RECHNERNETZE

Security Tools 4: Pretty Good Privacy

Bernd Lehle / Oliver Reutter

Was gibt's Neues im BelWü?

Peter Merdian / Paul Christ

Security Tools 4: Pretty Good Privacy

Bernd Lehle / Oliver Reutter

Beim Umgang mit E-Mail (elektronischer Post) mag bei vielen Benutzern das Bild von der Gelben Post im Hinterkopf auftauchen: Man schreibt einen Brief, steckt ihn in einen Umschlag, übergibt ihn dem verantwortlichen System und das sorgt dann dafür, daß er unversehrt beim Empfänger ankommt, der ihn öffnet und liest. In der Bundesrepublik Deutschland ist das Postgeheimnis von der Verfassung garantiert, und darf nur in speziellen Fällen und nach richterlichem Beschluß gebrochen werden. In der Welt der elektronischen Postzustellung sieht es leider etwas anders aus: Eine E-Mail Message ist nicht geheimer als eine Urlaubspostkarte und der eigene Briefkasten nicht sicherer als der eigene Mülleimer. Was man tun kann, um sein Postgeheimnis hier soweit zu sichern, daß nicht einmal der Staatsanwalt herankommt, beschreibt dieser Artikel.

Jeder Benutzer, der sich in dem Metier auskennt, wird sofort wissen, daß hier nur Verschlüsselung hilft. Wer heute im Internet Daten transportieren will, die nicht jeder lesen soll, macht sich erst gar keine Gedanken, ob die Leitungen sicher sinc

oder nicht: Er geht von unsicheren Leitungen aus und verschlüsselt seine Daten.

Diese Disziplin war bis vor einigen Jahren noch den hehren Kreisen der Mathematiker und Geheimdienste vorbehalten. Mit dem Auftauchen von Public Key-Verfahren und einfacher Software, die sie schnell, portabel und bedienerfreundlich implementiert, kann heute allerdings jeder seine Daten so verschlüsseln, daß sich selbst die Supercomputer der amerikanischen Oberschnüffler NSA (National Security Agency) daran die Zähne ausbeißen würden.

Der Prophet dieser neuen Privatsphäre im Internet ist der Amerikaner Phil Zimmermann, der das Programm PGP (Pretty Good Privacy) entwickelte, das heute auf allen gängigen Plattformen public domain erhältlich ist.

Wir wollen hier erst einen kleinen Einblick in die Grundlagen der Public Key-Verschlüsselung geben und dann am Beispiel von PGP die Benutzung detailliert darstellen.

Das von PGP verwendete Public Key-Verfahren heißt RSA, benannt nach seinen Entwicklern Ron Rivest, Adi Shamir und Len Adleman (MIT, 1977).

Die Grundlage von RSA bilden zwei sehr großen Primzahlen *p* und *q*. Zudem benötigt man noch den Encryption Key *e*, der eine beliebige Zahl sein kann, die kein Teiler von

$$(p-1)\cdot(q-1)$$

ist. Typischerweise ist es eine kleine Primzahl, aus der dann der Decryption Key *d* wie folgt über den Euklidischen Algorithmus berechnet wird:

$$d\cdot e = 1 \bmod ((p-1)\cdot(q-1))$$

So hängt also *d* mit *e* über *p* und *q* zusammen. Aus diesen Zahlen berechnet man dann das Produkt *n* = *pq*; *p* und *q* werden dann vergessen.

Verschlüsselt wird eine Zahl (oder ein Buchstabe) *m* nun mit: $c = m^e \bmod n$

Entschlüsselt wird mit: $m = c^d \bmod n$

Dies heißt, was mit *e* verschlüsselt wurde, kann nur mit *d* wieder entschlüsselt werden und umgekehrt. Man spricht - im Gegensatz zu den symmetrischen, die für Ver-/Entschlüsselung denselben Schlüssel benutzen - von einem asymmetrischen Verfahren. Die Kombinationen (*e*,*n*) und (*d*,*n*) sind die Schlüssel. Der Einwand, daß *d* jederzeit aus *e* wieder berechnet werden kann greift nur, wenn *p* und *q* bekannt sind. Die Schlüssel enthalten aber nur deren Produkt *n* und damit ist das

Knacken des Schlüssels nur über die Faktorisierung des Produkts *n* möglich, was bei genügend großen Primzahlen selbst auf Supercomputern Monate bis Jahre dauern kann.

Beim Public Key-Verfahren wird einer der beiden Schlüssel *(e,n)* oder *(d,n)* veröffentlicht und allen Kommunikationspartnern zur Verfügung gestellt. Der andere wird als ge-heimer Schlüssel sicher verwahrt. Hierbei wird die Tatsache ausgenutzt, daß beide Schlüssel nur kombiniert ver-/entschlüsseln können. Um jemandem eine geheime Botschaft zukommen zu lassen, verschlüsselt man sie mi dem öffentlichen Schlüssel des Partners. Der einzige Schlüssel, der dies nun wieder entschlüsseln kann ist der private des Empfängers. Die einmal verschlüsselte Nachricht kann man selbst nicht wieder entschlüsseln.

