Trusted Platform Module (TPM)
introduction

Mark D. Ryan
University of Birmingham

Computer Security module
October 2009

The ***Trusted Computing Group***

- An industry consortium including
  - Microsoft, HP, Dell, Sony, Lenovo, Toshiba, Vodafone, Seagate, . . .
  - (about 160 organisations in total)
- Main output is *Trusted Platform Module* spec
  - The specification is ***publicly available***
  - The TPM is a ***passive device*** (it does not *monitor* or *prohibit* anything; just performs actions if asked)
  - It is mandated to be ***opt-in***, not opt-out
  - It includes ***privacy-enabling*** functionality

The Trusted Platform Module

- A hardware chip currently included in 100M laptops
  - HP, Dell, Sony, Lenovo, Toshiba . . .
  - Soldered onto the motherboard, on the LPC bus
  - HP alone ships 1M TPM-enabled laptops each month
- Specified by the *Trusted Computing Group*
  - An industry consortium that includes Intel, HP, Microsoft, AMD, IBM, Sun, Lenovo. . . . and 130 other members
- Manufactured by many companies
  - Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond
- Supporting software to be rolled out over the next few years
  - MS BitLocker is the only mainstream application so far

TPM functionality

Secure storage

- Creation of RSA keys (with private part known only to the TPM)
- Encryption and decryption of user data with those keys

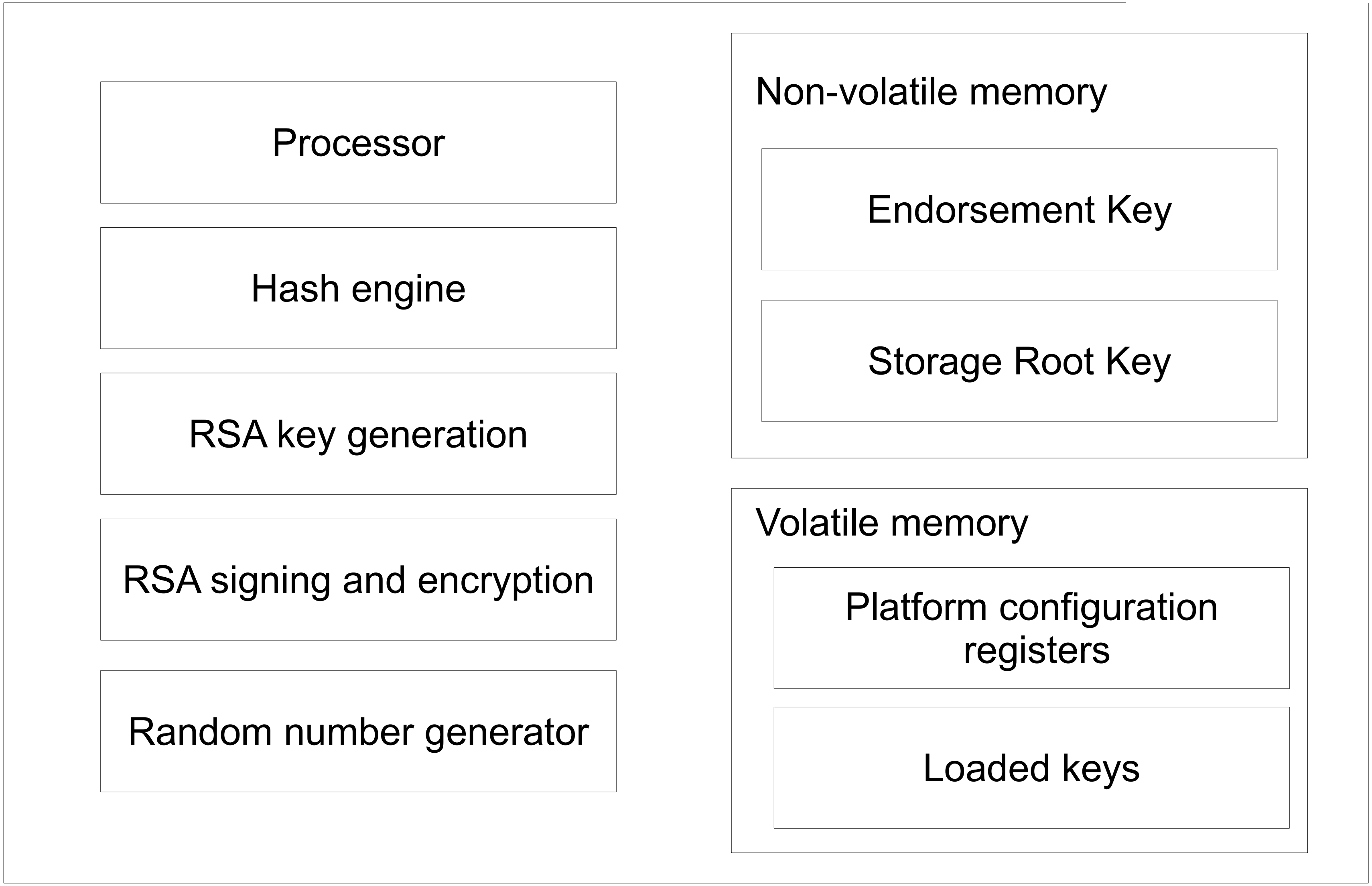
Platform integrity reporting

- "Measurement" and reporting of integrity of platform; may include measurement of BIOS, disk MBR, boot sector, operating system and application software

