Das GNU-Handbuch zum Schutze der Privatsphäre

Das GNU-Handbuch zum Schutze der Privatsphäre

Richten Sie bitte Ihre Fragen, Fehlermeldungen oder Anregungen, sofern sie dieses Handbuch betreffen, an die Mailingliste <gnupg-doc.de@gnupg.org>. Mike Ashley ist der Autor des orginalen englischen Version dieses Handbuchs, Beiträge lieferten auch Matthew Copeland, Joergen Grahn und David Wheeler.

J. Horacio MG hat das Handbuch ins Spanische übersetzt.

Harald Martin, Roland Goretzki und Peter Neuhaus haben das Handbuch ins Deutsche übersetzt.

Peter Neuhaus hat das Handbuch überarbeitet und erweitert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Vorwort

[*Grundgesetz Artikel 10, Absatz 1:*
"Das Briefgeheimnis sowie das
Post- und Fernmeldegeheimnis
sind unverletzlich."]

[*"Eckpunkte der deutschen Kryptopolitik",*
verabschiedet vom deutschen Bundeskabinett am 2. Juni 1999:
"Der Einsatz kryptographischer Verfahren ist von
außerordentlicher Bedeutung für eine effiziente
technische Kriminalprävention. Dies gilt sowohl
für die Gewährleistung der Authentizität und
Integrität des Datenverkehrs wie auch für den
Schutz der Vertraulichkeit."]

Elektronische Daten spielen im Zeitalter des Computers und der weltweiten Vernetzung eine herausragende Rolle. Privatleute, Firmen, Politiker, Organisationen und Behörden machen zunehmend Gebrauch von der bequemen, schnellen und preisgünstigen Möglichkeit, per E-Mail zu kommunizieren, und nutzen elektronische Speichermedien (Festplatten, Disketten, CDROMs), um darauf ihre persönlichen Daten, Forschungsergebnisse, Firmengeheimnisse, Kunden- oder Patienteninformationen, Statisken, Notizen, Entwürfe, Umsatzzahlen usw. zu speichern. Bei der Abwicklung von Geschäftsvorgängen (Bestellung, Bezahlung, Verträge) spielt das Internet eine immer wichtigere Rolle. Den Weg, den Ihre Daten über das Internet zu einer Zieladresse gehen, können Sie weder vorhersagen noch vorherbestimmen. Alle Daten, die unverschlüsselt (oder mit einer unsicheren Methode verschlüsselt) über's Netz gehen, sind quasi öffentlich. Man muß davon ausgehen, das diese Daten - von wem auch immer - mitgelesen oder manipuliert und - zu welchem Zweck auch immer - mißbraucht werden können. Daten, die Sie auf Ihrem Computer abgespeichert haben, sind meist nicht sicher vor unbefugten Zugriffen. Viele Rechner sind nicht einmal mit einem Paßwortschutz versehen, und selbst bei vorhandenem Paßwortschutz gibt es vielfältige Möglichkeiten, an diese Daten zu gelangen. Noch nie war es so einfach und effektiv möglich, in Ihre Privatsphäre einzudringen oder Zugang zu Ihren vertraulichen Informationen zu erlangen.

Warum Kryptographie?

Kryptographie (die Wissenschaft von der Verschlüsselung) gewährleistet

- *Vertraulichkeit*
- *Integrität* und
- *Authentizität*

Ihrer Daten und Ihrer Kommunikation.

Wenn Sie E-Mails unverschlüsselt verschicken, müssen Sie sich darüber im klaren sein, daß deren Inhalt weniger vertraulich ist als bei einer Postkarte. Die Administratoren sowohl Ihres Mailservers als auch des Empfängers könnten ohne weiteres ihre E-Mails lesen, abfangen oder verändern. Auf ihrem Weg zum Empfänger durchlaufen E-Mails unter Umständen etliche Rechner. Jeder, der Zugang zu einem dieser Rechner hat, auch jeder Cracker[1], der durch irgendwelche Sicherheitslöcher in diese Rechner eindringt, kann mühelos auf Ihre E-Mails zugreifen. Unter Umständen werden Ihre E-Mails sogar auf der Festplatte eines dieser Zwischenrechner gespeichert. Auch könnte der Carrier, also der, der die Datenleitungen zu Verfügung stellt (in Deutschland meist die Deutsche Telekom oder Colt-Telekom) die Datenpakete, die über seine Leitungen gehen, gezielt filtern. Es ist auch nicht auszuschließen, daß jemand diese Leitungen von außen anzapft.

Es geht aber nicht allein darum, sich gegen Cracker oder korrupte Sytemadministratoren zu schützen, sondern auch gegen das planmäßige Eindringen staatlicher Organisationen (des eigenen oder eines anderen Landes) in Ihre Privatsphäre. Die Geheimdienste vieler Länder filtern heutzutage nicht nur Telefongespräche, sondern zunehmend auch die Daten, die über das Internet transportiert werden, um daraus wirtschaftlich, politisch oder für die Strafverfolgung nutzbare Daten zu gewinnen. Eine Studie der “Kommission zur Technikfolgeabschätzung des Europaparlamentes” (STOA - Scientific and Technological Options Assessment) über die “Entwicklung von Überwachungstechnologie und dem Risiko des Mißbrauchs wirtschaftlicher Informationen” zeigt beispielsweise, daß das Belauschen elekronischer Kommunikation bereits systematisch und in großem Stil betrieben wird. Eines der prominentesten Beispiele ist das ECHELON-System, das von den USA, Kanada, Großbritannien, Australien und Neuseeland gemeinsam unterhalten wird. Ursprünglich zum Belauschen des Ostblocks konzipiert, sammeln heute über 120 Stationen mit großem Aufwand Informationen unter anderem durch Abhören von Satellitenverbindungen und Transatlantikkabeln, um Daten über Einzelpersonen, Organisationen, Regierungen, Wirtschaftsunternehmen, Forschungsprojekte und internationale Institutionen zu gewinnen. Auf europäischer Ebene plant die Arbeitsgruppe “Polizeiliche Zusammenarbeit” des Europa-Rats konkrete Maßnahmen für die Überwachung des Telekommunikations-Verkehrs. Das “ENFOPOL 98” genannte Dokument schließt ausdrücklich das Internet und zukünftige Technologien mit ein.

Auch Daten, die unverschlüsselt auf der Festplatte Ihres Rechners oder eines anderen Speichermediums liegen, sind vor unbefugten Zugriffen nicht sicher. Jemand könnte sich über eine Netzwerkverbindung Zugang verschaffen bzw. sich durch Diebstahl oder Einbruch in Besitz Ihrer Daten bringen. Wenn Sie Ihre Daten verschlüsselt haben, kann ein Angreifer - selbst wenn er physisch im Besitz der Daten ist - nicht auf diese zugreifen.

Ein weiteres Problem ist das Authentifizieren von elektronischen Daten. Wie bereits oben erwähnt, ist es möglich, die Absenderadresse und den Inhalt eines E-Mails zu fälschen. Gerade bei offizieller oder geschäftlicher Korrespondenz, dem Austausch von Dokumenten und dem Abwickeln von Geschäftsvorgängen über das Internet ist es wichtig, den Absender eindeutig zu identifizieren und die Integrität der Daten überprüfen zu können.

Die einzige Möglichkeit, um Vertraulichkeit, Integrität und Authentizität von elektronischen Dokumenten zu gewährleisten, ist die Benutzung wirkungsvoller kryptographischer Verfahren, wie sie etwa bei GnuPG Anwendung finden. Durch Verschlüsselung erreichen Sie, daß Ihre Daten nur von den Personen gelesen werden können, für die sie auch bestimmt sind. E-Mails werden quasi in einen Briefumschlag gesteckt, der nur vom Empfänger geöffnet werden kann. Darüberhinaus wird durch digitale Unterschriften eine eindeutige Zuordnung zum Urheber der Signatur möglich, und Manipulationen des Dokumentes oder Vortäuschen eines falschen Urhebers (Absenders) lassen sich feststellen.

In der Elektronischen Datenverarbeitung sollte für Sie die gleiche Sicherheit selbstverständlich sein wie in anderen Bereichen. Wahrscheinlich würden Sie weder ein intimes Liebesgeständnis, noch eine Mitteilung an Ihren Rechtsanwalt, noch Ihre wissenschaftliche oder geschäftliche Korrespondenz per Postkarte schicken. Auch lassen Sie wahrscheinlich keine vertraulichen Dokumente offen in Ihrer Wohnung oder an Ihrem Arbeitsplatz liegen. Ebensowenig würden Sie einen Kaufvertrag ohne rechtsgültige Unterschrift akzeptieren. Verschlüsselung und digitale Signaturen sollten also ein alltäglicher Vorgang für Sie sein. Ob Sie nun berufliches oder privates Interesse am Schutz Ihrer Daten haben: mangelndes Problembewußtsein ist das größte Risiko.

Warum GnuPG?

GnuPG (der GNU Privacy Guard) ist ein Programm zum Verschlüsseln und Signieren von digitalen Daten und arbeitet unabhängig von den jeweiligen Datenformaten (E-Mail, Textdateien, Bilddaten, Sourcecode, Datenbanken, komplette Festplatten usw.). Es entspricht der im RFC2440 festgelegten OpenPGP-Spezifikation und ist kompatibel zu PGP 5.x der Firma NAI. GnuPG verwendet dazu hauptsächlich ein hybrides Verfahren mit öffentlichem Schlüssel. Zum Verschlüsseln kann GnuPG aber ebenso ausschließlich symmetrische Verfahren einsetzen.

GnuPG ist derzeit eine der sichersten Anwendungen zum Verschlüsseln und Signieren von Daten. Bei sorgfältiger Anwendung ist eine Verschlüsselung mit GnuPG auch in absehbarer Zukunft nicht zu knacken. Im Gegensatz zu anderen Verschlüsselungsprogrammen wie beispielsweise PGP von der Firma NAI ist GnuPG freie Software. Das bedeutet unter anderem, daß der Programm-Quellcode frei verfügbar, frei von Patenten und frei von einschränkenden Lizenzbedingungen ist[2]. Jeder Anwender kann so das Programm auf seine Integrität hin prüfen. Das heißt beispielsweise, daß sich Hintertüren (Key Recovery) oder 'Generalschlüssel' (Key Escrow) nicht versteckt einbauen lassen und jeder Anwender die Möglichkeit hat, Fehler zu beseitigen, das Programm zu verbessern oder nach seinen Vorstellungen zu verändern. Darüberhinaus ist GnuPG nicht - wie beispielsweise amerikanische Verschlüsselungsprogramme - durch Ausfuhrbestimmungen künstlich in seiner Funktionalität und Sicherheit beschränkt.

Aufbau des Buches

Die grundlegenden Konzepte und Hintergründe der Verschlüsselung und digitaler Signaturen werden in Kapitel 1 “Konzepte” behandelt. Kapitel 2 “Grundlagen” führt in die Arbeit mit GnuPG ein; die wichtigsten Funktionen, Arbeitsschritte und Optionen werden dort am Beispiel erklärt. In Kapitel 3 “Schlüsselverwaltung” wird ausführlich auf das Editieren, Authentifizieren und Verwalten von Schlüsseln eingegangen. Auf die wichtigsten Aspekte des praktischen Einsatzes und das “Web of Trust” wird in Kapitel 4 “GnuPG im Alltagsgebrauch” eingegangen. Kapitel 5 gibt einen kurzen Überblick über die Kryptographie-Gesetzgebung. Im Anhang des Buches finden Sie ein ausführliches Glossar, das die verwendeten Fachausdrücke erklärt, ein Literaturverzeichnis, eine Sammlung von Internet-Ressourcen sowie eine Anleitung zur Installation von GnuPG.

Fußnoten

1. Eine Person, die vorsätzlich, unbefugterweise und oft mit bösartiger Absicht in fremde Rechnersysteme eindringt, im deutlichen Gegensatz zu “Hacker”, womit ein gutmeinender Computer-Freak gemeint ist (RFC 1983)
2. GnuPG steht unter der sogenannten GNU General Public License (GPL) der Free Software Foundation, die im Anhang des Buches abgedruckt ist.

Kapitel 1. Konzepte

GnuPG verwendet mehrere kryptographische Verfahren wie beispielsweise *symmetrische Verschlüsselung*, *Public-Key-Verschlüsselung* und *Einweg-Hashing*. Natürlich können Sie GnuPG auch ohne tiefere Kenntnis dieser Konzepte benutzen, doch wenn Sie GnuPG effektiv einsetzen möchten, sollten Sie ein wenig Hintergrundwissen haben.

Dieses Kapitel führt in die grundlegenden kryptographischen Konzepte ein, wie sie von GnuPG benutzt werden. Andere Bücher behandeln diese Themen viel detaillierter. Empfehlenswerte Bücher zum tieferen Studium sind beispielsweise Bruce Schneier (http://www.counterpane.com/schneier.html)s "Angewandte Kryptographie" (http://www.awl.de/katalog/item.asp?bnm=3893198547) oder Reinhard Wobsts "Abenteuer Kryptologie" (http://www.awl.de/katalog/item.asp?bnm=3827314135). Weitere Literaturhinweise finden sich im Anhang B.

Symmetrische Verschlüsselung

Eine symmetrische Verschlüsselung benutzt zum Ver- und Entschlüsseln denselben Schlüssel. Zwei Korrespondenzpartner, die eine symmetrische Verschlüsselung benutzen, müssen sich vorher über den Schlüssel einigen. Mit diesem Schlüssel verschlüsselt der Absender die Nachricht und schickt sie an den Empfänger, der sie unter Benutzung desselben Schlüssels wiederherstellt. Nach diesem Prinzip funktionierte beispielsweise die deutsche `Enigma`. Die jeweiligen Tages-Schlüssel wurden als Code-Bücher ausgegeben, und jeden Tag konsultierte dann ein Funker seine Kopie des Code-Buchs, um den aktuellen Tagesschlüssel zu ermitteln, mit dem der Funkverkehr für den betreffenden Tag dann ver- und entschlüsselt wurde. Zu den modernen Beispielen für symmetrische Verschlüsselungen gehören z.B. Blowfish und IDEA.

Ein gutes Verschlüsselungverfahren legt den Schwerpunkt der Sicherheit auf die Geheimhaltung des Schlüssels und nicht auf die Geheimhaltung des verwendeten Algorithmus. Mit anderen Worten, es ist keine Hilfe für einen Angreifer, wenn das Verschlüsselungsverfahren bekannt ist, solange er nicht im Besitz des Schlüssels selbst ist. Die von GnuPG benutzten Verschlüsselungsverfahren beruhen auf diesen Prinzipien.

Da die gesamte Sicherheit auf dem Schlüssel beruht, ist es wichtig, daß der Schlüssel mit verfügbaren Mitteln nicht zu erraten ist. Daraus folgt, daß der Vorrat an möglichen Schlüsseln, der sogenannte *key space*, möglichst groß sein muß. Während seiner Zeit in Los Alamos war der Nobelpreisträger Richard Feynman berühmt für seine Fähigkeit, Safes zu knacken. Um es noch geheimnisvoller zu machen, schleppte er einen Satz von Werkzeugen mit, zu denen ein altes Stethoskop gehörte. In Wirklichkeit wandte er jedoch eine ganze Reihe von Tricks an, um die Zahl der Kombinationen, die er ausprobieren mußte, zu reduzieren; dann fing er an zu raten, bis er die richtige Kombination fand. Mit anderen Worten, er verringerte die Größe des *key space*.

Die Briten benutzten im 2. Weltkrieg Maschinen, um Schlüssel zu erraten. Die deutsche `Enigma` hatte einen sehr großen *key space*, doch die Briten bauten spezialisierte Rechenmaschinen, `Bombes` genannt, um systematisch alle Schlüssel auszuprobieren, bis der jeweilige Tagesschlüssel gefunden war. Manchmal fanden sie den Tagesschlüssel innerhalb der Benutzungsdauer des neuen Schlüssels, an manchen Tagen fanden sie den richtigen Schlüssel überhaupt nicht.

Heute können Computer sehr schnell Schlüssel erraten, und eben deshalb ist in modernen Verschlüsselungsverfahren die Schlüsselgröße äußerst wichtig. Die DES-Verschlüsselung zum Beispiel benutzt einen 56-Bit-Schlüssel; das bedeutet, daß es 2^{56}, also genau 72.057.594.037.927.936 mögliche Schlüssel gibt (das sind *mehr als 72 Billiarden*). Obwohl das eine sehr große Zahl ist, kann ein normaler Mehrzweckcomputer den gesamten *key space* innerhalb von Tagen prüfen. Ein spezialisierter Computer braucht hierfür möglicherweise nur ein paar Stunden. Die moderneren Verschlüsselungsverfahren wie beispielsweise Blowfish und IDEA benutzen sämtlich 128-Bit-Schlüssel, was bedeutet, daß es 2^{128} (340.282.366.920.938.463.463.374.607.431.768.211.456!!!) mögliche Schlüssel gibt. Dies sind so unglaublich viel mehr Kombinationen als bei einer 56-Bit-Verschlüsselung, daß sogar selbst dann, wenn man alle Computer der Welt zusammen arbeiten ließe, das bisherige Alter des Universums noch eine zu kurze Zeit sein könnte, um den richtigen Schlüssel zu finden.

Public-Key-Verschlüsselung

Das Hauptproblem bei symmetrischen Verschlüsselungen ist nicht die Sicherheit der eingesetzten Verfahren, sondern der Austausch der Schlüssel. Wenn zwei Kommunikationspartner einmal die Schlüssel ausgetauscht haben, kann der betreffende Schlüssel für sicheren Datenaustausch benutzt werden. Die Frage ist nur, auf welchem sicheren Wege der Schlüsselaustausch stattgefunden hat. Wahrscheinlich wäre es für einen Angreifer viel leichter, den Schlüssel abzufangen, als alle möglichen Schlüssel im *key space* auszuprobieren (eine Erfahrung, die die Deutschen mit ihrer Enigma auch machen mußten). Ein weiteres Problem ist die Anzahl der insgesamt benutzten Schlüssel. Wenn die Zahl der Leute, die miteinander kommunizieren wollen, *n* beträgt, so werden insgesamt *n(n-1)/2* Schlüssel (also beispielsweise 45 Schlüssel bei 10 Leuten) benötigt. Dies mag für eine geringe Personenzahl noch angehen, läßt sich aber bei großen Personengruppen nicht mehr handhaben.

Der Sinn von Verschlüsselungsverfahren mit öffentlichem Schlüssel besteht darin, daß das Sicherheitsrisiko beim gegenseitigen Schlüsselaustausch gänzlich vermieden wird. Jeder hat ein Schlüsselpaar mit einem *öffentlichen* und einem *geheimen* Schlüssel. Zum Verschlüsseln einer Nachricht benutzt man den öffentlichen Schlüssel des Empfängers, und nur dieser kann sie mit seinem geheimen Schlüssel wieder entschlüsseln.

Dadurch löst man das Problem des Schlüsselaustausches bei symmetrischer Verschlüsselung. Sender und Empfänger brauchen sich nicht auf einen Schlüssel zu einigen. Erforderlich ist nur, daß der Absender eine Kopie des öffentlichen Schlüssels des Empfängers besitzt. Dieser eine öffentliche Schlüssel kann von jedem benutzt werden, der mit dem Empfänger kommunizieren will. Somit sind dann insgesamt nur *n* Schlüsselpaare notwendig, wenn *n* Leute verschlüsselt miteinander kommunizieren wollen.

Die Verschlüsselung mit öffentlichem Schlüssel beruht auf sogenannten Falltür-Algorithmen bzw. Einweg-Hashes. Das sind Funktionen, die leicht zu berechnen sind, doch umgekehrt ist es praktisch unmöglich, aus dem Ergebnis dieser Hash-Funktionen wieder den Ausgangswert zu berechnen. So ist es z.B. leicht, zwei Primzahlen miteinander zu multiplizieren, um eine Nichtprimzahl zu erhalten, es ist aber schwer, eine Nichtprimzahl in ihre Primfaktoren zu zerlegen. Falltür-Algorithmen sind ähnlich, haben aber eine “Falltür”. Das heißt: Wenn ein Stück Information bekannt ist, kann man leicht die Umkehrfunktion berechnen. Wenn Sie z.B. eine aus zwei Primfaktoren bestehende Zahl haben, so macht die Kenntnis eines der Faktoren es leicht, den zweiten zu berechnen.

Angenommen, ein Verfahren beruhe auf der Bildung einer Zahl aus Primfaktoren, dann enthält der öffentliche Schlüssel eine aus zwei großen Primfaktoren zusammengesetzte Zahl, und das Verschlüsselungsverfahren benutzt dann diese Nichtprimzahl zum Verschlüsseln der Nachricht. Das Verfahren zum Wiederherstellen dieser Nachricht erfordert dann die Kenntnis der Primfaktoren. So ist die Entschlüsselung möglich, wenn Sie den privaten Schlüssel haben, der einen der Faktoren enthält, ist aber praktisch unmöglich, wenn Sie ihn nicht haben.

Wie bei guten symmetrischen Verschlüsselungsverfahren beruht die Sicherheit auch bei Public-Key-Verfahren ausschließlich auf dem Schlüssel. Aus diesem Grund kann man die Schlüsselgröße als ein Maß für die Sicherheit des Systems nehmen. Allerdings kann man die Größe eines symmetrischen Schlüssels nicht mit der von Public-Key-Verfahren vergleichen, um Rückschlüsse auf deren relative Sicherheit ziehen zu können. Bei einem Brute-Force-Angriff auf eine symmetrische Verschlüsselung mit einer Schlüsselgröße von 80 Bit muß der Angreifer bis zu 2^{80}-1 Schlüssel ausprobieren, um den richtigen Schlüssel zu finden. Bei einem Brute-Force-Angriff auf eine Public-Key-Verschlüsselung muß der Angreifer bei einer Schlüsselgröße von 512 Bit eine in 512 Bit codierte (bis zu 155 Dezimalstellen umfassende) Nichtprimzahl in ihre Primfaktoren zerlegen. Der Rechenaufwand für den Angriff weist je nach der Verschlüsselung gewaltige Unterschiede auf. Während 128 Bit für symmetrische Schlüssel ausreichen, werden angesichts der heutigen Verfahren zur Faktorisierung grosser Zahlen für die meisten Zwecke öffentliche Schlüssel mit 1024 Bits empfohlen.

Hybride Verschlüsselungsverfahren

Public-Key-Verschlüsselung ist kein Allheilmittel. Viele symmetrische Verfahren sind vom Sicherheitsstandpunkt aus betrachtet wirksamer, und die Ver- und Entschlüsselung ist bei Public-Key-Verfahren aufwendiger als bei entsprechenden symmetrischen Systemen, sie sind aber nichtsdestoweniger ein wirksames Werkzeug für den sicheren Austausch von symmetrischen Schlüsseln. Das ist die Idee bei hybriden Verschlüsselungssystemen.

