# PGP - das elektronische Kuvert mit Siegel

Rainer Meisel

Verschlüsseln von Informationen und Schutz vor Informationsveränderung, das sind die Hauptaufgaben von PGP, die es auch mit einer für den Hausgebrauch ungewöhnlich hohen Qualität und Zuverlässigkeit erfüllt; es wird de facto militärische Sicherheit erreicht, vorausgesetzt man hält ein paar wichtige Spielregeln ein. PGP ist Freeware und für viele Plattformen verfügbar. Es wurde von Philip Zimmermann vom Massachusetts Institute of Technology entwickelt. PGP wendet das Prinzip von öffentlichen Schlüsseln, die sogenannte „Public Key"-Kryptographie an, wodurch es für die Verwendung in Computernetzwerken wie maßgeschneidert ist.

Benutzt wird PGP grundsätzlich als DOS Kommando, wobei alle wichtigen Parameter mit übergeben werden. Normalerweise wird es aber in die entsprechende Software, mit der es zusammen verwendet werden soll, eingebunden. Eine Möglichkeit ist z.B. PGP in einen Maileditor zu integrieren. Das Ver- und Entschlüsseln von Nachrichten passiert dann vollautomatisch, ohne daß der User davon Notiz nehmen muß, außer wenn ein Problem auftritt; Mail könnte zum Beispiel manipuliert worden sein. Dann gibt es einen entsprechenden Warnhinweis von PGP.

Um eine Nachricht für einen bestimmten Empfänger zu verschlüsseln, benötigt man dessen Public Key. Der Public Key wird zusammen mit dem Secret Key mit PGP generiert und kann dann über alle möglichen Informationskanäle verbreitet werden. Im Public Key sind, im Gegensatz zum Secret Key, nur so viele Informationen vorhanden, daß man ihn zum Verschlüsseln, aber keinesfalls zum Entschlüsseln von Nachrichten verwenden kann. Das ist das ganze Geheimnis des Public Key Verfahrens, nicht mehr und nicht weniger. Zum Entschlüsseln braucht man den zum Public Key passenden Secret Key, den nur der Empfänger der Nachricht hat. Hoffentlich!

Der Secret Key ist sozusagen das Heiligtum des Ganzen. Er ist zwar mit einer Wortsequenz, der Paßphrase geschützt, sollte aber so sicher wie möglich gegen unbefugten Zugriff geschützt sein. Wenn einmal der Secret Key gestohlen wird, ist nichts mehr zu machen, alle Nachrichten an den Eigner dieses Schlüssels können damit entschlüsselt werden.

PGP verwaltet alle Schlüsseln in sogenannten Keyrings, den Schlüsselbünden. Im Normalfall gibt es zwei davon, einen Pubring für die Public Keys und einen Secring für die Secret Keys. Man erspart sich dadurch viele Einzelfiles und hat einen sehr guten Überblick nebst vielen hilfreichen Funktionen zum Verwalten der Schlüssel. Identifiziert wird ein Key entweder mit dem bei der Erzeugung eingegebenen Namen oder mittels einer KeyID-Nummer.

Nicht nur zum Verschlüsseln von Informationen kann man PGP verwenden, sondern auch zum Schutz vor Fälschungen oder Veränderungen durch Dritte. Das geschieht, indem man mit seinem Secret Key die entsprechenden Daten signiert, also eine digitale Unterschrift dazuspeichert. Wie bei einer Prüfsumme ergibt jede Änderung an Inhalt oder Signatur selbst einen Fehler beim Check der Signatur. Überprüft werden kann die Signatur übrigens mit dem zum Secret Key passenden Public Key. Diese digitale Unterschrift kann direkt zu den Daten dazugespeichert oder als eigenes File mitgeliefert werden. Wenn ein Programmautor sein Archiv vor der Verbreitung in Mailboxen mit PGP signiert und der Empfänger seinen Public Key hat, dann gibt es praktisch keine Möglichkeit, unerkannt Teile des Archivs zu verändern oder zu entfernen. Auch ein unterwegs eingefangener Virus würde sofort erkannt werden.

Wie man sieht, hängt das ganze System von der Verbreitung der Public Keys ab. Es hat keinen Sinn, ein Programmpaket oder einen Text mit PGP zu signieren, wenn niemand den Public Key des Autors hat. Dazu wurden am Internet die sogenannten PGP Keyserver eingerichtet, die man natürlich auch über Gates von anderen Netzwerken erreicht. Dorthin kann man einerseits seinen eigenen Public Key schicken und andererseits fremde Public Keys holen. Die Adressen dieser Server und deren Benutzung sind in den Dokumentationen im PGP Paket enthalten. Die Betreiber der Keyserver garantieren nicht für die Authentität der darin gespeicherten Keys, womit wir beim nächsten Thema wären.

