

Sichere Email-Kommunikation

Verschlüsselung mit GnuPG

Florian Streibelt

<florian@freitagrunde.org>

6. Juli 2012



This work is licensed under the [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?



4!

- 1 Vorschau
- 2 Ziele der Verschlüsselung**
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?



4!

Ziele von Verschlüsselung

Wann ist die Kommunikation/sind die Daten 'sicher' ?

- Geheimhaltung (Confidentiality)
 - Nachricht soll nicht durch Dritte entziffert werden können.
- Integrität der Nachricht (Integrity)
 - Ist die Nachricht so verfasst worden, wie sie angekommen ist?
- Echtheit des Kommunikationspartners (Authenticity)
 - Stammt die Nachricht wirklich von dem, der er vorgibt zu sein?

→ Vertrauen in die Kommunikation

⇒ Ist (meist) erreicht, wenn die obigen Punkte eingehalten werden.

→ In der klassischen IT-Security kommt z.B. noch Availability hinzu.

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email**
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?



4!

Bestandsaufnahme

- Email-Verkehr kann sehr einfach abgehört und manipuliert werden
- ein sehr hoher Prozentsatz der Kommunikation erfolgt dennoch per Email!
- interne Daten werden damit per elektronischer Postkarte versandt
- im Rahmen der „Terrorismusbekämpfung“ immer mehr Befugnisse für div. Dienste
- gespeicherte Emails gelten nicht mehr als geschützte Kommunikation (Hausdurchsuchung)

Email 'abhören'

```
220 mail.freitagrunde.org ESMTP Postfix
HELO fls.local.freitagrunde.org
250 mail.freitagrunde.org
MAIL FROM: <florian@freitagrunde.org>
250 2.1.0 Ok
RCPT TO: <florian.streibelt@TU-Berlin.DE>
250 2.1.5 Ok
DATA
354 End data with <CR<LF>.<CR<LF>
From: "Florian Streibelt" <florian@freitagrunde.org>
To: "Florian Streibelt" <florian.streibelt@TU-Berlin.de>
Subject: Test...
Date: 16 Nov 13:15:56 CET 2010

Hallo ,
hier ist der Inhalt der Email....

.
250 2.0.0 Ok: queued as 81F5AB1C009
QUIT
221 2.0.0 Bye
```

⇒ Emails werden in der Regel im Klartext versendet und können einfach mitgelesen werden.

⇒ Kaum ein SMTP-Server checkt Zertifikate, SSL oft nicht 'enforced'

Lösung durch Kryptographie

Die Lösung ist:

- Kryptographie

= Wissenschaft von der Verschlüsselung und Verschleierung

Im Gegensatz:

- Kryptoanalyse

= Wissenschaft von der Entschlüsselung

A large, light gray graphic element consisting of a circle containing the number '4' and an exclamation mark '!' in white. The background of the slide features several overlapping, semi-transparent gray shapes, including a large circle and several triangles, creating a geometric pattern.

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte**
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?



4!

Wie ist Kryptographie entstanden?

- die Ursprünge reichen sehr weit zurück
- zunächst wurden die Daten „nur“ versteckt
 - ein geheimer Bote überbringt die Nachricht
- später wurden die Daten verschlüsselt
 - auch der Bote kann die Nachricht nun nicht mehr lesen oder manipulieren
 - bei Gefangennahme eines Boten bleibt die Nachricht geheim

Skytale von Sparta



Skytala von Sparta

- älteste bekannte (militär.) Verschlüsselungsmethode (ca. 2500 Jahre)
- von Plutarch überliefert
- Band wird um einen Zylinder gelegt, quer beschrieben und wieder abgerollt
- Schlüssel besteht aus dem Zylinderdurchmesser

⇒ Transpositionschiffre: Die Anordnung der Zeichen wird vertauscht

Bildquelle:

<http://de.wikipedia.org/w/index.php?title=Datei:Skytala%26EmptyStrip-Shaded.png&filetimestamp=20070216193708>

Skytale (Beispiel)

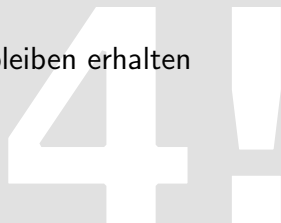
Beispiel Skytala

- Geheimtext:
FTRETRAUDUEGNEBISDRX
- Test mit Durchmesser 5cm:
FTDNSTRUEDRAEBREUGIX
- Test mit Durchmesser 4cm:
FREITAGSRUNDEDERTUBX



4!

- Julius Cäsar (100-44 v.Chr.) zugeschrieben
- Buchstaben werden um einige Stellen verschoben
- Klasse der Verschiebec chiffren
- ROT13 als „moderne“ Variante
- Problem: Buchstabenhäufigkeiten bleiben erhalten

A large, light gray graphic consisting of a circle containing a white number '4' and a white exclamation mark '!'.

