

Einführungskurs zu PGP, Version vom 18.02.1996

Vorwort

Pretty Good Privacy (abgek.: PGP) ist ein Verschlüsselungsprogramm für jedermann. Es ist Freeware und deshalb für Privatanwender kostenlos in vielen Mailboxen und von diversen Shareware-CDs zu beziehen. PGP wird mit einer ausführlichen englischen Anleitung geliefert, auch eine deutsche Doku und diverse weitere Texte dazu gibt es bereits. Was alle diese Anleitungen nicht tun, ist den Anfänger, der sich ohne die geringsten Kenntnisse dieses Programmes damit befassen möchte, schrittweise an die Bedienung von PGP heranzuführen. Dies soll dieser Kurs tun, dafür fehlt diesem Kurs einiges an Informationen über die Entstehung von PGP, die juristischen Probleme, die der Programmautor durch den Export seines Programmes hat, und einige andere PGP-Interna, die den PGP-Anfänger zunächst auch gar nicht interessieren, die er jedoch in den anderen Texten später nachlesen kann.

Dieser Kurs ist so aufgebaut, daß er von Anfang bis Ende durchgearbeitet werden sollte, weil jedes Kapitel auf den erworbenen Kenntnissen der vorhergehenden Kapitel aufbaut. Dazu ist manchmal ein wenig Vertrauen (zu mir als Verfasser dieses Kurses) und Geduld nötig, aber mir wurde von verschiedenen Beta-Testern dieses Kurses bestätigt, daß es sich lohnt, konsequent von Anfang bis Ende mitzuarbeiten. Einige von ihnen hatten vorher bereits versucht PGP anhand der Dokumentation zu verstehen, sind aber daran gescheitert, daß die schrittweise Einführung dort nicht stattfand. Mit diesem Kurs sollte es eigentlich jedem möglich sein, PGP zu verstehen und anzuwenden.

Die sinnvollste Art mit diesem Kurs zu arbeiten, ist den Text auszudrucken, um den Text immer vorliegen zu haben, egal was man gerade am Bildschirm macht, denn praktische Übung am PC ist ein Bestandteil dieses Kurses. Wenn man Utilities wie 4print oder Booklet zum Ausdrucken benutzt, kostet es nicht einmal viel Papier, damit lassen sich 4 Seiten auf eine Din-A4 Seite ausdrucken, und kleine Hefte falten, die man immer griffbereit haben kann.

Dieser Kurs richtet sich in erster Linie an Benutzer IBM-kompatibler PCs unter MS-DOS (oder kompatiblen Systemen), da ich zum einen selbst auf einem solchen System zu Hause bin, und zum anderen diese Maschinen die größte Verbreitung finden. Aber auch Benutzer anderer Systeme können an diesem Kurs teilnehmen, sie müssen lediglich die wenigen MS-DOS-spezifischen Hinweise (eigentlich sind es nur die Umgebungsvariablen) auf ihr System umsetzen.

Niemand ist perfekt, ich schon gar nicht, und dieser Text kann es auch nicht sein. Ich bin für jedes Feedback offen, sei es (möglichst konstruktive) Kritik oder Lob, die Mitteilung inhaltlicher, sachlicher oder orthographischer Fehler, oder auch miß- oder unverständliche Ausdrucksweise. Sachdienliche Hinweise bitte per e-mail oder per snailmail an unten genannte Adresse.

Wer diesen Text verbreiten möchte der darf das gern tun, sollte ihn jedoch unverändert lassen, damit die beiliegende, abgekoppelte Unterschrift gültig bleibt (die sollte natürlich auch immer mit verbreitet werden, weil im Text auf sie hingewiesen wird, gleiches gilt für die mitgelieferten Beispielschlüssel). Fehler im Text korrigiere ich gern auf Hinweis.

Nun aber los, ich wünsche dir viel Erfolg beim Erlernen von PGP.

Jürgen Pöttsch <yogi@newswire.de>
Am Böhnerfeld 37
41516 Grevenbroich

Besonderer Dank für Kritik und Verbesserungsvorschläge an:

Martin Gerken <smarty@darkness.gun.de>
Frank Prüfer <F.PRUEFER@LINK-L.cl.sub.de>

Was macht PGP?

PGP ...
... verschlüsselt Texte und Binärdateien
... unterschreibt Texte und Binärdateien
... entschlüsselt Texte und Binärdateien
... prüft Unterschriften von Texten und Binärdateien
... packt Texte und Binärdateien beim Verschlüsseln
... wandelt 8-Bit-Binärdateien in 7-Bit-ASCII
... erzeugt Schlüssel
... editiert Schlüssel
... verwaltet Schlüssel

Wofür braucht man PGP?

PGP braucht man im Allgemeinen zur Verschlüsselung privater Post, die man per e-mail verschicken will, ganz einfach um zu verhindern, daß andere als der Absender und der Empfänger diese Post lesen können (egal ob der Angriff absichtlich oder versehentlich geschieht). Ein weiterer Grund PGP zu benutzen, ist der, daß man seine öffentlichen Postings und privaten Mails fälschungssicher signieren kann.

Wie installiere ich PGP?

Zuerst einmal besorgst du dir das PGP-Paket aus dem Filemenue einer Mailbox, von einer Shareware-CD, oder per FTP von einem FTP-Server. Eine gute Wahl, wenn man Mailboxen sucht, sind z.B.:

BIONIC (Bielefeld)
Tel.: 0521/68000 (V.32bis)
0521/171188 (V.34)
0521/9680869 (ISDN)
Login: PGP

NEWSWIRE (Willich)
Tel.: 02154/5011 (V.32bis)
02154/87070 (V.34)
02154/951900 (ISDN)
Login: GAST

Dort findest du nicht nur die jeweils aktuellste Version von PGP (für alle Rechnerplattformen), sondern auch die Pakete für verschiedene Sprachversionen, Quelltexte und Hilfetexte.

Jetzt legst du ein Verzeichnis deiner Wahl für PGP an, (z.B. C:\PGP), entpackst das Archiv in dieses Verzeichnis, und trägst in der AUTOEXEC.BAT folgende zwei Variablen ein:

SET PGPPATH=C:\PGP (oder entsprechend deinen Gegebenheiten)
SET TZ=CET-1DST (darauf kommen wir noch im Verlauf des Kurses)

Die PATH-Variable erweiterst du durch Angabe des PGP-Verzeichnisses.

Nach einem Neustart des Rechners ist die Installation abgeschlossen.

PGP in deutsch

Wer möchte, kann seinem PGP auch Deutsch beibringen, ich werde im weiteren Verlauf des Kurses alle Fehlermeldungen und Ausgaben der deutschen Version wiedergeben. Die Befehle bleiben in beiden Versionen gleich und die Meldungen sind im allgemeinen wörtlich übersetzt, so daß man die englischen Meldungen gut mit den deutschen Meldungen vergleichen kann.

Du besorgst dir das deutsche Sprachpaket (Quellen: s.o.), und entpackst alle darin enthaltenen Dateien in dein PGP-Verzeichnis. Eventuelle Fragen, ob eine bereits bestehende Datei überschrieben werden soll, beantwortest du getrost mit "j" (oder "y", je nach Packer). Das war's schon, du arbeitest jetzt mit deutschen Hinweisen und Fehlermeldungen.

Weitere z.T. wichtige Hinweise zum deutschen Sprachkit, findest du im mitgelieferten README.LNG.

Erzeugen eines Schlüsselpaares

Jetzt generierst du dir ein Schlüsselpaar. Warum du das tust, und warum es ein "Paar" sein muß, wird im Verlauf des Kurses von selbst klar.

Wechsele also ins PGP-Verzeichnis und geben den Befehl "PGP -kg" ein ("-kg" steht für "key" und "generate"). PGP fragt dich nach einer Schlüssellänge und gibt einige Hinweise darauf, was es damit auf sich hat. Theoretisch kannst du hier fast beliebige Werte für die Schlüssellänge angeben, man sollte aber aus Kompatibilitätsgründen einen der drei vorgeschlagenen Werte oder 2048 Bit wählen. Veralterte PGP-Versionen (z.B. 2.3a) können diese großen Schlüssel zwar nicht lesen, aber da ohnehin (hoffentlich) niemand mehr mit diesen Steinzeitversionen arbeitet, muß man darauf auch keine Rücksicht nehmen, zumal kleinere Schlüssel einen Verlust eigener Sicherheit bedeuten.

Prinzipiell gilt:

- je länger ein Schlüssel ist, desto sicherer sind die damit verschlüsselten Daten,
- mit zunehmender Länge steigt aber auch die Zeit, die dein Rechner braucht um damit Nachrichten zu verschlüsseln.

Hast du dich für eine Schlüsselgröße entschieden, mußt du eine User-ID angeben. Hier kann man theoretisch angeben was man will, sollte sich aber an einen de facto Standard halten und seine ID in der Form ...

Vorname Nachname <user@host.domain>

... angeben, wer keine e-mail Adresse hat, gibt seine Telefonnummer, oder ein sonstiges eindeutiges und persönliches Merkmal an. Umlaute akzeptiert PGP hier nicht, damit erledigt sich die Frage nach deren Schreibweise. Man hat später jederzeit die Möglichkeit, weitere User-IDs anzufügen. Eine User-ID muß einen Realnamen tragen, um eindeutig und überprüfbar einer Person zugeordnet werden zu können, darauf komme ich später noch zurück.

Die nächste Frage gilt der "pass phrase". Eine pass phrase ist dasselbe wie ein "password", nur daß man hier wirklich ganze Sätze inklusive Sonderzeichen und Leerzeichen angeben kann. Im Umgang mit PGP wird diese pass phrase auch gern als "Mantra" bezeichnet, damit ist auch schon erklärt, was damit gemeint ist. (Vorschlag: für den Anfang, zum Testen der Funktionen von PGP, kann man auch "abc" als Mantra angeben, das läßt sich schneller eintippen und später leicht ändern, man sollte es aber nicht vergessen.) Hat man das Mantra eingegeben, bittet PGP zur Sicherheit, um Tippfehler auszuschließen, um nochmalige Eingabe des Mantras.

