GnuPG

Christian Barthel

2010-12-18

Inhaltsverzeichnis

1 Allgemein

Informationssicherheit:

- Vertraulichkeit
- Integrität
- Authentizität

GnuPG ist ein Programm zum Verschlüsseln und Signieren von digitalen Daten (entspricht OpenPGP-Spezifkation und ist kompatibel zu PGP 5.x). GnuPG verwendet ein hybrides Verfahren mit öffentlichem Schlüssel. Zum Verschlüsseln kann GnuPG aber ebenso symmetrische Verfahren einsetzen.

2 Konzepte

Verwendete kryptographische Verfahren:

- symmetrische Verschlüsselung
- Public-Key-Verschlüsselung
- Einweg-Hashing

2.1 Symmetrische Verschlüsselung

- benutzt zum Ver- und Entschlüsseln denselben Schlüssel
- Absender und Empfänger müssen den gleichen Schlüssel besitzen
- moderne Beispiele: Blowfish und IDEA
- geschichtliches: Enigma
- Schlüssel muss bei diesem Verfahren geheim bleiben
- äußerst wichtig ist die Schlüsselänge
- ein Schlüssel sollte möglichst lang sein
- DES hat 56 Bit: 2 hoch 56, also genau 72.057.594.037.927.936 Schüssel sind möglich
- dies scheint sehr viel zu sein, jedoch steigt die Rechenleistung sehr schnell an und so kann DES innerhalb von Tagen alle möglichen Schlüssel ausprobieren
- Blowfish und IDEA benutzen 128 Bit, also 2 hoch 128 mögliche Schlüssel: 340.282.366.920.938.463.463.374.607.431.768.211.456

2.2 Public-Key Verschlüsselung

- Hauptproblem symmetrischer Verfahren: Schlüsselaustausch
- Außerdem ist die Anzahl der Schlüssel ein weiterer negativer Aspekt des symmetrischen Verfahrens: n(n-1)/2
- Sinn des Public-Key Verfahrens ist es, dass das Sicherheeitsrisiko beim gegenseitigen Schlüsselaustausch gänzlich vermieden wird
- Jeder besitzt ein Schlüsselpaar: einen öffentlichen und einen geheimen Schlüssel
- zum Verschlüsseln einer Nachricht wird der öffentliche Schlüssel benutzt
- der Empfänger kann diese Nachricht wiederum nur mit seinem geheimen Schlüssel entschlüsseln
- die Verschlüsselung basiert auf einem sogenannten Falltür-Algorithmus bzw. Einweg-Hash: Es handelt sich um Funktionen, die einfach zu berechnen sind, doch umgekehrt ist es praktisch unmöglich, aus dem Ergebnis dieser Hash-Funktion wieder den Ausgangswert zu berechnen
- Es ist leicht, 2 Primzahlen 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: Wenn z. B. eine aus 2 Primfaktoren bestehende Zahl vorhanden ist, so macht die Kenntnis eines der Faktoren es leicht, den zweiten zu berechnen
- Angenommen, ein Verfahren beruht 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.
- Auch hier ist die Schlüsselgröße von entscheidender Bedeutung: bei 80 Bit muss der Angreifer bis zu 2 hoch 80-1 Schlüssel ausprobieren, um auf den richtigen zu stoßen
- Empfohlen: 1024/2048 Bits

2.3 Hybride Verfahren

- Public Key Verschlüsselung ist wichtig für den Schlüsselaustausch

- Hybride Verfahren benutzen sowohl eine symmetrische Verschlüsselung als auch ein asymmetrisches Public Key Verfahren
- es gibt Sitzungsschlüssel, die mit Hilfe des Public Key-Verfahren ausgetauscht werden

2.4 Digitale Unterschriften

- Hash-Funktion = kryptographische Prüfsumme
- es kann ein eindeutiges Abbild der Ursprungsdatei in einer wesentlich kürzeren Form dargestellt werden
- Digitale Unterschrift ist die Anwendung einer Hash-Funktion auf das Dokument
- Es ist unmöglich, zwei unterschiedliche Dokumente mit dem gleichen Hash-Ergebnis zu erstellen
- Außerdem kann aus einem Hash-Ergebnis nicht das ursprüngliche Dokument erzeugt werden
- Public Keys können auch zum Unterschreiben von Dokumenten benutzt werden
- der Unterzeichner verschlüsselt das Dokument mit seinem privatem Schlüssel
- jeder der die Unterschrift sehen möchte, kann die Signatur wieder mit dem öffentlichen Schlüsselentschlüsseln
- Hash-Algorithmen: SHA1, RIPE-MD160
- die Unterschrift kann überprüft werden, indem man selbst die Hash-Funktion auf die Kopie des Dokuments anwendet.
- Damit die Unterschrift nicht manipuliert werden kann wird mit dem privatem Schlüssel verschlüsselt, und jeder kann mit dem öffentlichen Schlüssel diese Unterschrift wiederum prüfen

3 Allgemein

3.1 Schlüsselpaare erzeugen

Um GnuPG verwenden zu können (Verschlüsseln, Entschlüsseln, Signieren) benötigen Sie zunächst einen geheimen und einen öffentlichen Schlüssel:

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

- Standardmäßig werden 2 Schlüsselpaare erezugt, basierend auf RSA.
- Als nächstes muss die Schlüsselgröße in Bits angegeben werden.
- Schließlich muss noch ein Verfallsdatum gewählt werden.
- Im nächsten Schritt muss dieser Schlüssel einer realen Person zugeordnet werden.
- Der Zugriff auf den Schlüssel wird durch eine Passphrase geschützt

3.2 Widerrufurkunde

Nachdem erstellen eines Schlüsselpaars sollte sofort eine Widerrufurkunde erzeugt werden. Vergessen oder verlieren sie Ihren Schlüssel, so können Sie mit dieser Widerrufurkunde andere davon in Kenntnis setzen, dass der dazugehörige öffentliche Schlüssel nicht mehr benutzt werden kann.

```
gpg --output revoke.asc --gen-revoke [Schlüsselid|Schlüsselname]
```

Die erzeugte Widerrufurkunde findet sich in der Datei revoke.asc wieder. Die Widerrufurkunde sollte ebenfalls geheim bleiben.