Cäsarchiffre (Beispiel)

Chiffrieralphabet mit $n=3$

- Klartextalphabet:

a b c d e f g h i j k l m n o p q r s t u v w x y z

- Geheimtextalphabet:

d e f g h i j k l m n o p q r s t u v w x y z a b c

Beispiel

- Klartext:

FREITAGSRUNDEDERTUB

- Geheimtext:

IUHLWDJVUXQGHGHUWXE

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung**
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?

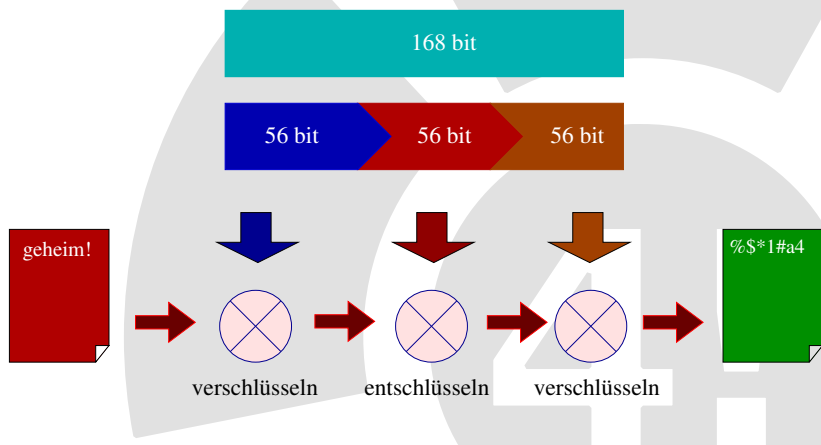


4!

- Voraussetzungen:
 - Schlüsselänge gleich Klartextlänge
 - Schlüsselerzeugung streng zufällig
 - Schlüssel wird nur einmal verwendet
- Probleme:
 - menschlicher Faktor
 - Beschaffung neuer OneTimePads...
- einzige beweisbar sichere Verschlüsselung
- hoher organ. Aufwand → militärische Umgebungen

- Data Encryption Standard von 1977
- symmetrische Blockchiffre mit 56 Bit
- im kommerziellen Bereich am häufigsten eingesetzt
- u.a. 1999 in ca. 22 Stunden gebrochen
- Weiterentwicklung: 3DES, benutzt 3 unterschiedliche DES Schlüssel hintereinander: 168 Bit
- Wurde (wird?) in Geldautomaten verwendet sowie in Behördenfunk und Kerberos

3DES Schema



Symmetrische vs. Asymmetrische Verschlüsselung

Nachteile aller bisherigen Methoden:

Zur Kommunikation über unsichere Kanäle (Internet) muss der Schlüssel zuerst über einen sicheren Kanal übertragen werden. ein Schlüssel pro Kommunikationspartner

Ausweg: asymmetrische Verschlüsselung

Es gibt einen geheimen und einen öffentlichen Schlüssel. Der öffentliche Schlüssel dient zum Verschlüsseln von Daten, nur der Geheime kann die Originaldaten wieder herstellen. (idealerweise)

- Das Prinzip wurde von Whitfield Diffie und Martin Hellman 1976 entwickelt und publiziert
- Ron Rivest, Adi Shamir und Leonard Adleman wollten dessen Sicherheit widerlegen
- ... und erfanden dabei RSA
- ältester und angesehenster asymmetrischer Algorithmus,
- löste als erster das Schlüsselverteilungsproblem
- basiert auf der Faktorisierung großer Zahlen...

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q



4!

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$



4!

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$
- Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(N) = (p - 1) \cdot (q - 1)$ (Eulersche φ -Funktion)



4!

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$
- Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(N) = (p - 1) \cdot (q - 1)$ (Eulersche φ -Funktion)
- Wähle eine zu $\varphi(N)$ Teilerfremde Zahl e mit $1 < e < \varphi(N)$



4!

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$
- Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(N) = (p - 1) \cdot (q - 1)$ (Eulersche φ -Funktion)
- Wähle eine zu $\varphi(N)$ Teilerfremde Zahl e mit $1 < e < \varphi(N)$
- Berechne Entschlüsselungsexponenten d mit:
 $e \cdot d \equiv 1 \pmod{\varphi(N)}$ (multiplikativ Inverses)

4!

