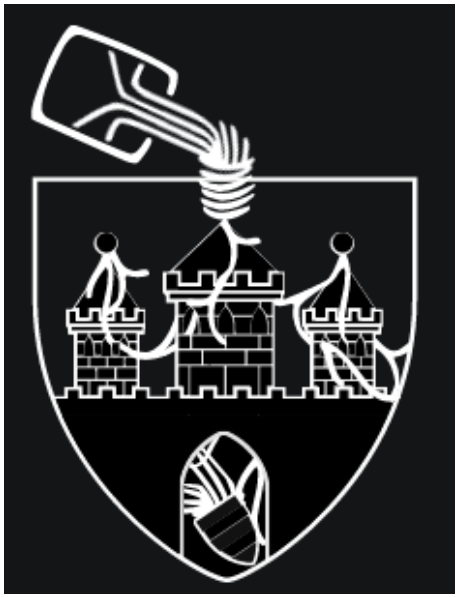
Sichere Kommunikation und Verschlüsselung am Beispiel vom GPG

Peter Gewald

Chaostreff Oldenburg

07. Mai 2015



https://ccc-ol.de

https://mainframe.io

Problem - Vertrauen

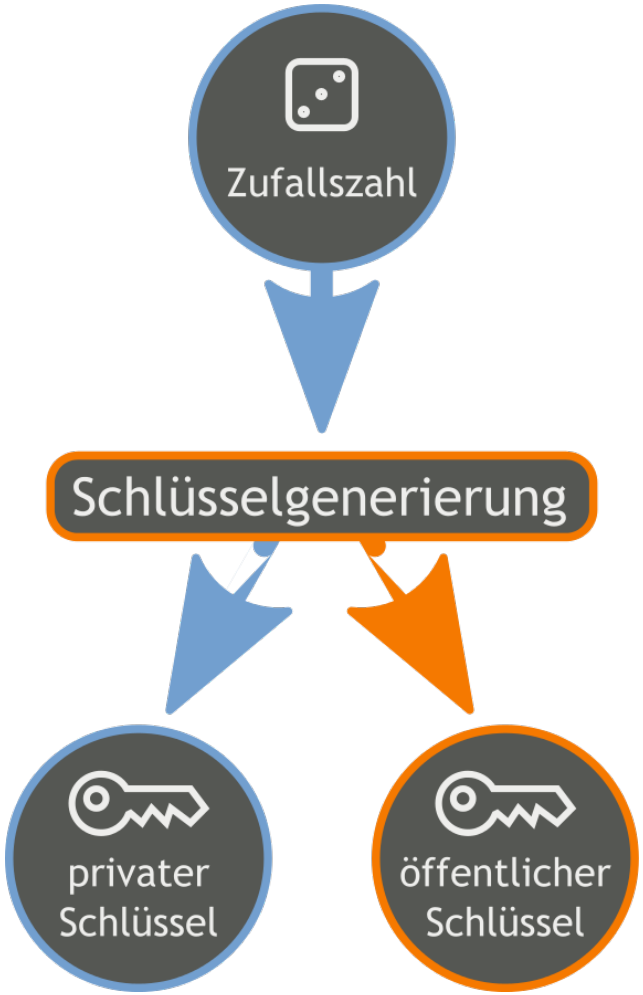
- Wem wird vertraut?
- Wer bestimmt das eine bestimmte Person/Institution vertrauenswürdig ist?
- Wie gelangt diese Information zu den Endanwendern?

Vertrauen - Lösungen

- Hierarchisch (zentral)
  - Es gibt eine oder mehrere Institutionen, denen der Endbenutzer vertraut
  - Diese Institution stellt anderen Institutionen vertrauen aus
- Web of Trust (dezentral)
  - Nutzer unterschreiben gegenseitig ihre Schlüssel
  - Vertrauensnetzwerk ohne zentrale Autoritäten

