

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

Hermann Härtig

Technische Universität Dresden

Summer Semester 2007



Goals

Understand:

- authenticated booting
- the difference to (closed) secure booting
- remote attestation
- sealed memory

Learn

to find out about TCPA/TCG documents TPMs etc



Some terms

Secure Booting

Authenticated Booting

(Remote) Attestation

Sealed Memory

Late launch / dynamic root of trust

Trusted Computing

Trusted Computing Base

Attention:

terminology has changed ...



Trusted Computing (Base)

Trusted Computing Base

The set of 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 SW 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 programs ?



Trusted Computing Terminology

Measuring

“process of obtaining metrics of platform characteristics”

Examples Hash- Codes of SW

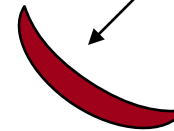
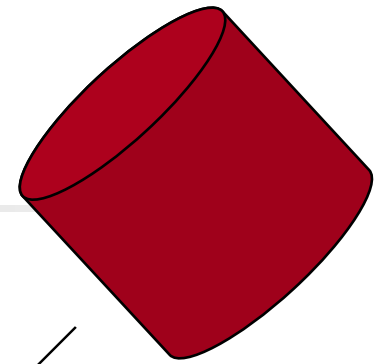
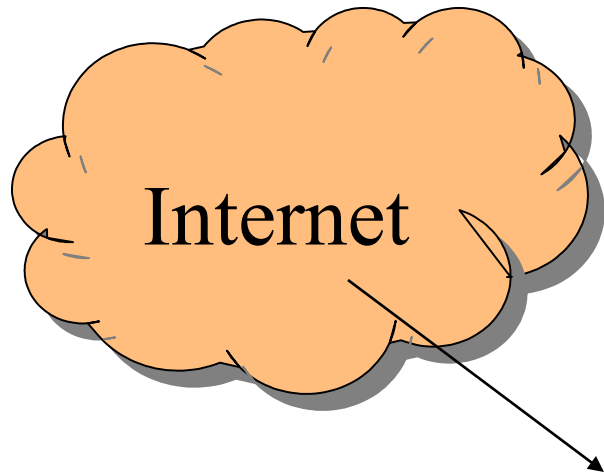
Attestation

“vouching for accuracy of information”

