



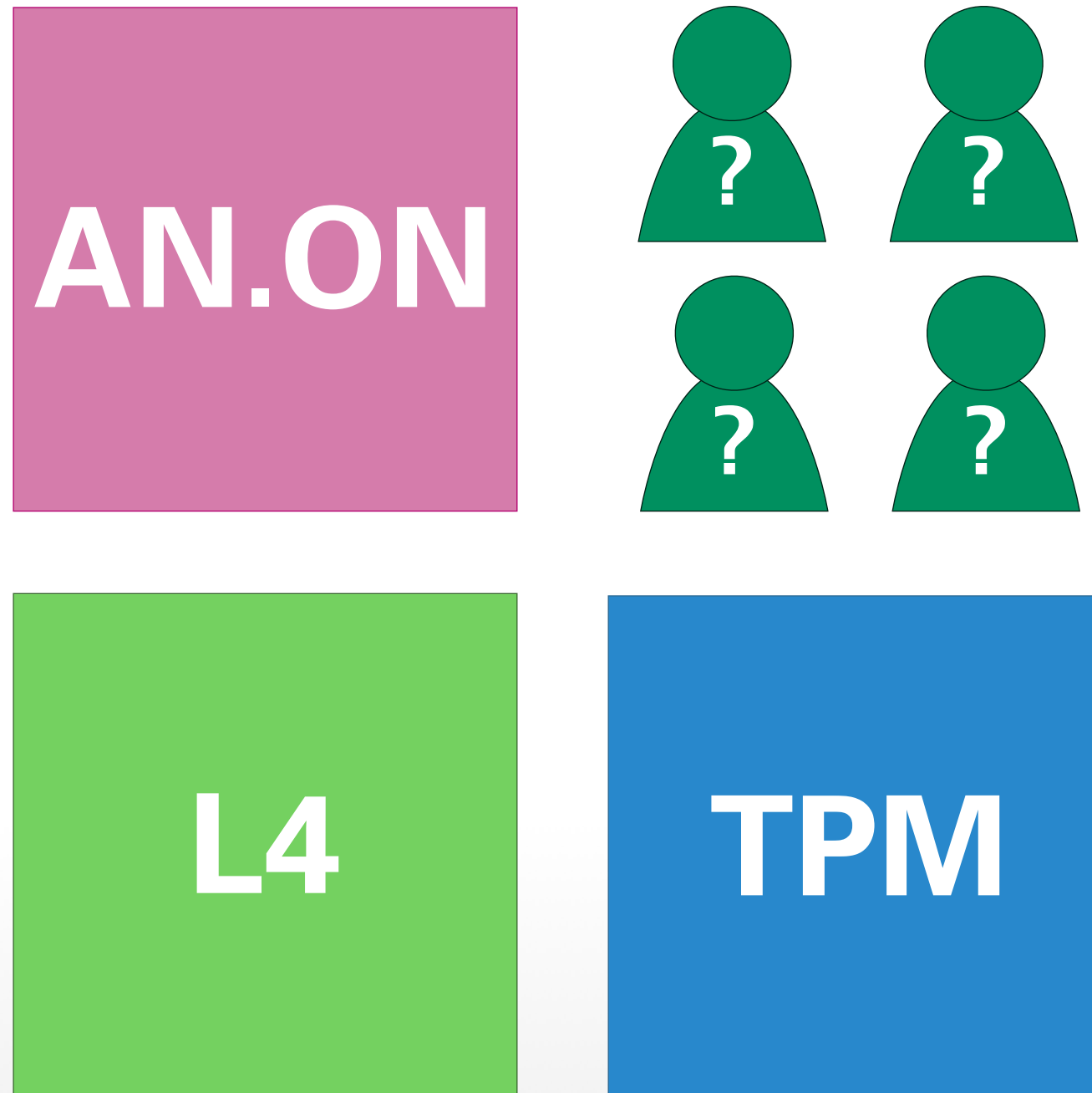
TECHNISCHE
UNIVERSITÄT
DRESDEN

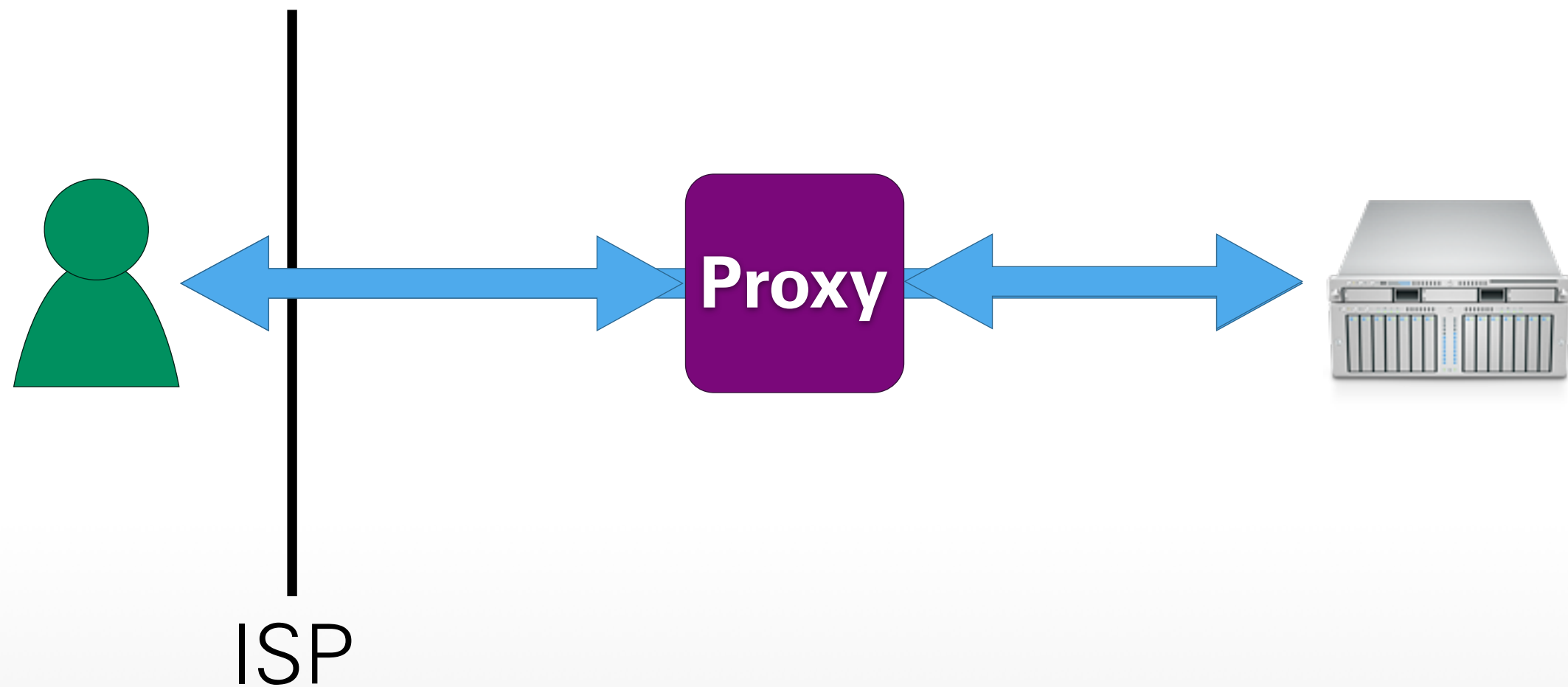
Department of Computer Science Institute for System Architecture, Operating Systems Group

