GPG — GNU Privacy Guard

David Frey

Was ist GPG?

GPG steht für GNU *Privacy Guard* und ist der OpenGPG-kompatible Nachfolger von PGP [1], Pretty Good Privacy.

GPG ist ein Werkzeug zur Ver- und Entschlüsselung von Daten, insbesondere e-Mail und wurde von **Werner Koch** und anderen entwickelt.

Features

1. Klassische Chiffrierung von lokalen Files, Schutz vor Diebstahl, neugierige Sysadmins. . . . ⇒ `des(1)`-Ersatz.
2. Verschicken und Empfangen von *chiffrierter* e-Mail,
   Beim Senden wird der Brief vor dem Absenden mit dem Public Key des Empfängers verschlüsselt und dort mit dem Secret Key desselben dechiffriert.
   Dies garantiert, dass nur der Empfänger den Brief entschlüsseln kann (*privacy*).
3. *Signieren* von e-Mail,
   Das elektronische Aequivalent der althergebrachten Unterschrift. Mit der Signatur, kann der Empfänger die *Echtheit* (authencity) der Nachricht überprüfen.
4. Verwalten, Generieren, Revoken von Keys
   Zuweisen von Trust, addieren von Signaturen, usw. usf.

Note: Das Signieren ist das am häufigsten verwendete Feature in `gpg`.

Ver- und Entschlüsseln von Botschaften

Folgende Terminologie wird nachfolgend benutzt:

- Plaintext m, die zu verschlüsselnde Nachricht
- Cyphertext c, die verschlüsselte Nachricht,
- Session Key k, der vom Verschlüsselungsalgorithmus benutzte temporäre Transport-Key,
- Public Key P ist der öffentliche Schlüssel,
- Secret Key S ist der geheime Schlüssel,
- Signatur s

Die Schlüssel tragen als Index den Namen des Senders resp. Empfängers, also a für Arthur, b für Beeblebrox etc.

Jeder GPG-Benutzer hat zwei Sorten von Keys, die in *Keyrings* zusammengefasst sind: die Public Keys (die Mail-Partner die GPG benutzen) und in der Defaultkonfiguration 2 Secret Keys: einen zum Signieren (DSA) und einen zum Verschlüsseln (ElGamal).

Klassische Verschlüsselung

Rumpelstilzchen verwendet GPG dazu,um ein wichtiges Dokument zu verschlüsseln. Den dazu verwendeten *Session Key* k muss er geheim halten.

Dieser Anwendungsfall ist für GPG atypisch, man könnte genausogut `des(1)` benutzen.

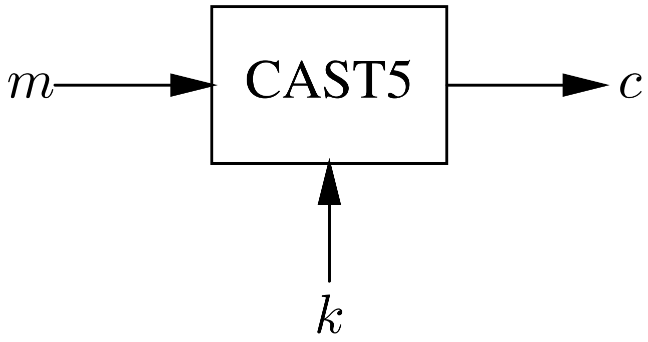


Abbildung 1: Private Verschlüsselung

```
~rumpel$cat secret.txt
Im Wald da ist es finster,
es blüht nicht mal der Gingster!
~rumpel$gpg -ac secret.txt
Enter passphrase: Ach wie gut das niemand weiss, dass ich  Rumpelstilzchen heiss!
Repeat passphrase: Ach wie gut das niemand weiss, dass ich  Rumpelstilzchen heiss!
~rumpel$cat secret.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.7 (GNU/Linux)

jA0EAwMCW3fRVsEtMEJgyVdwsJd6/uRgsKdCfLQ29D6ju1eVQ71OGxLVL/4QuNvL
+eMvIgVKT0QWCzB8JGkbTdcHjVrutBcoHkpt3L+OudCtQgIb0zHDNTiiXCILqt9m
C1Y8XYTbHOk=
=cPqt
-----END PGP MESSAGE-----
```

Klassische Entschlüsselung

```
~rumpel$gpg -d < secret.txt.asc
gpg: CAST5 encrypted data
Enter passphrase: Ach wie gut das niemand weiss, dass ich Rumpelstilzchen heiss!
Im Wald da ist es finster,
es blüht nicht mal der Gingster!
~rumpel$
```

Versand von verschlüsselter Mail

Arthur sendet Beeblebrox eine Mail, will aber sicherstellen, dass nur Beeblebrox die Mail lesen kann, und nicht jedermann, der im Mailpfad liegt.

