

# Gnu Privacy Guard

Phil Sutter

Computer Networking,  
Fachhochschule Furtwangen

19. Juni 2008

# Was ist GnuPG?

- Software zur Realisierung von Public-Key-Kryptographie
- Digitale Signaturen, Verschlüsselung von Nachrichten
- Implementiert OpenPGP-Standard (RFC2440, RFC4880)
- „Die freie Alternative“

# Public-Key-Kryptographie

- Zwei zueinander passende Keys
- Einer geheim ( $K_s$ ), einer öffentlich ( $K_p$ )
- $DEC(K_s, ENC(K_p, DATA))$  und  $DEC(K_p, ENC(K_s, DATA))$

# Wozu GnuPG?

- Hauptanwendungsbereich: Emails
- Von vielen Mail-Clients unterstützt
- Anerkennung von digitalen Signaturen problematisch

# Nebeneffekte

- KeyIDs und Fingerprints
- PGP-Keyserver
- Keysigning-Parties

# Web of Trust

- Signieren der Pubkeys anderer
- Dezentral: meist ist nur die Minderheit böse
- Zusätzlich: Trust-DB

# Key-Paar erstellen

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

- Interaktives Kommando
- Wichtig: Email-Adresse
- Keys werden im entspr. Keyring abgelegt

# Revocation-Key erstellen

```
gpg --gen-revoke <KeyID>
```

- Spezieller Signaturtyp
- Zur Invalidierung des eigenen Keys
- Output an sicherem Ort lagern

## Das „Pub“ in Pubkey

```
gpg --send-keys <KeyID>
```

- Sendet Pubkey an Keyserver aus *gpg.conf*
- Will man eventuell nicht
- Typische Falltürfunktion

# Schlüssel suchen

```
gpg --search-keys <EXPR>
```

- Interaktives Kommando
- Sucht auf Keyserver aus *gpg.conf*
- EXPR kann vieles sein

## Signieren, verschlüsseln oder beides

```
gpg --sign <FILE> oder gpg --detach-sign <FILE>  
gpg --encrypt <FILE>
```

- Kann auch kombiniert werden
- `--armor` ist nützlich
- Nutzt Default-Recipient aus *gpg.conf*  
(überschreiben mit `--recipient`)

## Verifizieren, entschlüsseln oder beides

```
gpg --verify <FILE>.pgp oder gpg --verify  
<FILE>.sig
```

```
gpg --output <FILE> --decrypt <FILE>.pgp
```

- **Verifikation:** Dateinamen müssen passen  
(alternativ Datei als zweiten Parameter angeben)
- `--decrypt` entfernt auch Inline-Signatur
- `--decrypt` verifiziert signierte Dateien automatisch

# Unter der Haube

- Keys an einem Keyring
- Mehrere UIDs pro Key
- Hybrides Krypto-Verfahren

# OpenPGP: V4 Fingerprint

160Bit SHA1-Hash über:

- Byte 0x99
- 2 Bytes Paketlänge
- komplettes Public Key Paket, beginnend mit Versionsfeld

# OpenPGP: transferable Public Keys

- 1 Public Key Packet
- 0-n Revocation Signatures
- 1-n User ID Packets
- 0-n Signature Packets per User ID Packet
- 0-n User Attribute Packets
- 0-n Signature Packets per User Attribute Packet
- 0-n Subkey Packets
- 1 Signature Packet per Subkey Packet
- 0-1 Revocation Signature Packet per Subkey

# OpenPGP: Verschlüsselung

- 1 Erzeugung einer Zufallszahl als symmetrischer Sitzungsschlüssel
- 2 Verschlüsselung des Sitzungsschlüssel mit dem Pubkey jedes Empfängers (bilden Message-Header)
- 3 Verschlüsselung der Nachricht mit Sitzungsschlüssel (bildet Message-Body)

# OpenPGP: digitale Signatur

- 1 Erzeugung eines Hashwertes für die Nachricht
- 2 digitales Signieren des Hashwertes
- 3 Anhängen des signierten Hashes an die Nachricht

## OpenPGP: Zusätzliche Features

- Komprimierung der erzeugten Daten
- Konvertierung nach Base64
- Langzahlarithmetik

# Web of Trust

- Pubkey signieren: `gpg --sign-key <EXPR>`  
(alternativ `--lsign-key`)
- Signatur publizieren: `gpg --send-keys`
- Pubkeys aktualisieren: `gpg --refresh-keys`

## Gültigkeit des eigenen Keys aktualisieren

```
gpg -edit-key <KeyID>
  expire   neue Gültigkeit angeben
  key 1    selektiert Encryption-Key (sub*)
  expire   neue Gültigkeit des Encryption-Keys
  save     speichert den Key
gpg -send-keys <KeyID>
```

## Sonstiges

- `gpg-agent` als Cache für die Passphrase
- `default-key` und `default-recipient-self` in *`gpg.conf`*
- `hidden-encrypt-to <KeyID>` in *`gpg.conf`*

# Links

- <http://gnupg.org>
- <http://rfc.net/rfc4880.html>