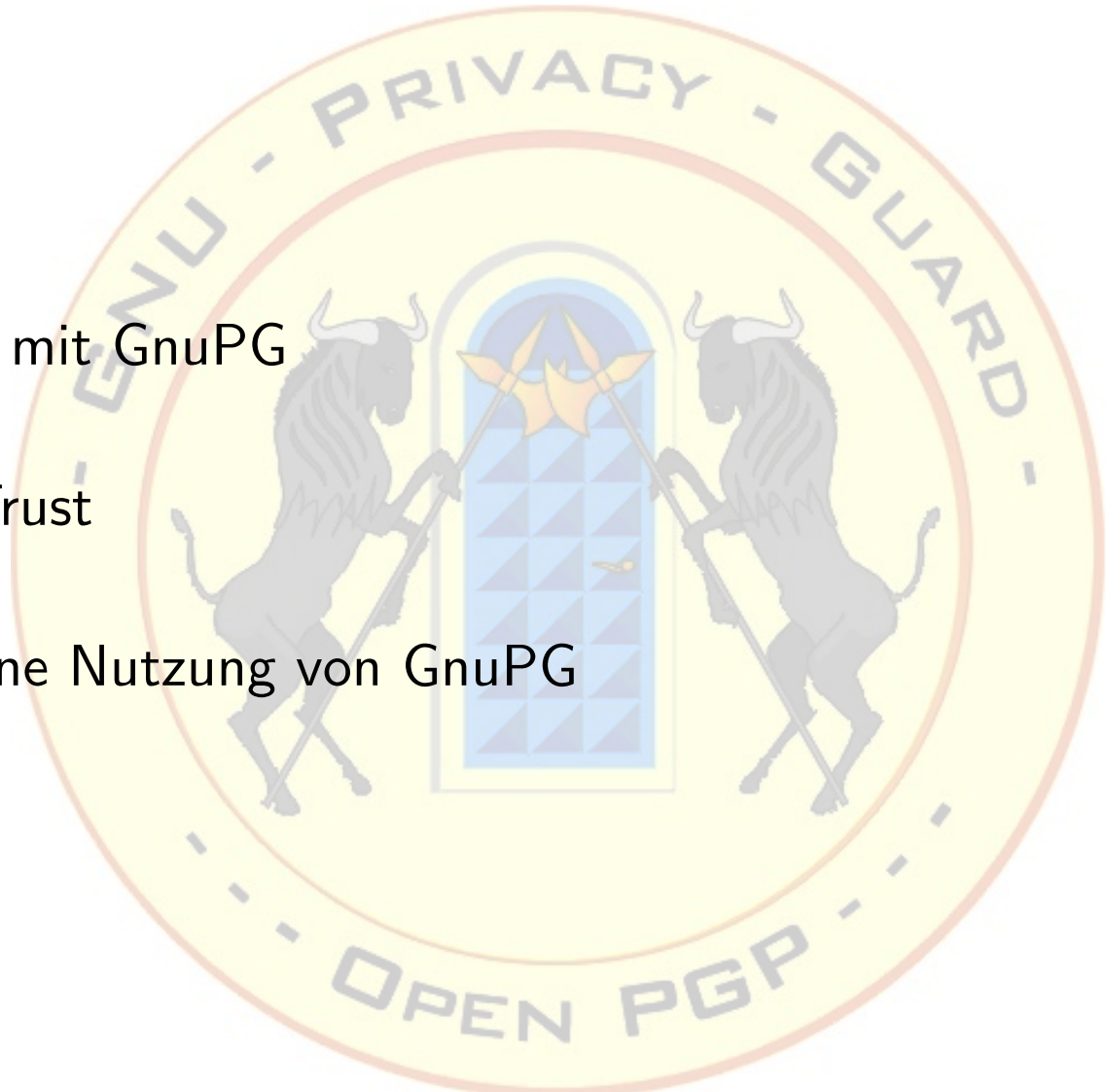
GnuPG - mit Sicherheit im Internet

Wie sicher bist Du, daß ich ich bin?