Neben dem Verschlüsseln bietet dieses Verfahren auch die Möglichkeit der digitalen Unterschrift. Dazu wird aus dem zu unterschreibenden Textdokument eine Prüfsumme gebildet und mit dem privaten Schlüssel verschlüsselt; sie wird an den Text gehängt und mitverschickt. Der Empfänger kann nun mit dem öffentlichen Schlüssel des Absenders die Unterschrift entschlüsseln und die erhaltene Prüfsumme mit einer selbst be-rechneten vergleichen. Wurde die Nachricht verändert, stimmen die Prüfsummen nicht mehr überein. Will ein Fälscher die Nachricht *und* die Unterschrift fälschen, braucht er dazu den privaten Schlüssel des Absenders, da ohne ihn aus der gefälschten Prüfsumme keine gültige Unterschrift erzeugt werden kann. Die Prüfsummen werden mit dem sogenannten MD 5-Verfahren erstellt, das es nicht zuläßt die Prüfsumme gezielt zu verändern.

Wenn man davon ausgeht, daß private Schlüssel wirklich geheim und bewacht sind, sind die beschriebenen Verfahren bei genügend großen Schlüsseln gegen alle Knack-Verfahren in menschlichen Zeiträumen immun. Man spricht daher von starker Kryptographie.

Löcher, die gleich am Beispiel von PGP noch etwas genauer beschrieben werden, sind in diesem Verfahren natürlich auch vorhanden. Schwierig ist es beispielsweise auf einem Mehrbenutzersystem wie UNIX den privaten Schlüssel auf dem Dateisystem wirklich ge-heim zu halten, vor allem vor dem Systembetreuer. Deshalb werden diese Schlüssel auch noch durch ein verbessertes Paßwort, eine sogenannte Paßwortphrase beliebiger Länge geschützt. Sie ist jedoch bei der Tastatureingabe wieder den Spähversuchen böswilliger Spione ausgesetzt. Eine Alternative wäre den Schlüssel nur auf einem nicht vernetzten Einplatz-Rechner, auf den Nachrichten aber per Diskette übertragen werden müssen, zu halten.

Ein anderes Problem ist der sogenannte Man in the Middle Attack. Hier spiegelt je-mand zwischen den beiden Kommunikationspartnern an beiden Enden vor, e wäre das vermutete Gegenüber. Er gibt beiden seinen eigenen öffentlicher Schlüssel und behauptet, es wäre der des Gegenübers. So kann er alle oben beschriebenen Verfahren aushebeln, indem er beide Partner täuscht. Hier hilf

nur, daß die beiden Kommunikationspartner Mittel und Wege finden ihre öffentlichen Schlüssel ohne Veränderung auszutauschen. Dies kann durch zentrale Schlüssel-Verwaltungsstellen geschehen oder durch die Signatur öffentlicher Schlüssel wie Dokumente mit einem privaten Schlüssel. Trägt nun ein öffentlicher Schlüssel die Unterschrift einer Person, der man traut, kann auch dem Schlüssel vertraut werden, da die Unterschrift nur mit dem privaten Schlüssel dieser Person erzeugt worden sein kann.

Ein letztes Problem ist die Langsamkeit des Verfahrens. Da es sich hier um Festkom-ma-Arithmetik jenseits aller Registerbreiten handelt, müssen die Zahlen symbolisch verarbeitet werden, was dazu führt, daß diese Verfahren bis zu 1000 mal langsamer sind als symmetrische, wie z.B. DES oder IDEA, die in Registern arbeiten.

PGP umgeht diese Probleme, indem es nicht die Nachricht mit RSA verschlüsselt, sondern einen 128 Bit langen Session Key. Dieser Schlüssel wird dann für ein schnelles, symmetrisches Verfahren benutzt, das die Nachricht verschlüsselt. Hier kommt allerdings nicht das in den USA entwickelte DES zum Einsatz, sondern das schweizerische IDEA.

Noch ein paar Worte zu den rechtlichen Problemen, die ständig im Zusammenhang mit PGP auftreten: Zum einen gibt es patentrechtliche Probleme, weil das IDEA-Verfahren von der schweizerischen Firma Ascom in Solothurı patentiert ist und bei kommerzieller Verwendung von PGP lizenziert werden muß. An Universitäten und von Privatpersonen darf es allerdings frei benutzt werden.

