MASTER OF SCIENCE
IN ENGINEERING

Trusted Platform Module (TPM) - PCR & Seals

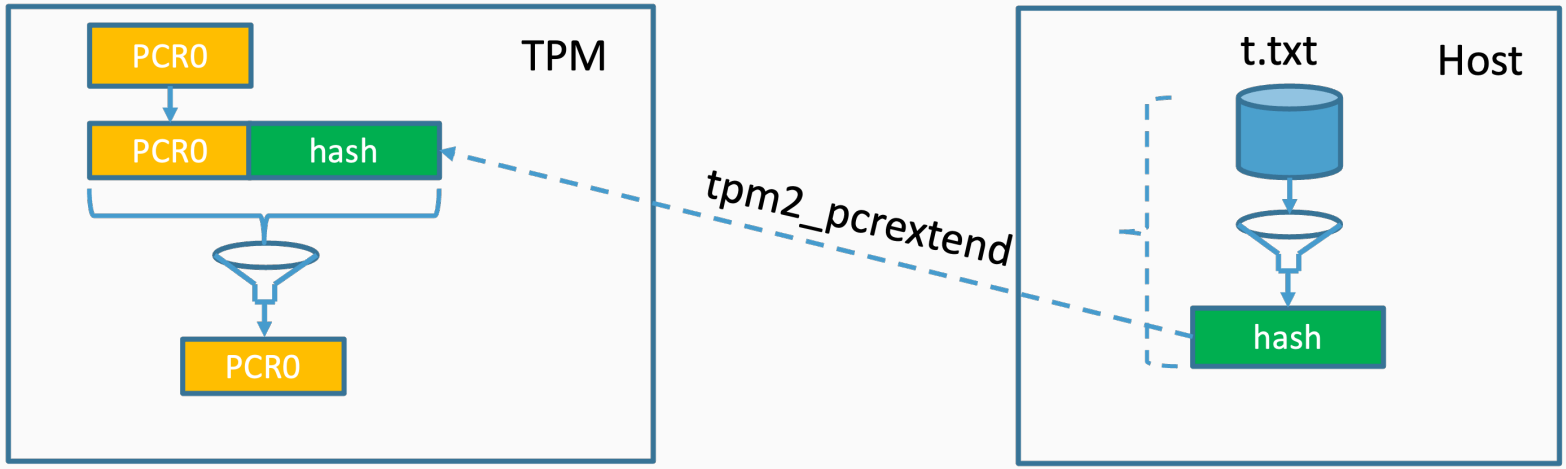
Luca Haab

December 15, 2025

9. Platform Configuration Registers (PCR)

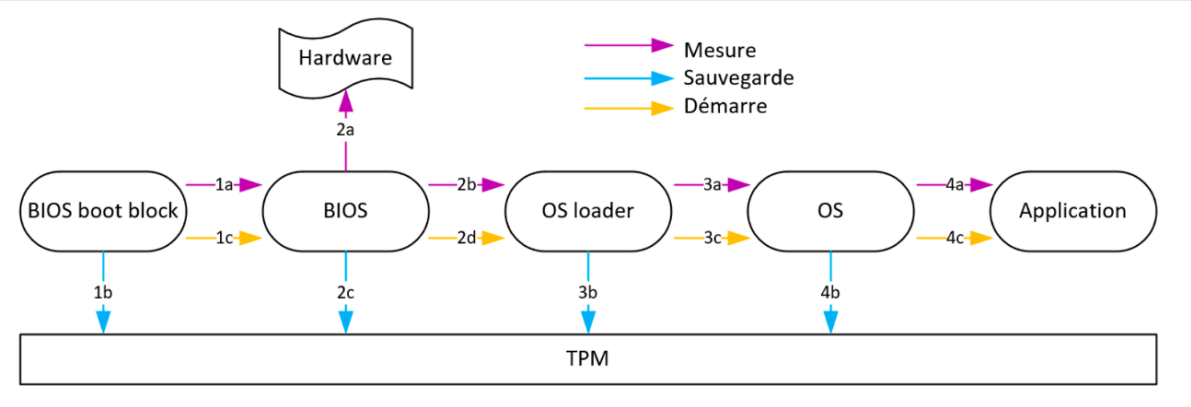
- Platform Configuration Registers (PCRs) are one of the essential features of a TPM. The prime use case is to provide a method to cryptographically record software state or configuration data used by a device.
- PCR are special TPM2 objects that can only be modified or written to by hash mechanism. In that, the incoming new value is concatenated with the existing value in the PCR and hashed. The PCR update calculation is called an extend

```
PCR new value = Digest of (PCR old value || data to extend)
// In words, it takes the old PCR value and concatenates some data to be extended
```