Um den Schlüssel zu erzeugen, braucht PGP jetzt ein paar wirklich zufällige Zahlen,

und bittet Dich, um ein paar Tastaturanschläge. Hier kannst du deinen Fingern freien Lauf lassen und solange tippern (egal was), bis es piept. Das Zufällige an dieser Eingabe ist nicht der Text, sondern der zeitliche Abstand zwischen den Anschlägen. Der ist so zufällig, daß es nicht einmal dir gelingen wird, zweimal exakt dieselben Zeiten zu erreichen. Es kann nicht schaden, hier einen normalen Text einzugeben, Tasten gedrückt halten zählt eh nicht, weil PGP jede Taste höchstens zweimal hintereinander akzeptiert.

PGP berechnet nun deine beiden Schlüssel und meldet nach wenigen Sekunden (bei langsamen Rechnern und langen Schlüsseln kann es auch bis zu einer Stunde dauern!), während derer es ein paar Punkte und Sternchen am Bildschirm anzeigt, daß es die Schlüssel generiert hat.

Herzlichen Glückwunsch, du besitzt jetzt dein eigenes Schlüsselpaar, und damit die erste Tür zur Teilnahme am Mailverkehr mit PGP geöffnet.

Warum man diese Schlüssel braucht, und was man damit macht, erfährst du in den nächsten Abschnitten dieses Kurses.

Bearbeiten des Schlüsselpaares

Du hast dir jetzt also ein Schlüsselpaar erzeugt. Daß es wirklich zwei Schlüssel sind, kannst du z.B. daran erkennen, daß PGP im PGP-Verzeichnis zwei neue Dateien angelegt hat, nämlich *SECRING.PGP* und *PUBRING.PGP*.

Secring steht für "secret key ring" (geheimer Schlüsselbund), pubring steht für "public key ring" (öffentlicher Schlüsselbund). Ein "key ring" ist tatsächlich so etwas wie ein Ring, an den man Schlüssel anhängen und auch wieder davon entfernen kann. Wie die Namen schon sagen, enthält *SECRING.PGP* alle deine gesammelten geheimen Schlüssel, während du in *PUBRING.PGP* alle deine gesammelten öffentlichen Schlüssel findest. Momentan befinden sich in beiden Dateien natürlich nur deine beiden eigenen Schlüssel, und das wollen wir jetzt kontrollieren.

Du gibst nacheinander die Befehle ...

```
PGP -kvv <userid> pubring.pgp
und
PGP -kvv <userid> secring.pgp
```

ein, wobei "<userid>" ein Teil deiner User-ID ist, der keine Leerzeichen enthalten darf, oder sonst in Anführungszeichen zu setzen ist.

Wir erhalten eine Ausgabe wie diese (hier für meinen Key):

```
-----
Typ  Bits    ID          Datum      Benutzer
öff   512/0D0AC4D9 1994/04/03 Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>

prv   512/0D0AC4D9 1994/04/03 Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
Es wurde ein passender Schlüssel gefunden.
```

(Einige Meldungen wurden hier weggelassen!)

Die Angaben bedeuten:

Typ: (engl.: Type)

öff - es handelt sich um einen public key (einen öffentlichen Schlüssel)

prv - es handelt sich um einen secret key (einen privaten Schlüssel)

Bits:

Die Schlüssellänge in Bit (hier handelt es sich um einen 512-Bit-Schlüssel)

ID (engl.: keyID)

Eine (wahrscheinlich weltweit) eindeutige 32 Bit breite Identifikationsnummer.

Datum: (engl.: Date)

Das Erstellungsdatum des Schlüssels

Benutzer: (engl.: User ID)

Eben die gewählte User-ID

Den secret Key (der Name spricht für sich) hütetest du ab sofort wie deinen Augapfel, während du den public Key so weit verbreiten solltest, wie möglich (Halt! Noch nicht, da fehlt noch was, und wie man ihn am besten verbreitet, erfährst du auch noch).

Wichtig:

Gib deinen secret key **niemals** weiter und achte auch darauf, daß niemand in seinen Besitz gelangt, der Zugriff auf deinen Rechner hat! Der secret Key ist zwar durch dein Mantra zusätzlich geschützt, aber dieses Mantra ist nur so sicher, wie Du es machst! Wenn Du z.B. ein sinnvolles Wort oder den Namen einer bekannten Person wählst (was leider viel zu oft vorkommt), dann kann ein Programm, das eine lexikalische Suche beherrscht, oder jemand der Dich persönlich kennt, dieses Mantra recht leicht herausfinden. Wie ein gutes Passwort aufgebaut wird, erfährst Du später in diesem Kurs.

Jeder, der in den Besitz deines secret keys (mit dem dazugehörenden Mantra) gelangt, kann Nachrichten in deinem Namen unterschreiben, wobei du Probleme haben wirst zu beweisen, daß die Unterschrift nicht von dir stammt, und er kann alle an dich gerichteten Nachrichten entschlüsseln.

Bevor du deinen public key verbreitest, **unterschreibst** du ihn! "Warum denn das?" wirst du vielleicht jetzt fragen, aber das ist schnell beantwortet. Stellen wir uns vor, ein "Freund" von dir erzeugt einen Schlüssel mit **deiner** User ID. Das ist, wie wir wissen, kein Problem, jeder kann einen Schlüssel unter beliebigem Namen erzeugen, das kann PGP nicht kontrollieren. Dabei erhält sein falscher Schlüssel eine andere key-ID, als dein echter Schlüssel. Diese key-ID ist aber für einen Amateur innerhalb der PUBRING.PGP editierbar, er kann also seine falsche ID durch deine echte ID ersetzen. Verbreitet er diesen falschen Schlüssel, ist dieser Schlüssel auf den ersten Blick durch nichts von deinem echten Schlüssel zu unterscheiden. Was der "Freund" aber nicht kann, er kann seinen falschen Schlüssel nicht mit deiner echten ID **unterschreiben**, das kannst **nur du** mit deinem secret key! Mit der Unterschrift unter deinem public key bestätigst Du, daß du im Besitz des passenden secret keys bist, und diese Unterschrift ist (im Gegensatz zur Key-ID) **nicht editierbar**, der Versuch eine Unterschrift zu fälschen, fällt PGP sofort auf.

Wenn Du mit der PGP-Version 2.6.3i arbeitest, brauchst Du Deinen Key nach der Erstellung nicht zusätzlich zu unterschreiben, PGP tut das automatisch und ohne Dein Zutun.

Nach dem Befehl...

`PGP -ks <name>`

... sieht unser Schlüssel z.B. so aus:

```
-----
öff   512/0D0AC4D9 1994/04/03 Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
unt   0D0AC4D9          Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
-----
```

Die Unterschrift (Type: unt, in der englischen Version heißt sie "sig") gehört ab sofort fest zu diesem Schlüssel, damit ist bestätigt, daß der Eigentümer (hier ich) seinen eigenen Schlüssel mit seinem secret key beglaubigt hat, es sich also um keine Fälschung handelt.

Merke: ein Schlüssel, der nicht von seinem Besitzer selbst unterschrieben ist, ist wertlos!

Nun sorgst du noch dafür, daß deine Schlüssel nicht verloren gehen können, z.B. durch einen Festplattendefekt. Dazu brauchst du nur deine beiden nagelneuen Schlüsselbunde (PUBRING.PGP **und** SECRING.PGP) auf eine leere Diskette zu kopieren und diese an einem **sicheren** Ort aufzubewahren (siehe dazu obige Hinweise zum secret key). Abgesehen vom Plattencrash gibt es noch eine Reihe von Fällen, in denen man diese Sicherheitskopie brauchen kann, denken wir nur wieder an deinen "guten Freund". Stellen wir uns vor, dein "Freund" nutzt einen unbeobachteten Moment an deinem Rechner und kopiert deine Schlüssel auf eine Diskette. Er ist sogar ein so "guter Freund", daß er die Schlüssel von deiner Platte löscht. Von nun an kann er unter deinem Namen verschlüsselte Nachrichten empfangen und mit deinem guten Namen Nachrichten unterschreiben. Das allein wäre schon schlimm, aber du könntest deinen Schlüssel nicht einmal für ungültig erklären. Bist du aber im Besitz einer Sicherheitskopie, kannst du (mit Hilfe von PGP) ein sogenanntes "key revokation certificate" (ein Schlüsselrückrufszertifikat, das übrigens die Typbezeichnung "zur" besitzt) erzeugen, und dies genau wie deinen öffentlichen Schlüssel verbreiten. Damit wird dein Schlüssel **weltweit** in allen key rings als zurückgerufen erklärt, und jedes PGP auf der Welt wird dann davor warnen, diesen Schlüssel zu benutzen. Zu all dem später mehr.

Verbreiten von Schlüsseln

Als ich im vorigen Absatz davon gesprochen habe, daß du deinen Schlüssel "weltweit" für ungültig erklären kannst, dann mag das etwas weit hergeholt klingen, ist es aber nicht. Es gibt ein Brett namens /Z-NETZ/ALT/PGP/SCHLUESSEL, das jedem die Möglichkeit gibt, seinen Schlüssel dort zu veröffentlichen. Damit ist er für jeden (der das Brett bezieht) deutschlandweit verfügbar. Aber damit nicht genug, es gibt mindestens einen User (wahrscheinlich mehrere), die sich ein Hobby daraus machen, diese Schlüssel zu sammeln, und regelmäßig an einen Keyserver weiterzuleiten. Von diesen Keyservern gibt es weltweit eine ganze Reihe, und sie synchronisieren sich täglich! Ein Schlüssel, der heute z.B. an pgp-public-keys@fbihh.uni-hamburg.de geschickt wird, wird morgen auch auf den Keyservern in Australien verfügbar sein, gleiches gilt natürlich auch über oben genanntes Rückrufszertifikat.

