TPM AND SECURE BOOT 101

Abolfazl Kazemi

Mahsan

Your Support
In Digital World

mahsan.co
info@mahsan.co

Agenda

1. What is Trusted Platform Module?
2. TPM use cases
3. What is Secure Boot?
4. Key Management in Secure Boot
5. Secure Boot and Linux [Ubuntu]

Agenda

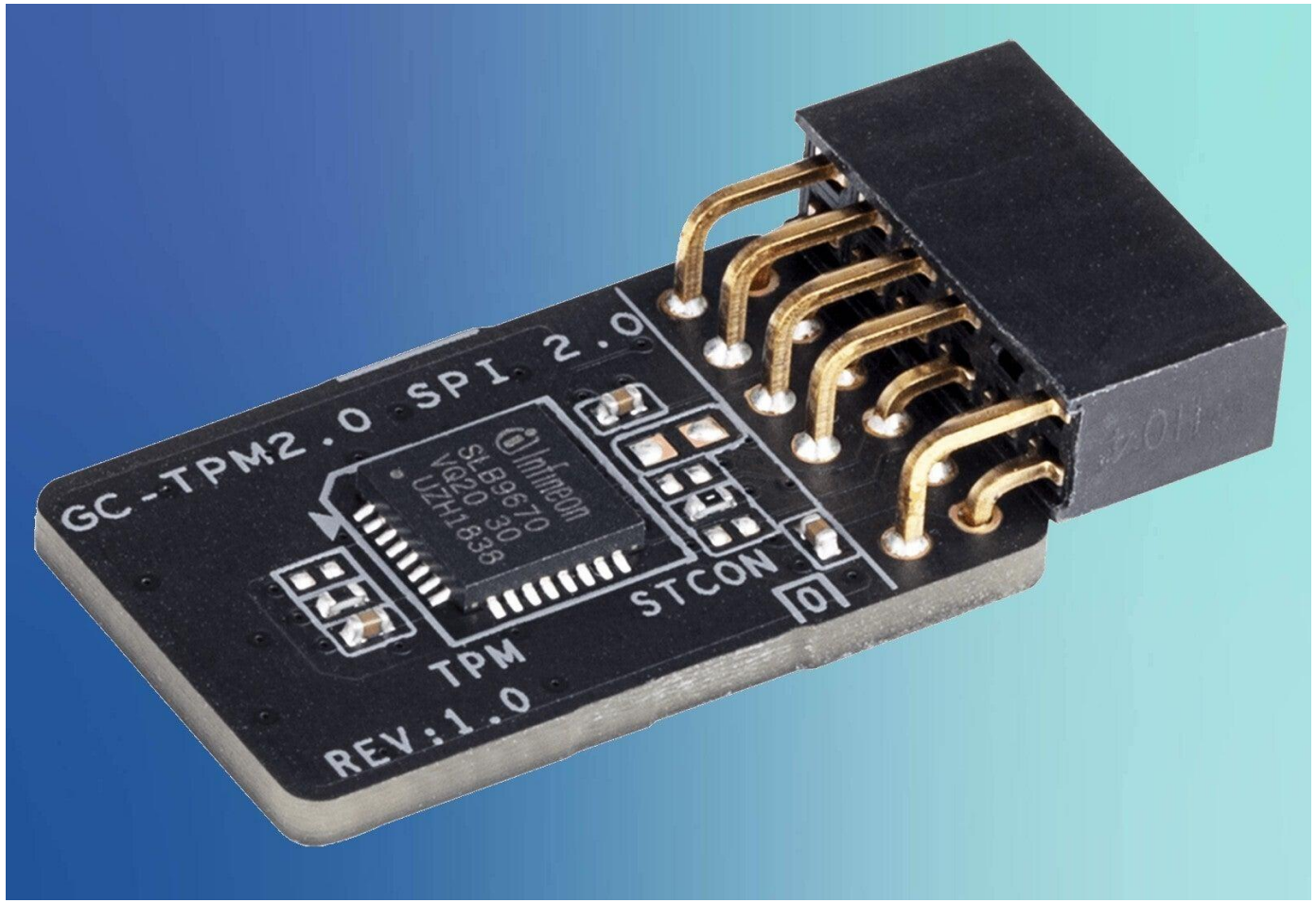
6 TPM Structure

7 TPM and Linux [Ubuntu] (SSH/FDE)

What is a TPM?

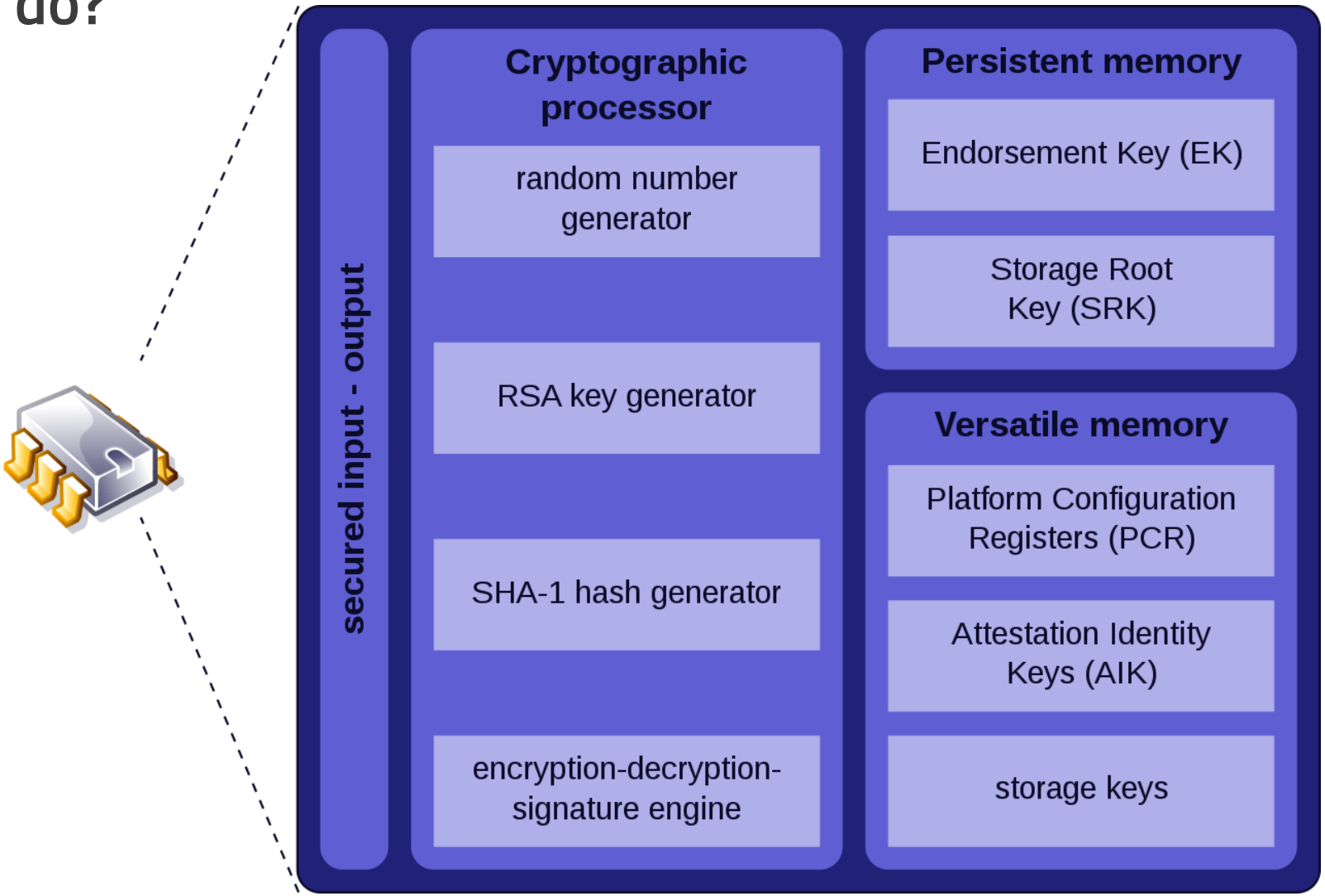
TPM Implementations:

- ❑ Discrete chipset
- ❑ Integrated into CPU
- ❑ Firmware in a Trusted Execution Mode
- ❑ Virtual TPM
- ❑ Software (Only for testing)



Source

What does a TPM do?

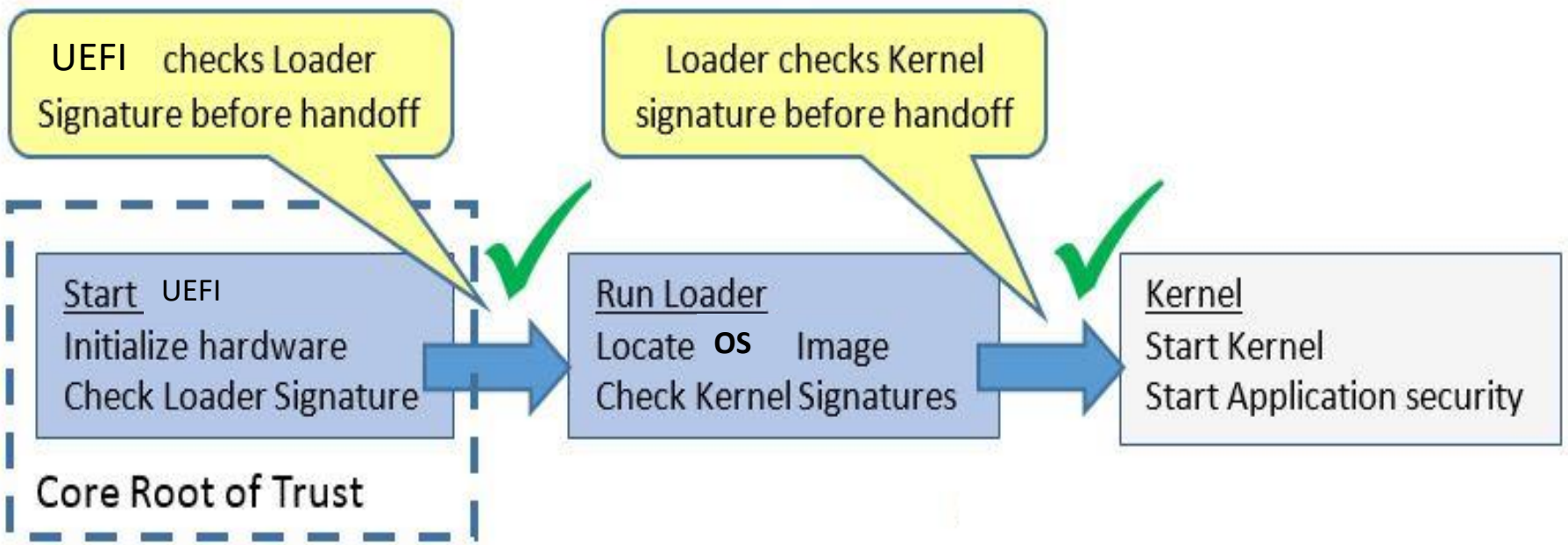


Source

TPM Use Cases

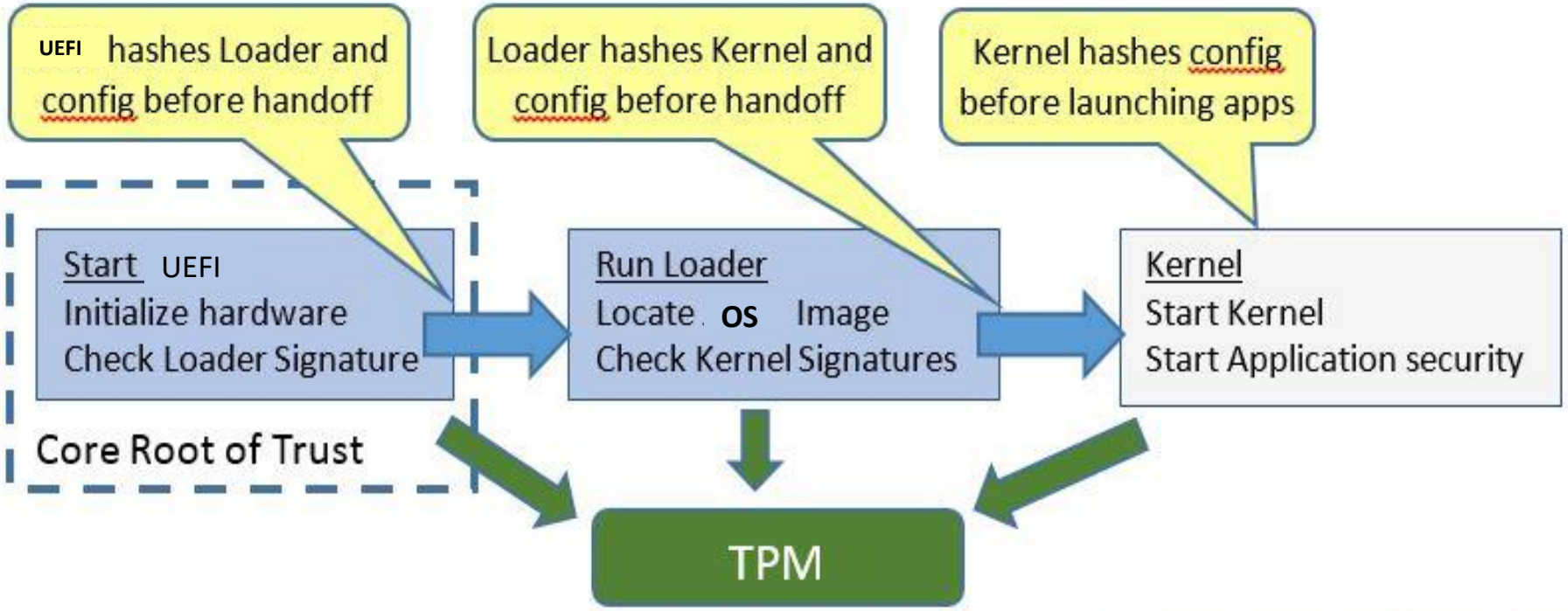
- Device Identification
- Secure Key Generation & Storage
- Data Encryption & Signing
- Random Number Generation
- NVRAM Storage
- Platform Configuration Registers (PCRs)
- Privacy Enablement