3.3 Anzeigen von Schlüsseln

Um nun mit anderen Benutzern kommunizieren zu könnn, müssen Sie Ihren öffentlichen Schlüssel austauschen.

```
$ ~ $ gpg --list-keys
pub   2048R/20DA92AC 2010-12-14 [expires: 2011-01-03]
uid                  Christian Barthel <christian@barthel-connect.de>
sub   2048R/1423BA72 2010-12-14 [expires: 2011-01-03]
```

3.4 Exportieren des öffentlichen Schlüssels

Mittels –export kann ein Schlüssel in eine Datei exportiert werden. Zur Identifikation des zu exportierenden Schlüssels dient entweder die Schlüssel-ID oder ein Teil der Benutzer-ID:

```
gpg --output cba.gpg --export christian@barthel-connect.de
```

Der Schlüssel wird mit der Endung *.gpg in einem binärem Format gespeichert. Dies kann unerwünscht sein, wenn der Schlüssel z. B. auf einer Webseite oder per E-Mail verteilt werden soll. Abhilfe schafft hier –armor oder -a. So wird der Schlüssel im ASCII-Format dargestellt.

```
$ ~/Documents/secret $ gpg -a --export christian@barthel-connect.de
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

mQENBE0HZN8BCACoIpMKW4D99lgWgNiFlp+BG80/SW7rXdsyy7PKYiaRfxm018pk
rtrpX8eTed45zsydueICGv3Fuk6bQNhocwSSp91msmc9aG25rAsky7h3rAnsl5b2
Vl1hH61dw96bD4G/gZAjrwv3FuFAlAgIVIaV0qU4zngNstZMbgob3Dt8PkpKzfFu
FSJxzSdZaRc44S0rJb5KCwRBRQBM24YwO9N6kBFDG0dSyoYulAI9bZq7RJxV/QYV
zwww2Uop1V2hQYvBpTMzkcGwXIKoYFjU1Zxyx4Jx2B0teVlYADuNY3DSe+QLm41K
4H1l1v2DbJvUTRbke8HsLbnJN0G3IxTfV/9bABEBAAG0MENocmlzdGlhbiBCYXJ0
aGVsIDxjaHJpc3RpYW5AYmFydGhlbC1jb25uZWN0LmRlPokBPgQTAQIAKAUCTQdk
3wIbAwUJABpeAAYLCQgHAwIGFQgCCQoLBBYCAwECHgECF4AACgkQsRjHPCDakqyp
Zwf/Zea4LNrVDx4qOx6kMUkoHkcum41r6RxNRV0mu4OVx0RL1meALBOYkDBEpS7B
oFp2ZE2kS6HNesad7HwzCcqnDr9xStf/ctvsPnaiccv6cffuxEVZl4R2SgjxGnwM
zboJHUC/K0kynVBPTBuEmq+LnHukqPbBNdHTKUQ+jKsig5PTNEM/QCz54v84vfsV
9FIlZjSQnaYSrLO7Sw78Baue/FaBst33N89JepIqhoebaXNNxeo3BRytPT6HLGkO
JjTxOq5Mxr9+4g4f0nPV+7ad4Xs446DctCOmmemMIZwtR0bxm/U7QAEGlbVNSDH0
WsmvbNTVSusae9Lo5D+e4Vy0e7QvQ2hyaXN0aWFuIEJhcnRoZWwgPGNocmlzdGlh
bi5iYXJ0aGVsQG9ubGluZS5kZT6JAT4EEwECACgFAk0IwcwCGwMFCQAaXgAGCwkI
BwMCBhUIAgkKCwQWAgMBAh4BAheAAAoJELEYxzwg2pKsiPUH/13nEQxzsrVT7rTf
ndaIVqaWV0Fqdb+uOxLOjgc6yDK/+L5JfEjbTwR/MJzW9HgCg/t8+ozOVzhzv25Z
w9RKgWSqoNnzBSLN1FYzHhZC+HVbj3AxZwDCBdsmWh1ujbTjvS9Vn/CjT/llxzD0
YOAk8SlYS1fxxRoTErd6Dmc4NaRn/+IZ+sCcVBC34jDDpW+YIPqrEKXiPkgvhrnc
p6KmkMj8FB7kD6EHtHPc0zengLCNWizWAaVQDLa/4MMwi/I13xEnmuJLmXmQvkDM
ibCGrXhjdb99vNgRihcVKsZcal21VZqRno1dU33pdXyxWmbxqczCLoRpNQkZXrtj
msE0Cmm5AQ0ETQdk3wEIAN1gy189PtaJZTuCuQD1ThukQIKIbITEFxaBYZVtxj9G
u4fsLHaysnOCISjbe3lX/WZNurYOI+17zLOqzz2FNlbhjOzmNyZsaV3Giv9mDO5B
h8hHX+BKuIIaO9FAdVYqEEkGdk879JXfA1rqC7Lj9fg/G0bP8ir9zH74w3hZG+6P
pPQRzvrb3pTVsxTOS3LVz+PQmOJ0Xsz/aYE0lTFB90W/ubAE/8I5/7EP1Yk8/X1p
XtZspMdkYYsaKtfcFigdgK1F/5dI+gEXpf//ynTW1Fo7ZtxTOAzw0OWZPd6Js+bX
l8cYBwAhFQjE7eQH7jNlC3T6ff2Eufzb/GnZBrhkM+UAEQEAAYkBJQQYAQIADwUC
TQdk3wIbDAUJABpeAAAKCRCxGMc8INqSrBRiB/9Ms8SiJdM8FPACBICu9RaJI716
ilkPqAUk7+8Hvg9yj4AObagVN0JezI6Z6cKIe+bkdf0Soi+W9Ef1qEOTCI+eDDz7
AlCxCxAC7c0hzHvh2F6Rl+/JX5MPfspDSjM+3s5io+UlLRpJV4Ddk/gjdDkIfZFp
5yQLfaO5KbMy4TOykaQQeqyvodYQDKx0oLzRsuBwxcrVlKB9agjg0ixZdIUiwaNa
oXWdt1XAxDZCFdBbcAngW9eSCFNkQKgMMAxXifWuIrVR2B0yL3W8G08xDpPG+C08
YEwWAEyR9KwmmrcgWGz+KbleBvttibJ9iUFzOHuEd9UxunbNPr2QYP9a4CB1
=Fvj5
-----END PGP PUBLIC KEY BLOCK-----
```

3.5 Importieren eines öffentlichen Schlüssels

Mit der Option –import kann ein öffentlicher Schlüssel zu Ihrem Schlüsselbund hinzugefügt werden:

```
$ ~/Documents/secret $ gpg --import fritz.asc
gpg: key A005AFE2: public key "fritz maier <fritz@maier.de>" imported
gpg: Total number processed: 1
```

```
gpg:               imported: 1
```

Die Ausgabe der Keys:

```
$ ~/Documents/secret $ gpg --list-key
/home/christian/.gnupg/pubring.gpg
---------------------------------
pub   2048R/20DA92AC 2010-12-14 [expires: 2011-01-03]
uid                  Christian Barthel <christian@barthel-connect.de>
sub   2048R/1423BA72 2010-12-14 [expires: 2011-01-03]

pub   1024D/A005AFE2 2007-03-13
uid                  fritz maier <fritz@maier.de>
sub   2048g/5EA3FF59 2007-03-13
```

