Email-Verschlüsselung mit Gnu Privacy Guard

Inhaltsverzeichnis

Kryptographie

Kryptographie (auch Kryptografie; von altgriechisch kryptos, bedeutet verborgen oder geheim) war ursprünglich die Wissenschaft der **Verschlüsselung** von Nachrichten und Informationen.
Heute befasst sie sich allgemein mit der Informationssicherheit, also der Konzeption, Definition und Konstruktion von Informationssystemen, die widerstandsfähig gegen unbefugtes Lesen und Verändern sind.
Kryptographie beschäftigte sich seit langem mit der Verschlüsselung, der Umwandlung einer Information von einem lesbaren Klartext in scheinbaren Unsinn (Geheimtext), sowie mit der Entschlüsselung, also mit dem Umwandeln eines Geheimtextes wieder in einen verständlichen Klartext.
Die Methode, die zum Ver- und Entschlüsseln dient, bezeichnet man als **Chiffre**. Das detaillierte Vorgehen einer Chiffre wird in jedem Schritt von sowohl dem **Algorithmus** und dem **Schlüssel** kontrolliert. Letztere ist ein geheimer Parameter, der idealerweise nur den kommunizierenden Parteien bekannt ist.
Als **Kryptosystem** wird die Gesamtheit aller möglichen Elemente, wie Klartexte, Geheimtexte, Schlüssel und V**erschlüsselungsalgorithmus** bezeichnet.

Geschichte der Kryptographie

Die Klassische Kryptographie gab es schon im dritten Jahrtausend v. Chr. in der altägyptischen Kryptographie des alten Reiches.

Im Mittelalter benutzten Gelehrte einfache Zeichenaustauschalgorithmen für den diplomatischen Briefverkehr, so etwa das Alphabetum Kaldeorum.

Ende des 19. Jahrhunderts, mit aufkommen der Telegrafie, gab es neue Überlegungen in der Kryptographie. So formulierte Auguste *Kerckhoffs von Nieuwenhof* mit *Kerckhoffs* Prinzip einen Grundsatz der Kryptographie, nachdem die Sicherheit eines kryptographischen Verfahrens allein auf der Geheimhaltung des Schlüssels basieren soll - das Verfahren selbst muss also nicht geheim gehalten werden. Im Gegenteil: es kann veröffentlicht und von vielen Experten untersucht werden.

Bereits im zweiten Weltkrieg, wo die mechanischen und elektromechanischen Kryptosysteme eingesetzt wurden, hatte man große Fortschritte in der mathematischen Kryptographie erzielt (siehe ENIGMA).

In der modernen Kryptographie mit des 20.Jahrhunderts endete die klassische Kryptographie, wo die Geheimhaltung der Ver- und Entschlüsselung Elementar wichtig war.
Nun müssen sich die modernen Verfahren dem offenen wissenschaftlichen Diskurs stellen.

Mitte der Siebziger gab es zwei wichtige Fortschritte. Erstens war dies DES (Data Encryption Standard), entwickelt von IBM und der NSA für einen sicheren und einheitlichen Standard für die behördenübergreifende Verschlüsselung zu schaffen. Sichere Varianten wie 3DES und AES folgten. Zum anderen war die Veröffentlichung des Artikels *New Directions in Cryptographie* von Whitfield Diffie und Martin Hellmann ausschlaggebend für die neue Methode der Schlüsselverteilung und gab den Anstoß zur Entwicklung von asymmetrischen Kryptosystemen (Public-Key-Verfahren).

Vor dieser Entdeckung waren die Schlüssel symmetrisch, und der Beitz eines Schlüssels erlaubte sowohl das Verschlüsseln als auch das Entschlüsseln einer Nachricht. Dabei traten aber auch Probleme auf wie z. B. ein sicherer Schlüsselaustausch, Anzahl benötigter Schlüssel pro Kommunikationspartner usw.

