Gnu Privacy Guard (GnuPG) Mini Howto

Michael Fischer v. Mollard <fischer@math.uni-goettingen.de> Version 0.1.1 vom 12. Februar 1999

Dieses Dokument beschreibt die Benutzung von GNU Privacy Guard (GnuPG), einem freien, OpenPGP kompatiblen Verschlüsselungssystem. Damit das Programm wirklich frei ist, wurde auf patentierte Algorithmen wie RSA verzichtet.

Bemerkung für die gedruckten Versionen: *Aufgrund von technischen Schwierigkeiten wird der doppelte Bindestrich, der die langen Optionen von GnuPG einleitet, in in Absätzen eingebetteten Text als ein einzelner dargestellt. Lesen Sie also bei langen Optionen (also denjenigen, die aus mehr als einem Buchstaben bestehen) zwei Bindestriche.*

Inhaltsverzeichnis

1 Konzepte

1.1 Public Key Verschlüsselung

Klassische Methoden zur Verschlüsselung benutzen nur einen Schlüssel. Der Sender verschlüsselt seine Nachricht mit diesem Schlüssel, und der Empfänger entschlüsselt ihn mit demselben wieder. Damit das funktioniert, muß der Empfänger vorher den Schlüssel bekommen haben, und zwar auf einem sicheren Kommunikationskanal, da sonst Unbefugte in Kenntnis des Schlüssels gelangen könnten. Also braucht man einen sicheren Kommunikationskanal, aber wenn man den hat, braucht man auch nicht mehr zu verschlüsseln.

Public Key Verfahren (auch: asymmetrischen Verfahren) beseitigen dieses Problem, indem zwei Schlüssel erzeugt werden: Der öffentliche, der über beliebige Kommunikationskanäle verschickt werden kann und der private, den nur der Besitzer kennt. Idealerweise ist der private Schlüssel nicht mit dem öffentlichen rekonstruierbar. Der Sender verschlüsselt die Nachricht mit dem öffentlichen Schlüssel des Empfängers. Entschlüsselt wird die Nachricht dann mit dem privaten Schlüssel des Empfängers. Nach diesem Schema kann man demnach effektiv verschlüsseln, ohne über einen sicheren Kommunikationskanal zu verfügen.

Ein ganz wichtiger Punkt ist aber die Geheimhaltung des privaten Schlüssels. Er darf auf keinen Fall in fremde Hände geraten, auch nicht über das Netz verbreitet werden. GnuPG via `telnet` zu benutzen, ist zum Beispiel eine ziemlich schlechte Idee. (Eigentlich sollte man `telnet` sowieso durch `ssh` ersetzen)

1.2 Digitale Unterschriften

Digitale Unterschriften sollen die Authenzität einer Nachricht beweisen. Würden Nachrichten von offizieller Seite signiert, wäre es deutlich schwerer, mit gefälschten Nachrichten Unruhe oder Schaden anzurichten (aktuelles Beispiel: Ein trojanische Pferd, verschickt als Patch eines bekannten Webbrowsers).

Ein digitale Signatur wird mit Hilfe des privaten Schlüssels aus dem Text erzeugt. Diese kann dann vom Empfänger mit dem öffentlichen Schlüssel des Senders überprüft werden. Dabei wird nicht nur der Absender (nur der kennt den privaten Schlüssel) überprüft, sondern auch, ob der Text unverändert angekommen ist.

1.3 Web of trust

Eine Schwachstelle der Public Key Algorithmen ist die Verbreitung der öffentlichen Schlüssel. Ein Benutzer könnte einen öffentlichen Schlüssel mit falscher User ID in Umlauf bringen. Wenn dann mit diesem Schlüssel Nachrichten kodiert werden, kann der Eindringling die Nachrichten dekodieren und lesen. Wenn er sie dann noch mit einem echten öffentlichen Schlüssel kodiert an den eigentlichen Empfänger weiterleitet, fällt dieser Angriff nicht einmal auf.