Ein weiteres Problem sind nationalstaatliche Beschränkungen von starken Kryptographie-Verfahren. So darf Kryptographie-Software mit Schlüsseln von über 64 Bit Länge beispielsweise nicht aus den USA exportiert werden. Wer es dennoch tut, macht sich des Schmuggelns von Munition schuldig. Würde PGP in den USA auf einen ftp-Server gelegt, läge derselbe Tatbestand vor wie bei der Mitführung einer Kiste Patronen im Fluggepäck!

Um niemanden in diese Gefahr zu bringen, gibt es eine amerikanische und eine internationale Version von PGP. Das RSA-Verfahren an sich ist publiziertes mathematisch-es Grundlagenwissen und daher nicht einschränkbar, d.h. jeder mathematisch versier-te Programmierer kann es nachbauen. So z.B. Stale Schumacher in Norwegen, der die internationale Version von PGP erstellt. Beiden Versionen sind völlig identisch in der Leistungsfähigkeit. Die internationale Version darf allerdings nicht in den USA verwendet werden, weil sie RSA-Algorithmen enthält, die dort patentiert, aber außerhalb der USA nicht anerkannt sind. Damit das amerikanische PGP frei verteilt werden kann, wird eine spezielle lizenzfreie Version der RSA-Algorithmen namens RSAREF verwendet. Alles klar?

Manche Länder, wie Frankreich oder Rußland, verbieten ihrem Volk grundsätzlich die Verwendung starker Kryptographie, wodurch die Verwendung von PGP dort illegal ist, außer man holt sich bei einer staatlichen Stelle eine Genehmigung. Da es aber derzeit keine staatlichen Stellen mit genügend Personal und Know-how gibt, um die Sache zu beurteilen, ist diese Angelegenheit momentan eher als Witz zu betrachten.

Auf europäischer Ebene laufen ständige Verhandlungen zu einer einheitlichen Regelung der Kryptographie in der EU. Die Ergebnisse sind allerdings ziemlich diffus, da die beauftragten Kommissionen es oft am Blick für die Realität missen lassen. In Deutschland ist die Verwendung aller PGP-Versionen legal und auch schon relativ weit verbreitet, sodaß viele Partner zur vertraulichen Kommunikation bereitstehen.

Technische Details der PGP-Verwendung

PGP kann man z.B. über ftp.uni-stuttgart.de:/pub/unix/securitybekommen. Obwohl im UNIX-Pfad, ist es dort für alle gängigen Plattformen - aktuelle Version (Stand April 96) 2.6.3i - verfügbar. Sämtliche 2.6-Versionen sind untereinander kompatibel. Die Version 2.3 sollte allerdings nicht mehr verwendet werden. In /sw ist die Version 2.6i bzw. 2.6.2i für einige Plattformen installiert. Die Endung i kennzeichnet die internationale Version.

Wer PGP für eine exotische Plattform haben will, muß u.U. den Schlüssel etwas patchen, hat aber meist Erfolg, da das Programm einfach geschrieben ist und eigentlich nur rechnet sowie Dateien liest bzw. anlegt. Wir empfehlen die englischsprachige Version, weil die deutsche leicht bizarr übersetzt ist. Da das Programm ursprünglich auf MS-DOS entwickelt wurde, benimmt es sich unter UNIX manchmal etwas daneben (Anlegen von Dateien namens stdin).

Hat man PGP nun auf seinem System installiert, sollte die erste Aktion das Erstellen ei-nes Schlüsselpaares sein, ohne das man in der PGP-Welt ein Niemand ist.

Wer auf die Devise *Real Hackers don`t read Manuals* vertraut, dem seien hier noch schnell zwei Optionen nahegelegt:

1. -h bietet allgemeine Hilfe

2. -k bietet Hilfe zur Schlüsselverwaltung.

Ansonsten viel Spaß!

Für alle Nicht-Hacker folgt hier eine detailliertere Erklärung.

Zuerst muß ein Schlüsselpaar erzeugt werden:

```
# pgp -kg

Pretty Good Privacy(tm)2.6.2i-Public-key encryption for the masses.
(c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good Software. 7 May
95 International version - not for use in the USA.Does not use
RSAREF . Current time: 1996/03/25 15:14 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- "edblbase;Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits:
```

Ohne als Militarist zu gelten kann ruhig ein 1024 Bit-Schlüssel gewählt werden. `slow` ist unerheblich, weil, wie bereits erwähnt, mit RSA nur minimale Datenmengen verschlüsselt werden. 512 Bit sind zu kurz, da bereits ein 429 Bit Key durch einen Workstation-Verbund von einigen tausend Rechnern geknackt wurde. Falls Sie den begründeten Verdacht haben, daß Sie von einer Institution ausgehorcht werden, die über starke Supercomputer verfügt, können Sie an dieser Stelle je nach Version auch höhere Schlüssellängen (bis 2048 Bit) angeben. Die Knack-Dauer eines Schlüssels steigt ex-ponentiell mit dessen Länge. Genauere Informationen dazu stehen in den FAQs am Ar-tikelende. Mit Schlüssellänge ist hier die Zahl *n* gemeint.

```
Choose 1, 2, or 3, or enter desired number of bits: 3
Generating an RSA key with a 1024-bit modulus.
You need a user ID for your public key. The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example: John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key:
```