Eine hybride Verschlüsselung benutzt sowohl eine symmetrische Verschlüsselung als auch ein asymmetrisches Public-Key-Verfahren. Die eigentliche Nachricht wird mit einem symmetrischen Sitzungsschlüssel verschlüsselt, welcher von einem Zufallsgenerator erzeugt wird. Dieser Sitzungsschlüssel wird dann mit dem öffentlichen Schlüssel des Empfängers verschlüsselt.

Sowohl PGP als auch GnuPG benutzen hybride Verschlüsselungsverfahren. Der mit dem öffentlichen Schlüssel des Empfängers verschlüsselte Sitzungsschlüssel und die symmetrisch verschlüsselte Nachricht werden automatisch zusammengefaßt. Der geheime Schlüssel des Empfängers wird zum Entschlüsseln des Sitzungsschlüssels verwendet, und dieser wird dann zum Entschlüsseln der eigentlichen Nachricht verwendet.

Ein hybrides Verschlüsselungsverfahren ist immer nur so gut wie der unsicherste Teil, egal ob das die Public-Key-Verschlüsselung oder die symmetrische Verschlüsselung ist. Da die symmetrischen Sitzungsschlüssel bei jedem Vorgang neu erzeugt werden, könnte ein Angreifer - selbst wenn er einen Sitzungsschlüssel entschlüsseln könnte - nur die mit dem betreffenden Sitzungsschlüssel verschlüsselte Nachricht entschlüsseln. Er müßte also für jede weitere Nachricht, die er lesen möchte, erneut einen Sitzungsschlüssel entschlüsseln.

Digitale Unterschriften

Eine Hash-Funktion [1] ist eine kryptographische Prüfsumme. Durch eine eindeutige Funktion wird aus einer Datei eine wesentlich kürzere Datensequenz erzeugt, die ein eindeutiges Abbild der Ursprungsdatei ist.

Die digitale Unterschrift eines Dokumentes ist das Ergebnis der Anwendung einer Hash-Funktion auf das Dokument. Um für digitale Unterschriften brauchbar zu sein, muß die Hash-Funktion jedoch zwei wichtige Eigenschaften haben:

Erstens sollte es unmöglich sein, zwei Dokumente zu finden, die dasselbe Hash-Ergebnis haben. Zweitens sollte es bei einem gegebenen Hash-Ergebnis schwer sein, das ursprünglich Dokument wiederherzustellen, aus dem dieser Hash erzeugt wurde.

Einige Public-Key-Verfahren könnten auch zum Unterschreiben von Dokumenten benutzt werden.[2] Der Unterzeichner verschlüsselt das Dokument mit seinem *privaten* Schlüssel. Jeder, der die Unterschrift prüfen und das Dokument sehen will, benutzt einfach den öffentlichen Schlüssel des Unterzeichners, um das Dokument zu entschlüsseln. Dieses Verfahren besitzt in der Tat die beiden Eigenschaften, die eine gute Hash-Funktion braucht, doch ist es in der Praxis zu langsam, um effektiv nutzbar zu sein.

Besser ist es, spezielle Hash-Algorithmen zu benutzen, welche diese beiden wichtigen Eigenschaften aufweisen; wie beispielsweise SHA1 und RIPE-MD160. Bei einem solchen Verfahren wird der Hash-Wert eines Dokumentes als Unterschrift verwendet. Man kann die Unterschrift dadurch prüfen, daß man auf die Kopie des Dokumentes ebenfalls die Hash-Funktion anwendet und den Hash-Wert, den man erhält, mit dem Hash-Wert des Originaldokumentes vergleicht. Wenn beide Werte übereinstimmen, dann sind beide Dokumente identisch.

Das Problem ist jetzt natürlich, Hash-Funktionen für digitale Unterschriften zu benutzen, ohne einem Angreifer das Manipulieren der Unterschrift zu ermöglichen. Wenn das Dokument und die Unterschrift unverschlüsselt geschickt werden, könnte ein Angreifer das Dokument verändern und eine entsprechende neue Unterschrift erzeugen, ohne daß der Empfänger es merkt. Wenn nur das Dokument verschlüsselt wird, könnte ein Angreifer die Unterschrift verfälschen und so das Scheitern einer Unterschriftsprüfung verursachen.

Eine dritte Möglichkeit besteht darin, ein hybrides Verfahren zu benutzen, um sowohl die Unterschrift als auch das Dokument zu verschlüsseln. Der Unterzeichner benutzt seinen privaten Schlüssel, und jedermann kann dessen öffentlichen Schlüssel benutzen, um die Unterschrift und das Dokument zu prüfen. Dies klingt zwar gut, ist aber in Wirklichkeit Unsinn. Wenn dieses Verfahren das Dokument wirklich sichern könnte, würde es dieses auch gegen Verfälschung sichern, und dann wäre die Unterschrift gar nicht nötig. Das ernstlichere Problem ist jedoch, daß dies keinen Schutz gegen Verfälschung bietet, weder für die Unterschrift noch für das Dokument. Bei diesem Verfahren wird nur der Sitzungsschlüssel für die symmetrische Verschlüsselung unter Benutzung des privaten Schlüssels des Unterzeichners verschlüsselt. Jeder kann den öffentlichen Schlüssel benutzen, um den Sitzungsschlüssel wiederherzustellen. Deshalb ist es für einen Angreifer einfach, den Sitzungsschlüssel wiederherzustellen und ihn zum Verschlüsseln von Ersatzdokumenten und Ersatzunterschriften zu benutzen, die er dann im Namen des Absenders an andere schickt.

Ein wirklich funktionierendes Verfahren ist es, nur die Unterschrift mit einem Public-Key-Verfahren zu verschlüsseln. Das heißt, es wird der geheime Schlüssel des Unterzeichners benutzt, um die digitale Unterschrift zu erzeugen, die dann jeder mit dem dazugehörigen öffentlichen Schlüssel checken kann. Das unterzeichnete Dokument kann man unverschlüsselt verschicken, wenn es öffentlich ist oder verschlüsselt, wenn es vertraulich ist. *Wenn das Dokument nach dem Unterzeichnen verändert wurde, wird die Unterschriftsprüfung negativ ausfallen.* Der von GnuPG standardmäßig benutzte Digital Signature Algorithm (DSA) arbeitet nach dieser Methode.

Fußnoten

1. Eine einfache Hash-Funktion ist $f(x) = 0$ für alle ganzen Zahlen x. Eine interessantere Hash-Funktion ist $f(x) = x \; mod \; 37$, welche x auf den Rest von x dividiert durch 37 abbildet.
2. Die Verschlüsselung muß die Eigenschaft haben, daß der aktuelle öffentliche oder private Schlüssel vom Verschlüsselungsverfahren als der öffentliche Schlüssel benutzt werden könnte. RSA ist ein Beispiel eines solchen Verfahrens, ElGamal dagegen nicht.

Kapitel 2. Grundlagen

Dieses Kapitel führt in die wesentlichen Funktionen des GNU Privacy Guard ein. Hier lernen Sie, wie man Schlüsselpaare erzeugt, Schlüssel austauscht und überprüft, Dokumente verschlüsselt, entschlüsselt und durch digitale Unterschriften authentifiziert.

Wie bereits in Kapitel 1 erwähnt, bedient sich GnuPG eines Public-Key-Verfahrens, um eine sichere Kommunikation zu gewährleisten. In einem solchen System hat jeder Benutzer ein Schlüsselpaar, bestehend aus einem geheimen Schlüssel und einem öffentlichen Schlüssel. Der geheime Schlüssel darf unter keinen Umständen jemand anderem zugänglich sein. Den öffentlichen Schlüssel sollte man für jeden, mit dem man kommunizieren möchte, zugänglich machen.

GnuPG benutzt ein erweitertes Schema, bei dem jeder Benutzer jeweils ein primäres Schlüsselpaar hat und optional weitere untergeordnete Schlüsselpaare haben kann. Das primäre und das untergeordnete Schlüsselpaar werden gebündelt, um die Schlüsselverwaltung zu erleichtern; das Bündel kann vereinfacht als ein Schlüsselpaar betrachtet werden.

Erzeugen eines neuen Schlüsselpaares

Damit Sie GnuPG zum Verschlüsseln, Entschlüsseln oder Signieren einsetzen können, benötigen Sie ein Schlüsselpaar, das aus einem geheimen und einem öffentlichen Schlüssel besteht. Mit der Kommandozeilen-Option `--gen-key` können Sie ein neues primäres Schlüsselpaar erzeugen:

```
alice$  gpg --gen-key
gpg (GnuPG) 1.0.1; Copyright (C) 1999 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Bitte wählen Sie, welche Art von Schlüssel Sie möchten:
   (1) DSA und ElGamal (voreingestellt)
   (2) DSA (nur signieren/beglaubigen)
   (4) ElGamal (signieren/beglaubigen und verschlüsseln)
Ihre Auswahl?
```

Mit GnuPG können Sie verschiedene Typen von Schlüsselpaaren erzeugen, doch muß der primäre Schlüssel Unterschriften liefern können. Es gibt daher nur drei Optionen. Option 1 erzeugt wirklich zwei Schlüsselpaare, nämlich ein DSA-Schlüsselpaar, das nur zum Unterschreiben geeignet ist, und außerdem noch ein untergeordnetes ElGamal-Schlüsselpaar für die Verschlüsselung. Option 2 erzeugt nur das DSA-Schlüsselpaar. Option 4[1] erzeugt ein einzelnes ElGamal-Schlüsselpaar, das sowohl zum Unterzeichnen als auch zum Verschlüsseln verwendbar ist. In allen Fällen ist es möglich, später noch weitere Unterschlüssel für die Verschlüsselung und Unterzeichnung hinzuzufügen. In der Regel sollten Sie hier die Standardoption auswählen.

Als nächstes wählen Sie die Schlüsselgröße. Bei einem DSA-Schlüssel muß diese zwischen 512 und 1024 Bits liegen, ein ElGamal-Schlüssel dagegen kann - zumindest theoretisch - eine beliebige Größe haben. Der GnuPG erfordert es allerdings, daß die Schlüssel nicht kleiner als 768 Bits sind. Wenn Option 1 mit einer Schlüsselgröße von mehr als 1024 Bit gewählt wurde, hat der ElGamal-Schlüssel die verlangte Größe, doch der DSA-Schlüssel wird maximal 1024 Bits haben.

```
Der DSA Schlüssel wird 1024 Bits haben.
Es wird ein neues ELG-E Schlüsselpaar erzeugt.
```

```
              kleinste Schlüssellänge ist  768 Bit
              standard Schlüssellänge ist 1024 Bit
      größte sinnvolle Schlüssellänge ist 2048 Bit
Welche Schlüssellänge wünschen Sie? (1024)
```

Je größer der Schlüssel ist, desto sicherer ist er gegen Brute-Force-Angriffe, doch sollte für die meisten Zwecke die Standard-Schlüsselgröße ausreichend sein, da es einfacher wäre, die Verschlüsselung zu umgehen, als sie zu knacken. Außerdem wird mit zunehmender Schlüsselgröße die Ver- und Entschlüsselung langsamer, und auch die Unterschrift wird länger. Einmal festgelegt, kann die Schlüsselgröße nicht nachträglich geändert werden.

Schließlich müssen Sie noch ein Verfallsdatum wählen. Wenn Option 1 gewählt wurde, gilt dieses Verfallsdatum sowohl für die ElGamal- als auch die DSA-Schlüsselpaare.

```
Bitte wählen Sie, wie lange der Schlüssel gültig bleiben soll.
         0 = Schlüssel verfällt nie
      <n>  = Schlüssel verfällt nach n Tagen
      <n>w = Schlüssel verfällt nach n Wochen
      <n>m = Schlüssel verfällt nach n Monaten
      <n>y = Schlüssel verfällt nach n Jahren
Der Schlüssel bleibt wie lange gültig? (0)
```

Für die meisten Fälle reicht ein Schlüssel ohne Verfallsdatum völlig aus. Allerdings sollte man das Verfallsdatum immer sorgfältig auswählen; denn, obwohl es sich auch noch nachträglich ändern läßt, kann es umständlich sein, das geänderte Verfallsdatum allen Ihren Kommunikationspartnern mitzuteilen.

Im nächsten Schritt müssen Sie eine Benutzer-ID (Benutzer-Kennung) angeben. Das dient dazu, den soeben erzeugten Schlüssel einer realen Person zuzuordnen.

```
Sie benötigen eine User-ID, um Ihren Schlüssel eindeutig zu machen; das
Programm baut diese User-ID aus Ihrem echten Namen, einem Kommentar und
Ihrer E-Mail-Adresse in dieser Form auf:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Ihr Name ("Vorname Nachname"):
```

Es wird zunächst nur eine Benutzer-ID erzeugt, doch können Sie später weitere Benutzer-IDs hinzufügen, wenn Sie den Schlüssel in verschiedenen Situationen benutzen wollen, also beispielsweise bei der Arbeit in Ihrer Firma oder für Ihre politische Arbeit. *Die Benutzer-ID sollten Sie mit aller Sorgfalt wählen, da Sie sie später nicht mehr ändern können.*

Damit Ihr geheimer Schlüssel nicht von anderen mißbraucht werden kann, wird er von GnuPG mit einem symmetrischen Verfahren verschlüsselt. Dazu geben Sie ein sogenanntes "Mantra" (einen Paßwort-Satz) ein, das Sie wiederum jedesmal benötigen, wenn Sie auf Ihren geheimen Schlüssel zugreifen.

```
Sie benötigen ein Mantra, um den geheimen Schlüssel zu schützen.

Geben Sie das Mantra ein:
```

Die Länge des Mantra ist theoretisch unbegrenzt. Sie sollten es mit Sorgfalt auswählen. Unter dem Gesichtspunkt der Sicherheit ist das Mantra einer der schwächsten Punkte im GnuPG (wie auch in anderen Verschlüsselungssystemen mit öffentlichen Schlüsseln), da es Ihr einziger Schutz ist, falls jemand in den Besitz Ihres privaten Schlüssels kommt.

Man sollte für das Mantra keine Wörter aus einem Wörterbuch oder Lexikon nehmen und nicht nur die Buchstaben des Alphabets, sondern auch Sonderzeichen verwenden. Je länger das Mantra ist, desto sicherer ist es, aber andererseits sollten Sie sich das Mantra auch gut merken können; nichts ist fataler als das Mantra auf einem Zettel oder in einer Datei zu notieren. Ein gut gewähltes Mantra ist entscheidend für Ihre Datensicherheit.

Es ist beispielsweise keine gute Idee, einen bekannten Ausspruch oder ein Zitat einer bekannten Persönlichkeit als Mantra zu nehmen. Das würde die Chance erhöhen, das Mantra zu erraten: ein Angreifer könnte einfach den Computer eine Zitatenliste durchprobieren lassen. Am besten denkt man sich einen unsinnigen Satz wie z.B: “Die Currywurst schmeckt nach Zimt und Zucker” oder “Helmut Kohl ist bekanntermaßen Vegetarier” aus. Ihrer Phantasie sind hierbei keine Grenzen gesetzt. Wenn Sie auch noch ein paar Rechtschreibfehler und Sonderzeichen einbauen, ist ein Wörterbuchangriff praktisch unmöglich: “Dat Körriwurst schmöckt nach #imt und #ucker”. *Benutzen Sie auch auf keinen Fall eines der soeben aufgeführten Beispiele!!*.

Erzeugen einer Widerrufurkunde

Nach dem Erzeugen Ihres Schlüsselpaars sollten Sie sofort mit der Option `--gen-revoke` eine Widerrufurkunde für Ihre Schlüssel erzeugen. Wenn Sie Ihr Mantra vergessen oder wenn Ihr privater Schlüssel kompromittiert oder verloren gegangen ist, können Sie mit dieser Widerrufurkunde andere davon in Kenntnis setzen, daß der dazugehörige öffentliche Schlüssel nicht mehr benutzt werden sollte. Ein zurückgerufener öffentlicher Schlüssel kann noch benutzt werden, um Unterschriften zu prüfen, die Sie vor dem Widerruf abgegeben haben, er kann jedoch nicht benutzt werden, um künftige Mitteilungen an Sie zu verschlüsseln. Vorausgesetzt, Sie haben noch Zugang zu Ihrem widerrufenen geheimen Schlüssel, so können Sie selbstverständlich noch Daten entschlüsseln, die vor dem Widerruf für Sie verschlüsselt worden sind.

```
alice$ gpg --output revoke.asc --gen-revoke mykey
[...]
```

wobei **mykey** entweder die Schlüssel-ID Ihres ersten Schlüsselpaares oder irgendein Teil einer dazugehörigen Benutzer-ID ist. Die erzeugte Widerrufurkunde wird in die Datei *revoke.asc*, bzw., wenn man die Option `--output` wegläßt, auf die Standard-Ausgabe geschrieben. Da die Widerrufurkunde kurz ist, ist es kein Problem, eine ausgedruckte Kopie der Widerrufurkunde irgendwo sicher aufzubewahren, z.B. in Ihrem Bankschließfach. Die Widerrufurkunde sollten Sie aber auf keinen Fall an Stellen aufbewahren, zu denen andere Personen Zugang haben, da im Prinzip jeder die Widerrufurkunde veröffentlichen und so den entsprechenden Schlüssel nutzlos machen könnte.

Austauschen von Schlüsseln

Um mit anderen zu kommunizieren, müssen Sie untereinander Ihre öffentlichen Schlüssel austauschen. Zum Auflisten der Schlüssel in Ihrem öffentlichen Schlüsselbund verwenden Sie die Befehlszeilen-Option `--list-keys`.

```
alice$ gpg --list-keys
/users/alice/.gnupg/pubring.gpg
---------------------------------------
pub  1024D/FB5797A9 2000-06-06 Alice (Rechtsanwältin) <alice@cyb.org>
sub  1024g/C8B3998F 2000-06-06
```

Exportieren eines öffentlichen Schlüssels

Um jemandem Ihren öffentlichen Schlüssel zu schicken, müssen Sie diesen zunächst exportieren. Hierzu benutzt man die Kommandozeilen-Option `--export`. Zur Indentifikation des zu exportierenden öffentlichen Schlüssels dient entweder die Schlüssel-ID oder irgendein Teil der Benutzer-ID.

```
alice$ gpg --output alice.gpg --export alice@cyb.org
```

Der Schlüssel wird in einem binären Format exportiert, doch kann dies unerwünscht sein, wenn Sie den Schlüssel per E-Mail verschicken oder auf einer WWW-Seite veröffentlichen wollen. GnuPG unterstützt daher die Kommandozeilen-Option `--armor`[2] die bewirkt, daß der Output im ASCII-Format ausgegeben wird. (Im Allgemeinen kann jeder Output von GnuPG - beispielsweise Schlüssel, verschlüsselte Dokumente oder Unterschriften - im ASCII-Format dargestellt werden, indem man die `--armor`-Option hinzufügt.)

```
alice$ gpg --armor --export alice@cyb.org
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

[...]
-----END PGP PUBLIC KEY BLOCK-----
```

Importieren eines öffentlichen Schlüssels

Ein öffentlicher Schlüssel kann zu Ihrem öffentlichen Schlüsselbund hinzugefügt werden, und zwar mit folgender Option: `--import`

```
alice$ gpg --import blake.gpg
gpg: Schlüssel B2690E6F: Öffentlicher Schlüssel importiert
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:           importiert: 1
alice$ gpg --list-keys
/users/alice/.gnupg/pubring.gpg
---------------------------------------
pub  1024D/FB5797A9 2000-06-06 Alice (Rechtsanwältin) <alice@cyb.org>
sub  1024g/C8B3998F 2000-06-06

pub  1024D/B2690E6F 2000-06-06 Blake (Staatsanwalt) <blake@cyb.org>
sub  1024g/F251B862 2000-06-06
```

Wenn ein Schlüssel einmal importiert ist, sollte er auf Authentizität überprüft werden. GnuPG arbeitet mit einem wirksamen und flexiblen Vertrauensmodell, bei dem Sie nicht jeden Schlüssel persönlich zu authentifizieren brauchen, den Sie importieren. Einige Schlüssel können dies jedoch erfordern. Ein Schlüssel wird dadurch authentifiziert, daß Sie den Fingerabdruck des Schlüssels überpüfen und dann den Schlüssel unterschreiben, um seine Gültigkeit zu bestätigen. Der Fingerabdruck eines Schlüssels kann schnell mit der Befehlszeilen-Option `--fingerprint` geprüft werden, um aber den Schlüssel zu bestätigen, müssen Sie ihn editieren.

```
alice$ gpg --edit-key blake@cyb.org

pub  1024D/B2690E6F  created: 2000-06-06 expires: never      trust: -/q
sub  1024g/F251B862  created: 2000-06-06 expires: never
(1)  Blake (Staatsanwalt) <blake@cyb.org>
```

```
Befehl> fpr
pub  1024D/B2690E6F 2000-06-06 Blake (Staatsanwalt) <blake@cyb.org>
             Fingerprint: 6A51 852C 7491 95B5 C5F0  731C 141F C008 B269 0E6F
```

Um den Fingerabdruck zu überprüfen, müssen Sie den Eigentümer des Schlüssels kontaktieren und die Fingerabdrücke vergleichen. Sie können persönlich oder per Telefon mit ihm sprechen oder auf beliebigem anderen Wege kommunizieren, solange nur garantiert ist, daß es sich um den rechtmäßigen Eigentümer handelt. Stimmen beide Fingerabdrücke überein, dann können Sie sicher sein, daß Sie eine echte Kopie des öffentlichen Schlüssels haben.

Nach dem Prüfen des Fingerabdrucks können Sie den Schlüssel unterschreiben, um ihn zu authentifizieren. Da die Schlüsselüberprüfung ein Schwachpunkt in der Kryptographie mit öffentlichem Schlüssel ist, sollten Sie *äußerste Sorgfalt* walten lassen und den Fingerabdruck eines Schlüssels *immer* gemeinsam mit dem Eigentümer prüfen, bevor Sie den Schlüssel unterschreiben.

```
Befehl> sign

pub  1024D/B2690E6F  created: 2000-06-06 expires: never      trust: -/q
             Fingerprint: 6A51 852C 7491 95B5 C5F0  731C 141F C008 B269 0E6F

     Blake (Staatsanwalt) <blake@cyb.org>

Sind Sie wirklich sicher, daß Sie vorstehenden Schlüssel mit Ihrem
Schlüssel beglaubigen wollen: "Alice (Rechtsanwältin) <alice@cyb.org>"

Wirklich unterschreiben?
```

Sie können sich jederzeit vergewissern, welche Unterschrift Sie hinzugefügt haben. Jede Benutzer-ID auf dem Schlüssel hat dann sowohl eine oder mehrere Eigenbeglaubigungen als auch eine Unterschrift von jedem Benutzer, der den Schlüssel authentifiziert hat.

```
Befehl> check
uid  Blake (Staatsanwalt) <blake@cyb.org>
sig!       B2690E6F 2000-06-06   [Eigenbeglaubigung]
sig!       FB5797A9 2000-06-06   Alice (Rechtsanwältin) <alice@cyb.org>
```

Ver- und Entschlüsseln von Dokumenten

Der öffentliche und der geheime Schlüssel haben jeweils eine spezifische Aufgabe beim Ver- und Entschlüsseln von Dokumenten. Das Public-Key-Verfahren kann man sich wie einen offenen Safe vorstellen. Wenn jemand ein Dokument unter Benutzung eines öffentlichen Schlüssels verschlüsselt, wird dieses Dokument in den Safe gelegt, der Safe geschlossen und das Kombinationsschloß mehrmals verdreht. Der entsprechende geheime Schlüssel ist die Kombination, mit der man den Safe wieder öffnen und das Dokument wieder herausholen kann. Mit anderen Worten, es kann nur die Person, die den geheimen Schlüssel hat, auf ein Dokument zugreifen, das unter Benutzung des dazugehörigen öffentlichen Schlüssels verschlüsselt worden ist.