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$
- Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(N) = (p - 1) \cdot (q - 1)$ (Eulersche φ -Funktion)
- Wähle eine zu $\varphi(N)$ Teilerfremde Zahl e mit $1 < e < \varphi(N)$
- Berechne Entschlüsselungsexponenten d mit:
 $e \cdot d \equiv 1 \pmod{\varphi(N)}$ (multiplikativ Inverses)
- Suche also Zahl d , für die gilt:
 $e \cdot d \pmod{\varphi(N)} = 1$ (erweiterter Euklidischer Algorithmus)

RSA (Prinzip)

- Wahl zweier grosser Primzahlen p und q
- Ermittlung des RSA-Moduln $N = p \cdot q$
- Ermittlung der Anzahl der zu n teilerfremden Zahlen $\varphi(N) = (p - 1) \cdot (q - 1)$ (Eulersche φ -Funktion)
- Wähle eine zu $\varphi(N)$ Teilerfremde Zahl e mit $1 < e < \varphi(N)$
- Berechne Entschlüsselungsexponenten d mit:
 $e \cdot d \equiv 1 \pmod{\varphi(N)}$ (multiplikativ Inverses)
- Suche also Zahl d , für die gilt:
 $e \cdot d \pmod{\varphi(N)} = 1$ (erweiterter Euklidischer Algorithmus)
- e und N sind der öffentliche Schlüssel
- d und N sind der private Schlüssel

Public und Private Key

- Der Schlüssel besteht aus zwei Teilen
- öffentlicher Teil: Zum Verschlüsseln
- privater Teil: Zum Entschlüsseln
- mathematisch gesehen sind beides nur Zahlen



4!

Public und Private Key

- Der Schlüssel besteht aus zwei Teilen
- öffentlicher Teil: Zum Verschlüsseln
- privater Teil: Zum Entschlüsseln
- mathematisch gesehen sind beides nur Zahlen
- Verschlüsseln: $secret = x^e$
- Entschlüsseln: $clear = secret^d$



4!

Public und Private Key

- Der Schlüssel besteht aus zwei Teilen
- öffentlicher Teil: Zum Verschlüsseln
- privater Teil: Zum Entschlüsseln
- mathematisch gesehen sind beides nur Zahlen
- Verschlüsseln: $secret = x^e$
- Entschlüsseln: $clear = secret^d$
- Zu Beweisen: $(x^e)^d = x$ für alle x

4!

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$



4!

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$



4!

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$



4!

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$$

Man erhält mit $e \cdot d = k \cdot \varphi(N) + 1$:

$$m^{(e \cdot d)} = m^{k \cdot \varphi(N) + 1}$$



4!

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$$

Man erhält mit $e \cdot d = k \cdot \varphi(N) + 1$:

$$m^{(e \cdot d)} = m^{k \cdot \varphi(N) + 1} = m \cdot m^{k \cdot \varphi(N)}$$

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

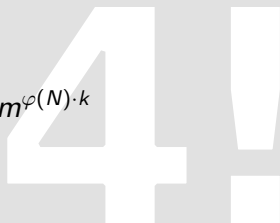
Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$$

Man erhält mit $e \cdot d = k \cdot \varphi(N) + 1$:

$$m^{(e \cdot d)} = m^{k \cdot \varphi(N) + 1} = m \cdot m^{k \cdot \varphi(N)} = m \cdot m^{\varphi(N) \cdot k}$$



Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$$

Man erhält mit $e \cdot d = k \cdot \varphi(N) + 1$:

$$m^{(e \cdot d)} = m^{k \cdot \varphi(N) + 1} = m \cdot m^{k \cdot \varphi(N)} = m \cdot m^{\varphi(N) \cdot k}$$

Nach dem Satz von Euler gilt $m^{\varphi(N)} = 1 \pmod{N}$,

$$\text{also } m \cdot m^{\varphi(N) \cdot k} = m \cdot 1^k = m$$

Zu zeigen: $(x^e)^d = x^{(e \cdot d)} \pmod{N}$

Nach Voraussetzung ist $N = p \cdot q$ mit q, p Primzahlen.

Damit gilt $\varphi(N) = (p - 1) \cdot (q - 1)$

Außerdem war $e \cdot d \equiv 1 \pmod{\varphi(N)}$

Wegen $e \cdot d \equiv 1 \pmod{\varphi(N)}$ gibt es ein $k \in \mathbb{N}$ mit

$$e \cdot d = k \cdot \varphi(N) + 1 \pmod{\varphi(N)}$$

Man erhält mit $e \cdot d = k \cdot \varphi(N) + 1$:

$$m^{(e \cdot d)} = m^{k \cdot \varphi(N) + 1} = m \cdot m^{k \cdot \varphi(N)} = m \cdot m^{\varphi(N) \cdot k}$$

