TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute for System Architecture, Operating Systems Group

TRUSTED COMPUTING

CARSTEN WEINHOLD

THIS LECTURE ...

- **Today: Trusted Computing Technology**
  - Lecture discusses basics in context of TPMs
  - More theoretical concepts also covered in lecture „Distributed Operating Systems“
- **Things you should have heard about:**
  - How asymmetric encryption is used
  - What a digital signature is
  - What a cryptographic hash function is

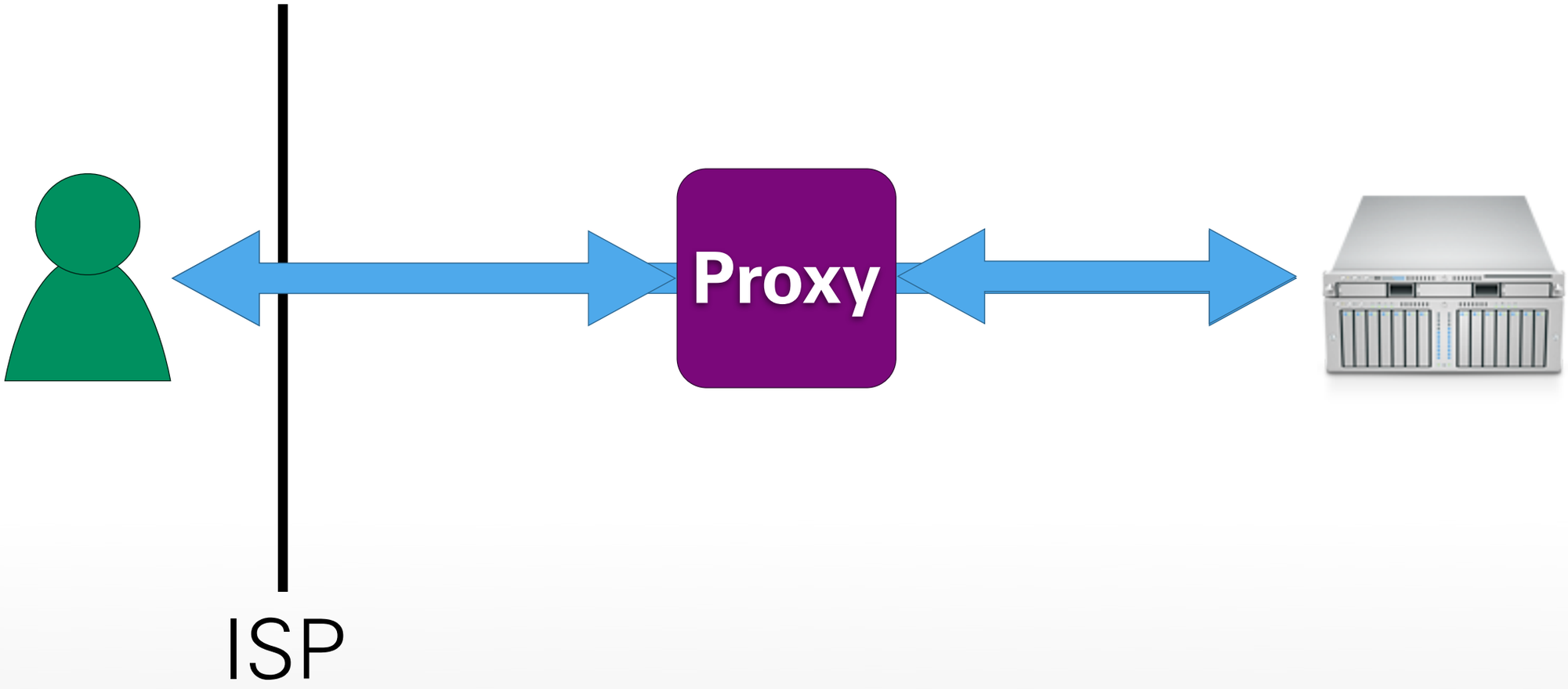
INTRODUCTION

AN.ON

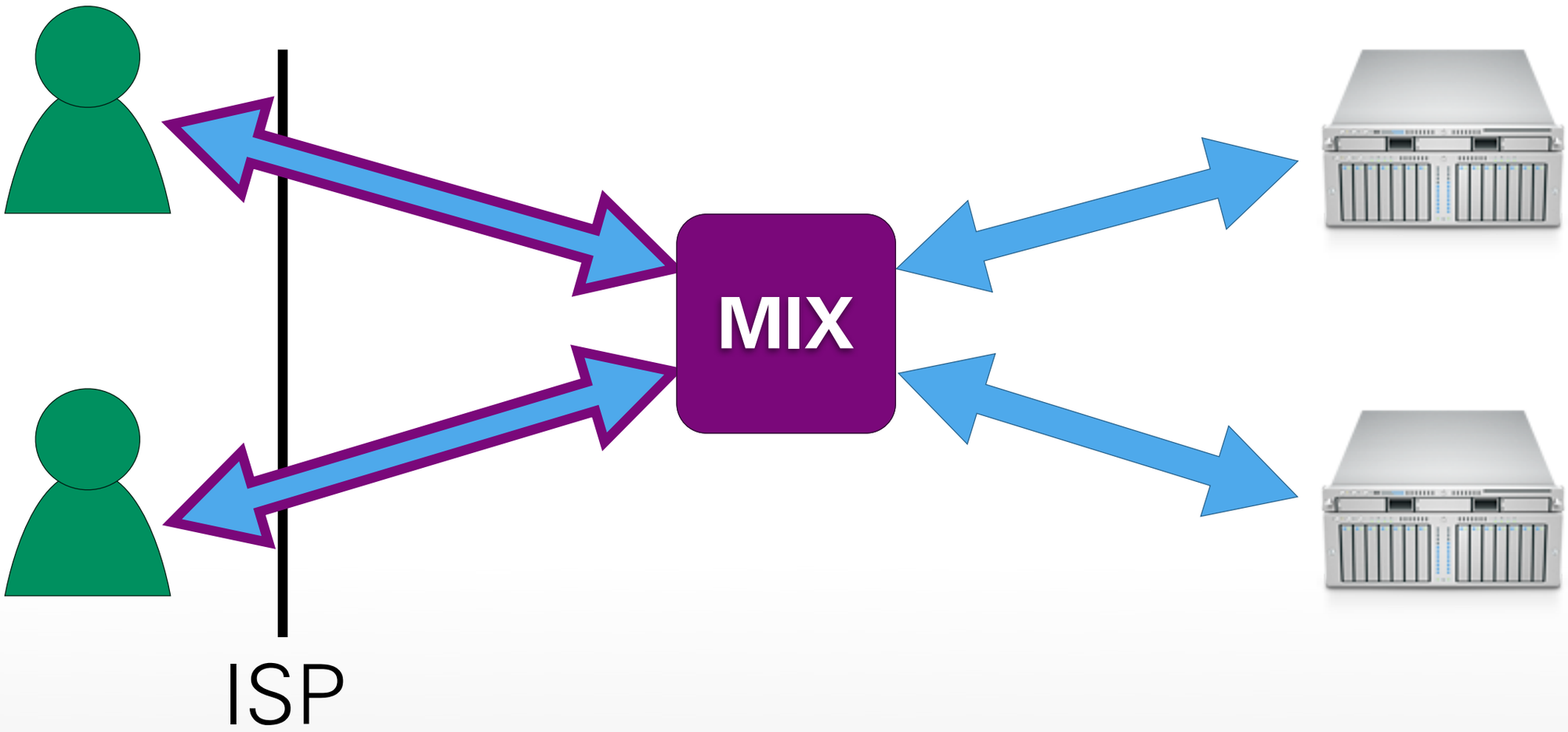