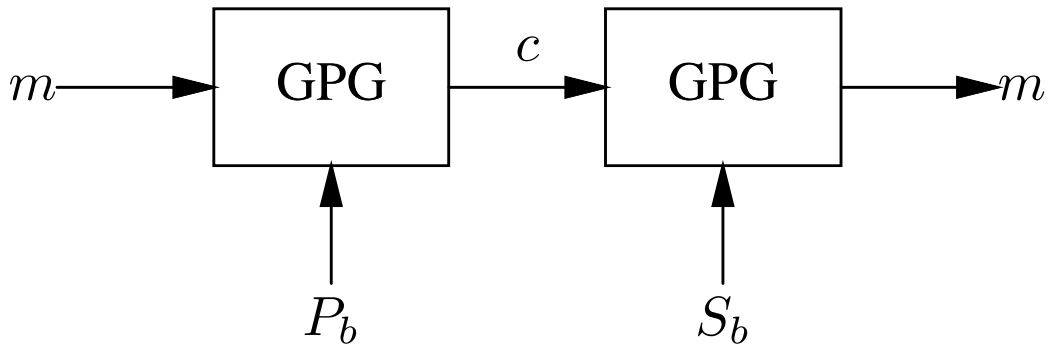


Abbildung 2: Chiffrierung

```
~arthur$cat heartofgold
Hoi Zaphod,

Du weisst, dass Du die Heart of Gold stehlen musst?

Gruss,
  Arthur Dent
~arthur$gpg -ear Beeblebrox < heartofgold|mail -s "Heart of Gold" beeblebrox
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: 81AF00A8: There is no indication that this key really belongs to the owner
1024g/81AF00A8 2002-06-16 "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
             Fingerprint: 8D81 D82C CD6C 53FE 7287  F208 187E 0C17 81AF 00A8

It is NOT certain that the key belongs to its owner.
If you *really* know what you are doing, you may answer
the next question with yes

Use this key anyway? yes
~arthur$
```

Empfang von verschlüsselter Mail

```
~beeblebrox$mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/beeblebrox": 1 message 1 new
>N  1 arthur@eos.lugs.c  Sun Jun 16 23:15   26/1069  Heart of Gold
&1
Message 1:
From arthur@eos.lugs.ch  Sun Jun 16 23:15:28 2002
Delivered-To: beeblebrox@eos.lugs.ch
From: arthur@eos.lugs.ch (Arthur Dent)

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.7 (GNU/Linux)

hQEOAxh+DBeBrwCoEAQAi1Koc98qcM9USrUpNxizPv0SePDiSEZIoUKy9MVRjmyA
cGGu93Z7S4VL5eZW2qKAIxC2EPvUK4hhHDUIEolWs52s+FVhg8jJ6EWjWfwhXJQ2
6A/nCkqtiRFj4N8FcMYt3MoQWNBHrcpHJZhlFMbPf6xyBq+sdREQNr4jDt5o0dcD
/A3n5zwFx6iLogANyezjZhEB7TgTYLJlZysKqgerFv1irwZ5ffIWM6TncDQnuJ7S
wzA+4FubSvkBsWlpNT4ambKNND/I3VMA5JAsBS/Tpr4buha0x5NTYuE6+qeXDUi9
HJPhectu8Zjy+oqusN+YGB+9W9qjdSHZYPYDGQ1Kk7wz0o4BBr2zM8u8ZP18M7DZ
YF9VjdWq2+1PVb+RmPCFp9n4zYPuguJjhJDm6l3c+U+RbxBLoZtFwG9xYYQEcZ8P
7IVT886tnPGvJ4QfqRKR7XhGkZxnWmVKn5CYi1Z+3xlVCpcrX2HFsTHRjbKhctci
emXhu5cnE6dVCN1mfG163k9rPGJoh7ExU/7u+Dqn/h03
=M5z2
-----END PGP MESSAGE-----

&|1 gpg

You need a passphrase to unlock the secret key for
user: "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
1024-bit ELG-E key, ID 81AF00A8, created 2002-06-16 (main key ID 650E1ECF)

Enter passphrase: I am President!
gpg: encrypted with 1024-bit ELG-E key, ID 81AF00A8, created 2002-06-16
      "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
Hoi Zaphod,

Du weisst, dass Du die Heart of Gold stehlen musst?

Gruss,
  Arthur Dent
& q
Saved 1 message in /home/beeblebrox/mbox
~beeblebrox$
```

Versand von verschlüsselter und signierter Mail (1)

Eine Mail soll verschlüsselt *und* signiert werden, so dass

1. sie nur der Empfänger lesen kann, und
2. der Empfänger weiss, dass die Nachricht vom Sender stammt.