Nach dem Satz von Euler gilt $m^{\varphi(N)} = 1 \pmod{N}$,

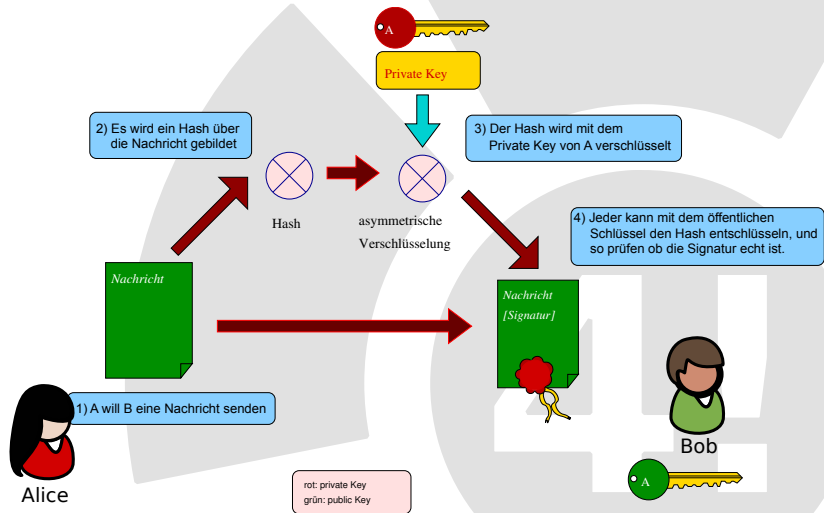
$$\text{also } m \cdot m^{\varphi(N) \cdot k} = m \cdot 1^k = m \quad \square \text{ q.e.d.}$$

Was ist eine Signatur?

- Bilden eines Hashes (Fingerabdruck) über Botschaft
- Verschlüsseln dieses Hashes mit dem privaten Schlüssel
- Empfänger entschlüsselt mit dem öffentlichen Schlüssel
- Empfänger vergleicht Hash mit dem selbst berechneten Hash.

4!

Signatur (Schema)



Symmetrische vs. Asymmetrische Verschlüsselung

	symmetrisch	asymmetrisch
⊖	shared secret, Schlüsselverteilungsproblem	bei großen Datenmengen sehr langsam
⊕	sehr schnell	kein Schlüsselverteilungsproblem

4!

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung**
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?

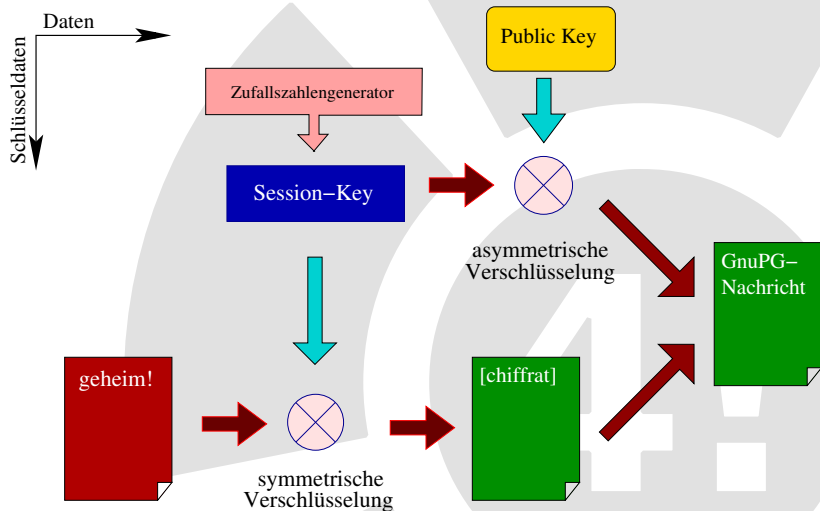


4!

hybrides Verfahren, kombiniert geschickt symmetrische und asymmetrische Algorithmen:

- zufälliger symmetrischer Schlüssel wird zur Verschlüsselung benutzt
 - dieser wird asymmetrisch verschlüsselt und mit der Nachricht verschickt
 - geheimer Schlüssel entschlüsselt den session key, damit kann der Empfänger die Daten lesen
- ⇒ man benötigt nur den öffentlichen Schlüssel des Kommunikationspartners
- ⇒ der geheime Schlüssel verlässt nie den Rechner (Spezialfall Chipkarte)

Schema GnuPG



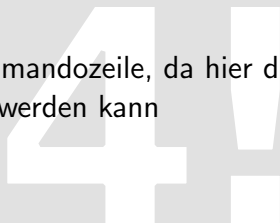
Geschichte (auch) von GnuPG

- PGP (Pretty Good Privacy) 1991 von Phil Zimmermann, zunächst Open Source, später (1997) Verkauf an NAI
- div. Inkompatibilitäten der PGP-Versionen führten 1998 zu RFC 2440
- ab 1998 Entwicklung von GnuPG (Gnu Privacy Guard), Förderung durch BMBF
- aktuelle Version(en) von GnuPG: 1.4.12 und 2.0.19
- seit geraumer Zeit Chipkartenunterstützung
- Open Source, diverse Portierungen

Auf den folgenden Folien wird gezeigt:

- erzeugen eines eigenen Schlüssels
- erzeugen eines Widerrufszertifikates
- Verschlüsseln einer Datei
- Entschlüsseln einer Datei

Alle Beispiele beziehen sich auf die Kommandozeile, da hier die größte Kontrolle über GnuPG ausgeübt werden kann

A large, light gray graphic consisting of a circle containing a white number '4' and an exclamation mark '!' to its right. The background of the slide features several overlapping light gray geometric shapes, including triangles and circles.

