

Sichere Kommunikation und Verschlüsselung am Beispiel vom GPG

Peter Gewalt

Chaostreff Oldenburg

07. Mai 2015



<https://ccc-ol.de>



<https://mainframe.io>

Problem - Vertrauen

Vertrauen

Verfahren

Fazit

Hands-on Theorie

Hands-on Let's go

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

Vertrauen - Lösungen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Hierarchisch (zentral)
 - Es gibt eine oder mehrere Institutionen, denen der Endbenutzer vertraut
 - Diese Institution stellt anderen Institutionen vertrauen aus

Vertrauen - Lösungen

Vertrauen

Verfahren

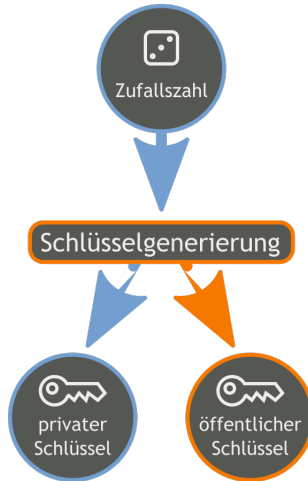
Fazit

Hands-on
TheorieHands-on
Let's go

- 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

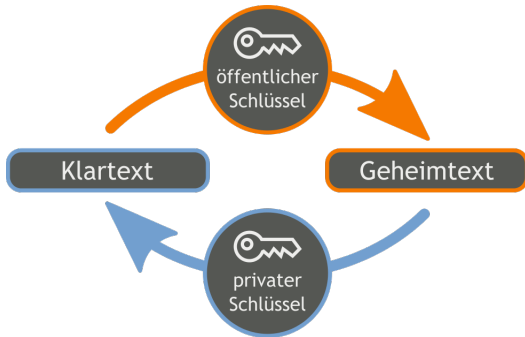
Public/Private Key Verfahren

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



Public/Private Key Verfahren

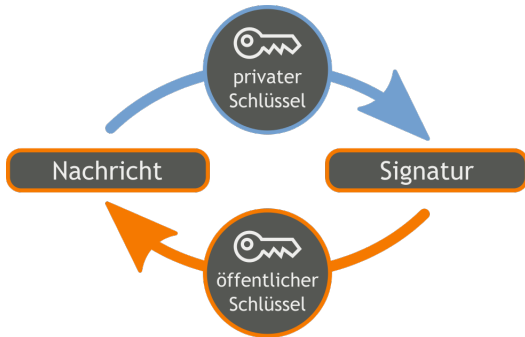
- Verschlüsselung (nur Empfänger Schlüsselpaar)



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)
 - Hierarchie, Zertifikatsliste (mal reingeschaut?)

-
- Das Diagramm zeigt die Vertrauensbildung zwischen fünf Personen:
- Eva (Freundin)** (rosa Hintergrund)
 - Manuel (Fremder)** (blauer Hintergrund)
 - Ingo (Ich)** (gelber Hintergrund)
 - Susi (Fremde)** (blauer Hintergrund)
 - Axel (Freund)** (rosa Hintergrund)
- Die Beziehungen sind wie folgt dargestellt:
- Eva zu Manuel:** direkte Verifizierung (& Vertrauen)
 - Eva zu Ingo:** indirekte Verifizierung (& Vertrauen)
 - Eva zu Axel:** indirekte Verifizierung (& Vertrauen)
 - Manuel zu Susi:** direkte Verifizierung (& Vertrauen)
 - Manuel zu Axel:** indirekte Verifizierung (kein Vertrauen)
 - Ingo zu Axel:** keine Verifizierung
 - Ingo zu Manuel:** indirekte Verifizierung (kein Vertrauen)
 - Axel zu Ingo:** direkte Verifizierung (& Vertrauen)
 - Axel zu Manuel:** indirekte Verifizierung (kein Vertrauen)
 - Susi zu Axel:** direkte Verifizierung (Vertrauen egal)