9. PCR: Secure Boot

- A platform begins with trusted software called the core root of trust measurement (CRTM). The CRTM measures (calculate a digest of) the next **software** to be run and extends that digest into an **even PCR**. It then extends that software's **configuration data** into an **odd PCR**. This software, perhaps a BIOS, in turn measures and extends the next software, perhaps a master boot record. The measurement chain continues through the early OS kernel code and perhaps further. Security-critical configuration files are also measured.
- The net result is that the PCR value represents the history of all measurements extended into it. Because of the one-way nature of a secure digest, there is no way to undo a measurement (to extend the PCR back to a desired value).



Info

Usually PCR are initialized to 0.

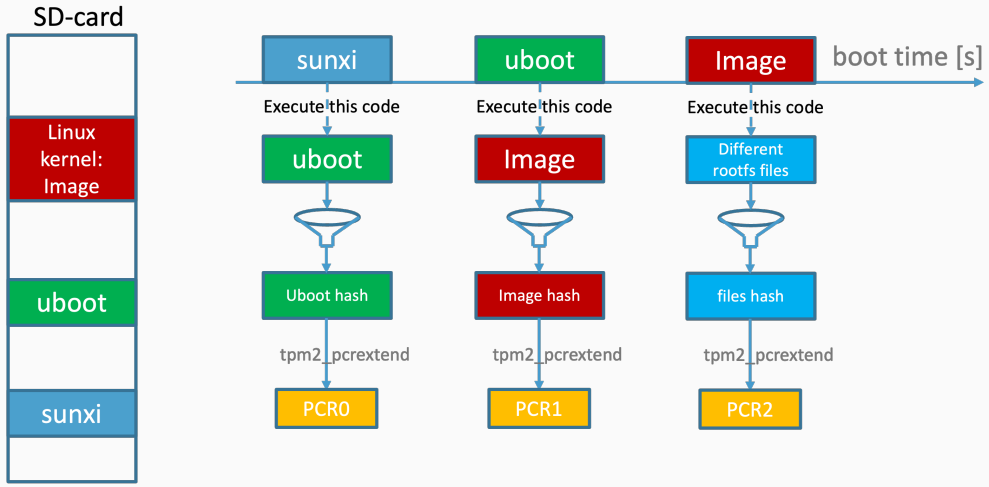
9. PCR: Exemplary PCR Number Allocation

In practice, a TPM contains multiple PCRs. A PC Client platform requires 24 PCRs and this minimum is expected to be the actual number in PCs. Automotive TPMs may have many more.

The platform TPM specification specifies the PCR attributes and a platform software specification standardizes what measurements go into which PCRs.

PCR Number	Allocation
0	BIOS
1	BIOS configuration
2	Option ROMs
3	Option ROM configuration
4	MBR (master boot record)
5	MBR configuration
6	State transitions and wake events
7	Platform manufacturer specific measurements
8-15	Static operating system
16	Debug
23	Application support

9. PCR: Nano Pi Secure Boot



- Nanopi boot: 1st sunxi ⟹ 2nd uboot ⟹ 3rd Linux
- The security of the secure boot process depends on the security of the first bootable code (sunxi code). The first bootable code is the root of trust.
- Example: sunxi hashes uboot and and extend the hash to PCR0
- The Linux open source Integrity Measurement Architecture (IMA) integrates boot-time measurements into the kernel.

9. PCR: Session, Policy Commands

- `tpm2_startauthsession`

  Starts a session with the TPM: There are two types of policy session:
  - *TPM_SE_TRIAL (default)*: used for building a policy
  - *TPM_SE_POLICY*, used for authenticating-authorizing, with a policy, the use of an object

- `tpm2_policypcr`

  Generates a *PCR policy event* with the TPM. A PCR policy event creates a policy bound to specific PCR values and is useful within larger policies constructed using policyor and policyauthorize events.

