Sicher verschlüsseln mit GnuPG

Werner Koch
wk@gnupg.org

Sommerakademie 2015 — Kiel, 31. August 2015

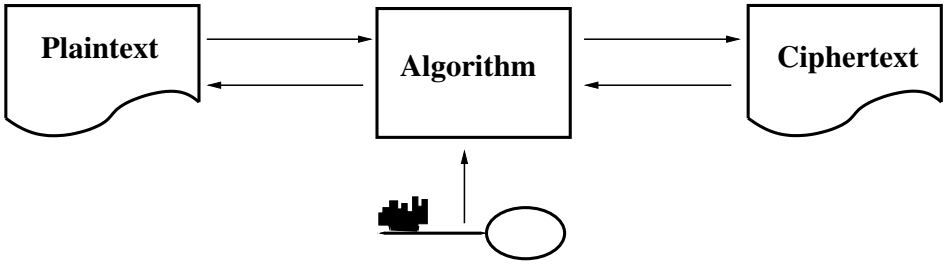
GnuPG

Schnellkurs Public Key Kryptographie Basisfunktionen Automatisieren mit GnuPG

Outline

GnuPG

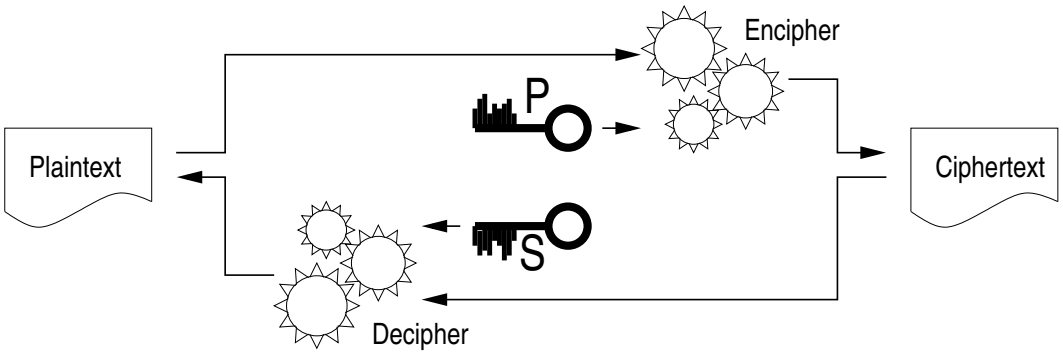
Symmetrische Verschlüsselung



- Derselbe Schlüssel wird zum Ver- und Entschlüsseln benutzt.
- Sender und Empfänger kennen beide diesen Schlüssel und halten ihn geheim („Shared Secret“).
  - Wie eine Passphrase zum Anmelden bei einem Online-Service.
  - Passphrase basierte Systeme sind unsicher, sofern die Passphrase nicht über eine Zufallsgenerator erzeugt wurde.
- Schlüsselaustausch und -verwaltung sind schwierig und nur praktikabel bei wenigen Relationen.

GnuPG

Asymmetrische Verschlüsselung (Public Key)



- Es wird ein Schlüsselpaar aus öffentlichem und privatem (geheimen) Schlüssel verwendet.
- Falltüralgorithmus aus Encipher und Decipher.
- Öffentlicher Schlüssel nur zum Verschlüsseln.
- Privater Schlüssel zum Entschlüsseln.
- Öffentliche Schlüssel im öffentlichen Verzeichnis.

GnuPG

Digitale Signaturen

Verfahren:

- Public-Key kann auch zum Signieren dienen.
- Privater Schlüssel zur Erstellung der Signatur.
- Öffentlicher Schlüssel zur Prüfung der Signatur.

Anwendungsgebiete:

- Datenintegrität.
- Erstellung von Zertifikaten.

GnuPG

Algorithmen

Gängige Public-Key Verfahren:

- RSA (verschlüsseln, signieren)
- DSA (signieren)
- Elgamal (verschlüsseln)
- ECC, Elliptische Kurven (verschlüsseln, signieren)
  - Kürzere Schlüssel (256 bit)
  - Gleiche Sicherheit (RSA mit 4096 bit)

GnuPG

Hybride Verfahren

Public-Key Verfahren sind wesentlich langsamer als symmetrische Verfahren.