9/19

Fazit

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

Ende-zu-Ende Verschlüsselung: GPG

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Alternatives Verfahren zu S/MIME
- Sehr verbreitet im Open Source Umfeld



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

Ende-zu-Ende Verschlüsselung: GPG

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Alternatives Verfahren zu S/MIME
- Sehr verbreitet im Open Source Umfeld
- Eher ungebräuchlich im geschäftlichen Umfeld
- Web of Trust Prinzip



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

Ende-zu-Ende Verschlüsselung: GPG

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- 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)
- msmtpt

GPG unter Linux - Konfiguration

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

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

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- `$ gpg --gen-key`

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- `$ gpg --gen-key`
 - RSA oder DSA?

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- \$ `gpg --gen-key`
 - RSA oder DSA?
 - Schlüssellänge?

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- \$ `gpg --gen-key`
 - RSA oder DSA?
 - Schlüssellänge?
 - Gültigkeit?

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- `$ gpg --gen-key`
 - RSA oder DSA?
 - Schlüssellänge?
 - Gültigkeit?
 - Name (kein Pseudonym)

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- \$ `gpg --gen-key`
 - RSA oder DSA?
 - Schlüssellänge?
 - Gültigkeit?
 - Name (kein Pseudonym)
 - Mailadresse

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

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

GPG unter Linux - Schlüsselpaar erstellen

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- `$ 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>
```

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

GPG unter Linux - Signieren

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Laden des fremden Keys:
\$ gpg --recv-keys <key-id>
- Prüfung des Fingerprints:
\$ gpg --fingerprint <key-id>

GPG unter Linux - Signieren

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Laden des fremden Keys:
`$ gpg --recv-keys <key-id>`
- Prüfung des Fingerprints:
`$ gpg --fingerprint <key-id>`
- Identitätsprüfung (Personalausweis, Führerschein)

GPG unter Linux - Signieren

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- 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>`

GPG unter Linux - Signieren

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- 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

GPG unter Linux - Signieren

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- 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 signierten Keys per Mail **an die signierten Mailadressen**

GPG unter Linux - Signieren mit caff

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Automatisch:
\$ caff <key-id>

GPG unter Linux - Signieren mit caff

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Automatisch:
 \$ caff <key-id>
- Benötigt konfigurierten SMTP-client
- Beispiel msmtp

GPG unter Linux - Signieren mit caff

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Automatisch:
\$ `caff <key-id>`
- Benötigt konfigurierten SMTP-client
- Beispiel msmtprc
- Config: `~/.msmtprc`
 - *host, port, user, password*

GPG unter Linux - Empfängerseite

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Datei *signature.asc* per Mail bekommen

GPG unter Linux - Empfängerseite

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Datei *signature.asc* per Mail bekommen
- Signatur importieren:
`gpg --import signatur.asc`

GPG unter Linux - Empfängerseite

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

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

```
gpg --import signatur.asc
```
- Signatur Uploaden:

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

GPG unter Linux - Empfängerseite

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- 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

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Entweder automatisch (Gültigkeit) oder manuell

GPG Keys Revoken

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Entweder automatisch (Gültigkeit) oder manuell
- Wichtig: **Vorher** Cross-signieren

GPG Keys Revoken

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

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

GPG Keys Revoken

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

- Entweder automatisch (Gültigkeit) oder manuell
- Wichtig: **Vorher** Cross-signieren
- Revoken \neq Löschen
- Widerrufszeugnis:
`gpg --gen-revoke <key-id>`
- Importieren, Uploaden

GPG Keys Revoken

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-on
Let's go

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

```
gpg --gen-revoke <key-id>
```
- Importieren, Uploaden
- ggf. direkt nach Generierung erzeugen
- Besser: Backup private key!

Hands-on - Let's go!

Vertrauen

Verfahren

Fazit

Hands-on
TheorieHands-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>`