

Gentoo

- [Install Gentoo](#)

Install Gentoo

Full instructions for a Gentoo installation with UEFI + GRUB, a single boot/ESP partition, LUKS-encrypted root, initramfs, and OpenRC (desktop stage3).

This follows the official Gentoo Handbook and the Rootfs encryption wiki, adapted for your requirements. It assumes an amd64 system, a single disk (replace `/dev/nvme0n1` or `/dev/sda` with your actual device), and a fresh install from the official Gentoo LiveCD/USB.

Warning: This will destroy all data on the target disk. Back up anything important. Have a second machine or the Live environment ready for recovery. Test the password carefully.

Recommended partition layout (single boot partition)

```
/dev/nvme0n1 (or /dev/sda)
├─ p1  EFI System Partition (ESP)   ~1 GiB   FAT32   mounted at /boot
└─ p2  LUKS encrypted root         rest     → filesystem (ext4/xfs/btrfs)
```

The ESP holds GRUB, the kernel, and the initramfs. The root is fully encrypted.

1. Boot the Live environment and prepare the disk

Boot the official Gentoo Minimal or LiveGUI ISO. Configure networking if needed (`dhcpcd`, `wpa_supplicant`, etc.).

Identify the disk:

```
lsblk -f
```

Create a GPT partition table and the two partitions with `fdisk`:

```
fdisk /dev/nvme0n1
```

- `g` → create new GPT

- `n` → partition 1, default start, size `+1G`
- `t` → type `1` (EFI System)
- `n` → partition 2, default start/end (rest of disk)
- `t` → type `23` (Linux root x86-64) or leave as Linux filesystem
- `w` → write and exit

Format the ESP:

```
mkfs.vfat -F32 /dev/nvme0n1p1
```

2. Create and open the LUKS volume

```
modprobe dm-crypt
cryptsetup luksFormat --type luks2 --cipher aes-xts-plain64 --key-size 512 --hash sha512 --
pbkdf argon2id /dev/nvme0n1p2
```

(Type `YES` and enter a strong passphrase. Remember it — there is no recovery without it.)

Optional (recommended on SSDs):

```
cryptsetup luksHeaderBackup /dev/nvme0n1p2 --header-backup-file /tmp/root_headers.img
# Copy the header backup somewhere safe later
```

Open it:

```
cryptsetup luksOpen /dev/nvme0n1p2 root
```

(This creates `/dev/mapper/root`.)

Optional persistent discard (SSDs):

```
cryptsetup refresh --persistent --allow-discards root
```

3. Create the root filesystem

Choose one (examples):

ext4 (simple & reliable):

```
mkfs.ext4 -L rootfs /dev/mapper/root
```

xfs:

```
mkfs.xfs -L rootfs /dev/mapper/root
```

btrfs (popular for desktops, supports subvolumes/snapshots):

```
mkfs.btrfs -L rootfs /dev/mapper/root
```

Mount everything:

```
mkdir -p /mnt/gentoo
mount /dev/mapper/root /mnt/gentoo
mkdir -p /mnt/gentoo/boot
mount /dev/nvme0n1p1 /mnt/gentoo/boot
```

(If using btrfs subvolumes, create them now and mount the `@` subvolume as root.)

4. Download and extract the desktop OpenRC stage3

Find the current desktop OpenRC stage3:

```
# On the live system
links https://www.gentoo.org/downloads/ # or use wget/curl
# Look for stage3-amd64-desktop-openrc-*.tar.xz
```

Or directly (check mirrors for the newest):

```
cd /mnt/gentoo
wget https://distfiles.gentoo.org/releases/amd64/autobuilds/current-stage3-amd64-desktop-
openrc/stage3-amd64-desktop-openrc-*.tar.xz
# (use the exact current filename)
tar xpvf stage3-*.tar.xz --xattrs-include='*.*' --numeric-owner
```

