Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

TPM, Trusted Platform Module

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

References

[1]: A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security. Will Artur & David Challener. 2015 by Apress Media, LLC, 978-1-4302-6583-2.

[2]: https://en.wikipedia.org/wiki/Trusted_Platform_Module

[3]: https://trustedcomputinggroup.org

[4]: https://trustedcomputinggroup.org/resource/tpm-library-specification/. // TPM description is describe in 4 files

TPM-Rev-2.0-Part-1-Architecture-01.16.pdf

TPM-Rev-2.0-Part-2-Structures-01.38.pdf

TPM-Rev-2.0-Part-3-Commands-01.38

TPM-Rev-2.0-Part-4-Supporting-Routines-01.38-code.pdf

[5]: https://google.github.io/tpm-js/#pg_welcome //Good site in order to check TPM functionalities //in a simulated environment

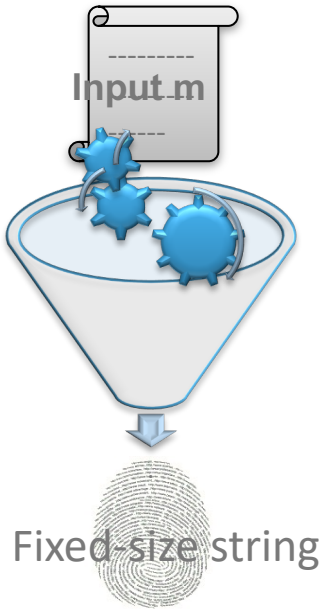
[6]: https://ebrary.net/24701/computer_science/a_practical_guide_to_tpm_20

[7]: Trusted Platform Module on Raspberry PI4, Anthony Claude Florent Alonso Lopez, 2021, Haute école l'ingénierie et d'architecture de Fribourg (Suisse)

[8]: Embedded Security with TPM, Jérome Bagnoud, 2021 HES-SO Master thesis

Hash functions

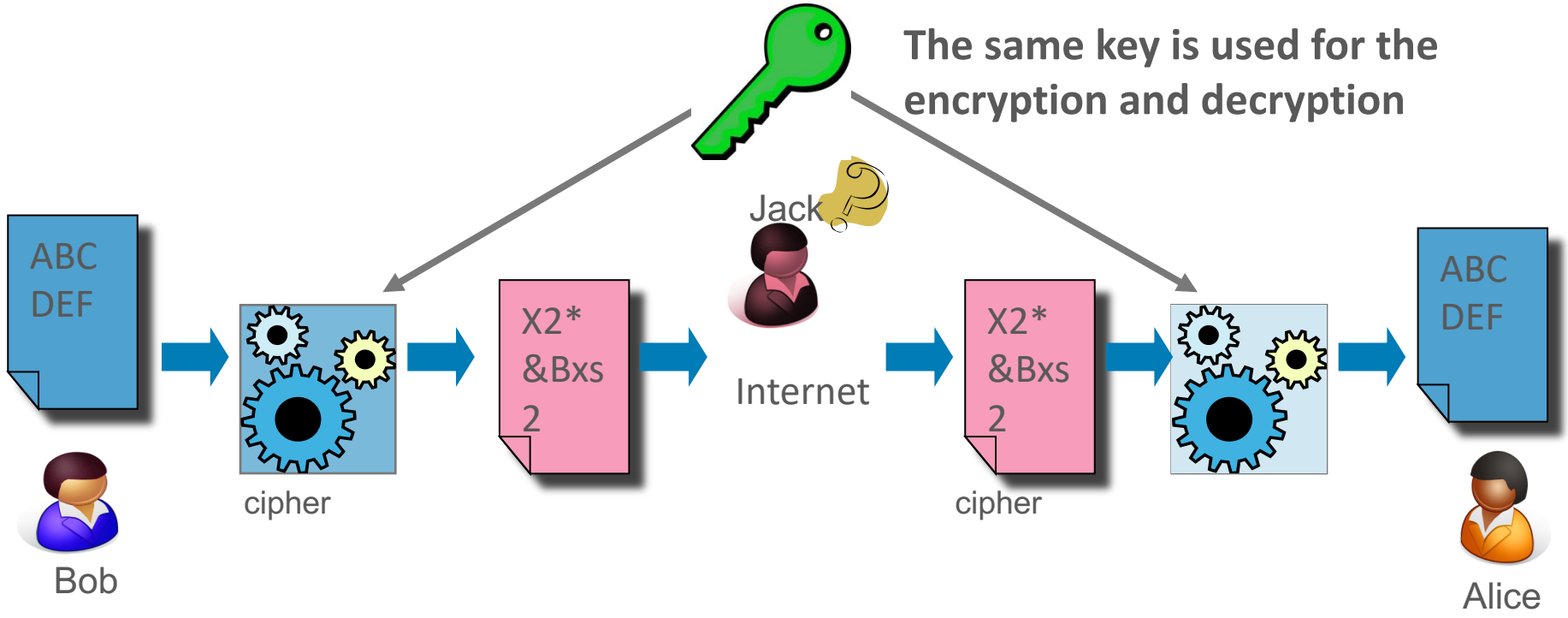
- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H(m)).
- The basic requirements for a cryptographic hash function are as follows.
  - The input can be of any length.
  - The output has a fixed length (MD5: 128 bits, SHA 1: 160 bits, …)
  - H(x) is relatively easy to compute for any given x.
  - H(x) is one-way.
  - H(x) is collision-free.
- Example:
  - MD5 (must no longer be used)
  - SHA-1 (must no longer be used)
  - SHA-2
  - SHA-3
  - Blake2



hash function

```
md5sum file
   a6a0e8d0522e2c5de921b1c455506320
Or
openssl dgst -md5 file
   MD5(file)= a6a0e8d0522e2c5de921b1c455506320
```

Symmetric ciphers



ciphers:

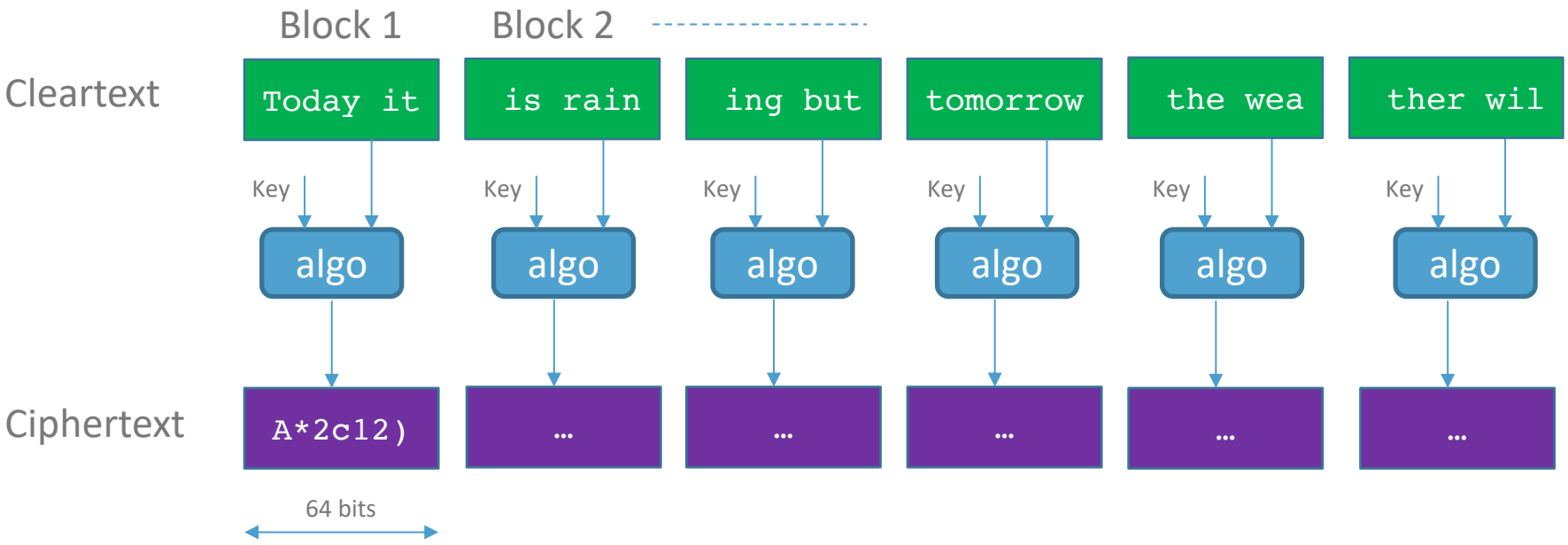
- AES
- CHACHA20
- …

Key length: min 128 bits

Symmetric ciphers, Block Cipher

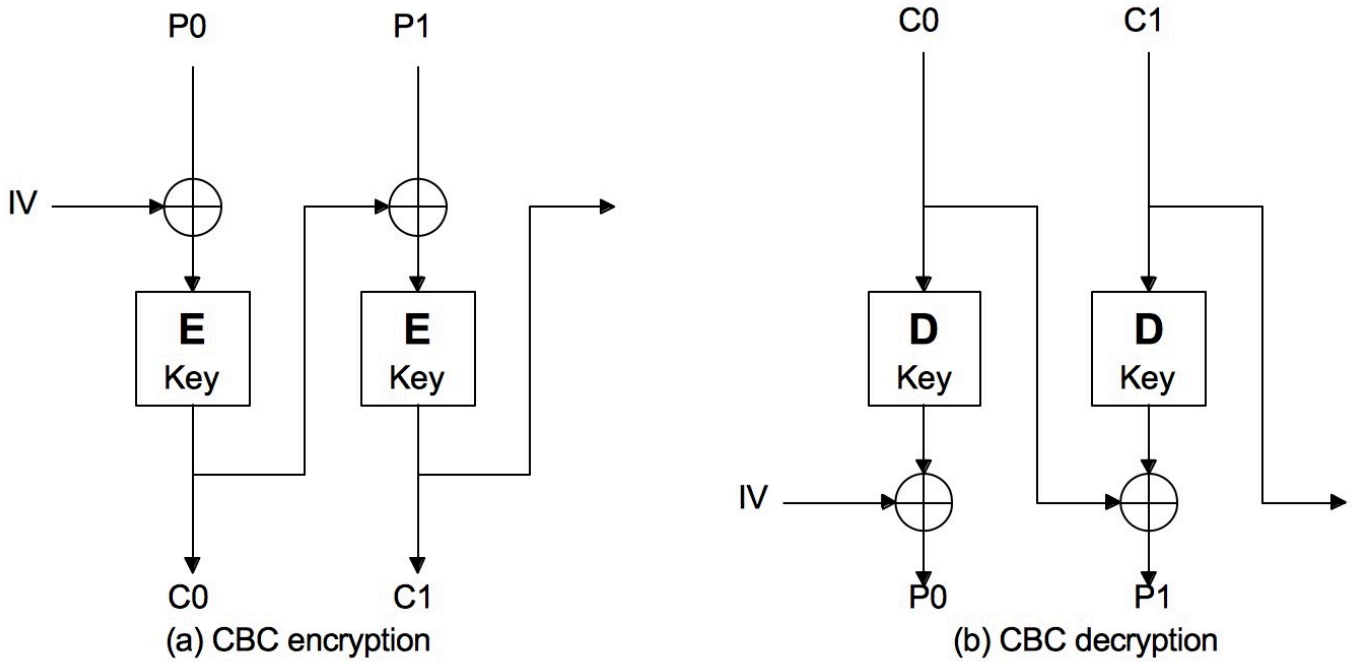
- Cleartext is divided in block. Each block has the same length (64, 128 bits)
- Each block is encrypted by an algorithm (AES, IDEA, 3DES, …) and a key
- Example:
- Cleartext:

Today it	is raining but	tomorrow	the weather will be better



Symmetric ciphers, CBC mode

- CBC (Cipher Block Chaining)
- Every block is coded with the result of the previous block
- IV: Initial Vector for the first block. IV is not secret



Symmetric ciphers: openssl

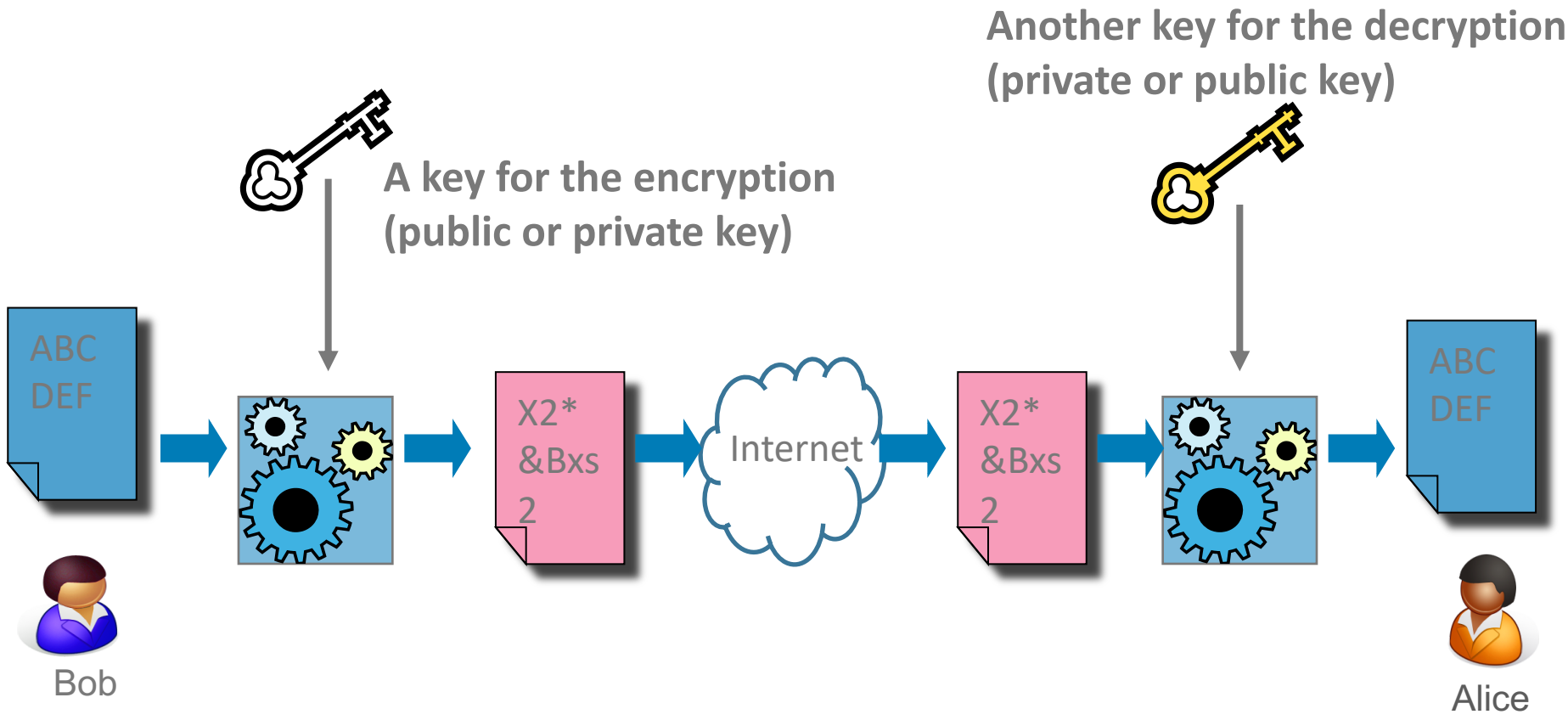
Encrypt the file t.txt with the algorithm aes-256-cbc and save encrypted file to t.enc: IV and key are derived from a password

```
openssl enc -aes-256-cbc -e -in t.txt -out t.enc
```

Decrypt the file t.enc with the algorithm aes-256-cbc and save decrypted file to t.txt

```
openssl enc -aes-256-cbc -d -in t.enc -out t.txt
```

Asymmetric cipher principle

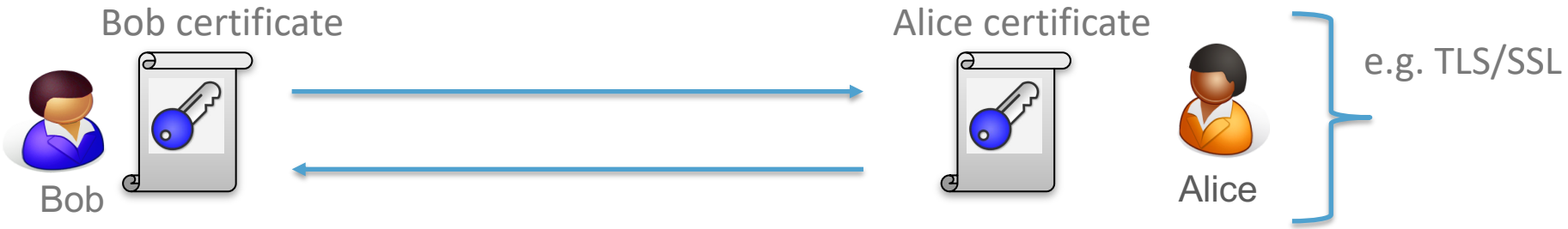


Algorithms

- Elliptical Curves (key length: 256 bits
- RSA, DSA, Diffie-Hellmann : min 2048 bits

