Forsvarets forskningsinstitutt

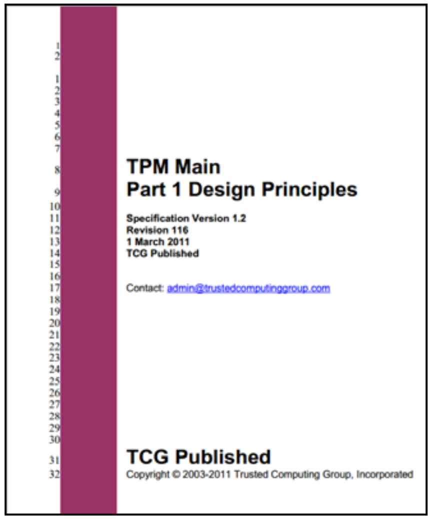
From TPM 1.2 to 2.0 and some more

Federico Mancini

AFSecurity Seminar, 30.11.2015

The trusted platform module - TPM

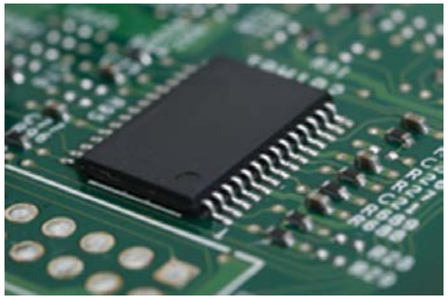
- The **TPM (Trusted Platform Module)** is both a set of specifications and its implementation.
- In particular a library as of 2.0
- The TPM is a *passive device* (it can only perform actions if asked to), soldered to the motherboard, that can be used to perform some cryptographic operations in a protected 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

Some context: Trust vs Security

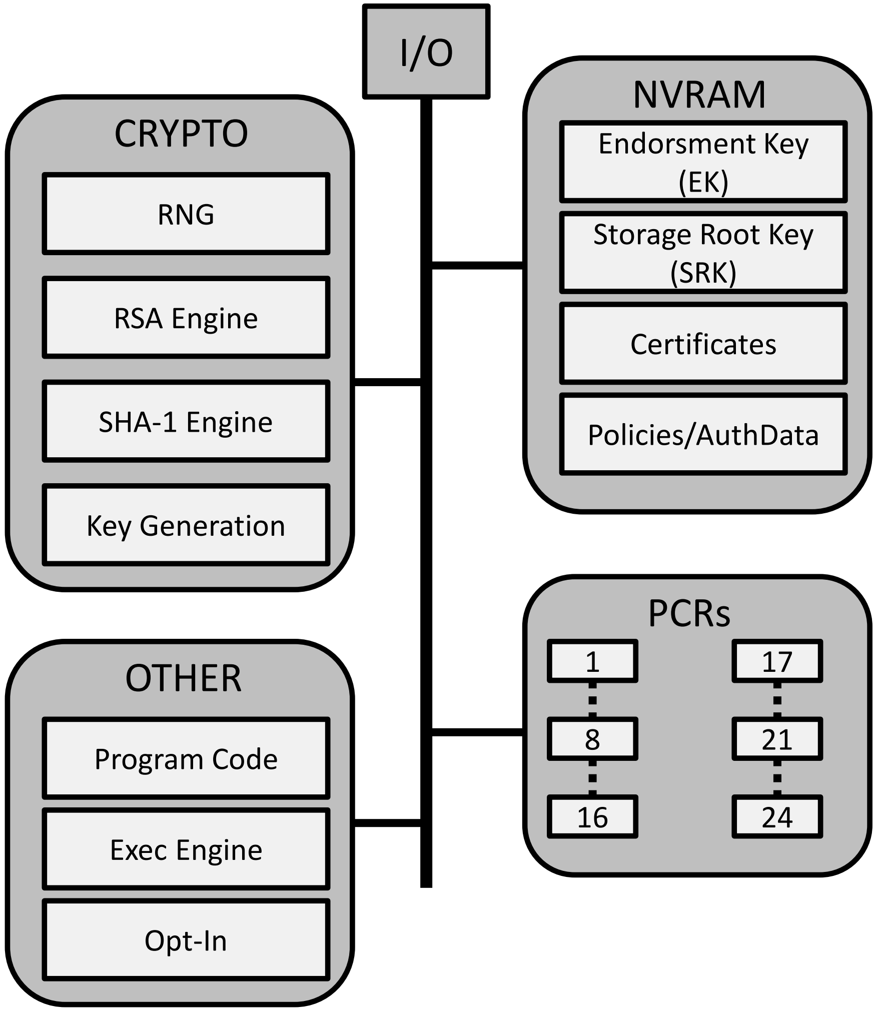
- A trusted system is one that behaves as expected, not a guarantee
- A trustworthy system is one whose behaviour can be predicted – the base on which one can decide whether to put their trust in the system or not (trustworthy computing)
- A secure system is one that has been certified to enforce correctly and reliably some specific security policy

- The TPM implementation can be secure, but not the platform on which it is attached
- The platform will be trusted to report certain values in a correct way, because it uses the TPM to do so
- I can take a decision about whether to further trust the platform with other tasks based on the TPM supported funcionalities

I can trust that a system is running Windows because the TPM says so. That does not make it secure.

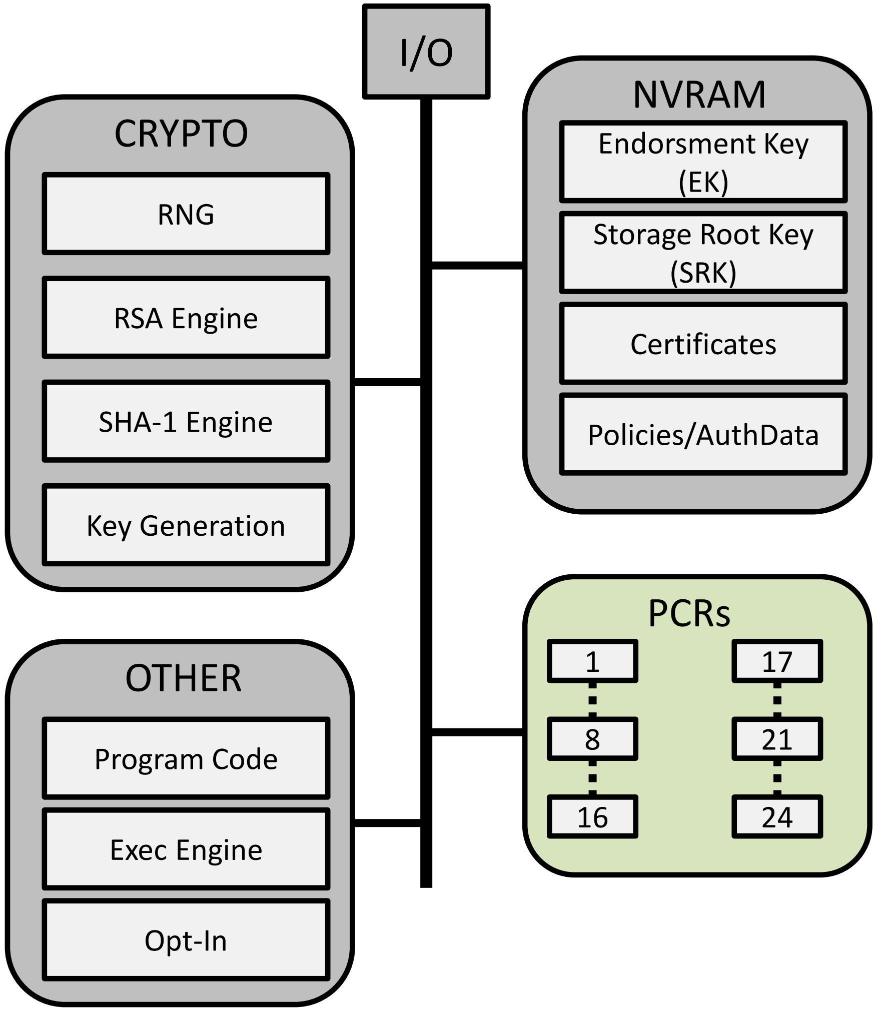
Inside a TPM

Infineon
TPM1.2
SLB 9635 TT 1.2


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

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 24 must be present.
- They are always reset at boot time and only then.



TPM main functionalities

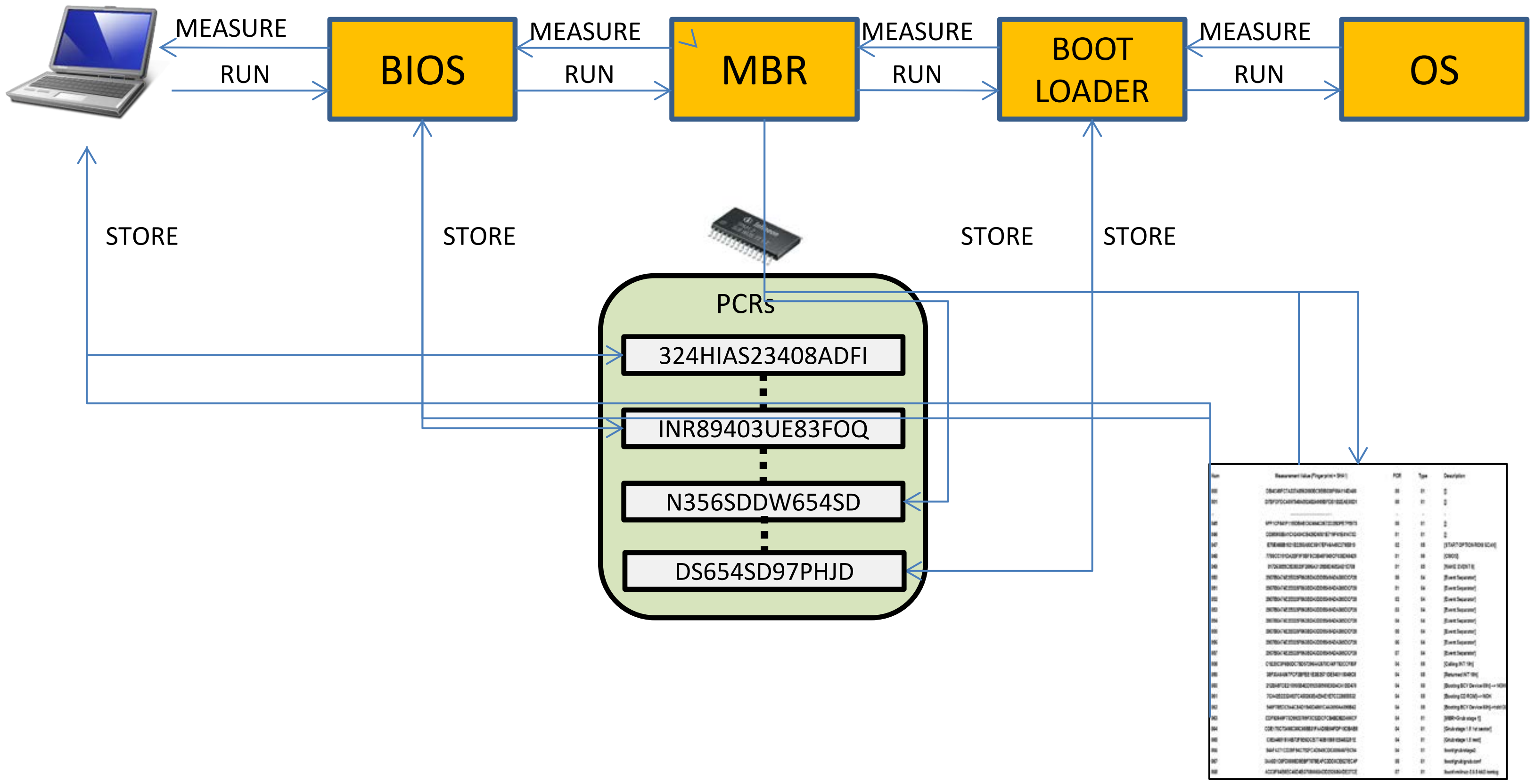
- **Better cryptographic services:**
  - Hardware protected crypto operations
  - Hardware protected data encryption
  - Hardware protection against password guessing

