GnuPG - eine Einführung

Über dieses Dokument

Version

- 20030203/tomm: erster Entwurf
- 20030204/tomm: zweiter Entwurf
- 20030205/tomm, cs: Bereinigung, Ergänzungen, Tippfehler entfernt
- 20030401/tomm, cs: Ergänzungen
- 20030625/cs, tomm: Ergänzungen

Autoren

- Tom Mattmann [tom DOT mattmann AT gmx DOT net]
- Chris Strässle [sacrelege AT swissonline DOT ch]

Zweck

Dieses Dokument, geschrieben anlässlich des LUG-Brugg-Treffens am 7. Februar 2003, beschreibt eine Möglichkeit, um den eMail-Verkehr zu schützen. Es geht insbesondere auf das Programm GnuPG ein.

Dieses Dokument ist kein Ersatz für die GnuPG-Dokumentation!

Inhalt

Weshalb ist Verschlüsselung wichtig?

Während verschlüsselte HTTP-Verbindungen (https) mittlerweile auf vielen Sites Einzug gehalten haben, wird das Thema eMail immer noch vernachlässigt. Viele Anwender benutzen entweder POP3- oder IMAP-Accounts, oder einen der nicht minder populären HTML-eMailer (zu selten HTTPS), bei denen sämtliche Daten unverschlüsselt über das Netz übertragen werden.

Weshalb ist die Sicherheit gefährdet? Hier einige Anhaltspunkte:

- Das Internet als Transportmedium ist wegen seines öffentlichen und dezentralen Charakters eigentlich nicht für die Übermittlung vertraulicher Nachrichten geeignet. Die Kontrolle der Netzabschnitte liegt in der Hand der jeweiligen Betreiber - und anderer Personen, die sich (vielleicht unbemerkt) Zugriff verschafft haben.
- Fehlkonfigurationen sind nicht auszuschliessen. Sie werden nicht nur von Crackern ausgenutzt...
- Jeder Administrator eines eMail-Servers kann ebenso mühelos auf sämtliche eMail-Konten zugreifen wie Personen, die physikalischen Zugriff auf wichtige Netzwerkadern (Backbones) haben.
- Vernetzte Arbeitsplätze sind alles andere als privat. Hier bieten verschlüsselte Dokumente zumindest ein minimales Mass an Privatsphäre.

Es ist also nicht übertrieben zu behaupten, dass eine **Nachricht auf Postkarte von weniger Personen eingesehen werden kann als eine unverschlüsselte eMail!**

Ausserdem: Verschiedene Institutionen, vor allem staatliche Organe, unternehmen grosse Anstrengungen, um die Benutzer (bzw. deren Daten) aufzuzeichnen und zu überwachen. Vor allem in den USA und auch in Europa wurden Gesetze eingeführt, welche die Provider dazu verpflichten, alle eMails (Verbindungsdaten und die gesamte eMail) eine Zeitlang auf ihrem Server zu halten. Einige Beispiele:

USA

Grundlage für die internationale Überwachung ist der digital copyright millenium act (DMCA). Seit dem 11. September 2001 wird dieser sehr rigoros angewandt. Staatliche Organisationen wie CIA und FBI erhalten auf einfache Weise Zugriff auf gespeicherte Daten.

- FBI: Carnivore (eMail-Sniffer)
- NSA: Betriebt weltweit Lauschposten ("Echelon"), um den gesamten Telekommunikationsverkehr mitzuschneiden. Kooperation mit England, Kanada, Australien, Neuseeland.

Europa

Auch die EU bzw. ihr angeschlossene Staaten haben die Gesetze zur Überwachung des Telekommunikationsverkehrs verschärft.

- Deutschland: [?]
- Frankreich: [?]

Schweiz

Seit Anfang 2002 ist das Bundesgesetz zur Überwachung des Post- und Fernmeldeverkehrs in Kraft. Es beinhaltet, dass Provider sämtliche Verbindungsdaten sowie den gesamten eMail-Verkehr während 6 Monaten zu Verfügung halten müssen, um diese auf Antrag dem UVEK (Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation) auszuhändigen.