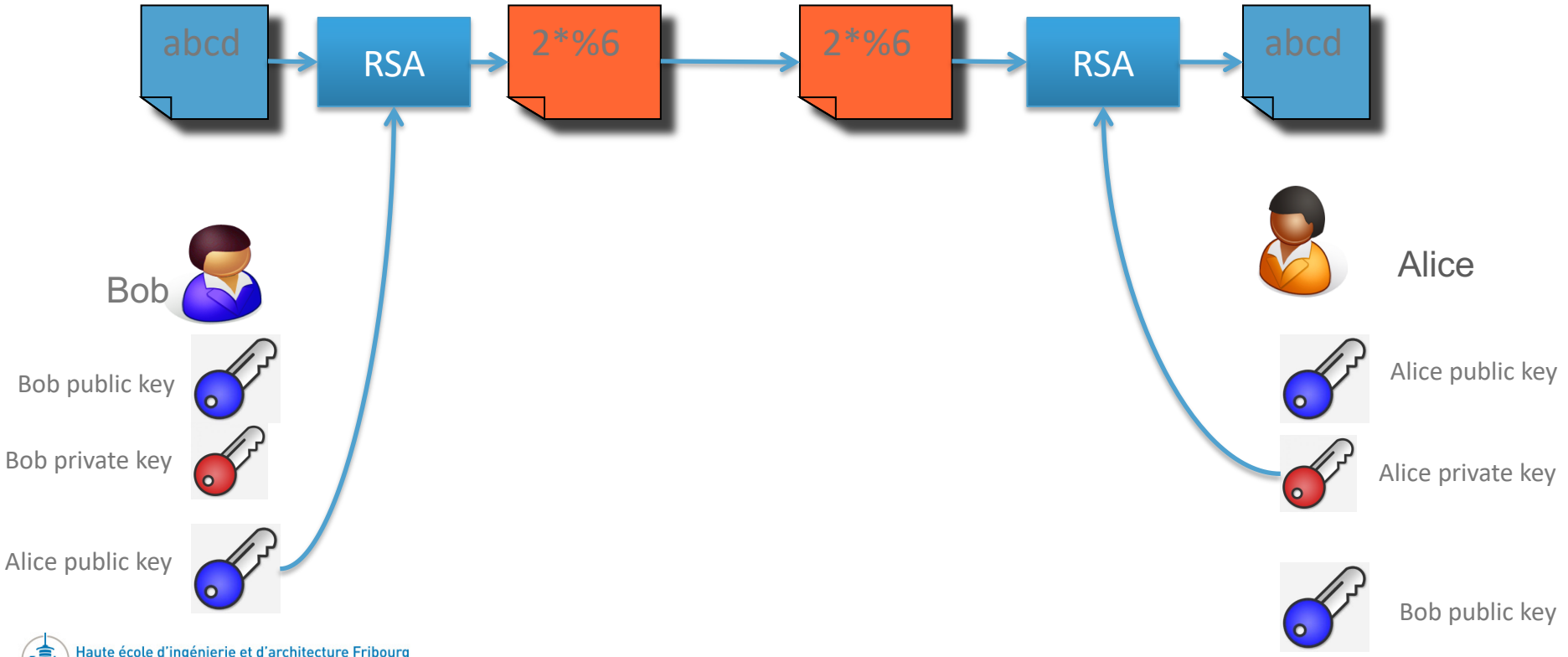
Asymmetric cipher

- Bob and Alice must exchange their public keys
- Generally the public keys are stored in a certificate



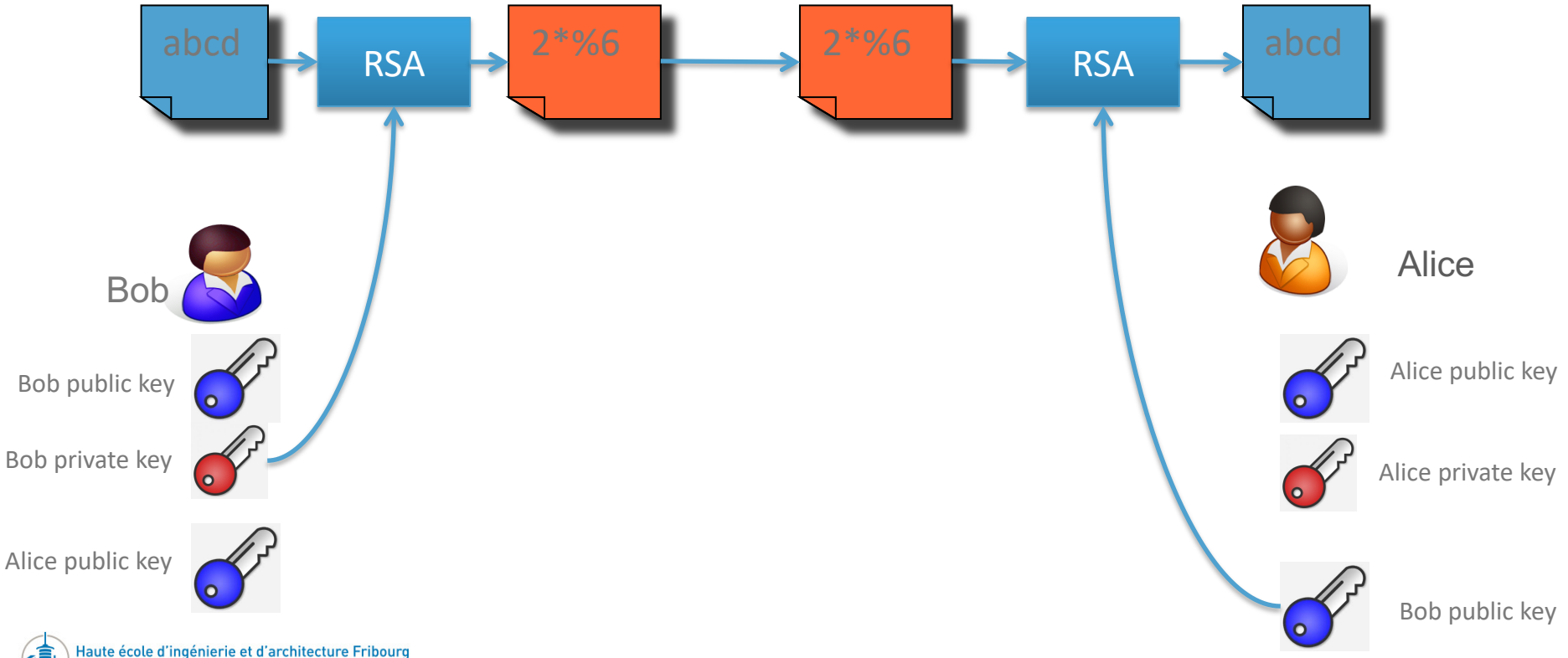
Asymmetric cipher: Confidentiality

- Encrypt with the public key and decrypt with the private key
  - Confidentiality, integrity



Asymmetric cipher: Digital signature

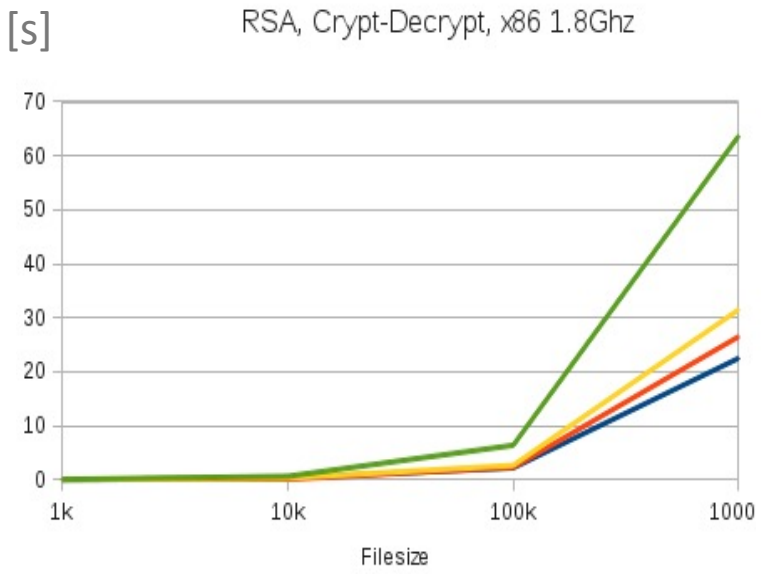
- Encrypt with the private key and decrypt with the public key
  - Authenticity, integrity

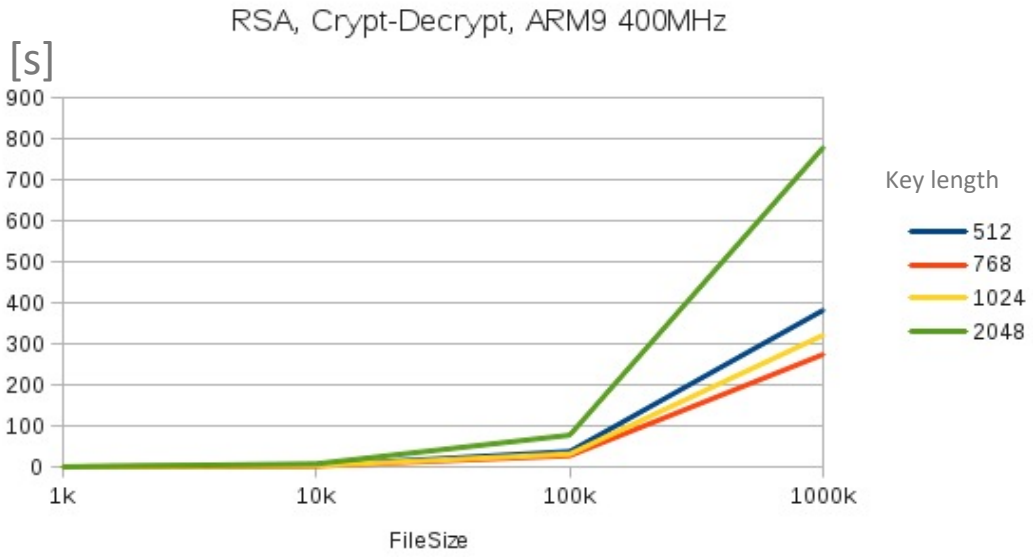


Asymmetric cipher

```
openssl genrsa -out rsa_key.pem 2048          // compute private-public keys
openssl rsa -in rsa_key.pem -pubout -out rsa_key_pub.pem // get public key
openssl rsa -in rsa_key.pem –text             // show private-public keys
openssl rsa –in rsa_key_pub.pem –pubin –text  // show public key

// encrypt t.txt with public key
openssl rsautl -encrypt -pubin -inkey rsa_key_pub.pem -in t.txt -out t.rsa

// decrypt t.rsa with private key
openssl rsautl -decrypt -inkey rsa_key.pem -in t.rsa -out t1.txt
```

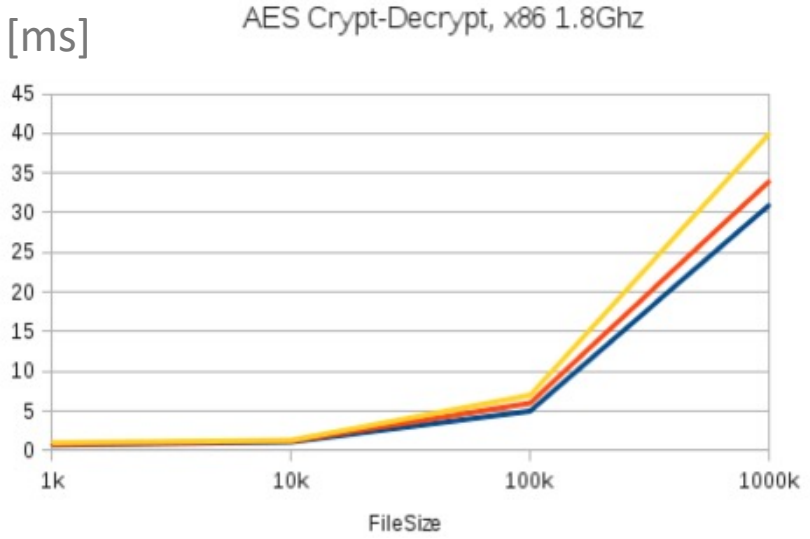
Asymmetric cipher performance

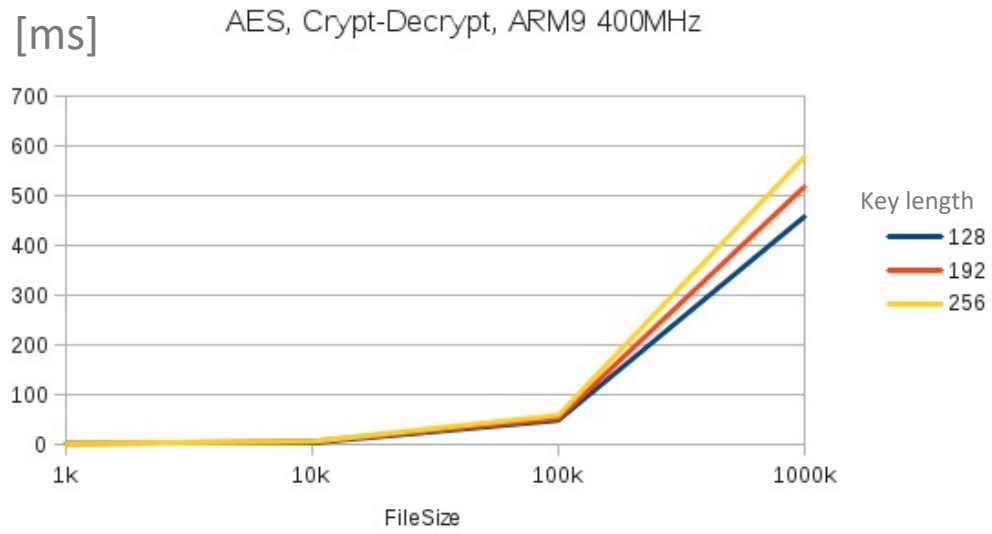




Asymmetric encryption on an ARM processor is 10-12 times slower than on an Intel x86

Symmetric cipher performance



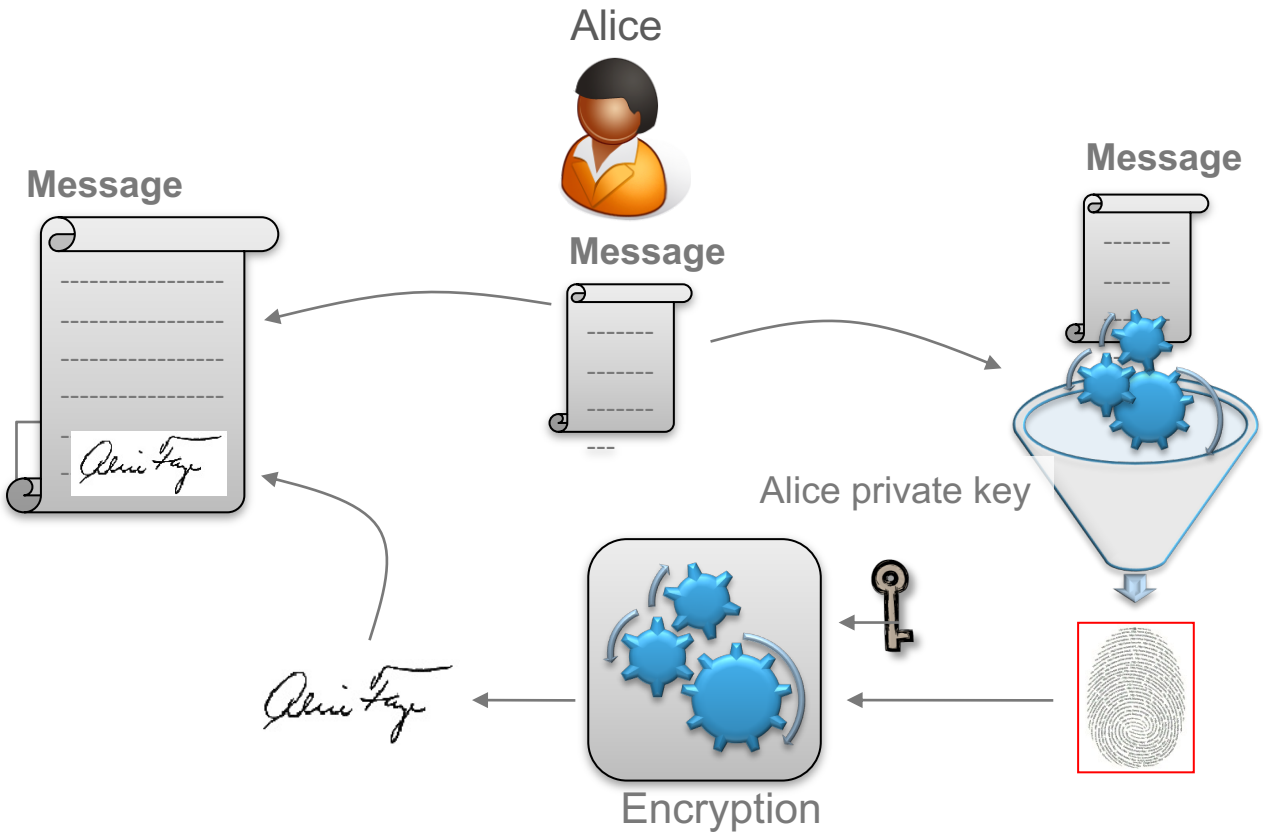


- Symmetric encryption on an ARM processor is 10-15 times slower than on an Intel x86
- Symmetric encryption is 1000 times faster than asymmetric encryption