Die von PGP (und damit auch von GnuPG) gewählte Lösung besteht im Unterschreiben von Schlüsseln. Ein öffentlicher Schlüssel kann von anderen Leuten unterschrieben werden. Diese Unterschrift bestätigt, daß der Schlüssel zu der in der UID angegebenen Person gehört. Der Benutzer kann festlegen, welchen Unterschriften er wie weit traut. Vertrauen ist dabei zwar reflexiv, aber nicht symmetrisch und transitiv. Ein Schlüssel gilt als vertrauenswürdig, wenn er von Leuten unterzeichnet wurde, denen man vertraut. Wenn man Schlüssel unterzeichnet, sollte man sich sicher sein, daß man die Identität desjenigen, dessen Schlüssel man unterschreibt, genau kennt. Eine Möglichkeit ist es, den Schlüssel persönlich bekommen zu haben, eine andere, den Fingerprint über zuverlässige Kanäle zu vergleichen.

1.4 Grenzen der Sicherheit

Wenn man Daten vertraulich halten will, sollte man sich nicht nur Gedanken über die Sicherheit des Verschlüsselungsalgorithmus machen, sondern über die Systemsicherheit allgemein. Die in GnuPG verwendeten Algorithmen gelten gemeinhin als nicht zu knacken. Daraus zu schließen, daß alle verschlüsselten Daten sicher seien, ist naiv. Es gibt auch noch andere Formen von Angriffen. Anfang Februar 1999 tauchte zum Beispiel ein Word Trojaner auf, der private PGP Schlüsselbunde auf der Festplatte suchte und via ftp verschickte (Meldung im Heise Newsticker vom 03.02.99). Ein privates Schlüsselbund läßt sich, insbesondere bei schlechtem Passwort, deutlich leichter knacken als eine einzelne Datei.

Denkbar sind auch Trojaner, die Tastatureingaben weiterleiten. Falls man die Nachrichten entschlüsselt auf dem Rechner lagert, können sie natürlich auch gelesen werden. Aufwendiger, aber technisch möglich ist es, die Abstrahlung des Monitors zu messen und sichtbar zu machen, so daß der Bildschirminhalt mitgelesen werden kann. Dann nützt es auch nichts, eine verschlüsselte Datei nur zum Lesen zu entschlüsseln. Zum Thema "Überwachung" gibt es den interessanten Artikel "Abhör-Dschungel" aus der c't 5/98, Seite 82 und "In die Röhre geguckt" c't 24/98, Seite 90.

Die obigen Möglichkeiten sollen keine Paranoia hervorrufen, sondern nur darauf hinweisen, daß Verschlüsselung von Daten nur ein Baustein eines Sicherheitskonzeptes sein kann. Um so erstaunlicher, daß es immer wieder Versuche gibt, Verschlüsselung von Daten zu be- beziehungsweise zu verhindern.

2 Installation

2.1 Bezugsquellen

Die offizielle Bezugsquelle ist die *GnuPG Homepage* `<http://d~ngm~~~we~r~vygt~~g.roads-uae.com/download.html>`. Dort gibt es auch eine Liste der Mirrors.

Aus rechtlichen Gründen darf GnuPG nicht aus Servern in den USA geladen werden, da Kryptographie in den USA als Rüstungsgut gilt. Aus diesem Grund gibt es übrigens PGP immer in nationalen und internationalen Versionen, wobei bei letzteren der Sourcecode in Buchform exportiert wird und er in Oslo wieder eingescant wird. Genaueres dazu auf der *Internationalen PGP Homepage* `<http://d~ngm~~~u6cvba~.roads-uae.com>`. Nichtsdestotrotz darf GnuPG in die USA eingeführt und benutzt werden, es darf dort auch auf ftp-Servern abgelegt werden. Es muß dabei nur garantiert werden, daß GnuPG nicht reexportiert wird.

Falls man schon eine lauffähige GnuPG oder PGP Version hat, sollte man die Signatur des Archivs überprüfen (siehe 5 (Signaturen)).