Wahl des Signatur- und/oder Verschlüsselungsalgorithmus

```
$ gpg --gen-key
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection?
```

- DSA: Digital Signature Algorithm, zum Signieren von Dateien (nicht DES!)
- ElGamal: El Gamal (Israel), ähnlich RSA
- RSA: Rivest, Shamir, Adleman, wie bereits vorgestellt

4!

Schlüssel erzeugen II

Wahl der Schlüssellänge

```
...  
RSA keys may be between 1024 and 4096 bits long.  
What keysize do you want? (2048)  
Requested keysize is 2048 bits  
...
```

- Man kann wohl davon ausgehen, dass 1024 Bit nicht mehr ausreichend sicher sind.
- Deshalb sollte eine Schlüssellänge von mindestens 2048 bit gewählt werden.
- Aber: das Arbeiten mit größeren Schlüsseln wird schnell sehr langsam. Hier ist wie immer zwischen Sicherheit und Komfort abzuwägen.

Schlüssel erzeugen II

Wahl der Gültigkeitsdauer

```
...  
Please specify how long the key should be valid.  
    0 = key does not expire  
    <n> = key expires in n days  
    <n>w = key expires in n weeks  
    <n>m = key expires in n months  
    <n>y = key expires in n years  
Key is valid for? (0)
```

- Gültigkeitsdauer hängt von eigenen Präferenzen ab.
- Oftmals wird ein Hauptschlüssel ohne feste Gültigkeit generiert, dann jeweils Unterschlüssel mit einjähriger Laufzeit.
- Bei Verlust der Passphrase und des Revocation Zertifikates verfällt der Schlüssel automatisch
- Partner werden so gezwungen, öfter auf Keyservern neu zu suchen

Schlüssel erzeugen III

Name, Kommentar, E-Mail-Adresse und Passphrase

```
...
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Florian Streibelt
Email address: florian@freitagsrunde.org
Comment: Freitagsrunde der TUB
You selected this USER-ID:
    "Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>"
```

- Eingabe von Namen und E-Mail-Adresse und Bestätigung der Angaben
- Dann Eingabe der Passphrase:
- kein „Wort“ aus dem Wörterbuch, möglichst ein verfremdeter Satz, z.B. „Ich habe den Linuxtag Chemnitz besucht.“
→ „1(h 4abe d3n L!nv)(tag Ch0mn:tz b?sucht+“

Schlüssel erzeugen IV

```
...
gpg: key F435050F marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   1024D/F435050F 2010-11-06
      Key fingerprint = B625 3451 AF4E F755 E407  FD88 A92B 359F F435 050F
uid           Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>
sub   2048g/6BB5B4FA 2010-11-06
```

Anzeige des Fingerprints am Ende der Schlüsselerzeugung:

- Der Fingerprint ist eine Prüfsumme, mit der die Integrität des Schlüssels geprüft werden kann.
- Die letzten 8 Byte sind die Key-ID, 16 Byte sind eindeutiger.

Widerrufszertifikat erzeugen

```
$ gpg --gen-revoke F435050F
sec 1024D/F435050F 2010-11-06
  Florian Streibelt (Freitagrunde der TUB) <florian@freitagrunde.org>
Create a revocation certificate for this key? (y/N)
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision?
```

Vorbeugung vor kompromittierten Schlüsseln und vergessenen
Passwörtern:

- Widerrufszertifikat ausdrucken
- an sicherem Ort aufbewahren
- möglichst nicht mit Arial drucken: ll = iL
(besser Times, Comic Sans, ...)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: GnuPG v1.4.1 (GNU/Linux)  
Comment: A revocation certificate should follow  
  
iFwEIBECABwFAkNuZYQVHQBQYXNzcGhyYXNIIHZlcmxvcmVuAAoJEKkrNZ/0NQUP  
mUsAniYuZuEp+Vrvhgrj/8KTTcBE0x4bAJ0YYyczU7zQlcJd7YsX00qGt12NSA==  
=F8X3  
-----END PGP PUBLIC KEY BLOCK-----
```

Vorsicht beim Ausdruck:

- Das Drucksystem könnte die Datei speichern,
- eventuell sogar der (Netzwerk-) Drucker selbst
- jeder mit Zugriff auf diese „Zeichenkette“ kann den Schlüssel unbrauchbar machen

Beispiel: Verschlüsseln

```
$ echo "Geheime Mitteilung" > geheim.txt
$ gpg -aer F435050F geheim.txt
  ls -l geheim*
-rw-r--r--  1 fls users  19 Nov  6 21:25 geheim.txt
-rw-r--r--  1 fls users 932 Nov  6 21:25 geheim.txt.asc

