Raspberry Workshop II

Im zweiten Teil des Raspberry Workshop geht es um:

- Filesysteme im Raspberry
- USB Sticks / Festplatten
    - Datenträger Partitionierung / Formatierung usw.
- Verschlüsselung

Filesysteme

Die wichtigstens FS

	FAT16	FAT32	NTFS	ext3	ext4
max Größe	4GB	8TB	256TB	32TB	1024PB
max Dateigröße	2GB	4GB	16TB	2TB	wie FS
max Datei Anzahl		268.435.456	4.294.967.295	Variabel	Variabel
Linux	Ja	Ja	bedingt	Ja	Ja

Sind einfach nur die Arten, wie die Daten auf dem Datenträger abgelegt werden.

Befehle:

- Dmesg
- Fdisk
- Mount
- Umount
- Mkfs (mkfs.msdos oder mkfs.vfat) stellt Filesystem her

Die folgenden, in Anführungszeichen stehenden Zeilen sind Eingabebefehle:

“df –h“ (listet die Speicherbereiche und Laufwerke auf)

```
pi@DF9PM:~ $ df -h
Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
/dev/root        13G    3,9G  8,4G   32% /
devtmpfs        459M       0  459M    0% /dev
tmpfs           463M       0  463M    0% /dev/shm
tmpfs           463M    6,4M  457M    2% /run
tmpfs           5,0M    4,0K  5,0M    1% /run/lock
tmpfs           463M       0  463M    0% /sys/fs/cgroup
/dev/mmcblk0p6   63M     21M   43M   33% /boot
tmpfs            93M       0   93M    0% /run/user/1000
/dev/sda1        15G    8,0K   15G    1% /media/pi/0201-AB0E
/dev/mmcblk0p5   30M    397K   28M    2% /media/pi/SETTINGS
/dev/sdb1       7,3G    606M  6,7G    9% /media/pi/USB DISK
pi@DF9PM:~ $
```

„Umount (sudo umount /media/pi/Stickname“ (hier 0201-AB0E)

```
pi@DF9PM:~ $ sudo umount /media/pi/0201-AB0E
pi@DF9PM:~ $
```

Das Laufwerk wird aus dem Systemzugriff genommen.
Keine Antwort ist eine gute Antwort - das heißt Befehl ausgeführt.

„df –h“ noch einmal schauen, ob der Stick weg ist.

```
Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
/dev/root        13G    3,9G  8,4G   32% /
devtmpfs        459M       0  459M    0% /dev
tmpfs           463M       0  463M    0% /dev/shm
tmpfs           463M    6,3M  457M    2% /run
tmpfs           5,0M    4,0K  5,0M    1% /run/lock
tmpfs           463M       0  463M    0% /sys/fs/cgroup
/dev/mmcblk0p6   63M     21M   43M   33% /boot
tmpfs            93M       0   93M    0% /run/user/1000
/dev/mmcblk0p5   30M    397K   28M    2% /media/pi/SETTINGS
pi@DF9PM:~ $
```

Wir sehen – der Stick (sda) wird nicht mehr angezeigt – alles OK.

Fdisk (Achtung - FDISK ist mächtig. Damit auch gefährlich)

```
Usage:
 fdisk [options] <disk>      change partition table
 fdisk [options] -l [<disk>] list partition table(s)

Options:
 -b, --sector-size <size>      physical and logical sector size
 -c, --compatibility[=<mode>]  mode is 'dos' or 'nondos' (default)
 -L, --color[=<when>]          colorize output (auto, always or never)
 -l, --list                    display partitions end exit
 -t, --type <type>             recognize specified partition table type only
 -u, --units[=<unit>]          display units: 'cylinders' or 'sectors' (default)
 -s, --getsz                   display device size in 512-byte sectors [DEPRECAT
ED]

 -C, --cylinders <number>      specify the number of cylinders
 -H, --heads <number>          specify the number of heads
 -S, --sectors <number>        specify the number of sectors per track

 -h, --help     display this help and exit
 -V, --version  output version information and exit

For more details see fdisk(8).
pi@DF9PM:~ $
```