Public/Private Key Verfahren

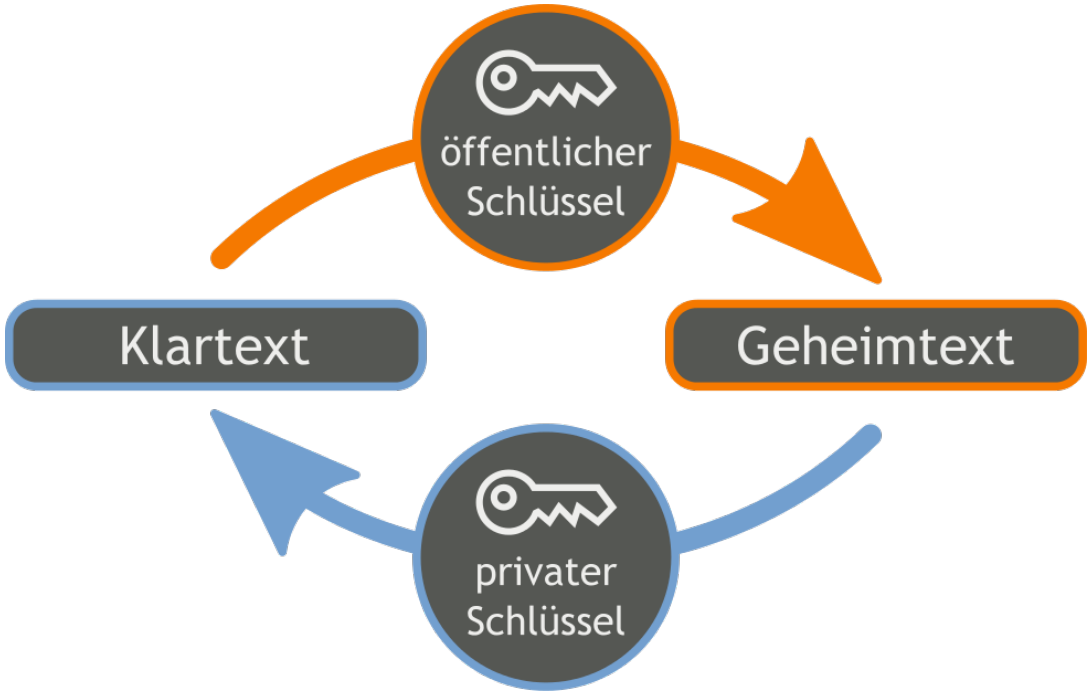
- Wie funktioniert asymetrische Verschlüsselung?
- Schlüsselpaare generieren (jeweils Sender, Empfänger)



https://upload.wikimedia.org/wikipedia/commons/9/92/Orange_blue_public_private_keygeneration_de.svg

Public/Private Key Verfahren

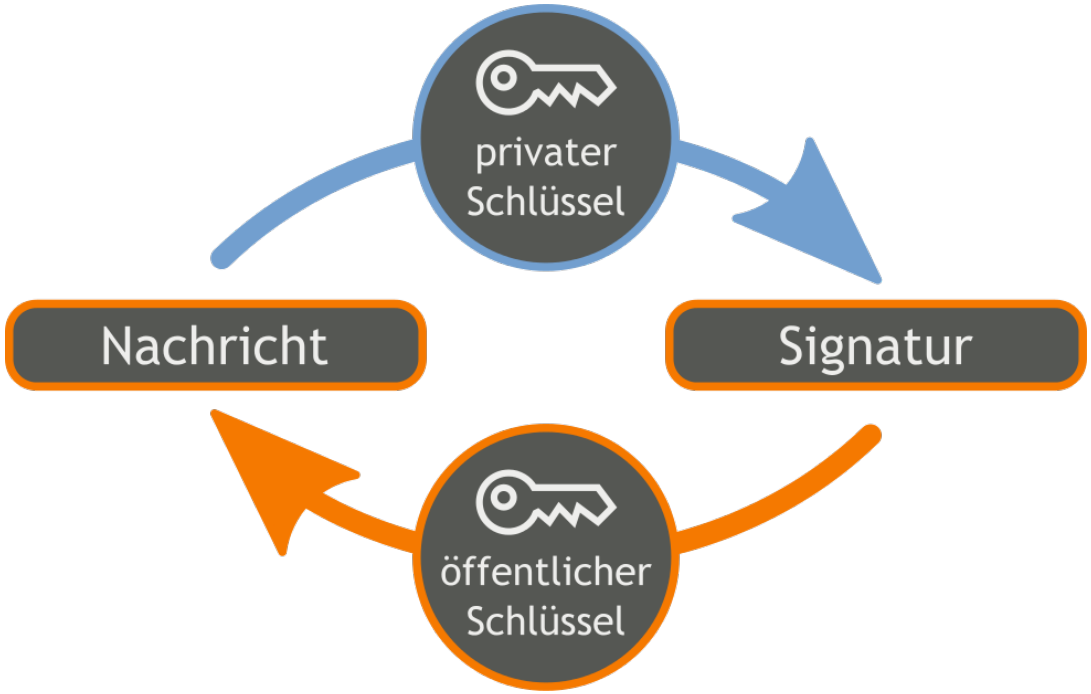
- Verschlüsselung (nur Empfänger Schlüsselpaar)



https://upload.wikimedia.org/wikipedia/commons/a/a2/Orange_blue_public_key_cryptography_de.svg

Public/Private Key Verfahren

- Signatur (nur Sender Schlüsselpaar)



