GnuPG for Beginners

Tristan Miller

Research Studios Austria FG

July 6 and 13, 2023

Overview

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Authentication

Trust in a key's owner

Further topics

- Introduction to symmetric and public-key cryptography
- GnuPG: A free implementation of the OpenPGP standard
  - Creating and managing keys
  - Encrypting and decrypting documents
  - Generating and verifying digital signatures
  - GnuPG trust model
  - Front ends

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

Background

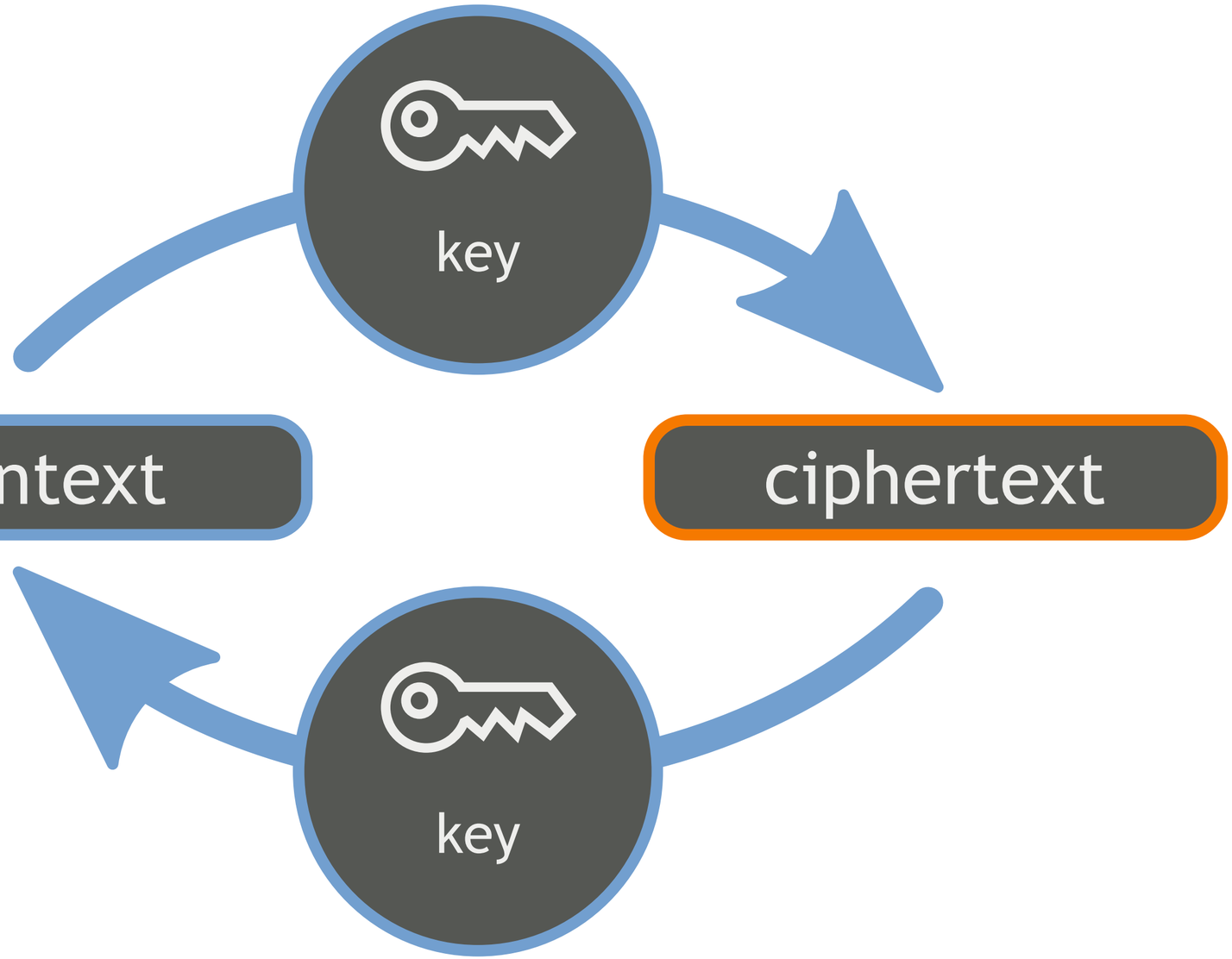
Symmetric ciphers

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- A **symmetric cipher** uses the same key for encryption and decryption.
- Two parties communicating using a symmetric cipher must agree on the key beforehand.