„fdisk –l /dev/sda“ (ich bekomme die Formatierung des Sticks gezeigt)

```
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: dos
Disk identifier: 0xcce26010

Device         Boot   Start      End  Sectors  Size Id Type
/dev/mmcblk0p1         2048  2285156  2283109  1,1G  e W95 FAT16 (LBA)
/dev/mmcblk0p2      2285157 30318591 28033435 13,4G  5 Extended
/dev/mmcblk0p5      2285568  2351101    65534   32M 83 Linux
/dev/mmcblk0p6      2351104  2480127   129024   63M  c W95 FAT32 (LBA)
/dev/mmcblk0p7      2482176 30318591 27836416 13,3G 83 Linux

Disk /dev/sda: 14,5 GiB, 15518924800 bytes, 30310400 sectors
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: dos
Disk identifier: 0xe83f87e8

Device     Boot Start      End  Sectors  Size Id Type
/dev/sda1        512 30310399 30309888 14,5G  b W95 FAT32

pi@DF9PM:~ $
```

Hier sehe ich den Stick, der Fat32 formatiert ist und mit SDA1 gemountet ist.

„fdisks –t“ (ich bekomme die Liste angezeigt)

```
Usage:
 fdisk [options] <disk>      change partition table
 fdisk [options] -l [<disk>] list partition table(s)

Options:
 -b, --sector-size <size>      physical and logical sector size
 -c, --compatibility[=<mode>]  mode is 'dos' or 'nondos' (default)
 -L, --color[=<when>]          colorize output (auto, always or never)
 -l, --list                    display partitions end exit
 -t, --type <type>             recognize specified partition table type only
 -u, --units[=<unit>]          display units: 'cylinders' or 'sectors' (default)
 -s, --getsz                   display device size in 512-byte sectors [DEPRECAT
ED]

 -C, --cylinders <number>      specify the number of cylinders
 -H, --heads <number>          specify the number of heads
 -S, --sectors <number>        specify the number of sectors per track

 -h, --help     display this help and exit
 -V, --version  output version information and exit

For more details see fdisk(8).
pi@DF9PM:~ $
```

Hier sieht man nun die Möglichkeiten von FDISK

„sudo fdisk /dev/sda“

```
pi@DF9PM:~ $ sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 14,5 GiB, 15518924800 bytes, 30310400 sectors
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: dos
Disk identifier: 0xe83f87e8

Device     Boot Start      End  Sectors  Size Id Type
/dev/sda1        512 30310399 30309888 14,5G  b W95 FAT32


Command (m for help):
```

Wir gebe den Befehl nochmals ein.
Grund ist die Anzeige in der Commandzeile.

Hier erfolgt nun die Eingabe:

„m“

```
  d   delete a partition
  l   list known partition types
  n   add a new partition
  p   print the partition table
  t   change a partition type
  v   verify the partition table

 Misc
  m   print this menu
  u   change display/entry units
  x   extra functionality (experts only)

 Save & Exit
  w   write table to disk and exit
  q   quit without saving changes

 Create a new label
  g   create a new empty GPT partition table
  G   create a new empty SGI (IRIX) partition table
  o   create a new empty DOS partition table
  s   create a new empty Sun partition table


Command (m for help):
```

Nach der Eingabe von „m“ zeigt fdisk die Möglichkeiten an, welche nun zu Verfügung stehen.

Eingabe - d

```
Command (m for help): d
Selected partition 1
Partition 1 has been deleted.
```

Die Partition ist gelöscht.

Eingabe „n“

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p):
```

Es wird nun gefragt, ob die zu erstellende Partition „primary“ oder „extended“ sein soll.

Eingabe „p“

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-30310399, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-30310399, default 30310399): +7G

Created a new partition 1 of type 'Linux' and of size 7 GiB.

Command (m for help):
```

Mit der Eingabe von „p“ wird nun eine Primary Partition erstellt.
Danach wird vom System gefragt:

1. Startsektor (2048 – lassen wir so). Einfach ENTER drücken.

   Bedeutet, dass ab dem Block 2048 die Partition beginnt. Die freien Blöcke davor sind für andere Funktionen (z.B. Boot USB Stick, usw.) vorgesehen.