Das Beispiel in Abbildung 2 hat einen Haken: Beeblebrox weiss nicht, ob Arthur diese Mail wirklich geschrieben hat, oder ob es ein anderer in seinem Namen tat. Um dies zu verhindern, signiert Arthur seine Mail mit seinem Secret Key.

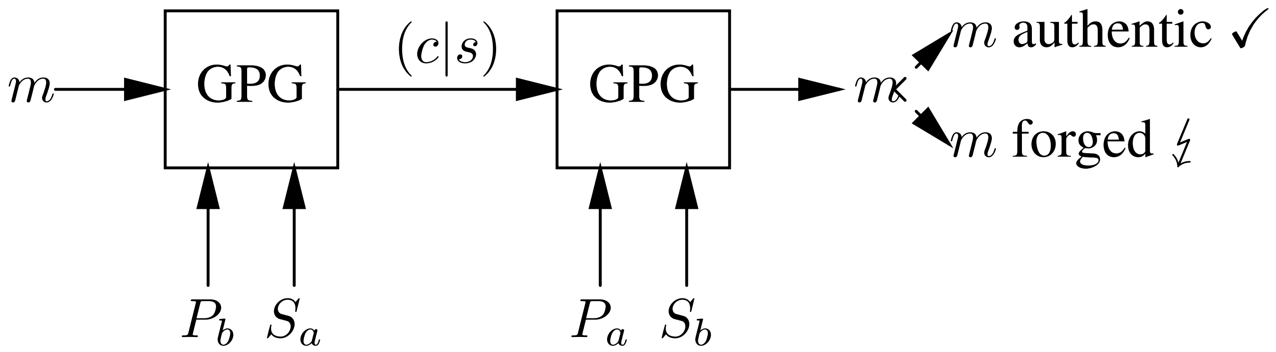


Abbildung 3: Chiffrierung und Signierung

Note: Dies ist der zweithäufigste Betriebsfall.

Versand von verschlüsselter und signierter Mail (2)

```
~arthur$gpg -esa -r beeblebrox|mail -s "Reply" beeblebrox

You need a passphrase to unlock the secret key for
user: "Arthur Dent <adent@mostly.harmless.earth>"
1024-bit DSA key, ID D8FBFB4E, created 2002-06-16

Enter passphrase: I want tea!
gpg: 81AF00A8: There is no indication that this key really belongs to the owner
1024g/81AF00A8 2002-06-16 "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
             Fingerprint: 8D81 D82C CD6C 53FE 7287  F208 187E 0C17 81AF 00A8

It is NOT certain that the key belongs to its owner.
If you *really* know what you are doing, you may answer
the next question with yes

Use this key anyway? yes
Aber nicht so plump, jetzt haben wir die Polizei am Hals!

Arthur
~arthur$
```

Empfang von verschlüsselter und signierter Mail (1)

```
~beeblebrox$mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/beeblebrox": 2 messages 2 new
>N  1 arthur@eos.lugs.c  Sun Jun 16 23:31   27/1134  Reply
& 1
Message 1:
From arthur@eos.lugs.ch  Sun Jun 16 23:31:57 2002
Delivered-To: beeblebrox@eos.lugs.ch
From: arthur@eos.lugs.ch (Arthur Dent)

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.7 (GNU/Linux)

hQEOAxh+DBeBrwCoEAP6Asudi7KwRsLiqQOUsUES+QWuu42KIsuFeGhAm/JK/mFV
CC76pFfC8BYwV436892OoHIEyLUCjQQsZhpzelYjJC+RuJnhIh3t/utxx7HEfu3B
Ex0G9Q77NhT/9Q52vfFSUI0PDTOil2GkUloETRxGd0uCHKfJpq1Yld0USVTRSngD
/08VoFGUcG00N3uZS+f0I80VvClgx663DiTIO99lCDOw+sR5hSANOfvUtijDqlIq
lxmDK95+zOQOEPdTTLHEyLsNEREGgCtMmf5g9DKjQ/shNzTmPY43X8MAoQzoK8/4
Qbs+W6cHRwhCbIYn3iQQemF40XpwnIiB9yViX9jrJ/ej0sACASjmJKhWBHDzfzhB
mWMHPYyZN1yI9p7pDflDqMNr4bj6ABdFBFU6pk8BGSOwocanuIepRMNtut8HB4N7
GEgf/+Qsf/AV4vCm9lZWm0RZ1k2ncmUzqvz4o3B5F567WkAsWXrm5ECQ9cM1D7KH
buPFH8x4iryh+oiRUEB4iQx4NTbwAACuwKg8bwKGvItw9vvw7XAxHfyVWKGBnDoM
xguP+/LlNyeeVdQGulf4TQ0MpMou+Qhs8aV0gRA1CJqDjHC/owk=
=1tJ+
-----END PGP MESSAGE-----
```