Platform authentication

- Creation of *attestation identity keys (AIK)*, with anonymity guarantees (DAA)

TPM architecture



Processor

Hash engine

RSA key generation

RSA signing and encryption

Random number generator

Non-volatile memory

Endorsement Key

Storage Root Key

Volatile memory

Platform configuration registers

Loaded keys

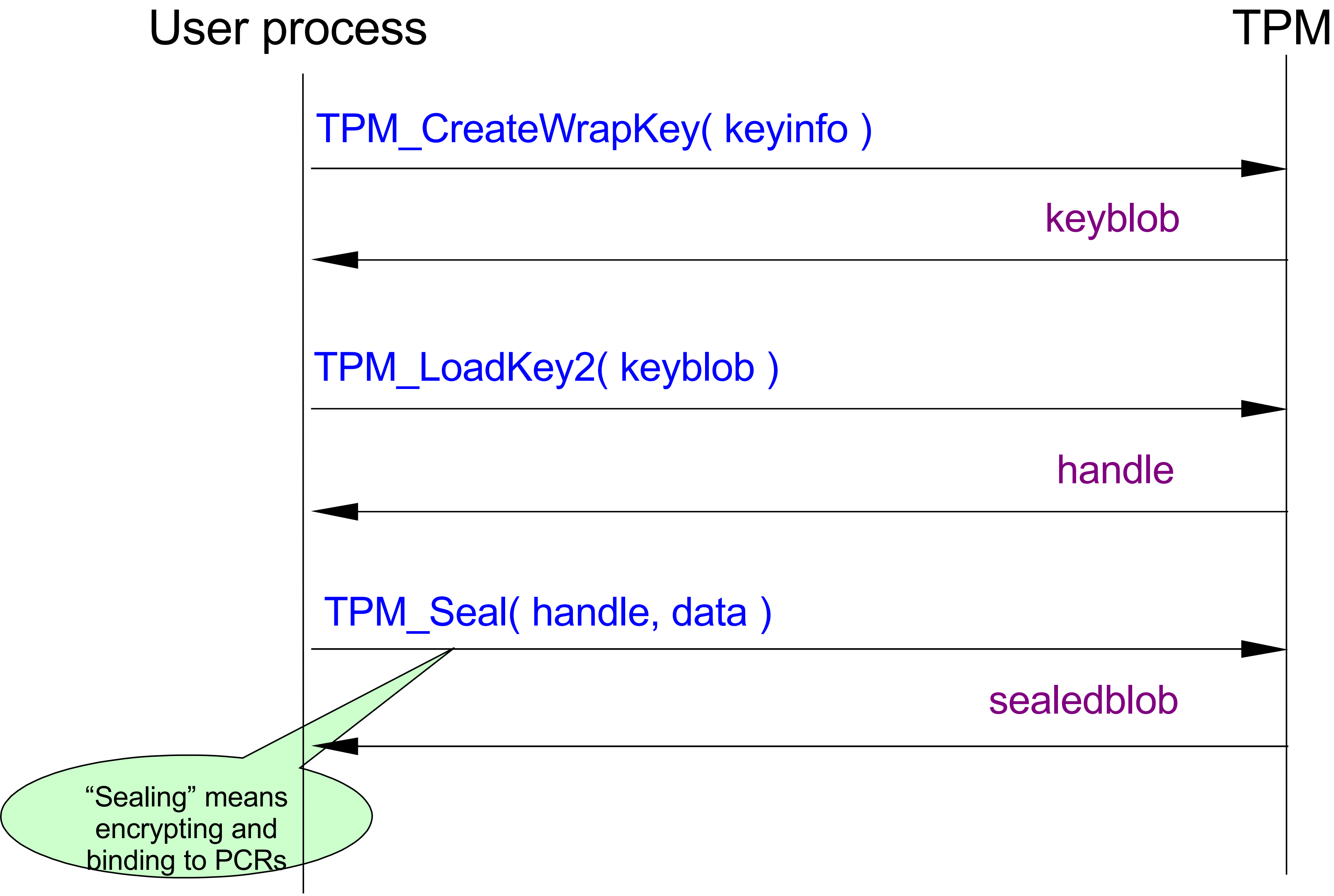
Secure storage

Secure storage

- Keys are created with TPM_CreateWrapKey
  - Passwords (known as “authdata”) are specified for each key
  - Keys are arranged in a tree hierarchy
  - The TPM returns the created key as a blob; the secret parts are encrypted with the parent key

