# TPM-Backed Full Disk Encryption (FDE)

A Complete Introduction to Hardware Roots of Trust, Sealing Architecture, Hardware Requirements, and Linux Configuration (Arch Linux, Ubuntu & Fedora)

## 1. Introduction: Why Pair FDE with a TPM?

Traditional Full Disk Encryption (such as standard Linux `dm-crypt`/LUKS) protects data at rest by encrypting the entire block device with a Master Key, which is itself unlocked by a user passphrase typed at every boot. While strong against offline theft, traditional FDE has two friction points: users must type a complex password before the OS loads, and remote/headless machines cannot reboot unattended.

**TPM-backed Full Disk Encryption** delegates key protection to a dedicated cryptographic processor—the **Trusted Platform Module (TPM 2.0)**. Instead of relying solely on a human-memorized password, the disk's decryption secret is cryptographically bound ("sealed") to the exact hardware and boot-software state of the machine. If the drive is removed and placed into another computer, or if an attacker tampers with the bootloader or kernel, the TPM refuses to release the key.

> **Security Trade-Off: Convenience vs. Physical Cold-Boot / Bus Sniffing**
>
> Unattended TPM-only unlock protects against stolen drives and unauthorized firmware modifications, but if an entire powered-off laptop is stolen, an attacker could attempt physical SPI/LPC bus sniffing or RAM extraction during auto-boot. Adding a **TPM + PIN** combines hardware state attestation with something you know, eliminating unattended bus-sniffing risks while keeping rate-limited anti-hammering protection inside the TPM.

## 2. How TPM-Backed Full Disk Encryption Works

The TPM does not encrypt your hard drive directly—it is far too slow for bulk I/O. Instead, the CPU performs high-speed AES-XTS disk encryption using a Volume Master Key, while the TPM acts as a hardware vault protecting that key.

### The Cryptographic Key Hierarchy

- **Storage Primary Seed (SPS):** A permanent, random secret burned into the TPM's non-volatile memory at manufacture. It never leaves the chip.
- **Storage Root Key (SRK):** Derived deterministically from the SPS. The SRK encrypts ("wraps") external secrets so they can be stored safely on disk (e.g., inside a LUKS2 JSON token header) while remaining unreadable without that specific TPM.
- **Volume Master Key (VMK):** The actual symmetric key used by the OS kernel (`dm-crypt`) to decrypt sectors on the SSD/NVMe drive.

## Measured Boot & Platform Configuration Registers (PCRs)

During startup, each component in the boot chain hashes the next component before executing it and sends that hash to the TPM. The TPM stores these measurements inside tamper-evident memory slots called **Platform Configuration Registers (PCRs)** using a one-way cryptographic extend operation:

```
PCR_new = SHA256( PCR_old || SHA256(Next_Boot_Component) )
```

Because PCRs can only be extended (never overwritten directly until a full hardware reboot), malware cannot spoof a clean boot state after loading.

| PCR Index | What Is Measured | Practical Impact on Linux FDE |
|---|---|---|
| **PCR 0** | Core System Firmware (UEFI BIOS code) | Breaks auto-unlock if you flash a BIOS/UEFI update. |
| **PCR 1** | Host Platform Configuration (UEFI settings) | Breaks auto-unlock if you change hardware or BIOS toggles. |
| **PCR 2 & 3** | Option ROMs & PCIe expansion firmware | Changes if you add/remove certain PCIe/Thunderbolt devices. |
| **PCR 4** | UEFI Boot Manager / Bootloader binary | Changes on every GRUB or `systemd-boot` binary update. |
| **PCR 7** | Secure Boot Policy & Enrolled Certificates | **Recommended anchor:** Remains stable across kernel updates as long as Secure Boot stays enabled with trusted keys. |
| **PCR 11** | Unified Kernel Image (UKI) sections | Used by `systemd-ukify` / `systemd-pcrlock` for signed PCR policies so kernel updates work seamlessly. |

## Sealing and Unsealing Lifecycle

**Step 1 — Enrollment (Sealing):** The OS generates a high-entropy LUKS2 key, asks the TPM to encrypt it under the SRK alongside an authorization policy specifying target PCR values (e.g., `PCR 7` or `PCR 0+7`), and saves the encrypted blob in the LUKS2 metadata header.

**Step 2 — Normal Boot (Unsealing):** UEFI firmware, Secure Boot, and the bootloader execute and extend the PCRs. Early userspace (`initrd`) passes the encrypted blob from the LUKS2 header to the TPM. The TPM checks whether its live PCR registers match the policy; if they match, it decrypts the key in memory and unlocks the root filesystem.

**Step 3 — Tamper / Update Fallback:** If a BIOS update or unauthorized bootloader alters a bound PCR, unsealing fails cleanly. The system falls back to prompting for your traditional recovery passphrase or recovery key, after which you re-seal against the new PCR values.