What is Secure Boot?

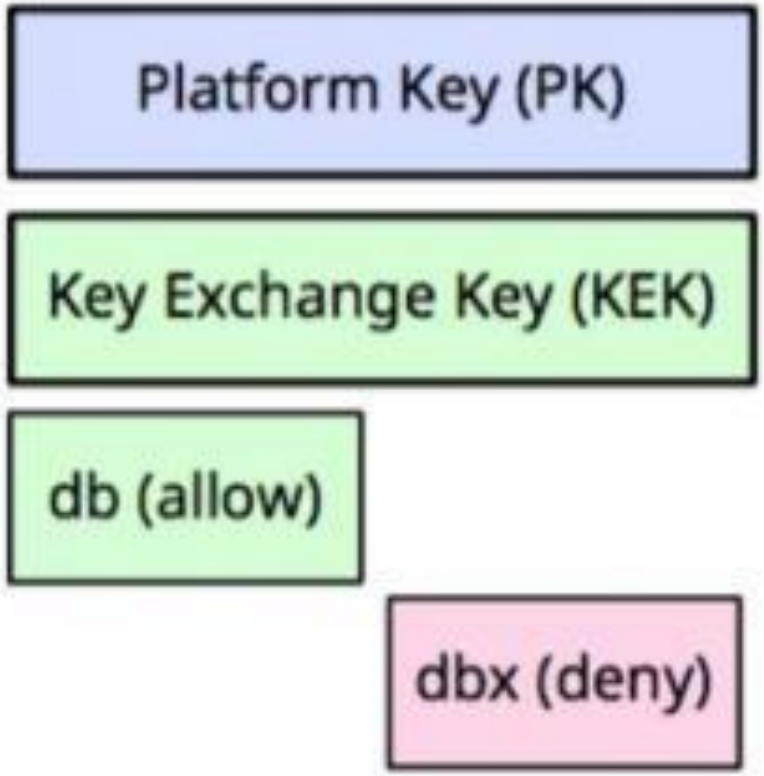


Source

What is Measured Boot?