2.2 Konfigurieren

GnuPG wird als Debian Package, als RedHat rpm oder als Sourcecode vertrieben. In den ersten beiden Fällen nimmt man die entsprechenden Werkzeuge zum installieren (ich kenne nur rpm), alles weitere bezieht sich auf den Sourcecode.

Da die Entwicklung unter Linux (x86) stattfindet, ist die Übersetzung dort meist gar kein Problem. Eine aktuelle Liste der offiziell unterstützten Betriebssysteme steht auch auf der *GnuPG Homepage*. Das folgende Vorgehen gilt aber betriebssystemunabhängig.

Nachdem das Archiv mit

```
tar xvzf gnupg-?.?.?.tar.gz
```

entpackt ist, tippt man

```
./configure
```

Dabei sollte nichts verblüffendes passieren. Mit

```
./configure --help
```

kann man sich, falls nötig, die möglichen Konfigurationsparameter ansehen. Falls es Probleme mit der der Internationalisierung (gettext) geben sollte, kann man mit `-with-included-gettext` eine mitgelieferte Version benutzen oder sie mit `-disable-NLS` abschalten.

2.3 Kompilieren

Danach sollte

```
make
```

problemlos laufen. Falls es dabei wider Erwarten Probleme gibt, sollte man (in dieser Reihenfolge): Selbst probieren (natürlich mit Lesen der Dokumentation), jemanden in der Nähe fragen, der sich auskennt und schließlich auf der Mailingliste (siehe 6 (Informationsquellen)) um Rat fragen. Falls es sich nur um falsche Pfade handelt, sollte man mit `make clean` (oder rabiater) das Verzeichnis säubern, neu konfigurieren und es dann noch einmal versuchen.

2.4 Einrichten

Mit

```
make install
```

werden die Programme und die Manpage in die dafür vorgesehenen Verzeichnisse kopiert. In `usr/local/share/gnupg/` (oder in dem bei ./configure angegebenen Verzeichnis) liegt die Datei `options.skel`. Wenn man diese nach `~/.gnupg/options` kopiert, werden die entsprechenden Einstellungen als Standard benutzt. Das Kopieren sollte eigentlich beim Anlegen von `~/.gnupg/` automatisch passieren. Jeder mögliche Eintrag ist gut dokumentiert, deshalb werden sie hier nicht beschrieben.

Man kann GnuPG als suid root laufen lassen (das heißt: das Programm läuft mit allen Rechten des Superusers), damit die Möglichkeit ausgeschlossen wird, daß Teile des Programmes ausgelagert werden und dann gelesen werden können. Wie real diese Gefahr ist, kann ich nicht beurteilen, allerdings ist auch mit suid root Vorsicht geboten, da ein trojanisches Pferd mit suid root beliebigen Schaden anrichten kann. Man kann die Warnmeldung, die ausgegeben wird, falls GnuPG nicht suid root läuft, mit `no-secmem-warning` in `~/gnupg/options` abschalten.

3 Umgang mit Schlüsseln

3.1 Erzeugen

Mit

```
gpg --gen-key
```

wird ein neues Schlüsselpaar erzeugt. Als Erstes wird man nach dem zu verwendenden Algorithmen gefragt. Genaueres zu den Algorithmen steht in der *PGP DH vs. RSA FAQ* `<http://d~ngm~a4~6krce~h.roads-uae.com.uk/ss/pgpfaq.html>` oder in 6.4 (Applied Cryptography). Man kann (und sollte) einfach den default Wert (DSA/ElGamal) nehmen.

Bei der Schlüssellänge muß man zwischen Sicherheit und Rechenzeit abwägen. Je länger ein Schlüssel, desto sicherer ist er, desto länger dauern aber auch Operationen mit ihm. Bei der Rechenzeit muß man aber berücksichtigen, daß der Schlüssel möglicherweise auch noch in einigen Jahren benutzt werden soll, wenn die durchschnittliche Rechenleistung stark angewachsen sein wird. GnuPG fragt ab einer Schlüssellänge von mehr als 1536 Bits, ob ein so großer Schlüssel wirklich nötig sei, andere Leute empfehlen mindestens 2048 Bits. Für DSA ist 1024 Bits Standard.