  Example: create a policy that only allows access to a key if PCR 0 contains a specific value (e.g. after a trusted boot process)

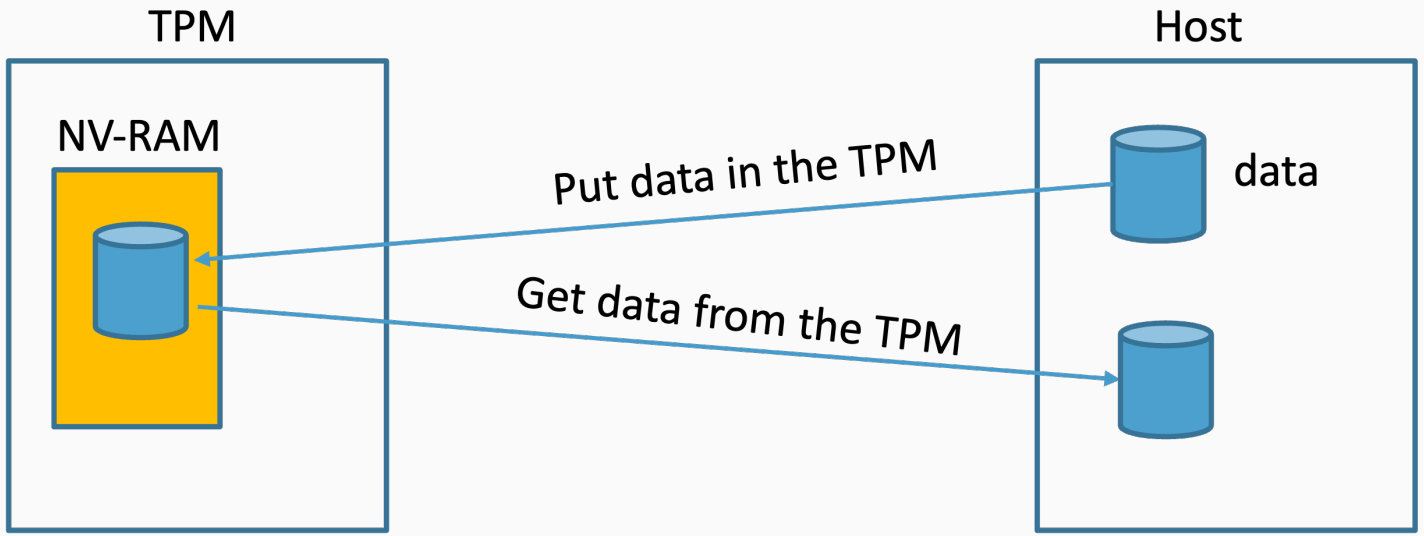
```
tpm2_pcrextend  : Write (extend) PCR
tpm2_pcrread    : Read PCR
tpm2_pcrreset   : Reset PCR
```

