Verschlüsselung unter Linux

Informationsquellen

LUKS:

- https://de.wikipedia.org/wiki/Dm-crypt
- https://wiki.immerda.ch/index.php/Hilfe:How-to:_Verschl%C3%BCsselung_mit_dm-crypt/LUKS
- http://linuxwiki.de/cryptsetup
- https://wiki.ubuntuusers.de/LUKS/
- https://wiki.ubuntuusers.de/LUKS/Containerdatei/

eCryptfs:

- http://ecryptfs.org/downloads.html
- https://de.wikipedia.org/wiki/ECryptfs
- https://wiki.ubuntuusers.de/ecryptfs/

EncFS

- https://wiki.archlinux.org/index.php/EncFS
- https://de.wikipedia.org/wiki/EncFS
- https://wiki.ubuntuusers.de/Howto/EncFS_in_der_Cloud/

Cryptomator

- https://cryptomator.org

Tools

Installation der notwendigen Software, hier `cryptsetup` und das kernel-Modul `dm-crypt`.

```
# Superuser werden
  sudo su
# cryptsetup installieren
  apt-get install cryptsetup
# verfügbare Verschlüsselungsmethoden des Kernels anzeigen
  cat /proc/crypto
```

LUKS - Linux Unified Key Setup

verschlüsselte Partition mit LUKS

erzeugen

```
sudo su
# verfügbare Partitionen anzeigen, z.B. auch eingesteckten USB-Stick
# hier wird /dev/sdb1 als Gerät gewählt
  lsblk
# Gerät mit Zufallsdaten überschreiben
  dd if=/dev/urandom bs=1M of=/dev/sdb1
# Gerät verschlüsseln
  cryptsetup luksFormat -c aes-xts-plain64 -s 512 -h sha512 -y /dev/sdb1
# Verschlüsselte Partition öffnen und ein sog. ''Mapper-Device'' zuweisen
  cryptsetup luksOpen /dev/sdb1 encrypted_usb_stick
# Wenn das Öffnen fehlerfrei war ist die Partition jetzt unter
''/dev/mapper'' sichtbar
  ls -l /dev/mapper/
# Jetzt kann die Partition ganz normal formatiert werden. Mittels der Option
''-m0'' wird statt 5% (default) 0% Speicherplatz für den Superuser
reserviert.
  mkfs.ext4 -m0 /dev/mapper/encrypted_usb_stick
# Label für die Partition vergeben
  tune2fs -L TI-2GB-ENC /dev/mapper/encrypted_usb_stick
# Partition einbinden
  mount /dev/mapper/usb-crypt /mnt
# Zugriffsrechte anpassen
  chown $USER:$USER /mnt
# Status der verschlüsselten Partition ermitteln
  cryptsetup status /dev/mapper/encrypted_usb_stick
```

einbinden

„On demand“, also nur bei Bedarf, per Klick im Dateimanager einbinden:

```
sudo su
cd /
# UUID der LUKS Partition ermitteln
  blkid /dev/sdb1
# Die Datei crypttab analog zu fstab anlegen. Sie wird von
"cryptdisks_[start|stop] ausgelesen und die darin genannten Partitionen
werden zum Einhängen unter /dev/mapper/ vorbereitet
  touch /etc/crypttab
# Einen Eintrag für die LUKS Partition hiunzufügen
  echo "encrypted_usb_stick UUID="ZUVOR_BESTIMMTE_ID" none luks,noauto" >>
/etc/crypttab
# Einen passenden Eintrag in der fstab erzeugen
  echo "/dev/mapper/encrypted_usb_stick /media/crypto ext4 noauto,nosuid,x-
gvfs-show,x-gvfs-name=LUKS-USB-Stick,x-gvfs-icon=Symname_Luks_stick 0 0" >>
/etc/fstab
```

verschlüssleter Container mit LUKS

erzeugen

```
# Superuser werden
  sudo su
# Containerdatei erzeugen, hier mit einer Größe von 250 MByte
  dd if=/dev/urandom of=luks_container bs=1M count=250
# Das nächste, frie Loop-Device zum einhängen der Containerdatei ermitteln
  losetup -f
  --> /dev/loopX
# Containerdatei zum öffnen vorbereiten in dem ein sog. Loop-Device erstellt
wird
  losetup /dev/loopX luks_container
# Containerdatei verschlüsseln
  cryptsetup luksFormat -c aes-xts-plain64 -s 512 -h sha512 -y /dev/loopX
# Verschlüsselte Containerdatei einhängen
  cryptsetup luksOpen /dev/loopX test_luks_container
# Dateisystem (hier ext4) innerhalb der verschlüsselten Containerdatei
erzeugen
  mkfs.ext4 /dev/mapper/test_luks_container
# Jetzt ist die verschlüsselte containerdatei fertig zur Einbindung in das
Dateisystem
# Mounten der verschlüsselten Containerdatei
  mount /dev/mapper/test_luks_container /mnt/MOUNTPOINT
```

einbinden

```
# Einen Eintrag für die LUKS Partition hiunzufügen
  echo "test_luks_container /home/<user>/tmp/luks_container none
luks,noauto" >> /etc/crypttab
# Einen passenden Eintrag in der fstab erzeugen
  echo "/dev/mapper/test_luks_container /mnt/crypto ext4 noauto,nosuid,x-
gvfs-show,x-gvfs-name=LUKS-USB-Stick,x-gvfs-icon=Symname_Luks_stick 0 0" >>
/etc/fstab
```