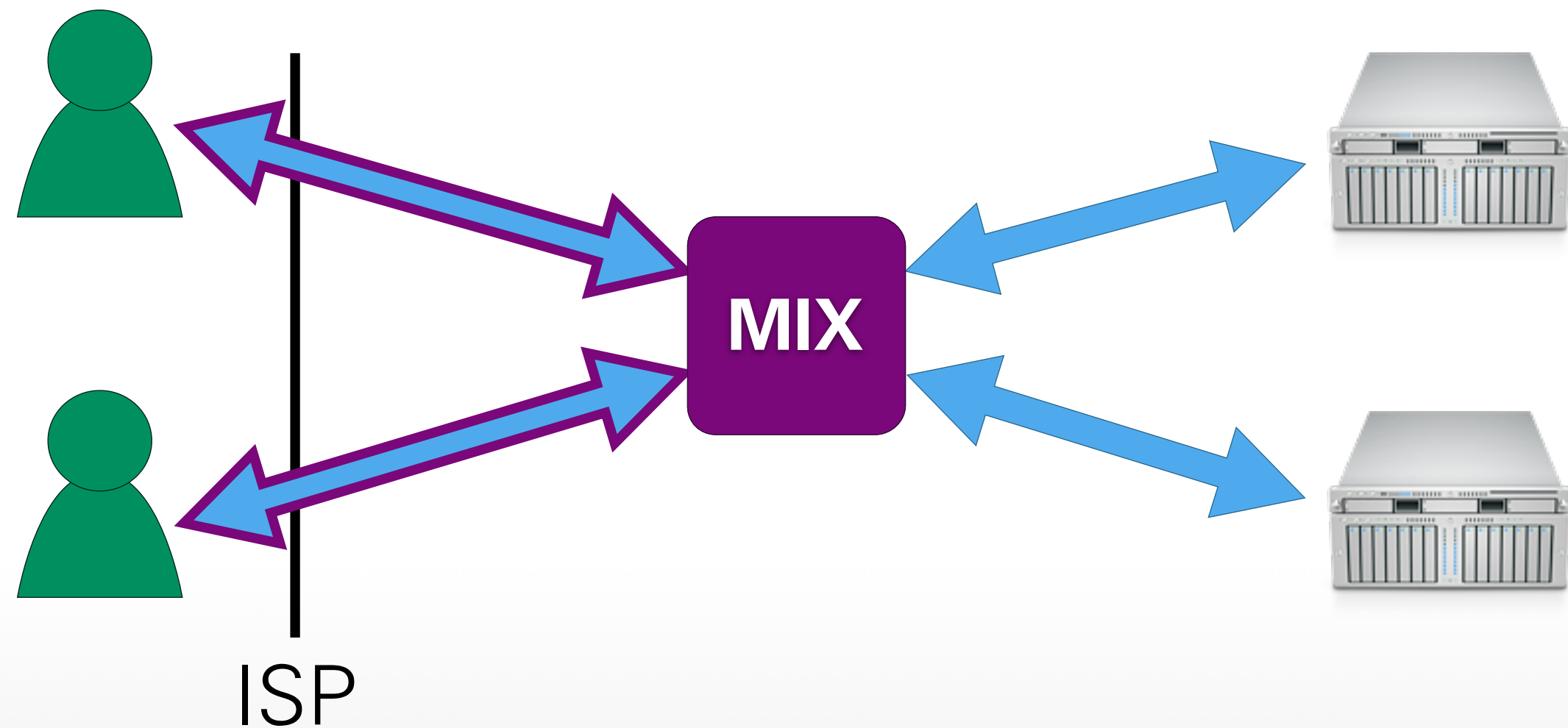
TRUSTED COMPUTING

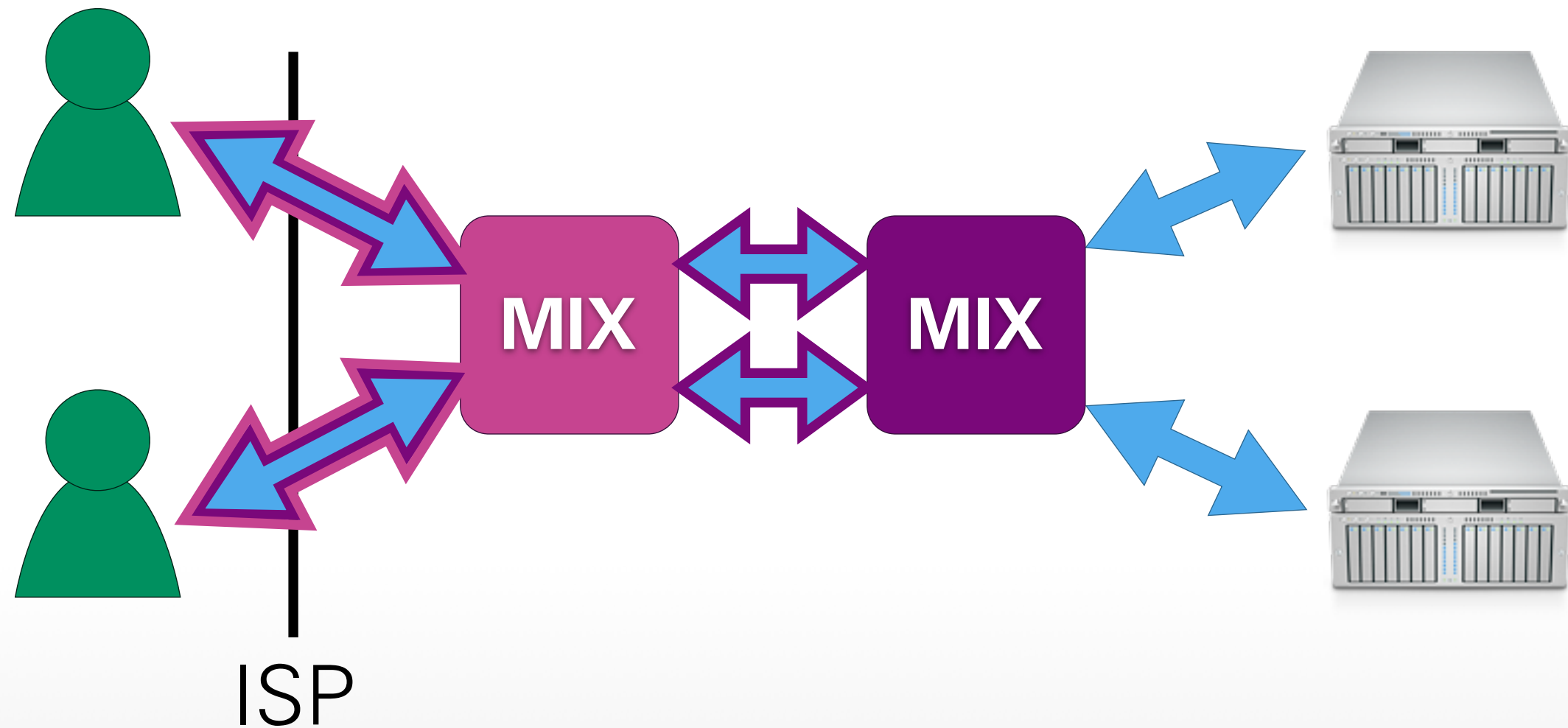
CARSTEN WEINHOLD

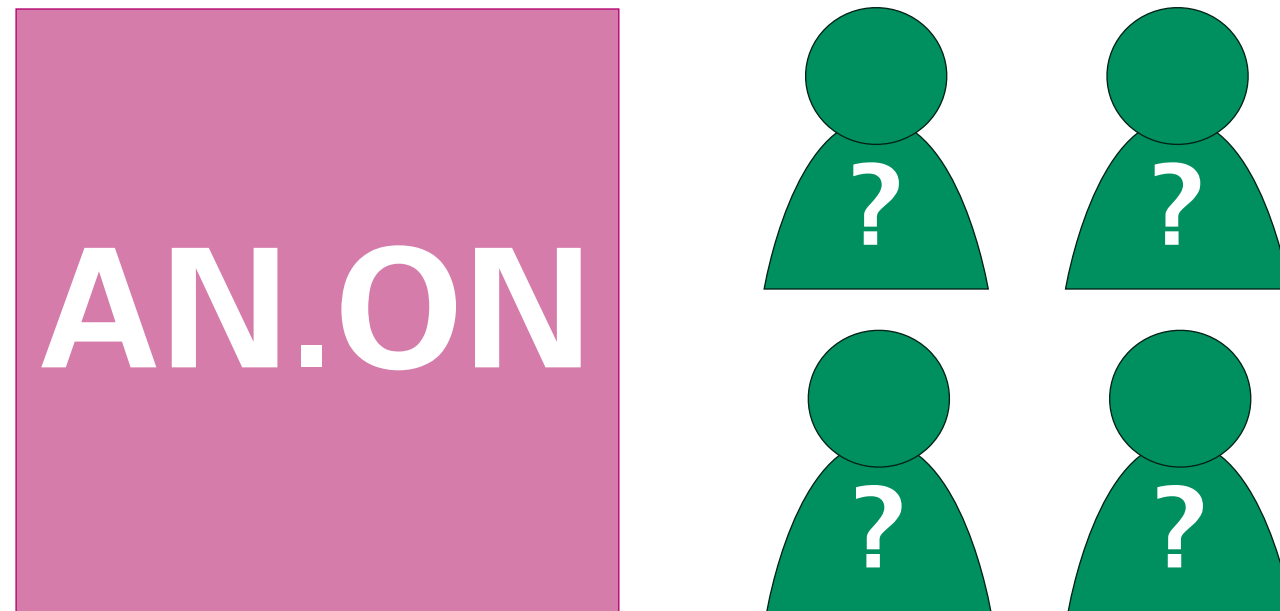
- **Today: Trusted Computing Technology**
 - Lecture discusses basics in context of TPMs
 - More theoretical concepts also covered in lecture „Distributed Operating Systems“
- **Things you should have heard about:**
 - How to use asymmetric encryption
 - Concept of digital signatures
 - Collision-resistant hash functions



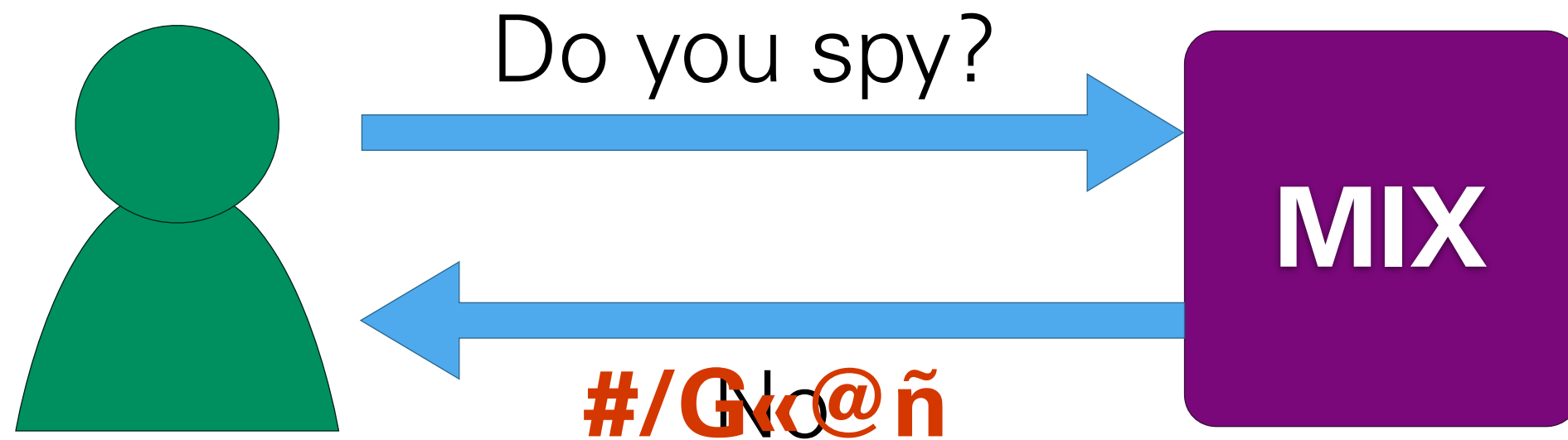


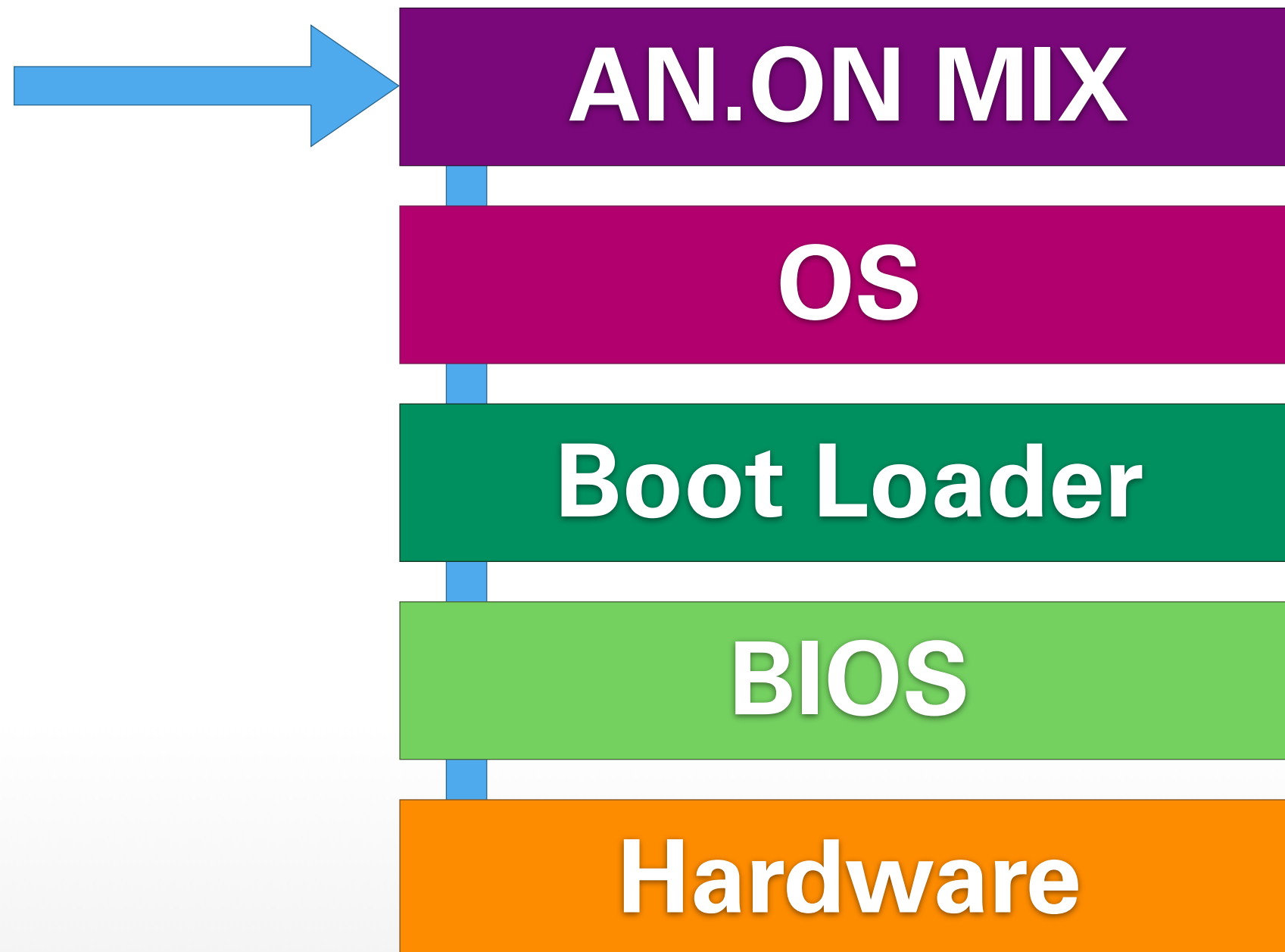






- Last proxy sees data in **plaintext**
- No additional end-to-end encryption?
 - Ideal for password phishing or identifying returning users (cookies, ...)
 - Dan Egerstad [1]: 100 passwords sniffed with 5 exit nodes
- TOR: increasing number of exit nodes in China, Russia, USA