9. PCR: Commands in Practice

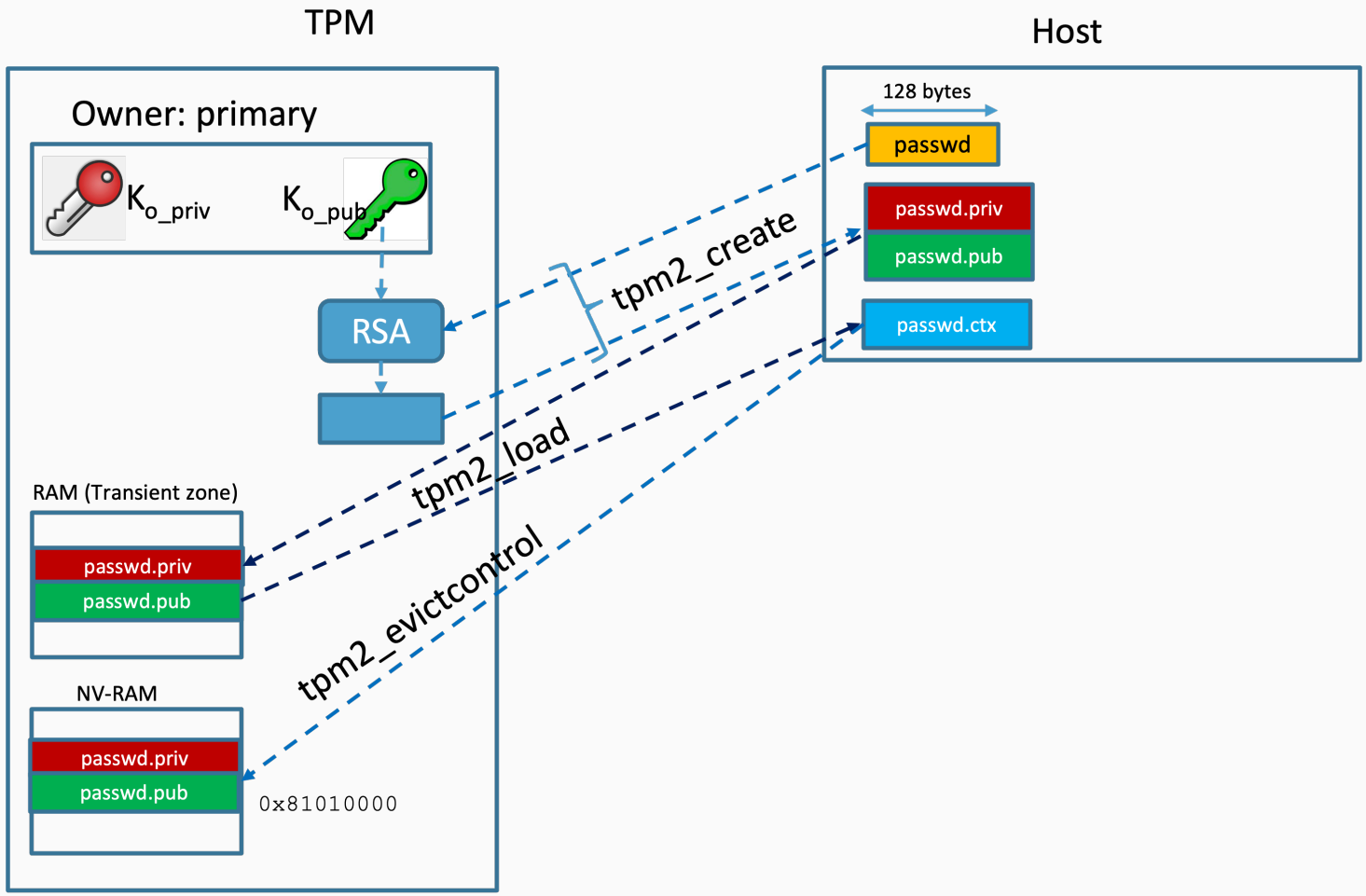
```
tpm2_pcrread
  sha1:
    0 : 0x36509D3791C92465563AD8447F99F19A5BB3A602
    1 : 0x0000000000000000000000000000000000000000
  sha256:
    0 : 0x0000000000000000000000000000000000000000000000000000000000000000
    1 : 0x0000000000000000000000000000000000000000000000000000000000000000
  sha384:
    0 : 0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    1 : 0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  sha512:
    0 : 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    1 : 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

tpm2_pcrread sha1:0
sha1:
  0 : 0x36509D3791C92465563AD8447F99F19A5BB3A602


tpm2_pcrextend 0:sha1=8c8393ac8939430753d7cb568e2f2237bc62d683

tpm2_pcrread
  sha1:
    0 : 0x3E8485A1670B31C2E5C4B818933F38C4EDF50265
    1 : 0x0000000000000000000000000000000000000000
```

10. Seal (*save*) Data on TPM

- Put (seal) and get (unseal) data in the TPM
- Data is stored in the NV-RAM (No volatile Ram)
- The max size of the data is 128 bytes