Sealed Memory

binding information to a configuration

DRM: Trust ./. No Trust in end user



{Digital Content}K

Decoder

TV

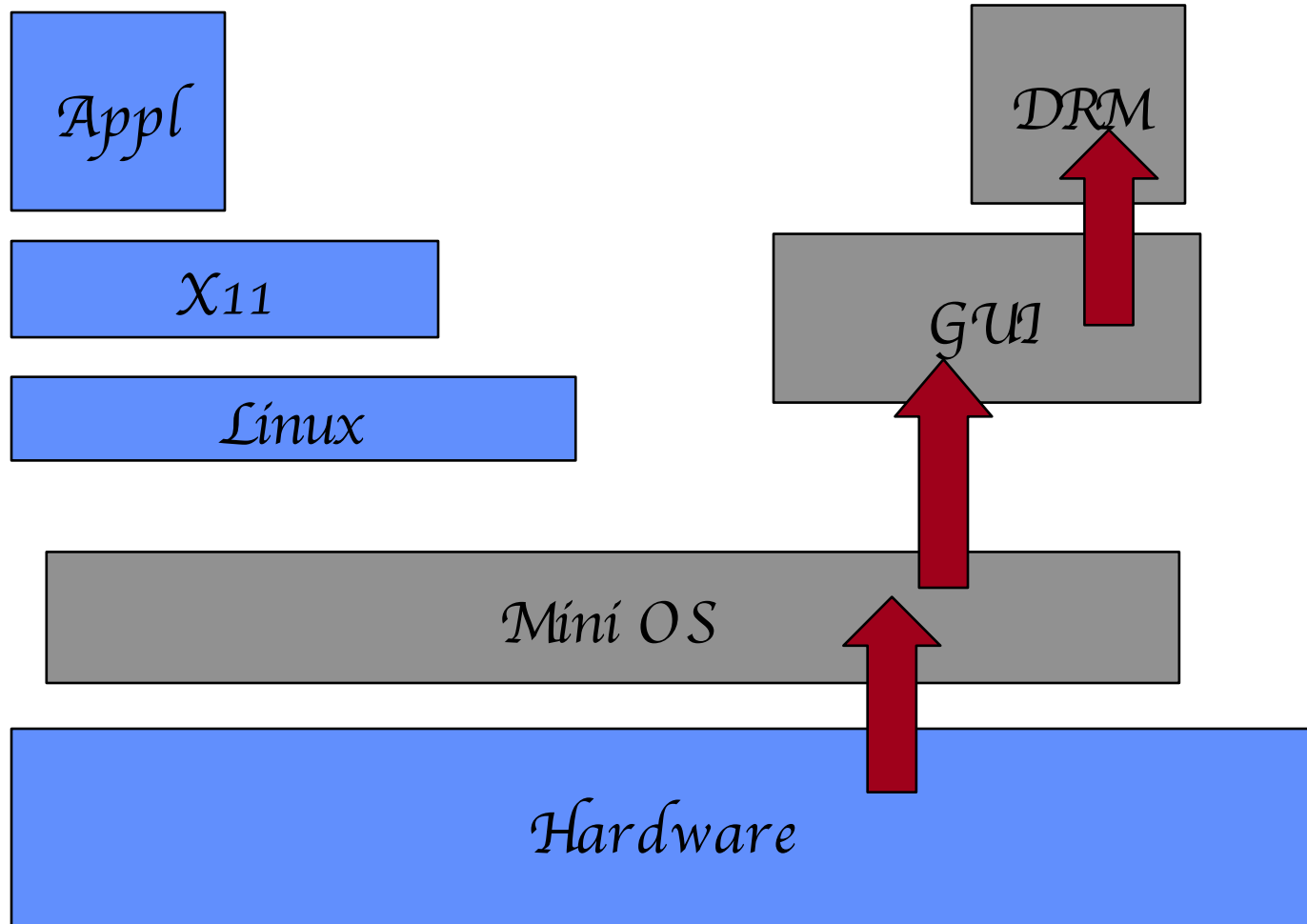
K



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

Secure Booting / Authenticated Booting





Notation

$SK^{\text{priv}} \quad SK^{\text{pub}}$ Asymmetric key pair of some entity S

$\{ M \}_{SK^{\text{priv}}}$ Digital Signature for message M
using the private key of signer S

$H(M)$ Collision-Resistant Hash

Certificate by authority Ca :