- The sender encrypts a message using the key, sends it to the receiver, and the receiver decrypts the message using the key.
- Examples: ROT13, AES, 3DES, Blowfish.
- Advantage: hard to crack, provided the key is big enough (128 bits is standard).
- Disadvantage: How to securely communicate the key?

Symmetric ciphers

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



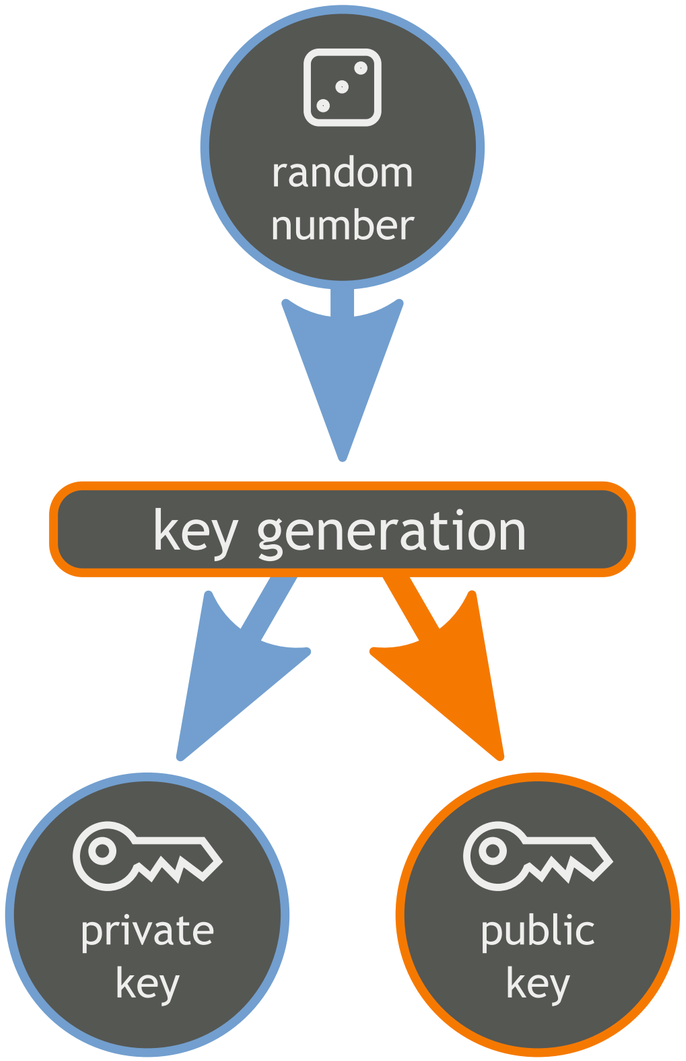
Public-key ciphers

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- **Public-key** ciphers use a pair of keys:
  - The **public key** is given to anyone who wishes to communicate and is used to encrypt a message.
  - The **private key** is kept secret and is used to decrypt a message.
- Advantage: simplified key exchange.
- Disadvantage: easier to crack, so key sizes must be much larger ($\geq$2048 bits is standard).
- **Hybrid ciphers** combine elements of both symmetric and public-key encryption.