Dann wird nach Namen, Kommentar und Email Adresse gefragt. Anhand dieser Angaben wird der Schlüssel identifiziert. Man kann die Angaben aber später noch ändern beziehungsweise ergänzen. Siehe 3.5 (Schlüsselbund verwalten) Man sollte eine länger gültige Email Adresse wählen, da die komplette Benutzerkennung unterschrieben wird. Wird dann etwas geändert, gelten die Unterschriften unter die geänderten Angaben nicht mehr.

Als letztes wird nach dem Passwort (beziehungsweise Passsatz (in der deutschen Übersetzung: Mantra) denn es können Leerzeichen vorkommen) gefragt, mit dem der private Schlüssel gesichert werden soll. *Verwenden sie ein gutes Mantra*. Ein gutes Mantra ist

- nicht zu kurz,
- enthält Sonderzeichen,
- ist kein Name und
- nicht mit Kenntnis des Benutzers leicht zu erraten (wie Telefonnummer, Bankleitzahl, Name und Anzahl der Kinder, ...)

Man kann durch willkürlich eingestreute GRoß/KlEinSchReibung und Leerzeichen weitere Sicherheit erhalten. Außerdem muß man es sich merken können, da der geheime Schlüssel ohne Mantra wertlos ist. Es kann in diesem Zusammenhang ein guter Gedanke sein, gleich ein Rückrufzertifikat zu erstellen. Siehe 3.4 (Widerrufen).

3.2 Exportieren

Mit

```
gpg --export [UID]
```

wird der Schlüssel mit der User ID UID exportiert. Wird keine UID angegeben, so wird der ganze Schlüsselbund exportiert. Voreingestellt ist Ausgabe auf `stdout`, man kann aber mit der Option `-o [Datei]` in eine Datei ausgeben. Es empfiehlt sich noch, mit der Option `-a` (`-armor`) zu arbeiten, da ich andernfalls Probleme hatte. Mit dieser Option werden die Schlüssel nicht im Binärformat ausgegeben, sondern als ASCII (7 Bit) Dateien.

Den exportierten Schlüssel kann man dann in der Welt verbreiten, wahlweise auf der Homepage, via finger, über Keyserver, ...

3.3 Importieren

Wenn man von irgendwoher einen öffentlichen Schlüssel bekommen hat, sollte man ihn in sein Schlüsselbund aufnehmen. Das wird mit

```
gpg --import [Datei]
```

erreicht. Wenn man den Dateinamen weglässt, wird von `stdin` gelesen.

3.4 Widerrufen

Es gibt verschiedene Gründe, einen alten Schlüssel zu widerrufen: Er könnte in fremde Hände geraten sein, die UID stimmt nicht mehr oder er ist einfach zu klein geworden. In all diesen Fällen ist der Befehl der Wahl

```
gpg --gen-revoke
```

Damit wird ein Schlüsselwiderruf-Zertifikat erzeugt. *Dafür braucht man den privaten Schlüssel*, denn sonst könnten solche Zertifikate auch von Fremden erzeugt werden. Das hat aber einen Nachteil: Ein Schlüssel, dessen Mantra ich nicht weiß, ist offensichtlich nutzlos. Aber weil ich das Mantra nicht weiß, kann ich ihn nicht widerrufen. Deshalb ist es geschickt, sich gleich bei der Erzeugung des Schlüssels ein Widerruf-Zertifikat zu erzeugen. Das sollte dann aber sicher verwahrt werden, am besten auf Diskette und auf Papier, damit es nicht in falsche Hände gerät.