2. Danach wird die Größe der Partition angegeben.

   In unserem Beispiel sind das 7GB
   Es erfolgt die Eingabe +7GB

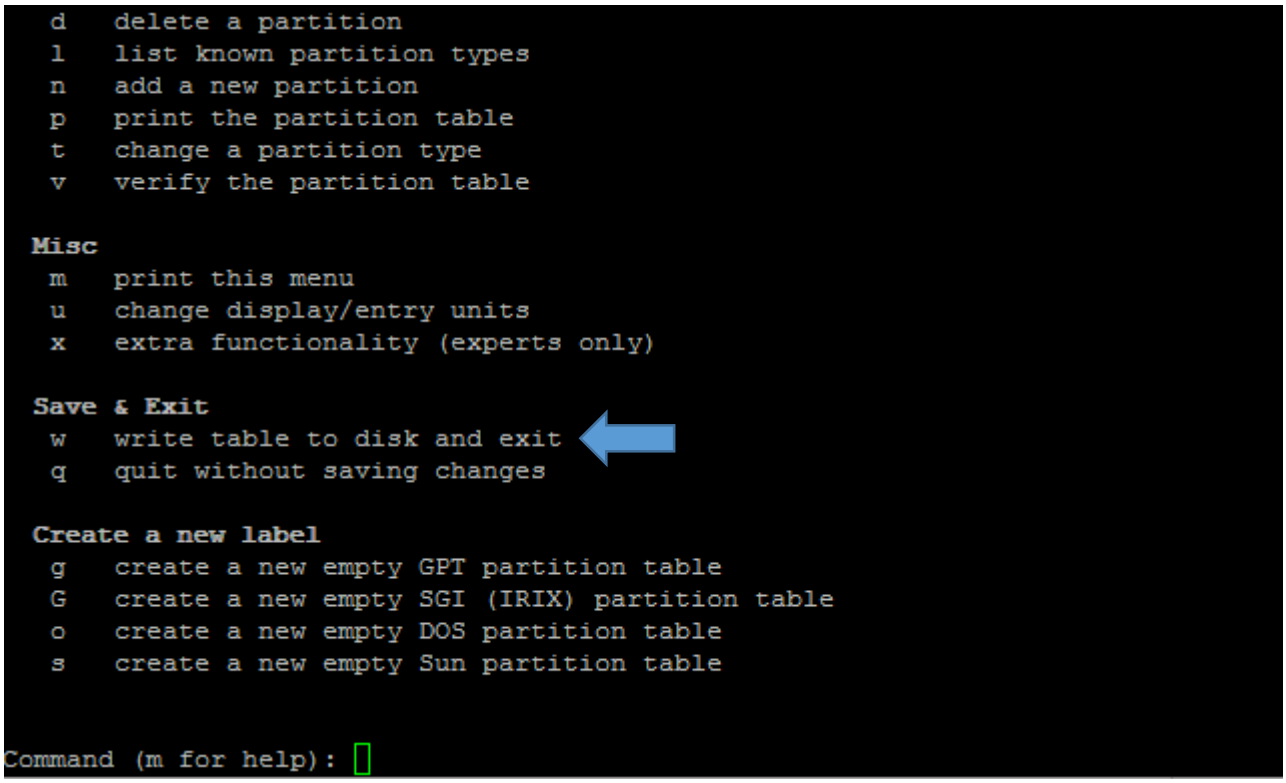
Eingabe „p“

```
Command (m for help): p
Disk /dev/sda: 14,5 GiB, 15518924800 bytes, 30310400 sectors
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: dos
Disk identifier: 0xe83f87e8

Device     Boot Start      End  Sectors Size Id Type
/dev/sda1        2048 14682111 14680064   7G 83 Linux


Command (m for help):
```

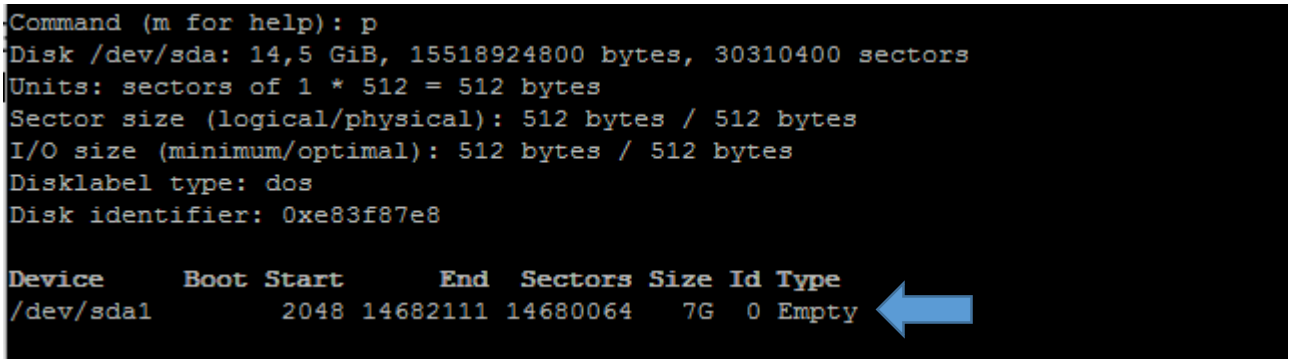
Hier wird einfach das Medium neu abgefragt und man sieht nun die neue Größe der ersten Partition von 7GB.