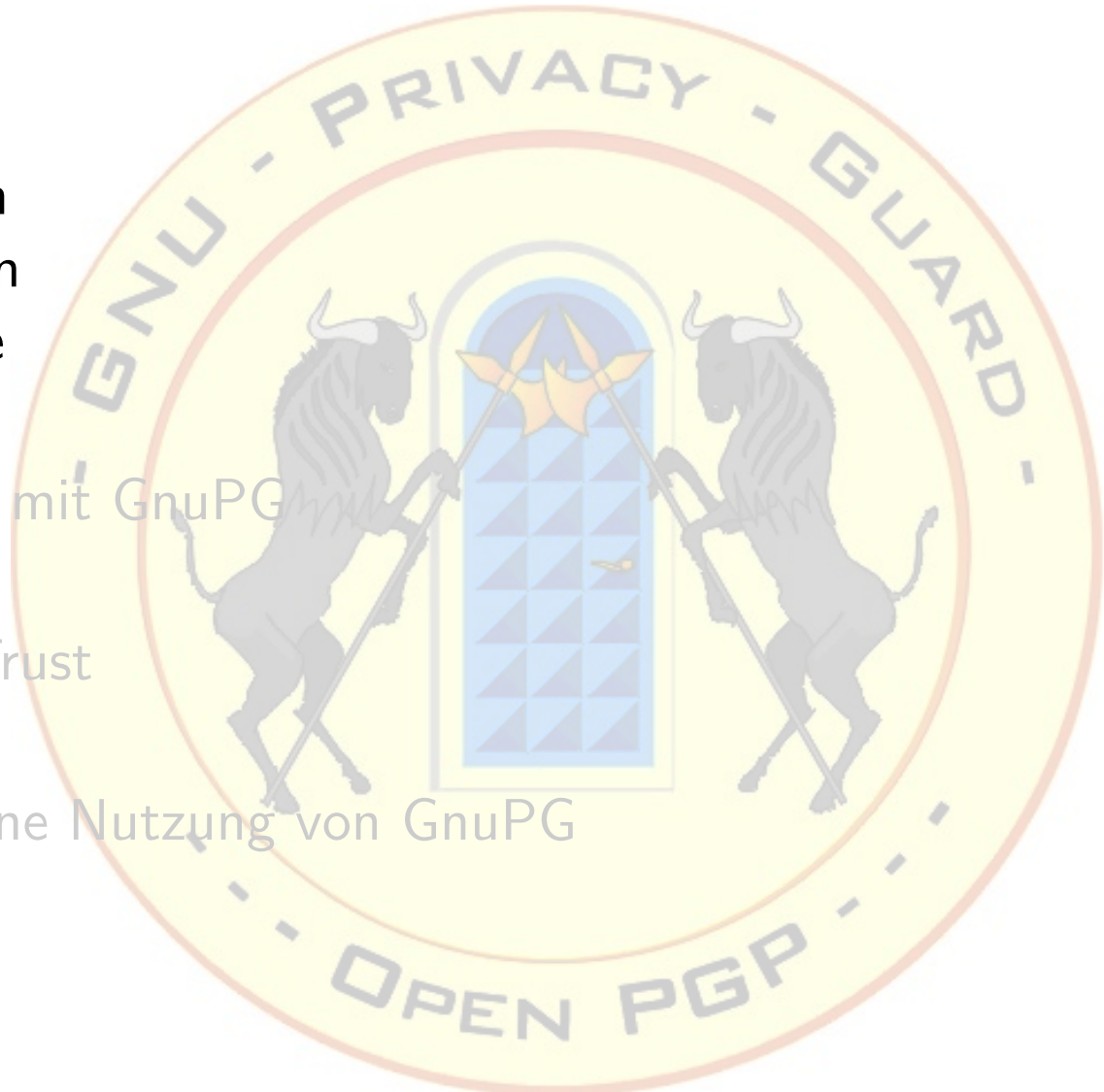
Elmar Hoffmann <elho@elho.net>

Linux User Group Mönchengladbach
14. November 2006

Übersicht



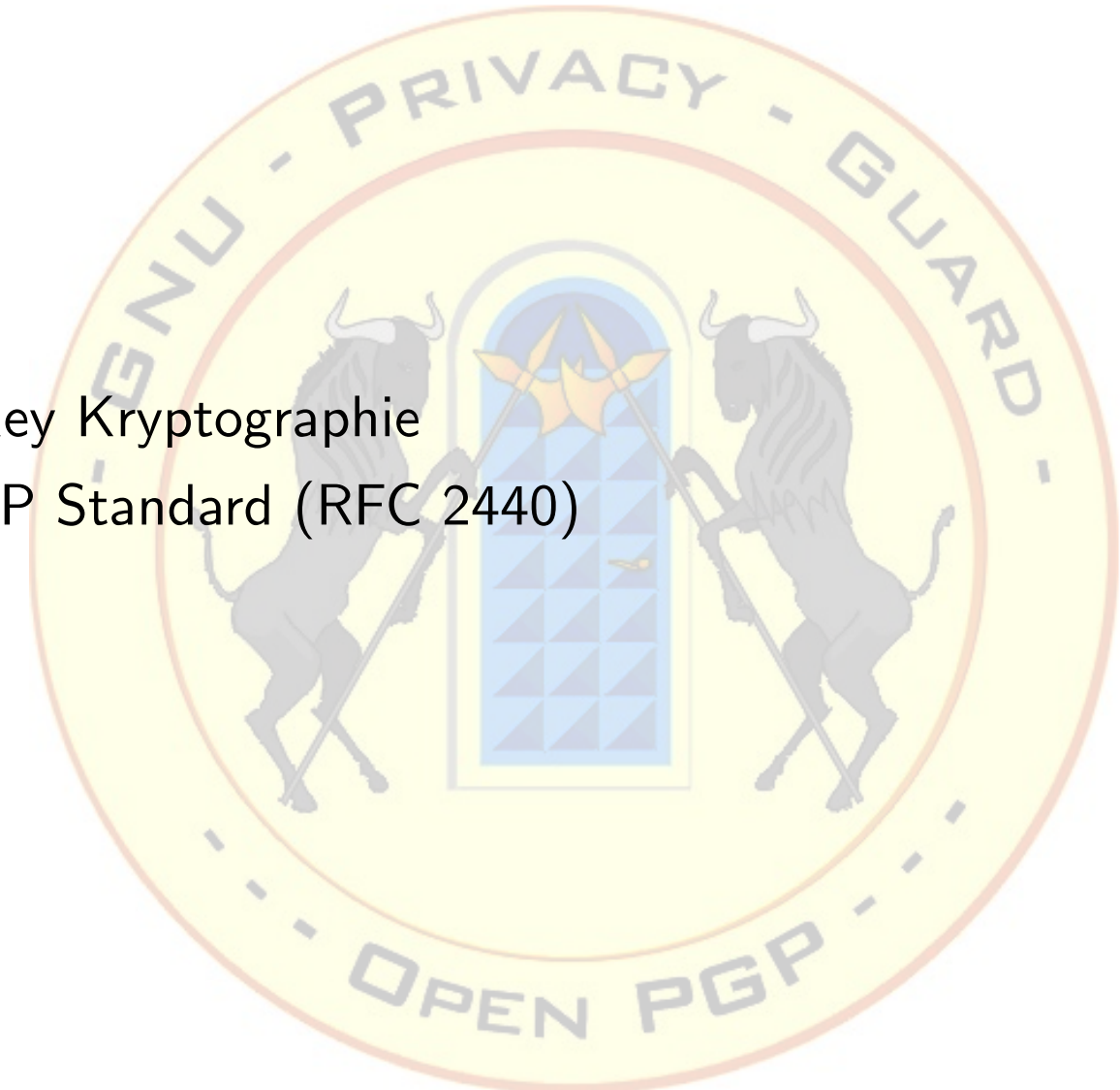
Übersicht



Motivation

- Signatur und Verschlüsselung von Emails
  - Emails sind Postkarten, keine Briefe
- Signatur von Software

Grundlagen



- Public Key Kryptographie
- OpenPGP Standard (RFC 2440)

Geschichte

- 1991: PGP
- 1997: GnuPG
- 1998: OpenPGP Standard
- 1999: GnuPG 1.0.0
- 2002: GnuPG 1.2.0
- 2004: GnuPG 1.4.0

Übersicht

Keys erzeugen

```
$ gpg --gen-key
gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?
```

Keys erzeugen

```
$ gpg --gen-key
gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?

DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

Keys erzeugen

```
$ gpg --gen-key
gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection?

DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)

Requested keysize is 2048 bits
```

Keys erzeugen II

```
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)
```

Keys erzeugen II

```
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)

Key does not expire at all
Is this correct? (y/N) y
```

Keys erzeugen III

```
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: John Doe
```

Keys erzeugen III

```
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: John Doe

Email address: jd@example.com
```

Keys erzeugen III

```
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: John Doe

Email address: jd@example.com

Comment:
```

Keys erzeugen III

```
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: John Doe

Email address: jd@example.com

Comment:

You selected this USER-ID:
    "John Doe <jd@example.com>"

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

Keys erzeugen III

```
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: John Doe

Email address: jd@example.com

Comment:

You selected this USER-ID:
    "John Doe <jd@example.com>"

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

You need a Passphrase to protect your secret key.

Enter passphrase:
Repeat passphrase:
```

Keys erzeugen IV

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.++++++++++.+++++.++++++++++.+++++..++++++++++.++++++++++++++++++++++++.+++++.+
++++++++.++++++++++++++++++++++++.++++++++++++++.++++++++++++++.++++++++++>+
++++++++>+++++.....>+++++....+++++^^^
gpg: key 43E1FB1F 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/43E1FB1F 2005-11-30
      Key fingerprint = CBCA 9EC7 5965 1172 B3F9  8DB1 A2F6 4C47 43E1 FB1F
uid                  John Doe <jd@example.com>
sub   2048g/0B2754E4 2005-11-30
```

Revocation Certificates erzeugen

```
$ gpg --gen-revoke 43E1FB1F >43E1FB1F_compromised.asc

sec  1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com>

Create a revocation certificate for this key? (y/N) y
```

Revocation Certificates erzeugen

```
$ gpg --gen-revoke 43E1FB1F >43E1FB1F_compromised.asc

sec  1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com>

Create a revocation certificate for this key? (y/N) y

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? 1
```

Revocation Certificates erzeugen

```
$ gpg --gen-revoke 43E1FB1F >43E1FB1F_compromised.asc

sec  1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com>

Create a revocation certificate for this key? (y/N) y

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? 1

Enter an optional description; end it with an empty line:
>
```

Revocation Certificates erzeugen

```
$ gpg --gen-revoke 43E1FB1F >43E1FB1F_compromised.asc

sec  1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com>

Create a revocation certificate for this key? (y/N) y

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? 1

Enter an optional description; end it with an empty line:
>

Reason for revocation: Key has been compromised
(No description given)
Is this okay? (y/N) y
```

Revocation Certificates erzeugen II

```
You need a passphrase to unlock the secret key for
user: "John Doe <jd@example.com>"
1024-bit DSA key, ID 43E1FB1F, created 2005-11-30

ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
```

Revocation Certificates erzeugen III