≈ Smart cards

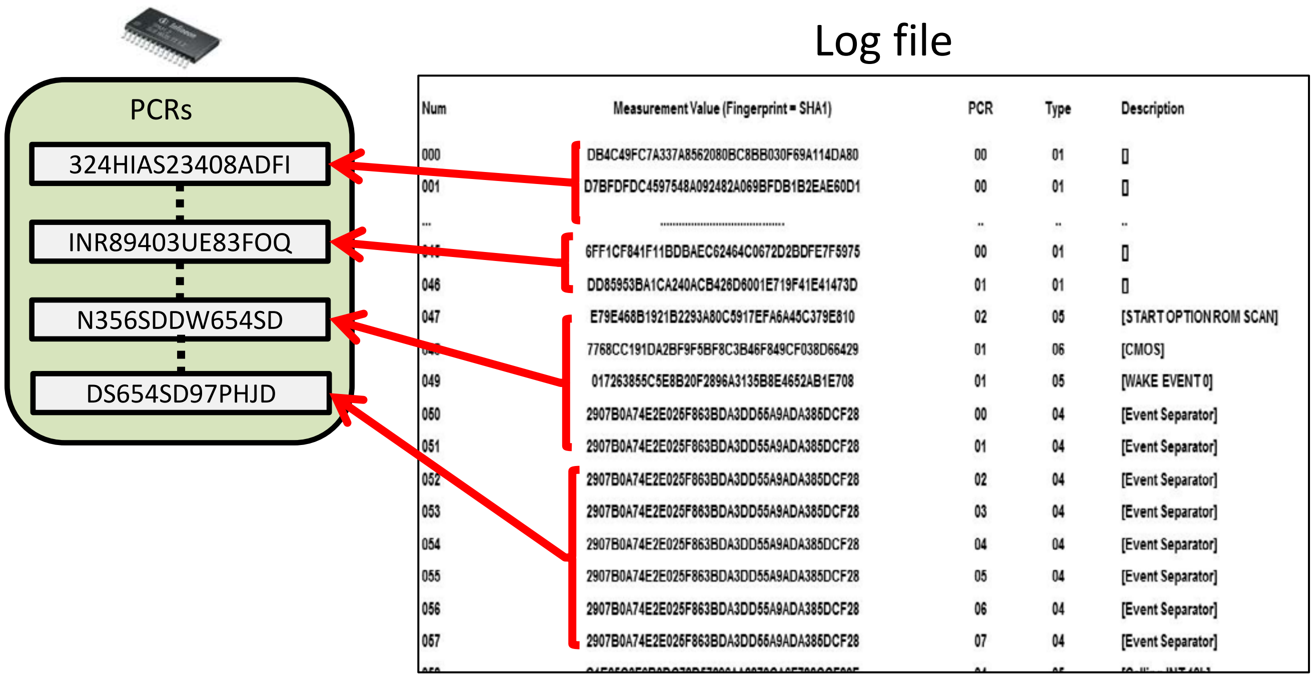
- **New functionalities:**
  - Platform integrity protection (*Trusted Boot)*
  - Platform Attestation
  - Sealing
  - Anonimity

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

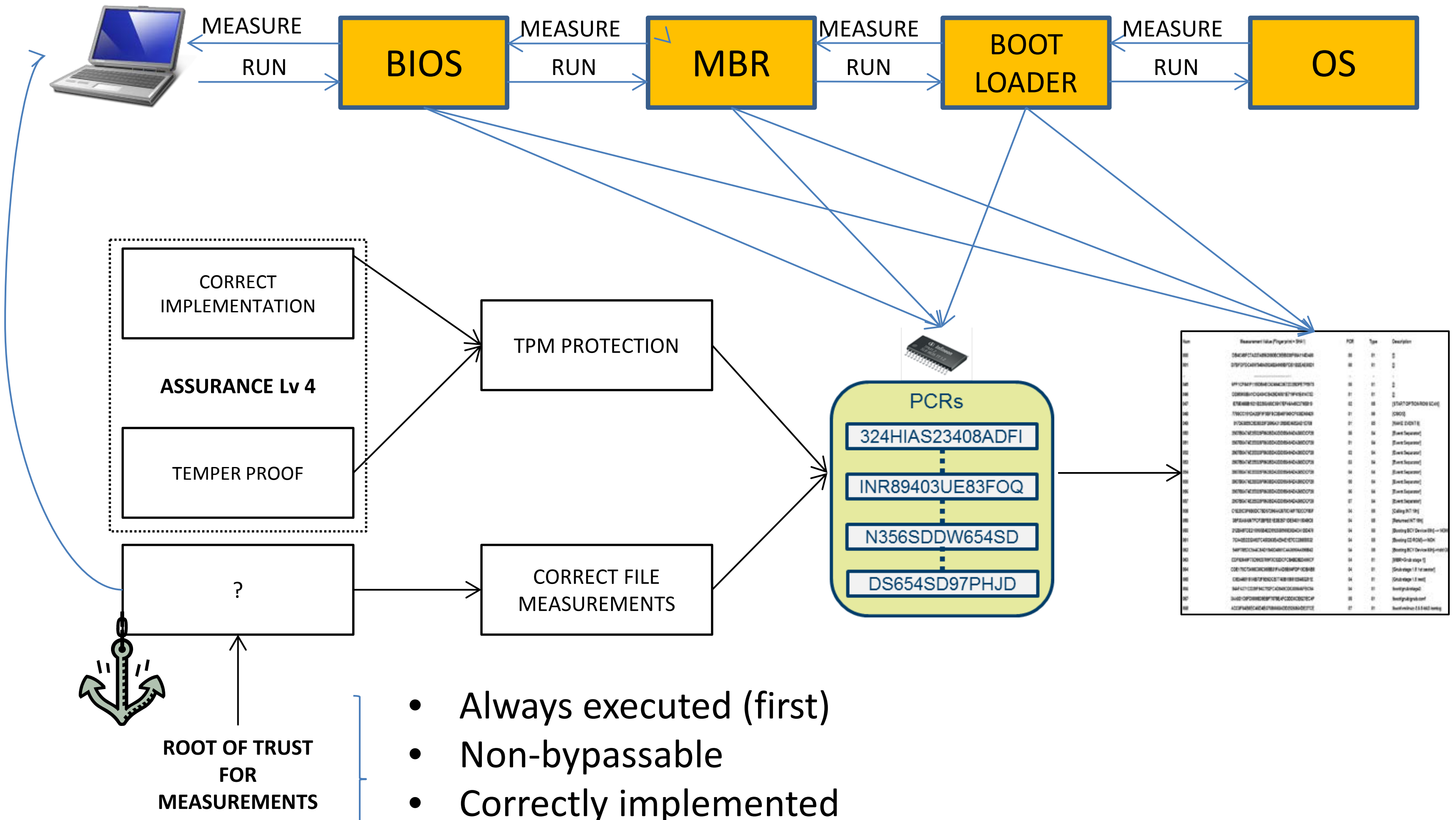


Num	Measurement Value (Fingerprint = SHA1)	PCR	Type	Description
000	DB4C49FC7A337A8562080BC8BB030F69A114DA80	00	01	▯
001	D7BFDFDC4597548A092482A069BFDB1B2EAE60D1	00	01	▯
...	....	..	..	..
[illegible]	6FF1CF841F11BDBAEC62464C0672D2BDFE7F5975	00	01	▯
046	DD85953BA1CA240ACB426D6001E719F41E41473D	01	01	▯
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?

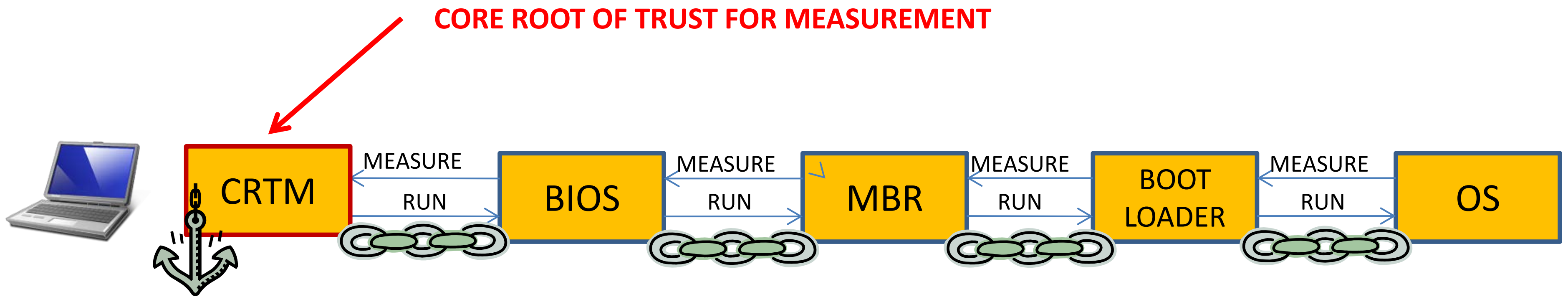
Where does the trust come from?

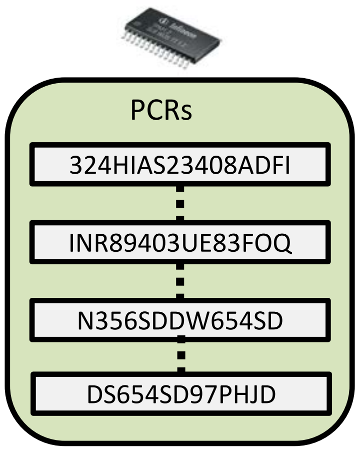


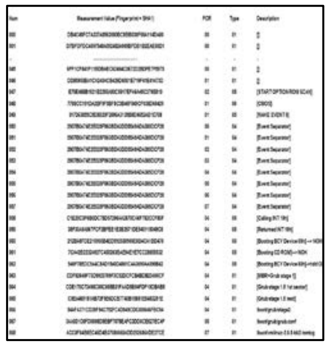
Num	Measurement Value (Fingerprint = SHA1)	PCR	Type	Description
000	[illegible]	00	01	[illegible]
001	[illegible]	00	01	[illegible]
-	-	-	-	-
045	[illegible]	00	01	[illegible]
046	[illegible]	01	01	[illegible]
047	[illegible]	02	05	[START OPTION ROM SCAN]
048	[illegible]	01	05	[CMOS]
049	[illegible]	01	05	[NVRAM EVENT 8]
050	[illegible]	00	04	[Event Separator]
051	[illegible]	01	04	[Event Separator]
052	[illegible]	02	04	[Event Separator]
053	[illegible]	03	04	[Event Separator]
054	[illegible]	04	04	[Event Separator]
055	[illegible]	05	04	[Event Separator]
056	[illegible]	06	04	[Event Separator]
057	[illegible]	07	04	[Event Separator]
058	[illegible]	04	05	[Calling INT 19h]
059	[illegible]	04	05	[Returned INT 19h]
060	[illegible]	04	05	[Booting BCV Device 80h] --> NOH
061	[illegible]	04	05	[Booting CD ROM] --> NOH
062	[illegible]	04	05	[Booting BCV Device 80h] --> [illegible]
063	[illegible]	04	01	[MBR+Grub stage 1]
064	[illegible]	04	01	[Grub stage 1.5 1st sector]
065	[illegible]	04	01	[Grub stage 1.5 rest]
066	[illegible]	04	01	/boot/grub/stage2
067	[illegible]	05	01	/boot/grub/grub.conf
068	[illegible]	07	01	/boot/vmlinuz-2.6.x [illegible]