- The function TPM_Seal encrypts data
  - It also “seals” it to specified PCR values
  - The command returns the sealed blob
  - The sealed blob is protected by another piece of authdata, specified at the seal time

TPM command message flow (abstract view)



“Sealing” means encrypting and binding to PCRs

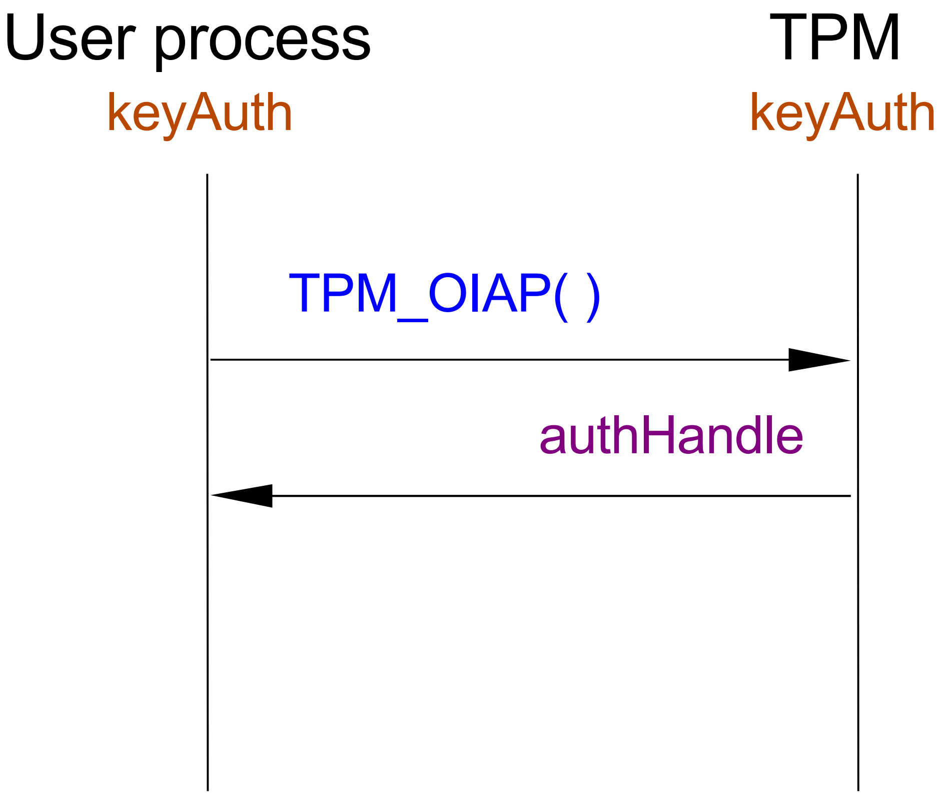
TPM authData

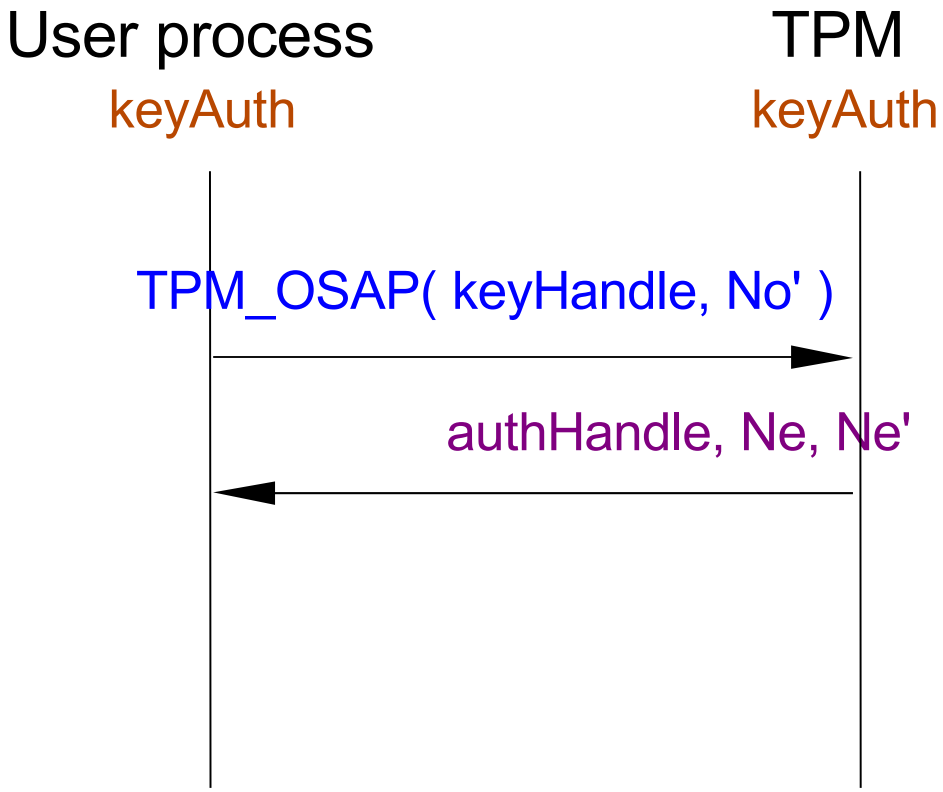
- To each TPM object or resource is associated an authData value
  - A 160-bit shared secret between user process and TPM
  - Think of it as a password that has to be cited to use the object or resource

