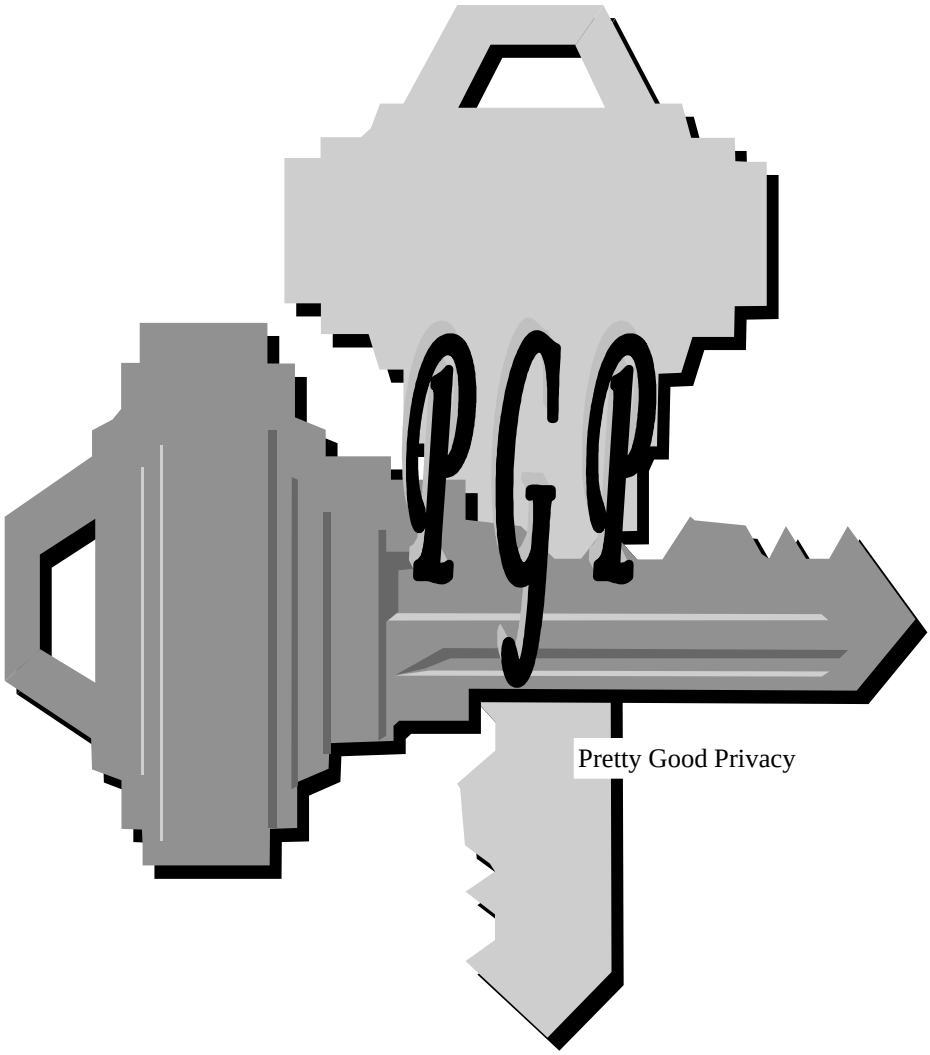
FernUniversität
Gesamthochschule in Hagen
Universitätsrechenzentrum

Manuela Jürgens
Abt. Wissenschaftliche Anwendungen



Briefumschläge für Ihre Emails

A/001/9608

1 Inhaltsverzeichnis

2 Einige Vorworte

Die Problematik des Verschlüsselns ist sicherlich fast genauso alt, wie das Benutzen einer Schrift. Dabei bedeutet Verschlüsseln mathematisch gesehen eigentlich nichts anderes, als eine neue Darstellung von Daten und Informationen. In modernen Zeiten wird man zum Chiffrieren zwar keine Stäbe, Schablonen oder Zahnräder benutzen, dafür aber computergestützte Verfahren, die inzwischen ein sehr hohes Maß an Sicherheit erreicht haben.

Was ist PGP?

PGP (Pretty Good Privacy) wurde von *Philip Zimmermann*[1] entwickelt, und ist momentan in der *Version 2.6.3i* im Rechenzentrum der FernUniversität verfügbar. Das Besondere ist, daß es zum Ver- und Entschlüsseln von Daten *zwei verschiedene* Schlüssel benutzt, die, bildlich gesehen, in ein und das gleiche Schloß passen. Es handelt sich hierbei um das sogenannte *asymmetrische Codierungsverfahren* mit dem Namen RSA, benannt nach den Entwicklern Rivest, Shamir und Adleman.

PGP gestattet gleichzeitig das *digitale Unterschreiben* von Emails, so daß diese auf Ihrem Weg zum Empfänger von niemandem unbemerkt verfälscht werden können. Zusätzlich ist damit sichergestellt, daß der Autor der Mail garantiert der Unterschreibende ist.

Wo finde ich PGP?

Auf dem *FTP-Server* des Rechenzentrums der FernUniversität finden Sie momentan sowohl die PGP-Software in der Version 2.6.3i, als auch diverses Zubehör zur Benutzung unter DOS und Windows.

Literatur

Zusammen mit der Software erhalten Sie immer zwei *User's Guides***,** geschrieben von Philip Zimmermann, die die grundlegende und weiterführende PGP-Benutzung erläutern. Eine deutsche Übersetzung wurde von der FoeBuD e.V. in Bielefeld herausgegeben und ist im World Wide Web unter der Adresse `www.zerberus.de/~christopher/pgp/` zu finden.

Die drei folgenden Bücher erleichtern Ihnen den Umgang mit PGP:

Das Buch *PGP Pretty Good Privacy* von *Simson Garfinkel* beschreibt nicht nur detailliert die PGP-Verwendung und die Installation auf verschiedenen Betriebssystemen, sondern erläutert auch noch ausführlich die Grundlagen und Historie der Kryptographie. (Erschienen: 1995 by O´Reilly & Associates, Inc; ISBN 1-56592-098-8).

Das Buch *Protect your Privacy* von *William Stallings* ist eine sehr ausführliche und übersichtliche Lektüre zu PGP und erleichtert den Lernprozeß Schritt für Schritt mit Hilfe von anschaulichen Graphiken. (Erschienen: 1995 by Prentice Hall PTR; ISBN 0-13-185596-4).

Eher grundsätzlich behandelt das Buch *Email Security - How to keep your Electronic Messages privat* von *Bruce Schneier* das Thema Verschlüsselung von Emails. Gleichzeitig beinhaltet es nicht nur detaillierte Informationen zu *PGP*, sondern auch zu *PEM (Privacy Enhanced Mail). (*Erschienen: 1995 by John Wiley & Sons, Inc.; ISBN 0-471-05318-X).

[1] Philip Zimmermann ist langjähriger, erfahrener Softwareentwickler. Sein Schwerpunkt liegt unter anderem im Bereich der Kryptographie und Authentisierung von Nachrichten.

3 PGP im Überblick

Zum besseren Verständnis des Arbeitens mit PGP, insbesondere der Verwendung von asymmetrischen Schlüsseln, möchte das vorliegende Kapitel Sie zunächst mit einigen Grundsätzen vertraut machen, ohne jedoch schon auf die eigentlichen PGP-Kommandos einzugehen.

Erst das anschließende Kapitel besteht dann im Wesentlichen aus einer beispielhaften Vorstellung der PGP-Befehle in der Reihenfolge, in der ein PGP-Anfänger sie in der Regel verwenden wird.

Die weiteren Kapitel dieser Broschüre werden dann auf einige Detailfragen eingehen, deren Studium ein Anfänger getrost zu einem späteren Zeitpunkt nachholen kann. Die letzten Kapitel dieses Dokumentes werden Sie abschließend noch über die Installations- und Konfigurationsmöglichkeiten, sowie über die Einbindung in diverse Email-Systeme informieren.

3.1 Was ist PGP?

PGP ist ein Programm zum Verschlüsseln und Entschlüsseln von Daten. Neben einer relativ *einfachen Benutzung*, seiner *Schnelligkeit* und einer guten *Schlüsselverwaltung* zeichnet es sich durch die Möglichkeit *digitaler Unterschriften* und einer automatischen *Datenkompression* aus.

3.2 Wozu benötige ich PGP?

Vertraulichkeit

Konzipiert wurde PGP in erster Linie für die Verschlüsselung von Mails. Sollten Sie sich fragen, weshalb Sie Ihre Mail verschlüsseln sollen, so fragen Sie sich doch auch einmal, warum Sie Ihre dienstliche und private Korrespondenz bei Benutzung der gelben Post nicht einfach per Postkarte verschikken, sondern zumindest einen zugeklebten Umschlag benutzen. Verschicken Sie Mails, so können im Prinzip an jedem Knoten, den Ihre Daten auf dem Weg zum Empfänger passieren, Mitleser lauern. Selbst wenn Sie nichts zu verbergen haben: Mails, die Sie verschicken, gehen niemanden etwas an.

Integrität

Um die Frage der Vertraulichkeit noch zu verschärfen: Wie können Sie sicher sein, daß Ihre Mails unterwegs nicht abgefangen und verfälscht werden, bevor Sie den Adressaten erreichen? PGP ermöglicht es Ihnen, selbst nicht verschlüsselte Mails mit Hilfe elektronischer Signaturen vor unbemerkter Verfälschung zu bewahren.

Authentizität

Und schließlich können Sie mit Ihrer PGP-Signatur auch noch allen Email-Partnern bestätigen, daß die Nachricht auch tatsächlich von Ihnen stammt und nicht möglicherweise von jemand anderem unter Ihrem Namen verschickt wurde.

Alles in allem: drei sehr gute Gründe PGP zu benutzen.

3.3 Wie arbeitet PGP?

konventionelle Verschlüsselung

Die Arbeitsweise von PGP unterscheidet sich deutlich von konventionellen Verschlüsselungsprogrammen: diese arbeiten nämlich in der Regel mit *einem* Schlüssel, der sowohl zum Ver- als auch zum Entschlüsseln benutzt wird. Das ist sicherlich kein Nachteil, wenn Sie zum Beispiel wichtige Dateien auf Ihrem PC verschlüsselt ablegen möchten. Sobald Sie konventionell verschlüsselte Dateien aber per Mail verschicken möchten, muß dieser Schlüssel zumindest den Empfängern bekannt sein. Der Kreis derer, die Ihren Schlüssel kennen, kann somit beträchtlich groß werden, und die Geheimhaltung Ihrer Daten ist nicht mehr gewährleistet. Und auf welchem Wege erhalten die Kommunikationspartner überhaupt Ihren Schlüssel? Wenn Sie ihn per Email übermitteln, unterliegt er den gleichen zuvor beschriebenen Gefahren, die jede Mail treffen können!

öffentliche und private Schlüssel

Die PGP Lösung: Sie arbeiten mit zwei Schlüsseln, sogenannten *öffentlichen* und *privaten*[2] Schlüsseln. Das trickreiche Konzept dieses Schlüsselpaares ist wichtig zum Verständnis der Arbeit mit PGP und wird deshalb hier kurz vorgestellt:

Jeder PGP-Benutzer verfügt über ein Schlüsselpaar, das aus einem öffentlichen und einem privaten Schlüssel besteht. Daten, die mit dem öffentlichen Schlüssel kodiert werden, können nur mit dem privaten Schlüssel dekodiert werden. Andererseits können Unterschriften nur mit dem privaten Schlüssel erstellt werden und mit dem zugehörigen öffentlichen Schlüssel auf ihre Echtheit überprüft werden.

Der *öffentliche* Schlüssel kann allen Mail-Partnern mitgeteilt werden, die diesen, bildlich gesprochen, an ihren *öffentlichen Schlüsselbund* hängen. Der *private* Schlüssel befindet sich entsprechend an eines jeden *privaten Schlüsselbund*, wird aber *niemals* jemandem ausgehändigt. Um diesen privaten Schlüssel mit dem größtmöglichen Schutz zu versehen, ist er zusätzlich noch mit einem Passwort versehen. Jedesmal, wenn Sie diesen Schlüssel benutzen möchten, fordert PGP Sie zur Eingabe des zugehörigen Passwortes auf. Sollte also tatsächlich einmal jemand in den Besitz Ihres geheimen Schlüssels kommen, kann er ihn ohne Kenntnis Ihres Passwortes nicht verwenden.

Möchten Sie nun eine verschlüsselte Mail verschicken, so benötigen Sie den *öffentlichen Schlüssel des Empfängers,* verschlüsseln damit die Daten und schicken diese auf die Reise. Der Empfänger — und nur dieser — kann jetzt mit seinem *geheimen Schlüssel* die Mail wieder entschlüsseln. Nicht einmal Sie selbst als Absender dieser Mail könnten Sie wieder entschlüsseln.

Um es noch einmal deutlich zu machen: Es gibt also im Prinzip *ein* Schloß in das *zwei verschiedene* Schlüssel passen. Der eine (öffentliche) Schlüssel schließt das Schloß ab und nur der zugehörige andere (geheime) Schlüssel kann das Schloß wieder aufschließen.

[2] Private Schlüssel werden im folgenden häufig auch „geheime“ Schlüssel genannt.

Ein Beispiel:

> Anna möchte Bob eine Mail schicken.
>
> Anna läßt sich den öffentlichen Schlüssel von Bob schicken. Das ist unkritisch, denn jeder darf den öffentlichen Schlüssel sehen[3].
>
> Anna verschlüsselt die Mail mit Bobs öffentlichen Schlüssel und schickt Sie ab.
>
> Bob empfängt diese Mail und entschlüsselt sie mit seinem privaten Schlüssel, den nur er kennt.

Ein einfaches Prinzip, das zudem noch die Suche nach den passenden Schlüsseln, die sich im Laufe der Zeit an dem ,,öffentlichen Schlüsselbund“ angesammelt haben, automatisch vornimmt.

3.4 Was sind digitale Unterschriften und wie werden sie benutzt?

Eine digitale Unterschrift soll dem Empfänger garantieren, daß die Mail auch wirklich von dem Absender stammt, von dem Sie zu sein scheint und daß der Inhalt der Nachricht nicht verändert wurde.

Für eine digitale Signatur benötigen Sie Ihren *geheimen* Schlüssel, während der Empfänger Ihren *öffentlichen* Schlüssel benutzt, um Ihre Signatur zu überprüfen. Jeder, der Ihren öffentlichen Schlüssel besitzt, kann auch automatisch Ihre Signatur bestätigen.

Ein Beispiel:

> Anna möchte Bob eine Mail schicken.
>
> Um zu gewährleisten, daß sie unterwegs nicht verfälscht wird, versieht sie die Mail mit einer digitalen Unterschrift und benutzt dazu ihren geheimen Schlüssel.
>
> Bob, der die Mail erhält, benutzt nun wiederum Annas öffentlichen Schlüssel und bekommt damit ihre Unterschrift bestätigt, sofern die Mail nicht verändert wurde.

[3] Zum Problem der Schlüsselfälschung finden Sie einige Anmerkungen im Kapitel 6 auf Seite 29.

3.5 Wie werden verschlüsselte Daten mit digitalen Unterschriften kombiniert?

Schauen wir uns direkt ein Beispiel an:

> Anna möchte Bob eine verschlüsselte Mail schicken und sie sicherheitshalber unterschreiben.
>
> Dazu erstellt sie zunächst den Inhalt der Mail und kann diesen nun in einem Arbeitsschritt verschlüsseln und signieren. PGP benutzt zur digitalen Unterschrift automatisch den privaten Schlüssel von Anna und anschließend, ebenfalls automatisch, Bobs öffentlichen Schlüssel zur Kodierung der Nachricht.
>
> Dieses Paket wird auf die Reise geschickt.
>
> Auch Bob benötigt zur Dekodierung der Nachricht und zur Bestätigung der Unterschrift nur einen Arbeitsgang. PGP entschlüsselt automatisch den Mailtext mit Bobs privaten Schlüssel und um auch noch die Signatur bestätigen zu können, wird diese mit Annas öffentlichem Schlüssel überprüft.

Die Sicherheit der Mail ist also gewährleistet:

- sie kann nur von Bob gelesen werden (Vertraulichkeit),
- sie ist garantiert von Anna geschrieben worden (Authentizität) und
- sie wurde auf dem Weg zu Bob nicht verfälscht (Integrität).

Um es noch einmal zusammenzufassen:

Sie benötigen *Ihren geheimen Schlüssel*

- um Mails, die für Sie kodiert wurden zu entschlüsseln und
- um digitale Unterschriften zu leisten

Sie benötigen *Ihren öffentlichen Schlüssel*

- um ihn an Kommunikationspartner zu verteilen

Sie benötigen die *öffentlichen Schlüssel Ihrer Kommunikationspartner*

- zum Verschlüsseln der Nachricht für die unterschiedlichen Empfänger und
- um die digitalen Unterschriften Ihrer Kommunikationspartner zu verifizieren.

4 Grundlegende PGP-Befehle

Dieses Kapitel beschreibt die grundlegenden PGP-Befehle in der Reihenfolge, in der sie beim ersten Arbeiten mit PGP benötigt werden. Dazu zählen unter anderem die Arbeitsschritte:

- ein eigenes Schlüsselpaar zu erstellen,
- einen eigenen öffentlichen Schlüssel vom Schlüsselbund „abzuhängen“ (kopieren); um ihn anschließend an Kommunikationspartner weiterzureichen,
- den öffentlichen Schlüssel von Kommunikationspartnern an den eigenen Schlüsselbund anzuhängen,
- Daten mit den vorhandenen Schlüsseln zu ver- und entschlüsseln und
- digitale Unterschriften mit dem privaten Schlüssel zu leisten.

4.1 Schlüsselverwaltung

4.1.1 Generieren eines öffentlich/privaten Schlüsselpaares

Das Kommando

```
pgp -kg
```

(key-generate) erzeugt ein Schlüsselpaar, das sowohl Ihren öffentlichen, als auch Ihren privaten Schlüssel enthält. Nach Absetzen des obigen Befehls werden Sie zur Beantwortung einer Reihe von Fragen am Bildschirm aufgefordert[4].

Schlüssellänge festlegen

Zuerst werden Sie gebeten, eine Schlüssellänge einzugeben, von denen es drei unterschiedliche gibt: je länger der Schlüssel, um so sicherer, aber auch langsamer läuft der Verschlüsselungsprozess ab.

```
C:\PGP> pgp -kg
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:00 GMT

Pick your RSA key size:
    1)   512 bits- Low commercial grade, fast but less secure
    2)   768 bits- High commercial grade, medium speed, good security
    3)  1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits: 2
Generating an RSA key with a 768-bit modulus.
```

[4] Die nachfolgend gerahmten Abschnitte zeigen beispielhaft den Dialog mit PGP. Die Eingaben, die Sie vornehmen müssen, sind fett gedruckt.

Schlüssel identifizieren

Anschließend werden Sie aufgefordert, Ihre ID einzugeben: das ist einfach nur Ihr Name, der zur eindeutigen Identifizierung Ihres Schlüssels benötigt wird. Die Namenskonvention sieht folgendermaßen aus:

Vorname Nachname <mailadresse>.