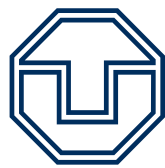




Platform Configuration Register

$$\text{PCR} := \text{SHA-1}(\text{PCR} \mid \mathbf{X})$$





BOOTING + TPM

AN.ON MIX

OS

Boot Loader

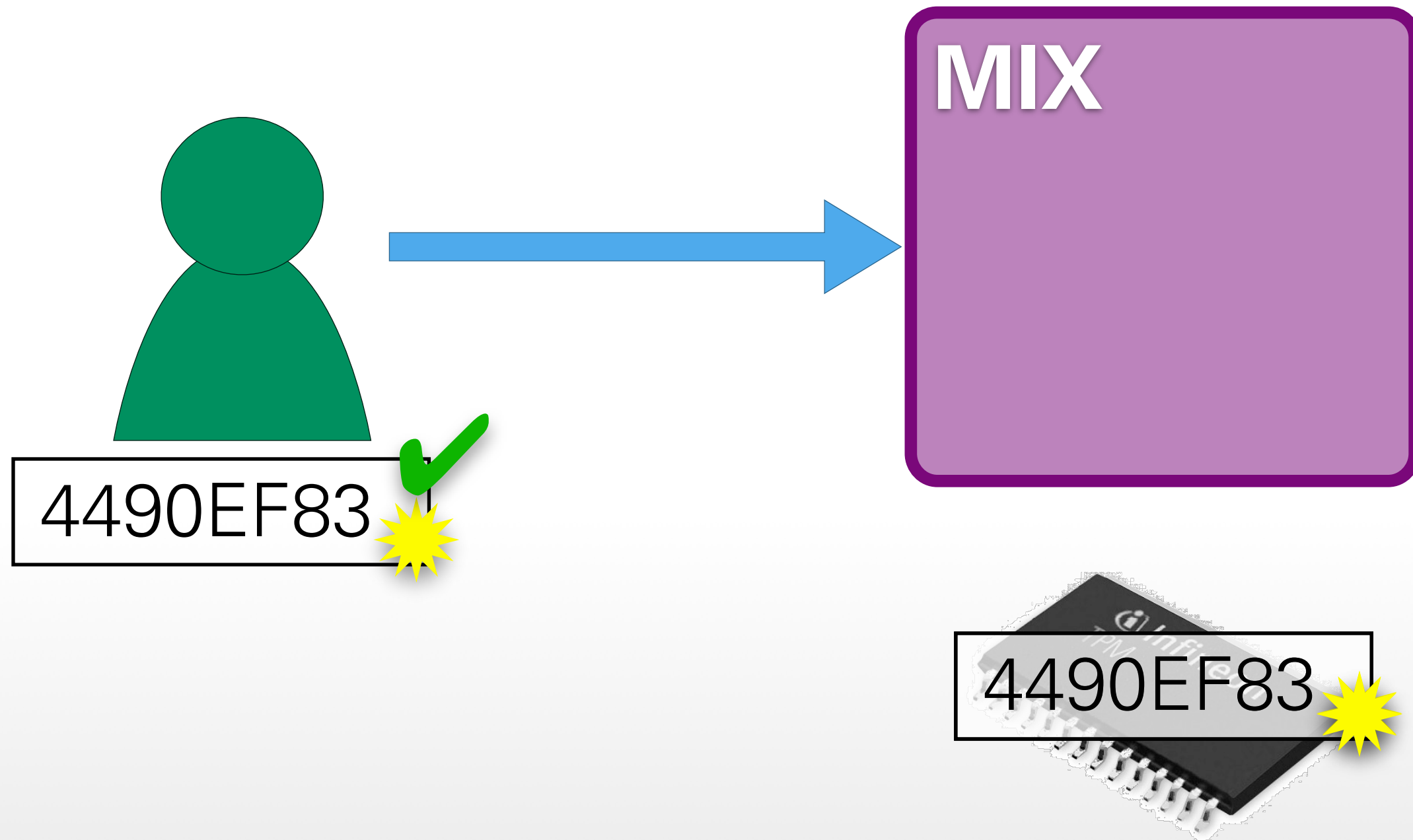
BIOS

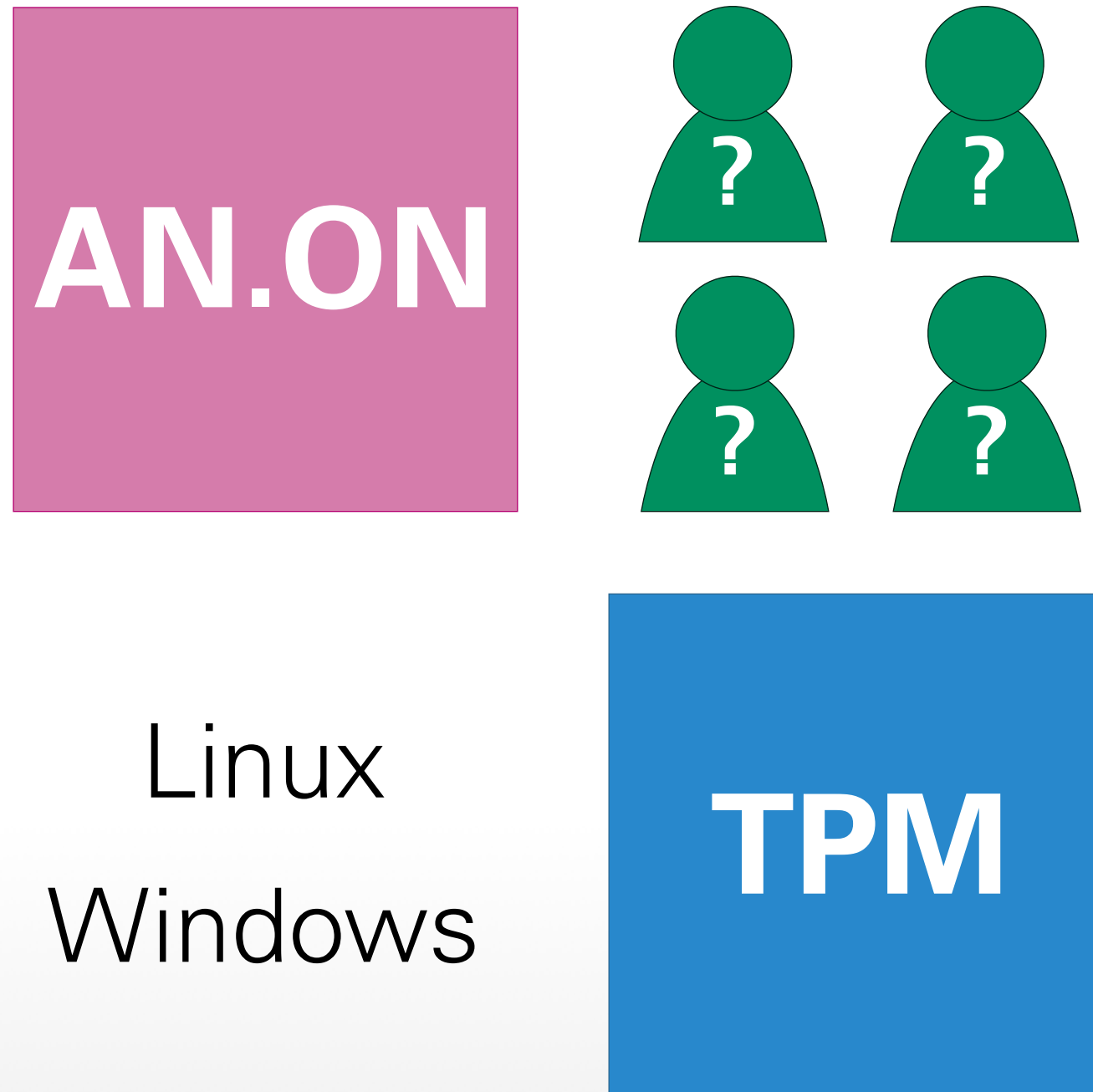


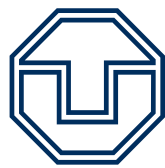
PCR

00000000

Remote Attestation

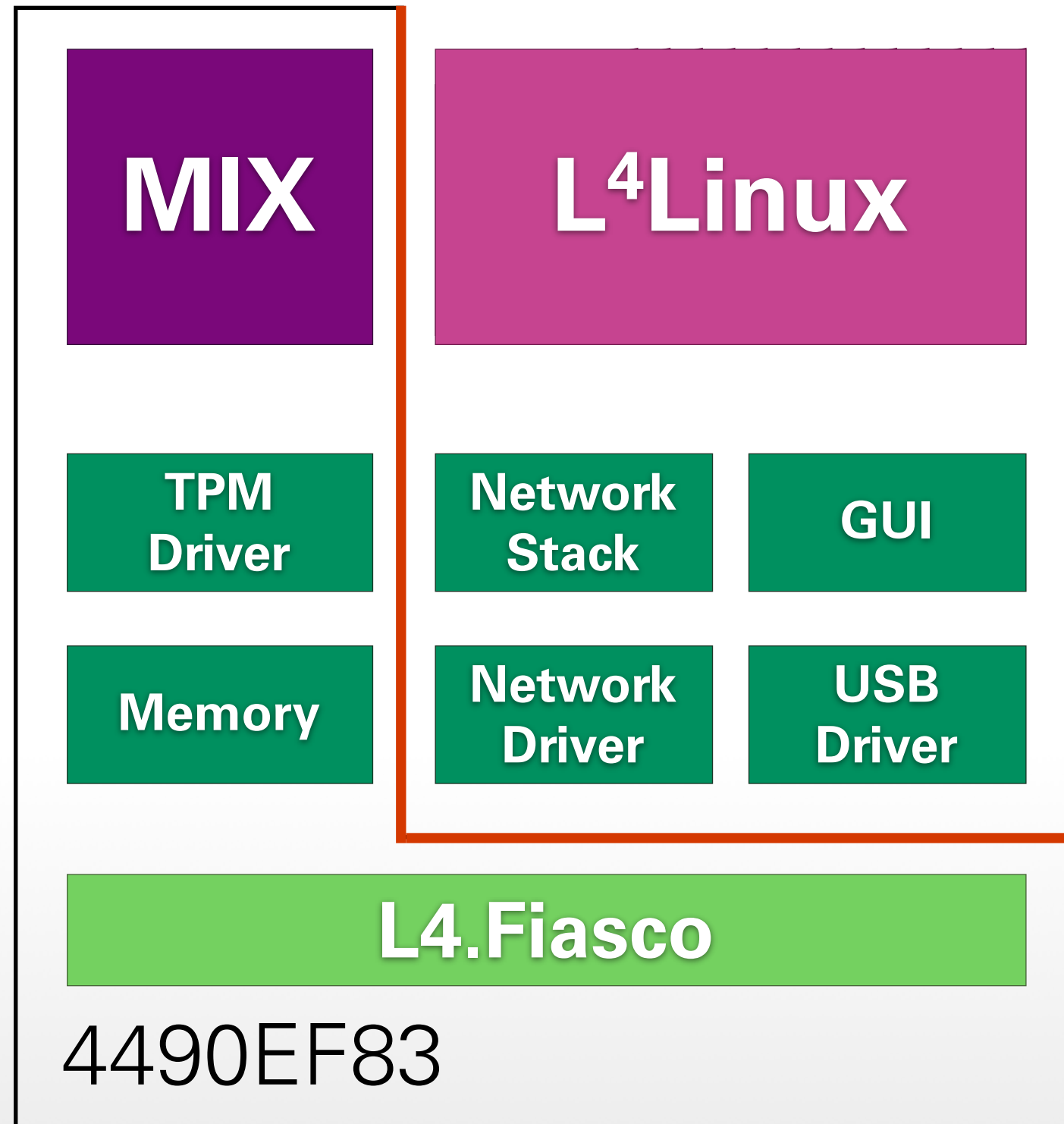
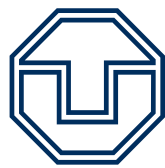


