Arbeiten mit GnuPG

Teil 1: Keyhandling und Keysigning

Wolfgang Stief

stief@guug.de

2004-05-10
sage-muc

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

gpg --fingerprint

- GnuPG User seit 2000
- signierter Key (c't) seit 2001
- GnuPG auf Solaris und Debian GNU/Linux

- Ich bin **nicht** allwissend. Bitte unterbrechen!
- Installation der Tools wird hier und heute nicht behandelt – ihr als Admins müsst das auch ohne fremde Hilfe hinkriegen :-)

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

--sign vs. --encrypt

- Unterscheidung *verschlüsseln* vs. *signieren*.
- Verschlüsselung: Mailbody incl. Attachments werden verschlüsselt, Mailheader **muss** unverschlüsselt bleiben (MTAs). Partner brauchen jeweils zwingend *Public Key* der Gegenstelle.
- Signatur: digitale Unterschrift, damit ist prinzipiell sichergestellt, dass der Absender auch der ist, der er vorgibt zu sein.
- **Achtung!** Signatur kann ein Fake sein!
- Abhilfe: signierte Schlüssel, Aufbau eines *Web of Trust* bzw. *Trusted 3rd Party*. Partner brauchen sinnvollerweise *Public Key* der Gegenstelle.
- GnuPG nutzt zum Verschlüsseln und Signieren unterschiedliche Keys (ElGamal vs. DSA).

--sign vs. --encrypt (cont'd)