Empfang von verschlüsselter und signierter Mail (2)

```
& |1 gpg

You need a passphrase to unlock the secret key for
user: "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
1024-bit ELG-E key, ID 81AF00A8, created 2002-06-16 (main key ID 650E1ECF)

Enter passphrase: I am President!
gpg: encrypted with 1024-bit ELG-E key, ID 81AF00A8, created 2002-06-16
      "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
Aber nicht so plump, jetzt haben wir die Polizei am Hals!

Arthur
gpg: Signature made Sun Jun 16 23:31:57 2002 CEST using DSA key ID D8FBFB4E
gpg: Good signature from "Arthur Dent <adent@mostly.harmless.earth>"
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Fingerprint: 9110 FB0B 622E 4B93 90BC  0A6F D285 7F30 D8FB FB4E
& exit
~beeblebrox$
```

Signieren von Dokumenten (1)

Vielfach ist der Inhalt des Dokuments nicht geheim und man will nur sicher sein, dass der/die Empfänger(in) sicher ist/sind, dass die Mail authentisch ist. Dann ist die Verschlüsselung überflüssig.

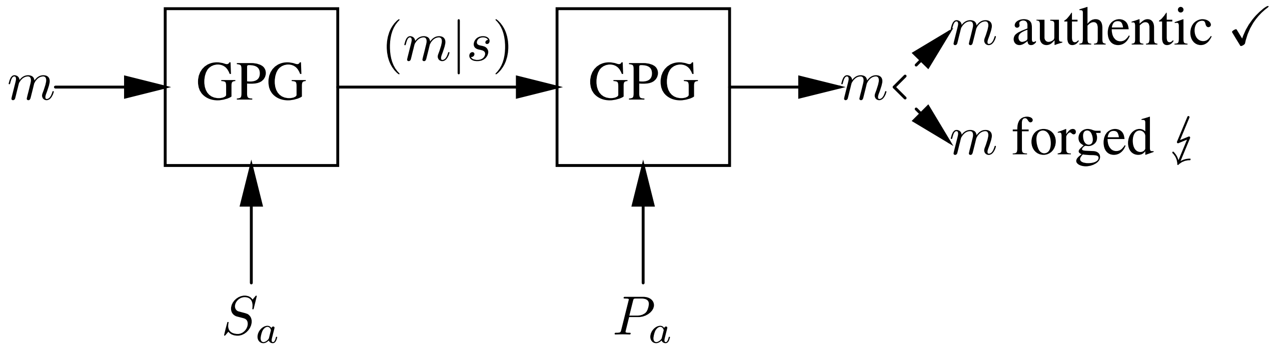


Abbildung 4: Signierung

Bemerkung: Bei eingehenderer Betrachtung stellt der/die Leser/-in fest, dass das *Signieren* den umgekehrten Prozess zum Chiffrieren darstellt: zum Verschlüsseln wird der Secret Key verwendet, was die Authenzität sicherstellt.

Note: Dies ist momentan der häufigste Betriebsfall.

Signieren von Dokumenten (2)

```
~arthur$cat restaurant
Wann treffen wir uns im Restaurant am Ende des Universums?
Ich habe Hunger

Gruss,
  Arthur
~arthur$gpg -sar beeblebrox < restaurant|mail -s "Restaurant" beeblebrox
gpg: WARNING: recipients (-r) given without using public key encryption
You need a passphrase to unlock the secret key for
user: "Arthur Dent <adent@mostly.harmless.earth>"
1024-bit DSA key, ID D8FBFB4E, created 2002-06-16

Enter passphrase: I want tea!
~arthur$
```

