GPG Smartcard Self-Help Documentation

Release 0.0.1

ax3l et al.

Jan 24, 2020

INSTALLATION

Here comes a welcome message

CHAPTER 1

Authors

Here is everyone who contributed to this document! :) Add yourself!

ax3l, informancer, Alexander (Nitrokey) the participants of Datenspuren 2017, ...

1.1 Debian 9 "stretch"

1.1.1 General

```
apt-get install gnupg2 gnupg-agent pcscd pcsc-tools scdaemon libusb-1.0-0 libccid
service udev restart
```

1.1.2 Yubikey

```
apt-get install libu2f-host0 yubikey-personalization yubikey-personalization-gui␣
↪ykneomgr
service udev restart
```

Note: There are new GUI and CLI tools currently in beta for Yubikeys. See Yubikey Manager and its CLI (`ykman`).

Further initial setup needs explained in [YubiEdit]:

```
# check USB devices
lsusb
# Yubikey listed?

# make sure CCID mode is enabled
# note: -m86 is also possible, it enables all modes
ykpersonalize -m6
```

(continues on next page)

(continued from previous page)

```
# check basic information of your yubikey
ykinfo -a

# check smart card readers
# pcsc_scan
# Yubikey listed?

# connect to key's GPG app and output version
gpg-connect-agent --hex "scd apdu 00 f1 00 00" /bye
# D[0000]  04 03 04 90 00
# OK
# -> means: v4.3.4

# OpenPGP Version 1 or 2 card?
gpg2 --card-status | grep Version
# Version ..........: 2.1
```

Note: to do: yubi touch setup

1.1.3 Nitrokey

...

1.1.4 Further references

- https://wiki.fsfe.org/TechDocs/CardHowtos/CardWithSubkeysUsingBackups
- https://malcolmsparks.com/posts/yubikey-gpg.html
- https://github.com/drduh/YubiKey-Guide/find/master#install—linux
- https://developers.yubico.com/libu2f-host/
- http://wiki.yobi.be/wiki/GnuPG (includes trouble shooting)
- https://www.nitrokey.com/documentation/installation
- GUI: Gnu Privacy Assistant (GPA)

1.2 List

1.2.1 Owned Keys

Keys that you own (aka: you have access to the private keys)

```
gpg2 --list-secret-keys
```

1.2.2 Interpret Output

```
sec   rsa4096/0xCCCCCCCCCCCCCCCC 2015-02-01 [SCA] [expires: 2020-01-31]
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
uid                  [ultimate] Jane Roe <jane@roe.de>
uid                  [ultimate] Dr. Jane Roe <j.roe@esa.int>
ssb   rsa4096/0xEEEEEEEEEEEEEEEE 2015-02-01 [E] [expires: 2020-01-31]
```

Line-By-Line

- line 1: the *primary key*
- line 2: fingerprint
- line 3-4: user IDs of the key (optional, used in Web-of-Trust and for Certification)
- line 5: the first *subkey*

Column-By-Column

1st column:

- `sec`: ... (secret key available?)
- `ssb`: ... (secret key of a subkey available?)
- `ssb*`: currently selected subkey during edits (next section: selected via `gpg2 --edit-key CCCCCC-key N`)
- `ssb>`: only stub of a subkey is available (next section: after secret part has been moved to a smartcard)
- `sec#`: the secret part is not available (removed from local key store)
- `uid`: this line is a user ID

2nd column:

- `algorithm/0xKeyID`

3rd column:

- date of generation

4th column:

- `[...]`: capabilities (see below)

5th column:

- date of expiration

Capabilities

Noted in each key with `[...]`. Can be delegated from the primary key to subkeys.

S: Sign. Signs data, such as e-mails.

C: Certification. Certifies (also: "signs") keys, e.g. keys of other people at a crypto party or during subkey generation. All primary keys *must* have this capability.

A: Authenticate. Can be used for logins such as SSH.

E: Encrypt. Encrypts (and decrypts?) data.