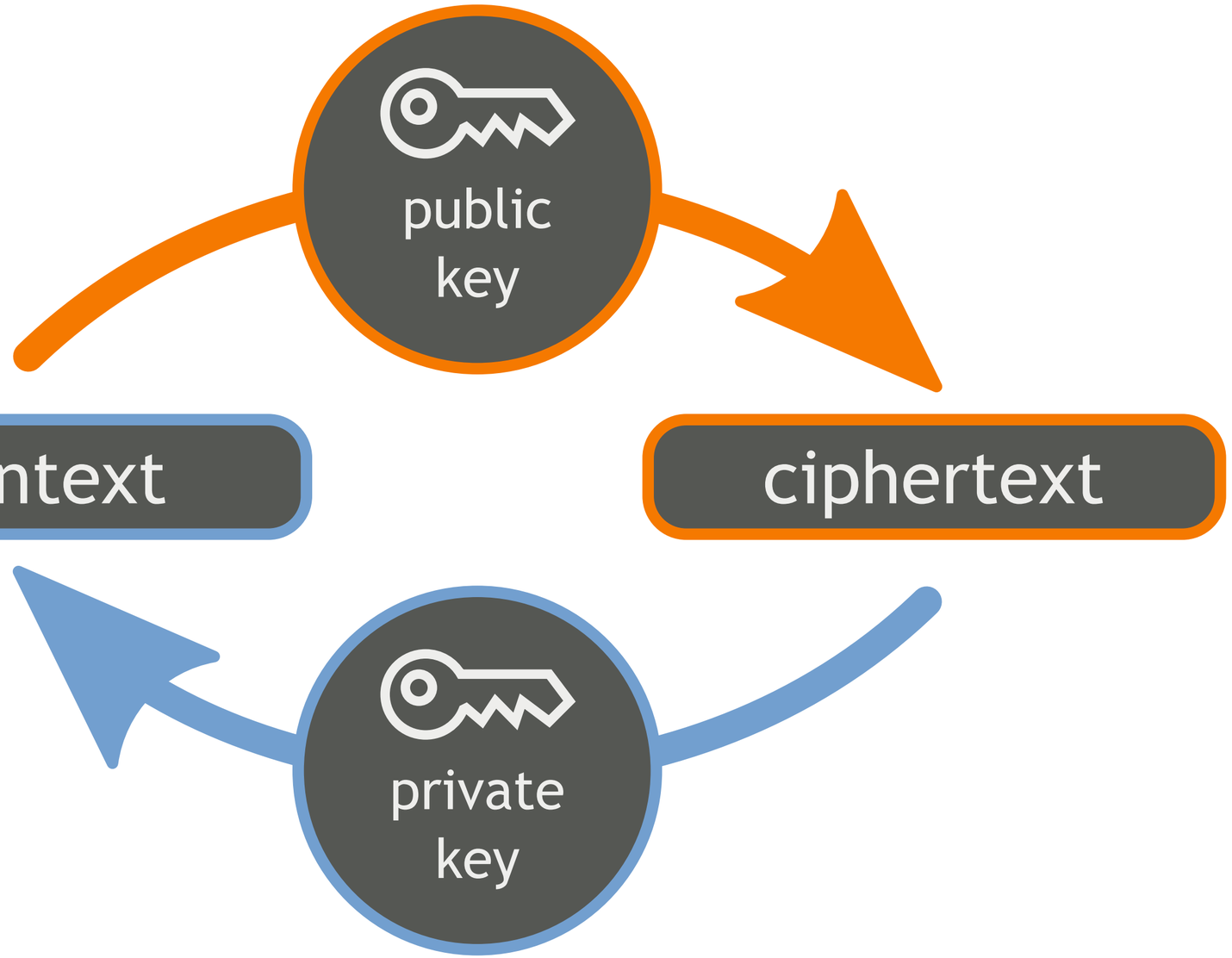
Public-key ciphers

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



Public-key ciphers

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



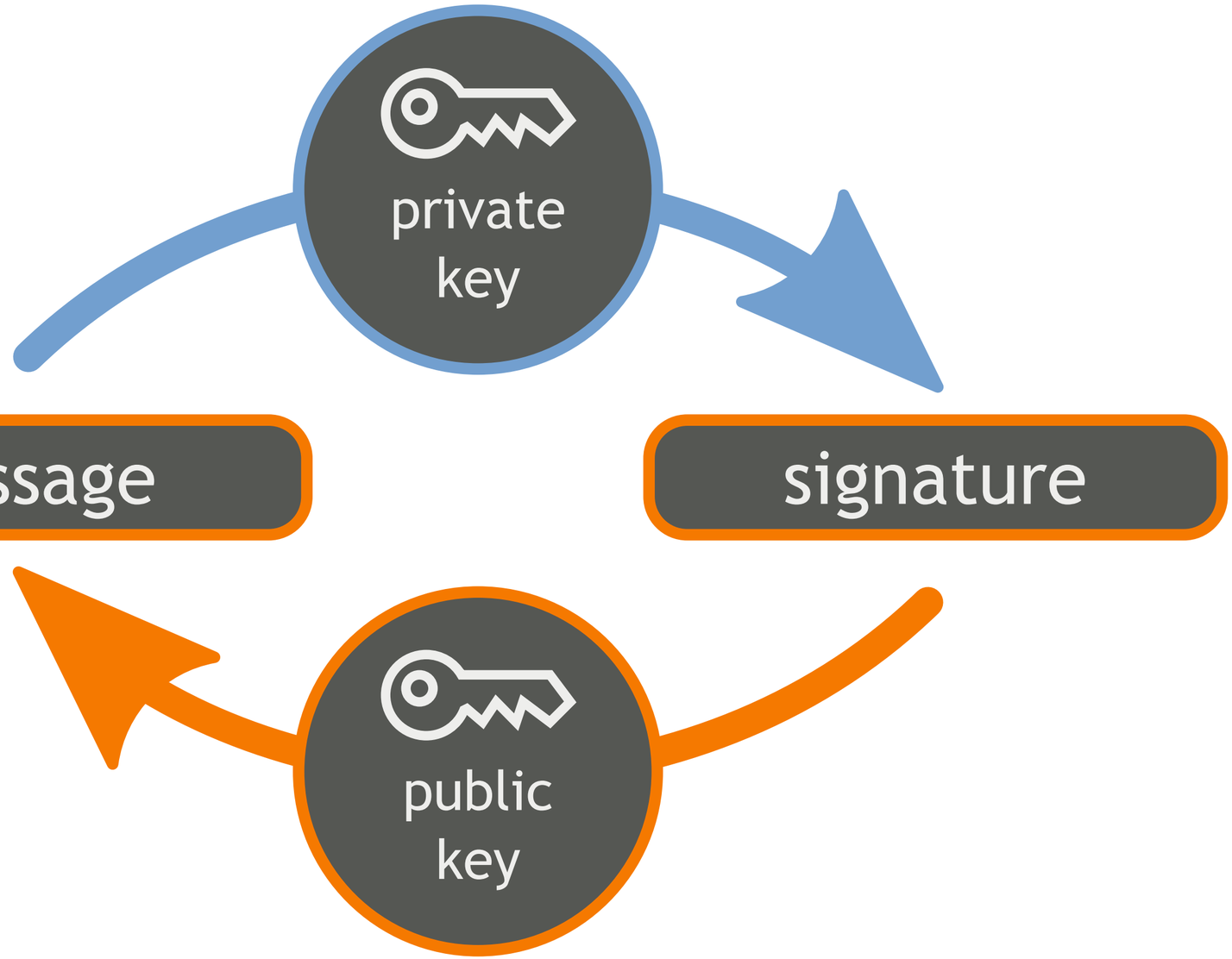
Digital signatures

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- A document's **digital signature** is the result of applying a one-way hash function to the document.
- The hash is then encrypted using the signer's **private key**.
- To verify the signature, the recipient decrypts the hash using the signer's **public key**.
- If the decrypted hash value matches the actual hash value of the document (as calculated by the recipient), then the recipient can be sure that the document he has received was exactly the same one the signer sent.