L4

TPM

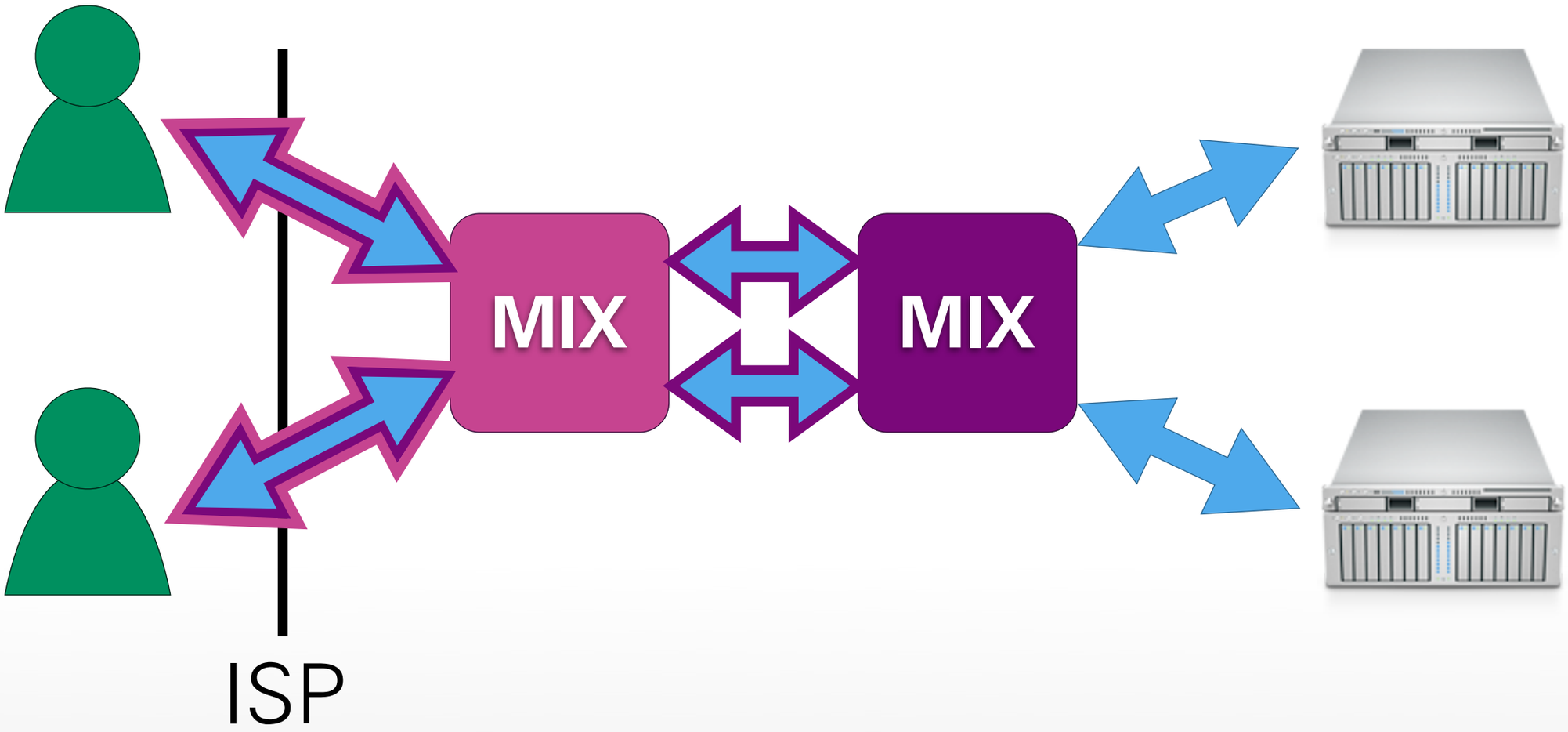
ANONYMITY



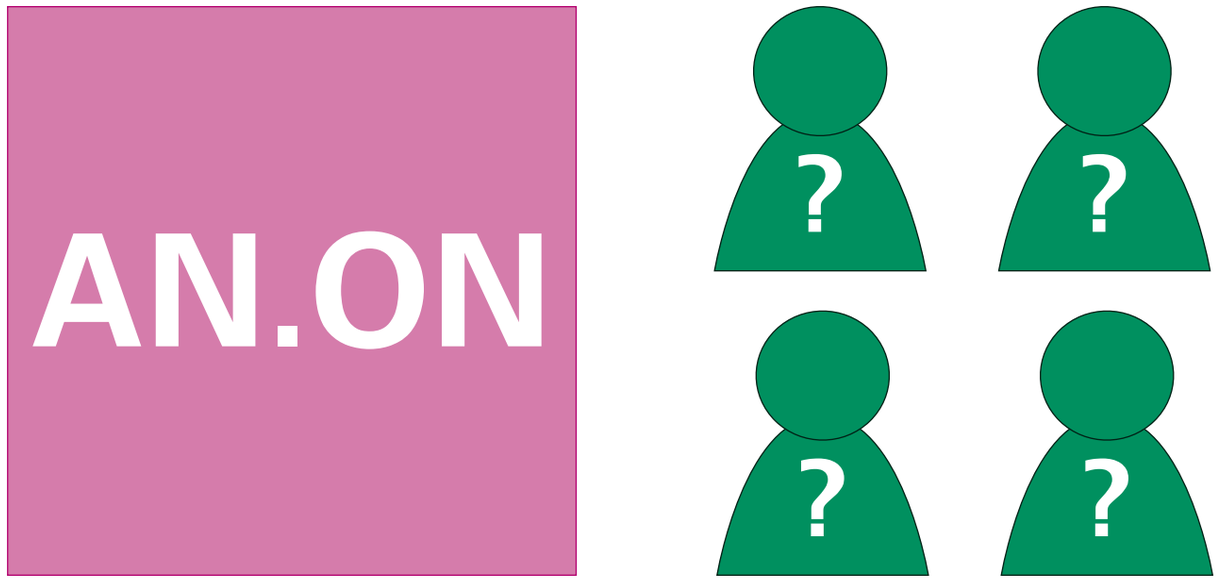
ANONYMITY



ANONYMITY



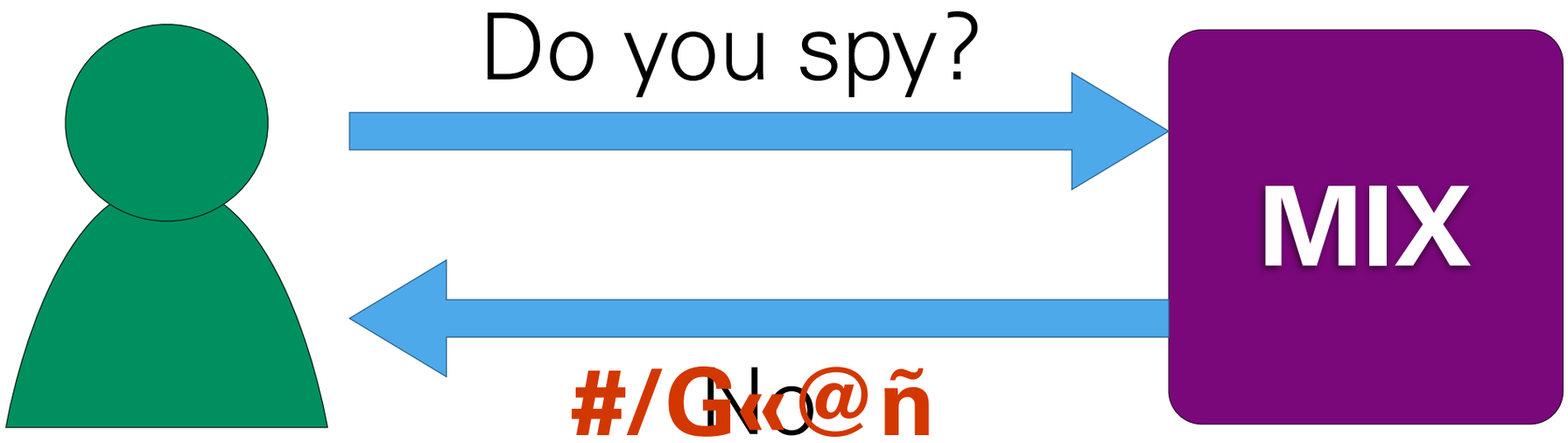
ANONYMITY



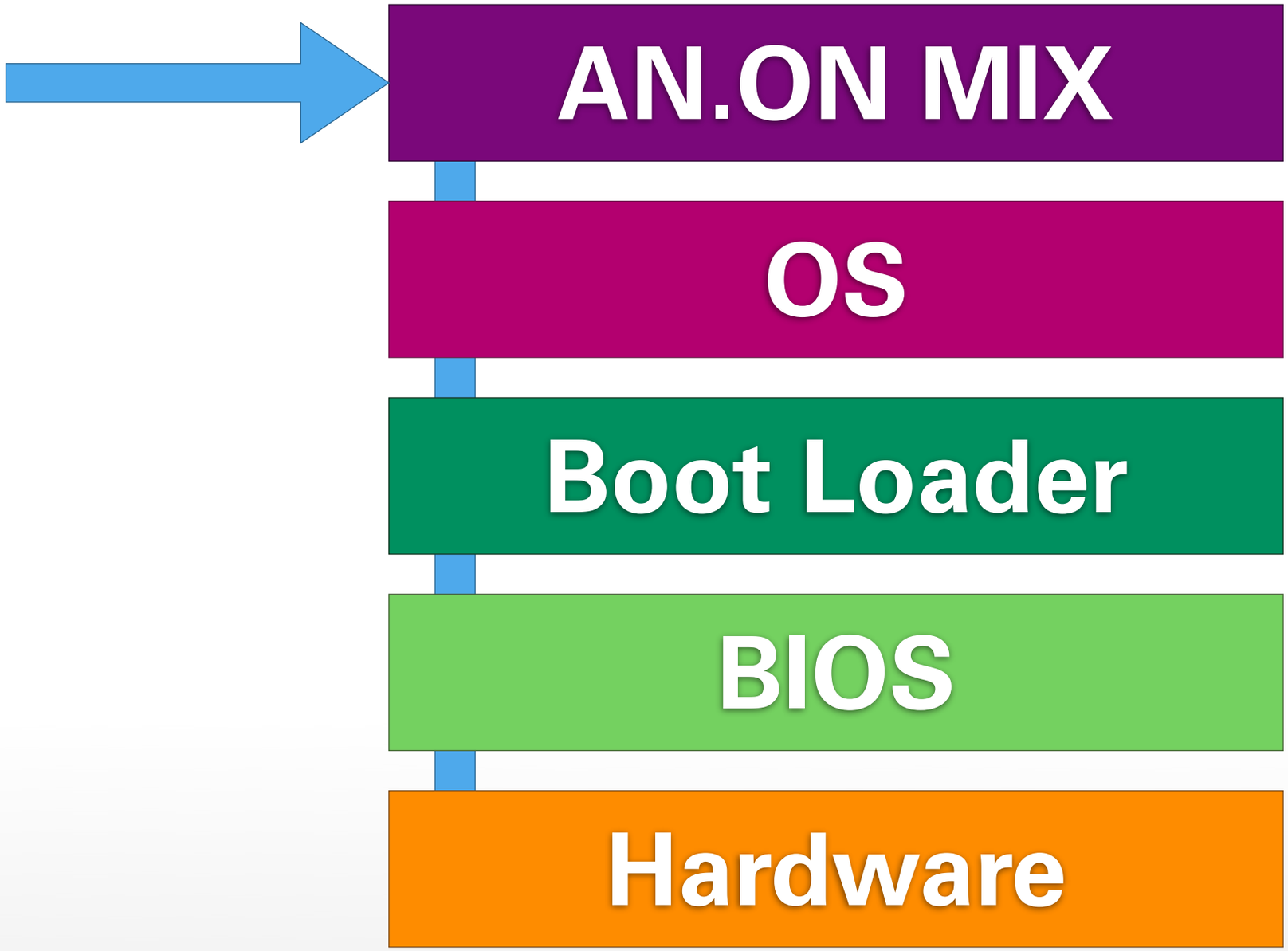
PROBLEM

- Last proxy sees data in **plaintext**
- Often no additional end-to-end encryption
- Ideal for password **phishing**
- TOR: increasing number of exit nodes in China, Russia, USA
- Dan Egerstad [1]: 100 passwords sniffed with 5 exit nodes

IDEA



SYSTEM LAYERS




Infineon
TPM

Platform Configuration Register

PCR := SHA-1(PCR | **X**)