- Ein zufälliger Sitzungsschlüssel von 256 Bit wird erzeugt,
- dieser wird mit einem Public-Key Verfahren an den Empfänger verschlüsselt,
- die Daten werden mit dem Sitzungsschlüssel symmetrisch verschlüsselt.

Vorteile:

- Effizient bei allen Datengrößen.
- Einfaches Verschlüsseln an mehrere Empfänger.

Zertifikate und PKI

Wie entscheiden ob der Schlüssel authentisch ist?

- Von Hand verwaltete Liste gültiger Schlüssel (z.B. im Adreßbuch).
- Ein Verzeichnis von gültigen Schlüsseln.
- Eine zentrale PKI (Public-Key Infrastructure), die auf einem hierarchisch aufgebauten System von Zertifizierungsstellen beruht.
- Eine dezentrale PKI wie das „Web-of-Trust“.
- Ein lokales „Trust-On-First-Use” Verfahren erkennt geänderte Schlüssel nach deren ersten Verwendung.

Erzeugen eines Schlüsselpaars

```
$ gpg --gen-key
gpg (GnuPG) 2.1.7; Copyright (C) 2015 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.

gpg: keybox '/home/wk/b/gnupg/kiel2015/pubring.kbx' created
Note: Use "gpg2 --full-gen-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: John Steed
Email address: steed@example.org
You selected this USER-ID:
    "John Steed <steed@example.org>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
```

gpg erzeugt den Schlüssel ...

und zeigt diesen dann an:

```
gpg: key 3F567FB6 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: public key of ultimately trusted key 912FCB93 not found
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
pub  rsa2048/3F567FB6 2015-08-12
      Key fingerprint = AF19 1E21 6B28 0B02 65E6  50C1 6415 179B 3F56 7FB6
uid         [ultimate] John Steed <steed@example.org>
sub   rsa2048/63B40B8C 2015-08-12
```

Backup

- Der private Schlüssel is wichtig.
- Ausdruck erstellen für Disaster Recovery:
  - Bis 2.0: Mittels `paperkey` drucken,
  - Seit 2.1: Ausgabe von `gpg -a --export-secret-key` drucken.
- Lokalen Drucker verwenden!
- Passphrase getrennt notieren!
- Backup unter Unix:

```
$ tar czf keys-DATUM.tar.gz --exclude random_seed ~/.gnupg
```

- Backup unter Windows:

```
> gpgconf --list-dirs
> cd DIR
> del random_seed
> gpgtar --skip-crypto -eo backup-keys-DATUM.tar .
```

GnuPG

Import von Schlüsseln

- Von Webseite holen und abspeichern:

```
$ wget -O a.key https://www.datenschutzzentrum.de/uploads/uld/uld.asc
$ gpg --import a.key
$ rm a.key
```

- Über Keyserver holen:

```
$ gpg --keyserver keys.gnupg.net --recv-key 0D75199E11357324
gpg: key 0D75199E11357324: public key "ULD-SH <mail[...]>" imported
gpg: public key of ultimately trusted key 0F1EB16A912FCB93 not found
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: Total number processed: 1
gpg:               imported: 1
```

GnuPG

Verschlüsseln

Teil 1

```
$ gpg -v -e -r mail@datenschutzzentrum.de datei.pdf
gpg: using PGP trust model
gpg: using subkey A749BED409A66C9A instead of primary key 0D75199E11357324
...
```

- Das `-v` ist optional um anzuzeigen was passiert.
- Das `-e` wählt Verschlüsselung aus.
- Das `-r` gibt den Empfänger an.
- `datei.pdf` ist die zu verschlüsselnde Datei.

Verschlüsseln

Teil 2

```
gpg: A749BED409A66C9A: There is no assurance this key belongs to [...]
sub  elg4096/A749BED409A66C9A 2008-04-11 ULD-SH <mail@[...]zentrum.de>
Primary key fingerprint:  D092 F1B5 AB9F D68E 4DA0  3633 [...]
      Subkey fingerprint: 4E31 0B46 A394 DE69 D56A  6F82 [...]

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
gpg: reading from 'datei.pdf'
gpg: writing to 'datei.pdf.gpg'
gpg: ELG/AES256 encrypted for: "A749BED409A66C9A ULD-SH <mail@[...]>"
```

- Der Schlüssel ist nicht bekannt.
- Der Fingerprint wird angezeigt.
- Nach Überprüfung mit „Ja“ oder „Yes“ antworten.
- Die verschlüsselten Daten sind abgespeichert.

