Hardware-based trust and integrity verification

Federico Mancini
Federico.Mancini@ffi.no

High Integrity Day, Bergen 11.02.2014

Trust and assurance

We say that a system it **trusted** if it is relied upon to a certain extent to enforce a specified security policy

To trust the system we need some **assurance** that the security mechanisms used to enforce the policy are correctly implemented and provided under all relevant circumstances (i.e., it is trustworthy).

Common Criteria

System to evaluate
(evaluation target)

Protection profile
(properties the system needs to fulfill)

Evaluation process for assurance level (EAL)

1
2
.
6
7

EAL levels

- EAL1: Functionally Tested
- EAL2: Structurally Tested
- EAL3: Methodically Tested and Checked
- EAL4: Methodically Designed, Tested and Reviewed
- EAL5: Semi-formally Designed and Tested
- EAL6: Semi-formally Verified Design and Tested
- EAL7: Formally Verified Design and Tested

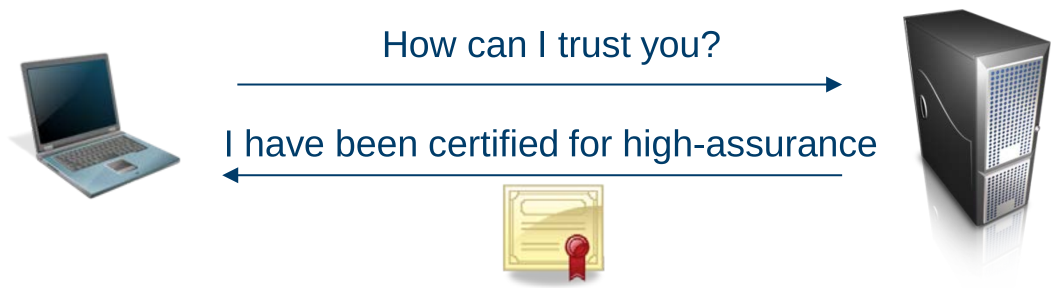
High-Assurance

PROBLEM

High-assurance levels are very difficult to obtain, very time consuming and expensive. The resulting system is an investment that must last many years to amortize the cost, and it is impossible to upgrade or modify because it would break the certification: a black box. Hence, it is often difficult to integrate in an existing infrastructure and manage it.

Trust = Certification + Integrity Protection

Given that we have a high-assurance system, how does a third party establish trust in it?



The certificate says that the system has been certified, but not the in the meanwhile it has not been compromised, i.e., that *its integrity is preserved*.

We trust it because we rely on its integrity protection mechanims, which are a requirement to achieve certification (temper-responsive and temper proof devices for instance, and formally verified software).

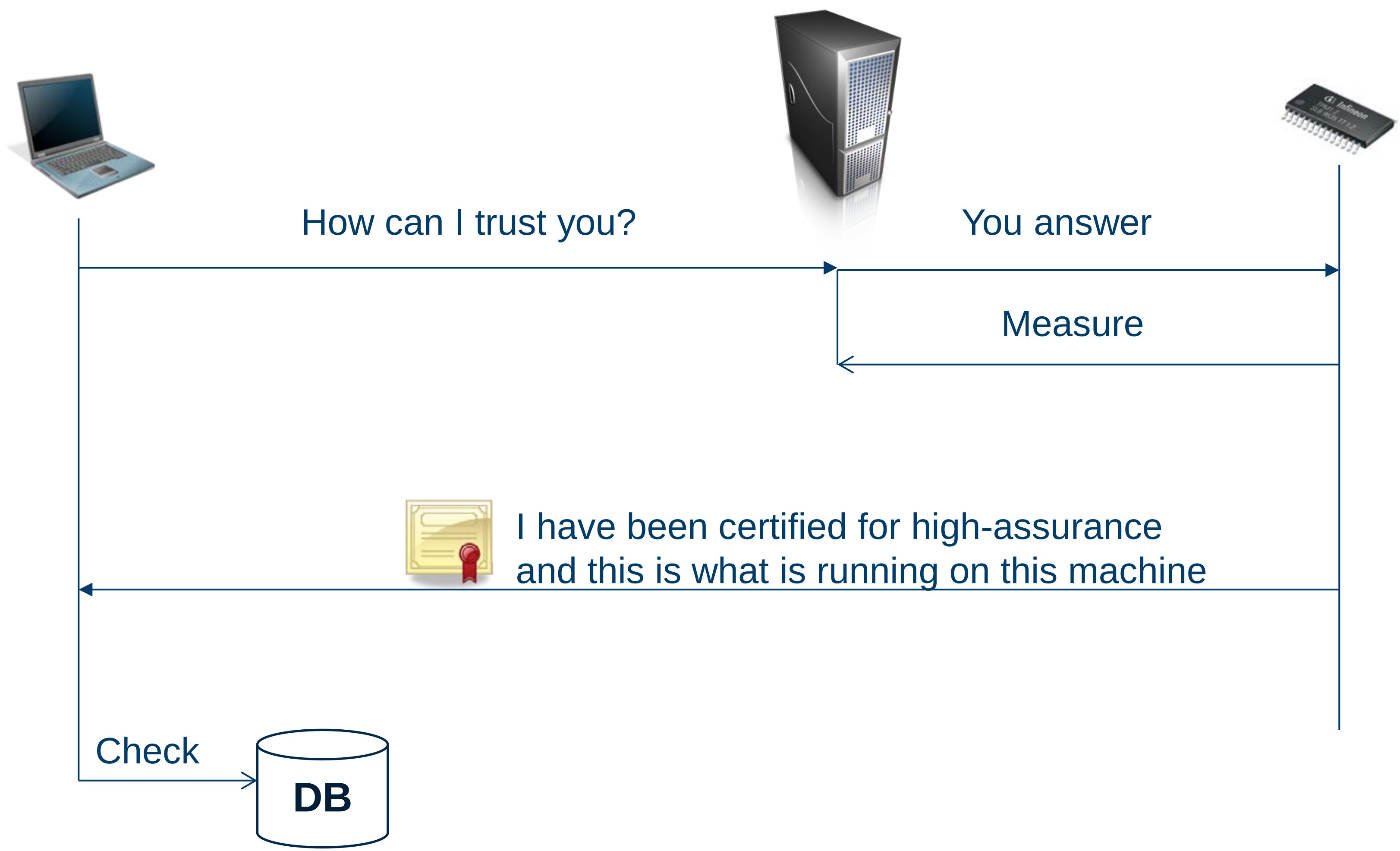
Integrity verification

Not all systems need to meet very high-assurance requirements, but maybe they need to have a specific configuration to be allowed to execute certain operations (e.g., a certain OS update with the last patches, or the latest BIOS version, or hardware with signed firmware)

How do we verify their configuration in a trustworthy manner without having to certify each machine every time the configuration changes?

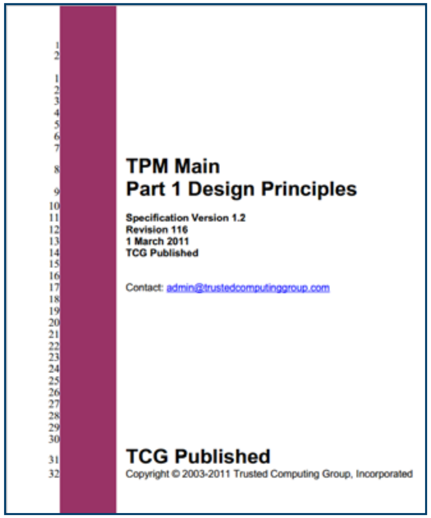
We need an independent entity that can measure and report the system integrity in a trustworthy manner. Then only this part of the system would need to be certified.

Integrity measuring and reporting



The trusted platform module - TPM

- The **TPM (Trusted Platform Module)** is both a set of specifications and its implementation.
- The TPM is a *passive device* (it can only perform actions if asked to), soldered to the motherboard, that can be used to perform some cryptographic operations in a protected environment.