Signieren von Dokumenten (3)

```
~beeblebrox$mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/beeblebrox": 3 messages 3 new
>N  1 arthur@eos.lugs.c  Mon Jun 17 00:29   21/753   Restaurant
& 1
Message 1:
From arthur@eos.lugs.ch  Mon Jun 17 00:29:31 2002
Delivered-To: beeblebrox@eos.lugs.ch
From: arthur@eos.lugs.ch (Arthur Dent)

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.7 (GNU/Linux)

owGbwMvMwCR4qbXe4Mbv336MpwuTGGx5Bb2TE0sUilKLSxJLixLzSrjCE/PyFEqK
UtPSUvMUyjOLFErzihUycxWC4EoUEnMVXPNSUhVSUosVQvMyy1KLiktzi+25PJMz
FDISk1IVPErz0lOLuLjci0qLi3W4FBQci0oySou4OuyZWcF2whwhyPT1LsOCNTne
xkwiiY8z/DqsY+3ECq+WJ65kmKf5y+7Hi62rc90/K3P+Pb5q/pOgmX8B
=5Arh
-----END PGP MESSAGE-----

& |1 gpg
Wann treffen wir uns im Restaurant am Ende des Universums?
Ich habe Hunger

Gruss,
  Arthur
gpg: Signature made Mon Jun 17 00:29:31 2002 CEST using DSA key ID D8FBFB4E
gpg: Good signature from "Arthur Dent <adent@mostly.harmless.earth>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Fingerprint: 9110 FB0B 622E 4B93 90BC  0A6F D285 7F30 D8FB FB4E
& quit
Saved 1 message in /home/beeblebrox/mbox
Held 1 message in /var/mail/beeblebrox
~beeblebrox$
```

Erzeugen eines Secret Keys (1)

Um Mails unterschreiben zu können und chiffrierte Mails dechiffrieren zu können, muss man zuerst einen Secret Key erzeugt haben:

```
arthur$gpg --gen-key
gpg (GnuPG) 1.0.7; Copyright (C) 2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: /home/arthur/.gnupg: directory created
gpg: /home/arthur/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
arthur$gpg --gen-key
gpg (GnuPG) 1.0.7; Copyright (C) 2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: keyring '/home/arthur/.gnupg/secring.gpg' created
gpg: keyring '/home/arthur/.gnupg/pubring.gpg' created
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) ElGamal (sign and encrypt)
   (5) 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) 1024
Requested keysize is 1024 bits
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
```

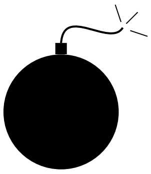
```
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct (y/n)? y

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: Arthur Dent
Email address: adent@mostly.harmless.earth
Comment:
You selected this USER-ID:
    "Arthur Dent <adent@mostly.harmless.earth>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

Enter passphrase:I want tea!
Repeat passphrase: I want tea!
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.
+++++++++++++++++++++++++++++++++++++++++++++..+++++..+++++++++++++++..++++++++++.+++

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 290 more bytes)
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/arthur/.gnupg/trustdb.gpg: trustdb created
public and secret key created and signed.
key marked as ultimately trusted.

pub  1024D/D8FBFB4E 2002-06-16 Arthur Dent <adent@mostly.harmless.earth>
     Key fingerprint = 9110 FB0B 622E 4B93 90BC  0A6F D285 7F30 D8FB FB4E
sub  1024g/166A37CF 2002-06-16
```

Erzeugen eines Secret Keys (2)

Neu erzeugte Keys sollten mindestens 1024 bits lang sein.

Nachdem man für sich einen Schlüssel generiert hat, sollte man verifizieren, dass er eine *Self-Signature* trägt. Damit wird verhindert, dass spätere Aenderungen am Key undetektiert bleiben (cf. [2])

Revocation-Request



Nachdem man den Secret-Key erzeugt hat, sollte man sofort einen **Revokation-Request** erzeugen (`gpg -gen-revoke`), um ihn später falls nötig, sofort zurückziehen zu können.

Ein Revokation-Request erklärt einen komprimittierten/verlorengegangen Key ungültig. Dazu wird er in den eigenen Keyring aufgenommen und der Key auf einen Keyserver exportiert.

Note: Die User-IDs sind von jedem ohne Eingabe eines Passwords editierbar.

Dies macht ist nur für importierte, `pgp`-Keys von Belang, da `gpg` schon bei der Erzeugung des Keys, diesem eine Self-Signature anhängt.

Einfügen eines Public Keys in den Keyring

Um Mails chiffriert versenden zu können und um Signaturen überprüfen zu können, braucht man den Key des Empfängers.