BOOTING + TPM

AN.ON MIX

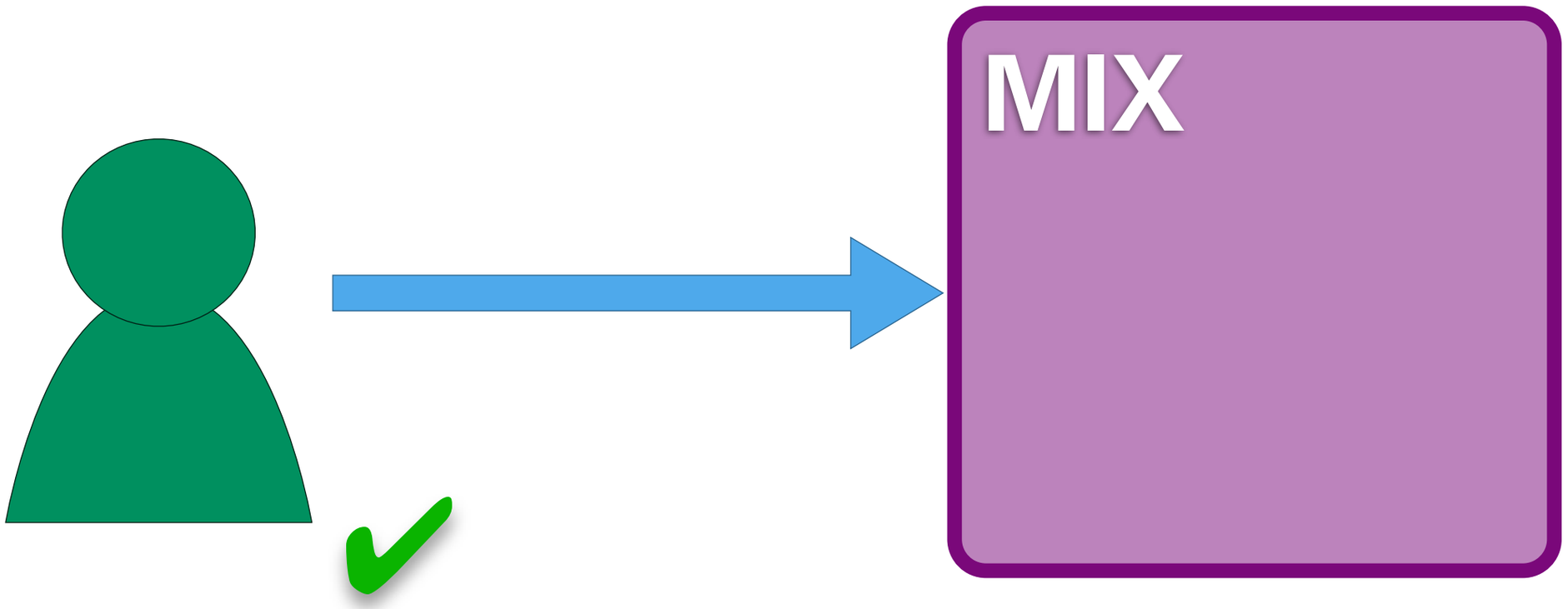
OS

Boot Loader

BIOS

PCR

Remote Attestation



ARCHITECTURE

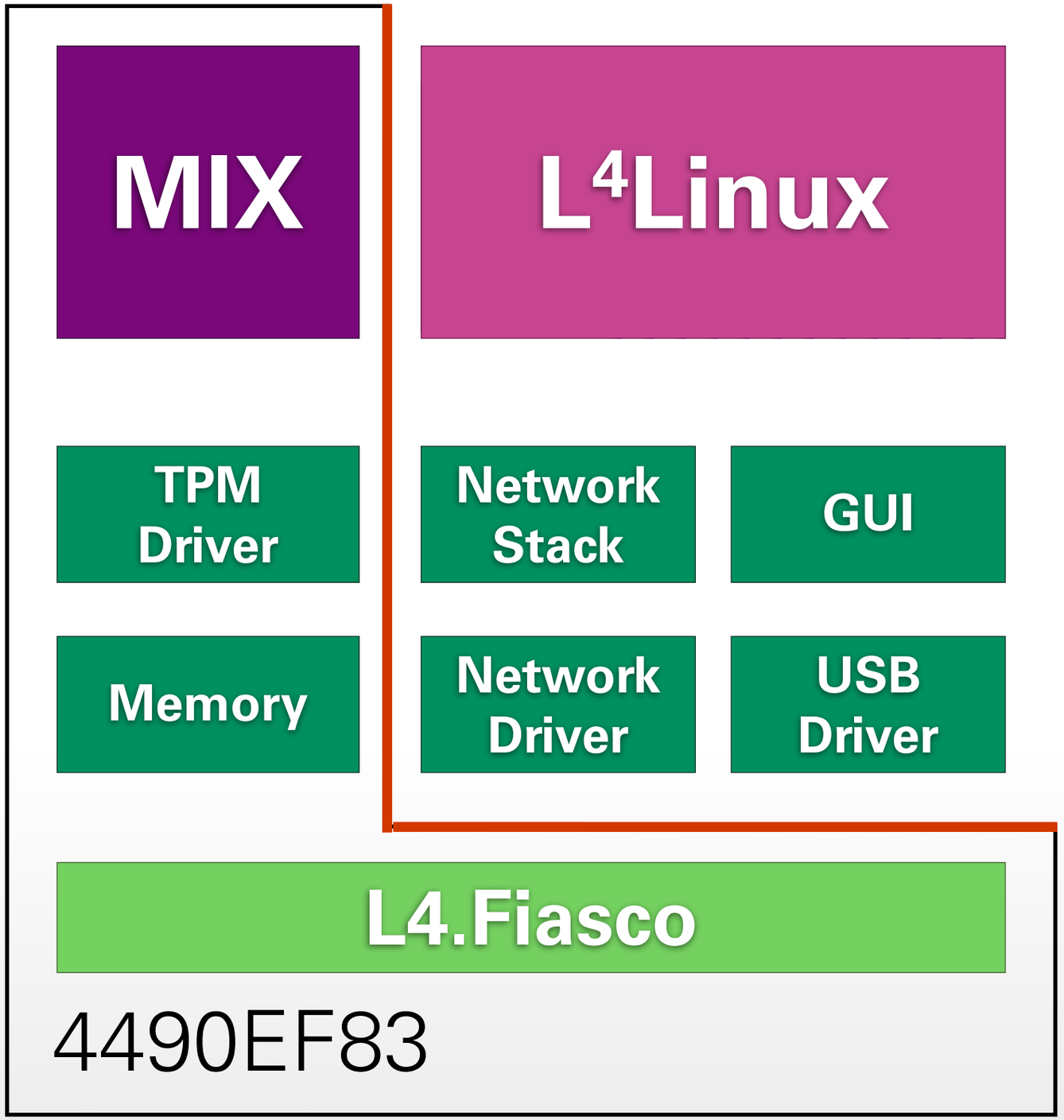
AN.ON

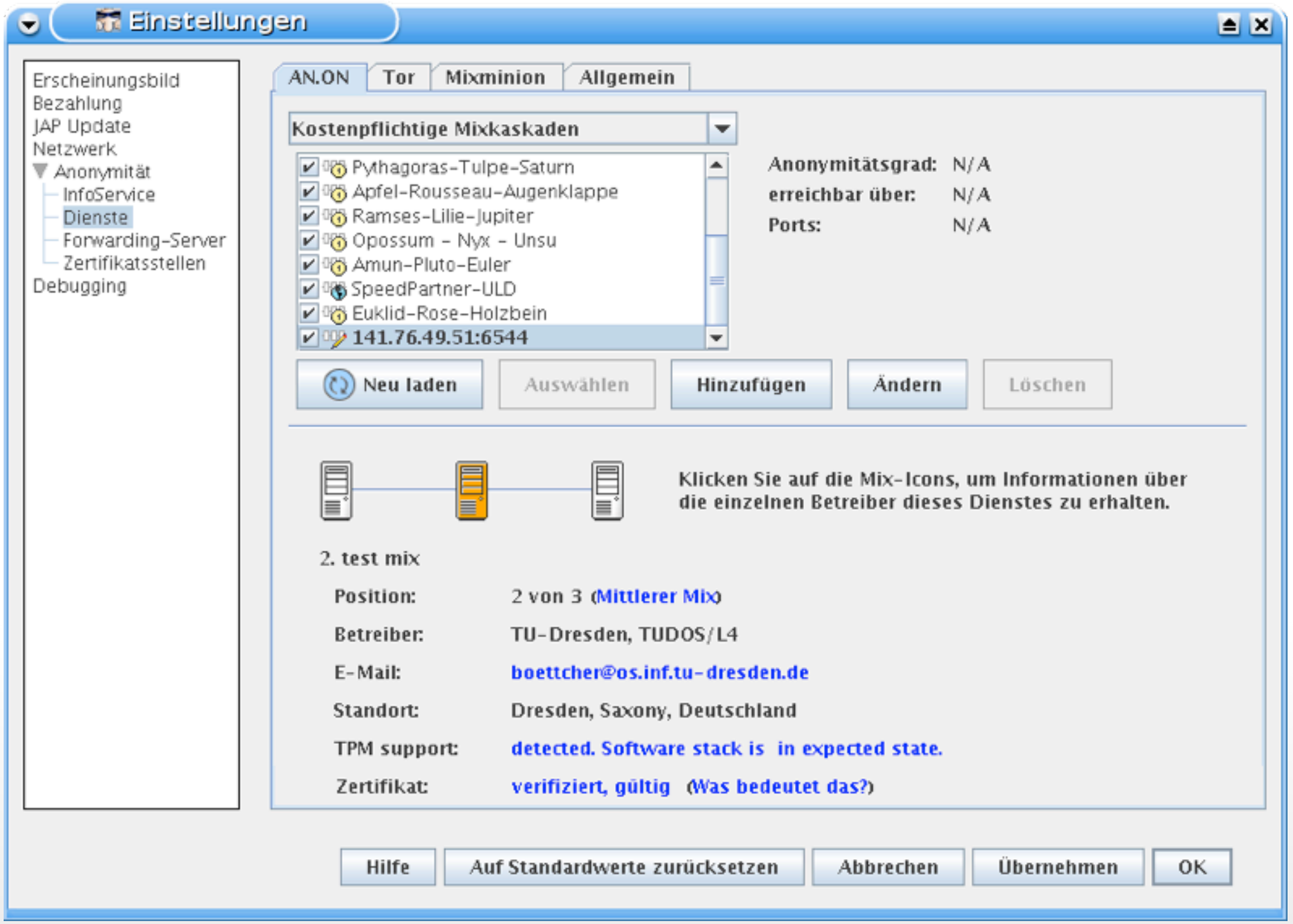
Linux

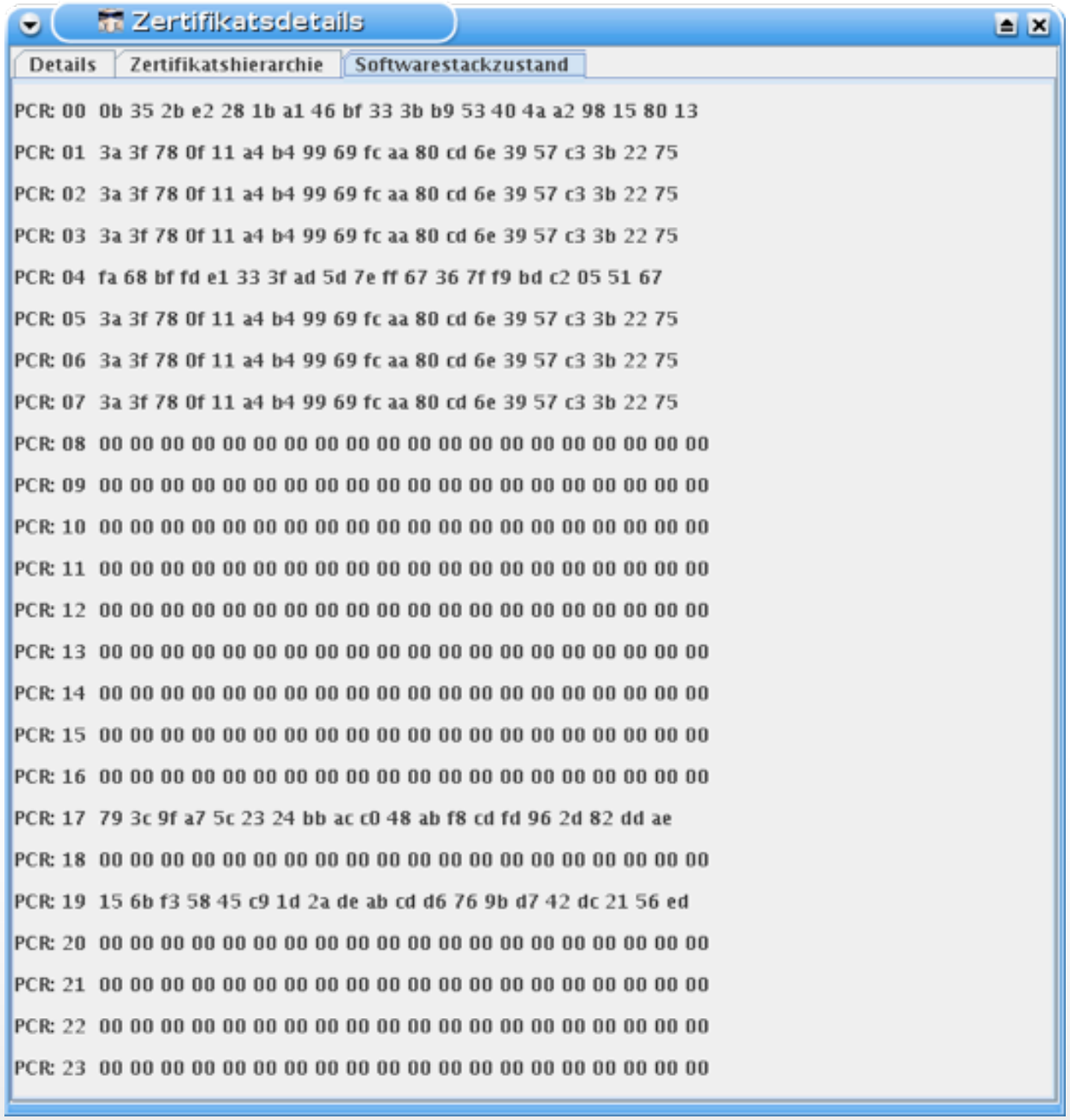
Windows

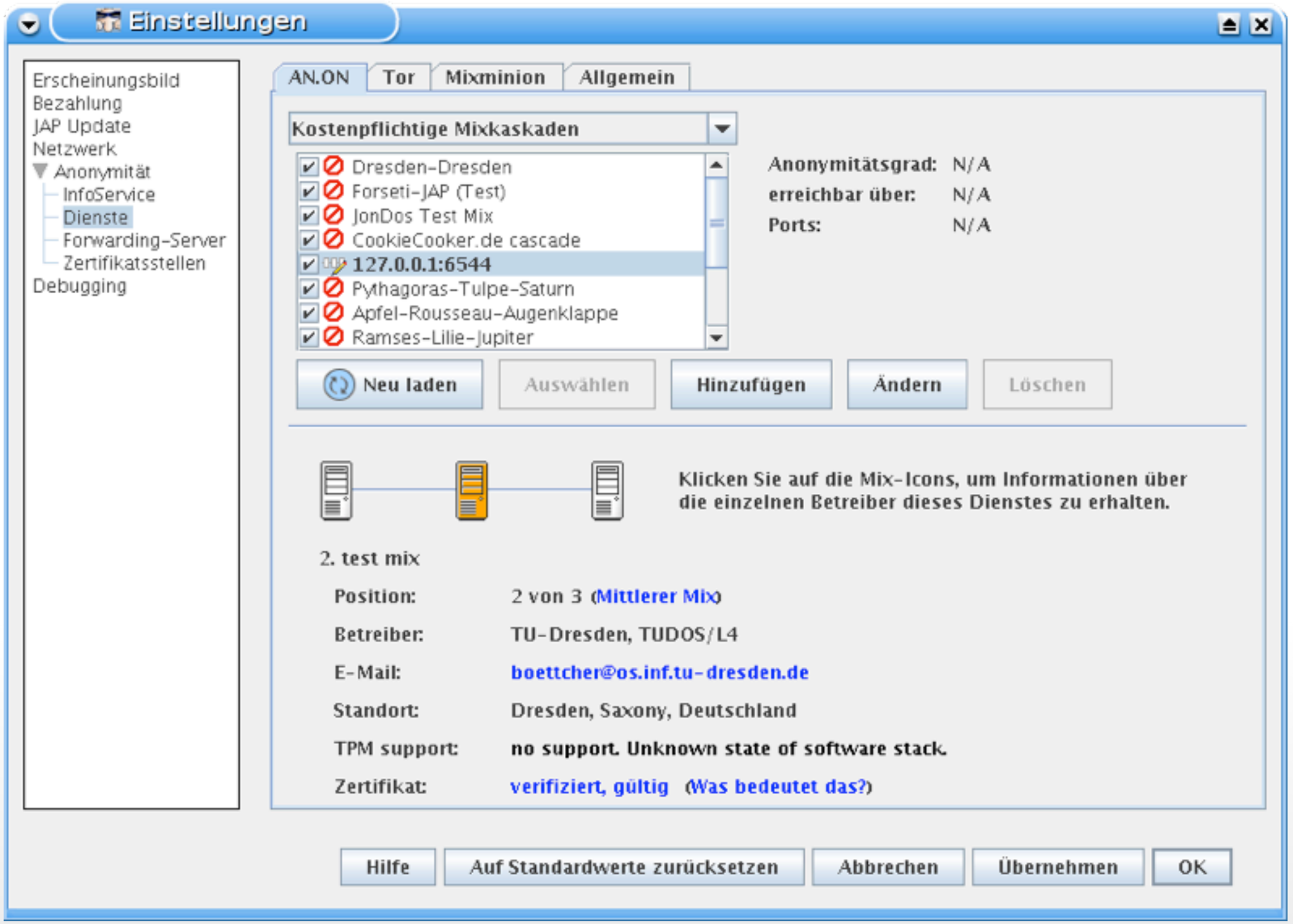
TPM

MONOLITHIC


MIX
L⁴Linux
TPM Driver
Network Stack
GUI
Memory
Network Driver
USB Driver
L4.Fiasco
4490EF83


Einstellungen
Erscheinungsbild
Bezahlung
JAP Update
Netzwerk
Anonymität
InfoService
Dienste
Forwarding-Server
Zertifikatsstellen
Debugging
AN.ON
Tor
Mixminion
Allgemein
Kostenpflichtige Mixkaskaden
Pythagoras-Tulpe-Saturn
Apfel-Rousseau-Augenklappe
Ramses-Lilie-Jupiter
Opossum - Nyx - Unsu
Amun-Pluto-Euler
SpeedPartner-ULD
Euklid-Rose-Holzbein
141.76.49.51:6544
Anonymitätsgrad: N/A
erreichbar über: N/A
Ports: N/A
Neu laden
Auswählen
Hinzufügen
Ändern
Löschen
Klicken Sie auf die Mix-Icons, um Informationen über die einzelnen Betreiber dieses Dienstes zu erhalten.
2. test mix
Position: 2 von 3 (Mittlerer Mix)
Betreiber: TU-Dresden, TUDOS/L4
E-Mail: boettcher@os.inf.tu-dresden.de
Standort: Dresden, Saxony, Deutschland
TPM support: detected. Software stack is in expected state.
Zertifikat: verifiziert, gültig (Was bedeutet das?)
Hilfe
Auf Standardwerte zurücksetzen
Abbrechen
Übernehmen
OK


Zertifikatsdetails
Details
Zertifikatshierarchie
Softwarestackzustand
PCR: 00 0b 35 2b e2 28 1b a1 46 bf 33 3b b9 53 40 4a a2 98 15 80 13
PCR: 01 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 02 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 03 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 04 fa 68 bf fd e1 33 3f ad 5d 7e ff 67 36 7f f9 bd c2 05 51 67