Das Verfahren für das Ver- und Entschlüsseln von Dokumenten ist bei diesem Modell einfach: eine Nachricht an Alice verschlüsseln Sie unter Verwendung von Alices öffentlichem Schlüssel, und diese entschlüsselt sie mit ihrem geheimen Schlüssel. Umgekehrt geht es genauso: Alice verschlüsselt eine

Nachricht an Sie mit Ihrem öffentlichen Schlüssel, welche Sie dann mit Ihrem geheimen Schlüssel entschlüsseln können.

Um ein Dokument zu verschlüsseln, benutzt man die Option `--encrypt`. Dazu müssen Sie die öffentlichen Schlüssel der vorgesehenen Empfänger haben. Sollten Sie auf der Kommandozeile den Namen der zu verschlüsselnden Datei nicht angeben, werden die zu verschlüsselnden Daten von der Standard-Eingabe gelesen. Das verschlüsselte Resultat wird auf die Standard-Ausgabe oder in die Datei, die durch die Option `--output` spezifiziert ist, geschrieben. Das Dokument wird darüberhinaus auch noch komprimiert.

```
alice$ gpg --output doc.gpg --encrypt --recipient blake@cyb.org doc
```

Mit der Option `--recipient` wird der öffentliche Schlüssel spezifiziert, mit dem das Dokument verschlüsselt werden soll. Entschlüsseln läßt sich das so verschlüsselte Dokument jedoch nur von jemandem mit dem dazugehörigen geheimen Schlüssel. Das bedeutet konsequenterweise aber auch, daß Sie selbst ein so verschlüsseltes Dokument nur wieder entschlüsseln können, wenn Sie Ihren eigenen öffentlichen Schlüssel in die Empfängerliste aufgenommen haben.

Zum Entschlüsseln einer Nachricht wird die Option `--decrypt` benutzt. Sie benötigen dazu den geheimen Schlüssel, für den die Nachricht verschlüsselt wurde und das Mantra, mit dem der geheime Schlüssel geschützt ist.

```
blake$ gpg --output doc --decrypt doc.gpg
Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Blake (Staatsanwalt) <blake@cyb.org>"
1024-Bit ELG-E Schlüssel, ID F251B862, erzeugt 2000-06-06 (Hauptschlüssel-ID B2690E6F)

Geben Sie das Mantra ein:
```

Mit GnuPG können Sie aber auch ohne Anwendung eines Public-Key-Verfahrens Dokumente verschlüsseln und stattdessen ein symmetrisches Verfahren benutzen. Der Schlüssel für die symmetrische Verschlüsselung wird aus einem Paßwort - besser noch, einem Paßwort-Satz - generiert, das *auf gar keinen Fall* dem Mantra zum Schutz Ihres privaten Schlüssels entsprechen sollte. Je länger das gewählte Paßwort ist, desto sicherer ist der Schlüssel. Wenn Sie diesen symmetrischen Schlüssel an jemanden weitergeben, sollten Sie dazu einen sicheren Weg wählen. Ein Dokument läßt sich so durch Benutzung der Option`--symmetric`verschlüsseln.

```
alice$ gpg --output doc.gpg --symmetric doc
Geben Sie das Mantra ein:
```

Symmetrische Verfahren empfehlen sich beispielsweise, wenn Sie die verschlüsselten Daten nicht weiter geben möchten, das Problem der Paßwortübergabe also entfällt. Ein mögliches Anwendungsbeispiel wäre, daß Sie alte E-Mails oder alte Datensätze aus Ihrer Umsatzstatisk auf ihrer Festplatte oder einer CDROM archivieren und vor fremden Zugriffen schützen möchten. Oder Sie können auch ganze Verzeichnisse oder Festplatten verschlüsseln.

Digitale Signaturen

Eine digitale Unterschrift oder Signatur ist am ehesten mit einem Siegel zu vergleichen. Mit dem Siegel wird die Integrität eines Dokumentes bestätigt, das sich in einem Umschlag befindet, und ermöglicht, daß sich eine nachträgliche Manipulation feststellen läßt. Wenn das Dokument nachfolgend in irgendeiner Weise verändert wird, ergibt die Prüfung der Signatur ein negatives Ergebnis. Außerdem ermöglicht die Signatur eine zweifelsfreie Zuordnung des Absenders. Eine digitale Unterschrift kann so demselben Zweck wie eine handgeschriebene Unterschrift dienen mit dem zusätzlichen Vorteil, eine Handhabe gegen Verfälschung zu bieten. Die GnuPG-Quelltextdistribution ist z.B. so unterschrieben, daß die Benutzer nachprüfen können, daß der Quelltext nachträglich nicht verändert worden ist und auch wirklich vom GnuPG-Team stammt.

Die rechtliche Verbindlichkeit von digitalen Unterschriften ist von Land zu Land verschieden. In Deutschland ist das Signaturgesetz augenblicklich einer Novellierung unterworfen. Weitere Informationen und Quellenverweise finden Sie in Kapitel 4.

Bei der Erzeugung und Prüfung von Unterschriften benutzt man das öffentlich/geheime Schlüsselpaar anders als bei der Ver- und Entschlüsselung. Die Unterschrift wird hier mit dem geheimen Schlüssel des Unterzeichnenden erzeugt und dann jeweils mit dem entsprechenden öffentlichen Schlüssel geprüft. So würde beispielsweise Alice ihren geheimen Schlüssel benutzen, um ihren letzten Beitrag für eine Zeitschrift zu signieren. Der Redakteur, der Alices Artikel bearbeitet, benutzt dann ihren öffentlichen Schlüssel, um die Unterschrift zu prüfen und so sicherzustellen, daß der Beitrag wirklich von Alice selbst stammt und auch nicht nachträglich verändert worden ist.

Als Konsequenz aus der Verwendung digitaler Signaturen ergibt sich, daß sich kaum abstreiten läßt, daß man eine digitale Unterschrift geleistet hat, da dies ja bedeuten würde, daß der geheime Schlüssel kompromittiert wurde.

Die Kommandozeilen-Option `--sign` wird zum Erzeugen einer digitalen Unterschrift verwendet. Mit der Option `--output` legen Sie fest, in welche Datei das signierte Dokument geschrieben wird.

```
alice$ gpg --output doc.sig --sign doc

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Alice (Rechtsanwältin) <alice@cyb.org>"
1024-bit DSA Schlüssel, 1024D/FB5797A9, erzeugt 2000-06-06

Geben Sie das Mantra ein:
```

Das Dokument wird vor dem Unterschreiben komprimiert und die Ausgabe erfolgt im binären Format.

Haben Sie ein unterschriebenes Dokument erhalten, können Sie die Unterschrift prüfen, und zwar optional ohne oder mit Entnahme des unterschriebenen Originaldokumentes. Zur bloßen Überprüfung der Unterschrift benutzen Sie die Option `--verify`. Wenn Sie außerdem das unterzeichnete Dokument entnehmen wollen, verwenden Sie die Option `--decrypt`.

```
blake$ gpg --output doc --decrypt doc.sig

gpg: Unterschrift vom Die 06 Jun 2000 17:19:52 CEST, DSA Schlüssel ID FB5797A9
gpg: Korrekte Unterschrift von "Alice (Rechtsanwältin) <alice@cyb.org>"
```

Dokumente mit Klartextsignatur

In Fällen, in denen es unerwünscht ist, das Dokument beim Unterschreiben zu komprimieren, benutzt man die Option `--clearsign`. Das bewirkt, daß eine in ASCII dargestellte Signatur das Dokument wie ein Briefumschlag umgibt, das Dokument an sich aber nicht verändert wird. Der Vorteil dieses Verfahrens ist, daß der Empfänger das Dokument auch ohne Prüfung der Signatur lesen kann.

```
alice$ gpg --clearsign doc
```

```
Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Alice (Rechtsanwältin) <alice@cyb.org>"
1024-Bit DSA Schlüssel, ID FB5797A9, erzeugt 2000-06-06

Geben Sie das Mantra ein:
```

GnuPG markiert dann im Klartext den Anfang des signierten Dokuments und hängt am Ende einen Block mit der eigentlichen OpenPGP-Signatur an.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hier steht irgend ein
von GnuPG signierter Text
[...]
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

iD8DBQE5Pf40WyoKbftXl6kRAsWJAJ4hj7FzPX8M9MWZav9u6yjbHXWGKwCfSiKA
wTaJ/lfY1ETv3R/uJrtGTbI=
=BDOH
-----END PGP SIGNATURE-----
```

Abgetrennte Signatur

Der Nachteil bei signierten Dokumenten ist, daß der Empfänger das Originaldokument aus der unterschriebenen Version erst wiederherstellen muß bzw. bei einem im Klartext unterschriebenen Dokument dieses gegebenenfalls noch editieren muß. Deshalb gibt es als Drittes noch die Möglichkeit, Dokumente mit abgetrennter Unterschrift zu signieren. Dazu verwendet man die Option `--detach-sig`. Die Signatur wird dann in einer separaten Datei abgelegt. Das eigentliche Dokument bleibt unverändert.

```
alice$  gpg --output doc.sig --detach-sig doc

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Alice (Rechtsanwältin) <alice@cyb.org>"
1024-Bit DSA Schlüssel, ID FB5797A9, erzeugt 2000-06-06

Geben Sie das Mantra ein:
```

Um die Signatur zu prüfen, benötigen Sie sowohl das eigentliche Dokument als auch die abgetrennte Unterschrift. Die Option `--verify` kann zum Prüfen der Signatur benutzt werden.

```
blake$  gpg --verify doc.sig doc
gpg: Unterschrift vom Die 06 Jun 2000 17:34:43 CEST, DSA Schlüssel ID FB5797A9
gpg: Korrekte Unterschrift von "Alice (Rechtsanwältin) <alice@cyb.org>"
```

Fußnoten

1. Mit der Option 3 läßt sich ein ElGamal-Schlüsselpaar erzeugen, mit dem Sie keine Unterschriften leisten können.

2. Viele Kommandozeilen-Optionen, die häufig benutzt werden, können auch in einer Konfigurationsdatei definiert werden.

Kapitel 3. Schlüsselverwaltung

Schlüsselverfälschungen sind ein nicht zu unterschätzender Unsicherheitsfaktor bei der Public-Key-Kryptographie. Ein Angreifer kann beispielsweise die Schlüsselbunde eines Benutzers manipulieren oder sich einen öffentlichen Schlüssel mit einer vorgetäuschten Identität erzeugen und ihn an andere zum Herunterladen und Benutzen schicken. Wenn z.B. Chloe unbemerkt die Nachrichten, welche Alice an Blake sendet, lesen will, dann könnte sie folgendermaßen vorgehen: zuerst erzeugt sie ein neues Schlüsselpaar mit einer gefälschten Benutzer-ID. Dann ersetzt sie Alices Kopie von Blakes öffentlichem Schlüssel durch den neuen Schlüssel. Anschließend fängt sie die Nachrichten ab, die Alice an Blake sendet. Diese Nachrichten kann sie dann mit dem neuen geheimen Schlüssel entschlüsseln. Dann verschlüsselt sie die Nachricht wieder, aber diesmal mit dem echten öffentlichen Schlüssel von Blake und schickt sie weiter an Blake. Chloe kann jetzt - ohne daß jemand etwas bemerkt - alle von Alice an Blake geschickten Nachrichten mitlesen.

Eine gute Schlüsselverwaltung ist entscheidend für die Integrität Ihrer eigenen Schlüsselbunde, wie auch der Schlüsselbunde anderer Benutzer. Der Kern der Schlüsselverwaltung von GnuPG ist das *Signieren von Schlüsseln* und verfolgt zwei Hauptzwecke: es erlaubt Ihnen, Verfälschungen an Ihrem Schlüsselbund zu entdecken, und es ermöglicht Ihnen, die Zugehörigkeit eines Schlüssels zu der von der jeweiligen Benutzer-ID genannten Person zu überprüfen. Schlüsselunterschriften werden in einem *Web of Trust* genannten Schema benutzt, um die Authentisierung auch auf Schlüssel auszudehnen, die nicht direkt von Ihnen selbst, sondern von anderen Personen, denen Sie zutrauen, Schlüssel nur nach sorgfältiger Prüfung zu signieren, signiert worden sind. Durch eine gewissenhafte Schlüsselverwaltung können Sie Schlüsselverfälschungen als einen praktischen Angriff auf ihre sichere und vertrauliche Kommunikation abwehren.

Verwaltung Ihres Schlüsselpaares

Ein Schlüsselpaar besteht aus einem öffentlichen und einem geheimen Schlüssel und einem Satz von Benutzer-IDs, um die Schlüssel einer realen Person zuzuordnen. Jeder dieser Bestandteile enthält Informationen über sich selbst. Bei einem öffentlichen Schlüssel sind dies seine ID sowie Angaben darüber, wann er erzeugt worden ist, wann seine Gültigkeit abläuft usw. Bei der Benutzer-ID sind das der Name der realen Person, die durch die ID identifiziert wird, eine optionale Bemerkung sowie eine E-mail-Adresse. Der geheime Schlüssel enthält dagegen keine Informationen über die Benutzer-ID.

Wenn Sie Informationen über ein Schlüsselpaar sehen möchten, dann rufen Sie am besten mit der Kommandozeilen-Option `--edit-key` den Schlüsseleditor auf. Zum Beispiel:

```
chloe$
gpg --edit-key chloe@cyb.org
Geheimer Schlüssel ist vorhanden.

pub  1024D/1B087D04  created: 2000-06-07 expires: never      trust: -/u
sub  2048g/6A3E902A  created: 2000-06-07 expires: never
sub  1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
sub   960D/C0A27DBE  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>

Befehl>
```

Zusammen mit dem öffentlichen Schlüssel wird angezeigt, ob der geheime Schlüssel verfügbar ist oder nicht. Alle Informationen über die Bestandteile des öffentlichen Schlüssels werden dann aufgelistet. Die erste Spalte gibt den Typ des Schlüssels an. Das Schlüsselwort `pub` identifiziert den öffentlichen Hauptschlüssel und das Schlüsselwort `sub` identifiziert einen untergeordneten öffentlichen Schlüssel (Subkey). Die zweite Spalte gibt Länge, Typ und ID des Schlüssels an. Dabei steht `D` für DSA-Schlüssel, `g` für einen nur zur Verschlüsselung geeigneten ElGamal-Schlüssel und `G` für einen ElGamal-Schlüssel, der sowohl zur Verschlüsselung als auch zum Unterschreiben verwendet werden kann. Das Datum der Erzeugung und das Verfallsdatum wird in den Spalten drei und vier angegeben. Die Benutzer-IDs werden nach den Schlüsseln angegeben.

Es stehen noch weitere Befehle zu Verfügung, um zusätzliche Informationen über die Schlüssel zu erhalten. Der Befehl **toggle** schaltet zwischen den öffentlichen und den geheimen Komponenten eines Schlüsselpaares um, wenn tatsächlich beides zur Verfügung steht.

```
Befehl> toggle

sec  1024D/1B087D04  created: 2000-06-07 expires: never
sbb  2048g/6A3E902A  created: 2000-06-07 expires: never
sbb  1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
sbb   960D/C0A27DBE  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>
```

Die Information ist ähnlich der Auflistung für die Komponente des öffentlichen Schlüssels. Das Schlüsselwort `sec` identifiziert den geheimen Hauptschlüssel und das Schlüsselwort `ssb` identifiziert die geheimen Subkeys. Die Benutzer-IDs vom öffentlichen Schlüssel werden der Bequemlichkeit halber auch aufgelistet.

Schlüssel-Integrität

Wenn Sie Ihren öffentlichen Schlüssel weitergeben, so geben Sie damit die öffentlichen Komponenten Ihres Hauptschlüssels und Ihrer Subkeys ebenso wie Ihre Benutzer-IDs weiter. Wenn Sie diese Informationen jedoch ungeschützt weitergeben, so besteht ein Sicherheitsrisiko, da es für einen potentiellen Angreifer möglich ist, den Schlüssel zu verfälschen. Der öffentliche Schlüssel kann durch Hinzufügen oder Ersetzen von Schlüsseln oder von Benutzer-IDs modifiziert werden. Der Angreifer könnte beispielsweise durch Verfälschen der E-Mail-Adresse einer Benutzer-ID die E-Mail an sich selbst umleiten. Durch Veränderung der öffentlichen Schlüssel wäre der Angreifer auch in der Lage, die zu ihm umgeleiteten Nachrichten zu entschlüsseln.

Die Benutzung digitaler Signaturen ist die Lösung für dieses Problem. Indem man den öffentlichen Schlüssel sowie die Benutzer-IDs mit seinem geheimen Schlüssel unterzeichnet, lassen sich Verfälschungen daran leicht feststellen. Dieser Vorgang wird Eigenbeglaubigung genannt; ein öffentlicher Schlüssel, der eigenbeglaubigte Benutzer-IDs enthält, wird *Zertifikat* genannt.

Ein Beispiel: Chloe hat zwei Benutzer-IDs und drei untergeordnete öffentliche Schlüssel bzw. Subkeys. Die Unterschriften auf den Benutzer-IDs können mit dem Befehl **check** im Schlüsseleditior geprüft werden.

```
chloe$  gpg --edit-key chloe
geheimer Schlüssel ist vorhanden.

pub  1024D/1B087D04  created: 2000-06-07 expires: never      trust: -/u
sub  2048g/6A3E902A  created: 2000-06-07 expires: never
```

```
sub  1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
sub   960D/C0A27DBE  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>

Befehl> check
uid  Chloe (Journalistin) <chloe@cyb.org>
sig!       1B087D04 2000-06-07   [Eigenbeglaubigung]
uid  Chloe (Freie Autorin) <chloe@tel.net>
sig!       1B087D04 2000-06-07   [Eigenbeglaubigung]
```

Wie erwartet, wird für jede Unterschrift der primäre Schlüssel mit der Schlüssel-ID `0x26B6AAE1` genommen. Die Eigenbeglaubigungen auf den Subkeys sind in dem öffentlichen Schlüssel enthalten, doch werden sie vom Schlüsseleditor nicht gezeigt.

Editieren von Schlüsseln

Zu Ihrem ursprünglichen Schlüsselpaar können Sie später sowohl neue Subkeys als auch neue Benutzer-IDs hinzufügen. Eine neue Benutzer-ID wird durch Verwendung des Befehls **adduid** erzeugt. Dabei werden Sie wieder nach Ihrem wirklichem Namen, E-Mail-Adresse und einer optionalen Bemerkung gefragt. Ein Subkey wird durch Verwendung des Befehls **addkey** hinzugefügt und kann von beliebigem Typ sein. Das ist so ähnlich, wie Sie es vom Erzeugen Ihres anfänglichen Schlüsselpaares kennen. Wenn Sie einen neuen Subkey oder eine neue Benutzer-ID erzeugen, so werden diese mit Ihrem geheimen Schlüssel eigenbeglaubigt; deshalb müssen Sie auch Ihr Mantra eingeben, wenn der Schlüssel erzeugt wird.

Zusätzliche Benutzer-IDs sind nützlich, wenn Sie für verschiedene Zwecke verschiedene IDs benötigen. So wollen Sie vielleicht eine Benutzer-ID für Ihre Arbeit, eine für Ihre politische Tätigkeit und eine weitere für private Korrespondenz haben. Ihre Mitarbeiter und Geschäftspartner, Politische Mitstreiter und Freunde werden Sie dann jeweils unter einer anderen ID kennen.

Zusätzliche Subkeys sind ebenfalls nützlich. Die zu Ihrem primären öffentlichen Schlüssel gehörigen Benutzer-IDs werden von den Leuten, mit denen Sie kommunizieren, authentisiert, deshalb erfordert eine Änderung des primären Schlüssels eine nochmalige Bestätigung. Wenn Sie mit vielen Leuten kommunizieren, kann dies schwierig und zeitaufwendig sein. Andererseits ist es gut, von Zeit zu Zeit die Subkeys für die Verschlüsselung zu ändern. Wenn ein Schlüssel kompromittiert wurde, ist die Sicherheit aller mit diesem Schlüssel verschlüsselten Daten gefährdet. Durch das Ändern der Schlüssel erreichen Sie jedoch, daß in der Zukunft zu verschlüsselnde Daten nicht auch noch gefährdet werden.

Subkeys und Benutzer-IDs können auch gelöscht werden. Dazu müssen Sie diese zunächst im Schlüsseleditor auswählen, indem Sie die Befehle **key** bzw. **uid** benutzen. So wählt beispielsweise der Befehl **key *2*** den zweiten Subkey aus; ein nochmaliger Aufruf des Befehls **key *2*** macht diese Auswahl wieder rückgängig. Wird **key** ohne Argument aufgerufen, wird die komplette Auswahl an Subkeys wieder aufgehoben. Das gleiche gilt für den Befehl **uid**. Wenn Sie die zu löschenden Benutzer-IDs ausgewählt haben, werden diese mit dem Befehl **deluid** aus Ihrem Schlüssel entfernt. Ebenso löscht der Befehl **delkey** alle ausgewählten Subkeys aus Ihren öffentlichen und geheimen Schlüsseln.

Für die lokale Schlüsselverwaltung ist das Löschen von Schlüssel-Komponenten ein geeignetes Mittel, um die öffentlichen Schlüssel anderer von unnötigem Ballast frei zu halten. Hingegen sollten Sie normalerweise keine Benutzer-IDs und Subkeys von Ihrem eigenen Schlüssel entfernen, da Sie so die Weiterverbreitung dieses Schlüssels verkomplizieren. Wenn ein anderer GnuPG-Benutzer Ihren aktuellen öffentlichen Schlüssel importiert, wird dieser standardmäßig mit dessen alter Kopie Ihres öffentlichen Schlüssels zusammengeführt. Dadurch werden effektiv alle Komponenten wieder hergestellt, die Sie gelöscht haben. Um den Schlüssel wirklich zu aktualisieren, müßte der Benutzer zuerst die alte Version Ihres Schlüssels löschen und dann die neue Version importieren. Dies bringt eine zusätzliche Belastung für Ihre Kommunikationspartner mit sich. Es ist daher auch keine gute Idee, Ihren aktualisierten Schlüssel zu einem Key-Server zu schicken. Zum Aktualisieren Ihres eigenen Schlüssels ist es folglich besser, die jeweiligen Schlüsselkomponenten zu widerrufen, statt sie zu löschen.

Widerrufen von Schlüssel-Komponenten

Um einen Subkey zu widerrufen, wählen Sie ihn im Schlüsseleditor aus, dann können Sie ihn mit dem Befehl **revkey** widerrufen. Der Schlüssel wird dadurch widerrufen, daß man dem Schlüssel eine Widerruf-Unterschrift hinzufügt. Anders als bei der Kommandozeilen-Option `--gen-revoke` tritt der Widerruf sofort in Kraft.

