Universelles Offsite-Backup mit iSCSI

Tim Oberschulte

Chemnitzer Linux-Tage, 22. März 2025

Inhalt

Gründe für ein Backup

→ Wiederherstellen wichtiger Daten

- Versehentliches Löschen
- Hardwaredefekt (Festplatte/SSD)
- Verlust der Hardware
- Diebstahl von Hardware
- Kryptotrojaner (*Ransomware*)

Wichtige Daten sind, welche man nur mit viel Mühe oder gar nicht wiederherstellen kann: z. B. Fotos, Zeugnisse, Bescheinigungen, Dokumente in die viel Arbeit geflossen ist etc.

Das Offsite-Backup

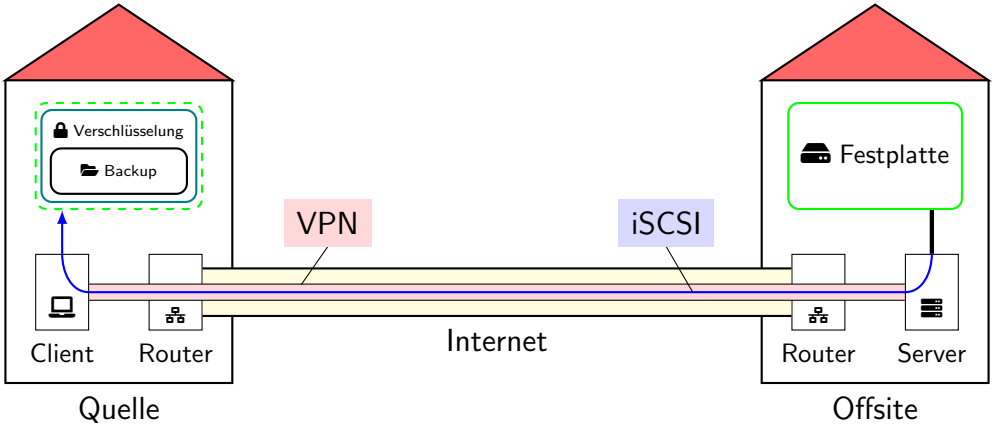
Siehe 3-2-1 Backup-Regel:

- 3 Kopien aller Daten
- 2 verschiedene Medien nutzen
- 1 an einem anderen Ort ←!

Warum an einem anderen Ort?

- Diebstahl der Hardware
- Abbrennen des Hauses
- Blitzeinschlag
- Großschadenslagen (z. B. Überflutung)

Setup


Verschlüsselung
Backup
Festplatte
VPN
iSCSI
Client
Router
Internet
Router
Server
Quelle
Offsite

Wahl eines geeigneten Standorts

- Möglichst weiter weg als beim Nachbarn
- Sicher und Vertrauenswürdig (Hardware kostet Geld)
- Trocken und kühl
- Zugänglich (wenn man es braucht muss man auch dran kommen)
- Schnelles Internet (je nach Datenmengen)

→ Zum Beispiel bei Freunden oder Familie

Verschlüsselung des Backups

Warum sollte ich (alle) meine Backups verschlüsseln?

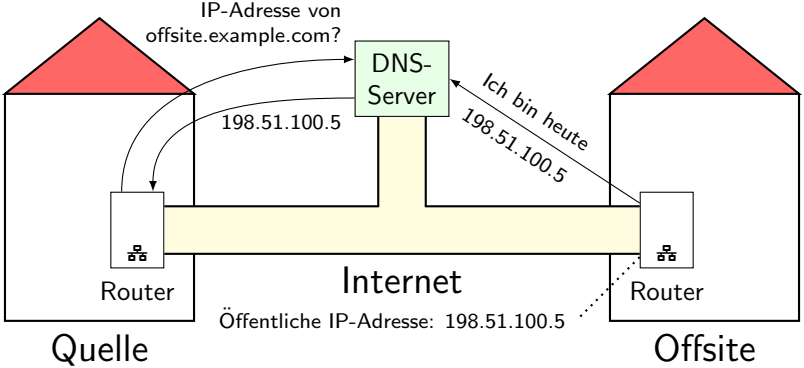
- Sicherheit der Daten bei physischem Diebstahl
- Zusätzliche Sicherheitsschicht bei der Datenübertragung über das Internet
- Glaubhafte Abstreitbarkeit von Wissen über die Festplatteninhalte durch Freunde und Familie