- Always executed (first)
- Non-bypassable
- Correctly implemented
- Immutable

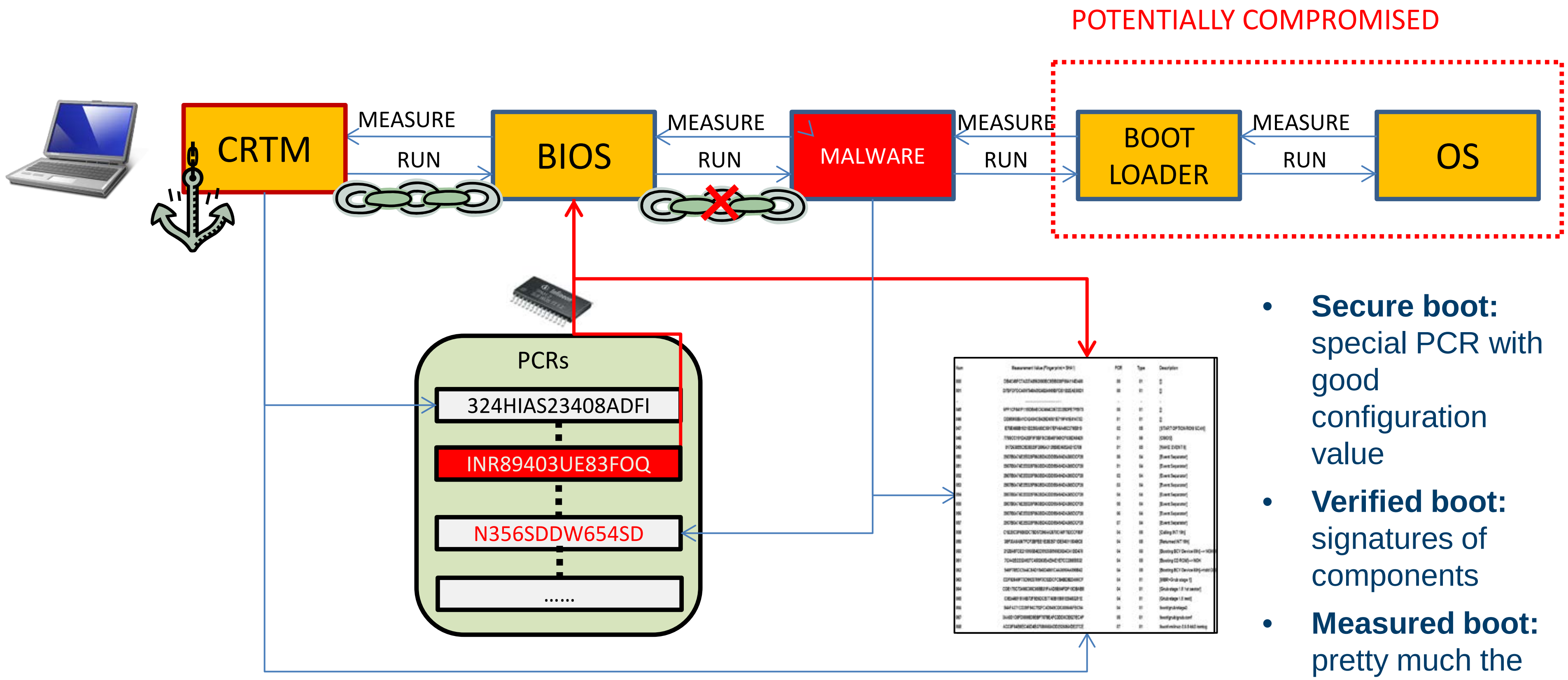
Root of Trust for Measurement







Root of Trust for Measurement

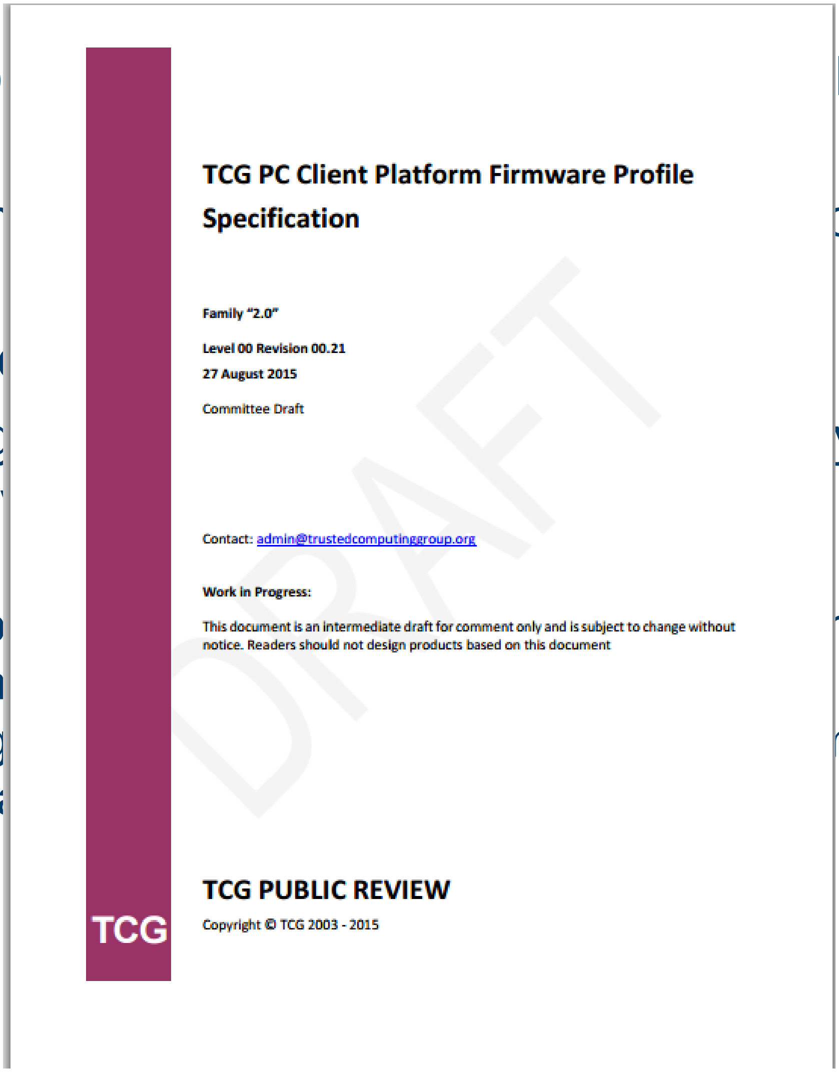


- **Secure boot:** special PCR with good configuration value
- **Verified boot:** signatures of components
- **Measured boot:** pretty much the same

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

S-CRTM Problems

- It should be p lt to do in practice because of th
- It is implemer of the BIOS.
- There is no cl it and every OEM does it on its
- Not surprising y implementation issues that re uncorrectly or not at all:
  - J. Butterwo rzog, "BIOS chronoman urement," Proceeding nputer and Communica


TCG PC Client Platform Firmware Profile Specification

Family "2.0"

Level 00 Revision 00.21

27 August 2015

Committee Draft

Contact: admin@trustedcomputinggroup.org

Work in Progress:

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document

TCG PUBLIC REVIEW

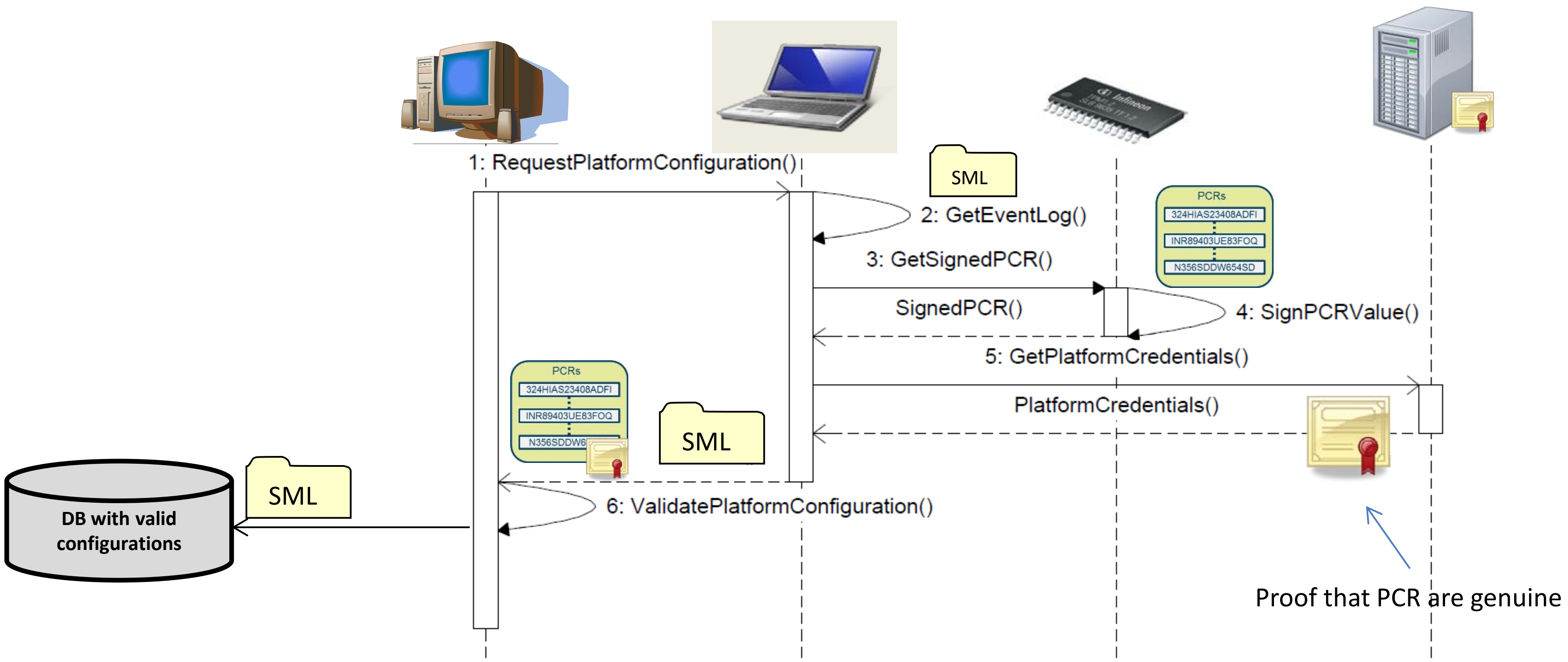
Copyright © TCG 2003 - 2015

Upgrades

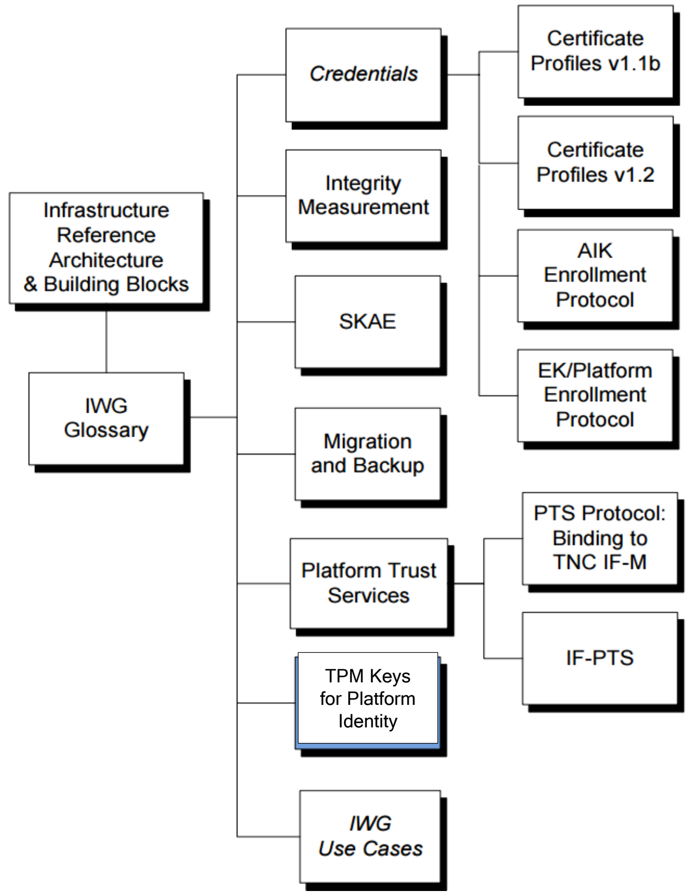
- What if I update my BIOS or some drivers?
- What I have have to upgrade or replace some hardware?
- How many possible «good» configurations can I have in a DB?
- Not practical

- TPM 2.0 tries to address this problem allowing more flexible policies about the PCR values

Attestation protocol: Root of trust for Reporting


1: RequestPlatformConfiguration()
SML
2: GetEventLog()
PCRs
324HIAS23408ADFI
INR89403UE83FOQ
N356SDDW654SD
3: GetSignedPCR()
SignedPCR()
4: SignPCRValue()
5: GetPlatformCredentials()
PlatformCredentials()
SML
DB with valid configurations
6: ValidatePlatformConfiguration()
Proof that PCR are genuine

PKI infrastructure

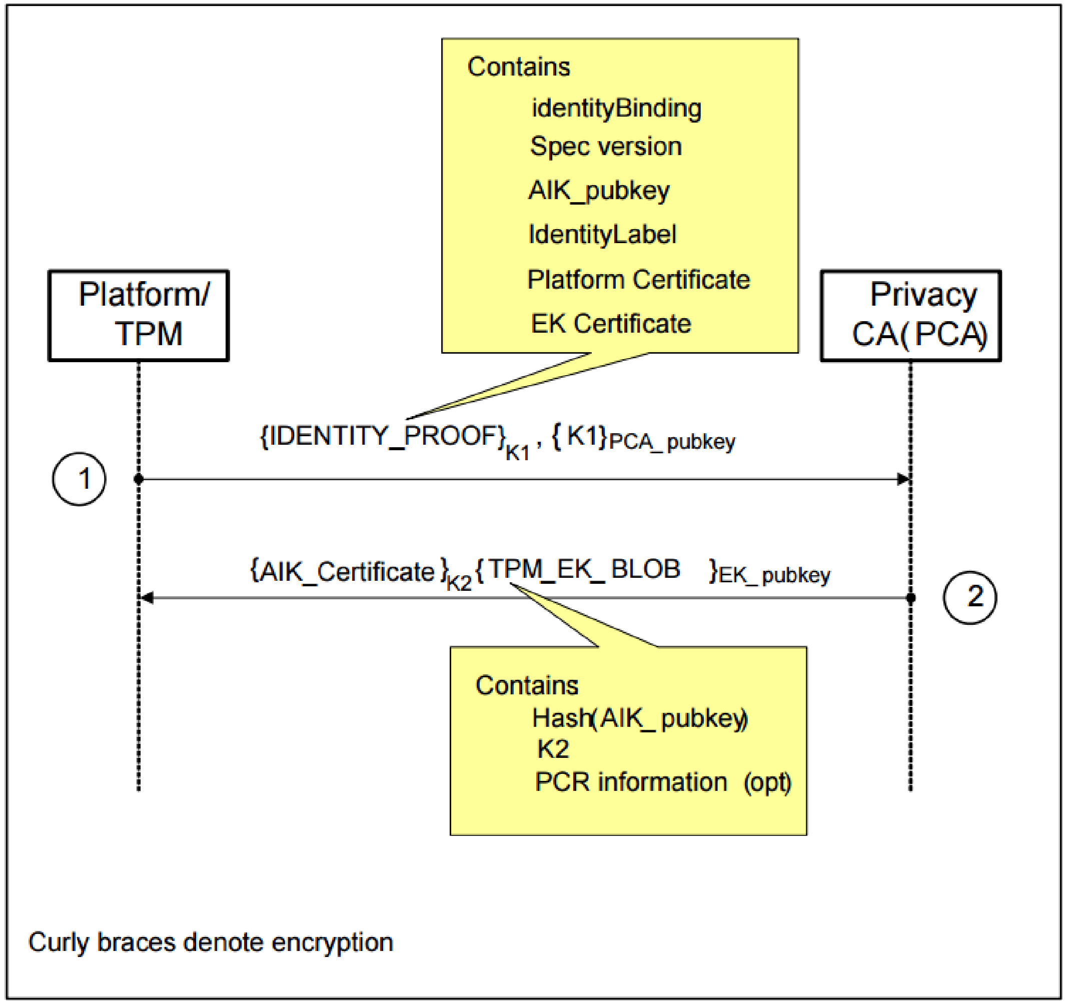

Infrastructure Reference Architecture & Building Blocks
IWG Glossary
Credentials
Integrity Measurement
SKAE
Migration and Backup
Platform Trust Services
TPM Keys for Platform Identity
IWG Use Cases
Certificate Profiles v1.1b
Certificate Profiles v1.2
AIK Enrollment Protocol
EK/Platform Enrollment Protocol
PTS Protocol: Binding to TNC IF-M
IF-PTS