```
Befehl> key 2

pub  1024D/1B087D04  created: 2000-06-07 expires: never      trust: -/u
sub  2048g/6A3E902A  created: 2000-06-07 expires: never
sub* 1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
sub   960D/6E82436B  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>

Befehl> revkey
Möchten Sie diesen Schlüssel wirklich wiederrufen? j

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Chloe (Journalistin) <chloe@cyb.org>"
1024-Bit DSA Schlüssel, ID 1B087D04, erzeugt 2000-06-07


pub  1024D/1B087D04  created: 2000-06-07 expires: never      trust: -/u
sub  2048g/6A3E902A  created: 2000-06-07 expires: never
sub  1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
rev! subkey has been revoked: 2000-06-07
sub   960D/6E82436B  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>
```

Beim Widerrufen einer Benutzer-ID wird anders verfahren. Durch Unterschriften auf einer Benutzer-ID wird bestätigt, daß der Eigentümer des Schlüssels tatsächlich identisch mit der in der Benutzer-ID genannten Person ist. In der Theorie beschreibt eine Benutzer-ID eine Person für immer, da diese Person sich nie ändert. In der Praxis können sich jedoch Elemente der Benutzer-ID, so z.B. die E-Mail-Adresse oder eine Bemerkung, mit der Zeit verändern und so die Benutzer-ID unbrauchbar machen.

Die Spezifikation von OpenPGP unterstützt den Widerruf einer Benutzer-ID nicht. Man kann sich aber dadurch helfen, daß man seine Eigenbeglaubigung für die entsprechende Benutzer-ID widerruft. Aus den zuvor beschriebenen Sicherheitsgründen werden die Korrespondenzpartner keiner Benutzer-ID ohne gültige Eigenbeglaubigung trauen, GnuPG lehnt den Import eines solchen Schlüssels sogar gänzlich ab.

Eine Unterschrift wird unter Verwendung des Befehls **revsig**. widerrufen. Da Sie eine beliebige Zahl von Benutzer-IDs unterschrieben haben können, verlangt der Schlüsseleditor von Ihnen für jede Unterschrift eine Entscheidung, ob sie widerrufen werden soll oder nicht.

```
Befehl> revsig
```

```
Befehl> revsig
Sie haben folgende User-IDs beglaubigt:
     Chloe (Journalistin) <chloe@cyb.org>
   beglaubigt durch 1B087D04 um 2000-06-07
   beglaubigt durch 1B087D04 um 2000-06-07
User-ID: "Chloe (Journalistin) <chloe@cyb.org>"
unterschrieben mit Ihrem Schlüssel 1B087D04 um 2000-06-07
Ein Widerrufszertifikat für diese Unterschrift erzeugen (j/N)n
User-ID: "Chloe (Freie Autorin) <chloe@tel.net>"
unterschrieben mit Ihrem Schlüssel 1B087D04 um 2000-06-07
Ein Widerrufszertifikat für diese Unterschrift erzeugen (j/N)j
Es werden nun folgende Beglaubigungen entfernt:
     Chloe (Freie Autorin) <chloe@tel.net>
   beglaubiigt durch 1B087D04 um 2000-06-07
Wirklich ein Unterschrift-Widerrufszertifikat erzeugen? (j/N) j

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Chloe (Journalistin) <chloe@cyb.org>"
1024-Bit DSA Schlüssel, ID 1B087D04, erzeugt 2000-06-07


pub  1024D/1B087D04  created: 2000-06-07 expires: never      trust: -/u
sub  2048g/6A3E902A  created: 2000-06-07 expires: never
sub  1792G/7D5D4DAE  created: 2000-06-07 expires: 2002-06-07
rev! subkey has been revoked: 2000-06-07
sub   960D/6E82436B  created: 2000-06-07 expires: 2002-06-07
(1)  Chloe (Journalistin) <chloe@cyb.org>
(2)  Chloe (Freie Autorin) <chloe@tel.net>
```

Eine widerrufene Benutzer-ID wird durch die Widerrufs-Signatur auf der Benutzer-ID angezeigt, wenn die Unterschriften auf den Benutzer-IDs des Schlüssels aufgelistet werden.

```
Befehl check

uid  Chloe (Journalistin) <chloe@cyb.org>
sig!       1B087D04 2000-06-07   [Eigenbeglaubigung]
uid  Chloe (Freie Autorin) <chloe@tel.net>
rev!       1B087D04 2000-06-07   [Widerruf]
sig!       1B087D04 2000-06-07   [Eigenbeglaubigung]
```

Ein Widerruf sowohl der Subkeys als auch der Eigenbeglaubigung auf Benutzer-IDs fügt dem Schlüssel eine Widerrufs-Signatur hinzu. Da also nur etwas hinzugefügt und nichts gelöscht wird, ist ein Widerruf für andere stets sichtbar, wenn Ihr aktueller öffentlicher Schlüssel weitergegeben und mit anderen älteren Kopien davon zusammengeführt wird. Der Widerruf garantiert deshalb, daß jeder die aktuelle Kopie Ihres öffentlichen Schlüssels haben kann.

Aktualisieren des Verfallsdatums

Das Verfallsdatum eines Schlüssels kann mit dem Befehl **expire** im Schlüsseleditor aktualisiert werden. Wenn kein Schlüssel ausgewählt ist, wird das Verfallsdatum des primären Schlüssels aktualisiert, ansonsten das des jeweils ausgewählten Subkeys.

Das Verfallsdatum eines Schlüssels ist mit der Eigenbeglaubigung des Schlüssels verbunden. Es wird dadurch aktualisiert, daß man die alte Eigenbeglaubigung löscht und eine neue hinzufügt. Da die Korrespondenzpartner die alte Eigenbeglaubigung noch nicht gelöscht haben, werden sie eine zusätzliche Eigenbeglaubigung auf dem Schlüssel sehen, wenn sie ihre Kopie Ihres Schlüssels aktualisieren. Die jüngste Eigenbeglaubigung hat jedoch jeweils Vorrang, und so werden alle Korrespondenzpartner unzweideutig die Verfallsdaten Ihrer Schlüssel kennen.

Authentisieren anderer Schlüssel

Wie in Kapitel 2 bereits bereits ausführlich besprochen, wird der öffentliche Schlüssel eines Korrespondenzpartners dadurch authentisiert, daß Sie persönlich den Fingerabdruck seines Schlüssels prüfen und dann seinen öffentlichen Schlüssel mit Ihrem geheimen Schlüssel unterschreiben. Durch das persönliche Prüfen des Fingerabdrucks können Sie sicher sein, daß der Schlüssel wirklich ihm gehört. Da Sie den Schlüssel unterschrieben haben, können Sie sicher sein, jede Verfälschung an ihm in der Zukunft zu entdecken. Leider ist dieses Verfahren umständlich, wenn Sie entweder eine große Zahl von Schlüsseln authentisieren müssen oder wenn Sie mit Leuten kommunizieren, welche Sie nicht persönlich kennen.

GnuPG geht dieses Problem mit einem Mechanismus an, der allgemein als *Web of Trust* bezeichnet wird. Im Web of Trust wird die Verantwortlichkeit für das Authentisieren öffentlicher Schlüssel an Personen übertragen, denen Sie zutrauen, bei der Authentisierung von Schlüsseln die nötige Sorgfalt walten zu lassen. Nehmen Sie zum Beispiel folgendes an:

- Alice hat Blakes Schlüssel unterschrieben und
- Blake hat Chloes Schlüssel und Dharmas Schlüssel unterschrieben.

Wenn Alice Blake hinsichtlich der ordnungsgemäßen Authentisierung von Schlüsseln vertraut, dann kann sie davon ausgehen, daß Chloes und Dharmas Schlüssel gültig sind, ohne daß sie diese persönlich prüfen muß. Sie benutzt einfach ihre authentisierte Kopie von Blakes öffentlichem Schlüssel, um zu prüfen, daß Blakes Unterschriften auf den öffentlichen Schlüsseln von Chloe und Dharma echt sind. Im allgemeinen wird, wenn Alice bei allen Partnern völlig darauf vertraut, daß diese die von ihnen unterschriebenen Schlüssel richtig authentisieren, auch jeder mit einem gültigen Schlüssel unterschriebene Schlüssel als gültig betrachtet. Der Ausgangspunkt ist Alices Schlüssel, dessen Gültigkeit vorausgesetzt wird.

Vertrauen in den Eigentümer eines Schlüssels

Vertrauen ist in der Praxis natürlich immer subjektiv. So ist beispielsweise Blakes Schlüssel für Alice gültig, da sie ihn selbst unterschrieben hat, aber vielleicht traut sie Blake kein richtiges Authentisieren der von ihm unterschriebenen Schlüssel zu. In diesem Fall könnte sie die Gültigkeit von Chloes und Dharmas Schlüssel bezweifeln, da sich diese nur auf Blakes Unterschrift stützt. Das Web of Trust trägt diesem Umstand Rechnung, indem es jedem öffentlichen Schlüssel in Ihrem Schlüsselbund eine Angabe darüber zuordnet, inwieweit Sie dem Eigentümer des Schlüssels dahingehend vertrauen, daß er Schlüssel erst nach gründlicher Prüfung authentisiert. Es gibt vier Vertrauensstufen:

Unbekannt

Es ist nichts über die Fähigkeit des Eigentümers bekannt, Schlüssel vor dem Signieren zu authentisieren. Alle Schlüssel in Ihrem öffentlichen Schlüsselbund, die Ihnen nicht gehören, fallen zunächst unter diese Vertrauensstufe.

Kein Vertrauen

Der Eigentümer ist dafür bekannt, andere Schlüssel nicht korrekt zu unterschreiben.

Teilweises Vertrauen

Der Eigentümer versteht die Implikationen des Unterschreibens von Schlüsseln und authentisiert Schlüssel richtig, bevor er sie unterschreibt.

Volles Vertrauen

Der Eigentümer hat ein ausgezeichnetes Verständnis hinsichtlich des Unterschreibens von Schlüsseln, und seine Unterschrift auf einem Schlüssel wäre so gut wie Ihre eigene.

Das Vertrauensmaß eines Schlüssels ist etwas, das Sie alleine dem Schlüssel zuordnen, und es wird als private Information betrachtet. Es wird nicht mit dem Schlüssel verpackt, wenn dieser exportiert wird; es wird sogar getrennt von Ihren Schlüsselbunden in einer gesonderten Trustdatenbank (`trustdb.gpg`) gespeichert.

Der GnuPG-Schlüsseleditor kann benutzt werden, um das Maß Ihres Vertrauens in den Eigentümer eines Schlüssels anzugeben. Der Befehl lautet **trust** (Andererseits fragt GnuPG auch nach, wenn es die Information braucht und noch kein Vertrauensmaß angegeben wurde). In diesem Beispiel gibt Alice das Maß ihres Vertrauens zu Blake an und aktualisiert dann entsprechend die Trustdatenbank, um neu zu ermitteln, welche Schlüssel auf der Basis ihrer neuen Einstufung von Blake gültig sind.

```
alice$  gpg --edit-key blake

pub  1024D/B2690E6F  created: 2000-06-06 expires: never      trust: -/f
sub  1024g/F251B862  created: 2000-06-06 expires: never
(1)  Blake (Staatsanwalt) <blake@cyb.org>

Befehl> trust
pub  1024D/B2690E6F  created: 2000-06-06 expires: never      trust: -/f
sub  1024g/F251B862  created: 2000-06-06 expires: never
(1)  Blake (Staatsanwalt) <blake@cyb.org>

Bitte entscheiden Sie, inwieweit Sie diesem User zutrauen,
den Schlüssel eines anderen Users korrekt zu prüfen (Vergleich mit
Lichtbildausweisen, Vergleich der Fingerabdrücke aus unterschiedlichen
Quellen ...)?

 1 = Weiß nicht so recht
 2 = Kein Vertrauen
 3 = Ich vertraue ihm normalerweise
 4 = Ich vertraue ihm vollständig
 s = Bitte weitere Informationen anzeigen
 m = Zurück zum Menü

Ihre Auswahl? 3

pub  1024D/B2690E6F  created: 2000-06-06 expires: never      trust: m/f
sub  1024g/F251B862  created: 2000-06-06 expires: never
(1)  Blake (Staatsanwalt) <blake@cyb.org>

Befehl> quit
```

Das Vertrauen [1] in den Schlüssel-Eigentümer und in die Gültigkeit des Schlüssels wird rechts neben dem Schlüssel angezeigt. An erster Stelle wird das Vertrauen in den Eigentümer angezeigt, dann das Vertrauen in die Gültigkeit des Schlüssels. Die vier Vertrauensstufen werden folgendermaßen abgekürzt:

- Unbekannt (`q`),
- kein Vertrauen (`n`),
- teilweises Vertrauen (`m`) und
- volles Vertrauen (`f`)

* *??? (pn)*

In diesem Fall ist Blakes Schlüssel voll gültig, da Alice ihn selbst unterschrieben hat. Anfangs fallen Blakes Schlüssel für sie unter die Vertrauensstufe “Unbekannt”, doch sie entscheidet sich dafür, ihn unter “Teilweises Vertrauen“ einzustufen.

Authentisieren von Schlüsseln im Web of Trust

Das Web of Trust ist ein flexibleres und komfortableres Verfahren zur Authentisierung eines Schlüssels. Früher wurde ein Schlüssel nur dann als gültig betrachtet, wenn er von Ihnen persönlich unterzeichnet war. Nach diesem Verfahren wird jetzt auch ein Schlüssel *K* als gültig betrachtet, wenn er die folgenden zwei Bedingungen erfüllt:

1. Schlüssel *K* ist von genügend gültigen Schlüsseln unterschrieben, das heißt, daß er entweder
   - von *Ihnen persönlich* oder
   - von *einem Schlüssel vollen Vertrauens* oder
   - von *drei Schlüsseln teilweisen Vertrauens* unterschrieben wurde.
2. Der Pfad unterschriebener Schlüssel, der vom Schlüssel *K* zurück zu Ihrem eigenen Schlüssel führt, besteht aus *maximal fünf Schritten*.

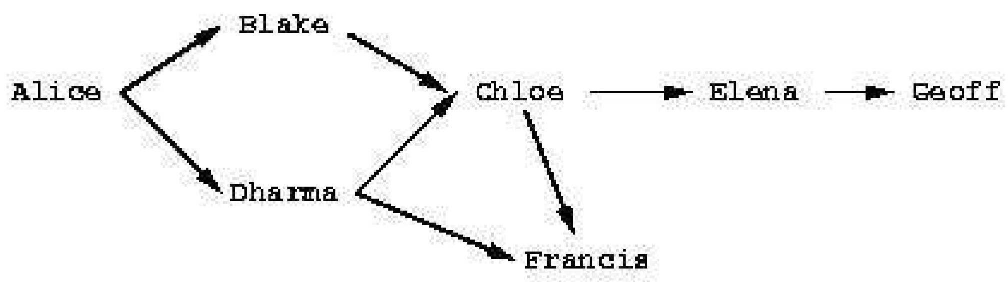
Die Pfadlänge, die Anzahl der erforderlichen Schlüssel Ihres teilweisen Vertrauens und die erforderliche Anzahl der Schlüssel Ihres vollen Vertrauens können Ihrer jeweiligen Vertrauensstufe angepaßt werden. Die oben angegebenen Zahlen sind die von GnuPG benutzten Standardwerte.

Abbildung 3-1 zeigt ein Web of Trust, das seinen Ausgangspunkt bei Alice hat. Das Diagramm zeigt anschaulich, wer wessen Schlüssel unterschrieben hat und welche Schlüssel Alice aufgrund ihres Vertrauens in die anderen Mitglieder des Web of Trust als gültig betrachtet.

* *Potential bug: `--completes-needed` on command line seems to be ignored when combined with `--update-trustdb`. Value is taken correctly if put in options file, however.*

In diesem Beispiel wird angenommen, daß zwei Schlüssel teilweisen Vertrauens oder ein Schlüssel vollen Vertrauens benötigt werden, um einen anderen Schlüssel zu authentisieren. Die maximale Pfadlänge beträgt drei Schritte.

Abbildung 3-1. Ein hypothetisches Vertrauensnetz



Vertrauen		Gültigkeit	
teilweise	**völlig**	**teilweise**	**völlig**
	Dharma		Blake, Chloe, Dharma, Francis
Blake, Dharma		Francis	Blake, Chloe, Dharma
Chloe, Dharma		Chloe, Francis	Blake, Dharma
Blake, Chloe, Dharma		Elena	Blake, Chloe, Dharma, Francis
	Blake, Chloe, Elena		Blake, Chloe, Elena, Francis

* **** überarbeiten und mit abb 3-1 abgleichen*

Beim Berechnen der gültigen Schlüssel in dem Beispiel gilt folgendes: Blakes und Dharmas Schlüssel werden immer als voll gültig betrachtet, da sie direkt von Alice unterschrieben worden sind. Die Gültigkeit der anderen Schlüssel hängt vom Vertrauen ab. Im ersten Fall genießt Dharma volles Vertrauen, woraufhin die Schlüssel von Chloe und Francis als gültig betrachtet werden. Im zweiten Beispiel genießen Blake und Dharma nur teilweises Vertrauen. Da nun zwei Schlüssel teilweisen Vertrauens nötig sind, um einen Schlüssel voll zu authentisieren, wird der Schlüssel von Chloe als voll gültig, der von Francis aber nur als teilweise gültig betrachtet. Falls Chloe und Dharma nur teilweises Vertrauen genießen, wird Chloes Schlüssel nur teilweise gültig sein, wähend Dharmas Schlüssel voll gültig ist. Der Schlüssel von Francis jedoch wird ebenfalls nur als teilweise gültig betrachtet, da nur ein voll gültiger Schlüssel zur Authentisierung anderer Schlüssel benutzt werden kann, und Dharmas Schlüssel der einzige voll gültige Schlüssel ist, der zum Unterschreiben des Schlüssels von Francis benutzt worden ist. Wenn teilweises Vertrauen in Blakes Schlüssel hinzukommt, kann Chloes Schlüssel voll gültig werden und kann dann zur vollen Authentisierung des Schlüssels von Francis und zur teilweisen Authentisierung des Schlüssels von Elena benutzt werden. Wenn schließlich Blake, Chloe und Elena volles Vertrauen genießen, reicht dies noch nicht aus, um den Schlüssel von Geoff zu authentisieren, da die maximal zulässige Länge des Zertifizierungspfades aus drei Schritten bestehen soll, die Pfadlänge von Geoff zurück zu Alice jedoch vier Schritte beträgt.

* *gehört vielleicht nach dailyuse*

Das Web of Trust ermöglicht es Ihnen, GnuPG genau Ihren Vorstellungen von Sicherheit anzupassen. Sie könnten beispielsweise auf mehreren kurzen Pfaden von Ihrem Schlüssel aus zu einem anderen Schlüssel

K bestehen, um diesem zu vertrauen. Vielleicht entscheiden Sie sich aber auch für längere Pfade oder sogar nur einen Pfad von Ihrem Schlüssel zu dem anderen Schlüssel *K*. Wenn Sie mehrfache kurze Pfade voraussetzen, so ist das eine starke Garantie dafür, daß Schlüssel *K* demjenigen gehört, von dem Sie dies annehmen. Der Preis dafür ist natürlich, daß die Authentisierung von Schlüsseln schwieriger ist, da Sie persönlich mehr Schlüssel unterschreiben müssen, als wenn Sie weniger und dafür längere Pfade akzeptieren.

Weitergabe von Schlüsseln

Im Idealfall wird ein Schlüssel durch persönliche Übergabe an Ihre Korrespondenzpartner weitergegeben. In der Praxis werden jedoch Schlüssel oft per E-Mail oder irgendein anderes elektronisches Kommunikationsmittel weitergegeben. Die Weitergabe per E-Mail ist durchaus annehmbar, wenn Sie nur einige wenige Korrespondenzpartner haben. Wenn Sie viele Korrespondenzpartner haben, könnten Sie beispielsweise Ihre(n) öffentlichen Schlüssel auf Ihrer Homepage im Web publizieren. Das setzt jedoch voraus, daß Ihre Korrespondenzpartner auch wissen, wo sie Ihre(n) Schlüssel finden können.

Um dieses Problem zu lösen, gibt es Key-Server, die öffentliche Schlüssel sammeln und weitergeben. Ein bei dem Server eingegangener öffentlicher Schlüssel wird entweder der Datenbank des Servers hinzugefügt oder mit Ihrem eventuell schon vorhandenen Schlüssel zusammengeführt. Wenn eine Anfrage nach einem Schlüssel beim Server eingeht, durchsucht dieser seine Datenbank und sendet den angeforderten öffentlichen Schlüssel zurück, wenn er ihn gefunden hat.

* *wk: bin da anderer Meinung! GnuPG und Keyserver werden das in Zukunft nicht mehr zulassen*

Ein Schlüssel-Server ist auch sinnvoll, wenn viele Leute häufig die Schlüssel anderer Leute unterschreiben. Ohne einen Schlüssel-Server würde Blake, wenn er Alices Schlüssel unterschreibt, an Alice eine Kopie ihres von ihm unterschriebenen Schlüssels schicken, so daß Alice den so aktualisierten Schlüssel ihrem Schlüsselbund hinzufügen und ihn auch an alle ihre Korrespondenzpartner weitergeben könnte. Mit dieser Mühe genügen Alice und Blake weitgehend ihrer Verantwortung gegenüber der Allgemeinheit durch den Aufbau engmaschiger Vertrauensnetze und helfen so, die Sicherheit von GPG zu verbessern. Dies ist jedoch sehr lästig, wenn das Unterschreiben von Schlüsseln häufig vorkommt.

Durch die Benutzung eines Schlüssel-Servers wird das etwas leichter. Wenn nun Blake Alices Schlüssel unterschreibt, so schickt er den unterschriebenen Schlüssel an den Schlüssel-Server, welcher dann Blakes Unterschrift seiner Kopie von Alices Schlüssel hinzufügt. Personen, die daran interessiert sind, ihre Kopie von Alices Schlüssel zu aktualisieren, wenden sich dann selbständig an den Schlüssel-Server, um sich den aktualisierten Schlüssel zu holen. Alice braucht sich mit der Weitergabe überhaupt nicht zu befassen und kann Unterschriften auf ihrem Schlüssel wie jeder andere auch einfach durch Anfrage bei einem Schlüssel-Server holen.

* `--keyserver` *must come before* `--send-keys` *or* `--recv-key`*. This appears to be a bug. ?? TESTEN!*

Ein oder mehr Schlüssel können unter Verwendung der Kommandozeilen-Option `--send-keys` an den Key-Server geschickt werden. Die Option erwartet eine Schlüssel-ID oder Benutzer-ID als Argument und schickt die so spezifizierten Schlüssel an den Key-Server. Der Key-Server, an den die Schlüssel geschickt werden sollen, wird durch die Kommandozeilen-Option `--keyserver` spezifiziert. In ähnlicher Weise wird die Option `--recv-keys` benutzt, um Schlüssel von einem Key-Server zu holen, doch müssen Sie hier den Schlüssel mit einer Schlüssel-ID spezifizieren. Im folgenden Beispiel aktualisiert Alice ihren öffentlichen Schlüssel mit neuen Unterschriften vom Key-Server `blackhole.pca.dfn.de` und schickt dann ihre Kopie von Blakes öffentlichem Schlüssel ebenfalls dorthin, um alle neuen Unterschriften, die sie hinzugefügt hat, weiterzugeben.