Damit Zaphon Beeblebrox Arthurs's Key verifizieren kann, muss er in vorher in seinen Keyring einfügen:

```
~beeblebrox$gpg --import arthur.asc
gpg: key D8FBFB4E: public key imported
gpg: key 650E1ECF: not changed
gpg: Total number processed: 2
gpg:               imported: 1
gpg:              unchanged: 1
```

Arthur kann den Inhalt seines Keyrings folgendmassen anschauen.

```
~arthur$gpg --list-sigs
/home/arthur/.gnupg/pubring.gpg
-------------------------------
pub  1024D/D8FBFB4E 2002-06-16 Arthur Dent <adent@mostly.harmless.earth>
sig 3       D8FBFB4E 2002-06-16   Arthur Dent <adent@mostly.harmless.earth>
sig 3       650E1ECF 2002-06-17   Zaphon Beeblebrox <beeblebrox@galactic.gov>
sub  1024g/166A37CF 2002-06-16
sig         D8FBFB4E 2002-06-16   Arthur Dent <adent@mostly.harmless.earth>

pub  1024D/650E1ECF 2002-06-16 Zaphon Beeblebrox <beeblebrox@galactic.gov>
sig 3       650E1ECF 2002-06-16   Zaphon Beeblebrox <beeblebrox@galactic.gov>
sub  1024g/81AF00A8 2002-06-16
sig         650E1ECF 2002-06-16   Zaphon Beeblebrox <beeblebrox@galactic.gov>
~arthur$
```

Einstufung eines erhaltenen Public Keys

Einem Public Key haften implizit folgende Eigenschaften an:

1. *Authencity*: Gehört der Key wirklich derjenigen Person, die sie vorzugeben scheint? ("Trust")
2. *Trustability*: Kann ich dieser Person trauen? ("Owner-Trust")
   Diese Frage ist unabhängig von 1. Es ist sehr wohl möglich, dass ich einen authentischen Key von einer moralisch nicht integren Person bekommen habe…

Bisher gingen wir der Einfachheit immer von einer vollständiger Glaubwürdigkeit aus.

Einstufung eines erhaltenen Public Keys

GPG/PGP kennen verschiedene Stufen von Glaubwürdigkeit:

- ▷ I will not answer. (default) ⇒ undefined
- ▷ I have not checked at all. ⇒ untrusted
- ▷ I have done casual checking. ⇒ marginal
- ▷ I have done very careful checking. ⇒ complete

Tabelle 1: Validity ("Trust")-Stufen

- ▶ Don't know ⇒ unknown (q)
- ▶ I do NOT trust ⇒ none (n)
- ▶ I trust marginally ⇒ marginal (m)
- ▶ I trust fully ⇒ full (f)
- ▶ I trust ultimately ⇒ ultimate (u)

Tabelle 2: (Owner-)Trust-Stufen

Note: Die Owner-Trust-Information wird ist privat und wird nicht weiterverteilt.

Signieren eines erhaltenen Public Keys

```
~beeblebrox$gpg --sign-key "Arthur Dent"

pub  1024D/D8FBFB4E  created: 2002-06-16 expires: never      trust: -/-
sub  1024g/166A37CF  created: 2002-06-16 expires: never
(1). Arthur Dent <adent@mostly.harmless.earth>


pub  1024D/D8FBFB4E  created: 2002-06-16 expires: never      trust: -/-
             Fingerprint: 9110 FB0B 622E 4B93 90BC  0A6F D285 7F30 D8FB FB4E

     Arthur Dent <adent@mostly.harmless.earth>

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? 3
Are you really sure that you want to sign this key
with your key: "Zaphon Beeblebrox <beeblebrox@galactic.gov>"

I have checked this key very carefully.

Really sign? y

You need a passphrase to unlock the secret key for
user: "Zaphon Beeblebrox <beeblebrox@galactic.gov>"
1024-bit DSA key, ID 650E1ECF, created 2002-06-16

Enter passphrase: I am President!
~beeblebrox$
```

Wie kommt man an Public Keys?

Dies ist der Kernpunkt. Folgende Möglichkeiten, in absteigender Reihenfolge der Präferenz, existieren:

1. Den des Public Keys *direct* von der Person auf Papier/Diskette erhalten. Man sieht die Person und kann daher zu 100% annehmen, dass der Key zur Person gehört[a]
2. Mit `finger` *user@somewhere* den Key abholen oder von der Webpage der Person abschreiben (copy&paste). Geht man, davon aus, dass das Account wirklich der Person gehört, so wird der Key stimmen. . . . Alternativ dazu kann man sich den Key via e-Mail zuschicken lassen.
3. Als letzter Ausweg bietet sich noch ein *Keyserver* an. Keyserver sind Behälter von Public Keys, wobei keine Garantie der Echtheit gegeben wird.
   - `wwwkeys.eu.pgp.net`
   - `keyring.debian.org`

[a] Ist der Public Key gefälscht, so ist die Person dann selber schuld, da sie die Nachricht nicht entschlüsseln kann.