10. Seal (*save*) Data on TPM



10. Seal (*save*) Data on TPM

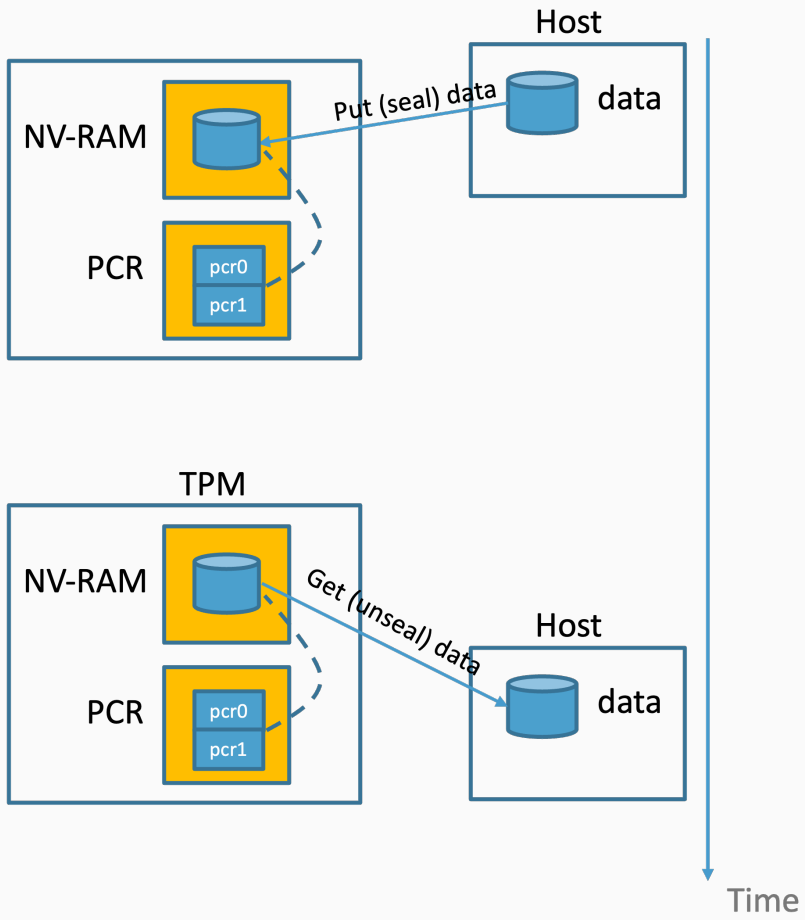
Example: a *password* is in the file `passwd`

```
tpm2_createprimary -C o -G rsa2048 -c primary
tpm2_create -C primary -i passwd -u passwd.pub -r passwd.priv
tpm2_load -C primary -u passwd.pub -r passwd.priv -c passwd.ctx
tpm2_evictcontrol -c passwd.ctx 0x81010000 -C o


shred passwd
rm -f passwd

tpm2_unseal -c 0x81010000 > passwd
```