- authData may be a weak (guessable) secret
  - May be based on a user-chosen password; e.g. in Microsoft Bitlocker.

- The TPM resists online guessing attacks of weak authdata by locking out a user that repeatedly tries wrong guesses
  - Details are left to manufacturer

OIAP and OSAP

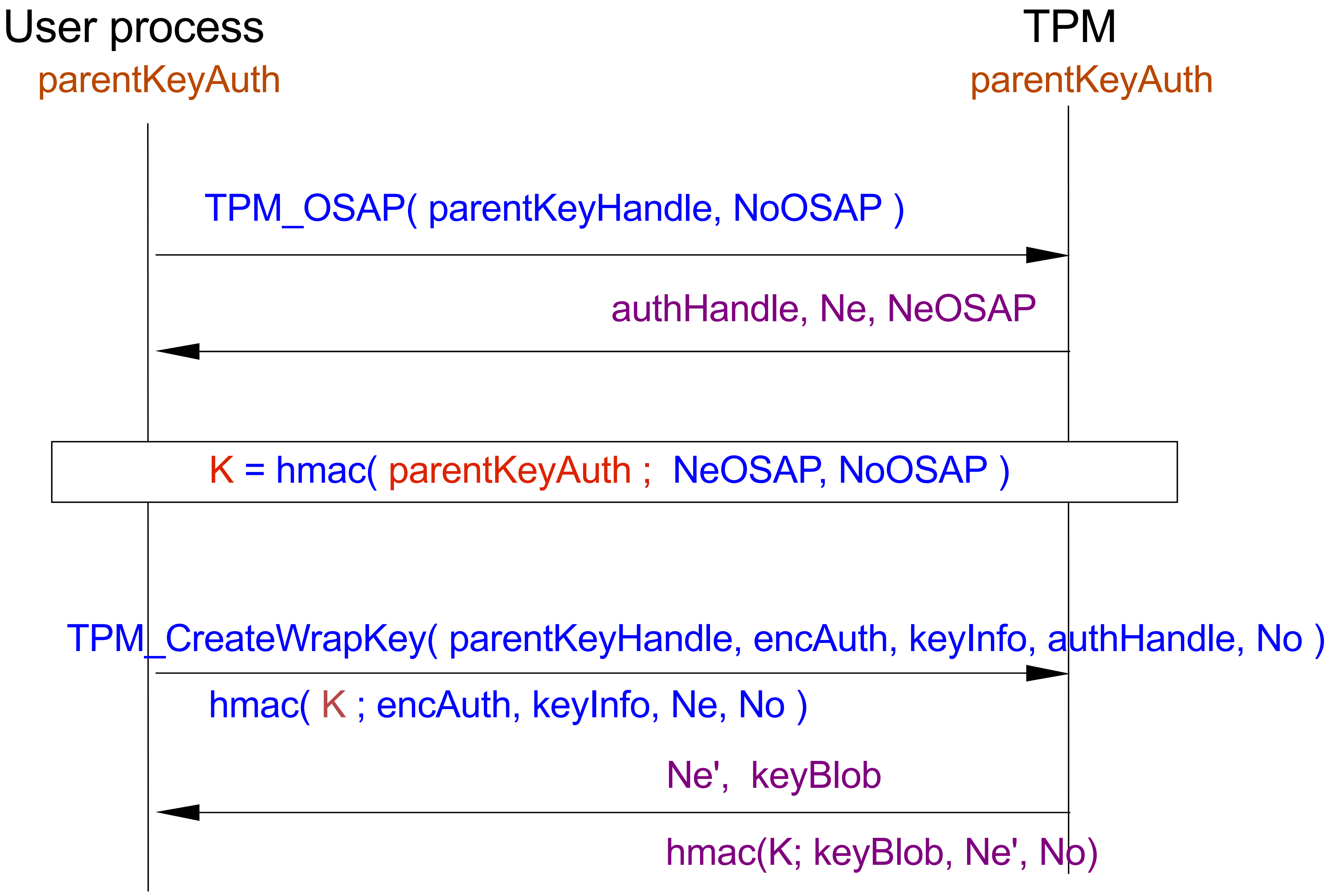




- Long-lived session
- Allows different objects in same session
- Authdata must be cited each command