Digital signature

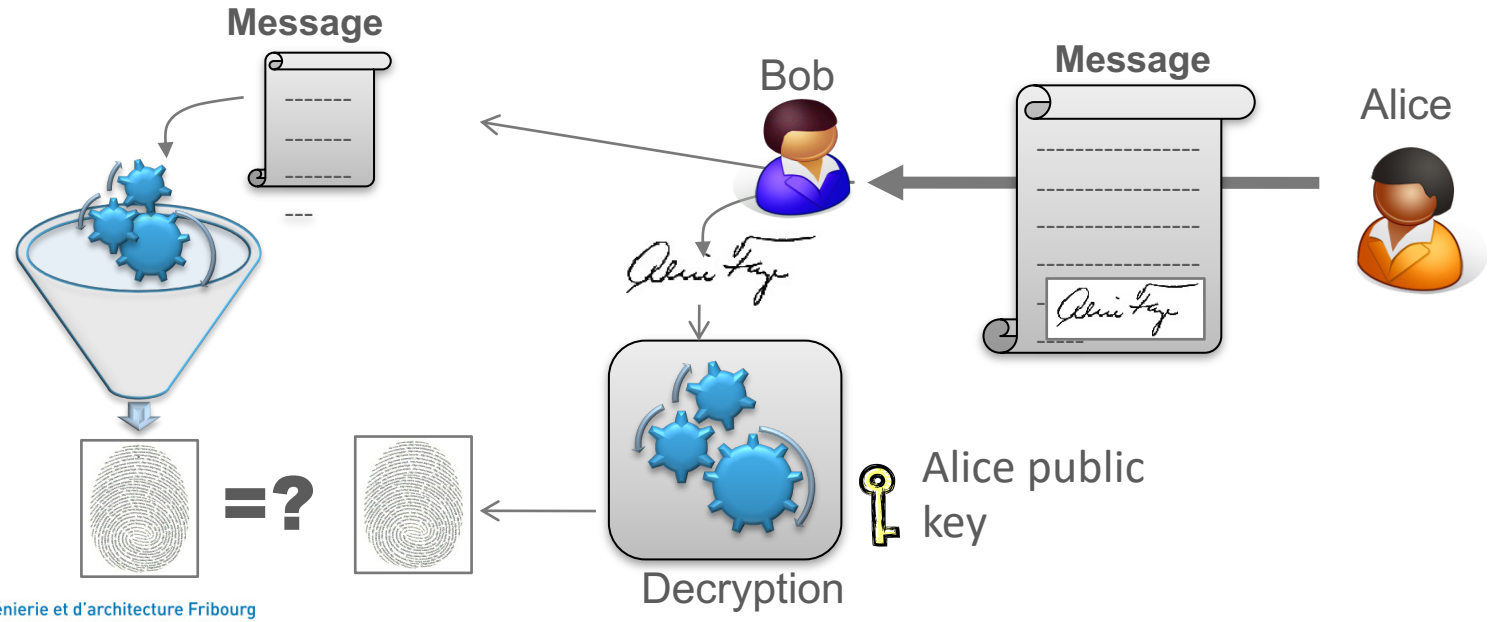
- Digital signature has two step:
  - A hash function computes the message fingerprint
  - The fingerprint is encrypted by an asymmetric cipher and a private key
- Goals of a digital signature are:
  - The message integrity
  - The origin-authentication of the message

Digital signature



Digital signature

- In order to check a signature:
  - Bob uses the Alice public key
  - Bob computes the message finperprint
  - Bob compares the two results.



Digital signature

```
openssl genrsa -out rsa_key.pem 2048       // compute private-public keys
openssl rsa -in rsa_key.pem –text          // show keys
openssl rsa -in rsa_key.pem -pubout -out rsa_key_pub.pem // get public key

// sign t.txt with public key
openssl dgst -sha256 -sign rsa_key.pem -out t.sign t.txt

// decrypt t.rsa with private key
openssl dgst –verify -sha256 rsa_key_pub.pem -signature t.sign t.txt
```

Trusted Platform Module [1], [2]

- A Trusted Platform Module, also known as a TPM, is a cryptographic coprocessor. It is not accelerator cryptographic coprocessor.
- TPM was conceived by a computer industry consortium called Trusted Computing Group (TCG) [3], and was standardized by International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) in 2009 as ISO/IEC 11889.
- It exists two versions:
  - Version 1.2: 2005, SHA-1, RSA
  - Version 2.0: 2014, SHA-256, RSA, ECC, AES

Example: Infineon TPM 96xx

TPM implementation [2]

There are different types of TPM 2.0 implementations (listed in order from most to least secure)

1. Discrete TPMs are dedicated chips that implement TPM functionalities in their own tamper resistant semiconductor package
2. Integrated TPMs are part of another chip. Intel: Platform Trust Technology (PTT), AMD: fTPM, ARM: TrustZone
3. Hypervisor TPMs (vTPMs) are virtual TPMs provided by and rely on hypervisors.
4. Software TPMs are software emulators of TPMs that run with no more protection than a regular program gets within an operating system.

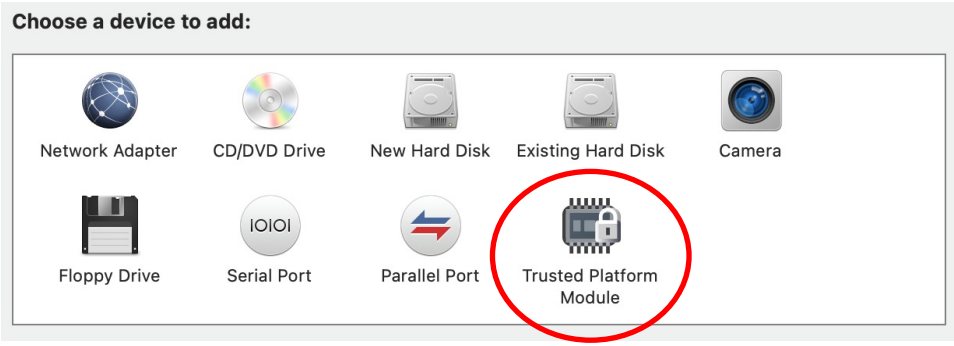
Discrete TPMs certified products [3]

[https://trustedcomputinggroup.org/membership/certification/tpm-certified-products/]

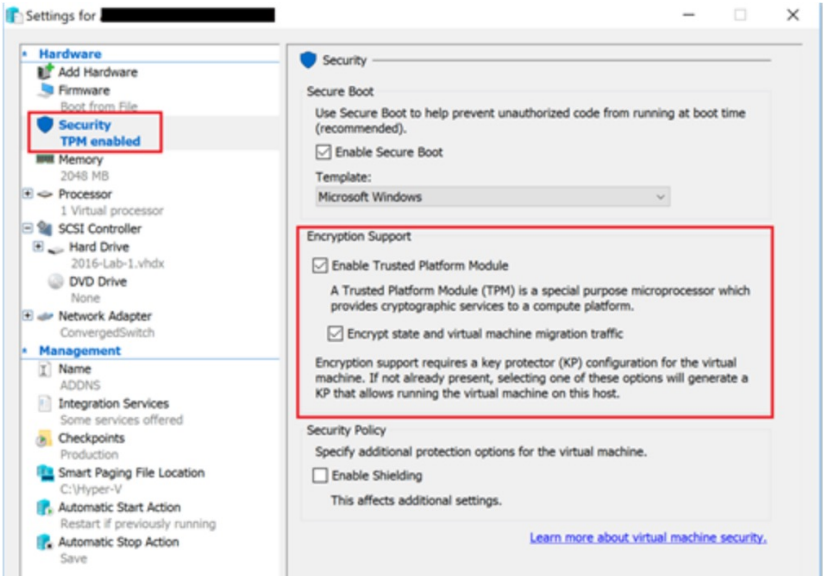
Company Name	Product Name	Product Revision	Specification	Details	Security Evaluation	Cert. Status	Cert. Complete Date
Infineon Technologies	TPM SLB9672	15.20, 15.21	Version 2.0 - Revision 1.38		Complete	Complete	2021.08.13
STMicroelectronics	TPM ST33GTPMA TPM ST33GTPMI	3.257, 6.257 3.256, 6.526	Version 2.0 - Revision 1.38		Complete Complete	Complete Complete	2020.11.30 2019.10.18
Nuvoton Technologies Corporation (NTC)	TPM NPCT75x	7.2.2.0 7.2.1.0	Version 2.0 - Revision 1.38		Complete Complete	Complete Complete	2020.07.10 2019.01.18
STMicroelectronics	TPM ST33TPHF2X	1.258, 2.272 1.256, 1.257, 2.256	Version 2.0 - Revision 1.38		Complete Complete	Complete Complete	2020.04.01 2019.10.18
STMicroelectronics	ST33TPxF2E	73.64, 73.65	Version 1.2 - Revision 116	01.02.49.40, 01.02.49.41	Complete	Complete	2019.11.07
STMicroelectronics	ST33TPxF2E	73.64, 73.65	Version 2.0 - Revision 1.38	00.49.00.40, 00.49.00.41	Complete	Complete	2019.11.07
STMicroelectronics	ST33TPxF20	74.64, 74.65	Version 2.0 - Revision 1.38	00.4A.00.40, 00.4A.00.41	Complete	Complete	2019.11.07
STMicroelectronics	ST33TPxF2E	73.20, 73.21	Version 1.2 - Revision 116	01.02.49.14, 01.02.49.15	Complete	Complete	2019.11.06
STMicroelectronics	ST33TPxF2E	73.20, 73.21	Version 2.0 - Revision 1.16	00.49.00.14, 00.49.00.15	Complete	Complete	2019.11.06

Hypervisor TPM

VMWare: Add device



Virtual Box

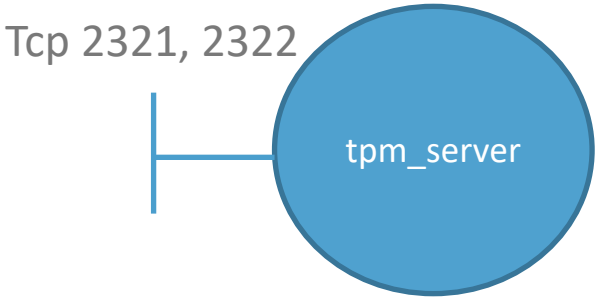


Software TPM

Software TPM (used for this course)

```
git clone https://github.com/stefanberger/swtpm.git
```

Other sotware TPM
https://sourceforge.net/projects/ibmswtpm2/



Install software TPM swtpm

```
git clone https://github.com/stefanberger/swtpm.git
cd swtpm
```

Install swtpm

see: https://github.com/stefanberger/swtpm/wiki

```
sudo dnf -y install libtasn1-devel expect socat python3-twisted fuse-devel glib2-devel
gnutls-devel gnutls-utils gnutls json-glib-devel

sudo dnf install libtpms-devel
sudo dnf install libseccomp-devel
./autogen.sh --with-openssl --prefix=/usr
make -j4
make -j4 check
sudo make install
export TPM2TOOLS_TCTI="swtpm:port=2321"
```

Start swtpm

```
swtpm socket --tpmstate dir=/home/schuler/work/tpm/swtpm2  --tpm2 --server type=tcp,port=2321 --ctrl
type=tcp,port=2322 --flags not-need-init,startup-clear
```

Or with log file

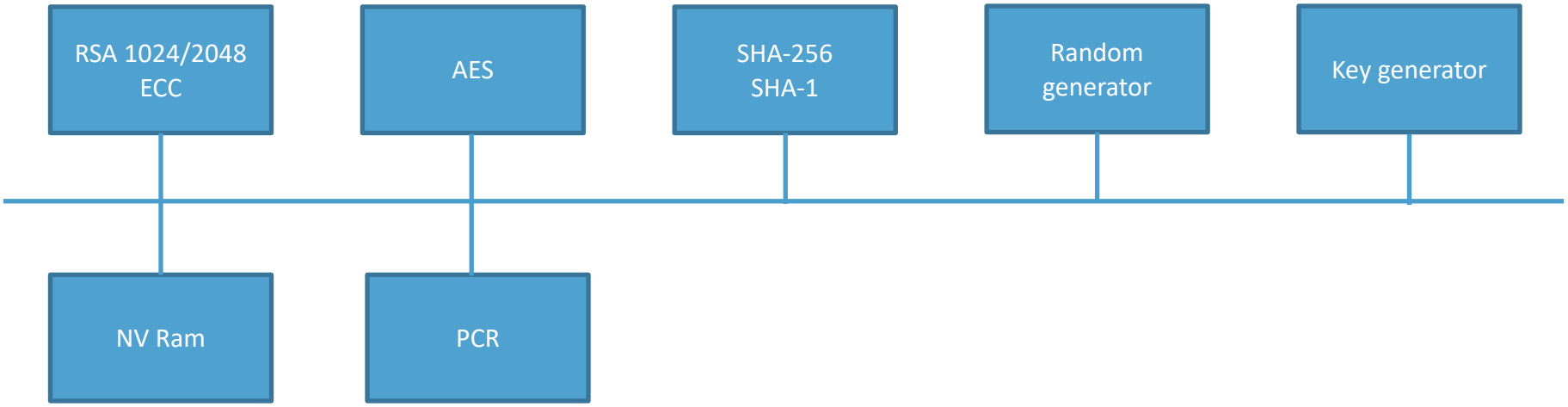
```
swtpm socket --tpmstate dir=/home/schuler/work/tpm/swtpm2  --tpm2 --server type=tcp,port=2321 --ctrl
type=tcp,port=2322 --log file=/home/schuler/work/tpm/swtpm2/log,level=2 --flags not-need-init,startup-
clear
```

TPM software libraries [2]

Summary of the existing open-source TPM libraries

TPM Libraries	API	TPM 2.0	TPM 1.2	Attestation server or example	Microsoft Windows	Linux	Bare metal
tpm2-tss[119]	SAPI, ESAPI and FAPI from the TCG specification	Yes	No	No, but there is a separate project*	Yes	Yes	Maybe**
ibmtss[120][121]	1:1 mapping to TPM commands + rich API (mild layer on top)	Yes	Partial	Yes, "IBM ACS"[122][123]	Yes	Yes	No
go-tpm[124]	1:1 mapping to TPM commands + rich API (mild layer on top)	Yes	Partial	Yes, "Go-attestation"[125]	Yes	Yes	No
wolfTPM[126]	1:1 mapping to TPM commands + rich API (wrappers)	Yes	No	Yes, examples are inside the library	Yes	Yes	Yes
TSS.MSR[127]	1:1 mapping to TPM commands + rich API (wrappers)	Yes	No	Yes, examples are inside the library	Yes	Yes***	No

TPM Internal architecture



- RSA 1024/2048, ECC: Asymmetric algorithms, encrypt-decrypt, sign
- AES: Symmetric algorithm, encrypt-decrypt, sign
- SHA-256, SHA-1: hash function
- Random generator: create random value
- Key generator: Create key for asymmetric algo
- NV Ram: Store different objects (keys, data, …) in NV Ram
- PCR (Platform Configuration Registers) stores hash values of different parts: code, files, partitions, …

Primary key hierarchies [5]

The TPM stores keys on one of four hierarchies:

- Endorsement hierarchy:

  The endorsement hierarchy is reserved for objects created and certified by the TPM manufacturer. The endorsement seed (eseed) is randomly generated at manufacturing time and never changes during the lifetime of the device. The primary endorsement key is certified by the TPM manufacturer, and because its seed never changes, it can be used to identify the device

- Platform hierarchy:

  The platform hierarchy is reserved for objects created and certified by the OEM that builds the host platform. The platform seed (pseed) is randomly generated at manufacturing time, but can be changed by the OEM by calling `tpm2_changepps`

- Owner hierarchy, also known as storage hierarchy:

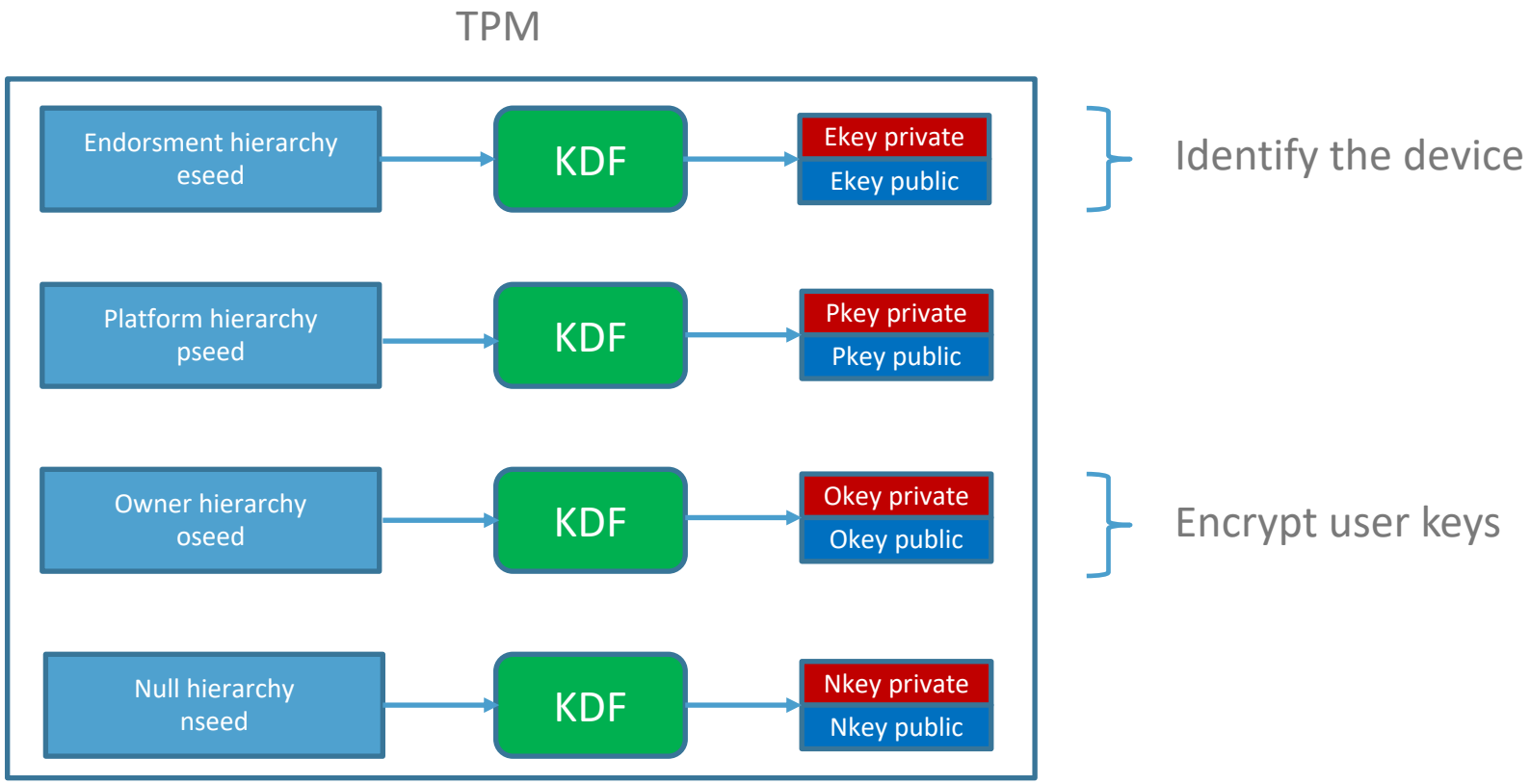
  The owner hierarchy is reserved for us - the primary users of the TPM. When a user takes a TPM, the owner hierarchy can be erased by the `tpm2_clear` command. `tpm2_clear` generates a new owner seed (oseed)

- Null hierarchy:

  The null hierarchy is reserved for ephemeral keys. The null seed is re-generated each time the host reboots

Primary key hierarchies [5]

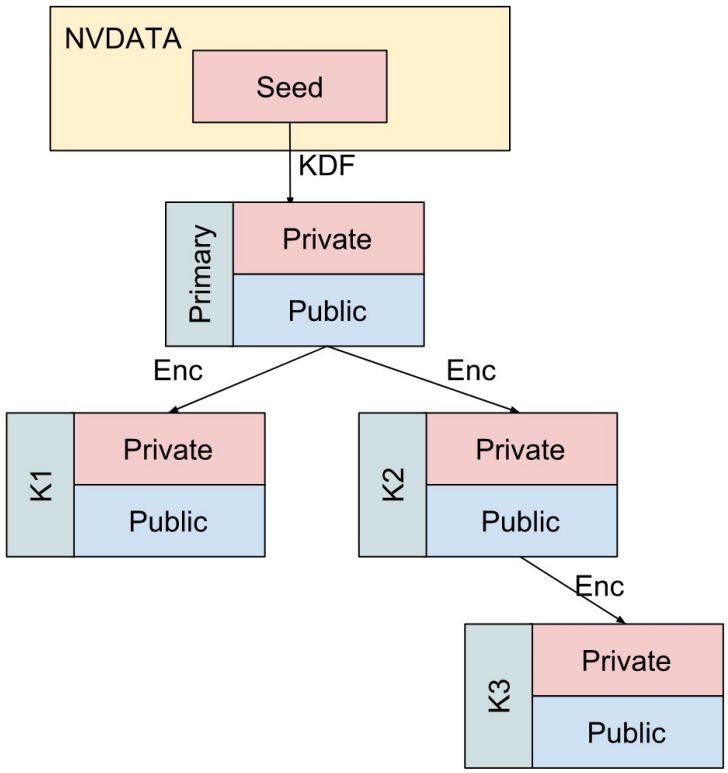
Primary keys are derived from the primary seeds using a deterministic key derivation function (KDF).



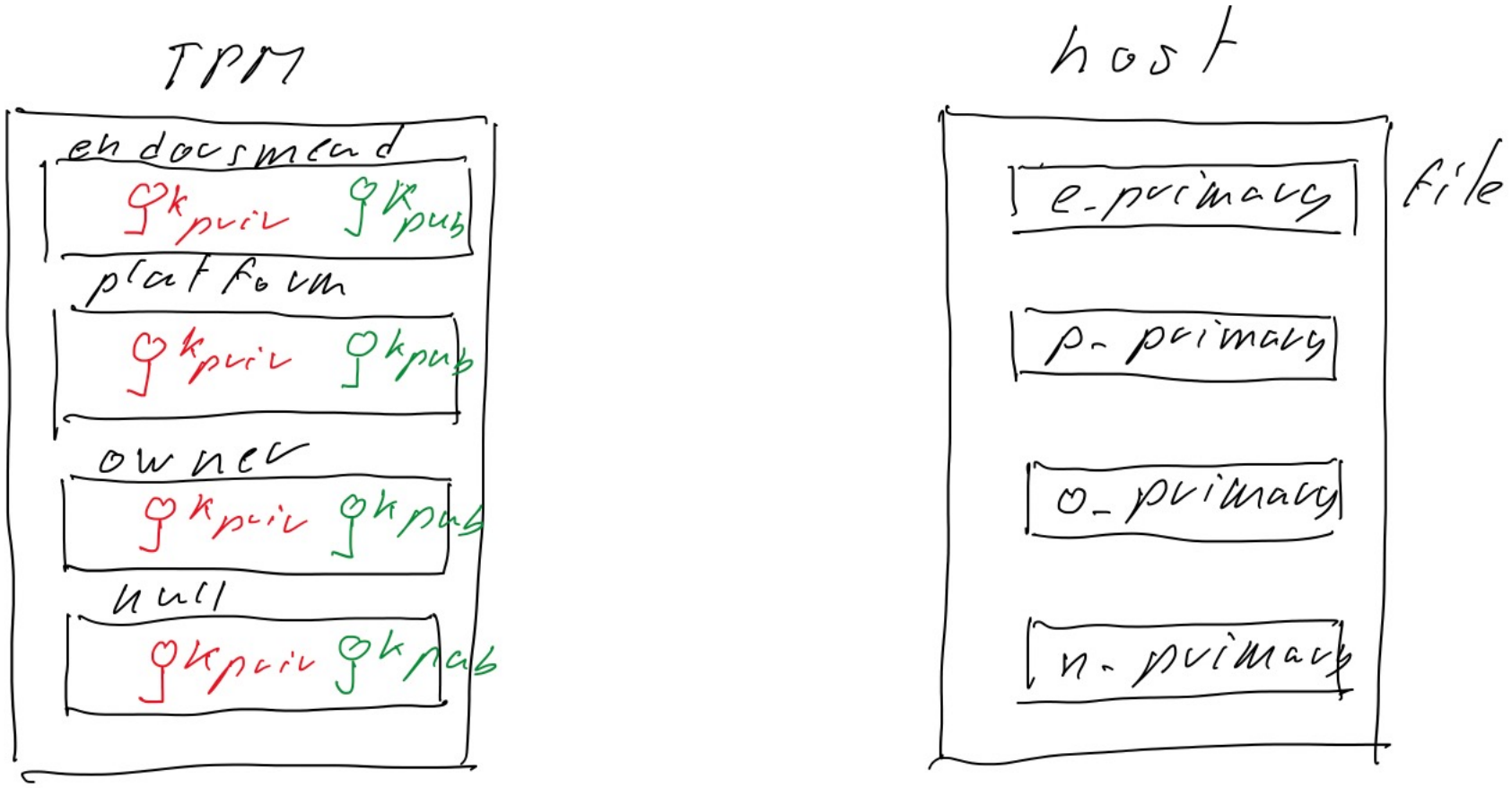
Cryptographic keys [5]

TPM generates strong, secure cryptographic keys.

- Strong in the sense that the key is derived from “true random source” and large key space
- Secure in the sense that the private key never leaves the TPM. When a key leaves the TPM - in order to be loaded and used later - it is wrapped (encrypted) by its parent key.



Primary keys [5]



Primary keys [5]

- Primary keys are derived from the primary seeds using a deterministic key derivation function (KDF). The template defines the characteristic of the primary key.
- The Private and Public keys are inside the TPM. The `primary.ctx` is a file saved outside the TPM. The `primary.ctx` contains reference to the primary keys

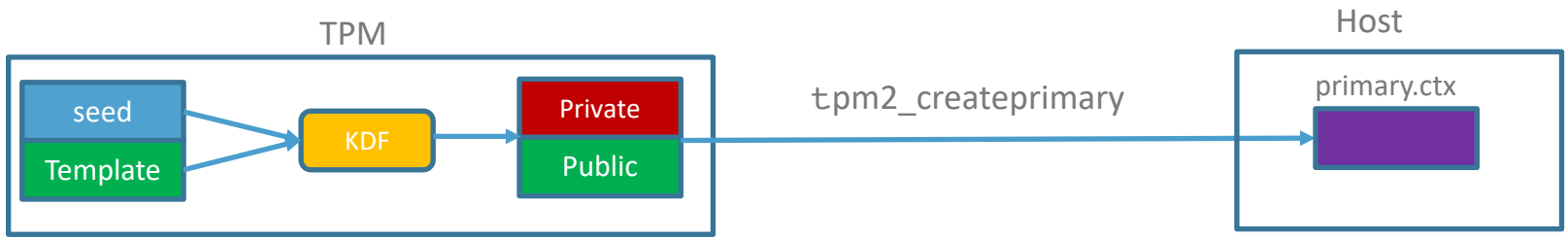
Create RSA endorsement key: `tpm2_createprimary -C e -G rsa2048 -c e_primary.ctx`

Create RSA platform key: `tpm2_createprimary -C p -G rsa2048 -c p_primary.ctx`

Create RSA owner key: `tpm2_createprimary -C o -G rsa2048 -c o_primary.ctx`

Create RSA null key: `tpm2_createprimary -C n -G rsa2048 -c n_primary.ctx`

```
man tpm2_createprimary
```



Capture with wireshark

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

It is possible to find all details in the file: TPM-Rev-2.0-Part-3-Commands-01.38 [4]

```
    35 0.002498156  127.0.0.1          127.0.0.1          TCP       74 40682 → 2321 [SYN] Seq=0 Win=65495
    36 0.002508801  127.0.0.1          127.0.0.1          TCP       74 2321 → 40682 [SYN, ACK] Seq=0 Ack=1
    37 0.002516872  127.0.0.1          127.0.0.1          TCP       66 40682 → 2321 [ACK] Seq=1 Ack=1 Win=
    38 0.002562606  127.0.0.1          127.0.0.1          TPM      197 40682 → 2321,  [TPM Request], Comma
    39 0.002574201  127.0.0.1          127.0.0.1          TCP       66 2321 → 40682 [ACK] Seq=1 Ack=132 Wi
```

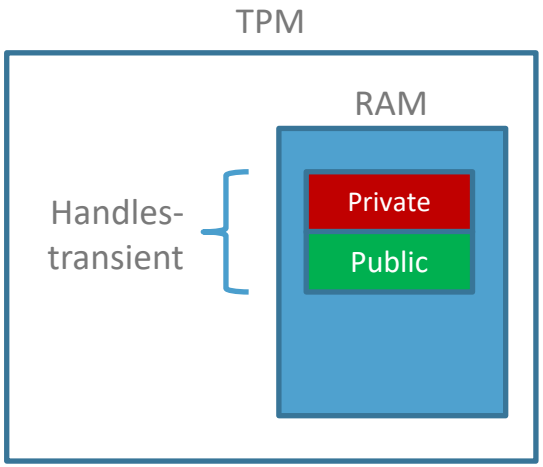
```
▸ Frame 38: 197 bytes on wire (1576 bits), 197 bytes captured (1576 bits) on interface lo, id 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 40682, Dst Port: 2321, Seq: 1, Ack: 1, Len: 131
▾ TPM2.0 Protocol
  ▾ TPM2.0 Header, TPM2_CC_CreatePrimary
      Tag: Command with authorization Sessions (0x8002)
      Command size: 131
      Command Code: TPM2_CC_CreatePrimary (0x00000131)
  ▾ Handle Area
      TPMI_RH_HIERARCHY: TPM2_RH_OWNER (0x40000001)
  ▾ Authorization Area
      AUTHAREA SIZE: 73
      TPMI_SH_AUTH_SESSION: Unknown (0x02000000)
      AUTH NONCE SIZE: 32
      AUTH NONCE: 469201b1b750da1982aa1f067d174ac0e58d13b345a96d79d457847936b78b8f
    ▸ Session attributes
      SESSION AUTH SIZE: 32
      SESSION AUTH: 557927f11548e1bad570c04a0aedb18f0e56ddd9a094ce049f1e2a97e9933f2d
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ·····E·
0010  00 b7 87 d8 40 00 40 06  b4 66 7f 00 00 01 7f 00   ····@·@· ·f·····
0020  00 01 9e ea 09 11 87 66  a8 c0 1f 1a 93 74 80 18   ·····f ·····t··
0030  02 00 fe ab 00 00 01 01  08 0a 61 42 ab 83 61 42   ·····aB ··aB
0040  ab 83 80 02 00 00 00 83  00 00 01 31 40 00 00 01   ·····1@···
0050  00 00 00 49 02 00 00 00  00 20 46 92 01 b1 b7 50   ···I···· · F····P
0060  da 19 82 aa 1f 06 7d 17  4a c0 e5 8d 13 b3 45 a9   ·····} · J·····E·
0070  6d 79 d4 57 84 79 36 b7  8b 8f 01 00 20 55 79 27   my·W·y6· ···· Uy'
0080  f1 15 48 e1 ba d5 70 c0  4a 0a ed b1 8f 0e 56 dd   ··H···p· J·····V·
0090  d9 a0 94 ce 04 9f 1e 2a  97 e9 93 3f 2d 00 04 00   ·····* ···?-···
00a0  00 00 00 00 1a 00 01 00  0b 00 03 00 72 00 00 00   ····· ····r···
00b0  06 00 80 00 43 00 10 08  00 00 00 00 00 00 00 00   ····C··· ·····
00c0  00 00 00 00 00                                    ·····
```

Create primary keys [5]

- After `tpm2_createprimary` command, public-private keys are stored in the TPM key cache area.
- Key cache is the `handles-transient` area of the TPM, `tpm2_getcap handles-transient` command shows the index of the public-private keys
- Example with owner hierarchy:

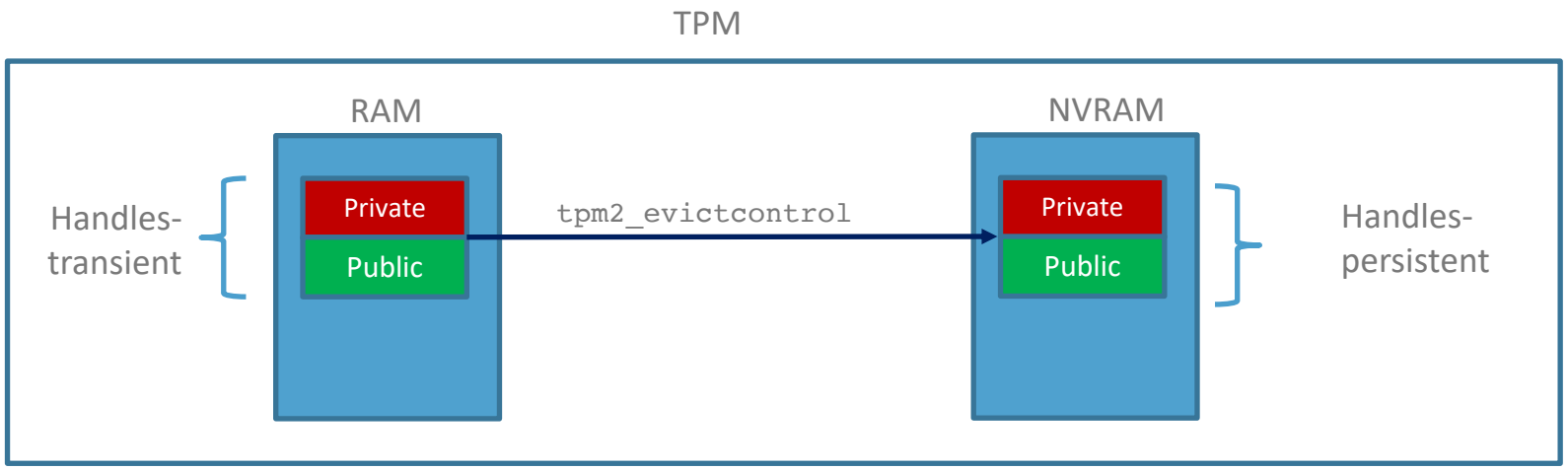
```
tpm2_createprimary -C o -G rsa2048 -c o_primary.ctx
tpm2_getcap handles-transient
- 0x80000000
```



Save primary keys to NVRAM [5]