3.5 Schlüsselbund verwalten

Der Schlüsselbund ist eine Datei, in der alle Schlüssel mit den dazugehörigen Informationen (bis auf die Ownertrust Werte, was das ist steht in 3.6 (Schlüssel signieren)) gespeichert werden. Mit

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

können alle Schlüssel des öffentlichen Schlüsselbundes angezeigt werden. Mit

```
gpg --list-sigs
```

werden zusätzlich noch die Signaturen angezeigt (siehe 3.6 (Schlüssel signieren)). Mit

```
gpg --fingerprint
```

werden die Schlüssel mit ihren "Fingerabdrücken" aufgelistet. Das sind (verhältnismäßig) kurze Zahlenfolgen, an denen sich der Schlüssel identifizieren läßt. Das kann praktisch sein, um sich über Telefon zu vergewissern, daß ein öffentlicher Schlüssel vom Gesprächspartner stammt. Fingerabdrücke im Abspann von Email oder Usenet Artikeln zu verschicken ist übrigens nicht sinnvoll.

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

listet die Schlüssel des privaten Schlüsselbundes auf. Unterschriften und Fingerabdrücke von privaten Schlüsseln haben keinen Informationswert.

Mit dem Befehl

```
gpg --delete-key UID bzw. gpg --delete-secret-key
```

kann man Schlüssel aus dem entsprechenden Schlüsselbund löschen.

Der letzte wichtige Befehl für den Umgang mit Schlüsseln lautet

```
gpg --edit-key UID
```

In dem dann folgenden Menü kann man unter anderem das Mantra und das Verfallsdatum ändern, Fingerabdrücke anzeigen lassen und Schlüssel signieren, womit wir beim nächsten Abschnitt wären.

3.6 Schlüssel signieren

Wie in der Einleitung erwähnt, ist die Echtheit eines öffentlichen Schlüssels die Achillesferse des Systems. Deshalb gibt es die Möglichkeit, Schlüssel zu unterschreiben. Damit bestätigt der Unterzeichnende, daß der in der User ID angegeben User tatsächlich der Besitzer des Schlüssels ist.

Nachdem man mit `gpg -edit-key UID` den zu unterzeichnenden Schlüssel ausgewählt hat, kann man ihn mit dem Kommando `sign` unterschreiben.

Unterschreiben Sie nur Schlüssel von deren Echtheit sie sich überzeugt haben. Das kann geschehen, in dem man entweder den Schlüssel persönlich bekommen hat (zum Beispiel auf einer Keysigning Party), oder man über Telefon den Fingerprint vergleicht. Man sollte keinen Schlüssel nur deshalb unterschreiben, weil man den anderen Unterschriften vertraut.

Anhand der Unterschriften und des "ownertrusts" ermittelt GnuPG die Gültigkeit des Schlüssels. Der Ownertrust ist ein Wert mit dem der Benutzer festlegt, in welchem Maße er dem Schlüsselinhaber zutraut, andere Schlüssel verläßlich zu unterzeichnen. Die möglichen Abstufungen sind "gar nicht", "weiß nicht", "teilweise" und "vollständig". Wenn der Benutzer also einem anderem nicht traut, kann er GnuPG über diesen Mechanismus anweisen, dessen Unterschrift zu ignorieren. Der Ownertrust wird nicht im Schlüsselbund gespeichert, sondern in einer separaten Datei.

4 Verschlüsseln und entschlüsseln

Falls man mehrere private Schlüssel hat, kann man mit der Option `-u UID` oder`-local-user UID` einen (oder mehrere) Schlüssel nach seiner UID auswählen. Diese Auswahl ersetzt den im Konfigurationsfile mit dem Befehl `default-key KeyID` einen Schlüssel standardmäßig ausgewählten Schlüssel.

Mit `-r UID` oder`-recipient UID` kann man den Empfänger in der Kommandozeile auswählen.

4.1 Verschlüsseln

Das Kommando zum Verschlüsseln lautet

```
gpg -e Empfänger [Datei]
```

oder

```
gpg --encrypt Empfänger [Datei]
```

Es ist sinnvoll, die Dateien auch zu signieren, genaueres siehe 5 (Signaturen).

