

# Daten verschlüsseln

mit GELI für FreeBSD 6.x

21. November 2007

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>GELI für FreeBSD</b>	<b>2</b>
2.1	Container vorbereiten . . . . .	2
2.2	Container erstellen . . . . .	3
2.3	Container öffnen/schließen . . . . .	4
2.4	Passwörter verwalten . . . . .	5
2.5	SWAP verschlüsseln . . . . .	5
2.6	System komplett verschlüsseln . . . . .	5

## 1 Einleitung

Dass die Verschlüsselung von Daten der Erhaltung einer Privatsphäre dient, bemerkt man spätestens, wenn ein USB-Stick verloren geht. Wird ein Laptop gestohlen, möchte man die private Fotosammlung sicher nicht im Internet sehen.

Investigative Journalisten, Rechtsanwälte und auch Priester haben das Recht und die Pflicht, Informationen über ihre Informanten zu schützen. Sie sollten sich frühzeitig Gedanken über ein Konzept zur Verschlüsselung machen. Eine gelesene und gespeicherte E-Mail unterliegt z.B. nicht mehr dem Postgeheimnis.

Diese Beispiele zeigen, dass unterschiedliche Anforderungen an eine Verschlüsselung bestehen können. Bevor man wild anfängt, alles irgendwie zu verschlüsseln, sollte man sich Gedanken über die Bedrohung machen, gegen die man sich schützen will:

1. **Schutz sensibler Daten** wie z.B. Passwortlisten, Revocation Certificates o.ä. erfordert die Speicherung in einem Container oder verschlüsselten Archiv, welches auch im normalen Betrieb geschlossen ist.
2. **Schutz aller persönlichen Daten** bei Verlust oder Diebstahl von Laptop oder USB-Stick erfordert eine Software, die transparent arbeitet ohne den Nutzer zu behindern und bei korrekter Anmeldung möglichst automatisch den Daten-Container öffnet (beispielsweise TrueCrypt für WINDOWS oder DM-Crypt für Linux).

3. **Backups auf externen Medien** enthalten in der Regel die wichtigen privaten Daten und sollten ebenfalls verschlüsselt sein. Dabei sollte die Wiederherstellung auch bei totalem Datenverlust möglich sein. Es ist nicht sinnvoll, die Daten mit einem PGP-Schlüssel zu chiffrieren, der nach einem Crash nicht mehr verfügbar ist.
4. Wer eine **Manipulation der Systemdaten** befürchtet, kann seinen Rechner komplett verschlüsseln (mit DM-Crypt für Linux oder GELI für FreeBSD) und von einem sauberen USB-Stick booten.
5. Zur **Herausgabe von Schlüsseln** im Fall einer Beschlagnahme des Rechners oder verschlüsselten Datenträgers gibt es immer wieder Missverständnisse. In Deutschland gelten folgende gesetzlichen Regelungen:
  - Richten sich die Ermittlungen gegen den Besitzer des Rechners oder Datenträgers muss man grundsätzlich keine Keys herausgeben.
  - Richten sich die Ermittlungen gegen Dritte, kann man die Herausgabe von Keys verweigern, wenn man sich auf das Recht zur Zeugnisverweigerung berufen oder glaubhaft(!) versichern kann, dass man sich damit selbst belasten würde. Im Zweifel sollte man einen Anwalt konsultieren.

In Großbritannien ist es bereits anders. Gemäß dem dort seit Oktober 2007 geltendem RIPA-Gesetz können Nutzer von Verschlüsselung unter Strafandrohung zur Herausgabe der Schlüssel gezwungen werden. Da diese Regelung nicht auf schwere Verbrechen beschränkt ist, kann man bei [Heise](#) nachlesen.

## 2 GELI für FreeBSD

GELI ist seit Version 6.0 fester Bestandteil von FreeBSD. Ähnlich wie DM-Crypt für Linux nutzt es ein generisches Framework zum Öffnen beliebiger Container als Device, ein Kernelmodul zur Verschlüsselung und ein Userland-Tool zur die Verwaltung der verschlüsselten Container.

GELI bietet folgende Feature:

- Es können komplette Slices, einzelne Partitionen und Dateien fester Größe als verschlüsselte Container genutzt werden.
- Für jeden Container können mehrere Keyfiles und/oder Passwörter genutzt werden.
- Verschiedene Algorithmen stehen zur Auswahl: AES, Blowfish, 3DES.
- Es ist möglich, dass System komplett zu verschlüsseln.