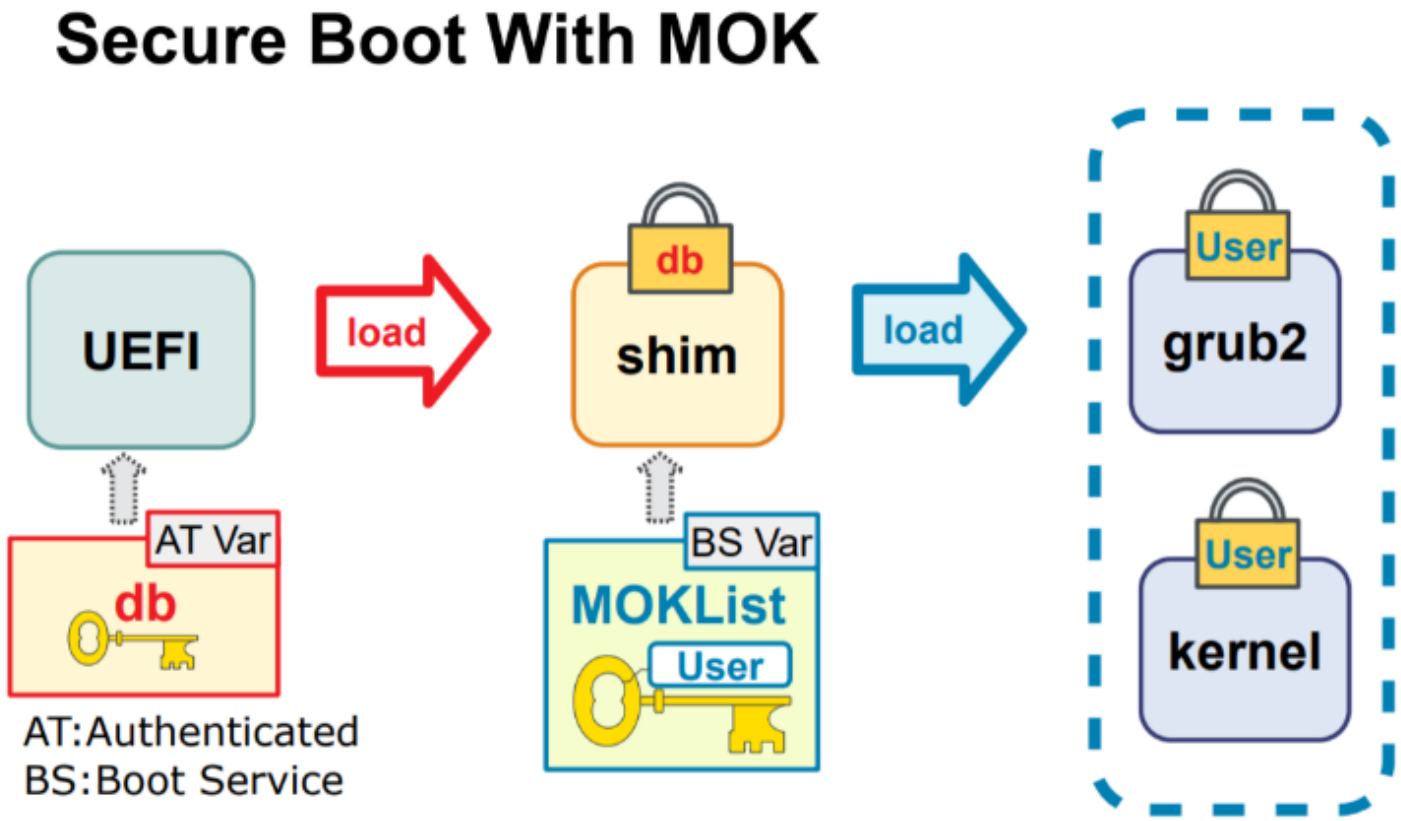


UEFI Keys Structure



Source

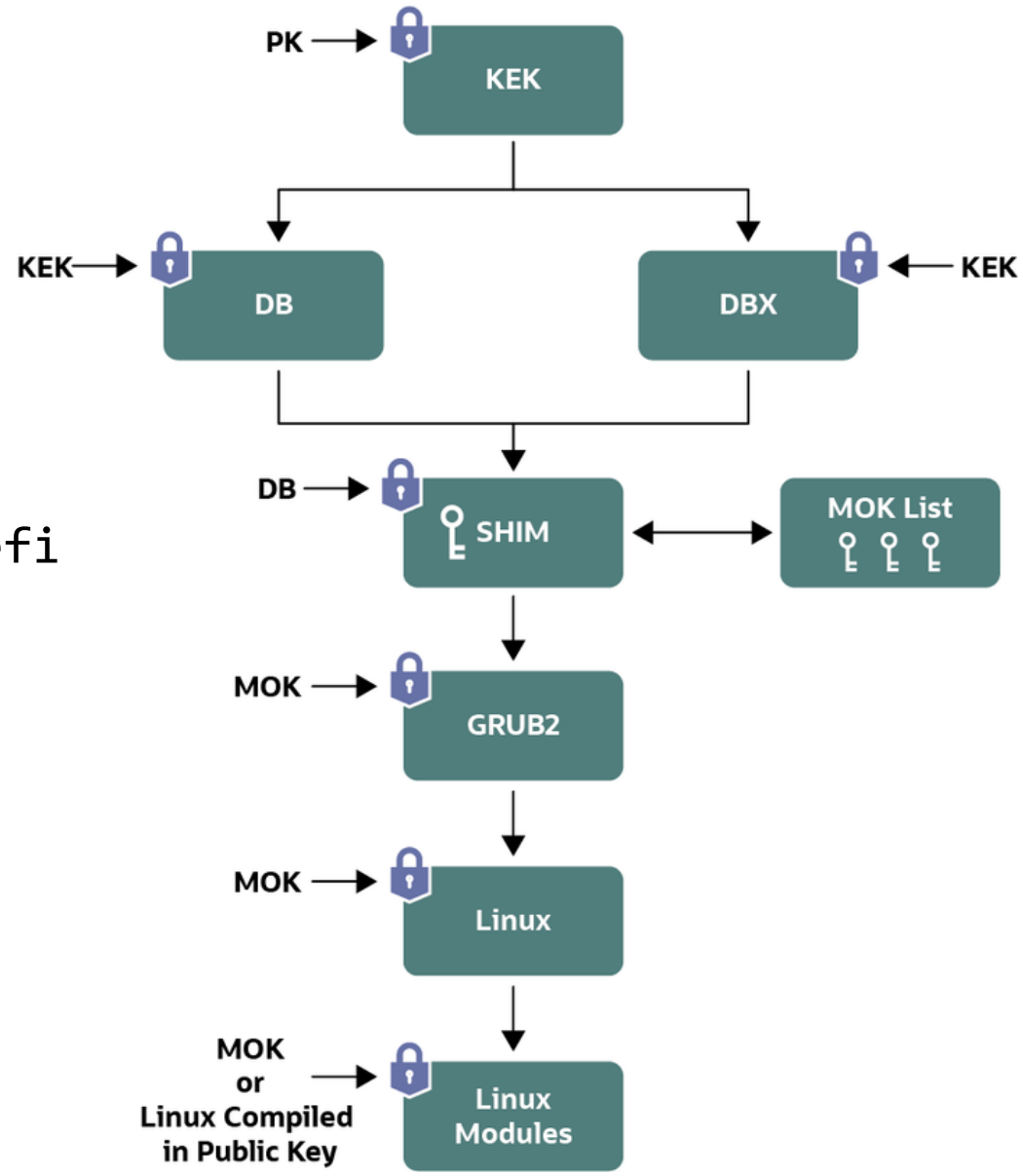
What (the heck) is shim?



Source

Putting it all together

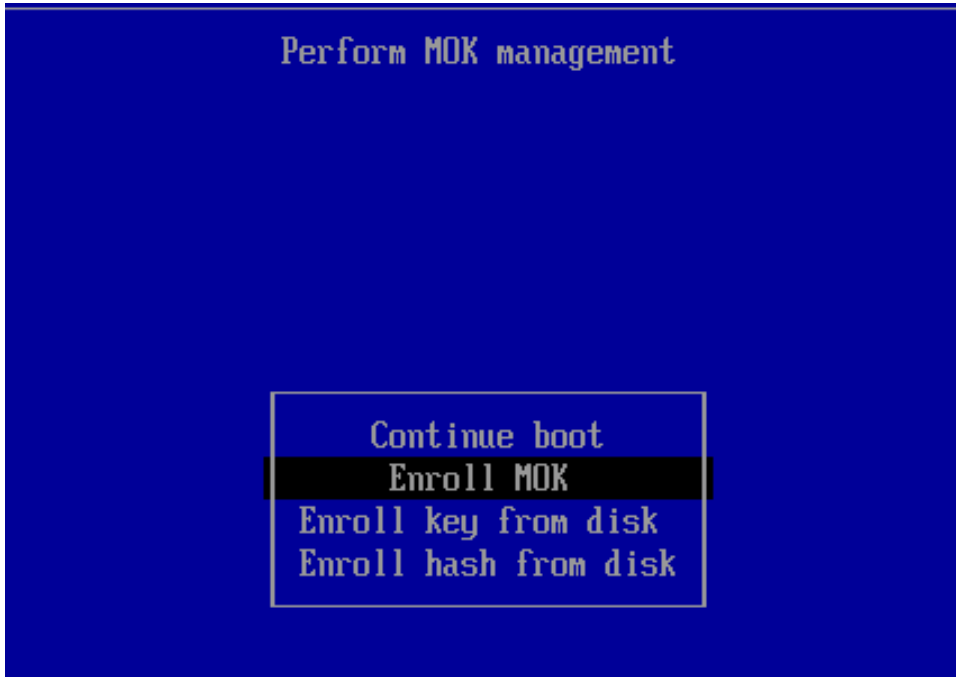
```
#ls /boot/efi/EFI/ubuntu/
...  grubx64.efi  mmx64.efi  shimx64.efi
```