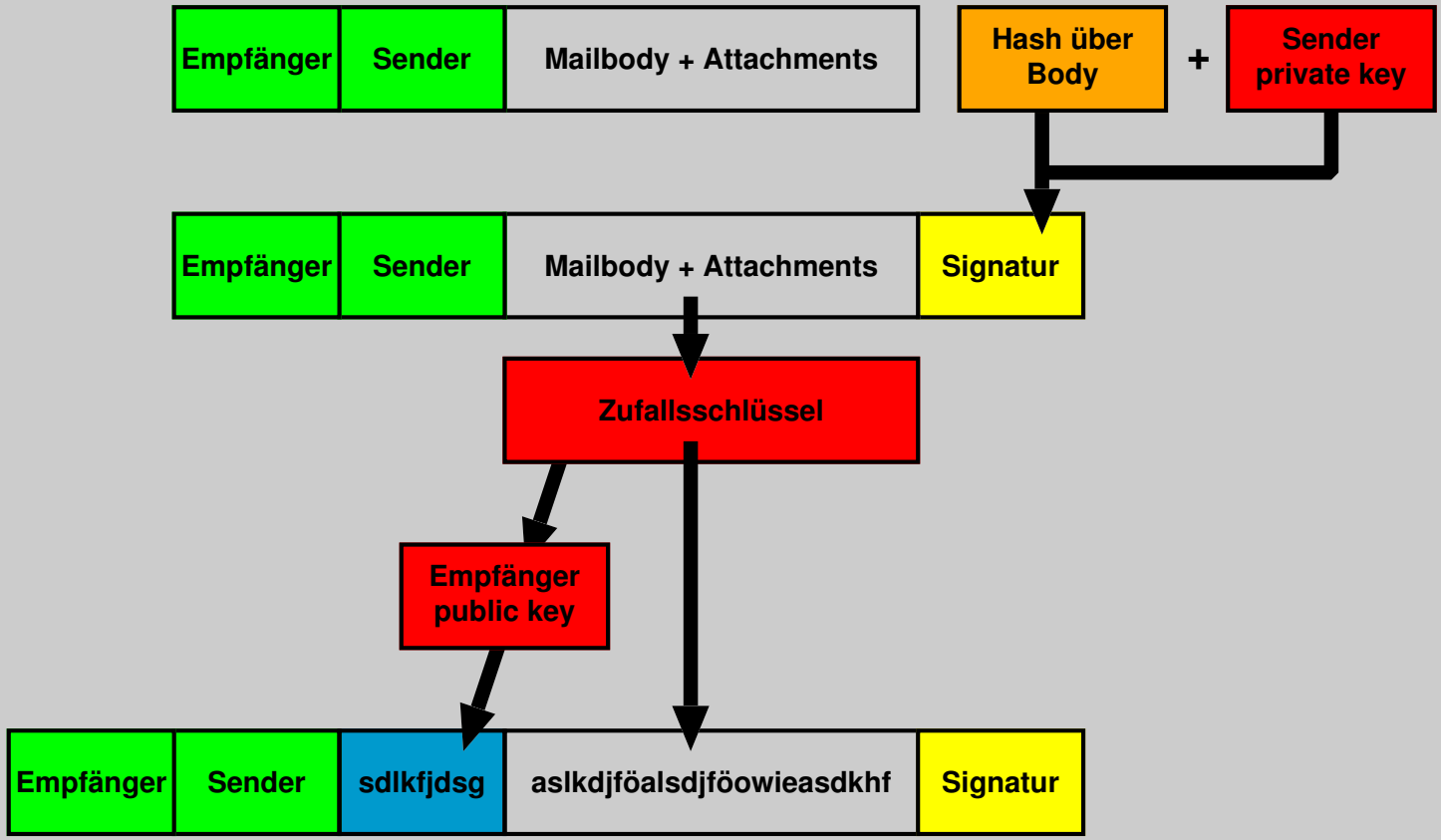

Abbildung 1: Unterscheidung: Verschlüsseln und Signieren von Mails

gpg --cipher-algo

- GnuPG kennt mehr Algorithmen als PGP! Das kann zu Problemen führen!
- ```
  evora<stief>/home/stief$ gpg --version
  gpg (GnuPG) 1.2.4
  Home:  /.gnupg
  Supported algorithms:
  Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA, ELG
  Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
  Hash: MD5, SHA1, RIPEMD160, SHA256
  Compression: Uncompressed, ZIP, ZLIB, BZIP2
  ```
- Default ist 1024bit DSA für Signierung (Hauptschüssel) und 2048bit ElGamal zum Entschlüsseln (Sub-Keys).
  ```

gpg --edit-key

- traditionell gpg als Command Line Tool
- Support für: FreeBSD (x86), OpenBSD (x86), NetBSD (x86), Windows 95/98/NT/2000/ME PocketConsole (Pocket PC), MacOS X, AIX, BSDI (i386), HP-UX 9-11, Irix, MP-RAS, OSF1, OS/2, SCO UnixWare, SunOS, Solaris (Sparc + x86), USL Unixware
- Frontends (GUI): jede Menge, verschiedene OS, Details unter http://www.gnupg.org/(en)/related_software/frontends.html#gui
- Für viele Mailclients sind Plugins verfügbar: http://www.gnupg.org/(en)/related_software/frontends.html#mua
- Vergleich und Stolperfallen: iX 3/2004, S. 126 ff.

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

gpg --fingerprint

```
evora<stief>/home/stief$ gpg --fingerprint stief@xyz.de
pub  1024D/96116155 2000-10-30 Wolfgang Stief <stief@xyz.de>
     Key fingerprint = C8ED 91A2 F5BD 7798 3433  DB01 33AD B7D9 9611 6155
uid                            Wolfgang Stief (Home) <ws@blubb.de>
uid                            Wolfgang Stief (GMX) <foobar@gmx.net>
uid                            Wolfgang Stief (Chaes) <w@foo.de>
sub  2048g/B65230E6 2000-10-30
```

User-IDs können beliebig viele existieren (uid). User-IDs können einzeln ungültig erklärt und zurückgezogen werden (Revocation Key).

- User-ID ist der Benutzername, zu dem der Schlüssel gehört.
- Kann im Prinzip beliebige Zeichenkette sein
- Hier: **nur natürliche Personen**, die an Hand amtlichem Lichtbildausweis identifiziert werden können!

gpg --fingerprint (cont'd)