Bis zur diesem Zeitpunkt ist noch nichts auf dem Stick passiert.
D.h es müssen die Systemoperationen noch ausgeführt werden.
Dazu wir nun ein „w“ eingegeben.

Eingabe „w“



Nun sieht man nach der Eingabe „p“ folgendes:



Der Type der Partition ist im Moment noch „Empty“!
Das müssen wir ändern.

Dazu erfolgt nun die Eingabe „t“ und wir bekommen folgendes angezeigt.

Eingabe – t gefolgt mit der Eingabe „l“

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): l

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix
Hex code (type L to list all codes): b
If you have created or modified any DOS 6.x partitions, please see the fdisk doc
umentation for additional information.
Changed type of partition 'Empty' to 'W95 FAT32'.
```

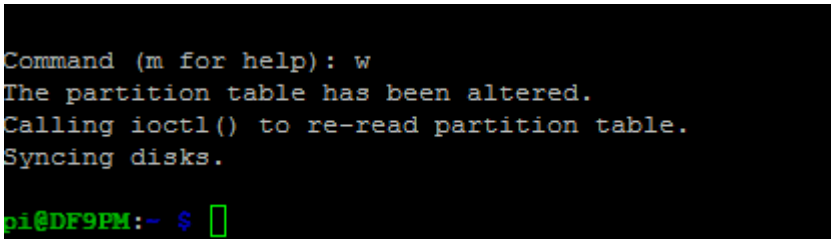
Man sieht hier im Anschluss nun die Partitionstype Liste und das System fragt, welche man auswählen will.
Wir nehmen „b“ für W95 FAT32.

Also Eingabe „b“

Gibt man nun erneut „p“ ein, sieht man die Veränderung des Type!

```
Command (m for help): p
Disk /dev/sda: 14,5 GiB, 15518924800 bytes, 30310400 sectors
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: dos
Disk identifier: 0xe83f87e8

Device     Boot Start      End  Sectors Size Id Type
/dev/sda1        2048 14682111 14680064   7G  b W95 FAT32