PCR: 05 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 06 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 07 3a 3f 78 0f 11 a4 b4 99 69 fc aa 80 cd 6e 39 57 c3 3b 22 75
PCR: 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 17 79 3c 9f a7 5c 23 24 bb ac c0 48 ab f8 cd fd 96 2d 82 dd ae
PCR: 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 19 15 6b f3 58 45 c9 1d 2a de ab cd d6 76 9b d7 42 dc 21 56 ed
PCR: 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR: 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


Einstellungen
Erscheinungsbild
Bezahlung
JAP Update
Netzwerk
Anonymität
InfoService
Dienste
Forwarding-Server
Zertifikatsstellen
Debugging
AN.ON
Tor
Mixminion
Allgemein
Kostenpflichtige Mixkaskaden
Dresden-Dresden
Forseti-JAP (Test)
JonDos Test Mix
CookieCooker.de cascade
127.0.0.1:6544
Pythagoras-Tulpe-Saturn
Apfel-Rousseau-Augenklappe
Ramses-Lilie-Jupiter
Anonymitätsgrad: N/A
erreichbar über: N/A
Ports: N/A
Neu laden
Auswählen
Hinzufügen
Ändern
Löschen
Klicken Sie auf die Mix-Icons, um Informationen über die einzelnen Betreiber dieses Dienstes zu erhalten.
2. test mix
Position: 2 von 3 (Mittlerer Mix)
Betreiber: TU-Dresden, TUDOS/L4
E-Mail: boettcher@os.inf.tu-dresden.de
Standort: Dresden, Saxony, Deutschland
TPM support: no support. Unknown state of software stack.
Zertifikat: verifiziert, gültig (Was bedeutet das?)
Hilfe
Auf Standardwerte zurücksetzen
Abbrechen
Übernehmen
OK

AN.ON

L4

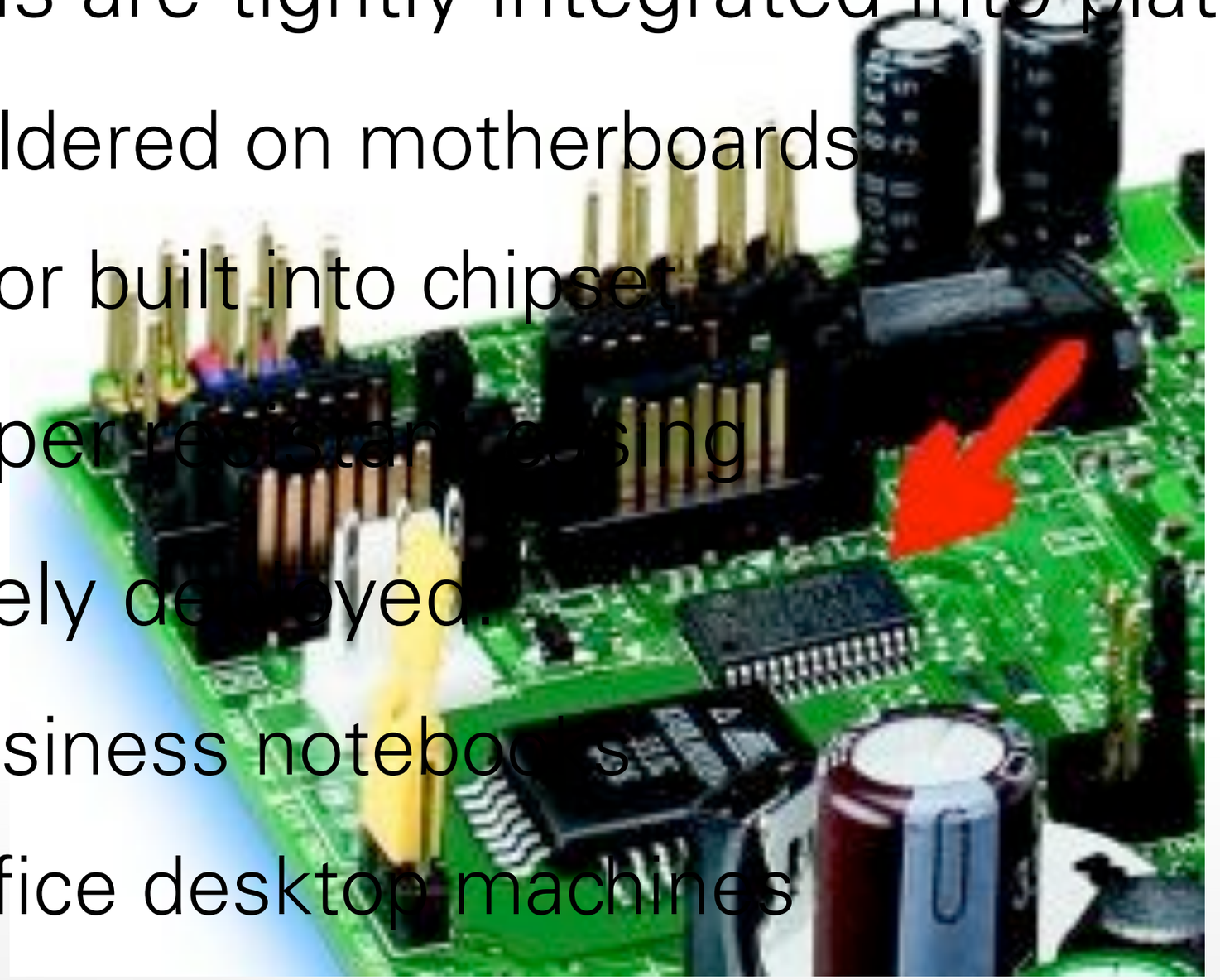
TPM

TECHNISCHE UNIVERSITÄT DRESDEN

THE TRUSTED PLATFORM MODULE

TPM HARDWARE

- TPMs are tightly integrated into platform:
  - Soldered on motherboards
  - ... or built into chipset
- Tamper resistant packaging
- Widely deployed:
  - Business notebooks
  - Office desktop machines
  - Some consumer notebooks