Hardware für den Server

- Raspberry Pi 4 mit 4 GB RAM ($\approx 60\,€$)
  - Raspberry Pi OS als Betriebssystem
- Gehäuse für den Raspberry Pi ($\approx 12\,€$)
- USB-C Netzteil ($\approx 10\,€$)
- Externe Festplatte (*USB 3.2 Gen 1* – 5 Gbit/s)
  - Seagate IronWolf Pro ($\approx 20\,€$/TB)
  - Western Digital Red ($\approx 25\,€$/TB)
  - USB Festplattengehäuse ($\approx 10\,€$)

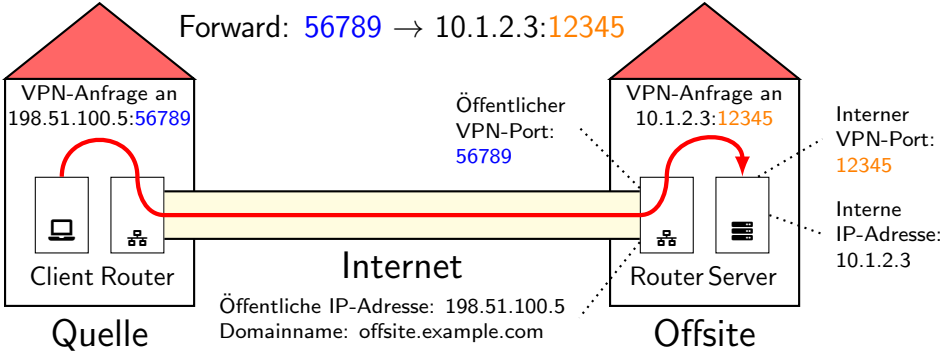
Domain einrichten

- Internetanschlüsse verwenden meist dynamische IP-Adressen
- Die öffentliche IP des Offsite-Routers muss bekannt sein
- Benötigt einen Vermittler, z. B. DynDns oder einen eigenen DNS Server



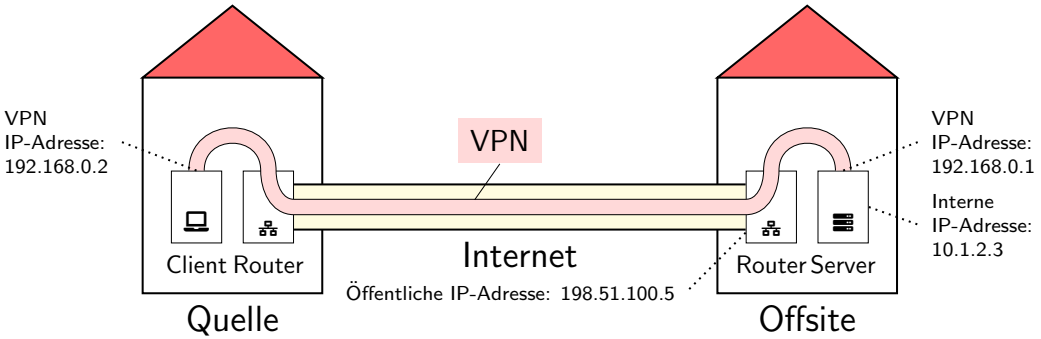
Portfreigabe

- Eingehende Verbindungen müssen im Offsite-Router konfiguriert werden
  - Statisches *Port Forwarding*
  - Inverses IP-*Masquerading*