Schlussbemerkungen

Dieser Artikel besprach die direkte Verwendung von GPG, d.h. GPG wurde direkt von der Commandline aus aufgerufen, ohne Einbindung in irgendwelche Mailers. Es gibt seit längerem ($\geq 7\,\mathrm{y}$) Mailer mit integriertem GPG-support. In einem solchen Fall kann man bei abzusendender Mail wählen, ob man die Mail verschlüsseln, signieren oder auch beides will, und gegebenenfalls den Pass-Phrase eingeben. Bei ankommender Mail wird typischerweise nachgeschaut ob die Mail GPG-verschlüsselt ist und/oder einen Key angehängt hat und falls ja entschlüsselt und/oder den Benutzer gefragt ob er den Key in seinen Keyring einfügen will.

Trust

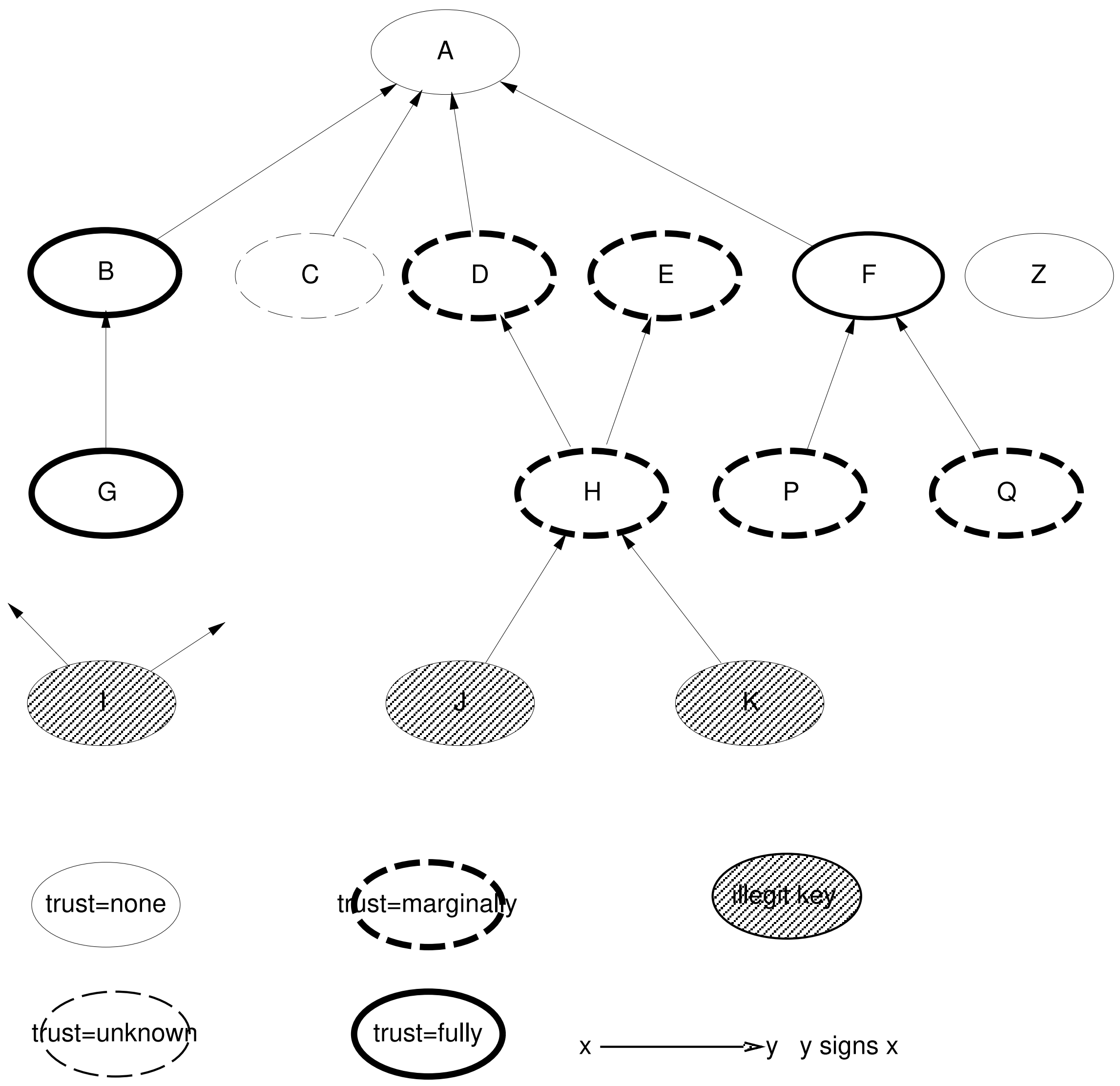


Abbildung 5: Trust-Graph (cf. [3], p. 586)