- It is possible to save these keys to NVRAM with the `tpm2_evictcontrol` command
- Keys are stored in the handles-persistent part of the TPM

```
tpm2_evictcontrol –c o_primary.ctx or
tpm2_evictcontrol –c 0x80000000

tpm2_getcap handles-persistent
- 0x81000000
```



Delete keys from NVRAM and RAM [5]

- `tpm2_evictcontrol` command can store keys to NVRAM and erase keys from NVRAM

```
tpm2_evictcontrol –c o_primary.ctx or
tpm2_evictcontrol –c 0x80000000
```

} Store keys to NVRAM

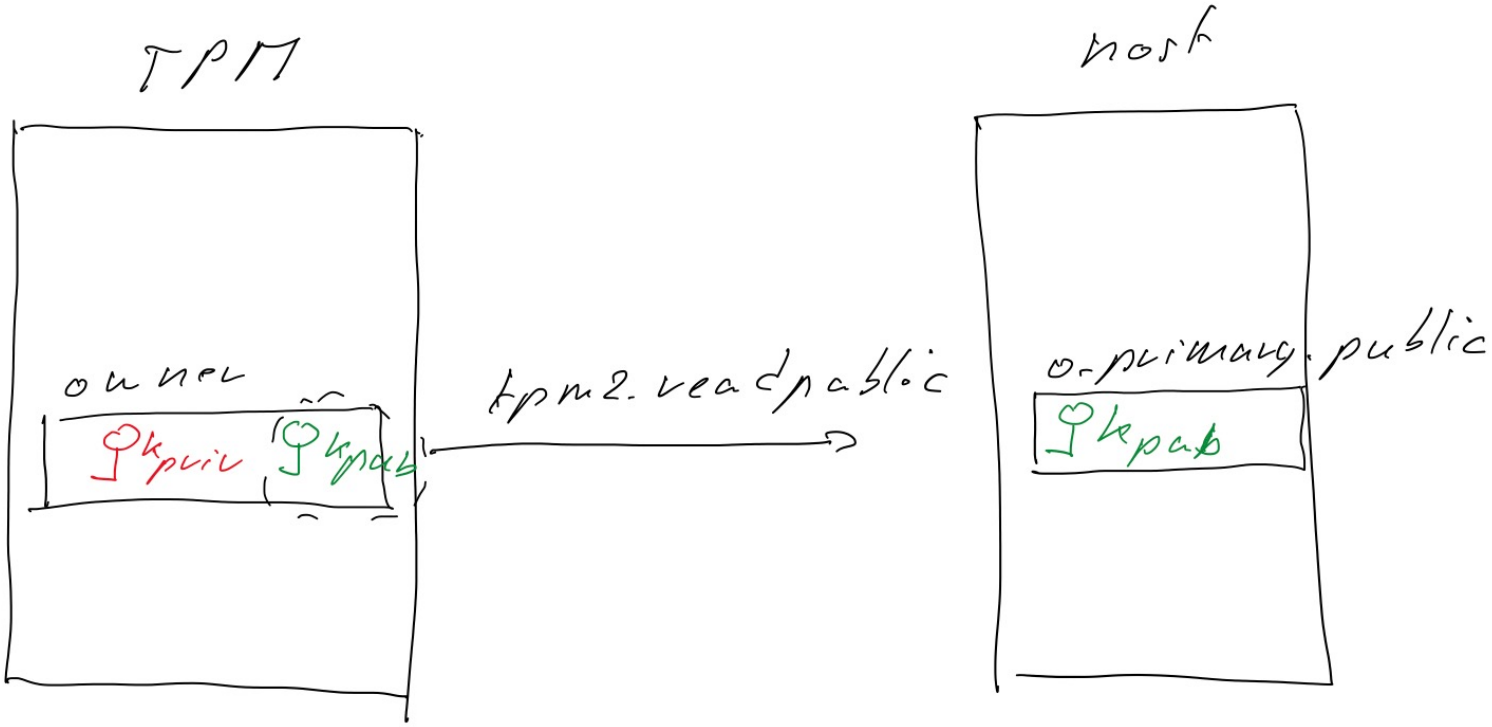
```
tpm2_getcap handles-persistent
-  0x81000000

tpm2_evictcontrol –c 0x81000000
```

} Delete keys from NVRAM

- `tpm2_flushcontext` command deletes keys from RAM

```
tpm2_getcap handles-transient
-  0x80000000
-  0x80000001

tpm2_flushcontext 0x80000000
tpm2_flushcontext -t
```

Get public key from TPM



Get public key from TPM

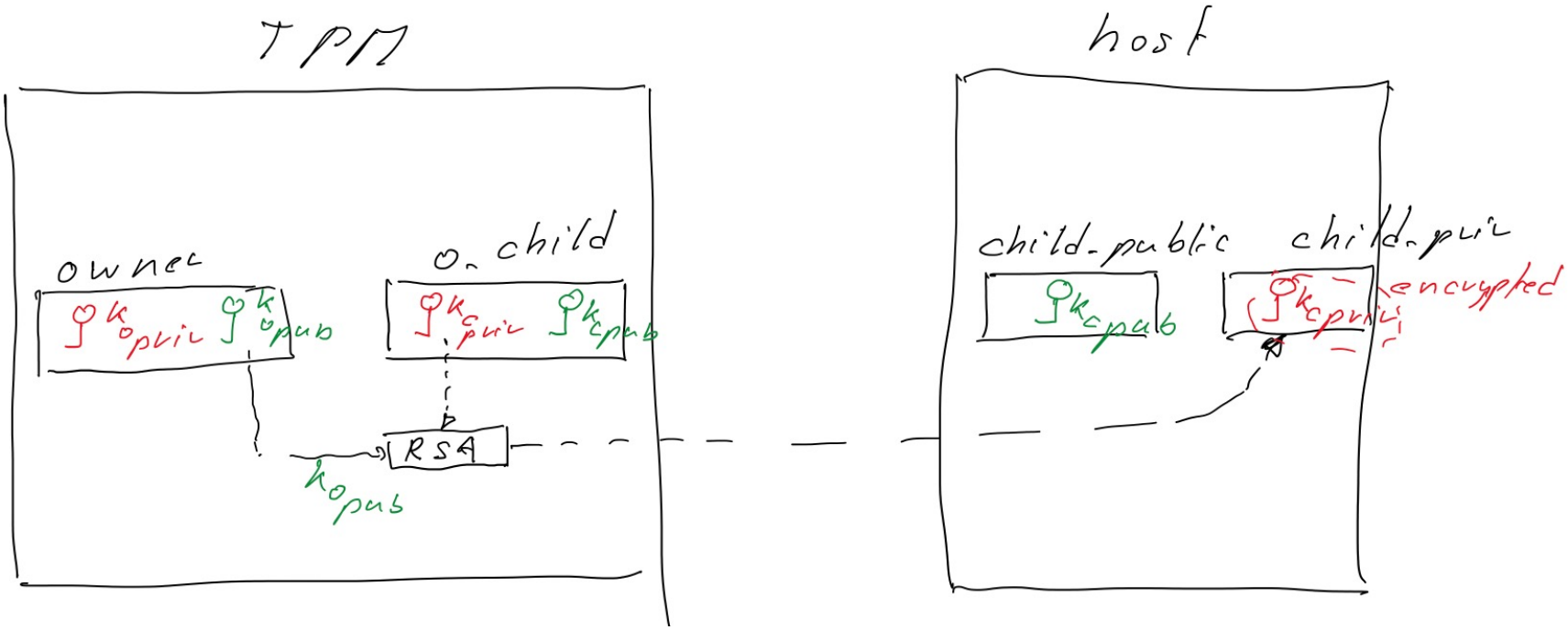
Tpm2_readpublic command get the public key from the TPM

```
tpm2_createprimary -C o -G rsa2048 -c o_primary.ctx
tpm2_getcap handles-transient
- 0x80000000

tpm2_readpublic -c o_primary.ctx --format PEM -o o_primary.public
Or
tpm2_readpublic -c 0x80000000 --format PEM -o o_primary.public
```

openssl command shows the public key

```
openssl rsa -pubin -in o_primary.public -text
```

Create Child asymmetric keys [5]

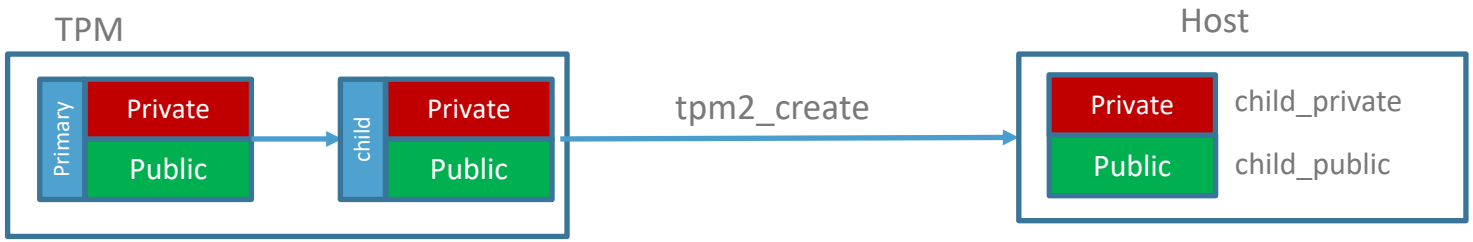


Create Child asymmetric keys [5]

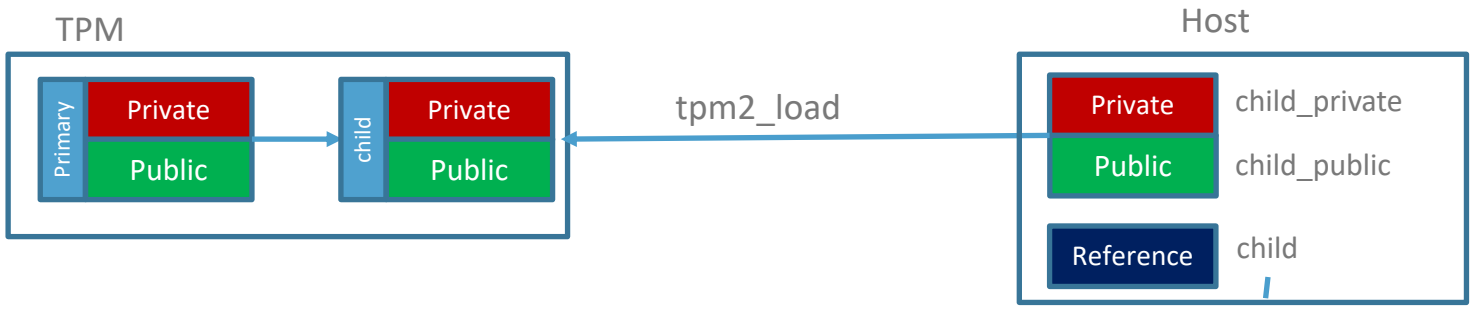
TPM2_Create command creates keys (asymmetric or symmetric).

- `child_private` file contains the private asymmetric key or symmetric keys. These keys are encrypted by the parent keys.
- `child_public` file contains public asymmetric keys and different references

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

tpm2_create -C primary -G rsa2048 -u child_public –r child_private
```



Load Child asymmetric keys [5]

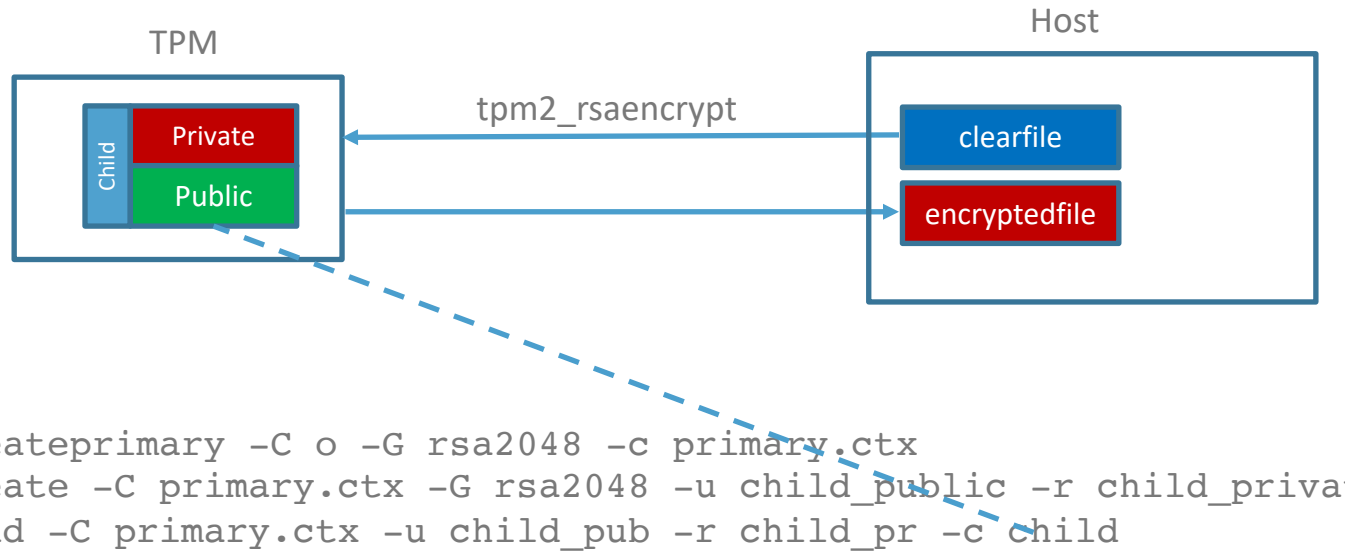


These child keys can be loaded to the TPM.

The loaded keys are in the handles-transient area (RAM). child file is a reference to the child keys

```
tpm2_createprimary -C o -G rsa2048 -c primary
tpm2_create -C primary -G rsa2048 -u child_public –r child_private

tpm2_load -C primary -u child_public -r child_private -c child

tpm2_getcap handles-transient
  - 0x80000000
```

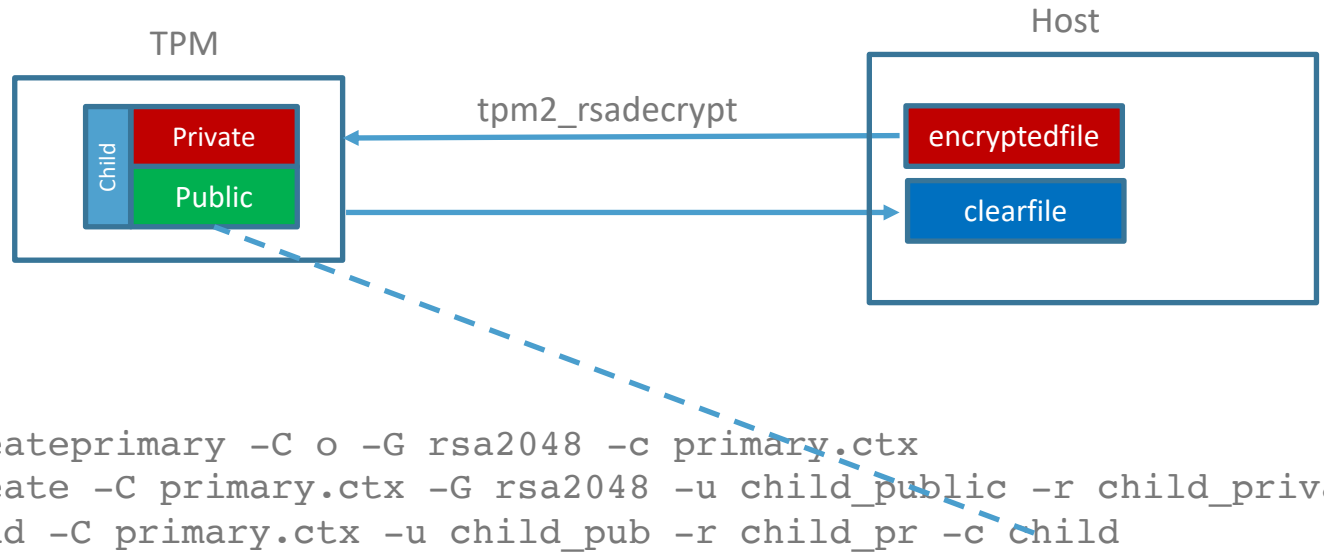
RSA- Encrypt with Child keys [5]




```
tpm2_createprimary -C o -G rsa2048 -c primary.ctx
tpm2_create -C primary.ctx -G rsa2048 -u child_public -r child_private
tpm2_load -C primary.ctx -u child_pub -r child_pr -c child


// encrypt with the public key
tpm2_rsaencrypt -c child -s rsaes clearfile –o encryptedfile