- No CA really supports any of these specifications
- Only Infineon ships TPM with actual pre-sintalled certificates signed by Verisign
- A TPM cannot create standard X.509v3 certificates
- It cannot even sign certificate requests
- A Privacy CA should take care of it, but none is implemented and deployed publically
- TPM 2.0 try to do something about this

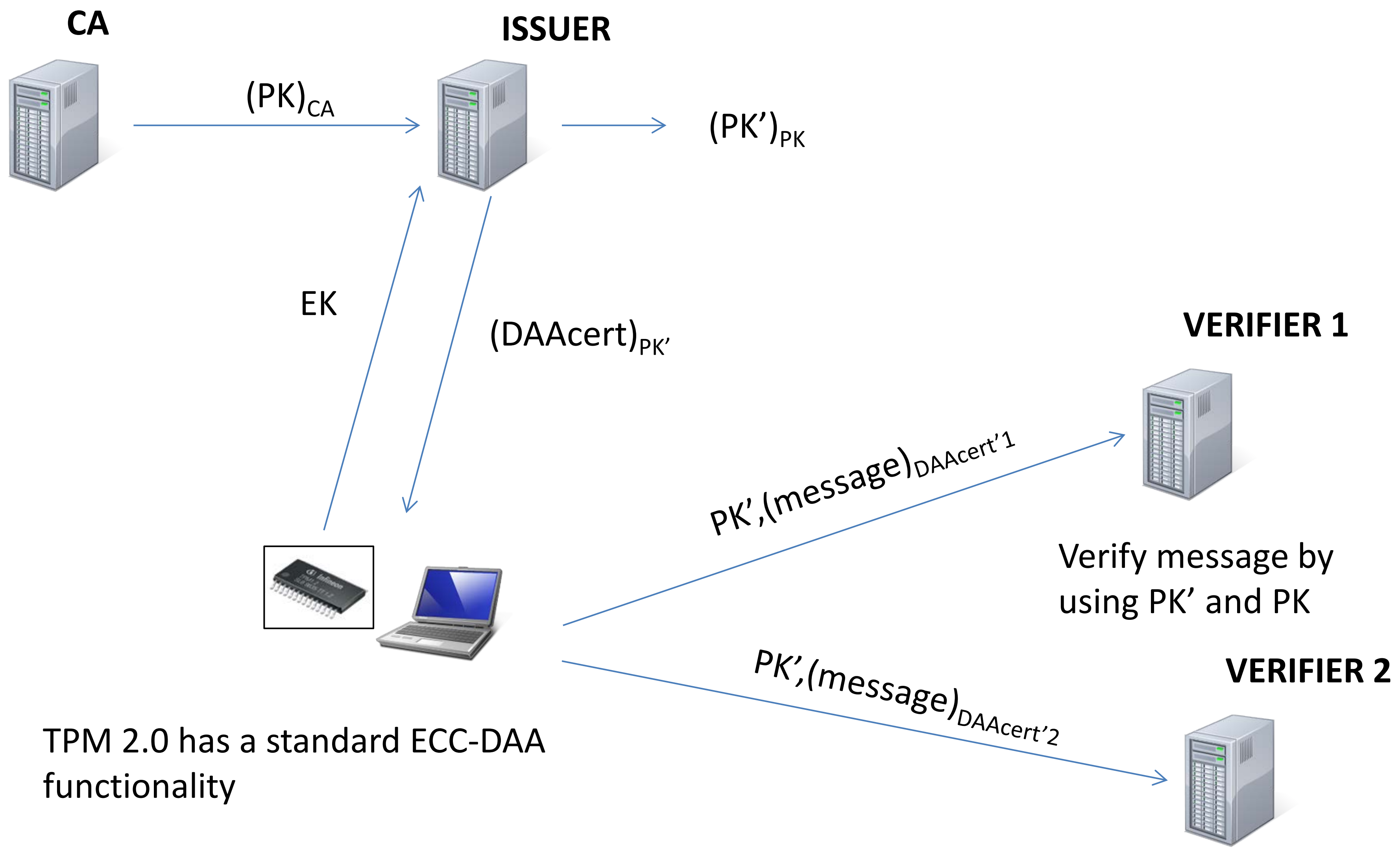
Anonimity

- Each TPM has a unique RSA key pair called Endorsment Key (EK) and a certificate certifying that the EK belongs to a genuine TPM
- This to allow third parties to verify that they are talking to a real TPM and that the compromise of one TPM does not affect all the other (otherwise one global TPM key could have been used)
- However, this also means that each TPM is identifiable.
- That is why the EK cannot be used for signing, but only encrypting and decrypting, and a TPM can create as many Attestation Identity Keys (AIK) as it wishes to be used to sign TPM generated content instead.
- But how to certify that these AIKs also are generated and protected in a genuine TPM?

Privacy CA (PCA)

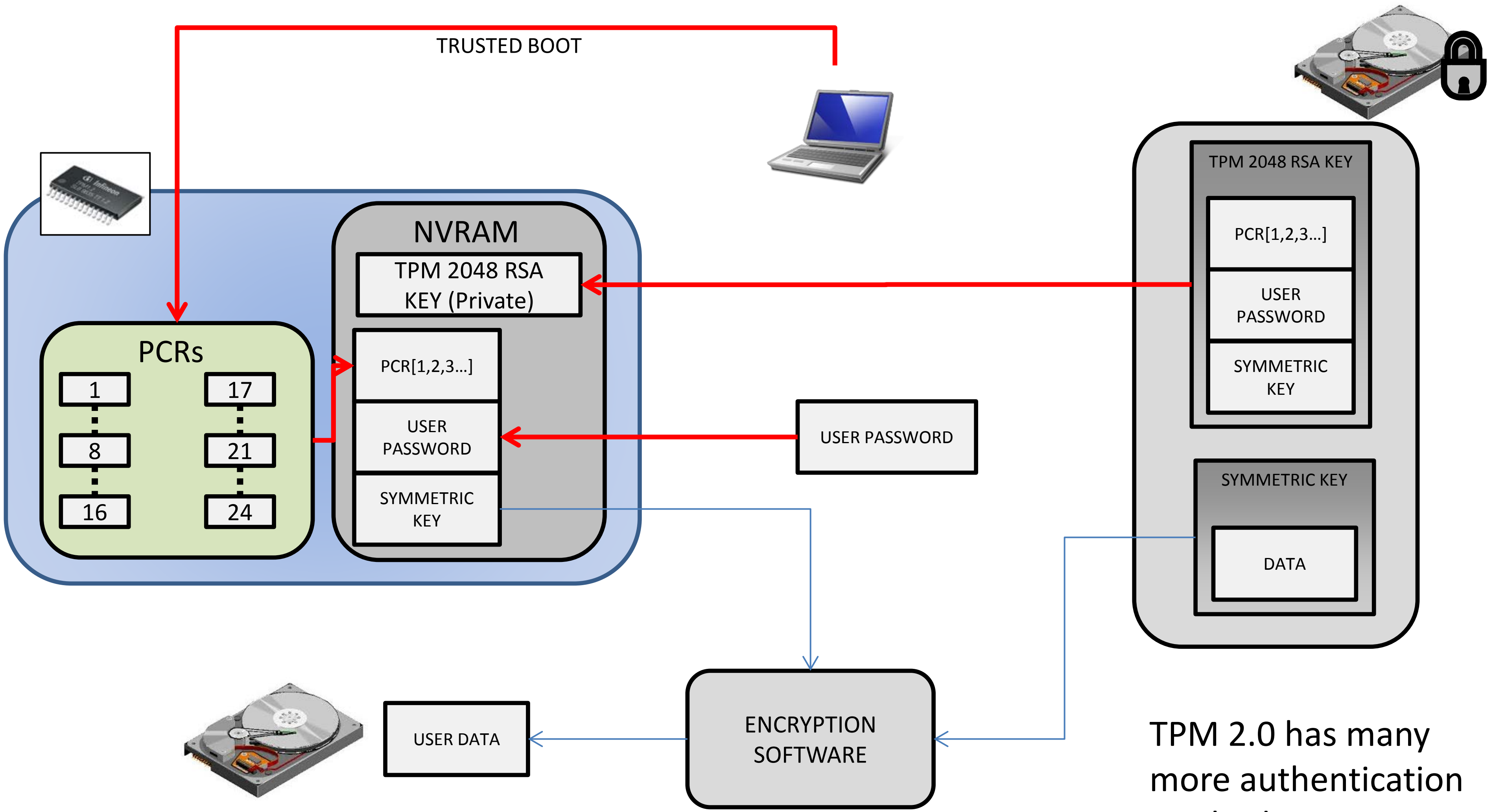


Direct Anonymous Attestation


CA
ISSUER
$(PK)_{CA}$
$(PK')_{PK}$
EK
$(DAAcert)_{PK'}$
VERIFIER 1
$PK',(message)_{DAAcert'1}$
Verify message by using PK' and PK
VERIFIER 2
$PK',(message)_{DAAcert'2}$

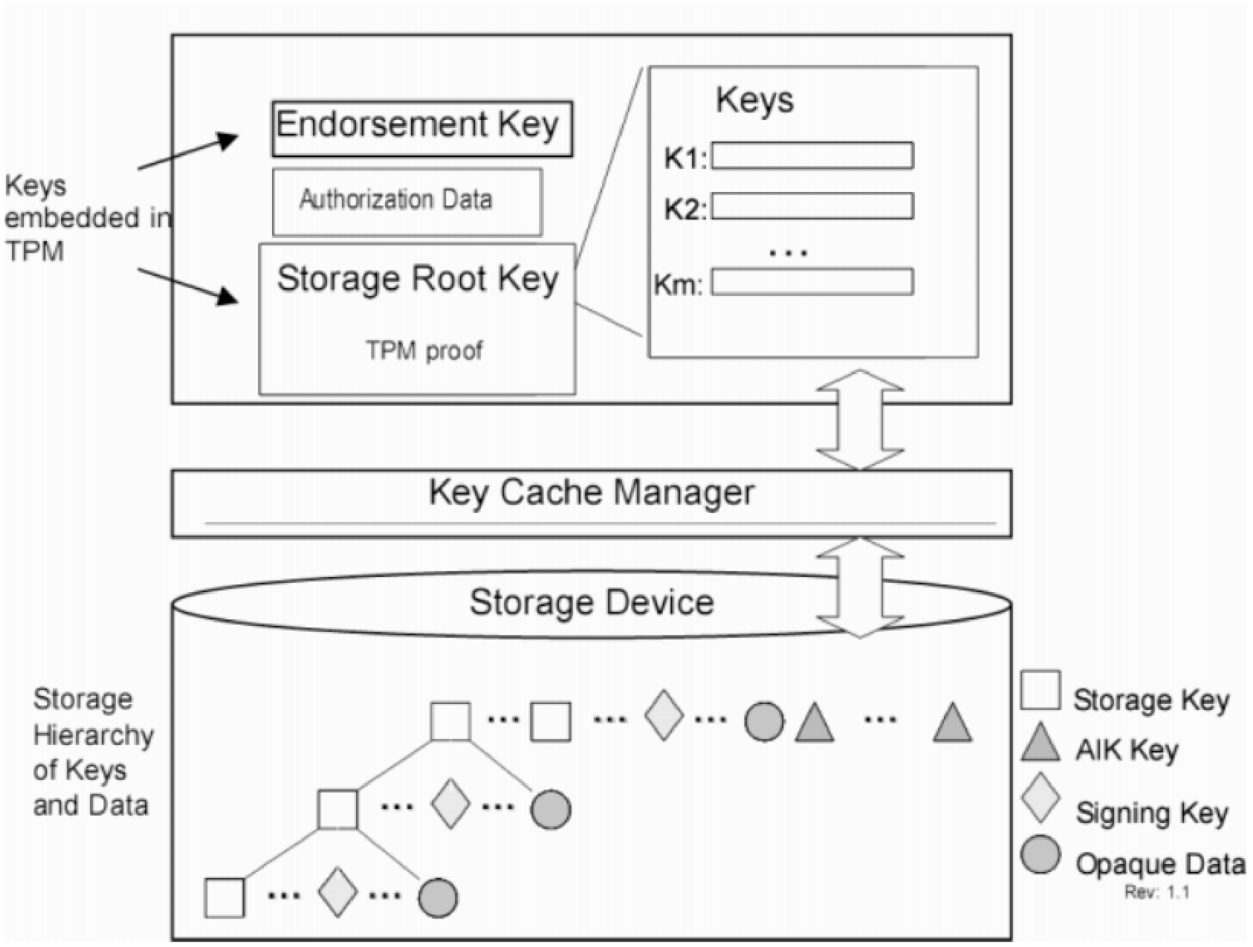
TPM 2.0 has a standard ECC-DAA functionality