```
alice$ gpg --keyserver wwwkeys.de.pgp.net --recv-key FB5797A9
gpg: Schlüssels FB5797A9 von wwwkeys.de.pgp.net wird angefordert ...
gpg: Schlüssel FB5797A9: 1 neue Signatur
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:          neue Signaturen: 1
alice$ gpg --keyserver wwwkeys.de.pgp.net --send-key blake@cyb.org
gpg: Senden an ‘wwwkeys.de.pgp.net’ erfolgreich (status=200)
```

Weltweit gibt es eine Vielzahl bekannter Key-Server. Die größeren Key-Server synchronisieren sich wechselseitig. Am Besten benutzen Sie einen gut erreichbaren Key-Server im Internet und tauschen dann regelmäßig über diesen Schlüssel aus. Eine kleine Auswahl gängiger Key-Server finden Sie im Anhang B des Buches.

Fußnoten

1. GnuPG überfrachtet das Wort “Vertrauen”, indem sowohl “Vertrauen in einen Eigentümer” als auch “Vertrauen in einen Schlüssel” gemeint sein kann. Dies kann Verwirrung stiften. Manchmal wird das Vertrauen in einen Eigentümer zur klareren Unterscheidung als *Ownertrust* bezeichnet. In diesem Handbuch ist jedoch der Begriff “Vertrauen” durchweg in der Bedeutung “Vertrauen in den Eigentümer eines Schlüssels” benutzt worden, und der Begriff “Gültigkeit” bezieht sich darauf, daß ein Schlüssel der mit der Schlüssel-ID verknüpften Person gehört.

Kapitel 4. GnuPG im Alltagsgebrauch

GnuPG ist nicht nur eine komplexe Software, sondern es gibt auch einige technische, gesellschaftliche und rechtliche Aspekte, die berücksichtigt werden sollten:

- Technisch muß es verschiedenen Situationen mit drastisch unterschiedlichen Sicherheitsanforderungen gerecht werden, was die Schlüsselverwaltung kompliziert.
- Die Benutzung von GnuPG ist nicht unbedingt eine rein persönliche Entscheidung. Um GnuPG effektiv nutzen zu können, müssen beide miteinander kommunizierenden Seiten es benutzen.
- Die Haltung der Gesetzgeber zur elektronischen Verschlüsselung und zu digitalen Signaturen unterscheidet sich von Land zu Land. Insbesondere die Frage einer legalen Benutzung von GnuPG bzw. Verschlüsselung im allgemeinen steht gegenwärtig bei vielen nationalen Regierungen zur Debatte.

Auf diese Aspekte wollen wir im folgenden eingehen.

Definition Ihres Sicherheitsbedarfs

Einer der wichtigsten Gründe, GnuPG zu benutzen, ist der Schutz Ihrer Privatsphäre. Das bedeutet, daß Sie mit anderen korrespondieren können, ohne daß Dritte die Möglichkeit haben, mitzulesen, und daß Sie vertrauliche Daten auf Ihrem Rechner dem unbefugten Zugriff anderer entziehen können. Ebenso gibt Ihnen GnuPG die Möglichkeit, Ihre Daten (E-Mail) durch digitale Signaturen zu authentifizieren und deren Integrität zu sichern.

Wie Sie GnuPG benutzen, sollte von der Zielstrebigkeit und Findigkeit derer abhängen, die unerlaubt Ihre verschlüsselten Nachrichten mitlesen wollen. Ein solcher “Lauscher” kann ein neugieriger Systemadministrator sein, der Ihre E-Mails mitliest, es könnte ein Industriespion sein, der versucht, Ihre Firmengeheimnisse auszuspähen, oder es könnte die Staatsanwaltschaft sein, die Ihnen auf den Fersen ist. Wenn Sie GnuPG benutzen, um mehr oder weniger zufälliges Mitlesen zu verhindern, wird das wahrscheinlich anders aussehen, als wenn Sie Ihre Daten gegen einen entschlossenen Angreifer schützen wollen.

> **Tipp:** Ihr Ziel sollte es dabei sein, daß der Aufwand zur Entschlüsselung Ihrer Daten so groß wird, daß der Wert der Daten diesen Aufwand nicht mehr rechtfertigt.

Wenn Sie GnuPG auf Ihren persönlichen Gebrauch abstimmen möchten, sind vor allem vier Punkte wichtig:

1. Die Wahl der Schlüssellänge Ihres öffentlichen und privaten Schlüsselpaars
2. Der Schutz Ihres geheimen Schlüssels
3. Die Verfallsdaten Ihrer Schlüssel und die Benutzung von Unterschlüsseln
4. Der Aufbau Ihres *Web of Trust*

Eine gut gewählte Schlüssellänge schützt Sie gegen Brute-Force-Angriffe auf verschlüsselte Daten. Der Schutz Ihres privaten Schlüssels hindert einen Angreifer daran, einfach Ihren privaten Schlüssel zum Entschlüsseln von verschlüsselten Nachrichten zu verwenden und Nachrichten in Ihrem Namen zu unterschreiben. Ein sorgfältig aufgebautes *Web of Trust* verhindert, daß ein Unbefugter sich als einer Ihrer Korrespondenzpartner ausgeben kann.

Wichtig ist die Frage, welchen Aufwand Sie entsprechend Ihren Sicherheitsanforderungen betreiben möchten, um Ihre Privatsphäre oder Ihre Firmendaten zu schützen.

Die Wahl der Schlüssellänge

Die Wahl der Schlüssellänge hängt von der Art des jeweiligen Schlüssels ab. Bei OpenPGP besteht ein Schlüsselbund gewöhnlich aus mehreren öffentlichen und geheimen Schlüsseln. Es sollte zumindest einen Hauptschlüssel zum Signieren und einen oder eventuell mehrere zusätzliche Unterschlüssel für die Verschlüsselung geben. Wenn man die Standardeinstellungen von GnuPG bei der Schlüsselerzeugung verwendet, ist der Hauptschlüssel ein DSA-Schlüssel, die Unterschlüssel sind ElGamal-Schlüssel.

* *wenn 1024 Bit nicht besonders gut sind, sollte man da keinen längeren Schlüssel nehmen??? (pn)*

* *Das sollte beim nächsten Treffen des "GnuPG-Kraenzchens" ein Thema sein! Ich habe die nächsten zwei Absätze umformuliert, bin aber nicht sicher, das Richtige getroffen zu haben. (rg)*

DSA erlaubt eine Schlüssellänge bis zu 1024 Bit. Das ist angesichts der heutigen Rechenleistungen nicht besonders lang, entspricht jedoch dem Standard. Warum das so ist und warum ein DSA-Schlüssel mit 1024 Bit zur Benutzung sogar empfohlen wird, geht aus dem folgenden Absatz hervor.

ElGamal-Schlüssel andererseits können beliebig lang sein. GnuPG ist ein hybrides Verschlüsselungsverfahren mit öffentlichem Schlüssel. Der öffentliche Schlüssel wird zum Verschlüsseln eines 128-Bit-Sitzungsschlüssels benutzt, und der private Schlüssel wird zu dessen Entschlüsselung verwendet. Allerdings beeinflußt die Schlüssellänge die Ver- und Entschlüsselungsgeschwindigkeit erheblich, da der Rechenaufwand bei diesen Algorithmen exponentiell mit der Länge des Schlüssels steigt. Außerdem ist der praktische Nutzen eines großen Schlüssels trotz seiner größeren Sicherheit durchaus zweifelhaft. Wenn der Schlüssel lang genug ist, um einem Brute-Force-Angriff zu widerstehen, wird der Angreifer wahrscheinlich zu einer anderen Methode greifen, um an Ihre unverschlüsselten Daten zu gelangen. Es könnte ihm leichter fallen, in Ihre Wohnung oder Ihr Büro einzudringen oder Sie möglicherweise sogar zu überfallen. 1024 Bit sind alles in allem eine zu empfehlende Schlüssellänge. Wenn Sie wirklich einen längeren Schlüssel brauchen, dann sollten Sie ohnehin einen Fachmann in Sachen Datensicherheit konsultieren.

Der Schutz Ihres geheimen Schlüssels

Das Allerwichtigste bei der Benutzung von GnuPG ist der Schutz Ihres geheimen Schlüssels. Wenn jemand Ihren geheimen Schlüssel in die Hand bekommt, dann kann er damit alle für diesen Schlüssel verschlüsselten Daten entschlüsseln, und er kann digitale Unterschriften in Ihrem Namen leisten. Wenn Sie Ihren geheimen Schlüssel verlieren, sind Sie nicht länger imstande, Daten zu entschlüsseln, die für Sie verschlüsselt worden sind, und Sie können keine Unterschriften mehr leisten. Den geheimen Schlüssels zu verlieren, ist eine Katastrophe für Ihre Datensicherheit.

Egal, wie Sie GnuPG benutzen, Sie sollten die Widerrufurkunde des öffentlichen Schlüssels und eine Sicherheitskopie Ihres geheimen Schlüssels auf einem schreibgeschützten Datenträger - beispielsweise einer CD-ROM oder Diskette - speichern und an einem sicheren Ort aufbewahren, z. B. in einem Bankschließfach oder gut versteckt in Ihrer Wohnung. Um eventuellen Datenträgerdefekten vorzubeugen, sollten Sie vielleicht auch jeweils einen ASCII-Ausdruck (*** gpg --armor) auf Papier aufbewahren. Was immer Sie tun, die Widerrufurkunde und die Sicherheitskopie Ihres geheimen Schlüssels sollten auf Datenträger gebracht werden, die eine sichere Aufbewahrung so lange ermöglichen, wie Sie Ihren Schlüssel voraussichtlich behalten werden, und Sie sollten diese sorgfältiger aufbewahren als die Kopie Ihres täglich benutzten geheimen Schlüssels.

Als weitere Sicherheitsmaßnahme speichert GnuPG Ihren privaten Schlüssel nicht in "roher" Form ab, sondern verschlüsselt ihn stattdessen unter Benutzung eines symmetrischen Verschlüsselungsverfahrens. Deshalb brauchen Sie das "Mantra", um mit Ihrem geheimen Schlüssel zu entschlüsseln oder zu signieren. Somit müßte ein Angreifer gleich zwei Probleme lösen, um Zugang zu Ihrem geheimen Schlüssel zu bekommen:

1. Er müßte tatsächlich den Schlüssel in die Hand bekommen.
2. Er müßte entweder dessen Verschlüsselung knacken oder an das Mantra kommen.

Die sichere Aufbewahrung Ihres geheimen Schlüssels ist wichtig, doch auch mit einigem Aufwand verbunden. Im Idealfall würden Sie den geheimen Schlüssel auf einem mobilen, schreibgeschützten Datenträger, wie z. B. einer Diskette, speichern und ihn auf einem nicht vernetzten Computer benutzen, zu dem nur Sie Zugang haben. Vielleicht ist das für Sie zu unbequem oder unmöglich. Vielleicht besitzen Sie auch keinen eigenen Computer und haben nur am Arbeitsplatz oder in der Schule Zugang zu einem Computer.

Das heißt aber nicht, daß Sie nun GnuPG nicht benutzen können oder sollten. Sie haben sich nur entschieden, daß Ihnen Ihre Daten zwar wichtig genug sind, um sie zu verschlüsseln, aber nicht so wichtig, daß Sie besondere Maßnahmen treffen müßten, um die erste Barriere sicherer zu machen. Es ist letztlich Ihre Entscheidung, ob Ihr Sicherheitsanspruch damit schon erfüllt ist oder nicht.

Absolut unerläßlich ist ein gutes Mantra, wenn Sie GnuPG benutzen. Jeder Angreifer, der Zugang zu Ihrem geheimen Schlüssel bekommt, muß dann noch die Verschlüsselung Ihres geheimen Schlüssels knacken. Es ist so gut wie sicher, daß ein Angreifer versuchen wird, das Mantra zu erraten, anstatt mit einem Brute-Force-Angriff den Schlüssel selbst herauszufinden.

Es ist nicht gerade leicht, sich eine ausreichend große Zahl von unzusammenhängenden Zeichen zu merken. Deshalb ist die Versuchung sehr groß, ein Mantra zu wählen, das leichter zu erraten ist als ein nach dem Zufallsprinzip erstellter 128-Bit-Schlüssel, und die meisten Leute erliegen dieser Versuchung, sodaß es für einen Lauscher besonders verlockend ist, zu versuchen, das Mantra zu erraten. Aber wenn Sie sich wirklich im klaren darüber sind, daß Sie eine Verschlüsselung schließlich deshalb benutzen, weil Sie *verhindern* möchten, daß man Ihre Daten mitlesen kann, dann werden Sie dieser Versuchung nicht erliegen und die notwendige Mühe auf sich nehmen.

Wenn das Mantra aus einem normalen Wort besteht, dann ist es ein leichtes, alle Wörter in den Wörterbüchern sämtlicher Sprachen der Welt auszuprobieren. Selbst wenn die Reihenfolge der Buchstaben oder Zeichen innerhalb des Wortes verändert worden ist, ist es immer noch leicht, Wörter aus dem Wörterbuch mit einem Katalog von Permutationen auszuprobieren. Dasselbe Problem stellt sich bei Zitaten. Im Allgemeinen sind Mantras, die auf Äußerungen der natürlichen Sprache beruhen, schlechte Mantras, da ihre Zufälligkeit gering ist und da es in der natürlichen Sprache eine Menge Redundanz gibt. Sie sollten Mantras aus der natürlichen Sprache tunlichst vermeiden.

Ein gutes Mantra ist eines, das nur sehr schwer zu erraten ist, obwohl *Sie* es sich gut merken können. Es sollte Zeichen aus der ganzen Reihe der druckbaren Zeichen auf Ihrer gesamten Tastatur enthalten. Dazu gehören auch Großbuchstaben, Ziffern und Sonderzeichen wie beispielsweise `}`, `#` oder `^`.

> **Tipp:** Seien Sie kreativ und nehmen Sie sich ein wenig Zeit bei der Wahl Ihres Mantras! Eine gutes Mantra ist wichtig für die Sicherheit Ihrer Daten und somit auch Ihrer Privatsphäre oder Firmengeheimnisse!

Auswählen der Verfallsdaten und Benutzung von Unterschlüsseln

Wenn Sie ein neues Schlüsselpaar erzeugen, werden standardmäßig ein DSA-Hauptschlüssel zum Unterschreiben und ein ElGamal-Unterschlüssel zum Entschlüsseln erzeugt. Dies ist von Vorteil, weil die Aufgaben der beiden Schlüssel verschieden sind und es sinnvoll sein könnte, den beiden Schlüsseln verschiedene Verfallsdaten zu geben. Der DSA-Hauptschlüssel wird benutzt, um digitale Unterschriften zu leisten, und er bestätigt Ihre Identität dadurch, daß andere ihn signiert haben. Der ElGamal-Unterschlüssel wird nur benutzt, um an Sie geschickte verschlüsselte Daten zu entschlüsseln. Typischerweise sollte eine digitale Signatur eine lange oder unbegrenzte Gültigkeitsdauer haben; Sie wollen ja auch die Unterschriften auf Ihrem Schlüssel, die Sie mühsam zusammengetragen haben, nicht verlieren. Andererseits sollte der ElGamal-Unterschlüssel in gewissen Zeitabständen gewechselt werden, um Ihre Datensicherheit zu erhöhen, da ein Angreifer, wenn der ElGamal-Unterschlüssel geknackt ist, alle Dokumente lesen kann, die für diesen Schlüssel verschlüsselt worden sind oder es noch werden.

In der Regel sollten Sie also eine unbeschränkte Gültigkeitsdauer für den DSA-Hauptschlüssel wählen. Es gibt jedoch Gründe, weshalb Sie vielleicht doch ein Verfallsdatum für Ihren Hauptschlüssel wählen sollten. Erstens kann es sein, daß Sie dem Schlüssel nur eine beschränkte Geltungsdauer geben wollen, z. B., wenn Sie den Schlüssel für ein zeitlich befristetes Ereignis wie etwa eine politische Kampagne benutzen wollen und danach nicht mehr. Ein weiterer Grund könnte in einer zusätzlichen Vorsichtsmaßnahme bestehen: Falls der Hauptschlüssel kompromittiert wird (und Sie möglicherweise auch keine Widerrufurkunde haben), würde ein Verfallsdatum den Schlüssel genau an diesem Datum unbrauchbar werden lassen.

> **Tipp:** Einer solchen Kompromittierung sollten Sie jedoch möglichst durch Sicherheitsvorkehrungen vorbeugen, wie in 4.1.2 beschrieben.

Das Erneuern von ElGamal-Unterschlüsseln ist zwar kein Problem, kann aber unbequem werden. Kurz vor dem Verfallsdatums sollten Sie einen neuen ElGamal-Unterschlüssel erzeugen und die davon abgeleiteten öffentlichen Schlüssel bekannt geben. Diejenigen, die mit Ihnen korrespondieren wollen, müssen ja, sobald der alte Schlüssel seine Gültigkeit verliert, Ihren aktualisierten öffentlichen Schlüssel bekommen, da sie mit dem dann ungültigen Schlüssel nicht mehr verschlüsseln können. Je nachdem, wie Sie die Verteilung Ihrer öffentlichen Schlüssel organisieren, kann dies eine mühsame Angelegenheit werden. Sie müssen aber Gott sei Dank keine neuen Unterschriften einholen, um Ihren neuen

Unterschlüssel zu authentisieren. Eine Unterschrift mit Ihrem authentifizierten DSA-Hauptschlüssel bestätigt die Echtheit des neuen Schlüssels.

Die erzielte zusätzliche Sicherheit mag diese Unbequemlichkeit wert sein oder nicht. Genauso wie Sie, kann ein erfolgreicher Angreifer immer noch alle Dokumente lesen, die mit einem verfallenen Unterschlüssel verschlüsselt worden sind. Das Wechseln der Unterschlüssel schützt nur Dokumente, die Sie nach diesem Wechsel verschlüsseln. Um die mit dem neuen Unterschlüssel verschlüsselten Dokumente zu lesen, müßte der Angreifer erneut in den Besitz Ihres Schlüssels *und* ihres Mantras kommen.

Es ist auch nicht nötig, mehr als einen gültigen Unterschlüssel in einem Schlüsselbund zu haben. Man erzielt keine zusätzliche Sicherheit dadurch, daß man zwei oder mehr aktive Unterschlüssel hat. Es können natürlich mehrere verfallene Schlüssel in einem Schlüsselbund sein, so daß in der Vergangenheit verschlüsselte Dokumente noch entschlüsselt werden können, doch braucht nie mehr als ein Unterschlüssel aktiv zu sein.

Verwaltung Ihres *Web of Trust*

Genauso wie beim Schutz Ihres geheimen Schlüssels müssen Sie auch bei der Verwaltung Ihres *Web of Trust* zwischen Bequemlichkeit und Sicherheit abwägen. Wenn Sie GnuPG lediglich zum Schutz gegen mehr oder weniger zufälliges Mitlesen und Dokumentenfälschungen benutzen, dann können Sie relativ vertrauensvoll hinsichtlich der digitalen Signaturen anderer Leute sein. Wenn Sie sich allerdings Sorgen machen, daß ein zu allem entschlossener Angreifer an Ihren Firmendaten oder am Eindringen in Ihre Privatsphäre interessiert ist, dann sollten Sie die Unterschriften anderer sorgfältig prüfen.

Ungeachtet Ihrer eigenen Sicherheitsbedürfnisse sollten Sie jedoch beim Unterschreiben anderer Schlüssel *immer Sorgfalt walten lassen*. Im Sinne des *Web of Trust* ist es nicht ratsam, einen Schlüssel zu unterschreiben, dessen Authentizität Sie gerade noch so weit vertrauen, wie es für Ihr eigenes Sicherheitsbedürfnis ausreichend ist. Andere, die einen höheren Sicherheitsbedarf haben, sollten sich auf Ihre Unterschrift verlassen können. Wenn man sich auf Ihre Signatur nicht verlassen kann, dann schwächt dies das *Web of Trust* und macht die Kommunikation für alle Benutzer von GnuPG schwieriger.

> **Tipp:** Lassen Sie also beim Unterschreiben von Schlüsseln dieselbe Sorgfalt walten, die Sie von anderen auch angewandt sehen möchten, wenn Sie sich auf deren Unterschriften verlassen.

Bei der Verwaltung Ihres *Web of Trust* sollten Sie sich auf zwei Dinge konzentrieren: Einerseits auf die Frage, wessen Schlüssel Sie genügend vertrauen, um sie selber zu signieren, und andererseits auf das Abstimmen der Optionen `--marginals-needed` und `--completes-needed`. Jeder Schlüssel, den Sie persönlich signieren, wird als gültig betrachtet, deshalb ist es - außer in kleinen Gruppen - keine gute Praxis, persönlich den Schlüssel jeder Person zu unterschreiben, mit der Sie kommunizieren. Sinnvoller ist es, sich daran zu gewöhnen, den Unterschriften anderer zu vertrauen.

Es ist wahrscheinlich die beste Strategie, beim Unterzeichnen von Schlüsseln genau die Authentizität des Schlüssels bzw. die Identität des Schlüsselbesitzers zu überprüfen und ansonsten durch Optionen zu bestimmen, wie sorgfältig GnuPG bei der Authentisierung sein soll. Ein konkretes Beispiel: Sie mögen einigen wenigen engen Freunden voll vertrauen, von denen Sie wissen, daß diese beim Unterschreiben von Schlüsseln sorgfältig vorgehen; den weiteren Schlüsselbesitzern in Ihrem Schlüsselbund vertrauen Sie in dieser Hinsicht nur teilweise. Danach können Sie `--completes-needed` auf `1` und

`--marginals-needed` auf `2` setzen. Wenn Sie hinsichtlich der Sicherheit stärker besorgt sind, können Sie auch die Werte `1` bzw. `3` oder `2` bzw. `3` wählen. Wenn Sie allerdings mit einem weniger großen Vertrauen hinsichtlich der Authentizität auskommen wollen und nicht so sehr mögliche Angriffe auf Ihre Privatsphäre oder Firmendaten befürchten, dann können Sie die Werte `1` und `1` einsetzen. Je höher die Werte für diese Optionen sind, desto schwieriger ist es, Ihnen einen gefälschten Schlüssel unterzuschieben.

Aufbau Ihres *Web of Trust*

Es reicht nicht aus, wenn nur Sie selbst GnuPG benutzen wollen. Um GnuPG zur sicheren Kommunikation mit anderen zu nutzen, müssen Sie ein funktionierendes *Web of Trust* aufbauen. Auf den ersten Blick scheint dies eine mühsame Aufgabe zu sein: Die Leute, mit denen Sie kommunizieren, müssen GnuPG [1] *ebenfalls* benutzen, und die Schlüssel müssen von ausreichend vielen Personen unterschrieben sein, so daß sie als authentisch zu betrachten sind. Dies sind wohlgemerkt keine technischen Schwierigkeiten, sondern soziale. Nichtsdestoweniger müssen Sie diese Schwierigkeiten meistern, wenn Sie GnuPG benutzen wollen.

Anfangs ist es noch nicht so wichtig, daß Sie mit allen Korrespondenzpartnern sicher kommunizieren können. Wenn Sie mit dem Gebrauch von GnuPG beginnen, suchen Sie sich einen kleinen Kreis von Leuten - Sie selbst und noch ein oder zwei andere -, die ebenfalls ihr Recht auf eine geschützte Privatsphäre in Anspruch nehmen wollen. Unterschreiben Sie jeweils, wenn Sie sich von der Identität der anderen Person überzeugt haben, deren öffentlichen Schlüssel und lassen Sie sich im Gegenzug Ihren Schlüssel signieren. Dieses kleine, robuste *Web of Trust* ist Ihr Ausgangspunkt. Sie werden dessen Wert zu schätzen lernen und - wenn Sie Ihr *Web of Trust* in der Zukunft weiter ausbauen - um so gewissenhafter und vorsichtiger sein.

Über Ihr anfängliches *Web of Trust* hinaus möchten Sie wahrscheinlich auch mit anderen Personen sicher kommunizieren; hierbei können zwei Schwierigkeiten auftreten:

1. Sie wissen nicht immer, ob Ihr Gegenüber GnuPG benutzt oder überhaupt benutzen will.
2. Selbst wenn Sie wissen, daß der andere GnuPG verwendet, könnten Sie Schwierigkeiten bei der Authentifizierung seines öffentlichen Schlüssels haben.

Das erste Problem rührt daher, daß viele Leute nicht öffentlich bekanntgeben, daß sie GnuPG benutzen. Am besten, Sie gehen mit gutem Beispiel voran und sorgen dafür, daß jeder Ihrer potentiellen Kommunikationspartner weiß, daß Sie GnuPG benutzen. Hierfür gibt es mehrere Möglichkeiten:

- Signieren Sie Nachrichten, die Sie an Ihre Korrespondenzpartner oder Mailinglisten verschicken, mit GnuPG.
- Veröffentlichen Sie Ihren öffentlichen Schlüssel auf Ihrer Website.
- Geben Sie Ihren öffentlichen Schlüssel auf einen Key-Server und veröffentlichen Sie die Schlüssel-ID in Ihrer E-Mail-Signatur oder auf Ihrer Visitenkarte.

Indem Sie Ihren Schlüssel bekannt geben, machen Sie es auch für andere akzeptabler, ihrerseits ihre Schlüssel bekannt zu geben. Außerdem erleichtern Sie es dadurch anderen, sicher mit Ihnen zu kommunizieren, da Sie die Initiative ergriffen und deutlich gezeigt haben, daß Sie GnuPG benutzen.

Die Authentisierung der öffentlichen Schlüssel ist schwieriger. Wenn Sie sich nicht persönlich von der Identität einer Person überzeugt haben, dann dürfen Sie deren Schlüssel auch *nicht* unterschreiben. In diesem Fall müssen Sie auf die Unterschriften von anderen vertrauen und hoffen, eine Kette von Unterschriften zu finden, die von dem betreffenden Schlüssel zurück zu Ihrem eigenen führt. Solch eine Kette kann nur zustande kommen, wenn Sie Ihren Schlüssel von anderen außerhalb Ihres anfänglichen *Web of Trust* haben unterschreiben lassen. Am einfachsten ist dies auf sogenannten Key-Signing-Partys (http://www.herrons.com/kb2nsx/keysign.html) zu erreichen: Das sind Zusammenkünfte, bei denen man sich gegenseitig authentifiziert und die öffentlichen Schlüssel unterzeichnet. Sollten Sie beispielsweise zu einer Konferenz gehen, halten Sie Ausschau nach einer Key-Signing-Party. Falls dort keine stattfindet, dann laden Sie doch einfach selbst zu einer ein. Auf jeden Fall sollten Sie aber Ihren GnuPG-Fingerabdruck immer bei sich haben (vielleicht auf Ihrer Visitenkarte) so daß Sie spontan mit anderen die Schlüssel tauschen können. Derjenige, dem Sie den Fingerabdruck gegeben haben, könnte dann, nachdem er Ihre Identität überprüft hat, bei der nächsten Gelegenheit Ihren öffentlichen Schlüssel unterschreiben.

Welchen Aufwand Sie betreiben, ist letztendlich Ihre Entscheidung und hängt allein von Ihren Sicherheitsbedürfnissen ab. Niemand ist verpflichtet, seinen Schlüssel öffentlich zu machen oder die Schlüssel anderer zu unterschreiben. Eine der Stärken von GnuPG ist die Flexibilität, mit der man die Benutzung den eigenen Ansprüchen anpassen kann. Sie werden jedoch feststellen, daß Sie Ihr *Web of Trust* ausbauen müssen, wenn sie GnuPG für Ihre sichere Kommunikation einsetzen möchten.

Fußnoten

1. In diesem Abschnitt bezieht sich GnuPG sowohl auf die GnuPG-Implementierung von OpenPGP als auch auf andere Implementierungen wie das PGP-Produkt von NAI.

Kapitel 5. Kryptogesetzgebung

Die gesetzlichen und politischen Rahmenbedingungen zur Benutzung von Verschlüsselungs-Software sind von Land zu Land sehr unterschiedlich und stetigem Wandel unterworfen. Deshalb möchten wir hier nur kurz die rechtliche Situation in der Bundesrepublik Deutschland anreißen. Die Internetseite vonBert-Jaap Koops (http://cwis.kub.nl/~frw/people/koops/bertjaap.htm) bietet eine ausgezeichnete "Übersicht über die Kryptographie-Gesetzgebung" (http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm), die Sie hinsichtlich rechtlicher Fragen in den verschiedenen Staaten zu Rate ziehen sollten.

Für eine Betrachtung der Kryptogesetzgebung sind vor allem folgende Punkte von Interesse:

- *Beschränkungen der Benutzung kryptographischer Verfahren*,
- *Beschränkungen hinsichtlich der Ausfuhr von kryptographischen Produkten* und
- die *Gültigkeit von digitalen Signaturen*.

Benutzungsbeschränkungen

Das Grundgesetz der Bundesrepublik Deutschland garantiert in Artikel 10 Absatz 1 die Unverletzlichkeit des Post- und Fermeldegeheimnisses. Darunter fällt auch das Verbergen des Nachrichteninhalts durch kryptographische Verfahren. Einschränkungen dieses Grundrechtes sind prinzipiell auf Grund eines Gesetzes möglich (Art. 10, Abs. 2 GG). Im Gegensatz zu vielen anderen Staaten gibt es jedoch derzeit in Deutschland keine rechtlichen Beschränkungen hinsichtlich des Einsatzes von Verschlüsselungsverfahren.

Nach den vom Bundeskabinett am 2. Juni 1999 verabschiedeten "Eckpunkten der deutschen Kryptopolitik" (http://www.sicherheit-im-Internet.de/download/krypto-d.pdf) spricht sich die Bundesregierung sogar deutlich für den "Einsatz sicherer kryptographischer Verfahren" zum "verbesserte[n] Schutz deutscher Nutzer in den weltweiten Informationsnetzen" aus und will deshalb "die Verbreitung sicherer Verschlüsselung in Deutschland aktiv unterstützen". In diesem Zusammenhang ist auch die Förderung (http://www.sicherheit-im-Internet.de/showdoc.php3?doc=bmwi_min_doc_1999942923793%38page=1) des GnuPG-Projektes durch das Bundesministeriums für Wirtschaft und Technologie (http://www.bmwi.de/) zu sehen.

Ausfuhrbeschränkungen

Das sogenannte "Wassenaar Abkommen" (http://www.wassenaar.org)stuft starke Kryptographie als Kriegswaffe ein und sieht vor, daß seine 33 Mitgliedsstaaten (zu denen auch die Bundesrepublik Deutschland gehört) die Ausfuhr von kryptographischen Produkten mit einer Schlüssellänge von mehr als 64 Bit kontrollieren. [1] Der Export kryptographischer und kryptanalytischer Technologien unterliegt zwar prinzipiell nach §§ 7 Abs. 1, 5 Abs. 1 AWG einem Genehmigungsvorbehalt, aber kryptographische Produkte, die frei auf dem Massenmarkt erhältlich sind, können gegenwärtig ohne Genehmigung aus der Bundesrepublik ausgeführt werden.

Digitale Signaturen

Mit der zunehmenden Bedeutung von Online-Banking, E-Commerce und Austausch von (amtlichen) Dokumenten über das Internet, hat auch der Gesetzgeber, hinsichtlich einer juristischen Bewertung der Gültigkeit und Anerkennung digitaler Signaturen, Handlungsbedarf erkannt. Das "Gesetz zur digitalen Signatur (Signaturgesetz, SigG)" vom 22. Juli 1997 legt die "Rahmenbedingungen für digitale Signaturen" fest "unter denen diese als sicher gelten und Fälschungen digitaler Signaturen oder Verfälschungen von signierten Daten zuverlässig festgestellt werden können", stellt diese allerdings nicht der gesetzlichen Schriftform gleich. Zweck des Gesetzes ist vielmehr "durch tatsächliche Sicherheit Vertrauen in die gesetzliche digitale Signatur" zu schaffen, "so daß sie vom Rechtsverkehr akzeptiert wird und Gerichte ihr im Rahmen der freien Beweiswürdigung die nötige Beweiskraft zuerkennen können". Eine Novellierung des Signaturengesetzes steht allerdings bevor.

Fußnoten

1. In den USA beispielsweise unterliegen kryptographische Produkte strengen Ausfuhrbestimmungen, die sich erst allmählich - unter wirtschaftlichem und wissenschaftlichen Druck - zu lockern scheinen.

Anhang A. Ressourcen im Internet

Es gibt im Internet zahlreiche Informationsquellen zu den Themen ... Weitere Informationsquellen zu den Themen GnuPG, Kryptographie, Datensicherheit, Datenschutz, Informationssicherheit finden Sie auf Webseiten und Mailinglisten im Internet.

GnuPG

Die aktuellen Informationen und Versionen von GnuPG finden Sie auf der GnuPG-Homepage http://www.gnupg.org.

Die neueste Version von GnuPG

Die aktuelle Version von GnuPG können Sie unter http://www.gnupg.org/de/download.html sowohl als .tar.gz, als .rpm oder als Binärversion für Windows herunterladen. Außerdem finden Sie dort die Signaturen und MD5 Prüfsummen, um die Integrität der Dateien zu überprüfen, Links zu Mirror-Sites und Links zu weiteren Programmen, die GnuPG unterstützen.

Die neueste Version des Handbuchs

Unter http://www.gnupg.org/de/docs.htmlfinden Sie die aktuelle Version dieses Handbuches. Außerdem können hier weitere hilfreiche und interessante Dokumente heruntergeladen werden.

GnuPG Mailinglisten

Es gibt mehrere Mailinglisten zum Thema GnuPG:

- announce@gnupg.org wird für die Ankündigung neuer Versionen und andere wichtige Nachrichten verwendet. Sie können diese Mailingliste abonnieren, indem Sie eine E-Mail mit dem Betreff (Subject) "subscribe" an announce-request@gnupg.org senden.
- gnupg-users@gnupg.org ist das allgemeine Diskussions- und Hilfsforum. Senden Sie eine E-Mail mit dem Betreff (Subject) "subscribe" an gnupg-users-request@gnupg.org.

Weitere Mailinglisten, die vor allem für Entwickler interessant sind finden Sie unter http://www.gnupg.org/de/docs.html. Ein Archiv dieser Mailinglisten ist online unter http://lists.gnupg.org verfügbar.

Links

Links zu anderen interessanten Webseiten und Projekten finden Sie unter http://www.gnupg.org/de/others.html und unter http://www.gnupg.org/de/crypto.html.

ftp.gnupg.org

Alternativ können Sie GnuPG als Tar-Archiv auch per Ftp von ftp://ftp.gnupg.org/pub/gcrypt/ herunterladen.

- ftp://ftp.gnupg.org/pub/gcrypt/gnupg/

  (Hier finden Sie die aktuelle stabile Version.)
- ftp://ftp.gnupg.org/pub/gcrypt/devel/

  (Hier gibt es die aktuellen Entwicklerversionen.)
- ftp://ftp.gnupg.org/pub/gcrypt/binary/

(Hier befinden sich vorkompilierte Binärversionen für Win32 und OS/2.)

Kryptographie allgemein

Hier finden Sie allgemeine und aktuelle Informationen zum Thema Kryptographie.

Crypto-gram

Der monatliche Newsletter des Autors und Kryptographieexperten Bruce Schneier. Um Crypto-gram zu abonnieren, senden Sie eine E-Mail an crypto-gram-subscribe@chaparraltree.com.

Unter http://www.counterpane.com/crypto-gram.html finden Sie ein Archiv des Newsletters.

telepolis

Das Onlinemagazin Telepolis (http://www.heise.de/tp) bietet News, Features und einen Themenschwerpunkt (http://www.heise.de/tp/deutsch/special/krypto/default.html) zu Kryptographie, Kommunikationssicherheit und Datenschutz.

Kryptographie-Gesetzgebung

Crypto Law Survey

Die Internet-Seite (http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm) von Bert-Jaap Koops (http://cwis.kub.nl/~frw/people/koops/bertjaap.htm) bietet einen hervorragenden internationalen Überblick über gesetzliche Regelungen und Ausfuhrbestimmungen für kryptographische Produkte.

Sicherheit im Internet

“Sicherheit im Internet - Sicherheit in der Informationsgesellschaft” ist eine Initiative des Bundesministeriums für Wirtschaft und Technologie, des Bundesministeriums des Innern und des Bundesamtes für Sicherheit in der Informationstechnik. Hier gibt es außer aktuellen Informationen und Pressemitteilungen aus den Ministerien Hintergrundartikel, einen Webkatalog mit weiteren Links und einen Newsletter.

Link-Sammlungen

Peter Gutmanns “crypto link farm”

Diese beinahe unerschöpfliche Linksammlung bietet einen ausgezeichneten Ausgangspunkt für die Suche nach weiteren Informationsquellen. Zu finden unter http://www.cs.auckland.ac.nz/~pgut001/links.html oder über den deutschen Mirror http://www.han.de/~ott/security-links.html.

Keyserver

Deutsche Keyserver:

- http://www.pca.dfn.de/dfnpca/pgpkserv/
- http://germany.keyserver.net/en/

Anhang B. Installation von GnuPG

Dieses Kapitel beschreibt die Installation von GnuPG auf verschiedenen Plattformen. Die Beschreibung bezieht sich auf die Version 1.0.1 von GnuPG. Bitte lesen Sie auf jeden Fall auch die Dateien "README" und "INSTALL" im GnuPG Source-Verzeichnis. Wo Sie aktuelle Versionen von GnuPG als TAR-Archiv, RPM oder Binary für Win32 bekommen, können Sie im Anhang B nachlesen.

Unix und GNU/Linux

Bevor Sie mit der Installation beginnen, vergewissern Sie sich bitte, daß Ihre Quelldateien nicht modifiziert wurden. Das ist ein sehr wichtiger Schritt, denn nur so können Sie sicher sein, daß niemand irgendwelche Hintertüren oder beabsichtigte Schwachstellen in den Code eingebaut hat.

Wenn Sie bereits eine Version von GnuPG installiert haben, können Sie einfach die Signatur überprüfen. *Benutzen Sie jedoch niemals die Version, die Sie gerade installieren möchten, für diese Überprüfung*. Der öffentliche Schlüssel zur Prüfung der Signatur ist:

```
pub  1024D/57548DCD 1998-07-07 Werner Koch (gnupg sig) <dd9jn@gnu.org>
```

Sollten Sie diesen Schlüssel noch nicht in Ihrem öffentlichen Schlüsselbund haben, müssen Sie ihn zuerst aus der Datei g10/pubring.asc aus den Sourcen importieren:

```
alice$  gpg --import src/gnupg-1.0.0/g10/pubring.asc
```

oder von einem Keyserver holen, also beispielsweise:

```
alice$  gpg --keyserver blackhole.pca.dfn.de --recv-keys 0x57548DCD
```

Dann können Sie die Signatur überprüfen mit

```
alice$  gpg --verify gnupg-1.0.1.tar.gz.asc
```

Sollten Sie eine vetrauenswürdige Version von PGP 2 oder PGP 5 installiert haben, können Sie die PGP 2 Signatur mit gnupg-1.0.1.tar.gz.sig überprüfen:

```
alice$  pgp gnupg-1.0.1.tar.gz.sig gnupg-1.0.1.tar.gz
```

Falls Sie weder GnuPG noch PGP installiert haben, dann benutzen Sie den MD5-Hashalgorithmus, um eine Prüfsumme des Tar-Files zu erzeugen.

```
alice$  md5sum gnupg-1.0.1.tar.gz
37eeae62c1823edc787996bfee70351a  gnupg-1.0.1.tar.gz
```

Vergleichen Sie dann bitte die Checksumme mit der, die Sie unter http://www.gnupg.org/download.html finden.

Falls Sie GnuPG systemweit installieren möchten, so daß es für alle User nutzbar ist, und falls sich bei Ihrem System die Sourcen für lokal installierte Software unter /usr/local/src/ befinden, kopieren Sie erst das TAR-File nach /usr/local/src, wechseln dann in dieses Verzeichnis und entpacken das TAR-File:

```
root#  cd /usr/local/src/
root#  gzip -d gnupg-1.0.1.tar.gz
root#  tar xvf gnupg-1.0.1.tar
```

Danach wechseln Sie in das neu angelegte Unterverzeichnis gnupg-1.0.1/ und führen dann nacheinander

```
root#  ./configure
root#  make
root#  make install
```

aus. Die ausführbare Datei "gpg" befindet sich dann in /usr/local/bin. Für weitere Optionen von **configure** benutzen Sie die Option `--help` und lesen die Datei INSTALL.

Um zu verhindern, daß vertrauliche Daten auf die Swap-Partition ausgelagert werden, empfiehlt es sich, das Set-User-ID Bit für "gpg" zu setzen:

```
root#  chmod u+s /usr/local/bin/gpg
```

GnuPG verhindert dann, daß Teile seines Speicherbereichs auf die Festplatte ausgelagert werden. Wenn Sie dies nicht tun möchten, können Sie die Option `no-secmem-warning` in die Datei ~/.gnupg/options einfügen, dann bekommen Sie diesbezüglich auch keine Warnmeldungen mehr.

Falls Sie keine Root-Rechte auf dem System haben oder der Systemadministrator nicht gewillt ist, GnuPG systemweit zu installieren, besteht immer noch die Möglichkeit zu einer User-Installation. Legen Sie dazu am besten ein Unterverzeichnis "src/" in Ihrem Home-Verzeichnis an, wenn Sie dies nicht schon haben. Entpacken Sie das Tar-File in "~/src" und führen dann:

```
alice$  ./configure --prefix=$HOME
```

aus. Dann können Sie genau wie oben ein **make** und **make install** durchführen. **make install** legt dann die Unterverzeichnisse "bin/" "lib/" "man/" "share/" in Ihrem Home-Verzeichnis an. Die ausführbare Datei befindet sich dann unter "$HOME/bin/gpg". $HOME/bin sollte natürlich in Ihrem Pfad liegen.

Anhang C. Referenz

gpg manpage

Name

`gpg` — encryption and signing tool

Synopsis