Wird ein Schlüssel importiert, so sollte dieser auch auf die Authentizität überprüft werden. Ein Schlüssel wird darduch authentifiziert, dass Sie den Fingerabdruck des Schlüssels überprüfen und dann den Schlüssel unterschreiben, um seine Gültigkeit zu bestätigen. Der Fingerabrduck einens Schlüssels wird mit der Option –fingerprint angezeigt. Um den Schlüssel zu bestätigen, müssen Sie ihn editieren.

```
$ ~/Documents/secret $ gpg --edit-key fritz@maier.de
gpg (GnuPG) 1.4.10; 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.


pub  1024D/A005AFE2  created: 2007-03-13  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  2048g/5EA3FF59  created: 2007-03-13  expires: never       usage: E
[ unknown] (1). fritz maier <fritz@maier.de>

Command> fpr
pub   1024D/A005AFE2 2007-03-13 fritz maier <fritz@maier.de>
 Primary key fingerprint: E9B3 4D8E 84E7 B4FB B1FA  EC28 BC3E 00ED A005 AFE2
```

Sobald Sie den fingerprint angezeigt haben, müssen Sie den Eigentümer kontaktieren und mit ihm über den Schlüssel sprechen. Ist die Authentizität gewährleistet, so kann der Schlüssel unterschrieben werden.

```
Command> sign

pub  1024D/A005AFE2  created: 2007-03-13  expires: never       usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: E9B3 4D8E 84E7 B4FB B1FA  EC28 BC3E 00ED A005 AFE2
```

```
     fritz maier <fritz@maier.de>

Are you sure that you want to sign this key with your
key "Christian Barthel <christian.barthel@online.de>" (20DA92AC)

Really sign? (y/N)

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14
```

Um die Unterschrift dann wiederum anzeigen zu können, kann die Option check verwendet werden:

```
Command> check
uid  fritz maier <fritz@maier.de>
sig!3        A005AFE2 2007-03-13  [self-signature]
sig!         20DA92AC 2010-12-17  Christian Barthel <christian.barthel@online.de
```

3.6 Ver- und Entschlüsseln

Ein Dokument kann nur mit dem zum öffentlichen Schlüssel passenden Schlüssel auch wieder entschlüsselt werden. Eine Nachricht an fritz verschlüsseln Sie mit dem öffentlichem Schlüssel von Arnolrd. Und dieser kann die geheime nachricht nur mit seinem privatem Key wiederum dechiffrieren. Um ein Dokument für eine fremde Person nun verschlüsseln zu können, müssen Sie zunächst den öffentlichen Key dieser Person haben. Mittels –encrypt wird eine Datei verschlüsselt. Der –output kann entwedern auf den Standardstrom geleitet werden, oder aber in eine Datei.

Im folgenden wird für fritz maier ein Dokument mit seinem öffentlichem Schlüssel verschlüsselt. Nun kann das Dokument nur noch von seinem privatem Key entschlüsselt werden.

```
$ ~/Documents/secret $ gpg --output doc.gpg --encrypt --recipient fritz@maier.de dokument.
```

Die Datei wird ebenfalls wieder im Binärformat abgespeichert. Um dies zu verhindern, wird wieder –armor angehängt.

```
$ ~/Documents/secret $ gpg --output  doc.pgp --armor --encrypt --recipient fritz@maier.de
$ ~/Documents/secret $ cat doc.pgp
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

hQIOA6fnig5eo/9ZEAf7Bgd509bRaLezui1LgruBEB7Sr2pcG7DtGVM+5+d1Aeyh
kqv1q15NQ93am1mKHDoemICDjK9G6aEw5H/jJVd8FS1Jtx4A/cyJj6ZMnZ1ltNRS
```

```
5l0DPu0o58wi18d87wnuo0BzYqnyixfOrJNJ9PB777nunx07J/IEvVRtdQKTlVGh
u2ZaretLFKwelkO3aqrPopwoII+F7qBqMmXBo4o2JWW60rYUuNengx+oH3tvTlZJ
5wV9YkytqmsHkmJVFJGebiUgZ9j8n8l1hX1C7KVhGomg5tudTkk+Zjtjw03AJKnZ
P18d8a1l3odbTcC2D3DWMM/tvkjAmDc7aV9THRO6eQf/aw6pH7J9JHw4yTJHdmzF
NEziOmaaaZdwpLKizLdhPG3T4EG+9eZ+BgalDzFwWirRwXJUNBJs/Fdl/SqDSk+g
oarfnvTOr6WWdgvDLFON2na4pHclu/sB8b1LlgM7N/MnSwmmW2u14X127VwFPgU6
bYwYFyl63bOqE3so/B4ofnUwpMRNCKM+EtcaQOkTh/tMw/blqqMG6BcKVIuGLVuS
pdmlo8frKwwDWXOV+SO7WgHB361Nkc0BNMGtUcXOz+qOENUbczWT1Re5i1FAeoZu
r3YhWBmkAHIIRkqc0+RzPtMqjaNTn+xNanQSMwceUD5zpg42jG6O+R2aE55LDPFf
HNJSAcHQHR+9b004G2zhDm8AnKMmamtLWbukvTHFrxHUMGFyg0v7om2goaaOEJTo
SbmQiTqTcPh6zwKObteAiXL26QNRiFPaY9xCRDFwtMS2ulRVVw==
=mF3f
-----END PGP MESSAGE-----
```

Will man ein Dokument entschlüsseln, so wird die Option –decrypt benutzt.

```
$ ~/Documents/secret $ gpg --output doc --decrypt doc.pgp

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 1423BA72, created 2010-12-14 (main key ID 20DA92AC)

gpg: encrypted with 2048-bit RSA key, ID 1423BA72, created 2010-12-14
      "Christian Barthel <christian.barthel@online.de>"
$ ~/Documents/secret $ cat do
cat: do: No such file or directory
$ ~/Documents/secret $ cat doc
Hallo Welt
```

Des weiteren kann mit GnuPG auch das symmetrische Verschlüsselungsverfahren benutzt werden. Sie sollten allerdings dann auf ein entsprechend langes und komplexes Passwort setzen:

```
$ ~/Documents/secret $ gpg --output doc.asc --armor --symmetric dokument.txt
gpg: problem with the agent - disabling agent use
Enter passphrase:
File ‘doc.asc’ exists. Overwrite? (y/N) y
$ ~/Documents/secret $ cat doc.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

jA0EAwMCfHuSUrMGIy9gyS2uDtiMSjTBroN+DN66iQsm83y3V2KP5l6qxnz5uJ0+
Tu0hP4nnPMaUBKOf/jI=
=6+S+
```