Zusammenfassung

Der Telekommunikationsbereich wird von einer Vielzahl von Organisationen überwacht - aus den unterschiedlichsten Beweggründen. Leider **legen diese Organisationen entweder gar nicht oder nicht vollständig offen, was, wann, wo, wie lange... gespeichert wird, wer darauf Zugriff hat**.

Es ist anzunehmen, dass **umfangreiche Benutzerprofile** gepflegt werden, sowohl von staatlicher Seite wie auch von anderen Organisationen, die Teile des Internet kontrollieren. Dieses Dokument will diese Tatsache nicht Werten im Sinne einer "politischen" Betrachtung, sondern die Anwender, mich, Dich, uns, daran erinnern:

Es ist **Sache des Anwenders, um seine Privatsphäre besorgt zu sein!** Was im Web HTTPS, ist bei eMail GnuPG/PGP.

Was ist Verschlüsselung?

Verschlüsselung ist der Vorgang, eine Nachricht so zu codieren, dass sie auf öffentlichen Wegen transportiert, aber nur einem (vom Absender :-) bestimmten Empfänger gelesen werden kann. Prinzipiell gibt es zwei Verfahren, eine Nachricht zu verschlüsseln:

- symmetrische Verschlüsselung
- asymmetrische Verschlüsselung

Wichtige Begriffe in diesem Zusammenhang sind:

- Klartext: die Ursprüngliche, nicht verschlüsselte Nachricht
- Geheimtext, die verschlüsselte Nachricht
- Schlüssel: Parameter für den Algorithmus, der zum Verschlüsseln und/oder Entschlüsseln verwendet wird

Symmetrische Verschlüsselung

Das Hauptmerkmal bei der symmetrischen Verschlüsselung ist, dass Sender und Empfänger denselben Algorithmus anwenden, mit demselben Schlüssel. Ein einfaches Beispiel ist die ROT13-Verschlüsselung, der eine einfache Zeichenersetzung durchführt. Dabei wird jeden Buchstaben des Originaltextes mit demjenigen ersetzt, der sich 13 Stellen weiter hinten im Alphabet befindet.

Damit der Empfänger eine ROT13-verschlüsselte Nachricht entschlüsseln kann, muss er den Vorgang beim Verschlüsseln einfach **umkehren,** in diesem Fall also jeden Buchstaben mit demjenigen ersetzen, der sich 13 Stellen weiter vorne im Alphabet befindet.

Exkurs: Mono- und Polyalphabetische Verschlüsselung

Codebeispiel zu ROT13:

- Verschlüsselungsprogramm: perl -p -e 'tr/A-Za-z/N-ZA-Mn-za-m/'
- Entschlüsselungsprogramm: identisch, da Alphabet aus 26 Zeichen besteht!
- Geheimtext: EBG13 vfg tne avpug fvpure, nore rva thgrf Orvfcvry.
- Klartext: ?

Natürlich kann man das Verfahren weiter verfeinern, zum Beispiel nicht immer um 13 Positionen schieben, sondern um eine vorher definierte Sequenz (die also den Schlüssel darstellt).

Man kann diese Ersetzung von Zeichen auch vornehmen, indem man zwei Alphabete erstellt. Das erste ist das Klartextalphabet, das zweite das Geheimtextalphabet:

```
Klar:    a | b | c | d | e | ...
Geheim:  p | l | z | m | t | ...
```

Dieses Verfahren nennt man **monoalphabetische** Verschlüsselung. Auch kann man ohne weiteres mehrere Geheimtextalphabete hinzufügen, und dann während des Chiffrierens zwischen ihnen wechseln (**polyalphabetische** Verschlüsselung). Die Alphabete bzw. die Reihenfolge (Sequenz), in der sie verwendet werden, bilden hier den Schlüssel. Da diese Sequenz in der Regel kürzer ist als der Klartext, wird er blockweise angewandt (**Blockchiffre**).