Verfügen Sie nicht über eine Mailadresse, so können Sie statt dessen auch Ihre Telefonnummer eintragen. Die ID wird, wie gesagt, nur zur eindeutigen Identifizierung Ihres Schlüssels benötigt.

```
You need a user ID for your public key.
The desired form for this
user ID is your name, followed by your Email address enclosed in
<angle brackets>, if you have an Email address.
For example:  John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key:
Anna Schmitt <anna.schmitt@fernuni-hagen.de>
```

Schlüsselschutz durch Paßwort-Satz

Danach müssen Sie einen *Paßwort-Satz* eingeben, der Ihren privaten Schlüssel schützt, falls er einmal in falsche Hände geraten sollte. Dieser Paßwort-Satz sollte aus mehreren Wörtern, Leerzeichen, Satzzeichen und sonstigen Sonderzeichen bestehen und möglichst nicht zu kurz sein. Aber – merken Sie ihn sich gut. Sie benötigen ihn nämlich immer wieder, wenn Sie Ihren privaten Schlüssel benutzen. Bei Verlust des Paßwort-Satzes gibt es in der momentanen PGP-Version *keine* Möglichkeit, ihn zu rekonstruieren. Der Paßwort-Satz unterscheidet im übrigen Groß- und Kleinschreibung und wird standardmäßig nicht am Bildschirm angezeigt. Um Tippfehlern vorzubeugen, werden Sie aber aufgefordert, den Paßwort-Satz ein zweites Mal zur Bestätigung einzugeben. Es erübrigt sich sicherlich, darauf hinzuweisen, daß Sie den Paßwort-Satz möglichst nicht aufschreiben und auch nicht auf Ihrem PC abspeichern sollten.

```
You need a pass phrase to protect your RSA secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase:
Enter same pass phrase again:
Note that key generation is a lengthy process.
```

Zufallszahlen

Nach Eingabe des Paßwort-Satzes wird Ihr Schlüsselpaar auf Basis zufällig ermittelter Zahlen erzeugt. Das funktioniert so, daß Sie einfach ein paar Zeichen eingeben, die durch die Schwankungen in Ihren Tastenanschlägen den Schlüssel erzeugen. Drücken Sie deshalb nicht nur einfach dieselbe Taste, sondern wählen Sie einige zufällig aus. Sobald Ihr PC piept, wird die Generierung des Schlüsselpaares begonnen.

```
We need to generate 408 random bytes.  This is done by measuring the
time intervals between your keystrokes.  Please enter some random text
on your keyboard until you hear the beep:
...................**** ..........................****
0 *  -Enough, thank you.
Key generation completed.
```

Die Schlüssel sind erstellt

Ihr öffentlicher Schlüssel wird automatisch an Ihren öffentlichen Schlüsselbund gehängt, Ihr privater entsprechend an Ihren privaten Schlüsselbund, wo er zusätzlich durch den Paßwort-Satz geschützt ist. Eine Kopie Ihres öffentlichen Schlüssels können Sie an alle Freunde und Bekannte im Netz weitergeben[5], die diesen dann an ihren öffentlichen Schlüsselbund hängen, *Ihren privaten Schlüssel geben Sie bitte niemals ab*.

Die Schlüsselbunde werden auf Ihrem PC abgelegt unter den Namen

PUBRING.PGP	für öffentliche Schlüssel
SECRING.PGP	für private Schlüssel

Für Fortgeschrittene

Die Tastenanschläge und die Schwankungen dazwischen werden von PGP dazu verwendet, Zufallszahlen zu erzeugen. Bei jeder Zufallszahl überprüft PGP anschließend, ob es sich um eine Primzahl handelt. Sobald zwei Primzahlen gefunden werden, werden diese zur Erstellung Ihres Schlüsselpaares verwendet. Die Punkte und Pluszeichen oder Sterne, die bei der Schlüsselerstellung am Bildschirm erscheinen, zeigen an, ob die Zahl keine Primzahl ist (Punkt) oder eventuell eine Primzahl ist (Plus/Stern).

4.1.2 Anlisten aller Schlüssel an einem Schlüsselbund

Wollen Sie nachsehen, welche Schlüssel sich bereits an Ihrem öffentlichen und privaten Schlüsselbund befinden, so setzen Sie das Kommando

```
pgp -kv [schlüsselbesitzer][schlüsselring]
```
[6]

(key-view) ab. Angelistet werden standardmäßig alle Schlüssel am öffentlichen Schlüsselbund:

```
c:\PGP>pgp -kv
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:05 GMT

Key ring: 'c:\pgp\pubring.pgp'
Type bits/keyID    Date       User ID
pub   768/5F8D8B79 1996/05/13 Anna Schmitt <anna.schmitt@fernuni-hagen.de>
pub   512/43A8CACD 1995/02/16 Bernhard Vogeler <Bernhard.Vogeler@FernUni-Hagen.de>
pub  1024/C08CAEED 1995/03/23 Isabel Rode <isabel.rode@fernuni-hagen.de>
pub   512/A8473C45 1994/10/28 Enno de Vries <Enno.deVries@FernUni-Hagen.DE>
```

[5] Zu diesem Zweck existieren eine Reihe sogenannter Keyserver, die eine komfortable Weitergabe von Schlüsseln gestatten. Mehr dazu im Kapitel 12 auf Seite 67.

[6] Angaben in eckigen Klammern sind wahlweise: werden sie weggelassen, so werden Standardeinstellungen von PGP verwendet, ansonsten gelten die von Ihnen angegebenen Werte.

```
pub   512/CC1322F5 1995/01/30 carsten schippang <carsten.schippang@fernuni-hagen.de>
pub  1024/8C5DADC5 1996/03/28 Brigitte Wiberg <Brigitte.Wiberg@fernuni-hagen.de>
pub   512/070DB651 1994/10/28 Manuela Juergens <manuela.juergens@fernuni-hagen.de>
                              Manuela Juergens <manuela@manitu>
pub   512/FD17EC31 1995/03/02 Klaus Sternberger <klaus.sternberger@fernuni-hagen.de>
8 matching keys found.
```

Direkt nach der Erstellung des ersten Schlüsselpaares befindet sich wahrscheinlich nur Ihr eigener Schlüssel am Bund. In unserem Bespiel sehen Sie aber, daß sich bereits einige Schlüssel am Schlüsselbund angesammelt haben, unter anderem der öffentliche Schlüssel von Anna.

Um den Schlüssel eines bestimmten Schlüsselbesitzers anzusehen, können Sie den Namen des Schlüsselbesitzers, also seine ID, direkt angeben.

```
c:\PGP>pgp -kv Anna
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:06 GMT

Key ring: 'c:\pgp\pubring.pgp', looking for user ID "Anna".
Type bits/keyID    Date       User ID
pub   768/5F8D8B79 1996/05/13 Anna Schmitt <anna.schmitt@fernuni-hagen.de>
1 matching key found.
```

Der Name des Schlüsselbesitzers muß dabei nur soweit angegeben werden, bis er eindeutig ist. Hängt also auch der Schlüssel von Anna Müller an Ihrem Bund, so muß für den Schlüssel von Anna Schmitt das Kommando

pgp -kv “Anna S“

angegeben werden. Groß-/Kleinschreibung muß nicht berücksichtigt werden; sobald jedoch Leerzeichen im Namen auftreten, muß der Name in Anführungszeichen gesetzt werden.

Sind selbst die Namen der Schlüsselbesitzer nicht eindeutig, so geben Sie zur weiteren Identifizierung noch die Mail-Adresse, bzw. Telefonnummer an.

Für Fortgeschrittene

Sollte es einmal passieren, daß tatsächlich sowohl der Name, als auch die Email-Adresse oder Telefonnummer eines Schlüssels übereinstimmen, so können Sie zur Identifikation eines Schlüssels auch die hexadezimale keyID verwenden: im obigen Beispiel finden Sie Annas keyID als hexadezimale Zahl 5F8D8B79. Diesen Wert können Sie in allen PGP-Befehlen verwenden, vorausgesetzt, Sie schreiben 0x vor die keyID. Damit deuten Sie an, daß es sich um einen hexadezimalen Wert handelt. Sie können sich Annas Schlüssel also z.B. auch über den Befehl

```
pgp   -kv   0x5F8D8B79
```

anzeigen lassen.

Um die Schlüssel am privaten Schlüsselring zu sehen, geben Sie das Kommando

pgp -kv secring.pgp

ein.

4.1.3 Einen öffentlichen Schlüssel vom Schlüsselbund abhängen

Nachdem Sie Ihren öffentlichen Schlüssel erstellt haben, möchten Sie ihn irgendwann sicherlich auch an andere weitergeben, damit Sie anschließend verschlüsselte Daten austauschen können. Dazu hängen Sie ihn einfach *als Kopie* von Ihrem Schlüsselbund ab.[7]

```
pgp -kx schlüsselbesitzer schlüsseldatei
```

(key-extract).

Beispiel:

```
c:\pgp>pgp -kx Anna annas.key[8]
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:07 GMT

Extracting from key ring: 'c:\pgp\pubring.pgp', userid "Anna".

Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13

Key extracted to file 'annas.key'.
```

Annas öffentlicher Schlüssel wird in der Datei ANNAS.KEY abgelegt. Sie schickt diese Datei an Bob, der den Schlüssel an seinen öffentlichen Bund hängt und ihn anschließend verwendet, um Daten für Anna zu verschlüsseln.

Transportrisiko

Das Verschicken des öffentlichen Schlüssels birgt natürlich das Risiko, daß die Datei möglicherweise unterwegs abgefangen und verfälscht wird. Deshalb kann sie mit einer digitalen Unterschrift versehen werden, aber dazu später mehr.

4.1.4 Hinzufügen weiterer Schlüssel

Um weitere Schlüssel am Schlüsselbund aufzunehmen, benutzen sie das Kommando:

```
pgp -ka schlüsseldatei [schlüsselring]
```

(key-add). Der Schlüssel, der in der angegebenen Schlüsseldatei enthalten ist, wird standardmäßig dem öffentlichen Schlüsselbund hinzugefügt.

---

[7] Wollen Sie den Schlüssel per Email verschicken, so beachten Sie bitte Kapitel 5 auf Seite 25. In diesem Fall müssen Sie nämlich unbedingt die zusätzliche Option -a verwenden.

[8] UNIX bevorzugt als Namenserweiterung der Schlüsseldatei .PGP

Wenn Anna von Bob den öffentlichen Schlüssel in der Datei BOBS.KEY zugeschickt bekommt, so hängt sie diesen an ihren öffentlichen Schlüsselbund mit

```
c:\pgp>pgp -ka bobs.key
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:09 GMT

Looking for new keys...
pub   768/5ADF7C41 1996/05/14  Bob Meyer <bob.meyer@fernuni-hagen.de>

Checking signatures...

Keyfile contains:
   1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)? n
```

PGP findet in der angegebenen Datei den Schlüssel von Bob, sucht nach eventuell vorhandenen Unterschriften[9] und fordert Sie anschließend noch auf, den Schlüssel zu zertifizieren um damit Ihr Vertrauen in den neuen Schlüssel zu bestätigen. Bevor Sie jedoch das Kapitel 6 auf Seite 29 nicht gelesen haben, sollten Sie hier unbedingt ein *n* eingeben.

Befindet sich der neu einzufügende Schlüssel bereits am Bund, so wird er nicht noch einmal eingefügt.

Besitzt ein Schlüssel eine digitale Unterschrift, so wird diese mit dem Schlüssel zusammen abgespeichert. Bei bereits vorhandenen Schlüsseln wird die Unterschrift dann lediglich hinzugefügt.

4.1.5 Löschen von öffentlichen Schlüsseln

Um Schlüssel wieder vom Schlüsselbund zu entfernen, geben Sie ein:

```
pgp -kr  schlüsselbesitzer
```

(key-remove). Der Schlüssel des angegebenen Besitzers wird standardmäßig im öffentlichen Schlüsselring gesucht und anschließend gelöscht.

Beispiel:

```
c:\pgp>pgp -kr bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:10 GMT
Removing from key ring: 'c:\pgp\pubring.pgp', userid "bob".

Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
```

[9] Wie gesagt: Was es damit auf sich hat, werden Sie noch kennenlernen.

```
768-bit key, Key ID 5ADF7C41, created 1996/05/14

Are you sure you want this key removed (y/N)? y
Key removed from key ring.
```

Für Fortgeschrittene

Beim Löschen eines Schlüssels können Sie natürlich auch die ID Ihres eigenen Schlüssels angeben. Nachdem Sie Ihren Schlüssel tatsächlich am öffentlichen Schlüsselbund gelöscht haben, stellt PGP fest, daß ein weiterer Eintrag im geheimen Schlüsselbund vorhanden ist und erfragt, ob dieser auch gelöscht werden soll. Benutzen Sie diesen Befehl aber bitte sehr umsichtig: Wenn bereits mehrere Kommunikationspartner Ihren Schlüssel verwenden, müssen Sie diese darüber informieren, daß es Ihren Schlüssel nicht mehr gibt. Wenn Sie über keine Sicherungskopie Ihres Schlüssels verfügen, gibt es keinerlei Rekonstruktionsmöglichkeit. Die Folge ist, daß Sie Dateien, die Ihre Kommunikationspartner mit dem gelöschten Schlüssel für Sie kodiert haben, nicht mehr entschlüsseln können.

4.1.6 Sperren von Schlüsseln

Angenommen, Sie vermuten, daß jemand in den Besitz Ihres privaten Schlüssels und Ihres Paßwort-Satzes gelangt ist. Dann sollten Sie möglichst bald Ihren Schlüssel sperren und dieses auch allen mitteilen, die Ihren öffentlichen Schlüssel benutzen. Ein Schlüsselpaar sperren Sie durch Eingabe von

```
pgp -kd schlüsselbesitzer
```

(key disable). Sie erstellen mit dem obigen Befehl ein sogenanntes *Key Compromise Certificate*. Es handelt sich um eine Rückrufurkunde für Ihren Schlüssel, die auf jeden Fall Ihre Signatur trägt. Nachdem Sie Ihren Paßwort-Satz eingegeben haben, werden sowohl der öffentliche, als auch der private Schlüssel gesperrt. Diese Urkunde kennzeichnet Ihren PGP-Schlüssel als unbrauchbar und das Verfahren läßt sich auch nicht wieder rückgängig machen. Sie sollten die Urkunde in einer Datei speichern, mit

```
pgp -kxa schlüsselbesitzer dateiname
```

und diese Datei an wirklich *alle* Kommunikationspartner schicken, die über Ihren Schlüssel verfügen.

Am besten erstellen Sie gleichzeitig ein vollkommen neues Schlüsselpaar mit einem neuen Paßwort-Satz, hängen Ihren neuen öffentlichen Schlüssel ebenfalls vom Bund ab und verteilen ihn zusammen mit der Rückrufurkunde an Ihre Gesprächspartner.

Sollten Sie selbst einmal eine Rückrufurkunde von einem Gesprächspartner erhalten, so können Sie sie verwenden durch

```
pgp -ka dateiname
```

Dadurch wird ein entsprechender Sperrvermerk an dem Schlüssel befestigt, was Sie leicht mit dem Befehl

```
pgp -kv
```

überprüfen können.

Gesperrte Schlüssel können zum Verschlüsseln nicht mehr verwendet werden. Signaturen, die mit gesperrten Schlüsseln gemacht wurden, liefern einen entsprechenden Warnhinweis.

4.1.7 Verlorene Schlüssel

Sie haben Ihren Schlüssel verloren? Sprich: Sie haben Ihren Paßwort-Satz vergessen? Dann haben Sie ein großes Problem, da es keine Chance gibt, den Paßwort-Schutz in der aktuellen PGP-Version irgendwie zu knacken.

Sie können Ihren Schlüssel nicht einmal sperren, denn dafür benötigen Sie Ihren Paßwort-Satz. Sie können lediglich alle, die Ihren öffentlichen Schlüssel besitzen, bitten, diesen zu sperren.

Sie müssen sich anschließend ein neues Schlüsselpaar erzeugen und den öffentlichen Schlüssel neu verteilen.

Für Fortgeschrittene

Falls Sie dem Fall der Fälle vorbeugen wollen, können Sie eine Rückrufurkunde vorbereiten. Sie sollten dabei aber mit größter Vorsicht vorgehen, damit Sie Ihren Schlüssel nicht tatsächlich sperren; die Sperre läßt sich nicht wieder rückgängig machen.

- *Kopieren Sie direkt nach der Schlüsselerzeugung Ihren öffentlichen Schlüssel in eine Datei,*
- *sperren Sie anschließend Ihren öffentlichen Schlüssel wie im vorigen Kapitel beschrieben und speichern Sie die Rückrufurkunde an einem für Fremde unzugänglichen Platz,*
- *löschen Sie Ihren gesperrten Schlüssel und*
- *fügen Sie schließlich Ihren anfangs abgehängten Schlüssel wieder ein.*

So können Sie selbst bei verlorenem Passwort die sicher aufbewahrte Rückrufurkunde verteilen. Das gleiche gilt auch für die Fälle, in denen Sie Ihre Schlüsselbunde versehentlich gelöscht haben und über keinerlei Kopien mehr verfügen.

4.2 Verschlüsseln und Entschlüsseln von Daten

Um es noch einmal zusammenzufassen: Bevor Sie verschlüsselte Daten austauschen können

- sollten Sie sich ein eigenes Schlüsselpaar erstellt haben,
- den öffentlichen Schlüssel Ihrer Gesprächspartner besitzen
- und Ihren öffentlichen Schlüssel an die Gesprächspartner weiterleiten.

4.2.1 Verschlüsseln von Daten

Daten werden verschlüsselt mit dem *öffentlichen Schlüssel* des Empfängers. Dieser Schlüssel befindet sich zusammen mit dem Namen des Empfängers am

öffentlichen Schlüsselbund, dem sogenannten *public key ring*. Dies ist standardmäßig die Datei PUBRING.PGP.

Die Verschlüsselung nehmen Sie folgendermaßen vor:

```
pgp -e dateiname empfängername
```

(encrypt).

Ein Beispiel:

Sie besitzen Annas öffentlichen Schlüssel und möchten damit eine Datei für sie kodieren.

```
C:\PGP>pgp -e brief.txt anna
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:30 GMT

Recipients' public key(s) will be used to encrypt.
Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13
.
Ciphertext file: brief.pgp
```

PGP sucht an Ihrem öffentlichen Schlüsselbund nach der Zeichenkette „anna“; und verwendet dann anschließend den zugehörigen öffentlichen Schlüssel zur Kodierung.

Auch hier müssen Sie beachten: sobald der Name des Schlüsselbesitzers nicht eindeutig ist, müssen so viele Zeichen eingegeben werden, wie zur Identifizierung nötig sind, also zum Beispiel:

pgp -e brief.txt “anna sch“

Sobald Sonderzeichen im Empfängernamen auftauchen, wie hier das Leerzeichen, setzen Sie die Zeichenkette einfach in Gänsefüßchen.

Verschlüsselte Datei

Der verschlüsselte Text befindet sich anschließend automatisch in der Datei BRIEF.PGP[10]. Vorab führt PGP eine Datenkompression durch, so daß die verschlüsselte Datei durchaus kleiner sein kann, als das Original.

Zertifizierungshinweis

Möglicherweise bekommen Sie nach Eingabe des Verschlüsselungsbefehls von PGP eine Frage gestellt, deren Beantwortung Sie sich genau überlegen sollten. Dazu ein weiteres Beispiel. Anna möchte eine Datei für Bob verschlüsseln:

```
C:\PGP>pgp -e brief.txt bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:32 GMT
```

[10] PGP nennt solche verschlüsselten Dateien Cyphertext-File.

```
Recipients' public key(s) will be used to encrypt.
Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 5ADF7C41, created 1996/05/14
…
WARNING:  Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Bob Meyer <bob.meyer@fernuni-hagen.de>".

Are you sure you want to use this public key (y/N)? y
Ciphertext file: brief.pgp
```

Sehen Sie sich die Warnung im obigen Beispeil einmal genau an. Sie erinnert Sie daran, daß Sie dem Schlüssel von Bob noch nicht Ihr volles Vertrauen ausgesprochen haben. Sie können so einen Schlüssel benutzen, aber mit Vorsicht. Wann Sie einem Schlüssel trauen können, lesen Sie im Kapitel 6 auf Seite 29.