```
$ cat 43E1FB1F_compromised.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.2 (GNU/Linux)
Comment: A revocation certificate should follow

iEkEIBECAAkFAkOTbXACHQIACgkQovZMROPh+x9N6ACfa+SIW0qaUJ+VE4nTHjG9
YPYLzd0AoItF4fPs/2oF1axp+4az75yr/Cyw
=enCG
-----END PGP PUBLIC KEY BLOCK-----

$ gpg 43E1FB1F_compromised.asc
gpg: standalone signature of class 0x20
gpg: Signature made Sun 04 Dec 2005 11:28:00 PM CET using DSA key ID 43E1FB1F
gpg: standalone revocation - use "gpg --import" to apply
```

Nach Benutzung den Key wieder auf Keyserver hochladen!

Daten signieren

```
$ gpg --detach-sign --armor foo.tar.gz

You need a passphrase to unlock the secret key for
user: "John Doe <jd@example.com>"
1024-bit DSA key, ID 43E1FB1F, created 2005-11-30

Enter passphrase:
```

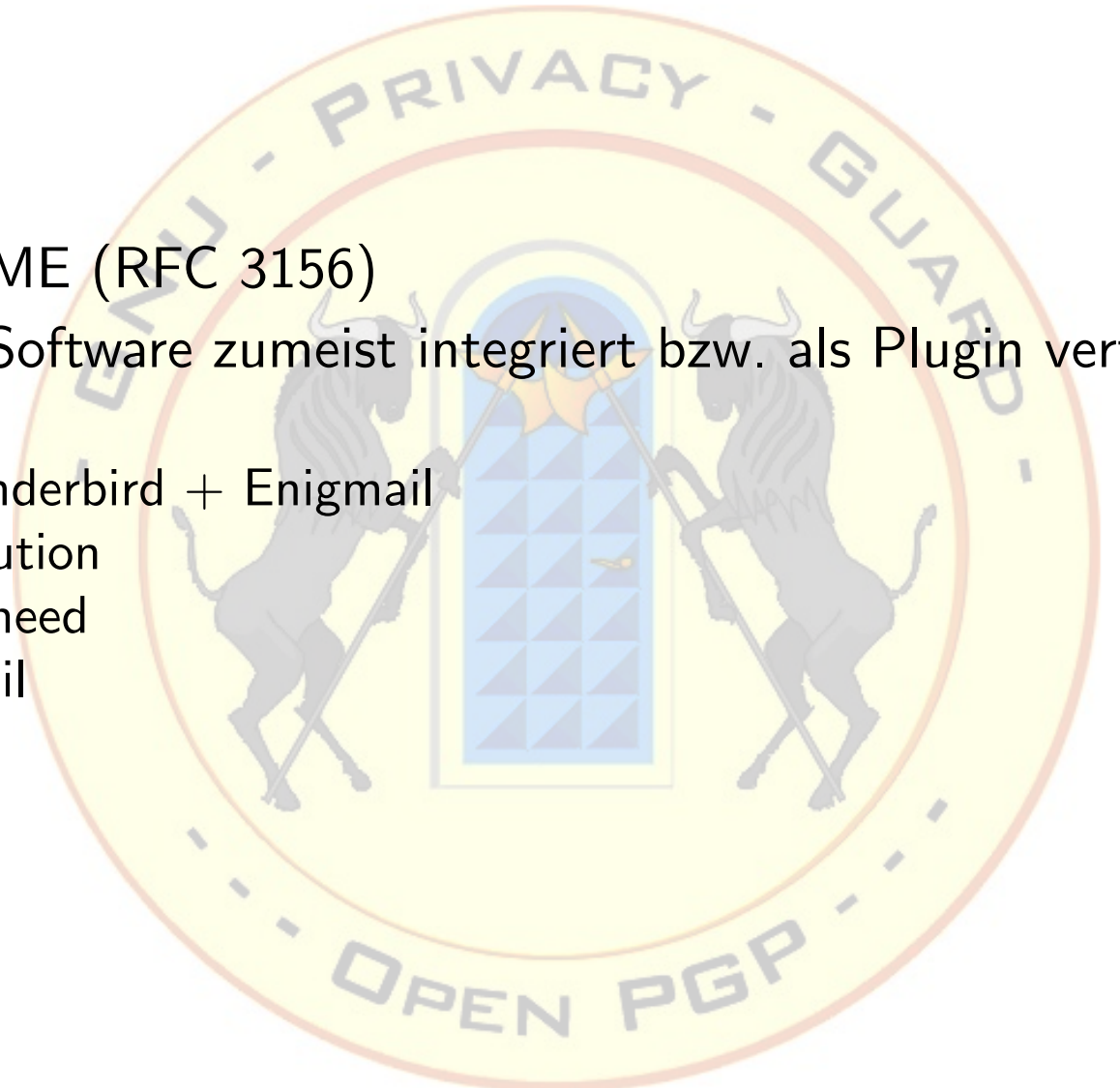
Daten signieren

```
$ gpg --detach-sign --armor foo.tar.gz

You need a passphrase to unlock the secret key for
user: "John Doe <jd@example.com>"
1024-bit DSA key, ID 43E1FB1F, created 2005-11-30

Enter passphrase:

$ ls foo.tar.gz*
foo.tar.gz  foo.tar.gz.asc
```

Daten signieren

```
$ gpg --detach-sign --armor foo.tar.gz

You need a passphrase to unlock the secret key for
user: "John Doe <jd@example.com>"
1024-bit DSA key, ID 43E1FB1F, created 2005-11-30

Enter passphrase:

$ ls foo.tar.gz*
foo.tar.gz  foo.tar.gz.asc

$ gpg foo.tar.gz.asc
gpg: Signature made Sat 10 Dec 2005 01:09:44 PM CET using DSA key ID 43E1FB1F
gpg: Good signature from "John Doe <jd@example.com>"
```

Email signieren / verschlüsseln



- PGP/MIME (RFC 3156)
- In freier Software zumeist integriert bzw. als Plugin verfügbar
  - mutt
  - Thunderbird + Enigmail
  - Evolution
  - Sylpheed
  - KMail

  ...

Keys signieren

- “Keys signieren” ist genau genommen falsch!
  - Richtig: User IDs signieren
- Signatur bestätigt, daß die Identität des Key-Besitzers mit der User ID übereinstimmt
  - Dies beinhaltet alle Komponenten der User ID!

Keys signieren II

```
$ gpg --sign-key CF3401A9

pub  4096R/CF3401A9  created: 2005-02-17  expires: never       usage: CS
                     trust: full          validity: unknown
[ unknown] (1). Elmar Hoffmann <elho@elho.net>


pub  4096R/CF3401A9  created: 2005-02-17  expires: never       usage: CS
                     trust: full          validity: unknown
 Primary key fingerprint: 8736 FE21 A2DF DDC9 8E5A  AD73 9579 52D7 CF34 01A9

     Elmar Hoffmann <elho@elho.net>

Are you sure that you want to sign this key with your
key "John Doe <jd@example.com>" (43E1FB1F)

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

Keys signieren II

```
You need a passphrase to unlock the secret key for
user: "John Doe <jd@example.com>"
1024-bit DSA key, ID 43E1FB1F, created 2005-11-30

Enter passphrase:
```

Anschließend den signierten Key wieder auf Keyserver hochladen!