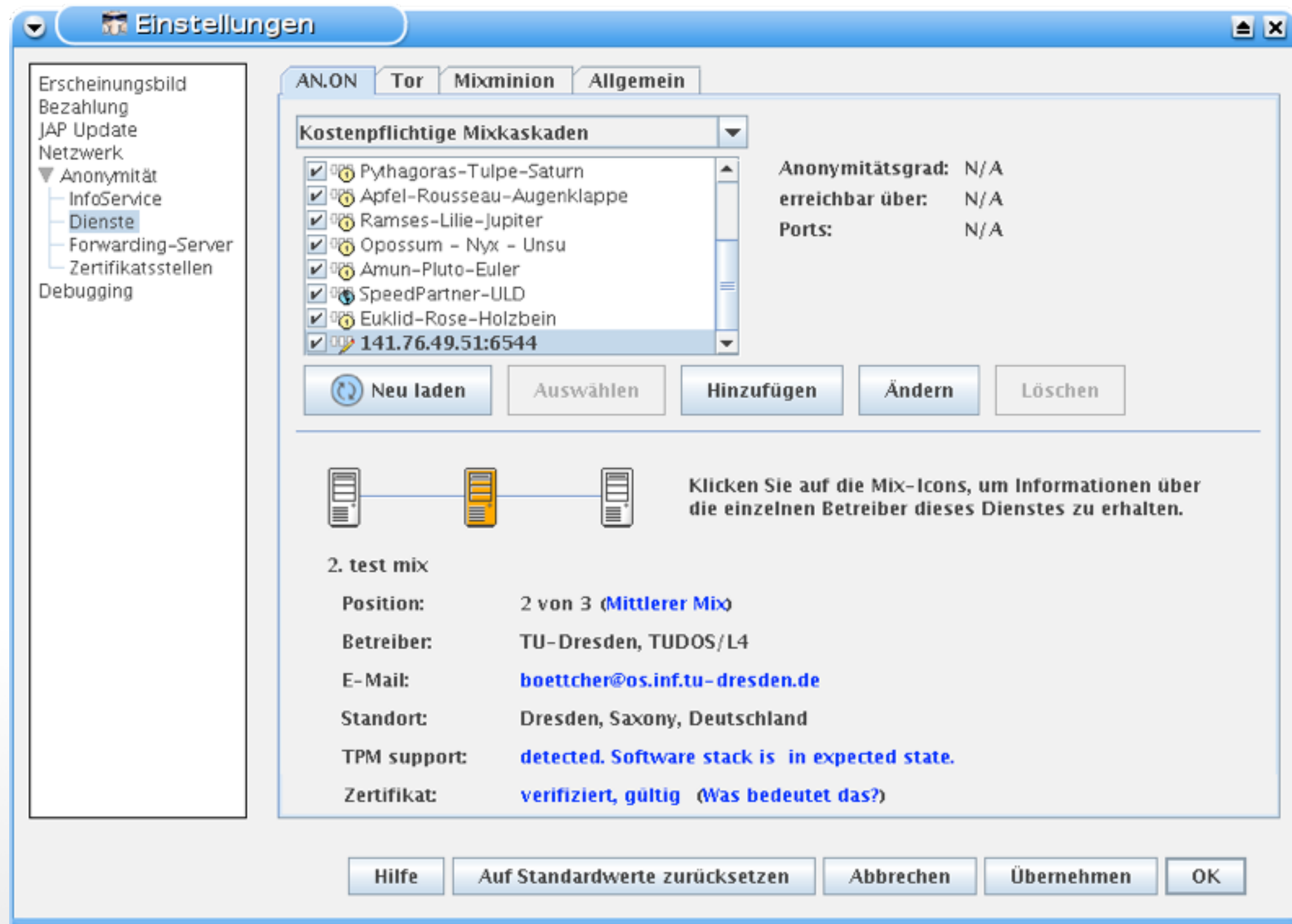


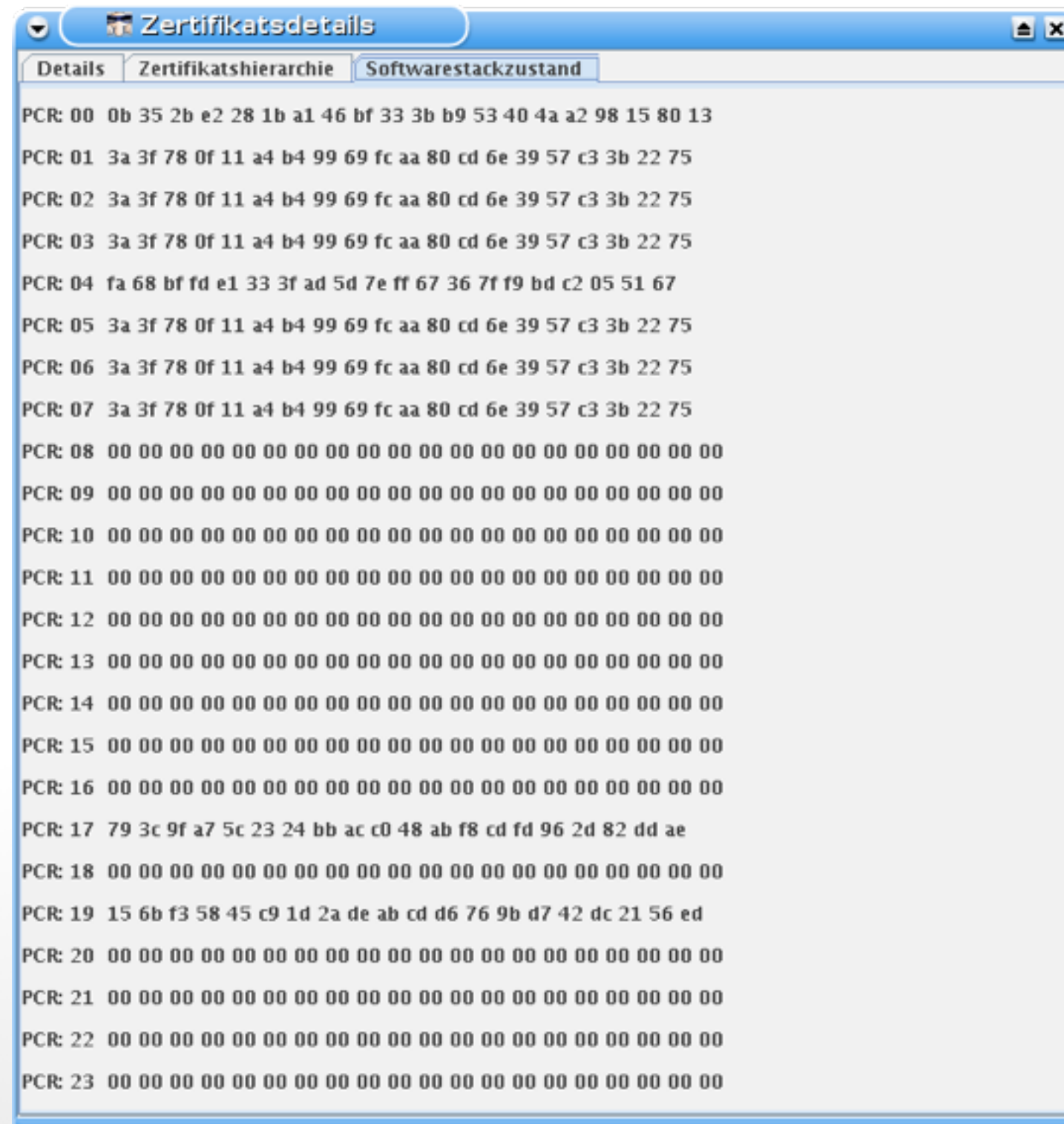


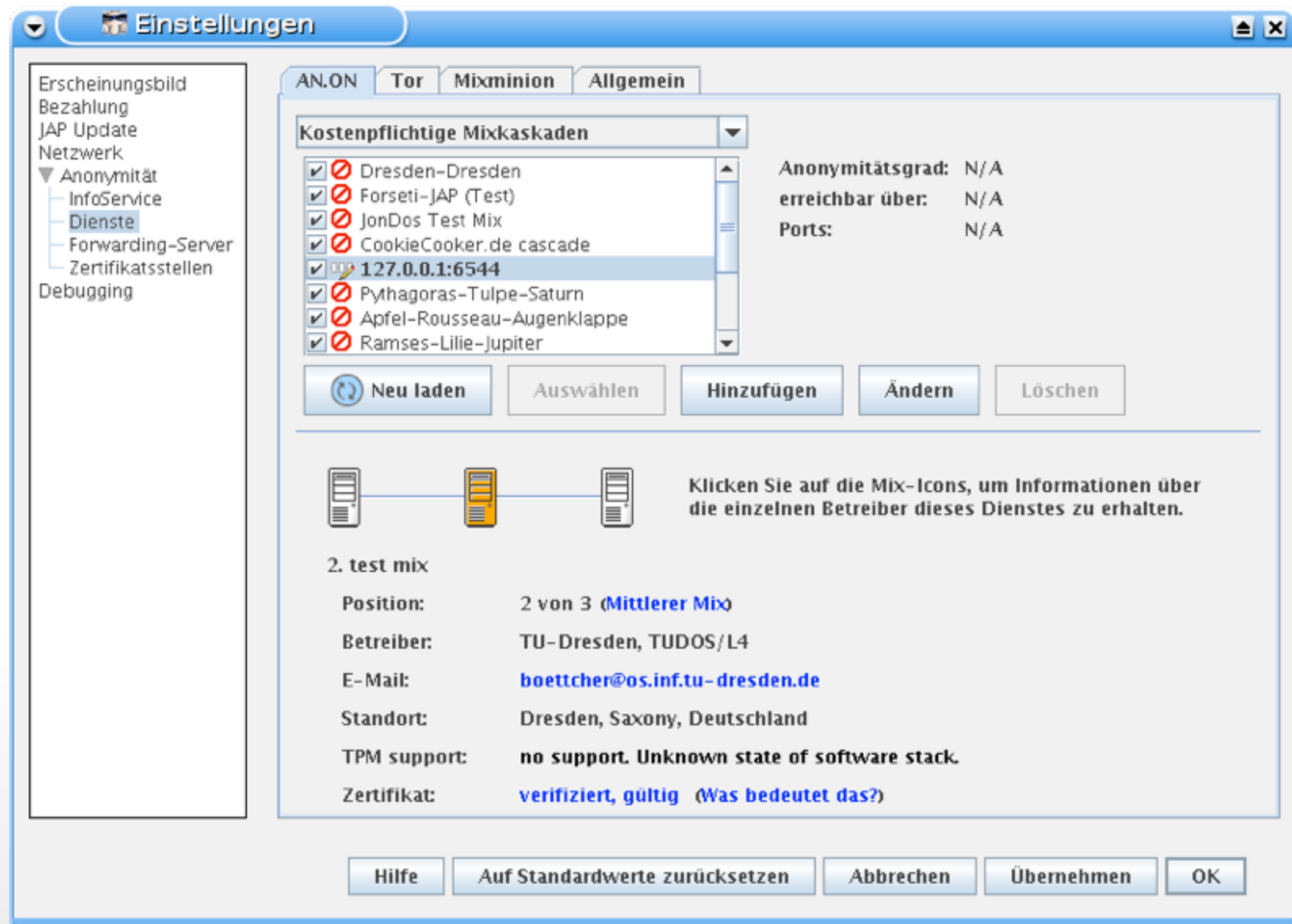
MONOLITHIC

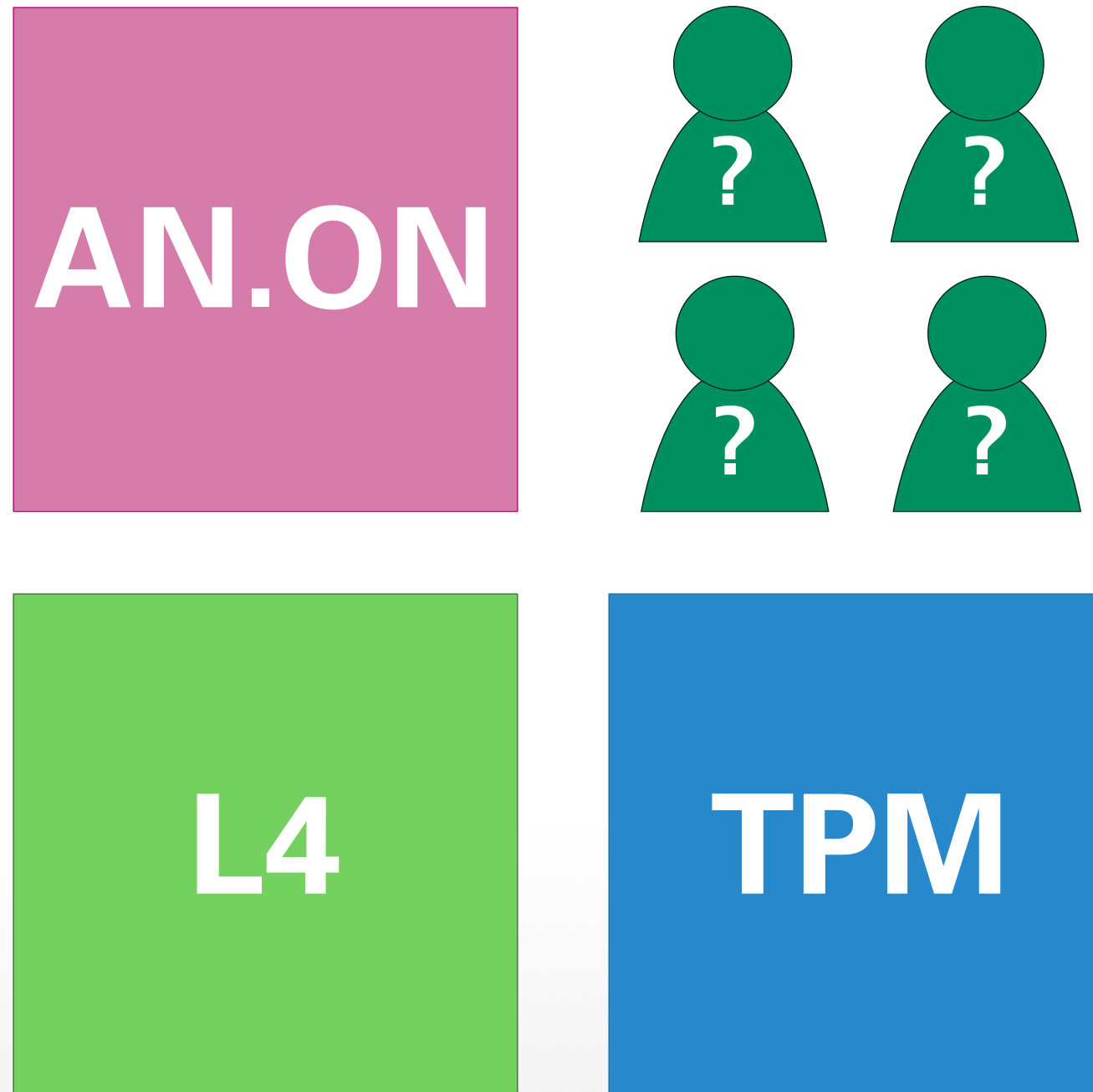






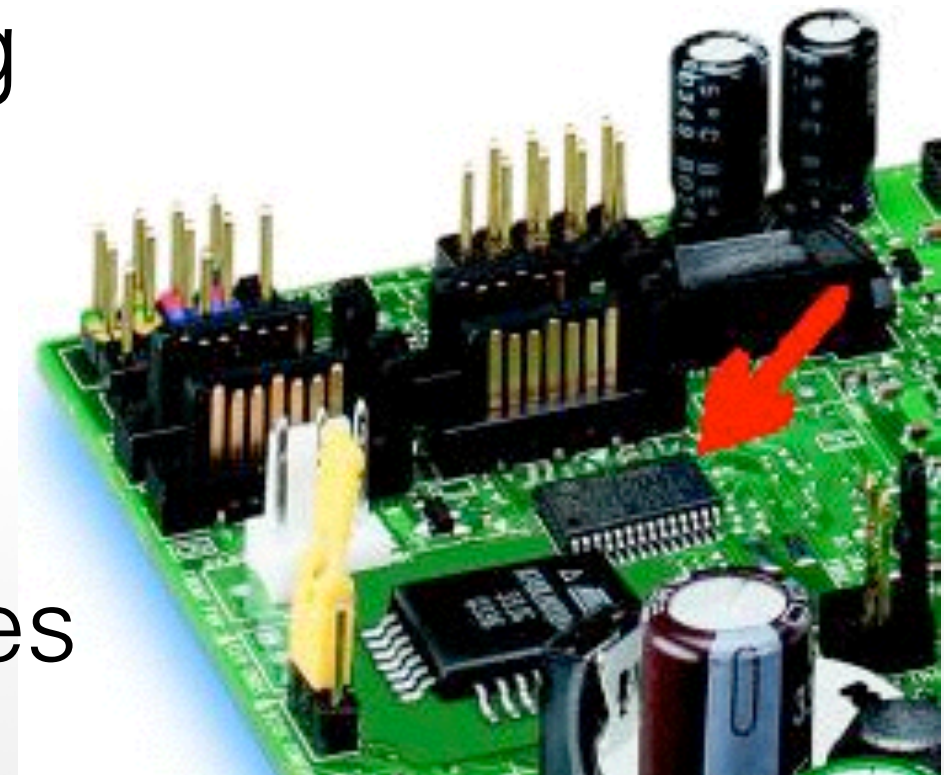






THE TRUSTED PLATFORM MODULE

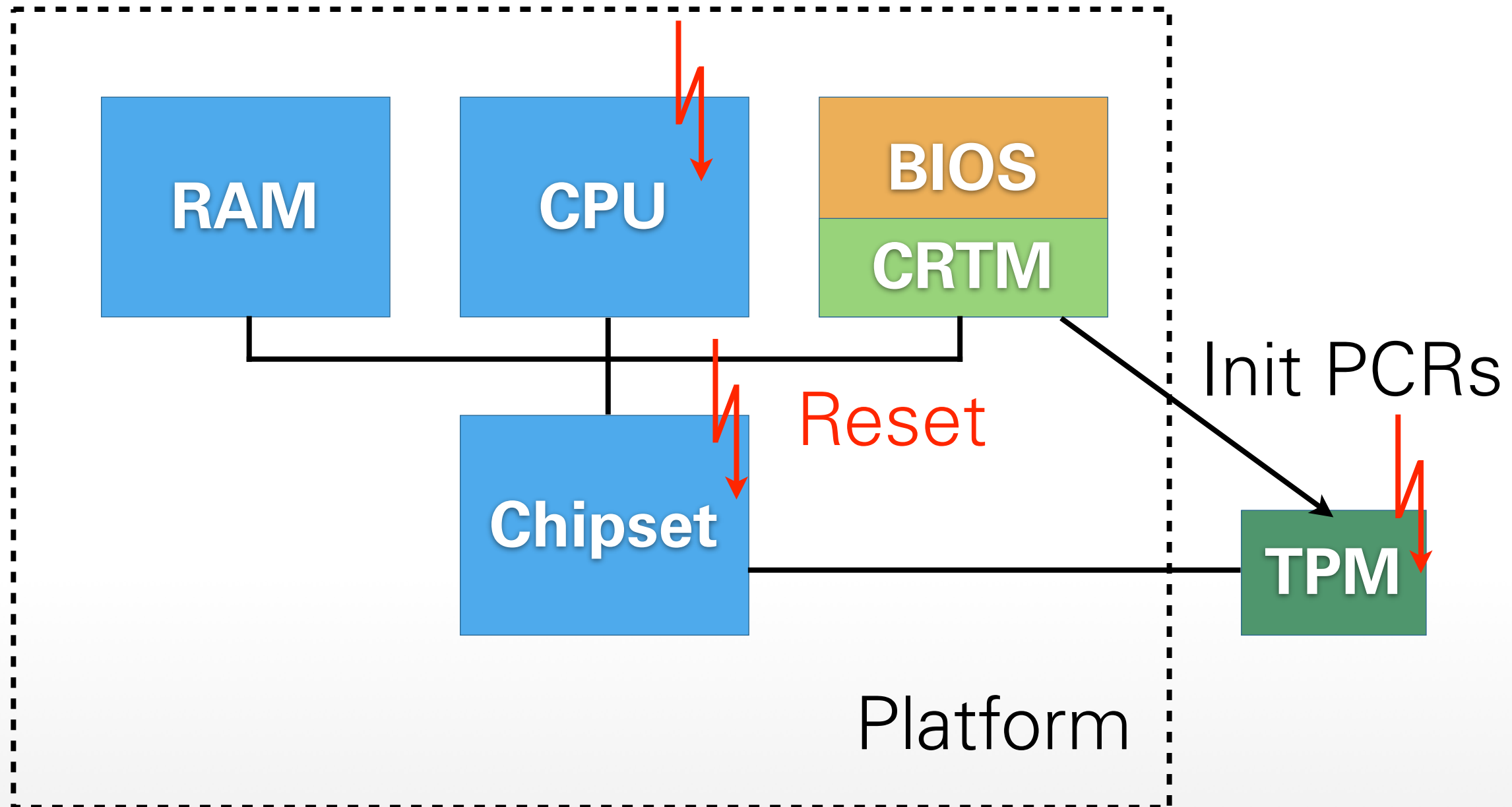
- TPMs are tightly integrated into platform:
 - Soldered on motherboards
 - ... or built into chipset
- Tamper resistant casing
- Widely deployed:
 - Business notebooks
 - Office desktop machines
 - Windows 8/RT tablets



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

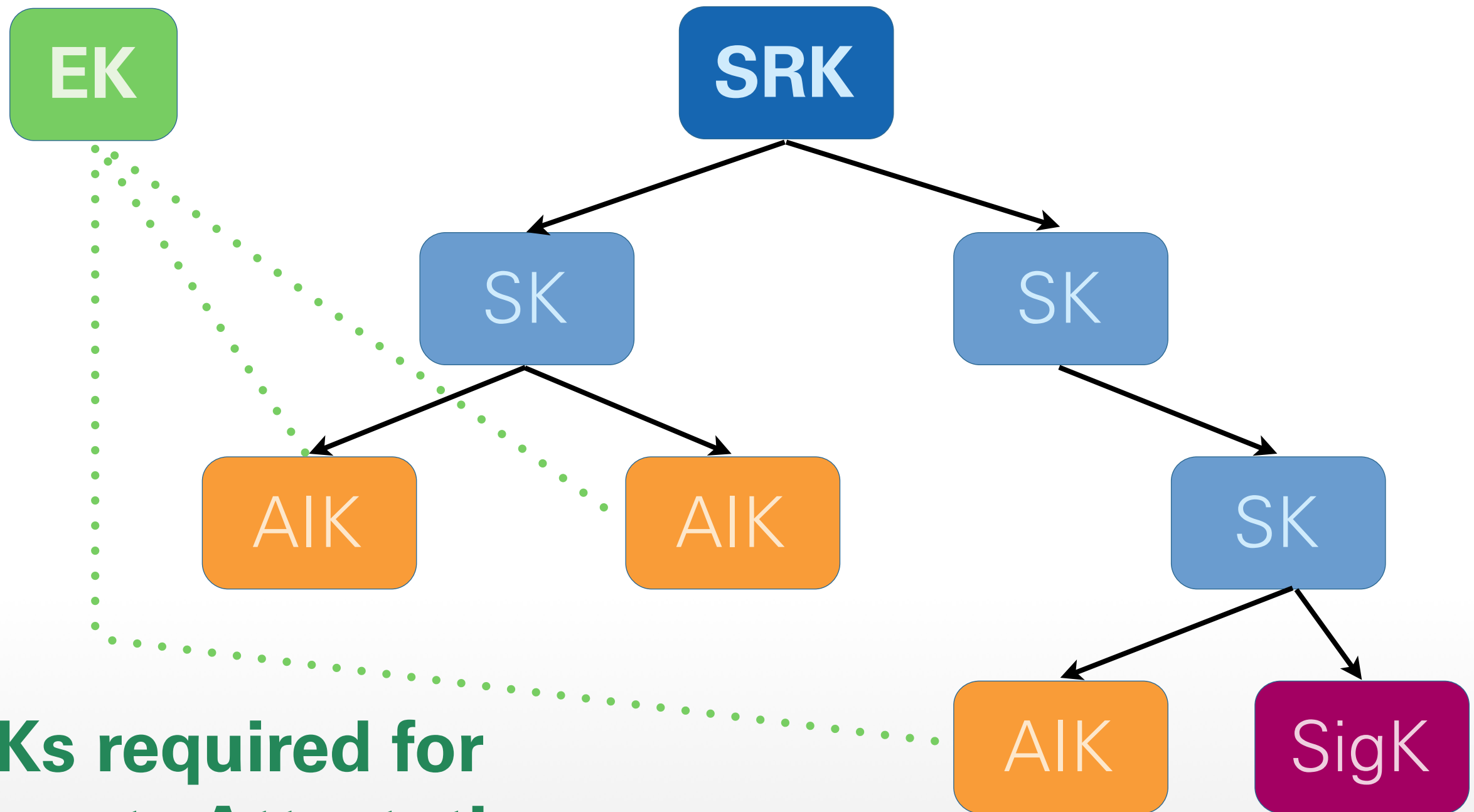
- TPM is cryptographic coprocessor:
 - **RSA** (encryption, signatures), **AES** (encryption), **SHA-1** (cryptographic hashes)
 - Other crypto schemes (e.g., **DAA**)
 - Random number generator
 - Platform Configuration Registers (**PCRs**)
 - Non-volatile memory
- TPMs are passive devices!