```
gpg
 [--homedir name]
 [--options file]
 [options]
 command
 [args]
```

DESCRIPTION

gpg is the main program for the GnuPG system.

COMMANDS

gpg recognizes these commands:

-s, --sign

Make a signature. This command may be combined with --encrypt.

--clearsign

Make a clear text signature.

-b, --detach-sign

Make a detached signature.

-e, --encrypt

Encrypt data. This option may be combined with --sign.

-c, --symmetric

Encrypt with symmetric cipher only This command asks for a passphrase.

--store

Store only (make a simple RFC1991 packet).

--decrypt [*file*]

Decrypt *file* (or stdin if no file is specified) and write it to stdout (or the file specified with --output). If the decrypted file is signed, the signature is also verified. This command differs from the default operation, as it never writes to the filename which is included in the file and it rejects files which don't begin with an encrypted message.

--verify [[*sigfile*] [*signed-files*]]

Assume that *sigfile* is a signature and verify it without generating any output. With no arguments, the signature packet is read from stdin (it may be a detached signature when not used in batch mode). If only a sigfile is given, it may be a complete signature or a detached signature, in which case the signed stuff is expected in a file without the ".sig" or ".asc" extension (if such a file does not exist it is expected at stdin; use a single dash ("-") as filename to force a read from stdin). With more than 1 argument, the first should be a detached signature and the remaining files are the signed stuff.

--verify-files [*files*]

This is a special version of the --verify command which does not work with detached signatures. The command expects the files to bee verified either on the commandline or reads the filenames from stdin; each anem muts be on separate line. The command is intended for quick checking of many files.

--list-keys [*names*]
--list-public-keys [*names*]

List all keys from the public keyrings, or just the ones given on the command line.

--list-secret-keys [*names*]

List all keys from the secret keyrings, or just the ones given on the command line.

--list-sigs [*names*]

Same as --list-keys, but the signatures are listed too.

--check-sigs [*names*]

Same as --list-sigs, but the signatures are verified.

--fingerprint [*names*]

List all keys with their fingerprints. This is the same output as --list-keys but with the additional output of a line with the fingerprint. May also be combined with --list-sigs or --check-sigs. If this command is given twice, the fingerprints of all secondary keys are listed too.

--list-packets

List only the sequence of packets. This is mainly useful for debugging.

--gen-key

Generate a new key pair. This command is normally only used interactive.

There is an experimental feature which allows to create keys in batch mode. See the file `doc/DETAILS` in the source distribution on how to use this.

--edit-key *name*

Present a menu which enables you to do all key related tasks:

sign

Make a signature on key of user *name* If the key is not yet signed by the default user (or the users given with -u), the program displays the information of the key again, together with its fingerprint and asks whether it should be signed. This question is repeated for all users specified with -u.

lsign

Same as --sign but the signature is marked as non-exportable and will therefore never be used by others. This may be used to make keys valid only in the local environment.

revsig

Revoke a signature. GnuPG asks for every signature which has been done by one of the secret keys, whether a revocation certificate should be generated.

trust

Change the owner trust value. This updates the trust-db immediately and no save is required.

disable
enable

Disable or enable an entire key. A disabled key can normally not be used for encryption.

adduid

Create an alternate user id.

deluid

Delete an user id.

addkey

Add a subkey to this key.

delkey

Remove a subkey.

revkey

Revoke a subkey.

expire

Change the key expiration time. If a key is selected, the time of this key will be changed. With no selection the key expiration of the primary key is changed.

passwd

Change the passphrase of the secret key.

uid *n*

Toggle selection of user id with index *n*. Use 0 to deselect all.

key *n*

Toggle selection of subkey with index *n*. Use 0 to deselect all.

check

Check all selected user ids.

pref

List preferences.

toggle

Toggle between public and secret key listing.

save

Save all changes to the key rings and quit.

quit

Quit the program without updating the key rings.

The listing shows you the key with its secondary keys and all user ids. Selected keys or user ids are indicated by an asterisk. The trust value is displayed with the primary key: the first is the assigned owner trust and the second is the calculated trust value. Letters are used for the values:

-

No ownertrust assigned / not yet calculated.

e

Trust calculation has failed.

q

Not enough information for calculation.

n

Never trust this key.

m

Marginally trusted.

f

Fully trusted.

u

Ultimately trusted.

--sign-key *name*

Sign a public key with you secret key. This is a shortcut version of the subcommand "sign" from --edit.

--lsign-key *name*

Sign a public key with you secret key but mark it as non-exportable. This is a shortcut version of the subcommand "lsign" from --edit.

--delete-key *name*

Remove key from the public keyring

--delete-secret-key *name*

Remove key from the secret and public keyring

--gen-revoke

Generate a revocation certificate for the complete key. To revoke a subkey or a signature, use the --edit command.

--export [*names*]

Either export all keys from all keyrings (default keyrings and those registered via option --keyring), or if at least one name is given, those of the given name. The new keyring is written to stdout or to the file given with option "output". Use together with --armor to mail those keys.

--send-keys [*names*]

Same as --export but sends the keys to a keyserver. Option --keyserver must be used to give the name of this keyserver. Don't send your complete keyring to a keyserver - select only those keys which are new or changed by you.

--export-all [*names*]

Same as --export, but does also export keys which are not compatible to OpenPGP.

--export-secret-keys [*names*]
--export-secret-subkeys [*names*]

Same as --export, but does export the secret keys. This is normally not very useful and a security risk. the second form of the command has the special property to render the secret part of the primary key useless; this is a GNU extension to OpenPGP and other implementations can not be expected to successful import such a key.

--import [*files*]
--fast-import [*files*]

Import/merge keys. This adds the given keys to the keyring. The fast version does not build the trustdb; this can be done at any time with the command --update-trustdb.

--recv-keys *key IDs*

Import the keys with the given key IDs from a HKP keyserver. Option --keyserver must be used to give the name of this keyserver.

--export-ownertrust

List the assigned ownertrust values in ASCII format for backup purposes

--import-ownertrust [*files*]

Update the trustdb with the ownertrust values stored in *files* (or stdin if not given); existing values will be overwritten.

--print-md *algo* [*files*]

Print message digest of algorithm ALGO for all given files of stdin. If "*" is used for the algorithm, digests for all available algorithms are printed.

--gen-random *0|1|2* [*count*]

Emit COUNT random bytes of the given quality level. If count is not given or zero, an endless sequence of random bytes will be emitted. PLEASE, don't use this command unless you know what you are doing, it may remove precious entropy from the system!

--gen-prime *mode bits* [*qbits*]

Use the source, Luke :-). The output format is still subject to change.

--version

Print version information along with a list of supported algorithms.

--warranty

Print warranty information.

-h, --help

Print usage information. This is a really long list even it does list not all options.

OPTIONS

Long options can be put in an options file (default "~/.gnupg/options"). Do not write the 2 dashes, but simply the name of the option and any required arguments. Lines with a hash as the first non-white-space character are ignored. Commands may be put in this file too, but that does not make sense.

gpg recognizes these options:

-a, --armor

Create ASCII armored output.

-o, --output `file`

Write output to `file`.

-u, --local-user `name`

Use `name` as the user ID to sign. This option is silently ignored for the list commands, so that it can be used in an options file.

--default-key `name`

Use `name` as default user ID for signatures. If this is not used the default user ID is the first user ID found in the secret keyring.

-r, --recipient `name`

Encrypt for user id `name`. If this option is not specified, GnuPG asks for the user-id unless --default-recipient is given

--default-recipient `name`

Use `name` as default recipient if option --recipient is not used and don't ask if this is a valid one. `name` must be a non empty.

--default-recipient-self

Use the default key as default recipient if option --recipient is not used and don't ask if this is a valid one. The default key is the first one from the secret keyring or the one set with --default-key.

--no-default-recipient

Reset --default-recipient and --default-recipient-self.

--encrypt-to `name`

Same as --recipient but this one is intended for in the options file and may be used together with an own user-id as an "encrypt-to-self". These keys are only used when there are other recipients given either by use of --recipient or by the asked user id. No trust checking is performed for these user ids and even disabled keys can be used.

--no-encrypt-to

Disable the use of all --encrypt-to keys.

-v, --verbose

Give more information during processing. If used twice, the input data is listed in detail.

-q, --quiet

Try to be as quiet as possible.

-z *n*

Set compression level to *n*. A value of 0 for *n* disables compression. Default is to use the default compression level of zlib (normally 6).

-t, --textmode

Use canonical text mode. If -t (but not --textmode) is used together with armoring and signing, this enables clearsigned messages. This kludge is needed for PGP compatibility; normally you would use --sign or --clearsign to selected the type of the signature.

-n, --dry-run

Don't make any changes (this is not completely implemented).

-i, --interactive

Prompt before overwriting any files.

--batch

Use batch mode. Never ask, do not allow interactive commands.

--no-batch

Disable batch mode. This may be of use if --batch is enabled from an options file.

--yes

Assume "yes" on most questions.

--no

Assume "no" on most questions.

--always-trust

Skip key validation and assume that used keys are always fully trusted. You won't use this unless you have installed some external validation scheme.

--keyserver *name*

Use *name* to lookup keys which are not yet in your keyring. This is only done while verifying messages with signatures. The option is also required for the command --send-keys to specify the keyserver to where the keys should be send. All keyservers synchronize with each other - so there is no need to send keys to more than one server. Using the command "host -l pgp.net | grep wwwkeys" gives you a list of keyservers. Because there is load balancing using round-robin DNS you may notice that you get different key servers.

--honor-http-proxy

Try to access the keyserver over the proxy set with the variable "http_proxy".

--keyring *file*

Add *file* to the list of keyrings. If *file* begins with a tilde and a slash, these are replaced by the HOME directory. If the filename does not contain a slash, it is assumed to be in the home-directory ("~/.gnupg" if --homedir is not used). The filename may be prefixed with a scheme:

"gnupg-ring:" is the default one.

"gnupg-gdbm:" may be used for a GDBM ring. Note that GDBM is experimental and likely to be removed in future versions.

It might make sense to use it together with --no-default-keyring.

--secret-keyring `file`

Same as --keyring but for the secret keyrings.

--homedir `directory`

Set the name of the home directory to `directory` If this option is not used it defaults to "~/.gnupg". It does not make sense to use this in a options file. This also overrides the environment variable "GNUPGHOME".

--charset `name`

Set the name of the native character set. This is used to convert some strings to proper UTF-8 encoding. Valid values for `name` are:

iso-8859-1

This is the default Latin 1 set.

iso-8859-2

The Latin 2 set.

koi8-r

The usual Russian set (rfc1489).

--utf8-strings
--no-utf8-strings

Assume that the arguments are already given as UTF8 strings. The default (--no-utf8-strings) is to assume that arguments are encoded in the character set as specified by --charset. These options effects all following arguments. Both options may used multiple times.

--options `file`

Read options from `file` and do not try to read them from the default options file in the homedir (see --homedir). This option is ignored if used in an options file.

--no-options

Shortcut for "--options /dev/null". This option is detected before an attempt to open an option file.

--load-extension `name`

Load an extension module. If `name` does not contain a slash it is searched in "/usr/local/lib/gnupg" See the manual for more information about extensions.

--debug *flags*

Set debugging flags. All flags are or-ed and *flags* may be given in C syntax (e.g. 0x0042).

--debug-all

Set all useful debugging flags.

--status-fd *n*

Write special status strings to the file descriptor *n*. See the file DETAILS in the documentation for a listing of them.

--logger-fd *n*

Write log output to file descriptor *n* and not to stderr.

--no-comment

Do not write comment packets. This option affects only the generation of secret keys. Output of option packets is disabled since version 0.4.2.

--comment *string*

Use *string* as comment string in clear text signatures.

--default-comment

Force to write the standard comment string in clear text signatures. Use this to overwrite a --comment from a config file.

--no-version

Omit the version string in clear text signatures.

--emit-version

Force to write the version string in clear text signatures. Use this to overwrite a previous --no-version from a config file.

-N, --notation-data *name=value*

Put the name value pair into the signature as notation data. *name* must consists only of alphanumeric characters, digits or the underscore; the first character must not be a digit. *value* may be any printable string; it will encoded in UTF8, so sou should have check that your --charset is set right. If you prefix *name* with an exclamation mark, the notation data will be flagged as critical (rfc2440:5.2.3.15).

--set-policy-url *string*

Use *string* as Policy URL for signatures (rfc2440:5.2.3.19). If you prefix it with an exclamation mark, the policy URL packet will be flagged as critical.

--set-filename *string*

Use *string* as the name of file which is stored in messages.

--use-embedded-filename

Try to create a file with a name as embedded in the data. This can be a dangerous option as it allows to overwrite files.

--completes-needed *n*

Number of completely trusted users to introduce a new key signer (defaults to 1).

--marginals-needed *n*

Number of marginally trusted users to introduce a new key signer (defaults to 3)

--max-cert-depth *n*

Maximum depth of a certification chain (default is 5).

--cipher-algo *name*

Use *name* as cipher algorithm. Running the program with the command --version yields a list of supported algorithms. If this is not used the cipher algorithm is selected from the preferences stored with the key.

--digest-algo *name*

Use *name* as message digest algorithm. Running the program with the command --version yields a list of supported algorithms. Please note that using this option may violate the OpenPGP requirement, that a 160 bit hash is to be used for DSA.

--s2k-cipher-algo *name*

Use *name* as the cipher algorithm used to protect secret keys. The default cipher is BLOWFISH. This cipher is also used for conventional encryption if --cipher-algo is not given.

--s2k-digest-algo *name*

Use *name* as the digest algorithm used to mangle the passphrases. The default algorithm is RIPE-MD-160. This digest algorithm is also used for conventional encryption if --digest-algo is not given.

--s2k-mode *n*

Selects how passphrases are mangled. If *n* is 0 a plain passphrase (which is not recommended) will be used, a 1 (default) adds a salt to the passphrase and a 3 iterates the whole process a couple of times. Unless --rfc1991 is used, this mode is also used for conventional encryption.

--compress-algo *n*

Use compress algorithm *n*. Default is 2 which is RFC1950 compression. You may use 1 to use the old zlib version which is used by PGP. The default algorithm may give better results because the window size is not limited to 8K. If this is not used the OpenPGP behavior is used, i.e. the compression algorithm is selected from the preferences; note, that this can't be done if you do not encrypt the data.

--disable-cipher-algo *name*

Never allow the use of *name* as cipher algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--disable-pubkey-algo *name*

Never allow the use of *name* as public key algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--throw-keyid

Do not put the keyid into encrypted packets. This option hides the receiver of the message and is a countermeasure against traffic analysis. It may slow down the decryption process because all available secret keys are tried.

--not-dash-escaped

This option changes the behavior of cleartext signatures so that they can be used for patch files. You should not send such an armored file via email because all spaces and line endings are hashed too. You can not use this option for data which has 5 dashes at the beginning of a line, patch files don't have this. A special armor header line tells GnuPG about this cleartext signature option.

--escape-from-lines

Because some mailers change lines starting with "From " to "<From " it is good to handle such lines in a special way when creating cleartext signatures. All other PGP versions do it this way too. This option is not enabled by default because it would violate rfc2440.

--passphrase-fd *n*

Read the passphrase from file descriptor *n*. If you use 0 for *n*, the passphrase will be read from stdin. This can only be used if only one passphrase is supplied. Don't use this option if you can avoid it.

--rfc1991

Try to be more RFC1991 (PGP 2.x) compliant.

--openpgp

Reset all packet, cipher and digest options to OpenPGP behavior. Use this option to reset all previous options like --rfc1991, --force-v3-sigs, --s2k-*, --cipher-algo, --digest-algo and --compress-algo to OpenPGP compliant values. All PGP workarounds are also disabled.

--force-v3-sigs

OpenPGP states that an implementation should generate v4 signatures but PGP 5.x recognizes v4 signatures only on key material. This options forces v3 signatures for signatures on data.

--force-mdc

Force the use of encryption with appended manipulation code. This is always used with the newer cipher (those with a blocksize greater than 64 bit). This option might not be implemented yet.

--allow-non-selfsigned-uid

Allow the import of keys with user IDs which are not self-signed. This is only allows the import - key validation will fail and you have to check the validity of the key my other means. This hack is needed for some German keys generated with pgp 2.6.3in. You should really avoid using it, because OpenPGP has better mechanics to do separate signing and encryption keys.

--ignore-time-conflict

GnuPG normally checks that the timestamps associated with keys and signatures have plausible values. However, sometimes a signature seems to be older than the key due to clock problems. This option makes these checks just a warning.

--lock-once

Lock the databases the first time a lock is requested and do not release the lock until the process terminates.

--lock-multiple

Release the locks every time a lock is no longer needed. Use this to override a previous --lock-once from a config file.

--lock-never

Disable locking entirely. This option should be used only in very special environments, where it can be assured that only one process is accessing those files. A bootable floppy with a standalone encryption system will probably use this. Improper usage of this option may lead to data and key corruption.

--no-random-seed-file

GnuPG uses a file to store it’s internal random pool over invocations. This makes random generation faster; however sometimes write operations are not desired. This option can be used to achive that with the cost of slower random generation.

--no-verbose

Reset verbose level to 0.

--no-greeting

Suppress the initial copyright message but do not enter batch mode.

--no-secmem-warning

Suppress the warning about "using insecure memory".

--no-armor

Assume the input data is not in ASCII armored format.

--no-default-keyring

Do not add the default keyrings to the list of keyrings.

--skip-verify

Skip the signature verification step. This may be used to make the decryption faster if the signature verification is not needed.

--with-colons

Print key listings delimited by colons.

--with-key-data

Print key listings delimited by colons and print the public key data.

--with-fingerprint

Same as the command --fingerprint but changes only the format of the output and may be used together with another command.

--fast-list-mode

Changes the output of the list commands to work faster; this is achieved by leaving some parts empty. Some applications don't need the user ID and the trust information given in the listings. By using this options they can get a faster listing. The excact behaviour of this option may change in future versions.

--list-only

Changes the behaviour of some commands. This is like --dry-run but different in some cases. The semantic of this command may be extended in the future. Currently it does only skip the actual decryption pass and therefore enables a fast listing of the encryption keys.

--no-literal

This is not for normal use. Use the source to see for what it might be useful.

--set-filesize

This is not for normal use. Use the source to see for what it might be useful.

How to specify a user ID

There are different ways on how to specify a user ID to GnuPG; here are some examples:

Used to locate the default home directory.

234567C4
0F34E556E
01347A56A
0xAB123456

Here the key ID is given in the usual short form.

234AABBCC34567C4
0F323456784E56EAB
01AB3FED1347A5612
0x234AABBCC34567C4

Here the key ID is given in the long form as used by OpenPGP.

1234343434343434C434343434343434
123434343434343C3434343434343734349A3434
0E12343434343434343434EAB3484343434343434
0xE12343434343434343434EAB3484343434343434

The best way to specify a key ID is by using the fingerprint of the key. This avoids any ambiguities in case that there are duplicated key IDs (which are really rare for the long key IDs).

=Heinrich Heine <heinrichh@uni-duesseldorf.de>

Using an exact to match string. The equal sign indicates this.

<heinrichh@uni-duesseldorf.de>

Using the email address part which must match exactly. The left angle bracket indicates this email address mode.

+Heinrich Heine duesseldorf

All words must match exactly (not case sensitive) but can appear in any order in the user ID. Words are any sequences of letters, digits, the underscore and all characters with bit 7 set.

#34

Using the Local ID. This is a very low level method and should only be used by applications which really need it. The hash character indicates this method. An application should not assume that this is only a number.

Heine
*Heine

By case insensitive substring matching. This is the default mode but applications may want to explicitely indicate this by putting the asterisk in front.

RETURN VALUE

The program returns 0 if everything was fine, 1 if at least a signature was bad, and other error codes for fatal errors.