$\{ ID, SK^{\text{pub}}, \text{other properties} \}_{CaK^{\text{priv}}}$



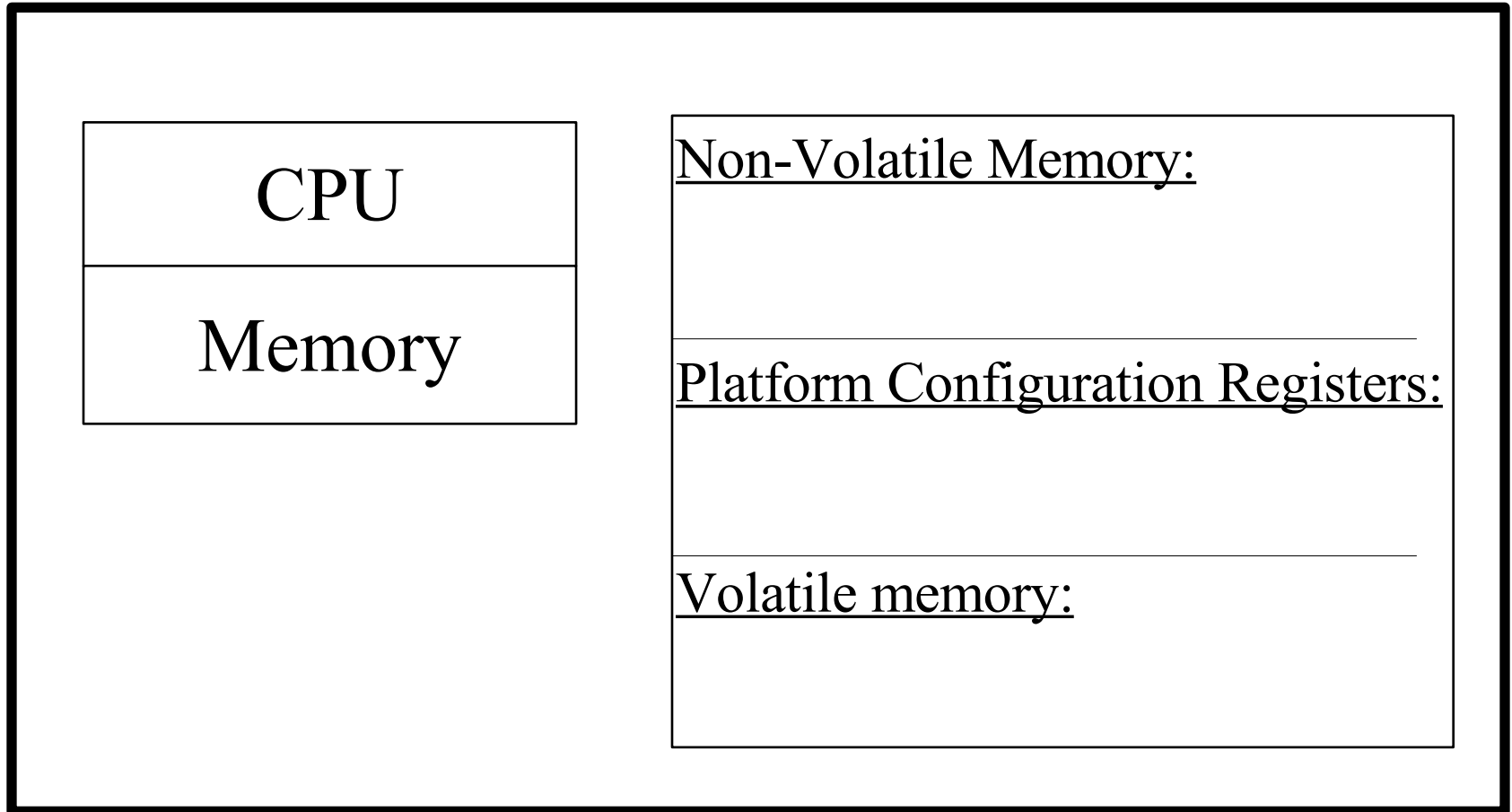
Identification of Software

Program vendor: Foosoft FS

- $H(\text{Program})$
- $\{\text{Program, ID- Program}\}_{\text{FSK}^{\text{priv}}}$
use FSK^{pub} to check



Tamperresistant black box (TRB)





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 (AB)

Phases:

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



Authenticated Booting (AB)

CPU

Memory

Non-Volatile Memory:
“Endorsement Key” EK
preset by Manufacturer

Platform Configuration Registers:
Hash-Code obtained during boot

Volatile memory:



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

Booting:

TRB “measures” OS- Code (computes $H(\text{OS-Code})$)
stores in PCR
no other way to write PCR

Attestation:

Challenge: nonce

TRB generates Response:

$\{\text{PCR, nonce}\}_{\text{EK}^{\text{priv}}}$



Remaining problems

Now we know identities: $H(\text{loaded-OS})$ and EK^{pub}

Problems to solve:

- OS versioning
- Remote attestation on each message (what about reboot ?)
- not only OS on platform (SW stacks or trees)
- Privacy: remote attestation always reveals EK^{pub}
- Black box too big
- Sealed memory



AB (Variant 2, allow OS versions)

CPU

Memory

Non-Volatile Memory:
“Endorsement Key” EK
preset by Manufacturer

Platform Configuration Registers:
OSK^{pub} used to check OS

Volatile memory:



Vendors of TRB and OS

TRB_generates key pair:

stores in TRB NV Memory: EK^{priv}

emits: EK^{pub}

TRB vendor certifies: $\{\text{"a valid EK"}, EK^{pub}\}TVK^{priv}$

OS-Vendor certifies: $\{\text{"a valid OS"}, OSK^{pub}\}OSVK^{priv}$

and digns OS-Code: $\{\text{OS-Code}\}OSK^{priv}$

serve as identifiers: EK^{pub} and OSK^{pub}



Booting & Attestation (Variant 2)

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:



$\{PCR, nonce\}_{EK^{priv}}$



AB (Variant 3, check for reboot)

Motivation:

$$\{OSK^{pub}, nonce\}_{EK^{priv}}$$
$$\{H(OS), nonce\}_{EK^{priv}}$$

always requires access to and usage of EK

Instead:

create new keypair on every reboot:

$$OS_{running}K^{priv} \quad OS_{running}K^{pub}$$



Booting (Variant 3)

Booting:

TRB checks OS- Code using some OSK_{pub}

stores OSK_{pub} in PCR

creates $OS_{running}K$ keypair

certifies: $\{ OS_{running}K_{pub}, H(OS) \}_{EK_{priv}}$



Attestation (Variant 3)

Attestation:

Challenge: nonce

OS generates response:

$\{ \text{OSrunning}K^{\text{pub}}, H(\text{OS}) \} E_{K^{\text{priv}}}$

$\{ \text{nonce} \} \text{OSrunning}K^{\text{priv}}$



Attestation:

Challenge: nonce

OS generates response:

$\{ \text{OSrunningK}^{\text{pub}}, H(\text{OS}) \} \text{EK}^{\text{priv}}$

$\{ \text{nonce} \} \text{OSrunningK}^{\text{priv}}$

use OSrunningK keypair to establish secure channel



Assumptions

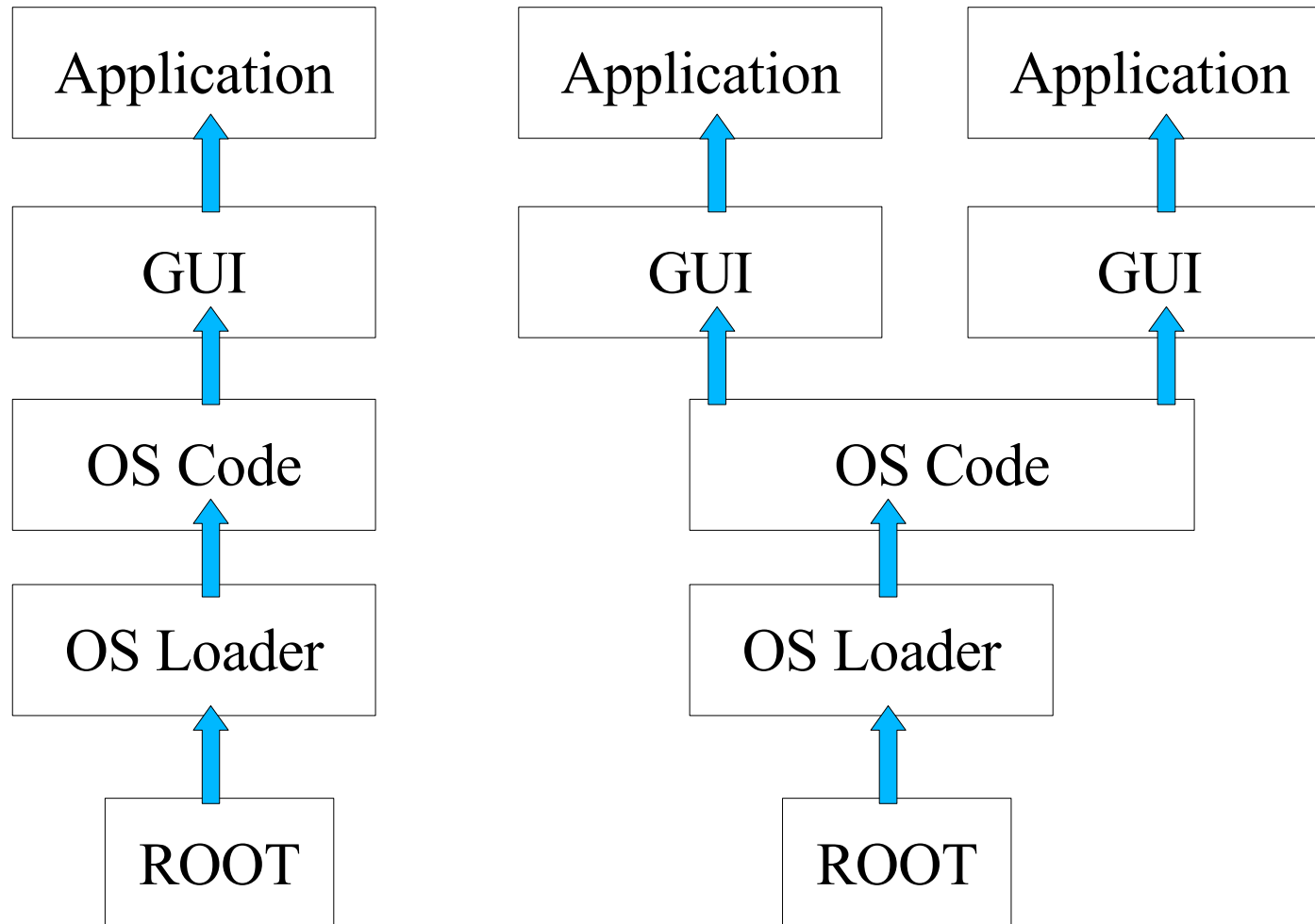
TRB can protect: EK, PCR

OS can protect: OSrunningK^{priv}

Rebooting destroys content of

- PCR and Memory Holding OSrunningK^{priv}

Software stacks and trees





Software stacks and trees

“Extend” Operation

- stack: $PCR_n = H(PCR_{n-1} || value)$
- tree: difficult (unpublished ?)