Sealing/Binding



TPM 2.0 has many more authentication methods

TPM 1.2 Key Hierarchy



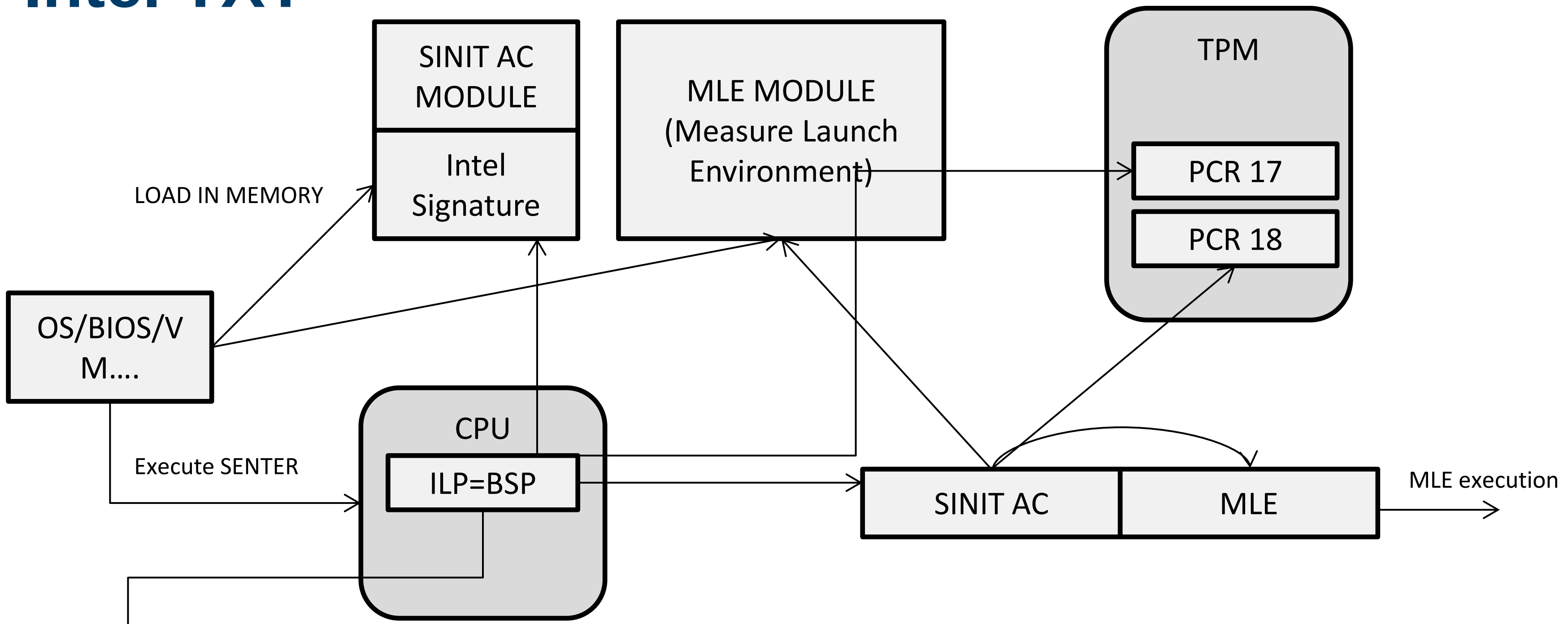
Problems

- 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
- Users do not want someone else to have control of their machines
- Many restrictions on the key usage

After the OS

- **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

Intel SGX (Secure Guard Extensions) is coming next to add flexibility. Reminds of Flicker.

TPM 2.0

With content copied and pasted shamelessly from:

- David Challener, «TPM 2.0 – Re-envisioning the TPM», Trusted Infrastructure Workshop, 2013 The Pennsylvania State University, University Park, PA.
- David Wooten, "TPM 2.0", Partner Architect Microsoft Corp., 25 October 2013, Trusted Computing Group.
- Liqun Chen, «From TPM 1.2 to TPM 2.0», ETISS 2013, Gratz, Austria.
- Will Arthur and David Challener, «A Practical Guide to TPM 2.0», Apress Open, 2014.
- Graeme Proudler, Liqun Chen and Chris Dalton, «Trusted Computing Platforms – TPM 2.0 in context», Springer 2014.
- Various TCG specifications

TPM 2.0

- **Support for new algorithms:** flexibility for the inclusion of a variety of algorithms, Elliptic curve-based algorithms and SHA-2 . and potentially multiple “algorithm sets” on a single TPM.
- **Support for more than one “bank” of PCRs:** enables the TPM to keep track of platform state using more than one distinct hash algorithm
- **Inclusion of three ownership hierarchies:** a “platform hierarchy” for platform protection, an “endorsement hierarchy” for privacy control and a “storage hierarchy” for general cryptographic usage
- **Support for enhanced authorization:** support for very flexible and fine-grained control over how and when TPM-protected data and keys can be accessed
- **More complex policies:** policies can be combined with boolean operator and support more scenarios
- **Flexible keys:** only two types of keys

Algorithms

TPM 1.2 supports

RSA encryption

RSA signature

RSA-DAA

SHA-1

HMAC

One-time-pad with XOR

AES (optional)

TPM 2.0 supports

RSA encryption and signature

ECC encryption and signature

ECC-DAA

ECDH

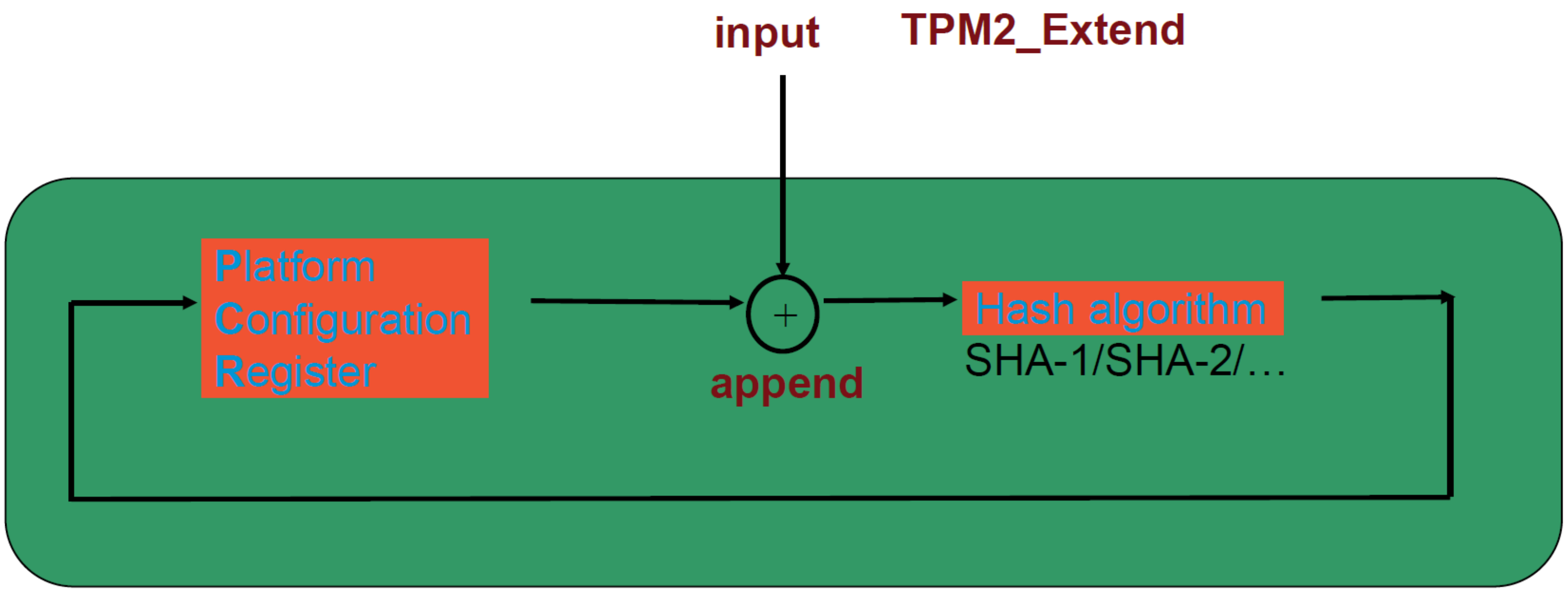
SHA-1, SHA-256

HMAC

AES and one-time-pad with XOR

Manufacturer can add any algorithms with TCG IDs

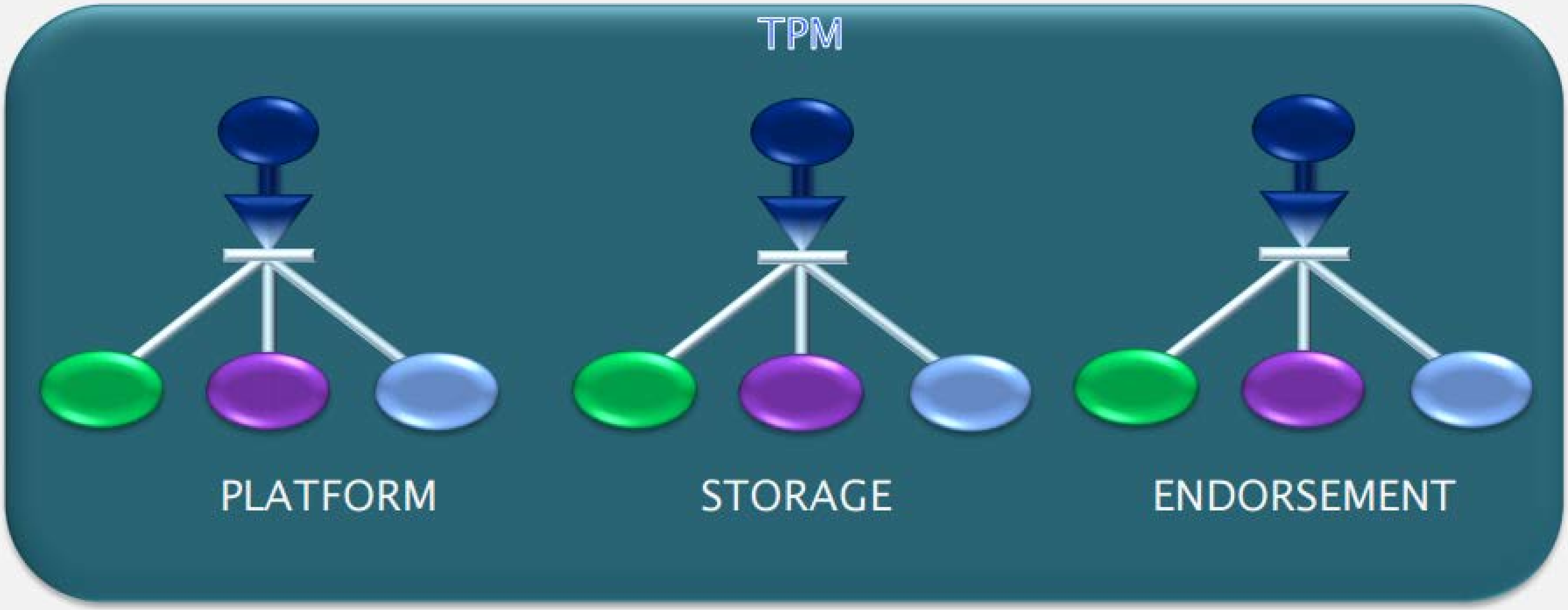
More PCRs


input
TPM2_Extend
Platform Configuration Register
+
append
Hash algorithm
SHA-1/SHA-2/…

TPM 2.0 - Hierarchies

- In TPM 1.2, everything is under the control of the “Owner”
  - If the TPM is not enabled, activated, and owned; there isn’t much that can be done with it
  - If you are the Owner, you control both the security and privacy functions
- In TPM 2.0, there are three separate domains
  - Security – functions that protect the security of the user
  - Privacy – functions that expose the identity of the platform/user
  - Platform – functions that protect the integrity of the platform/firmware services
- Each domain has its own resources and controls
  - Security – *ownerAuth*, storage hierarchy, hierarchy enable
  - Privacy – *endorsementAuth*, endorsement hierarchy
  - Platform – *platformAut*h, platform hierarchy