4.2 Entschlüsseln

Das Kommando zum Entschlüsseln lautet

```
gpg [-d] [Datei]
```

oder

```
gpg [--decrypt] [Datei]
```

Auch hier gilt: Voreingestellt ist Ausgabe auf `stdout`, man kann aber mit der Option `-o [Datei]` in eine Datei ausgeben.

5 Signieren und Signaturen prüfen

Mit dem Befehl

```
gpg -s (oder --sign) [Datei]
```

unterschreibt man eine Datei mit seinem privaten Schlüssel. Sie wird dabei gleichzeitig komprimiert, ist dann also nicht mehr ohne weiteres lesbar. Mit

```
gpg --clearsign [Datei]
```

belässt man die Datei lesbar, mit

```
gpg -b (oder --detach-sign) [Datei]
```

erzeugt man eine Unterschrift in einer separaten Datei. Letzteres ist insbesondere zum signieren von Binärdateien wie Archiven zu empfehlen. Auch bei diesen Befehlen kann die Option `-armor` nützlich sein.

Üblicherweise wird sowohl signiert als auch verschlüsselt, der Befehl lautet dann vollständig

```
gpg [-u Sender] [-r Empfänger] [--armor] --sign --encrypt [Datei]
```

Die Optionen `-u` (`-local-user`) und `-r` (`-recipient`) funktionieren wie oben erläutert.

Wenn eine verschlüsselte Datei signiert ist, so wird beim Entschlüsseln die Signatur mitgeprüft. Die Signatur einer unverschlüsselten Datei prüft man mit

```
gpg [--verify] [Datei]
```

immer natürlich vorausgesetzt, daß man im Besitz des entsprechenden öffentlichen Schlüssels ist.

6 Informationsquellen

6.1 GnuPG

- Die *GnuPG Homepage* `<http://d˜ngm˜˜˜we˜r˜vygt˜˜g.roads-uae.com>`

- Die GnuPG Mailingliste. Inklusive Archiv und Beschreibung auf der *GnuPG Homepage* zu finden.
- Die beiliegende Dokumentation. Bisher (Stand 0.9.2) noch nicht sehr umfangreich, aber eben unverzichtbar. Nicht vergessen:

  ```
  gpg --help
  ```

  hilft!

6.2 PGP

PGP ist das ältere und (noch) weiter verbreitete Kryptographie Programm. Deshalb gibt es dazu auch viel mehr Informationen, sie sind aber teilweise so allgemein, daß sie auch für GnuPG nützlich sein können.

- Die *Internationale PGP Homepage* `<http://d˜ngm˜˜˜u6cvba˜.roads-uae.com>`
- Die *PGP DH vs. RSA FAQ* `<http://d˜ngm˜a4˜6krce˜h.roads-uae.com.uk/ss/pgpfaq.html>` gibt Informationen über die verwendeten Algorithmen.

6.3 Keyserver

- *Keyserver.net* `<http://d˜ngm˜e0g6kyw˜x˜0p˜˜4.roads-uae.comt>`
- `<http://d˜ngnpanx˜vbeenwr˜˜eawr.roads-uae.comp.net>`

6.4 Bücher

- B. Schneier, "Applied Cryptography, Second Edition", Wiley, 1996 Deutsche Ausgabe unter dem Titel "Angewandte Kryptographie", Addison-Wesley, 1996

7 Über dieses Dokument

7.1 Versionen

Version 0.1 war die erste öffentliche Version dieses Dokumentes.

Änderungen in Version 0.1.1

- Neuer Abschnitt "Grenzen der Sicherheit"
- Erklärung der Signatur verbessert
- kleinere Detailverbesserungen nach Hinweisen von Werner Koch (danke!)

Alle Änderungen sind in einem diff File aufgeführt, das man an gleicher Stelle wie *dieses Dokument* `<http://www.stud.uni-goettingen.de/~s070674/GnuPGMiniHowto/>` finden kann.

Anregungen, Kritik, Verbesserungen und Erweiterungen einfach an Michael Fischer v. Mollard (*fischer@math.uni-goettingen.de*) senden, damit dieses Dokument weiter verbessert werden kann.