Es bietet sich der eigene Name samt E-Mail-Adresse an:
Otto B. Nutzer <`otto@uni-stuttgart.de`>

```
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: Meine Oma faehrt im Huehnerstall Motorrad
Enter same pass phrase again: Meine Oma faehrt im Huehnerstall
Motorrad
```

Man sollte hier unbedingt ausnutzen, daß mehr als acht Buchstaben zur Verfügung stehen. Das Eigegebene wird nicht angezeigt. Bitte verwenden Sie keine Zeichen, die Sie nicht auf jeder Tastatur wiederfinden! Insbesondere sei hier vor Umlauten gewarnt.

```
Note that key generation is a lengthy process. We need to generate
968 random bits. This is done by measuring the time intervals
```

```
between your keystrokes. Please enter some random text on your
keyboard until you hear the beep:
```

Geben Sie nun einen beliebigen Text ein. Dabei werden die Abstandszeiten zwischen dem Drücken der Tasten gemessen und als Zufallszahlen verwendet. Hier empfiehlt es sich die Paßwortphrase abzutippen, damit sie sich besser einprägt. Sollte sie Ihnen nämlich entfallen, ist Ihr Schlüsselpaar wertlos und Sie müssen allen Partnern einen neuen Schlüssel geben.

```
.....************
.....****
Key generation completed.
```

Nun haben Sie ein Schlüsselpaar, das typischerweise in einem Verzeichnis namens `pgp` oder `.pgp` steht:

```
1 drwxr-xr-x 2 zrzv0111 zr0111 512 Mar 25 16:30 .
4 drwxr-xr-x 20 zrzv0111 zr0111 2048 Mar 25 16:30 ..
1 -rw------- 1 zrzv0111 zr0111 190 Mar 25 16:30 pubring.pgp
1 -rw------- 1 zrzv0111 zr0111 408 Mar 25 16:30 randseed.bin
2 -rw------- 1 zrzv0111 zr0111 523 Mar 25 16:30 secring.pgp
```

Hier sehen Sie den öffentlichen Schlüsselring, einen Satz Zufallszahlen und den privaten Schlüsselring. Ihr privater Schlüsselring steht allerdings nicht im Klartext auf der Platte, sondern ist mit IDEA verschlüsselt. Der Schlüssel dafür ist die 128 Bit MD5-Prüfsumme Ihrer Paßwortphrase.

Den öffentlichen Schlüssel können Sie sich mit der Option -kv (key view) oder -kvv (key view verbose) anschauen:

```
# pgp -kv

Pretty Good Privacy(tm) 2.6.2i - Public-key encryption for the
masses. (c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good
Software. 7 May 95 International version - not for use in the USA.
Does not use RSAREF. Current time: 1996/03/25 15:41 GMT
Key ring: /home/otto/.pgp/pubring.pgp`
Type bits/keyID Date User ID
pub 1024/04A5D509 1996/03/25 Otto B. Nutzer <otto@uni-stuttgart.de>
1 matching key found.
```

Momentan ist nur Ihr eigener Schlüssel enthalten, aber es kommen sicher bald mehrere hinzu. Anfügen kann man Schlüssel mit der Option `-ka` (key add). Wenn Sie eine Datei `herbert.key` haben, die einen Schlüssel enthält, geben Sie einfach nur `pgp -ka herbert.key` ein und der öffentliche Schlüssel wird angefügt. Noch leichter wird es, wenn jemand seinen öffentlichen Schlüssel in der `finger`-Information bereit hält; man kann dann die Ausgabe des `finger`-Kommando direkt mit der Option `-f` in `pgp` umleiten. Meinen Public Key

bekommen Sie folgendermaßen:

```
# finger bernd@visbl.rus.uni-stuttgart.de | pgp -kaf

Pretty Good Privacy(tm) 2.6.2i - Public-key encryption for the
masses. (c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good
Software. 7 May 95 International version - not for use in the USA.
Does not use RSAREF. Current time: 1996/03/25 15:46 GMT
Looking for new keys... pub 768/64EABDF5 1995/07/07 Bernd Lehle
<lehle@rus.uni-stuttgart.de>
Checking signatures...
Keyfile contains: 1 new key(s)
Looking for new keys... No new keys or signatures in keyfile.
```