Verwendungszwecke:

- Verschlüsseln von Daten, die nicht weitergegeben werden müssen
- die Passwortübergabe entfällt also
- z. B. für eine verschlüsselte E-Mail-CD
- Festplatte,
- Archive

3.7 Digitale Signaturen

Eine digitale Unterschrift/Signatur ist am ehesten mit einem Siegel zu vergleichen, wie Sie früher im Briefverkehr vorkamen. Das Siegel bestätigt die Integrität eines Dokumentes. Es gewährleistet, dass der Umschlag von niemanden geöffnet wurde. Eine nachträgliche Manipulation würde sich sofort feststellen lassen. Im OpenSource-Umfeld werden z. B. Quellcodes so signiert, dass man sehr leicht feststellen kann, ob der Quelltext naträglich verändert wurde.

Bei der Erzeugung und Prüfung von Unterschriften wird das Public-Key Verfahren entgegengesetzt angewandt. Die Unterschrift wird mit dem geheimen Schlüssel des Unterzeichnenden erzeugt und dann jeweils mit dem entsprechendem öffentlichen Schlüssel wieder dechiffriert. fritz würde so z. B. mit dem öffentlichen Schlüssel die Signatur prüfen, die Sie mit dem privatem Schlüssel verschlüsselt haben.

Zum Erzeugen wird die Optionsfolge –sign verwendet.

```
christian@notebook02 ~/Documents/secret $ gpg -a --output doc.sig --sign dokument.txt

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14

File ‘doc.sig’ exists. Overwrite? (y/N) y
christian@notebook02 ~/Documents/secret $ cat doc.sig
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

owEBTQGy/pANAwACAbEYxzwg2pKsAawdYgxkb2t1bWVudC50eHRNC94eSGFsbG8g
V2VsdAqJARwEAAECAAYFAk0L3h4ACgkQsRjHPCDakqypHwf/dMLB/oXyjscGTlCw
YvTKDgALgMmBGuHRAaDHS9JHbk+uG0BfjwuK8kK147j3tQ/+oZ+Z2Muzzb7Ntdiu
ptO3K+1PcopqgGKAlfTQrmSbb8mVdUzBjHvFk7MfSf2LkxCybPZO+hPaN5jGNA0B
eAlXDj0kQS9prcc1w6k+VBFz1+fVF3ClqHKJa07fmFAMgcgHav85o5j+1DcmkkjC
J6UBU9GfUFJbxWXqhIn6dWa/UHdr8aWKOPjL2quX8nzSlp1MbijJ9ijtuuCiDK81
cIvQ4M3u3SJYkmxWN5/4mXrpH0xRRgIcIQqmRtPkYikI6PPaofKtD8ywySu8BYHy
QXjSTQ==
=HAGz
-----END PGP MESSAGE-----
```

Lässt man den Parameter -a weg, so wird die Ausgabe im Binärformat gespeichert.

```
christian@notebook02 ~/Documents/secret $ gpg --output doc2 --decrypt doc.sig
gpg: Signature made Fri 17 Dec 2010 11:03:10 PM CET using RSA key ID 20DA92AC
gpg: Good signature from "Christian Barthel <christian.barthel@online.de>"
gpg:                 aka "Christian Barthel <christian@barthel-connect.de>"
```

Sie können die Signatur eines Dokuments prüfen und das unterschriebene Originaldokument gleichzeitig Entnehmen, oder sie prüfen die Signatur ohne eine entnahme des Inhalts.

```
christian@notebook02 ~/Documents/secret $ gpg --verify doc.sig
gpg: Signature made Fri 17 Dec 2010 11:03:10 PM CET using RSA key ID 20DA92AC
gpg: Good signature from "Christian Barthel <christian.barthel@online.de>"
gpg:                 aka "Christian Barthel <christian@barthel-connect.de>"
```

3.7.1 Klartextsignatur

In manchen Fällen ist es nicht erwünscht, das Dokument beim Unterschreiben zu komprimieren. Hierfür wurde die Option –clearsign entwickelt. Das Format wird dadurch auf ASCII umgestellt und umhüllt ähnlich wie ein Briefumschlag das Dokument, das nicht verändert werden soll.

```
christian@notebook02 ~/Documents/secret $ cat dokument.txt
Hallo Welt
christian@notebook02 ~/Documents/secret $ gpg --output test --clearsign dokument.txt

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14

christian@notebook02 ~/Documents/secret $ cat test
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hallo Welt
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)

iQEcBAEBAgAGBQJNC+AmAAoJELEYxzwg2pKsIXcH/0+Y6egSea4NSbHYelZVn2Sy
h++T0TaxzqNQU+iSuN4VVRmiszFOLsBBRxZQ7fvnO/sdahI/KLeCl1PmcNnglNSn
KVj6ln2WIpgp9EsLOW3r6F8WPRnYDRTsCqzF4vzwKo69OiyeqUCnVojCW9ygKuGz
qj3Lv/VGHOOZKPHgQMJJ+iyamKV3w+3sNhQ0UlkoEJmtBPz7WFdO7iL+A0D/bCVY
```

```
4wk15AFZ7CiHzhHhDLZjUNuovmO0QeXFCSJishSJIHM3fL4d4n2jYJitphLrAYyK
1412uTvvSAgfYwOoSLvS+EJW/VMWKNrm+JDjNbyMMi+rc4KdPr9cQIn4A5QB1VA=
=+c/e
-----END PGP SIGNATURE-----
```

Oder eine verschlüsselte Nachricht mit angehängter Signatur:

```
christian@notebook02 ~/Documents/secret $ cat doc.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

jA0EAwMCfHuSUrMGIy9gyS2uDtiMSjTBroN+DN66iQsm83y3V2KP5l6qxnz5uJ0+
Tu0hP4nnPMaUBKOf/jI=
=6+S+
-----END PGP MESSAGE-----
christian@notebook02 ~/Documents/secret $ gpg --output doc.asc.sig --clearsign doc.asc

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14

christian@notebook02 ~/Documents/secret $ cat doc.asc.sig
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- -----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

jA0EAwMCfHuSUrMGIy9gyS2uDtiMSjTBroN+DN66iQsm83y3V2KP5l6qxnz5uJ0+
Tu0hP4nnPMaUBKOf/jI=
=6+S+
- -----END PGP MESSAGE-----
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)

iQEcBAEBAgAGBQJNC+D/AAoJELEYxzwg2pKstQAH/js5hoQOrNLbXhsRUZBt4Nq7
zIUWh10iq8OspgO2Vm5DESv8zYyMzQJVoruneeFnVY6SnKempfJGOv/W/Lyg1eLP
mVNApJw8/9aQ+JOCWoyElDEXLjWqRhEkN3np5xCsgrKGHZ1cTL2IkwUbvIMMRXkK
hhEnLkm7ym8FvRAJsSAjbjJdQjH3u7FtLbLYskCWRfMaJKfLaEiNTJXBqC6GOzYy
dIPGl4yZBpBFCJdiCneV+1h9S+UDf3o+OtAc9uEGwag4TGX51PRdSc/FRDgeHfEM
5hgHLpLsfN7Eod564CeiUykLNYq/WwADSNtorSZL7t1tRFYGWn1aUfP76QZ65vo=
=QCDy
-----END PGP SIGNATURE-----
```