Das Problem bei dieser Art der symmetrischen Verschlüsselung ist, dass eine **Häufigkeitsanalyse** (von Buchstaben) oft schon bei wenigen Hundert Zeichen Geheimtext auf den Klartext (und somit Schlüssel) schliessen lässt. Die Häufigkeitsanalyse nutzt die Tatsache, dass in jeder Sprache bestimmte Zeichen (im Deutschen etwa das "e" und "t") und Zeichenkombinationen ("ei", "er") wesentlich häufiger vorkommen als andere.

Die eben beschriebenen Algorithmen sind teilweise mehrere Tausend Jahre alt. Begünstigt durch die rasante technische Entwicklung, wurden in den 20er und 30 Jahren des vorigen Jahrhunderts erste elektromechanische Chiffriergeräte entwickelt. Die bekannteste ist wahrscheinlich die von der Deutschen Wehrmacht eingesetzte Enigma. Ihr Code ist ohne elektronische Hilfe nicht mehr zu brechen, was in den USA unmittelbar zur Entwicklung der ersten programmierbaren Computer führte (ENIAC, 1946).

Durch die Verbreitung der Computer wurden nochmals wesentlich leistungsfähigere Algorithmen ausgedacht, die wir hier aber nicht im Detail beschreiben. Ihnen ist gemeinsam, dass die Algorithmen so gut gewählt sind, dass ein direkter Angriff praktisch unmöglich wird, selbst wenn grössere Datenmengen mitgeschnitten werden können. Die Sicherheit hängt primär von der Schlüssellänge ab, oder anders gesagt von der Anzahl möglichen Kombinationen, die ein Angreifer "ausprobieren" müsste, um den Geheimtext zu entschlüsseln. Deshalb ist

die symmetrische Verschlüsselung mit einem guten Algorithmus und entsprechend gewählter Schlüssellänge sehr sicher.

Exkurs: Schlüsselknacker im Web

Um Dir ein Bild vom Aufwand zu machen, der nötig ist, um einen symmetrischen Schlüssel mit bekannter Länge zu finden, besuche Distributed.Net (www.distributed.net). Beispielsweise benötigte das Brechen eines 64 bit langen RC5-Schlüssels über 3 Jahre, und insgesamt über 300'000 Freiwillige, die Rechenkapazität bereitstellten.

Bekannte symmetrische Verschlüsselungsverfahren sind:

- ROT13
- XOR
- DES und 3DES
- RC2, RC4, RC5
- Blowfish
- AES
- IDEA

Zusammenfassung

Vorteile der symmetrischen Verschlüsselung:

- sie ist relativ einfach zu implementieren und deshalb schnell
- moderne Algorithmen sind sehr sicher

Nachteile der symmetrischen Verschlüsselung:

- der Schlüsselaustausch ist problematisch, da er die einzige Sicherheit darstellt
- die Länge des Schlüssels hat unmittelbar Einfluss auf die Sicherheit
- keine Authentifizierung möglich, nur Verschlüsselung

Asymmetrische Verschlüsselung

Bei der Asymmetrischen Verschlüsselung werden für Ver- und Entschlüsselung **unterschiedliche Schlüssel** verwendet. Mathematiker haben lange nach entsprechenden Algorithmen gesucht und schliesslich auch gefunden. Diese Algorithmen arbeiten mit sehr grossen Primzahlen bzw. deren Produkten (Multiplikation), da es sehr schwierig ist, diese in Primfaktoren zu zerlegen, was aber nach heutigem Stand des Wissens für das Knacken des Schlüssels erforderlich ist.

Anders gesagt: Es ist einfacher, eine grosse Primzahl zu finden als herauszufinden, ob eine gegebene grosse Zahl eine Primzahl ist.

Wesentlicher Bestandteil der asymmetrischen Verschüsselung ist ein **Schlüsselpaar**. Er besteht aus einem **privaten** und einem **öffentlichen** Schlüssel:

- Der **private Schlüssel** wird dazu verwendet, mit dem öffentlichen Schlüssel verschlüsselte Nachrichten zu **entschlüsseln.** Er ist das einzige Geheimnis und muss deshalb besonders geschützt werden. Er darf **NIEMALS an Dritte weitergegeben** werden.
- Der **öffentliche Schlüssel** wird dazu verwendet, um Nachrichten

zu **verschlüsseln.** Natürlich verwenden wir nicht unseren eigenen öffentlichen Schlüssel (da nur wir selbst so verschlüsselte Nachrichten entschlüsseln können) - sondern den öffentlichen Schlüssel des **Kommunikationspartners.** Öffentliche Schlüssel sind kein Geheimnis und können bedenkenlos an Dritte weitergegeben werden.