Mathematischer Hintergrund (Encryption)[3], p. 476ff

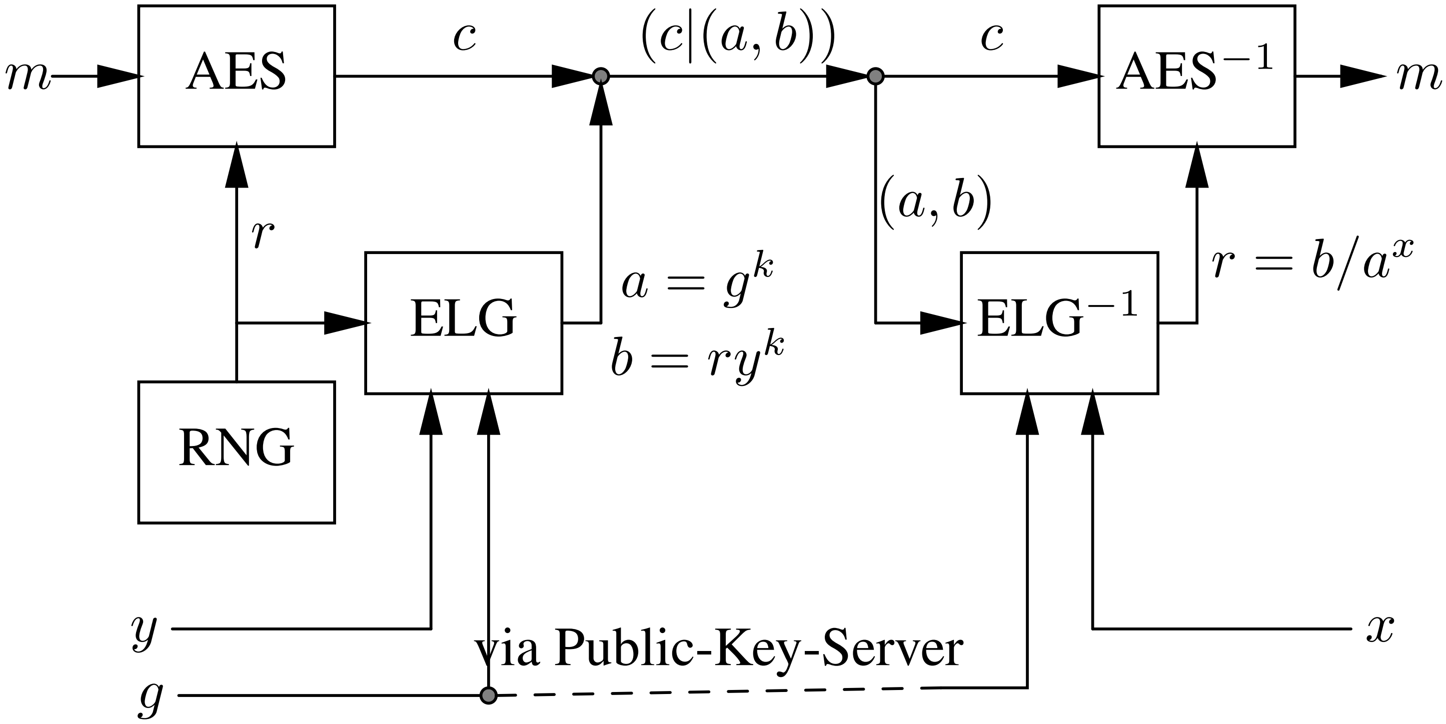


Abbildung 6: Implementierungsdetail

Wieso funktioniert das ganze?

(Alle Arithmetik wie üblich $\mod p$ und k relativ prime zu $p-1$)

$$r = b/a^x \overset{b=ry^k}{\overset{\downarrow}{=}} ry^k/a^x \overset{y=g^x}{\overset{\downarrow}{=}} rg^{xk}/a^x \overset{a=g^k}{\overset{\downarrow}{=}} rg^{xk}/g^{kx} \equiv r$$

g,y stammen aus dem Public Key, x aus dem Secret Key.

Literatur

[1] The Internet Society. *RFC2440: OpenPGP Message Format*, November 1998.

[2] Mike Ashley et al. *The GNU Privacy Handbook*. 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA, 1999.

[3] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc, New York, second edition, 1996.
Comprehensive survey of modern cryptography.
ISBN 0-471-11709-9.

[4] Brenno J.S.A.A.F. de Winter and Michael Fischer v. Mollard. *Gnu Privacy Guard (GnuPG) Mini Howto*, April 1999.

[5] Douglas Adams. *The Hitch Hikers's Guide to the Galaxy: A Trilogy in Five Parts*. William Heinemann Ltd, Pan Books Ltd.