Du kannst deinen Schlüssel natürlich auch selbst an einen Keyserver schicken, um das zu tun, mußst du ihn erst einmal aus seinem Schlüsselbund extrahieren, und das geht mit dem Befehl:

```
PGP -kxa <userid> [filename]
```

das "-kxa" steht dabei für "Key eXtract Ascii" und du erhältst damit eine kleine ASCII-Datei, mit dem public key der zu "<userid>" gehört. Diese Datei kannst du jetzt entweder in /Z-NETZ/ALT/PGP/SCHLUESSEL posten (Betreff ist beliebig), oder per PM an Freunde schicken, oder selbst per PM an o.g. Keyserver schicken, du erhältst dann eine Bestätigung des Servers, daß der Schlüssel aufgenommen wurde.

So sollte die Datei aussehen, die den Schlüssel enthält:

Typ	Bits	ID	Datum	Benutzer
öff	512	0D0AC4D9	1994/04/03	Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: 2.6.3i

```
mQBNai2fDI4AAAEcANsCGkhLlXSXMgjQ0NgzSt56Pd/Qho8s5NYIayyHgR2w9D6J
VBqyXLfeSq0tynurxdPcedW/rmQIBpmZ2Q0KxNkABRG0J0p1ZXJnZW4gUG9ldHpz
CsTZAQHzlgh9Ek0BGmsNnZ9EmOlZz2pZX9yxj33iqaDwwjRi3vXtOiFws5gBVDOG
```

```
TKlDt09PDdRjdv2jSz0zGpxSkzFOkWA7/A==
=naUN
-----END PGP PUBLIC KEY BLOCK-----
```

Sie kann kürzer sein als diese hier (aber auch länger, das hier ist ohnehin nur ein gekürztes Beispiel, kein echter Schlüssel), wichtig ist die Zeile mit "BEGIN PGP PUBLIC KEY BLOCK" und "END PGP PUBLIC KEY BLOCK", denn nur wo public key draufsteht, ist auch public key drin. ;-)

Editieren von Schlüsseln

Nachdem du jetzt sicher fleißig deine Schlüssel erzeugt und unterschrieben hast, kannst du daran gehen sie eventuell noch etwas zu erweitern. Vielleicht hast du mehrere e-mail Adressen, und da PGP dazu dienen soll, verschlüsselte Nachrichten per e-mail auszutauschen, ist es eine gute Idee deine verschiedenen User-IDs in deinem Schlüssel zu dokumentieren. Versuche es, und gib den Befehl "PGP -ke <userid>" ein ("-ke" steht für "Key Edit", was "<userid>" bedeutet, dürfte mittlerweile klar sein). Da du einen Schlüssel bearbeitest, der auch in deinem geheimen Schlüsselbund (SECRING.PGP) vorhanden ist, wirst du nach der pass phrase gefragt (deshalb vorher der Hinweis, das Mantra für diese Probierphase möglichst kurz zu halten). Nach Eingabe des richtigen Mantras wird dir deine User-ID angezeigt und gefragt, ob du eine neue User-ID eingeben willst. Du willst (auch wenn du keine zweite e-mail Adresse hast solltest du hier zur Übung einfach eine erfundene Adresse eingeben), und bestätigst die Frage mit "j" für "Ja", in der englischen Version wäre es "y" für "yes". Für die neue User-ID hältst du dich wieder an die oben beschriebene Konvention.

Es folgt die Frage, ob du diese ID zur primären ID machen willst. Da du dich aber (hoffentlich) in deiner Stammbox zu Hause fühlst, bzw. gar keine zweite User-ID hast und das hier nur zur Übung machst, verneinst du diese Frage, damit wird die neue ID zur sekundären User-ID. Die Frage ob du eine neue pass phrase eingeben willst, verneinst du ebenfalls, weißt aber jetzt auch schon, wie du das Mantra ändern kannst. Du erhältst noch kurz eine Bestätigung, daß SECRING.PGP und PUBRING.PGP aktualisiert wurden, und kannst dir jetzt deinen erweiterten Schlüssel ansehen (wir erinnern uns: PGP -kvv userid), er sieht jetzt z.B. so aus:

```
öff  512/0D0AC4D9 1994/04/03 Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
unt  0D0AC4D9                Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
                                Juergen Poetzsch <100540,501@compuserve.com>
```

Die neue User-ID wurde unten angehängt, für sie gibt es keine Typangabe und keine eigene Key-ID, da für sie Typ und ID gesamten Keys gilt, sie ist eben nur ein Teil dieses Schlüssels.

Was schon für die primäre User-ID galt, gilt auch für die alternative User-ID: du solltest sie unterschreiben um fälschungssicher zu dokumentieren daß du selbst diese ID erstellt hast (auch hier gilt: ab Version 2.6.3i unterschreibt PGP eine alternative User-ID automatisch). Du gibst wieder ein "PGP -ks userid" ein, wobei diesmal "userid" ein eindeutiger Teil der neuen User-ID sein muß, sonst beschwert sich PGP, daß dieser Schlüssel (damit meint es die primäre User-ID) bereits von dir unterschrieben sei.

Ein Blick auf den Schlüssel:

```
öff  512/0D0AC4D9 1994/04/03 Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
unt  0D0AC4D9                Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
                                Juergen Poetzsch
<100540,501@compuserve.com>
unt  0D0AC4D9                Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>
```

Zu beachten ist hier, daß die zweite "sig", also die zweite Unterschrift *nur* für

die User-ID gilt, unter der sie steht, nicht für den gesamten Schlüssel. Das ist hier noch nicht wichtig, aber wir kommen später darauf zurück, da wird es wichtig sein.

Zur Übung (und zur Abwechslung) änderst du zwischendurch einmal kurz dein Mantra. Auch dafür eignet sich, wie wir mittlerweile wissen, der Befehl `PGP -ke name`. Die Frage nach der neuen User-ID kannst du hier verneinen, dafür bejahst du die Frage nach Änderung der pass phrase. Du gibst ein neues Mantra ein (das aber auch gleich mit dem alten sein darf), bestätigst es noch einmal zur Sicherheit und bekommst angezeigt, daß `SECRING.PGP` aktualisiert wurde, zur Änderung von `PUBRING.PGP` aber kein Anlaß bestand. Das ist verständlich, denn das Mantra schützt nur den secret key, auf den public key hat es keinen Einfluß.

Das Mantra ist das wichtigste Sicherheitsmerkmal deines secret keys, sollte er irgendwann einmal in falsche Hände gelangen, schützt ihn nur das Mantra vor Mißbrauch. Ein Mantra sollte deshalb möglichst schwer zu "erraten" sein. Dazu gibt es verschiedene Kriterien ein sicheres Mantra zu erzeugen, dabei geht es darum einem Angreifer das Ausprobieren deines Passwortes (Besser gesagt des "Pass-Satzes") weitgehend schwer zu machen:

1. je länger, desto besser, (macht Computern mehr Arbeit beim Ausprobieren) es kann gar nicht lang genug sein,
2. je kryptischer, desto besser, also besser Sätze mit wild gemischten Buchstaben ohne Sinn, als sinnvolle Wörter (macht Menschen das Erraten schwerer),

Ideal sind Mantras, die keinen Sinn ergeben, und möglichst lange Ketten zufälliger Zeichen enthalten, bunt gemischt in Groß- und Kleinschreibung.

Beispiele für schlechte Mantras:

```
"Passwort",  
"Mantra",  
"pgp",  
"abc",  
<Name der Freundin, Ehefrau, Geliebten>
```

Beispiele für gute Mantras:

```
"$%# GrMbl? ,\! SchOKol#d& ...=63 |&&"  
oder:  
"raTe#mal/wIemeinBa??worD_H*jsst}"
```

Noch ein Wort zu User-IDs: wie du gesehen hast, ist es sehr leicht, eine User-ID zu erzeugen. Es ist zwar ebenso leicht sie wieder zu löschen (das tun wir gleich), dabei gibt es aber ein kleines Problem. Wenn du deinen Key mit einer neuen User-ID versehen hast, und ihn verbreitest, später aber diese User-ID wieder entfernst, dann ist sie nur in deinem eigenen Keyring entfernt. Alle anderen Keyrings (z.B. auf den weltweit verstreuten Keyservern) erfahren davon nichts. Und selbst wenn du deinen neuen Key dann an einen der Keyserver schickst, nimmt der davon keine Notiz, er registriert nur hinzukommende User-IDs und neue Unterschriften, entfernt aber keine User-IDs. Man sollte deshalb keine e-mail-Adressen in seinen Key aufnehmen, die man nur kurzzeitig vorhat zu nutzen.

Nun aber zum Entfernen einer User-ID aus unserem Keyring: dazu dient der Befehl `"PGP -kr User-ID"` ("`kr`: Key Remove"). Damit läßt sich sowohl der gesamte Key aus dem Keyring entfernen, als auch nur eine einzelne User-ID, deshalb ist beim Kommando `"-kr"` äußerste Vorsicht angesagt, denn ein gelöschter Key ist unwiederbringlich weg, es sei denn, man hat eine Sicherheitskopie des Keyrings (hast du doch mittlerweile, oder etwa nicht?). Sobald ein Key mehrere User-IDs hat, fragt PGP nach, ob der gesamte Key, gelöscht werden soll. Du willst natürlich nur deine letzte eingefügte User-ID wieder löschen und antwortest auf die Frage, ob der gesamte Key gelöscht werden soll, mit `"n"`. Du wirst nun zu jeder einzelnen User-ID gefragt, ob du sie löschen willst.

Antwortest du mit "j", verschwindet die User-ID aus dem public key und zwar mit allen ihren Unterschriften.

Bleibt zum Editieren von Schlüsseln eigentlich nur noch eine Übung: das Entfernen einer Unterschrift. Wie du (hoffentlich noch) weißt, gehört eine Unterschrift immer nur zu einer User-ID. Willst du eine Unterschrift entfernen, mußt du angeben, zu welcher User-ID die Unterschrift gehört, die du entfernen möchtest. Der Befehl lautet dann: "PGP -krs User-ID" ("-krs": Key Remove Signature). Trägt eine User-ID mehrere Unterschriften (was unsere IDs noch nicht tun), dann wirst du auch hier zu jeder Unterschrift einzeln gefragt, ob man sie entfernen möchte.