   -s : RSA padding: -s null: no padding, -s rsaes: pkcsv1.5 padding, -s oaep:
   oaep padding
```

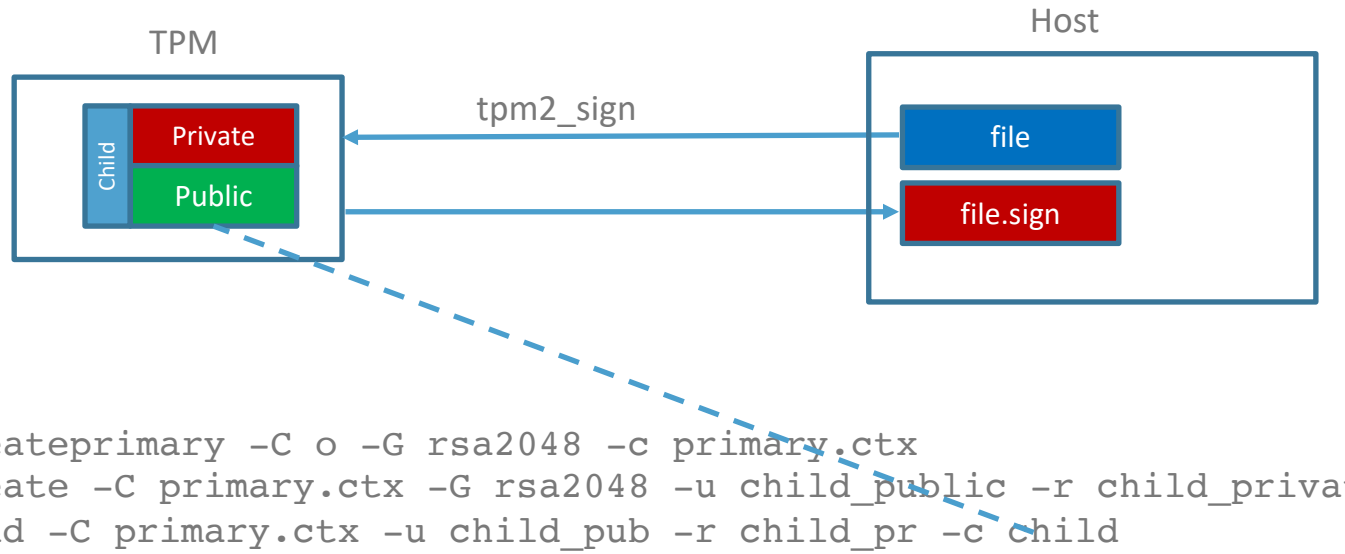
Remark: The clearfile max length = the length of the key less the padding. Example: key length=2048 bits = 256bytes
Padding null: max length= 256bytes, rsaes: max length= 256 -11=245

RSA- Decrypt with Child keys [5]




```
tpm2_createprimary -C o -G rsa2048 -c primary.ctx
tpm2_create -C primary.ctx -G rsa2048 -u child_public -r child_private
tpm2_load -C primary.ctx -u child_pub -r child_pr -c child


// decrypt with the private key
tpm2_rsadecrypt -c child -s rsaes encryptedfile -o clearfile
```

RSA- Sign with Child keys [5]

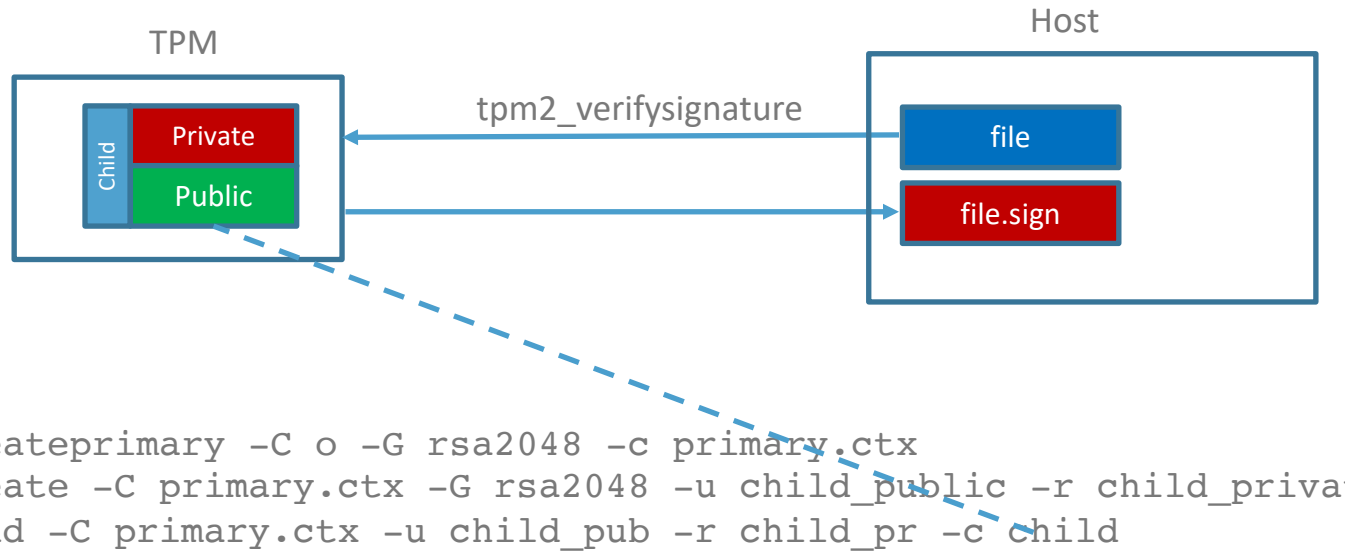



```
tpm2_createprimary -C o -G rsa2048 -c primary.ctx
tpm2_create -C primary.ctx -G rsa2048 -u child_public -r child_private
tpm2_load -C primary.ctx -u child_pub -r child_pr -c child
```

```
tpm2_sign -c child -g sha256 -o file.sign file //sign with the private key
```

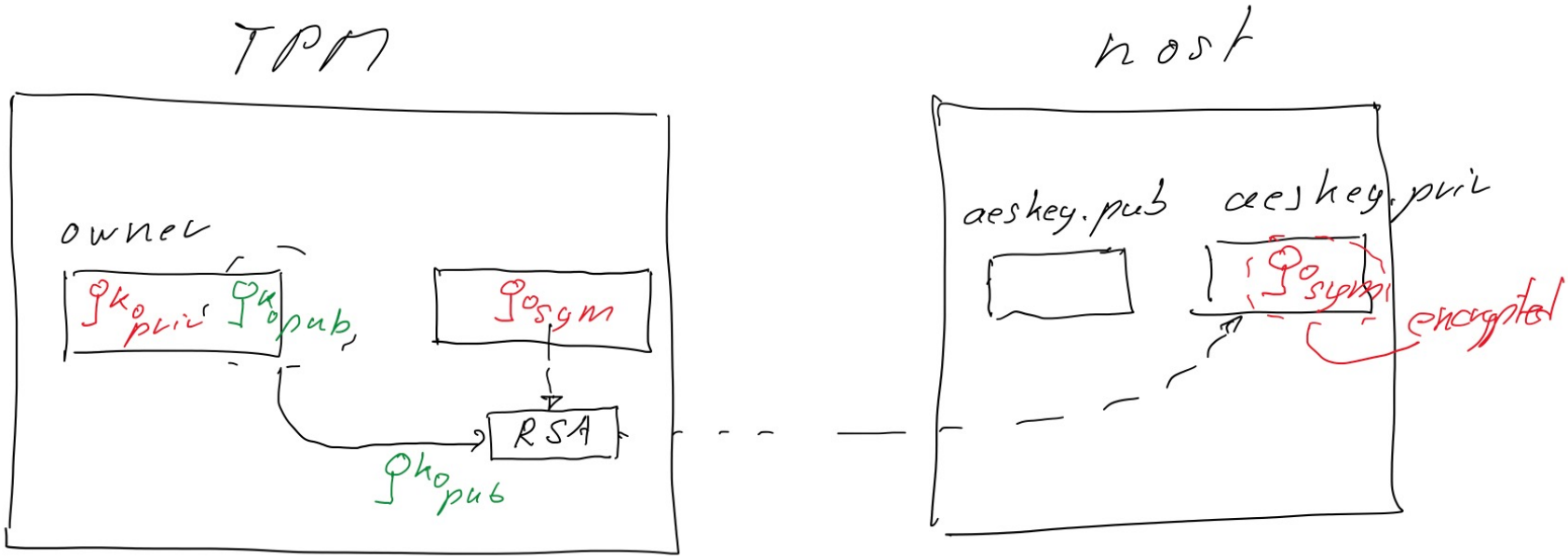
Remark: The message hash is computed by the TPM (be careful if the message is big)

RSA- Verify signature with Child keys [5]




```
tpm2_createprimary -C o -G rsa2048 -c primary.ctx
tpm2_create -C primary.ctx -G rsa2048 -u child_public -r child_private
tpm2_load -C primary.ctx -u child_pub -r child_pr -c child


// Verify with the private key
tpm2_verifysignature -c child -g sha256 -s file.sign -m file
```

Symmetric encryption-decryption



Symmetric encryption-decryption

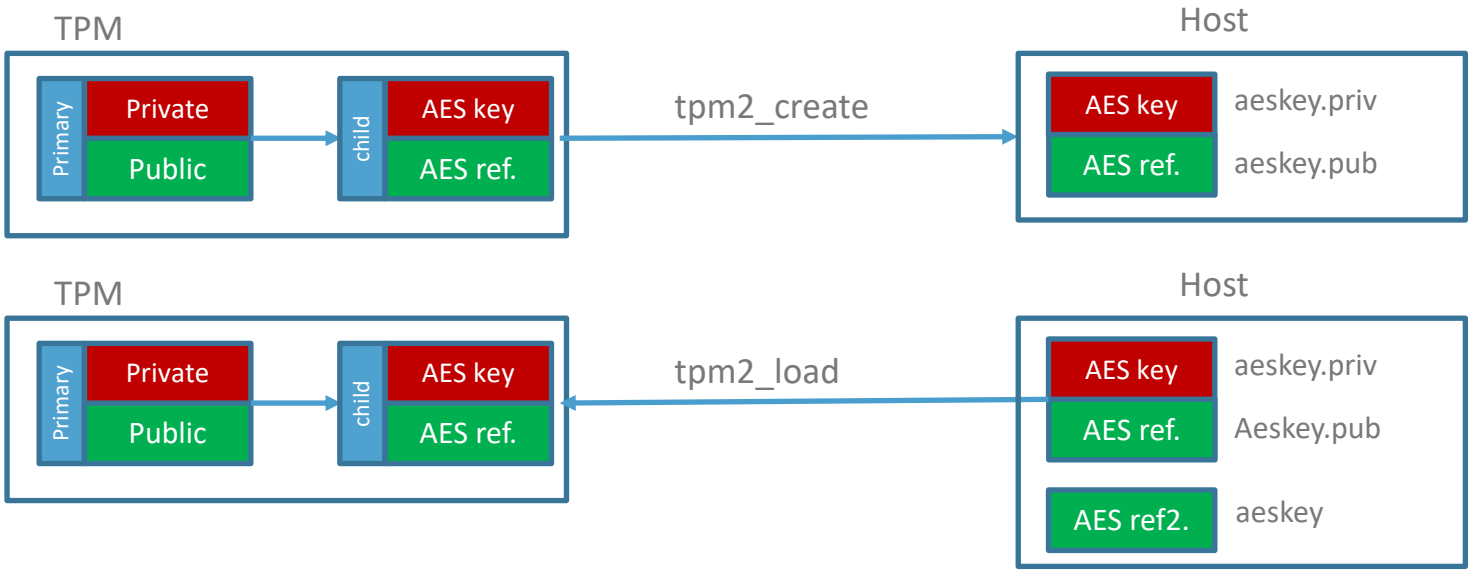
See this slide and the next

`tpm2_create` command creates an AES symmetric key. This key is encrypted by primary public key, and saved to aeskey.priv.

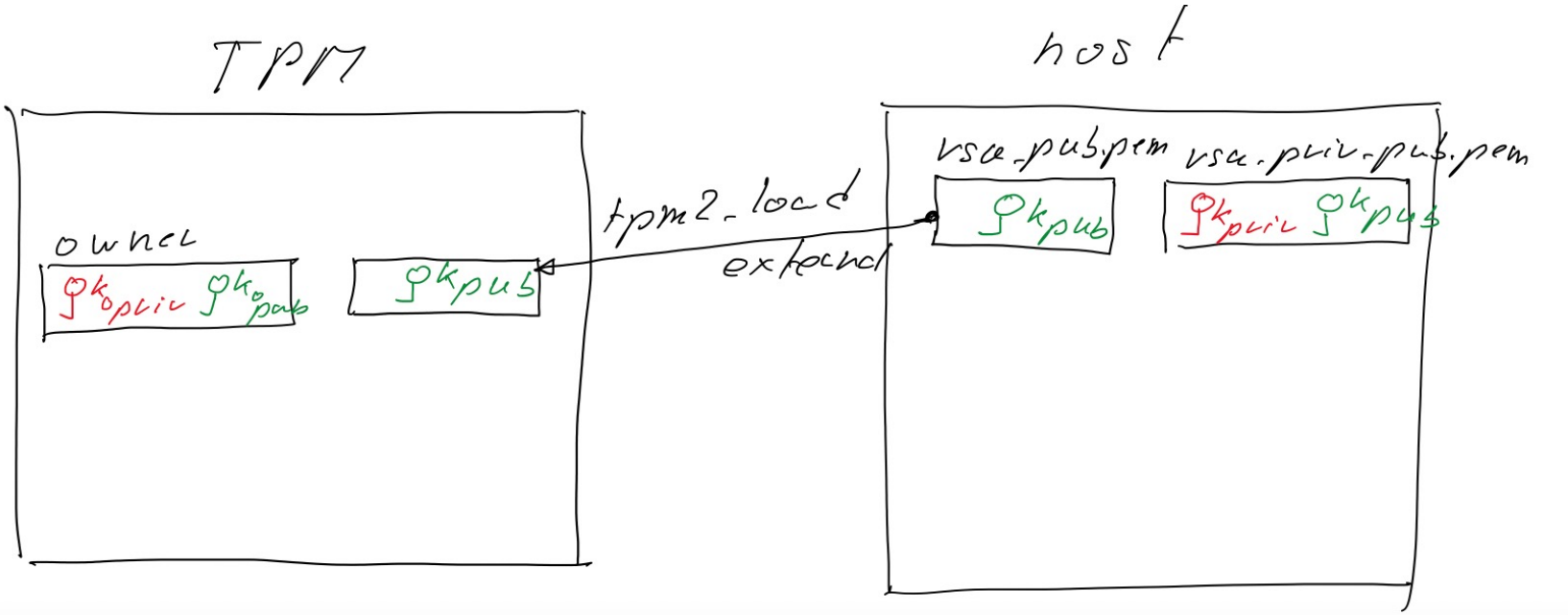
- aeskey.pub contains some references to aes key
- Aes key is loaded to the TPM with tpm2_load, aeskey contains other references to aes key
- In order to encrypt-decrypt, it is necessary to have a Initial Vector with 16 bytes

Symmetric encryption-decryption

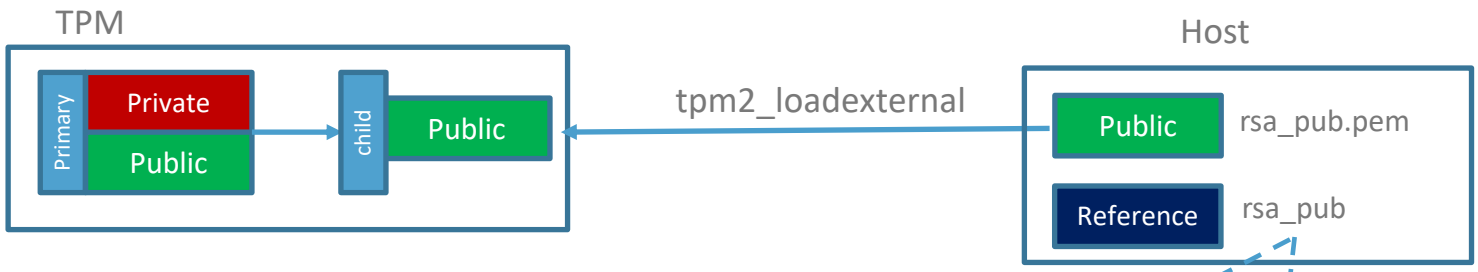
```
tpm2_createprimary -C o –G rsa2048 -c primary
tpm2_create -C primary –G aes128cbc -u aeskey.pub -r aeskey.priv
tpm2_load -C primary -u aeskey.pub -r aeskey.priv  -c aeskey

echo -n 234sdfweaw34rft6 > IV      // 16 bytes

tpm2_encryptdecrypt -c aeskey -e --iv=iv --pad -o encryptedfile clearfile
tpm2_encryptdecrypt -c aeskey -d --iv=iv --pad -o clearfile encryptedfile
```



Load external public key



Load external public key



It is possible to import external public key to the TPM.
Generate rsa public key with openssl (rsa_priv_pub.pem contains private and public keys, rsa_pub.pem contains only public key)

```
openssl genrsa -out rsa_priv_pub.pem 2048
Openssl rsa –in rsa_priv_pub.pem –out rsa_pub.pem -pubout
```

`tpm2_loadexternal` command loads rsa_pub.pem into the TPM the owner hierarchy. The key is in the handles-transient area.

The `rsa_pub` file contains references to the loaded key (used for the next commands)

