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.

Related articles

- Self-encrypting drives
- Smartcards
- Trusted Platform Module/1.2

Note: There are two very different TPM specifications: 2.0 and 1.2, which also use different software stacks. **This article only describes TPM 2.0**, for the older TPM 1.2 see /1.2.

1 Checking TPM support

Most modern computers support TPM 2.0, as it has been **required (https://www.computerworld.com/article/3101427/microsoft-mandates-windows-10-hardware-change-for-pc-security.html)** for Windows 10 certification since 2016. To check support on your system, use any of the following methods:

- check the logs, e.g., by running `journalctl -k --grep=tpm` as root
- read the value of `/sys/class/tpm/tpm0/device/description` **[1] (https://github.com/tpm2-software/tpm2-tools/issues/604#issuecomment-342784674)** or `/sys/class/tpm/tpm0/tpm_version_major` :

```
$ cat /sys/class/tpm/tpm0/device/description
TPM 2.0 Device
```

- use `systemd-analyze(1) (https://man.archlinux.org/man/systemd-analyze.1)` to check for TPM 2.0 and the necessary software dependencies:

```
$ systemd-analyze has-tpm2
```

TPM 2.0 allows direct access via `/dev/tpm0` (one client at a time), kernel-managed access via `/dev/tpmrm0` , or managed access through the `tpm2-abrmd (https://archlinux.org/packages/?name=tpm2-abrmd)` resource manager daemon. **According to (https://groups.google.com/g/linux.debian.bugs.dist/c/OWBHw4og6Vk/m/rnwwkVapDAAJ)** a systemd project member, using `tpm2-abrmd (https://archlinux.org/packages/?name=tpm2-abrmd)` is no longer recommended. There are two choices of userspace tools, `tpm2-tools (https://archlinux.org/packages/?name=tpm2-tools)` by Intel and `ibm-tss (https://aur.archlinux.org/packages/ibm-tss/)`[AUR] by IBM.

TPM 2.0 requires **UEFI** boot; BIOS or Legacy boot systems can only use TPM 1.2.

Some TPM chips can be switched between 2.0 and 1.2 through a firmware upgrade (which can be done only a limited number of times).

2 Usage

Many informative resources to learn how to configure and make use of TPM 2.0 services in daily applications are available from the **tpm2-software community (https://tpm2-software.github.io/)**.

2.1 LUKS encryption

It is possible to encrypt volumes using keys securely stored in the TPM. This approach ensures that your drives remain locked unless the TPM is present and specific conditions are met, such as the integrity of the firmware or **Secure Boot** state (see **#Accessing PCR registers**).

This mechanism can be used to automatically decrypt the root volume during the boot process, similarly to how BitLocker works on Windows or FileVault on macOS. While this provides strong protection if the drive is removed from the computer with the TPM, data protection will only rely on basic measures like user passwords and system settings if the entire PC is stolen. To mitigate this, you can:

- Consider encrypting user data, such as individual home folders, with a different mechanism, such as **fscrypt** or **systemd-homed**.
- Use a **TPM pin** to benefit from the security properties of the TPM, while avoiding completely unattended unlocking.

> **Warning:** Be aware that this method makes you more vulnerable to **cold boot attacks**.

systemd-cryptenroll and **Clevis** allow locking LUKS volumes with a key stored in the TPM. Additionally, systemd-cryptenroll enables tying the encryption to signed policies instead of static PCR values (See `systemd-cryptenroll(1) (https://man.archlinux.org/man/systemd-cryptenroll.1)`).

2.2 SSH

For TPM sealed SSH keys, there are two options:

- **ssh-tpm-agent** — ssh-agent compatible agent using TPM backed keys.

  https://github.com/Foxboron/ssh-tpm-agent || `ssh-tpm-agent (https://archlinux.org/packages/?name=ssh-tpm-agent)`
  See **Store ssh keys inside the TPM: ssh-tpm-agent (https://linderud.dev/blog/store-ssh-keys-inside-the-tpm-ssh-tpm-agent/)**.

- **tpm2-pkcs11** — PKCS#11 interface for Trusted Platform Module 2.0 hardware.

  https://github.com/tpm2-software/tpm2-pkcs11 || `tpm2-pkcs11 (https://archlinux.org/packages/?name=tpm2-pkcs11)`

See SSH configuration (https://github.com/tpm2-software/tpm2-pkcs11/blob/master/docs/SSH.md) and Using a TPM for SSH authentication (https://incenp.org/notes/2020/tpm-based-ssh-key.html) (2020-01).

2.3 GnuPG

GnuPG, since version 2.3, supports moving compatible keys into the TPM. See Using a TPM with GnuPG 2.3 (https://gnupg.org/blog/20210315-using-tpm-with-gnupg-2.3.html) for the instructions.