Wenn du jetzt üben möchtest, fügst du an deinen Key beliebige User-IDs an, unterschreibst sie, löschst die Unterschriften, unterschreibst erneut, löschst User-IDs usw. Deiner Phantasie sind hier keine Grenzen gesetzt.

Im nächsten Kapitel gehen wir an's Interessanteste, wir verschlüsseln Nachrichten.

Verschlüsseln von Nachrichten

Vielleicht hast du dir schon einmal Gedanken darüber gemacht, wie man Dateien oder Texte so verschlüsseln kann, daß sie nicht jeder gebrauchen kann. Vielleicht hast du auch schon einmal mit PKZIP oder ARJ und den darin enthaltenen Verschlüsselungsfunktionen gearbeitet. Theoretisch könnte man diese Packer dazu benutzen, seine e-mails zu verschlüsseln, und sie dem Empfänger zu schicken. Das Problem an der Sache wäre das Passwort, denn der Empfänger muß den Text mit demselben Passwort auspacken, mit dem ihn der Absender eingepackt hat.

Man muß sich bei dieser konventionellen Art der Verschlüsselung entweder einmal persönlich treffen, oder per Brief oder Telefon ein Passwort ausmachen, das dann beiden bekannt ist. Bei regem Mailverkehr mit mehreren Partnern, hat man dann schnell eine beachtliche Sammlung an Passworten, die man sich aufschreiben muß um sie nicht zu vergessen. Was davon zu halten ist Passworte aufzuschreiben, brauche ich wohl nicht zu erwähnen.

Anders ist das alles bei PGP, da brauchst du einen Empfänger weder gesehen noch angerufen zu haben, er braucht nicht einmal von dir gehört zu haben, und du kannst ihm dennoch verschlüsselte Post schicken, die **nur er** entschlüsseln kann. Dazu holst du dir seinen public key von einem Keyserver, und verschlüsselst damit deinen Text, den du ihm schicken möchtest. Der Empfänger (und nur er, weil nur er im Besitz des secret keys ist, der zum verschlüsselnden public key passt), kann diese verschlüsselte Nachricht nun problemlos mit seinem secret key entschlüsseln.

Nochmal deutlich die Funktion der beiden Schlüssel:

verschlüsselt wird mit dem **public key**,
entschlüsselt wird mit dem **secret key**.

Möchte der Empfänger der verschlüsselten Nachricht dir antworten, braucht auch er **deinen** public key um verschlüsseln zu können, den kannst du ihm in deiner verschlüsselten Nachricht gleich mitschicken, oder aber er holt ihn sich ebenfalls von einem Keyserver. Du **entschlüsselst** die Antwort dann mit **deinem** secret key.

Damit ist im Prinzip schon alles Wissenswerte zu den Vorgängen beim Austausch verschlüsselter Nachrichten gesagt!

Schauen wir einmal kurz ein paar Absätze zurück, da habe ich davon gesprochen, daß es bei konventioneller Verschlüsselung umständlich ist, mit den vielen Passworten umzugehen, wenn man mit vielen Leuten verschlüsselte Post austauschen will. Das wäre mit den vielen public keys bei PGP noch viel schlimmer, wenn PGP uns nicht eine (halbwegs komfortable) Schlüsselverwaltung zur Verfügung stellen würde.

Bei der Erklärung der Schlüsselbunde habe ich davon gesprochen, daß sich in unserem

PUBRING bisher nur unser eigener Schlüssel befindet, das wollen wir jetzt ändern. Zu diesem Zweck habe ich dem Archiv in dem sich dieser Text befand einige public keys beigelegt (*.asc), wenn das Archiv unverändert weitergegeben wurde, dann müsstest du jetzt auch im Besitz dieser Schlüsselsammlung sein. Um einen dieser Schlüssel in Deinen Keyring aufzunehmen, gibst du einfach den Befehl "PGP -ka dateiname" ein, logischerweise ist "dateiname" der Name der Datei, die den gewünschten Schlüssel enthält. PGP sucht dann in dieser Datei nach Schlüsseln, die es noch nicht in seinem Schlüsselbund hat, und addiert sie gegebenenfalls ("-ka": Key Add). Neue User-IDs und Unterschriften werden den schon bekannten Schlüsseln hinzugefügt.

ACHTUNG: PGP wird dich darauf hinweisen, daß einer oder mehrere Schlüssel aus dieser Schlüsseldatei nicht ausreichend beglaubigt sind, das liegt daran, daß zwar (hoffentlich) der Schlüsselinhaber seinen Schlüssel selbst unterschrieben hat, PGP aber nicht weiß, ob die Person deren Name in der User-ID angegeben ist, auch wirklich der Erzeuger des Schlüssels ist. Schließlich kann jeder unter jedem Namen einen Schlüssel erzeugen. Die Frage ob Du diesen Schlüssel selbst beglaubigen willst, beantwortest du mit **nein**, und (wenn es ginge) 1000 mal nein, denn auch du kannst nicht wissen, wer den Schlüssel wirklich erzeugt hat.

Ich komme später darauf zurück, bis dahin beglaubigst du **auf keinen Fall** einen Schlüssel bzw. eine User-ID, außer deiner eigenen!!!

Nachdem du einen (oder mehrere) fremde Schlüssel in deinen PUBRING aufgenommen hast, kannst du die mitgelieferten Schlüsseldateien eigentlich löschen, die weitere Schlüsselverwaltung übernimmt PGP. Die Befehle "PGP -kv" (key view) und "PGP "-kvv" (key view verbose) zeigen dir jetzt alle in deinem Schlüsselbund enthaltenen Key-IDs und deren verschiedenen User-IDs an. Der Unterschied zwischen "-kv" und "-kvv" ist einfach der, daß mit "-kv" die Schlüssel mit allen ihren User-IDs, mit "-kvv" aber zusätzlich mit den dazugehörenden Signaturen angezeigt werden.

Jetzt aber ran an's Verschlüsseln

Dazu suchst du dir einen beliebigen Text oder eine beliebige Binärdatei aus, die du für irgendjemanden verschlüsseln willst. Dieser "irgendjemand" kann jeder sein, dessen Schlüssel du in deinem Schlüsselbund hast, eingeschlossen du selbst. Um eine Datei zu verschlüsseln gibst du den Befehl "PGP -e dateiname User-ID" ("e": Encrypt). "Dateiname" ist dabei natürlich die Datei, die du verschlüsselst, "User-ID" ist ein Teil der User-ID des Empfängers. Wenn du eine fremde User-ID zum Verschlüsseln wählst, wird PGP meckern daß der verwendete Schlüssel nicht beglaubigt ist, und du nicht sicher sein kannst, daß der Schlüssel wirklich dem gehört, dessen Namen er trägt. Da du die Schlüssel, die du eingelesen hast, absichtlich nicht beglaubigt hast, ignorierst du die Warnung zunächst, und beantwortest die Frage, ob du den Schlüssel trotzdem benutzen willst, mit "j".

PGP verschlüsselt nun also deine Datei, und erzeugt eine neue Datei die den gleichen Namen erhält wie deine Ausgangsdatei, sowie die Erweiterung "PGP" (z.B. würde AUTOEXEC.BAT zur AUTOEXEC.PGP). Schaust du dir die neue Datei z.B. mit einem Editor an, siehst du nichts Brauchbares, denn PGP hat eine Binärdatei erzeugt. Das ist zum Versand per e-mail nicht gerade erwünscht, da einige Systeme oder Gates die merkwürdigsten Dinge mit Binärdateien anstellen. Du willst PGP deshalb dazu veranlassen, eine 7-Bit-ASCII-Datei zu erstellen (so etwas ähnliches wie eine UUENCODEte mail). Dazu brauchst du dem Encode-Befehl nur noch ein "a" anzuhängen, also lautet unser Befehl "PGP -ea dateiname User-ID". Deine verschlüsselte Datei erhält dann die Endung "ASC" für ASCII, und läßt sich mit einem Editor ansehen (viel sinnvoller wird dir das was du da siehst aber wohl auch nicht erscheinen, es ist eben verschlüsselt).

Dateigrößen

Siehst du dir die Dateigrößen der Originaldatei und der beiden verschlüsselten Dateien an, stellst du fest, daß die binärvercryptete Datei weitaus kleiner ist als das Original, die ASCII-vercryptete Version zwar nicht ganz so klein, aber immer noch kleiner als das Original ist. Das liegt daran, daß PGP jede Datei vor dem Verschlüsseln mit einem ZIP-Algorithmus packt. Durch die Umwandlung in 7-Bit-ASCII wird die gepackte Datei allerdings wieder etwas größer (wie beim UUencoden).

Datei entschlüsseln

Wenn du beim Verschlüsseln deine eigene User-ID angibst, kannst du die verschlüsselte Datei auch wieder entschlüsseln, dazu reicht der PGP-Befehl ohne Angabe von Parametern, außer dem Dateinamen versteht sich. "PGP dateiname" macht demnach aus DATEI.PGP oder auch DATEI.ASC wieder die Ursprungsdatei, allerdings ohne Dateinamenserweiterung, sprich aus DATEI.PGP oder DATEI.ASC wird DATEI. Existiert diese Datei bereits, wird gefragt, ob PGP die Datei überschreiben soll.

Hast du beim Verschlüsseln eine andere User-ID als deine eigene angegeben, schlägt jeder Versuch fehl, die Datei zu entschlüsseln! PGP wird dir sagen, daß die Datei verschlüsselt ist, wessen secret key notwendig ist um sie zu entschlüsseln, und daß du nicht im Besitz dieses Schlüssels bist. Damit hat sich für PGP die Sache erledigt, und du schaust in die Röhre. ;-)

Zusammenfassung des Ver- und Entschlüsselns

Um eine Datei an jemanden zu verschlüsseln, dessen public key du hast, brauchst du dir kein Passwort zu merken, sondern kannst einfach einen Teil seiner User-ID (z.B. den Nachnamen) angeben. Der Befehl "PGP -e[a] dateiname User-ID" verschlüsselt die Datei (mit "a" als 7-Bit-ASCII-Datei), der Befehl "PGP dateiname" entschlüsselt eine Datei, vorausgesetzt sie ist an dich verschlüsselt und du besitzt den passenden secret key. Übrigens kannst du auch versuchen eine Schlüsseldatei (die mit dem PUBLIC KEY BLOCK) zu entschlüsseln, dabei wird PGP feststellen, daß es sich um einen oder mehrere Schlüssel handelt, dir diese anzeigen, und fragen ob du ihn/sie in deinen Schlüsselbund aufnehmen willst.