Bevor man mit dem Erstellen eines verschlüsselten Containers beginnt, sollte man sich wie üblich Gedanken über den Schlüssel machen. Eine Passphrase sollte einfach zu merken aber schwer zu erraten sein. Ein gutes zufälliges Keyfile lässt sich mit folgendem Kommando generieren:

```
> dd if=/dev/random of=key_01.key bs=128 count=1
```

## 2.1 Container vorbereiten

Je nach verwendetem Container sind verschiedenen Vorbereitungen nötig. Eine Containerdatei ist zu erstellen und als Loop-Device einzuhängen. Eine Partition oder ein gesamtes Slice ist mit zufälligen Daten zu überschreiben. Alle Schritte sind als *root* auszuführen:

### 1. Erstellen einer Containerdatei:

Als erstes ist eine leere Datei der gewünschten Größe zu erstellen (im Beispiel 1GB). Anschließend wird diese Datei als neues Device geöffnet und steht unter */dev/md1* für die Verschlüsselung zur Verfügung:

```
# dd if=/dev/zero of=geheim.f bs=1m count=1024
# mdconfig -a -t vnode -f geheim.f
```

### 2. Löschen einer Partition:

Am besten löscht man eine vorhandene Partition durch (mehrfaches?) Überschreiben mit zufälligen Daten. Dieser Vorgang kann einige Zeit in Anspruch nehmen.

```
# dd if=/dev/random of=/dev/ad0s1f
```

## 2.2 Container erstellen

Als Container wird im folgenden Beispiel das Loop-Device */dev/md1* genutzt. Wer eine Partition verschlüsseln möchte, muss in allen Kommandos *md1* durch *ad0s1f* oder ähnliches ersetzen.

### 1. Als erstes wird ein neuer GELI Provider initialisiert:

```
# geli init -l 256 -s 4096 -K key_01.key /dev/md1
Enter new passphrase:
Reenter new passphrase:
```

Der Parameter *-l 256* legt als Verschlüsselung AES256 fest, den stärksten zur Verfügung stehenden Algorithmus und mit *-s 4096* wird die Sektorgröße gegenüber dem Standard vergrößert, was zu einer Verbesserung der Performance führt.

Mit der Option *-K* wird ein Keyfile angegeben. Diese letzte Option kann entfallen, wenn nur eine Passphrase für den Zugriff verwendet werden soll. Die Nutzung einer Passphrase könnte mit dem Parameter *-p* deaktiviert werden.

Soll der Provider später beim Booten automatisch geöffnet werden, ist zusätzlich der Parameter *-b* anzugeben.

### 2. Als zweiten Schritt wird der GELI Provider geöffnet und dem GEOM-Framework unterstellt:

```
# geli attach -k key_01.key /dev/md1
Enter passphrase:
```

Der Name des GEOM Providers bekommt den Zusatz *.eli*, der Inhalt ist also als */dev/md1.eli* erreichbar.

3. Um zu verhindern, dass mögliche Schnüffler erahnen, an welcher Stelle im Container Daten gespeichert wurden, ist er komplett zu beschreiben. Ich denke, dass verschlüsselten Nullen nicht von Zufallszahl unterscheidbar sind, paranoider Nutzer können statt */dev/zero* auch */dev/random* als Datenquelle nutzen. Der Vorgang dauert dann wesentlich länger.

```
# dd if=/dev/zero of=/dev/md1.eli bs=1m
```

4. Wer eine Slice komplett verschlüsselt hat, muss an dieser Stelle mit dem Tool *bsdlabel* partitionieren. In diesem Beispiel oder bei Verschlüsselung einzelner Partitionen kann es entfallen und mit dem Anlegen eines neuen Filesystems fortgefahren werden:

```
# newfs /dev/md1.eli
```

5. Abschließend wird der Container wieder geschlossen:

```
# geli detach /dev/md1.eli
```

## 2.3 Container öffnen/schließen

Das Öffnen eines verschlüsselten Containers erfordert 2-3 Schritte als *root*:

1. Einbinden der Containerdatei. Dieser erste Schritt kann entfallen, wenn man eine verschlüsselte Partition oder Slice öffnet.
2. Öffnen des GELI Providers. Wichtig ist der Parameter *-d*. Er sorgt dafür, dass der Provider später automatisch mit dem *umount* wieder geschlossen wird.

```
# geli attach -d -k key_01.key /dev/md1
Enter passphrase:
```

Es ist das gleiche Keyfile, wie bei der Initialisierung anzugeben (allerdings mit kleinem *-k*!). Auch die Abfrage der Passphrase ist mit *-p* zu deaktivieren, wenn diese bei der Initialisierung nicht genutzt wurde.

3. Verzeichnis erstellen, Mounten des verschlüsselten Filesystems und setzen des Eigentümers:

```
# mkdir /home/<user>/geheim
# mount /dev/md1.eli /home/<user>/geheim
# chown -R <user> /home/<user>/geheim
```

Das Schließen des Containers geht wesentlich einfacher, als Vorschlag diesmal mit *sudo*:

```
> sudo umount /home/<user>/geheim
```

## 2.4 Passwörter verwalten

GELI kann für jeden Container mehrere Kombinationen von Keyfiles und Passphrase verwalten. Um einer weiteren Person Zugriff auf den Container zu gewähren kann eine zweite Schlüsselkombination hinzugefügt werden:

```
# geli setkey -n 1 -K key_02.key /dev/md1
Enter passphrase:
Enter new Passphrase:
Reenter new Passphrase:
```

Die Option `-K keyfile` kann entfallen, wenn der Zugriff nur mit einer Passphrase gesichert werden soll. Mit der Option `-P` kann die Abfrage der Passphrase deaktiviert werden.

