

# Singular file system (LUKS, encrypted)

LUKS (Linux Unified Key Setup) is the standard for Linux hard disk encryption. By providing a standard on-disk-format, it does not only facilitate compatibility among distributions, but also provides secure management of multiple user passwords. LUKS stores all necessary setup information in the partition header, enabling to transport or migrate data seamlessly.

Management of LUKS encrypted devices is done via the `cryptsetup` utility.

**NOTE:** Why should you encrypt your data? Encryption ensures that no one but the rightful owner has access to the data. Encryption is therefore not only used to hide sensitive data from prying eyes, it also serves to protect your privacy. Encryption should be considered especially for portable devices such as laptops. In the event of loss or theft, encryption ensures that personal data and secrets (passwords, key files, etc.) do not fall into the wrong hands and are less likely and not as easily be abused.

The simplest, most basic encrypted partitioning scheme in a Linux operating system consists of 3 partitions:

| Type                 | File System | Description   |
|----------------------|-------------|---|
| EFI System Partition | vfat        | Stores boot loaders and bootable OS images in <code>.efi</code> format        |
| Root File System     | LUKS2       | Stores the Linux OS files (kernel, system libraries, applications, user data) |
| Swap                 | Plain       | Stores swapped memory pages from RAM during high memory pressure              |

This guide assumes the following:

- There is only 1 disk that needs partitioning
- `/dev/nvme0n1` is the primary disk

## Preparing the disk

Determine the disks that are installed on your system. This can easily be done with `fdisk`:

```
fdisk -l
```

It outputs a list of disk devices with one or more entries similar to this:

```
Disk /dev/nvme0n1: 232.89 GiB, 250059350016 bytes, 488397168 sectors
Disk model: Samsung SSD 840
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

The line starting the device file with `/dev/` is the relevant one. Start partitioning the disk with `cdisk`:

**WARNING:** Make sure you are modifying the correct device, else you *will* lose data!

```
cdisk /dev/nvme0n1
```

If the disk has no partition table yet, `cdisk` will ask you to specify one. The default partition table format for UEFI systems is `gpt`. Create a layout with at least 3 partitions:

| Size        | FS Type             |
|-------------|---------------------|
| 1G          | EFI System          |
| (RAM size)  | Linux Swap          |
| (remaining) | Linux root (x86-64) |

**NOTE:** Specifying the correct file system type allows some software to automatically detect and assign appropriate mount points to partitions. See [Discoverable Partitions Specification](#) for more details.

You can verify that the partitions have been created by running `fdisk -l` again:

```
Disk /dev/nvme0n1: 232.89 GiB, 250059350016 bytes, 488397168 sectors
Disk model: Samsung SSD 840
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
```

Disk identifier: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

| Device         | Start    | End       | Sectors   | Size   | Type                |
|----------------|----------|-----------|-----------|--------|---------------------|
| /dev/nvme0n1p1 | 2048     | 2099199   | 2097152   | 1G     | EFI System          |
| /dev/nvme0n1p2 | 2099200  | 35653631  | 33554432  | 16G    | Linux swap          |
| /dev/nvme0n1p3 | 35653632 | 488396799 | 452743168 | 215.9G | Linux root (x86-64) |

This time `fdisk` will also list the partitions present on the disk.

**NOTE:** You might notice a pattern with how Linux structures its block devices. Partitions also count as "devices" which you can interact with. Each partition has an incrementing counter attached to its name to specify its order in the partition layout.

## Formatting partitions

Before writing a file system to the disk a LUKS container needs to be created with the `cryptsetup` utility:

**WARNING:** Do **NOT** forget your passphrase! In case of loss you won't be able to access the data inside the container anymore!

```
cryptsetup luksFormat /dev/nvme0n1p3
```

Open the newly created LUKS container and supply the passphrase you just set:

**NOTE:** `cryptroot` is used as an example here. It is the "mapper name" under which the opened LUKS container will be available at, in this example: `/dev/mapper/cryptroot`. You may use whatever name you like.

```
cryptsetup open /dev/nvme0n1p3 cryptroot
```

## Formatting and mounting partitions

Create file systems for the ESP and the root file system:

```
mkfs.fat -F 32 /dev/nvme0n1p1  
mkfs.ext4 /dev/mapper/cryptroot
```

Mount the file systems:

```
mount /dev/mapper/cryptroot -o noatime /mnt  
mount --mkdir /dev/nvme0n1p1 /mnt/efi
```

**NOTE:** For an additional layer of security and privacy, swap space is going to be set up to be re-encrypted with a random passphrase on every boot in a later step. This way contents that have been swapped out of RAM and onto disk become inaccessible after the machine has been powered off.

---

Revision #3

Created 5 March 2024 16:39:31 by Sebin

Updated 3 February 2025 16:10:23 by Sebin