Schauen wir uns den öffentlichen Schlüsselring nun genau (verbose) an:

```
# pgp -kvv

Pretty Good Privacy(tm) 2.6.2i - Public-key encryption for the
masses. (c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good
Software. 7 May 95 International version - not for use in the USA.
Does not use RSAREF. Current time: 1996/03/25 15:52 GMT
Key ring: /home/otto/.pgp/pubring.pgp`
Type bits/keyID Date User ID
pub 768/64EABDF5 1995/07/07 Bernd Lehle <lehle@rus.uni-stuttgart.de>

sig 64EABDF5 Bernd Lehle <lehle@rus.uni-stuttgart.de>
sig 8E0A49D1 (Unknown signator, can`t be checked) Bernd Lehle
pub 1024/04A5D509 1996/03/25 Otto B. Nutzer <otto@uni-stuttgart.de>
2 matching keys found.
```

Die erste Zeile zeigt, daß nun mein öffentlicher Schlüssel mit einer Länge von 768 Bit eingebunden ist. Somit können Sie mir geheime Nachrichten schicken. Die weiteren Zeilen fangen mit sig an, was bedeutet, daß mein öffentlicher Schlüsse Unterschriften trägt. Die erste Signatur ist meine eigene, d.h. der Schlüssel kann nur von der Person kommen, die den zugehörigen privaten Schlüssel hat.

Die zweite können Sie nicht lesen, da Sie den öffentlichen Schlüssel des Unterzeichnenden nicht haben; Sie brauchen ihn aber zwingend, um die Unterschrift zu entziffern, die mit seinem privaten Schlüssel erstellt wurde. Zum Nachprüfen können Sie sich den benötigten Schlüssel mit`#finger ley-pgp@ftp.cert.dfn.de | pgp -kaf` holen und ihn anfügen.

Der eigene öffentliche Schlüssel sollte gleich nach der Erzeugung mit pgp -ks Nutzer unterschrieben werden. Dies beweist, daß er jemanden gehört, der auch den geheimen Schlüssel hat, also weniger leicht gefälscht sein kann. Nutzer muß dabei innerhalb des Rings eindeutig sein, da sonst der erste Schlüssel genommen wird, auf den der Name paßt.

Wie benutze ich den Schlüssel?

Wenn Sie mir beispielsweise vertraulich mitteilen wollen, daß Sie ab heute PGP benutzen, müssen Sie folgende Schritte unternehmen: Sie erstellen eine Datei, die diese Nachricht enthält, nennen wir sie`hallo.txt "edblbase;Hallo, ich habe jetzt pgp! Otto."` Sie wird nun mit der Option `-e` (encrypt) verschlüsselt. Damit die Nachricht danach als Mail verschickt werden kann, muß der verschlüsselte Text im ASCII-Format vorliegen, was Sie mit der Option `-a` erreichen:

```
# pgp -ea hallo.txt

Pretty Good Privacy(tm) 2.6.2i - Public-key encryption for the
masses. (c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good
Software. 7 May 95 International version - not for use in the USA.
Does not use RSAREF. Current time: 1996/03/25 16:03 GMT
Recipients` public key(s) will be used to encrypt. A user ID is
required to select the recipient`s public key.
Enter the recipient`s user ID:
```

Hier ist nun ein im Schlüsselring eindeutiger Name gefragt:

```
Enter the recipient`s user ID: Bernd
Key for user ID: Bernd Lehle <lehle@rus.uni-stuttgart.de> 768-bit
key, Key ID 64EABDF5, created 1995/07/07
Also known as: Bernd Lehle
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: "edblbase;Bernd Lehle
<lehle@rus.uni-stuttgart.de>".
```

Dies bedeutet, daß mein öffentlicher Schlüssel keine Unterschrift von jemandem trägt, dem Sie vertrauen - wie sollte er auch. Wenn Sie das stört, und Sie mir vertrauen, dann unterschreiben Sie meinen Schlüssel für Ihren Privatgebrauch mit `pgp -ks Bernd` einfach selbst und die Meldung taucht nicht wieder auf.

```
Are you sure you want to use this public key (y/N)? y
.
Transport armor file: hallo.txt.asc
```

Ein `transport armor` ist immer nötig, wenn Daten via Mail verschickt werden. Schauen wir einmal nach, was aus dem Text geworden ist:

```
-----BEGIN PGP MESSAGE----- Version: 2.6.2i
hGwDIEeAeWTqvfUBAv9WLqNOmTkOHPJD9VpMjkRabYLTSrSuMTKZosMcIRJZLbwn
2kWD3yqO2cVafQN21mxxaY5oSQ1cfs2eeq8tRIlDhfb4m1vAxdx91Qj0uZ/JgbeG
lhKbfjai5url5Xx6U9ymAAAAPClWstI+ZOGi7Xbry/zWn2+J+buKKrLXS5Tr/0Xh
mJ1IZwnjVnNzVuxNIliuOF169bPFGyq4j2mI5YqoEg== =uhlI
-----END PGP MESSAGE-----
```

Nicht wiederzuerkennen. Die `BEGIN`- und `END`-Zeilen sowie die Versionsnummer dürfen nicht verändert werden, da sonst eine Entschlüsselung unmöglich wird.