Note: to do: document each entry/line

1.3 Change Home Directory

By default, all `gpg2` configuration data resides in `$HOME/.gnupg/`.

If you want to experiment with GPG, test a new smartcard or follow this document, you can also create a temporary new location to experiment with. This prevents accidentially editing/deleting/modifying your already existing keys and to work from a live system or on an "offline" USB stick.

```
# assuming the directory exists,
# otherwise create it with
#   mkdir /media/my-usb-stick/my-gpg-home/
#   chmod -R og-rwx /media/my-usb-stick/my-gpg-home/
export GNUPGHOME=/media/my-usb-stick/my-gpg-home/
```

This setting is valid until you close your terminal.

The directory should only be accessible by the current user (see `chmod` comment above), otherwise you will see warnings of the form `# gpg: WARNING: unsafe permissions on homedir '/media/my-usb-stick/my-gpg-home/'`.

Note: On the first call to `gpgp2` with a changed (fresh) home it will create a "keybox" (`pubring.kbx`) and a "trustdb" (`trustdb.gpg`).

```
gpg2 --list-secret-keys

# gpg: keybox '/media/my-usb-stick/my-gpg-home/pubring.kbx' created
# gpg: /media/my-usb-stick/my-gpg-home/trustdb.gpg: trustdb created
```

1.4 Export

How to backup your keys.

1.4.1 Public Key

```
gpg2 --export --armor CCCCCCCCCCCCCCCC > pubkey.ascii
```

1.4.2 Full Key

(including private part, probably what you are looking for)

```
gpg2 --export-secret-key --armor CCCCCCCCCCCCCCCC > fullkey.ascii
```

Note: An exported private key, e.g. stored on a USB stick is unprotected if you loose they storage media. You can encrypt and decrypt the backup file with a simple password via:

```
# encrypt with password
gpg -c fullkey.ascii
# generates an encrypted fullkey.ascii.gpg file
rm fullkey.ascii

# descrypt with password
gpg -d fullkey.ascii.gpg > fullkey.ascii
# generates a decrypted fullkey.ascii file
```

1.4.3 Stubs

(for keys moved to smartcards)

1.5 Create a Key

. . . on your local laptop.

```
gpg2 --gen-key
# here we select
#   RSA
#   size: 2048 or 4096
```

Warning: For Yubikeys, only generation 4 and newer support RSA key sizes of 4096!

Note: One can also generate keys directly on the smartcard. Doing so has a list of pro's and con's that we need to inform you about first (to do).

Now, *list your new keys*.

1.5.1 Revocation Key

Note: A revocation key is not sensitive for encrypted data. Still, anyone who can access your revocation keys can publicly and provably mark your keys as invalid. For most use cases, just keep is as safe as your *privaye key backups*.

1.6 Create Subkeys

. . . on your local laptop.

Note: Reminder from `gpg2 --list-secret-keys`:

```
sec   rsa4096/0xCCCCCCCCCCCCCCCC 2015-02-01 [SCA] [expires: 2020-01-31]
# ...
```

```
gpg2 --expert --edit-key CCCCCCCCCCCCCCCC

gpgp> addkey
# select 8 add a new RSA sub-key to your key
# select A, then S, then E to get a pure authentication key. Then Q to continue.
# select same expiry as for the rest of the key and then answer y to save changes.

gpg> quit
```

1.7 Renew a Key

1.7.1 Local

```
gpg2 -- ...
```

1.7.2 From USB Backup

1.7.3 Smartcard

1.8 Personalize

You can personalize some of the meta information of your smartcard directly via the gpg2 interface. This allows you to, e.g. set passwords or additional meta information.

```
# start the edit of the smartcard
gpg2 --card-edit

gpg/card> admin
Admin commands are allowed

gpg/card> passwd
# ...

# setting a URL for a public key dramatically easens the startup
# needs when connecting the smartcard to an other device
# (laptop, Android Cell Phone with OpenKeyChain) since the pub
# key is not installed on the device
gpg/card> url
URL to retrieve public key: https://keybase.io/user/key.asc
# note: some older versions of GPG only support http:// :(

# the options below are purely administrational if you have several
# smartcards or want to use them in specific workflows.
# the information are not authenticated in any way and optional
gpg/card> name
Cardholder's surname: Doe
```

