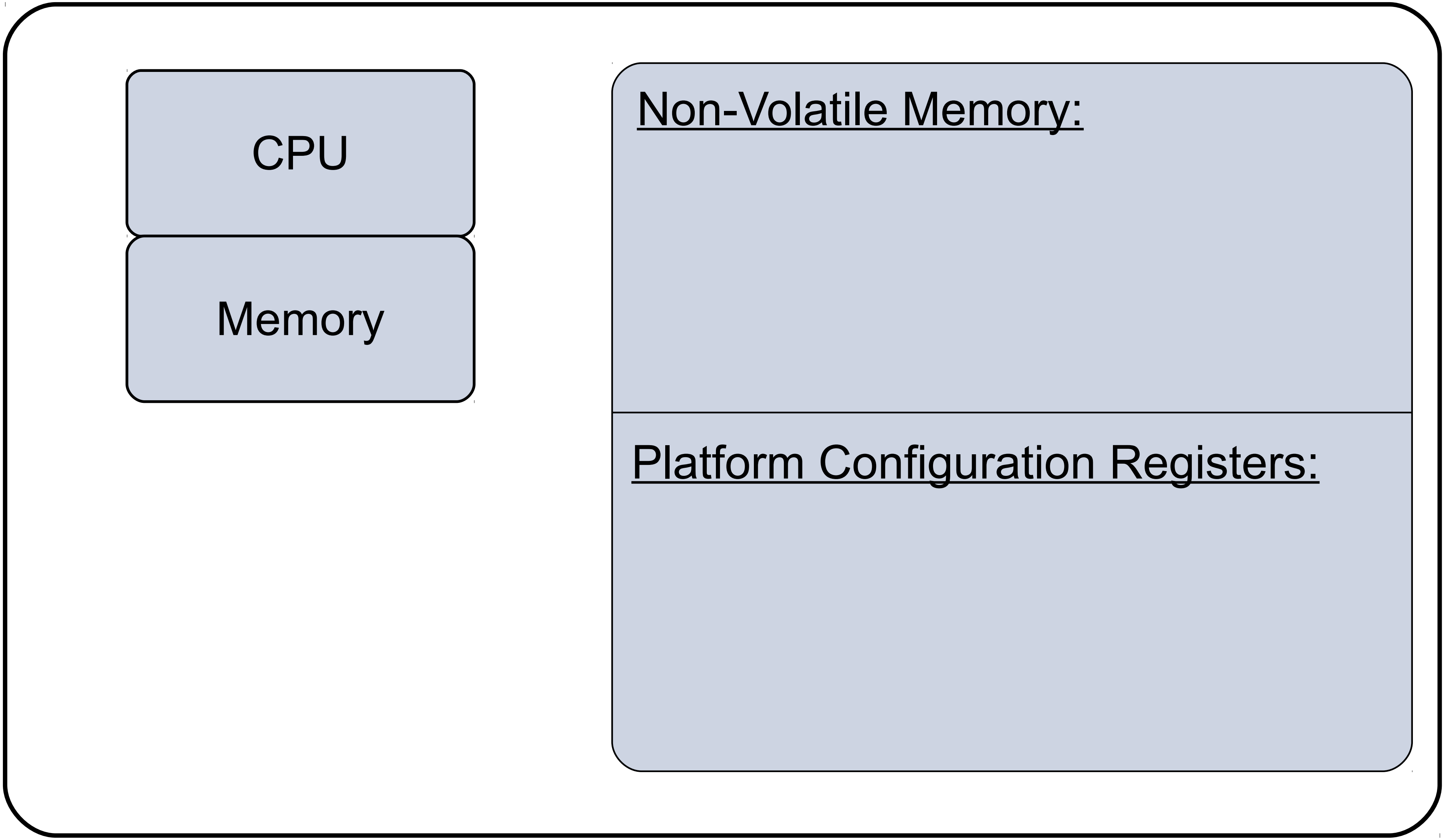
  (rather a combination of symmetric

  and asymmetric methods)

Identification of Software

- Program vendor: Foosoft FS
- Two ways to identify Software: Hash / public key
  - H(Program)
  - {Program, ID- Program}FSK^{priv}
    use FSK^{pub} to check the signature must be made available, e.g. shipped with the Program
- The „ID” of SW must be made available somehow.

Tamperresistant black box



Ways to “burn in” the OS or secure booting

- Read-Only Memory
- Allowed H(OS) in NV memory preset by manufacturer
  - load OS- Code
  - compare H(loaded OS code) to preset H(OS)
  - abort if different
- Preset FSK^{pub} in NV memory preset by manufacturer
  - load OS- Code
  - check signature of loaded OS-Code using FSK^{pub}
  - abort if check fails

Authenticated Booting, using HASH

Steps:

- Preparation by Manufacturers (TRB and OS)
- Booting & “Measuring”
- Remote attestation

Authenticated Booting, using HASH

CPU

Memory

Non-Volatile Memory:

"Endorsement Key" EK
preset by Manufacturer

Platform Configuration Registers:

PCR:
Hash-Code obtained during boot

Vendors of TRB and OS

- TRB generates key pair: „Endorsement Key“ (EK)
  - stores in TRB NV Memory: EK^{priv}
  - emits: EK^{pub}

- TRB vendor certifies: {“a valid EK”, EK^{pub}}TVK^{priv}
- OS-Vendor certifies: {„a valid OS“, H(OS)}$OSVK^{priv}$
- serve as identifiers: EK^{pub} and H(OS)

Booting & Attestation, using HASH

Booting:

- TRB “measures” OS- Code (computes H(OS-Code))
- stores in PCR
- no other way to write PCR

Attestation:

- Challenge: nonce
- TRB generates Response: {PCR, nonce' }EK^{priv}

Authenticated Booting, using public key

CPU

Memory

Non-Volatile Memory:

"Endorsement Key" EK
preset by Manufacturer

Platform Configuration Registers:

PCR:
OSK^{pub} used to check OS

Vendors of TRB and OS, using Key

- TRB_generates key pair:
  - stores in TRB NV Memory: EK^{priv}
  - emits: EK^{pub}
- TRB vendor certifies: {“a valid EK”, EK^{pub}}TVK^{priv}
- **OS-Vendor certifies: {„a valid OS“, OSK^{pub}}$OSVK^{priv}$**
- **and signs OS-Code: {OS-Code}OSK^{priv}**
- serve as identifiers: EK^{pub} and OSK^{pub}

Booting & Attestation, using Key

Booting:

- TRB checks OS- Code using some **OSK**^{pub}
- **stores OSK**^{pub} **in PCR**
- no other way to write PCR

Attestation:

- Challenge: nonce
- TRB generates Response: {PCR, nonce' }EK^{priv}

A Race condition

Boot Linux

→ new active OS

Nonce

Content

Boot Windows

→ new active OS

Content

Boot Linux

→ new active OS

Auth. Booting considering reboot

- attestation required at each request
- Do not use EK

This is one way of doing it:

create new keypairs on every reboot

Booting (AB considering reboot)

Booting:

- TRB checks OS- Code using some OSK^{pub}
- store OSK^{pub} in PCR
- create 2 keypairs for the booted OS (“Active OS”):
  - ActiveOSAuthK /* for Authentication
  - ActiveOSConsK /* for Concellation
- certifies: $\{ActiveOSAuthK^{pub}$, ActiveOSConsKpub,$OSK^{pub}\}EK^{priv}$
- Hand over ActiveOSKeys to booted OS

Attestation (AB considering reboot)

Remote Attestation:

- Challenge: nonce
- Active OS generates response:

{ ActiveOSConsKpub, ActiveOSAuthK^{pub}, OSK^{pub}}EK^{priv} /* see previous slide

{nonce'} ActiveOSAuthK^{priv}

Encrypted Channel via the active OS:

- { message } ActiveOSConsK^{pub}

Assumptions

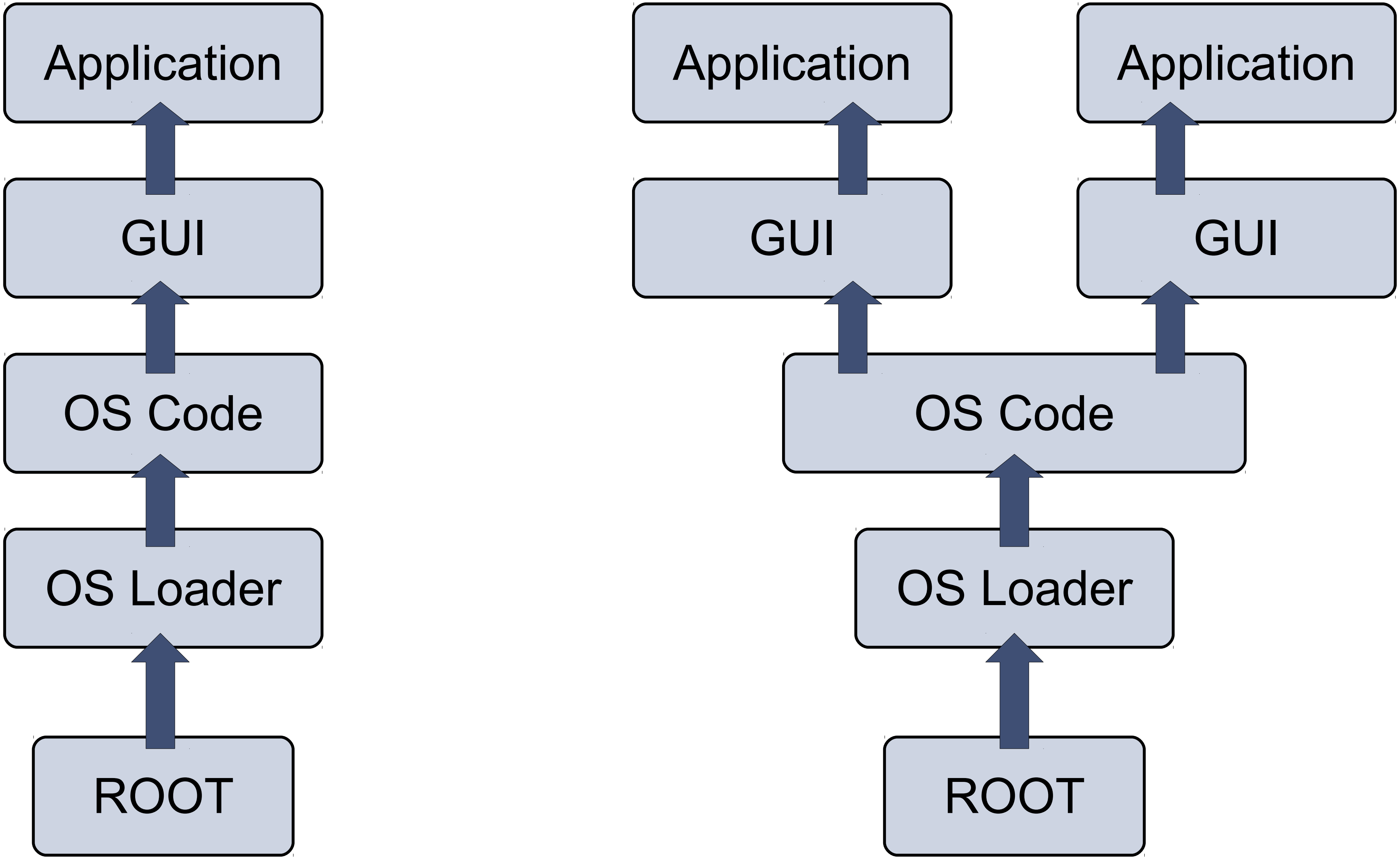
TRB can protect: EK, PCR

OS can protect: $ActiveOSAuthK_{priv}$ $ActiveOSConsK_{priv}$

Rebooting destroys content of

- PCR
- Memory Holding $ActiveOSAuthK_{priv}$ $ActiveOSConsK_{priv}$

Software stacks and trees



Software stacks and trees

2 Problems:

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

Software stacks and trees

- “Extend” Operation
  - stack: $PCR_n = H(PCR_{n-1} \,||\, \text{next-component})$
  - tree: difficult (unpublished ?)