TPM Main
Part 1 Design Principles

Specification Version 1.2
Revision 116
1 March 2011
TCG Published

Contact: admin@trustedcomputinggroup.com

TCG Published
Copyright © 2003-2011 Trusted Computing Group, Incorporated

- **Main Goal**: increase trust in a platform

TPM main functionalities

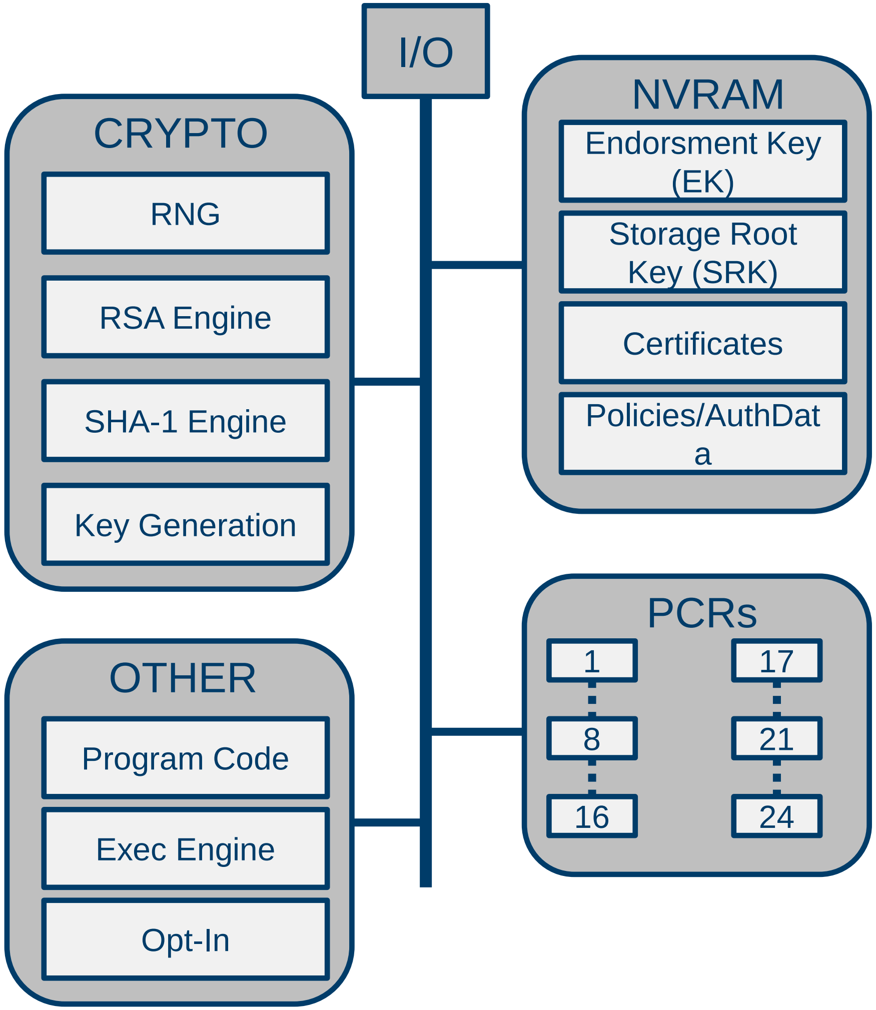
- **Better cryptographic services:**
  - Hardware protected crypto operations
  - Hardware protected data encryption
  - Hardware protection against password guessing

≈ Smart cards

- **New functionalities:**
  - Platform integrity protection (*Trusted Boot)*
  - Platform Attestation
  - Sealing
  - Anonimity

Inside a TPM

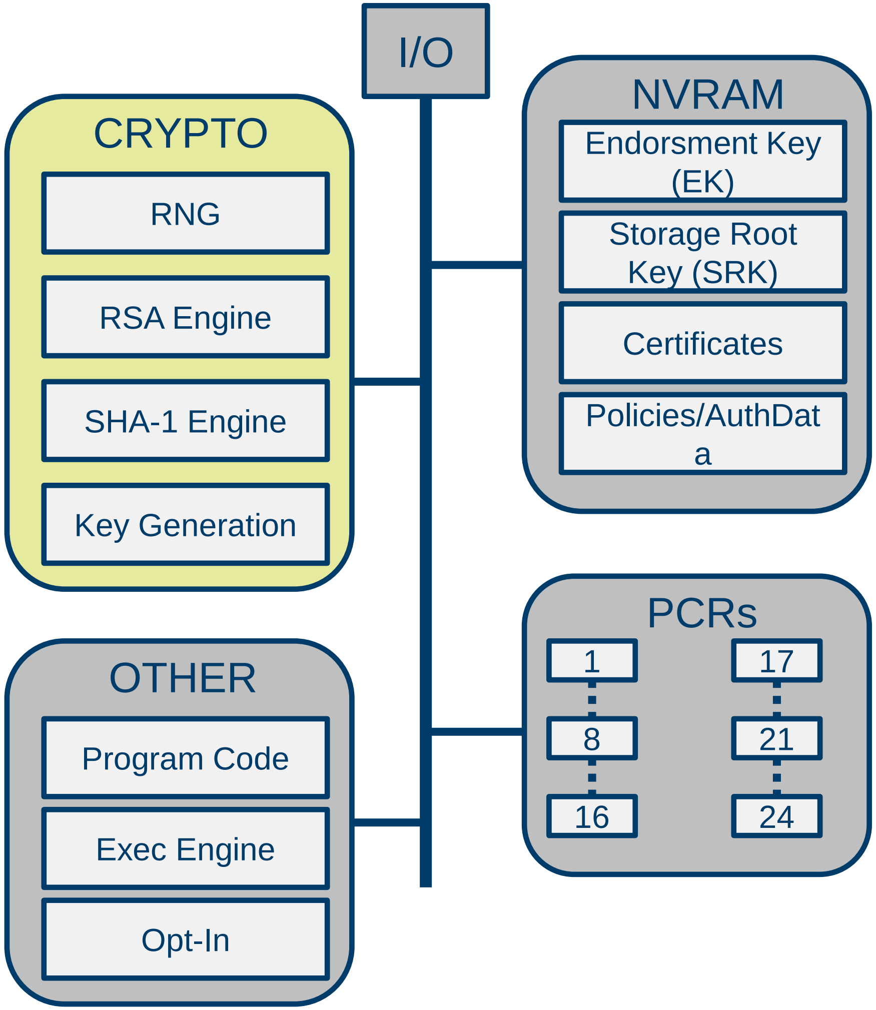
TPM1.2
SLB 9635 TT 1.2
Infineon



I/O
CRYPTO
RNG
RSA Engine
SHA-1 Engine
Key Generation
NVRAM
Endorsment Key (EK)
Storage Root Key (SRK)
Certificates
Policies/AuthData
OTHER
Program Code
Exec Engine
Opt-In
PCRs
1
8
16
17
21
24

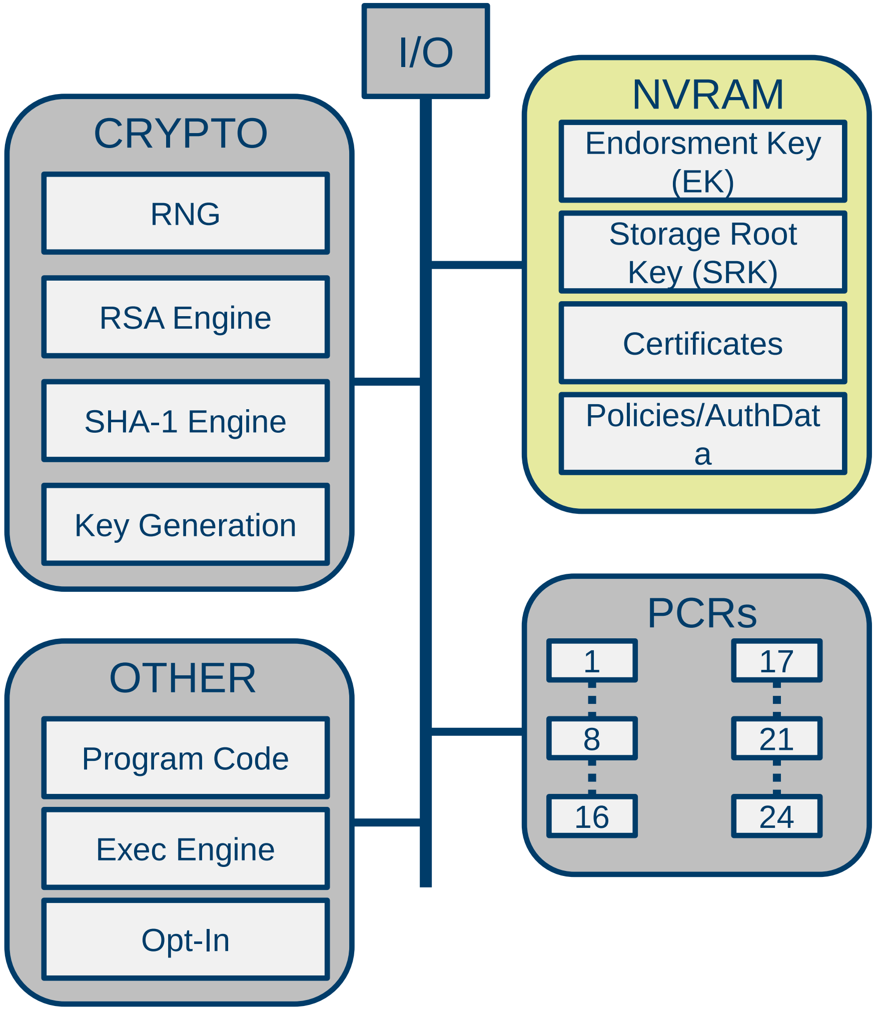
TPM - CRYPTO

- **RNG:** (True) Random Number Generator
- **SHA-1 Engine:** To compute hashes
- **RSA Engine:** For encryption, decryption and signing with asymmetric keys.
- **Key generator:** To generate RSA key pairs and symmetric keys.