## 3. Required Hardware & Firmware

Before configuring TPM-backed FDE, verify that your hardware satisfies these baseline requirements:

| Component | Requirement & Verification Notes |
|---|---|
| **TPM Chip (2.0 Mandatory)** | Must be **TPM 2.0** (TCG PC Client Profile, Library Spec v1.32+). Legacy TPM 1.2 is unsupported by modern `systemd-cryptenroll` and Ubuntu FDE. Can be a discrete chip (dTPM) on the motherboard or firmware-based (fTPM: Intel PTT / AMD fTPM). |
| **UEFI Firmware** | **UEFI 2.5 or newer** with TCG EFI Protocol 2.0 support. Legacy CSM / BIOS boot mode cannot measure boot components into TPM 2.0 PCRs. |
| **Secure Boot** | Must be **Enabled** (User/Deployed Mode). Without Secure Boot enforcing signature checks, binding to PCR 7 provides no protection against unsigned bootloaders. |
| **Hardware Root of Trust (Optional for Auto-Unlock)** | For strict zero-PIN unlock (required by Ubuntu's installer for PIN-less mode), CPUs generally need **Intel Boot Guard** or **AMD Platform Secure Boot (PSB)** (common on post-2021 business laptops) to verify UEFI firmware before execution. |

### Quick Linux Hardware Verification Commands

```
# 1. Confirm TPM 2.0 device node exists in kernel
ls -l /dev/tpm*        # Expect /dev/tpm0 and /dev/tpmrm0

# 2. Check TPM specification version and manufacturer
cat /sys/class/tpm/tpm0/tpm_version_major   # Must output: 2

# 3. Verify Secure Boot and systemd TPM2 stack readiness
bootctl status | grep -E "Secure Boot|TPM2"
```

## 4. Distribution-Specific Setup Guides

### A. Arch Linux (LUKS2 + systemd-cryptenroll + Unified Kernel Image)

Arch Linux gives you full control via `systemd-cryptenroll`. Using a Unified Kernel Image (UKI) signed with `sbctl` ensures kernel updates do not break TPM unlocking.

1. **Install Required Packages:**

```
sudo pacman -S tpm2-tools tpm2-tss sbctl systemd-ukify
```

2. **Configure mkinitcpio for systemd-based initramfs:** Edit `/etc/mkinitcpio.conf` so `HOOKS` uses `systemd` instead of `udev`/`encrypt`:

```
HOOKS=(base systemd autodetect microcode modconf kms keyboard sd-vconsole block
sd-encrypt filesystems fsck)
```

   Regenerate your initramfs/UKI with `sudo mkinitcpio -P`.

3. **Set Up Custom Secure Boot Keys (if not already active):**

```
sudo sbctl create-keys
sudo sbctl enroll-keys --microsoft   # Keeps Microsoft keys so GPU/UEFI Option
ROMs still load
sudo sbctl sign-all                  # Signs your EFI bootloader and UKIs
```

4. **Enroll TPM 2.0 into Your LUKS2 Volume:** Replace `/dev/nvme0n1p2` with your encrypted root partition. Adding `--tpm2-with-pin=yes` adds a pre-boot PIN; omit it for unattended boot.

```
# Create a backup recovery key first!
sudo systemd-cryptenroll --recovery-key /dev/nvme0n1p2

# Bind LUKS2 keyslot to TPM2 PCR 7 (Secure Boot state)
sudo systemd-cryptenroll --tpm2-device=auto --tpm2-pcrs=7 /dev/nvme0n1p2
```

5. **Enable Auto-Unlock in Kernel Parameters / Crypttab:** Add `rd.luks.options=tpm2-device=auto` to your kernel command line (or `tpm2-device=auto` in `/etc/crypttab.initramfs`) and rebuild the UKI.

### B. Ubuntu Desktop (24.04 LTS / 25.10 / 26.04 LTS Snap-Integrated TPM FDE)

Ubuntu implements TPM-backed FDE directly inside its installer using signed kernel snaps and full-disk LUKS2 volumes.

1. **Pre-Install BIOS Preparation:** Boot into UEFI firmware settings, enable **UEFI Boot**, enable **Secure Boot**, and clear/enable **TPM 2.0**.
2. **Installer Disk Setup:** Boot the official Ubuntu Desktop ISO. On the *Disk setup* screen, choose **Erase disk and install Ubuntu → Advanced features → Use hardware-backed disk encryption**. (Note: Do not check "Install third-party drivers" during initial setup if your firmware lacks third-party Secure Boot keys, as unsigned DKMS modules conflict with Snap-backed TPM FDE.)
3. **Save Your Recovery Key:** Immediately save or scan the QR code for the 48-digit recovery key displayed post-install.
4. **Add a Post-Install PIN or Passphrase (Recommended):**

```
# View or manage recovery keys and PINs via CLI or the Ubuntu Security Center GUI
sudo snap recovery --show-keys
sudo snap install snap-tpmctl
sudo snap-tpmctl add-pin
```

### C. Fedora Workstation (LUKS2 + Dracut + systemd-cryptenroll)

Fedora uses `dracut` for initramfs generation and supports TPM2 unlocking out of the box on standard LUKS installations.

1. **Instruct Dracut to Include TPM2 Drivers:**

```
echo 'add_dracutmodules+=" tpm2-tss "' | sudo tee /etc/dracut.conf.d/tpm2.conf
```

2. **Enroll TPM 2.0 on Your LUKS Partition:**

```
sudo systemd-cryptenroll --tpm2-device=auto --tpm2-pcrs=7 /dev/nvme0n1p3
```

3. **Update `/etc/crypttab` and Rebuild Initramfs:** Append `,tpm2-device=auto` to the options column of your root LUKS entry in `/etc/crypttab` (e.g., `luks-UUID UUID=... none discard,tpm2-device=auto`), then run:

```
sudo dracut --force --regenerate-all
```

**Essential Best Practices & Recovery Hygiene**

- **Always Keep a Recovery Key:** Never wipe your manual passphrase slot after enrolling a TPM. If your motherboard dies or a firmware update alters PCRs unexpectedly, a recovery key stored offline is the only way to recover your data.
- **Re-Enrolling After Firmware Updates:** If a BIOS update changes PCR 0/7 and prompts for your passphrase at boot, log in, wipe the stale TPM slot (`sudo systemd-cryptenroll --wipe-slot=tpm2 /dev/sdX`), and re-run the enrollment command.