Das Public Key System hat einen Haken, die Zuverlässigkeit der Public Keys betreffend. Dieser Haken wurde zwar excellent vom PGP-Autor berücksichtigt, verursacht aber einen gewissen Aufwand und macht die Sache komplizierter als notwendig. Lassen Sie mich diesen Haken in einem kleinen Beispiel verdeutlichen: User A und User B hängen beide als Point an einer Mailbox und wollen sich Daten übertragen, die nicht für Dritte bestimmt sind. Also senden sie sich gegenseitig ihre Public Keys über die Mailbox. Der böse Sysop fängt aber jetzt den Key von User A ab und schickt statt dessen seinen eigenen, unter dem Namen von User A generierten Public Key an B. User B bekommt diesen Fake-Schlüssel und schreibt ganz fleißig geheime Nachrichten an User A, verschlüsselt mit dem falschen Schlüssel. Der Sysop der Mailbox kann nun diese Nachrichten entschlüsseln und in Ruhe lesen, nachdem er ja den passenden Secret Key hat. Anschließend verschlüsselt er die Nachrichten mit dem echten Key von User A und schickt ihm die Nachrichten weiter. User A wird nichts Auffälliges feststellen und User B schon gar nicht, aber alle Nachrichten werden vom Sysop gelesen. Ich glaube, jeder kann sich vorstellen, wie leicht es ist, falsche Schlüssel, überhaupt auf solchen Keyservern, zu verbreiten.

Es gibt nun mehrere Maßnahmen, um dieses Problem in Griff zu bekommen. Die einfachste ist die Möglichkeit, einen Fingerprint eines Keys ausgeben zu lassen und z.B. telefonisch mit dem Eigner zu vergleichen. So ein Fingerprint besteht aus 16 zweistelligen Hexadezimalzahlen, die für jeden Schlüssel einzigartig sind. Eine andere Maßnahme stellt die Möglichkeit des Signierens von Public Keys dar. Jeder PGP-User kann die Public Keys anderer User signieren und bestätigt praktisch mit seiner Unterschrift die Richtigkeit und Authentität des Keys. Das Ganze geht auch mehrfach und in mehreren Levels. Wenn man also einen neuen Key in seinen Ring aufnimmt und dieser von jemanden signiert ist, dem man voll vertraut, dann kann man eben mit dessen Public Key den neuen Key checken.

In der Praxis geht das dann so vor sich, daß PGP bei der Aufnahme eines neuen Keys in den Ring ein paar Fragen stellt, je nach dem, von wem oder ob überhaupt der Key signiert ist. Zuerst erkundigt sich PGP grundsätzlich ob man dem Eigner des neuen Keys vetraut, in vier Stufen gegliedert. Das bestimmt dann wie PGP auf weitere neue Keys reagiert, die von diesem User unterschrieben sind. Dann will PGP noch wissen, falls der Schlüssel z.B. überhaupt nicht unterschrieben war, wie sicher man ist, dass das auch der Key vom Eigner ist. Je nach dieser Angabe gibt es dann Warnungen auf Zuverlässgkeit beim Verschlüsseln usw.

Den Aufwand, den man mit den Vertrauensangaben von Schlüsseln betreibt, kann man selbst bestimmen. Einerseits kann man jeden Schlüssel gutgläubig in seinen Ring aufnehmen oder die Sicherheit bis zum Exzess treiben, indem man jeden Key Eigner anruft und den Fingerprint vergleicht. Aber eines ist klar, ein Programmarchiv mittels eines mitgelieferten Public Keys auf Sicherheit zu prüfen ist sinnlos. Das wird jeder verstehen, der den Sinn von PGP versteht.

Es ist sicher nicht sinnvoll, Nachrichten à la „Der Witz war gut!" mit einer PGP-Signatur zu versehen, aber PGP schafft Möglichkeiten, die für viele Leute Arbeit, Zeit und Geld sparen. Ein Beispiel ist die Versendung von Shareware-Keyfiles über Netzwerke. Wenn diese mit PGP verschlüsselt sind, kann man sie nicht nur als Textfile verschicken, sondern sind auch für alle, die sie abfangen, nicht zu gebrauchen. Auch wenn Leute sich darüber ärgern, daß unter ihrem Namen ungute Mails in Netzwerken verbreitet werden, sollten Sie auf PGP zurückgreifen.

## Wo finde ich PGP

Grundsätzlich ist PGP auf einigen Fido Mailboxen zu finden.

## Kommandozeilenbeispiele

generieren eines Schlüsselpaars: `pgp -kg`

Key in den Bund aufnehmen: `pgp -ka Key.txt`

Durchsucht die Datei `Key.txt` nach einem oder mehreren Keys und nimmt diese in den Pubring auf.