- Session may be shortlived
- Just one object
- Because K is cached, authdata need not be cited for each command

TPM_CreateWrapKey in more detail



User process

parentKeyAuth

TPM

parentKeyAuth

TPM_OSAP(parentKeyHandle, NoOSAP)

authHandle, Ne, NeOSAP

K = hmac(parentKeyAuth ; NeOSAP, NoOSAP)

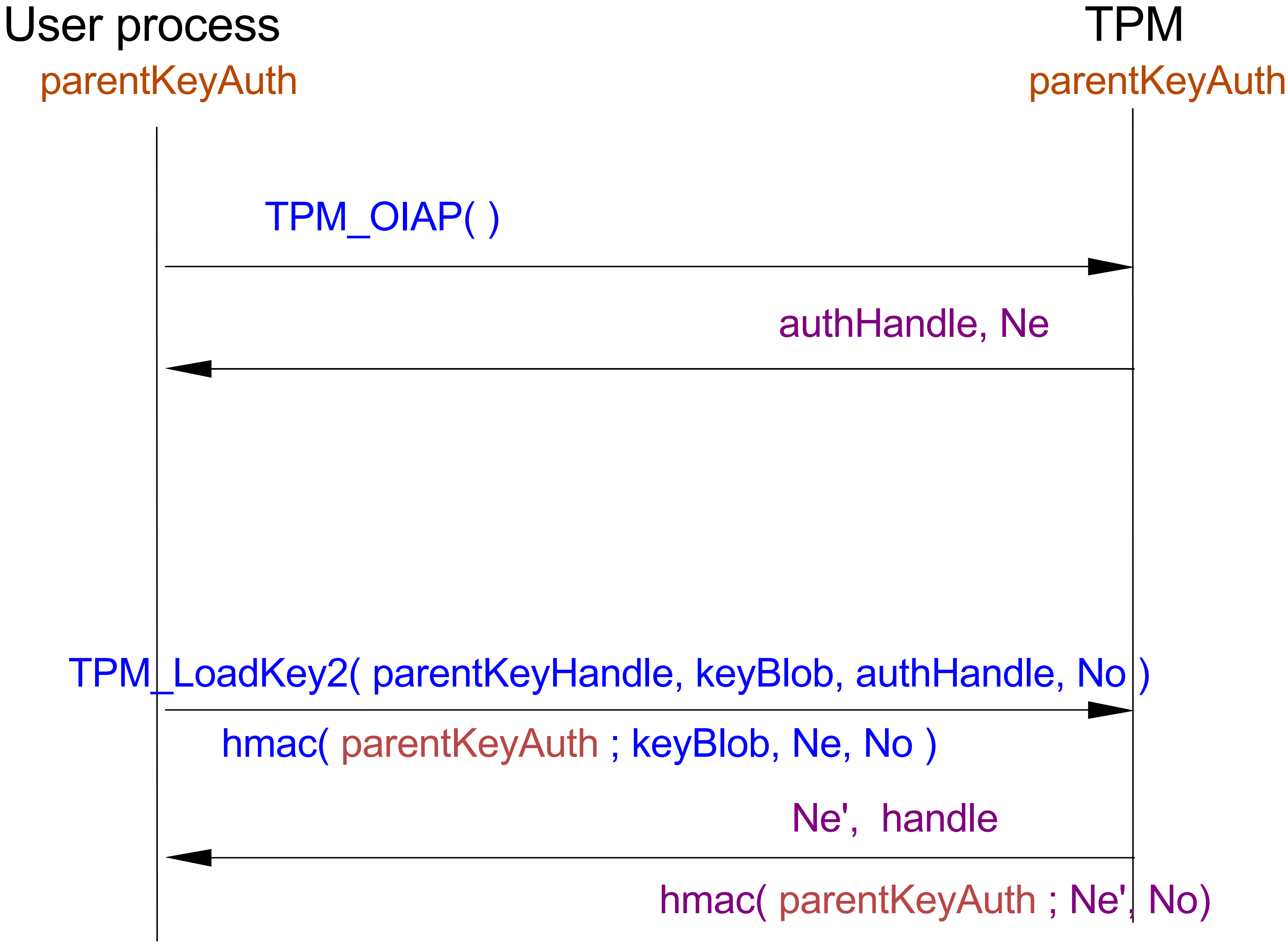
TPM_CreateWrapKey(parentKeyHandle, encAuth, keyInfo, authHandle, No)

hmac(K ; encAuth, keyInfo, Ne, No)