Vertrauensstufen

Zunächst mußt du dir darüber klarwerden, wem und auf was du überhaupt in Bezug auf PGP vertrauen kannst. Da sind zunächst einmal die Schlüssel, und da sind die User, deren Schlüssel du in deinem Keyring hast. Beide werden von PGP (und deshalb auch von dir) unterschiedlich behandelt, und in unterschiedliche Vertrauensstufen einsortiert.

Vertrauen zu einem Schlüssel

Wenn du einen Schlüssel über /Z-NETZ/ALT/PGP/SCHLUESSEL oder einen der Keyserver erhältst, dann weißt du über diesen Schlüssel **gar nichts**. Du weißt zwar wem er gehören **soll**, aber nicht wem er wirklich gehört und wer ihn erzeugt hat. Es wäre ein Leichtes, einen Key mit einem beliebigen Usernamen und dem dazu passenden Realnamen zu erzeugen, ihn selbst zu unterschreiben, und ihn über das Netz zu verteilen. Deshalb gehört dieser Schlüssel aber nicht dem, dessen Namen er trägt, du hast es also mit einem "Fake" zu tun.

Wie du weißt, kann (und muß!) man seinen eigenen Schlüssel mit seinem secret key signieren um seine Echtheit zu bestätigen. Ebenso kann man aber auch fremde Schlüssel signieren, und damit bestätigen, daß man sie für echt hält. Aus den im vorigen Absatz genannten Gründen **darf** man **niemals** einen Schlüssel unterschreiben, von dem man nicht 100%ig ... nein besser 1000%ig überzeugt ist, daß er dem gehört, dessen Namen er trägt!!! Auch wenn die Angriffsmöglichkeiten manchmal, z.B. innerhalb einer Mailbox,

äußerst gering sein mögen, unterschreibt man **niemals** einen Key, den man nicht persönlich vom Inhaber erhalten hat (persönlich heißt auch: **nicht** per PM!!), und dessen Identität man nicht wirklich überprüft hat (z.B. durch Vergleich der User-ID mit dem Personalausweis des Überbringers). Aus diesem Grund ist es auch witzlos eine User-ID ohne Realnamen zu erzeugen.

Wer einen Schlüssel unterschreibt, wird von PGP ausdrücklich gefragt (hier in der deutschen Version):

SORGFÄLTIG LESEN: Bist Du, gestützt auf eigenes, direktes Wissen aus erster Hand, absolut sicher, daß du zuverlässig beglaubigen kannst, daß der oben angezeigte öffentliche Schlüssel wirklich zu der oben genannten Person gehört? (j/N)

Den Satz muß man sich mal auf der Zunge zergehen lassen: ... eigenes, direktes Wissen... aus erster Hand... absolut sicher... zuverlässig... beglaubigen... wirklich...

Sechs Hinweise darauf, wie ernst man dieses Thema nehmen sollte.

Merke: unterschreibe **niemals** einen Schlüssel, den du nicht vom Inhaber persönlich von Angesicht zu Angesicht ausgehändigt bekommst, und den du nicht persönlich gut kennst, oder per Ausweiskontrolle überprüft hast. Du bescheinigst mit deinem guten Namen, daß der Schlüssel dem gehört, dessen Namen er trägt, nicht mehr, aber auch nicht weniger.

Es kommt immer wieder vor, daß Leute ihren Schlüssel im Netz verbreiten und andere dazu auffordern, ihn zu unterschreiben. Es gab sogar den Fall, daß ein Sysop jedem, der seinen Key unterschreibt, Downloadfreiraum versprochen hat. Diese Leute disqualifizieren sich, indem sie deutlich kundtun, PGP nicht verstanden zu haben.

Ausnahme von der Regel

Es gibt einen Fall, in dem man auf ein Treffen von Angesicht zu Angesicht verzichten kann, und dieses "Treffen" durch den heißen Draht, das Telefon, ersetzen kann. Wenn man jemanden so gut kennt, daß man seine Stimme definitiv und mit 100%iger Sicherheit am Telefon erkennt, **und** ihn selbst zu Hause anruft (also sich nicht von einem Stimmenimitator anrufen läßt), dann kann man seinen Schlüssel per Voice mit ihm vergleichen. Ein mit "-kxa" extrahierter Schlüssel hat, wie dir ein Blick auf die diesem Kurs beiliegenden Schlüssel zeigt, nicht unter 300 Byte Länge, die noch dazu mehr als cryptisch zu lesen sind. Niemand wird Lust dazu haben, diese Zeichenfolge vorzulesen und ebensowenig wird der Partner am anderen Ende der Leitung dem gerne zuhören.

Rettung naht: der PGP-Key-Fingerprint!

Der einfache Befehl "PGP -kvc User-ID" gibt einen einmaligen Fingerabdruck eines Schlüssels aus, der z.B. für meinen Schlüssel so aussieht:

Fingerabdruck des Schlüssels:

BA B3 36 CE CC 1E F9 E5 43 12 3D 16 26 57 86 F5

Der Schlüsselinhhaber braucht diesen Fingerabdruck nur seinem Freund am Telefon vorzulesen (nicht umgekehrt! der Schlüsselinhhaber liest vor), und bestätigt ihm damit, daß er einen echten Schlüssel in den Händen hat, vorausgesetzt beide Fingerprints sind identisch.

Es relativ sinnlos, diesen Fingerprint in die Footer seiner Netzpostings zu schreiben (auch wenn es viele Leute tun), es macht aber durchaus Sinn diesen Fingerprint auf seiner Visitenkarte zu verewigen. Visitenkarten werden in der Regel persönlich

übergeben, und wer von mir persönlich eine Visitenkarte erhält (sicherheitshalber zusätzlich mit einem Blick auf den Personalausweis), kann sich meinen Schlüssel irgendwoher besorgen und mit dem Fingerprint auf meiner Visitenkarte vergleichen.

Vertrauen zu einem User

Um einen Schlüssel zu unterschreiben brauchst du den Inhaber nicht zu kennen, du brauchst nur sicher zu sein, daß der Inhaber des Schlüssels identisch mit dem Inhaber des Namens ist (oder umgekehrt? ;-), dann darfst du den Schlüssel unterschreiben.

Du kannst deinem PGP aber auch mitteilen, ob und wie weit du einem User vertraust. Damit gibst du an, wie du einen Menschen dahingehend einschätze, wie gut er PGP verstanden hat, und wie verantwortungsbewußt er mit seiner Unterschrift umgeht. Dazu gibt es 4 Vertrauensstufen, die angeben wie sehr du einem Menschen vertraust:

1=Ich weiß nicht
2=Nein
3=In der Regel
4=Ja, immer

Das hat **nichts** damit zu tun, ob du seinen Schlüssel für echt hältst, sondern nur damit, daß du ihn/sie für vertrauenswürdig hältst. Dieser Unterschied ist sehr wichtig!

Logischerweise kannst du einem User (und damit seinem Key in deinem Keyring) nicht dein Vertrauen bescheinigen, solange du nicht von der Echtheit seines Keys überzeugt bist, sonst kann es passieren, daß du einem Menschen dein Vertrauen aussprichst, wobei der Key über den Du das bestätigst gar nicht dem gehört, dem du vertraust.

Das könnte, ich gebe es zu, jetzt etwas verwirrend klingen, aber für dein PGP ist der Schlüssel eines Menschen das einzige, was es von ihm kennt, und wenn du einem Menschen dein Vertrauen aussprichst, dann tust du das über seine Key-ID.

Und wie bekunde ich mein Vertrauen?

Einen Schlüssel beglaubigst du genau so wie deinen eigenen:
"PGP -ks User-ID" ("ks": Key Sign).

Die Vertrauensparameter für einen User stellst du so ein, wie du deinen Eigenen Key bearbeitest: "PGP -ke User-ID" ("ke": Key Edit).

PGP erkennt anhand des Inhalts von SECRING.PGP, ob du deinen eigenen, oder einen fremden Schlüssel unterschreiben oder bearbeiten willst und bietet dir demnach unterschiedliche Möglichkeiten der Behandlung.

Zu Testzwecken kannst du jetzt einmal einen fremden Key unterschreiben, z.B. gibst du "PGP -ks yogi" ein. Du belügst dein PGP, indem du ihm bestätigst sicher zu sein (was du aber nicht bist), daß der Schlüssel wirklich dem gehört, dessen Namen er trägt. Siehst du dir den Schlüssel dann mit "PGP -kvv yogi" an, findest du deine Unterschrift an oberster Stelle unter der User-ID. Diese Unterschrift ist natürlich wertlos, du entfernst sie deshalb sofort wieder mit "PGP -krs yogi" ("krs": Key Remove Signature), wobei du nur die wertlose Unterschrift entfernst. Du kannst zwar auch alle anderen Sigs entfernen, das hat jedoch beim eventuellen Weiterverbreiten des Keys keinen Einfluß, da (wie schon einmal erwähnt) beim Aufnehmen eines Schlüssels in einen Schlüsselbund nur *hinzugekommene* User-IDs und Unterschriften berücksichtigt werden, aber keine entfernten.

Und was hat es mit dem Vertrauen zu einem User auf sich?