Bei einem asymmetrischen Kryptosystem wird ein Paar zusammenpassender Schlüssel eingesetzt. Der eine ist ein öffentlicher Schlüssel, der – im Falle eines Verschlüsselungsverfahrens – zum Verschlüsseln von Nachrichten für den Schlüsselinhaber benutzt wird. Der andere ist ein privater Schlüssel, der vom Schlüsselinhaber geheim gehalten werden muss. Mit dieser Methode wird nur ein einziges Schlüsselpaar für jeden Teilnehmer benötigt, da der Besitz des öffentlichen Schlüssels die Sicherheit des privaten Schlüssels nicht aufs Spiel setzt. Ein solches System kann auch zur Erstellung einer digitalen Signatur genutzt werden. Die digitale Signatur wird aus den zu signierenden Daten oder ihrem Hashwert und dem privaten Schlüssel berechnet. Die Korrektheit der Signatur – und damit die Integrität und Authentizität der Daten – kann durch entsprechende Operationen mit dem öffentlichen Schlüssel überprüft werden. Public-Key-Verfahren können auch zur Authentifizierung in einer interaktiven Kommunikation verwendet werden.

Gnu Privacy Guard nutzt die asymmetrische Verschlüsselung (eigentlich eine hybride Verschlüsselung!!!).

Gnu Privacy Guard (GnuPG)

GnuPG hat sich zum Ziel gesetzt, einer möglichst großen Benutzergruppe die Verwendung von kryptographischen Methoden zur vertraulichen Übermittlung von elektronischen Daten zu ermöglichen.

GnuPG unterstützt dazu folgende Funktionen:

- Verschlüsselung von Daten (z.B. Emails), um vertrauliche Informationen an einen oder mehrere Empfänger zu übermitteln, die nur von den Empfängern wieder entschlüsselt werden können.
- Erzeugung einer Signatur über die versendeten Authentizität und Integrität zu gewährleisten.

Beide Funktionen können kombiniert werden. Dabei wird zuerst die Signatur gebildet und an die Daten angehängt. Dieses Paket wiederum kann dann verschlüsselt an die Empfänger gesendet werden. Zudem können vorab Dateianhänge verschlüsselt und danach an die Email angehängt werden.
GnuPG verschlüsselt Nachrichten mit asymmetrischen Schlüsselpaaren. Der öffentliche Schlüssel kann mit anderen Nutzern über eine Vielzahl von Kanälen ausgetauscht werden, z.B. über Internet Schlüsselserver. Es sollte aber damit sehr behutsam umgegangen werden, um Identitätsmanipulationen vorzubeugen, da öffentliche Schlüssel bzw. Identitätsbestimmungen mit dem originären Eigentümer des Schlüssels gefälscht werden können.

Um genau zu sein, verwendet GnuPG ein hybrides Verfahren zur Verschlüsselung. Da asymmetrische Verschlüsselung extrem rechenintensiv ist, wird die eigentliche Nachricht tatsächlich symmetrisch verschlüsselt. Hierfür wird intern ein "Einweg-Schlüssel" erzeugt. Mit diesem Schlüssel wird zunächst die Nachricht verschlüsselt, was nun wesentlich schneller geht. Es wird nur der relativ kurze, symmetrische Schlüssel mit dem asymmetrischen Schlüssel (öffentlicher Schlüssel) gesichert. Der jetzt gesicherte symmetrische Schlüssel wird sodann zusammen mit der gesicherten Nachricht versandt. Der Empfänger nutzt seinen privaten Schlüssel, um Zugang zum geheimen, symmetrischen Schlüssel zu erlangen, und kann damit schließlich die Nachricht

wiederherstellen. Die Erstellung und Verwendung des symmetrischen Schlüssels erfolgt programmintern und für jeden einzelnen Verschlüsselungsvorgang neu und ist für den Anwender vollkommen transparent. Da bei diesem Verfahren sowohl symmetrische als auch asymmetrische Verschlüsselung Verwendung findet, spricht man von *hybrider Verschlüsselung*.
Zurzeit werden verschieden starke Schlüssel eingesetzt mit einer Länge bis zu 4096 Bit.
GnuPG verwendet aus patentrechtlichen Gründen die Algorithmen von Elgamal, CAST5, Tripple-DES(3DES), AES(Rijndael) und Blowfish.