```
evora<stief>/home/stief$ gpg --fingerprint stief@xyz.de
pub  1024D/96116155 2000-10-30 Wolfgang Stief <stief@xyz.de>
     Key fingerprint = C8ED 91A2 F5BD 7798 3433  DB01 33AD B7D9 9611 6155
uid                            Wolfgang Stief (Home) <ws@blubb.de>
uid                            Wolfgang Stief (GMX) <foobar@gmx.net>
uid                            Wolfgang Stief (Chaes) <w@foo.de>
sub  2048g/B65230E6 2000-10-30
```

Fingerprint ist ein One-Way-Hash über den Schlüssel. Der Schlüssel ist über Fingerprint und Key-ID eindeutig identifizierbar.

Subkeys sind – wie der Name sagt – Unterschlüssel (sub). Anzahl ist beliebig. Subkeys können einzeln ungültig gemacht werden.

Key-ID ist hier in Kurzform (4 Byte) dargestellt (pub). Langform (8 Byte) bekommt man mit der zusätzlichen Option `--with-colons`.

gpg --keyserver

Woher unbekannte Public-Keys nehmen?

- tauschen mit Kommunikationspartner (Mail, Speichermedium, Papier, Webseite etc.)
- auf Keyservern suchen und von dort importieren
- weltweites Netz von Servern, www.pgp.net
- Key nur einmal hochspielen, Server replizieren sich automatisch
- Abfrage/Suche über Webinterface möglich

`gpg --sign-key`

Am I am I?

- eigener Key muss zunächst selbst signiert werden
- darüber Nachweis, dass Eigentümer den Key selbst erstellt hat
- zwingende Voraussetzung für Schlüsselzertifizierung durch CAs
- zwingende Voraussetzung für Keysigning innerhalb Web of Trust
- macht GnuPG beim Schlüsselerstellen automatisch

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

gpg --gen-key

Schlüsseltyp und -länge, Gültigkeitsdauer, Mantra (== Passphrase)

```
gpguser@evora:~$ gpg --gen-key
gpg: keyring ‘/home/gpguser/.gnupg/secring.gpg’ created
gpg: keyring ‘/home/gpguser/.gnupg/pubring.gpg’ created
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
              minimum keysize is  768 bits
              default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024)
Requested keysize is 1024 bits
```

gpg --gen-key cont'd

```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct (y/n)? y
You need a User-ID to identify your key; the software constructs the user id
from Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
Real name: John Doe
Email address: doe@foo.bar
Comment: Blablubb
You selected this USER-ID:
    "John Doe (Blablubb) <doe@foo.bar>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
```

gpg --gen-key cont'd

```
You need a Passphrase to protect your secret key.

Enter passphrase:
Repeat passphrase:

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.++++++++.++++++++++.+++++..++++++++++..........>+++++........+++++
gpg: /home/gpguser/.gnupg/trustdb.gpg: trustdb created
public and secret key created and signed.
key marked as ultimately trusted.

pub  1024D/9CB035D0 2004-05-06 John Doe (Blablubb) <doe@foo.bar>
     Key fingerprint = 8558 61FD EECF 271D EBC0  4FE9 4319 4F10 9CB0 35D0
sub  1024g/3ECB438C 2004-05-06
```

gpg --sign-key --fingerprint

- Key ist nach Erzeugung bereits self-signed:

```
gpguser@evora:~$ gpg --sign-key doe@foo.bar
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
pub  1024D/9CB035D0  created: 2004-05-06 expires: never      trust: u/u
sub  1024g/3ECB438C  created: 2004-05-06 expires: never
(1). John Doe (Blablubb) <doe@foo.bar>
"John Doe (Blablubb) <doe@foo.bar>" was already signed by key 9CB035D0
Nothing to sign with key 9CB035D0
Key not changed so no update needed.
```

- Fingerprint:

```
gpguser@evora:~$ gpg --fingerprint 9CB035D0
pub  1024D/9CB035D0 2004-05-06 John Doe (Blablubb) <doe@foo.bar>
     Key fingerprint = 8558 61FD EECF 271D EBC0  4FE9 4319 4F10 9CB0 35D0
sub  1024g/3ECB438C 2004-05-06
```

gpg --keyserver --send-key

- **Vor** Keysigning-Party **muss** der Public Key auf einen Keyserver hochgeladen werden
- Hochladen: `gpg --keyserver www.de.pgp.net --send-key doe@foo.bar`
- alternativ: exportieren des Schlüssels in ASCII und hochladen über ein Webinterface (aber das ist **unsportlich** :-)

```
gpguser@evora:~$ gpg --armor --export doe@foo.bar
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.4 (GNU/Linux)

mQGhBECahywRBACwPQgQx0yX/hEQW9Q4C/eb6BNw77rwg0HJVCuD7oZvkQA3qc70
9ni17BcyrRyMR3lJX36E8ZcPczlfLW7Jts/JpKjFy7TuJbNPQMrI2MQDdwu/8Mjz
[...]
gf0tpt0M7VKaMIhJBBgRAgAJBQJAmoctAhsMAAoJEEMZTxCcsDXQD8MAoMERe6W/
6rSMy4FWWELzZgwz8z9oAJ9mRRQR74qHdEe+h3ydum3Q76bsNw==
=5Ycq
-----END PGP PUBLIC KEY BLOCK-----
```

- mögliches Webinterface: http://www.pca.dfn.de/pgpkserv/#submit

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

Let's paaarty...

Prinzipieller Ablauf:

1. *Key-Paar generieren.*
2. *Public Key an Keyserver schicken.*
3. *Key-ID + Fingerprint an Koordinator.*
4. Zur Party auftauchen. **Nicht vergessen:** amtlichen Lichtbildausweis (besser zwei), Key-ID, Schlüsseltyp, Schlüssellänge, Fingerprint vom eigenen Schlüsselpaar. Alles **auf Papier**.
5. Eigene Key-Info auf dem Papier des Koordinators überprüfen.
6. Identität und Key-Info der anderen anhand Ausweis und Fingerprint überprüfen.
7. *Keys der anderen Teilnehmer zuhause vom Server runterladen und signieren.*
8. *Signierte Keys wieder hochladen.*

Zur Keysigning-Party ist **keine** Elektronik notwendig!

Schlüssel und Tools

Begriffe

Vor der Party

Während der Party

Nach der Party

gpg --send-keys --recv-keys --list-sigs

Key suchen:

```
gpg --keyserver wwwkeys.eu.pgp.net --search-keys doe@foo.bar
```

Key runterladen:

```
gpg --keyserver wwwkeys.eu.pgp.net --recv-keys stief@chaes.de
```

Key überprüfen anhand des Fingerprints und der Key-ID:

```
gpg --fingerprint <keyid>
```

Key signieren:

```
gpg --sign-key stief@chaes.de
```

Signierten Key wieder hochladen:

```
gpg --keyserver wwwkeys.eu.pgp.net --send-keys stief@chaes.de
```

Achtung! Ein auf einem Keyserver vorhandener Key heisst **nicht automatisch**, dass er auch authentisch ist! Vertrauen ist an dieser Stelle gut, Kontrolle mitunter besser.

Beispiel Command Line (--sign-keys)

```
gpguser@evora:~$ gpg --sign-key 96116155
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
pub  1024D/96116155  created: 2000-10-30 expires: never      trust: -/-
sub  2048g/B65230E6  created: 2000-10-30 expires: never
(1). Wolfgang Stief (Chaes) <stief@chaes.de>
(2)  Wolfgang Stief (GfhR) <stief@gfhr.de>
[...]