Command (m for help):
```

Nach der Eingabe von „w“ wird die Partition neu geschrieben und der Stick zeigt danach eine Partition mit 7GB an.
Eingabe „w“

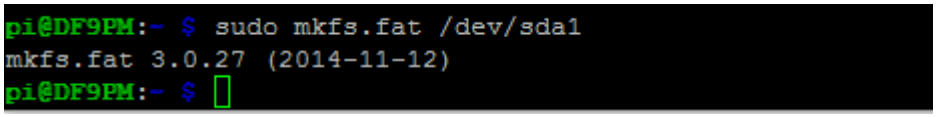


Jetzt kann man das Ganze wiederholen und eine zweite Partition (Restkapazität) erstellen. Die Vorgehensweise ist die Gleiche.

Achtung – wir haben nun nur die Partition(s) erstellt.
Der Stick ist noch nicht formatiert und kann noch nicht benutzt werden.

Dazu ist es notwendig, nun den Stick, bzw. die Partition noch zu formatieren.

Eingabe - „sudo mkfs.fat /dev/sda1“



Denkt dran – keine Nachricht ist eine gute Nachricht und
nach einem Moment ist der Stick nun formatiert.
Man muss nun den Stick abziehen und neu stecken.
Danach sieht man die Partition(s) mit ihren neuen Größen!

Diese Prozedur gilt natürlich nicht nur für USB Sticks, sondern auch für externe Festplatten!

Das war’s.

Weiter geht’s.

Verschlüsselung von Daten im Raspberry

Hier die Verschlüsselungsarten:

Symetrisch verschlüsseln mit GNUPG

Verschhlüsselungsalgorithmen:

- 3DES, (auch Triple-DES genannt) ist eine verstärkte Version des DES-Algorithmus.
- CAST5
- BLOWFISH, vom Sicherheitsguru Bruce Schneier entworfen.
- AES, benutzt 128 Bit Blockgröße (je größer die AES-Blockgröße, desto sicherer).
- AES192, AES mit 192 Bit Blockgröße.
- AES256, AES mit 256 Bit Blockgröße.
- TWOFISH, die Nachfolge-Version von Blowfish.