Ne', keyBlob

hmac(K; keyBlob, Ne', No)

TPM_LoadKey2 in more detail



User process

parentKeyAuth

TPM

parentKeyAuth

TPM_OIAP( )

authHandle, Ne

TPM_LoadKey2(parentKeyHandle, keyBlob, authHandle, No)

hmac(parentKeyAuth ; keyBlob, Ne, No)

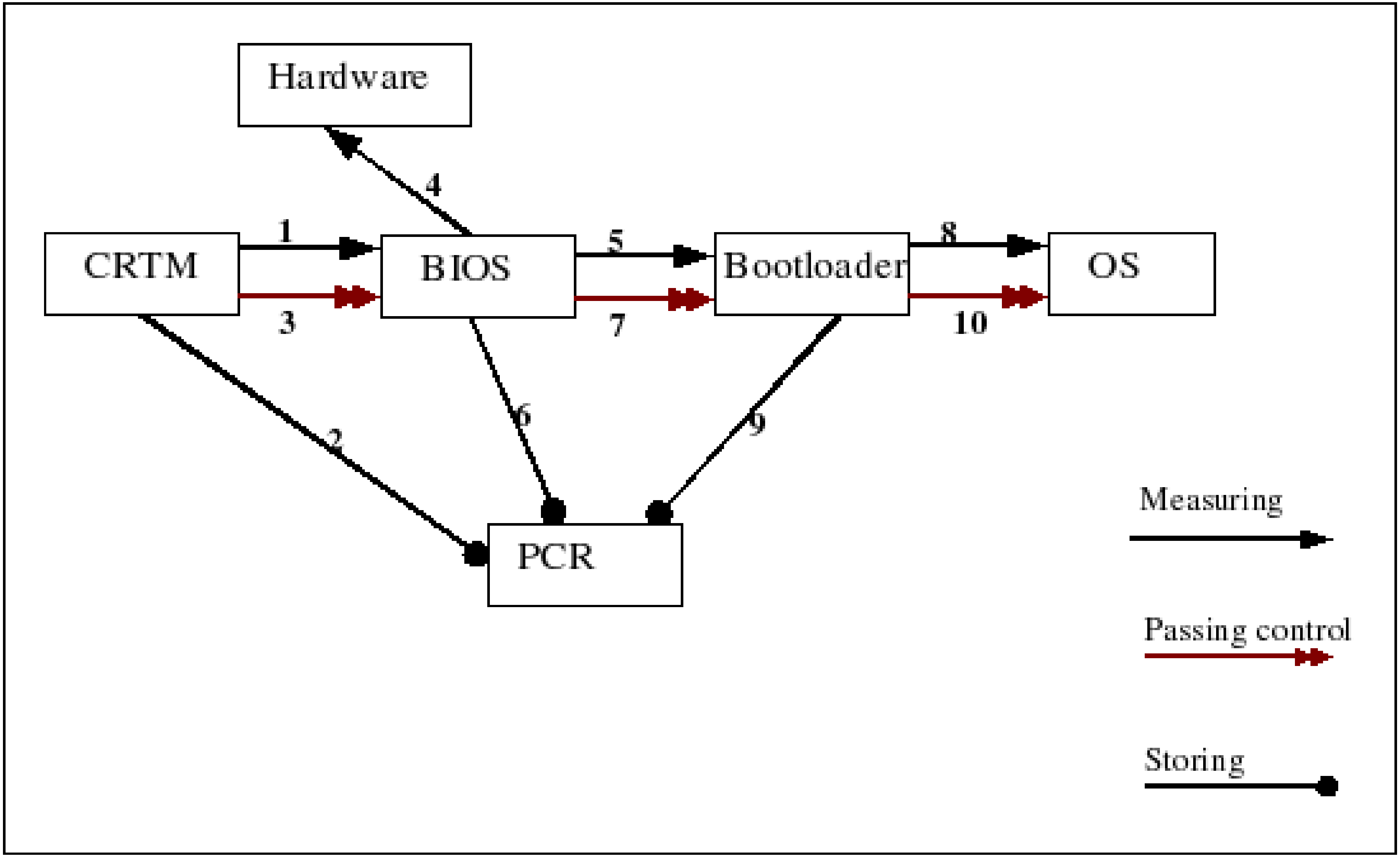
Ne', handle

hmac(parentKeyAuth ; Ne', No)

Platform measurement

- The TPM has 24 Platform Configuration Registers (PCRs)
  - Used to record platform configuration
  - x is a “measurement” of some part of the platform
  - TPM_Extend(p,x) “stores” the value x on the PCR p
    - TPM_Extend(p,x) means:
      p := SHA1(p || x)
  - p contains a proof of the record of the values that have been extended into it.