5. Prepare for chroot

```
cp /etc/resolv.conf /mnt/gentoo/etc/
mount --types proc /proc /mnt/gentoo/proc
```

```
mount --rbind /sys /mnt/gentoo/sys
mount --make-rslave /mnt/gentoo/sys
mount --rbind /dev /mnt/gentoo/dev
mount --make-rslave /mnt/gentoo/dev
mount --bind /run /mnt/gentoo/run
mount --make-slave /mnt/gentoo/run
```

Enter the chroot:

```
chroot /mnt/gentoo /bin/bash
source /etc/profile
export PS1="(chroot) ${PS1}"
```

6. Portage configuration (common make flags)

Edit `/etc/portage/make.conf`:

```
# Common recommended flags
COMMON_FLAGS="-O2 -pipe -march=native"
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
FCFLAGS="${COMMON_FLAGS}"
FFLAGS="${COMMON_FLAGS}"

# Parallel jobs – adjust to your CPU/RAM (rough rule: min(nproc, RAM_GB/2))
MAKEOPTS="-j$(nproc) -l$(nproc)"

# Useful global USE flags for a desktop system (add/remove as needed)
USE="X alsa pulseaudio dbus elogind udev -systemd \
    opengl vulkan \
    -qt5 -kde gnome \           # or the opposite depending on DE
    networkmanager \
    crypt"

# CPU-specific (run after first emerge of app-portage/cpuid2cpuflags)
# CPU_FLAGS_X86="aes avx ..."
```

```
ACCEPT_LICENSE=""  
INPUT_DEVICES="libinput"  
VIDEO_CARDS="..." # amdgpu, nvidia, intel, etc.
```

Sync the tree and set the profile:

```
emerge-webrsync  
eselect profile list  
eselect profile set default/linux/amd64/23.0/desktop/openrc # or the current desktop openrc  
profile
```

7. Install essential packages

```
emerge --ask sys-fs/cryptsetup  
# If using btrfs: emerge --ask sys-fs/btrfs-progs  
echo "sys-kernel/linux-firmware linux-fw-redisistributable" >> /etc/portage/package.license  
emerge --ask sys-kernel/gentoo-sources sys-kernel/linux-firmware  
emerge --ask sys-boot/grub:2 sys-boot/efibootmgr  
# For initramfs generation (recommended modern options):  
emerge --ask sys-kernel/installkernel  
# Choose one generator:  
echo "sys-kernel/installkernel dracut grub" >> /etc/portage/package.use/installkernel  
emerge --ask sys-kernel/dracut # or ugrd
```

8. Kernel configuration

```
eselect kernel set 1  
cd /usr/src/linux  
make menuconfig
```

Critical options (search with ):

- Device Drivers → Multiple devices driver support → Device mapper support → Crypt target support
- Cryptographic API → enable AES, XTS, SHA-*, etc. (or use hardware crypto)
- File systems → the one you chose (ext4/xfs/btrfs)
- Enable EFI runtime services, EFI stub if desired

Build and install:

```
make -j$(nproc)
make modules_install
make install          # with installkernel this also generates the initramfs
```

9. Initramfs for LUKS

With Dracut (common): Create `/etc/dracut.conf.d/luks.conf`:

```
add_dracutmodules+=" crypt "
# or more complete: add_dracutmodules+=" crypt dm "
hostonly="yes"
```

Get the LUKS UUID:

```
blkid /dev/nvme0n1p2
```

Find the version of the kernel you actually built:

```
ls /lib/modules/
# or
ls /boot/vmlinuz*
```

Then create `/etc/dracut.conf.d/i18n.conf` with this content (correct for OpenRC):

```
# Correct mapping for Gentoo OpenRC
i18n_vars="/etc/conf.d/keymaps:keymap-KEYMAP,extended_keymaps-EXT_KEYMAPS
/etc/conf.d/consolefont:consolefont-FONT,consoletranslation-FONT_MAP /etc/rc.conf:unicode-
UNICODE"
```

Then either let `installkernel` generate it or run:

```
dracut --force --hostonly --kver 6.18.52-gentoo # or the installed kernel version
```

With UGRD (Gentoo-specific, often simpler for LUKS): It usually auto-detects; just ensure the `USE` flag is set and rebuild the initramfs via `installkernel`.