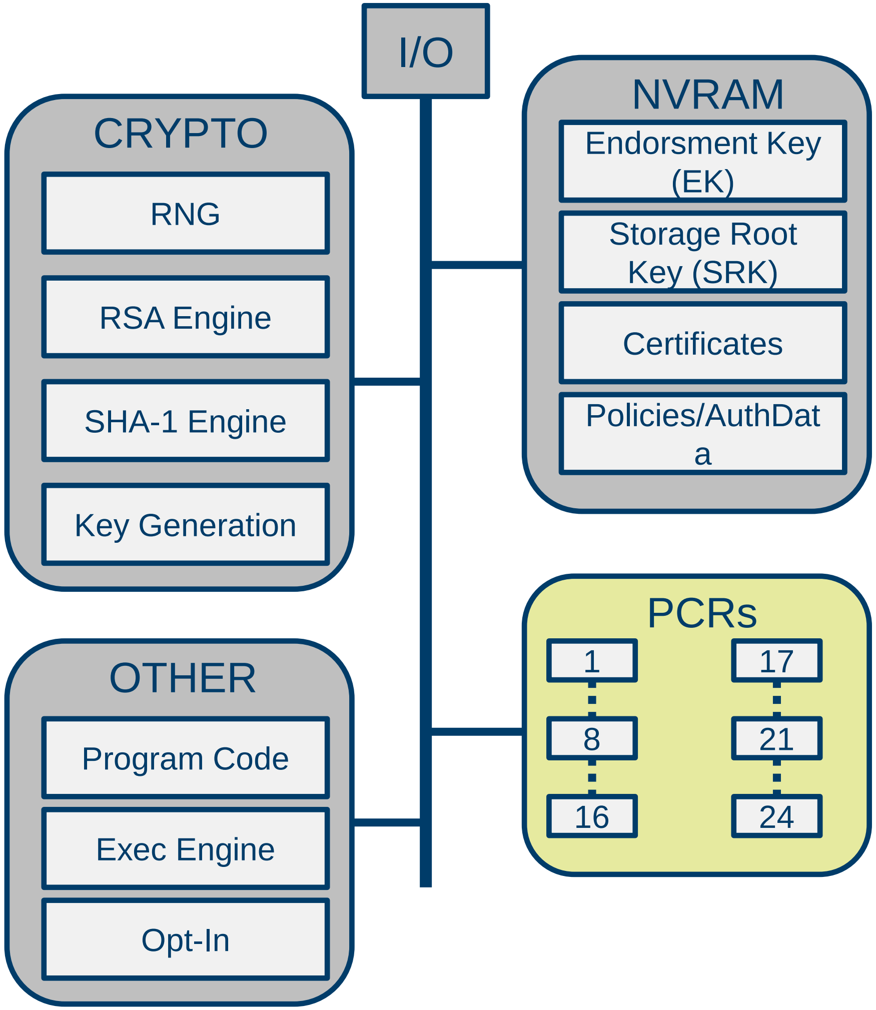
Non-Volatile Memory

- **EK:** installed by manufacturer. Unique per TPM.
- **SRK:** created when user takes ownership of TPM.
- **Certificates:** Manufacturer, Conformance Entity, Validation Entity, Trusted Third Party
- **Policies/AuthData:** shared secret to access objects (authentication+authorization)



Platform Configuration Registers - PCR

- 20 bytes registers to store SHA-1 hashes.
- Cannot be written directly, only extended: *PCR = SHA-1(Current value || new hash)*
- 1-8 reserved. At least 16 must be present.
- They are always reset at boot time and only then.

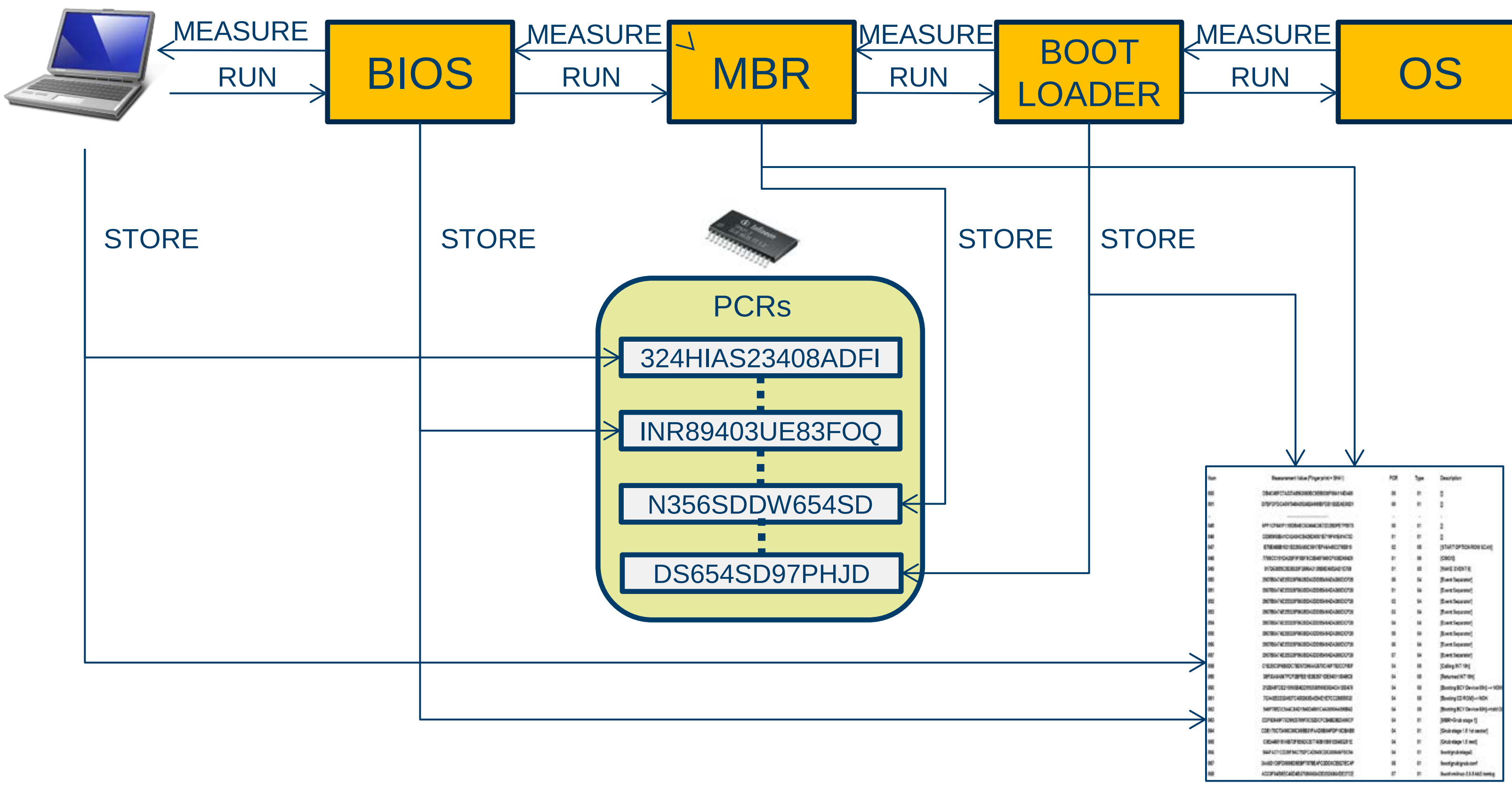


What can you do with a TPM?