Um GnuPG in verschiedenen Anwendungskontexten zu benutzen, sind zahlreiche Frontends erstellt worden. Hier können die folgenden Frontend-Typen unterschieden werden:

- Frontends, die die Funktionen des kommandozeilenorientierten Programmes über eine graphische Oberfläche zur Verfügung stellen, wie z.B. der Gnu Privacy Assistant (GPA), Seahorse und KGpg für Gnome und KDE, GPGTools für Apple OS X, sowie Gpg4win für Windows
- Mailprogramme, die GnuPG entweder direkt (z.B. Evolution, KMail, Mutt) oder über ein Plug-In (Enigmail für Mozillas Email-Programme, EudoraGPG für Eudora, GPGol für Outlook, GPGMail für Apple Mail) einbinden können.
- Chatprogramme wie Gabber, Miranda IM, licq, Kopete, PSI oder Gajm, die so teilweise auch plattformübergreifende verschlüsselte Chats über Netzwerke wie ICQ ermöglichen.
- Serverbasierende Frontends wie GNU Anubis oder freenigma, die als SMTP-Relay-Server oder als MTA eine zentralisierte und transparente Email-Verschlüsselung erlauben.
- Für den Webbrowser Mozilla Firefox gab es ein Add-on namens FireGPG, das auf jeder Internetseite GPG-Blöcke erkennt und verarbeitet, es wird jedoch seit Juni 2010 bzw. der Version 0.8 nicht mehr weiterentwickelt.
- Daneben gibt es noch weitere Schnittstellen für die Nutzung von GnuPG aus verschiedenen Skriptsprachen wie PHP, Perl und Python.

Installation

Das Paket ***gnupg*** ist bei allen derzeitigen Ubuntu-Versionen vorinstalliert. Optional kann man noch folgendes Paket installieren ***gnupg-doc***, welches die Dokumentation der Software enthält.

Sie befindet sich danach in **/usr/share/doc/gnupg-doc/GNU_Privacy_Handbook/de/html/index.html**

Zudem ist es sinnvoll oder teilweise auch nötig den GPG-Agent zur Passwortverwaltung zu benutzen.

Bei Nutzung von GnuPG in externen Mailprogrammen muss evtl. ein Plugin nachinstalliert werden.
Bei Mozillas Thunderbird wird z.B. Enigmail als Add-on heruntergeladen. Danach ist OpenPGP in Thunderbird verfügbar, das GnuPG verwendet.

Schlüsselpaar

GnuPG speichert alle Schlüssel, die man mit der Zeit sammelt, in einem „Schlüsselbund“ im Home-Verzeichnis ~/.gnupg/.
Genau genommen handelt es sich um zwei Schlüsselbunde. Eines (secring.gpg) enthält die eigenen, geheimen Schlüssel, und das andere (pubring.gpg) enthält die öffentlichen Schlüssel, sowohl die eigenen als auch die der Kommunikationspartner. Alle GnuPG-Operationen werden mit dem Kommandozeilenwerkzeug **gpg** durchgeführt.

Schlüssel auflisten

Mit den folgenden Befehlen kann man seine GnuPG-Schlüssel auflisten lassen. Die Optionen können dabei jeweils in der langen, beschreibenden Version, oder in der kurzen, ein-buchstabigen Variante benutzt werden.
Entweder die eigenen, geheimen Schlüssel:

```
gpg --list-secret-keys
gpg --K
```

Oder die öffentlichen Schlüssel:

```
gpg --list-keys
gpg --k
```

“Fingerprint” anzeigen

Ein Fingerprint (dt.: Fingerabdruck) ist ein relative kurzer Hash-Wert, mit dem man Schlüssel verifizieren kann. Dieser identifiziert einen Schlüssel so wie der menschliche Fingerabdruck einen Menschen eindeutig identifiziert. Nach heutigem Kenntnisstand ist es so gut wie aussichtslos zu versuchen, einen zweiten Schlüssel mit demselben Fingerprint erschaffen zu wollen. Man kann (und sollte) das dafür nutzen, sich von der Echtheit solcher Schlüssel zu überzeugen, die man von einem Schlüsselserver oder per Email erhalten hat. Dieser Fingerabdruck besteht aus einer Zeile Hexadezimalzahlen, die man bspw. auf Visitenkarten drucken oder am Telefon übermitteln kann.