Stell dir vor, du kennst jemanden, von dem du weißt, daß er PGP voll verstanden hat, und der garantiert verantwortungsvoll mit seiner PGP-Unterschrift umgeht. Den wirst du in deinem Keyring als "voll vertrauenswürdig" einstufen. Erhältst du nun irgendwann einen Schlüssel über das Netz, der von dieser voll vertrauenswürdigen Person unterschrieben wurde, dann wird dein PGP das zur Kenntnis nehmen und sich nicht beschweren, wenn du diesen unbeglaubigten Schlüssel in deinen Keyring aufnimmst, oder wenn du ihn benutzt. PGP wird so tun, als hast du diesen Schlüssel selbst unterschrieben. Das bedeutet **nicht**, daß du deshalb diesen Schlüssel auch unterschreiben darfst, aber du darfst ihn ruhigen Gewissens benutzen.

PGP wird einen Schlüssel als echt ansehen, wenn ...

... er von mindestens **einer** Person unterschrieben ist, die du als voll vertrauenswürdig (Stufe 4) eingestuft hast,

... er von mindestens **zwei** Personen unterschrieben ist, die du als "in der Regel vertrauenswürdig" (Stufe 3) eingestuft hast.

Dieses Verhalten von PGP kannst du selbst einstellen, z.B. kannst du die minimale Anzahl benötigter "Stufe 4"-Unterschriften erhöhen, oder die Anzahl benötigter "Stufe 3"-Unterschriften ebenfalls auf 1 setzen, oder auch entsprechend erhöhen (dazu kommen wir noch).

Mit diesem Wissen kannst du ein wenig mit deinen gesammelten Schlüsseln spielen, sie unterschreiben, Unterschriften wieder löschen, Vertrauensparameter setzen usw. Aber bitte: verbreite **keinen** Schlüssel, den du nur so zum Test unterschrieben hast! Mit den Vertrauensparametern kannst du theoretisch machen, was du willst, niemand kann dir verbieten allen Menschen blind zu vertrauen (oder auch niemandem zu vertrauen), und die Vertrauensparameter betreffen nur dich und dein PGP, davon dringt nichts nach Aussen, deine Unterschriften sind aber weltweit zu sehen und du machst dich mit leichtfertig erteilten Unterschriften selbst zu einer Vertrauensperson der Stufe 2 (absolut nicht vertrauenswürdig).

Zur kurzen Zusammenfassung:

Du weißt jetzt ...

- ... wie du ein eigenes Schlüsselpaar erzeugst,
- ... wie du deinen public key extrahierst und verschickst,
- ... wie du fremde Schlüssel einliest,
- ... wie du deinen eigenen Schlüssel signierst
- ... wie (und vor allem wann!) du fremde Schlüssel signierst,
- ... wie du User nach Vertrauenswürdigkeit einstufst
- ... wie du Dateien verschlüsselst,
- ... wie du Dateien entschlüsselst.

Aber eins von dem, was ich schon ganz zu Anfang angekündigt habe, weißt du noch nicht:

Was hat es mit Unterschriften auf sich?

Bevor hier ein Mißverständnis aufkommt, ich rede jetzt nicht davon Schlüssel zu unterschreiben, das Thema wurde im vorigen Kapitel bis zum Erbrechen behandelt, nein es geht um eine Unterschrift unter einer Mail, oder unter einer Binärdatei.

Im täglichen Leben dienen Unterschriften dazu, daß eine Person halbwegs fälschungssicher etwas bekundet. In der Regel unterschreibt man ein Schriftstück, um zu dokumentieren, daß man vom Inhalt dieses Schriftstückes Kenntnis genommen hat.

Nichts anderes tut die PGP-Unterschrift (auch Signatur genannt), und das noch sicherer als im täglichen Leben, denn es ist weitaus schwieriger eine PGP-Unterschrift zu fälschen, als eine Handgeschriebene (es ist nahezu unmöglich).

Wenn du dich zurückerinnerst wie eine Verschlüsselung vor sich ging, dann fällt dir wieder ein:

verschlüsselt wird mit dem **public key**
entschlüsselt wird mit dem **secret key**

Zum Unterschreiben merkst du dir entsprechend:

unterschrieben wird mit dem **secret key**
die ***Unterschrift** wird **geprüft** mit dem **public key**

Das ist logisch, denn jeder kann deinen public key haben, darf damit aber nichts unterschreiben können, andererseits muß jeder mit deinem public key deine Unterschrift auf Echtheit prüfen können.

Da PGP nicht nur Dateien unterschreiben kann, sondern auch verschlüsseln (wir hörten bereits davon ;-)), gibt es eine Reihe von Möglichkeiten die dir zur Verfügung stehen, wie du diese Unterschriften an ein Dokument anhängen kannst, hier die drei Wichtigsten:

- Du kannst eine Datei verschlüsseln und dabei gleichzeitig unterschreiben, dann befinden sich verschlüsselte Datei und Unterschrift untrennbar in derselben Datei. Dies ist der Standardfall bei PMs, wobei nur der rechtmäßige Empfänger auch die Unterschrift prüfen kann.
- Du kannst eine Datei im Klartext verschicken und eine ASCII-Unterschrift anhängen. Die Unterschrift steht dann als ASCII-Block unter dem Text. Dies ist der Standardfall bei öffentlichen Nachrichten, bei denen man Wert darauf legt, daß sie unverfälscht ankommen.
- Du kannst eine Unterschrift abgekoppelt von einer Datei erstellen, so daß die Datei unverändert bleibt, und die Unterschrift getrennt von ihr verschickt werden kann. Dies ist gebräuchlich bei Verbreitung von Software, wobei man mit seiner Unterschrift ein Originalpaket garantieren möchte.

Wir konstruieren Beispiele:

Fall 1:

Du möchtest eine verschlüsselte PM an jemanden schicken, und sie gleichzeitig unterschreiben. Du suchst dir also wieder eine beliebige Datei auf deiner Festplatte aus, die du verschlüsseln willst, und erinnerst dich daran, daß "PGP -ea dateiname userid" eine verschlüsselte Datei namens dateiname.ASC erzeugte, die du nur dann entschlüsseln konntest, wenn du dich selbst als Empfänger angegeben hattest. Das gilt auch hier, füge dem Befehl ein "s" hinzu (für "Sign"), er lautet damit "PGP -sea dateiname User-ID". Damit wird die Datei unterschrieben, gepackt und verschlüsselt. Prüfen kannst du die Unterschrift beim Entschlüsseln der Datei ("PGP dateiname"), aber nur, wenn du sie auch an dich selbst verschlüsselt hast.

Fall 2:

Du möchtest eine Nachricht in ein öffentliches Brett schicken, und legst Wert darauf, daß diese Nachricht a) keinesfalls verändert irgendwo ankommt, und b) (für jedermann überprüfbar) von niemand anderem als von dir persönlich kommt. Die Nachricht soll also nicht verschlüsselt, sondern klar lesbar, aber dennoch unterschrieben sein. Der Befehl "PGP -sat dateiname" erledigt das für dich, du erhält eine Datei etwa mit folgendem Inhalt:

-----BEGIN PGP SIGNED MESSAGE-----

Hier steht der Nachrichtentext

-----BEGIN PGP SIGNATURE-----

Version: 2.6.2i

Und hier steht die Unterschrift.

-----END PGP SIGNATURE-----

Überprüft man die Datei mit "PGP dateiname", wird etwas wie die folgende Nachricht angezeigt:

Diese Datei trägt eine Unterschrift.
Zur Überprüfung wird der öffentliche Schlüssel benötigt. .
BESTÄTIGTE Unterschrift von "Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>".
Unterschrift erzeugt: 1995/08/29 13:45 GMT

(Ab Version 2.6.3i erhält die letzte Zeile den Zusatz:
"... mit 512-Bit Schlüssel 0x0D0AC4D9")

Wird aber auch nur ein einziges Bit in der Nachricht geändert, erhält man die folgende Mitteilung:

Diese Datei trägt eine Unterschrift. Zur Überprüfung wird der öffentliche Schlüssel benötigt. . WARNUNG: Die Unterschrift stimmt nicht mit dem Datei-Inhalt überein!

FEHLERHAFTE Unterschrift von "Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>".
Unterschrift erzeugt: 1995/08/29 13:45 GMT

(Zur letzten Zeile bei Version 2.6.3i: siehe oben)

Wurde in einer Klartextnachricht mit Unterschrift diese Unterschrift geändert erhält man die Meldung:

FEHLER: Falsche Prüfsumme der Versandhülle.

Du siehst also, daß eine Unterschrift **immer** zum Inhalt der Datei passen muß, die sie unterschreibt. Jede Manipulation am Inhalt der Nachricht oder an der Unterschrift selbst, **fällt sofort auf** wenn du die Unterschrift prüfst.

Fall 3:

Du hast ein Programm geschrieben, kannst dafür garantieren, daß es 100%ig virenfrei ist, und willst es verbreiten. Hier würde es keinen Sinn machen, die EXE-Datei zu verschlüsseln und dabei zu unterschreiben, wie im Fall 1, sie im Klartext mit angehängter Unterschrift zu verbreiten wäre noch unsinniger. In beiden Fällen würde dein Programm nicht mehr laufen, bzw. wenn du ein Archiv unterschreibst, ließe es sich nicht mehr entpacken. Deshalb gibt es Möglichkeit Nr.3: vom Dokument abgekoppelte Unterschriften.

"PGP -sb dateiname" erzeugt die ASCII-Unterschriftsdatei "dateiname.SIG". Nehmen wir an, du willst die Datei PROG.EXE unterschreiben, dann gibst du "PGP -sb prog.exe" an, wobei eine Datei namens PROG.SIG entsteht. Diese Datei kannst du zusammen mit PROG.EXE in ein Archiv packen und verbreiten. Jeder Empfänger dieses Pakets kann nun, sofern er deinen public key besitzt, diese Unterschrift prüfen, der Befehl dazu lautet: "PGP prog.sig prog.exe".

Hier gilt, was du oben über Unterschriften und das dazugehörige Dokument gelernt hast: jeder Patch oder Virenbefall fällt sofort auf, womit du beweisen kannst, daß die Datei verändert worden sein muß, **nachdem** du sie unterschrieben hast.