Die Algorithmen sind alle recht sicher und nehmen sich nicht viel. Der Unterschied für den Benutzer ergibt sich meist nur in der Geschwindigkeit, mit der die Algorithmen verschlüsseln.

Wer auf der sicheren Seite sein will, wird sich vermutlich für AES256 oder Twofish entscheiden.

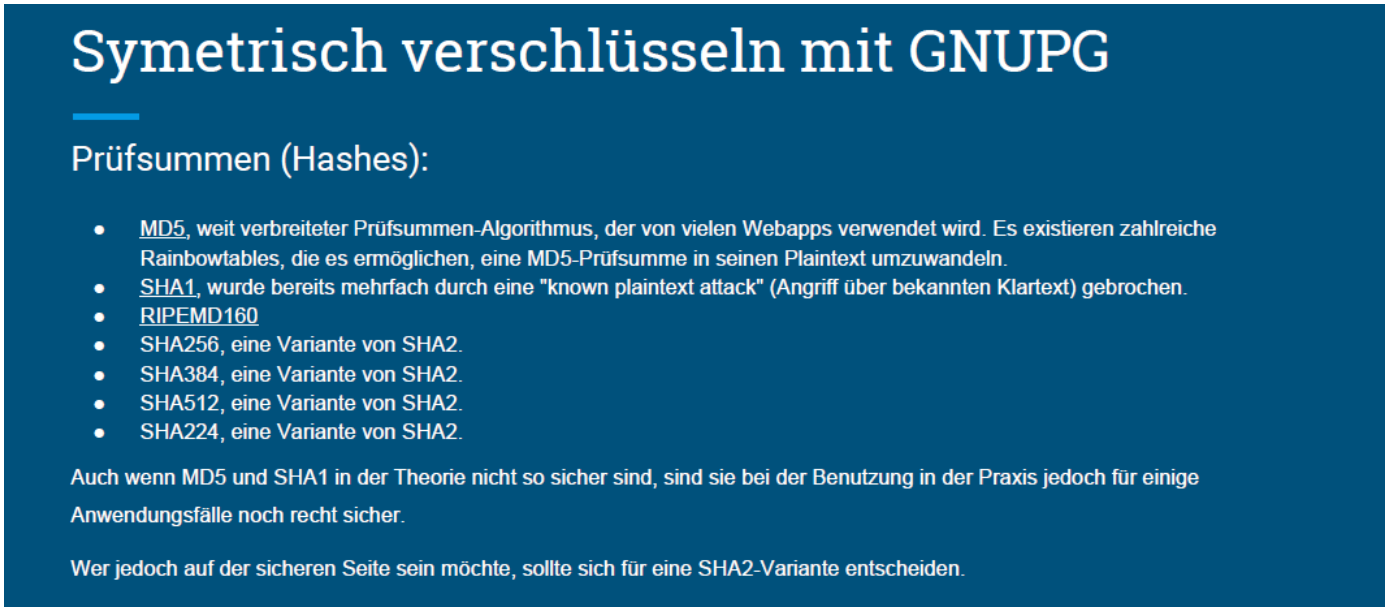
Auf Eurem Raspberry die ist das bereits installiert und man kann dies abfragen. Dazu

Eingabe - „gpg --version“

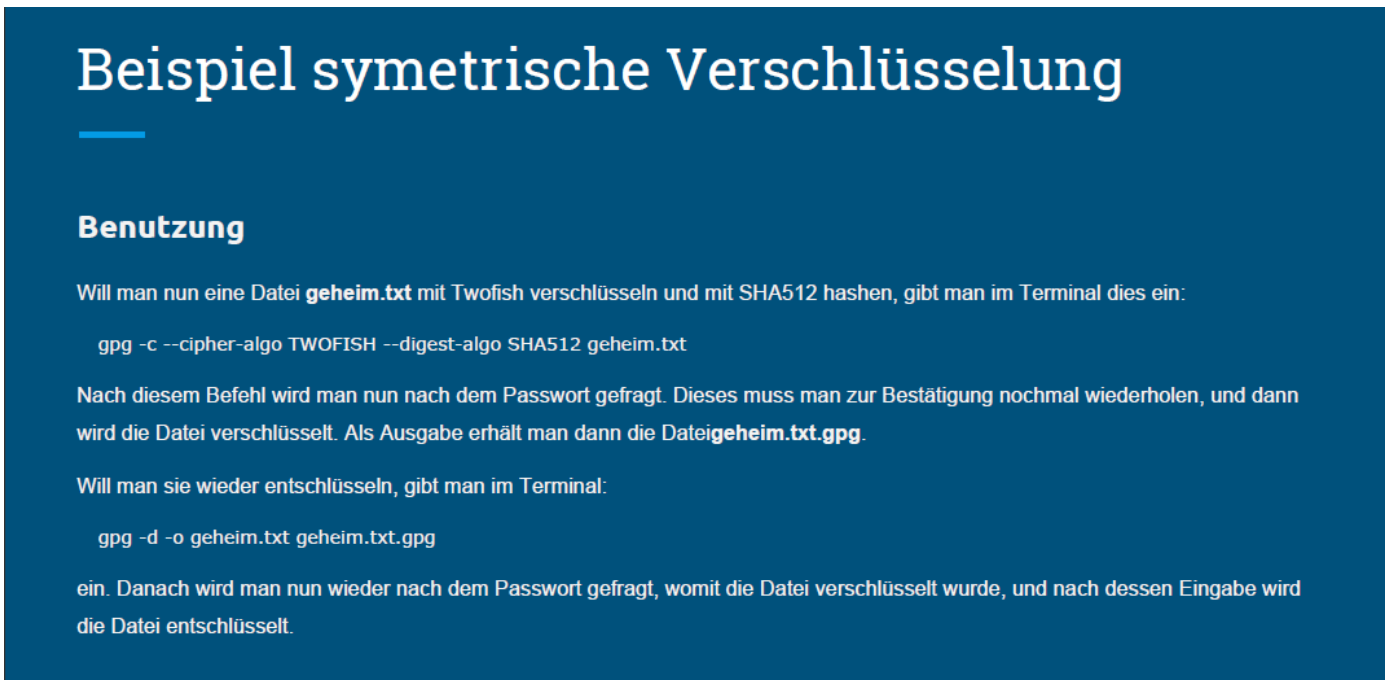
```
pi@DF9PM:~ $ gpg --version
gpg (GnuPG) 1.4.18
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg
Unterstützte Verfahren:
Öff. Schlüssel: RSA, RSA-E, RSA-S, ELG-E, DSA
Verschlü.: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
           CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Komprimierung: nicht komprimiert, ZIP, ZLIB, BZIP2
pi@DF9PM:~ $
```

Hier seht Ihr die Erläuterung der vorhandenen Verschlüsselung.

Wenn man nun diese Verschlüsselung benutzt, dann sollte man das noch wissen:

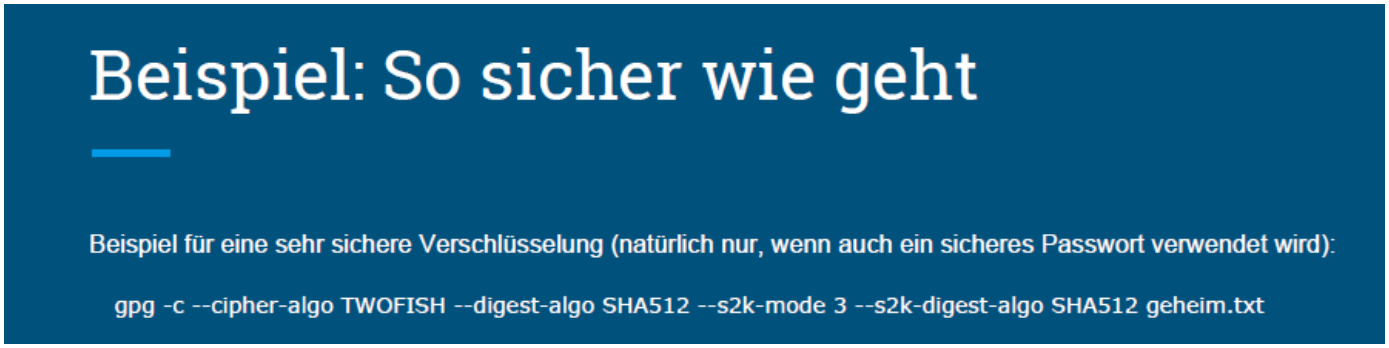


Nehmen wir also eine symmetrische Verschlüsselung:



Hier sehen wir also die Verschlüsselung und Entschlüsselung einer Datei namens: geheim.txt.

Es geht aber auch noch sicherer – so sicher wie geht:



D.h. wir verschlüsseln eine Datei: (besonders sicher)

Eingabe: (Beispiel)

„gpg -c --cipher-algo TWOFISH --digest-algo SHA512 --s2k-mode 3 --s2k-digest-algo SHA512 geheim.txt“