Info

Source: https://tpm2-software.github.io/2020/04/13/Disk-Encryption.html

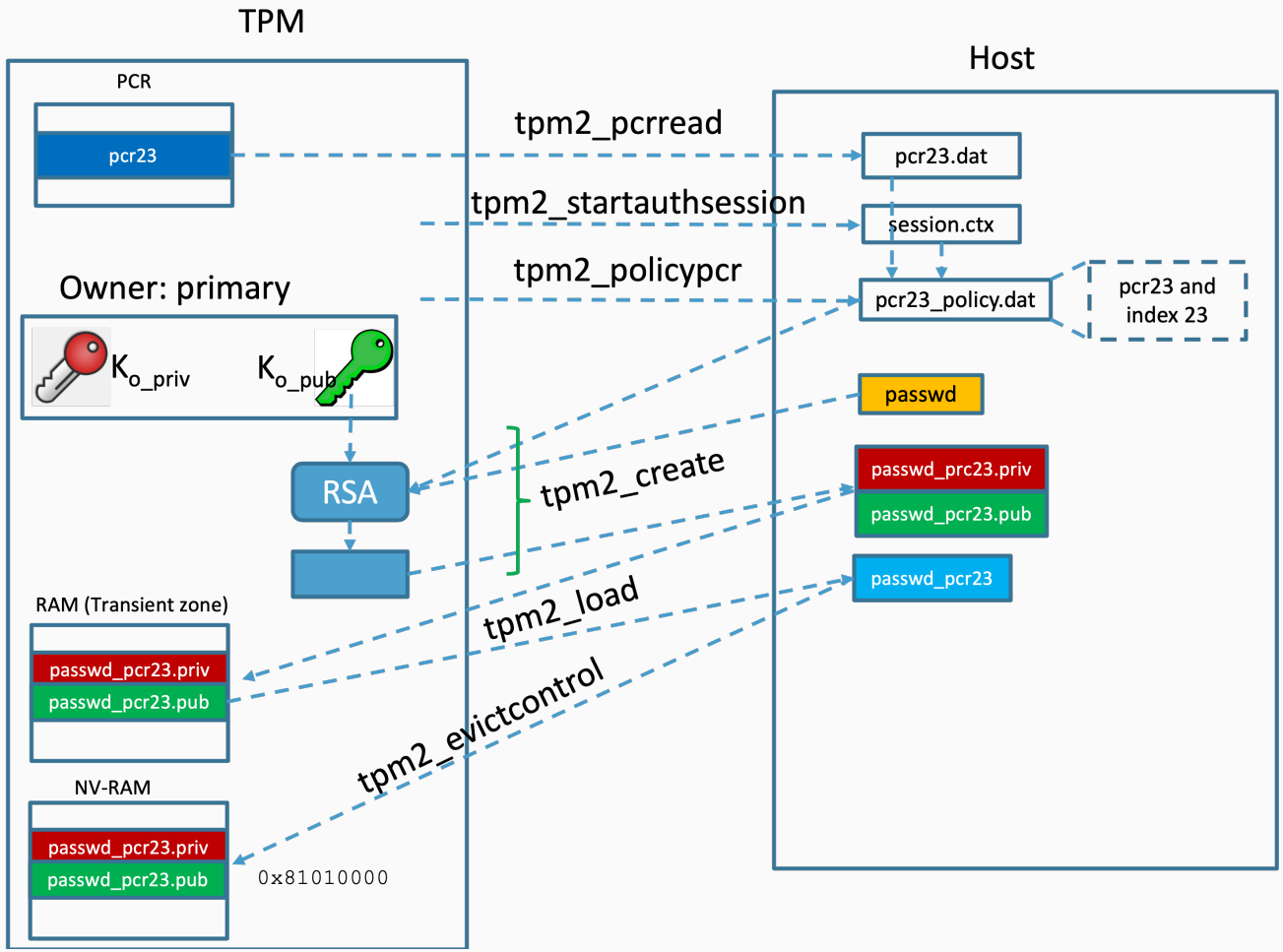
10. Seal (*save*) Data on TPM



Data are saved in the TPM (NV_RAM) and there is a link (policy pcr) with the values of some PCR registers.

It is possible to get (unseal) data, only if the values of the corresponding PCV registers are not changed.