Key pairs:

- OS controls applications ->
generate key pair per application
- OS certifies
{ Application 1, App1K^{pub} } OSrunningK^{priv}
{ Application 2, App2K^{pub} } OSrunningK^{priv}



Remote Attestation and Privacy

Remote attestation reveals platform identity: EK^{pub}

add intermediate step:

- Attestation Identity Key (AIK)
- Trusted third party as anonymizer (TTP)



Remote Attestation and Privacy

CPU

Memory

Non-Volatile Memory:

EK preset by Manufacturer

AIK signed by third party

Platform Configuration Registers:

Volatile memory:



Remote Attestation and Privacy

Generate AIK in TRB

send $\{ \text{AIK} \} \text{EK}^{\text{priv}}$ to trusted third party

third party certifies: $\{ \text{AIK}, \text{“good ID”} \} \text{TTPK}^{\text{priv}}$

AIK used instead of EK during remote attestation,
response:

$\{ \text{AIK}, \text{“good ID”} \} \text{TTPK}^{\text{priv}}$

$\{ \text{OSrunningK}^{\text{pub}}, \text{H(OS)} \} \text{AIK}^{\text{priv}}$

$\{ \text{nonce} \} \text{OSrunningK}^{\text{priv}}$



Late Launch

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

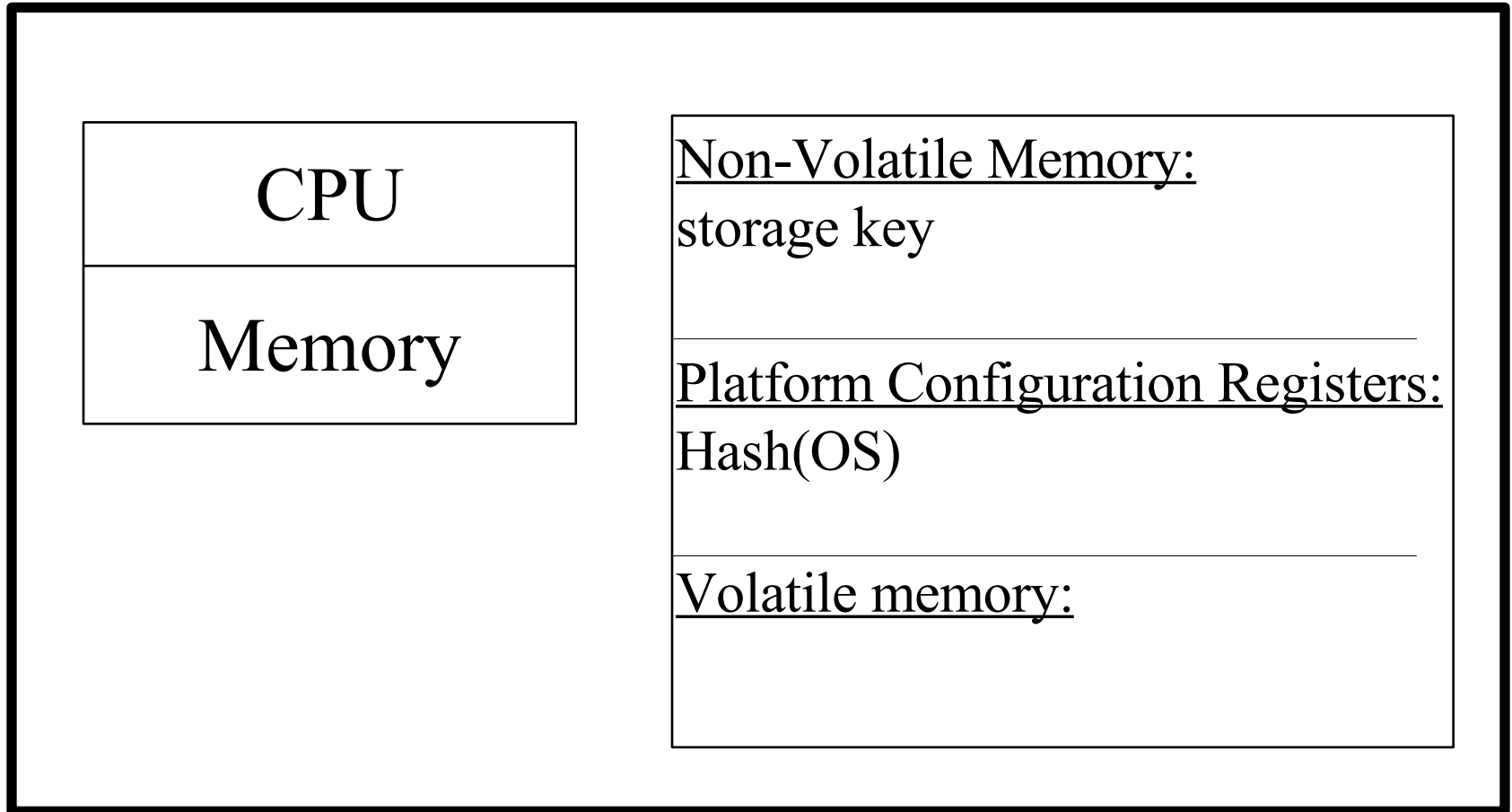
Bind sensitive information to specific configuration
(for example: keys to specific machine, specific OS)