In diesem Beispiel wird mit "-sb" eine Binärdatei mit der Endung SIG erstellt. Wenn du gerne eine lesbare Signatur haben möchtest, kannst du auch den Befehl "PGP -sab dateiname" verwenden, da die Unterschrift aber meist ohnehin mit in ein Archiv gepackt wird, ist gegen die Erstellung einer binären Unterschrift nichts einzuwenden. *.ASC-Dateien können Schlüssel, verschlüsselte Nachrichten oder Signaturen enthalten, bei *.SIG-Dateien ist der Inhalt schon am Namen ersichtlich.

Übrigens: diesem Kurs ist im Originalarchiv eine abgekoppelte Unterschrift beigelegt, wenn dein PGP diese Unterschrift bestätigt, dann kannst Du sicher sein, daß ich jeden Buchstaben in diesem Text genau so geschrieben habe, wie du ihn hier vorliegen hast.

Du machst deinen Schlüssel ungültig

Es wurde bereits im ersten Teil des Kurses angesprochen, es kann Fälle geben, die es erfordern daß man seinen Schlüssel widerruft. Das könnte passieren, wenn irgendjemand in den Besitz deines secret keys gekommen ist, schlimmstenfalls sogar mit deiner Passphrase.

Um in einem solchen Fall wenigstens kundzutun, daß der Schlüssel nicht mehr gültig ist, muß man seinen Schlüssel zurückrufen. Dazu erstellt man sich ein sogenanntes "key revocation certificate" (eine Schlüsselerückrufsurkunde). Dazu dient der Befehl:

"PGP -kd User-ID",

damit würde aus meinem Schlüssel der jetzt so aussieht...

Typ	Bits	ID	Datum	Benutzer
öff	512/0D0AC4D9	1994/04/03	Juergen Poetzsch	<YOGI@NEWSWIRE.gun.de>
unt	0D0AC4D9		Juergen Poetzsch	<YOGI@NEWSWIRE.gun.de>

... ein Rückrufszeugertifikat, das so aussieht:

Typ	Bits	ID	Datum	Benutzer
zur	512/0D0AC4D9	1994/04/03	*** ZURÜCKGEZOGEN ***	Juergen Poetzsch <YOGI@NEWSWIRE.gun.de>

Das "zur" soll "zurückgezogen" bedeuten, in der englischen Version erscheint hier ein "rev" für "revokation".

Dieses Rückrufszeugertifikat funktioniert wie ein normaler Schlüssel! Es wird so aus dem Keyring extrahiert (-kxa), so weit wie möglich verbreitet, und so in andere Keyrings aufgenommen wie ein normaler Schlüssel. Der betreffende Schlüssel wird dann weltweit als ungültig betrachtet und jeder Anwender wird darauf hingewiesen, wenn er ihn benutzen will. Für Rückrufszeugertifikate gilt das gleiche was schon für User-IDs und Unterschriften gesagt wurde, sie lassen sich nicht rückgängig machen. Du kannst zwar einen zurückgerufenen Schlüssel aus Deinem eigenen Keyring entfernen und den Originalschlüssel wieder aufnehmen, aber ein verbreitetes Rückrufszeugertifikat läßt sich durch Aufnahme des ursprünglichen Schlüssels nicht ersetzen, der Schlüssel bleibt dann zurückgerufen.

Worst case

Der schlimmste anzunehmende Fall, sozusagen der Super-GAU, tritt ein, wenn dein Schlüsselpaar in falsche Hände gerät, und du keine Sicherheitskopie davon besitzt. Dann kannst du nur noch einen öffentlichen Aufruf an alle PGP-User starten, damit die

deinen Key quasi "von Hand" als ungültig markieren. Auch dazu dient der Befehl "PGP -kd userid", damit kann ein beliebiger Key (außer deinem eigenen) gesperrt und auch wieder entsperrt werden. Das hat den gleichen Effekt wie ein Rückruf, nur daß dieser Befehl lediglich Einfluss auf den eigenen public Keyring hat (im Gegensatz zum Rückrufs-zertifikat).

Kleinvieh macht auch Mist

unter "Kleinvieh" ist zu verstehen, daß es eine Reihe von Kleinigkeiten an PGP einzustellen gibt, die nicht unbedingt wichtig sind, damit PGP richtig arbeitet, aber sie können einem das Leben schon unnötig schwer machen (eben "Mist" machen ;-)).

Dir bisher bereits bekannte Einstellmöglichkeiten sind die Variablen PATH (zu erweitern um den Pfad in dem sich PGP.EXE befindet), PGPPATH (das Verzeichnis in dem PGP seine Keyrings findet), und TZ (eine Zeitzoneangabe). Dazu noch etwas Vertiefendes bzw. weitere Konfigurationsmöglichkeiten.

TZ

Die Variable TZ wurde bisher nicht erklärt, das soll hier nachgeholt werden. TZ beschreibt eine Zeitzone. Unsere Erde ist in Ost/West-Richtung in 24 Zeitzonen aufgeteilt. Nullpunkt dieser Zeitzonen ist Greenwich, ein Ort in der Nähe von London, auf den der 0. Längengrad willkürlich festgelegt wurde. Wenn es in Greenwich 0:00 Uhr ist, dann ist es in Deutschland im Winter 1:00 Uhr, im Sommer aber 2:00 Uhr. Wenn man international Nachrichten austauschen möchte, dann sollte man sich einig werden, welche Erstellzeit man im Kopf seiner Nachrichten oder in seinen PGP-verschlüsselten oder unterschriebenen Nachrichten angibt. Deshalb einigt man sich auf Greenwich als Bezugspunkt. Schreibe ich z.B. hier in Deutschland eine Nachricht um 16:13 MESZ (mitteleuropäische Sommerzeit), dann erhält meine Nachricht die Erstellzeit 14:13 GMT (Greenwich mean time). Schreibt jemand in New York seine Nachricht um 08:13 EST (eastern standard time), dann erhält seine Nachricht ebenfalls die Erstellzeit von 14:13 GMT. Damit kann man problemlos die Erstellzeiten von Nachrichten miteinander vergleichen und bei Kenntnis der Zeitzone auch ebenso problemlos auf die Ortszeit des Erstellers, oder auf seine eigene Ortszeit zurückrechnen.

Der Inhalt der Variablen TZ besteht aus drei Teilen:

1. der Name der Zeitzone (3 Buchstaben)
2. ein Korrekturwert bezüglich der Greenwich-Zeit
3. optional eine Sommerzeitangabe (3 Buchstaben).

Der Name der Zeitzone und die Sommerzeitangabe ist dem Anwender überlassen, solange er sich an die Konvention hält, die besagt, daß beide Angaben aus drei Buchstaben bestehen müssen. Richtig für Mitteleuropa ist "CET-1DST", im Klartext "central european time" minus eine Stunde (zu Greenwich) und "daylight saving time", was schlicht bedeutet, daß wir hier im Sommer auf Sommerzeit umstellen. Man könnte genauso gut "ABC-1DEF" oder "CDU-1SPD" angeben, der Effekt wäre derselbe, nämlich daß im Winter (vom 1. Sonntag im Oktober bis zum 1. Sonntag im April) eine Stunde von der Ortszeit abgezogen wird, im Sommer (1. Sonntag im April bis 1. Sonntag im Oktober) aber 2 Stunden abgezogen werden.

Bei einer Angabe wie "CET-1" (central european time -1 Stunde) wird keine Sommerzeit berechnet, da die Angabe (die drei Buchstaben) für die Sommerzeit fehlt.

Ob deine TZ-Variable richtig eingestellt ist, kannst du prüfen, indem du PGP ohne Parameter aufrufst, die Zeile "Aktuelles Datum und Uhrzeit" muß das heutige Datum anzeigen, und die GMT-Angabe muß der aktuellen Uhrzeit um eine Stunde nachgehen (im Sommer um zwei Stunden!).

PGPPASS

Nicht zu verwechseln mit PGPPATH!!

Während PGPPATH den Pfad angibt in dem PGP "zu Hause" ist, kannst du mit "SET PGPPASS=dies ist meine pass phrase" dein Passwort schon in der AUTOEXEC.BAT setzen, dann fragt PGP z.B. beim automatischen Entschlüsseln nicht mehr nach einem Passwort. Doch **Vorsicht:** mit diesem Feature kann jeder der Zugriff auf deinen Rechner hat, nicht nur alle deine verschlüsselte Nachrichten lesen, er kann auch mit einem einfachen SET-Befehl dein Passwort im klarsten Klartext auf den Bildschirm zaubern. Dieses PGP-Featurer ist ein ausgesprochener Risikofaktor und sollte **nur dann** verwendet werden, wenn **kein Fremder** Zugriff auf deinen Rechner hat.

CONFIG.TXT

Ein Bestandteil des PGP-Paketes ist eine Datei namens CONFIG.TXT (beim IBM-PC, beim AMIGA heißt sie PGP.CONFIG, das hier gesagte gilt für beide Dateien (soweit ich informiert bin)).

Wie die Dateinamenserweiterung schon sagt, kann man diese Datei mit einem herkömmlichen Texteditor ansehen und bearbeiten. Du findest dort eine Reihe von Voreinstellungen, die auch verständlich kommentiert sind, so daß sich eine Erklärung hier eigentlich erübrigt. Man sollte sich aber wirklich die Mühe machen, diese Datei einmal durchzulesen, dafür ist sie extra so ausführlich kommentiert.

Erwähnt seien hier nur einige der wichtigsten Voreinstellungen:

Completes_Needed und **Marginals_Needed**, das ist die bereits angesprochenen Mindestanzahl "voll vertrauenswürdiger" und "normal vertrauenswürdiger" Unterschriften, ab der PGP einen fremden Schlüssel als echt und unbedenklich betrachten soll.

EncryptToSelf ist eine Variable, mit der du erzwingen kannst, daß **jede** Nachricht, die du an jemand anderen verschlüsselst, auch **gleichzeitig** an dich selbst verschlüsselt wird. Damit hast du die Möglichkeit, Nachrichten die z.B. gebounced (zum Absender zurückgeschickt) werden, selbst wieder zu entschlüsseln. Allerdings dauert die Verschlüsselung deiner Nachrichten damit auch länger.