3.7.2 Getrennte Signatur

Nachteilhaft bei signierten Dokumenten ist es, dass der Empfänger das Originaldokument aus der unterschriebenen Version erst wieder erstellen muss bzw. bei einem im Klartext unterschriebenen Dokument dieses gegebenfalls noch editierten muss. Als dritte Möglichkeit kann man die Datei mit abgetrennter Unterschrift signieren. Dazu wird die Option –detach-sig verwendet.

```
christian@notebook02 ~/Documents/secret/abgetrennt $ cat neu
TESSSST
christian@notebook02 ~/Documents/secret/abgetrennt $ gpg -a --output doc.sig --detach-sig

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14

File ‘doc.sig’ exists. Overwrite? (y/N) y
christian@notebook02 ~/Documents/secret/abgetrennt $ cat neu
TESSSST
christian@notebook02 ~/Documents/secret/abgetrennt $ ls -alh
total 16K
drwxr-xr-x 2 christian christian 4.0K 2010-12-17 23:17 .
drwxr-xr-x 3 christian christian 4.0K 2010-12-17 23:17 ..
-rw-r--r-- 1 christian christian  490 2010-12-17 23:17 doc.sig
-rw-r--r-- 1 christian christian    8 2010-12-17 23:17 neu
christian@notebook02 ~/Documents/secret/abgetrennt $ cat doc.sig
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)

iQEcBAABAgAGBQJNC+GNAAoJELEYxzwg2pKsQ9sH/257GMGa6scMzq/+X5/o7GwI
hLUnMIub1Q3IE+WBmwTGtzXwFBKstiBql1tnyckwBBZ6ZVgB9xNQXahYXnKP6G5O
A6wxt1NofMGR+BAT/wBqfKNtgS1uIGHOeuGz2VLMjnQl2nqAOG2GOmGFCYhLe6X1
4uEKlG9gHdTxricfTF4YwOdOBn7TvsN+67ZsCds8S7tJG5Lc7XZMeI4gPYQLdPSQ
OyAtYd/anwyBPYbz+smGaZiZTWTACuzrHlxU4YEhS2T8AXIz5LxmTC8PIYPIBmJS
Ct6HsxXTrE633uKCKIL6ymPOmsTUQWzLo13WGJ474fAOrUiboL3Nz2yYSedI4dY=
=32Zt
-----END PGP SIGNATURE-----
```

Die Überprüfung:

```
christian@notebook02 ~/Documents/secret/abgetrennt $ gpg --verify doc.sig neu
gpg: Signature made Fri 17 Dec 2010 11:17:49 PM CET using RSA key ID 20DA92AC
gpg: Good signature from "Christian Barthel <christian.barthel@online.de>"
gpg:                 aka "Christian Barthel <christian@barthel-connect.de>"
```

4 Schlüsselmanagement

Schlüsselfälschungen sind in Public-Key-Verfahren sehr leicht möglich. Beispielsweise kann sich ein Angreifer die Schlüsselbunde eines Benutzers manipulieren oder er erzeugt Schlüssel mit einer vorgetäuschen Identität.
Beispiel: Peter möchte die Nachrichten unbemerkt mitlesen, die Alice an Wolfgang sendet. Zuerst kann Peter ein Schlüsselpaar erzeugen, mit einer falschen Identität. Dann ersetzt Peter Alice Kopie von Wolfgangs öffentlichem Schlüssel. Anschließend fängt Peter alle Nachrichten ab, die Alice an Wolfgang sendet. Diese Nachrichten kann Peter nun entschlüsseln. Damit Wolfgang jedoch keinen Verdacht schöpft verschlüsselt nun Peter die Nachrichten, die er zuvor von Alice geklaut hat, mit dem öffentlichem Schlüssel von Wolfgang.

4.1 Verwaltung des eigenen Schlüsselpaars

Ein Schlüsselpaar besteht aus einem öffentlichem und einem privatem Schlüssel. Um die Schlüssel einer realen Person zuzuordnen, können ein oder mehrere Benutzer-IDs den Schlüssel hinzugefügt werden.

```
barthelc@notebook02 ~ $ gpg --edit-key christian@barthel-connect.de
gpg (GnuPG) 1.4.10; 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.

Secret key is available.

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>
```

Die erste Spalte gibt den Typ des Schlüssels an.

pub identifiziert den öffentlichen Hauptschlüssel

sub identifiziert einen untergeordneten öffentlichen Schlüssel (Subkey)

In der zweiten Spalte wird die Länge, der Typ und die ID des Schlüssels angezeigt, z. B. 2048R/20DA92AC

Länge in Bits, in diesem Beispiel handelt es sich um 2048 Bit. Maximal sind bis zu 4096 erlaubt, häufig werden auch noch 1048 benutzt.

Typ, hier steht das **R** für RSA. Weitere Möglichkeinen: **D** für DSA, **g** für einen nur zur Verschlüsselung geeigneten ElGamal-Schlüssel und **G** für einen ElGamal-Schlüssel, der sowohl verschlüsseln und unterschreiben kann.

ID, die den Key eindeutig identifiziert. In diesem Beispiel **20DA92AC**.

Das Erstell- und Verfallsdatum folgt in Spalte 3 und 4. Die Benutzer-IDs werden nach den Schlüsseln angegeben.

Um noch weitere Informationen über Schlüssel abrufen zu können, kann der Befehl **toogle** verwendet werden. Toggle schaltet zwischen den öffentlichen und den geheimen Komponenten eines Schlüsselpaares um, falls beides zur Verfügung steht.

```
Command> toggle

sec  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03
ssb  2048R/1423BA72  created: 2010-12-14  expires: never
(1)  Christian Barthel <christian@barthel-connect.de>
(2)  Christian Barthel <christian.barthel@online.de>
```

sec identifiziert den gehimen Hauptschlüssel, das Schlüsselwort **ssb** identifiziert geheime Subkeys.