Really sign all user IDs? y

How carefully have you verified the key you are about to sign actually belongs
to the person named above?  If you don't know what to answer, enter "0".

   (0) I will not answer. (default)
   (1) I have not checked at all.
   (2) I have done casual checking.
   (3) I have done very careful checking.

Your selection? (enter '?' for more information): 3
```

Beispiel Command Line (--sign-keys) (cont'd)

```
Are you really sure that you want to sign this key
with your key: "John Doe (Blablubb) <doe@foo.bar>" (9CB035D0)
I have checked this key very carefully.
Really sign? y

You need a passphrase to unlock the secret key for
user: "John Doe (Blablubb) <doe@foo.bar>"
1024-bit DSA key, ID 9CB035D0, created 2004-05-06
Passphrase:
gpguser@evora:~$

gpguser@evora:~$ gpg --list-sigs stief@chaes.de
pub  1024D/96116155 2000-10-30 Wolfgang Stief (Chaes) <stief@chaes.de>
sig 3       96116155 2003-04-25   Wolfgang Stief (Chaes) <stief@chaes.de>
sig 3       9CB035D0 2004-05-06   John Doe (Blablubb) <doe@foo.bar>
[...]
sig         B3B2A12C 2000-12-20   [User id not found]
[...]
```

Config-File

```
# GnuPG can import a key from a HKP keyerver if one is missing
# for sercain operations. Is you set this option to a keyserver
# you will be asked in such a case whether GnuPG should try to
# import the key from that server (server do syncronize with each
# others and DNS Round-Robin may give you a random server each time).
# Use "host -l pgp.net | grep www" to figure out a keyserver.
keyserver wwwkeys.eu.pgp.net