- 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 identified by Endorsement Key **EK**:
 - Generated in manufacturing process
 - Certified by manufacturer
 - Unique among all TPMs
 - Can only decrypt, serves as root of trust
- Creating entirely new **EK** possible (e.g., for use in corporate environments)
- Private part of **EK** never leaves TPM

- 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



**AIKs required for
Remote Attestation**

- Special key type for remote attestation:
Attestation Identity Key (**AIKs**)
 - TPM creates AIK + certificate request
 - **Privacy CA** checks certificate request,
issues certificate and encrypts under **EK**
 - TPM can decrypt certificate using **EK**
- **AIK** certificate:
 - „This **AIK** has been created by a valid TPM“
 - TPM identity (**EK**) cannot be derived from it

Application

OS

Boot Loader

BIOS

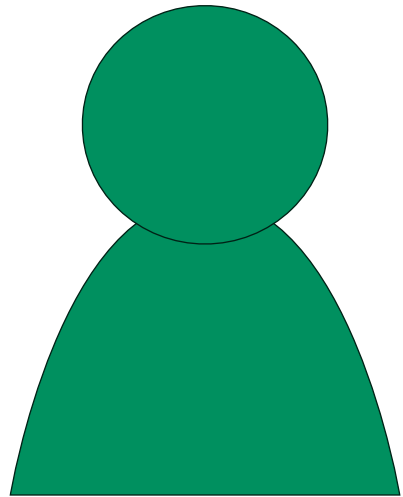
Authenticated
Booting



PCR

00000000000000000000000000000000

TPM_Quote(AIK, Nonce, PCR)



System

Challenger

4490EF83 ✓

AE58B991 ✨

Remote Attestation with
Challenge/Response

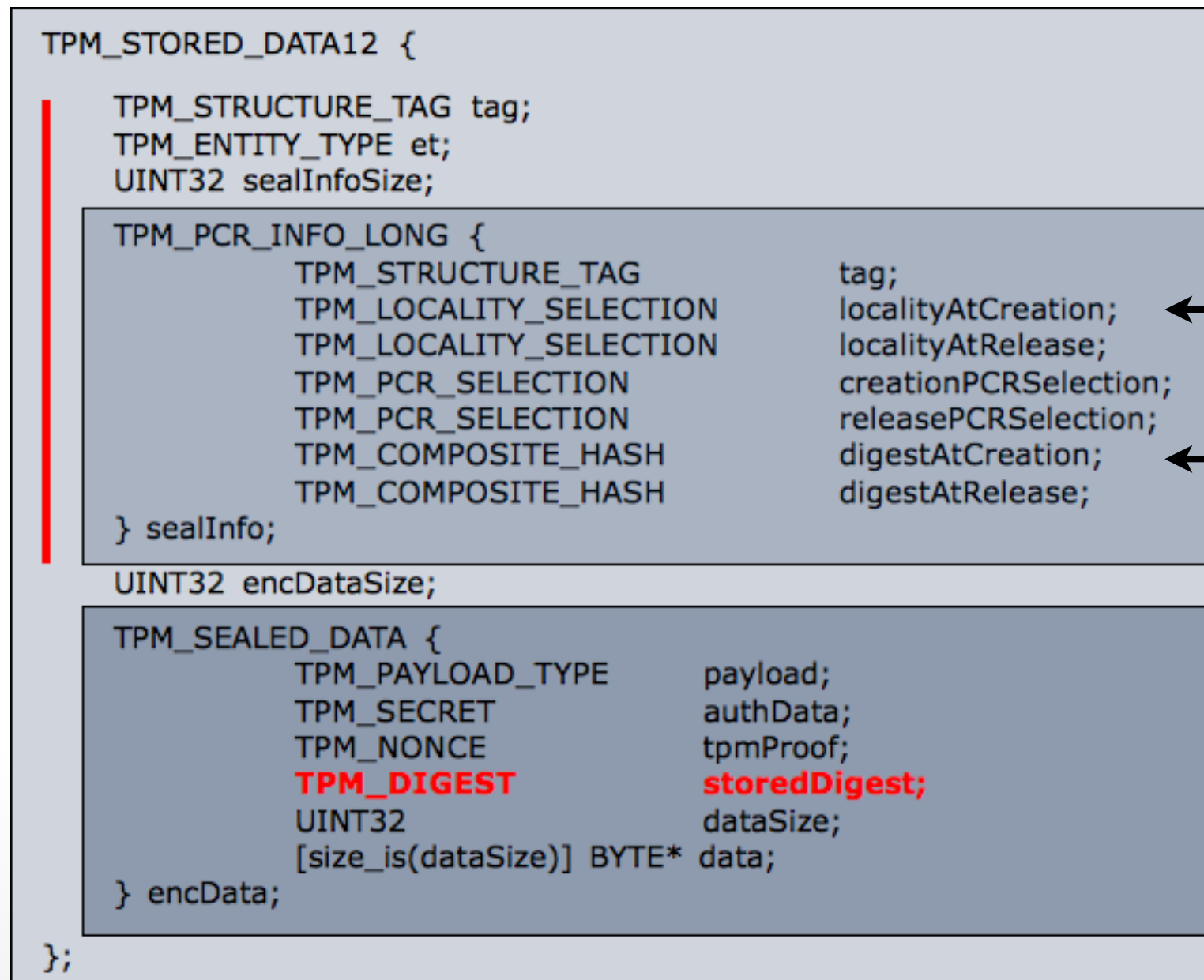
4490EF83

AE58B991 ✨

4490EF83

- 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



Only the TPM_SEALED_DATA structure is encrypted

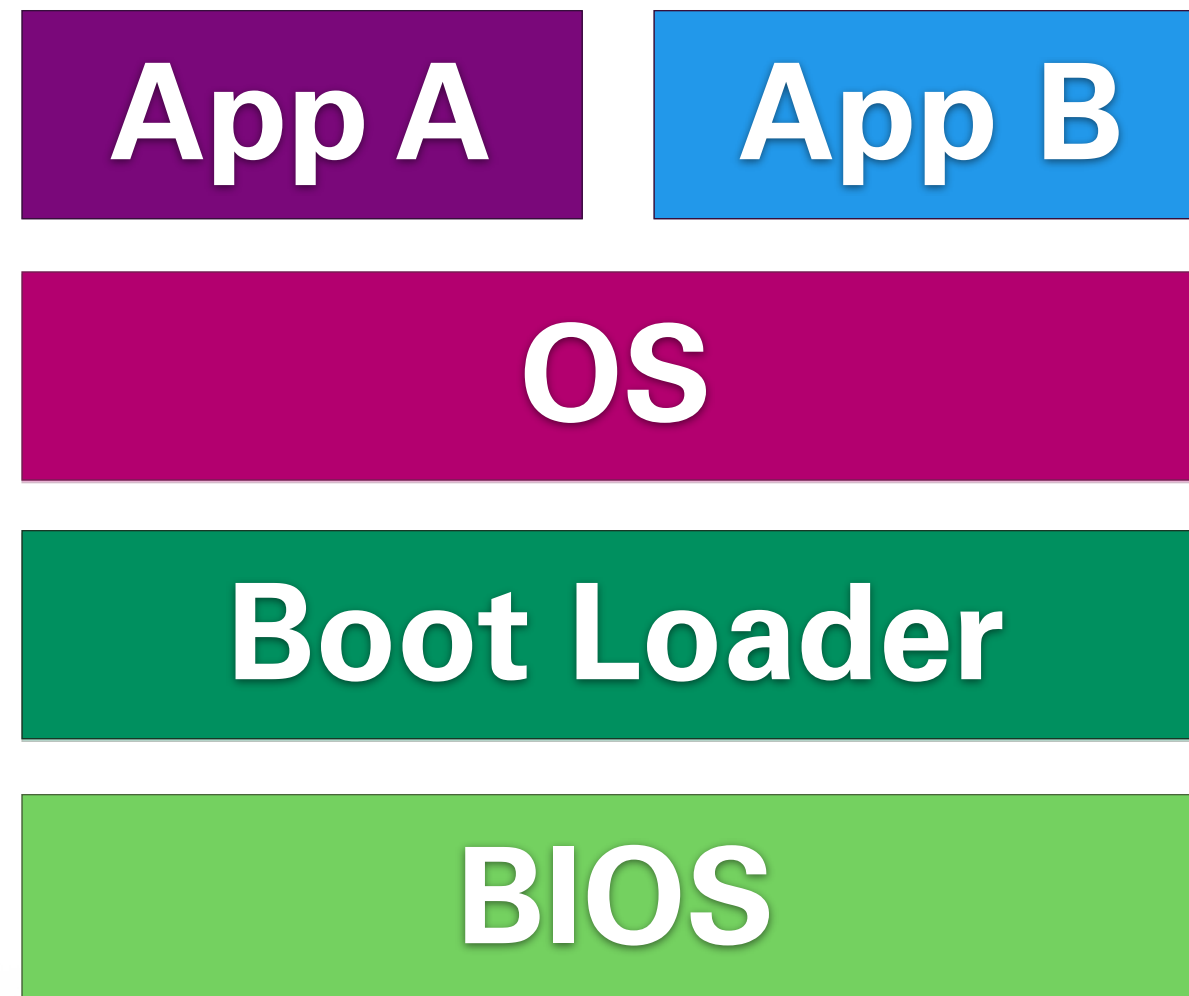
- Sealed data is stored outside the TPM
- Vulnerable to replay attacks:
 - Multiple versions of sealed blob may exist
 - Any version can be passed to TPM
 - TPM happily decrypts, if crypto checks out
- Problem:
 - What if sealed data must be current?
 - How to prevent use of older versions?

- TPMs provide **monotonic counters**
- Only two operations: **increment, 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

- 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

TPMS IN NIZZA

ARCHITECTURE

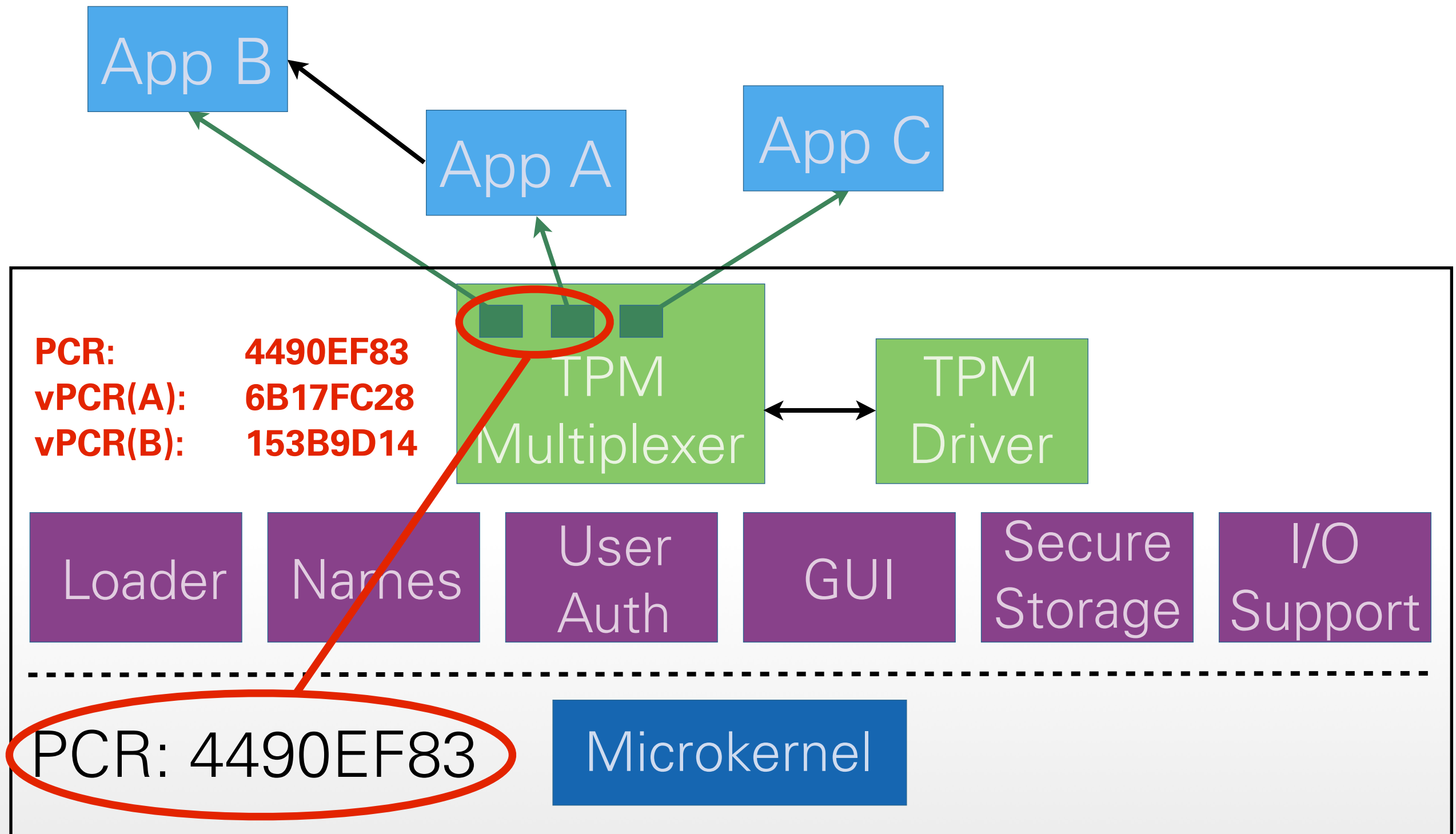
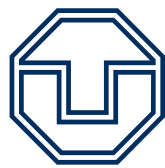