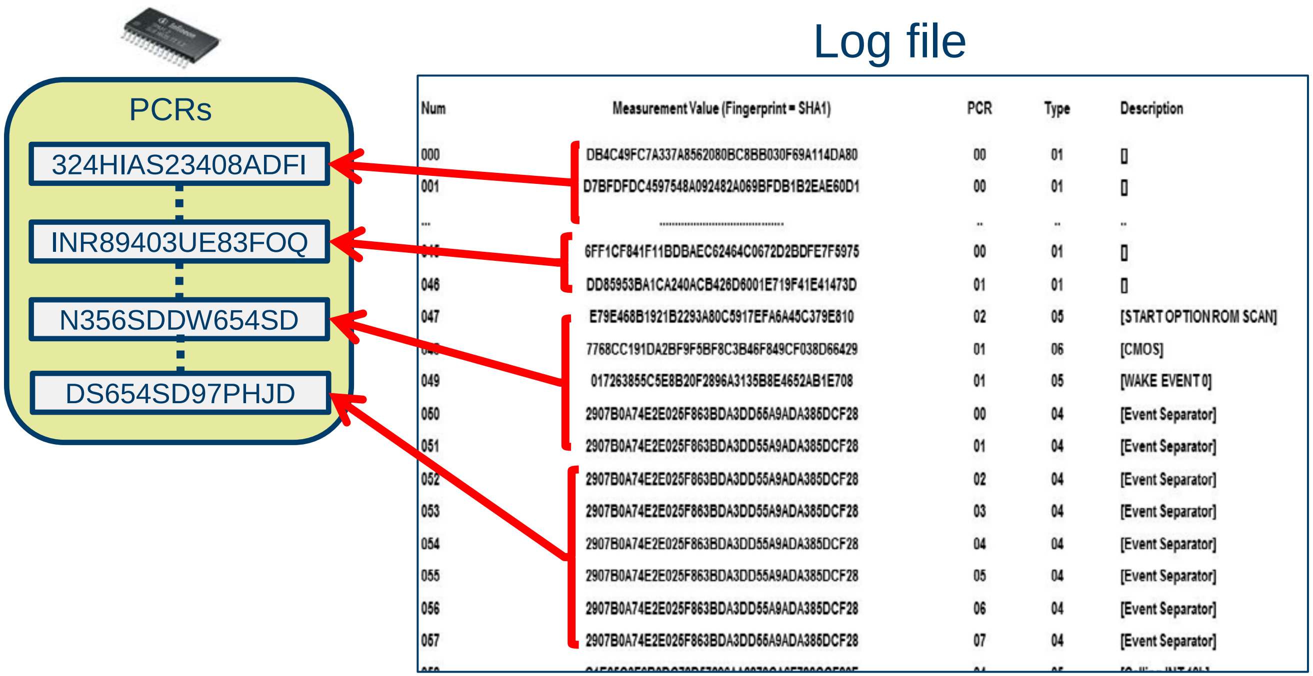
- Trusted Boot
- Secure Storage
- Remote Attestation

Trusted Boot

Each component involved in the boot process is measured, and the measurement stored both in the TPM PCRs and in a Log File.



Integrity protection

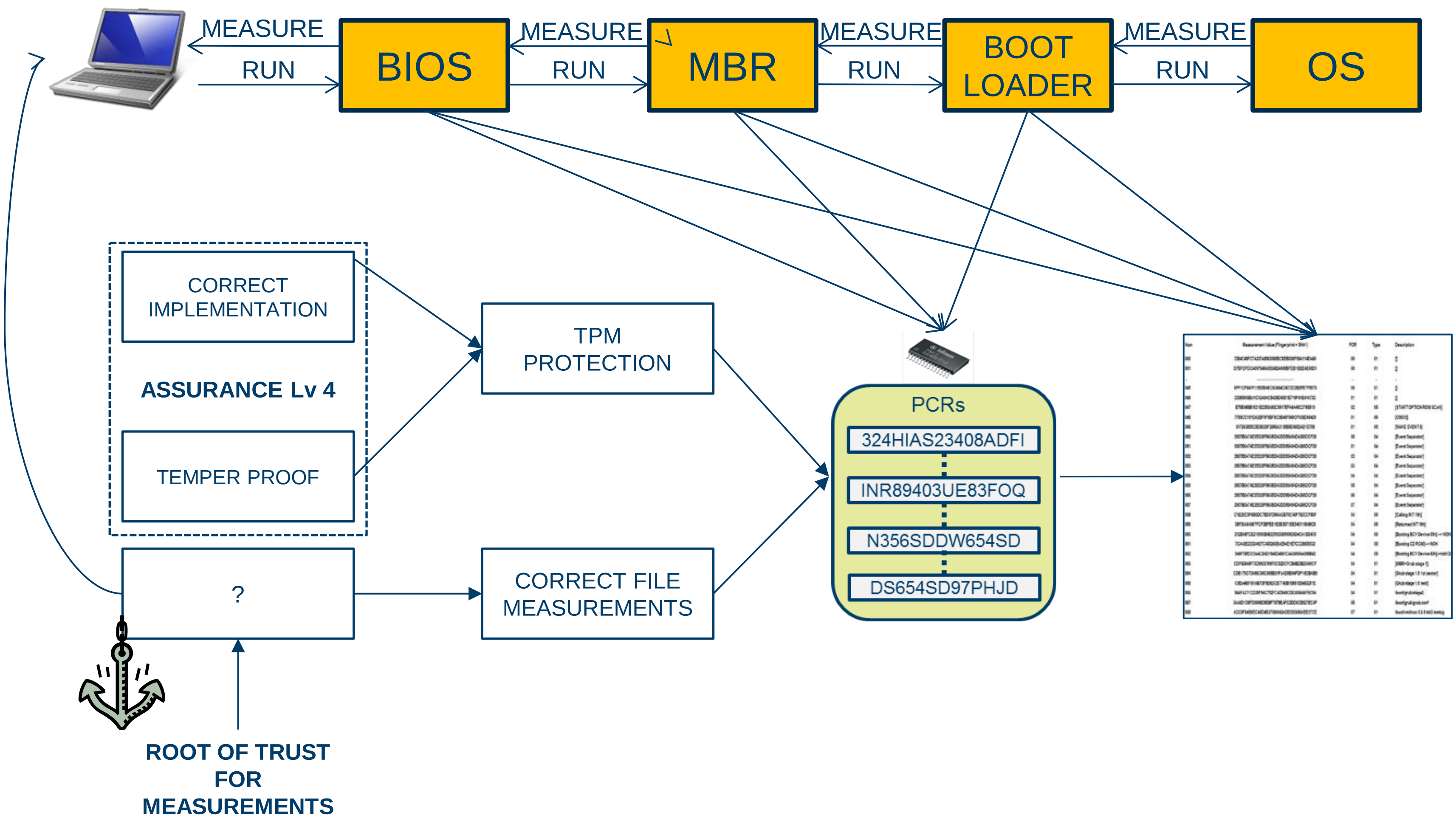


Num	Measurement Value (Fingerprint = SHA1)	PCR	Type	Description
000	DB4C49FC7A337A8562080BC8BB030F69A114DA80	00	01	[]
001	D7BFDFDC4597548A092482A069BFDB1B2EAE60D1	00	01	[]
...	....	..	..	..
[illegible]	6FF1CF841F11BDBAEC62464C0672D2BDFE7F5975	00	01	[]
046	DD85953BA1CA240ACB426D6001E719F41E41473D	01	01	[]
047	E79E468B1921B2293A80C5917EFA6A45C379E810	02	05	[START OPTION ROM SCAN]
[illegible]	7768CC191DA2BF9F5BF8C3B46F849CF038D66429	01	06	[CMOS]
049	017263855C5E8B20F2896A3135B8E4652AB1E708	01	05	[WAKE EVENT 0]
050	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	00	04	[Event Separator]
051	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	01	04	[Event Separator]
052	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	02	04	[Event Separator]
053	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	03	04	[Event Separator]
054	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	04	04	[Event Separator]
055	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	05	04	[Event Separator]
056	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	06	04	[Event Separator]
057	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	07	04	[Event Separator]