TPM 2.0 - Hierarchies



To support these three hierarchies, the only persistent storage required in the TPM are the three, Primary Seeds – 256-/512-bits each

TPM 2.0 – Platform Hierarchy

- **Platform hierarchy**
  - For platform firmware BIOS/UEFI
  - When the platform boots, the platform hierarchy is enabled and platformAuth is set to a new value
    - Allows use of the TPM to ensure the integrity of the firmware
    - This is not a capability that should be under control of the user, so it isn’t
  - PlatformAuth can be used to:
    - Allocate NonVolatile memory resources
    - Initialize the TPM
    - Control the enables of the other hierarchies
  - Before platform firmware turns control of system to OS, phEnable can be turned off or platformAuth can be randomized
    - PlatformAuth would be placed in secure location (SMM) so that only platform firmware would be able to access it

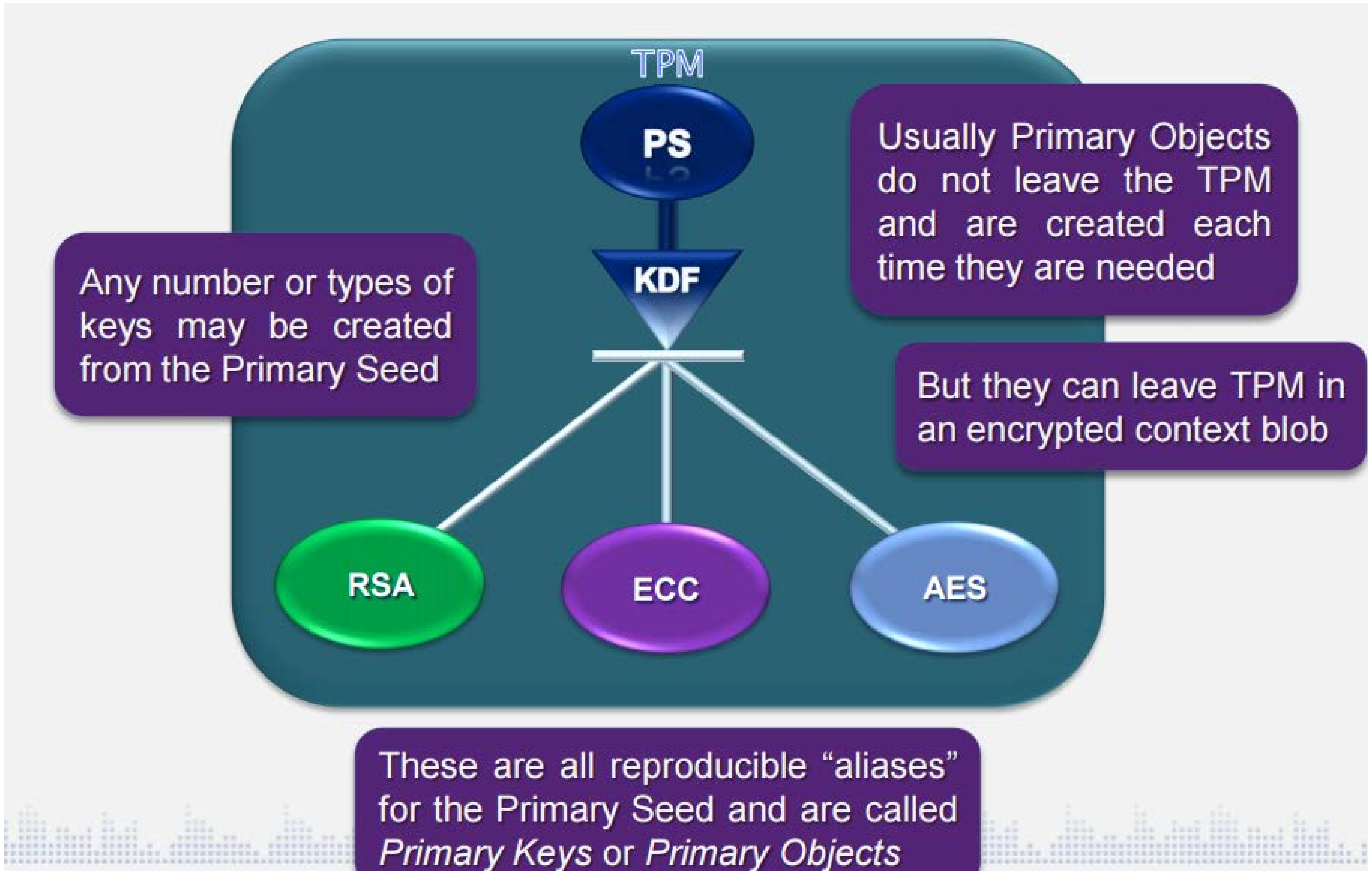
TPM 2.0 – Endorsement Hierarchy

- **Endorsement hierarchy**
  - For privacy administrator

- **Endorsment keys**
  - As many as one wish
  - Created from a secret seed
  - Can be used to sign
  - Can be used with different algorithms
  - Belongs to its own key hierarchy

  - Examples:
    - One can create a signing EK to sign a CSR and get a Device ID directly from the certificate autorithy that has a list of valid EK
    - If EK comes with EK credentials, it should not be allowed to sign to preserve privacy

TPM 2.0 - One seed to rule them all



Any number or types of keys may be created from the Primary Seed

Usually Primary Objects do not leave the TPM and are created each time they are needed

But they can leave TPM in an encrypted context blob

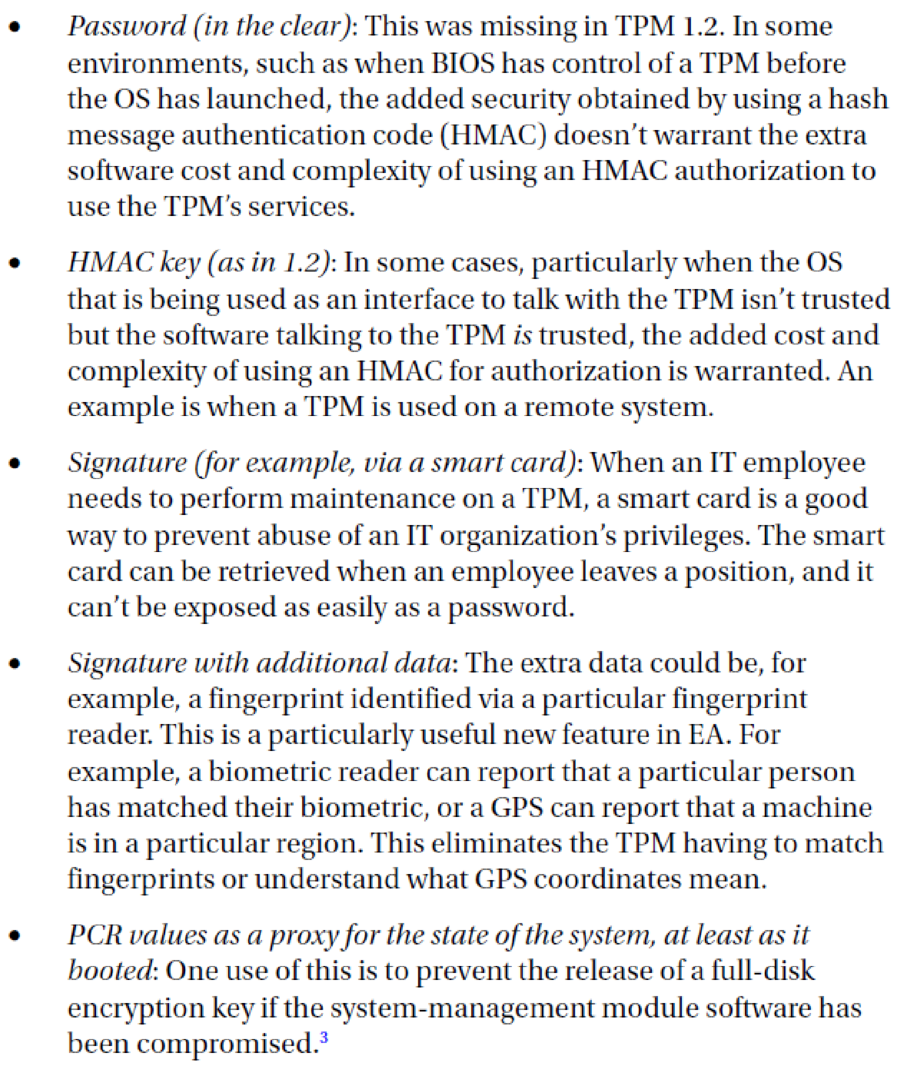
These are all reproducible “aliases” for the Primary Seed and are called *Primary Keys* or *Primary Objects*

TPM 2.0 – Keys

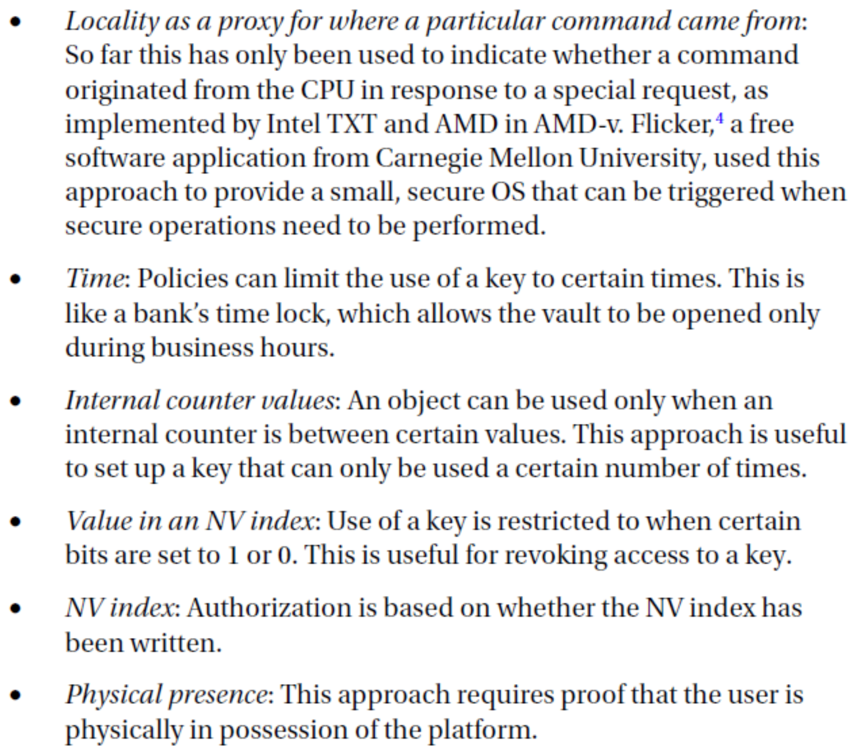
- No more many types of keys, only *restricted* and *unrestricted*, with possibility to choose what they do. Even EK.
- Restricted simply means that they will not sign any external data that resembles a TPM data structure.

Attribute			
restricted	*sign*	*decrypt*	Nominal Usage
0	0	0	External data that is protected (via bind or seal) by the hierarchy
0	0	1	A key for protecting data
0	1	0	A key for signing data
0	1	1	A key for protecting and signing external data
1	0	1	A storage key, for constructing the hierarchy
1	1	0	A key for signing TPM data (certificates, quotes) (an “AK”)
1	0	0	Forbidden combination (don’t know what it means)
1	1	1	Forbidden combination (it has no useful purpose and is incompatible with the USA’s FIPS specifications)

TPM 2.0 – Enhanced Authorization (EA)



- *Password (in the clear)*: This was missing in TPM 1.2. In some environments, such as when BIOS has control of a TPM before the OS has launched, the added security obtained by using a hash message authentication code (HMAC) doesn't warrant the extra software cost and complexity of using an HMAC authorization to use the TPM's services.
- *HMAC key (as in 1.2)*: In some cases, particularly when the OS that is being used as an interface to talk with the TPM isn't trusted but the software talking to the TPM *is* trusted, the added cost and complexity of using an HMAC for authorization is warranted. An example is when a TPM is used on a remote system.
- *Signature (for example, via a smart card)*: When an IT employee needs to perform maintenance on a TPM, a smart card is a good way to prevent abuse of an IT organization's privileges. The smart card can be retrieved when an employee leaves a position, and it can't be exposed as easily as a password.
- *Signature with additional data*: The extra data could be, for example, a fingerprint identified via a particular fingerprint reader. This is a particularly useful new feature in EA. For example, a biometric reader can report that a particular person has matched their biometric, or a GPS can report that a machine is in a particular region. This eliminates the TPM having to match fingerprints or understand what GPS coordinates mean.
- *PCR values as a proxy for the state of the system, at least as it booted*: One use of this is to prevent the release of a full-disk encryption key if the system-management module software has been compromised.[3]



- *Locality as a proxy for where a particular command came from*: So far this has only been used to indicate whether a command originated from the CPU in response to a special request, as implemented by Intel TXT and AMD in AMD-v. Flicker,[4] a free software application from Carnegie Mellon University, used this approach to provide a small, secure OS that can be triggered when secure operations need to be performed.
- *Time*: Policies can limit the use of a key to certain times. This is like a bank's time lock, which allows the vault to be opened only during business hours.
- *Internal counter values*: An object can be used only when an internal counter is between certain values. This approach is useful to set up a key that can only be used a certain number of times.
- *Value in an NV index*: Use of a key is restricted to when certain bits are set to 1 or 0. This is useful for revoking access to a key.
- *NV index*: Authorization is based on whether the NV index has been written.
- *Physical presence*: This approach requires proof that the user is physically in possession of the platform.