Anzeigen lässt sich der Fingerabdruck eines Schlüssels mit:

```
gpg --fingerprint <ID oder Name des Schlüssels>
```

Schlüsselpaar erzeugen

Ein neues Schlüsselpaar aus privatem und zugehörigem öffentlichen Schlüssel kann ganz einfach im Terminal erzeugt werden:

```
gpg --gen-key
```

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

Bei der Wahl des Schlüssels sollte nur mit entsprechendem Hintergrundwissen von der Standardauswahl abgewichen werden !!!

Anschließend wird nach der Schlüsselstärke gefragt. Je nach Art des Schlüssels sind unterschiedliche Werte möglich. Der mögliche Minimal- bzw. Maximalwert wird vom Programm ausgegeben. Je höher desto sichere – aber auch umso langsamer – arbeitet der Schlüssel.

```
RSA Schlüssel können zwischen 1024 und 4096 Bits lang sein.
Welche Schlüssellänge wünschen Sie? (2048)
```

Anschließend muss man auswählen, wie lange der Schlüssel gültig sein soll. Hier sollte man der Versuchung widerstehen, eine unbeschränkte Gültigkeit festzulegen, da dieser Schlüssel dann auch bei Verlust ewig gültig ist und niemals von den Keyservern verschwindet. Irgendetwas zwischen 1 und 5 Jahren ist dagegen ein praktischer Wert.

```
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
Wie lange bleibt der Schlüssel gültig? (0)
```

Hinweis:
Das Ablaufdatum kann auch noch im Nachhinein geändert werden. Um diese Änderung seinen Gesprächspartnern mitzuteilen, muss man ihnen seinen öffentlichen Schlüssel erneut schicken bzw. ihn erneut zum Schlüsselserver hoch laden.

Nun muss der Name und die EMail-Adresse sowie optional ein Kommentar angegeben werden, für die das Schlüsselpaar gelten soll:

```
Sie benötigen eine Benutzer-ID, um Ihren Schlüssel eindeutig zu machen; das
Programm baut diese Benutzer-ID aus Ihrem echten Namen, einem Kommentar und
der E-Mail-Adresse in dieser Form auf:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
Ihr Name ("Vorname Nachname"): Vorname Nachname
E-Mail-Adresse: user@example.com
Kommentar: optional
Sie haben diese User-ID gewählt:
    "Vorname Nachname (optional) <user@example.com>"
Ändern: (N)ame, (K)ommentar, (E)-Mail oder (F)ertig/(B)eenden?
```

Zu guter Letzt wird nochmal gefragt, ob die Angaben ok sind, (dies kann mit F bestätigt werden) und eine Passphrase für den privaten Schlüssel wird benötigt, um diesen zu schützen. Diese Passphrase braucht man später, um Mails signieren und entschlüsseln zu können. Das Passwort sollte nicht leicht zu erraten sein und mindestens 8 Stellen besitzen. Nun wird der Schlüssel erstellt, wobei eine große Menge echter (nicht Pseudo-)Zufallswerte benötigt werden. Es kann deswegen sein, dass man vom System gebeten wird, ein wenig mit Maus und Tastatur zu spielen um solche Werte besser erzeugen zu können.

```
Wir müssen eine ganze Menge Zufallswerte erzeugen. Sie können dies
Unterstützen, indem Sie z. B. in einem anderen Fenster/Konsole irgendetwas
tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.
(Es werden noch xx Byte benötigt.)
```

Nun liegt der Schlüssel in ~/.gnupg als secring.gpg und man kann ihn wie oben beschrieben auflisten lassen. Mit dem öffentlichen Schlüssel verschlüsselte Nachrichten oder auch Dateien, können nur mithilfe des dazugehörigen privaten Schlüssels wieder entschlüsselt werden.

Achtung:
Wenn man plant, den Schlüssel an mehr als eine Handvoll Leute weiterzugeben (bzw. ihn auf einen Keyserver hochzuladen), denen man bei Verlust oder Kompromittierung des privaten Schlüssels persönlich

Bescheid sagen kann, sollte man sich am besten direkt nach der Erstellung des Schlüssels ein sogenanntes Widerrufszertifikat erstellen, mit dem man später den Schlüssel – auch ohne dass dieser vorhanden sein muss – für ungültig erklären kann.