VPN mit Wireguard – Netzwerkübersicht

- Aufbau eines Virtuellen privaten Netzwerks (VPN) zwischen Client und Server
- Beide Geräte haben danach eine VPN IP-Adresse



VPN mit Wireguard – Vorgehen

1. Installieren von Wireguard[1]
   - Raspberry Pi OS: `apt-get install wireguard wireguard-tools`
   - Debian/Ubuntu: `wireguard`
   - Fedora/Arch: `wireguard-tools`
2. Erstellen der Schlüsselpaare
3. Erstellen der Konfigurationsdateien
4. (Auto)Starten von Wireguard

[1] `https://www.wireguard.com/install/`

VPN mit Wireguard – Schlüsselpaare

Auf Client und Server erstellen wir einen *private* (geheimen) und einen *public* (öffentlichen) Schlüssel.

```
su
mkdir /etc/wireguard
cd /etc/wireguard
umask 077
wg genkey > privatekey
wg pubkey < privatekey > publickey
```

Die Datei `publickey` kopieren wir auf den je anderen Computer

VPN mit Wireguard – Konfigurationsdatei Server

/etc/wireguard/offsite.conf (Server)

```
[Interface]
Address = 192.168.0.1/24
PrivateKey = Server privatekey
ListenPort = 12345

[Peer]
PublicKey = Client publickey
AllowedIPs = 192.168.0.2/32
```

Eigene VPN IP-Adresse

Aus der Datei `privatekey` des Servers

Interner VPN-Port

Aus der Datei `publickey` des Clients

VPN IP-Adresse des Clients

VPN mit Wireguard – Konfigurationsdatei Client

/etc/wireguard/offsite.conf (Client)

```
[Interface]
Address = 192.168.0.2/24
PrivateKey = Client privatekey

[Peer]
PublicKey = Server publickey
AllowedIPs = 192.168.0.1/32
Endpoint = offsite.example.com:56789
```

Eigene VPN IP-Adresse

Aus der Datei privatekey des Clients

Aus der Datei publickey des Servers

VPN IP-Adresse des Servers

Offsite Domain

Öffentlicher VPN-Port

VPN mit Wireguard – Starten

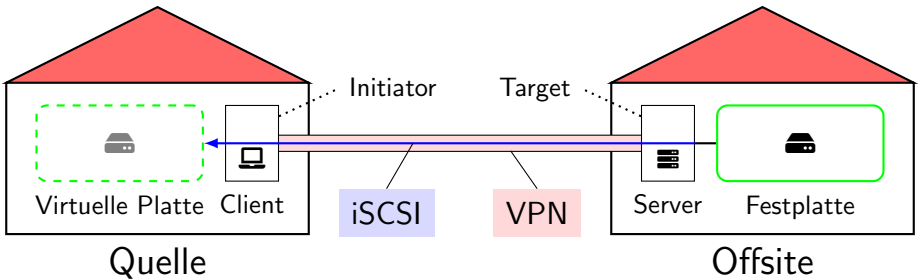
```
su
wg-quick up offsite   #Starten
wg-quick down offsite #Stoppen
wg show               #Status anzeigen

# Direkt beim Booten:
systemctl enable wg-quick@offsite.service
systemctl daemon-reload
```

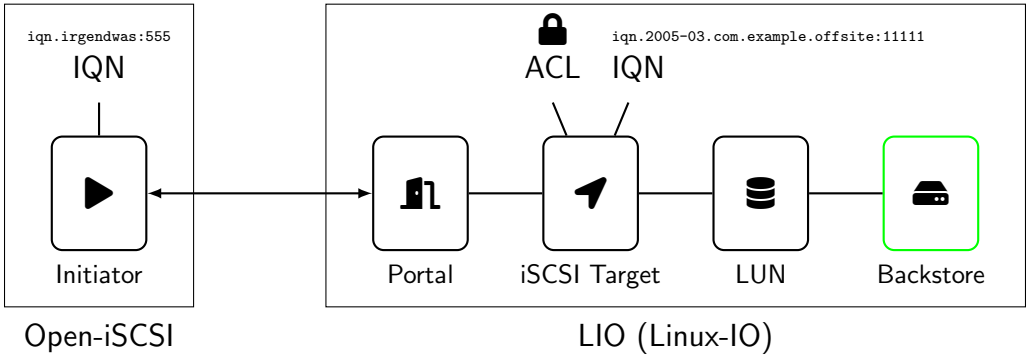
Name der Konfigurationsdatei
Hier: `/etc/wireguard/offsite.conf`

