

Trusted Platform Module

[Trusted Platform Module](#) (TPM) is an international standard for a secure cryptoprocessor, which is a dedicated microprocessor designed to secure hardware by integrating cryptographic keys into devices.

In practice a TPM can be used for various different security applications such as [secure boot](#), key storage and random number generation.

TPM is naturally supported only on devices that have TPM hardware support. If your hardware has TPM support but it is not showing up, it might need to be enabled in the BIOS settings.

List of Platform Configuration Registers

Platform Configuration Registers (PCR) contain hashes that can be read at any time but can only be written via the extend operation, which depends on the previous hash value, thus making a sort of blockchain. They are intended to be used for platform hardware and software integrity checking between boots (e.g. protection against [Evil Maid attack](#)). They can be used to unlock encryption keys and proving that the correct OS was booted.

The [UAPI Group](#) describes the:

PCR	Used by	Notes
PCR0	System Firmware executable code	May change if you upgrade your firmware
PCR1	System Firmware settings	Settings like boot order, etc.
PCR2	Extended executable code	Extended or pluggable executable code (aka OpROMs)
PCR3	Extended executable data	Set during Boot Device Select UEFI boot phase
PCR4	Boot Manager Code + Boot Attempts	Measures boot manager the devices the firmware tried to boot from
PCR5	Boot Manager Configuration + Data	Can measure configuration of boot loaders; includes GPT Partition Table
PCR6	S4/S5 Resume + Power State Events	
PCR7	Secure Boot State	Full contents of PK/KEK/db to validate each boot application

PCR	Used by	Notes
PCR8	Hash of kernel cmdline	Supported by grub and systemd-boot
PCR9	Hash of initrd + EFI Load Options	Kernel 6.1 might measure the kernel cmdline
PCR10	IMA	Protection of the Integrity Measurement Architecture measurement log
PCR11	Hash of Unified kernel image	ELF kernel image, embedded initrd and other payload of the PE image
PCR12	Overridden kernel cmdline	Will be disregarded if Secure Boot is enabled and UKI has embedded kernel cmdline
PCR13	System extension images	System extensions built to extend a base system via overlay images
PCR14	shim	MOK certificates and hashes
PCR15	LUKS key, machine ID, mount points	Root FS encryption key, machine ID, UUID of root volume, FS mounts, UUIDs, etc.

For further details on how PCR 11-13 are used, see [systemd-stub\(7\)](#).

Packages

Package	Usage
<code>tpm2-tss</code>	Implementation of the TCG Trusted Platform Module 2.0 Software Stack (TSS2)
<code>tpm2-tools</code>	Trusted Platform Module 2.0 tools based on <code>tpm2-tss</code>
<code>tpm2-abrmd</code>	A ccess B roker and R esource M anagement D aemon
<code>tpm2-tss-engine</code>	OpenSSL engine for Trusted Platform Module 2.0 devices
<code>tpm2-pkcs11</code>	PKCS#11 interface for Trusted Platform Module 2.0 hardware
<code>tpm2-totp</code>	Attest the trustworthiness of a device against a human using time-based one-time passwords

Configuration

1. Add user to the `tss` group

```
sudo usermod -aG tss $USER
```

2. Enable access broker

```
sudo systemctl enable --now tpm2-abrmd
```

3. Logout and login again

Usage

TPM2-based LUKS key

You can use the trusted platform module in your computer as a key store to unlock LUKS encrypted volumes with them.

Arch Linux comes with `systemd` which itself comes with `systemd-cryptenroll` and allows you to specify a TPM2 device for key storage.

Using a TPM for this purpose automates the process of unlocking your LUKS volumes, given that certain conditions are met, e.g. the firmware of the machine and the Secure Boot state.

WARNING: When using this method on your root volume, there are a few caveats to be aware of.

Provided the PCR slots you chose to seal against are considered valid by the system, the TPM will automatically unlock the LUKS volume at boot without the need to enter a password.

However, this also means that in case of theft, the data is no longer protected by the encryption. Furthermore, this also makes you more vulnerable to cold boot attacks, since the computer just has to be booted up to gain access to the decrypted data, without even the need to tamper with the device in order to crack the encryption.

It is therefore **strongly recommended** to at least pass the `--tpm2-with-pin=yes` option to `systemd-cryptenroll` to still have a mechanism for user verification (available with `systemd` version 251).