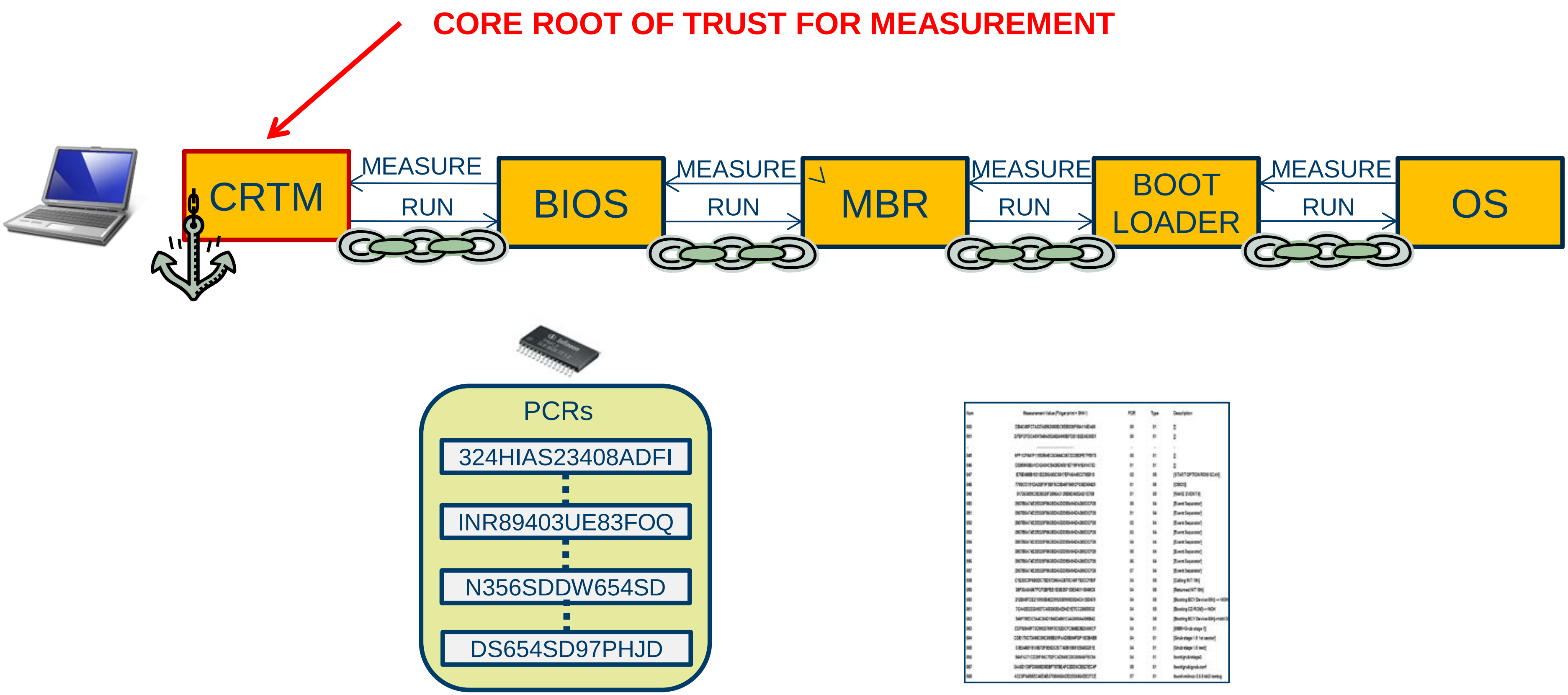
PCR values can be used to verify the integrity of the log file

- Why should we trust the PCR values?
- What if a malware was installed that stored fake measurements?
- Who measured the system?

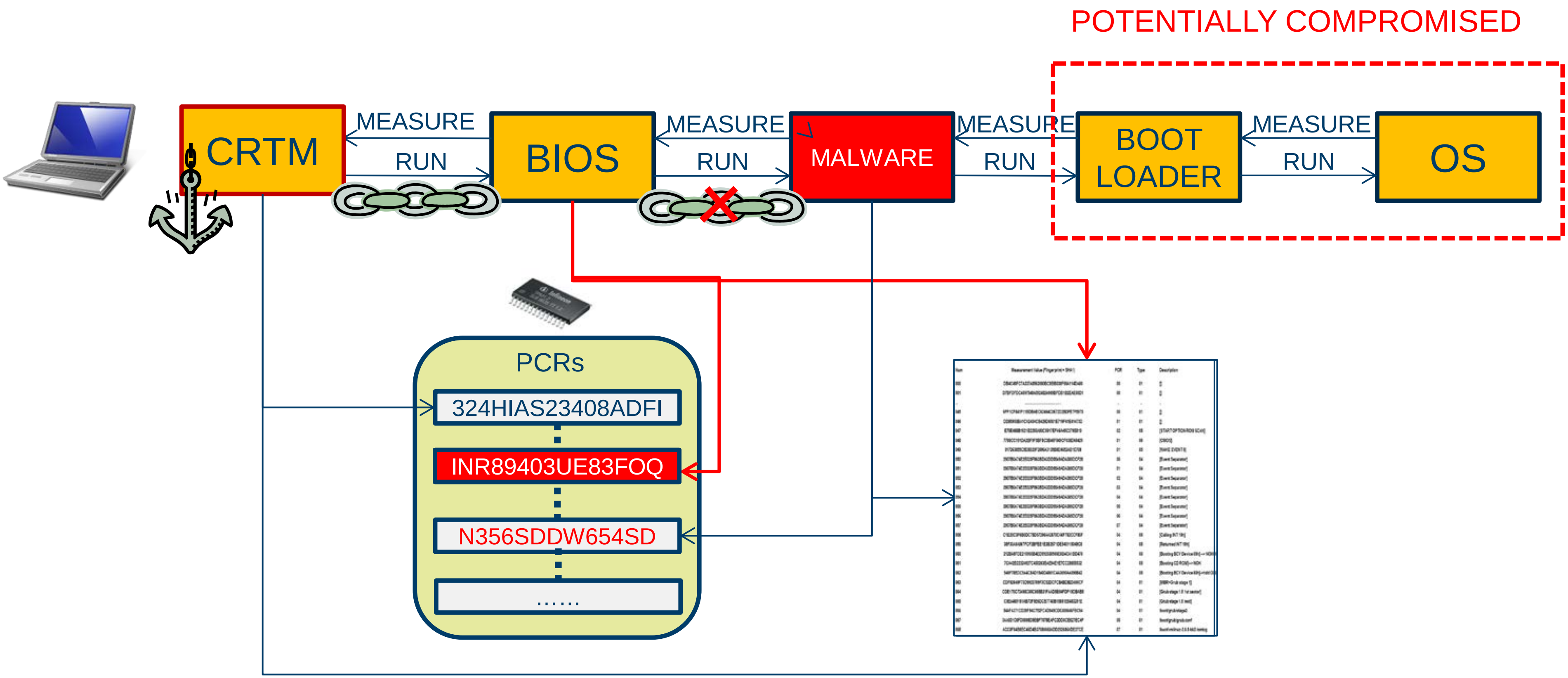
Where does the trust come from?



Root of Trust for Measurement



Root of Trust for Measurement



Guarantee that there is always a component that will measure the malware

Trusted boot and attestation

The computer has started and everything has been measured. This is going to happen no matter the measurements. So what now? Who verifies the integrity?

Prove to a third party the integrity of your platform: **Remote Attestation**.

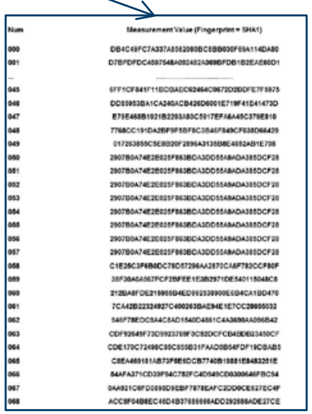
Seal computer to current configuration so that it is unusable if someone tamper with it: **BitLocker**.