vergrößern

```
# Superuser werden
  sudo su
# Containerdatei vergrößern, hier um 100 MByte
  dd if=/dev/urandom bs=1M count=100 >> luks_container
# Container einhängen ohne das Dateisystem zu mounten
  losetup -f
  --> /dev/loop1
  losetup /dev/loop1 luks_container
  cryptsetup luksOpen /dev/loop1 test_luks_container
```

```
# gesamten verfügbaren Platz in der Containerdatei nutzbar machen
  cryptsetup resize test_luks_container
# Dateisystem an die neue größe anpassen, Dateisystemprüfung vorher und
nachher durchführen
  fsck.ext4 /dev/mapper/test_luks_container
  resize2fs -f/dev/mapper/test_luks_container
  fsck.ext4 /dev/mapper/test_luks_container
```

EncFS

Laut EncFS Security Audit {en} vom 14.01.2014 enthält EncFS in der Version 1.7.4 einige potentielle Schwachstellen. Das Fazit der Prüfung: EncFS ist wahrscheinlich noch sicher, solange ein potentieller Angreifer nur (genau) eine Version der verschlüsselten Daten erhält, wie z.B. bei Diebstahl oder Verlust eines Datenträgers. Kann ein potentieller Angreifer allerdings mehr als eine Version der verschlüsselten Daten einsehen, ist EncFS laut der Sicherheitsprüfung nicht mehr geeignet. Die verbreitete Verwendung von EncFS zur Verschlüsselung von Daten in der Cloud ist ein solcher Risikofall.
Inzwischen wurde die Version 1.8.2 veröffentlicht, leider ohne explizite Anmerkungen oder Hinweise im Changelog {en} bezüglich des Audits (Stand: November 2015). Aus einer Diskussion zur Sicherheit im github Repository {en} geht hervor, dass die kritische Lücke Issue #9 {en} ist und voraussichtlich in Version 2.0 gefixt wird. Eine Version 2.0 {en} wurde zwar angekündigt, aber bisher nicht veröffentlicht.
Quelle: https://wiki.ubuntuusers.de/EncFS/

```
# Superuser werden
  sudo su
# Installation
  apt-get install encfs cryptkeeper
# Gruppe "fuse" hinzufügen
  groupadd fuse
# aktuellen Benutzer der Gruppe "fuse" hinzufügen
  gpasswd -a <USER> fuse
# weiter als normaler Benutzer
  exit
# beliebiges verschlüsseltes Verzeichnis (hier z.B. ~/foobar bzw. das
versteckte Verzeichnisse ~/.foobar anlegen
  encfs ~/.foobar ~/foobar
# EncFS Verzeichnis aus einem Script heraus mounten
  encfs ~/.foobar ~/foobar --extpass'echo "<PASSWORD>"'
# oder auch so
  echo "<PASSWORD>" | encfs -S ~/.foobar ~/foobar
# oder auch so
  zenity --password | encfs -S ~/.foobar ~/foobar
# oder auch so
  zenity --title="EncFS Password" --width=300 --ok-label="Mount" --entry --
text="Please enter your EncFS Password" --hide-text | encfs -S ~/.foobar
~/foobar
# Verschlüsseltes Verzeichnis aushängen
```

```
  fusermount -u ~/foobar
# Gnome Encfs Manager installieren
  echo "deb http://ppa.launchpad.net/gencfsm/ppa/ubuntu xenial main" >>
/etc/apt/sources.list
  sudo apt-get update
  sudo apt-get install gnome-encfs-manager
```

eCryptfs

eCryptfs ist für die Verwendung mit dem HiDrive Angebot von Strato in Verbindung mit `rsync` nicht geeignet da eCryptfs die Datei- und Verzeichnisnamen auf bis zu 255 Zeichen erweitert. Dem seht die maximal zulässige Dateinamenlänge von 251 bzw. eine Pfadlänge von 1020 Zeichen entgegen.
Quelle: https://www.strato.de/faq/article/2031/Wie-richte-ich-rSync-ein.html

```
# Installation
  sudo apt-get install ecryptfs-utils
# weiter als normaler Benutzer
  exit
# Verschlüsseltes Standardverzeichnis ~/Private bzw. die versteckten
Verzeichnisse ~/.ecryptfs oder ~/.Private anlegen
  ecryptfs-setup-private
# Standardverzeichnis ändern
  mv .Private .Chr
  ln -l .Chr .Private
  mv Private Chr
  cp .ecryptfs/Private.nmt .ecryptfs/Private_orig.nmt
  echo "/home/<USER>/Chr" > .ecryptfs/Private.nmt
# alles wieder los werden
  ecryptfs-umount-private
  cd ~
  sudo su
  rm -rf Private .Private .ecryptfs
```

Cryptomator

Installation über eine externe Paketquelle - hier exemplarisch für Ubuntu 18.04:

```
sudo add-apt-repository ppa:sebastian-stenzel/cryptomator
sudo apt-get update
sudo apt-get install cryptomator
```