All these forms for authorizations can also be combined to create complex access policies.

TPM 2.0 – Locking to a PCR

- You can lock *not* just to a certain set of PCRs equals a certain value
- You can also lock to: “Any set of PCRs / values signed by an authority, as represented by this public key”

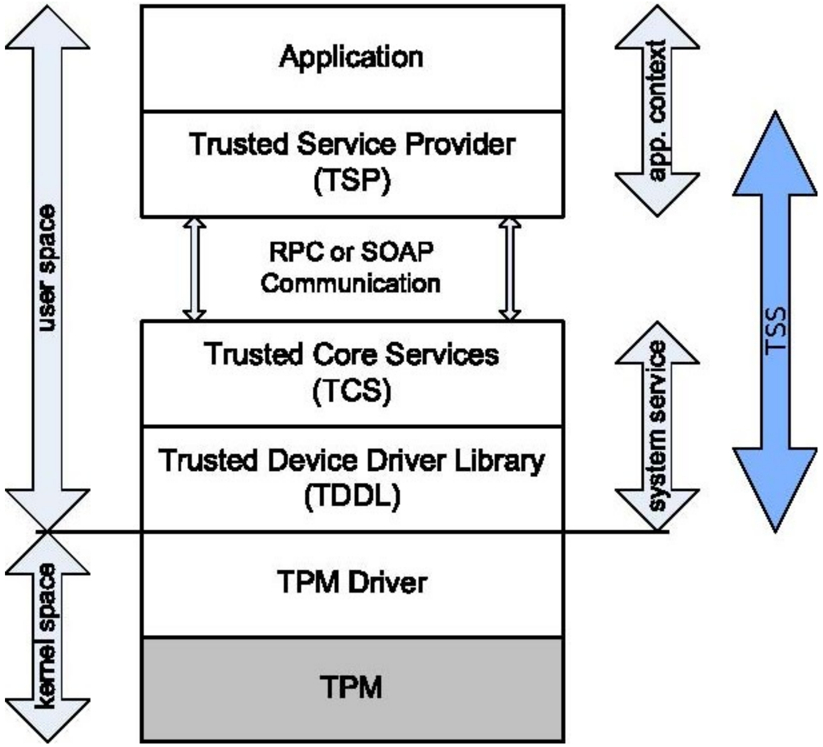
Examples:
- You can lock to “PCR 0 (the BIOS) as signed by DELL”
    - Thereafter upgrading your BIOS to a signed DELL BIOS won’t cause problems!
- You can lock to “PCR values signed by IT”
    - Thereafter IT need only sign new values to make them useable

How to do that

TPM2_PolicyAuthorize() allow a policy to have an “authority” determine if some policy is OK rather than have the policy hardwired in

- **Example:** The *policyHash* representing a set of good PCR is known to the OEM
- The OEM signs a digest that represents the *policyHash* representing the good PCR and distributes it along with their BIOS update
- The user can create a policy that says, “if the OEM approves the PCR settings they are OK with me”
- Use TPM2_PolicyPCR() to set the policyHash to the current value of the PCR and then use TPM2_PolicyAuthorize() to apply the OEM’s stamp of approval to those PCR

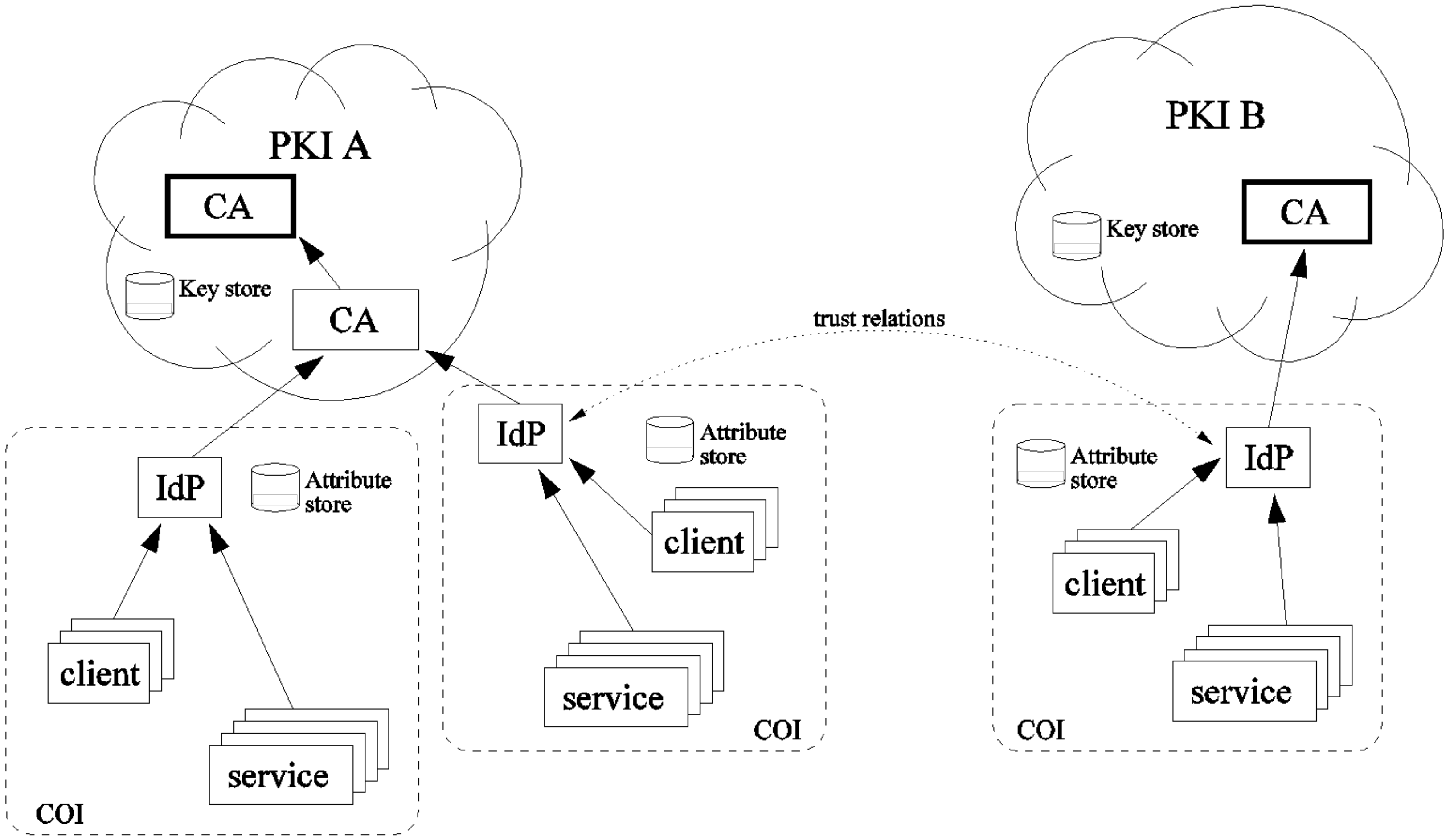
Other TPM related things - Trusted Software Stack (TSS)


Application
Trusted Service Provider (TSP)
RPC or SOAP Communication
Trusted Core Services (TCS)
Trusted Device Driver Library (TDDL)
TPM Driver
TPM
user space
kernel space
app. context
system service
TSS

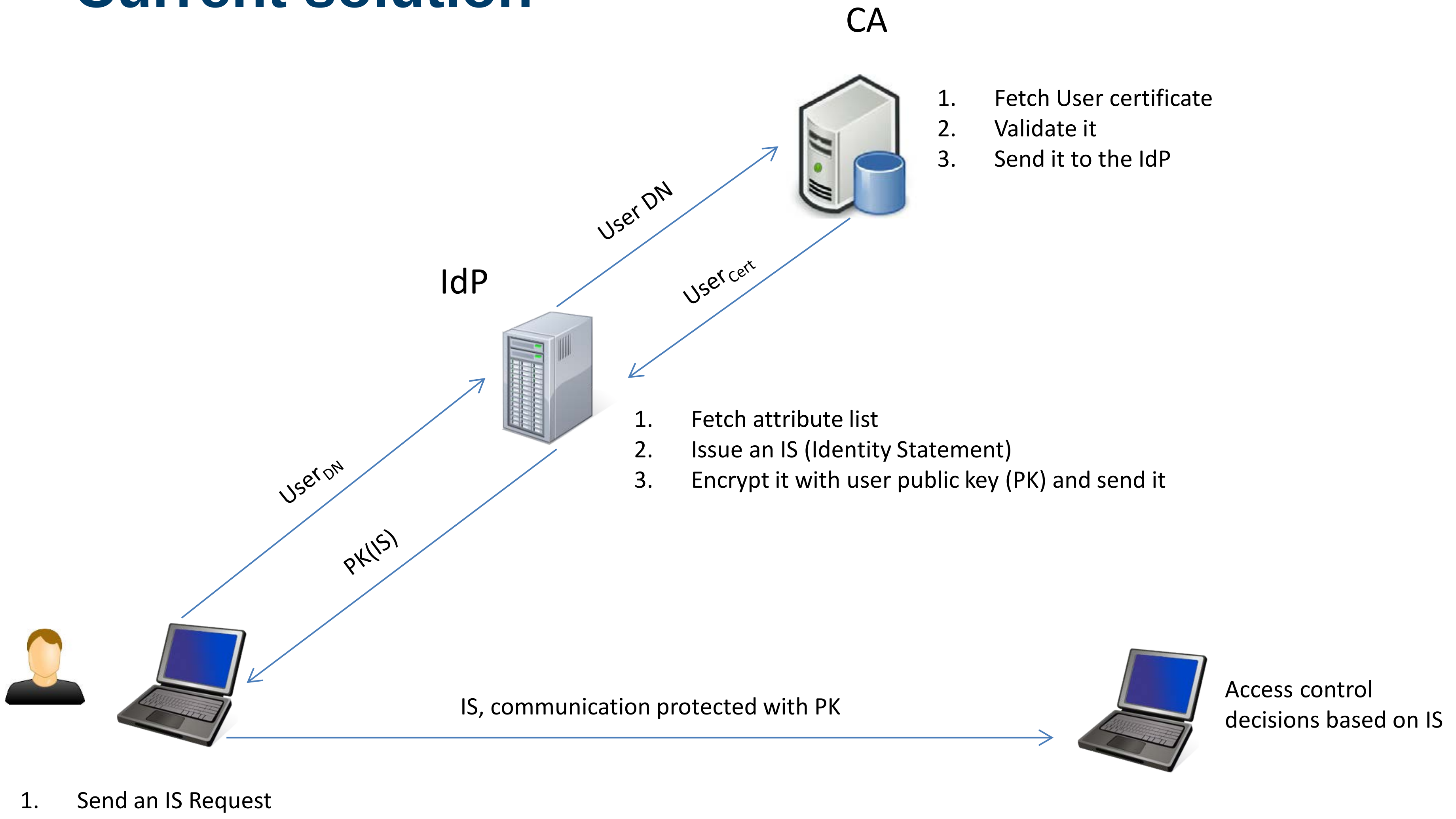
From JSR 321 wiki

- Teh software stack is needed by applications to communicate with theTPM.
- Trousers library written in C for Linux offers TSS 2.0 support, TSS.net from Microsoft also supports 2.0 specs, while the JSR321 for Java only supports TSS 1.0
- Drivers for TPM are now integrated in all modern OSes.