iSCSI – Einführung

- iSCSI leitet SCSI Befehle über das Netzwerk
- Der Client heißt *Initiator*
- Der Server heißt *Target*
- Die *Target* Festplatte erscheint als wäre sie am Client-PC angeschlossen



iSCSI – Nomenklatur



- LUN = Logical Unit Number: Logischer Speicherbereich
- ACL = Access Control List: Zugangsberechtigungen
- IQN = iSCSI Qualified Name: Eindeutiger Name

iSCSI – Client Konfiguration Teil 1

- Paket `open-iscsi` installieren
- Automatisch wird eine *initiator IQN* erstellt

```
systemctl enable --now iscsid.service
cat /etc/iscsi/initiatorname.iscsi
```

Inhalt von `/etc/iscsi/initiatorname.iscsi`

```
InitiatorName=iqn.irgendwas:555
```
← Für später merken

iSCSI – Server Kernel Teil 1

- Raspberry Pi OS fehlt das notwendige Kernelmodul
- Kernel mit dem Modul neu bauen[2]

Raspberry Pi 4

```
sudo apt install git bc bison flex libssl-dev make
git clone --depth=1 https://github.com/raspberrypi/linux
cd linux
KERNEL=kernel8
make bcm2711_defconfig
```

[2] Offizielle Anleitung:
https://www.raspberrypi.com/documentation/computers/linux_kernel.html