4.2 Schlüsselintegrität

Sobald Sie den öffentlichen Teil eines Schlüsselpaares weitergeben, geben Sie den Hauptschlüssel und Ihre Subkeys, sowie Ihre Benutzer-ID weiter. Diese Informationen sollten nicht ungeschützt weitergegeben werden, da es ansonsten für einen potentiellen Angreifer möglich ist, die Schlüssel zu verfälschen. Z. B. könnte Peter, der Angreifer, Ihre Schlüssel so manipulieren, dass die E-Mails an seine Mail-Adresse gehen, und nicht in Ihrem Posteingang erscheinen. Diese Probleme werden unter anderem durch digitale Signaturen gelöst. Indem man den öffentlichen Schlüssel sowie die Benutzer-IDs mit seinem geheimen Schlüssel unterzeichnet, lassen sich Verfälschungen daran leicht feststellen. Natürlich kann nun jeder mit dem öffentlichen Schlüssel diesese Zertifikat prüfen und ansehen.

Im folgenden sehen Sie für den Benutzer Christian zwei Benutzer-IDs hinterlegt. Die Unterschriften auf den Benutzer-IDs können mit dem Befehl **check** im Schlüsseleditor geprüft werden.

```
barthelc@notebook02 ~ $ gpg --edit-key christian@barthel-connect.de
gpg (GnuPG) 1.4.10; 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.

Secret key is available.

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
```

```
                    trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>

Command> check
uid  Christian Barthel <christian.barthel@online.de>
sig!3        20DA92AC 2010-12-15  [self-signature]
uid  Christian Barthel <christian@barthel-connect.de>
sig!3        20DA92AC 2010-12-14  [self-signature]
```

Für jede Unterschrift wird der primäre Schlüssel verwendet. Die Eigenbeglaubigungen auf den Subkeys sind in dem öffentlichen Schlüssel enthalten, doch werden Sie vom Schlüsseleditor nicht angezeigt.

4.3 Editieren von Schlüsseln

Zu einem Schlüsselpaar können später weitere Subkeys und Benutzer-IDs hinzugefügt werden. Eine Benutzer-ID wird durch Verwendung des Befehls **adduid** hinzugefügt. Dabei werden die gleichen Angaben benötigt, wie Sie auch bei der Key Erzeugung abgefragt werden. Ein Subkey wird durch Verwendung des Befehls **addkey** hinzugefügt. Wird ein Subkey oder eine neue Benutzer-ID erzeugt, so werden diese mit dem geheimen Schlüssel eigenbeglaugibt. Deshalb wird auch die Passpharse benötigt, wenn der Schlüssel erzeugt wird.

```
Command> addkey
Key is protected.

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14

Please select what kind of key you want:
   (3) DSA (sign only)
   (4) RSA (sign only)
   (5) Elgamal (encrypt only)
   (6) RSA (encrypt only)
Your selection? 3
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
```

```
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y
Really create? (y/N) y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++++++++++++++++.+++++..+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
sub  1024D/0DCAB980  created: 2010-12-18  expires: never       usage: S
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>


Command> list

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
sub  1024D/0DCAB980  created: 2010-12-18  expires: never       usage: S
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>
```

Z. B. kann es praktisch sein, zusätzliche Benutzer-IDs für Arbeit, Freizeit, politische Tätigkeiten und ähnliches anzulegen.

Da die zu dem primären öffentlichen Schlüssel gehörigen Benutzer-IDs von den Leuten, mit denen kommuniziert wird, authentisiert werden, erfordert eine Änderung des primären Schlüssels eine nochmalige Bestitigung. Kommunizieren Sie mit vielen Leuten, so kann dies schwierig und zeitaufwendig sein. Anererseits sollte man von Zeit zu Zeit, die Subkeys für die Verschlüsselung ändern. Durch das Ändern der Schlüssel erreicht man, dass in der Zukunft zu verschlüsselnde Daten auch sicher sind.

Subkeys und Benutzer-IDs können ebenfalls gelöscht werden. Dazu wählt man zunächst mittels **key** oder **uid** den gewünschten Eintrag aus (Ganzzahl). Danach kann mittels **deluid** oder **delkey** der selektrierte Wert entfernt werden.

```
Command> key 2

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
```

```
sub* 1024D/0DCAB980  created: 2010-12-18  expires: never       usage: S
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>

Command> delkey
Do you really want to delete this key? (y/N) y

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>
```

4.4 Widerrufen von Schlüsseln

Sie sollten allerdings nicht unbedingt aktualisierte Versionen Ihres Schlüssels verteilen. Die Kommunikationspartner müssent dafür zunächst Ihren aktuellen Key enfernen und den neuen öffentlichen Key wieder importieren. Besser ist es, Subkeys zu widerrufen.

```
Command> key 2

pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
sub* 1024D/9D0A2E67  created: 2010-12-18  expires: never       usage: S
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>

Command> revkey
Do you really want to revoke this subkey? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
Your decision? 3
Enter an optional description; end it with an empty line:
> Key wurde aktualisiert
>
Reason for revocation: Key is no longer used
Key wurde aktualisiert
Is this okay? (y/N) y

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
```

```
2048-bit RSA key, ID 20DA92AC, created 2010-12-14


pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
This key was revoked on 2010-12-18 by RSA key 20DA92AC Christian Barthel <christian.barthe
sub  1024D/9D0A2E67  created: 2010-12-18  revoked: 2010-12-18  usage: S
[ultimate] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>
```

Zum Widerrufen einer Benutzer-ID muss anders verfahren werden. In der Theorie sollte sich eine Benutzer-ID niemals ändern. In der Praxis können sich allerdings Elemente wie E-Mail, Bemerkung oder Beschreibung ändern. Eine Unterschrift wird mittels **revsig** widerrufen.

```
Command> revsig
You have signed these user IDs on key 20DA92AC:
     Christian Barthel <christian.barthel@online.de>
   signed by your key 20DA92AC on 2010-12-15
     Christian Barthel <christian@barthel-connect.de>
   signed by your key 20DA92AC on 2010-12-14

user ID: "Christian Barthel <christian.barthel@online.de>"
signed by your key 20DA92AC on 2010-12-15
Create a revocation certificate for this signature? (y/N) y
user ID: "Christian Barthel <christian@barthel-connect.de>"
signed by your key 20DA92AC on 2010-12-14
Create a revocation certificate for this signature? (y/N) n
Really create the revocation certificates? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  4 = User ID is no longer valid
  Q = Cancel
Your decision?  4
Enter an optional description; end it with an empty line:
Is this okay? (y/N) y

You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14


pub  2048R/20DA92AC  created: 2010-12-14  expires: 2011-01-03  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
This key was revoked on 2010-12-18 by RSA key 20DA92AC Christian Barthel <christian.barthe
sub  1024D/9D0A2E67  created: 2010-12-18  revoked: 2010-12-18  usage: S
```

```
[ revoked] (1). Christian Barthel <christian.barthel@online.de>
[ultimate] (2)  Christian Barthel <christian@barthel-connect.de>
```