A Trusted IdM for Tactical Operations



Current solution



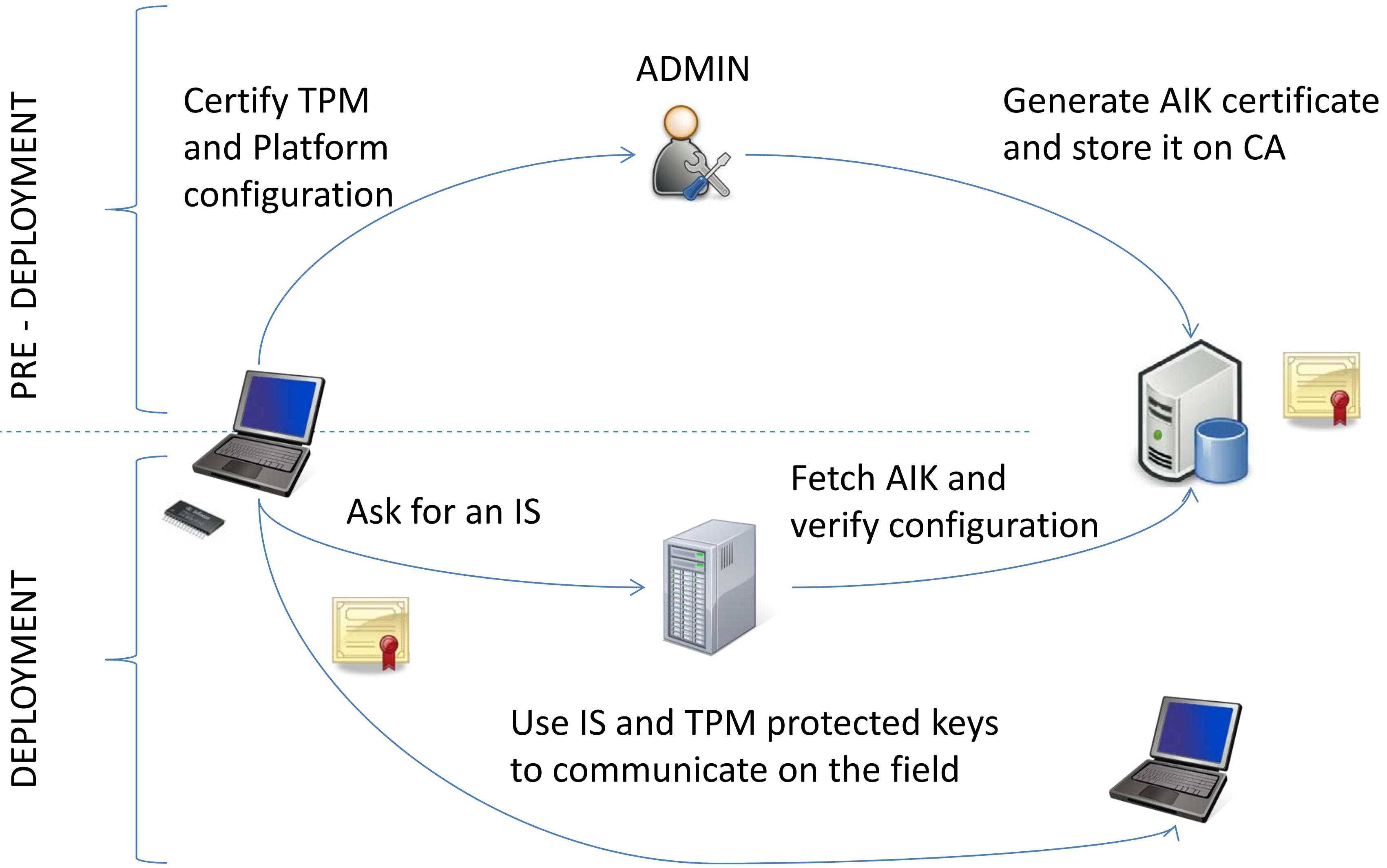
1. Fetch User certificate
2. Validate it
3. Send it to the IdP

1. Fetch attribute list
2. Issue an IS (Identity Statement)
3. Encrypt it with user public key (PK) and send it

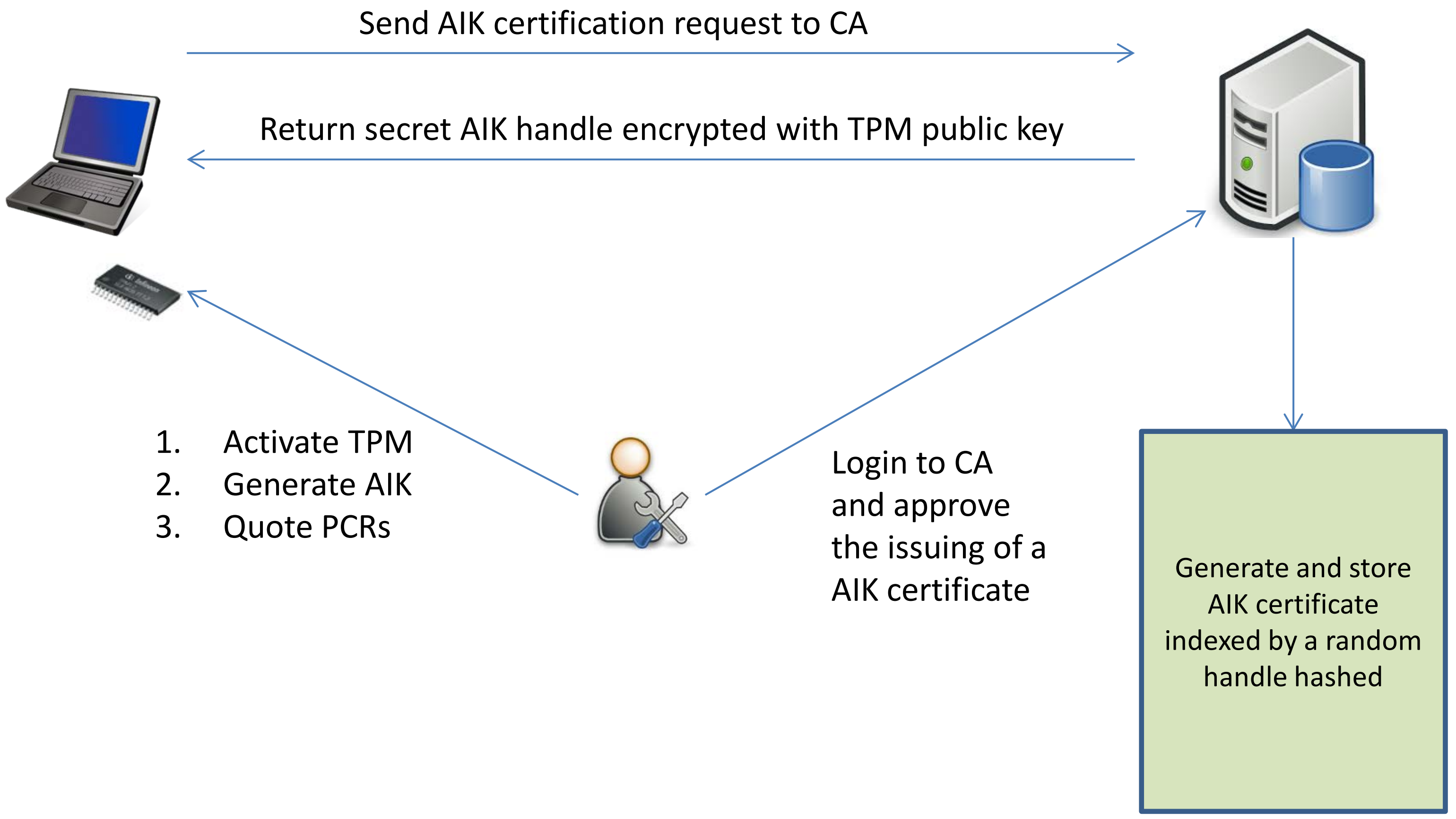
Access control decisions based on IS

1. Send an IS Request

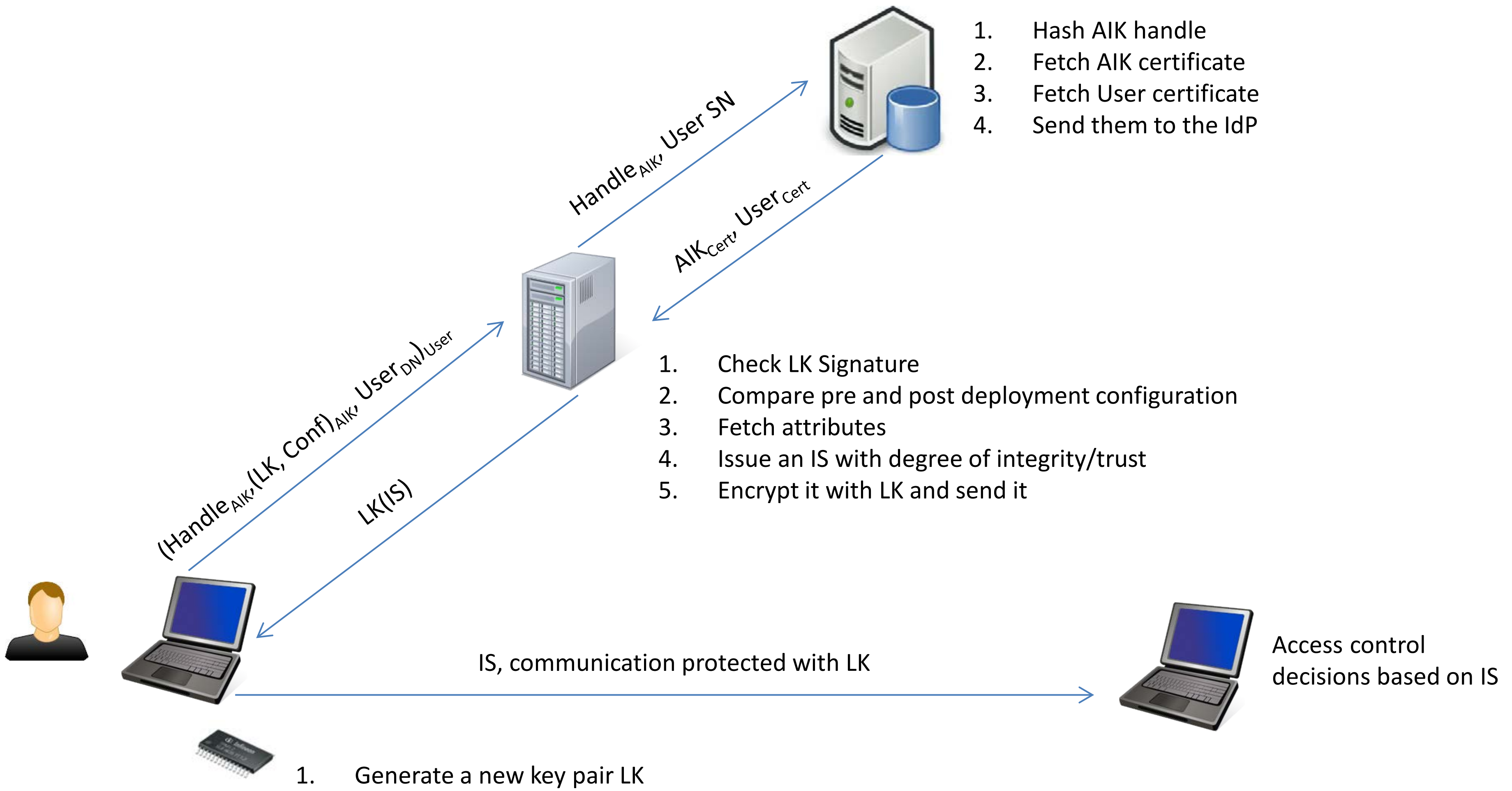
New Solution with TPM support


PRE - DEPLOYMENT
DEPLOYMENT
ADMIN
Certify TPM and Platform configuration
Generate AIK certificate and store it on CA
Ask for an IS
Fetch AIK and verify configuration
Use IS and TPM protected keys to communicate on the field

Pre-deployment scenario


Send AIK certification request to CA
Return secret AIK handle encrypted with TPM public key
1. Activate TPM
2. Generate AIK
3. Quote PCRs
Login to CA and approve the issuing of a AIK certificate
Generate and store AIK certificate indexed by a random handle hashed

Deployment scenario



Handle$_{AIK}$, User SN
AIK$_{Cert}$, User$_{Cert}$
(Handle$_{AIK}$,(LK, Conf)$_{AIK}$, User$_{DN}$)User
LK(IS)
IS, communication protected with LK

1. Hash AIK handle
2. Fetch AIK certificate
3. Fetch User certificate
4. Send them to the IdP

1. Check LK Signature
2. Compare pre and post deployment configuration
3. Fetch attributes
4. Issue an IS with degree of integrity/trust
5. Encrypt it with LK and send it

Access control decisions based on IS

1. Generate a new key pair LK
2. Bind it to the current configuration
3. Sign it with the AIK and the user certificate

Some experience with implementing with TPM

- Few available TSS, not implementing all needed functionalities
- No PCA implementation available, only some limited prototype
- Only Infineon provides EK certificates signed by Verisign
- We had to change the TSS code to unpack the TPM certificate request and change it to Java objects
- A special x.509v3 extension called SKAE (Subject Key Attestation Evidence) is to be used to integrated TPM info in a certificate. OpenSSL does not support it.
- AIK cannot sign CSR (Certificate Signing Requests)
- Heavy to send your AIK certificate in all transactions
- We do not have a DB with approved configurations, but a pre-deployment phase where we issue an AIK bound to specific PCR values and check that the same values are still there when the user want to certificate a new legacy key
- We re-parsed all TPM data-structure to use standard Java objects instead, since TPM-enables and non-TPM nodes had to be interoperable.

Products using TPM

- Windows 8/10 verified/authenticated boot
- Bitlocker
- Google Chromebook
- Various vendors implementing certificate/key storage on TPM
- Integrity Measurement Architecture in Linux kernel
- Some Google routers
- Strong Swan
- …..