Provide information using secure channels

How to store information in the absence of
communication channels?



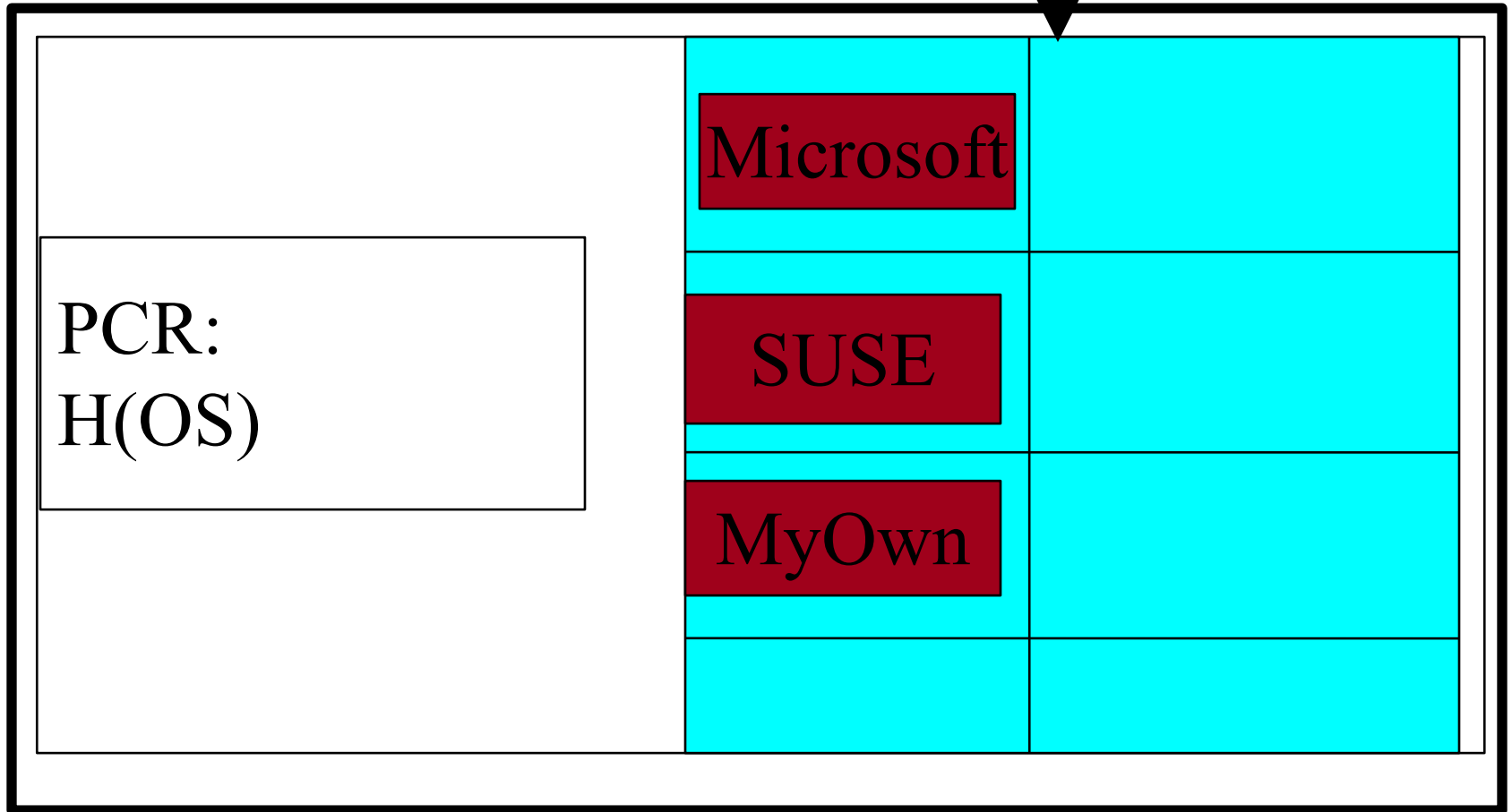
Tamperresistant black box (TRB)



Sealed Memory

Tamperresistant black box

add/delete entry
read
write





Sealed Memory

Seal(SW config, message):

- encrypt("SW config, message", Storage-Key)

Unseal(sealed message):

- decrypt("sealed message", Storage-Key) -> "SW config, message"
- If SW config == PCR then emit message else abort



Migration ?