Zusammenfassung

Bekannte Asymmetrische Verschlüsselungsverfahren:

- RSA
- DSA

Vorteile der asymmetrischen Verschlüsselung:

- keine Probleme mit Schlüsselaustausch
- Algorithmus ist so gewählt, dass er nicht sicherheitsrelevant ist

Nachteile der asymmetrischen Verschlüsselung:

- sie ist aufwendig und deshalb langsam
- die gesamte Sicherheit hängt vom privaten Schlüssel ab
- Schlüsselverwaltung ist aufwendig

Kombinierte Verfahren

Wie wir gesehen haben, ist die symmetrische Verschlüsselung relativ schnell, aber deren Sicherheit vor allem von der Geheimhaltung des Schlüssels abhängig. Asymmetrische Verfahren sind eher langsam, und die Schlüsselverwaltung ist aufwendig.

GnuPG/PGP verwenden deshalb ein kombiniertes Verfahren:

- Klartext wird mit einem zufällig gewählten symmetrischen Schlüssel verschlüsselt
- Geheimtext und symmetrischer Schlüssel werden zusammengefasst und mit dem öffentlichen Schlüssel des Empfängers verschlüsselt

Der Empfänger erhält also eine doppelt verschlüsselte Nachricht. Mit seinem privaten Schlüssel kann er den symmetrischen Schlüssel auslesen, und mit diesem dann den Geheimtext wieder in Klartext umwandeln.

Was sind digitale Unterschriften?

Durch die Verschlüsselung einer Nachricht ist zwar erstmal deren Inhalt gegen neugierige Augen geschützt, aber die wahre Identität des Verfassers und die Unversehrtheit (Integrität) der Nachricht sind ungewiss. Die Identität ist deshalb nicht gewiss, weil die Nachricht mit

dem öffentlichen Schlüssel des Empfängers verschlüsselt wird, und der ist jedermann zugänglich. eMail-Absenderadressen (und überhaupt alle Informationen im Mail-Header) lassen sich zu leicht fälschen.

Aus diesen Gründen kann eine Nachricht mit einer digitalen Unterschrift versehen werden. Diese Unterschrift ist eine Prüfsumme, die zur Nachricht passt. Das funktioniert folgendermassen:

1. Der Absender generiert ein Hash für die Nachricht.
2. Der Absender verschlüsselt den Hash mit seinem privaten Schlüssel. Der Empfänger kann ihn also mit dem öffentlichen Schlüssel des Absenders entschlüsseln. Damit weiss der Empfänger, woher die Nachricht stammt, und dass sie "unterwegs" nicht verändert wurde.
3. Dann wir das "kombinierte Verfahren" (siehe oben) angewandt, um die Nachricht zu verschlüsseln.

Exkurs: Hash-Funktionen

Eine hash-Funktion ist eine typische Einweg-Funktion, die eine Prüfsumme für einen Datenblock (Text, Datei, ...) zurückliefert. Die Prüfsumme ist üblicherweise wenige Bytes lang, unabhängig von der Grösse des Datenblocks. Hier als Beispiel eine einfache additive Hash-Funktion aus K&R ANSI C:

```
unsigned long
    k_r(unsigned char *str)
    {
      unsigned long hash = 0;
      int c;
      while (c = *str++)
        hash += c;
      return hash;
    }
```

Die Funktion nimmt einen Zeiger auf eine Zeichenkette (unsigned char * str) entgegen, die mit einem 0-Byte endet (while (*str++)). In der Schleife wird einfach eine Summe gebildet. Die Summe selbst ist je nach Plattform 32 oder 64 Bit gross (unsigned long hash), und darf/kann auch überlaufen.
Hier wird auch der "Einweg"-Charakter der Hash-Funktion deutlich: Es ist unmöglich, vom Hash auf den übergebenen String zu schliessen (sofern dieser aus mindestens zwei Zeichen besteht...).