Enrolling a new key

Certain preconditions are necessary to use TPM2 in conjunction with a LUKS encrypted volume:

- `tpm2-tss` must be installed

- the volume uses LUKS2 encryption (default when using `cryptsetup`)
- the initramfs must be `systemd`-based (`mkinitcpio` hooks: `systemd` and `sd-encrypt`)

Start by getting a list of TPM2 devices available in your machine:

TIP: If there are no devices listed, make sure the TPM is enabled in your device's firmware. Devices from 2016 onwards usually have a TPM 2.0, as it is a [requirement from Microsoft](#) for Windows 10 certification for hardware manufacturers.

```
systemd-cryptenroll --tpm2-device=list
```

To enroll a new TPM-based key into a LUKS slot specify the TPM device to generate the key from and the PCRs to seal against, followed by the LUKS volume to save a new slot to (using `/dev/nvme0n1p2` as an example):

ATTENTION: The more PCRs you bind to, the more hardened your setup becomes. But at the same time you can also end up with a less flexible setup — e.g. binding your TPM LUKS key to PCRs 8, 9 and/or 11 harden your system against attempts to boot a kernel image which's hashes aren't measured into these PCRs but the moment your kernel changes (i.e. you update your kernel, change initrd generation, etc.) the PCRs stop validating and trigger a passphrase or recovery key prompt!

TIP: If your device only has one TPM (which is usually the case) you can supply `--tpm2-device=auto` to use the only device available.

```
systemd-cryptenroll --tpm2-device=/path/to/tpm2_device --tpm2-pcrs=0+7 --tpm2-with-pin=yes /dev/nvme0n1p2
```

It will ask you for a PIN to enter, which you will be asked to put in every time you boot the system.

Recovery key

It is also generally advisable to let `systemd-cryptenroll` generate a recovery key, in case the key stored in the TPM doesn't validate anymore for whatever reason.

A recovery key is generated automatically with a character set that's easy to type in while still having high entropy.

To generate a recovery key and have it saved to a slot in the LUKS device (using `/dev/nvme0n1p2` as an example):

```
systemd-cryptenroll --recovery-key /dev/nvme0n1p2
```

Unlocking at boot

Making systemd use the TPM to unlock the volume can be done in one of two ways:

1. add kernel parameters, telling systemd which device to unlock with the TPM
2. use a `/etc/crypttab.initramfs` file to be included in the initramfs to point systemd to the correct volume

For the kernel command line, add the following:

TIP: Again, if your device only has one TPM you can supply `tpm2-device=auto` to use the only device available.

```
rd.luks.options=tpm2-device=/path/to/tpm2_device
```

If you'd rather use a `crypttab.initramfs` file, the syntax is as follows:

```
# <name> <device> <passphrase> <options>
root UUID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX none tpm2-device=auto
```

TOPT code during boot

You can use `tpm2-totp` as a means for attestation that the system you are trying to boot is trustworthy by making it display a TOTP code during boot. If the code displayed during boot matches the one in your TOTP app, the system can be considered untampered. Otherwise you have to assume the system is no longer trustworthy and has been tampered with in one way or another.

ATTENTION: If you want to use `tpm2-totp` in conjunction with Plymouth you will have to use the `tpm2-totp-git` package from the AUR, since Arch Linux does not officially support Plymouth and the Repo package lacks the initcpio hooks needed for Plymouth support.

Use the `plymouth-tpm2-totp` hook in conjunction with the `encrypt` hook, or `sd-plymouth-tpm2-totp` in conjunction with the `sd-encrypt` hook.

Generate authentication code and seal against PCRs 0 (hash of UEFI firmware) and 7 (Secure Boot state):

```
sudo tpm2-totp --pcrs=0,7 generate    # official package
sudo tpm2-totp --pcrs=0,7 init        # AUR package
```

This will output a QR code on the terminal that you can scan with your TOTP app of choice.

Test your TOTP by requesting a 6-digit code to be printed and compare with your TOTP app to verify the codes match:

```
tpm2-totp calculate    # official package
tpm2-totp show        # AUR package
```

Add the `tpm2-totp` hook to the HOOKS array in `/etc/mkinitcpio.conf` **before** the `encrypt` hook, else you won't be able to see the TOTP during boot.

TPM-based SSH keys

Revision #12

Created 10 September 2021 15:40:57 by Sebin

Updated 6 June 2025 20:12:16 by Sebin