4.2.2 Entschlüsseln einer Datei

Um eine Nachricht zu entschlüsseln, benutzen Sie das Kommando

```
pgp verschlüsselte_datei
```

Beispiel: Anna möchte die Datei, die Bob ihr zugeschickt hat, entschlüsseln:

```
C:\PGP>pgp brief
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:36 GMT

File is encrypted.  Secret key is required to read it.
Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: Pass phrase is good.  Just a moment......
Plaintext filename: brief
```

Passwort-Eingabe

Zur Entschlüsselung verwendet PGP automatisch den *geheimen Schlüssel* von Anna, der durch Ihren Passwort-Satz geschützt ist. Aus diesem Grund wird Sie aufgefordert, Ihr Passwort einzugeben. Ist die Eingabe korrekt, wird die verschlüsselte Datei BRIEF.PGP entschlüsselt und unverschlüsselt in der Datei BRIEF abgelegt. Diese kann von Anna problemlos gelesen werden. Wünschen Sie eine andere Ausgabedatei für die entschlüsselte Nachricht, so können Sie den Dateinamen als Option beim PGP-Kommando mit angeben:

pgp verschlüsselte_datei [-o dateiname]

4.2.3 Verschlüsseln einer Datei für mehrere Empfänger

Soll eine verschlüsselte Datei an mehrere Empfänger übermittelt werden, so kann der Verschlüsselungsprozess in einem Arbeitsschritt ausgeführt werden: geben Sie einfach alle Empfänger an, und PGP sucht der Reihe nach die entsprechenden öffentlichen Schlüssel an Ihrem Schlüsselbund.

pgp -e brief.txt anna bob charlie

Die daraus resultierende Datei BRIEF.PGP kann anschließend jeweils mit den geheimen Schlüsseln von Anna, Bob und Charlie entschlüsselt werden.

4.2.4 Verschlüsseln einer Datei bei gleichzeitiger digitaler Unterschrift

Authentizität einer Nachricht

Eine digitale Unterschrift erfüllt den gleichen Zweck, den auch eine normale Unterschrift erfüllt: man kann von der Echtheit des Absender überzeugt sein. Über die Möglichkeit einer Urkundenfälschung, die prinzipiell auch bei PGP gegeben ist, informiert Sie das Kapitel 6 auf Seite 29.

Wenn Sie sich also nun entschließen, eine Datei nicht nur zu verschlüsseln sondern gleichzeitig auch noch zu unterschreiben, dann können Sie das in einem PGP -Aufruf durchführen:

```
pgp -es Datei    Empfänger
```

(encrypt and sign).

Beispiel: Anna will Bob eine unterschriebene verschlüsselte Nachricht schikken:

```
C:\PGP>pgp  -es brief.txt bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:37 GMT

A secret key is required to make a signature.
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Anna Schmitt <anna.schmitt@fernuni-hagen.de>"

Enter pass phrase: Pass phrase is good.
Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13
Just a moment....

Recipients' public key(s) will be used to encrypt.
Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 5ADF7C41, created 1996/05/14
…
WARNING:  Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Bob Meyer <bob.meyer@fernuni-hagen.de>".
But you previously approved using this public key anyway.
Ciphertext file: brief.pgp
```

PGP führt den Auftrag in mehreren Arbeitschritten aus:

Zunächst wird Annas *privater* Schlüssel von ihrem privaten Schlüsselbund benutzt um die digitale Unterschrift zu erstellen. Da der private Schlüssel immer durch ihr Passwort geschützt ist, muß Anna dieses zur Bestätigung korrekt eingeben.

Danach wird automatisch Bobs *öffentlicher* Schlüssel, der an Annas öffentlichem Schlüsselbund hängt, benutzt, um die Datei zu verschlüsseln. Das Ergebnis befindet sich in komprimierter Form anschließend zusammen mit Annas Unterschrift in der Datei BRIEF.PGP[11].

Für Fortgeschrittene

Wenn Sie mit mehreren privaten Schlüsseln arbeiten, so hängen diese alle an einem privaten Schlüsselbund. Für die Signatur wird in solchen Fällen standardmäßig der erste gefundene geheime Schlüssel verwendet. Ist das nicht der Schlüssel, den Sie verwenden möchten, so müssen Sie dem PGP-Kommando mittels der Option -u den tatsächlich benötigten Schlüssel angeben, also z.B.

pgp -es brief.txt bob -u charlie

Dann wird für die Signatur der geheime Schlüssel von Charlie verwendet.

4.2.5 Entschlüsseln einer unterschriebenen Datei

Zum Entschlüsseln einer unterschriebenen Datei verfahren Sie wie bei allen anderen verschlüsselten Dateien auch. PGP erkennt automatisch, ob eine digitale Unterschrift beigefügt wurde oder nicht und informiert Sie über die Korrektheit des Absenders. Wenn Anna also eine von Bob verschlüsselte und unterschriebene Datei erhält, geht Sie folgendermaßen vor:

```
c:\pgp> pgp brief
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:42 GMT

File is encrypted.  Secret key is required to read it.
Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: Pass phrase is good.  Just a moment......
File has signature.  Public key is required to check signature. .
Good signature from user "Bob Meyer <bob.meyer@fernuni-hagen.de>".
Signature made 1996/05/14 09:57 GMT
Plaintext filename: brief
```

Zur Entschlüsselung wird wieder Annas geheimer Schlüssel verwendet und das macht erneut die Eingabe Ihres Passwortes erforderlich.

Anschließend erkennt PGP Bobs digitale Unterschrift und verwendet nun automatisch *Bobs öffentlichen Schlüssel*, um die Korrektheit der Signatur zu bestätigen. Achten sie also immer auf Textzeilen der Art: Good signature ...,

[11] Sie erhalten aber auf jedenfall einen Hinweis darauf, daß Bobs Schlüssel noch nicht von Ihnen als echt bestätigt wurde. Verwendet wird der Schlüssel nun allerdings automatisch, da Sie bereits voher einer Verwendung zugestimmt haben.

wie sie im obigen Beispiel zu sehen ist. Enthält eine Datei eine Signatur, deren öffentlichen Schlüssel Sie nicht besitzen, so werden Sie ebenfalls darauf hingewiesen. PGP kann die Echtheit des Schlüssels dann nicht überprüfen.

4.2.6 Konventionelle Verschlüsselung

PGP kann, falls gewünscht, auch eine konventionelle Verschlüsselung, also die Kodierung mit Hilfe eines einzigen Schlüssels, vornehmen. Das ist zum Beispiel dann nützlich, wenn Sie eine Datei nur verschlüsselt abspeichern möchten, sie aber nicht über Datenleitungen oder Disketten weiterverschikken wollen. In diesem Fall werden *Sie selbst* die Datei verschlüsseln *und* entschlüsseln. Zur konventionellen Verschlüsselung benutzen Sie:

```
pgp -c dateiname
```

(conventional).

Die angegebene Datei wird verschlüsselt mit der Dateinamenserweiterung .PGP abgelegt. Dazu wird *keiner* der öffentlichen oder privaten Schlüssel benutzt. Statt dessen werden Sie nach einem Paßwort gefragt, das als konventioneller Schlüssel dient. Dieses Paßwort ist unabhängig von dem Paßwort-Satz, den Sie zum Schutz Ihres privaten Schlüssels benutzen. Zur Entschlüsselung benutzen Sie das bereits vorgestellte Kommando

```
pgp verschlüsselte_datei
```

Sie werden dann zur Eingabe des zugehörigen Paßwortes aufgefordert, das Sie zur konventionellen Verschlüsselung benutzt haben.

5 Verschicken verschlüsselter Daten per Email

Der Austausch von Schlüsseln und Nachrichten wird sicherlich in der Regel über diverse Email-Systeme vorgenommen werden. Viele Electronic-Mail-Systeme verlangen, daß Nachrichten und Dateien aus ASCII-Text bestehen. Die von PGP standardmäßig erstellten verschlüsselten Dateien beinhalten jedoch binäre Zeichen. PGP unterstützt aber auch die Erstellung von Dateien im sogenannten ASCII-Radix-64-Format. Das sind Binärdateien, die nur druckbare ASCII-Zeichen enthalten. Diese Dateien sind allerdings um rund 33% größer als die Standard-Binärdateien.

Um das Radix-64 Format zu erhalten, können Sie fast allen PGP-Befehlen den Buchstaben **a** hinzufügen.

Angenommen Anna möchte Bob eine verschlüsselte und digital unterschriebene Nachricht per Mail zuschicken und benötigt deshalb das Radix-64 Format, so benutzt sie folgenden Befehl:

Datei im Radix-64 Format

```
C:\PGP>pgp  -esa  mail.txt  bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 14:47 GMT

A secret key is required to make a signature.
You specified no user ID to select your secret key,
so the default user ID and key will be the most recently
added key on your secret keyring.

You need a pass phrase to unlock your RSA secret key.
Key for user ID "Bob Meyer <bob.meyer@fernuni-hagen.de>"

Enter pass phrase: Pass phrase is good.
Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 2DDB81C9, created 1996/05/14
Just a moment....

Recipients' public key(s) will be used to encrypt.
Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 2DDB81C9, created 1996/05/14
.
Transport armor file: mail.asc
```

Die verschlüsselte Nachricht befindet sich nun in der Datei MAIL.ASC und kann per Email verschickt werden. Der verschlüsselte Inhalt besteht aus reinem ASCII, wie das nachfolgende Beispiel zeigt:

```
-----BEGIN PGP MESSAGE-----
Version: 2.6.3i
hGwDrNj16S3bgckBAv0XNeSDtuPUzGD2tnd4l0tfDEVCMzOV4/kTzg/rHmrs+/yI
wKlrq9iIj3tsPV6SvugXIlM2AGjspAf8nhMbDCpnjOVroIx/682S5Keg6bSq0D5c LtKO-
QDuZZZ5bKZ1Osr+mAAAA99tAIc3C4HSYlVZMkjBcgsS/K5jDUQvpeviLgyb/
1eI8C5HMs2Xauj5J44q6nGDHZrQLRQ6Jw0fk2zktGEX/9MoKSPQY0x19Sb1yCWkp
1GUrsTYMBwyErnXKxgVQfEOuJ5BgNxWCIZql9sW8a8iDDjBrcG0lHWyf86bQijpG
6FRezVoSrr9Cq1blcW6rWuFo58T9mUQqaaDExGqfVXbu1bcAPio/BrV+UESs8rCo
nzj7Kc8fZhK7cq/EPXmkdbMq4xoeL1LCno9n1leXkiEZ2k8ImhaIZ9/rr2VRQa1W
oIG76E6G3Jj/0XsZwxNxLwXiD1DyQZDA37Q=
=Ucz+
-----END PGP MESSAGE-----
```

Zum Entschlüsseln der Datei wird das bereits bekannte Kommando

pgp mail

benutzt. PGP sucht automatisch zuerst nach der Datei MAIL.ASC, erkennt das Radix-64 Format, konvertiert es in ein binäres Format und entschlüsselt es anschließend.

Schlüssel im Radix-64 Format

Wollen Sie Ihren öffentlichen Schlüssel per Email verschicken, so benutzen Sie zum Extrahieren ebenfalls die zusätzliche Option **a** und Ihr Schlüssel erscheint im Radix-64 Format.

```
pgp -kxa  schlüsselbesitzer schlüsseldatei
```

Haben Sie beim Verschlüsseln oder Extrahieren Ihres Schlüssels die Angabe **a** vergessen, so können Sie PGP jederzeit mit der -a-Option aufrufen um eine Konvertierung ins Radix-64 Format vorzunehmen.

```
pgp   -a   schlüsseldatei
```

```
c:\pgp>pgp -a annas.key
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 15:10 GMT

Input file 'annas.key' looks like it may have been created by PGP.
Is it safe to assume that it was created by PGP (y/N)? y
Transport armor file: annas.asc
```

Die Datei ANNAS.ASC mit dem Schlüssel sieht anschließend ungefähr folgendermaßen aus:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQBtAzGXNjoAAAEDAMb12EdWMAfXqoY2RbZrxKbVlPo3EPclNSSZrr0rF60w3e8s
5mA4OP3zD5iIOHUbYf54yJudsAw28IjuI37Yy2/YX+R+1AnFGTxkGJ8abrMgXZUV
offJErKBgxBlX42LeQAFEbQsQW5uYSBTY2htaXR0IDxhbm5hLnNjaG1pdHRAZmVy
bnVuaS1oYWdlbi5kZT4=
=0cFl
-----END PGP PUBLIC KEY BLOCK-----
```

Anhand der ersten und letzten Zeile können Sie jeweils erkennen, ob es sich um eine verschlüsselte Message oder um einen Schlüssel handelt.

Wenn Sie standardmäßig das Radix-64 Format verwenden möchten, so können Sie diese Einstellung in der Konfigurationsdatei festlegen. Näheres hierzu lesen Sie im Kapitel 12 auf Seite 67.

Für Fortgeschrittene

Innerhalb des Internets ist normalerweise die Größe einer zu verschickenden Datei auf 50.000 Byte beschränkt. PGP berücksichtigt dies bei Verwendung der Option a zur Erstellung des Radix-64-Formats automatisch: Es splittet die verschlüsselten Daten in einzelne Dateien auf, die die Namensendungen .as1, .as2, .as3 usw. bekommen. Der Empfänger ruft PGP dann einfach für die erste Datei mit der Endung .as1 auf und PGP fügt alle zusammengehörenden Dateien selbständig wieder zusammen.

6 Vertrauen zu Schlüsseln

Grundsätzlich sollten Sie der Echtheit eines Schlüssels, den Sie nicht *persönlich und direkt* empfangen haben, mißtrauen. Insbesondere wenn Sie einen Schlüssel per Email oder gar über einen Key-Server erhalten haben, sollten Sie solange skeptisch bleiben, bis Sie sich mit Hilfe verschiedener Verfahren von der Echtheit des Schlüssels hundertprozentig überzeugen konnten.

Das Problem mit der Echtheit eines Schlüssels besteht darin, daß eine dritte Person einen eigenen Schlüssel unter fremdem Namen erstellt und diesen im Netz verteilt. Dazu ein Beispiel:

> Bob möchte Anna eine verschlüsselte Mail schicken und bittet Sie per unverschlüsselter Email um Ihren öffentlichen Schlüssel.
>
> Was Bob und Anna nicht wissen: Joe, der sich mit Computern bestens auskennt, „belauscht" die beiden, fängt Annas Schlüssel auf dem Weg zu Bob einfach ab und erstellt stattdessen einen eigenen Schlüssel, den er unter Annas Namen an Bob weiterleitet.
>
> Jedesmal wenn Bob jetzt eine Nachricht für Anna verschlüsselt, verwendet er, ohne es zu wissen, eigentlich Joes öffentlichen Schlüssel und schickt die so kodierte Mail an Anna.
>
> Joe fängt die Mail wieder ab, kann mit seinem privaten Schlüssel die Nachricht entschlüsseln und, schlimmer noch: er kann den Inhalt sogar unbemerkt verändern, bevor er ihn anschließend mit Annas tatsächlichem öffentlichen Schlüssel erneut kodiert und Anna zuschickt.
>
> Anna entschlüsselt dann die Mail; weder sie, noch Bob bemerken etwas von Joe, der die ganze Unterhaltung unbemerkt verfolgen und manipulieren kann.

Wenn Sie den obigen Trick nicht sofort verstanden haben, machen Sie sich bitte die Mühe und lesen Sie den Abschnitt noch einmal durch. Es ist wirklich wichtig, die Problematik zu verstehen.

Die oben beschriebenen Gefahren können Sie mit PGP umgehen, wenn Sie konsequent einen Vertrauensbeweis für alle Schlüssel fordern, die Sie nicht direkt und persönlich in Empfang genommen haben. Erst dann sind die beiden wichtigsten Anforderungen an PGP, nämlich

- *Integrität*, das heißt, Ihre Dateien wurden auf dem Weg nicht verfälscht und
- *Authentizität*, das heißt, die empfangenen Daten sind garantiert von dem richtigen Absender

gewährleistet.

Welche Möglichkeiten bieten sich Ihnen, um sich von der Echtheit eines Schlüssels zu überzeugen?

Sie können den Schlüssel zum Beispiel per Telefon überprüfen, indem Sie dem Kommunikationspartner den ASCII-Code des Schlüssels durchgeben und vergleichen.

Fingerprints, um Schlüssel zu verifizieren

Da das aber sehr mühsam ist, gibt es zu diesem Zweck eine Erleichterung: Jeder Schlüssel besitzt eine Art Fingerabdruck, den sogenannten *Fingerprint*, den Sie sich mit dem Befehl

```
pgp -kvc [schlüsselbesitzer]
```

anzeigen lassen können. Wenn Sie auf die Angabe eines Schlüsselbesitzers verzichten, so werden die Fingerprints aller Schlüssel Ihres öffentlichen Schlüsselbundes angezeigt.

```
C:\pgp>pgp -kvc anna
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 16:20 GMT

Key ring: 'c:\pgp\pubring.pgp', looking for user ID "anna".
Type bits/keyID    Date       User ID
pub   768/5F8D8B79 1996/05/13 Anna Schmitt <anna.schmitt@fernuni-hagen.de>
          Key fingerprint =  67 BD 1C D9 54 20 D9 23  DC FF 18 7D 10 04 51 58
1 matching key found.
```

Der Fingerprint besteht aus 16 hexadezimalen Ziffern des Schlüssels, die Sie nun bequem per Telefon verifizieren können. Der Fingerabdruck eines Schlüssels ist eindeutig.

Wählen Sie zur Überprüfung des Fingerprints vorzugsweise nicht Emails, sondern eine alternative Möglichkeit (eben Telefon oder Brief). Eine Fingerprint-Verifikation per Email ist schließlich wieder den zuvor beschriebenen Gefahren und Manipulationen ausgesetzt.

Einen Schlüssel unterschreiben

Wenn Sie von der Echtheit eines Schlüssels überzeugt sind, und wirklich nur dann, können Sie ihn mit *Ihrer Unterschrift* versehen. *Sie bestätigen damit auch anderen Benutzern, daß Sie zu diesem Schlüssel hundertprozentiges Vertrauen haben*. Sie bürgen sozusagen mit Ihrem guten Namen für die Echtheit des Schlüssels.

Ihre Unterschrift unter einem Schlüssel hat eine wichtige Funktion: Derjenige, der einen Schlüssel unterschreibt, fungiert als sogenannter *Introducer*, also jemand der zwei andere Personen miteinander bekannt macht.

Wenn Bob zum Beispiel Anna kennt und vertraut und Ihren Schlüssel persönlich in Empfang genommen hat, bzw. den Fingerprint per Telefon verifiziert hat, so kann er ihn mit seiner Unterschrift versehen. Er gibt Anna anschließend eine Kopie ihres eigenen Schlüssels, der jetzt Bobs Unterschrift trägt, zurück.

Wenn Anna nun Charlie kennenlernt, der ein guter Freund von Bob ist, so vertraut Charlie möglicherweise aufgrund von Bobs Unterschrift ebenfalls Annas Schlüssel und verwendet ihn. Ob Charlies Vertrauen jedoch soweit geht, daß er Annas Schlüssel ebenfalls signiert ist fraglich und auch nicht erwünscht. Charlie sollte Annas Schlüssel ausschließlich dann unterschreiben, wenn er sich persönlich und unabhängig von anderen Unterschriften von der Echtheit von Annas Schlüssel überzeugt hat.

Wenn er Annas Schlüssel unterschreibt, so gilt er nun ebenfalls als Introducer für Anna, woraufhin vielleicht auch Erna, Charlies Tante, nun Annas Schlüssel vertraut, obwohl sie weder Bob noch Anna persönlich kennt.

Je mehr Unterschriften Annas Schlüssel im Laufe der Zeit sammelt, um so größer ist die Wahrscheinlichkeit, daß eine fremde Person Annas Schlüssel vertraut, weil er die Unterschrift einer ihm bekannten Person trägt, der er vertraut.

Eine ausführliche Beschreibung der Problematik bezüglich der Glaubwürdigkeit von Schlüsseln und ihren Besitzern finden Sie in der Dokumentation, die der PGP-Software beigefügt ist. Bevor Sie die Schlüssel anderer Besitzer unterschreiben, lesen Sie sich das entsprechende Kapitel unbedingt durch.

Wenn Sie sehen wollen, welche Unterschriften ein Schlüssel zur Bestätigung trägt, geben Sie das Kommando

```
pgp -kvv [schluesselbesitzer]
```

ein.