MyName ist wichtig, wenn man mehrere eigene Schlüssel verwendet. Hier gibt MyName an, welcher der Schlüssel standardmäßig zum Unterschreiben verwendet werden soll.

Damit ist eigentlich alles über PGP gesagt, was man darüber wissen muß, um damit arbeiten zu können. Wer auf den Geschmack gekommen ist, und mehr wissen will, dem sei die (im Vorwort bereits angesprochene) Originaldokumentation zu PGP empfohlen, die es auch in einer deutschen Übersetzung gibt.

Angriffe

Ein "Angriff" im Cryptologie-Jargon bezeichnet jede Form eines Versuchs, Daten, die nicht für die Augen des "Angreifers" bestimmt sind, lesbar zu machen und zu lesen. Im Gegensatz zu einem Angriff wie man ihn sich im täglichen Leben vorstellt, nimmt hier der Angegriffene von diesem Angriff überhaupt nichts wahr, was aber gerade das Gefährliche an einem solchen Angriff ist.

Wie könnte ein solcher Angriff vor sich gehen?

Theoretisch könnte ein Angriff von jedem erfolgen, über dessen System deine PMs geroutet werden, also jeder Sysop. Nehmen wir als Beispiel eine fiktive Mailbox und

nennen sie "Disneyland" In der Disneyland gibt es einen User, wir nennen ihn Donald@disneyland, und eine Userin, die nennen wir Daisy@disneyland. Die beiden tauschen regelmäßig miteinander PMs aus. In dieser fiktiven Mailbox gibt es auch noch den fiesen Sysop namens Carlo (Walt Disney möge mir nachsehen, daß ich mir Namen seiner Figuren geliehen habe). Carlo macht sich einen Spaß daraus, den PM-Verkehr seiner beiden User "abzuhören", dafür hat er sich extra ein Programm installiert, das eine Kopie der PMs der beiden, in sein eigenes Postfach legt.

Irgendwann haben die PMs zwischen Donald und Daisy ein so vertrautes Niveau erreicht, daß die beiden beschließen, PGP zu installieren, jeweils ein Schlüsselpaar zu erstellen, und diese Schlüssel per PM auszutauschen. Carlo bekommt das natürlich sofort mit, und reagiert prompt, indem er selbst auch zwei Schlüsselpaare erstellt, einen mit Donalds Namen, und einen mit Daisies. Zusätzlich verbessert er sein Watchdog-Programm so, daß es die PMs nicht mehr kopiert, sondern komplett abfängt. Die public keys die sich die beiden zuschicken, tauscht er durch seine gefälschten Schlüssel aus und behält die Originalschlüssel in seinem Keyring. Weder Donald, noch Daisy erfahren etwas von dieser Tauschaktion, denn sie erhalten beide einen Schlüssel, der den Namen des anderen trägt, wie verabredet.

Da haben wir das Problem: ein Schlüssel, den man über das Netz bekommt, sagt überhaupt nichts, er kann von jedem kommen und muß nicht zwangsläufig dem gehören, dem er zu gehören scheint. Naiv wie sie sind, unterschreiben Donald und Daisy aber gegenseitig ihre Schlüssel (die falschen), und schicken sie sich wieder zu. Kein Problem für Carlo, er besitzt ja seine Fake-Keys, einschließlich der dazugehörigen secret keys. Er unterschreibt einfach den für Donald gefälschten Key mit Daisies gefälschtem Key und schickt ihn an Donald, Daisies gefälschten Key unterschreibt er mit Donalds gefälschtem Key und schickt ihn an Daisy (klingt kompliziert, ist es auch, aber wenn man etwas drüber nachdenkt, wird es klarer).

Zur Verdeutlichung:

Donalds echter Key:

```
pub 512/11111111 1994/04/03 Donald Duck <donald@disneyland>
```

Daisies echter Key:

```
pub 512/22222222 1994/04/03 Daisy Duck <daysi@disneyland>
```

Carlos gefälschte Keys:

```
pub 512/33333333 1994/04/03 Donald Duck <donald@disneyland>
```

```
pub 512/44444444 1994/04/03 Daisy Duck <daysi@disneyland>
```

Die Keys unterscheiden sich nur anhand der Key-ID, die User-IDs sind jeweils gleich. Donald schickt also seinen Key (ID: 11...) an Daysy, Carlo fängt ihn ab und ersetzt ihn durch Key 33..., gleiches gilt für Daisies echten Key 22..., der wird durch Key 44... ersetzt.

Wenn Daisy den Key 33... unterschreibt, von dem sie glaubt er gehöre Donald, fängt Carlo diesen Key ab und schickt Donald stattdessen Key 11..., den er mit Key 44... unterschrieben hat, von dem ja Donald glaubt, er gehöre Daisy.

Wenn Donald den Key 44... unterschreibt, von dem er glaubt er gehöre Daisy, fängt Carlo diesen Key ab und schickt Daisy stattdessen Key 22..., den er mit Key 33... unterschrieben hat, von dem ja Daisy glaubt, er gehöre Donald.

Und wieder sind Donald und Daysy glücklich und zufrieden, denn sie haben jetzt sogar eine beglaubigte Unterschrift, die ihren Schlüssel zierte. Sie gehen nun ruhigen Gewissens daran, ihre PMs verschlüsselt auszutauschen. Kein Problem für Carlos Software, sie empfängt z.B. eine PM von Donald (die er ja mit Daisies falschem Key verschlüsselt hat), entschlüsselt sie, kopiert sie in Carlos Postfach, verschlüsselt sie wieder mit Daisies echtem Key und schickt sie weiter an Daisy, die nun wiederum

eine PM erhält, die so verschlüsselt ist, daß sie sie mit ihrem secret key entschlüsseln kann, die Welt ist also (für sie) in bester Ordnung. Die Antwort von Daisy an Donald erleidet das gleiche Schicksal, sie wird ebenfalls bei Carlo entschlüsselt, kopiert, mit Donalds echtem Schlüssel verschlüsselt und an Donald weitergeleitet.

Carlo hat seine Angriffs-Software mittlerweile so verfeinert, daß sie sogar die Nachrichten der beiden auf Fingerprints scannt, und diese gegebenenfalls auch austauscht! Und wenn sie nicht gestorben sind, dann tauschen sie noch heute verschlüsselte Nachrichten aus, und sind sicher, daß niemand ihre PMS lesen kann.

Eine hübsche Geschichte, nicht wahr? Was, du meinst, das sei etwas weit hergeholt? Das ist es keineswegs. Jeder halbwegs begabte Programmierer mit knapp überdurchschnittlichen PGP-Kenntnissen ist in der Lage, ein Scannerprogramm zu schreiben, daß selbsttätig einen PGP-Key in PMs erkennt, feststellt, ob es ihn schon gefaked hat, ihn gegebenenfalls austauscht und in der PM ersetzt. Dasselbe Programm ist auch problemlos in der Lage verschlüsselte Mails zu erkennen, für den Sysop zu entschlüsseln, mit dem echten Schlüssel des Originalempfängers wieder zu verschlüsseln und an ihn weiterzuleiten. Selbst Fingerprints in den verschlüsselten Mails zu fälschen und dem Fake-Schlüssel anzupassen ist eine der leichtesten Programmierübungen.

Damit jetzt niemand denkt PGP sei unsicher, PGP ist das wahrscheinlich sicherste Verschlüsselungsprogramm der Welt, wenn man es richtig bedient und Verantwortungsbewußt damit umgeht. Die einzigen, die in meiner fiktiven Geschichte fehlerfrei gearbeitet haben, waren PGP, Carlo und seine Angriffssoftware, die Fehler lagen bei Donald und Daisy. Ein simpler Telefonanruf hätte gereicht, um Carlo durch Vergleich der Fingerprints auf die Schliche zu kommen und zumindest festzustellen, daß man nicht den Key dessen besitzt, den man an der Strippe hat. Der Telefonanruf bietet zwar auch keine letztendliche Sicherheit, denn auch da gibt es Angriffsmöglichkeiten, aber dazu müßte der Angreifer schon zwei Datenwege manipulieren.

Der gravierendste Fehler der beiden war aber, daß sie einen Schlüssel unterschrieben haben, von dem sie nur **geglaubt haben** er stamme von dem, dessen Namen er trägt, ohne es es aber wirklich **zu wissen**.

Was kann der Angreifer nicht?

Er kann keine Unterschriften Dritter fälschen. Nehmen wir an, Daisy und Donald haben einen gemeinsamen Bekannten: Dagobert. Nehmen wir weiter an, Dagobert hat die neu erstellten Keys von Donald und Daisy unterschrieben, bevor die beiden ihre Schlüssel tauschen. Dann könnte sich Carlo zwar auch noch einen Schlüssel mit Dagoberts Namen erstellen und damit seine Fakes unterschreiben, aber das PGP von Daisy und Dagobert würde sofort feststellen, daß diese Unterschrift nicht mit dem Schlüssel übereinstimmt, den es selbst von Dagobert besitzt.

Jede Unterschrift macht einen Schlüssel sicherer gegen Manipulationen und mit Unterschriften von Usern denen man vertraut, kann man eine Vertrauenskette (Cryptologie-Jargon: trusted chain) aufbauen, die es uns erlaubt einen Schlüssel zu benutzen, den wir nur über das Netz bekommen haben, und nicht vom Inhaber persönlich.

Happy end

Tja, damit sind wir am Ende unseres PGP-Einführungskurses. Was hier zur Sprache kam, war nur das Wesentliche, was man für die Benutzung von PGP wissen muß. Vieles wurde nicht gesagt, z.B. der interne Aufbau geheimer und öffentlicher Schlüssel, wie der in PGP integrierte Packalgorithmus funktioniert und vieles mehr. Das war aber auch nicht

Ziel dieses Kurses, diese Informationen erhält der Interessierte auch aus vielen bereits existierenden Texten von FTP-Servern oder aus dem Brett /Z-NETZ/ALT/PGP/ALLGEMEIN.