Schlüssel importieren

Erhält man den öffentlichen Schlüssel eines Kommunikationspartners oder will man ein Schlüsselpaar (also einen privaten und einen öffentlichen Schlüssel) importieren, so muss man ihn zur weiteren Verwendung erstmals in den Schlüsselbund importieren.

```
gpg --import /tmp/dateiname.asc
```

Man sollte aber auf keinen Fall Schlüssel ungeprüft verwenden!!! Entweder man nimmt persönlichen Kontakt zum Kommunikationspartner auf und gleicht den Fingerprint ab, oder man nutzt das GnuPG/Web of Trust.

Schlüssel exportieren

Ebenso kann man einen Schlüssel exportieren. Das ist z. B. notwendig, wenn man seinen öffentlichen Schlüssel auf seiner Webseite zum Download anbieten will, oder wenn man den Schlüssel eines Kommunikationspartners signiert hat und ihm diesen jetzt mit dieser Signatur zukommen lassen möchte (siehe auch GnuPG/Web of Trust).

```
gpg –a --output gpg-key.asc --export <optional Schlüssel-ID oder Name>
```

Der Schlüssel befindet sich danach in der Datei gpg-key.asc im aktuellen Verzeichnis und kann als EMail-Anhang verschickt oder auf den Webspace hochgeladen werden. Bei dieser Befehlsvariante wird der private Teil eines Schlüsselpaares – falls vorhanden – nicht exportiert.

Um auch den privaten Schlüssel exportieren zu können, müssen andere Befehlsoptionen verwendet werden. Dies dient dazu, ein versehentliches

Veröffentlichen des privaten Schlüssels zu verhindern. Der folgende Befehl sollte also nur mit Bedacht verwendet werden:

```
gpg –a --output gpg-secret-key.asc --export-secret-keys <optional Schlüssel_ID oder Name>
```

Schlüssel signieren

Um einen Schlüssel zu signieren, wird folgender Befehl verwendet:

```
gpg --sign-key <ID oder Name des Schlüssels>
```

Vertrauensstufe importieren/exportieren

Jedem Schlüssel ist der Grad des Vertrauens in diesen zugewiesen. Auch diese Eigenschaft lässt sich exportieren, um sie z. B. auf einem anderen Rechner wiederzuverwenden. Mehr Informationen zum Thema Vertrauen findet man im Artikel GnuPG/Web of Trust unter http://wiki.ubuntuusers.de/GnuPG/Web_of_Trust.
Das Exportieren dieser Information geschieht mit dem Befehl

```
gpg --export-ownertrust > trust.txt
```

Zu beachten ist, dass hierbei die Information zu jedem Schlüssel im Schlüsselring exportiert wird. Eine Einschränkung ist nicht möglich.

Eine so erzeugte Datei kann anschließend mit

```
gpg --import-ownertrust < trust.txt
```

importiert werden.

Zusätzliche EMail-Adressen

Viele Leute besitzen inzwischen mehrere EMail-Adressen, aber die wenigsten wollen für jede einen eigenen Schlüssel erzeugen und sich die ganzen unterschiedlichen Passphrasen merken. Deswegen kann man dem eigenen Schlüsselpaar zusätzliche Identitäten hinzufügen. Der Schlüssel gilt dann für alle diese EMail-Adressen. Das geht so:

```
gpg --edit-key <Schlüssel-ID oder Name>
Befehl> adduid
```

Dann auf Nachfrage den neuen Namen (bzw. denselben), die zusätzliche EMail-Adresse und optional einen zugehörigen Kommentar angeben, mit F bestätigen und die Passphrase des Schlüssel eingeben. Danach das Schlüssel-Edit-Programm verlassen:

```
Befehl> save
```

Der Schlüssel ist jetzt mit der zusätzlichen Identität verknüpft, was man durch einen Aufruf von gpg –list-keys überprüfen kann.

Verlängerung des Schlüssels mit GnuPT

Solange die Schlüssel noch gültig sind, kann man dies bequem über den Eintrag »Schlüssel bearbeiten« des Kontextmenüs von GnuPT erledigen. Im erscheinenden Dialog kann hierfür das Kommando »EXPIRE« ausgeführt werden.