$ cat geheim.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.1 (GNU/Linux)

hQIOA4C2E61rtbT6EAf+PXrzs5CwHhRXsb3UK+5wgnV4eLYUDv0dVag5akMj+nvU
ekEBb0ucjZN20HKb9wgKAXo3+HmkiG4IYEIQm5d2wp6He6wRUo2DoxZR1Gmfczoy
sQFC/fGWAq/wclgKiyEyYCECao0R6Y8PrNS0Zyybq+5XJOH4n8ybDJFTo6FqIYfb
uTZTCxtRqpB7T5MUMt0hUJTX3LNF46QtIOxNXhmeYM9pd5AJMmXkPbwSw1CsHvdG
gbGajMmoewYswSD5WMeB66dxtlQQxZfQNRMDn/9i6Kpfj0rh/WBf/3NrPsoy04y6
l7lpR/eeeGyBooiY2ti0/rNimXipW3zh3bgejnpzAAGAmFwOyE5aJlelBVQ5EFxW
et1VutRo7hVjAimWNPxDK8pJUDxJ2P0h9Kbe/Tq6tV+yp5viwnl3Ec5F3YqWpVaC
ebdlgh7mxAvsz4266hGFZsClmvl9UqBeKd7fIZxhYlpG/+omgTDWyye8gA5kUZg3
O/v7ic2xHeDsHoxOYvol9cGteelgj+eBBCsZT79Nhy5ZoCvrouBaeLh3GNDInrs
nFvjGKAIG8BMPRofdD6BkGix+pp5pLkDOrVfVMgqdlIEHpoh0KXzO9l4q5Bux+EC
bMltOcTX0lr1rxAVqqE0qqKCLLiHZ8TJ1fh6polmOTryQAwimP3b8uNiB9NeT7sW
H9JVAVCNf3FUMmac1TR4/JmfPNjI41k4yQ84N5fJmcbgbbmjgvLk75RhpEnLqaxrG
PnDx6jhqQ+zUSznwrg67KflqCLUzPKnTTIxc6d1rzoWVhKAEiTEBw==
=>XPN
-----END PGP MESSAGE-----
```

Beispiel: Entschlüsseln

```
$ cat geheim.txt.asc | gpg
```

```
You need a passphrase to unlock the secret key for  
user: "Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>"  
2048-bit ELG-E key, ID 6BB5B4FA, created 2010-11-06 (main key ID F435050F)
```

```
gpg: encrypted with 2048-bit ELG-E key, ID 6BB5B4FA, created 2010-11-06  
"Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>"  
Geheime Mitteilung
```

alternativ, direkt in eine Datei:

```
$ gpg geheim.txt.asc
```

```
You need a passphrase to unlock the secret key for  
user: "Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>"  
2048-bit ELG-E key, ID 6BB5B4FA, created 2010-11-06 (main key ID F435050F)
```