Schlüssel vertrauenswürdig setzen

```
$ gpg --lsign-key  0D75199E11357324
...
pub  dsa3072/0D75199E11357324
     created: 2008-04-11  expires: never       usage: SC
     trust: unknown       validity: unknown
 Primary key fingerprint: D092 F1B5 AB9F D68E 4DA0  3633 [...]

     ULD-SH <mail@datenschutzzentrum.de>

Are you sure that you want to sign this key with your
key "John Steed <steed@example.org>" (6415179B3F567FB6)

The signature will be marked as non-exportable.

Really sign? (y/N) y
```

- `--lsign-key` zum lokalen Signieren verwenden.
- Per Telefon oder mittels einer Publikation Fingerprint prüfen,
- bei positivem Resultat mit „Yes“ oder „Ja” bestätigen.

Schlüsselstatus anzeigen

```
$ gpg --fingerprint 0D75199E11357324
gpg: checking the trustdb
...
pub   dsa3072/0D75199E11357324 2008-04-11
      Key fingerprint = D092 F1B5 AB9F D68E 4DA0  3633 0D75 199E 1135 7324
uid                  [ full  ] ULD-SH <mail@datenschutzzentrum.de>
sub   elg4096/A749BED409A66C9A 2008-04-11
```

- Allgemeines Kommando um den Fingerprint anzuzeigen.
- In diesem Beispiel ist der Schlüssel vertrauenswürdig (full).
- Eine Liste der Vertrauensstufen findet sich im Handout.

Signieren
(standard)

```
$ gpg -v -s datei.pdf
gpg: using PGP trust model
gpg: writing to 'datei.pdf.gpg'
gpg: RSA/SHA256 signature from: "6415179B3F567FB6 John Steed <steed@example
```

- Das `-s` (oder `--sign`) wählt Signieren aus.
- `datei.pdf` ist die zu signierende Datei.
- `datei.pdf.sig` ist die erstellte Datei mit Signatur.

Signieren
(abgetrennt)

```
$ gpg -v -b datei.pdf
gpg: using PGP trust model
gpg: writing to 'datei.pdf.sig'
gpg: RSA/SHA256 signature from: "6415179B3F567FB6 John Steed <[...]>"
```

- Das `-b` (oder `--detach-sign`) wählt Signieren aus.
- `datei.pdf` ist die zu signierende Datei.
- `datei.pdf.sig` ist die erstellte abgetrennte Signatur.

Signieren
(anderer Schlüssel)

```
$ gpg -v -b -u peel datei.pdf
gpg: writing to 'datei.pdf.sig'
gpg: EDDSA/SHA256 signature from: "EA9644E68E27FD07 Emma Peel <[...]>"
```

- Das `-u` (oder `--local-user`) wählt den Signaturschlüssel aus.
- Name, Mailadresse oder Keyid wird benötigt.
- Dieser Schlüssel benutzt einen ECC Algorithmus.

Signatur prüfen

```
$ gpg -v --verify datei.pdf.sig datei.pdf
gpg: Signature made Sun 16 Aug 2015 09:24:06 AM CEST
gpg:                using EDDSA key EA9644E68E27FD07
...
gpg: Good signature from "Emma Peel <peel@example.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs [...]
Primary key fingerprint: CA4A EF4F 0065 91A8 DF82  761F EA96 44[...]
gpg: binary signature, digest algorithm SHA256, key algorithm ed25519
```

- `--verify` wählt die Prüfung einer Signatur aus.
- `datei.pdf.sig` ist die abgetrennte Signatur.
- `datei.pdf` ist die zu prüfende Datei.
- Die Signatur is gültig, aber ...
- es ist nicht klar ob dies Emmas Schlüssel ist, ...
- deswegen den Fingerprint prüfen.

Wie man OpenPGP Dateien erkennt

- Mittels eines Tools anhand des Inhalts: Entweder durch einen Versuch mit `gpg`, oder auf Unix, mit dem Tool `file`.
- Mittels der Bibliotheksfunktion `gpgme_data_identify` von Libgpgme.
- Bei „armored“ Dateien auch visuell anhand des Inhalts (z.B. `-----BEGIN PGP MESSAGE-----`).
- Anhand der Dateiendung (per Konvention):
  - `.sig` Binäre abgetrennte Signatur.
  - `.pub` Datei mit öffentliche Schlüsseln.
  - `.sec` Datei mit privaten Schlüsseln.
  - `.asc` „Armored“ OpenPGP Datei.
  - `.gpg` Andere binäre OpenPGP Datei.
  - `.pgp` Dito, aber von PGP verwendet.