10. fstab and crypttab

`/etc/fstab` example:

```
# /etc/fstab: static file system information.
#
# <fs>                <mountpoint> <type>  <opts>                <dump/pass>

# EFI System Partition (your single boot partition)
UUID=XXXX-XXXX        /boot          vfat     umask=0077,noatime    0 2

# Encrypted root filesystem
UUID=yyyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyyy /  ext4  defaults,noatime  0 1
# (or xfs / btrfs – change the type accordingly)

# Optional: if you have a swap file or partition
# /dev/mapper/root-swap none  swap  sw  0 0
# or for a swap file:
# /swapfile            none  swap  sw  0 0
```

`/etc/crypttab` (for OpenRC):

```
root UUID=your-luks-uuid none luks
```

11. GRUB configuration

```
echo 'GRUB_PLATFORMS="efi-64"' >> /etc/portage/make.conf
emerge --ask sys-boot/grub

# Edit /etc/default/grub
GRUB_CMDLINE_LINUX="rd.luks.uuid=YOUR-LUKS-UUID root=/dev/mapper/root"
# or the older genkernel-style: crypt_root=UUID=... root=/dev/mapper/root
# Add other options as needed (quiet, splash, etc.)

grub-install --target=x86_64-efi --efi-directory=/boot --bootloader-id=Gentoo
grub-mkconfig -o /boot/grub/grub.cfg
```

12. Timezone

```
# List available timezones
ls /usr/share/zoneinfo
```

```
# Example for Europe/Berlin (change to your zone)
echo "Europe/Berlin" > /etc/timezone

# Apply it
emerg --config sys-libs/timezone-data
```

13. Locale

Generate the locales you want

```
nano /etc/locale.gen
```

Uncomment (or add) the locales you need. Common example:

```
en_US.UTF-8 UTF-8
de_DE.UTF-8 UTF-8
# add others if you want
```

Generate them:

```
locale-gen
```

Check they exist:

```
locale -a
```

Select the system locale

```
eselect locale list
```

You'll see something like:

```
Available targets for the LANG variable:
[1]  C
[2]  C.UTF-8
[3]  POSIX
[4]  en_US.UTF-8
[5]  de_DE.UTF-8
[ ]  (free form)
```

Set the one you want (example for English UTF-8):

```
eselect locale set 4
```

Reload the environment:

```
env-update && source /etc/profile
```

14. Hostname

```
# Set the hostname (replace "gentoo" with whatever you want)
echo "gentoo" > /etc/hostname

# Also set it in the OpenRC config
nano /etc/conf.d/hostname
```

12. Final system configuration

- Set root password: `passwd`
- Create a regular user: `useradd -m -G users,wheel,audio,video,plugdev -s /bin/bash youruser` then `passwd youruser`
- Timezone, locale, hostname as usual (handbook chapters)
- Install a desktop environment / display manager later after first boot
- For OpenRC + desktop: emerge `eelogind`, `dbus`, NetworkManager, etc., and enable services with `rc-update`

Exit chroot, unmount, and reboot:

```
exit
umount -l /mnt/gentoo/dev{/shm,/pts,}
umount -R /mnt/gentoo
cryptsetup luksClose root
reboot
```

On boot you should get a GRUB menu, then a passphrase prompt from the initramfs, then OpenRC.

Additional tips

- Keep a backup of the LUKS header in a safe place.

- After first boot, run `emerge --ask app-portage/cpuid2cpuflags` and add the resulting `CPU_FLAGS_X86=...` to `make.conf`, then rebuild world if desired.
- For a nicer desktop experience later: install your preferred DE (Plasma, GNOME, XFCE, etc.) and a display manager.
- Secure Boot is possible but requires extra signing steps (not covered here).
- Always consult the current Handbook and the Rootfs encryption page for updates:
<https://wiki.gentoo.org/wiki/Handbook:AMD64>
https://wiki.gentoo.org/wiki/Rootfs_encryption

This setup gives you a clean, single-boot-partition, LUKS-encrypted root with GRUB on UEFI and OpenRC. Enjoy the compile times!