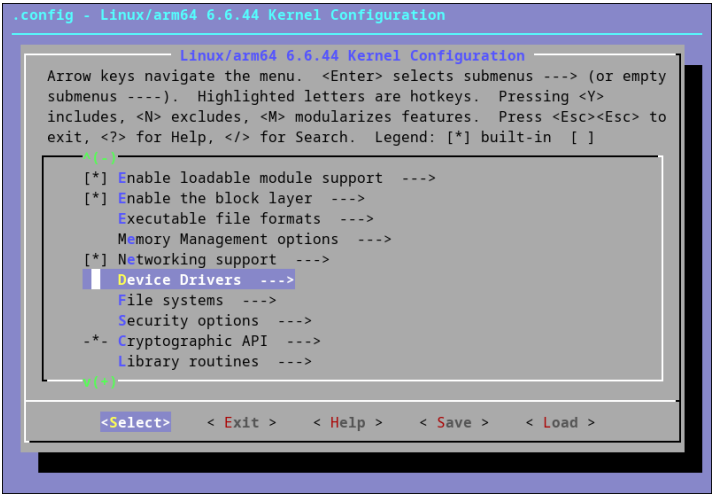
iSCSI – Server Kernel Teil 2

Raspberry Pi 4

```
make menuconfig
```

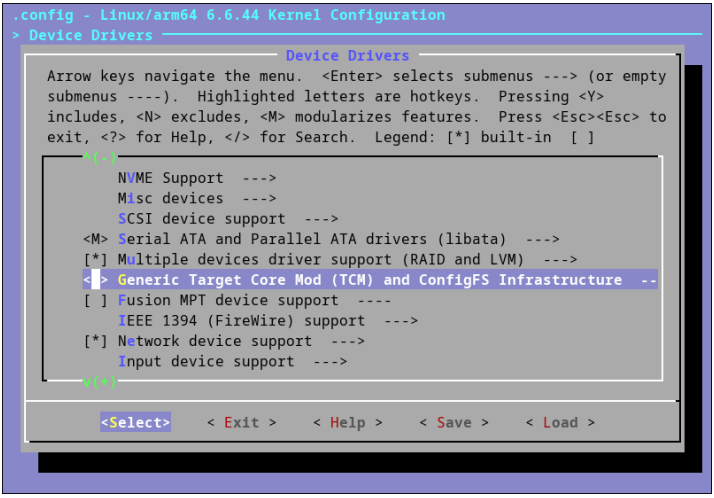
- Aktivieren des notwendigen Moduls *Generic Target Core Mod*
  - Zu `Device Drivers` navigieren (Pfeiltasten + Enter)
  - `Generic Target Core Mod (TCM) and ConfigFS Infrastructure` wählen und Leertaste drücken
- Optional: Dem Kernel einen Namen geben
  - Zu `General Setup` navigieren
  - `Local version - append to kernel release` setzen z. B. `-v7l-mit-iSCSI`

iSCSI – Server Kernel Teil 3



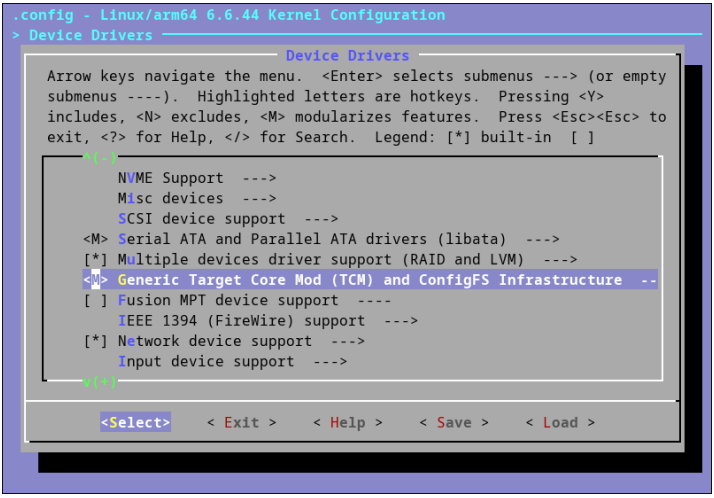
Enter

iSCSI – Server Kernel Teil 3



Leertaste

iSCSI – Server Kernel Teil 3



Speichern und beenden

iSCSI – Server Kernel Teil 4

- Bauen und Installieren des Kernels

Raspberry Pi 4

```
make -j6 Image.gz modules dtbs # Dauert sehr lange ;)
sudo make -j6 modules_install
sudo cp /boot/firmware/$KERNEL.img /boot/firmware/$KERNEL-backup.img
sudo cp arch/arm64/boot/Image.gz /boot/firmware/$KERNEL.img
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/firmware/
sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/firmware/overlays/
sudo cp arch/arm64/boot/dts/overlays/README /boot/firmware/overlays/
sudo reboot # Reboot damit der neue Kernel gestartet wird
```

- Nach dem Neustart überprüfen mit `uname -a`

iSCSI – Server Setup

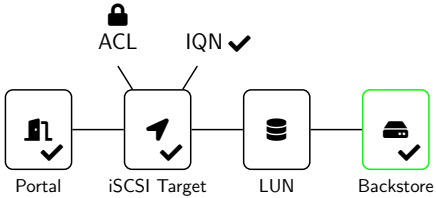
- Die Konfiguration von iSCSI erfolgt über `targetcli`
- Aufgebaut wie eine eigene kleine Shell