```
C:\pgp>pgp -kvv anna
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 16:25 GMT

Key ring: 'c:\pgp\pubring.pgp', looking for user ID "anna".
Type bits/keyID    Date       User ID
pub   768/5F8D8B79 1996/05/13 Anna Schmitt <anna.schmitt@fernuni-hagen.de>
sig       1CA3F065             Bob Meyer <bob.meyer@fernuni-hagen.de>
sig       5F8D8B79             Anna Schmitt <anna.schmitt@fernuni-hagen.de>
sig       2DDB81C9             (Unknown signator, can't be checked)
1 matching key found.
```

Annas Schlüssel besitzt also bereits 3 Unterschriften:

Eine von Bob: nach der telefonischen Überprüfung des Fingerprints ist Bob sicher, daß der Schlüssel auch wirklich von Anna erstellt worden ist.

- Er unterschreibt also den Schlüssel[12],
- hängt Annas Schlüssel anschließend mit dem bekannten Kommando

  pgp -kxa anna annas.key

  von seinem Schlüsselbund ab, wobei Bobs Unterschrift dann auch automatisch in der Datei ANNAS.KEY vorhanden ist,
- schickt Anna den Schlüssel zu
- und Anna hängt ihren eigenen von Bob unterschriebenen Schlüssel mit dem bekannten Kommando

  pgp -ka annas.key

  an ihren Schlüsselbund an. PGP stellt automatisch fest, daß die Datei einen Schlüssel enthält, der sich bereits an dem Bund befindet und erweitert den Schlüssel lediglich um die Unterschrift von Bob.

Da sich Bobs öffentlicher Schlüssel auch an Annas Schlüsselbund befindet, kann PGP nun überprüfen, ob Bobs Unterschrift auch korrekt und nicht gefälscht ist. Ein Schlüssel mit einer so versehenen *guten* Unterschrift ist in der Regel vertrauenswürdig.

Eine zweite Unterschrift ist Annas eigene Signatur. Grundsätzlich sollte jeder seinen eigenen Schlüssel unterschreiben. Er ist so absolut sicher vor Manipulationen, die ansonsten von wirklichen Kennern der PGP-Interna insbesondere an den Schlüsselidentifikationen, also zum Beispiel der Mail-Adresse vorgenommen werden könnten.

Eine dritte Unterschrift ist zwar vorhanden, aber an Annas öffentlichen Schlüsselbund befindet sich der zugehörige Schlüssel des Unterschreibenden nicht. Deshalb kann eine Korrektheit des Schlüssels nicht überprüft werden und PGP meldet einen unbekannten Unterschreiber.

Fazit: Sammeln Sie möglichst viele Unterschriften unter Ihrem eigenen Schlüssel, so daß möglichst viele Kommunikationspartner Ihren Schlüssel aufgrund der Unterschriften verifizieren können. Seien Sie gleichzeitig umsichtig bei der Vergabe Ihrer Unterschrift unter einen Schlüssel.

[12] Sie werden gleich noch kennenlernen, wie das geht.

Um einen Schlüssel zu unterschreiben, verwenden Sie den Befehl

Schlüssel unterschreiben

```
pgp -ks schlüsselbesitzer
```

(key sign).

Ein Beispiel: Anna will nun auch Bobs Schlüssel unterschreiben.

```
c:\pgp>pgp -ks bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 16:50 GMT

Looking for key for user 'bob':

Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 1CA3F065, created 1996/05/15
          Key fingerprint =  13 C3 BF 2A 78 5B B2 FE  5A 61 37 EA 65 BF 6F 04

READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Anna Schmitt <anna.schmitt@fernuni-hagen.de>"

Enter pass phrase: Pass phrase is good.  Just a moment....
Key signature certificate added.

Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on available
evidence.  If you think it does, then based on your estimate of
that person's integrity and competence in key management, answer
the following question:

Would you trust "Bob Meyer <bob.meyer@fernuni-hagen.de>"
to act as an introducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?
2
```

Nachdem Anna bestätigt hat, daß sie wirklich sicher ist, daß der Schlüssel Bob gehört, muß sie zur Echtheit Ihrer Unterschrift Ihren Passwort-Satz angeben. Anschließend trägt Bobs Schlüssel Annas korrekte Unterschrift. Anna sollte Bob den unterschriebenen Schlüssel danach zuschicken.

Im obigen Beispiel finden Sie aber noch eine abschließende Frage von PGP: Sie sollen nämlich entscheiden, ob Ihr Vertrauen in Bob so weit geht, daß Schlüssel die von ihm unterschrieben wurden, auch automatisch Ihr Vertrauen genießen. Dieses Konzept der verschiedenen Personen, die als Introducer fungieren dürfen, erzeugt unter Umständen eine lange verschachtelte Kette von Vertrauensbekundungen. Für den Anfang sind Sie gut damit bedient, hier lediglich die Auswahl 1 oder 2 zu treffen. Wenn Sie sich mit diesem sogenannten *Trust Model* allerdings näher befassen möchten, sollten Sie sich das

bereits eingangs erwähnte Buch von William Stallings zu Gemüte führen. Das recht komplizierte Trust Model wird hier detailliert erläutert.

Für Fortgeschrittene

Es gibt einen Befehl, mit dessen Hilfe Sie sich sehr ausführliche Information über die einzelnen Vertrauensketten Ihrer gesammelten öffentlichen Schlüssel anzeigen lassen können.

pgp -km

zeigt Ihnen alle Schlüssel mit dem jeweiligen Vertrauensstatus an, wobei Ihr eigener Schlüssel automatisch die höchste Vertrauenstufe ultimately-trusted *besitzt.*

Eine Unterschrift wieder entziehen

Möchten Sie einem öffentlichen Schlüssel Ihre Unterschrift wieder entziehen, so können Sie dafür den Befehl

```
pgp -krs schlüsselbesitzer
```

(key remove signatur) verwenden.

7 Einige weitere Kommandos

Es gibt eine ganze Reihe weiterer Kommandos, die in der Regel nicht unbedingt zur tagtäglichen PGP-Verschlüsselung benötigt werden. PGP-Einsteiger, die eigentlich nur einen schnellen Einblick in die Thematik benötigen, um möglichst rasch abhörsichere Emails in die Welt zu schicken, können dieses Kapitel also getrost überschlagen.

7.1 Vertrauliche Informationen schützen

Um vertrauliche Information nicht nur auf dem Weg durchs Netz zu schützen, sondern jegliche Spuren der geheimen Information auch auf dem Rechner selbst zu verhindern, stehen Ihnen drei weitere PGP-Befehle zur Verfügung.

7.1.1 Klartext restlos löschen

Nachdem Sie eine vertrauliche Information in eine Datei getippt haben, können Sie dafür sorgen, daß PGP diese direkt nach der Verschlüsselung von Ihrem Rechner restlos entfernt. Dazu wird nicht, wie bei vielen Lösch-Befehlen der Betriebssysteme üblich, lediglich der Verzeichniseintrag gelöscht, während die einzelnen Fragmente der Datei erhalten bleiben. Stattdessen werden die Dateifragmente vor dem Löschen mit Nullen bzw. Zufallszahlen gefüllt[13] und erst dann wird die Datei auch im Verzeichniseintrag als gelöscht gekennzeichnet.

Vorsicht!

Aber Vorsicht: Je nachdem welchen Editor Sie zur Erstellung Ihrer Originaldatei verwendet haben, werden von diesem automatisch temporäre und Sicherungsdateien angelegt, die von PGP natürlich nicht gelöscht werden.

Das Löschen des Originaltextes wird als *wiping* bezeichnet und kann in Kombination mit den PGP -Verschlüsselungskommandos als Option *w* hinzugefügt werden.

```
pgp -esw dateiname empfänger
```

7.1.2 Verschlüsseln nur für die Bildschirmanzeige

Um auch dafür zu sorgen, daß der Empfänger Ihrer Nachricht den Inhalt nicht leichtsinnig auf seinem Rechner abspeichert, können Sie mit Hilfe einer weiteren Option PGP dazu veranlassen, daß der Empfänger den entschlüsselten Inhalt lediglich an seinem Bildschirm betrachten kann, nicht aber, wie standardmäßig üblich, von PGP eine Datei mit dem unverschlüsselten Inhalt erstellt wird. Diese Technik wird bezeichnet als: „for your eyes only“ und erzeugt auf dem Bildschirm eine Ausgabe, die ähnlich wie der Filter *more* unter DOS und UNIX funktioniert. Die PGP-Option heißt auch entsprechend *m* und kann beim Verschlüsseln zusätzlich mitangegeben werden.

```
pgp -esm dateiname empfänger
```

[13] welches Verfahren benutzt wird, hängt von der PGP-Version ab.

Der Empfänger kann beim anschließenden Entschlüsseln den unkodierten Dateiinhalt nur am Bildschirm lesen. Will er die Mitteilung ein zweites Mal lesen, so muß er sie auch ein zweites Mal entschlüsseln.

Speichern erzwingen

Durch Umlenkung der Bildschirmausgabe auf eine Datei, kann vom Empfänger allerdings eine Ausgabe auf dem Rechner erzwungen werden. Das Kommando soll auch nur vor *unbeabsichtigtem* Speichern vertraulicher Nachrichten bewahren.

7.1.3 Entschlüsseln nur am Bildschirm

Die im vorigen Kapitel vorgestellte Option *m* kann auch beim Entschlüsseln von Nachrichten eingesetzt werden: Wenn Sie als Empfänger einer Nachricht auf jeden Fall ausschließen wollen, daß der Inhalt auf Ihrem Rechner abgelegt wird, so wählen Sie zum Entschlüsseln den Befehl

```
pgp -m dateiname
```

Damit wird der Nachrichtentext ausschließlich am Bildschirm aufgelistet.

7.2 Ändern der Schlüssel-Identifikations-Merkmale

Ändert sich Ihr Name oder Ihre Email-Adresse, möchten Sie möglicherweise Ihrem Schlüssel weitere alternative Mail-Adressen hinzufügen oder Ihren Passwort-Satz ändern, so können Sie das mit dem Befehl

```
pgp -ke eigener_schlüsselname
```

(key-edit).

Wichtig!

Der Name des Schlüssels muß auf jeden Fall Ihr eigener Name sein. PGP erkennt dann, daß dieser Schlüsselname sowohl am öffentlichen, als auch am privaten Schlüsselbund zu finden ist. Das gleiche Kommando ändert nämlich bei Angabe eines anderen Schlüsselbesitzers die zugehörigen Vertrauensparameter, aber dazu gleich mehr.

Ein Beispiel

```
c:\pgp>pgp -ke anna
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 15:10 GMT

Editing userid "anna" in key ring: 'c:\pgp\pubring.pgp'.

Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13

You need a pass phrase to unlock your RSA secret key.
Key for user ID "Anna Schmitt <anna.schmitt@fernuni-hagen.de>"

Enter pass phrase: Pass phrase is good.
Current user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
Do you want to add a new user ID (y/N)? y
Enter the new user ID: schmitt@manitu
```

```
Make this user ID the primary user ID for this key (y/N)? n
Do you want to change your pass phrase (y/N)? n
Secret key ring updated...
Public key ring updated.
```

Annas Schlüssel beinhaltet anschließend eine alternative Mail-Adresse. Das kann unter Umständen bei der Integration von PGP und Mail-Systemen notwendig werden. Näheres zu dieser Problematik lesen Sie im Kapitel 11.1 auf Seite 61. Der Schlüssel wird automatisch sowohl im öffentlichen, als auch im privaten Schlüsselbund aktualisiert.

Schauen wir uns nun noch Annas neue Schlüsselidentifikation an:

```
c:\pgp>pgp -kv schmitt
Key ring: 'c:\pgp\pubring.pgp', looking for user ID "schmitt".
Type bits/keyID    Date       User ID
pub   768/5F8D8B79 1996/05/13 Anna Schmitt <anna.schmitt@fernuni-hagen.de>
                              schmitt@manitu
1 matching key found.
```

Vertrauensparameter ändern

Geben Sie bei dem Befehl

```
pgp -ke schlüsselbesitzer
```

den Namen eines fremden Schlüssels an, so können Sie hiermit Ihre Vertrauensbekundungen ändern.

```
c:\pgp>pgp -ke bob
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 17:02 GMT

Editing userid "bob" in key ring: 'c:\pgp\pubring.pgp'.

Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 1CA3F065, created 1996/05/15

No secret key available.  Editing public key trust parameter.

Key for user ID: Bob Meyer <bob.meyer@fernuni-hagen.de>
768-bit key, Key ID 1CA3F065, created 1996/05/15
This key/userID association is fully certified.
  Axiomatically trusted certification from:
  Anna Schmitt <anna.schmitt@fernuni-hagen.de>
Current trust for this key's owner is: unknown

Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on available
evidence.  If you think it does, then based on your estimate of
that person's integrity and competence in key management, answer
the following question:

Would you trust "Bob Meyer <bob.meyer@fernuni-hagen.de>"
to act as an introducer and certify other people's public keys to you?
```

```
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?
2
Public key ring updated.
```

Mit Hilfe des obigen Befehls kann der Trust-Parameter, das heißt die Größe des Vertrauens in eine Unterschrift, festgelegt werden. Eine ausführliche Beschreibung des Trust Models finden Sie in dem Buch von William Stallings. Hier nur soviel: Wenn Sie Bob gut kennen und auch ganz sicher sind, daß Sie von ihm den korrekten Schlüsssel besitzen, dann müssen Sie aber dennoch nicht auch automatisch jedem von Bob unterschriebenen fremden Schlüssel Ihr Vertrauen schenken. Möglicherweise sind Sie ja sicher, daß Bob zwar ein sehr netter Kerl ist, aber trotzdem etwas großzügig mit seinen Schlüsselunterschriften umgeht. Dann können Sie PGP, wie im obigen Beispiel zu sehen, mit der Festlegung des Trustparameters **No** dazu bringen, Bobs Unterschrift unter fremden Schlüsseln einfach zu ignorieren.

7.3 ASCII-Nachrichten als Klartext unterschreiben

Normalerweise wird von PGP jede Nachricht, die von Ihnen digital unterschrieben wird, automatisch in einer binären Form abgespeichert. Sie wissen jedoch bereits, daß Sie mit Hilfe der Option **a** PGP dazu bringen können, nur ASCII-Daten zu verwenden.

```
c:\pgp>pgp  -sa  brief.txt
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 17:05 GMT

A secret key is required to make a signature.
You specified no user ID to select your secret key,
so the default user ID and key will be the most recently
added key on your secret keyring.

You need a pass phrase to unlock your RSA secret key.
Key for user ID "schmitt@manitu"

Enter pass phrase: Pass phrase is good.
Key for user ID: Anna Schmitt <anna.schmitt@fernuni-hagen.de>
768-bit key, Key ID 5F8D8B79, created 1996/05/13
Also known as: schmitt@manitu
Just a moment....
Transport armor file: brief.asc
```

Schauen Sie sich den obigen Befehl einmal genau an: die Datei BRIEF.TXT wird lediglich unterschrieben und als ASCII-Datei abgelegt. Da *keinerlei Verschlüsselung* stattfindet, wird auch keine Schlüsselangabe des Empfängers der Datei benötigt. Doch wie sieht die Datei BRIEF.ASC. schließlich aus?

```
-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

owHrZChlZmUwnKOg2tgskBrf213JyKhnw/RXbhObVdnZj88n+z3KrH5Tkjwx4HIj
w3mVPLaOOr87meGG2zbUFpta/S/P6LspoeSj2t2SeXjmSjXWWWeY3bbc7KtZrveH
8w2PxuHaj4ats7eJsonUfluc2/u5yMRsWle0ZcS/zWs8kziTijJT0/RKKkoYgMAn
MzUpVcExLy9Rh5crJTO1WCGzuEQhNTNPITsHSKYWKTiBlCuklQKZKZnJGbxcvFxO
+Um8XAA=
=1Gt3
-----END PGP MESSAGE-----
```

Unleserlich! Und das obwohl lediglich unterschrieben und nicht verschlüsselt wurde. Warum das so ist, erfahren Sie im Kapitel 9 auf Seite 49.

Wenn Sie aber nun eine unterschriebene Nachricht zum Beispiel in die News setzen möchten, um sicherzustellen, daß sie beim Transport nicht verändert wird, dann können nur die News-Teilnehmer Ihre Nachricht entschlüsseln, die auch über PGP verfügen. Das ist nicht so schön.

Es gibt aber natürlich eine Möglichkeit, eine Unterschrift so an eine Nachricht anzuhängen, daß der Text auch leserlich bleibt. Dazu benötigen Sie zwei zusätzliche PGP Angaben:

1. die Option **t** (Textmode=on), um PGP mitzuteilen, daß Sie eine ASCII- und keine binäre Datei verwenden.
2. die Option **+clearsig=on**, damit der Klartext Ihrer Nachricht erhalten bleibt und lediglich eine Unterschrift hinzugefügt wird.

Nach Absetzen des Befehls

```
pgp -sat +clearsig=on brief.txt
```

erhalten Sie die Datei

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

Liebe Benutzer und Benutzerinnen
ich melde mich hiermit aus dem Urlaub zurück

Anna

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3i

iQB1AwUBMZxn94GDEGVfjYt5AQG8uwL/a6JRhTJdPGcMAyNxSMjdDB+Y8q5kFtbu
4iKfTvlQsgBG+QWgm3Wl4j8vTxULuipldKivFkTeA3ZxrW5nsl8xj0HdUz4RWJdQ
PrlDqc8H5gk/lqjAX+h6KpaUa/1Du0mk
=lsvG
-----END PGP SIGNATURE-----
```

Nun kann wirklich jeder die obige Nachricht lesen. Diejenigen, die über Annas öffentlichen Schlüssel verfügen, können zusätzlich mit PGP die Unter-

schrift überprüfen. Sollte irgendjemand zwischenzeitlich die obige Datei verändert haben, so erhält man beim Aufruf von PGP folgende Meldung:

```
c:\pgp>pgp  brief
Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the masses.
(c) 1990-1996 Philip Zimmermann, Phil's Pretty Good Software. 1996-01-14
International version - not for use in the USA. Does not use RSAREF.
Current time: 1996/05/13 17:13 GMT

File has signature.  Public key is required to check signature.
WARNING: Bad signature, doesn't match file contents!

Bad signature from user "Anna Schmitt <anna.schmitt@fernuni-hagen.de>".
Signature made 1996/05/17 11:50 GMT

Plaintext filename: brief
```

Die verfälschte Ausgabedatei BRIEF wird trotzdem erstellt, ihr Inhalt sollte aber mit Skepsis betrachtet werden.