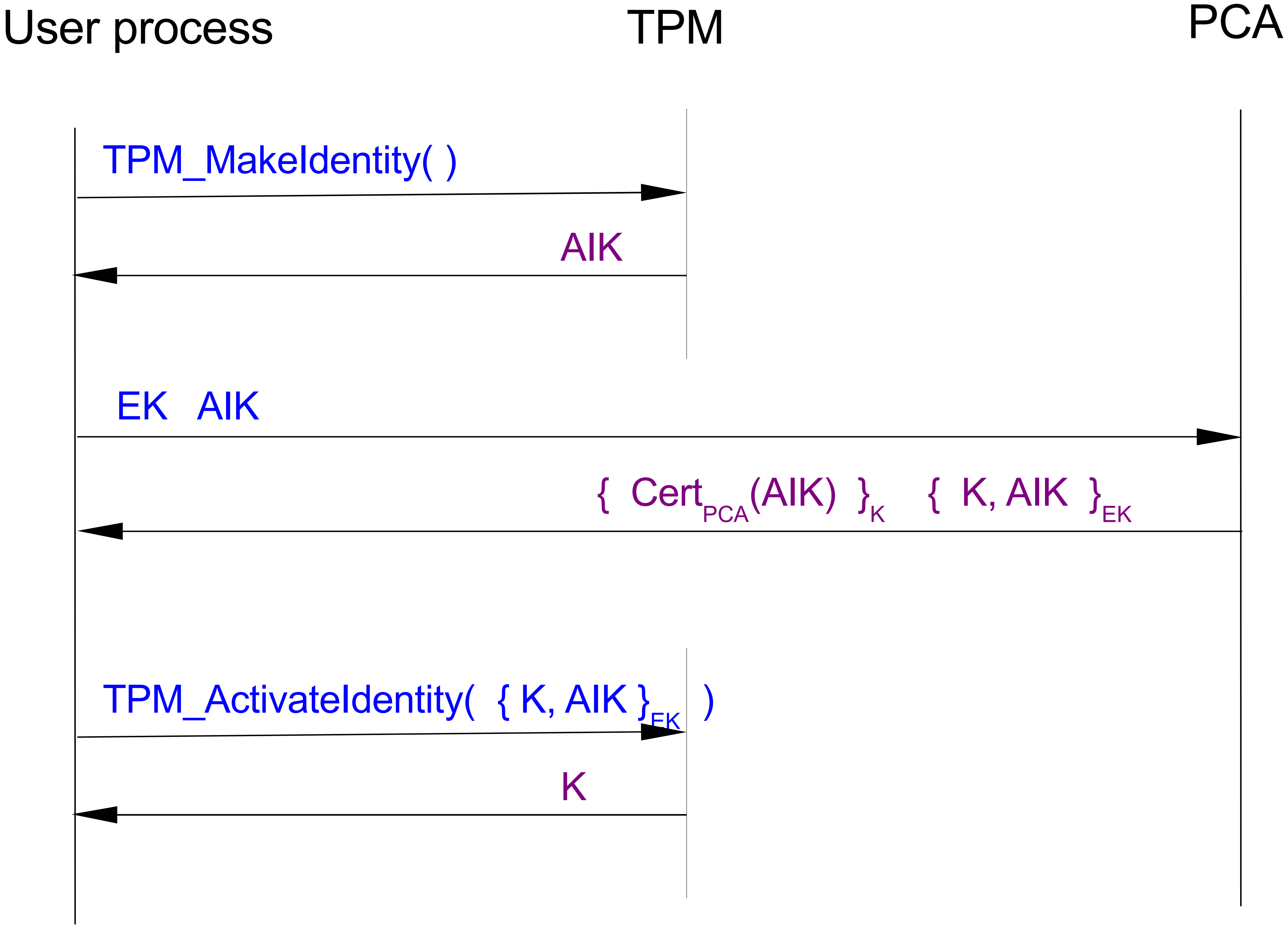
Core root of trust for measurement



Platform integrity reporting

- TPM_Quote returns a signature (using a TPM key) on the PCR p.
- A remote party can use that to be convinced of the integrity of the platform
- The key used is an attestation identity key (AIK), that has a certificate demonstrating that it is a real TPM key.

