PRODUCTION DEPLOYMENT RUNBOOK • SINGLE-BOOT ARCHITECTURE

Arch Linux: TPM 2.0 FDE, Btrfs Snapshots & Encrypted Hibernation

End-to-end provisioning with LUKS2, systemd-boot, Unified Kernel Images (UKIs), sbctl Secure Boot, Snapper subvolumes, encrypted swapfile resume, and recovery diagnostics.

BOOTLOADER & PAYLOAD	ENCRYPTION & SEALING	FILESYSTEM & ROLLBACK	SUSPEND-TO-DISK
systemd-boot + Signed UKI	LUKS2 + TPM 2.0 (PCR 7)	Btrfs + Snapper Subvolumes	Encrypted Btrfs Swapfile

1. Architecture & Security Design

This stack eliminates unencrypted `/boot` partitions and fragile multi-partition swap layouts while supporting atomic filesystem rollbacks and secure hibernation:

- **Single Encrypted Container:** Only the EFI System Partition (`/efi`) is unencrypted, holding `systemd-boot` and signed Unified Kernel Images (`.efi`). Everything else—root, home, logs, and hibernation swap—resides inside a single LUKS2 volume.
- **Rolling-Release Safe PCR Binding:** Binding TPM 2.0 unsealing exclusively to `PCR 7` (Secure Boot policy & certificates) rather than `PCR 4` or `PCR 9` allows routine Arch kernel and initramfs upgrades without breaking auto-unlock, provided `sbctl` automatically re-signs every updated UKI.
- **Snapper-Ready Btrfs Layout:** Separating `@`, `@home`, `@log`, `@pkg`, and `@snapshots` prevents root rollbacks from reverting user data, logs, or package caches, while `@swap` isolates the hibernation swapfile from CoW snapshots.

> **Hibernation & Cold-Boot Threat Model Note**
>
> Hibernation writes active RAM contents (including the mounted LUKS2 volume key) into the encrypted swapfile. If an attacker steals a hibernated laptop, unattended `PCR 7` auto-unlock will decrypt the disk and resume RAM. For physical theft protection with hibernation enabled, enroll TPM 2.0 with a **pre-boot PIN** (`--tpm2-with-pin=yes`).

2. Partitioning, LUKS2 & Btrfs Subvolumes

Boot the Arch Linux live ISO in UEFI mode, verify network connectivity, and confirm TPM 2.0 presence (`ls /dev/tpmrm0`). Assume target drive `/dev/nvme0n1`:

```
# 1. Wipe partition table and create 1 GiB EFI + remaining LUKS2 root
sgdisk --zap-all /dev/nvme0n1
sgdisk -n 1:0:+1G -t 1:ef00 -c 1:"EFI" /dev/nvme0n1
sgdisk -n 2:0:0   -t 2:8309 -c 2:"cryptroot" /dev/nvme0n1

# 2. Format EFI System Partition
mkfs.fat -F 32 -n EFI /dev/nvme0n1p1

# 3. Format LUKS2 container with strong passphrase (Slot 0 fallback)
cryptsetup luksFormat --type luks2 --cipher aes-xts-plain64 --key-size 512   --hash sha256 --
pbkdf argon2id /dev/nvme0n1p2
cryptsetup open /dev/nvme0n1p2 cryptroot

# 4. Create Btrfs filesystem and subvolumes
mkfs.btrfs -L archroot /dev/mapper/cryptroot
mount /dev/mapper/cryptroot /mnt
for sub in @ @home @log @pkg @snapshots @swap; do
  btrfs subvolume create /mnt/$sub
done
umount /mnt
```

Subvolume Mount Layout

Subvolume	Mountpoint	Mount Options	Purpose
`@`	`/`	`noatime,compress=zstd:1,space_cache=v2`	Root filesystem (target for Snapper rollbacks)
`@home`	`/home`	`noatime,compress=zstd:1,space_cache=v2`	User files preserved across root snapshots
`@snapshots`	`/.snapshots`	`noatime,compress=zstd:1,space_cache=v2`	Snapper snapshot storage directory
`@log`	`/var/log`	`noatime,compress=zstd:1,space_cache=v2`	System & audit logs preserved across rollbacks
`@pkg`	`/var/cache/pacman/pkg`	`noatime,compress=zstd:1,space_cache=v2`	Pacman cache excluded from snapshot bloat
`@swap`	`/swap`	`noatime,nodatacow,space_cache=v2`	Dedicated non-CoW subvolume for hibernation swapfile

```
# Mount subvolumes and EFI partition
BTRFS_OPTS="noatime,compress=zstd:1,space_cache=v2"
mount -o subvol=@,$BTRFS_OPTS /dev/mapper/cryptroot /mnt
mkdir -p /mnt/{home,.snapshots,var/log,var/cache/pacman/pkg,swap,efi}

mount -o subvol=@home,$BTRFS_OPTS /dev/mapper/cryptroot /mnt/home
mount -o subvol=@snapshots,$BTRFS_OPTS /dev/mapper/cryptroot /mnt/.snapshots
mount -o subvol=@log,$BTRFS_OPTS /dev/mapper/cryptroot /mnt/var/log
mount -o subvol=@pkg,$BTRFS_OPTS /dev/mapper/cryptroot /mnt/var/cache/pacman/pkg
mount -o subvol=@swap,noatime,nodatacow /dev/mapper/cryptroot /mnt/swap
mount -o umask=0077 /dev/nvme0n1p1 /mnt/efi
```