Dieses Ungetüm verschicken Sie mir nun per Mail. Sobald es bei mir ankommt, muß ich es entweder abspeichern oder es direkt per Pipeline durch pgp ohne Argumente schikken. Das Programm erkennt an der BEGIN-Zeile selbst, was es mit der Nachricht zu tun hat. Angenommen, ich speichere es unter `mail.otto`, geht es folgendermaßen weiter:

```
# pgp mail.otto

Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the
masses. (c) 1990-96 Philip Zimmermann, Phil`s Pretty Good Software.
1996-01-18 International version - not for use in the USA. Does not
use RSAREF. Current time: 1996/03/25 16:22 GMT
File is encrypted. Secret key is required to read it. Key for user
ID: Bernd Lehle <lehle@rus.uni-stuttgart.de> 768-bit key, key ID
64EABDF5, created 1995/07/07 Also known as: Bernd Lehle
You need a pass phrase to unlock your RSA secret key. Enter pass
phrase:

Da mein privater Schlüssel gefragt ist, muß ich ihn durch die
Paßwortphrase freigeben:

Enter pass phrase: Pass phrase is good. Just a moment......
Plaintext filename: mail.otto Output file "esinglbase;mail.otto`
already exists. Overwrite (y/N)? y
```

Dies ist eine Marotte von PGP. Es könnte den Text einfach anzeigen, aber es will unbedingt einen File schreiben. Nun habe ich die ursprüngliche Botschaft in `mail.otto` und kann sie lesen. Direkt anzeigen kann man den Text mit `pgp -m` (more).

Jetzt will ich auch auf die freudige Botschaft antworten. Dabei reicht es mir aber, daß die Antwort signiert ist, sie muß nicht verschlüsselt sein. Dazu schreibe ich die Antwort direkt in `mail.otto` und signiere sie mit der Option `-as` (ASCII signature):

```
# pgp -as mail.otto

Pretty Good Privacy(tm) 2.6.3i - Public-key encryption for the
masses. (c) 1990-96 Philip Zimmermann, Phil`s Pretty Good Software.
1996-01-18 International version - not for use in the USA. Does not
use RSAREF. Current time: 1996/03/25 16:30 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: Bernd Lehle <lehle@rus.uni-stuttgart.de> 768-bit
key, key ID 64EABDF5, created 1995/07/07
```

```
Enter pass phrase:
```

Da zum Unterschreiben der private Schlüssel benötigt wird, muß er auch hier erst mittels Paßwortphrase freigegeben werden:

```
Enter pass phrase: Pass phrase is good. Just a moment....
Output file "esinglbase;mail.otto.asc` already exists. Overwrite
(y/N)? y
Transport armor file: mail.otto.asc
#
```

Schauen wir nach, was wir erzeugt haben:

```
-----BEGIN PGP SIGNED MESSAGE-----
> Hallo, ich hab jetzt pgp ! Otto.
Freut mich ! Bernd
-----BEGIN PGP SIGNATURE----- Version: 2.6.3i Charset: ascii
iQB1AwUBMVbLuiBHgHlk6r31AQGZmAMAoBdhjemZy3YCWmL6SCSdOe6+kjsdf8J934fF

kikhsddkuCQl4FLHGVFshmjE4VnmZBn5UfGIEPhPClcHlJ3vgCJO87dfh339jhxA5TYX

YQI8LnMDZlV7xjRUVDonHGL+bOr0cR5mC =4e30
-----END PGP SIGNATURE-----
```

Hier tauchen wieder die `BEGIN`- und `END`-Zeilen auf. Diesmal ist aber auch die Unterschrift unten angefügt, die die Authentizität des Textes garantiert. Sie besteht aus einer mit dem geheimen Schlüssel verschlüsselten Prüfsumme des Textes zwischen den Markierungszeilen.

Anmerkung:
Normalerweise wird bei `pgp -as` der Text auch noch Base 64-codiert, um einen Verlust von Umlauten zu verhindern, der den Text verfälschen würde. Dies ist nicht immer er-wünscht; man kann es abwenden, indem man `pgp -as +clearsig=on` explizit angibt oder in der Konfigurationsdatei `config.txt` die Zeile `ClearSig = on` einträgt.