```
tpm2_loadexternal -C o -G rsa -u rsa_pub.pem -c rsa_pub
```

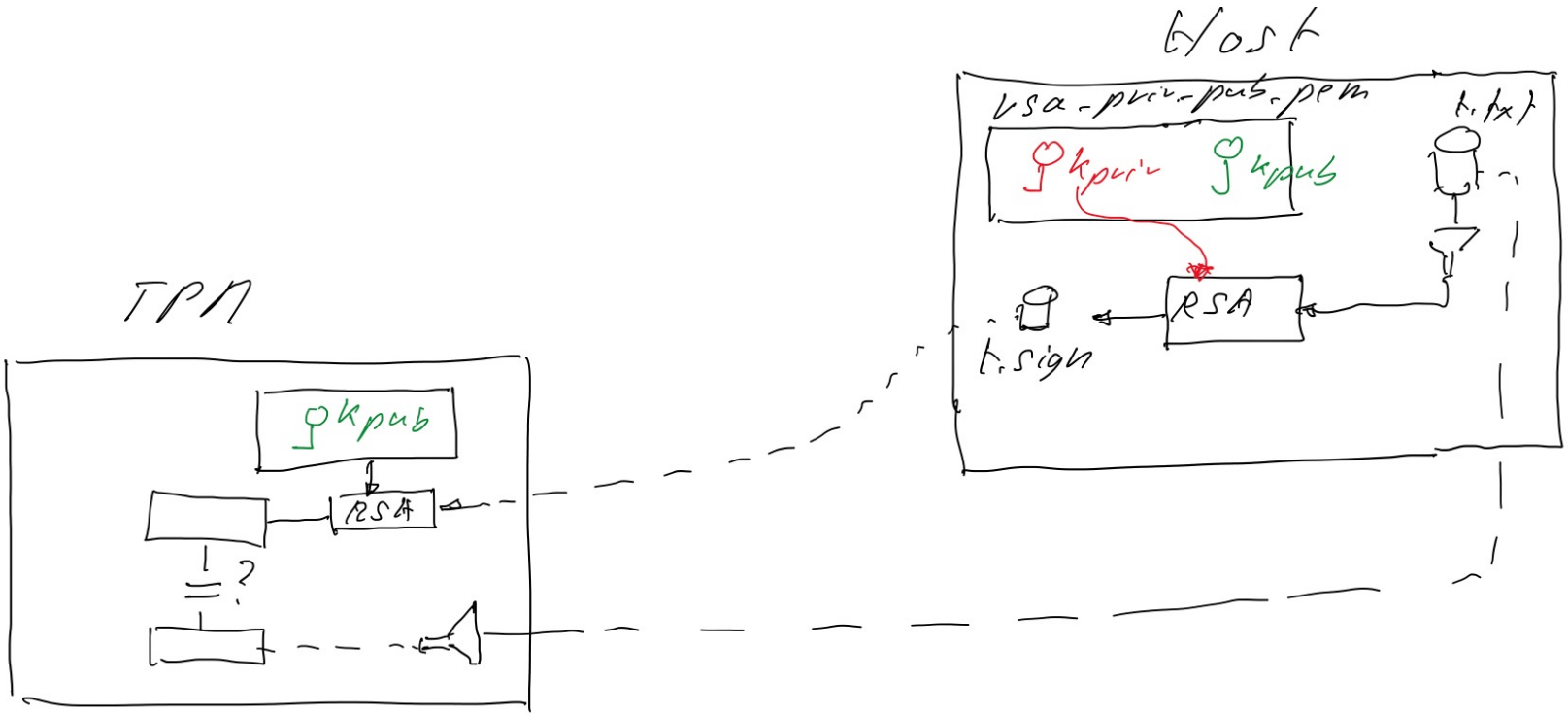
Verify signature with a loaded external public key

On the Host, sign a file with the private key

```
openssl dgst -sha256 -sign rsa_priv_pub.pem -out t.sign t.txt
```

On the TPM, verify the signture with the pubic key

```
tpm2_loadexternal -C o -G rsa -u rsa_pub.pem -c rsa_pub
tpm2_verifysignature -c rsa_pub -g sha256 -f rsassa -s t.sign -m t.txt
```



Load external private-public keys

It is possible to import external private-public keys to the TPM. Private key can be imported only on the NULL hierarchy

```
openssl genrsa -out rsa_priv_pub.pem 2048
```

`tpm2_loadexternal` command loads rsa_pub.pem into the TPM the owner hierarchy. The key is in the handles-transient area.

The `rsa_priv_pub` file contains references to the loaded key (used for the next commands)

```
tpm2_loadexternal -C n -G rsa -r rsa_priv_pub.pem -c rsa_priv_pub
```

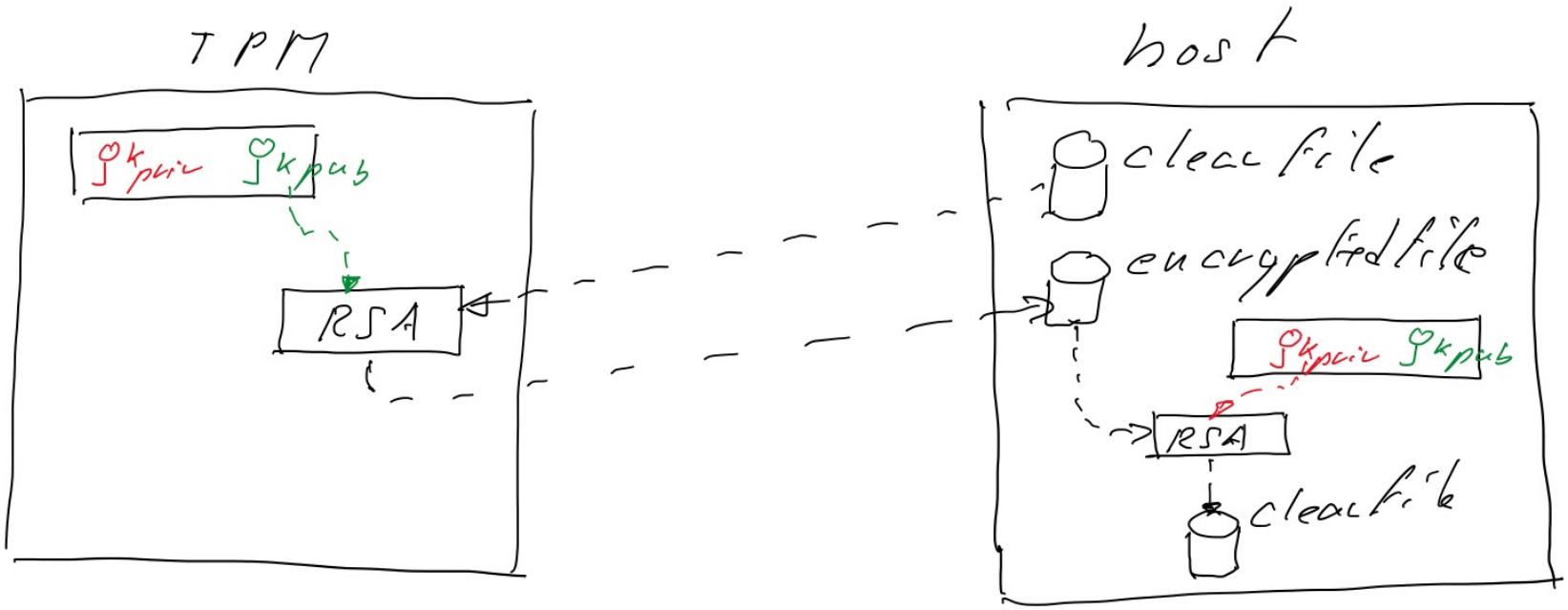
Encrypt on TPM, decrypt on host

On the TPM encrypt with the the public key:

```
tpm2_rsaencrypt -c rsa_priv_pub -s rsaes clearfile -o encryptedfile
```

On the host decrypt with the private key

```
openssl rsautl -decrypt -inkey rsa_priv_pub.pem -in encryptedfile -out clearfile
```

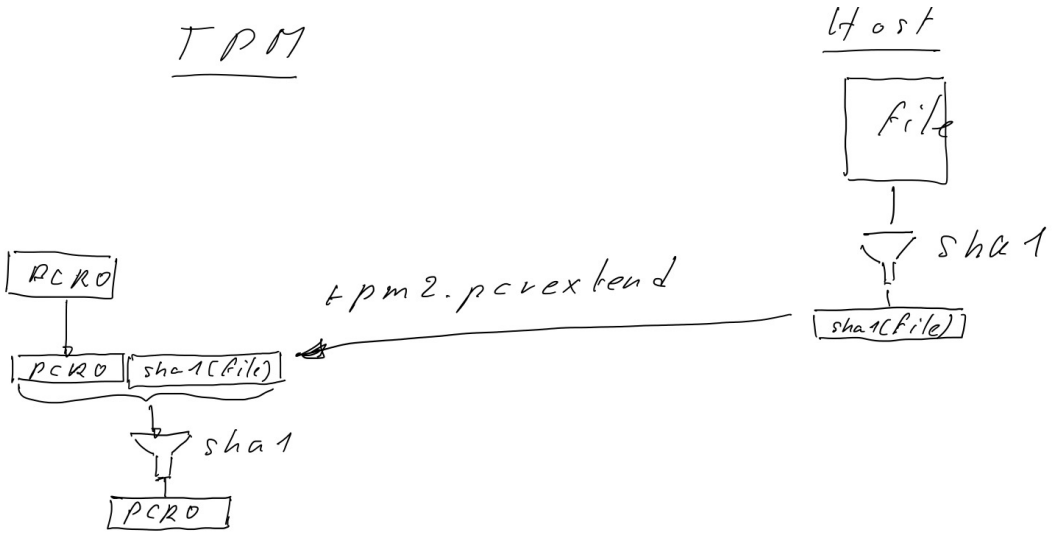


Platform Configuration Registers [6, chap 12]

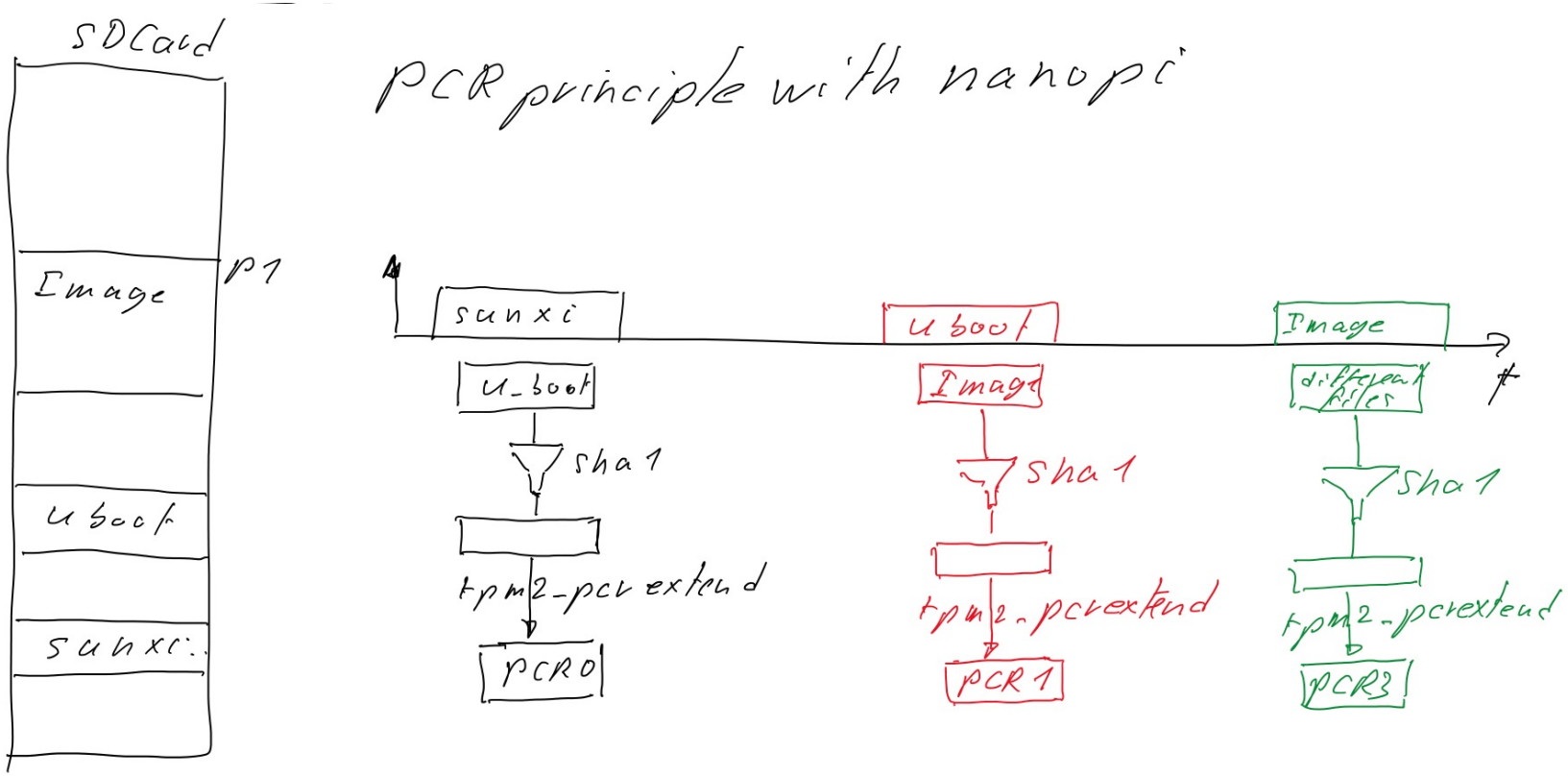
- Platform Configuration Registers (PCRs) are one of the essential features of a TPM. Their prime use case is to provide a method to cryptographically record (measure) software state: both the software running on a platform and configuration data used by that software.

- The PCR update calculation, called an extend, is a one-way hash so that measurements can't be removed.

```
PCR new value = Hash (PCR old value || data to extend)

Example: sha1sum file
          8c8393ac8939430753d7cb568e2f2237bc62d683  t.txt
tpm2_pcrextend 0:sha1=8c8393ac8939430753d7cb568e2f2237bc62d683
```

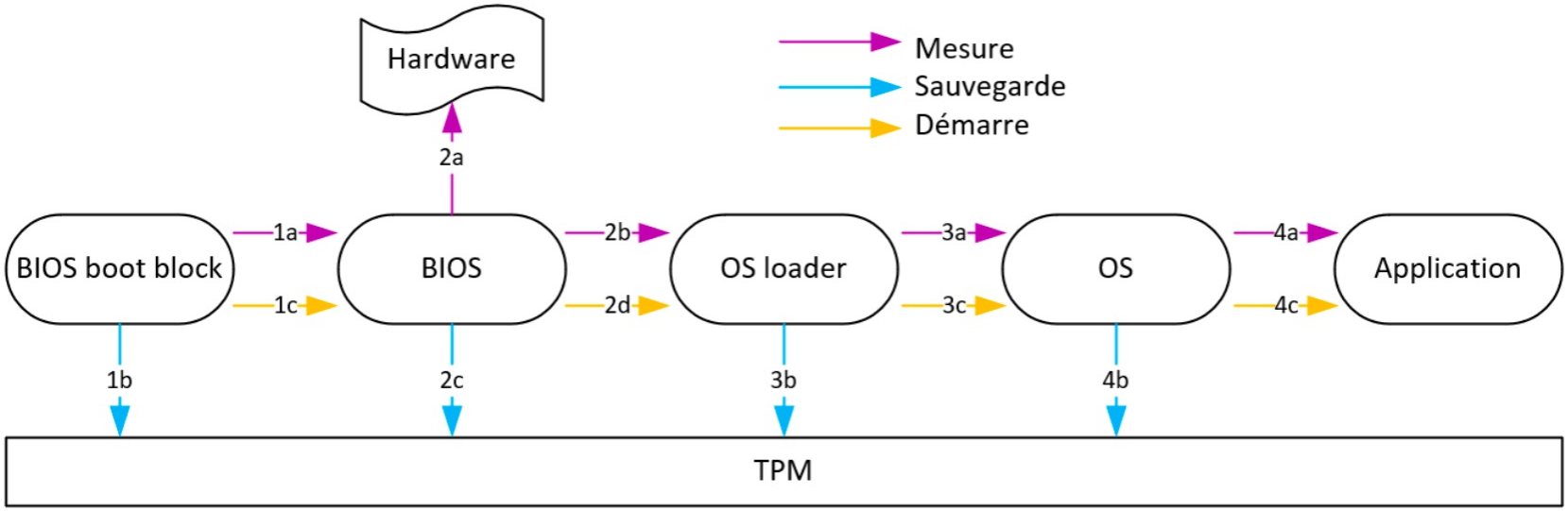


PCR, secure boot principle with nanopi



- The security of the secure boot process depends on the security of the CRTM (for nanopi sunxi code). The CRTM, being the first software to run, can't be measured or validated. It's a root of trust. The platform manufacturer can protect the CRTM from attack by making it immutable, putting it in ROM, or otherwise preventing software updates
- The Linux open source Integrity Measurement Architecture (IMA) integrates boot-time measurements into the kernel

PCR, secure boot principle with nanopi



PCR: Secure boot [1]

- Number of PCRs: In practice, a TPM contains multiple PCRs. The 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

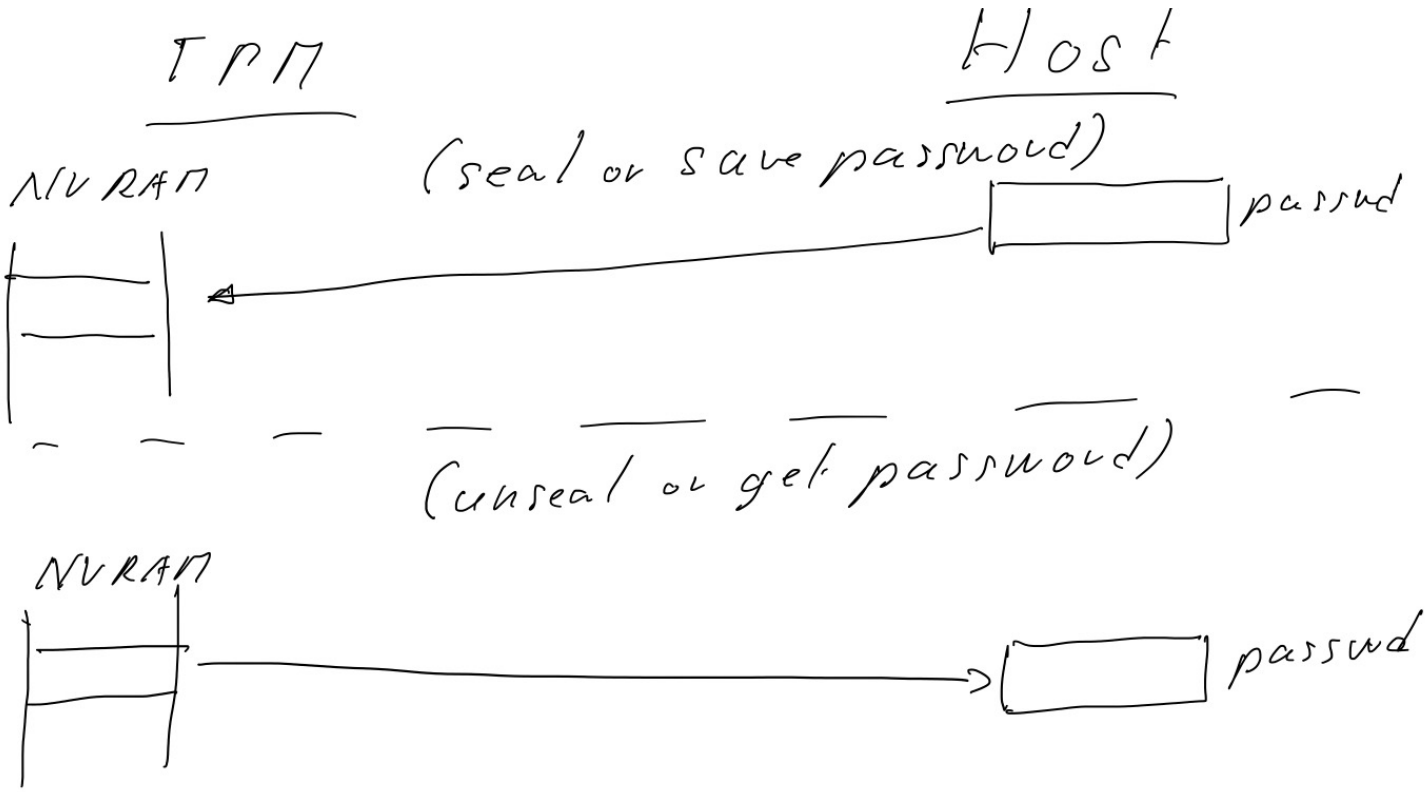
PCR commands