Verlängerung der Schlüssel mit GnuPG in der Kommandozeile

Wenn die Schlüssel bereits abgelaufen sind, kann man GnuPT nicht nutzen. Die Verlängerung der Schlüssel kann daher nur direkt mit GnuPG erledigt werden. Nachfolgend sind die erforderlichen Schritte beschrieben:
1. In der Kommandozeile

```
gpg --edit-key mustermann@example.com
```

ausführen.

Die aktuellen Daten des Schlüssels werden angezeigt:

```
gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
  This is free software: you are free to change and redistribute it.
  There is NO WARRANTY, to the extent permitted by law.

  Geheimer Schlüssel ist vorhanden.

  pub  1024D/12345678  erzeugt: 2006-01-01  verfallen: 2008-12-31  Aufruf: SC
                       Vertrauen: uneingeschränkt Gültigkeit: verfallen
  sub  2048g/87654321  erzeugt: 2006-01-01  verfallen: 2008-12-31  Aufruf: E
  [verfall.] (1). Max Mustermann <max.mustermann@example.com>
  [verfall.] (2)  Max Mustermann <mustermann@example.com>
```

2. Optional: Unterschlüssel (Sub-Key) auswählen

Die folgenden Befehle werden standardmäßig auf den Hauptschlüssel angewendet.

Falls nicht die Gültigkeit des Hauptschlüssels, sondern die Gültigkeit des ersten Unterschlüssels verlängert werden soll, ist dieser mit »key 1« auszuwählen und die Eingabetaste zu betätigen. Der zweite Unterschlüssel wird mit »key 2« adressiert usw.

3. Befehl »expire« eingeben und Eingabetaste betätigen

Die verschiedenen Optionen zur Erweiterung der Gültigkeit werden angezeigt:

```
Ändern des Verfallsdatums des Hauptschlüssels.
  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
  Wie lange bleibt der Schlüssel gültig? (0)
```

4. Gewünschten Zeitraum – z. B. »1y« für 1 Jahr – eingeben und Eingabetaste betätigen

```
  Key verfällt am 01/12/11 22:22:22
  Ist dies richtig? (j/N)
```

5. Eingabe des Zeitraums mit »j« bestätigen und Eingabetaste betätigen

```
Sie benötigen eine Passphrase, um den geheimen Schlüssel zu entsperren.
  Benutzer: "Max Mustermann <max.mustermann@example.com>"
  1024-Bit DSA Schlüssel, ID 12345678, erzeugt 2006-01-01
```

5. Passphrase eingeben und mit der Eingabetaste bestätigen

Die aktualisierten Daten des Schlüssels werden jetzt angezeigt:

```
pub  1024D/12345678  erzeugt: 2006-01-01  verfällt: 2010-01-12  Aufruf: SC
                     Vertrauen: uneingeschränkt Gültigkeit: uneingeschränkt
sub  2048g/87654321  erzeugt: 2006-01-01  verfallen: 2008-12-31  Aufruf: E
[ uneing.] (1). Max Mustermann <max.mustermann@example.com>
[ uneing.] (2)  Max Mustermann <mustermann@example.com>
```

6. Vorgehensweise für weitere Unterschlüssel wiederholen

Falls auch der Sub-Key abgelaufen ist, ist die Vorgehensweise zu wiederholen und in Punkt 2. der gewünschte Unterschlüssel auszuwählen.

7. Mit »quit«beenden und die Eingabetaste bestätigen

```
Änderungen speichern?
```

8. Taste »j« und die Eingabetaste zur Speicherung der Daten betätigen

Die Daten werden gespeichert und GnuPG beendet.

Widerrufszertifikat erstellen

Um zu einen späteren Zeitpunkt seinen eigenen öffentlichen Schlüssel als ungültig erklären zu lassen, besteht die Möglichkeit, ein Widerrufszertifikat zu erstellen. Dies ist zum Beispiel nötig, wenn der eigene private Schlüssel und/oder die Passphrase an dritte gpg –gen-revoke
Dazu wird mittels:

```
gpg --gen-revoke <ID oder Name des Schlüssels>
```

ein Zertifikat erstellt.