Wenn nun diese Antwort empfangen wird, kann man sie wieder abspeichern und ebenfalls einfach an `pgp` übergeben:

```
# pgp answer.bernd

Pretty Good Privacy(tm) 2.6.2i - Public-key encryption for the
masses. (c) 1990-1995 Philip Zimmermann, Phil`s Pretty Good
Software. 7 May 95 International version - not for use in the USA.
Does not use RSAREF. Current time: 1996/03/25 16:47 GMT
Warning: Unrecognized ASCII armor header label "edblbase;Charset:"
ignored.
File has signature. Public key is required to check signature. .
Good signature from user "edblbase;Bernd Lehle
```

```
<lehle@rus.uni-stuttgart.de>". Signature made 1996/03/25 16:44 GMT
Die Unterschrift ist also in Ordnung. Dies kann PGP allerdings nur
feststellen, wenn es meinen öffentlichen Schlüssel hat. Wenn Sie
eine Nachricht von jemand erhalten, dessen öffentlichen Schlüssel
Sie nicht haben, müssen Sie sich bemühen diesen auf möglichst
vertrauenswürdigem Weg zu erhalten (möglichst mit Unterschriften von
Leuten, denen Sie trauen). Erst wenn er an Ihrem öffentlichen
Schlüsselring hängt, können Sie die Unterschrift prüfen.
Plaintext filename: answer.bernd Output file
"esinglbase;answer.bernd` already exists. Overwrite (y/N)? y

answer.bernd enthält nun den Text ohne Steuerzeilen.
```

Ich hoffe, daß das Prinzip und die Anwendung jetzt verständlicher geworden sind.

Zum Abwenden weiterer Unklarheiten nachfolgend einige Frequently Asked Questions.

Q: Was mache ich, wenn ich an mehrere Empfänger dieselbe Mail verschicken will?
A: `pgp -ea text.file User1 User2 User3 ...`

Q: Kann ich Dateien auch zum Eigengebrauch verschlüsseln?
A: Ja, mit `pgp -e file Otto` **durch den eigenen öffentlichen Schlüssel oder** `pgp -c file`**. Bei** `pgp -c` **wird eine neue Paßwortphrase abgefragt und die Datei dann mit IDEA verschlüsselt.**

Q: Muß ich immer alles abspeichern und dann die Dateien bearbeiten? Geht das nicht einfacher?
A: Das kommt auf die benutzte Mail Software an. Wer unter UNIX zum Schreiben von Mails einen Editor benutzt, der Makros zuläßt (z.B. vi und emacs), kann PGP-Makros auf Funktionstasten legen. Bei manchen Programmen mit graphischer Oberfläche (zmail für UNIX, einige POP Mail Clients für Windows und Mac) ist ein PGP-Knopf vorhanden, der dann PGP entsprechend aufruft. Eine Version von elm, die PGP verwenden kann, ist in Arbeit. Bis dahin kann man bei elm zum Entschlüsseln und Unterschrift prüfen einfach `| pgp` **eingeben. Pine bietet diese Funktionalität leider nicht.**

Q: Woher kommt `config.txt`**?**
A: Es ist Bestandteil des Paketes. Wer `/sw` **verwendet findet es unter:**
`/sw/<platform>/pgp-<Version>/doc/config.txt`

Q: Woher bekomme ich öffentliche Schlüssel? Wie verteile ich meinen?
A: Viele Benutzer bieten sie per `finger` **in ihrem** `.plan` **an. Es gibt aber auch sogenannte Public Key Server, in denen man Schlüssel suchen oder deponieren kann. Der deutsche liegt unter** http://www.de.pgp.net/pgp/ **im Web.**
Per Mail ist er unter `pgp-public-keys@keys.de.pgp.net` **erreichbar. Einfach als**

Subject der Mail `HELP` angeben.

Q: Was kann ich falsch machen?
A: Die Paßwortphrase vergessen, den privaten Schlüssel verlieren, einen zu kurzen Schlüssel (<512 Bit) nehmen, Verschlüsselung übertreiben, Verschlüsselung nicht ernst nehmen. Es macht zum Beispiel keinen Sinn, sich einen 2048 Bit-Schlüssel zu machen und ihn dann lesbar auf der Platte liegen zu lassen. Auch das Anhängen eines 2048 Bit-Schlüssels mit 20 Unterschriften zur besseren Verteilung an jede Mail ist nicht gerade fachmännisch, da es die Mail um 100 Zeilen länger macht.

Q: Was soll ich überhaupt verschlüsseln?
A: Jede Kommunikation, die auf dem Weg zum Empfänger abgefangen und gelesen werden kann, bei der man genau das verhindern will. Die Gründe dafür sollte jeder selbst wissen. Die Arbeitsgruppe Systemsicherheit verschlüsselt nur im Einsatzfall gegen Eindringlinge.

Q: Wie kann ich meinen privaten Schlüssel schützen?
A: Auf UNIX die Permission richtig setzen (am besten `-r--------, mode 400`). Die Paßwortphrase nicht aufschreiben und sicherstellen, daß sie nicht vom Terminal abgehört wird.