Hier wird die Datei „geheim.txt“ schlüsselt.

Den Namen (hier geheim.txt) solltet natürlich auf Euren Dateinamen, den Ihr verschlüsseln wollt – anpassen.

Es wird dann nach einem Passwort gefragt.

Das wird zweimal eingegeben und ist wichtig für die Endschlüsselung.

Also wenn Ihr ein Passwort vergebt, dann notiert es Euch, denn:

Wenn ihr es nicht mehr wisst, dann bekommt Ihr die Datei nie mehr geöffnet.

Danach sieht man eine neue Datei mit der Endung: Dateiname.txt.gpg

Die unverschlüsselte Datei kann dann gelöscht werden. Diese braucht ihr ja nicht mehr und es ist natürlich unsicher, diese zu behalten, da sie den Inhalt unverschlüsselt freigibt.

Also löschen.

Diese Vorgehensweise gilt für eine Datei.

Möchte man mehrere Dateien verschlüsseln, dann sollte man vorher die Dateien in eine ZIP Datei packen.

Man hat danach eine gepackte ZIP - Datei mit vielen Dateien und kann diese dann endgültig dann endgültig mit „gpg“ verschlüsseln.

D.h. ZIP (auch auf dem Raspberry sehr wichtig)

Zippen heißt, dass man die Datei oder die Dateien zusammenpackt (komprimiert) und so

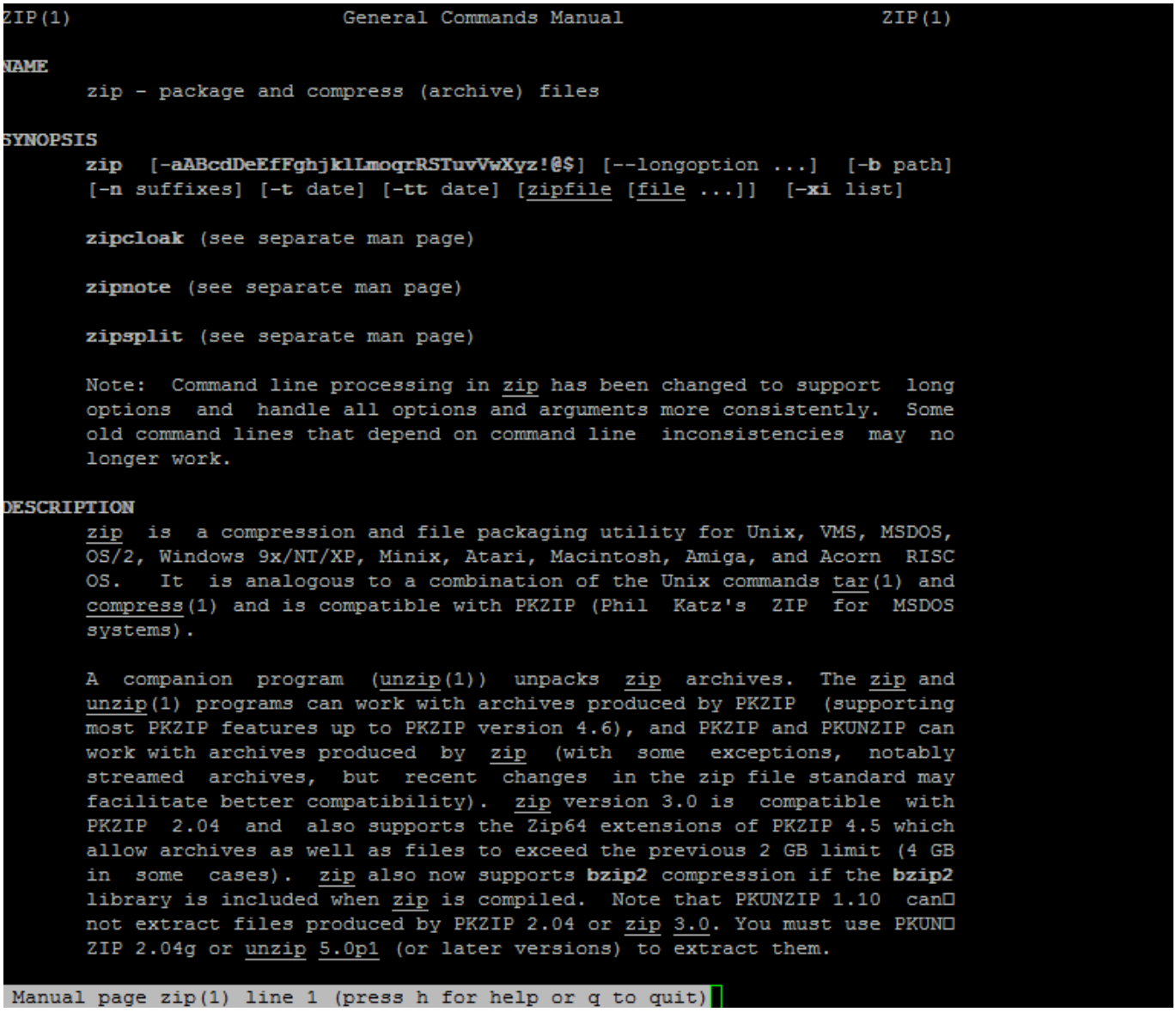
- Platz zu sparen
- Mehrere Dateien in **ein** Packet zu packen.
- Pakete zu verschlüsseln.

Dazu muss man aber Zip auf dem Raspberry installieren:

Das geht mit dem Befehl:

“Sudo apt-get install zip”

Zip Manual aufrufen:
Eingabe – „man zip“