Attestation using a Privacy CA



User process

TPM

PCA

TPM_MakeIdentity( )

AIK

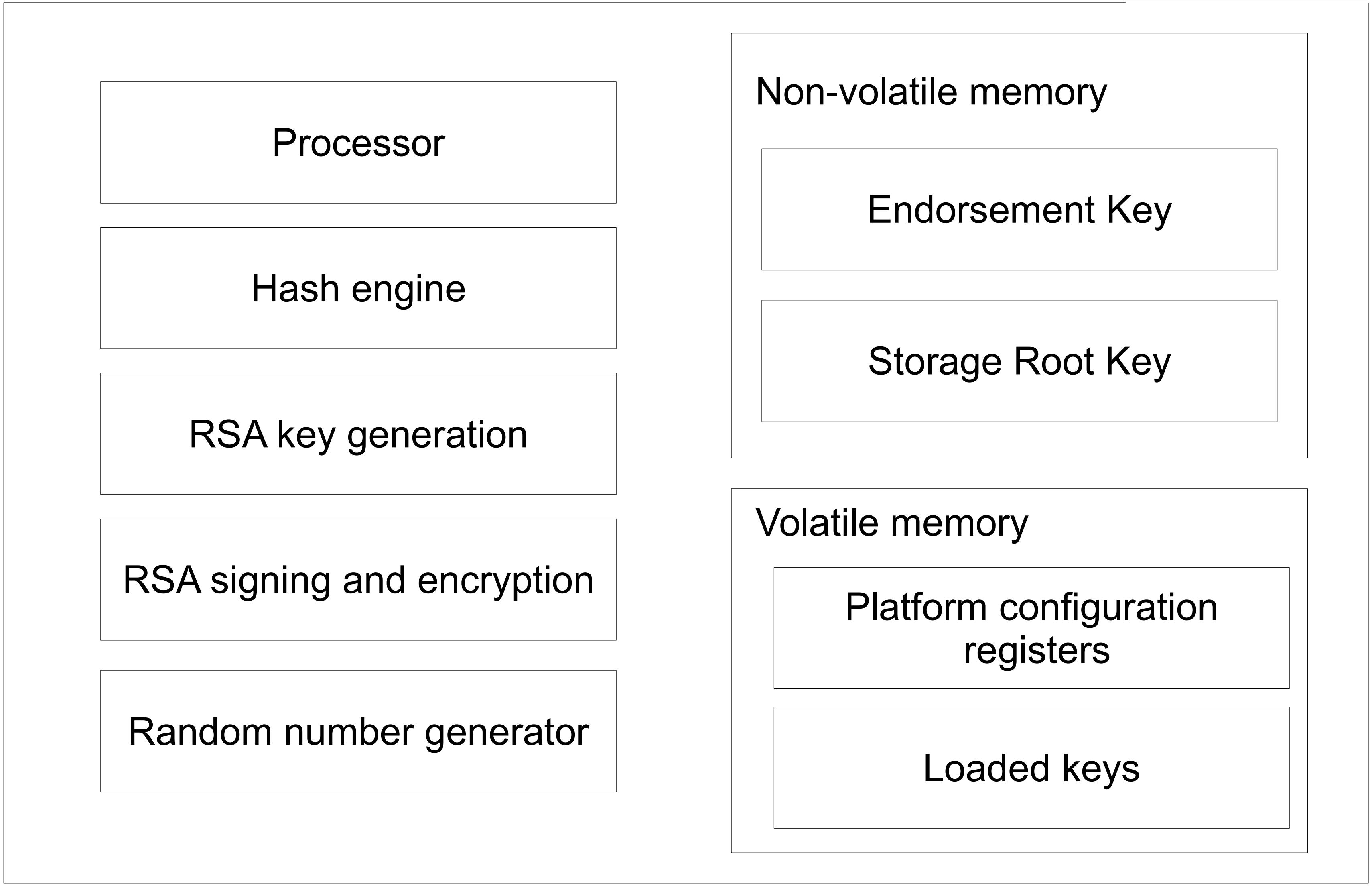
EK AIK

$\{ \text{Cert}_{PCA}(\text{AIK}) \}_K \quad \{ \text{K, AIK} \}_{EK}$

TPM_ActivateIdentity($\{ \text{K, AIK} \}_{EK}$)

K

TPM architecture



Processor

Hash engine

RSA key generation

RSA signing and encryption

Random number generator

Non-volatile memory

Endorsement Key

Storage Root Key

Volatile memory

Platform configuration registers

Loaded keys

TPM: summary

- Commands
    - Authdata
- Storage
- Platform integrity measurement
- Platform integrity reporting
    - Attestation
    - Privacy preserving

MS BitLocker and TPM

How to ensure only MSBL has access to volume decryption key? [Simplified story]

- On boot, control passes to pre-bios.
- Pre-bios measures bios, extends PCR, passes control.
- Bios measures other hardware and MBR, extends PCR, passes control.
- MBR measures MSBL, extends PCR, passes control. Begin window.
- MBSL retrieves vol id key and extends PCR with “stop value”. End window.
- MBSL starts decrypting disk and launches OS.