Raspberry Pi 4

```
su
apt install targetcli
targetcli
```

iSCSI – Server Setup targetcli 1

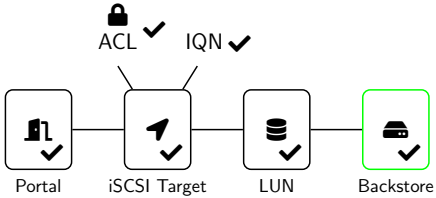
```
cd /backstores/block
create block0 /dev/sda
cd /backstores/block/block0
set attribute block_size=4096
cd /iscsi/
create iqn.2005-03.com.example.offsite:11111
cd /iscsi/iqn.2005-03.com.example.offsite:11111/tpg1/portals
create
```



Kommandos für `targetcli`
Keine normale Shell

iSCSI – Server Setup targetcli 2

```
cd /iscsi/iqn.2005-03.com.example.offsite:11111/tpg1/luns
create /backstores/block/block0
cd /iscsi/iqn.2005-03.com.example.offsite:11111/tpg1/acls
create iqn.irgendwas:555
cd /iscsi/iqn.2005-03.com.example.offsite:11111/tpg1
set attribute authentication=1
set auth userid=initiators_nutzername
set auth password=initiators_passwort
```



Kommandos für `targetcli`
Keine normale Shell

iSCSI – Client Konfiguration Teil 2

/etc/iscsi/iscsid.conf

```
node.session.auth.authmethod = CHAP
node.session.auth.username = initiators_nutzername
node.session.auth.password = initiators_passwort
```

```
su
iscsiadm -m discovery -p 192.168.0.1 -t sendtargets
iscsiadm -m node -T iqn.2005-03.com.example.offsite:11111 -l
# Hier das Backup machen
iscsiadm -m node -T iqn.2005-03.com.example.offsite:11111 -u
```

Server VPN IP-Adresse

Verschlüsselung aufsetzen

- Die Festplatte ist jetzt auf dem Client zu sehen als wäre sie normal angeschlossen
- Wir nehmen an die Festplatte ist `/dev/sdc` bzw. `/dev/disk/by-uuid/ABCD-EF12`
- Verschlüsselung über `dm-crypt` und LUKS[3]

```
su
cryptsetup luksFormat /dev/disk/by-uuid/ABCD-EF12
cryptsetup luksOpen /dev/disk/by-uuid/ABCD-EF12 offsite_disk
# Das Blockdevice liegt nun unter /dev/mapper/offsite_disk
cryptsetup luksClose offsite_disk
```

[3] https://wiki.archlinux.org/title/Dm-crypt/Device_encryption

Formatieren der verschlüsselten Disk

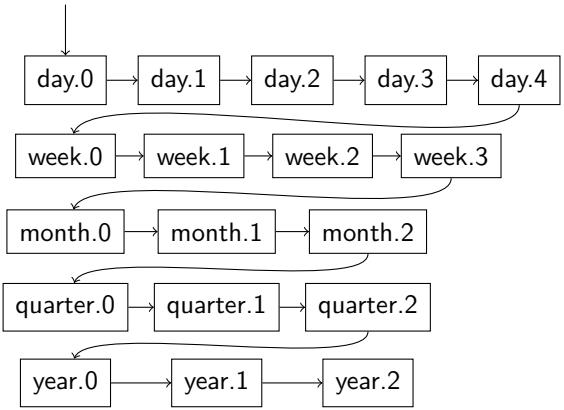
- Im entschlüsselten Laufwerk soll nun ein Dateisystem erstellt werden
- Als Beispiel hier das Dateisystem *Btrfs*

```
su
mkfs.btrfs /dev/mapper/offsite_disk

mkdir /mnt/offsite
mount /dev/mapper/offsite_disk /mnt/offsite
# Hier das Backup machen
umount /mnt/offsite
```

Vorschlag eines Backupprogramms

- Programm *rsnapshot* erzeugt inkrementelle Backups
- Stufen selbst festlegbar[4]
- Mehr zu Rsnapshot im Vortrag: *Alles backupen mit Rsnapshot* um 10:00 Uhr



[4] `https://wiki.archlinux.org/title/Rsnapshot`

Rsnapshot mit Btrfs

- Btrfs Subvolumes statt Ordner für die Kopien
  - Sehr schnelle Snapshots
  - RAID nicht über Btrfs (instabil)!