8 PGP installieren und konfigurieren

8.1 PGP unter MS-DOS

Wenn Sie PGP unter MS-DOS installieren wollen, so können Sie die Software PGP2.6.3I[14] von unserem FTP-Server ziehen; sie befindet sich dort unter `/pub/pgp/pgp263i.zip`. Es handelt sich hierbei um ein komprimiertes Format. Zum „Auspacken" der eigentlichen Dateien benötigen Sie zusätzlich das Programm PKUNZIP, das sich ebenfalls auf unserem FTP-Server im Verzeichnis `/pub/dos/archiver` befindet.

- Legen Sie sich zunächst ein passendes Verzeichnis an, z.B. `c:\pgp`, in dem Sie ihre Software installieren möchten.
- Dekomprimieren Sie die Datei `PGP263I.ZIP` z.B. mit

  ```
  pkunzip -d pgp263i
  ```

- Es werden automatisch die Dateien `PGP263II.ZIP` und `PGP263II.ASC` erstellt. Die zweite der beiden Dateien enthält Signaturen, damit Sie die Software auf unverfälschte Übermittlung überprüfen können. Wie das geschieht, lesen Sie auf der nächsten Seite im Kästchen für Fortgeschrittene.
- Packen Sie nun die Datei `PGP263II.ZIP` auf die gleiche Weise aus.
- Als nächstes müssen Sie MS-DOS Umgebungsvariablen besetzen. Damit Sie diese Arbeiten nicht ständig wiederholen müssen, sollten Sie die Variablen in Ihrer AUTOEXEC.BAT belegen:

  ```
  SET PGPPATH=C:\PGP
  SET PATH=%PATH%;C:\PGP
  SET TZ=MET-1DST
  SET TMP=C:\WINDOWS\TMP
  ```

  Die Variable `PGPPATH` muß auf das Verzeichnis verweisen, indem sich u.a. Ihre PGP-Software und Ihre Schlüsselbunde befinden. Die `PATH`-Variable sollten Sie um das PGP-Verzeichnis erweitern, damit PGP beim Aufruf automatisch gefunden wird. Die `TZ`-Variable besetzen Sie mit der für uns gültigen, oben angegebenen Zeitzone.
- Anschließend können Sie Ihr erstes Schlüsselpaar generieren.

[14] Dies ist zum Zeitpunkt der Broschürenerstellung die aktuelle PGP-Version, die zur Benutzung außerhalb der USA gültig ist.

Für Fortgeschrittene

Sobald Sie mit der PGP-Arbeit vertraut geworden sind, sollten Sie auf jeden Fall nachträglich Ihre PGP-Version verifizieren; und zwar möglichst, bevor Sie Ihren Schlüssel ausgetauscht haben.

Zusammen mit der PGP-Software haben Sie eine Datei `KEYS.ASC` erhalten. Hier sind u.a. einige Schlüssel aufgeführt, die den Mitwirkenden an der PGP-Software gehören. Diese Schlüssel benötigen Sie, wenn Sie Ihre PGP-Version verifizieren möchten. Wenn Sie die Schlüssel nicht an Ihrem normalen Schlüsselbund eintragen möchten,können Sie auch einen alternativen Schlüsselbund verwenden, den Sie folgendermaßen erzeugen können:

```
pgp    -ka  keys.asc    keys.pgp
```

Anschließend geben Sie den Befehl

```
pgp   pgp263ii.asc
```

ein. Wenn PGP die benötigten Schlüssel an Ihrem normalen Schlüsselbund nicht findet, erhalten Sie jetzt eine Aufforderung, den Namen eines anderen Bundes anzugeben:

```
keys.pgp
```

Sie werden darüber informiert, daß diese Datei zwar eine Unterschrift trägt, aber über keinen Text verfügt. Geben Sie deshalb anschließend den Namen der ZIP-Datei an:

```
pgp263ii.zip
```

Sie sollten daraufhin die Meldung:

```
Good Signature from:

Stale Schumacher <staalesc@ifi.uio.no>
```

erhalten.

8.2 PGP unter Windows 95

Die Installation unter Windows 95 verläuft analog zur Installation unter MS-DOS, kann jedoch hier wahlweise auch über die komfortable Menuoberfläche erfolgen. Die Umgebungsvariable `TMP` müssen Sie eventuell mit

```
SET TMP=C:\WIN95\TMP
```

belegen.

8.3 PGP unter UNIX

Wenn Sie eine UNIX-Installation vornehmen wollen, so sollten Sie sich hierzu die ausführliche Beschreibung in der Datei `SETUP.DOC` anschauen, die zusammen mit der PGP-Software verteilt wird. Die Software finden Sie auf unserem FTP-Server im Verzeichnis `/pub/pgp/pgp263i.tar.gz`.

Nachdem Sie die Dateien entpackt haben, müssen Sie noch das `makefile` für Ihr spezielles UNIX-System ausführen.

Auch hier sollten Sie anschließend auf jedem Fall dem Tip für Fortgeschrittene folgen, der bei der MS-DOS-Installation empfohlen wird, geben dann aber bitte statt den Namen der ZIP-Datei, den Namen des TAR-Files an.

8.4 PGP auf dem Publikumsrechner Bonsai

Wenn Sie die bereits installierte PGP-Version auf unserem Publikumsrechner Bonsai verwenden wollen, so sollten Sie folgendermaßen vorgehen:

- Legen Sie sich unter ihrem Heimatdirectory ein Verzeichnis, z.B. `~/pgp` an.
- Als nächstes müssen Sie UNIX-Umgebungsvariablen besetzen. Damit Sie diese Arbeiten nicht ständig wiederholen müssen, sollten Sie die Variablen in Ihrer `.profile`-Datei belegen:

```
export pgppath=~/pgp
export path=$PATH:/usr/local/bin
export tz=met-1dst
```

- Die Variable `PGPPATH` muß auf das Verzeichnis verweisen, indem sich später u.a. Ihre Schlüsselbunde befinden. Die `PATH`-Variable sollten Sie, falls das nicht schon voreingestellt ist, um das Verzeichnis `/usr/local/bin` erweitern, da sich dort die PGP-Software befindet. Die `TZ`-Variable besetzen Sie mit der für uns gültigen, oben angegebenen Zeitzone.

Anschließend können Sie PGP verwenden, sollten jedoch unbedingt darauf achten, daß das PGP-Verzeichnis mit Ihren Schlüsselbunden von *keinem, außer Ihnen selbst* gelesen werden kann.

Achtung: Verzeichnis schützen!

8.5 PGP konfigurieren

Sobald Sie PGP erfolgreich installiert haben, können Sie sich daran begeben, gewisse Voreinstellungen vorzunehmen. Zusammen mit der PGP-Software haben Sie zwei Dateien erhalten:

`language.txt`

`config.txt`

In der Datei `language.txt` befinden sich die Meldungen und Fehlermeldungen, die bei der Benutzung von PGP am Bildschirm erscheinen in englischer, deutscher, spanischer und französischer Sprache. Die Voreinstellung, welche Sprache benutzt werden soll, ist in der Datei `config.txt` festgelegt. Standardmäßig ist die englische Sprache eingestellt[15]. Auch die Hilfestellungen und Befehlsübersichten, die mit dem Kommando

`language.text`

```
pgp -h
```

[15] Da die Übersetzung in die deutsche Sprache nach meiner ganz persönlichen Meinung ein wenig „gewöhnungsbedürftig“ ausgefallen ist, würde ich die Verwendung der englischen Sprache bevorzugen.

angezeigt werden, erscheinen in der ausgewählten Sprache.

config.txt

In der Datei `config.txt` können diverse weitere Voreinstellungen eingetragen werden. Der folgende Ausschnitt zeigt Ihnen einen Teil der `config.txt`. Leerzeilen und Zeilen die mit # beginnen werden von PGP ignoriert. Das # wird deshalb als Kommentarzeichen verwendet. Durch Entfernen oder Hinzufügen des Kommentarzeichens können diverse Voreinstellungen außer Kraft gesetzt oder eingestellt werden.

```
# Sample config.txt file for PGP 2.6.3i.
# Blank lines are ignored, as is anything following a '#'.
# Keywords are not case-sensitive.
# Whatever appears in here can be overridden on the command line,
# by specifying (for example) "+armor=on".
#.....
MyName = "Anna Schmitt"

# The language we will be using for displaying messages to the user.
#.....
Language = en
#.....
Armor = on      # Use -a flag for ASCII armor whenever applicable
TextMode = on   # Attempt to use -t option where applicable
ClearSig = on   # Use unarmored plaintext for unencrypted signed message
#.....
# Verbose = 2         # verbose diagnostic messages
# Verbose = 0         # be quiet
#.....
# ShowPass = on       # Echo Passwort when user types it

# BakRing is the path to a backup copy of your secret keyring, usually
# on floppy disk.  Your secret keys will be compared with the backup
# copy when doing a keyring check (pgp -kc)
BakRing = "a:\secring.pgp"

# Number of completely trusted signatures needed to make a key valid.
Completes_Needed = 1

# Number of marginally trusted signatures needed to make a key valid.
Marginals_Needed = 2

# How many levels of introducers may introduce other introducers.
Cert_Depth = 2

# Paths to keyrings and seed file for random generator.
PubRing = "c:\pgp\pubring.pgp"
SecRing = "c:\pgp\secring.pgp"
RandSeed = "c:\pgp\randseed.bin"
```

Die obige Datei ist gekürzt. Die wichtigsten eingestellten Werte sind in fetter Schrift dargestellt.

MyName Standardmäßig verwendet PGP für digitale Unterschriften automatischen den ersten Schlüssel am privaten Schlüsselbund.

Arbeiten Sie mit mehreren geheimen Schlüsseln, so können Sie bei den verschiedenen PGP-Befehlen die Option `-u` verwenden, um einen anderen als den ersten Schlüssel zur Signatur zu verwenden. Der Parameter

MyName = "Anna Schmitt"

legt alternativ Annas geheimen Schlüssel fest, der automatisch verwendet wird, wenn die Option `-u` nicht angegeben ist.

Language Sämtliche Fragen, Warnungen und Erläuterungen, die von PGP am Bildschirm ausgegeben werden, erscheinen in der hier eingestellten Sprache. Durch Festlegung der Sprache auf

Language = de

erscheinen diese Texte in deutscher Sprache.

Armor Um eine Datei so zu verschlüsseln, daß sie emailfähig ist, können Sie den PGP-Befehlen die Option `-a` hinzufügen, die für eine Verschlüsselung im Radix-64-Format sorgt, das nur aus ASCII-Zeichen besteht. Sollten Sie jedoch überwiegend ASCII-Dateien benötigen, so können Sie diesen Parameter in der Konfigurationsdatei permanent einstellen:

Armor = on

sorgt dafür, daß Dateien auch ohne die Option `-a` im Radix-64-Format erzeugt werden.

TextMode Dieser Parameter ist äquivalent zur PGP-Option `-t`, mit der Sie festlegen können, daß die zu verschlüsselnden Dateien ASCII-Text sind. Insbesondere beim Erstellen von Klartext-Unterschriften wird diese Option benötigt.

ClearSig Wollen Sie Text nicht verschlüsseln, sondern lediglich unterschreiben, so können Sie PGP mit diesem Parameter anweisen, den eigentlichen Nachrichten-Text lesbar zu belassen. Genaueres über diese Funktion können Sie bei Bedarf noch einmal im Kapitel 7.3 auf Seite 38 nachlesen.

Verbose Hiermit können Sie die Ausführlichkeit der Meldungen festlegen. Mögliche Werte sind 0, 1 oder 2. Bei der Einstellung 2 können Sie insbesondere den Exit-Code von PGP ausgeben lassen, was zum Beispiel bei Programmaufrufen interessant sein kann. (Standardeinstellung: 1)

ShowPass Aus Sicherheitsgründen wird der Passwort-Satz bei Eingabe durch den Benutzer am Bildschirm nicht angezeigt. Und das sollten Sie nach Möglichkeit auch so belassen. Nur wenn Sie die allergrößten Schwierigkeiten haben, Ihr Passwort „blind“ einzutippen, empfiehlt es sich, diesen Parameter einzustellen. (Standardeinstellung: off)

BakRing In regelmäßigen Abständen sollten Sie Ihren eigenen öffentlichen Schlüssel auf Echtheit überprüfen; insbesondere dann, wenn Sie PGP auf einem vernetzten oder Mehrbenutzersystem verwenden. Speichern Sie dazu Ihren geheimen Schlüssel, also

die Datei `secring.pgp` möglichst auf einem für Fremde unzugänglichen Medium, z.B. einer schreibgeschützten Diskette. Mit dem Befehl

```
pgp -kc
```

können Sie Ihren gesamten öffentlichen Schlüsselbund überprüfen. Ist der Parameter **Bakring** mit einem Wert belegt, so wird gleichzeitig ein Vergleich mit Ihrer Sicherungskopie des geheimen Schlüssels durchgeführt. (Standardeinstellung: "")

Completes_Needed/Marginals_Needed

Diese beiden Parameter ermöglichen die Einstellung des Vertrauensgrades, den PGP einem Schlüssel zubilligt. In unserem Beispiel genügen eine voll vertrauenswürde bzw. zwei teilweise vertrauenswürdige Unterschriften unter einem Schlüssel, um diesen vollständig als echt zu bestätigen. (Standardeinstellungen: wie im Beispiel eingetragen)

Cert_Depth PGP kann dazu veranlasst werden, bis zu einer vorgegebenen Verschachtelungstiefe von unterschriebenen Schlüsseln, diese als echt anzuerkennen.

Setzen Sie diesen Parameter beispielsweise auf den Wert 1, so wird ein Schlüssel nur dann als echt anerkannt und voll bestätigt, wenn er von einer Person unterschrieben ist, deren Schlüssel Sie selbst am Schlüsselbund tragen und unterschrieben haben. Bobs Schlüssel gilt also als echt, wenn er von Anna unterschrieben ist und Sie selbst Annas Schlüssel unterschrieben haben.

Beim Wert 2 würde es, wie im obigen Beispiel, genügen, wenn Bobs Schlüssel von Charlie unterschrieben ist, Charlies Schlüssel die Unterschrift von Anna trägt und Sie selbst wiederum Annas Schlüssel unterschrieben haben.

Der Parameter kann mit Werten zwischen 0 und 8 belegt werden. (Standardeinstellung: 4)

PubRing / SecRing

Der Ablageplatz Ihrer beiden Schlüsselbunde ist normalerweise das Verzeichnis, auf das der `PGPPATH` verweist. Standardmäßig sind die Namen der Dateien `pubring.pgp` und `secring.pgp.` Diese Voreinstellung kann mit diesen beiden Parametern verändert werden.

RandSeed Bei der Erzeugung Ihres Schlüsselpaares werden Sie aufgefordert, zur Erstellung von Zufallszahlen einige Tastenanschläge vorzunehmen. Diese Sammlung von Zufallszahlen wird von PGP als Datei `randseed.bin` in verschlüsselter Form zur weiteren Verwendung abgelegt. Standardmäßig wird diese Datei ebenfalls in dem Verzeichnis erwartet, auf das der `PGPPATH` verweist. Mit diesem Parameter ändern Sie die Voreinstellung.

Alternativ können die Parameter der Datei `config.txt` auch beim PGP-Aufruf direkt in der Kommandozeile angegeben werden. Das empfiehlt sich besonders dann, wenn eine Einstellung nur im Ausnahmefall verwendet werden soll. Benutzen Sie einfach den gewünschten Parameternamen zusammen mit dem einzustellenden Wert und einem vorangestellten +-Zeichen beim PGP-Aufruf.

Beispiele:

```
pgp -e +armor=on   mail.txt      anna

pgp -sat +clearsig=on  brief.txt
```

9 Ein paar Details zur PGP-Verschlüsselung

Dieses Kapitel ist für Ihre tagtägliche Verschlüsselungarbeit mit PGP sicherlich nicht wichtig. Aber falls Sie sich für ein paar Details interessieren, die PGP bei den unterschiedlichen Verschlüsselungs-/Unterschriftskombinationen verwendet, so sind Sie hier genau richtig. Für eine exakte Beschreibung der mathematischen Verfahren, muß ich Sie allerdings wieder auf die Literatur verweisen.

drei Verschlüsselungsalgorithmen

PGP verwendet nicht nur einen Verschlüsselungsalgorithmus, sondern drei. Diese sind bekannt unter den Namen: RSA, MD5 und IDEA. Wann welcher Algorithmus aus welchen Gründen eingesetzt wird, möchte dieses Kapitel kurz vorstellen. Dazu zunächst eine kurze Erläuterung zu den Algorithmen:

RSA

RSA ist ein Verfahren, zur Erstellung asymmetrischer Schlüssel. Es wurde benannt nach seinen Entwicklern Ron *R*ivest, Adi *S*hamir und Len *A*dleman und von diesem am MIT entwickelt. MIT wurde daraufhin ein Patent erteilt, dessen alleinige Rechte für den Verkauf und die Lizensierung die kalifornische Firma Public Key Partners (PKP) besitzt. Dieses Patent gilt nur innerhalb der USA.

IDEA

IDEA ist ein symmetrisches Verschlüsselungsverfahren, das sowohl zum Verschlüsseln, als auch zum Entschlüsseln lediglich einen einzigen Schlüssel verwendet. Dieses Verfahren ist in Europa patentiert und gehört den Firmen ETH Zürich und Ascom Tech AG. Eine nicht-kommerzielle Verwendung dieses Algorithmus ist allerdings erlaubt und unterliegt keinerlei Lizenzgebühren.

MD5

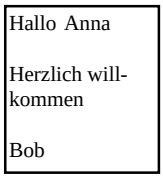
MD5 (Message Digest) ist ein Algorithmus, der ebenfalls von Ron Rivest entwickelt wurde. Er erzeugt aus einer Nachricht einen 128-Bit Hash-Code. Ein Hash-Code ist eine Zahl, die nach einem vorgegebenen Verfahren aus einer Nachricht berechnet wird. Solange die Nachricht nicht verändert wird, ergibt sich immer wieder der gleiche Hash-Code. MD5 ist als Public Domain freigegeben.

Kombination der Verfahren

Die oben aufgeführten Algorithmen werden von PGP automatisch in einer sinnvollen Kombination verwendet, die ich Ihnen gleich noch vorstellen möchte. Hinzu kommt noch ein Verfahren zur Komprimierung von Daten, das ZIP-Programm, bei dem es sich um Freeware handelt, die mit Erlaubnis des Autors frei verteilt werden darf. Ein weiteres Verfahren zur Erstellung von mailfähigen Dateien ist die Radix-64-Konversion, die ja in der Broschüre schon mehrfach erwähnt wurde.