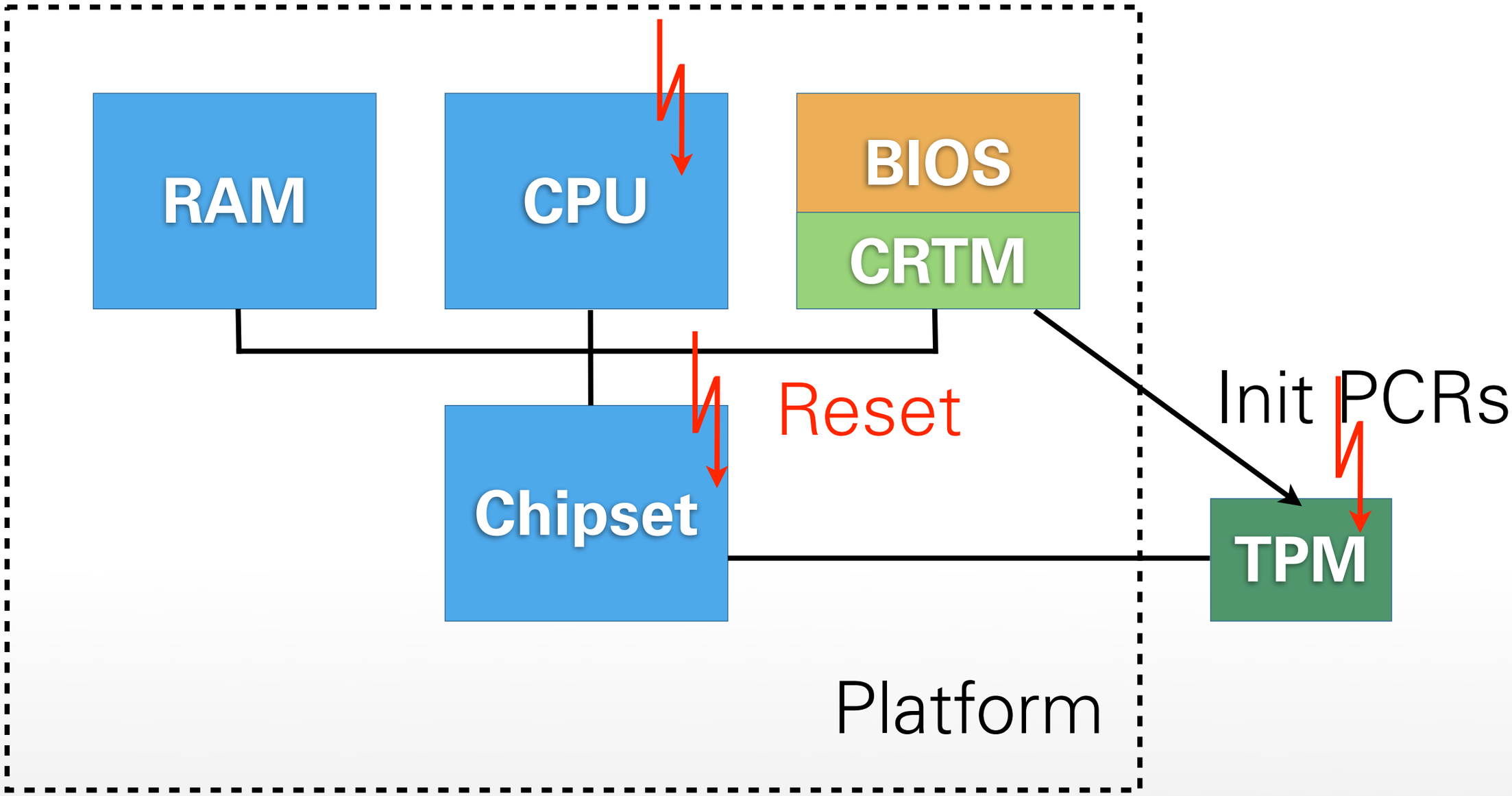


http://www.heise.de/bilder/61155/0/0

TPM OVERVIEW

- TPM is cryptographic coprocessor:
  - **RSA** (encryption, signatures), **AES** (encryption), **SHA-1** (cryptographic hashes)
  - Other crypto schemes (e.g., **DAA**)
  - Random number generator
  - Platform Configuration Registers (**PCRs**)
  - Non-volatile memory
- TPMs are passive devices!

TPM & PLATFORM



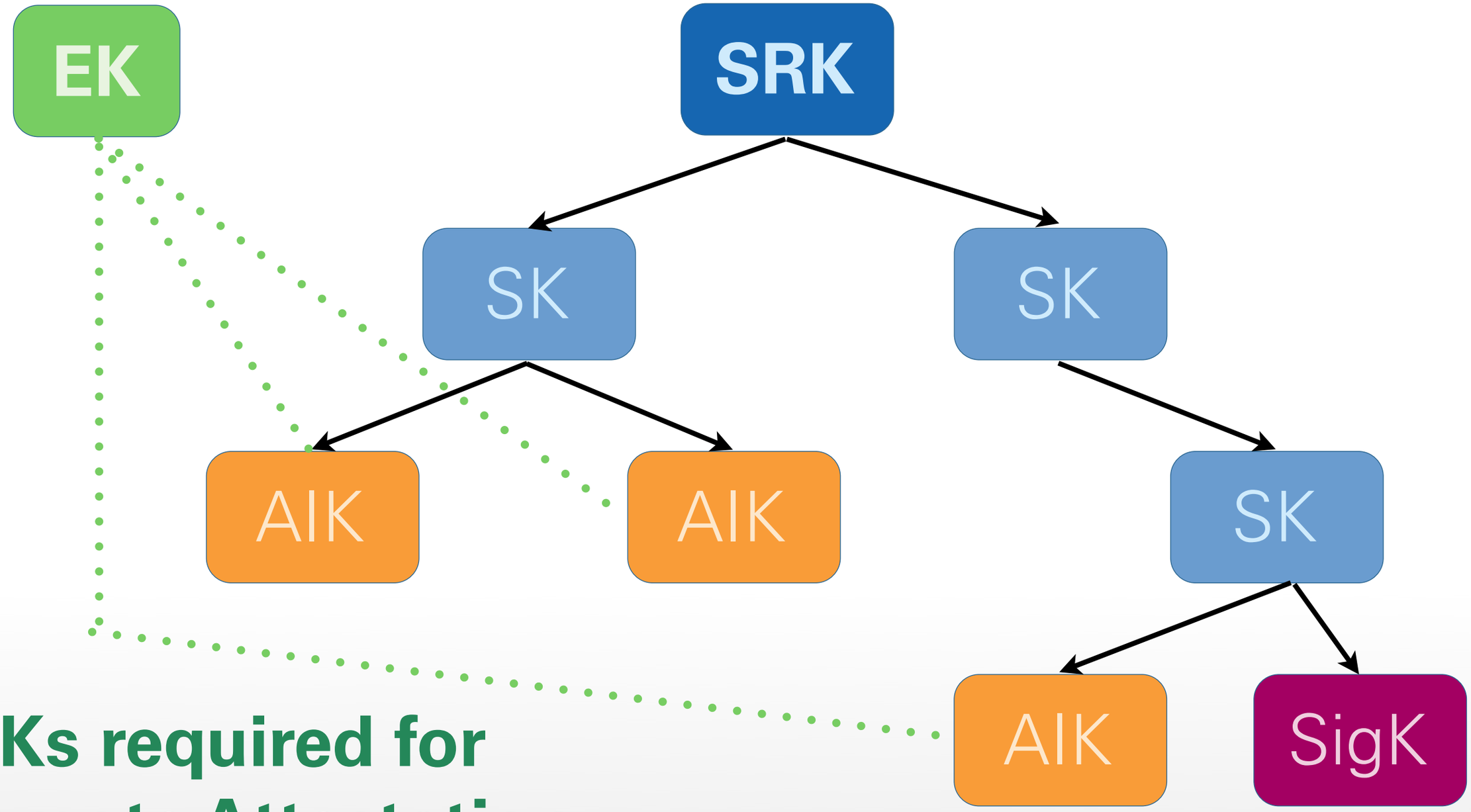
TPM SPECS

- TPMs specified by Trusted Computing Group [2]
- Multiple hardware implementations
- TPM specifications [3,4] cover:
  - Architecture, interfaces, security properties
  - Data formats of input / output
  - Schemes for signatures, encryption, ...
  - TPM life cycle, platform requirements

TPM IDENTITY

- TPM identified by Endorsement Key **EK**:
  - Generated in manufacturing process
  - Certified by manufacturer
  - Root of signatures issued by TPM
    - Unique among all TPMs
- Creating entirely new **EK** possible (e.g., for use in corporate environments)
- Private part of **EK** never leaves TPM

KEY HIERARCHY



AIKs required for Remote Attestation

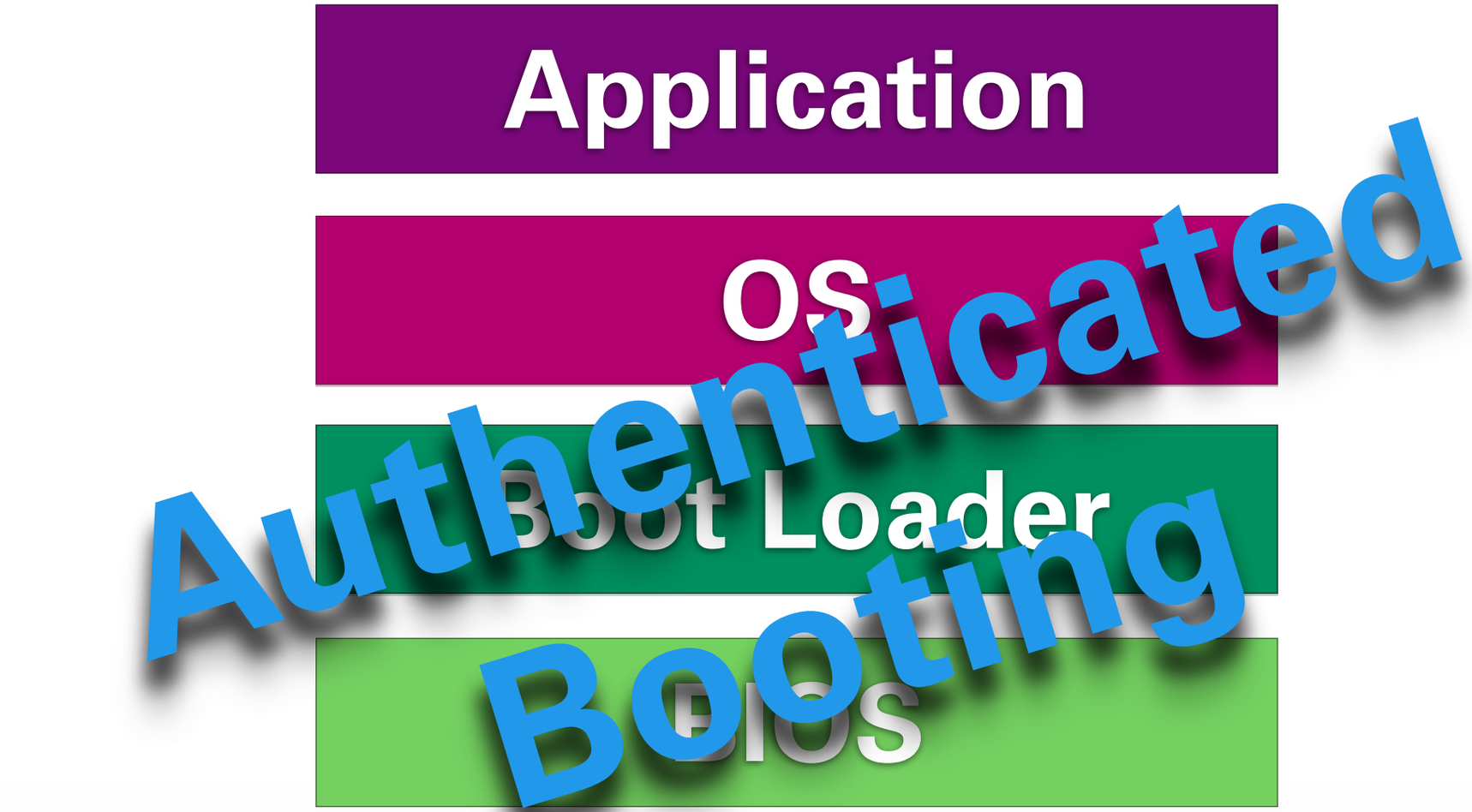
KEY HIERARCHY

- All keys except for **EK** are part of key hierarchy below Storage Root Key **SRK**:
  - **SRK** created when user „takes ownership“
  - Key types: **storage**, **signature**, **identity**, ...
- Storage keys are parent keys at lower levels of hierarchy (like **SRK** does at root level)
- Keys other than **EK** / **SRK** can leave TPM:
  - Encrypted under parent key before exporting
  - Parent key required for loading and decrypting

AIK

- Special key type for remote attestation: Attestation Identity Key (**AIKs**):
  - Created locally by TPM
  - Encrypted under **EK** and sent to privacy CA
  - Privacy CA issues certificates for **AIKs** based on **EK** and **PCR** configuration
- **AIK** certificate:
  - „This **AIK** has been created by a valid TPM“
  - TPM identity (**EK**) cannot be derived from it