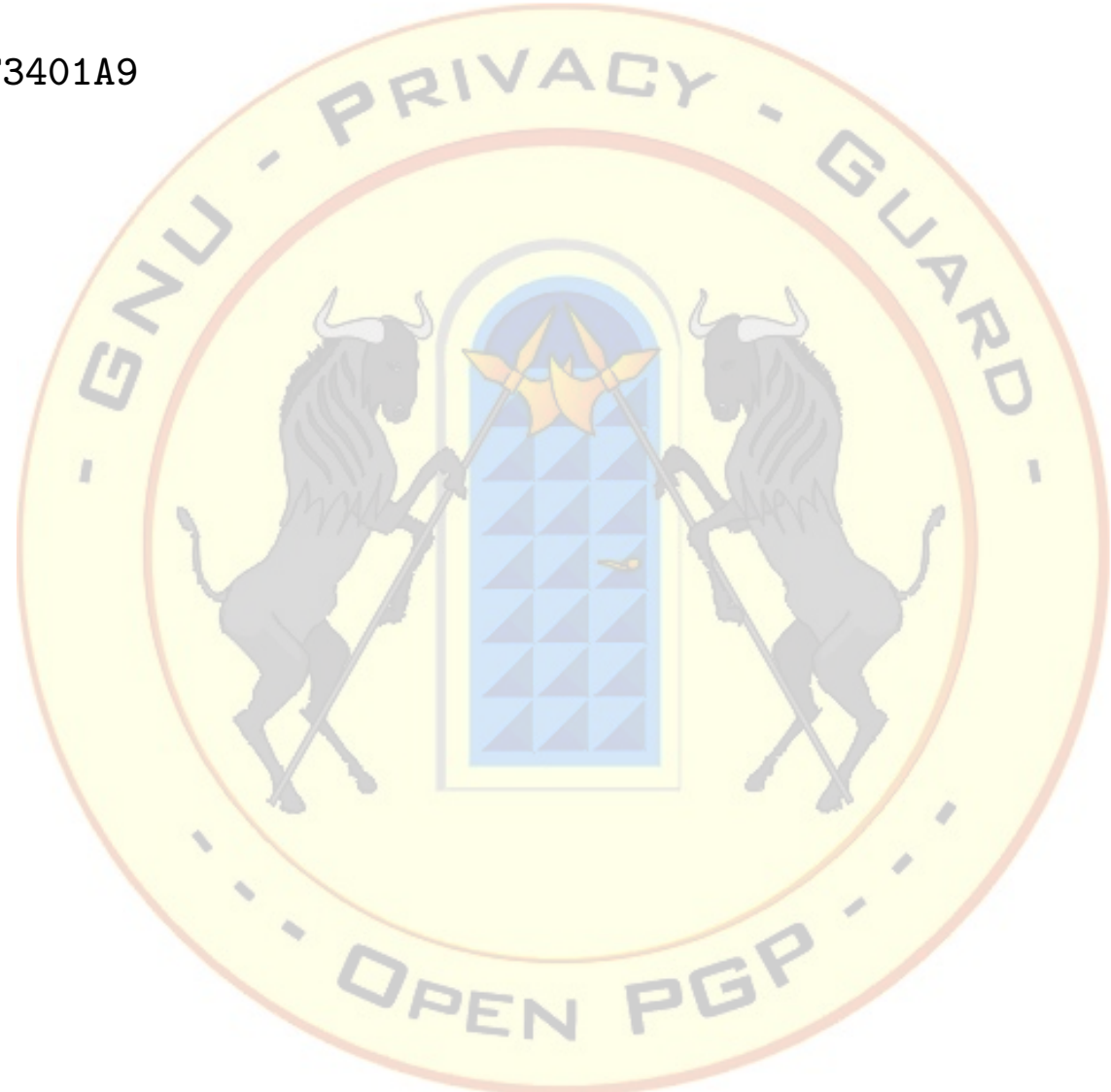
Übersicht

Trust

- Persönliches verifizieren der Identität beliebiger Personen nicht praktikabel
- Lösung: Signaturen von vertrauenswürdigen Personen zur Verifikation hinzuziehen
  - John Doe signiert den Key von Alice
  - Alice signiert den Key von Bob
  - Wenn John Doe Alice vertraut, ist auch der Key von Bob für John Doe gültig

Trust II

```
$ gpg --edit CF3401A9

Command> trust
```



Trust II

```
$ gpg --edit CF3401A9

Command> trust

pub  4096R/CF3401A9  created: 2005-02-17  expires: never       usage: CS
                     trust: unknown       validity: unknown
[ unknown] (1). Elmar Hoffmann <elho@elho.net>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 4
```

Trust III

```
pub  4096R/CF3401A9  created: 2005-02-17  expires: never       usage: CS
                     trust: full          validity: unknown
[ unknown] (1). Elmar Hoffmann <elho@elho.net>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

Command> quit
```

Trust III

```
pub  4096R/CF3401A9  created: 2005-02-17  expires: never       usage: CS
                     trust: full          validity: unknown
[ unknown] (1). Elmar Hoffmann <elho@elho.net>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

Command> quit

$ gpg --check-trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:   1  signed:   1  trust: 0-, 0q, 0n, 0m, 1f, 0u