- Key pairs per step:
  - OS controls applications → generate key pair per application
  - OS certifies
    - { Application 1, $App1K^{pub}$ } $ActiveOSK^{priv}$
    - { Application 2, $App2K^{pub}$ } $ActiveOSK^{priv}$

Late Launch

- 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

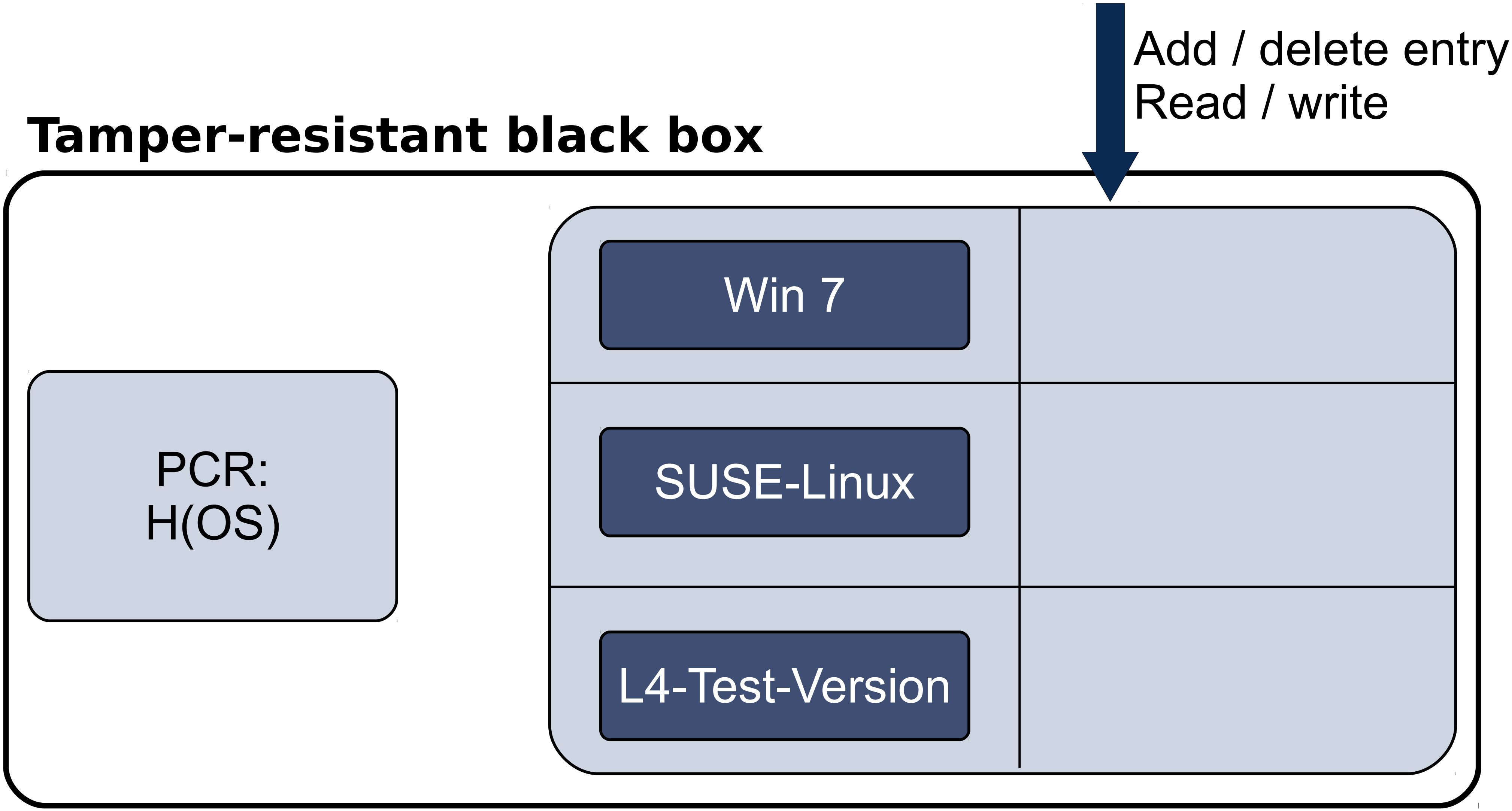
Problem:

- Send information using secure channels
- Bind that information to Software configuration
- Work offline:

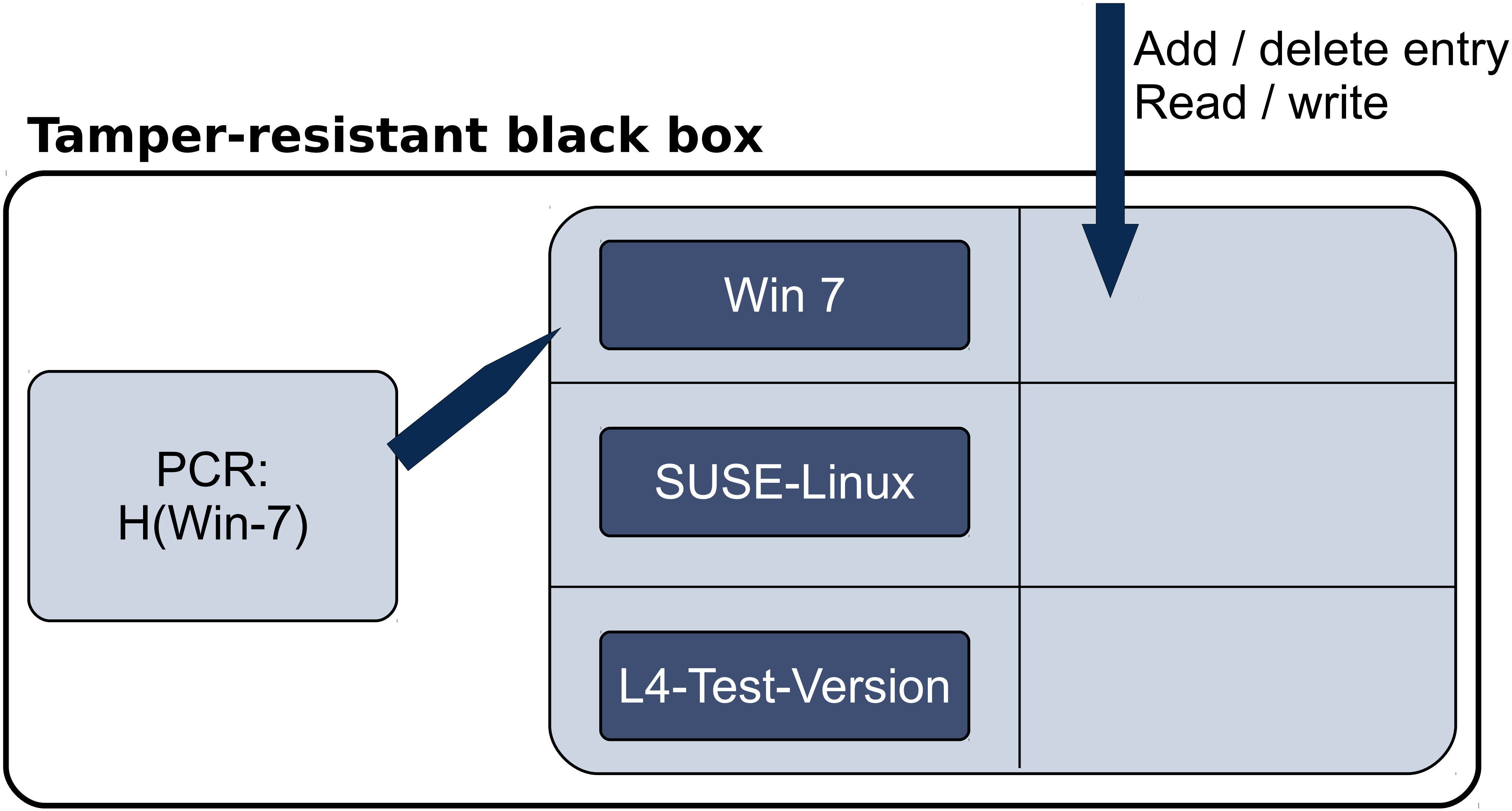
  How to store information in the absence of communication channels?
- For example DRM:

  bind encryption keys to specific machine, specific OS

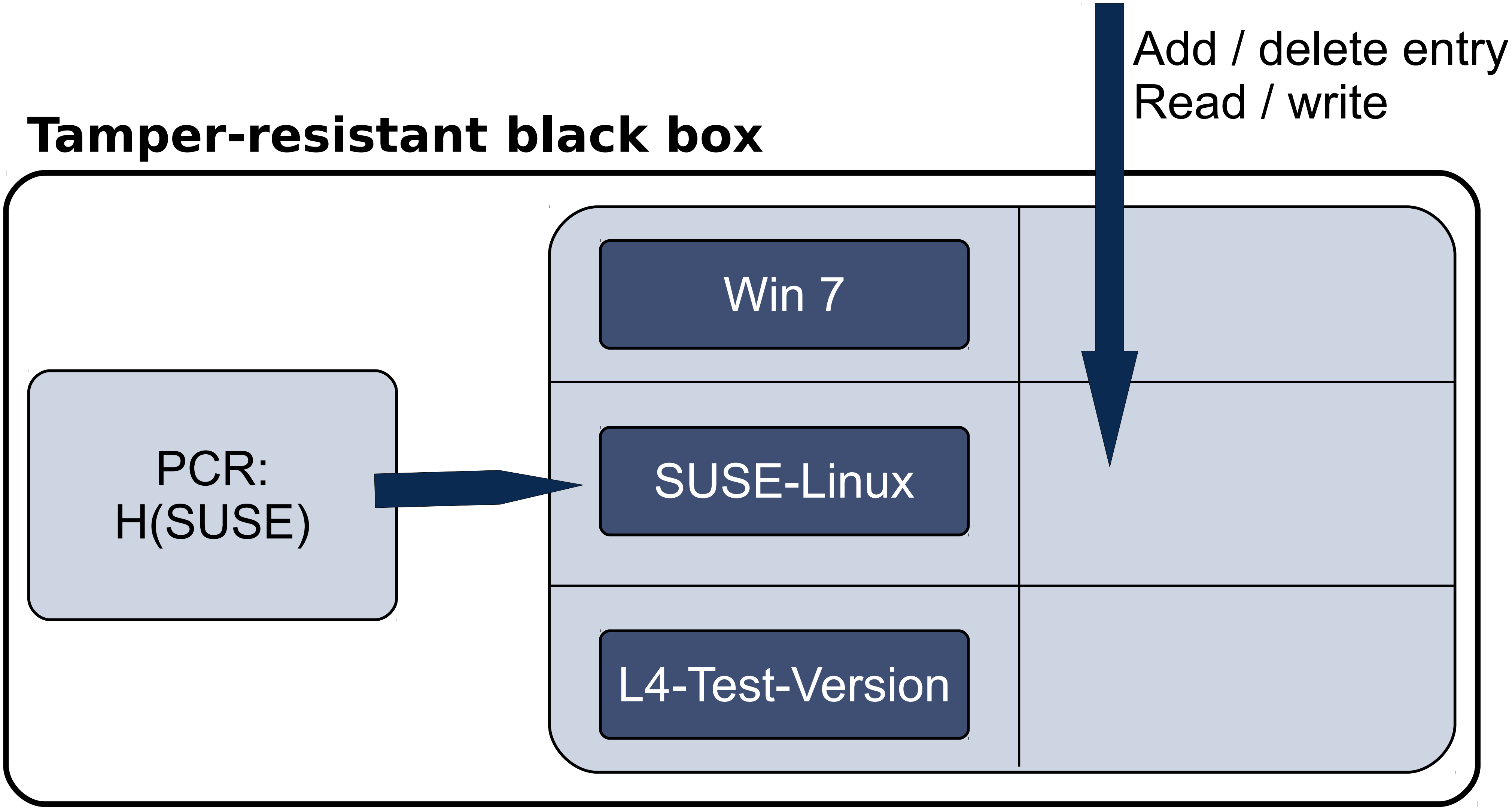
Sealed Memory



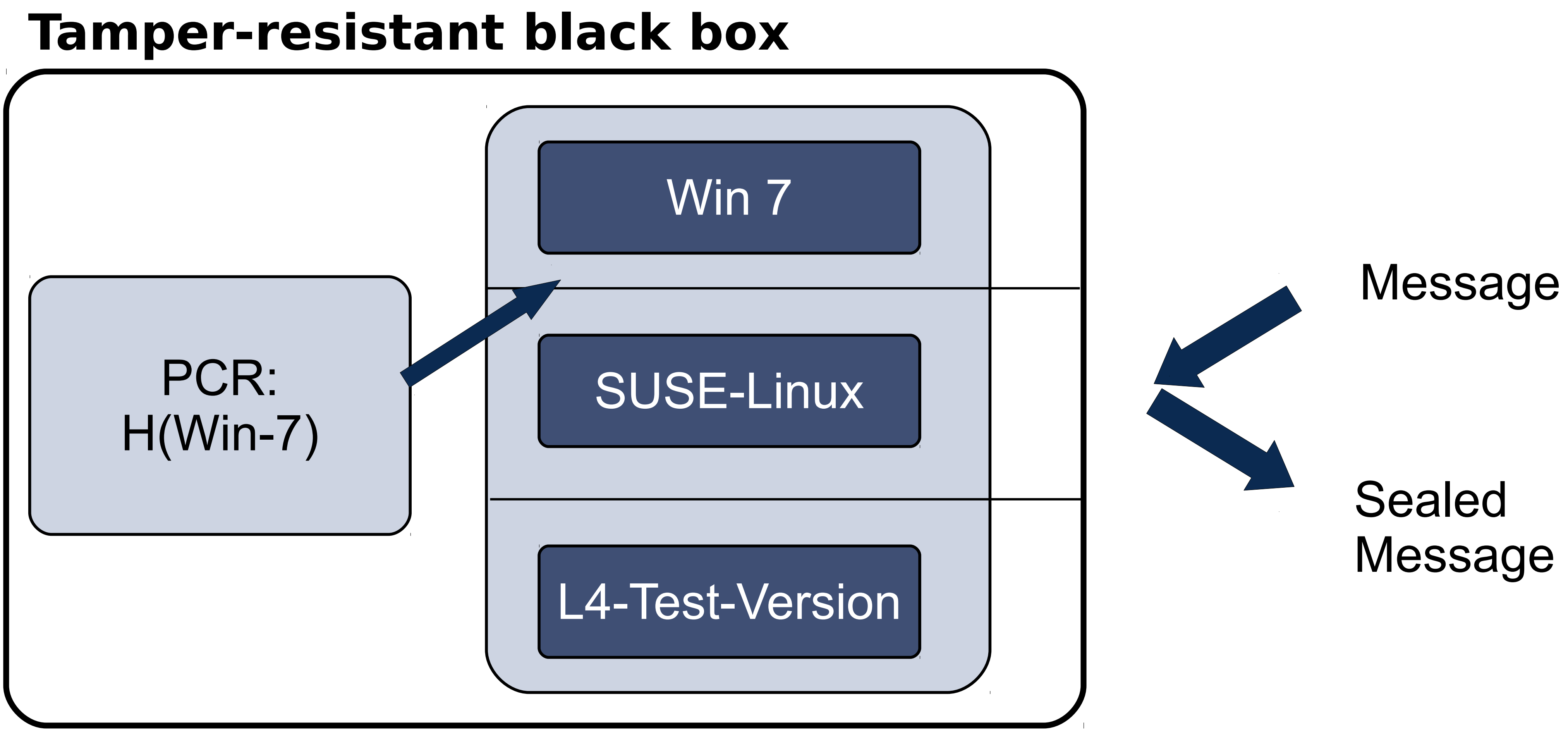
Sealed Memory



Sealed Memory

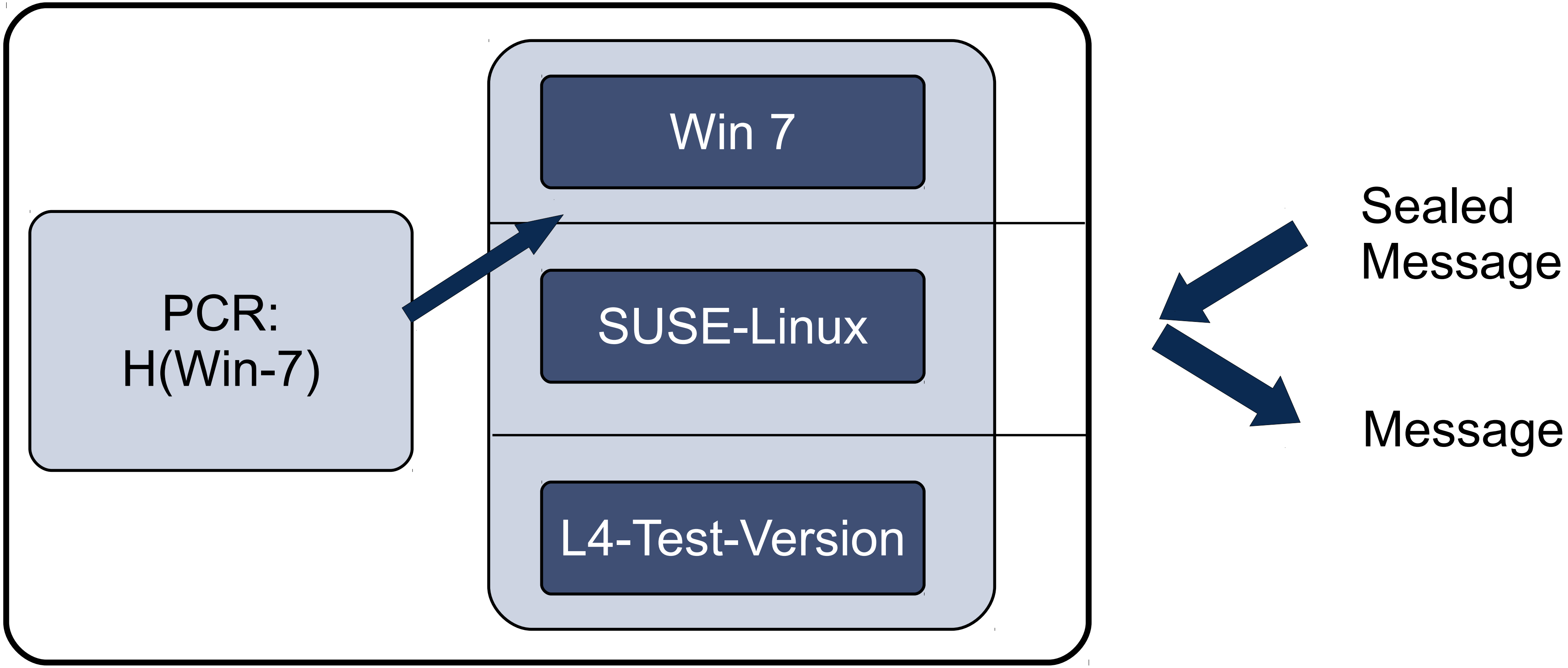


Sealed Memory: Seal Operation

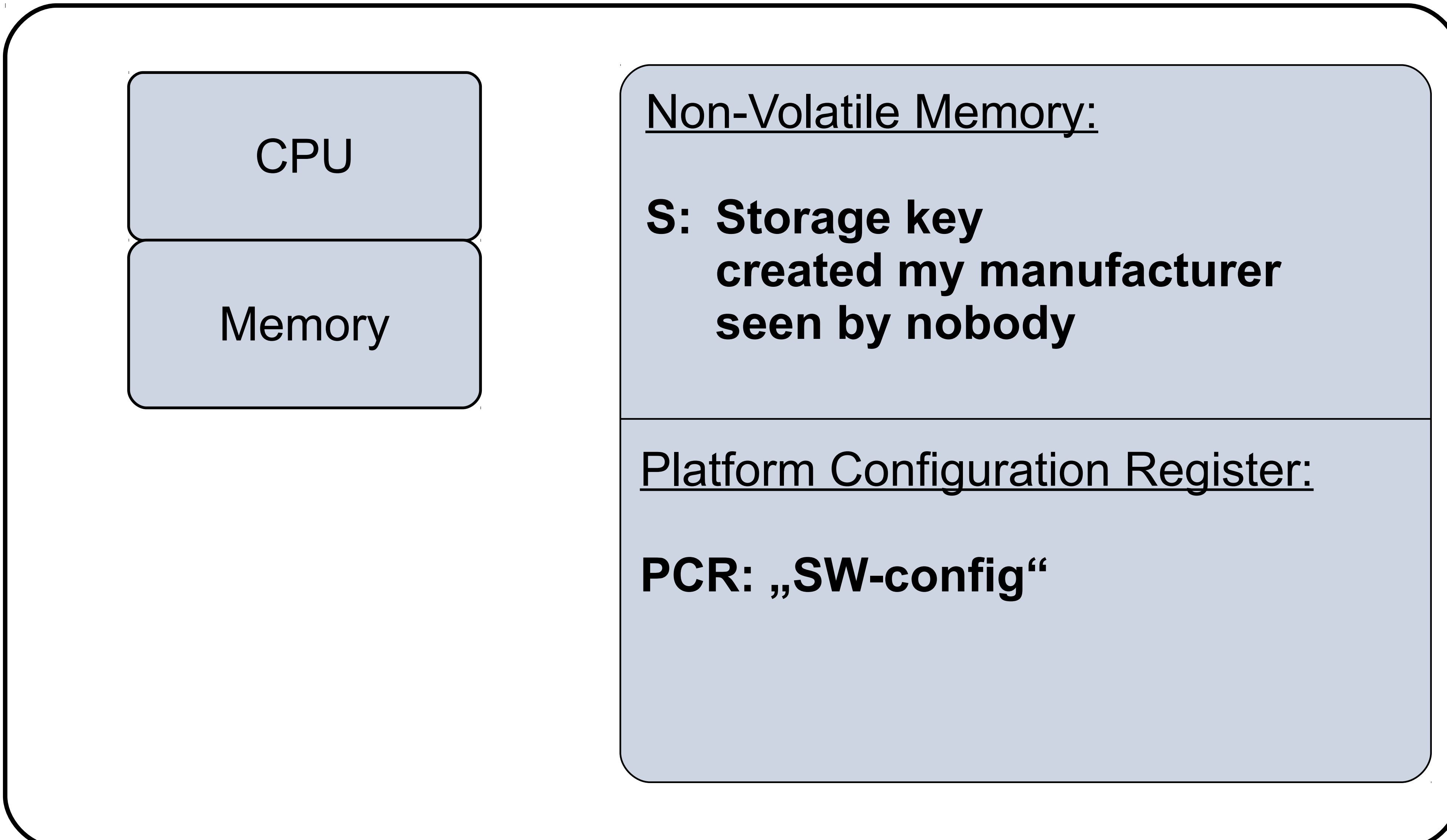


Sealed Memory: Unseal Operation

Tamper-resistant black box



Tamperresistant black box (TRB)



Sealed Memory

- Seal(message):

  encrypt(“PCR, message”, Storage-Key)

  → “sealed message”;

  emit sealed message

- Unseal(sealed_message):

  decrypt(“sealed_message”, Storage-Key)

  → “SW config, message”;

  If SW config == PCR then emit message else abort fi

Sealed Memory for future configuration

- Seal(message, **FUTURE_Config**):

  encrypt("**FUTURE_Config**, message", Storage-Key)

  → "sealed message";
  emit sealed_message

- "seals" information such that it can be unsealed by a future configuration (for example: future version)

