GnuPG-Card Anleitung

Andere Anleitungen

GnuPG-Card

- Anleitung der Free Software Foundation Europe
- Anleitung von spin.atimoicobject.com

Best Practice (Nutzung von Haupt- und Subkeys) für gnupg

- http://www.openpgp-schulungen.de/kurzinfo/schluesselqualitaet/
- http://www.openpgp-schulungen.de/inhalte/
- http://www.hauke-laging.de/sicherheit/openpgp.html
- http://blog.andreas-haerter.com/2010/04/12/gnupg-schluesselhierarchie-passender-algorithmus-fuer-jede-aufgabe-schluesselaustausch-ohne-verlust-des-web-of-trust#fn__8
- https://alexcabal.com/creating-the-perfect-gpg-keypair/
- http://www.bootc.net/archives/2013/06/07/generating-a-new-gnupg-key/

Nutzung der Gnupg Card

- Die verschiedenen Möglichkeiten zum Einsatz der Karte

Grundsätzliche Infos über Smart Cards

Sogenannte Smart Cards können zur Authentifizierung, Verschlüsselung und Signatur mit unterschiedlichen Diensten und Programmen dienen.

Es gibt zwei unterschiedliche Konzepte:

- Die „pki cards“ (PKCS#11), siehe hier: https://wiki.debian.org/Smartcards
- und die vor allem unter Linux sehr gut nutzbaren Gnupg-Cards https://wiki.debian.org/Smartcards/OpenPGP

Während erstere den Quasi-Standard für x509 Zertfikate und eine Public-Key-Infrastructure darstellen scheint die gnupg Variante unter Linux einfacher einzurichten und besser unterstützt zu sein.

Ähnliche Konzpete verfolgen auch

- der Yubikey

- eine Mischung aus USB-smartcard und One-Time-Password Generator

- und der Cryptostick der Privacy-Foundation.

Ein Überblick über SmartCard, Cryptostick und Yubikey findet sich im Privacy-Handbuch

Unterschiede der verschiedenen Systeme:

System	Vorteile	Nachteile
Gnupg Card	Karten leicher erhältlich und preiswerter. Unterstützung unter Linux einfacher	Authentifizierung mit Clients wie Thunderbird etc. nicht so einfach
PKCS#11	De Facto Standard	komplizierter in der Einrichtung, Karten teurer?, Beschaffung komplizierter, Kompatibilität nicht immer gegeben

Vor- und Nachteile sind z.B. hier diskutiert worden:
http://lists.gnupg.org/pipermail/gnupg-users/2006-February/027936.html

Diese Anleitung beschäftigt sich nur mit der Einrichtung der Gnupg-Card.

Voraussetzungen

Wir benötigen

- eine Gnupg-Card
- einen kompatiblen Cardreader

Beides ist am einfachsten im FLOSS Shop zu erhalten. Eine Gnupg-Card erhält auch, wer der Free Software Foundation Europe beitritt.

Software installieren

Als erstes installieren wir die Software. Wichtig: die gnupg-card funktioniert nur mit gnupg Version 2.

```
apt-get install gnupg2 scdaemon gpgsm pcscd
```

Wenn der PKCS#11-Support verwendet werden soll evtl. auch:

```
apt-get install gnupg-pkcs11-scd
```

Card-Reader einrichten

In der Anleitung der Free Software Foundation Europe wird beschrieben, wie der Card-Reader so

eingerichtet wird, dass das USB-Device die richtigen Rechte und Gruppenzugehörigkeit erhält.

Unter Debian Wheezy war das für den Gemalteo Shell Token nicht nötig. Der Card-Reader funktionierte nach der Installation der Software „Out of the Box“. Das liegt möglicherweise daran, dass die ccid-library noch mal anders funktioniert als die gnupg built in ccid.

Das Device des Card-Readers muss die richtigen Dateirechte erhalten, damit die User darauf zugreifen können. Das erfolgt mittels udev und wird hier beschrieben

Als schneller Workaroundzum Testen oder zur Nutzung innerhalb von Live-Systemen (zur sicheren Generierung der Schlüssel) kann das folgendermaßen getan werden

```
lsusb
Bus 001 Device 008: ID ... Card-Reader ....
```

damit wissen wir die Bus und Device ID:

```
ls -l /dev/bus/usb/001/008
```

als temporärer Workaround:

```
chmod 666 /dev/bus/usb/001/008
```

Schlüssel erzeugen und auf Karte transferrieren

Es gibt grundsätzlich mehrere Möglichkeiten, die Schlüssel für die Karte zu erzeugen

1. Keypair (Hauptkey) kann direkt auf der Karte erzeugt werden
2. Ein Keypair wird außerhalb erzeugt werden. Anschließend werden Subkey generiert (für signieren, verschlüsseln und Authentifizieren) und auf die Karte importiert. Diese Variante ist die von der empfohlene und im Folgenden beschriebene.