gpg: depth: 2  valid:   1  signed:   0  trust: 1-, 0q, 0n, 0m, 0f, 0u
```

Keyserver

- Zentrale Quelle für Keys
- Leider doch nicht so einfach
  - Viele veraltete Server mit diversen Problemen
  - Lösung: subkeys.pgp.net

```
$ gpg --recv-keys CF3401A9
gpg: requesting key CF3401A9 from hkp server subkeys.pgp.net
gpg: key CF3401A9: public key Ëlmar Hoffmann <elho@elho.net>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Web of Trust

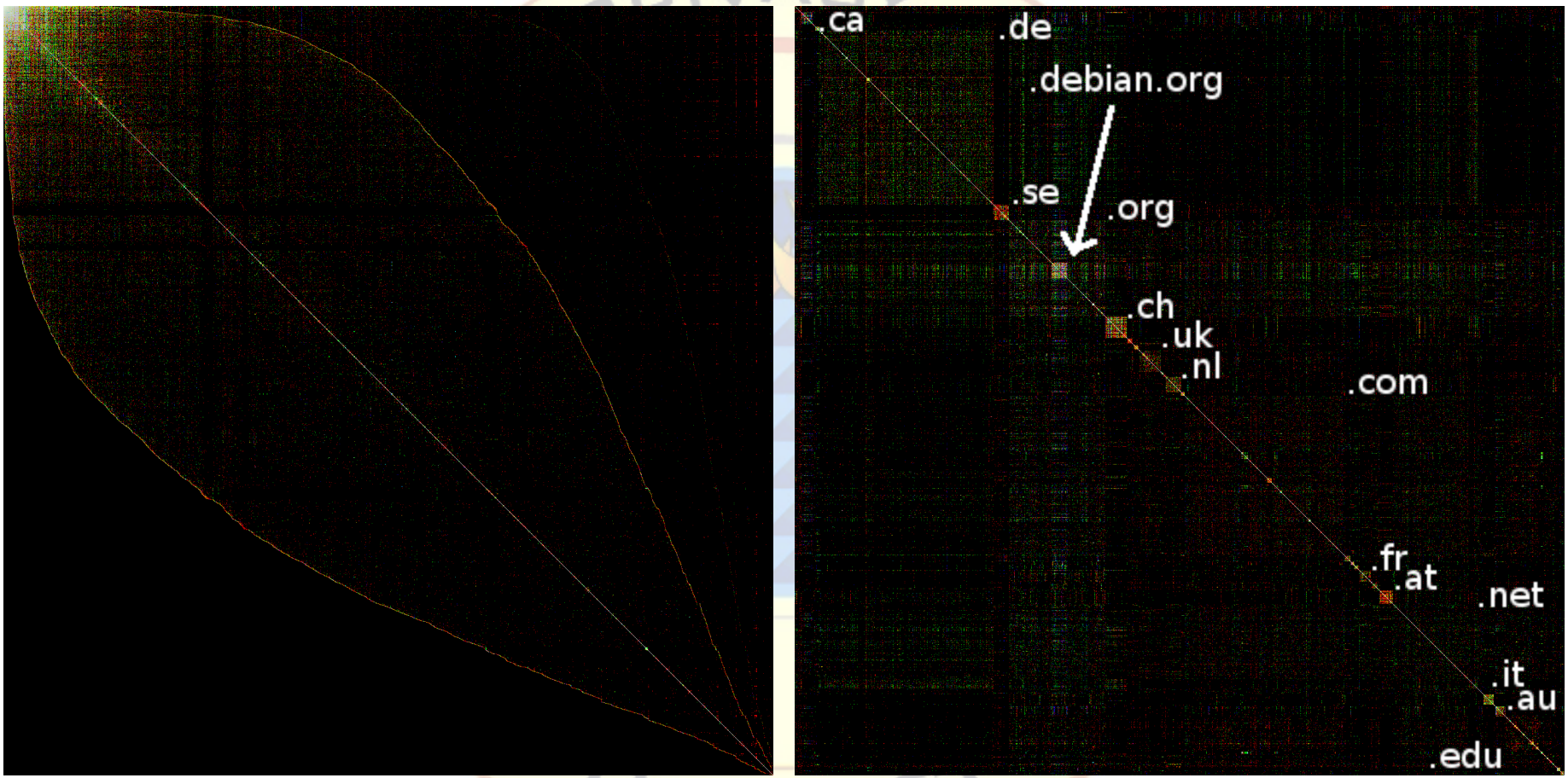
- Netz von Keys, die einander signiert haben
- Das “Strong Set”
  - Das größte zusammenhängende Netz solcher Art
  - Von jedem Key in diesem Netz existiert ein Pfad von Signaturen zu jedem anderen Key
  - “Mean Shortest Distance” (MSD)
    - mittlere Distanz von allen anderen Keys zu diesem
  - Wie findet man einen Pfad von/zu sich?
    - PGP Pathfinder & key statistics:
      `http://www.cs.uu.nl/people/henkp/henkp/pgp/pathfinder/`
- Biglumber - key signing coordination

  `http://www.biglumber.com/`

Dissecting the Leaf of Trust

- Adjazenzmatrix
  - Key auf der X-Achse hat Key auf der Y-Achse signiert
- 24000 Keys im Strong Set zu der Zeit
- “Bedeutende” Keys sind deutlich sichtbar
  - c’t / Heise Krypto-Kampagne
  - Debian
- `http://www.lysator.liu.se/~jc/wotsap/leafoftrust.html`

Dissecting the Leaf of Trust II



Übersicht

Subkeys

- In aller Regel hat man einen, ohne es wissen zu müssen