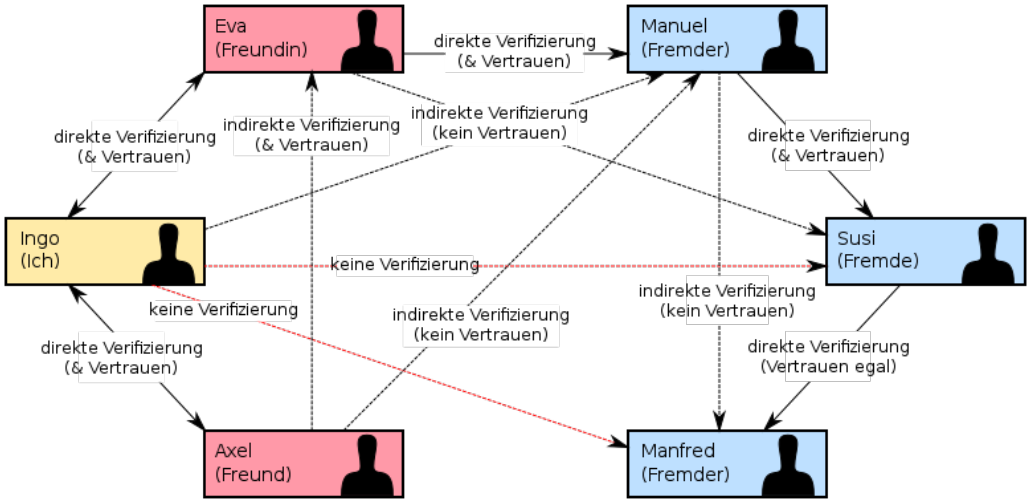
https://upload.wikimedia.org/wikipedia/commons/2/29/Orange_blue_digital_signature_de.svg

Austausch des öffentlichen Schlüssels

- Zentrales Modell: Indirekter Austausch
  - A bekommt öffentliche Schlüssel über Dritte (z.B. Browser, Mailprogramm)
  - Hierachie, Zertifikatsliste (mal reingeschaut?)

Austausch des öffentlichen Schlüssels

- Web of Trust: Direkter Austausch
  - A und B tauschen ihre öffentlichen Schlüssel aus
  - Ggf. Keyserver als Tauschplattform



https://upload.wikimedia.org/wikipedia/commons/thumb/4/49/Web_of_Trust_2.svg/2000px-Web_of_Trust_2.svg.png

Fazit

- Dezentrale Organisation sicherer für erfahrende Anwender
- ...aber auch aufwändiger

Ende-zu-Ende Verschlüsselung: GPG

- Alternatives Verfahren zu S/MIME
- Sehr verbreitet im Open Source Umfeld
- Eher ungebräuchlich im geschäftlichen Umfeld
- Web of Trust Prinzip
- Plugins für viele Clients
- Metadaten nicht verschlüsselt!

https://upload.wikimedia.org/wikipedia/de/thumb/6/6b/GnuPG.svg/300px-GnuPG.svg.png

GPG unter Linux - Installation

Debian Pakete:

- gnupg2
- signing-party (für caff)
- msmtp

GPG unter Linux - Konfiguration

- Config: ~/.gnupg/gpg.conf
- `keyserver pgp.mit.edu`
- Vorteil: Kein manuelles `--keyserver <keyserver>`

GPG unter Linux - Schlüsselpaar erstellen

- `$ gpg --gen-key`
  - RSA oder DSA?
  - Schlüssellänge?
  - Gültigkeit?
  - Name (kein Pseudonym)
  - Mailadresse
  - Passphrase (Langes_Passwort > S4l4t)
- Upload (sofern öffentlich):

  `$ gpg --send-keys <key-id>`

GPG unter Linux - Signieren

- Laden des fremden Keys:

  ```
  $ gpg --recv-keys <key-id>
  ```

- Prüfung des Fingerprints:

  ```
  $ gpg --fingerprint <key-id>
  ```

- Identitätsprüfung (Personalausweis, Führerschein)
- Signieren:

  ```
  $ gpg --sign-key <key-id>
  ```

- Passphrase eingeben
- Upload-Varianten:

  Unsicher Direkter Upload auf Keyserver

  Sicher Verschicken des signiergen Keys per Mail **an die signierten Mailadressen**

GPG unter Linux - Signieren mit caff

- Automatisch:

```
$ caff <key-id>
```

- Benötigt konfigurierten SMTP-client
- Beispiel msmtp
- Config: ~/.msmtprc
  - *host*, *port*, *user*, *passwort*

GPG unter Linux - Empfängerseite

- Datei *signature.asc* per Mail bekommen
- Signatur importieren:

  ```
  gpg --import signatur.asc
  ```

- Signatur Uploaden:

  ```
  gpg --send-keys <meine key-id>
  ```

- Done! Ready for GPG-Mails!

GPG Keys Revoken

- Entweder automatisch (Gültigkeit) oder manuell
- Wichtig: **Vorher** Cross-signieren
- Revoken $\neq$ Löschen
- Widerrufszertifikat:

  ```
  gpg --gen-revoke <key-id>
  ```

- Importieren, Uploaden
- ggf. direkt nach Generierung erzeugen
- Besser: Backup private key!

Hands-on - Let's go!

1. Schlüsselpaar erstellen
   - `gpg --gen-key`
   - (keyserver definieren)
   - `gpg --send-keys <key-id>`
2. Signieren mit Identitätsprüfung (z.B. ohne caff)
   - `gpg --recv-keys <key-id>`
   - `gpg --fingerprint <key-id>`
   - `gpg --sign-key <key-id>`
3. Signaturen per Mail verschicken
   - `caff <key-id>`
4. Uploaden auf den Keyserver (Empfänger)
   - `gpg --import signatur.asc`
   - `gpg --send-keys <meine key-id>`