2.4 Other good examples of TPM 2.0 usage

- Configuring Secure Boot + TPM 2 (https://threat.tevora.com/secure-boot-tpm-2/) (2018-06, Debian)
- Using the TPM - It's Not Rocket Science (Anymore) (https://www.youtube.com/watch?v=XwaSyHJIos8) - Johannes Holland & Peter Huewe (2020-11, Youtube): examples for OpenSSL with `tpm2-tss-engine` (https://archlinux.org/packages/?name=tpm2-tss-engine)

3 Accessing PCR 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 TCG PC Client Specific Platform Firmware Profile Specification (https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/) defines the registers in use, and The Linux TPM PCR Registry (https://uapi-group.org/specifications/specs/linux_tpm_pcr_registry/) assigns Linux system components using them.

The registers are:

PCR	Description	Extended by
PCR0	Core System Firmware executable code (aka Firmware). May change if you upgrade your UEFI.	Firmware
PCR1	Core System Firmware data (aka UEFI settings)	Firmware
PCR2	Extended or pluggable executable code (aka OpROMs)	Firmware
PCR3	Extended or pluggable firmware data. Set during Boot Device Select UEFI boot phase.	Firmware
PCR4	Boot Manager Code and Boot Attempts. Measures the boot manager and the devices that the firmware tried to boot from.	Firmware
PCR5	Boot Manager Configuration and Data. Can measure configuration of boot loaders; includes the GPT Partition Table.	Firmware
PCR6	Resume from S4 and S5 Power State Events	Firmware
PCR7	Secure Boot State. Contains the full contents of PK/KEK/db, as well as the specific certificates used to validate each boot application[2] (https://superuser.com/questions/1640985/how-to-enable-bitlocker-when-booting-windows-10-from-a-non-microsoft-boot-manage)	Firmware, shim (adds MokList, MokListX, and MokSBState)
PCR8[1]	Hash of the kernel command line	GRUB (https://lists.gnu.org/archive/html/grub-devel/2017-07/msg00003.html), systemd-boot (https://github.com/systemd/systemd/pull/2587)
PCR9[1]	Hash of the initrd and EFI Load Options	Linux (measures the initrd and EFI Load Options, essentially the kernel cmdline options)
PCR10[1]	Reserved for Future Use	
PCR11[1]	Hash of the Unified kernel image	`systemd-stub(7)` (https://man.archlinux.org/man/systemd-stub.7)
PCR12[1]	Overridden kernel command line, Credentials	`systemd-stub(7)` (https://man.archlinux.org/man/systemd-stub.7)
PCR13[1]	System Extensions	`systemd-stub(7)` (https://man.archlinux.org/man/systemd-stub.7)
PCR14[1]	shim's MokList, MokListX, and MokSBState.[3] (https://github.com/rhboot/shim/blob/main/README.tpm)	shim
PCR15[1]		Unused
PCR16[1]	Debug. May be used and reset at any time. May be absent from an official firmware release.	
PCR23	Application Support. The OS can set and reset this PCR.	

1. Use case defined by the OS and might change between various Linux distros and Windows devices.

On Windows, BitLocker uses PCR8-11 (Legacy) or PCR11-14 (UEFI) for its own purposes. Documentation from tianocore[4] (https://github.com/tianocore-docs/edk2-TrustedBootChain/blob/main/4_Other_Trusted_Boot_Chains.md).

`tpm2-totp` (https://archlinux.org/packages/?name=tpm2-totp) facilitates this check with a human observer and dedicated trusted device.

The current PCR values can be listed with `systemd-analyze(1)` (https://man.archlinux.org/man/systemd-analyze.1):

```
$ systemd-analyze pcrs
```

Or, alternatively with `tpm2_pcrread(1)` (https://man.archlinux.org/man/tpm2_pcrread.1) from `tpm2-tools` (https://archlinux.org/packages/?name=tpm2-tools):

```
# tpm2_pcrread
```

4 Troubleshooting

4.1 TPM2 LUKS2 unlocking still asking for password

If you followed the **instruction described above** for automatically unlocking luks2 devices with enrolled keys in a TPM2 hardware module, but still receive a prompt to input a password during the initramfs boot stage. You may need to **early load** the kernel module (you can obtain its name with `systemd-cryptenroll --tpm2-device=list`) that is responsible for handling your specific TPM2 module.

5 See also

- **Gentoo:Trusted Platform Module**
- TPM-JS testing tool: **source (https://github.com/google/tpm-js)** - **live web version (https://google.github.io/tpm-js/)**.