PCRs

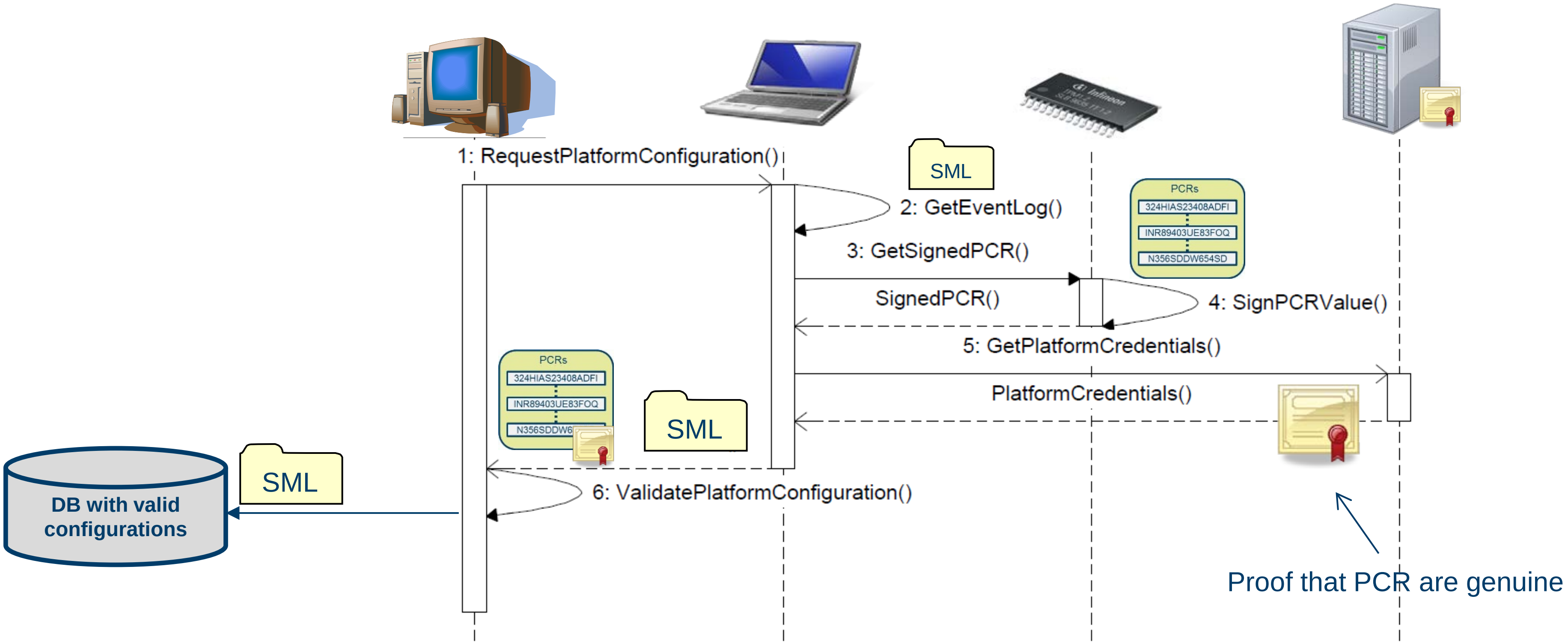
324HIAS23408ADFI

INR89403UE83FOQ

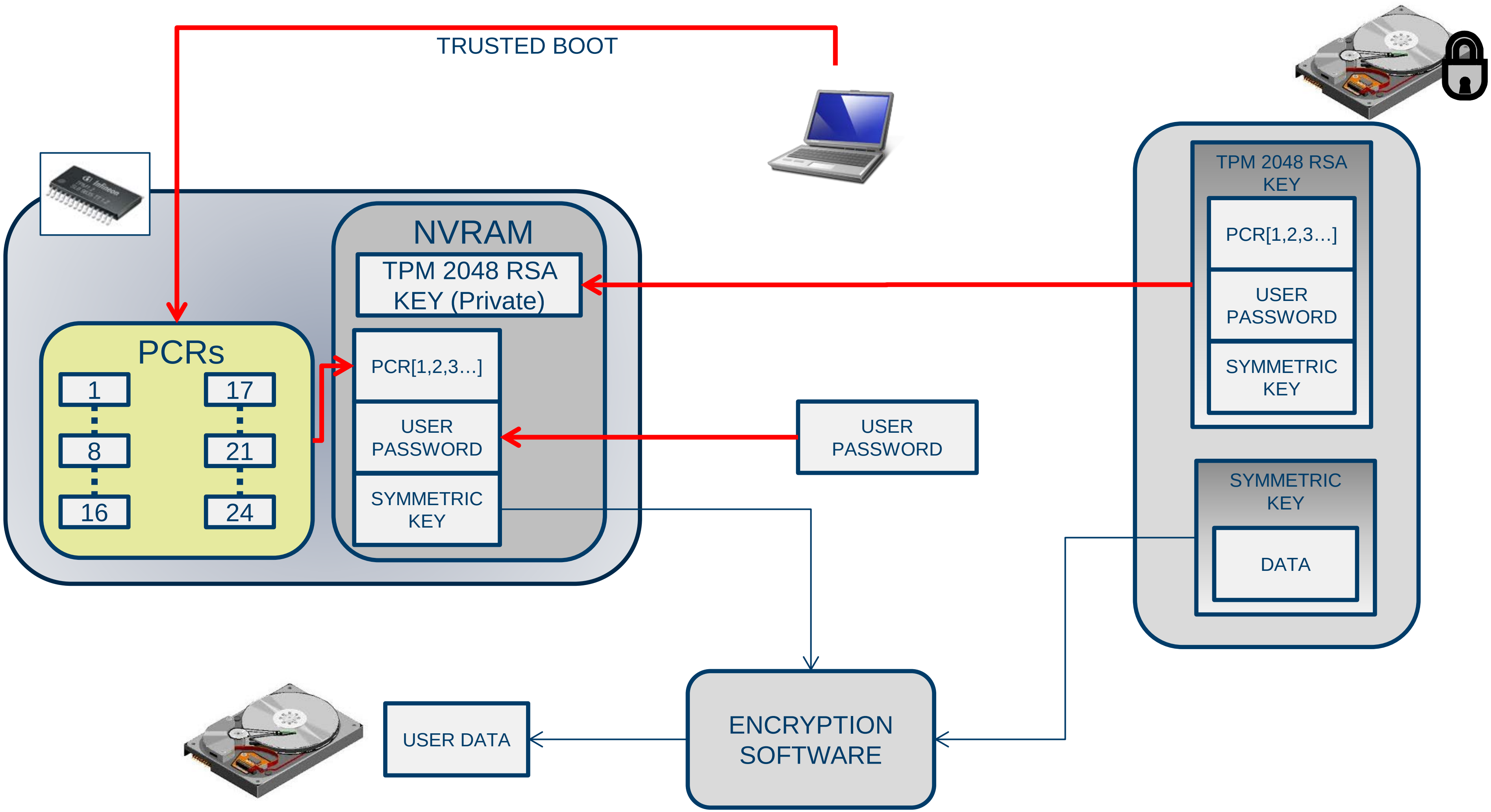
N356SDDW654SD



Attestation protocol: Root of trust for Reporting



Sealing/Binding



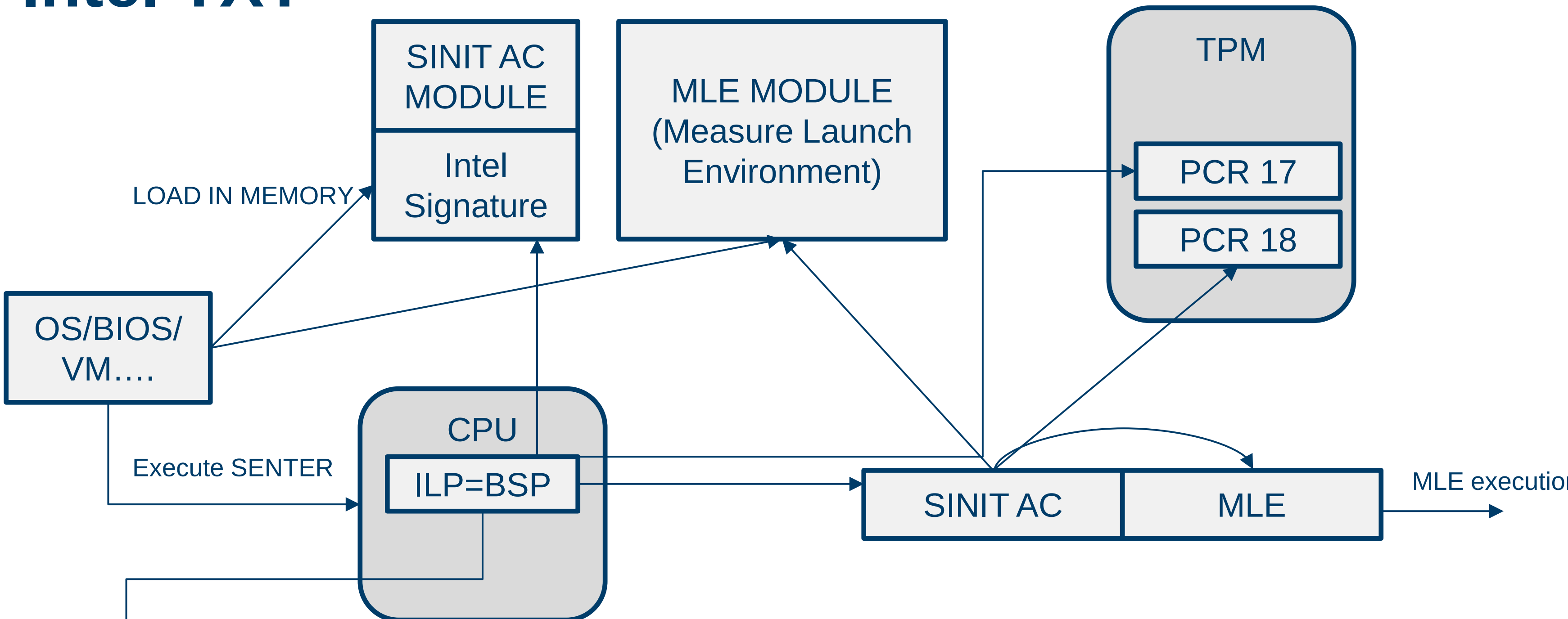
Problems

- We cannot say much about what happens «after» the OS takes control
- We need to maintain a potentially huge database of valid platform configurations
- We need an infrastructure parallel to PKI to manage TPM certificates