3. Base Installation & Encrypted Btrfs Swapfile

Install base packages including Btrfs utilities, TPM 2.0 userspace tools, Secure Boot manager, and Snapper (replace `amd-ucode` with `intel-ucode` on Intel CPUs):

```
pacstrap -K /mnt base linux linux-firmware amd-ucode btrfs-progs   tpm2-tss sbctl snapper
snap-pac efibootmgr networkmanager vim

genfstab -U /mnt >> /mnt/etc/fstab
arch-chroot /mnt
```

Create Swapfile & Calculate Physical Resume Offset

Btrfs requires a dedicated non-CoW swapfile and exact physical block offset for kernel hibernation (`resume_offset=`). Size the swapfile to at least **100% of physical RAM** (e.g., `32g` for 32 GiB RAM):

```
# Inside arch-chroot: create 32 GiB Btrfs swapfile and enable in fstab
btrfs filesystem mkswapfile --size 32g --uuid clear /swap/swapfile
swapon /swap/swapfile
echo "/swap/swapfile none swap defaults 0 0" >> /etc/fstab

# Extract exact physical resume_offset for kernel cmdline
btrfs inspect-internal map-swapfile -r /swap/swapfile
# Example output: 533760  <-- Record this exact integer!
```

4. systemd-boot, UKI & Kernel Command Line

Obtain the UUID of the outer LUKS2 partition (`/dev/nvme0n1p2`) and configure the Unified Kernel Image cmdline with both LUKS unlocking and hibernation resume parameters:

```
# Get outer LUKS2 partition UUID (NOT /dev/mapper/cryptroot UUID)
LUKS_UUID=$(blkid -s UUID -o value /dev/nvme0n1p2)
OFFSET=$(btrfs inspect-internal map-swapfile -r /swap/swapfile)

# Write /etc/kernel/cmdline
mkdir -p /etc/kernel
cat <<EOF > /etc/kernel/cmdline
rd.luks.name=${LUKS_UUID}=cryptroot rd.luks.options=tpm2-device=auto root=/dev/mapper/
cryptroot rootflags=subvol=@ rw resume=/dev/mapper/cryptroot resume_offset=${OFFSET} quiet
loglevel=3
EOF
```

Configure systemd-based mkinitcpio & UKI Output

Edit `/etc/mkinitcpio.conf` to replace busybox hooks with `systemd` hooks (required for `systemd-cryptsetup` TPM 2.0 auto-unlock and `sd-vconsole`):

```
# /etc/mkinitcpio.conf (key lines)
MODULES=(btrfs amdgpu)   # Use i915 for Intel graphics
BINARIES=(/usr/bin/btrfs)
HOOKS=(base systemd autodetect microcode modconf kms keyboard sd-vconsole block sd-encrypt
filesystems fsck)
```

Edit `/etc/mkinitcpio.d/linux.preset` so `mkinitcpio` builds signed-ready UKIs directly onto the ESP and removes legacy unencrypted initramfs images:

```
# /etc/mkinitcpio.d/linux.preset
ALL_kver="/boot/vmlinuz-linux"
PRESETS=('default' 'fallback')

default_uki="/efi/EFI/Linux/arch-linux.efi"
default_options="--splash /usr/share/systemd/bootctl/splash-arch.bmp"

fallback_uki="/efi/EFI/Linux/arch-linux-fallback.efi"
fallback_options="-S autodetect"
```

```
# Install systemd-boot, generate UKIs, and clean legacy /boot images
mkdir -p /efi/EFI/Linux
bootctl install
mkinitcpio -P
rm -f /boot/initramfs-linux*.img
```

5. Custom Secure Boot Keys & Snapper Configuration

Before rebooting into firmware setup, configure Snapper on `/` while inside `arch-chroot` (avoiding Snapper's default subvolume conflict with our pre-created `@snapshots`):

```
# Initialize Snapper cleanly with pre-created @snapshots subvolume
umount /.snapshots
rm -rf /.snapshots
snapper --no-dbus -c root create-config /
btrfs subvolume delete /.snapshots
mkdir /.snapshots
mount -a
chmod 750 /.snapshots
```