Source

Changing MOK List

```
# mokutil --import Mok.der
# mokutil --list-new
```



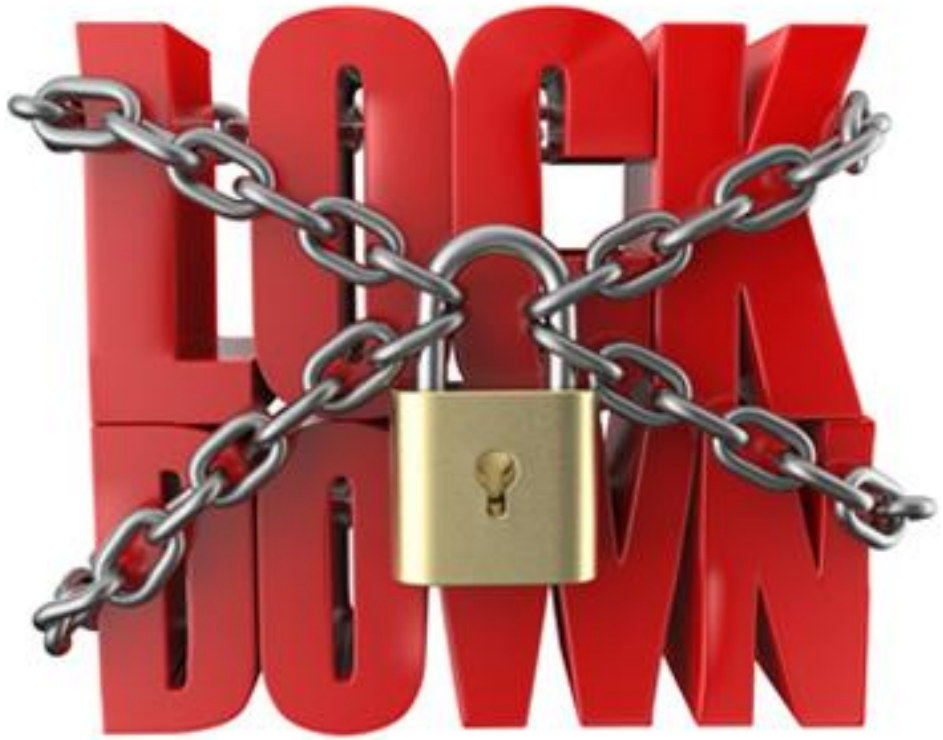

```
integrity: Loading X.509 certificate: UEFI:MokListRT (MOKvar table)
integrity: Loaded X.509 cert 'Mahsan-MOK: c9dbb66deabd3cd806472e780e7671e257410f5f'
Loading compiled-in module X.509 certificates
```

Signing the Linux Kernel

```
root@akjvm02:~# sbverify --list ./vmlinuz-5.13.0-52-generic
signature 1
image signature issuers:
 - /C=GB/ST=Isle of Man/L=Douglas/O=Canonical Ltd./CN=Canonical Ltd. Master Certificate Authority
image signature certificates:
 - subject: /C=GB/ST=Isle of Man/O=Canonical Ltd./OU=Secure Boot/CN=Canonical Ltd. Secure Boot Signing (2017)
   issuer:  /C=GB/ST=Isle of Man/L=Douglas/O=Canonical Ltd./CN=Canonical Ltd. Master Certificate Authority
root@akjvm02:~# sbsign --key Mok.key --cert Mok.pem ./vmlinuz-5.13.0-52-generic --output ./vmlinuz-5.13.0-52-generic.newsign
Image was already signed; adding additional signature
root@akjvm02:~# sbverify --list ./vmlinuz-5.13.0-52-generic.newsign
signature 1
image signature issuers:
 - /C=GB/ST=Isle of Man/L=Douglas/O=Canonical Ltd./CN=Canonical Ltd. Master Certificate Authority
image signature certificates:
 - subject: /C=GB/ST=Isle of Man/O=Canonical Ltd./OU=Secure Boot/CN=Canonical Ltd. Secure Boot Signing (2017)
   issuer:  /C=GB/ST=Isle of Man/L=Douglas/O=Canonical Ltd./CN=Canonical Ltd. Master Certificate Authority
signature 2
image signature issuers:
 - /CN=Mahsan-MOK
image signature certificates:
 - subject: /CN=Mahsan-MOK
   issuer:  /CN=Mahsan-MOK
```

Linux Lockdown

- Was added in Linux 5.4
- Will be automatically enabled if the system boots in EFI Secure Boot
- Only validly signed modules may be loaded
- Designed to prevent both direct and indirect access to a running kernel image
- Use of **debugfs** is not permitted



Source

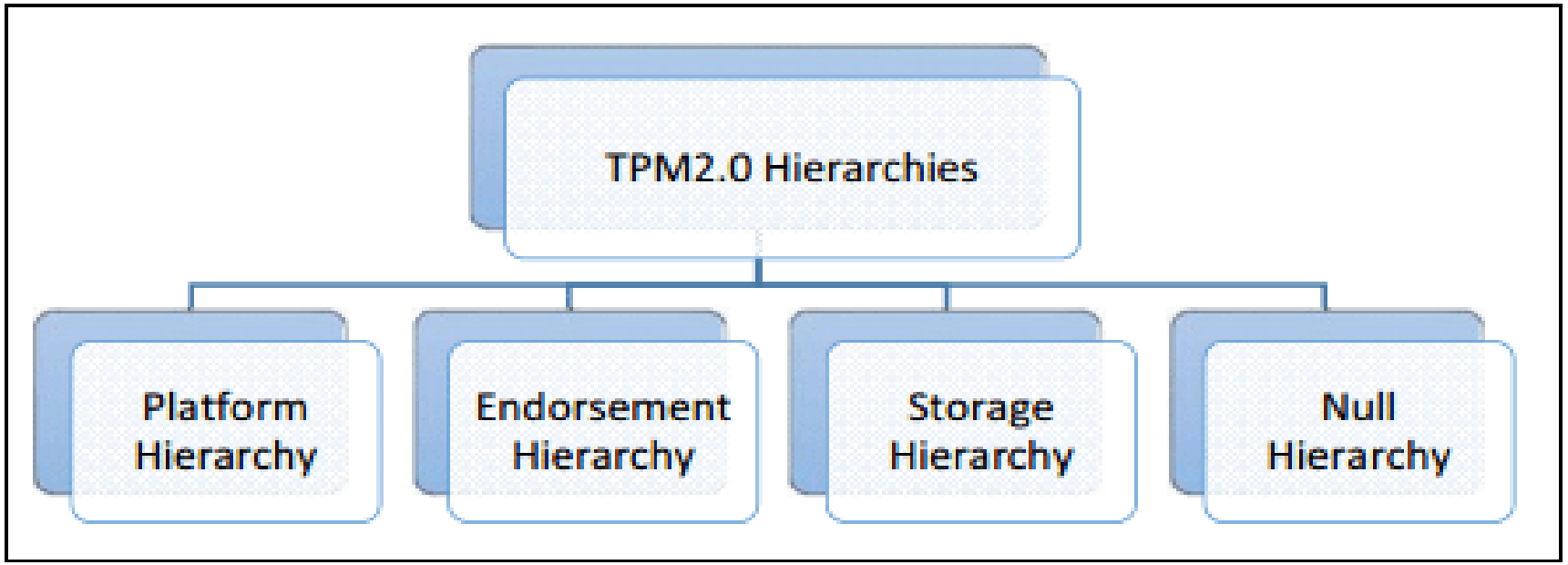
Putting Lockdown into Practice 1

```
root@akjubuvm01:~# mokutil --sb-state
SecureBoot enabled
root@akjubuvm01:~# cat /sys/kernel/security/lockdown
none [integrity] confidentiality
root@akjubuvm01:~# modinfo test.ko
filename:       /root/test.ko
license:        GPL
srcversion:     8110BDA05F2C22DD9878A25
depends:
retpoline:      Y
name:           hello_3
vermagic:       5.13.0-52-generic SMP mod_unload modversions
root@akjubuvm01:~# insmod test.ko
insmod: ERROR: could not insert module test.ko: Operation not permitted
root@akjubuvm01:~# dmesg | tail -1
[ 1204.536278] Lockdown: insmod: unsigned module loading is restricted; see man kernel_lockdown.7
```