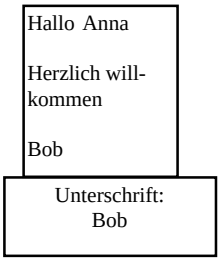
PGP 's Arbeitsschritte

Die einzelnen Arbeitsschritte sollen nun anhand der nachfolgenden Bilder dargestellt werden. Die Idee zu dieser übersichtlichen Darstellung stammt übrigens von William Stallings.



- Bob erstellt eine Nachricht für Anna.
- Er verwendet anschließend den PGP-Befehl:

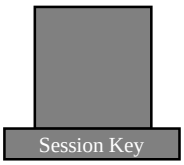
```
pgp -esa mail.txt Anna
```



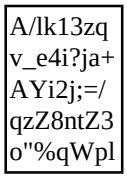
- Die Datei wird daraufhin zunächst mit Bobs Unterschrift versehen:
  - mit MD5 wird ein Hash-Code aus der Nachricht generiert,
  - der Hash-Code wird mit RSA verschlüsselt, unter Verwendung von Bobs privatem Schlüssel.
  - Dieser verschlüsselte Hash-Code ist die digitale Unterschrift von Bob und wird der Datei hinzugefügt.

- Die gesamte Datei, also Text und Unterschrift, werden automatisch mit ZIP komprimiert.

- Die komprimierte Datei wird nun verschlüsselt:
  - Dazu wird eine Zufallszahl ermittelt[16], die als sogenannter Session Key dient.
  - Diesen Session Key verwendet IDEA zur Verschlüsselung der Datei.

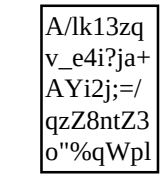


- Da der Empfänger den Session Key zum Entschlüsseln auch benötigt, wird er der Datei hinzugefügt, allerdings mit RSA verschlüsselt, unter Verwendung von Annas öffentlichem Schlüssel.
- Der Session-Key wird für jede Verschlüsselung neu ermittelt.



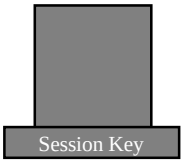
- Zur Emailfähigkeit werden schließlich die binären Daten in druckbare Zeichen, also ASCII-Zeichen umgewandelt.
- Diese Datei gelangt nun zum Empfänger.

[16] entweder wird eine neue berechnet, oder, falls vorhanden, auf den Pool von Zufallszahlen in der Datei `randseed.bin` zugegriffen.



- Anna erhält Bobs Mail und entschlüsselt sie mit

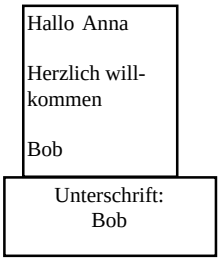
  ```
  pgp mail
  ```



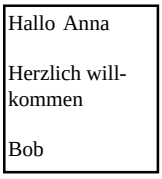
- Aus den ASCII-Zeichen werden die originalen Binärzeichen wiederhergestellt.

- Der Session-Key wird ermittelt und mit dem geheimen Schlüssel von Anna entschlüsselt (RSA).

- Mit diesem Session-Key wird schließlich die Datei entschlüsselt (IDEA).



- Dann wird der Text dekomprimiert (ZIP)



- Da die Datei Bobs Unterschrift trägt, wird diese mit Bobs öffentlichen Schlüssel entschlüsselt. Dazu wird zunächst mit
  - RSA der der Message beigefügte Hash-Code dekodiert und ein
  - neuer Hash-Code aus der übertragenen Datei berechnet (MD5)
  - Wenn beide übereinstimmen, wurde die Datei nicht verändert.

Für jeden einzelnen Schritt wird also ein geeignetes Verschlüsselungsverfahren ausgewählt. Vielleicht ist Ihnen die scheinbar komplizierte Kombination von RSA und IDEA aufgefallen und sie fragen sich, weshalb der Nachrichtentext nicht einfach komplett mit RSA verschlüsselt wird. Theoretisch ist das sicherlich möglich, jedoch ist eine RSA-Verschlüsselung ziemlich langsam. Das würde für umfangreiche Texte nicht zumutbare Wartezeiten bedeuten. Deshalb wird für die eigentliche Verschlüsselung der Nachricht das überaus sichere und schnelle IDEA-Verfahren verwendet. Da es sich um ein konventionelles Verfahren handelt, muß auf jedenfall ein Austausch des Schlüssels stattfinden und der erfolgt nach dem zuvor beschriebenen Muster.

Die PGP Versionen

Insbesondere der von PGP verwendete RSA-Algorithmus, dessen Patent das Massachussets Institute of Technology (MIT) besitzt und dessen einzige Lizenz im Besitz der Firma PKP ist, führte 1994 dazu, daß einige rechtliche Fragen zur Legalität und Illegalität von verschiedenen PGP-Versionen aufkamen:

PGP 2.3 ist die „klassische“ PGP-Version; in der Version PGP 2.3a wurde ein Fehler korrigiert. Diese Version verwendet das RSA-Verfahren und verstößt damit bei Verwendung innerhalb der USA gegen das Patentrecht. Außerhalb der USA ist eine Verwendung völlig legal.

PGP 2.6.2 ist die aktuelle offizielle Freeware-Version zur Benutzung innerhalb der USA. Sie verwendet die RSAREF-Library, die von der RSA Data Security, Inc. zur nicht-kommerziellen Nutzung frei zur Verfügung gestellt wird und verletzt damit nicht das Patentrecht. Ein Export in andere Länder ist verboten.

PGP 2.7 ist die kommerzielle Version von ViaCrypt für den Gebrauch innerhalb der USA.

PGP 2.6.3i ist die momentan aktuelle internationale Version, die von Stale Schumacher in Norwegen veröffentlicht wurde. Sie verwendet nicht die RSA-Library, sondern eine andere Implementation des RSA-Algorithmus, genannt MPILIB. Diese Version wird außerhalb der USA verwendet, entspricht aber in Ihrer Funktionalität der amerikanischen Version.

PGP 3.0 ist die Version, auf die zur Zeit mit großer Spannung gewartet wird. Größere Veränderungen und insbesondere eine weitaus höhere Geschwindigkeit bei großen Schlüsselbunden sind in Aussicht gestellt. Wann mit der neuen Version zu rechnen ist, darüber wird momentan lediglich spekuliert. Warten wir es ab ...

10 PGP- Frontends

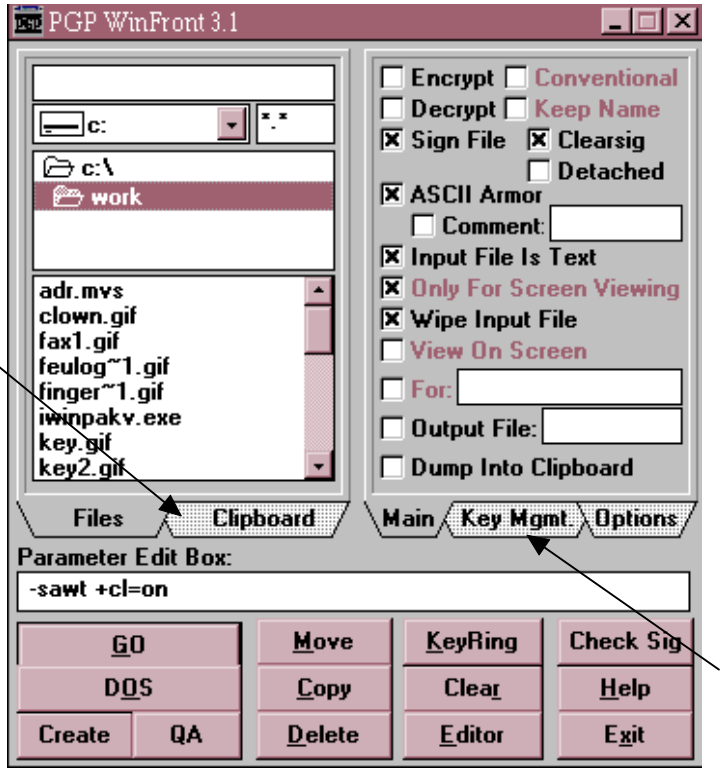
10.1 PGP unter Windows

Da inzwischen der Trend eindeutig zu fenstergestützten Systemen geht, wurde schnell der Wunsch nach einer Oberfläche laut, die eine einfache PGP - Benutzung unter Windows gestattet. Diese Programme erleichtern hauptsächlich die Auswahl der benötigten PGP-Parameter, stellen den PGP-Befehl mit den richtigen Optionen zusammen und führen ihn anschließend in einem DOS-Fenster aus.

Eine sehr komfortable Benutzeroberfläche für Windows 3.1[17] stellt das Freeware-Programm PGP WinFront, Version 3.1 zur Verfügung.

Es baut, ganz nach Ihren Wünschen, die Sie durch Ankreuzen aussprechen, den PGP-Befehl auf und führt ihn in einem DOS-Fenster aus. Nach Anklicken des zugehörigen Icons kann folgendes Fenster für PGP benutzt werden:

Das nebenstehende Menu bietet Ihnen drei Abschnitte:

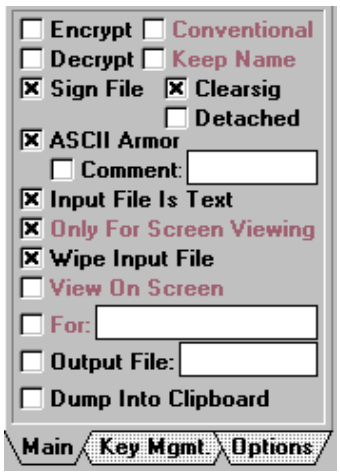


Im linken Fensterteil haben Sie die Möglichkeit, Dateinamen zur Verschlüsselung auszuwählen, bzw auf die Registerkarte `Clipboard` zu klicken, um den dort vorhandenen Inhalt für weitere Aktionen zu verwenden[18].

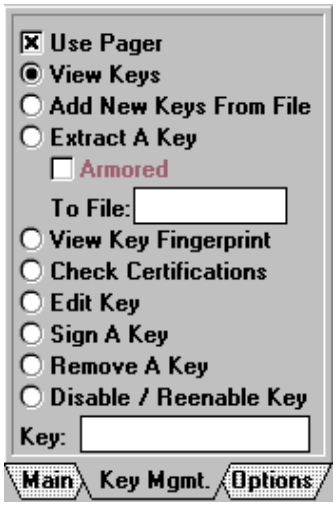
Im rechten Fensterteil können Sie durch Ankreuzen die einzelnen PGP-Optionen auf zwei von den drei vorhandenen Registerkarten zusammenstellen.

[17] Diese Freeware funktioniert weitestgehend auch unter Windows 95.

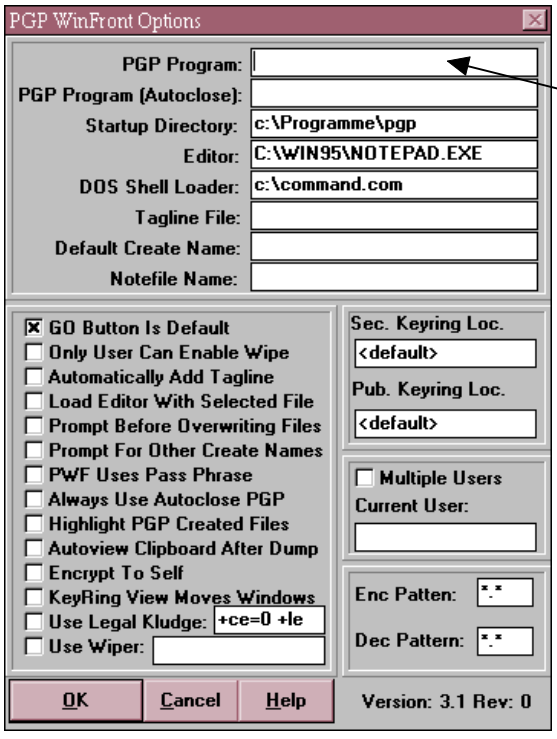
[18] Ein View auf die Zwischenablage muß allerdings unter Windows 95 eventuell nachkonfiguriert werden. Das Verzeichnis für das Clipboard muß sich auf jedenfall im Pfad befinden.



Auf der Registkarte `Main` können Sie für die ausgewählte Datei oder den Clipboard-Inhalt eine PGP-Aktion starten. Das Ergebnis können Sie ebenfalls wieder in eine Datei oder in die Zwischenablage stellen. Das ist insbesondere für den Austausch mit anderen Programmen, z.B. für Emails, News und WWW-Anwendungen praktisch.



Die Registerkarte `Key Mgmt.` stellt Ihnen ein Hilfmittel zur Schlüsselverwaltung zur Verfügung. Durch „Ankreuzen“ können Sie Ihren Schlüsselbund anlisten lassen, Schlüssel unterschreiben, editieren, hinzufügen usw.



Die dritte Registerkarte ermöglicht Ihnen die Konfigurierung von PGP WinFront, und sollte als einer der ersten Schritte ausgefüllt werden:

Geben Sie hier mindestens das Verzeichnis an, in dem sich Ihr PGP-Programm befindet und füllen Sie den Rest nach Bedarf aus

Je nachdem, welche Kreuzchen Sie auf den unterschiedlichen Registerkarten gesetzt haben, wird in der *Parameter Edit Box* angezeigt, wie der PGP-Befehl automatisch zusammengesetzt wird.

Im unteren Teil des Fensters steuern Sie weitere Funktionen:

Der Button **GO**: führt den zuvor zusammengestellten PGP-Befehl in einem DOS-Fenster aus.

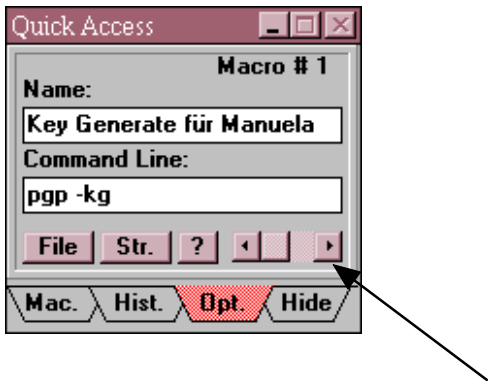
Der Button **DOS**: zeigt ein DOS-Fenster.

Der Button **CREATE**: erzeugt mit dem voreingestellten Editor eine neue Datei, deren Namen Sie zuvor festlegen müssen.

Der Button **QA** führt Sie in ein Fenster, in dem unter anderem die PGP-Befehle protokolliert werden und in dem Sie eigene häufig benötigte PGP-Befehle selbst einfügen können.

Zum Beispiel gibt es normalerweise keine Möglichkeit, im PGP-WinFront ein neues Schlüsselpaar zu erzeugen. Mit dem **QA**-Button können Sie den Befehl selbst hinzufügen:

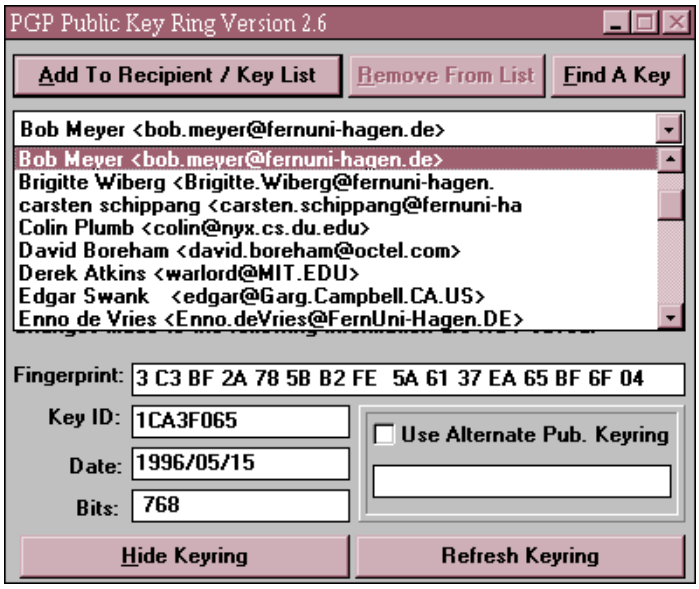
Wählen Sie zunächst die Registerkarte `Opt.` Anfangs sind die beiden Felder Name: und Command Line: leer. Tragen Sie hier einen beliebigen Namen und zum Beispiel den PGP-Befehl, der ein neues Schlüsselpaar erzeugt, ein. Klicken Sie dann auf den Pfeil nach rechts.



Wählen sie anschließend die Registerkarte `Mac.` aus. Hier befindet sich jetzt Ihr neues Kommando zur Erstellung eines Schlüsselpaares. Auf diese Weise können Sie PGP Win-Front um weitere häufig von Ihnen benötigte PGP-Befehle erweitern.

Die Buttons **MOVE, COPY, DELETE**: verschieben, kopieren bzw. löschen die zuvor ausgewählte Datei.

Der Button **KEYRING**: dieser Knopf ist sehr hilfreich: er zeigt Ihnen für alle verfügbaren Schlüssel weitere Informationen an, insbesondere den Fingerprint.



Durch Anklikken des Buttons **ADD TO RECIPIENT / KEY LIST** können Sie mehrere Schlüssel in eine Empfänger-Liste aufnehmen[19], für die dann anschließend Dateien verschlüsselt werden können. Zusätzlich können Sie auch nach ausgewählten Schlüsseln suchen. Der Knopf **HIDE KEYRING** „versteckt“ die Schlüsselanzeige wieder.

Der Button **CLEAR**: leert den Parameterbereich.

Der Button **EDITOR**: ruft den voreingestellten Editor mit einer leeren Datei auf.

Der Button **CHECKSIG**: kann verwendet werden, wenn eine sogenannte „Detached Signatur“ geprüft werden soll.

Die Buttons **HELP**/**EXIT**: führen in ein Hilfesystem, bzw. verlassen die PGP WinFront-Umgebung.

PGP WinFront gibt es inzwischen in neueren Versionen, die auch speziell für Windows 95 gedacht sind. Es handelt sich dabei jedoch um Shareware, für die ein geringes Entgeld an den Software-Autor gezahlt werden sollte. Auf unserem FTP-Server finden Sie lediglich die Freeware-Version PGP WinFront 3.1.

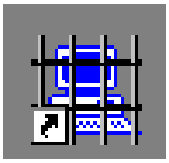
10.2 PGP unter Windows 95

Unter Windows 95 existieren momentan vielfältige unterschiedliche PGP-Frontends. Bei den meisten handelt es sich um kostenpflichtige Shareware-Produkte, wobei der Preis in der Regel lediglich in Höhe von ungefähr 50$ angegeben wird. Die Entwicklung dieser Oberflächen scheint zum aktuellen Zeitpunkt noch nicht abgeschlossen zu sein, denn ständig erscheinen neue Versionsnummern zu den einzelnen Produkten.