How to transfer information from one TRB to another
for example: key for decryption of videos

- Send information to third party
- Destroy information locally and prove to third party
- Thirds party provides information to another entity



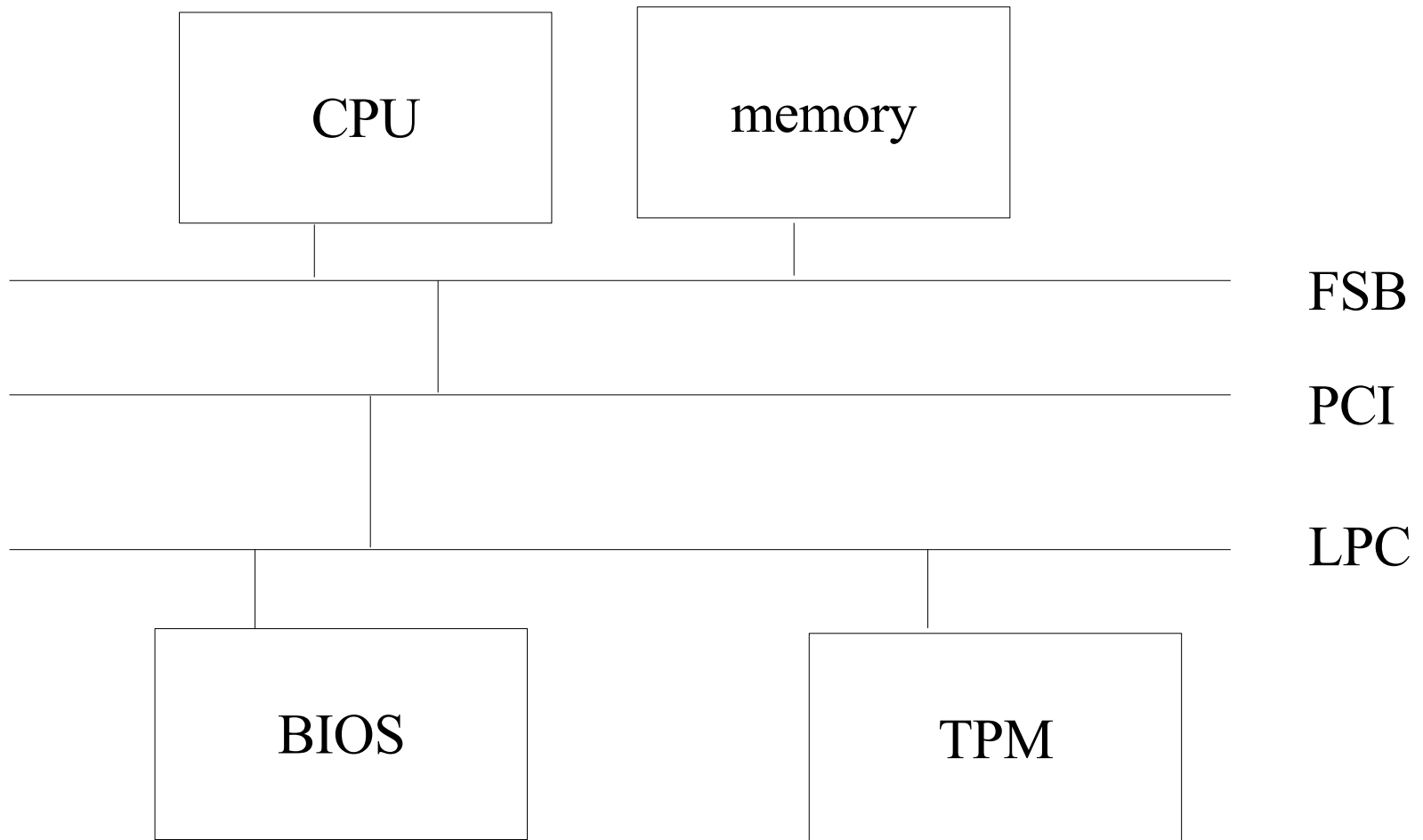
Tamper Resistant Box ?