GnuPG

Komprimieren

- Verschlüsselte Daten können nicht mehr komprimiert werden.
- `gpg` komprimiert deswegen die Daten bevor sie Verschlüsselt werden.
- Bereits verschlüsselte Daten werden i.d.R. erkannt und die Komprimierungsstufe wird ausgeschaltet.
- Mit `-c 0` kann die Komprimierung explizit ausgeschaltet werden.
- Beim Entschlüsseln werden die Daten automatisch dekomprimiert. Vorsicht: ZIP Bombe.
- Eventuell die Option `--max-output` benutzen.

GnuPG

S/MIME

- GnuPG unterstützt auch S/MIME (X.509/CMS).
- S/MIME ist nicht kompatibel zu OpenPGP.
- Es werden andere Schlüssel verwendet und diese müssen durch eine CA zertifiziert werden.
- Das Tool `gpgsm` wird anstatt von `gpg` verwendet.
- Auch für X.509 Zertifikate und CSRs für Webserver.

GnuPG

Pipelines

Beispiel: In der Wolke speichern

```
$ tar cf - /var/log \
| gpg --batch -e --always-trust -r 0x12345678abcdef0 \
| ssh backup@archive 'cat >"backup-$(date +%Y-%m-%d).tar.gpg"'
```

- `tar` kopiert rekursiv alle Dateien aus `/var/log/` nach „stdout“,
- `gpg` liest, verschlüsselt und gibt nach „stdout“ aus.
  - `--batch` verhindert hierbei jede Nachfrage.
  - `-e` fordert Verschlüsselung an.
  - `--always-trust` vertraut allen angegebenen Schlüsseln.
  - `0x12345678abcdef0` ist eine Key-ID.
    (Besser: Fingerprint benutzen)
- `ssh` verbindet mit Host „archive“, führt dort `cat` aus und schreibt in eine Datei.

GnuPG

Pipelines

Beispiel: Aus der Wolke holen

```
$ cd restored-logs
$ ssh backup@archive 'cat DATEI.tar.gpg' \
| gpg --batch -d  --max-output 0x80000000 \
| tar xpf -
```

- Ins Zielverzeichnis wechseln,
- per `ssh` die Datei nach „stdout“ senden,
- `gpg` entschlüsselt die Daten.
  - `-d` (oder `--decrypt`) kann entfallen.
  - `--max-output` gibt die maximal erwartete Länge der Ausgabe in Bytes an.
- `tar` entpackt das entschlüsselte Archiv.

Unbeaufsichtigte Benutzung

Sinnvolle Optionen:

- `--status-fd 2` erzeugt maschinell verarbeitbare Ausgaben.
- `--batch` schaltet alle Abfragen aus.
- `--yes` benutzt implizit „Ja“ für viele Abfragen;
  - aber nicht überall.
- `--max-output N` kann zur Verhinderung von ZIP Bomben benutzt werden.
- `--trust-model=always` kann u.U. mit `-r` benutzt werden (Alternative zu `--lsign-key`).

Ratschläge

- Nach Möglichkeit, Schlüssel immer per Fingerprint angeben.
- Signaturschlüssel explizit auswählen.
- `--encrypt-to` benutzen um verschlüsselte Daten auch selbst entschlüsseln zu können.
  - Achtung: Verhindert Anonymität des Senders.
  - `--no-encrypt-to` schaltet dies explizit aus.
- Definierte Konfigurationsdatei und GNUPGHOME nutzen.
- Auf Servern keine Passphrase setzen bzw. Smartcard benutzen.
- Immer aktuelle Versionen von GnuPG verwenden.

GnuPG

Zusammenfassung

- GnuPG ist vielseitig zu verwenden,
- Leicht in Skripte einbindbar.
- Sichere, etablierte Algorithmen und Protokolle.
- Kostengünstig.
- Zukunftssicher.

Vielen Dank.

- `https://gnupg.org`
- `https://wiki.gnupg.org`