Example

- Win8: Seal („SonyOS, Sony-Secret")
  → SealedMessage (store it on disk)

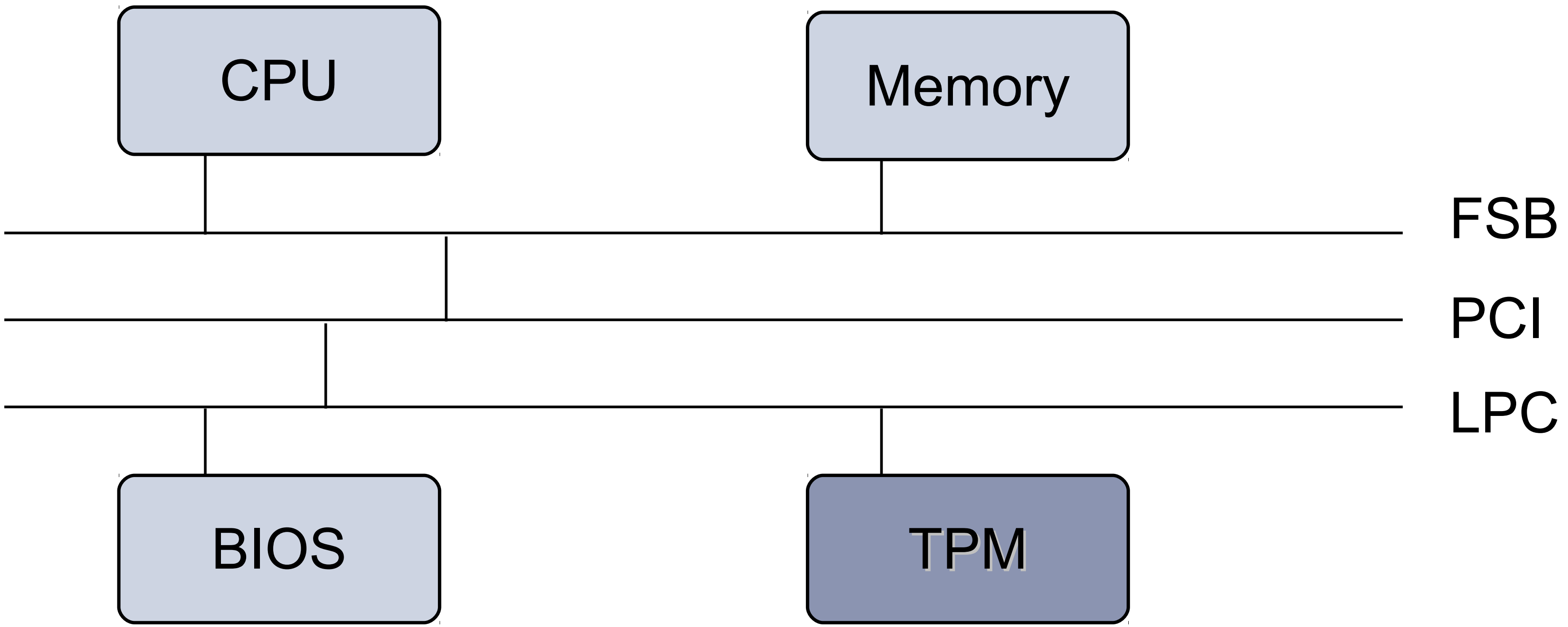
- L4: Unseal (SealedMessage)
  → SonyOS, Sony-Secret → PCR#SonyOS → abort

- SonyOS: Unseal(SealedMessage
  → SonyOS, Sony-Secret → PCR==SonyOS → ok

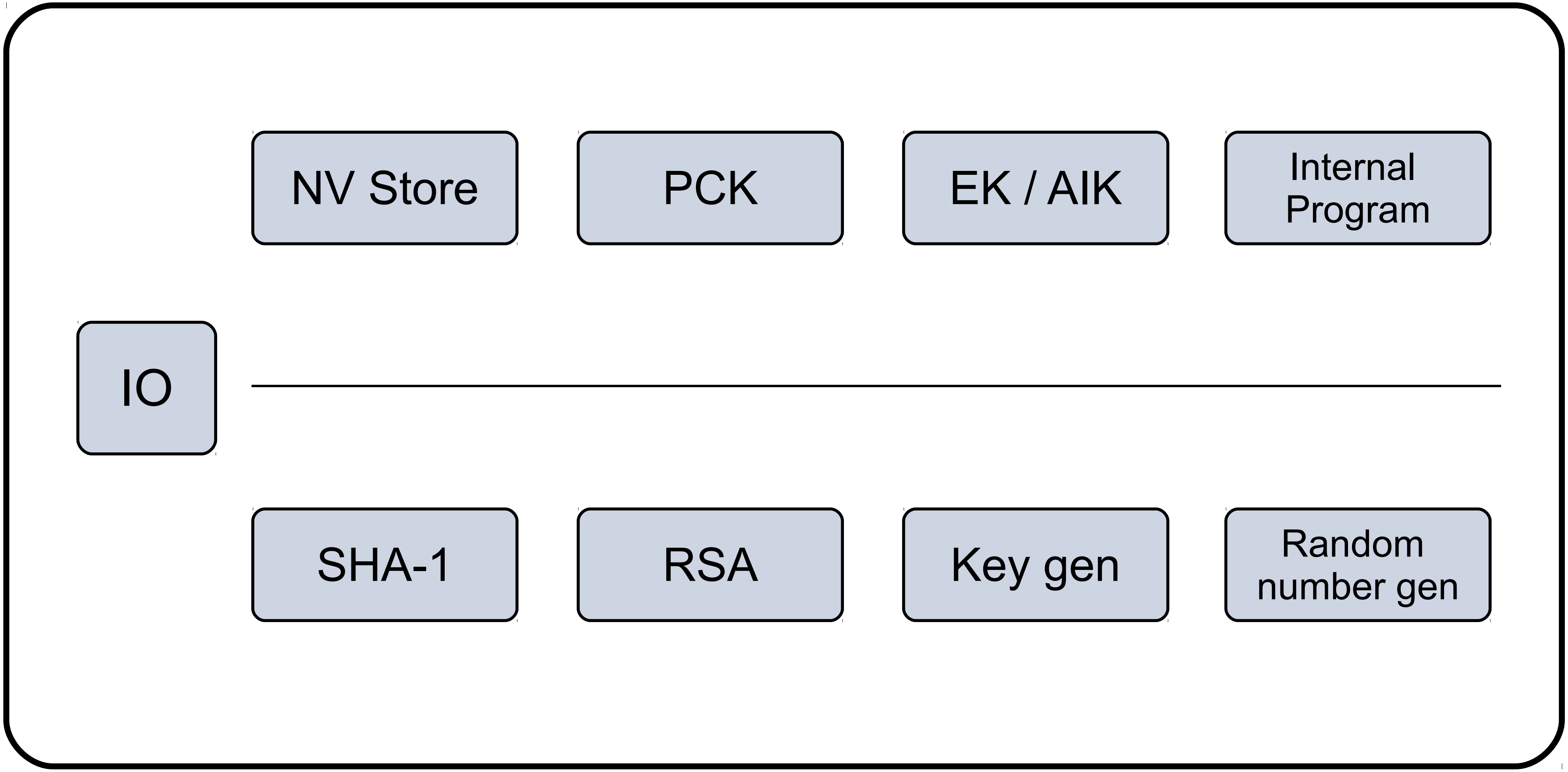
Tamper Resistant Box ?