IBM 4758 ...

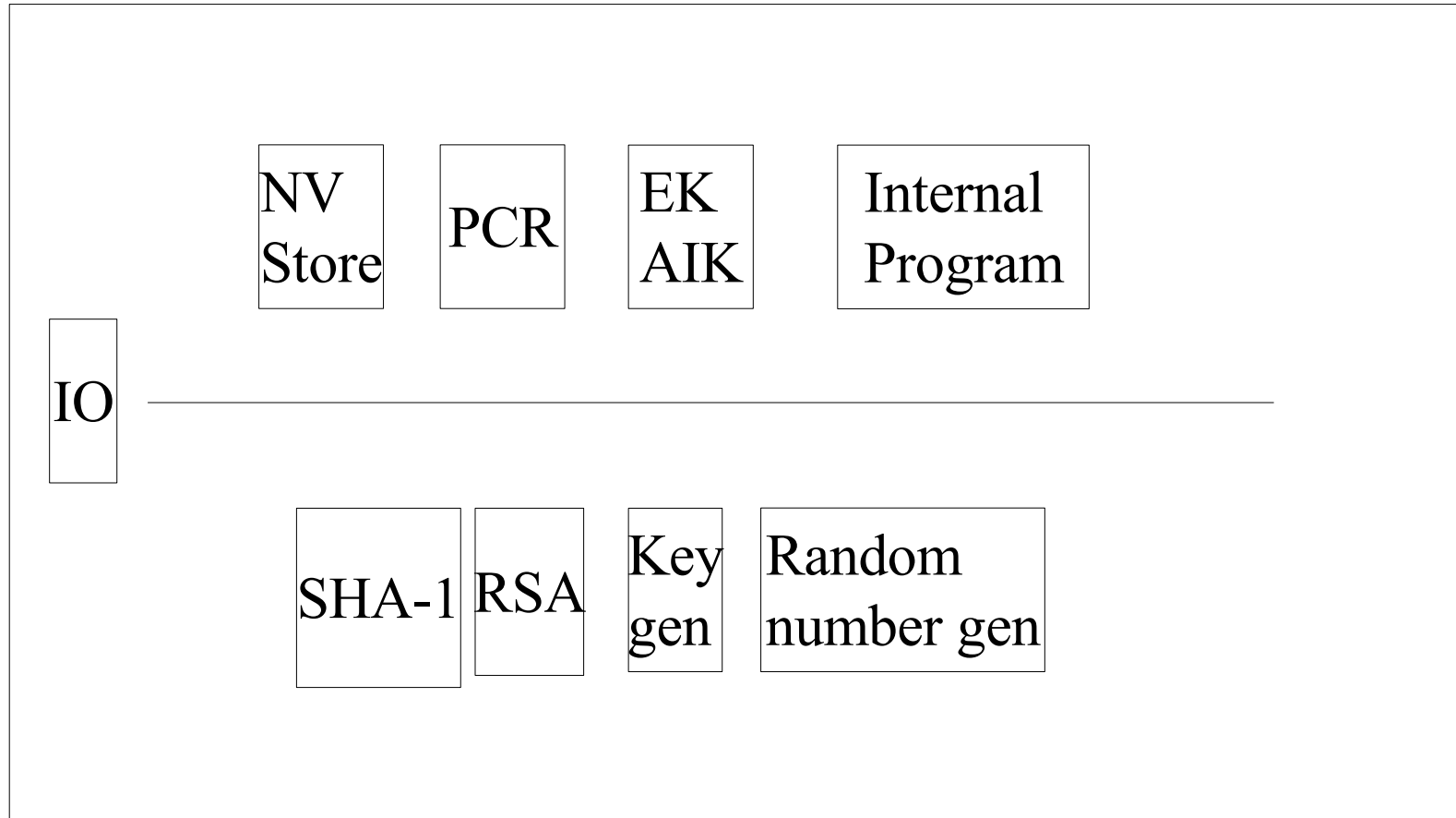
“Trusted Platform Modules”



TCG PC Platforms



TPM





References

Specifications:

[https://www.trustedcomputinggroup.org/
groups/TCG_1_3_Architecture_Overview.pdf](https://www.trustedcomputinggroup.org/groups/TCG_1_3_Architecture_Overview.pdf)

Important Foundational Paper:

Authentication in distributed systems: theory and practice

Butler Lampson, Martin Abadi, Michael Burrows, Edward Wobber

ACM Transactions on Computer Systems (TOCS)