```
ZIP(1)                    General Commands Manual                   ZIP(1)

NAME
       zip - package and compress (archive) files

SYNOPSIS
       zip  [-aABcdDeEfFghjklLmoqrRSTuvVwXyz!@$] [--longoption ...]  [-b path]
       [-n suffixes] [-t date] [-tt date] [zipfile [file ...]]  [-xi list]

       zipcloak (see separate man page)

       zipnote (see separate man page)

       zipsplit (see separate man page)

       Note:  Command line processing in zip has been changed to support  long
       options  and  handle all options and arguments more consistently.  Some
       old command lines that depend on command line  inconsistencies  may  no
       longer work.

DESCRIPTION
       zip  is  a compression and file packaging utility for Unix, VMS, MSDOS,
       OS/2, Windows 9x/NT/XP, Minix, Atari, Macintosh, Amiga, and Acorn  RISC
       OS.   It  is analogous to a combination of the Unix commands tar(1) and
       compress(1) and is compatible with PKZIP (Phil  Katz's  ZIP  for  MSDOS
       systems).

       A  companion  program  (unzip(1))  unpacks  zip  archives.  The zip and
       unzip(1) programs can work with archives produced by PKZIP  (supporting
       most PKZIP features up to PKZIP version 4.6), and PKZIP and PKUNZIP can
       work with archives produced  by  zip  (with  some  exceptions,  notably
       streamed  archives,  but  recent  changes  in the zip file standard may
       facilitate better compatibility).  zip version 3.0 is  compatible  with
       PKZIP  2.04  and  also supports the Zip64 extensions of PKZIP 4.5 which
       allow archives as well as files to exceed the previous 2 GB limit (4 GB
       in  some  cases).  zip also now supports bzip2 compression if the bzip2
       library is included when zip is compiled.  Note that PKUNZIP 1.10  can□
       not extract files produced by PKZIP 2.04 or zip 3.0. You must use PKUN□
       ZIP 2.04g or unzip 5.0p1 (or later versions) to extract them.

 Manual page zip(1) line 1 (press h for help or q to quit)
```

Nun könnt Ihr die Funktionalität von ZIP auf dem Raspberry benutzen.

Dazu die gibt es folgenden Befehl:

Tar und zip

Beispiel tar:

tar cvzf k07.tgz geheim.txt Bilder Dokumente

c=create, v=verbose, z=compress, f=file

zip -r k07.zip geheim.txt Bilder Dokumente

-r=rekursiv

Im oberen Beispiel seht Ihr „cvzf“ hinter dem tar.

Diese Kürzel stehen für die Optionen:

C=create, v=verbose, z=compress, f=file

So - das war’s für den Raspberry Workshop II

Weiterhin viel Spaß mit Eurem Raspberry.

Kontaktadresse Stefan DF6PA.
eMail: df6pa@darc.de
Tel 0151-54723700