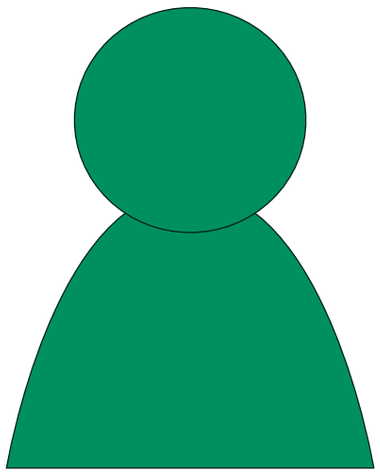
BOOTING + TPM



PCR

AIKS & QUOTES

TPM_Quote(AIK, Nonce, PCR)



Challenger

System

Remote Attestation

AE58B991

SEALED MEMORY

- Applications require secure storage
- TPMs can lock data to **PCR** values:
  - **TPM_Seal()**:
    - Encrypt user data under specified storage key
    - Encrypted blob contains **expected PCR** values
  - **TPM_Unseal()**:
    - Decrypt encrypted blob using storage key
    - Compare **current** and **expected PCR** values
    - Release user data only if **PCR** values match

SEALED BLOBS

```
TPM_STORED_DATA12 {
    TPM_STRUCTURE_TAG tag;
    TPM_ENTITY_TYPE et;
    UINT32 sealInfoSize;
    TPM_PCR_INFO_LONG {
            TPM_STRUCTURE_TAG          tag;
            TPM_LOCALITY_SELECTION     localityAtCreation;
            TPM_LOCALITY_SELECTION     localityAtRelease;
            TPM_PCR_SELECTION          creationPCRSelection;
            TPM_PCR_SELECTION          releasePCRSelection;
            TPM_COMPOSITE_HASH         digestAtCreation;
            TPM_COMPOSITE_HASH         digestAtRelease;
    } sealInfo;
    UINT32 encDataSize;
    TPM_SEALED_DATA {
            TPM_PAYLOAD_TYPE           payload;
            TPM_SECRET                 authData;
            TPM_NONCE                  tpmProof;
            TPM_DIGEST                 storedDigest;
            UINT32                     dataSize;
            [size_is(dataSize)] BYTE*  data;
    } encData;
};
```

FRESHNESS

- Sealed data is stored outside the TPM
- Vulnerable to replay attacks:
  - Multiple versions of sealed blob may exist
  - Any version can be passed to TPM
    - TPM happily decrypts, if crypto checks out
- Problem:
  - What if sealed data must be current?
  - How to prevent use of older versions?

COUNTERS

- TPMs provide **monotonic counters**
- Only two operations: **inc**, **read**
- Password protected
- Prevent replay attacks:
  - Seal expected value of counter with data
  - After unseal, compare unsealed value with current counter
  - Increment counter to invalidate old versions

TPM SUMMARY

- Key functionality of TPMs:
  - Authenticated booting
  - Remote attestation
  - Sealed memory
- Problems with current TPMs:
  - No support for virtualization
  - Slow (hundreds of ms / operation)
  - Linear chain of trust

TECHNISCHE UNIVERSITÄT DRESDEN

TPMS IN NIZZA ARCHITECTURE

BOOTING + TPM

App A

App B

OS

Boot Loader

BIOS

PCR 83E2EE9A

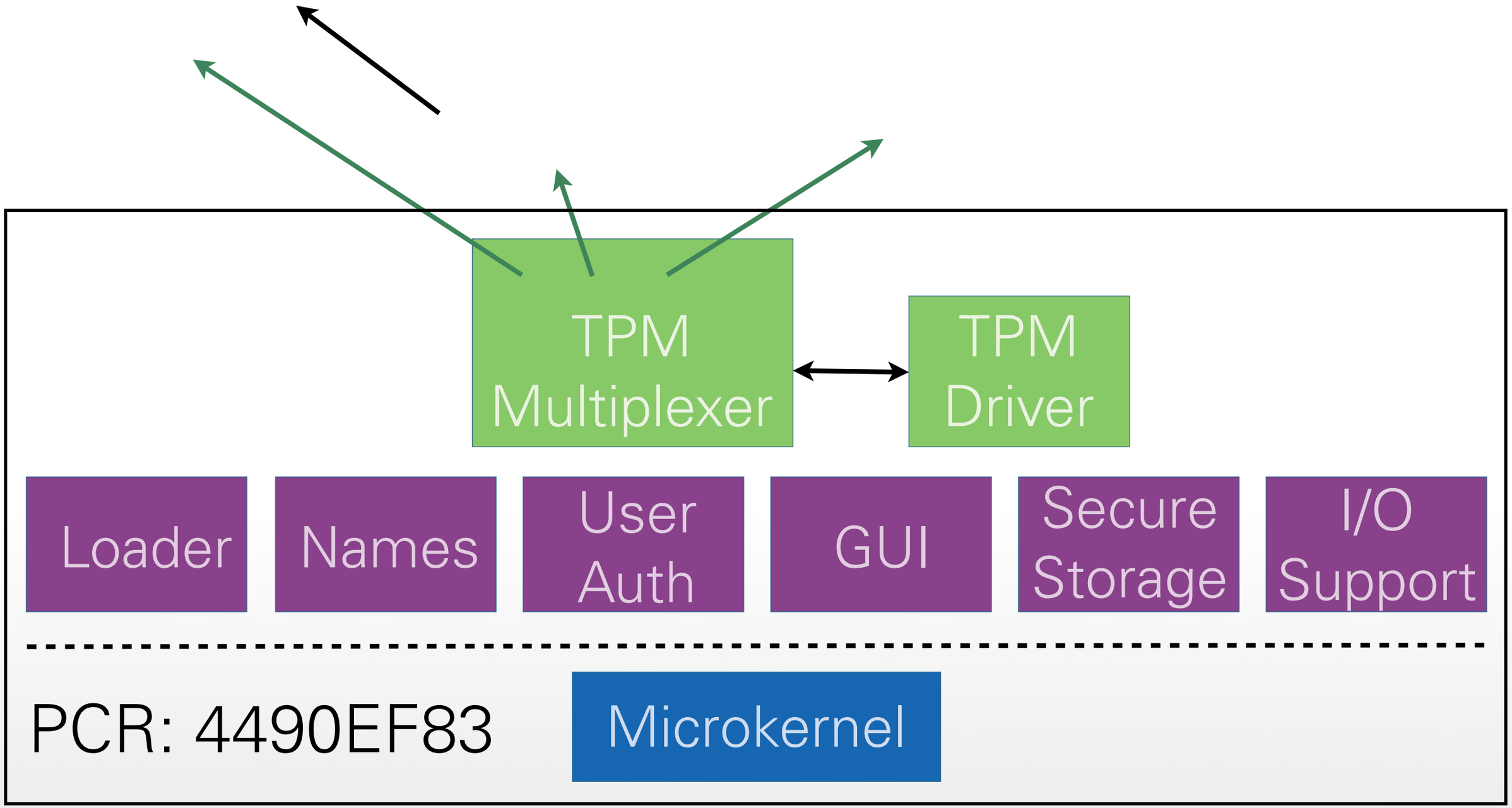
MULTIPLE APPS

- Use one PCR per application:
  - Application measurements independent
  - Number of PCRs is limited (max 24)
- Use one PCR for all applications:
  - Chain of trust / application log grows
  - All applications reported in remote attestation (raises privacy concerns)
  - All applications checked when unsealing

EXTENDING TPMS

- Idea: extend PCRs in software:
  - Measure only base system into PCRs (microkernel, basic services, TPM driver, ...)
  - „Software TPM“ provides „software PCRs“ for each application
- More flexibility with „software **PCRs**“:
  - Chain of trust common up to base system
  - Extension of chains of trust for applications fork above base system
  - Branches in **Tree of Trust** are independent

SOFTWARE PCRS

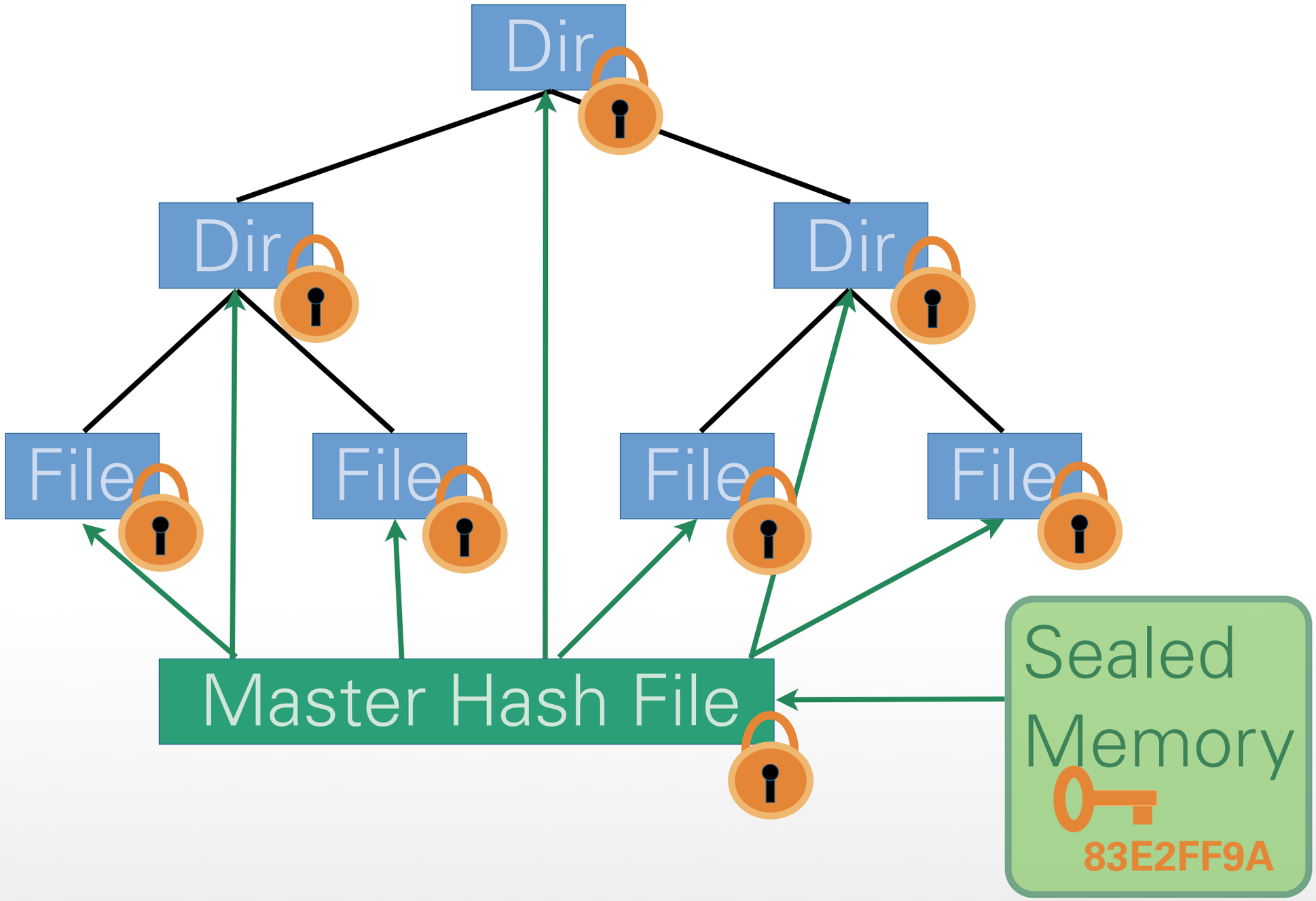


- Operations on software PCRs:
  - Seal() / Unseal(), Quote(), Extend()
  - Add_child(), Remove_child()
  - Performed using software keys (AES, RSA)
    - Software keys protected with real TPM
- Link between software **PCRs** and real **PCRs**: certificate for RSA signature key
- Implemented for L4: TPM multiplexer **Lyon**