Der Widerruf kann mittels **check** überpfüt werden.

```
Command> check
uid  Christian Barthel <christian.barthel@online.de>
rev!         20DA92AC 2010-12-18  [revocation]
sig!3        20DA92AC 2010-12-15  [self-signature]
uid  Christian Barthel <christian@barthel-connect.de>
sig!3        20DA92AC 2010-12-14  [self-signature]
```

4.5 Aktualisieren des Schlüssel-Verfallsdatums

Die Aktualisierung erfolgt mittels **expire**. Ist kein Schlüssel ausgewählt (mit key x), dann wird der primäre Schlüssel aktualisiert, ansonsten der jeweilige Subkey.

```
Command> expire
Changing expiration time for the primary key.
gpg: WARNING: no user ID has been marked as primary.  This command may
              cause a different user ID to become the assumed primary.
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 2
Key expires at Mon 20 Dec 2010 11:45:03 AM CET
Is this correct? (y/N) y
You need a passphrase to unlock the secret key for
user: "Christian Barthel <christian.barthel@online.de>"
2048-bit RSA key, ID 20DA92AC, created 2010-12-14


pub  2048R/20DA92AC  created: 2010-12-14  expires: 2010-12-20  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
This key was revoked on 2010-12-18 by RSA key 20DA92AC Christian Barthel <christian.barthe
sub  1024D/9D0A2E67  created: 2010-12-18  revoked: 2010-12-18  usage: S
[ultimate] (1)  Christian Barthel <christian.barthel@online.de>
[ultimate] (2). Christian Barthel <christian@barthel-connect.de>

Command>
```

```
pub  2048R/20DA92AC  created: 2010-12-14  expires: 2010-12-20  usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/1423BA72  created: 2010-12-14  expires: 2011-01-03  usage: E
This key was revoked on 2010-12-18 by RSA key 20DA92AC Christian Barthel <christian.barthe
sub  1024D/9D0A2E67  created: 2010-12-18  revoked: 2010-12-18  usage: S
[ultimate] (1)  Christian Barthel <christian.barthel@online.de>
[ultimate] (2). Christian Barthel <christian@barthel-connect.de>
```

4.6 Aktualisieren anderer Schlüssel

Der öffentliche Schlüssel eines Kommunikationspartners wird dadurch authentisiert, dass Sie persönlich den Fingerabdruck seines Schlüssels prüfen und dann seinen öffentlichen Schlüssel mit Ihrem geheimen Schlüssel unterschreiben. Sobald jemand den Schlüssel manipuliert, werden Sie es sofort erkennen. Außerdem kann man sich durch den Vergleich des Fingerabdrucks mit dem Eigentümer, dass es sich um den wirklichen öffentlichen Schlüssel handelt. Nachteilhaft ist jedoch, dass dieses Verfahren bei vielen Kommunikationspartnern sehr umständlich sein kann. Dieses Problem soll mit dem sogenannten Web of Trust angegangen werden. Es läuft dabei wie ein Bewertungsverfahren auf ebay ab. Angenommen Benutzer A hat den Schlüssel von Benutzer B unterschrieben und Benutzer B hat die Schlüssel von Benutzer C und Benutzer D unterschrieben. Benutzer A kanns omit auch Benutzer C und Benutzer D vertrauen, ohne das Sie persönlich prüfen muss, ob es sich um Fälschungen oder ähnliches handelt.

4.7 Vertrauen in den Eigentümer eines Schlüssels

Der Schlüssel von Benutzer B ist für Benutzer A gültig, da dieser den Schlüssel selbst unterschrieben hat. Aber vielleicht traut Benutzer A Benutzer B kein vernünftiges Authentisieren zu. In diesem Fall könnte Benutzer A also den Benutzern C und misstrauen, da nur die Unterschrift von Benutzer B vorhanden ist. Das Web of Trust versucht diesen Missstand zu beseitigen, indem es jedem öffentlichem Schlüssel in Ihrem Schlüsselbund eine Angabe zuordnet, inwieweit Sie einem Eigentümer eines Schlüssels vertrauen können. Es gibt 4 verschiedene Vertrauensstufen:

1. **Unbekannt**: Es ist nicht über den Eigentümer bekannt. Alle Schlüssel in Ihrem öffentlichem Schlüsselbund, die Ihnen nicht gehören, fallen zunächst unter diese Stufe
2. **Kein Vertrauen**: Der Eigentümer ist dafür bekannt, andere Schlüssel nicht korrekt zu unterschreiben.
3. **Teilweises Vertrauen**: Der Eigentümer versteht die Implikationen des Unterschreibens von Schlüsseln und authentisiert Schlüssel richtig, bevor

er sie unterschribt.

4. **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.

Vertrauensinformationen sind private Informationen und werden nicht mit dem Schlüssel verpackt. Sie müssen extra exportiert werden.

Der Befehl, um Ihr Vertrauen gegenüber einen Schlüsseleigentümer zu konfigurieren, lautet **trust**.

```
barthelc@notebook02 ~/Documents/secret $ gpg --import fritz.asc
gpg: key A005AFE2: public key "fritz maier <fritz@maier.de>" imported
gpg: Total number processed: 1
gpg:               imported: 1
barthelc@notebook02 ~/Documents/secret $ gpg --list-key
/home/barthelc/.gnupg/pubring.gpg
---------------------------------
pub   2048R/20DA92AC 2010-12-14 [expires: 2010-12-20]
uid                  Christian Barthel <christian@barthel-connect.de>
uid                  Christian Barthel <christian.barthel@online.de>
sub   2048R/1423BA72 2010-12-14 [expires: 2011-01-03]

pub   1024R/2DF1033C 2010-12-14 [expired: 2010-12-16]
uid                  Heinrich Heine (Der Dichter) (test) <heinrichh@duesseldorf.de>

pub   1024R/15D1C02C 2010-12-15 [expired: 2010-12-16]
uid                  Test MeinTest (bla) <test@test.de>

pub   1024D/A005AFE2 2007-03-13
uid                  fritz maier <fritz@maier.de>
sub   2048g/5EA3FF59 2007-03-13
```

```
barthelc@notebook02 ~/Documents/secret $ gpg --edit-key fritz@maier.de
gpg (GnuPG) 1.4.10; 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.


pub  1024D/A005AFE2  created: 2007-03-13  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  2048g/5EA3FF59  created: 2007-03-13  expires: never       usage: E
[ unknown] (1). fritz maier <fritz@maier.de>

Command> trust
pub  1024D/A005AFE2  created: 2007-03-13  expires: never       usage: SC
```

```
                     trust: unknown       validity: unknown
sub  2048g/5EA3FF59  created: 2007-03-13  expires: never       usage: E
[ unknown] (1). fritz maier <fritz@maier.de>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu


Your decision? 4

pub  1024D/A005AFE2  created: 2007-03-13  expires: never       usage: SC
                     trust: full          validity: unknown
sub  2048g/5EA3FF59  created: 2007-03-13  expires: never       usage: E
[ unknown] (1). fritz maier <fritz@maier.de>
Please note that the shown key validity is not necessarily correct
unless you restart the program.
```