```
pub   1024D/43E1FB1F 2005-11-30
uid                  John Doe <jd@example.com>
sub   2048g/0B2754E4 2005-11-30
```

- Using multiple subkeys in GPG

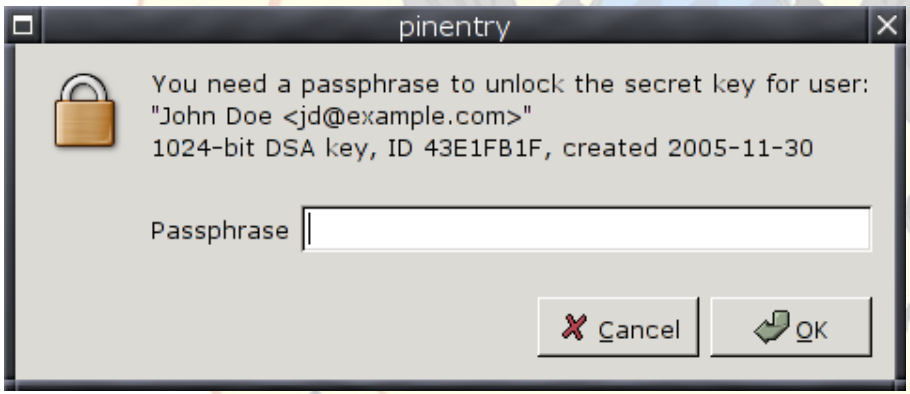
  `http://fortytwo.ch/gpg/subkeys`

- Moving keys and subkeys - How to migrate a (sub)key into a new key

  `http://atom.smasher.org/gpg/`

GPG-Agent

- Cache für Passphrase(s)
  - Vergleichbar mit dem ssh-agent
- Passphrase-Dialog mittels pinentry
  - pinentry-curses
  - pinentry-gtk2



  - pinentry-qt
  - pinentry-x11

GPG-Agent Quickstart

- gpg-agent und pinentry installieren
- `use-agent` in `~/.gnupg/gpg.conf` aktivieren
- gpg-agent starten (`~/.bash_profile`, Xsession)
  - `eval `gpg-agent --daemon``
  - keychain

    `http://www.gentoo.org/projects/keychain/`

gpg-key2ps

- Erzeugt Ausdruck mit Fingerprints
- `gpg-key2ps 43E1FB1F | lp`

pub 1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com> Key fingerprint = CBCA 9EC7 5965 1172 B3F9 8DB1 A2F6 4C47 43E1 FB1F sub 2048g/0B2754E4 2005-11-30
pub 1024D/43E1FB1F 2005-11-30 John Doe <jd@example.com> Key fingerprint = CBCA 9EC7 5965 1172 B3F9 8DB1 A2F6 4C47 43E1 FB1F sub 2048g/0B2754E4 2005-11-30

- `http://pgp-tools.alioth.debian.org/`

caff

- CA - Fire and Forget
- Erinnerung: Es werden User IDs signiert, nicht Keys
  - User IDs sind i.a.R. Email-Adressen, somit macht es Sinn, diese zu überprüfen
- Challenge Response Verfahren sind umständlich und lästig für beide Seiten
- Versand der Signatur für die jeweilige User ID, verschlüsselt an den Key
- `http://pgp-tools.alioth.debian.org/`

Übersicht

Weitere Informationen

- The GNU Privacy Handbook

  `http://www.gnupg.org/documentation/guides.html`
- GnuPG MiniHOWTO

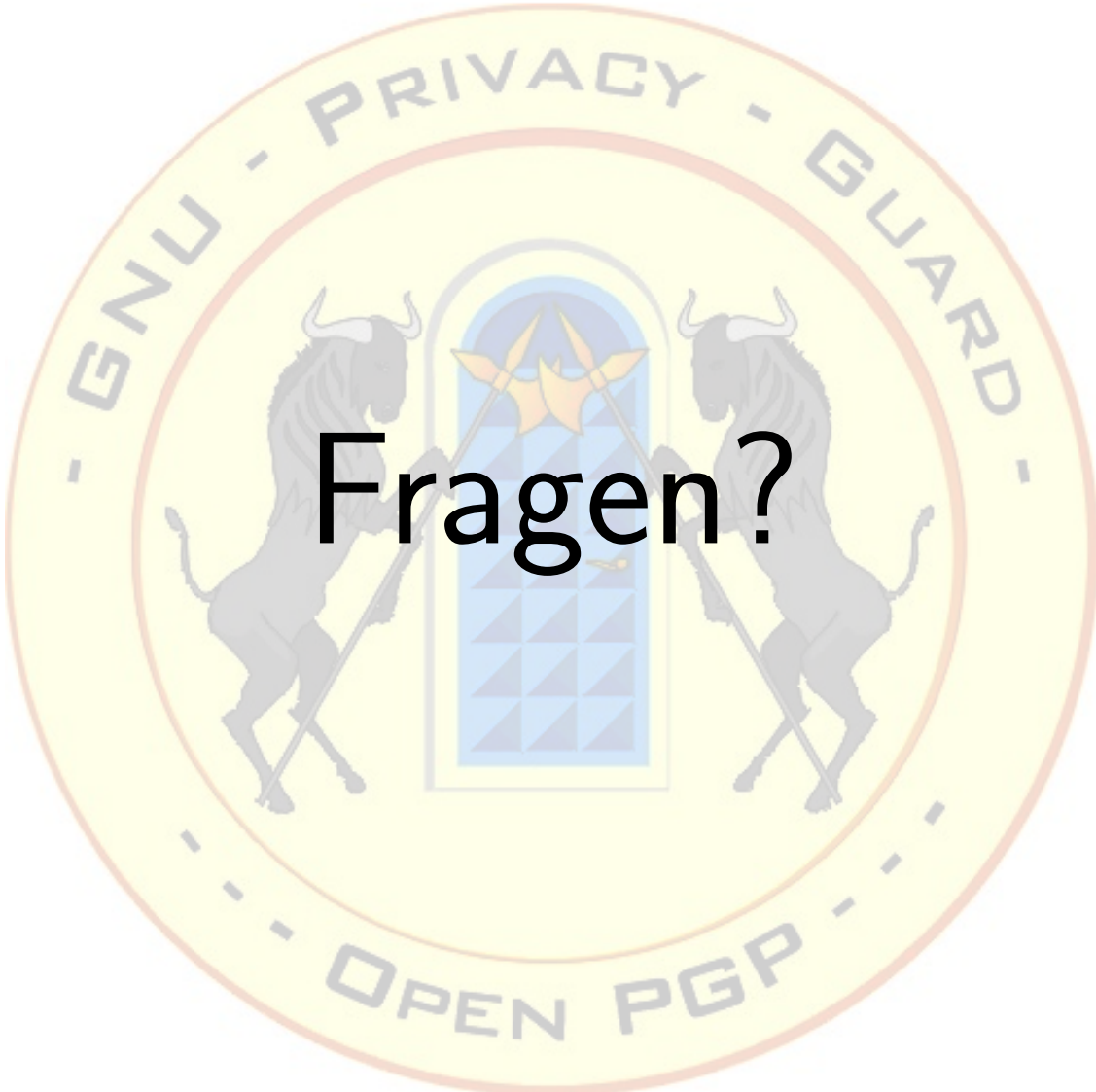
  `http://www.gnupg.org/documentation/howtos.html`
- OpenPGP Charter (RFC 2440 draft)

  `http://www.ietf.org/html.charters/openpgp-charter.html`

- GnuPG - mit Sicherheit im Internet

  `http://www.elho.net/pub/`

Fragen



Fragen?