Q: Was ist eine Key-ID?
A: Eine Art Nummer, um einen Schlüssel innerhalb eines -rings zu identifizieren. Die Wahrscheinlichkeit, daß zwei Schlüssel dieselbe Key-ID haben ist zwar sehr gering, aber nicht gleich Null. Außerdem kann man einen Key mit einer bestimmten Key-ID absichtlich erzeugen, daher darf man sie nicht als eindeutig ansehen.

Q: Was ist ein Key Finger Print?
A: Der Finger Print ist eine MD 5-Prüfsumme des Schlüssels. Daß zwei Schlüssel den selben Finger Print haben, ist äußerst unwahrscheinlich, zumal ein Finger Print auch nicht gezielt beeinflußbar ist. Der Finger Print eignet sich daher als Ausweis für einen Schlüssel und kann anstelle des Schlüssels zur Kontrolle weitergegeben werden. Ein Finger Print ist wesentlich kürzer als ein Schlüssel Berechnen kann man den Finger Print eines Schlüssels mit`pgp -kvc <Name>`

Q: Kann ich meine Paßwortphrase ändern, wenn sie abgehört wurde? Oder meine User-ID, wenn ich eine andere Mail-Adresse bekomme?
A: Natürlich geht das: Mit `pgp -ke User [Schlüsselring]` können Sie die Paßwortphrase eines Schlüssels in dem betreffenden -ring (z.B.`secring.pgp`) ändern.
Eine neue User-ID kann nur angefügt werden. Dadurch wird verhindert, daß Unterschriften übertragen werden oder verloren gehen.

Q: Wie sicher sind die Schlüssel? Warum ist der Session Key nur so kurz?

A: Die Sicherheit eines Schlüssels hängt von den Angriffsmethoden ab, die das Verschlüsselungsverfahren bietet. RSA ist durch Faktorisierung des Produkts der beiden Primzahlen angreifbar, IDEA bisher nur durch Probieren aller Möglichkeiten. Da dieses aber ungleich mehr Rechenaufwand erfordert, können IDEA-Schlüssel bei gleicher Sicherheit wesentlich kürzer sein.
Zum Brechen eines 512 Bit RSA-Schlüssels werden etwa 30 000 MIPS-Jahre veranschlagt (ein Prozessor, der 30 Milliarden Instruktionen pro Sekunde leistet, rechnet daran ein Jahr). Ein vergleichbar sicherer IDEA-Schlüssel muß dafür nur 64 Bit lang sein. Bei einem doppelt so langen 1024 Bit RSA-Schlüssel würde dieselbe Maschine bereits 10 Millionen Jahre rechnen. Der entsprechende IDEA-Schlüssel käme auf etwa 100 Bit. Der größte RSA-Schlüssel, den PGP anbietet hat 2048 Bit. Um ihn zu brechen kann man einen weiteren Faktor von einer Milliarde hinzurechnen. Etwa so sicher ist der von PGP verwendete128 Bit IDEA Session Key. Diese Zahlen können sich aber bei Entdeckung neuer Verfahren drastisch ändern.
Der bereits geknackte 429 Bit RSA-Schlüssel benötigte etwa 5000 MIPS-Jahre.

Q: Wie komme ich an mehr Informationen?
A: Im O`Reilly-Verlag ist ein gutes Buch von Simpson Garfinkel erschienen:*PGP*.
Wer Online-Information bevorzugt, sollte die dem PGP beigefügten Dokumentation le-sen oder sich unter
http://www.uni-mannheim.de/studorg/gahg/PGP/HANDBUCH/informieren.
Weitere Web Pages können Sie auf den großen Suchmaschinen finden, wenn Sie nach PGP suchen.
Es gibt natürlich auch News Groups zu dem Thema, z.B.`alt.security.pgp` und `sci.crypt`

Fingerprints einiger Ansprechpartner am RUS

· Sascha Buhr 2E B9 91 AE 04 8C 16 E8 67 A5 14 7F CB 7A C6 93

· Dr. Lothar Ehnis C1 BF DC 39 91 CE 40 5C 96 50 21 36 9F E5 4E 0A

· Herbert Franz 75 77 BA FE 72 A1 66 95 8E 15 8B 15 C3 0B D3 23

· Dr. Lisa Golka 05 6F 43 E2 E8 95 4B DB 7D 97 CF 68 3B DA 6E 2E

· Bernd Lehle 3E B0 35 8D 59 D5 AE AA 5A F9 60 80 9E E0 55 48

· Helmut Springer AE 42 C3 2C A1 3E 55 6D B3 AC 3C D2 F3 CF FF E7

Bernd Lehle, NA-5531
E-Mail: `Lehle@rus.uni-stuttgart.de`

Oliver Reutter, NA-4513