Hauptkey mit Subkeys

- Eine ausführlichere Anleitung (incl. Backup der keys findet sich hier: http://www.bootc.net/archives/2013/06/07/generating-a-new-gnupg-key/)

Das Generieren der Schlüsselpaare findet vorzugsweise auf einem Live-System statt, damit sicher gestellt werden kann, dass die Schlüssel nicht bereits bei der Erstellung von Dritten gelesen werden können. Ist der geheime Schlüssel einmal auf der Karte, kann er, so zumindest die Theorie, diese nicht mehr verlassen. Alle Operationen finden auf der Karte statt. Daher lohnt es sich, bei der Erstellung der Schlüssel wirklich sehr sorgfältig vorzugehen:

- vertrauensvolle Hardware
- Live-System möglichst ohne Netzwerkverbindungen

Ein angepasstes Live-System mit aller nötigen Software sowohl zum Erzeugen der Keys als auch für die Verwendung der Gnupg-Karte kann beim Datenkollektiv bestellt werden.

Ihr erhaltet einen Bootbaren USB-Stick mit

- angepasstem xfce4 Desktop und allen nötigen Crypto-Tools
- einer zusätzlichen LUKS-verschlüsselten Partition, die z.B. für die Backups der privaten Schlüssel dienen kann.

Hauptschlüssel-Paar generieren:

Ist alles so weit vorbereitet, geht es jetzt an das eigentliche Erstellen der Schlüssel. In einem Terminal werden jeweils die markierten Kommandos ausgeführt.

Als erstes wird der Hauptschlüssel erstellt, der nur zum Signieren der Unterschlüssel dient:

```
gpg --full-generate-key
```

Bei der Auswahl:

```
RSA (sign only)
```

- ggf. unbegrenzte Haltbarkeit

Erstellen der Unterschlüssel (Subkeys)

Mit

```
gpg --list-keys
```

finden wir unsere neue Key-ID heraus. Diese nutzen wir nun zum Editieren des Schlüssels:

```
gpg --expert --edit-key KEYIDXYZ
```

Im Anschluss generieren wir drei Subkeys für folgende Zwecke:

1. Signieren
2. Verschlüsseln
3. Authentifizieren

```
gpg>  addkey
```

legt einen neuen Subschlüssel an. Wir müssen jetzt je einen zum Signieren, Verschlüsseln und Authentifizieren anlegen, d.h. die folgende Prozedur wird **dreimal** wiederholt.

Bei der folgenden Auswahl:

```
Please select what kind of key you want:
(3) DSA (sign only)
```

```
(4) RSA (sign only)
(5) Elgamal (encrypt only)
(6) RSA (encrypt only)
(7) DSA (set your own capabilities)
(8) RSA (set your own capabilities)
Your selection?
```

nutzen wir „8“ „Set your own capabilities“. Signieren und Verschlüsseln funktioniert zwar auch mit den vorgegebenen, Authentifizieren funktioniert aber nur mit benutzerdefinierten Fähigkeiten.

Jetzt kann mit den angegebenen Buchstaben jeweils eine Fähigkeit zu- oder abgeschaltet werden.

Zuletzt sichert ein

```
gpg> save
```

unsere Änderungen.

Backup

Dieser Schritt muss unbedingt vorgenommen werden, **bevor** die Subkeys auf die Karte transferriert werden, da das Kommando `keytocard` die Schlüssel aus dem Schlüsselbund auf die Karte verschiebt. Im Secring auf der Platte bleiben nur Stubs.

- Der gesamte Keyring (Unterverzeichnis .gnupg) sollte nach der Prozedur auf ein verschlüsseltes Medium bzw. in einem Container gesichert werden.
- eine weitere Variante zum Backup von Keys ist paperkey

- ein Programm um ausdruckbare Daten zu erhalten und den key auf Papier auszudrucken.