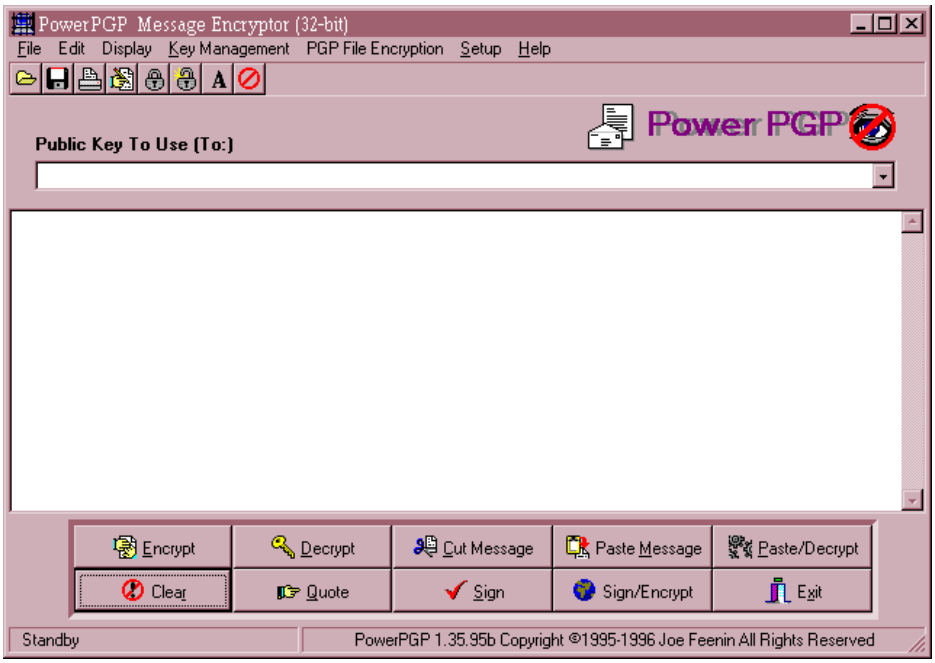
Eine gute Wahl treffen Sie, wenn Sie auch unter Windows 95 das bereits im vorigen Kapitel beschriebene PGP Winfront 3.1 verwenden. Sie können sich jedoch auch im Internet umsehen und eine für Sie adequate Lösung suchen. Unser Favorit ist momentan PowerPGP, eins der wenigen Freeware-

[19] Das ist allerdings nur dann möglich, wenn Sie die Registerkarte `Main` eingestellt haben. Die Registerkarte `Key Mgmt.` ermöglicht lediglich die Auswahl *eines* Schlüssels.

Angebote, das ich Ihnen hier kurz vorstellen möchte. PowerPGP finden Sie ebenfalls auf unserem FTP-Server.

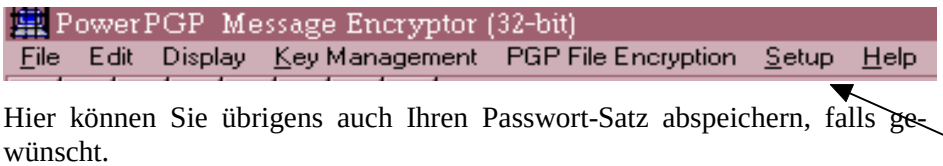


Nach Installation und Aufruf der Software über das nebenstehende Icon erscheint folgendes Fenster:



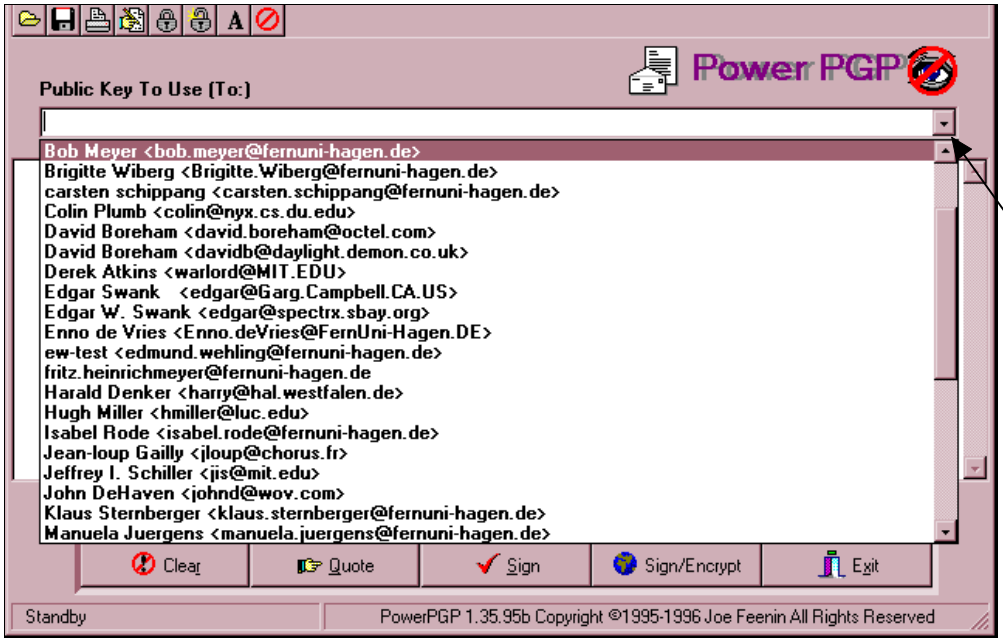
PowerPGP ist, wie die meisten PGP-Frontends so gestaltet, daß eine Nachricht, die in das Textfenster eingegeben wird, mit Hilfe von Menu-Punkten und Buttons z.B. verschlüsselt und signiert werden kann. Die kodierte Nachricht wird dann durch einfachen Tastendruck in das Clipboard gebracht und kann durch jede beliebe Anwendung, sei es Pegasus für Mails oder FreeAgent für News, von dort mit Hilfe des `Edit`-Befehls der Menu-Leiste und der anschließenden Auswahl von `Paste,` in die Anwendung kopiert werden. Das hört sich zunächst umständlich an, ist aber in der Handhabung recht einfach. Zur Erläuterung dieser Funktion wird im nachfolgenden Beispiel Pegasus verwendet.

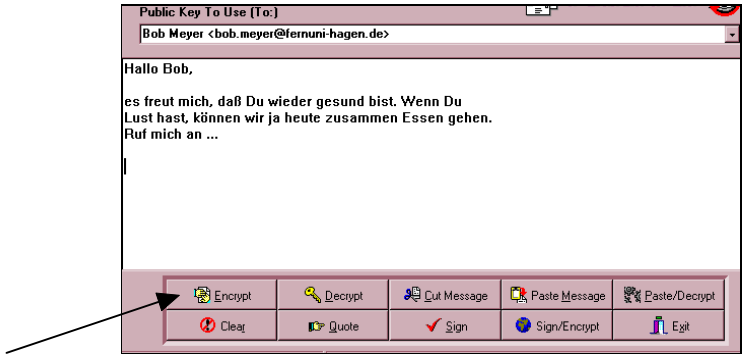
Zunächst aber sollten Sie PowerPGP mit Hilfe des `Setup`-Befehls der Menu-Leiste konfigurieren, indem Sie dort den richtigen Pfad zu Ihrem PGP eintragen.



Hier können Sie übrigens auch Ihren Passwort-Satz abspeichern, falls gewünscht.

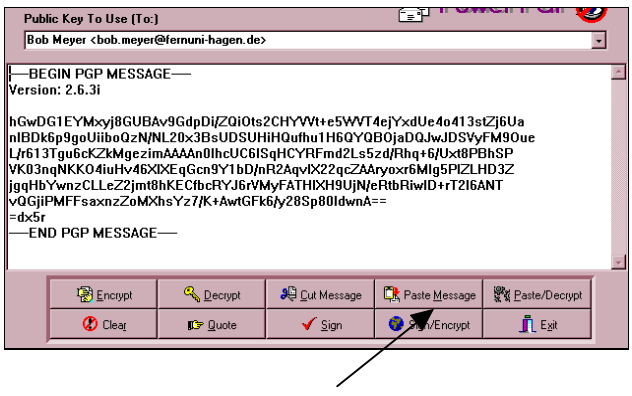
Anschließend können Sie sich Ihren Schlüsselbund im PowerPGP durch Klicken auf den rechtsstehenden Pfeil ansehen und diejenigen Schlüssel auswählen, die Sie zur Kodierung verwenden möchten.



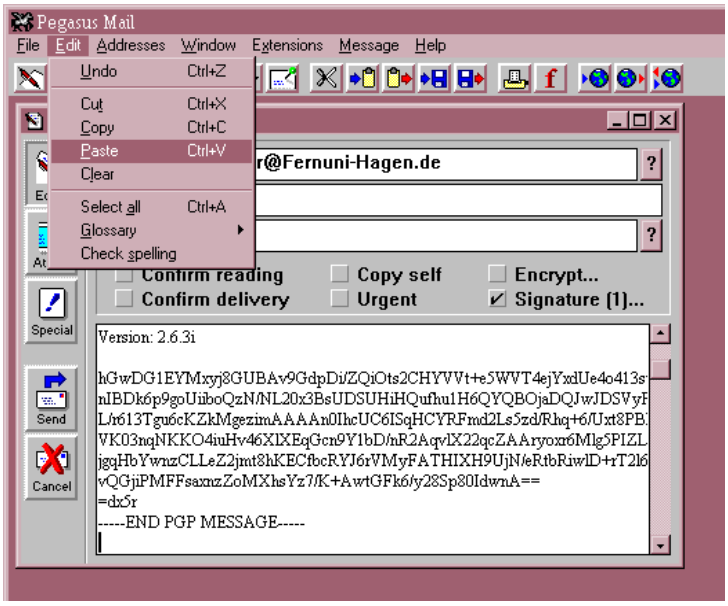


Angenommen, Sie wollen Bob eine verschlüsselte Nachricht schicken. Wählen Sie dann zunächst Bobs öffentlichen Schlüssel aus und schreiben Sie anschließend den Message-Text in den Nachrichten-Bereich.

Klicken Sie schließlich auf den **ENCRYPT**-Button, wird in einer DOS-Box am Bildschirm der PGP-Lauf durchgeführt. Das verschlüsselte Ergebnis erscheint im Textbereich.



Klicken Sie nun noch auf den Button **PASTE MESSAGE**; der Nachrichteninhalt verschwindet am Bildschirm und gelangt stattdessen in das Clipboard.



Rufen Sie nun zum Beispiel Pegasus oder ein beliebiges anderes Mail-System auf und wählen Sie dort den Befehl `Edit Paste`, mit dessen Hilfe Sie den Inhalt des Clipboards in den Textbereich von Pegasus einfügen können. Positionieren Sie zuvor aber unbedingt den Cursor in dem Textbereich!

Füllen Sie alle weiteren vom Mail-System benötigten Informationen aus und schicken Sie das ganze an den Empfänger ab.

Angenommen, Bob antwortet Ihnen ebenfalls verschlüsselt. Sie erhalten dann zum Beispiel die Message mit Pegasus, wählen den Befehl `Edit Select All` und anschließend `Edit Copy`. Die verschlüsselte Antwort befindet sich wieder in dem Clipboard. Wechseln Sie nun noch zurück ins PowerPGP, wählen dort den Button **PASTE/DECRYPT** und schon steht der entschlüsselte Nachrichtentext in dem Messagefenster.

Die weiteren Buttons im unteren Teil des Fensters sind sicherlich selberklärend. Alle Befehle, die über Buttons angesteuert werden, können gleichzeitig auch aus der Menuleiste heraus aufgerufen werden.

Die Menuleiste von PowerPGP bietet aber noch weitere Funktionen: Sie können zum Beispiel komfortabel Ihre Schlüsselbunde verwalten und statt einer Verschlüsselung von direkt eingegebenen Nachrichten, auch eine Kodierung für Dateien durchführen.

Weitere Hilfestellungen können Sie der Programmbeschreibung entnehmen, die Sie nach Anklicken des **HELP**-Buttons am Bildschirm erhalten.

11 Integration in Mail-Systeme

Die bisher beschriebene Vorgehensweise verschlüsselt lediglich Ihre Dateien bzw. den Inhalt des Clipboards mit PGP. Um die kodierten Daten dann zu verschicken, müssen Sie in einem zweiten Schritt ein Mail-System benutzen und die Daten mittels der Zwischenablage austauschen. Die direkte Integration beider Systeme ist bisher teilweise zumindest für einige Mail-Systeme realisiert worden.

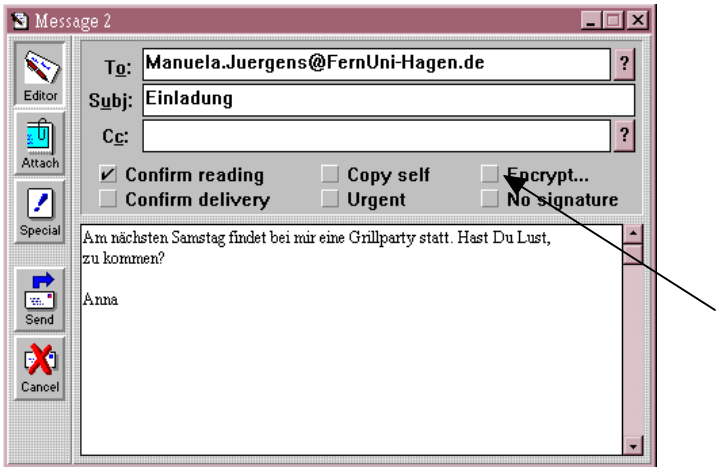
11.1 Pegasus unter Windows 3.1 und Windows 95

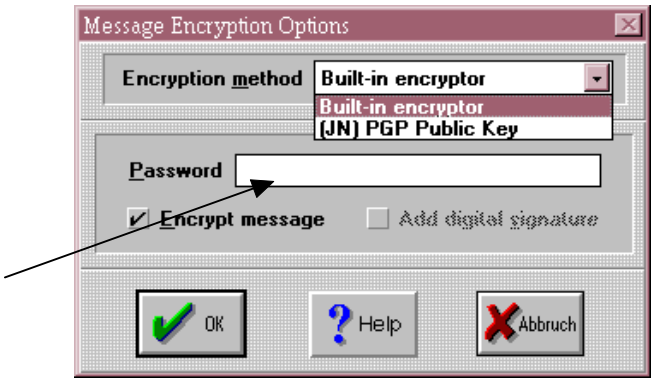
Pegasus ist eine sehr komfortable Mail-Oberfläche, die als Freeware verfügbar ist. Ab der Version 2.23 existiert eine PGP-Schnittstelle PGPJN, benannt nach dem Autor John Navas. Nach korrekter Installation dieser Zusatzsoftware, erkennt Pegasus zusätzlich zum internen Verschlüsselungsalgorithmus auch PGP. PGPJN finden Sie einschließlich einer Installationsanleitung auf unterem FTP-Server. Beachten Sie jedoch, daß PGPJN in der aktuellen Version nicht mit dem neusten 32-Bit-Pegasus für Windows 95 zusammen arbeitet. Sofern Sie sich jedoch mit der Pegasus Version 2.23 unter Win 95 begnügen, dürften Sie keinerlei Probleme haben. Einschränkungen, die sich durch Verwendung von PGPJN für Pegasus und PGP ergeben, entnehmen Sie bitte der beigefügten Dokumentation.

Vorausgesetzt, Sie haben sowohl Pegasus, PGP, als auch PGPJN korrekt installiert, so können Sie anschließend problemlos eine Verschlüsselung für Nachrichtentexte durchführen.

Rufen Sie zunächst Pegasus auf und erstellen Sie im Message-Fenster die gewünschte Nachricht.

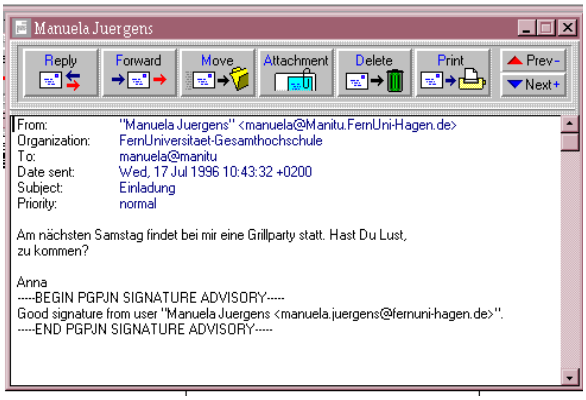
Klicken Sie anschließend auf den Button **ENCRYPT** und Sie gelangen automatisch in ein Fenster, in dem Sie nun den gewünschten Verschlüsselungsalgorithmus auswählen können





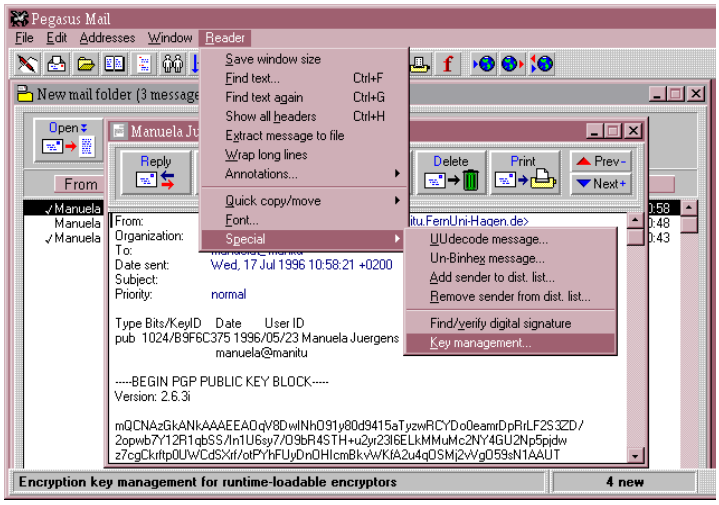
Entscheiden Sie sich für PGPJN, so können sie die Message verschlüsseln und / oder signieren. Sollten Sie sich zusätzlich für eine digitale Unterschrift entschließen, so müssen Sie *unbedingt auch gleichzeitig Ihren Passwort-Satz hier eintagen*. Klicken Sie auf **OK**, so gelangen Sie wieder in das Nachrichten-Fenster zurück. Sie werden hier allerdings von der Verschlüsselung nichts sehen. Diese wird nämlich erst durchgeführt, wenn Sie die Message absenden.

Ganz wichtig ist: *Die Mailadresse, die Sie angegeben haben, muß in dieser Form auch an Ihrem Schlüsselbund auftauchen. PGPJN sucht nämlich genau nach dem Begriff an Ihrem Keyring, den Sie als Adressaten eingetragen haben. Unter Umständen wird es dadurch erforderlich. daß Ihre Kommunikationspartner unter Alias-Namen am Schlüsselbund bekannt sind.*



Sobald Sie eine verschlüsselte Mail erhalten und diese im Mailfenster zweimal anklicken, erscheint ein weiteres Feld, in das Sie Ihren Passwort-Satz eintragen. Der Mailinhalt wird anschließend unverschlüsselt und eventuell mit Signatur angezeigt. Bei jedem erneuten Lesen, wird natürlich auch wieder die Eingabe Ihres Passwort-Satzes erforderlich.

Wenn Sie den öffentlichen Schlüssel eines Gesprächspartners zugeschickt bekommen, so können sie ihn über Pegasus an Ihren Schlüsselbund anhängen.



Öffnen Sie dazu einfach die Mail, in der sich der Schlüssel befindet und wählen Sie anschließend auf der Menuleiste den Befehl `Reader Special Key Management...`

Sobald Sie das Key Management auswählen, wird Ihnen der mitgelieferte Schlüssel angezeigt und Sie können ihn an den Keyring hängen

PGPJN finden Sie ebenfalls auf unserem FTP-Server.