```
tpm2_pcrread
    sha1:
      0 : 0x36509D3791C92465563AD8447F99F19A5BB3A602
      1 : 0x0000000000000000000000000000000000000000
    sha256:
      0 : 0x0000000000000000000000000000000000000000000000000000000000000000
      1 : 0x0000000000000000000000000000000000000000000000000000000000000000
    sha384:
      0 : 0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
      1 : 0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    sha512:
      0 : 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
      1 : 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
tpm2_pcrread sha1:0
    sha1:
      0 : 0x36509D3791C92465563AD8447F99F19A5BB3A602
```

```
tpm2_pcrextend 0:sha1=8c8393ac8939430753d7cb568e2f2237bc62d683
```

LUKS, seal (save) password on TPM



LUKS, seal (save) password on TPM

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

The password is in the passwd file

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

Seal passwd to the TPM

```
shred passwd
rm -f passwd
```

Delete passwd file form the host

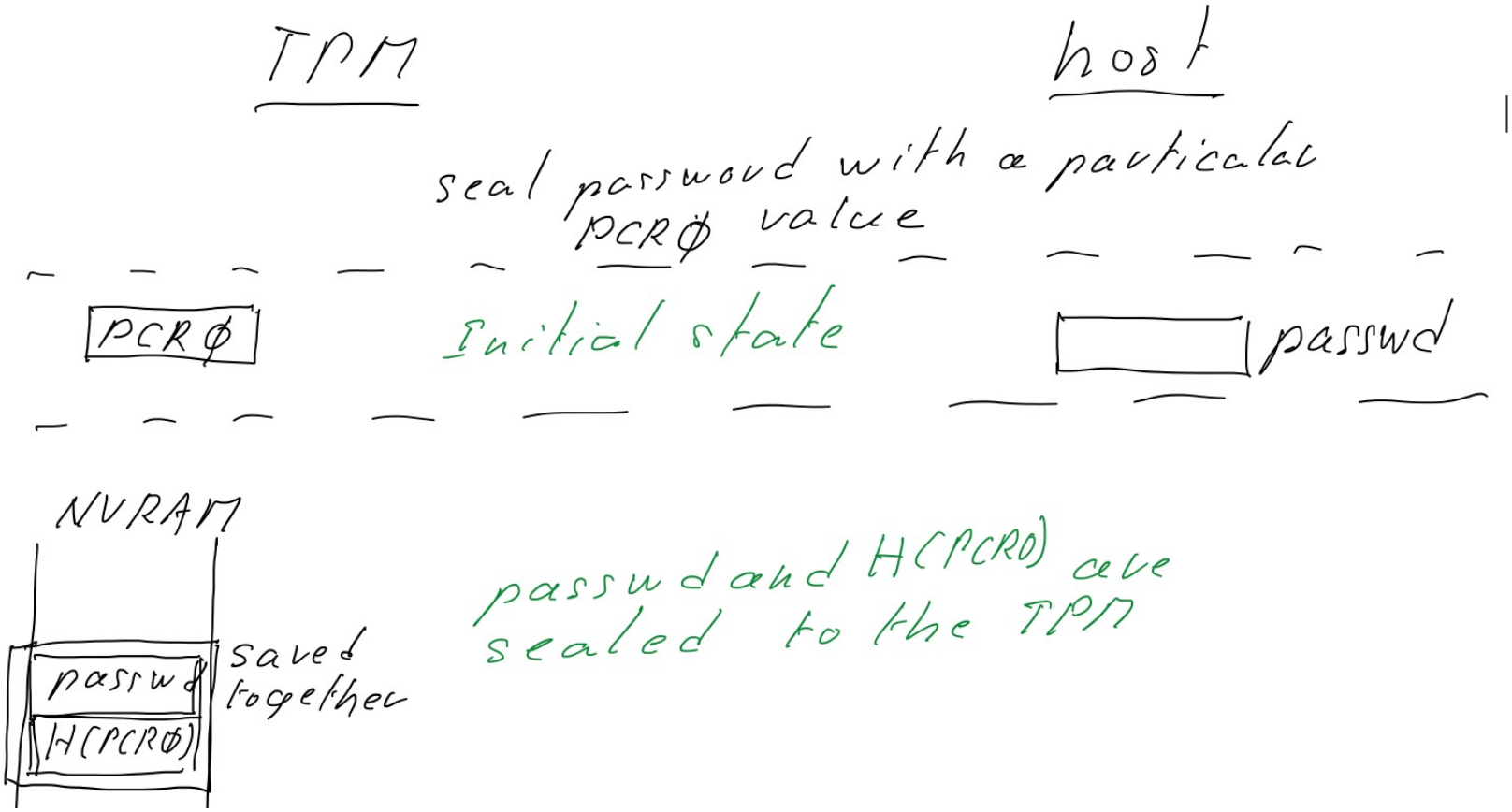
```
tpm2_unseal -c 0x81010000 > passwd
```

Unseal passwd from TPM

LUKS, seal password on TPM and protect with PCR policy

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

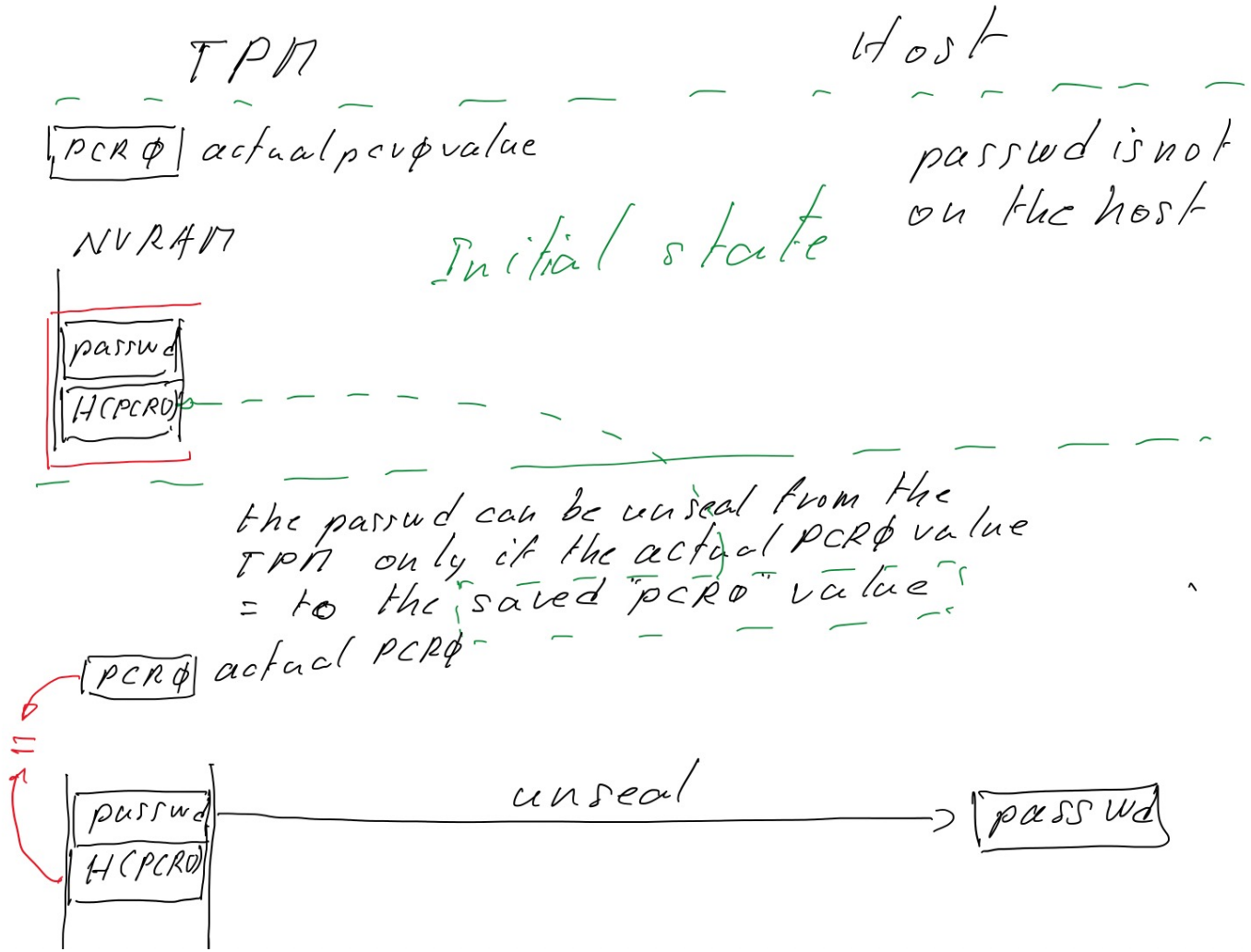
Seal the passwd and H(PCR0) to the TPM



LUKS, seal password on TPM and protect with PCR policy

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

Unseal (get) the passwd from the TPM only when PCR0 is correct



LUKS, seal password on TPM and protect with PCR policy

The password is in the passwd file

```
sha1sum passwd → 8c8393ac8939430753d7cb568e2f2237bc62d683
tpm2_pcrreset 0
tpm2_pcrextend 0:sha1=8c8393ac8939430753d7cb568e2f2237bc62d683
```

Init PCR0

```
tpm2_createprimary -C o –G rsa2048 -c primary
tpm2_startauthsession -S session
tpm2_policypcr -S session  -l sha1:0 -L pcr0_policy
tpm2_flushcontext session
```

1

pcr0_policy = hash(PCR0)

```
tpm2_create -C primary  -g sha256 \
                -u passwd_pcr0.pub -r passwd_pcr0.priv \
                –i passwd -L pcr0_policy
```

2

Link passwd and hash(PCR0) and encrypt with public key

```
tpm2_load -C primary -u passwd_pcr0.pub   \
                     -r passwd_pcr0.priv -c passwd_pcr
tpm2_evictcontrol -c passwd_pcr0 0x81010000 -C o
tpm2_flushcontext session
```

3

Load to TPM NV-Ram

```
shred passwd
rm -f passwd
```

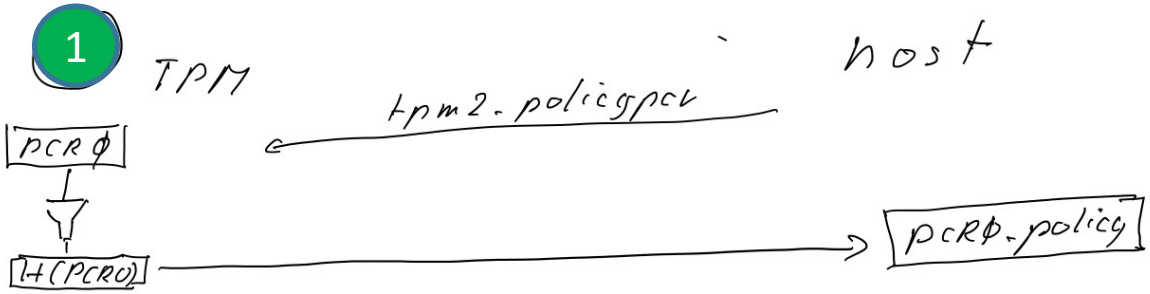
Delete passwd file form the host

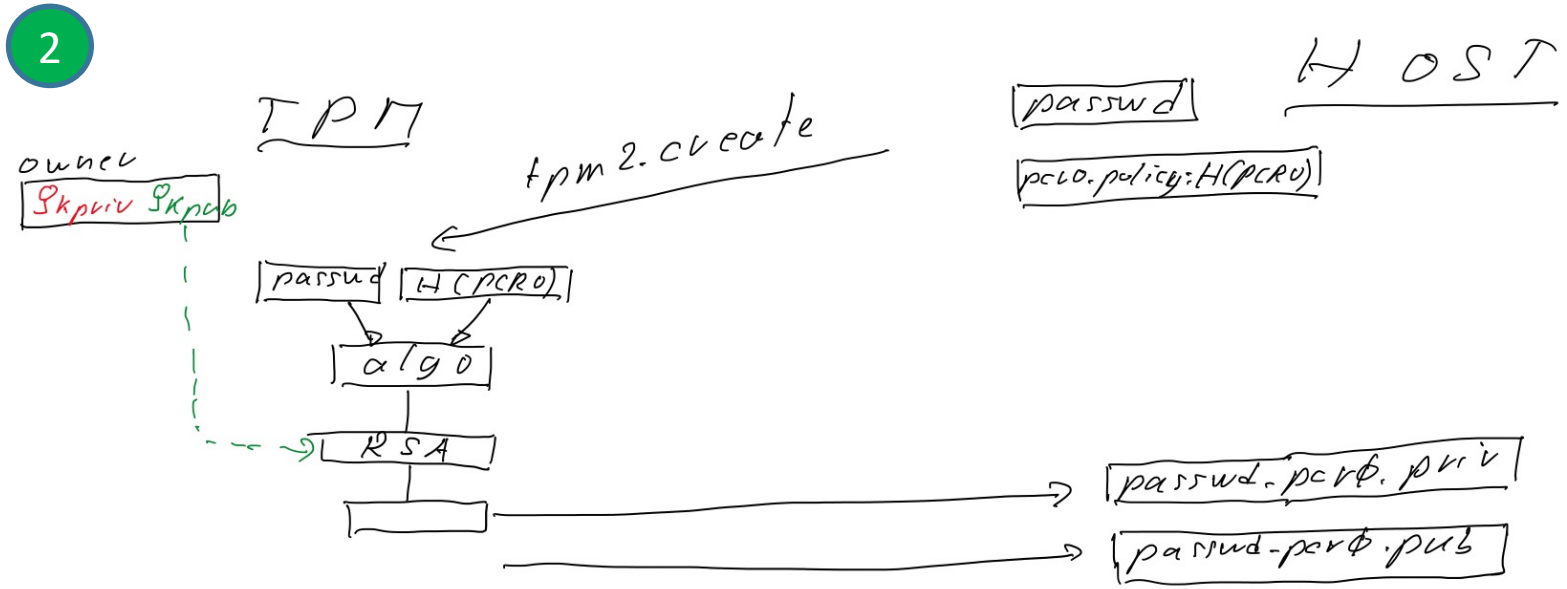
```
tpm2_startauthsession --policy-session -S session
tpm2_policypcr -S session -l sha1:0
tpm2_unseal -p session:session -c 0x81010000 > passwd
```

4

Get actual PCR0 value and get passwd only if actual and saved PCR0 values are correct

LUKS, seal password on TPM and protect with PCR policy





LUKS, seal password on TPM and protect with PCR policy