(continues on next page)

(continued from previous page)

```
Cardholder's given name: Jane

gpg/card> lang
Language preferences: en

gpg/card> sex
Sex ((M)ale, (F)emale or space): f

gpg/card> login
Login data (account name): jane

gpg/card> quit
```

1.8.1 References

- [YubiEdit]
- https://www.gnupg.org/howtos/card-howto/en/ch03s03.html

1.9 Import

Full key to stub key...

Warning: Importing a GPG key to a smart-card is a *one-way operation* for your private key! Backup your *full key* via an *export first*!

Note: Reminder from `gpg2 --list-secret-keys` after we created a *sub-keys*.

```
sec   rsa4096/0xCCCCCCCCCCCCCCCC 2015-02-01 [SC] [expires: 2020-01-31]
uid                  [...]
ssb   rsa4096/0xEEEEEEEEEEEEEEEE 2015-02-01 [E] [expires: 2020-01-31]
ssb   rsa4096/0xAAAAAAAAAAAAAAAA 2015-02-01 [A] [expires: 2020-01-31]
```

```
gpg2 --edit-key CCCCCCCCCCCCCCCC

# move primary key to signature key slot on card
gpg> toggle
gpg> keytocard
# select: (1) Signature key

# repeat for encryption key
gpg> key 1
gpg> keytocard
# select: (2) Encryption sub-key

# repeat for authentication sub-key
gpg> key 2
gpg> keytocard
# select: (3) Authentication key
```

(continues on next page)

(continued from previous page)

```
gpg> quit
# Save changes? (y/N) y

gpg2 --list-secret-keys
# [...]
# ssb* [...]
```

See [YubiImport] and [YubiEdit]

1.10 Change Device

If you are moving with your configured smartcard to another device, its GPG keyring needs to learn the key before the card can be used. The fastest way to achieve this is to retreive the public key as configured in the smartcard.

```
gpg2 --edit-card

# receive key stubs from URL
gpg> fetch
gpg> quit

gpg2 --list-keys
# now visible
gpg2 --list-secret-keys
# [...]
# ssb* [...]
```

1.11 Algorithms

ECC ws RSA and hardware support.

1.12 Key Generation on Device

1.12.1 Pros

- key never leaves the device
- very easy setup, e.g. integrated in Enigmail

1.12.2 Cons

- backup possible on generation if at all?
- entropy and RNG on device need to be trustworthy (often issues)
- recent Infineon RSA key issues can also weaken such generated keys [1], [2]

1.12.3 How

```
# start the edit of the smartcard
gpg2 --card-edit

generate
```

1.12.4 Notes

The GPG manual on `--card-edit` lists the possibility to store a off-device backup on `generate`. Unclear: Is this possible with all devices?

1.13 Renewal

There are several ways to "renew" the lifetime of existing keys. This chapter informs about several common ones and their pros/cons (to do).

1.13.1 Keep Key, Extend Deadline

- pro: easy update & publication
- pro: no new signatures needed
- con: adds no forward secrecy

1.13.2 Keep Primary Key, Update Subkeys

- pro: no new signatures needed
- neutral: update process a bit more complex
- unclear: adds (no) forward secrecy?

1.13.3 Generate New Primary Key, Sign with old Primary

- pro: forward secrecy from this point onward
- neutral: update process a bit more complex
- con: only transitive trust from old key

CHAPTER 2

Installation

CHAPTER 3

GPG2 Basics

CHAPTER 4

Keygen

CHAPTER 5

Smartcard

CHAPTER 6

Background

Bibliography

[YubiImport] https://developers.yubico.com/PGP/Importing_keys.html

[YubiEdit] https://developers.yubico.com/PGP/Card_edit.html