Schlüssel auf die Gnupg-Karte kopieren

Achtung: Bitte sicher stellen, dass vor dem Kopieren auf die Karte ein Backup mit allen Subkeys auf einem sicheren Medium erstellt wurde. Die folgenden Kommandos löschen die Subkeys aus dem Schlüsselbund und verschieben sie auf die Karte, von der sie zwar benutzt - aber nicht wieder hergestellt werden können.

Im nächsten Schritt kopieren wir unsere Subkeys auf die gnupg-Karte. Hierfür benötigen wir das Passwort für den geheimen Hauptschlüssel und die Admin-Pin für die Karte.

```
gpg --expert --edit-key KEYIDXYZ
```

Ein

```
gpg> toggle
```

schaltet in die Ansicht der privaten Schlüssel um.

Jetzt markieren wir der Reihe nach unsere Subkeys:

```
gpg> key 1
```

und kopieren sie auf die Karte:

```
gpg> keytocard
```

Das wiederholen wir jetzt mit Schlüssel 2 u. 3. (Die Schlüssel müssen jeweils wieder entsperrt werden.)

öffentlichen Schlüssel exportieren

Den öffentlichen Schlüssel (der nicht auf der Karte gespeichert ist) exportieren wir auf ein zugängliches Medium oder zu einem Keyserver. Wir brauchen diesen später.

```
gpg --export --armor KEYID> public_key.asc
```

Geheimen Schlüsselbund auf den Nutzer_innen Rechnern erstellen

Auch wenn die geheimen Schlüssel nun auf der Karte sind - und nicht auf den jeweiligen Computern: dennoch muss ein Schlüsselbund mit den Verweisen auf die geheimen Schlüsseln auf die Karte erstellt werden.

Während bei Smartcards, die von `gnupg 1` unterstützt werden ein einfacher Aufruf von

```
gpg --card-status
```

reicht, um Verweise auf die Karte zu generieren, muss bei `gnupg 2` folgender Aufruf erfolgen:

```
gpg2 --secret-keyring .gnupg/secring.gpg --card-status
```

in manchen Fällen funktioniert dies jedoch nur über den internen Befehl `fetch`

```
$ gpg2 --card-edit
...
gpg/card> fetch
...
gpg: Total number processed: 1
gpg:               imported: 1 (RSA: 1)
$ gpg2 --card-status
...
```

Übrigens bei einer Erweiterung des öffentlichen Schlüssels

mit neuen Schlüsseln, deren neue geheime Gegenstücke auf einer neuen Karte sind, müssen erst alle Verweise auf die alte Karte gelöscht werden.

Karte wechseln

Beim Ersetzen einer Karte muss folgendes beachtet werden:

1. Die Keys müssen analog der Anleitung auf die neue Karte kopiert werden

~~- der Verweis auf den privaten Schlüssel (also nur der Rest des privaten Schlüssels) muss auf den Rechnern, auf denen die Karte zum Einsatz kommt gelöscht werden. Ein `gpg --delete-secret-keys KEYID` erledigt das.~~ **~~Sicher stellen, dass es der Schlüsselbund mit den Verweisen ist, der bearbeitet nicht - und nicht z.B. der mit den Originalschlüsseln~~**.

1. seit gnpg2.1 werden die secret Keys nicht mehr in secring.pgp gespeichert - sondern unter `.gnupg/private-keys-v1.d/`
2. Die zu den privaten Keys der Karte passenden Verweise müssen aus `.gnupg/private-keys-v1.d/` gelöscht werden.
3. Danach müssen die Verweise auf die neue Karte wieder erstellt werden: `gpg2 –secret-keyring .gnupg/secring.gpg –card-status`

Anzeigen der zur gnupg-Karte passenden Dateien unter `.gnupg/private-keys-v1.d/`:

Das ist hier von Werner beschrieben:
https://lists.gnupg.org/pipermail/gnupg-users/2016-April/055766.html

```
gpg --with-keygrip -k KEYID
```

Die ausgegebenen Keygrips beziehen sich auf die Dateien unter: `.gnupg/private-keys-v1.d/`

```
rm ''.gnupg/private-keys-v1.d/KEYGRIP.key''
```

Achtung: immer Backup des .gnupg-Verzeichnisses machen, damit nicht aus Versehen andere secret Keys gelöscht werden.