Putting Lockdown into Practice 2

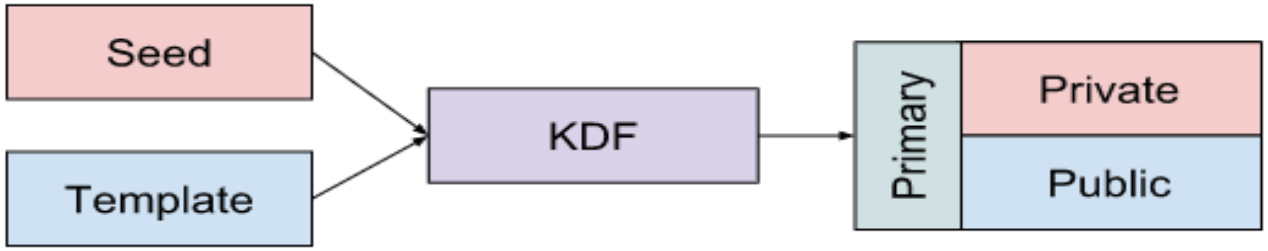
```
root@akjubuvm01:~# alias sign-file=/usr/src/linux-headers-5.13.0-52-generic/scripts/sign-file
root@akjubuvm01:~# export KBUILD_SIGN_PIN=[redacted]
root@akjubuvm01:~# sign-file sha256 Mok.priv Mok.pem test.ko
root@akjubuvm01:~# insmod test.ko
root@akjubuvm01:~# dmesg | tail -1
[ 2000.623355] Hello, world 3
root@akjubuvm01:~# modinfo test.ko
filename:       /root/test.ko
license:        GPL
srcversion:     8110BDA05F2C22DD9878A25
depends:
retpoline:      Y
name:           hello_3
vermagic:       5.13.0-52-generic SMP mod_unload modversions
sig_id:         PKCS#7
signer:         akjmahsan.mah
sig_key:        0E:01:B1:59:6E:BB:EF:9B:9C:53:E5:9B:22:DF:EB:54:FA:64:E9:F7
sig_hashalgo:   sha256
signature:      2E:5D:AF:08:2A:0E:87:30:AD:26:C2:C5:5E:89:97:3B:3E:62:2A:48:
                10:2C:CD:86:0B:65:2E:24:D9:85:10:[redacted]
```

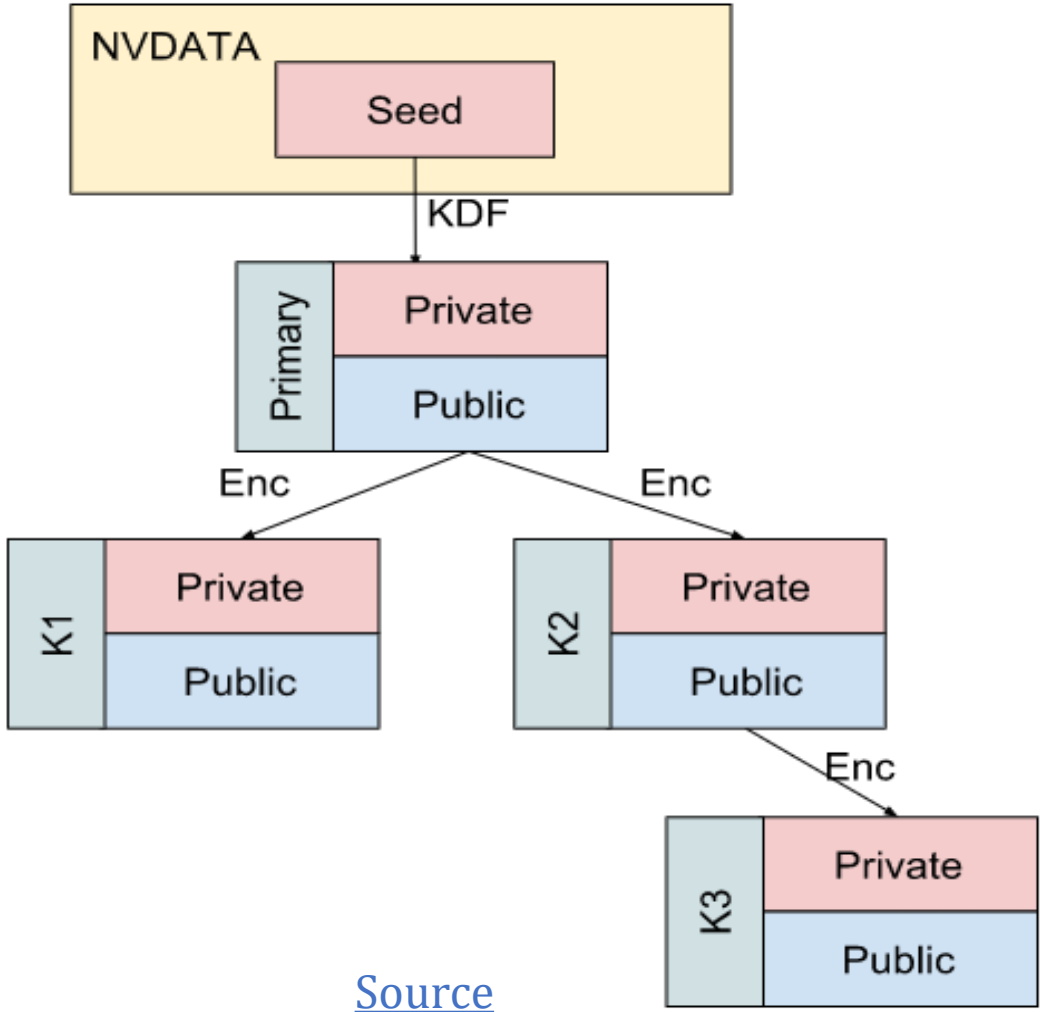
Back to TPM [Structure]