- Ideally, includes CPU, Memory, …
- In practice
  - Additional physical protection, for example IBM 4758 … look it up in Wikipedia
  - Recent HW versions
    - TPM: separate “Trusted Platform Modules” (replacing BIOS breaks TRB)
    - Add a new privilege mode:
      - ARM TrustZone
      - Intel SGX

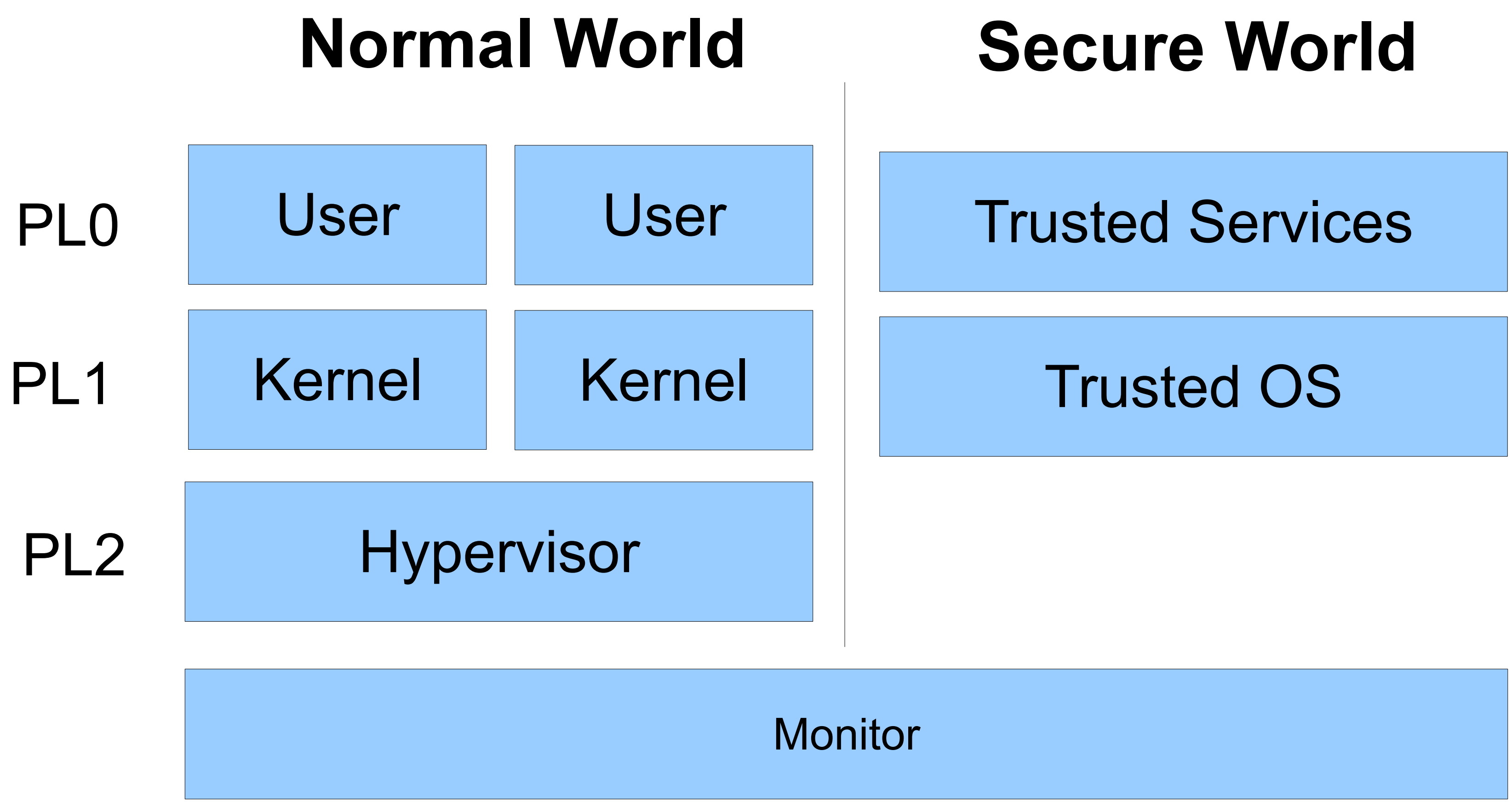
TCG PC Platforms: “Trusted Platform Module” (TPM)



TPM


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

ARM TrustZone



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
- https://software.intel.com/sites/default/files/329298-001.pdf
- http://www.slideshare.net/daniel_bilar/intel-sgx-2013
- ARM Trustzone