Skript für die Ausführung

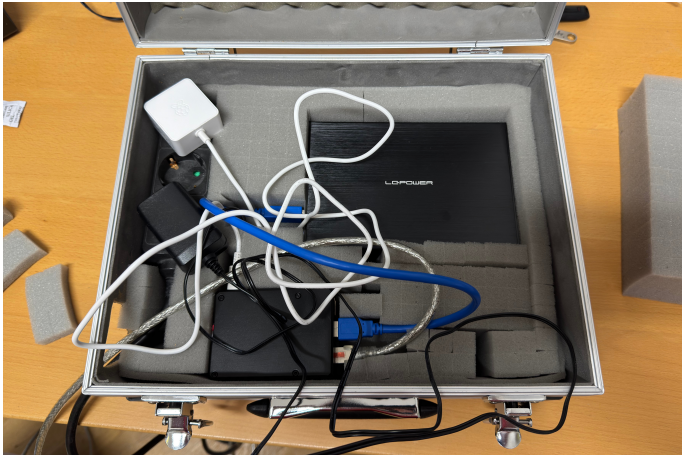
```
wg-quick up offsite
iscsiadm -m discovery -p 192.168.0.1 -t sendtargets
iscsiadm -m node -T iqn.2005-03.com.example.offsite:11111 -l
sleep 10 # Discovern der neuen Platte dauert einen Moment

cryptsetup luksOpen /dev/disk/by-uuid/ABCD-EF12 offsite_disk
mount /dev/mapper/offsite_disk /mnt/offsite
rsnapshot daily # Oder komplexere Aufrufe
umount /mnt/offsite
cryptsetup luksClose offsite_disk

iscsiadm -m node -T iqn.2005-03.com.example.offsite:11111 -u
wg-quick down offsite
```

Gehäuse

- Werkzeugkoffer aus dem Baumarkt mit zuschneidebarem Schaumstoff-Inlay
- Vibrationsarm
- Portabel



Verschiedenes

- Keine Lust auf Basteln?
  - NAS Systeme von Synology unterstützen auch iSCSI (ab $\approx 150\,€$)
  - Man kann scheinbar auch Wireguard darauf einrichten
  - Habe ich nicht getestet!
- Schaltbare Steckdose zum An- und Abschalten (z. B. über Home Assistant)
- Sichert eure LUKS-Header![5]
- Vorsicht bei der Verwendung von Keyfiles statt Passwörtern
- Firewall einrichten

[5] `https://wiki.archlinux.org/title/Dm-crypt/Device_encryption#Backup_and_restore`

Welche Alternativen gibt es?

- Festplatten manuell transportieren ("Sneakernet")
  - \+ Keine zusätzliche Hardware erforderlich
  - \+ Kein Netzwerksetup nötig
  - – Regelmäßiges Hin- und Herfahren
- Storage im Rechenzentrum ("Cloud")
  - \+ Schnelle Internetanbindung
  - – Je nach Festplattenkapazität deutlich teurer
  - – Physischer Zugriff nicht möglich (im Schadenfall)
- Fertige Backuplösungen z. B. Borg[6]
  - \+ Kein Setup von iSCSI, Verschlüsselung und Dateisystem
  - – Bringt ein eigenes Dateisystem mit (nicht KISS[7])

[6] `https://www.borgbackup.org/`

[7] Keep It Simple, Stupid (Mach es so einfach wie möglich)

Zusammenfassung

- Netzwerk einrichten (Server aus dem Internet erreichen)
- VPN aufsetzen (Kommunikation zwischen Client und Server)
- iSCSI einrichten (Festplatte vom Server erscheint beim Client)
- Festplatte verschlüsseln, formatieren und mounten
- Backups erstellen