TECHNISCHE UNIVERSITÄT DRESDEN

A SECOND LOOK AT VPFS

VPFS SECURITY



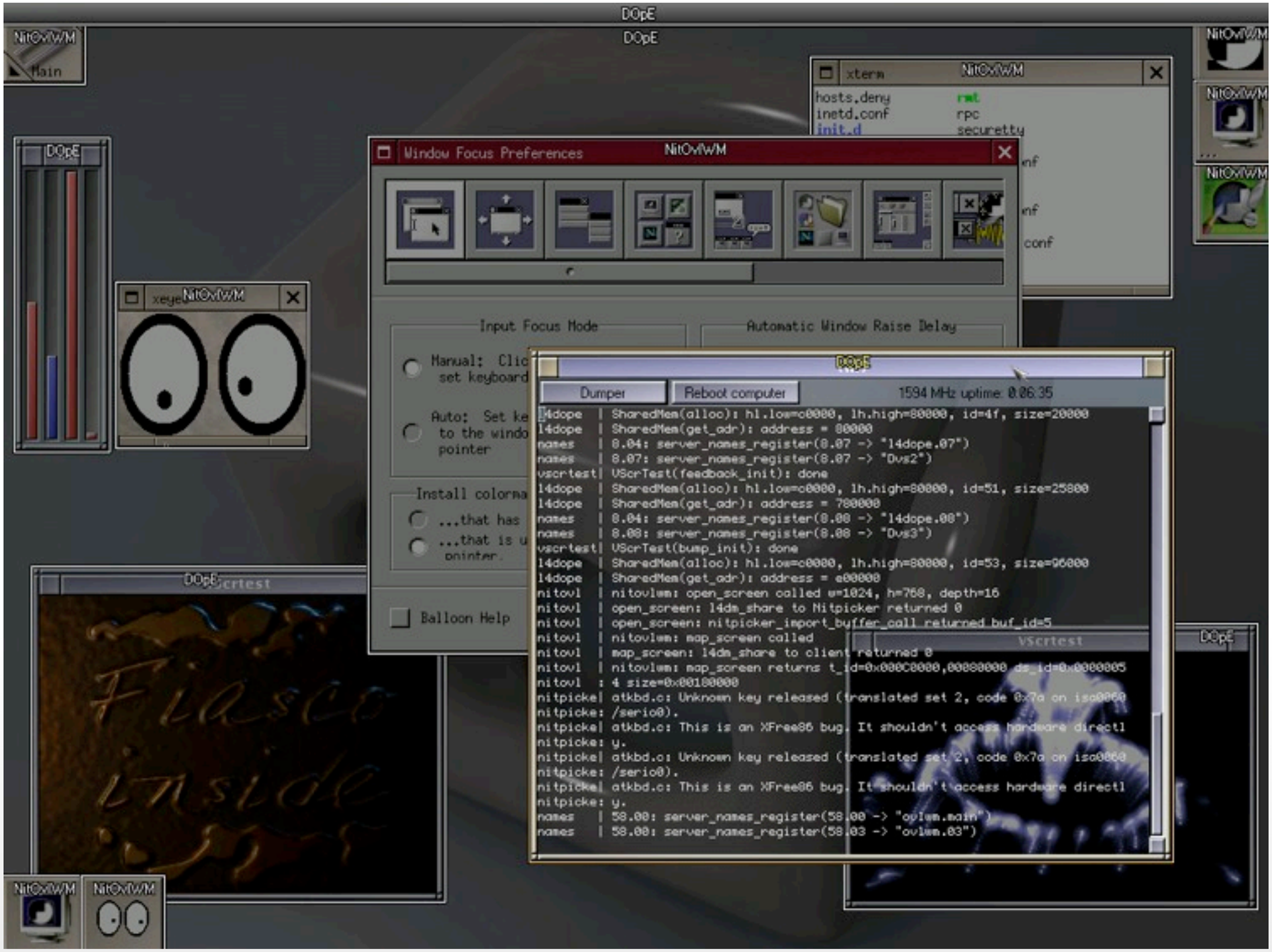
VPFS SECURITY

- VPFS uses sealed memory:
  - Secret encryption key
  - Master hash sum
- VPFS uses remote attestation:
  - Trusted backup storage required, because data in untrusted storage can be lost
  - Secure access to backup server needed
  - VPFS challenges backup server: „Will you store my backups reliably?“

TECHNISCHE UNIVERSITÄT DRESDEN

A CLOSER LOOK AT THE WHOLE PICTURE

NITPICKER


DOpE
DOpE
NitOvIW/M
Main
xterm
NitOvIW/M
hosts.deny
inetd.conf
init.d
rat
rpc
securetty
DOpE
xeye
NitOvIW/M
Window Focus Preferences
NitOvIW/M
Input Focus Mode
Automatic Window Raise Delay
Manual: Clic
set keyboard
Auto: Set ke
to the windo
pointer
Install colorma
...that has
...that is u
pointer.
Balloon Help
DOpE
Dumper
Reboot computer
1594 MHz uptime 0:06:35
14dope | SharedMem(alloc): hl.low=c0000, lh.high=80000, id=4f, size=20000
14dope | SharedMem(get_adr): address = 80000
names | 0.04: server_names_register(0.07 -> "14dope.07")
names | 0.07: server_names_register(0.07 -> "Ovs2")
vscrtest| VScrTest(feedback_init): done
14dope | SharedMem(alloc): hl.low=c0000, lh.high=80000, id=51, size=25800
14dope | SharedMem(get_adr): address = 780000
names | 0.04: server_names_register(0.00 -> "14dope.00")
names | 0.00: server_names_register(0.00 -> "Ovs3")
vscrtest| VScrTest(bump_init): done
14dope | SharedMem(alloc): hl.low=c0000, lh.high=80000, id=53, size=96000
14dope | SharedMem(get_adr): address = e00000
nitovl | nitovlwm: open_screen called w=1024, h=768, depth=16
nitovl | open_screen: l4dm_share to Nitpicker returned 0
nitovl | open_screen: nitpicker_import_buffer_call returned buf_id=5
nitovl | nitovlwm: map_screen called
nitovl | map_screen: l4dm_share to client returned 0
nitovl | nitovlwm: map_screen returns t_id=0x000C0000,00000000 ds_id=0x0000005
nitovl : 4 size=0x00100000
nitpicke| atkbd.c: Unknown key released (translated set 2, code 0x7a on isa0060
nitpicke: /serio0).
nitpicke| atkbd.c: This is an XFree86 bug. It shouldn't access hardware directl
nitpicke: y.
nitpicke| atkbd.c: Unknown key released (translated set 2, code 0x7a on isa0060
nitpicke: /serio0).
nitpicke| atkbd.c: This is an XFree86 bug. It shouldn't access hardware directl
nitpicke: y.
names | 58.00: server_names_register(58.00 -> "ovlwm.main")
names | 58.00: server_names_register(58.03 -> "ovlwm.03")
DOpE
vscrtest
DOpE
crtest
Flasco
inside
NitOvIW/M
NitOvIW/M

TRUST NITPICKER

- *User cannot just trust what he / she sees on the screen!*
- Solution:
  - Remote attestation
  - For example with trusted device:
    - User‘s cell phone sends **nonce** to PC
    - PC replies with quote of **nonce** + **PCR** values
    - User can decide whether to trust or not

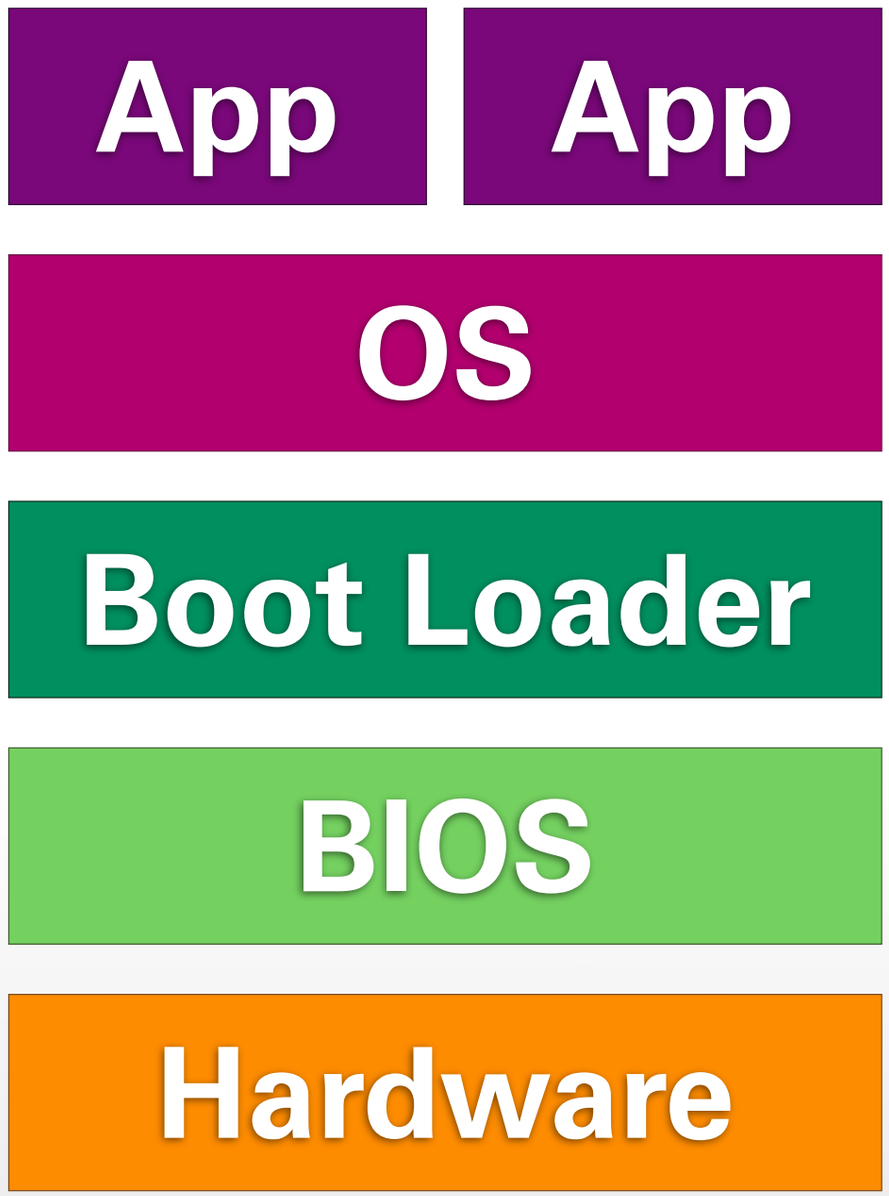
TECHNISCHE UNIVERSITÄT DRESDEN

A SECOND LOOK AT THE CHAIN OF TRUST

- When you press the power button ...
  - First code to be run: BIOS boot block
  - Stored in small ROM
  - Starts chain of trust:
    - Initialize TPM
    - Hash BIOS into TPM
    - Pass control to BIOS
- BIOS boot block is **C**ore **R**oot of **T**rust for **M**easurement (**CRTM**)

CHAIN OF TRUST

- Discussed so far:
  - **CRTM** & chain of trust
  - How to make components in chain of trust smaller
- **Observation:** BIOS and boot loader only needed for booting
- **Question:** can chain of trust be shorter?