Stolperstellen

gnupg sollte aus diversen Gründen nicht mit dem gnome keyring genutzt werden. Dazu sollten zum einen die Autostart-Einträge der Desktop-Umgebung angepasst werden:

```
mv /etc/xdg/autostart/gnome-keyring-gpg.desktop /etc/xdg/autostart/gnome-keyring-gpg.desktop.inactive
```

beim Aufruf von

```
gpg2 --card-status
```

gibts ggf. Beschwerden:

```
gpg: selecting openpgp failed: Nicht unterstütztes Zertifikat
gpg: OpenPGP Karte ist nicht vorhanden: Nicht unterstütztes Zertifikat
```

Grund: Die Umgebungsvariable GPG_AGENT_INFO ist gesetzt. Erst ab Version 2.1 ignoriert gpg2 diese. Ein

```
unset GPG_AGENT_INFO
```

verschafft Linderung und sollte ggf in den Startscripten untergebracht werden.

Update für Debian Stretch:

Seit Debian Stretch wird der gpg-agent vom systemd in der user-Session gestartet. Das deaktivieren des gnome-keyring-daemons sollte hier nicht mehr nötig sein.

Erfahrungen mit Kartenlesegeräten

Übersicht über unterstützte Kartenlesegeräte:

http://pcsclite.alioth.debian.org/ccid/supported.html

Bei der Wahl eines Kartenlesegerätes sollte eines mit der Unterstützung sogenannter "extended APDU" gewählt werden.

Ohne „extended APDU“ Unterstützung sind Schlüsselgrößen ab 2048 Bit nicht möglich.

Vollständig unterstützte Kartenleser

gemalto usb shell token v2

Ein USB-Kartenleser in USB-Stick Format ohne Keypad

- funktioniert out of the box mit dem pcscd

http://pcsclite.alioth.debian.org/ccid/supported.html#0x046A0x003E

Ein USB-Kartenleser mit Pinpad

- funktioniert out of the bos mit dem pcscd

Nicht oder nicht vollständig unterstützte Kartenleser

ReinerSCT Secoder

Für diesen Kartenleser sind zusätzliche Treiber nötig, die sich in den folgenden Debian-Paketen finden:

```
apt-get install fxcyberjack libifd-cyberjack6
```

bzw. für Stretch:

```
apt-get install libifd-cyberjack6
```

Allerdings scheint trotzdem das Pinpad nicht unterstützt zu werden:
https://blogs.fsfe.org/jzarl/2013/08/20/card-reader-reinersct-cyberjack-pinpad/

Omnikey CardMan 6121

Ein USB-Kartenleser in USB-Stick Format ohne Keypad

Lt. https://www.gnupg.org/howtos/card-howto/en/ch02s02.html sollte dieser Cardreader von der gnupg-internen libccid untersützt werden. Allerdings funktioniert das nur aufgrund fehlender Unterstützung von extended APDU nur mit Schlüsseln < 2048 Bit. Bei Schlüsseln ab 2048 Bit wird ein keytocard mit folgendem Fehler beendet:

```
gpg: ccid_transceive failed: (0x10002)
gpg: apdu_send_simple(0) failed: invalid value
```

Laut einem Ubuntu-User soll es möglich sein, das Gerät auch mit 2048 Bit Schlüsseln zu verwenden. Aufgrund der fehlenden APDU Unterstützung scheint das aber bei der angegebenen Schlüssellänge unwahrscheinlich.

Omnikey 3621

Dieser Kartenleser mit Pinpad funktioniert nur mit dem gnupg internen ccid Treiber. Um diesen zu nutzen, muss der pcscd gestoppt werden. Beim Schreiben auf die Karte kommt es bei Schlüsseln ab 2048 Bit zu folgender Fehlermeldung:

```
scdaemon[29150]: apdu_send_simple(0) failed: unknown status error
scdaemon[29150]: Der Fingerabdruck kann nicht gespeichert werden:
Kartenfehler
gpg: Fehler beim Schreiben des Schlüssels auf die Karte: Kartenfehler
```

Gleiche Ursache: Extended APDU sind nicht unterstützt. Das Pinpad wird wohl auch nicht unterstützt.

From:
https://wiki.datenkollektiv.net/ - **datenkollektiv.net**

Permanent link:
https://wiki.datenkollektiv.net/public/gnupg/gnupg-card-howto?rev=1596901223

Last update: **2020/08/08 17:40**