PCR

83E2BE9A

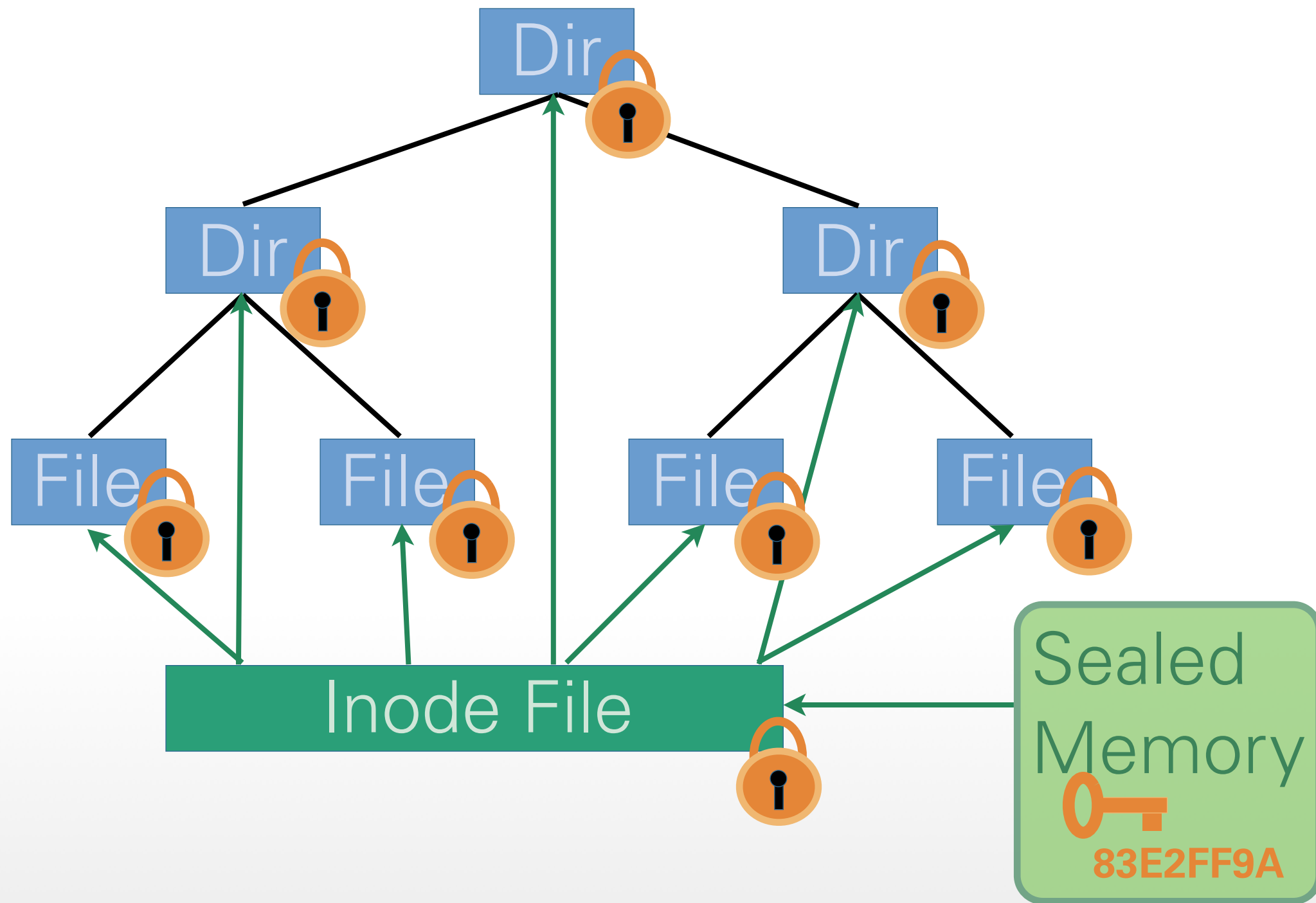
- Use one PCR per application:
 - Application measurements independent
 - Number of PCRs is limited (max 24)
- Use one PCR for all applications:
 - Chain of trust / application log grows
 - All applications reported in remote attestation (raises privacy concerns)
 - All applications checked when unsealing

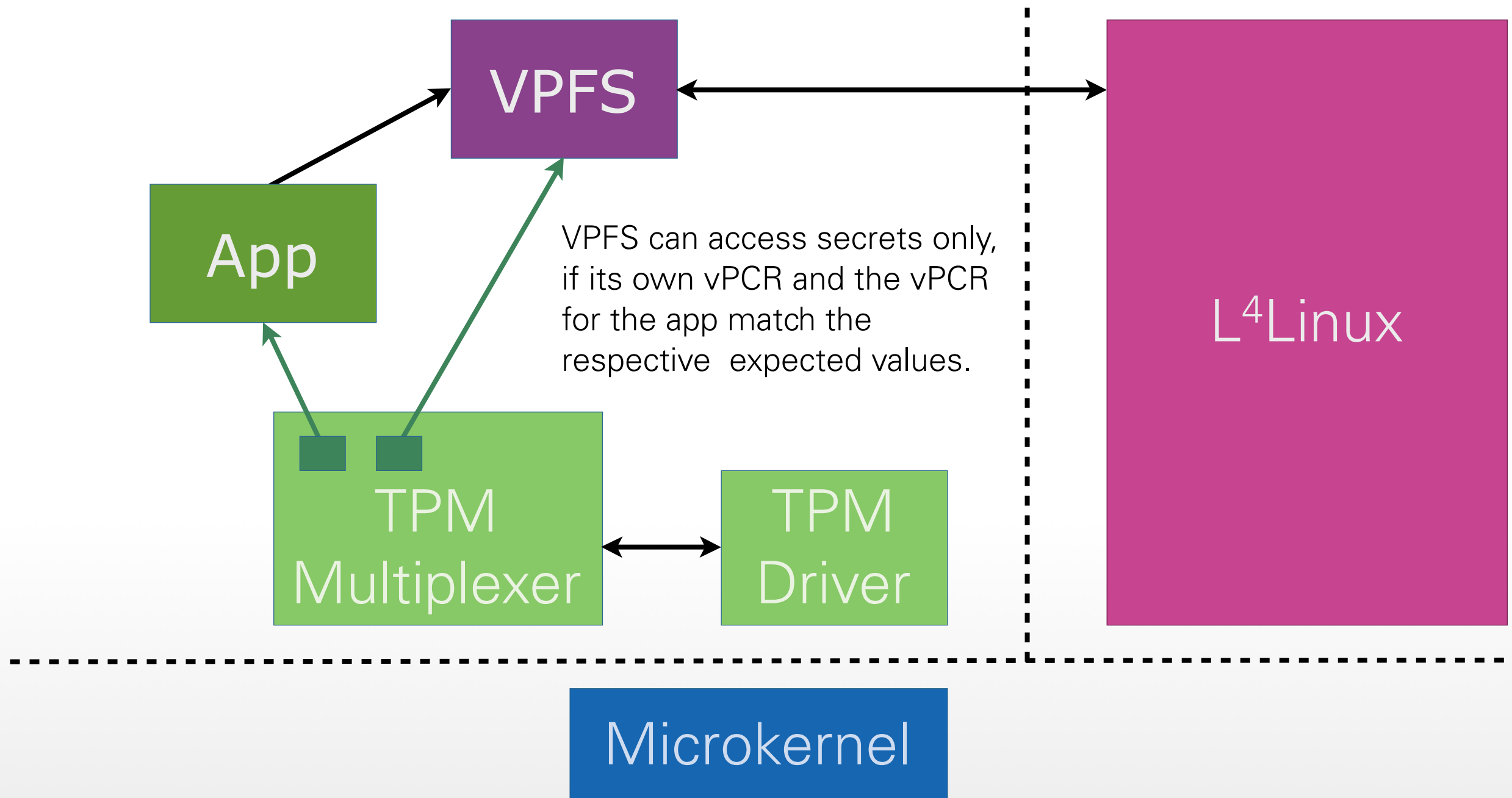
- Idea: per-application PCRs in software:
 - Measure only base system into TPM PCRs (microkernel, basic services, TPM driver, ...)
 - „Software TPM“ provides „software PCRs“ for each application
 - More flexibility with „software **PCRs**“:
 - Chain of trust common up to base system
 - Extension of chains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



- 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**

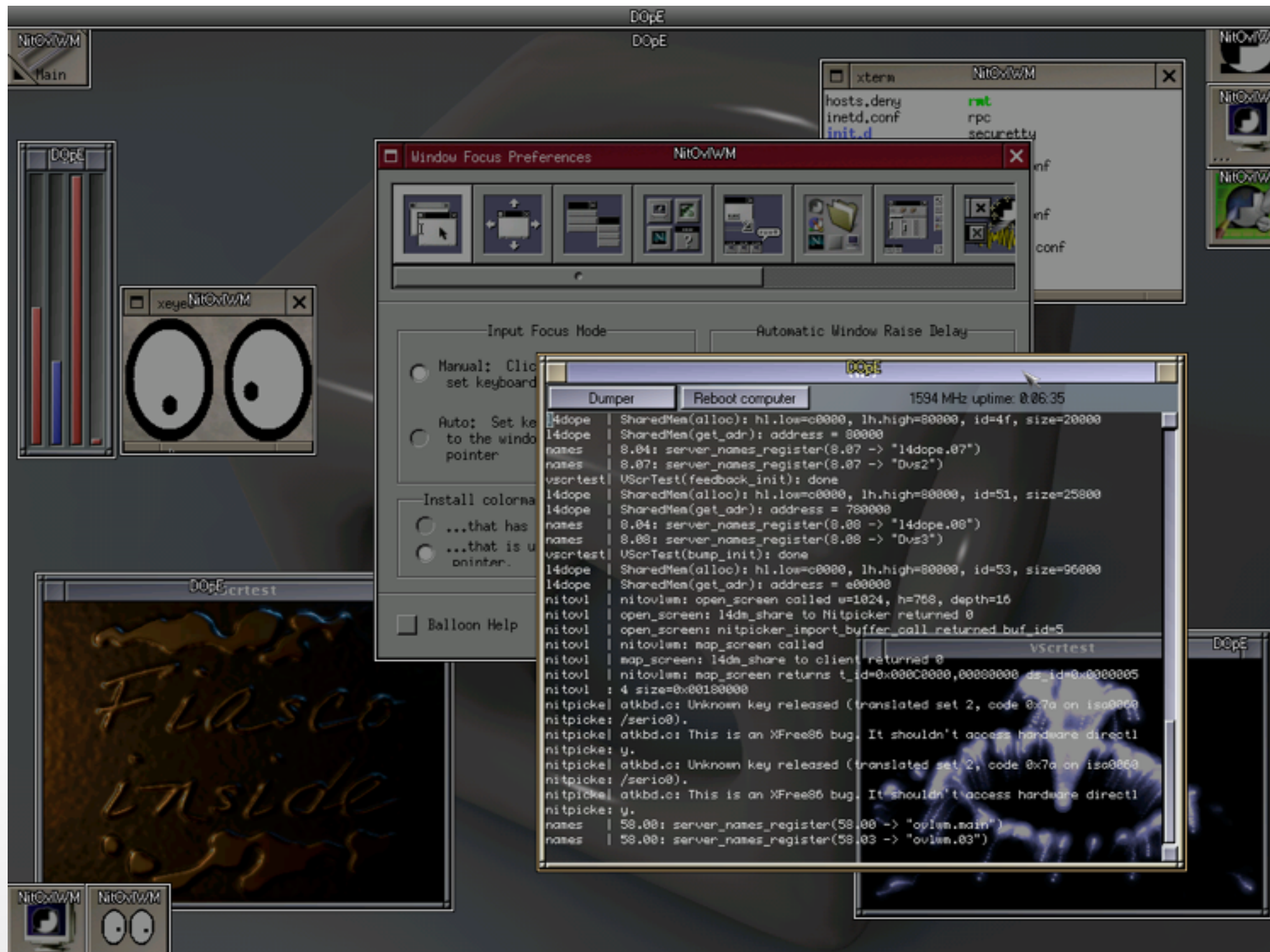
A SECOND LOOK AT VPFS





- VPFS uses **sealed memory**:
 - Secret encryption key
 - Root hash of Merkle hash tree
- Second use case is **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?“