# The environment variable http_proxy is only used when the
# this option is set.
honor-http-proxy
```

- Proxy setzt man ggf. im Environment:

  `gpguser@evora:~$ export http_proxy="http://w3proxy:8080/"`

Keys über Webinterface – für Commandline Muffel

1. http://www.heise.de/security/dienste/pgp/keyserver.shtml oder http://www.pca.dfn.de/pgpkserv/#extract
2. Stichwort (Nachname, Mailadresse etc.) in Suchfeld eingeben.
3. In Trefferliste auf passende Key-ID klicken.
4. ASCII incl. `-----BEGIN PGP...` und `-----END PGP...` in Datei speichern.
5. Key importieren: `gpg --armor --import <filename>`
6. Fingerprint und Key-ID überprüfen: `gpg --fingerprint <keyid>`
7. Key signieren: `gpg --sign-key <keyid>`
8. Signierten Key exportieren: `gpg --armor --export <keyid> [ > <filename> ]`
9. ASCII-Output in Formular am Keyserver pasten: http://www.pca.dfn.de/pgpkserv/#submit

gpg --edit-key revkey

Wichtig: Keys am Keyserver können nicht gelöscht werden!

Was also tun bei falschem, komprommitiertem, ungültigem Key?

- Keys, Subkeys, User-IDs und Signatures können zurückgezogen werden (*revoking*).
- Für Keys gibt es `--gen-revoke`: generiert ein Revocation Certificate für den **kompletten** Key.
  Revocation Key am besten gleich beim Erzeugen des Keypaares mit anlegen!
- Sollen Subkeys, User-IDs oder Signaturen revoked werden, gibt es Befehle im `--edit-key` (`revkey`, `revuid` und `revsig`).
- Um Änderung am Keyserver bekannt zu machen, geänderten Key **erneut** auf Keyserver hochladen.

Pflegeanleitung

- **Unbedingt** auf Secret Key gut Acht geben! Kommt der in falsche Hände, sind alle damit behandelten Keys und Zertifikate ungültig und damit quasi unbrauchbar!
- Gleiches gilt im Prinzip für Revocation Key!
- Möglichkeit, einem *fremden* Key zu vertrauen (Trust), damit ist Secret Key teilweise *entbehrlich* und kann weggesperrt werden.
- Weitere Möglichkeit: Secret Key auf Wechselspeicher (USB, CF-Card etc.) und bei Verlassen des Arbeitsplatzes mitnehmen.
- Bei befristeten Keys rechtzeitig für Update des Ablaufdatums sorgen! Signaturen mit abgelaufenem Key sind von wenig Aussagekraft!

Was kommt im zweiten Teil (Juni)?

- Ver- und Entschlüsseln
- Trusts
- Schlüsselpflege
- Keyrings
- Optionen im Config-File

Literatur

- Albrecht Beutelspacher
  Kryptologie
  5. Auflage, Vieweg, 1996
- Jörg Schwenk
  Sicherheit und Kryptographie im Internet
  1. Auflage, Vieweg, 2002
- Simson Garfinkel
  PGP. Pretty Good Privacy
  O'Reilly, 1996
- Deutsche Bank, TU Darmstadt, Universität Siegen, Secude, FZI
  CrypTool – eLearning-Programm für Kryptographie
  http://www.cryptool.de/

Quellen

- GnuPG generell: http://www.gnupg.org/
- HowTo in deutsch: http://www.gnupg.org/(en)/howtos/de/index.html
- iX Magazin für professionelle Informationstechnik, 3/2004, S. 126 ff.
- c't Kryptokampagne: http://www.heise.de/security/dienste/pgp/
- GNU Privacy Projekt: http://www.gnupp.de/
- GnuPG Keysigning Party HOWTO: http://www.cryptnet.net/fdp/crypto/gpg-party.html

Fin

Danke für's Zuhören.

Fragen?

Thanks for Review and Input (alphabetical): Stephan Leicht, Stefan Peinkofer, Bernhard Schneck

Anhang

Während des Vortrags (2004-05-10) wurden einige Punkte diskutiert, die ich teilweise nicht in Vergessenheit geraten lassen möchte:

- Trennung von beruflichem und privatem Key?
  Macht Sinn, ja. Weil: wenn man Arbeitgeber wechselt, werden typischerweise Mailadressen ungültig. Und man will/muss evtl. Keysigning mit unterschiedlichen Leuten haben.
- Zertifizierte Public Keys können sehr gross werden.
  Einige offizielle Stellen (z. B. RIPE) akzeptieren nur Keys bis zu einer bestimmten Maximalgrösse. Evtl. sollte man also achtgeben, nicht Hinz und Kunz zu signieren. Abhilfe: Man signiert einen Institutsschlüssel (den einer CA) und vertraut dem dann. Damit sind alle Keys, die den selben Institutsschlüssel signiert haben ebenfalls vertrauenswürdig.
- Man kann sich mitunter nicht immer aussuchen, mit wem man Keys signen will bzw. muss. Mitgefangen, mitgehangen. Allerdings: Eine Signatur sagt nur was aus zur geprüften Identität, nicht zum geprüften Charakter :-)
- *Private Key* und *Revocation Key* **niemals gemeinsam** aufbewahren! Für beide gilt: gut

wegsperren!

- Signaturen im Key können **nicht** gelöscht werden! Revoking einzelner Signatures ist möglich.
- Revoking Key muss nicht zwingend beim Erzeugen des Keypaares angelegt werden. Allerdings: Man braucht dazu die Passphrase. Wenn man die irgendwann mal vergisst, kann man auch keinen Revoking Key mehr erzeugen!
- Beim Erzeugen eines Revoking Keys kann man aus mehreren Gründen für das Revoken wählen (*expired*, *lost password*, *compromised*, *no special reason*). Der Grund wird im Key dann auch angezeigt.
- Beim Signen nicht nur fremde Angaben auf eigener Keylist kontrollieren, sondern auch eigene Angaben auf fremder Keylist!