Protecting critical applications

- **Trusted Execution Environment (TEE) and Dynamic Root of Trust:** A secure and sanitized environment is created in hardware on the fly in order to run code securely, even if the system is compromised. TPM can be used to attest that code was securely run. Implementations:
  - Intel TXT
  - AMD-V
  - ARM TrustZone
- **Separation Kernel/MILS:** A secure separation kernel or hyper-visor is securely loaded with trusted boot, and different security domains are run in parallel. One domain is dedicated to TPM operations, so that the user or other processes cannot interphere.

Intel TXT



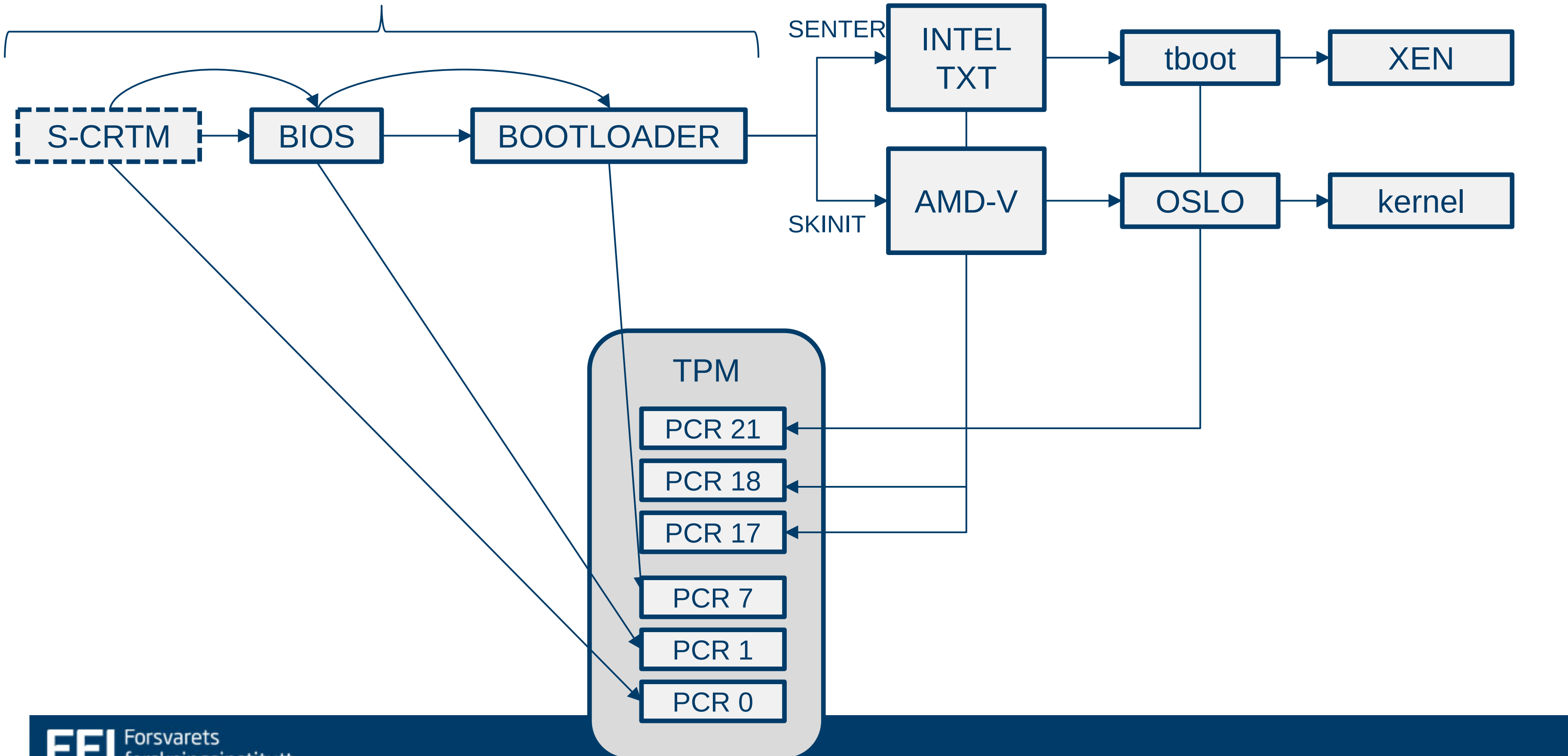
1. Stop other processors
2. Mask all external events
3. Validate SINIT AC signature
4. Reset PCR 17-20 to a special value and extend 17 with SINIT hash
5. Unlock the chipset , load the SINIT AC and pass control to it

1. Test Hardware configuration
2. Initialize SMM handling
3. Enable DMA handling
4. Load and measure MLE
5. Store MLE hash in TPM
6. Pass control to MLE

Example of MLE: tboot and OSLO [8]

Dynamic root of trust:
BIOS out of trusted base, can be executed at any moment

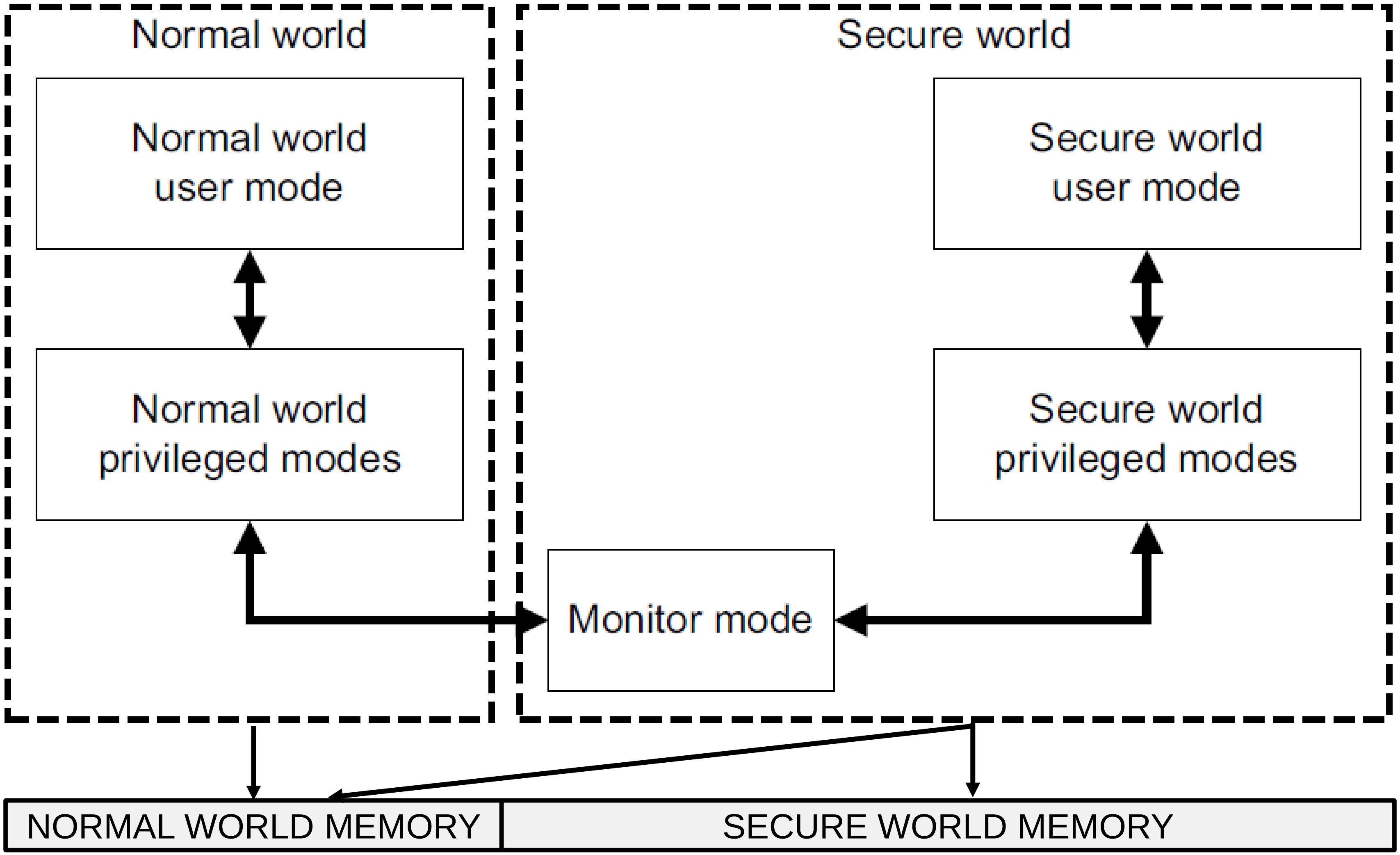
Static root of trust and trusted boot:
Must trust BIOS
Possible only at booting time