Enroll Secure Boot Keys with sbctl

1. Reboot into UEFI Firmware Setup (`systemctl reboot --firmware-setup`).
2. Clear existing Secure Boot keys to enter **Setup Mode** (keep Secure Boot enabled or enable after enrollment).
3. Boot into your new Arch installation (using your LUKS passphrase for this first boot) and run:

```
# Verify Setup Mode is active
sbctl status

# Create custom keys and enroll with Microsoft certificates (prevents GPU/NIC Option ROM
bricking)
sbctl create-keys
sbctl enroll-keys --microsoft

# Sign systemd-boot and both Unified Kernel Images (-s saves paths to sbctl's pacman hook
database)
sbctl sign -s /efi/EFI/systemd/systemd-bootx64.efi
sbctl sign -s /efi/EFI/BOOT/BOOTX64.EFI
sbctl sign -s /efi/EFI/Linux/arch-linux.efi
sbctl sign -s /efi/EFI/Linux/arch-linux-fallback.efi

# Verify all binaries show "Signed"
sbctl verify
```

> **Automatic Kernel Signing on Updates**
>
> The `sbctl` package installs a pacman hook at `/usr/share/libalpm/hooks/zz-sbctl.hook`. Because you passed `-s` above, every future `pacman -Syu` that rebuilds a UKI will automatically re-sign it before reboot.

6. TPM 2.0 Enrollment & Recovery Key Hygiene

Reboot once more and confirm Secure Boot is active (`bootctl status` should report `Secure Boot: enabled (user)`). Next, generate an offline recovery key and bind LUKS2 to TPM 2.0 `PCR 7`:

```
# 1. Enroll an offline recovery key (print or store in a password manager—NEVER on the
encrypted drive)
systemd-cryptenroll --recovery-key /dev/nvme0n1p2

# 2A. Option A: Unattended TPM 2.0 auto-unlock (bound to Secure Boot PCR 7)
systemd-cryptenroll --wipe-slot=tpm2 --tpm2-device=auto --tpm2-pcrs=7 /dev/nvme0n1p2

# 2B. Option B (Recommended with Hibernation): TPM 2.0 + Pre-Boot PIN
systemd-cryptenroll --wipe-slot=tpm2 --tpm2-device=auto --tpm2-pcrs=7 --tpm2-with-pin=yes /
dev/nvme0n1p2

# 3. Inspect active LUKS2 keyslots (Slot 0: passphrase, Slot 1: recovery, Slot 2: tpm2)
cryptsetup luksDump /dev/nvme0n1p2
```

Reboot and test both cold boot (verifying TPM 2.0 unlocks `cryptroot`) and hibernation (`systemctl hibernate` followed by power-on resume).

7. Troubleshooting & Maintenance Checklist

Symptom / Event	Root Cause	Resolution
Prompted for passphrase after UEFI / BIOS firmware update	Firmware update altered Secure Boot variables or TPM measurement state (`PCR 7`)	Unlock with Slot 0 passphrase or recovery key, then re-seal: `sudo systemd-cryptenroll --wipe-slot=tpm2 --tpm2-device=auto --tpm2-pcrs=7 /dev/nvme0n1p2`
System reboots cleanly instead of resuming from `systemctl hibernate`	Incorrect `resume_offset=`, swapfile smaller than active RAM, or Secure Boot lockdown blocking unsigned hibernation image	Re-verify offset via `btrfs inspect-internal map-swapfile -r /swap/swapfile`, ensure swap ≥ RAM, and check `journalctl -b -1 -u systemd-hibernate.service`.
Boot fails with `Security Violation` after kernel update	UKI was rebuilt without running `sbctl` signing hook	Boot fallback UKI or live ISO, mount `/efi`, and run `sbctl sign-all`.
Rolled back a Snapper snapshot and now hibernation fails	`/etc/kernel/cmdline` inside snapshot had an outdated `resume_offset` if swapfile was recreated	Never recreate `/swap/swapfile` inside `@`; keep it isolated in `@swap` so its physical block offset never changes across rollbacks.
TPM lockout after multiple failed pre-boot PIN attempts	TPM 2.0 dictionary-attack (DA) protection triggered	Unlock using your Slot 0 LUKS passphrase, then clear TPM lockout counter via `tpm2_dictionarylockout -c`.

Pre-Production Verification Checklist

- [ ] `bootctl status` shows **Secure Boot: enabled (user)** and **TPM2 Support: yes**.
- [ ] `sbctl verify` shows green checkmarks for `systemd-bootx64.efi`, `BOOTX64.EFI`, and all UKIs in `/efi/EFI/Linux/`.
- [ ] `cryptsetup luksDump /dev/nvme0n1p2` confirms 3 active slots: Password (0), Recovery (1), and TPM2 token (2).
- [ ] Offline recovery key is stored safely outside the machine (printed or in an external vault).
- [ ] `snapper -c root list` creates a test snapshot without mounting errors, and `systemctl hibernate` resumes open desktop windows intact.