Source

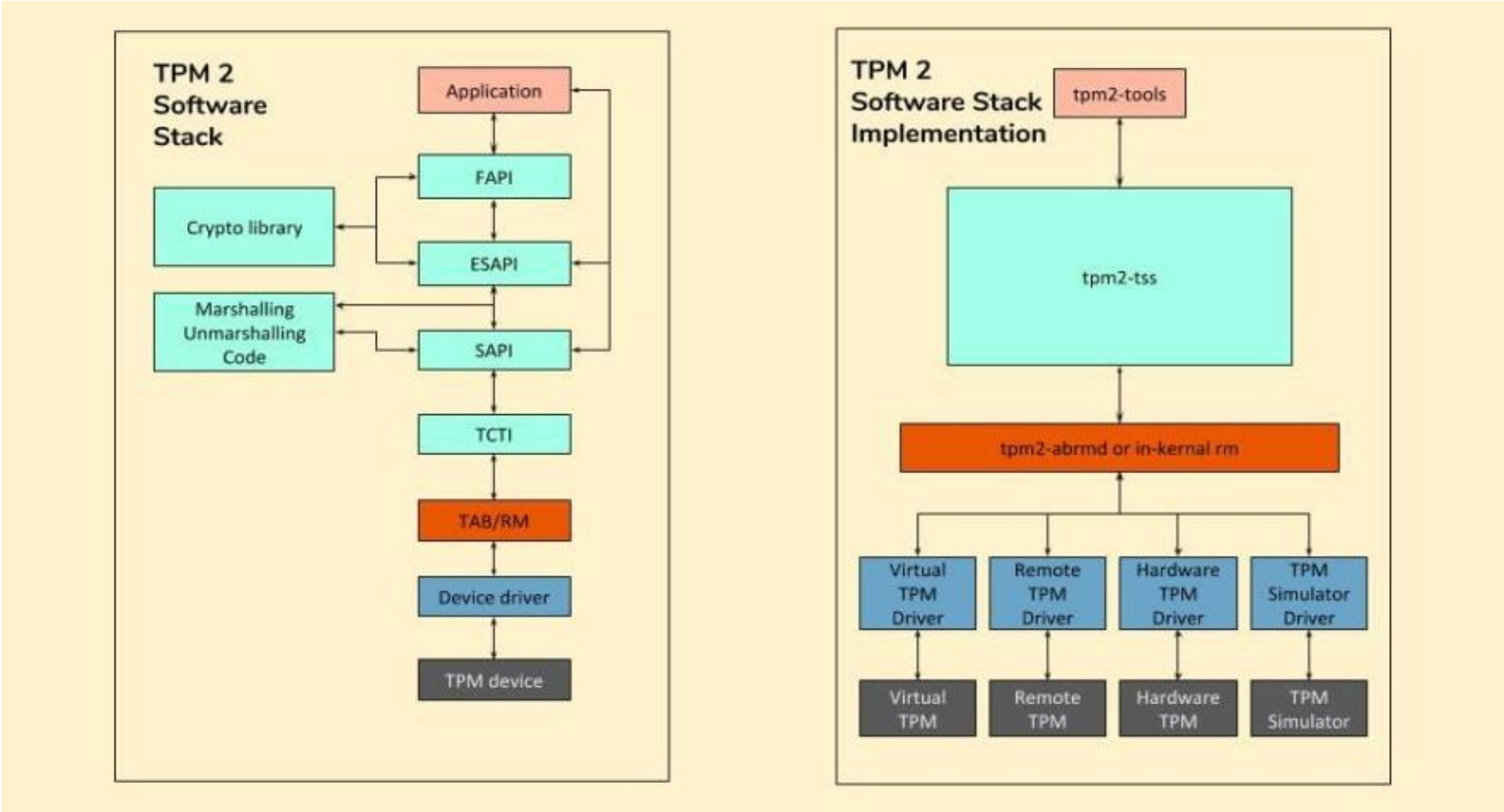
TPM Key Generation





Source

TPM Software Stack



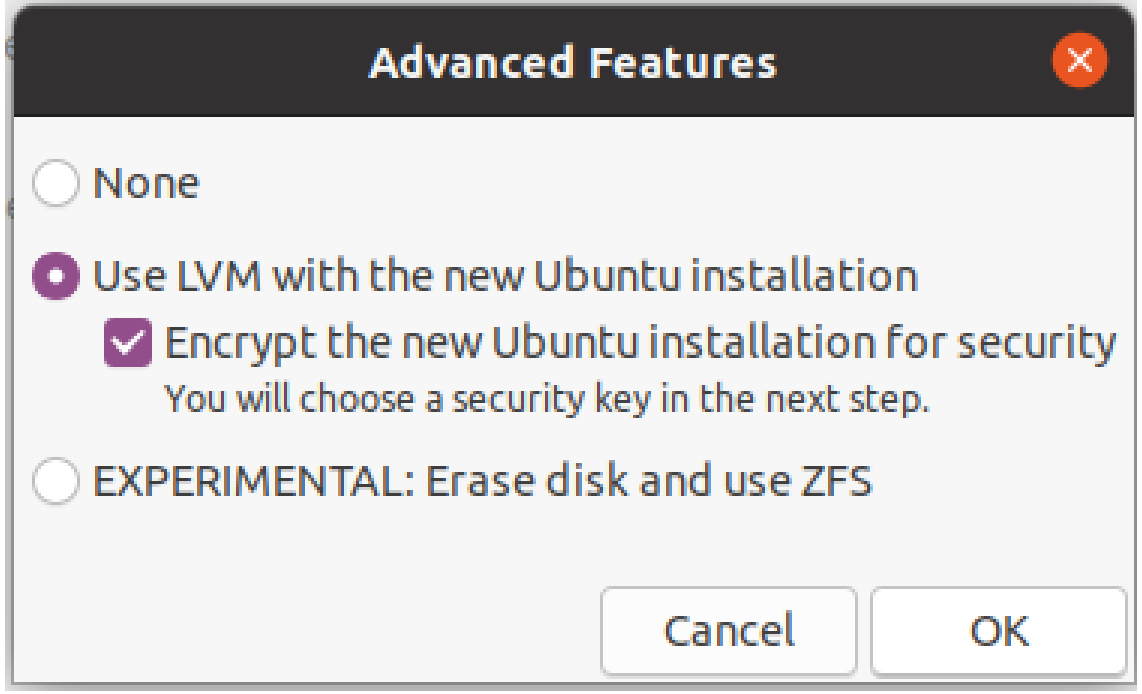
Source

TPM and PKCS#11

```
$ tpm2_ptool init
action: Created
id: 1

$ tpm2_ptool addtoken --pid=1 --label=mylabel --sopin=XXXX --userpin=YYYY
$ tpm2_ptool addkey --label=mylabel --userpin=YYYY --algorithm=ecc256

$ ssh-keygen -D /usr/local/lib/pkcs11/libtpm2_pkcs11.so > key.pub

$ ssh -I /usr/local/lib/pkcs11/libtpm2_pkcs11.so myserver.example
Enter PIN for 'mylabel': YYYY

$ cat ~/.ssh/config
Host myserver.example
  PKCS11Provider /usr/local/lib/pkcs11/libtpm2_pkcs11.so
  IdentityAgent none
```