10. Seal (*save*) Data on TPM



10. Seal (*save*) Data on TPM

Example: a *password* is in the file passwd

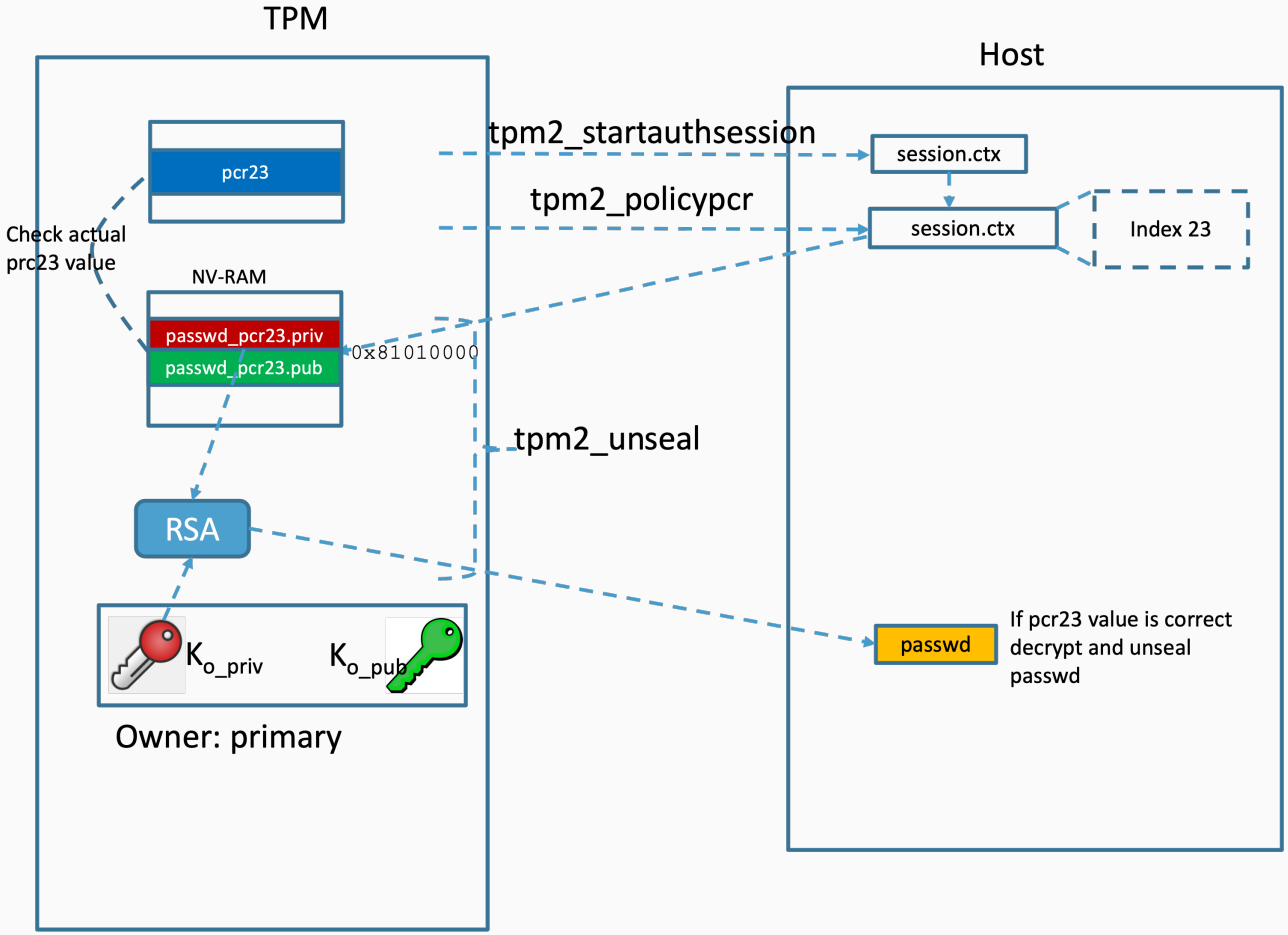
```
sha1sum passwd // 8c8393ac8939430753d7cb568e2f2237bc62d683
tpm2_pcrreset 23
tpm2_pcrextend 23:sha1=8c8393ac8939430753d7cb568e2f2237bc62d683

// Seal and protect it with PCR23
tpm2_pcrread sha1:23 -o pcr23.dat
tpm2_startauthsession -S session.ctx
tpm2_policypcr -S session.ctx -l "sha1:23" -f pcr23.dat -L
pcr23_policy.dat
// Create and Seal the passwd object
tpm2_create -C o_primary.ctx -i passwd \
            -L pcr23_policy.dat -u passwd_pcr23.pub -r passwd_pcr23.priv
// Load the object into the TPM and make it persistent
tpm2_load -C o_primary.ctx -u passwd_pcr23.pub \
          -r passwd_pcr23.priv -c passwd_pcr23
tpm2_evictcontrol -c passwd_pcr23 0x81010000 -C o

// Delete files on the host
shred passwd
rm -f passwd
rm session.ctx pcr23.dat pcr23_policy.dat passwd_pcr23.pub \
passwd_pcr23.priv passwd_pcr23

// Unseal and get the password - but only if PCR23 value is correct
tpm2_startauthsession --policy-session -S session.ctx
tpm2_policypcr -S session.ctx -l "sha1:23"
// BINGO!! You got passwd back
tpm2_unseal -p session:session.ctx -c 0x81010000 > passwd
```

Info

Make sure you have:

```
tpm2_createprimary -C o -G rsa2048 -c primary.ctx
```

10. (un)Seal Data from TPM



Session and Authorization: Compared and Contrasted

- Sessions are used for managing context and enforcing policies, but they don't inherently handle authentication by themselves.
- Authentication is required for ensuring that the user or process performing the operation is authorized.
- For security, use authenticated sessions for any sensitive TPM operations (like modifying PCRs, sealing/unsealing data, creating objects).
- By binding TPM operations to authenticated sessions, you can ensure that only authorized users can modify or interact with the content stored in the TPM.

> **Info**
>
> A more technically accurate name for `tpm2_startauthsession` command would have been `tpm2_startsession`.

Source: *A Practical Guide to TPM 2.0 - Using the Trusted Platform Module in the New Age of Security*