EXAMPLES

gpg -se -r *`Bob file`*

sign and encrypt for user Bob

gpg --clearsign *file*

make a clear text signature

gpg -sb *file*

make a detached signature

gpg --list-keys *user_ID*

show keys

gpg --fingerprint *user_ID*

show fingerprint

gpg --verify *pgpfile*
gpg --verify *sigfile* [*files*]

Verify the signature of the file but do not output the data. The second form is used for detached signatures, where *sigfile* is the detached signature (either ASCII armored of binary) and [*files*] are the signed data; if this is not given the name of the file holding the signed data is constructed by cutting off the extension (".asc" or ".sig") of *sigfile* or by asking the user for the filename.

ENVIRONMENT

HOME

Used to locate the default home directory.

GNUPGHOME

If set directory used instead of "~/.gnupg".

http_proxy

Only honored when the option --honor-http-proxy is set.

FILES

~/.gnupg/secring.gpg

The secret keyring

~/.gnupg/secring.gpg.lock

and the lock file

~/.gnupg/pubring.gpg

The public keyring

~/.gnupg/pubring.gpg.lock

    and the lock file

~/.gnupg/trustdb.gpg

    The trust database

~/.gnupg/trustdb.gpg.lock

    and the lock file

~/.gnupg/random_seed

    used to preserve the internal random pool

~/.gnupg/options

    May contain options

/usr[/local]/share/gnupg/options.skel

    Skeleton options file

/usr[/local]/lib/gnupg/

    Default location for extensions

WARNINGS

Use a *good* password for your user account and a *good* passphrase to protect your secret key. This passphrase is the weakest part of the whole system. Programs to do dictionary attacks on your secret keyring are very easy to write and so you should protect your "~/.gnupg/" directory very well.

Keep in mind that, if this program is used over a network (telnet), it is *very* easy to spy out your passphrase!

BUGS

On many systems this program should be installed as setuid(root). This is necessary to lock memory pages. Locking memory pages prevents the operating system from writing memory pages to disk. If you get no warning message about insecure memory 3our operating system supports locking without being root. The program drops root privileges as soon as locked memory is allocated.

Anhang D. Glossar

3DES

Triple DES. Symmetrischer Verschlüsselungsalgorithmus, der auf einer dreifachen Verschlüsselung mit DES basiert. Gilt nach heutigen Maßstäben als sicher. Wird vom OpenPGP-Standard zwingend vorgeschrieben und gilt daher als der kleinste gemeinsame Nenner unter den verwendeten Verschlüsselungsalgorithmen. 3DES hat eine Schlüssellänge von 168 Bit, die wirksame Schlüssellänge ist aber aufgrund des eingesetzten Verfahrens 112 Bit.

Algorithmus

Allgemein: (mathematisches) Verfahren, das in kleinste Teilschritte/Handlungsanweisungen unterteilt ist.

Siehe auch: Verschlüsselungsalgorithmus.

ASCII

American Standard Code for Information Interchange. Standard-Zeichensatz für das englische Alphabet. Besteht aus 128 Zeichen, jedes Zeichen wird durch eine 7 Bit lange Zahl dargestellt.

Siehe auch: Binär.

ASCII-armor

Die Ausgabe erfolgt nicht in binärer Form, sondern in Form von am Bildschirm darstellbaren ASCII-Zeichen.

Asymmetrische Verschlüsselung

Bei asymmetrischen Verfahren wird zum Verschlüsseln ein anderer Schlüssel eingesetzt als zum Entschlüsseln. Asymmetrische Verschlüsselung findet in Public-Key-Verfahren Verwendung. Asymmetrische und symmetrische Verschlüsselung lassen sich zu hybrider Verschlüsselung kombinieren. Sowohl zum Verschlüsseln als auch zum Überprüfen von digitalen Signaturen wird der öffentliche Schlüssel eingesetzt, zum Entschlüsseln und zum Signieren der geheime Schlüssel.

Authentisierung

Das Beglaubigen der Identität. Hierzu ist die Eingabe eines Mantra erforderlich. Durch das Beglaubigen der Identität wird verhindert, daß sich eine andere Person als Urheber eines Dokumentes ausgeben kann oder ein für einen bestimmten Empfänger verschlüsseltes Dokument unbefugt lesen kann.

Benutzer-ID

(Engl. User-ID) Identifikation des Benutzers durch Name und E-Mail-Adresse und optional durch einen zusätzlichen Kommentar.

Binär

Im Zweier-Zahlensystem (Binärsystem) dargestellt. Intern werden alle Daten in einem Computer binär dargestellt.

Blowfish

Von Bruce Schneier (http://www.counterpane.com/schneier.html) entwickelter, frei verwendbarer symmetrischer Verschlüsselungsalgorithmus mit 128 Bit Schlüssellänge, der Teil der OpenPGP-Spezifikation ist.

Brute Force

Angriff auf verschlüsselte Daten, bei dem alle möglichen Schlüsselkombinationen durchprobiert werden. Brute-Force-Angriffe sind extrem rechenaufwendig. Selbst wenn alle Rechner der Welt zusammengeschaltet wären, würde das Durchprobieren aller Kombinationen bei einem 128-Bit-Schlüssel einige Milliarden Jahre dauern.

CAST5

Symmetrischer Verschlüsselungsalgorithmus mit 128 Bit Schlüssellänge. In der OpenPGP-Spezifikation vorgeschrieben.

Certification Authority

Zertifizierungsinstanz innerhalb eines hierarchischen Zertifizierungsmodells. Certification Authorities lassen sich auch problemlos in das von GnuPG favorisierte Modell des "Web of Trust" einbeziehen.

Siehe auch: Web of Trust.

Cracker

Person, die vorsätzlich, unbefugterweise und oft mit bösartiger Absicht in fremde Rechnersysteme eindringt, im deutlichen Gegensatz zu “Hacker”, worunter man allgemein einen gutmeinenden Computer-Freak versteht (siehe hierzu auch RFC 1983).

default

Standard, Standard-Einstellung, Voreinstellung.

DES

Digital Encryption Standard. Symmetrischer Verschlüsselungsalgorithmus mit einer Schlüssellänge von 56 Bit. Kann nach dem heutigen Stand der Technik - wenn auch mit erheblichem Aufwand - geknackt werden.

Diffie-Hellmann

Public-Key-Algorithmus. Wird zum sicheren Austausch von Schlüsseln verwendet.

Digest Algorithmus

Hashalgorithmus.

Digitale Signatur

Auch digitale Unterschrift. Aus dem geheimen Schlüssel und einer Datei wird durch Anwendung einer Hash-Funktion eine eindeutige Zeichenfolge gebildet. Mit Hilfe des öffentlichen Schlüssels kann nun jeder überprüfen, ob die Datei tatsächlich von dem angegebenen Urheber bzw Absender stammt und ob der Inhalt verfälscht wurde. Die digitale Signatur ermöglicht es, die Integrität und Authentizität eines elektronischen Dokumentes unabhängig vom Datenformat zu verifizieren.

DSA

Digital Signature Algorithm. Ein von der NSA entwickelter, sehr sicherer Algorithmus zum Signieren von Daten. DSA verwendet den Hash-Algorithmus SHA1.

Eigenbeglaubigung

Auch Selbstunterzeichnung. Indem der Benutzer seinen öffentlichen Schlüssel sowie die Benutzer-ID selbst mit seinem geheimen Schlüssel unterzeichnet, bestätigt er deren Authentizität, und so lassen sich Verfälschungen daran sehr leicht feststellen.

Einweg-Hash

Eine nicht umkehrbare Hashfunktion. Es ist nicht möglich, aus der durch die Hashfunktion erzeugten eindeutigen Prüfsumme die ursprünglichen Daten wieder herzustellen oder auch nur Rückschlüsse darauf zu ziehen.

ElGamal

Auch ELG-E.Asymmetrischer Verschlüsselungs-Algorithmus, der sowohl zum Verschlüsseln als auch zum Signieren benutzt werden kann. Seit 1997 nicht mehr von Patenten gedeckt. Dieser Algorithmus gilt nach den heutigen Maßstäben als sicher und muß von jedem OpenPGP-System unterstützt werden.

Entropie

Begriff aus der Thermodynamik. Maß für die Unordnung eines Systems. Zum Erzeugen echter Zufallswerte benötigt GnuPG Entropie. Diese kann man beispielsweise durch (willkürliche) Festplattenzugriffe, Mausbewegungen oder Tastatureingaben erzeugen.

Falltür-Algorithmus

(Engl. Doortrap Algorithm) Ein Algorithmus, der leicht zu berechnen ist, dessen Umkehrfunktion aber sehr schwer zu berechnen ist. So ist es z.B. leicht, zwei Primzahlen miteinander zu multiplizieren, um eine Nichtprimzahl zu erhalten, es ist aber schwer, eine große Nichtprimzahl in ihre Primfaktoren zu zerlegen.

Fingerabdruck

(Engl. fingerprint) Eindeutige Prüfsumme (Hash) des öffentlichen Schlüssels; ist wesentlich kürzer als der Schlüssel selbst. Wird zum Überprüfen bzw. Verifizieren eines öffentlichen Schlüssels herangezogen.

Freie Software

Software, die nach Belieben - auch kommerziell - benutzt werden darf, z. B. um den Quelltext einzusehen und nach eigenen Vorstellungen abzuändern, oder um die Software in veränderter oder unveränderter Form - ohne ihr allerdings eigene Einschränkungen aufzuerlegen - an andere weiterzugeben. Beispiele für freie Software sind Linux und GnuPG. Siehe auch http://www.gnu.org/philosophy/free-sw.html.

Geheimschlüssel

(Engl. Secret Key, Private Key) Bei asymmetrischen Verfahren der Hauptschlüssel, der sowohl zum Entschlüsseln des Geheimtextes, zum digitalen Signieren von Dokumenten als auch zur Generierung des öffentlichen Schlüssels und der Widerrufurkunde verwendet wird.

Siehe auch: Öffentlicher Schlüssel.

Geheimtext

Die mit Hilfe eines Verschlüsselungsverfahrens verschlüsselten Daten.

Siehe auch: Klartext.

GNU

Gnu's Not Unix. Abkürzung für das seit 1984 bestehende GNU-Projekt der Free Software Foundation (http://www.fsf.org/fsf/fsf.html), dessen Ziel die Schaffung eines auf freier Software basierenden, Unix-ähnlichen Betriebssystems ist. LINUX beruht in großen Teilen auf GNU-Software. Siehe auch http://www.gnu.org

GNU-GPL

GNU General Public License. Eine Lizenz für Freie Software, der auch GnuPG unterliegt. Jeder kann Software, die unter der GPL steht, frei benutzen, modifizieren und weitergeben; die einzigen Einschränkungen sind, daß man keine weiteren Einschränkungen bei der Weitergabe auferlegen darf und daß die Lizenz an sich nicht verändert werden darf.

GnuPG

GNU Privacy Guard. Freie, dem OpenPGP-Standard entsprechende Verschlüsselungs-Software.

Hashfunktion

Auch kryptographische Prüfsumme oder Message Digest. Eine Hashfunktion ist eine Funktion, die aus einer Datei eine eindeutige Prüfsumme errechnet. Beispiele für Hashalgorithmen sind SHA1 und MD5.

Hybride Verschlüsselung

Verschlüsselungsverfahren, bei dem sowohl symmetrische als auch asymmetrische Verschlüsselungsalgorithmen eingesetzt werden. Bei GnuPG (und PGP) werden beispielsweise die eigentlichen Daten mit einem nach dem Zufallsprinzip erzeugten symmetrischen Sitzungsschlüssel verschlüsselt. Dieser Sitzungsschlüssel wird dann, um einen sicheren Schlüsseltausch zu ermöglichen, mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und von GnuPG zu einem Paket zusammengepackt. Auf der Empfängerseite entschlüsselt GnuPG zuerst mit dem geheimen Schlüssel des Empfängers den Sitzungsschlüssel, mit dem dann die ursprünglichen Daten wiederhergestellt werden können.

Siehe auch: Public-Key-Verschlüsselung.

IDEA

International Data Encryption Standard. Symmetrischer Verschlüsselungsalgorithmus mit 128 Bit Schlüssellänge. Der IDEA-Algorithmus ist patentiert; kommerzieller Einsatz erfordert den Erwerb einer Lizenz. IDEA gilt nach den heutigen Maßstäben als sicher und kann von OpenPGP-Systemen optional unterstützt werden. IDEA ist kein OpenPGP-Standard-Algorithmus und wird nicht von allen OpenPGP-Systemen unterstützt.

IETF

Internet Engineering Task Force. Gremium, das technische Standards für das Internet koordiniert. In den sogenannten Workinggroups der IETF, in denen sich Entwickler und Wissenschaftler aus eigenem Antrieb organisieren, werden die Standards erarbeitet und in RFC ("Request forComments") genannten Dokumenten beschrieben. Siehe auch http://www.ietf.org.

Key Escrow

Schlüsselhinterlegung. Maßnahme, um beispielsweise ein Mitlesen verschlüsselter Nachrichten durch staatliche Stellen oder den Arbeitgeber zu ermöglichen.

Key Recovery

Die Möglichkeit, durch eine in die Verschlüsselungs-Software eingebaute "Hintertür" für Unbefugte (staatliche Stellen, Arbeitgeber) das Wiederherstellen des geheimen Schlüssels zu vereinfachen. Key Recovery stellt einen Eingriff in das Grundrecht auf informationelle Selbstbestimmung dar und wird von GnuPG nicht unterstützt. Da GnuPG freie Software ist, kann der Quelltext daraufhin überprüft werden.

Keyserver

Zentrale Internet-Datenbanken zur Verwaltung und Veröffentlichung von öffentlichen Schlüsseln. Ein Keyserver führt keine Zertifizierung der verwalteten Schlüssel durch; dazu bedarf es zusätzlicher Maßnahmen wie z. B. der Einrichtung des Web of Trust.

key space

Der Anzahl an möglichen Schlüsseln. Je länger der Schlüssel ist, desto größer ist auch die mögliche Anzahl an verschiedenen Schlüsseln und desto schwieriger ist es, den Schlüssel durch Raten oder Ausprobieren herauszubekommen.

Klartext

Allgemein: unverschlüsselte Daten.

Siehe auch: Geheimtext.

Kompromittierung

Unbeabsichtigte oder unautorisierte Offenlegung des (Geheim-)Schlüssels bzw. der verschlüsselten Daten.

Kryptographie

Wissenschaft von der Verschlüsselung von Daten und deren Anwendung.

LINUX

Betriebssystem der Unix-Familie, das als freie Software für verschiedene Rechner-Plattformen verfügbar ist. Benannt nach dem Initiator Linus Torvalds, der 1991 die erste Version im Internet veröffentlicht hat, wird es von einem weltweiten Netzwerk von Programmierern ständig weiterentwickelt. Linux basiert in großen Teilen auf dem GNU-Projekt und ist der GNU GPL unterstellt. LINUX ist derzeit das meistverwendete Betriebssystem auf dem Gebiet der Internetserver und gewinnt immer mehr Marktanteile im Desktop-Bereich.

Mantra

Ein Paßwort-Satz. Der geheime Schlüssel ist bei GnuPG noch einmal selbst mit einem Mantra geschützt. Ohne das Mantra kann man den geheimen Schlüssel weder zum Entschlüsseln noch zum Signieren verwenden. Ein sicheres Mantra sollte möglichst lang sein, möglichst wenig Wörter aus dem Wörterbuch/Lexikon enthalten und trotzdem leicht zu behalten sein. Um sich das Mantra zu merken, sollte man es *niemals* aufschreiben, sondern quasi wie ein "Mantra" *still* in sich "hineinmurmeln".

MD5

Message Digest 5. Ein kryptographisch sicherer Hashalgorithmus.

NSA

National Security Agency (http://www.nsa.gov). Amerikanischer Geheimdienst, der sich vorrangig mit Kryptographie und dem weltweiten gezielten Abhören der elektronischen Kommunikation beschäftigt.

Öffentlicher Schlüssel

(Engl. public key) Bei asymmetrischen und hybriden Verschlüsselungsverfahren der frei zugängliche Schlüssel. Mit dem öffentlichen Schlüssel können Daten verschlüsselt und Signaturen überprüft werden. Zum Signieren und Verschlüsseln ist der geheime Schlüssel erforderlich.

OpenPGP

Protokoll, das den Austausch von verschlüsselten Daten, Signaturen und Schlüsseln regelt. Spezifiziert in RFC 2440.

PGP

Pretty Good Privacy. Eine von Philip Zimmermann in den USA entwickelte, weit verbreitete Verschlüsselungs-Software. PGP benutzt den patentierten Algorithmus IDEA und fordert für kommerzielle Anwender den Erwerb einer Lizenz. Der Quelltext von PGP ist öffentlich nicht verfügbar, die Integrität der Software wird von Experten in Frage gestellt.

Primzahl

Auch Primfaktor. Zahl, die nur durch sich selbst oder die Zahl 1 teilbar ist. Die Zerlegung in Primfaktoren spielt bei vielen Verschlüsselungsalgorithmen eine zentrale Rolle.

Privatsphäre

(Engl. Privacy) Im Rahmen dieses Handbuches das Schützen vertraulicher Informationen vor dem Zugriff oder der Manipulation durch Dritte. Weil Daten, die über das Internet oder ein lokales Netzwerk verschickt werden, leicht mitgelesen, abgefangen oder manipuliert werden können, und weil Daten auf einem nicht vernetzten Einzelplatzrechner in der Regel auch nicht sicher vor unbefugten Zugriffen sind, ist die einzige Möglichkeit zum Schutz dieser Privatsphäre eine wirkungsvolle Verschlüsselung.

Public-Key-Verschlüsselung

Verschlüsselungsverfahren mit einem öffentlichen (engl. public key) und einem geheimen Schlüssel (engl. secret key). Der öffentliche Schlüssel wird sowohl zum Verschlüsseln als auch zum Überprüfen von Signaturen benötigt, der geheime zum Entschlüsseln und zum Signieren. Public-Key-Verschlüsselung setzt ein asymmetrisches Verschlüsselungsverfahren voraus. In der Praxis wird meistens die hybride Verschlüsselung eingesetzt.

Quelltext

(Engl. source code) Ein Computer-Programm in seiner ursprünglichen, vom Menschen lesbaren Textform; kann nicht direkt vom Prozessor ausgeführt werden, sondern muß vorher von einem Compiler oder Interpreter in Binärcode übersetzt werden. Kann von Programmierern oder Systemadministratoren verändert, den eigenen Anforderungen angepaßt und auf eventuelle Sicherheitsrisiken geprüft werden. Der Quelltext der meisten gebräuchlichen kommerziellen Software ist nur dem Hersteller zugänglich; erst in letzter Zeit gewinnt freie Software, bei welcher der Benutzer Zugriff auf die Quelltexte hat, zunehmend an Bedeutung. Der Quelltext von GnuPG, welches der GNU GPL unterliegt ist jedermann frei zugänglich.

RIPE-MD-160

Ein Hash-Algorithmus.

RFC

Request For Comments. Dokumente, die (unter anderem) technische Standards für das Internet beschreiben. Die RFCs werden von den sog. Workinggroups der IETF erarbeitet. Der Standard für OpenPGP beispielsweise ist in RFC 2440 (http://www.gnupg.org/rfc2440.html) spezifiziert. Weitere Informationen sowie alle RFCs finden Sie unterhttp://www.rfc-editor.org.

RSA

Ein Algorithmus zum Signieren und asymmetrischen Verschlüsseln von Daten. RSA steht für Rivest, Shamir und Adelman, die Erfinder des Algorithmus. Die Spezifikation von OpenPGP unterstützt die optionale Verwendung von RSA. Dieser Algorithmus ist von Patenten geschützt und daher nicht frei verwendbar.

Schlüssel

Datensequenz, die benutzt wird, um mit einer Verschlüsselungs-Software aus dem Klartext Geheimtext zu erzeugen (Verschlüsselung) und um aus dem Geheimtext den Klartext wieder herzustellen (Entschlüsselung). Auch zum Signieren und Überprüfen einer digitalen Signatur wird ein Schlüssel benötigt.

Schlüssel ID

(Engl. key ID) Eindeutige Kennzeichnung eines Schlüssels. Bestehend aus Schlüssellänge, verwendetem Algorithmus, den letzten 8 Stellen des Fingerabdrucks, dem Erzeugungsdatum und der Benutzer-ID.

Schlüsselbund

(Engl. key ring) Eine Sammlung öffentlicher oder geheimer Schlüssel wird als Schlüsselbund bezeichnet. Bei GnuPG die Dateien pubring.gpg und secring.gpg. Da ein Teilnehmer für jeden Gesprächspartner, dem er verschlüsselte E-Mail schicken will, dessen öffentlichen Schlüssel seinem öffentlichen Schlüsselbund hinzufügt, kann dieser recht groß werden.

Schlüsseleditor

Ein Werkzeug zum Anzeigen und Bearbeiten von Schlüsseln. Der GnuPG-Schlüsseleditor z. B. ist ein interaktives Kommandozeilen-Interface. Es wird mit `gpg --edit-key` , gefolgt von der Schlüssel-ID, aufgerufen.

Schlüssellänge

Sowohl bei symmetrischer als auch bei asymmetrischer Verschlüsselung beruht die Sicherheit ganz und gar auf dem Schlüssel. Deshalb ist die Schlüssellänge ein Maß für die Sicherheit des Systems, doch kann man die Länge eines symmetrischen Schlüssels nicht mit der eines asymmetrischen Schlüssels vergleichen, um Rückschlüsse auf ihre relative Sicherheit ziehen zu können. Heutzutage gelten symmetrische Schlüssel als sicher ab einer Länge von etwa 80 Bit, asymmetrische etwa ab 1024 Bit, vorausgesetzt, daß das zugrundeliegende Verschlüsselungsverfahren keine Schwachstellen enthält.

Schlüsselpaar

Bei Public-Key Verfahren der geheime und der dazugehörige öffentliche Schlüssel.

Schlüssel-Server

Siehe: Keyserver

SHA1

Secure Hash Algorithm One. Von der NSA entwickelter Hashalgorithmus mit einer Schlüssellänge von 160 Bit.

Selbstunterzeichnung

Siehe: Eigenbeglaubigung

Signatur

Auch digitale Signatur. Aus der zu signierenden Datei und dem Geheimschlüssel wird mittels eines Einweg-Hashalgorithmus eine digitale Signatur erzeugt, deren Echtheit man mit dem öffentlichen Schlüssel überprüfen kann. Wird die Datei oder die Signatur verändert, ergibt sich bei der Überprüfung der Signatur eine Fehlermeldung. Mit digitalen Signaturen kann man die Echtheit von digitalen Dokumenten wie beispielsweise Texten, Fotografien, Quelltext bestätigen.

Sitzungsschlüssel

Der symmetrische Schlüssel, mit dem bei OpenPGP-Verfahren die eigentlichen Daten verschlüsselt werden. Dieser Sitzungsschlüssel wird dann mit dem asymmetrischen öffentlichen Schlüssel verschlüsselt. Auf diese Weise kann der Sitzungsschlüssel sicher übertragen werden.

Symmetrische Verschlüsselung

Symmetrisch verschlüsselte Daten müssen mit dem gleichen Schlüssel entschlüsselt werden, mit dem sie auch verschlüsselt worden sind. Symmetrische Verschlüsselung findet Verwendung bei der sicheren Aufbewahrung geheimer Daten. Für den Austausch geheimer Nachrichten ist symmetrische Verschlüsselung nur in Kombination mit asymmetrischer Verschlüsselung sinnvoll (hybrides Verfahren).

subkey

Jeder Schlüssel besteht normalerweise aus zwei Schlüsselpaaren: einem Hauptschlüsselpaar zum Unterschreiben bzw. Prüfen der Unterschrift und einem Unterschlüsselpaar zum Ver- bzw. Entschlüsseln von Nachrichten. (Es kann auch mehrere Unterschlüsselpaare geben.)

Trust-Datenbank

Datei, in der die Vertrauensstufen, die man den verschiedenen Schlüsselbesitzern zuordnet, verwaltet werden.

Twofish

Von Bruce Schneier (http://www.counterpane.com/schneier.html) entwickelter, symmetrischer Verschlüsselungsalgorithmus mit wahlweise 128 oder 256 Bit Schlüssellänge. Der OpenPGP-Standard spezifiziert 256 Bit.

Unix

Familie von Multi-User-Multi-Tasking-Multi-Platform-Betriebssystemen. Während Unix früher ausschließlich auf mittelgroßen und großen Rechenanlagen eingesetzt wurde, gewinnt es heute in Form von LINUX einen wachsenden Markt im Internet, in der Industrie und im Privatbereich.

Verschlüsselung

Das Verändern eines Textes, Bildes bzw. allgemein einer Datei unter Verwendung eines Schlüssels und nach einen festgelegten Verfahren (Verschlüsselungsalgorithmus), mit dem Ziel, den betreffenden Inhalt für andere unkenntlich zu machen, wobei der Vorgang unter Verwendung des Schlüssels wieder umkehrbar ist.

Verschlüsselungsalgorithmus

Methode, nach der aus dem Klartext der Geheimtext erzeugt wird. Es wird unterschieden zwischen symmetrischen und asymmetrischen Verschlüsselungsalgorithmen. Beispiele für Verschlüsselungsalgorithmen: 3DES, Blowfish, ElGamal und Twofish.

Siehe auch: Algorithmus.

Vertrauensstufen

(Engl. trustlevel) Maß für das Vertrauen in die Fähigkeit des Schlüsselbesitzers, Schlüssel sorgfältig zu authentifizieren und zu signieren. Die vier Vertrauensstufen werden folgendermaßen abgekürzt:

q Unbekannt
n kein Vertrauen
m teilweises Vertrauen

f volles Vertrauen

GnuPG entscheidet, ausgehend von dieser Einstufung, ob es von anderen Kommunikationspartnern signierte und authentifizierte Schlüssel als echt anerkennt. Für ein sicheres Web of Trust ist es entscheidend, daß man den einzelnen Benutzer-IDs in seinem Schlüsselbund wohlüberlegte Vertrauensstufen zuweist.

Web of Trust

("Netzwerk gegenseitigen Vertrauens") Hier werden Schlüsselunterschriften benutzt, um die Gültigkeit auch auf Schlüssel auszudehnen, die nicht direkt von Ihnen selbst, sondern von anderen Personen signiert worden sind. Dabei ist nicht das Vertrauen in die andere Person entscheidend, sondern das Vertrauen in deren Fähigkeit, Schlüssel sorgfältig zu authentifizieren und richtig zu signieren. Verantwortungsbewußte Benutzer, die eine gute Schlüsselverwaltung praktizieren, können das Verfälschen des Schlüssels als einen praktischen Angriff auf sichere Kommunikation mit Hilfe von GnuPG abwehren.

Widerrufurkunde

Wenn ein geheimer Schlüssel kompromittiert worden ist, sollte man - um Schäden und Mißbrauch zu vermeiden - die davon abgeleiteten öffentlichen Schlüssel für ungültig erklären. Dies geschieht durch die Veröffentlichung einer aus dem geheimen Schlüssel erzeugten Widerrufurkunde.

Weitere Bücher zum Thema Kryptographie

Reinhard Wobst: *“Abenteuer Kryptologie - Methoden, Risiken und Nutzen der Datenverschlüsselung”* Addison-Wesley Verlag, München, 1998. ISBN 3-8273-1413-5

Bruce Schneier: *“Applied Cryptography - Protocols, Algorithms, and Source Code in C”* John Wiley & Sons, 1995. ISBN 0-471-11709-9

Bruce Schneier: *“Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C”* Addison-Wesley Verlag, München, 1996. ISBN 3-89319-854-7

Christopher Creutzig, Andreas Buhl und Philip Zimmermann: *“PGP Pretty Good Privacy - Der Briefumschlag für Ihre elektronische Post”* Art D’Ameublement 1999. ISBN 3-9802182-9-5

Gisbert W. Selke: *“Kryptographie. Verfahren, Ziele, Einsatzmöglichkeiten”* O’Reilly Verlag, Köln, 2000. ISBN 3-89721-155-6