Wie benutze ich GnuPG?

Installation von GnuPG

Auf der *nix-Plattform wird in der Regel aus den Quellen kompiliert. Quellcode und Installationshinweise finden sich unter:

```
http://www.gnupg.org/
```

Grafische Helfer

Moderne eMail-Programme (MUAs) bieten in der Regel eine vollständig grafische Oberfläche, in der auch GnuPG eingebettet ist. Auf die Bedienung der MUAs wird hier nicht weiter eingegangen. RTFM! ;-)

Auch wenn Du einen grafischen Helfer hast, **lies unbedingt den Abschnitt "Vorsichtsmassnahmen"**!

Diese MUAs mit grafischer Einbettung von GnuPG sind uns bekannt:

- Sylpheed (mit gpgme gelinkt)
- Mozilla Mailer (mit EnigMail plugin)

Bei der Verwendung eines Web-Mailers oder beim Scripten ist es nötig, GnuPG direkt zu steuern. Die folgenden Abschnitte beschreiben die wichtigsten Optionen. Mehr Informationen liefern z.B.:

- man gpg
- gpg --help

Vorbereitung

Erzeugen eines Schlüsselpaars

Als erstes benötigen wir ein Schlüsselpaar, bestehend aus einem privaten und einem öffentlichen Schlüssel.

- Den privaten Schlüssel verwenden nur wir selbst, um verschlüsselte Mails zu entschlüsseln, die von anderen mit unserem öffentlichen Schlüssel verschlüsselt wurden.
- Den öffentlichen Schlüssel geben wir an unsere Kommunikationspartner weiter.

Jetzt aber zur Sache, wir starten GnuPG!

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

Ich verwende die momentan neuste Version 1.2.1, die Ausgabe kann bei anderen Versionen leicht abweichen.

```
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?
```

Erstmal werden werden wir gefragt, welchen Algorithmus wir verwenden möchten. Da RSA wegen Patenten ausfällt, nehmen wir DSA / ElGamal (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)
```

Jetzt folgt die Frage nach der Schlüssellänge. Ich verwende gerne 1280 Bit, ist aber eher eine Sache des persönlichen Geschmacks als der Sicherheit. [keine Empfehlung]

```
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)
```

Das Schlüsselpaar wird mit einem Gültigkeitsdatum versehen. Nach Ablauf dieses Datums ist der öffentliche Schlüssel nicht mehr gültig und wir müssen ein neues Schlüsselpaar generieren.

```
  You need a User-ID to identify your key;
  the software constructs the user id
  from Real Name, Comment and Email Address [...]
```

Jetzt geben wir unseren Namen, eMail-Adresse und einen Kommentar an. Diese Angaben werden im öffentlichen Schlüssel hinterlegt.

```
  You need a Passphrase to protect your secret key.
  Enter passphrase:
```

Das Mantra (passphrase) schützt den privaten Schlüssel und wird immer dann benötigt, wenn wir eine Nachricht entschlüsseln. Einerseits sollte das Mantra ausreichend kompliziert sein, damit der Schlüssel gut geschützt ist, aber dennoch so, dass Du sie Dir merken kannst.

```
  gpg: /home/tomm/.gnupg/trustdb.gpg: trustdb created
  public and secret key created and signed.
  key marked as ultimately trusted.
  pub  1024D/D62E1779 2003-02-02 Tom Mattmann ([LUG TEST]) <tomm@localhost>
       Key fingerprint = 1188 0158 53AC E1F6 8838  2ACF 023B A25E D62E 1779
  sub  1280g/430AF050 2003-02-02 [expires: 2004-02-02]
```

Wir haben es fast geschafft, der Schlüssel ist erstellt!

Erzeugen des Widerruf-Zertifikats

Mit dem Widerruf-Zertifikat können wir unsere Kommunikationspartner darüber informieren, dass unser öffentlicher Schlüssel nicht mehr benutzt werden soll. Der Schlüssel wird dazu als widerrufen markiert, bleibt aber im Schlüsselbund. Dadurch kann er weiterhin benutzt werden, um Signaturen zu überprüfen.

Sollte unser privater Schlüssel einmal in falsche Hände gelangen, verlorengehen, oder wir unser Postfach züglen müssen, werden wir das Widerrufs-Zertifikat einsetzen. Deshalb erzeugen wir gleich mal ein solches:

```
gpg --armor --gen-revoke tom

sec  1024D/D62E1779 2003-02-02 Tom Mattmann ([LUG TEST]) <tomm@localhost>
Create a revocation certificate for this key?
```

Wenn der Schlüssel passt, sagen wir ja (y).