```
gpg: encrypted with 2048-bit ELG-E key, ID 6BB5B4FA, created 2010-11-06  
"Florian Streibelt (Freitagsrunde der TUB) <florian@freitagsrunde.org>"
```

Features von GnuPG

- Chipkartenunterstützung: OpenPGP-Card
- Foto-IDs: eingebettete jpeg-Dateien
- Vorsicht: bitte kleine Dateien benutzen
- mehrere uid's Pro Schlüssel
- pro Emailadresse eine User-ID
- mehrere Schlüssel
- werden alle zusammen signiert
- z.B. für unsichere Umgebungen



4!

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning**
- 8 Tools
- 9 Hardware...?



4!

Das Problem

- Jeder kann Schlüssel für beliebige Namen und Emailadressen erzeugen.
- Viele falsche Schlüssel existieren.
- Eine Suche nach Bill Gates ergibt bereits einige hundert Treffer

Die Lösung: Keysigning

- Inhaber von PGP-Schlüsseln treffen sich und verifizieren ihre Identitäten.

Keysigning

- Beim Keysigning überprüft man den Fingerprint des Schlüssels
 - und die user-id (den Namen) anhand eines Ausweises
 - Danach signiert man den Schlüssel des anderen lokal auf dem eigenen Rechner
 - Idealerweise signiert man jede uid auf dem Schlüssel einzeln
 - Nun schickt man den signierten Schlüssel verschlüsselt an die angegebene(n) Emailadresse(n)
 - Damit sind nun auch die Emailadressen verifiziert!
 - Es gibt scripte dafür: `caff`, oder `mycaff` ¹
 - das Web of Trust lebt, wie der Name sagt, vom Vertrauen:
- ⇒ bitte niemals blind signieren, nur nach persönlichem Treffen!

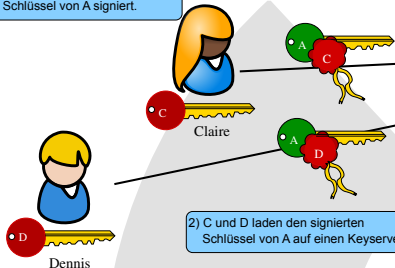
¹Eigenwerbung: <http://user.cs.tu-berlin.de/~mutax/ksp-scripts/mycaff/>

Web of Trust

- Hat man ein genügend großes Web of Trust, muss nicht mehr jeder jeden Schlüssel signieren!
- Jedem öffentlichen Schlüssel anderer Personen kann man verschiedene Vertrauensstufen zuweisen
- Bekommt man eine Nachricht von einem bisher unbekanntem Schlüssel kann man prüfen, ob eine gewisse Anzahl vertrauenswürdiger User diesen unterschrieben haben
- ist dies der Fall, kann man davon ausgehen, dass die Identität genügend geprüft und sicher ist.
- wichtiges Hilfsmittel: Keyserver, die öffentliche Schlüssel und Wiederrufsurkunden vorhalten

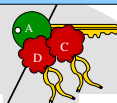
Schema Web of Trust

1) C und D haben den öffentlichen Schlüssel von A signiert.

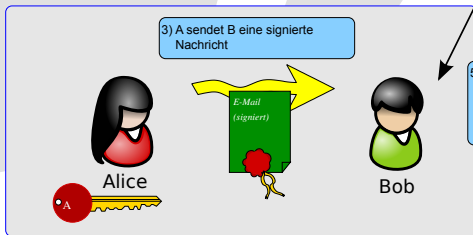


2) C und D laden den signierten Schlüssel von A auf einen Keyserver

4) B ruft den Schlüssel ab und importiert die Signaturen von C und D



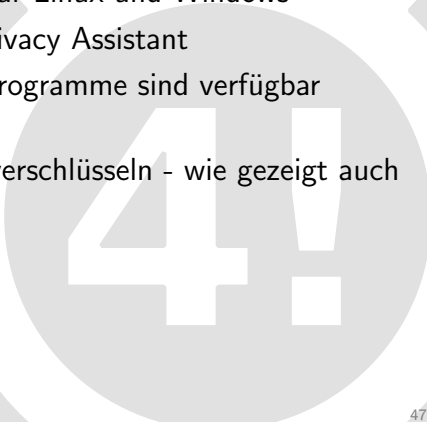
3) A sendet B eine signierte Nachricht



5) B vertraut C und D, deshalb auch ihren Unterschriften auf dem Schlüssel von A. Damit akzeptiert er die Signatur in der Email

rot: private Key
grün: public Key

- auf der Kommandozeile hat man die größte Kontrolle
- es gibt einige grafische Tools für Linux und Windows™
- unter Linux: gpa - den Gnu Privacy Assistant
- Plugins für die meisten Emailprogramme sind verfügbar (Thunderbird, Evolution)
- GnuPG kann beliebige Daten verschlüsseln - wie gezeigt auch lokale Dateien



4!

GNU Privacy Assistant - Schlüsselverwaltung

Datei Bearbeiten Schlüssel Server Fenster Hilfe

Ändern Löschen Signieren Import Export Übersicht Details Dateien

Schlüsselverwaltung

▲	Schlüsselkennung	Verfallsdatum	Benutzervertrauen	Gültigkeit des Schlüssels	Benutzerkennung
🔑	05E281DE	19.09.2005	Ultimativ	Abgelaufen	Florian Streibelt (interActive-Systems) <florian.streibelt@interActive
🔑	AC804673	kein Verfallsdatum	Ultimativ	Widerrufen	[Widerrufen] Florian Streibelt <mutax@ringworld.org>
🔑	731F33F6	31.10.2004	Ultimativ	Widerrufen	[Widerrufen] Florian Streibelt <florian.streibelt@brainMedia.de>
🔑	1079C4D9	21.09.2003	Ultimativ	Widerrufen	[Widerrufen] Florian Streibelt (key for business mail) <Florian.Stre
🔑	C7C84CE1	18.09.2005	Ultimativ	Abgelaufen	Florian Streibelt (ringworld) <mutax@ringworld.org>
🔑	D34EE3D7	23.10.2004	Ultimativ	Abgelaufen	Florian Streibelt (Technische Universität Berlin, FB Informatik/Fak IV)
🔑	82F61240	21.10.2010	Ultimativ	voll gültig	Florian Streibelt
🔑	CE68DFEC	26.01.2010	Ultimativ	voll gültig	Florian Streibelt

Details | Signaturen | Untergeordnete Schlüssel

Dieser Schlüssel hat einen öffentlichen und einen geheimen Teil
Der Schlüssel kann zur Zertifizierung, zum Signieren und zur Verschlüsselung verwendet werden.

Benutzerkennung: Florian Streibelt:
 Florian Streibelt (general purpose key) <Florian.Streibelt@TU-Berlin.DE>