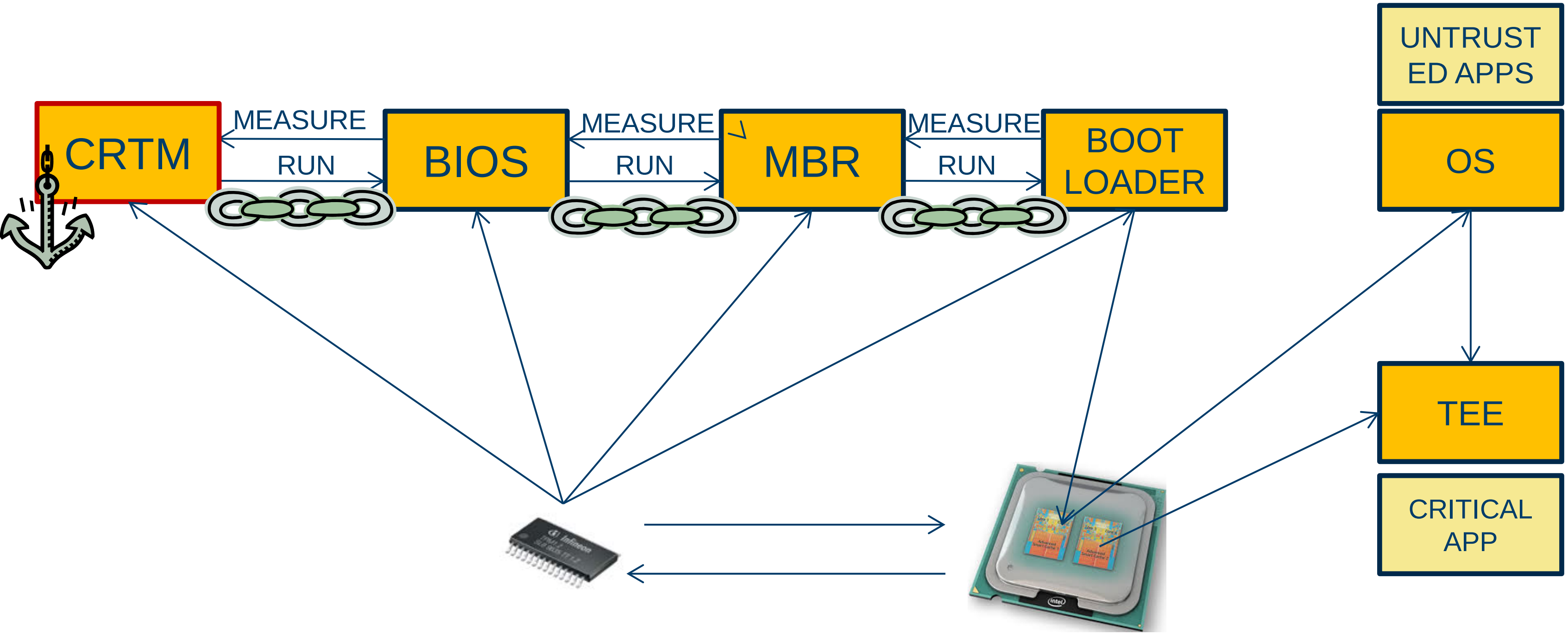
Problems with DRTM

- It is designed mainly as support to virtualization, i.e., to securely launch hypervisors like XEN.
- It has been already broken in various ways (together with XEN itself)
- We still need to trust the Hypervisor/Kernel/OS to execute our application securely at runtime.

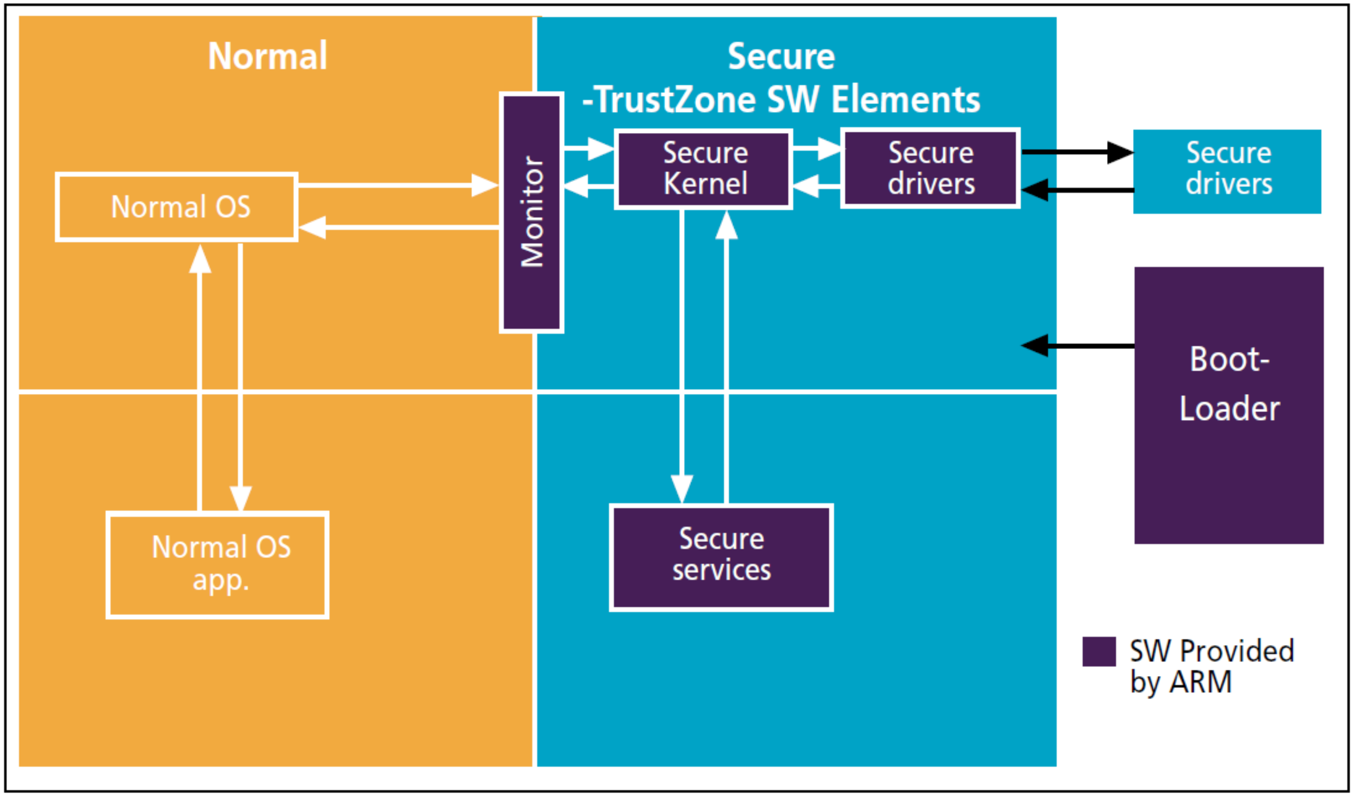
ARM TrustZone



Complete picture: TPM + INTEL TXT



Complete picture: ARM TRUSTZONE



Conclusions

- TPM and DRTM can help greatly to increase trust in commodity PC and servers
- Huge growth in the last year (1 billion TPM deployed)
- Recently approved by NSA for use in public offices to improve security
- Tighter integration with commercial OSes (Windows 8, Chrome OS)
- Still a lot of problems to fix, but great potential