```
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision?
```

Hier werden wir nach dem Grund gefragt, weshalb wir das Widerrufs-Zertifikat (revocation certificate) erstellen. Mir gefällt "Key is no longer

used" (Auswahl 3) wesentlich besser als der Vorschlag "Key has been compromised" (Auswahl 1). Wähle!

```
Enter an optional description; end it with an empty line:
```

Wir können dem Zertifikat auch etwas Text mitgeben, vielleicht das Tagesdatum und ein Aufruf, sich unseren neuen öffentlichen Schlüssel zu besorgen?

Abschliessend müssen wir wiedermal das Mantra eingeben, da wir mit dem privaten Schlüssel arbeiten, und erhalten das Widerrufs-Zertifikat.

Vorsichtsmassnahmen

Jetzt speichern wir den öffentlichen Schlüssel, den privaten Schlüssel und das Widerruf-Zertifikat auf einem langlebigen Datenträger, den wir unter Verschluss halten. Zusätzlich drucken wir alles aus (Papier hält wesentlich länger als Disketten und CDs...) und halten diese Papiere ebenfalls unter Verschluss.

Dann **löschen wir das Widerruf-Zertifikat,** da sonst irgendein uns übelwollender Kollege dieses einfach verteilen könnte!

Den Datenträger und/oder die Ausdrucke benutzen wir im Notfall (z.B. wenn unsere Festplatte streikt und wir den privaten Schlüssel nach der Neu-Installation wieder benötigen), aber wir **geben diese niemals an Dritte weiter.**

Verwalten des Schlüsselbunds

Damit wir mit GnuPG arbeiten können, müssen wir uns die öffentlichen Schlüssel der Kommunikationspartner besorgen und sie zu unserem elektronischen Schlüsselbund hinzufügen. Doch schauen wir jetzt zuerst mal unseren Schlüsselbund an.

Anzeigen des Schlüsselbunds

- gpg --list-keys <name> [<name> ...]
- gpg --list-key <name> [<name> ...]
- gpg --list-public-keys <name> [<name> ...]

Für <name> kannst Du wahlweise einen Teil des Namens, der eMail-Adresse oder der Schlüssel-ID einsetzen, probier's eifach aus!

Eine nützliche Option ist `--armor`: sie veranlasst gpg, die Ausgabe "lesbar" zu machen, anstatt binäre Zeichen auf die Standardausgabe zu schieben. Die meisten Befehle mit unmittelbarer Ausgabe der Ergebnisse verstehen diese Option, allerdings muss sie als erstes Argument übergeben werden.

Arbeiten mit einem Schlüsselserver

Wir verwenden gewöhnlich den Server `pgp.mit.edu`, da er seinen Dienst

kostenlos anbietet und gut erreichbar ist. Du kannst ihn auch via http erreichen und über eine einfache Maske nach Schlüsseln suchen und diese im `--armor`-Format anzeigen. Dadurch kannst Du auch mit copy/paste Schlüssel herunterladen und dann in den Schlüsselbund einfügen.

Um nicht ständig die Option `--keyserver` mit dem entsprechenden Server angegeben zu müssen (also --keyserver pgp.mit.edu), solltest Du den bevorzugten Keyserver in der GnuPG-Konfigurationsdatei hinterlegen.

- gpg --keyserver <my.key.server> --send-keys <name> [<name> ...]
  Mit diesem Befehl werden die angegeben Schlüssel an den Schlüsselserver übermittelt.
- gpg --keyserver <my.key.server> --search-keys <name> [<name> ...]
  Dieser Befehl führt eine Suche nach einem Schlüssel durch. Die Suche wird case-insensitive durchgeführt und die passenden Schlüssel werden angezeigt, damit man sie gleich in den eigenen Schlüsselbund aufnehmen kann.

Mit fremden Schlüsseln arbeiten

- Hinzufügen eines Schlüssels zum Schlüsselbund aus einer Datei:
  gpg --import <dateiname>
- Hinzufügen eines Schlüssels zum Schlüsselbund von einem Schlüsselserver:
  gpg --keyserver <my.key.server> --recv-keys <key-ID>

Sobald wir einen Schlüssel zum Schlüsselbund hinzufügen, wird ihm ausserdem eine **Vertrauensstufe** zugewiesen. Sie beschreibt, wie gut wir den Empfänger bzw. seinen Schlüssel kennen. Wenn hohe Sicherheit nötig ist, **müssen** wir z.B. persönlich mit dem Herausgeber des Schlüssels nachprüfen, ob wir wirklich seinen Schlüssel erhalten haben, oder ob sich jemand als ihn ausgibt und uns täuschen will.

- Aendern der Vertrauenswürdigkeit eines Schlüssels:
  gpg --edit-key <name>

Nach Eingabe des `--edit-key`-Befehls befinden wir uns im interaktiven Teil von GnuPG, und der entsprechende Schlüssel wird angezeigt. Wenn wir diesen Schlüssel nun mit unserem Schlüssel unterschreiben wollen, müssen wir folgendes eingeben:

```
Command> sign
```

Jetzt werden Schlüssel-ID, Fingerprint und ein paar andere Informationen ausgegeben und wir werden gefragt, für wie Vertrauenswürdig wir den Schlüssel halten bzw. wie wir den Schlüssel mit unserer Unterschrift einstufen wollen:

```
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.
```

- Löschen eines Schlüssels:
  gpg --delete-key <name@host>

Ver- und Entschüsseln von Nachrichten

Verschlüsseln einer Nachricht

Um eine Nachricht zu verschlüsseln, brauchen wir erstmal den öffentlichen Schlüssel des Empfängers. Diesen fügen wir unserem Schlüsselbund hinzu (siehe Verwalten des Schlüsselbunds).

- Verschlüsseln einer Nachricht:
  gpg --armor --encrypt
  bzw.:
  gpg --armor --encrypt-files <datei1 [datei2 ...]>

Entschlüsseln einer Nachricht

- Entschlüsseln einer Nachricht:
  gpg --decrypt
  bzw.:
  gpg --decrypt-files <datei1 [datei2 ...]>

Weitere Themen

Wir haben hier nur die nötigsten Handgriffe mit GnuPG beschrieben, und einige Möglichkeiten einfach verschwiegen. Um Deine Neugier zu wecken, listen wir sie hier einfach mal auf:

- Wenn wir einen öffentlichen Schlüssel geprüft haben, können wir ihm nicht nur einen anderen Vertrauensstatus zuweisen, sondern ihn auch signieren. Diesen signierten Schlüssel können wir wiederum weitergeben, und die neuen Empfänger sehen, dass Du ihn geprüft hast.
- Es gibt öffentliche Schlüssel-Server (key server), auf welchen man seinen öffentlichen Schlüssel hinterlegen kann, bzw. andere öffentliche Schlüssel beziehen kann. Einige Schlüssel-Server bieten auch vertrauenswürdige Schlüssel - d.h. die Administratoren haben die Echtheit des Schlüssel-Inhabers geprüft.

Weiterführende Literatur

- Gnu Privacy Guard homepage: www.gnupg.org, und man gpg

Quellcode und die Handbücher

- Geheime Botschaften, Simon Singh, ISBN 3-446-19873-3
  Geschichte der Kryptographie und Kryptanalyse
- Applied Cryptography, Bruce Schneider
  Die ultimative Algorithmensammlung für die Kryptographie
- Vom Ende der Anonymität, Christiane Schulzki-Haddouti, ISBN 3-88229-185-0
  Wer jetzt immer noch nicht paranoid ist, wird es mit diesem Buch!