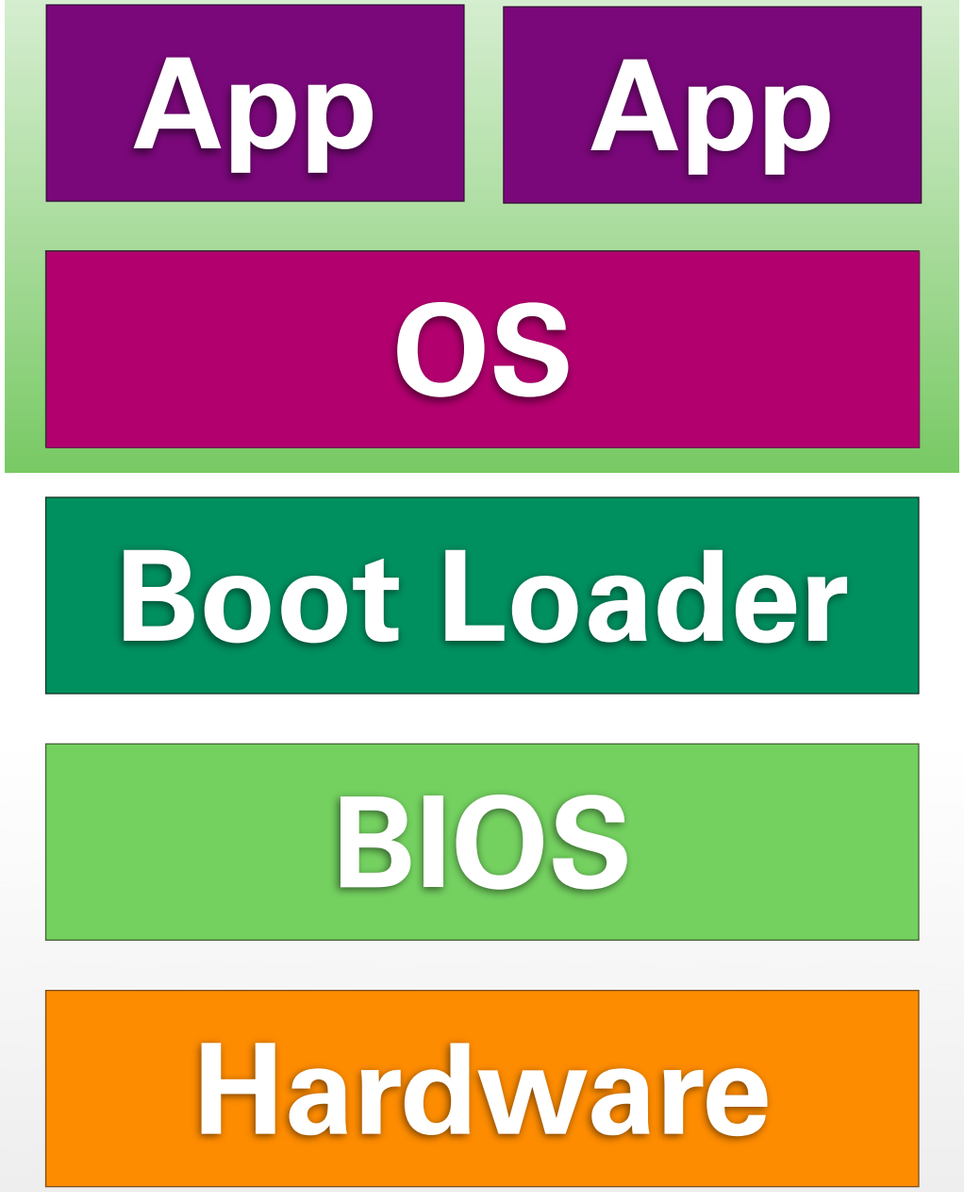


DRTM

- **CRTM** starts chain of trust early
- **D**ynamic **R**oot of **T**rust for **M**easurement (**DRTM**) starts it late:
  - Special CPU instructions (AMD: skinit, Intel: senter)
  - Put CPU in known state
  - Measure small „secure loader“ into TPM
  - Start „secure loader“
- **DRTM**: Chain of trust can start anywhere

DRTM: OSLO

- First idea: **DRTM** put right below OS
- Smaller TCB:
  - Large and complex BIOS / boot loader removed
  - Small and simple **DRTM** bootstrapper added
- Open Secure Loader **OSLO**: **1,000** SLOC, **4KB** binary size [6]

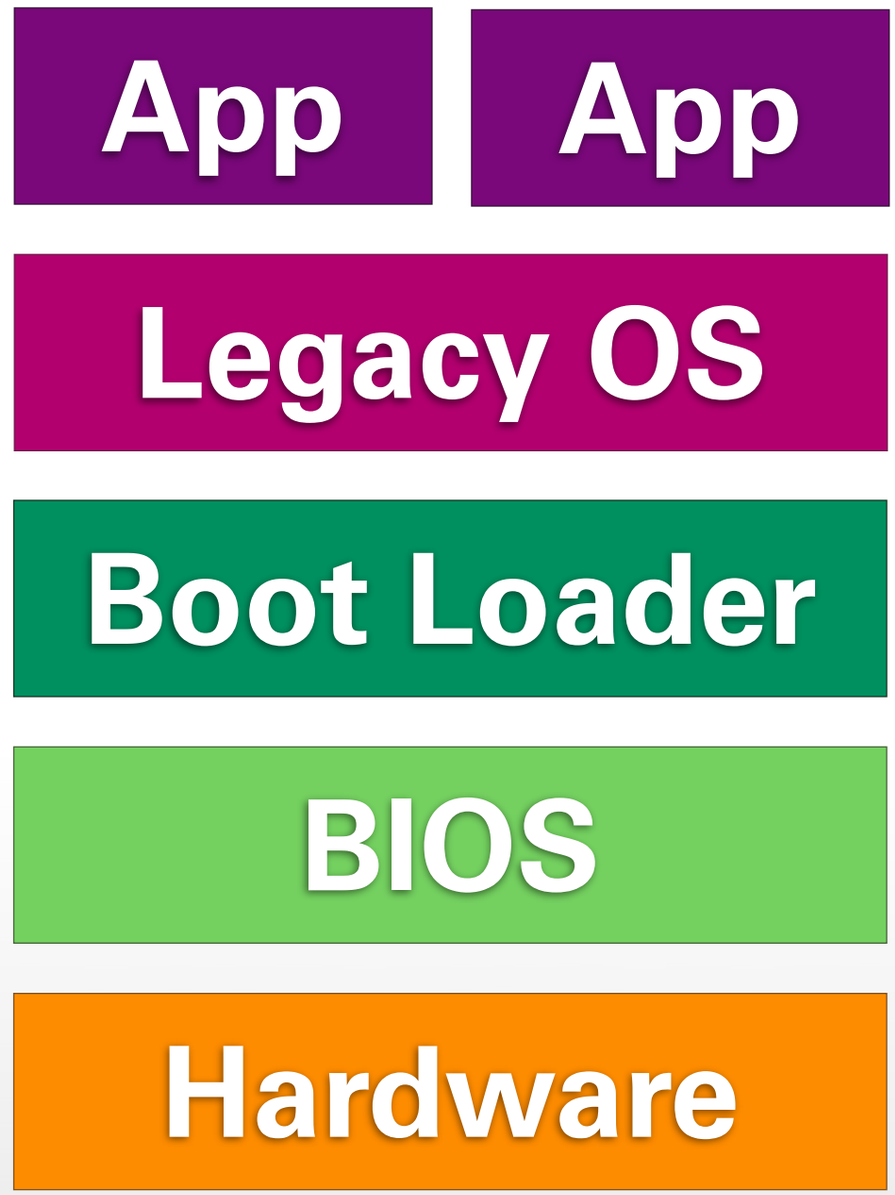


DRTM CHALLENGE

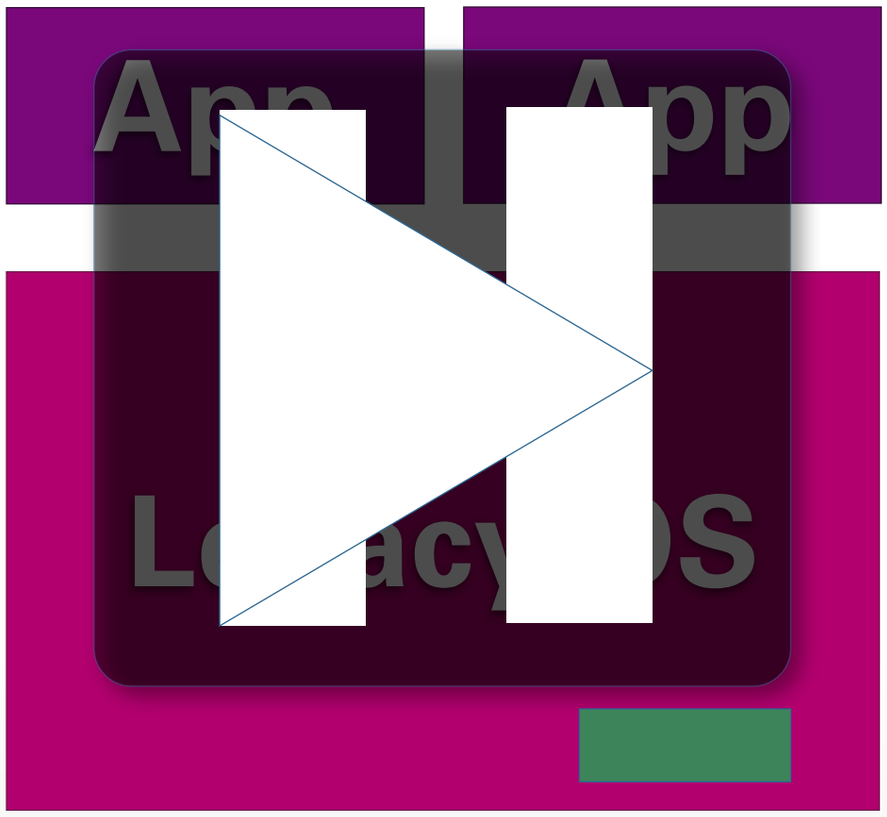
- DRTM remove boot software from TCB
- Key challenges:
  - „Secure loader“ must not be modified
  - Requires careful checking of platform state (e.g., that secure loader is actually in locked RAM, not in insecure device memory)

DRTM: FLICKER

- New **DRTM** can be established anytime
- Flicker [7] approach:
  - Pause legacy OS
  - Execute critical code as **DRTM** using skinit
  - Restore CPU state
  - Resume legacy OS



DRTM: FLICKER



Hardware

FLICKER DETAILS

- Pause untrusted legacy OS, stop all CPUs
- Execute skinit:
  - Start Flicker code as „secure loader“
  - Unseal input / sign data / seal output
- Restore state on all CPUs
- Resume untrusted legacy OS
- If needed: create quote with new PCRs
- *TCB in order of only few thousand SLOC!*

FLICKER LIMITS

- Problems with Flicker approach:
  - Untrusted OS must cooperate
  - Only 1 CPU active, all other CPUs stopped
  - Secure input and output only via slow TPM functionality (seal, unseal, sign)
- Works for some server scenarios (e.g., handling credentials)
- Client scenarios require more functionality (e.g., trusted GUI for using applications)

THERE IS A MTM ...

- TPMs specified for mobile platforms, too
- **MTMs** protect network operator and user
- However, in reality:
  - Simple implementations in smartphones, etc.
  - Non-modifiable boot ROM loads OS
  - OS is signed with manufacturer key, checked
  - Small amount of flash integrated into SoC
  - Not open: **closed** or **secure boot** instead of **authenticated booting**

WHAT'S NEXT?

- Later today:
  - Practical exercise
- February 1:
  - Lecture *„Debugging Operating Systems“*
  - Complex lab

References

- [1] http://www.heise.de/security/Anonymisierungsnetz-Tor-abgephisht--/news/meldung/95770
- [2] https://www.trustedcomputinggroup.org/home/
- [3] https://www.trustedcomputinggroup.org/specs/TPM/
- [4] https://www.trustedcomputinggroup.org/specs/PCClient/
- [5] Carsten Weinhold and Hermann Härtig, „VPFS: Building a Virtual Private File System with a Small Trusted Computing Base“, Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008, 2008, Glasgow, Scotland UK
- [6] Bernhard Kauer, „OSLO: Improving the Security of Trusted Computing“, Proceedings of 16th USENIX Security Symposium, 2007, Boston, MA, USA
- [7] McCune, Jonathan M., Bryan Parno, Adrian Perrig, Michael K. Reiter, and Hiroshi Isozaki, "Flicker: An Execution Infrastructure for TCB Minimization", In Proceedings of the ACM European Conference on Computer Systems (EuroSys'08), Glasgow, Scotland, March 31 - April 4, 2008