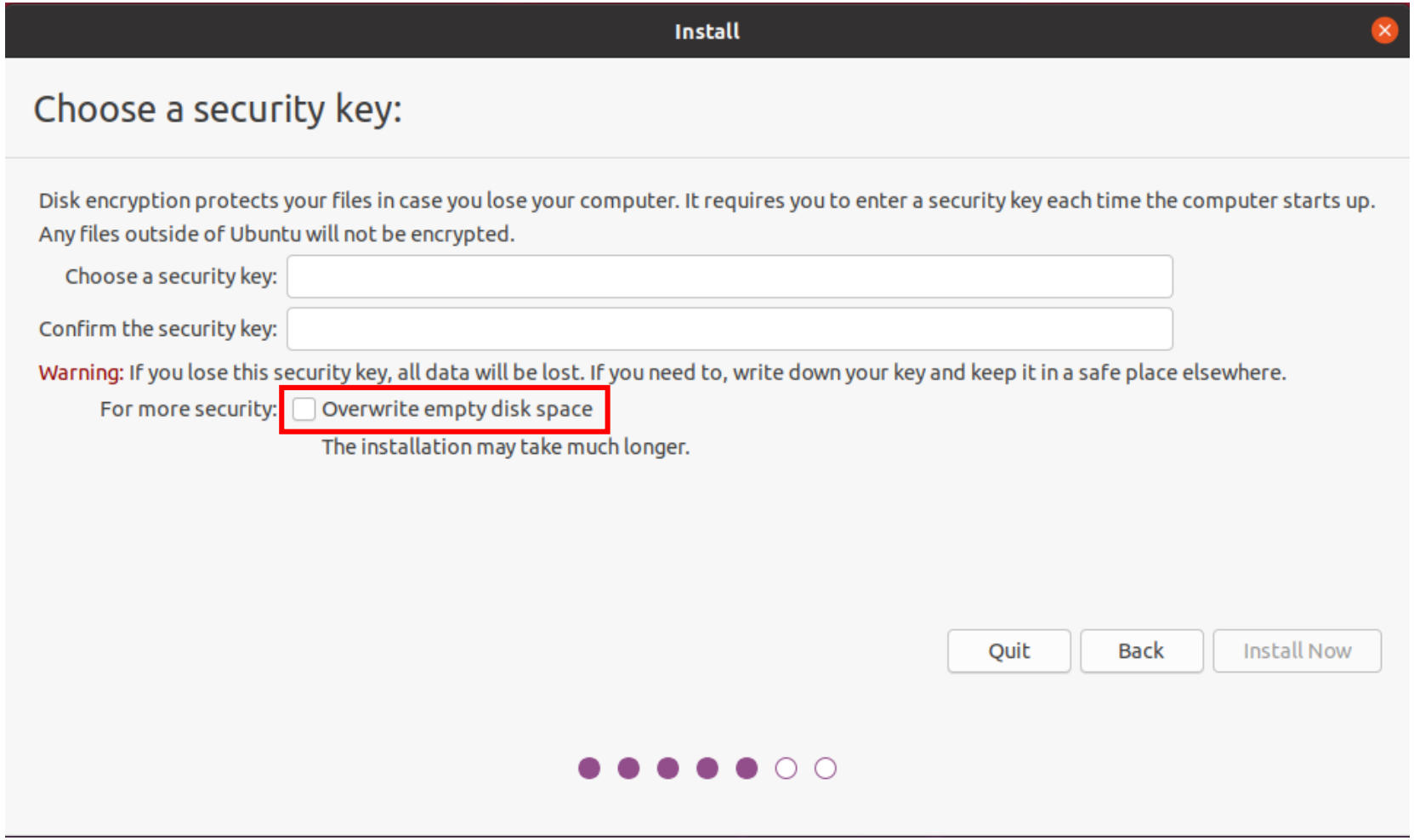
Source

TPM and Full Disk Encryption [except /boot/]



Source

TPM and Full Disk Encryption [except /boot/]



TPM and Full Disk Encryption [except /boot/]

TPM and FDE <STEPS>

1. Install TPM2-TSS and TPM2-TOOLS
2. Generate a new key for LUKS to be stored in TPM
3. Store the key in TPM
4. Write a script to read the key from TPM on Boot and add it to **crypttab**
5. Generate a new **initrd [initramfs]** file to get the key and decrypt the disk on Boot

Source

TPM and FDE

I. Key Generation and TPM Storage

```
cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 64 | head -c64 > root.key
cryptsetup luksAddKey /dev/sda3 root.key
```

```
tpm2_nvdefine 0x1500016 -C o -s 64 -a 0x2000A -T device
tpm2_nvwrite 0x1500016 -C o -i root.key -T device
```

TPM and FDE

II. Setting crypttab to read the key from TPM

```
#!/bin/sh
key=$( tpm2_nvread 0x1500016 -C o -s 64 -T device )
echo -n $key
```

```
sda3_crypt UUID=a8e7db54-cf16-4dad-b516-010bffd45e75 none
luks,discard,keyscript=/usr/local/sbin/key
```

TPM and FDE

III. Generating a new initrd file (/etc/initramfs-tools/hooks/decryptkey)

```
#!/bin/sh
PREREQ=""
prereqs()
{
    echo "$PREREQ"
}
case $1 in
prereqs)
    prereqs
    exit 0
    ;;
esac

. /usr/share/initramfs-tools/hook-functions
copy_exec /usr/local/bin/tpm2_nvread
copy_exec /usr/lib/x86_64-linux-gnu/libtss2-tcti-device.so.0

exit 0
```

```
mv /boot/initrd.img-`uname -r` /boot/initrd.img-`uname -r`.orig
mkinitramfs -o /boot/initrd.img-`uname -r` `uname -r`
```

Things to Consider

- For more security enable Secure Boot and FDE together
- Limit the TPM access to trusted entities
- Harden the OS and set grub/root passwords
- Set a password for UEFI

Thanks

Mahsan

Your Support In Digital World