11.2 Private Idaho für Windows 95

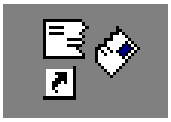
11.2.1 Direktes Verschicken/Empfangen von Mails

Private Idaho ist ein System, das Ihnen zwei Vorteile bietet:

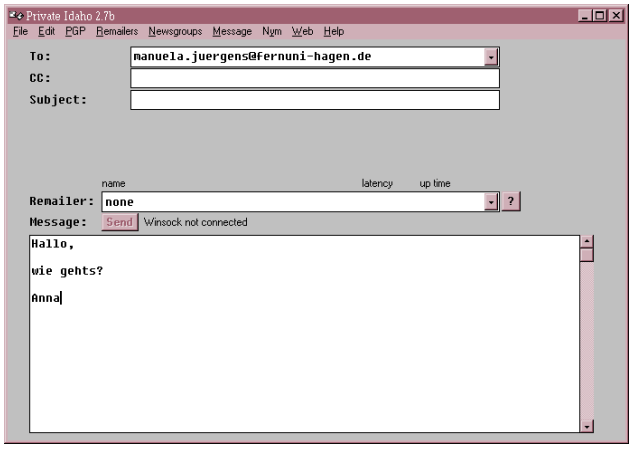
1. zunächst einmal können Sie mit diesem System direkt mit PGP verschlüsselte Mails verschicken und empfangen,
2. und zusätzlich können Sie einen sehr komfortablen Austausch von verschlüsselten Texten mit anderen Systemen, wie zum Beispiel dem News-System vornehmen.

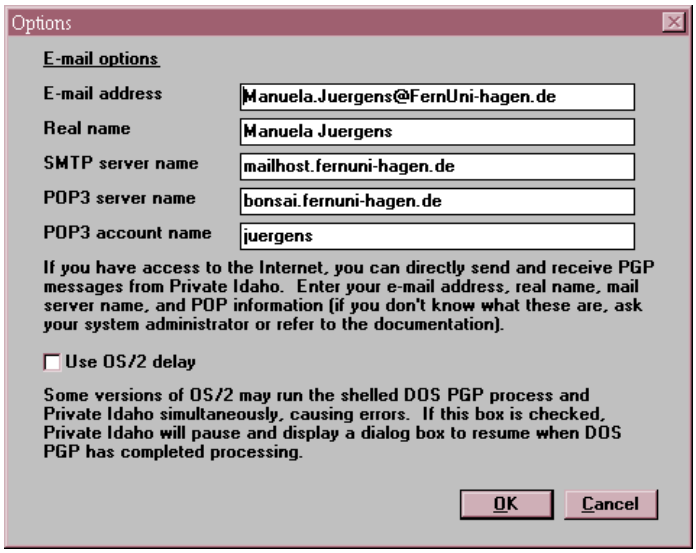
Private Idaho ist Freeware und befindet sich zur Zeit in der Version 2.7b auf unserem FTP-Server.

Wenn Sie Mails direkt mit Private Idaho verschicken möchten, so genügt es, einfach das entsprechende Icon anzuklikken. Sie erhalten daraufhin das folgende Fenster:



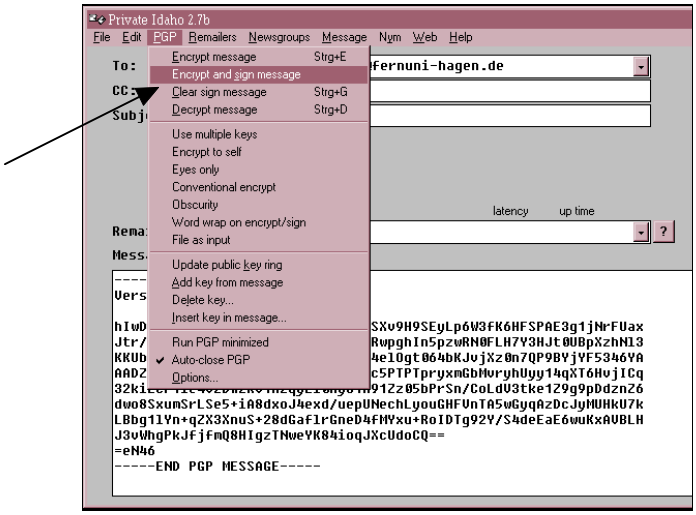
Bevor Sie nun die Adresse des Empfängers und den Nachrichtentext eintragen, sollten Sie zunächst den Befehl `File` aus der Menuleiste anwählen und die Eintragung `User` anklicken. Private Idaho wird dann automatisch in den `Expert`-Modus versetzt, der Ihnen erweiterte Möglichkeiten bietet.





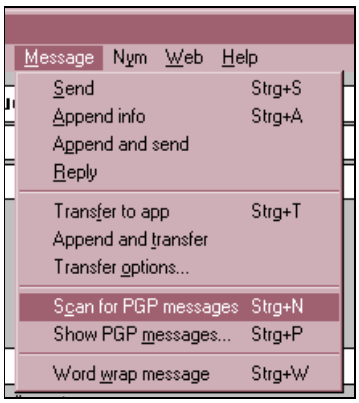
Danach sollten Sie noch die Konfiguration Ihrer Mailumgebung festlegen. Wählen Sie dafür den Befehl `File Options` und füllen Sie die Felder wie benötigt aus.

Anschließend müssen Sie noch die Befehlsfolge `File Connect` auswählen, so daß Sie über den Button **SEND** Ihre Mails verschicken können.



Um Ihre Mail zu kodieren und zu signieren, benötigen Sie den Befehl `PGP` aus der Menuleiste. Füllen Sie auch hier zunächst die `Options` aus, um eventuell benötigte Voreinstellungen einzutragen. Danach können Sie dann die gewünschte Verschlüsselung durchführen: Die verschlüsselte Mail erscheint im Nachrichtenfenster und kann auf den Weg zum Empfänger geschickt werden.

Neben dem **SEND**-Button können Sie die korrekte Übertragung Ihrer Nachricht verfolgen.



Über den Befehl `Message Scan for PGP messages` können können Sie sich die Mails ansehen, die PGP-Verschlüsselungen oder Signaturen enthalten. Diese erscheinen in einem eigenen Fenster, das jederzeit durch den Befehl `Show PGP messages` wieder angezeigt werden kann.

Durch Doppelklick auf die entsprechende Nachricht wird PGP automatisch gestartet und das unverschlüsselte Ergebnis am Bildschirm angezeigt.

11.2.2 Austausch von Daten mit anderen Anwendungen

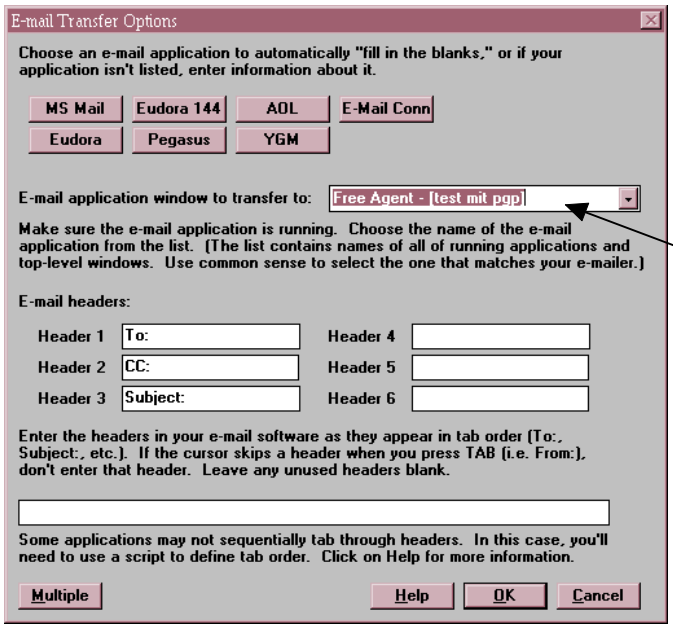
Private Idaho ist nicht nur in der Lage, verschlüsselte und signierte Mails zu verschicken und zu empfangen, sondern kann auch Daten für andere Anwendungen vorbereiten und mit diesen austauschen.

Ganz wichtig: Sie sollten immer zuerst die Anwendung starten, in der Sie die kodierten Daten benötigen und erst dann Private Idaho aufrufen.

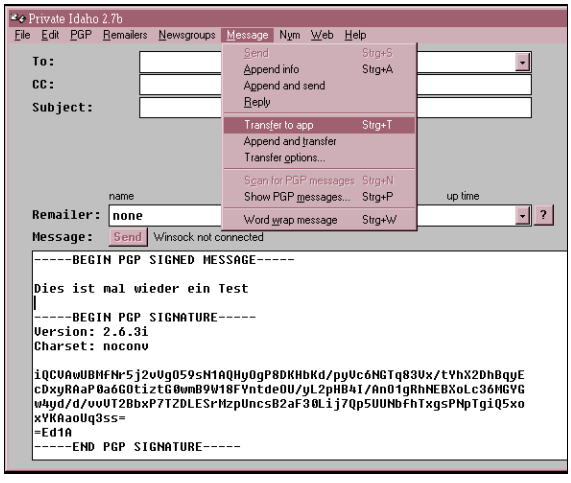
Sehen wir uns das an einem kleinen Beispiel an: Anna möchte eine signierte Mitteilung ins News-System setzen. Sie verwendet dazu FreeAgent und startet diesen zunächst.

Anna wählt die Gruppe `feu.test`, um die Verschlüsselung zuerst zu testen und wählt den Befehl `Post New Article` aus der Menuleiste.

Spätestens jetzt ruft Sie auch Private Idaho auf. Hier muß zunächst der Befehl `Message Transfer Options` ausgewählt werden, um den Datenaustausch mit FreeAgent vorzubereiten. In dem sich öffnenden Fenster trägt Anna nun den FreeAgent als Anwendung ihrer Wahl ein, so daß später ein einfacher Datenaustausch vorgenommen werden kann.



In das Message-Fenster von Private Idaho schreibt Anna Ihre Test-News, signiert den Beitrag anschließend über `PGP Clear Sign Message`, gibt auf Anfrage Ihren Passwort-Satz ein und schon erscheint der unterschriebene Text im Private Idaho Message Fenster. Über den Befehl `Message Transfer to App ...` wird der signierte Text direkt in den im Hintergrund schlummernden FreeAgent übertragen und kann von dort sofort gesendet werden.



Auf die gleiche bequeme Art und Weise können verschlüsselte Daten an beliebige Anwendungen weitergereicht werden.

Private Idaho bietet noch viele weitere Möglichkeiten, die Sie bitte der der Software beigefügten Dokumentation entnehmen.

11.3 Emacs als Mail-System

Emacs ist zwar ein Editor, trotzdem können Sie ihn zum Verschicken von Mails einspannen. Mit

`ESC+x mail`	zum Erstellen und Verschicken von Mail und mit
`ESC+x rmail`	zum Empfangen und Lesen von Mail

schalten Sie in den sogenannten Mail-Modus um. Wenn Sie anschließend die PGP-Umgebung laden möchten, so geben Sie ein:

```
ESC+x load-lib mailcrypt
```

Ihnen stehen nun eine Reihe von typischen PGP-Befehlen zur Verfügung, z.B. zum Entschlüsseln, Verschlüsseln, Signieren und desweiteren mehr. Mit

```
ESC+x mailcrypt - TAB-Taste
```

erhalten Sie eine Übersicht über die möglichen PGP-Befehle.

11.4 Elm

Mit Hilfe von Elm, einem weit verbreiteten Email-System unter UNIX, können Sie Ihre Mails ganz automatisch ver- und entschlüsseln und zwar nachdem Sie folgende Schritte ausgeführt haben:

1. Rufen Sie Elm, falls Sie noch nie damit gearbeitet haben einmal auf, damit die benötigten Verzeichnisse und Dateien automatisch für Sie angelegt werden.
2. Wechseln Sie in das Verzeichnis `~/.elm`.
3. Erstellen Sie hier, falls noch nicht vorhanden, die Datei elmrc, mit den Einträgen
   ```
   editor=/usr/local/bin/mailpgp
   pager=/usr/local/bin/morepgp
   ```
4. Bei neu erstellten Mails stellt Elm Ihnen anschließend automatisch Fragen, ob Verschlüsselung und Signatur gewünscht wird.
5. Bei eintreffenden Mails findet beim Ansehen der Nachricht eine automatische Entschlüsselung statt.

11.5 Weitere Mail-Systeme

Weitere Mail-Systeme, wie zum Beispiel MH, PRIVTOOL und EUDORA verfügen über Schnittstellen zu PGP oder sogar über integrierte PGP-Funktion. Einige davon werden eventuell zu einem späteren Zeitpunkt installiert. Wir werden Sie dann entsprechend informieren. Falls Sie sich aber jetzt schon dafür interessieren, werden Sie jederzeit im Internet fündig.

12 Key-Server

Zur einfachen Verteilung von öffentlichen Schlüsseln existieren eine Reihe von Key-Servern. Der Key-Server für Deutschland befindet sich an der Universität in Hamburg. Aber: dieser Service ist natürlich keine Garantie für die Glaubwürdigkeit eines Schlüssels. Er erleichtert lediglich die Bekanntgabe Ihres Schlüssels und den Erhalt von Schlüsseln, mit deren Besitzer Sie kommunizieren möchten.

Das Arbeiten mit einem Key-Server erfolgt über das Verschicken von Mails. Das Kommando mit dem Sie den Server nutzen möchten, schreiben Sie als Subject in Ihre Mail. Wollen Sie zu allererst einige Hilfestellungen für die Serverbenutzung in Erfahrung bringen, so versenden Sie folgende Mail:[20]

```
To:  pgp-public-keys@informatik.uni-hamburg.de
From:  manuela.juergens@fernuni-hagen.de
Subject: help
```

Sie erhalten Informationen über alle möglichen weiteren Subject-Angaben.

Möchten Sie auf diese Art Ihren Schlüssel veröffentlichen, so schicken Sie eine Mail der Art:

```
To:  pgp-public-keys@informatik.uni-hamburg.de
From:  manuela.juergens@fernuni-hagen.de
Subject: add

Type Bits/KeyID    Date    User ID
pub  1024/B9F6C375 1996/05/23 Manuela Juergens <manuela.juergens@fernuni-hagen.de>
manuela@manitu
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i
mQCNAzGkANkAAAEEAOqV8DwINhO91y80d9415aTyzwRCYDo0eamrDpRrLF2S3ZD/
2opwb7Y12R1qbSS/In1U6sy7/O9bR4STH+u2yr23I6ELkMMuMc2NY4GU2Np5pjdw
z7cgCkrftp0UWCdSXrf/otPYhFUyDnOHIcmBkvWKfA2u4qOSMj2vVgO59sN1AAUT
tDRNYW51ZWxhIEp1ZXJnZW5zIDxtYW51ZWxhLmp1ZXJnZW5zQGZlcm51bmktaGFn
ZW4uZGU+iQCVAwUQMaRd2Wa3ZBkKnjf5AQFYpgP/aN0BIjV5SlcRxlKR89FXEYCO
Fa+a6eLN895qn4CHkyLdAdpPzreWihNYal3PuZnxaDUNv0eNFxx6tBeUU8vM3lIc
UTwxgvK57dFGvgP8DZD00u8DU+9IX1k8hyNdQkcdFrzdzGZh65qKzWOaislVqwQn
3GEFLTvmkf/vtGLGyXeJAJUDBRAxpAF7Pa9WA7n2w3UBAddyA/43z1YSQI/znCWH BCAN-
jFijB5rRU3XSCTW2AseluQeJdE0Wiadu/yQjUHTacOIOh1J3aOZzWEIJAIo3
JJVVdAZ4toTfz8+GWY5g2w+lt7dAV4U9YTwYPiTLN4iRJtoFgnR0zWwt/gJqSCpp
O0nmd+LfGzov/bPkjH4jr3wjSHObfLQObWFudWVsYUBtYW5pdHU=
=Q2r0
-----END PGP PUBLIC KEY BLOCK-----
```

[20] Als Absender hinter **From:** benutzen Sie natürlich bitte Ihre eigene Userid.

Es genügt, den Schlüssel an einen Key-Server zu senden; er wird dann automatisch an die anderen Server weitergeleitet.

Mögliche Aufträge sind unter anderem:

`add`	Hinzufügen von Schlüsseln
`index`	Anlisten aller Schlüssel (-kv)
`get userid`	Kopieren des Schlüssels von userid
`get`	Kopieren des gesamten Schlüsselbundes (Vorsicht: der kann sehr groß sein!)
`mget ausdruck`	Kopieren aller Schlüssel, die dem angegebenen Aus druck entsprechen

13 Sachregister

S

T

U

V

W

Z

14 Zusammenfassung einiger PGP-Kommandos

Schlüsselverwaltung

pgp -kg	erzeugt ein neues Schlüsselpaar (Key Generate)
pgp -ka datei [schlüsselbund]	hängt den neuen Schlüssel aus der angegebenen Datei an den Schlüsselbund. (Key Add)
pgp -kr [benutzer] [schlüsselbund]	hängt den Schlüssel von `benutzer` vom Schlüsselbund ab (Key Remove)
pgp -kv [benutzer] [schlüsselbund]	listet alle oder den angegebenen Schlüssel auf (Key View)
pgp -kvv [benutzer] [schlüsselbund]	listet alle oder den angegebenen Schlüssel einschließlich der Signaturen auf (Key View Verbose)
pgp -kc [benutzer] [schlüsselbund]	listet u.a. alle oder den angegebenen Schlüssel auf und überprüft die Signaturen auf Glaubwürdigkeit (Key Check)
pgp -kvc [benutzer] [schlüsselbund]	listet alle oder den angegebenen Schlüssel einschließlich des Fingerprints auf (Key View and Check)
pgp -ks [benutzer] [schlüsselbund]	unterschreibt den angegebenen Schlüssel (Key Sign)
pgp -krs benutzer [schlüsselbund]	entfernt die Unterschrift vom Schlüssel (Key Remove Signature)
pgp -kx [benutzer] [datei] [schlüsselbund]	speichert den gewünschten Schlüssel in einer Datei ab (Key Extract)
pgp -kxa benutzer datei [schlüsselbund]	wie `-kx`, allerdings wird der Schlüssel nur mit ASCII-Zeichen abgespeichert (Key Extract ASCII)
pgp -kd benutzer [schlüsselbund]	der angegebene Schlüssel wird gesperrt (Key Disable)

Verschlüsselung/Signaturen

pgp -c datei	die angegebenen Datei wird konventionell verschlüsselt (Crypt)
pgp -e datei benutzer	verschlüsselt die angegebene Datei für den gewünschten Benutzer (Encrypt)
pgp -ew datei benutzer	wie `-e`, nur wird die Originaldatei anschließend automatisch gelöscht (Encrypt and Wipe)
pgp -s datei [-u mein_schlüssel]	unterschreiben einer Datei (Sign)
pgp -es datei benutzer [-u mein_schlüssel]	verschlüsseln und unterschreiben einer Datei für den angegebenen Benutzer (Encrypt and Sign)
pgp -esa datei benutzer [-u mein_schlüssel]	wie `-es`, allerdings wird die Datei im ASCII-Format abgelegt (Encrypt and Sign with ASCII)
pgp -sat datei [-u mein_schlüssel]	unterschreiben einer lesbaren Datei, also ohne Verschlüsselung (Sign with ASCII and Text)

Entschlüsselung

pgp verschlüsselte_datei [-o entschlüsselt]	entschlüsselt die angegebene Datei und speichert sie in der angegebenen Ausgabedatei ab. Ist die verschlüsselte Datei gleichzeitig unterschrieben, so wird automatisch die Signatur überprüft
pgp -m verschlüsselte_datei	die Ausgabe der entschlüsselten Datei wird lediglich auf dem Bildschirm vorgenommen (More)

Hilfestellungen

pgp -h	zeigt eine Zusammenfassung der PGP-Befehle am Bildschirm an (Help)
pgp -k	zeigt die KeyManagement-Befehle an