Verschlüsseln: `pgp -ea Msg.txt "Franz Fiala"`

Verschlüsselt die Datei `Msg.txt` mit dem Public Key von Franz Fiala.

## Umgang mit PGP-Servern

Der Umgang ist recht einfach. Man schickt eine E-Mail an den Server. Im Subject steht das Kommando und eventuell im Mailbody der betreffende PGP Public Key.

## Die Kommandos:

| Command | Message body contains |
|---|---|
| `ADD` | Your PGP public key (key to add is body of msg) (-ka). Mit diesem Befehl hängt man einen Key in den Ring des Servers. |
| `INDEX` | List all PGP keys the server knows about (-kv). Liefert eine Liste der verfügbaren Public Keys. |
| `VERBOSE INDEX` | List all PGP keys, verbose format (-kvv). Auch eine Liste der Keys, aber mit mehr Infos. |
| `GET` | Get the whole public key ring (-kxa *). Damit bekommt man alle Schlüssel (Vorsicht! Riesendatei) |
| `GET <userid>` | Get just that one key (-kxa <userid>). Schickt einen einzelnen Public Key. |
| `MGET <userid>` | Get all keys which match <userid>. Gleich wie GET aber mit Patternmatching (Wildcards). |
| `LAST <n>` | Get all keys uploaded during last <n> days. Wird verwendet, um die neuen Keys der letzten n Tage zu holen. |

Also wenn man z.B. seinen Public Key an den Server schicken will, dann schreibt man den Befehl 'ADD' ins Subject, der Key ist im Msgbody.

## Liste der Server

pgp-public-keys@uit.no
pgp-public-keys@informatik.uni-hamburg.de
pgp-public-keys@kub.nl
pgp-public-keys@pgp.ox.ac.uk
pgp-public-keys@pgp.pipex.net
pgp-public-keys@dsi.unimi.it
pgp-public-keys@goliat.upc.es
pgp-public-keys@srce.hr
pgp-public-keys@kiae.su
pgp-public-keys@ext221.sra.co.jp
pgp-public-keys@sw.oz.au
pgp-public-keys@pgp.mit.edu (*)
public-key-server@martigny.ai.mit.edu (*)
pgp-public-keys@pgp.iastate.edu (*)
pgp-public-keys@burn.ucsd.edu (*)
pgp-public-keys@pgp.dhp.com (*)
pgp-public-keys@jpunix.com (*)
pgp-public-keys@gondolin.org (*)

(*) Key servers in the USA only accept keys labelled "Version: 2.4" or later.

## Sites accessible via WWW:

http://www.service.uit.no/pgp/servruit.eng.html
http://www.cl.cam.ac.uk/PGP/pks-toplev.html
http://www.nic.surfnet.nl/pgp/pks-toplev.html
http://goliat.upc.es/~alvar/pks/pks-toplev.html
http://martigny.ai.mit.edu/~bal/pks-toplev.html

## Key server keyrings accessible via FTP:

ftp://ftp.uit.no/pub/crypto/pgp/keys/pubring.pgp
ftp://ftp.informatik.uni-hamburg.de/pub/virus/crypt/pgp/pubkring.pgp
ftp://ftp.dsi.unimi.it/pub/security/crypt/PGP/public-keys.pgp
ftp://ftp.ox.ac.uk/pub/crypto/pgp/keys/pubring.pgp
ftp://ftp.sunet.se/pub/security/tools/crypt/pgp/keys/pubring.pgp
ftp://ftp.funet.fi/pub/crypt/cryptography/pgp/keys/pubring.pgp
ftp://pgp.mit.edu/pub/keys/public-keys.pgp
ftp://pgp.iastate.edu/pub/pgp/public-keys.pgp
ftp://burn.ucsd.edu/Crypto/public-keys.pgp
ftp://jpunix.com/pub/PGP/

## Schnellste Zusammenstellung über PGP

ftp://ftp.tuwien.ac.at/pub/auxc/privacy/crypto/pgp

## Weitere Info auch auf

http://www.pgp.net□

---

The Evolution of a Programmer

High School/Jr.High

```
10 PRINT "HELLO WORLD"
20 END
```

First year in College

```
program Hello(input, output)
begin
  writeln('Hello World')
end.
```

Senior year in College

```
(defun hello
(print
(cons 'Hello (list 'World))))
```

New professional

```
#include <stdio.h>

void main(void)
{
  char *message[ ] = {"Hello ", "World"};
  int i;

  for(i = 0; i < 2; ++i)
    printf("%s", message[i]);
  printf("\/n");
}
```