A CLOSER LOOK AT THE WHOLE PICTURE

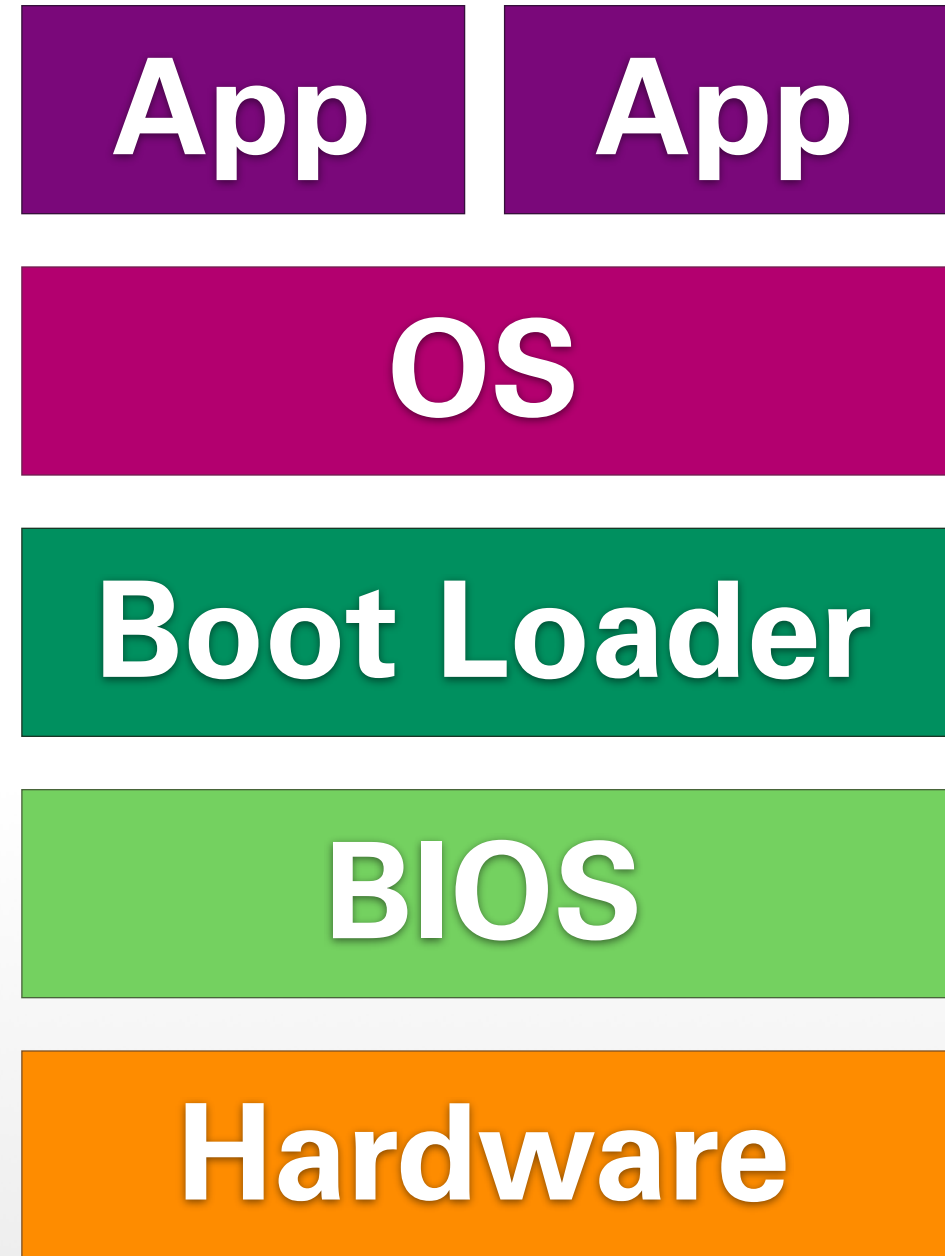


- *User cannot just trust what he / she sees on the screen!*
- Solution:
 - Remote attestation
 - For example with trusted device:
 - User's cell phone sends **nonce** to PC
 - PC replies with quote of **nonce** + **PCR** values
 - User can decide whether to trust or not

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**)

- 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?



App

The diagram illustrates a vertical stack of components in a chain of trust. From bottom to top, the components are: Hardware (orange), BIOS (light green), Boot Loader (teal), OS (magenta), and two App boxes (purple) at the top. Each component is represented by a colored rectangular box with its name in white text.

App

OS

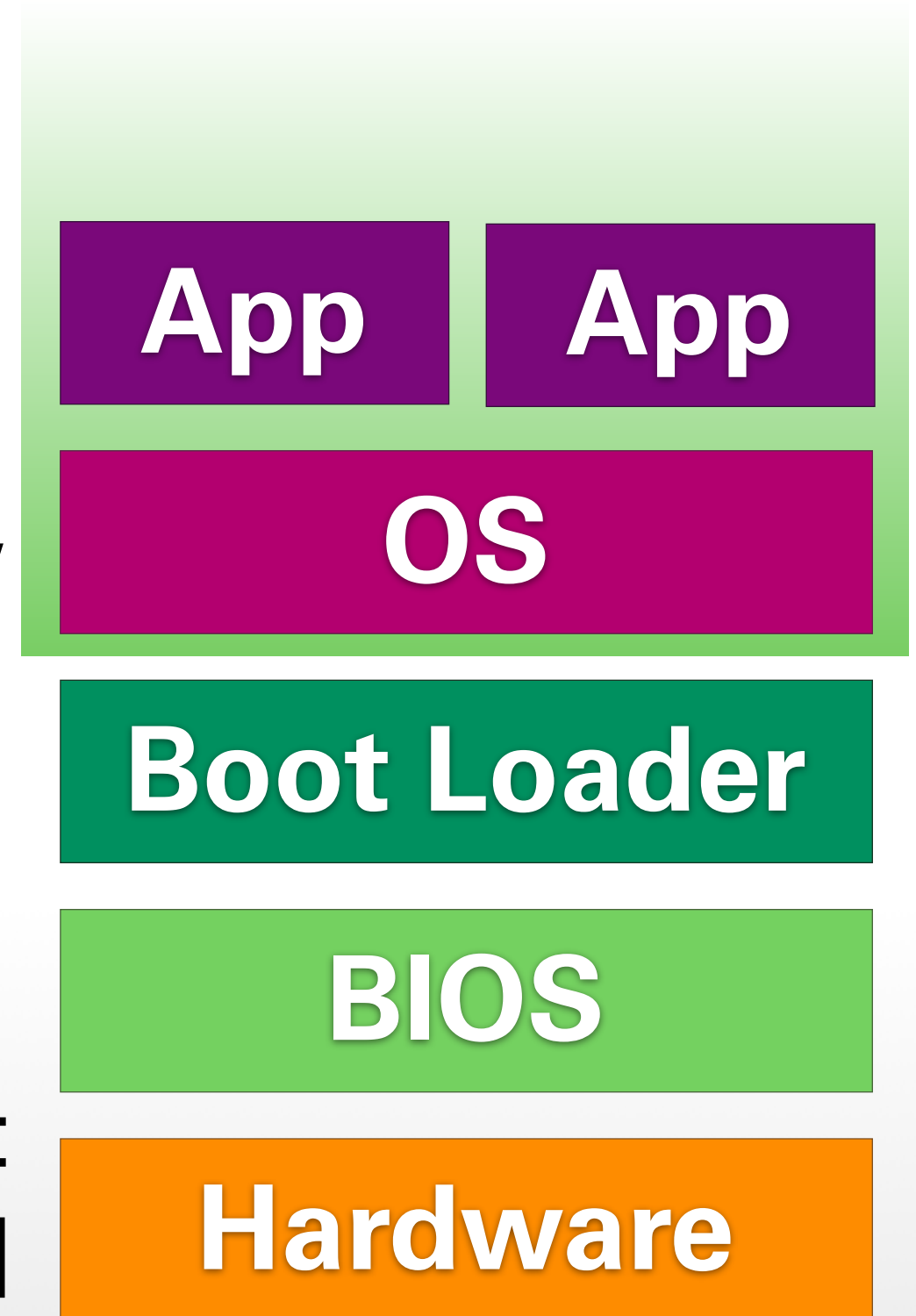
Boot Loader

BIOS

Hardware

- **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

- 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 remove boot software from TCB
- Key challenges:
 - „Secure loader“ must not be compromised
 - Requires careful checking of platform state
 - Secure loader must actually run in locked RAM, not in insecure device memory
- DRTM can also run after booting OS

- 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

App

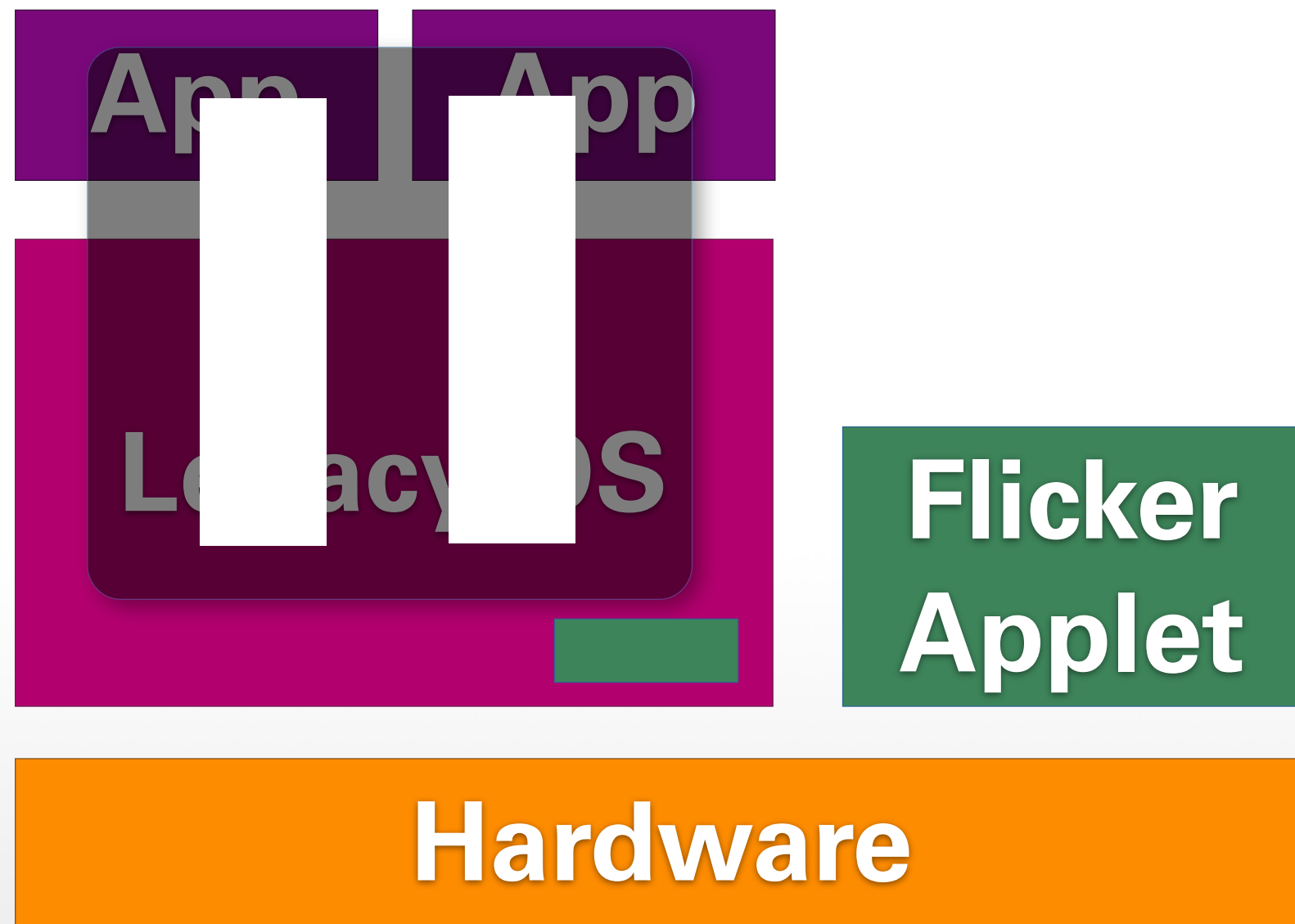
App

Legacy OS

Boot Loader

BIOS

Hardware



- 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!*

- 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)

- **ARM TrustZone [8]:**
 - New processor mode for critical software
 - Private memory partition (accessible only in secure processor mode)
 - Can be used to implement software TPM
- **Intel SGX [9]** specified, not yet available:
 - Secure enclaves: protected regions of address space for code, stack, heap
 - Sealed memory and remote attestation

- 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
 - Cryptographic co-processor: software can use (but not obtain) encryption key
- Not open: **closed** or **secure boot** instead of **authenticated booting**

- January 28, 2014:
 - Lecture „*Resilience*“
 - Paper Reading Exercise “Microkernels Meet Recursive Virtual Machines”

- [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
- [8] <http://arm.com/products/processors/technologies/trustzone/index.php>
- [9] <http://software.intel.com/en-us/intel-isa-extensions#pid-19539-1495>