Der Schlüssel kann später auch geändert werden. Das Beispiel zeigt die Änderung für die zweite Person:

```
# geli setkey -n 1 -k key_02.key -K key_02_new.key /dev/md1
Enter passphrase:
Enter new Passphrase:
Reenter new Passphrase:
```

Mit dem kleinen `-k` wird das alte Keyfile angegeben, mit großem `-K` das neue Keyfile. Gleichfalls ist mit kleinem `-p` die alte Abfrage der Passphrase zu deaktivieren und mit großem `-P` die zukünftige Abfrage der Passphrase.

Ein Löschen der Schlüssel ist mit folgendem Kommando möglich:

```
# geli delkey -n 1 /dev/md1
```

## 2.5 SWAP verschlüsseln

Das Verschlüsseln des SWAP-Bereiches geht ganz einfach:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
# geli onetime -d ad0s1b<br>
# swapon
```

## 2.6 System komplett verschlüsseln

Mit einem komplett verschlüsselten FreeBSD erhält man ein hochsicheres System. Der Aufwand ist allerdings beträchtlich.

Die FreeBSD Installations-CD enthält ein Live-System, welches für die Installation eines komplett verschlüsselten Systems genutzt werden. Nach dem Booten des Live-Systems ist ein symbolischer Link von `/dist/lib` nach `/lib` zu erstellen, falls es noch nicht existiert. Dann kann es losgehen.

1. Die Festplatte aufteilen und das für FreeBSD vorgesehene Slice mit `dd` reinigen. Ich verwende im folgenden `/dev/ad0` für die Installation von FreeBSD.

2. Das gesamte Slice verschlüsseln. Wichtig ist der Parameter `-b`, da der Container später beim Booten geöffnet werden soll. Auf ein Keyfile verzichte ich diesmal:

```
# geli init -b -l 256 -s 4096 /dev/ad0
```

3. Den Container öffnen und partitionieren. Für die Partitionierung ist das Tool `bsdlabel` zu verwenden!

```
# geli attach /dev/ad0
Enter passphrase:
# bsdlabel -w /dev/ad0s1.eli
# bsdlabel -e /dev/ad0s1.eli
```

Für die weiteren Schritte gehe ich davon aus, dass die Partitionen a (Root), b (swap) und d (/usr) angelegt wurden. Wer zusätzlich eigene Partitionen für /tmp und /var verwendet, muss die Befehle für das Anlegen der Dateisystem und zum mounten ergänzen.

4. Die Dateisysteme sind auf den Partitionen anzulegen (swap braucht kein Dateisystem):

```
# newfs /dev/ad0.elia
# newfs /dev/ad0.elid
```

5. Die Partitionen sind für die Installation einzubinden:

```
# mkdir /installation
# mount /dev/ad0.elia /installation
# mkdir /installation/usr
# mount /dev/ad0.elid /installation/usr
```

6. Dann kann die Installation beginnen. Es ist das Script `install.sh` zu nutzen, `sysinstall` funktioniert (noch) nicht.

```
# export DESTDIR=/installation/
# mount /cdrom
# cd /cdrom/6.1-RELEASE/base
# ./install.sh
```

7. Um das verschlüsselte System booten zu können, braucht man einen USB-Stick, welcher den usw. enthält. Wurde der Stick unter `/usbdisk` eingebunden, sind folgende Kommandos zur nötig:

```
# cp -Rpv /installation/boot /usbdisk
# cd /usbdisk/boot/kernel
# gzip kernel geom_eli.ko acpi.ko
```

8. Anschließend ist dem Kernel mitzuteilen, dass er das Modul `geom_eli.ko` beim Booten zu laden hat:

```
# echo geom_eli_load=\"YES\" >> /usbdisk/boot/loader.conf
```

9. Die Datei *fstab* der neuen Installation ist anzupassen, so dass beim Booten die verschlüsselten Partitionen zu laden sind. Der Inhalt meiner *fstab*:

```
/dev/ad0.elia /      ufs  rw 1 1  
/dev/ad0.elid /usr  ufs  rw 1 1  
/dev/ad0.elib none  swap sw 0 0
```

10. Diese Datei ist anschließend auch auf den USB-Stick zu kopieren:

```
# mkdir /usbdisk/etc  
# cp /installation/etc/fstab /usbdisk/etc/fstab
```

11. Jetzt ist es Zeit, das Live-System herunterzufahren und den Rechner mit dem vorbereiteten USB-Stick zu booten. Ist das neu installierte FreeBSD erwacht, kann die Installation mit *sysinstall* fortgesetzt werden. Eine Kopie des USB-Sticks sollte angelegt werden, da man bei Verlust des Sticks nicht mehr an seine Daten kommt.