E-Mail-Verschlüsselung mit GnuPG

Augsburger Linux-Infotag

16. April 2016

Philipp Kammerer
B1 Systems GmbH
kammerer@b1-systems.de

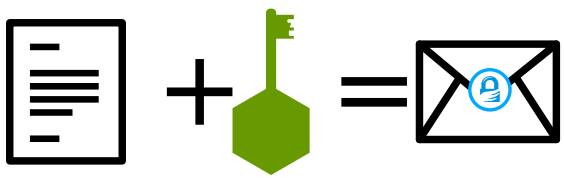
Theoretischer Hintergrund – Kurze Geschichte

- 1991: Veröffentlichung von PGP (*Pretty Good Privacy*)
- 1998: OpenPGP-Standard beschreibt die Funktionsweise der asymmetrischen Verschlüsselung.
- GnuPG wird als Freie Implementation des Standards entwickelt

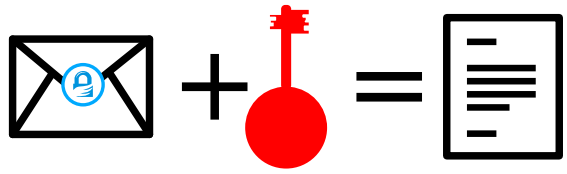
Funktionsweise

- Wer verschlüsselte Nachrichten empfangen möchte, benötigt ein Schlüsselpaar
- Verschlüsselt wird mit dem *öffentlichen* Schlüssel des Empfängers
- Entschlüsseln kann nur der *private* Schlüssel des Empfängers

Verschlüsseln mit öffentlichem Schlüssel



Entschlüsseln mit privatem Schlüssel



Verwendete Software

- E-Mail-Client (Thunderbird)
- Kryptografie-Software (GnuPG)
- Verbindung zwischen den beiden (Enigmail)

Geht's auch anders?

- Andere Mail-Clients werden auch unterstützt
- Für die Windows User: `https://www.gpg4win.de/`
- Outlook nur 32 Bit Version.

Schlüsselverwaltung

Separate Container-Dateien für öffentliche und private Schlüssel

	gpg	**gpg2**
öffentlich	`~/.gnupg/pubring.gpg`	`~/.gnupg/pubring.kbx`
privat	`~/.gnupg/secring.gpg`	`~/.gnupg/private-keys-v1.d/`

Praxis zum Mitmachen

1. Software installieren
2. Schlüssel erstellen
3. Widerrufszertifikat erstellen (optional, aber empfohlen)
4. GnuPG, Enigmail und Account konfigurieren
5. Schlüssel vom Gesprächspartner beziehen
6. E-Mails verschlüsseln
7. Keyserver nutzen

Software installieren

- Thunderbird bzw. Icedove
- gnupg2
- Entropie/Zufalls-Generator `haveged` (optional, aber empfohlen)

Software installieren

1. E-Mail-Account(s) einrichten
2. Plugin Enigmail installieren,
   Thunderbird beenden (nicht neustarten)

Schlüssel (Zertifikat) erstellen

Grafisch:

Beim ersten Starten von Enigmail automatisch über den Assitenten; Sonst manuell über die Schlüsselverwaltung.

Terminal:

Enigmail nutzt, wenn vorhanden, automatisch gpg2. Wenn man über das Terminal arbeiten will, vorher einen Alias in `~/.bash_aliases` anlegen.

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

```
$ gpg --gen-revoke <keyID> > <email/ID>-rev.asc
```

GnuPG und Enigmail konfigurieren

- Enigmail konfiguriert sich mit den Standardeinstellungen selbst.
- Bei mehreren privaten Schlüsseln ggf. in den Accounteinstellungen den Schlüssel verknüpfen.

Unterstützung für lange Key-IDs aktivieren:

```
$ echo "keyid-format xlong" >> ~/.gnupg/gpg.conf
```

Schlüssel vom Gesprächspartner beziehen

- Der öffentliche Schlüssel des Gesprächspartners muss zunächst importiert werden.
- Praktische Methode: Keyserver

Grafisch:

Direkt beim Senden einer E-Mail oder über die Schlüsselverwaltung.

Terminal:

```
$ gpg --keyserver <foo> --search <e-mail>
```

Schlüssel vom Gesprächspartner beziehen

Wenn der Schlüssel nicht über den Keyserver ausgetauscht wird:
→ Individuelle Methode: Datei

Den eigenen öffentlichen Schlüssel per E-Mail als Dateianhang an den Gesprächspartner verschicken.

Grafisch

Im- und Exportfunktion beim Senden und Empfangen

Terminal

Export: `$ gpg -a -o <dateiname.asc> --export <e-mail>`

Import: `$ gpg --import <dateiname.asc>`

E-Mails verschlüsseln

Sobald der öffentliche Schlüssel des Empfängers im Schlüsselbund ist, verschlüsselt Enigmail in der aktuellen Version automatisch.

Falls nicht, müssen die Einstellungen von Enigmail und ggf. die Account-Einstellungen überarbeitet werden.

Keyserver nutzen

Grafisch

Öffentlichen Schlüssel über die Schlüsselverwaltung hochladen.

Terminal

```
$ gpg --keyserver <foo> --send-keys <KeyID>
```

Schlüssel signieren – Keysigning Party

1. Gehört der Schlüssel wirklich zu einer bestimmten Person?
2. Wenn ja: Fingerprint abgleichen
3. Stimmt überein: Schlüssel signieren
4. Danach ggf. wieder auf einen Keyserver hochladen (`--send-keys`)

Terminal

```
$ gpg --fingerprint <e-mail>
$ gpg --edit-key <KeyID> oder <e-mail>
      sign ... (interaktiv der Menüführung folgen)
      save
$ gpg --keyserver <foo> --send-keys <KeyID>
```

Besonderheiten

Was kann man sonst noch machen?

Dateiverschlüsselung

Dateiverschlüsselung

```
$ gpg (-a) --recipient <KeyID> --encrypt <datei>
$ gpg -o <ausgabedatei> --decrypt <eingabedatei>
```

`-a`: Die Ausgabedatei wird mit ASCII-Zeichen erstellt. Lässt man `-a` weg, wird eine binäre Datei erstellt.

Dateien signieren und Signaturen prüfen

Dateien signieren und Signaturen prüfen

```
$ gpg --detach-sign (-a) <file>
$ gpg --verify <file>
```

`--local-user` um einen abweichenden Key für die Signatur zu wählen

Schlüsseleigenschaften bearbeiten

```
$ gpg --edit-key <e-mail> oder <KeyID>
```

Schlüsseleigenschaften bearbeiten

Befehl	**Aktion**
`quit`	Beenden ohne zu speichern
`save`	Speichern und Beenden
`expire`	Verfallsdatum ändern
`trust`	Vertrauensstufe festlegen
`sign`	Schlüssel signieren
`delsig`	Signatur löschen (noch nicht auf Keyserver)
`revsig`	Signatur widerrufen; wenn auf dem Keyserver
`adduid`	UID (weitere E-Mail) hinzufügen
`uid <n>`	UID auswählen
`deluid/revuid`	UID löschen/widerrufen

Neue Signaturen und TrustDB

Alle Schlüssel im Keyring (mit eventuellen neuen Signaturen) vom Server synchronisieren:

```
$ gpg --keyserver <foo> --refresh-keys
```

TrustDB updaten. Erfordert einen Schlüssel mit Ultimate Trust

```
$ gpg --update-trustdb
```

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096