Die eigentliche Prozedur beginnt aber erst nach einer Sicherheitsabfrage, bei der die entsprechenden Schlüsseldaten nochmals genau überprüft werden sollten:

```
sec  1024D/12345678 2006-06-01 Max Mustermann <max.mustermann@exampel.com>

Ein Widerrufszertifikat für diesen Schlüssel erzeugen? (y/N)

Jetzt wird folgende Auswahl angeboten:

Grund für den Widerruf:
  0 = Kein Grund angegeben
  1 = Hinweis: Dieser Schlüssel ist nicht mehr sicher
  2 = Schlüssel ist überholt
  3 = Schlüssel wird nicht mehr benutzt
  Q = Abbruch
(Wahrscheinlich möchten Sie hier 1 auswählen)
Ihre Auswahl?
```

Dies dient nur zur Information und hat auf das Zertifikat sonst keinen Einfluss. Sinnvollerweise wählt man hier 1, wenn man einen vorsorglichen Widerruf erzeugt. Wenn irgendwann mal 2 oder 3 zutreffen sollten, kann man immer noch ein neues Widerrufszertifikat erzeugen.

Nun kann noch eine eigens verfasste Information angegeben werden. Die Eingabe hier wird durch eine leere Zeile beendet. Abschließend wird man zur Eingabe des Passwortsatzes aufgefordert. Das Zertifikat wird dann auf der Konsole ausgegeben:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.2.2 (GNU/Linux)
Comment: A revocation certificate should follow

iEkEIBECAAkFAkZMUB8CHQAACgkQ543ylotjOHgprACgzTk24kFEFbxmeqt9UThB
```

```
/yqmZBIAnjO9TqC1YQqS25c84hsvkzslUisH
=99Xw
-----END PGP PUBLIC KEY BLOCK-----
```

Dieser Abschnitt stellt das Zertifikat dar und sollte als Datei gesichert werden. Man kann diese Ausgabe auch gleich bei der Erzeugung in eine Datei umleiten:

```
gpg --output Widerruf_<Kennung>.asc --gen-revoke <Kennung>
```

Mit dem eben erzeugten Zertifikat kann der Schlüssel bei Bedarf widerrufen werden. Dazu muss dieses Zertifikat in den Schlüsselring importiert werden. Angenommen, das Zertifikat ist in der Textdatei Widerruf_12345678.asc gespeichert, so müsste folgendes Kommando ausgeführt werden:

```
gpg --import Widerruf_12345678.asc
```

Es erfolgt keine Rückfrage. Der öffentliche Schlüssel wird sofort widerrufen.

Dateioperationen

Mit den folgenden Befehlen kann man Dateien mit GnuPG verschlüsseln und /oder signieren. In den meisten Fällen, wenn man GnuPG z. B. für die EMail-Kommunikation benutzt, muss man sich mit diesen Optionen nicht weiter beschäftigen, da die Mailprogramme sich selber um die Verschlüsselung kümmern. Die folgenden Befehle benötigt man also nur, wenn man bspw. Dateien verschlüsselt auf der eigenen Festplatte speichern will, oder wenn man ein Softwarepaket, welches man auf der eigenen Homepage zum Download anbietet, digital signieren will.
In den folgenden Beispielen wird davon ausgegangen, dass man eine fiktive Datei test.txt behandeln will.

Verschlüsseln

Wenn man etwas verschlüsseln will, musss auch angegeben werden, für wen das geschehen soll. Genauer gesagt, mit wessen öffentlichen Schlüssel die Datei verschlüsselt werden soll. Der Befehl hierfür sieht folgendermaßen aus:

```
gpg --encrypt –a --recipient <Schlüssel-ID oder Name> test.txt
```

Es entsteht eine Datei test.txt.asc

Hinweis:
Wenn man die Datei mit dem öffentlichen Schlüssel von jemand anders verschlüsselt hat, bedeutet dies, dass nur derjenige die Datei wieder entschlüsseln kann. Selber kann man dies nicht, da man nicht in Besitz des geheimen Schlüssels des Kommunikationspartners ist. Wenn man die Datei selber noch benötigt, sollte man also eine Kopie der unverschlüsselten Datei behalten. Alternativ kann diese zusätzlich mit dem eigenen Schlüssel verschlüsselt werden – die Option --recipient kann mehrfach angegeben werden.

Entschlüsseln

Der Inhaber des privaten Schlüssels kann diese Datei jetzt auf folgende Weise wieder in die ursprüngliche Version verwandeln (Hierfür wird natürlich die Passphrase benötigt).