 Florian Streibelt (Usenet ONLY) <news@F-Streibelt.de>
 Florian Streibelt (some other account) <mutax@ringworld.org>
 Florian Streibelt (business key) <florian.streibelt@interActive-Systems.de>
 Florian Streibelt <mutax@cs.tu-berlin.de>

Schlüsselkennung: 82F61240
 Fingerabdruck: 5BE7 F008 8B83 9357 1108 984A 3B8E A41F 82F6 1240
 ungültig ab: 21.10.2010

Benutzervertrauen: Ultimativ
 Gültigkeit: voll gültig
 Art: DSA 1024 bit
 erzeugt am: 22.10.2004

Standard-Schlüssel: Florian Streibelt (interActive-Systems) <florian.streibelt@interActive-Systems.de> 05E281DE

GNU Privacy Assistant - Schlüsselverwaltung

Datei Bearbeiten Schlüssel Server Fenster Hilfe

Ändern Löschen

Schlüssel

Schlüsselkennung

- 05E281DE
- 1073C4D9
- 731F93F6
- AC804673
- C7C84CE1
- D34EE3D7

Details Signaturen

Schlüssel erzeugen

Verschlüsselungsalgorithmus: DSA und ElGamal (Voreinstellung)

Schlüssellänge (Bit): 1024

Benutzerkennung:

E-Mail:

Kommentar:

Passwortsatz:

Passwortsatz wiederholen:

Verfallsdatum:

unbegrenzt gültig

verfällt nach Tagen

wird ungültig am:

← Oktober 2005 →

Mo	Di	Mi	Do	Fr	Sa	So
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

OK Abbrechen

Benutzerkennung

- Florian Streibe
- [Widerrufen] F
- [Widerrufen] F
- [Widerrufen] F
- Florian Streibe
- Florian Streibe
- [Widerrufen] E
- [Widerrufen] E
- [Widerrufen] M

Standard-Schlüssel: Florian Streibelt (interActive-Systems) <florian.streibelt@interActive-Systems.de> 05E281DE

OpenPGP Key Management

File Edit View Keyserver Generate

Filter for user ID's or key ID's containing:

Account / User ID	Key ID	Type	Calculated Trust	Owner Trust	Expiry	
Florian Streibelt	CE68DFEC	pub/sec	ultimate	ultimate	01/26/2010	
Florian Streibelt	82F61240	pub/sec	ultimate	ultimate	10/21/2010	
- Florian Streibelt (gen...)			ultimate	ultimate	10/21/2010	
- Florian Streibelt (Use...)			ultimate	ultimate	10/21/2010	
- Florian Streibelt (som...)			ultimate	ultimate	10/21/2010	
- Florian Streibelt (busi...)			ultimate	ultimate	10/21/2010	
- Florian Streibelt <mut...>			ultimate	ultimate	10/21/2010	
- User attribute (JPEG i...)				ultimate	10/21/2010	
<i>Florian Streibelt (inter... 05E281DE</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>expired</i>	<i>ultimate</i>	<i>09/19/2005</i>	
<i>Florian Streibelt (key ... 1073C4D9</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>disabled</i>	<i>ultimate</i>	<i>09/21/2003</i>	
<i>Florian Streibelt (ring... C7C84CE1</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>expired</i>	<i>ultimate</i>	<i>09/18/2005</i>	
<i>Florian Streibelt (Tec... D34EE3D7</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>expired</i>	<i>ultimate</i>	<i>10/23/2004</i>	
<i>Florian Streibelt <flor... 731F33F6</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>revoked</i>	<i>ultimate</i>	<i>10/31/2004</i>	
<i>Florian Streibelt <mut... AC804673</i>	<i>pub/sec</i>	<i>pub/sec</i>	<i>revoked</i>	<i>ultimate</i>		

- 1 Vorschau
- 2 Ziele der Verschlüsselung
- 3 Bestandsaufnahme Email
- 4 Geschichte
- 5 Grundlagen von Verschlüsselung
- 6 Praktische Anwendung
- 7 Hintergründe Keysigning
- 8 Tools
- 9 Hardware...?



4!



Abbildung: scr335 mit FSFE-Fellowship Karte ²

- externer Kartenleser ohne Pinpad
- Chipkarte mit privatem Schlüssel
- asymmetrische Verschlüsselung im Chip

²Quelle: <http://gnupg.org>

Hardware - Cardreader mit Pinpad, Chipkarte



Abbildung: Kobil-Reader Klasse 2 mit FSFE-Fellowship Karte ³

- externer Kartenleser mit Pinpad
- Chipkarte mit privatem Schlüssel
- asymmetrische Verschlüsselung im Chip

³Quelle: <http://gnupg.org>



Abbildung: GPFCryptoStick, Version 2 ⁴

- included OpenPGP smart card
- encrypted MicroSD storage
- ARM-processor, USB interface
- symmetric encryption in IC (e.g. AES-256)
- platform independent PIN interface

⁴Quelle: https://www.privacyfoundation.de/wiki/GPFCryptoStick#Version_2

Download der Folien:

[http://docs.freitagrunde.org/Veranstaltungen/
keysigning/2012/](http://docs.freitagrunde.org/Veranstaltungen/keysigning/2012/)

Nicht vergessen: Keysigning-Party kommenden Freitag!

ANMELDUNG:

<http://keysigning-fr12.streibelt.net/>

Anleitungen:

<http://wiki.freitagrunde.org/Keysigning-Party>

**GG, Art.10, Abs1:
Das Briefgeheimnis
sowie das Post- und
Fernmeldegeheimnis
sind unverletzlich...**

=cSNN

-----END PGP MESSAGE-----