Das Vertrauen in einem Schlüssel wird rechts neben dem Schlüssel angezeigt. An erster Stelle wird das Vertrauen in den Eigentümer, dannach in den Schlüssel aufgelistet.

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

4.8 Authentisieren von Schlüsseln im Web of Trust

Das Web of Trust ist ein flexibleres und komfortableres Verfahren zur Authentisierung eines Schlüssels. Wurde früher nur ein Schlüssel als gültig betrachtet, wenn er von Ihnen persönlich unterzeichnet wurde, so wird jetzt auch ein Schlüssel K als gültig betrachtet, wenn dieser 2 Bedingungen erfüllt:

- Schlüssel K ist von genügend gültigen Schlüsseln unterschrieben, d. h.
  - von Ihnen persönlich
  - von einem Schlüssel vollen Vertrauens oder

  - von 3 Schlüsseln teilweisen Vertrauens

- Der Pfad unterschriebener Schlüssel, der vom Schlüssel K zurück zu Ihrem eigenen Schlüssel führt, besteht aus max 5 Schritten

Diese Eigenschaften können nach jeweiligen Bedürfnissen angepasst werden.

4.9 Weitergabe von Schlüsseln

Im Idealfall wird ein Schlüssel durch eine persönliche Übergabe an den Kommunikationspartner getätigt. In der Praxis ist dies jedoch nicht immer möglich. Oft werden deshalb Schlüssel per E-Mail oder HTTP-Seite weitergegeben. E-Mail wird dann gerne benutzt, wenn es sich um nicht allzu viele Partner handelt. Haben Sie viele Partner, so können Sie den Schlüssel auf Iherer Homepage im Web publizieren. Natürlich müssen die Partner wissen, dass Sie Ihren Schlüssel hier finden können.

Als alternative gibt es Key-Server, die öffentliche Schlüssel sammeln und weitergeben. 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.

Um einen Schlüssel an einen Keyserver zu senden, kann die Option –send-keys verwendet werden. An welchen Keyserver die Schlüssel gesendet werden sollen, wird durch die Option –keyserver bestimmt. In ähnlicher Weise können auch Keys von einem Keyserver geholt werden: –recv-keys.

```
barthelc@notebook02 ~/Documents/secret $ gpg --keyserver wwwkeys.de.pgp.net --recv-key FB5
gpg: requesting key FB5797A9 from hkp server wwwkeys.de.pgp.net
gpg: key FB5797A9: public key Alice (Rechtsanwältin) <alice[at]cyb.org> imported
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   3  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: next trustdb check due at 2010-12-20
gpg: Total number processed: 1
gpg:               imported: 1

barthelc@notebook02 ~/Documents/secret $ gpg --list-key

pub   1024D/FB5797A9 2000-06-06
uid                  Alice (Rechtsanwältin) <alicev[at]cyb.org>
sub   1024g/C8B3998F 2000-06-06

alice$ gpg --keyserver wwwkeys.de.pgp.net --send-key blake[at]cyb.org
gpg: Senden an wwwkeys.de.pgp.net erfolgreich (status=200)
```

5 Tabellarischer Überblick

5.1 Grundlegende Befehle

Wirkung	Befehl
Key-Paar erzeugen	gpg –gen-key
Widerrufsurkunde erstellen	gpg –output revoke.asc –gen-revoke [Schlüsselid|Schlüsselname]
Keys anzeigen	gpg [–list-keys | -K | k]
Keys exportieren	gpg –export [-a] [–output dateiname.asc|.gpg] [Schlüsselid|Schlüsselname]
Keys importieren	gpg –import key.asc|*.gpg
Verschlüsseln	gpg –output AUSGABE.pgp –armor –encrypt –recipient [Public-Schlüsselid|Public-Schlüsselname] dokument.txt
Entschlüsseln	gpg –output ausgabe –decrypt verschlüsselt.pgp
Symmetrisch Verschlüsseln	gpg –output doc.asc –armor –symmetric dokument.txt
Signieren eines Dokuments	gpg -a –output doc.sig –sign dokument.txt
Signierung prüfen	gpg –output doc2 –decrypt doc.sig
Signierung im Klartext	gpg –clearsign doc
Signierung als Anhang	gpg –output doc.sig –detach-sig doc
Signierung als Anhang prüfen	gpg –verify doc.sig doc

5.2 Schlüssel Verwaltung

Wirkung	Befehl
Key-Details anzeigen	gpg –edit-key christian@barthel-connect.de
Geheime Schlüssel	command> toogle
Integrität prüfen	command> check
UID auswählen	command> uid 2
Key auswählen	command> key 3
UID/Key löschen	[deluid|delkey]
UID/Key hinzufügen	command> [addkey|adduid]
Widerrufszertifikat	command> revkey
Benutzer-IDs widerrufen	command> revsign
Aktualisieren des Verfallsdatums	command> expire

5.2.1 Vertrauensstufen Web of Trust

1. **Unbekannt**: Es ist nicht über den Eigentümer bekannt. Alle Schlüssel in Ihrem öffentlichem Schlüsselbund, die Ihnen nicht gehören, fallen zunächst unter diese Stufe

2. **Kein Vertrauen**: Der Eigentümer ist dafür bekannt, andere Schlüssel

nicht korrekt zu unterschreiben.

3. **Teilweises Vertrauen**: Der Eigentümer versteht die Implikationen des Unterschreibens von Schlüsseln und authentisiert Schlüssel richtig, bevor er sie unterschribt.

4. **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.

Wirkung	**Befehl**
`Key Vertrauenseinstellungen`	gpg –edit-key [keyid|keyname]
`Trust-Level ändern`	command> trust
`Integrität prüfen`	command> check

5.2.2 Keyserver

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