```
gpg --decrypt --output entschluesselt.txt test.txt.asc
```

In entschluesselt.txt steht nun wieder der Text. Wenn man den Parameter --output (und den Dateinamen) weglässt, gibt GnuPG den entschlüsselten Text in der Konsole aus.

Hinweis:
Anstelle der langen Optionsnamen --encrypt, --decrypt, --recipient und --output kann man auch die Kurzversionen –e, -d, -r und –o verwenden. Die Option –a ausgeschrieben --armor bedeutet, dass die verschlüsselte Datei im *ASCII-Armor*-Format gespeichert wird. Lässt man diese Option weg, wird das Ergebnis in einem Binärformat abgelegt und erhält die Endung **.gpg** bzw. bei Signaturen **.sig**, anstatt **.asc**.
Die Ausgabe im *ASCII-Armor*-Format benötigt 33 Prozent mehr Speicherplatz als die im Binärformat, enthält jedoch nur druckbare Zeichen. Sie eignet sich daher besser zum übertragen über das Internet wie etwa bei Emails oder zum Einbinden in HTML-Seiten. Sollen die Daten hingeggpg --decrypten verschlüsselt auf der Festplatte abgelegt werden, ist das Binärformat gerade bei großen Dateien vorzuziehen.
Im Internetgebrauch sollte man sich daher den Einsatz der Option –a angewöhnen.

Signieren

Es gibt mehrere Verfahren, eine Datei digital zu signieren. In den meisten Fällen wird man die folgende Option verwenden:

```
gpg --detach-sign –a test.txt
```

Nachdem man seine Passphrase erfolgreich eingegeben hat, erhält man eine Datei test.txt.asc. Diese enthält eine digitale Signatur der Originaldatei mit der sich die Authentizität beweisen lässt. Hier ein Beispiel:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.1 (GNU/Linux)
iD8DBQFDTqEJxHY64NcyylkRAvZ2AJoDxlQ4VuqpIDfcTFnHEye4PGfNIACdEcRw
QVoj5npyj4VyaEzLzx4vdSs=
=ySv1
-----END PGP SIGNATURE-----
```

Man kann auch die Kurzform –b verwenden. Verwendet man stattdessen die Option --clearsign, so wird nicht nur die Signatur in die Ausgabedatei geschrieben, sondern die ganze Ursprungsdatei mit angehängter Signatur am Ende. Das kann praktisch sein, wenn man nur eine einzige Datei übertragen will. Allerdings muss dann der Empfänger im Falle von ausführbarem Code diese Signatur erstmal wieder aus der Datei entfernen, um sie verwenden zu können.
Zum Signieren wird immer der default-Schlüssel verwendet. Sollte man mehrere geheime Schlüssel besitzen, kann man mit dem Parameter --local-user den Schlüssel angeben, den man für die Signatur verwenden will.

Signatur verifizieren

Eine Signatur überprüft man mit folgendem Kommando:

```
gpg –verify test.txt.asc
```

```
gpg: Signature made Mon 19 Jun 2006 17:06:05 CEST using DSA key ID <Key_ID>
gpg: Good signature from “Person von der die Signatur stammt”
```

Wenn die signierte Datei nicht denselben Dateinamen wie die Signatur (abzgl. Der **.asc**- oder **.sig**-Endung) trägt, kann man diesen Dateinamen extra angeben:

```
gpg --verify signaturdatei.asc test.txt
```

Sollte die Signatur nicht zu dem Text passen, weil die Datei nachträglich manipuliert wurde, wird dies durch BAD sigantur from … angezeigt. Das dritte mögliche Ergebnis ist, dass GnuPG die Signatur nicht verifizieren kann, weil man den benötigten Schlüssel nicht im Schlüsselbund hat. In diesem Fall muss man sich erstmal den öffentlichen Schlüssel des Absenders besorgen.

Quellen:

http://www.wikipedia.de

http://wiki.ubuntuusers.de/

http://www.stephan-nix.de

http://www.wikibooks.org