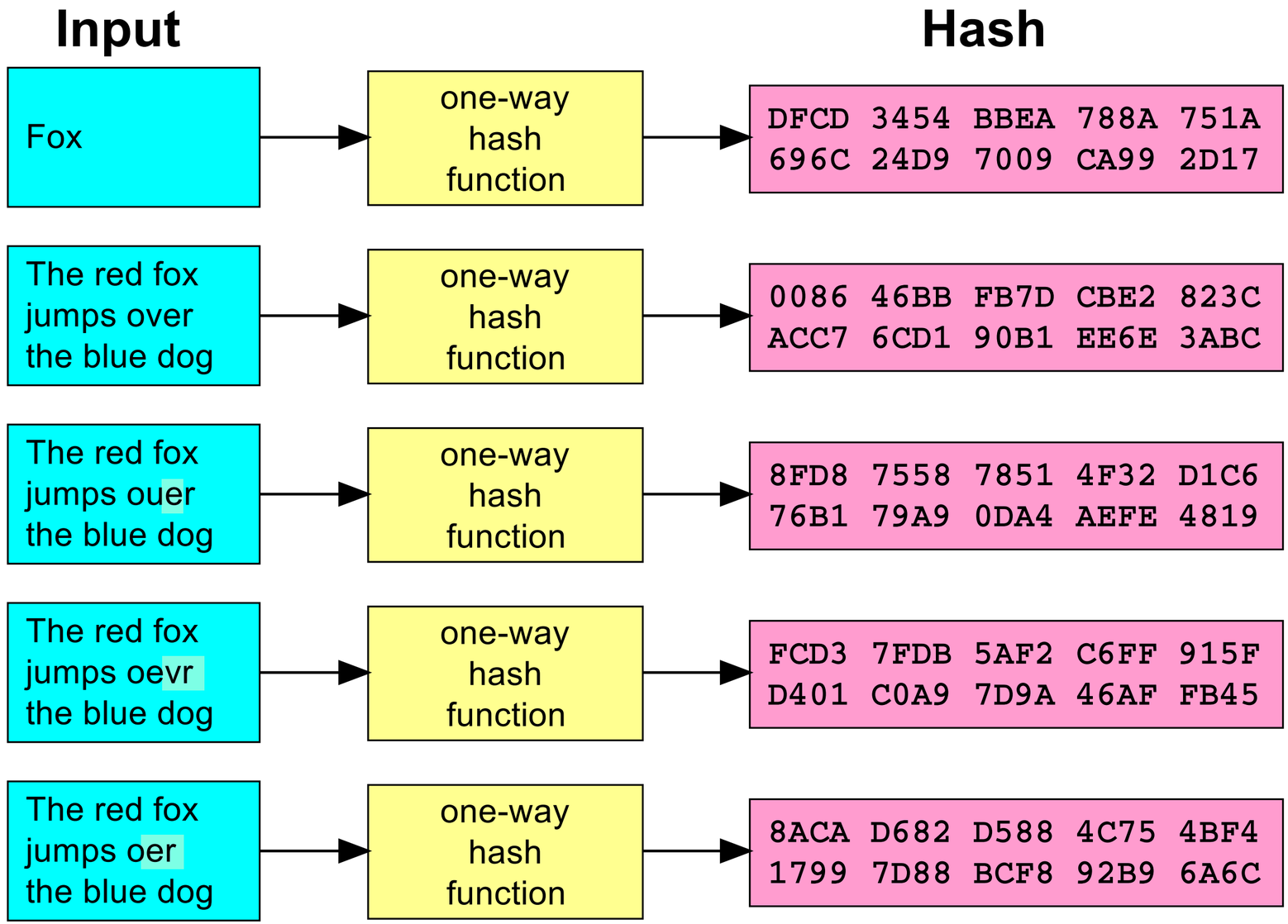
Digital signatures

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



Digital signatures

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



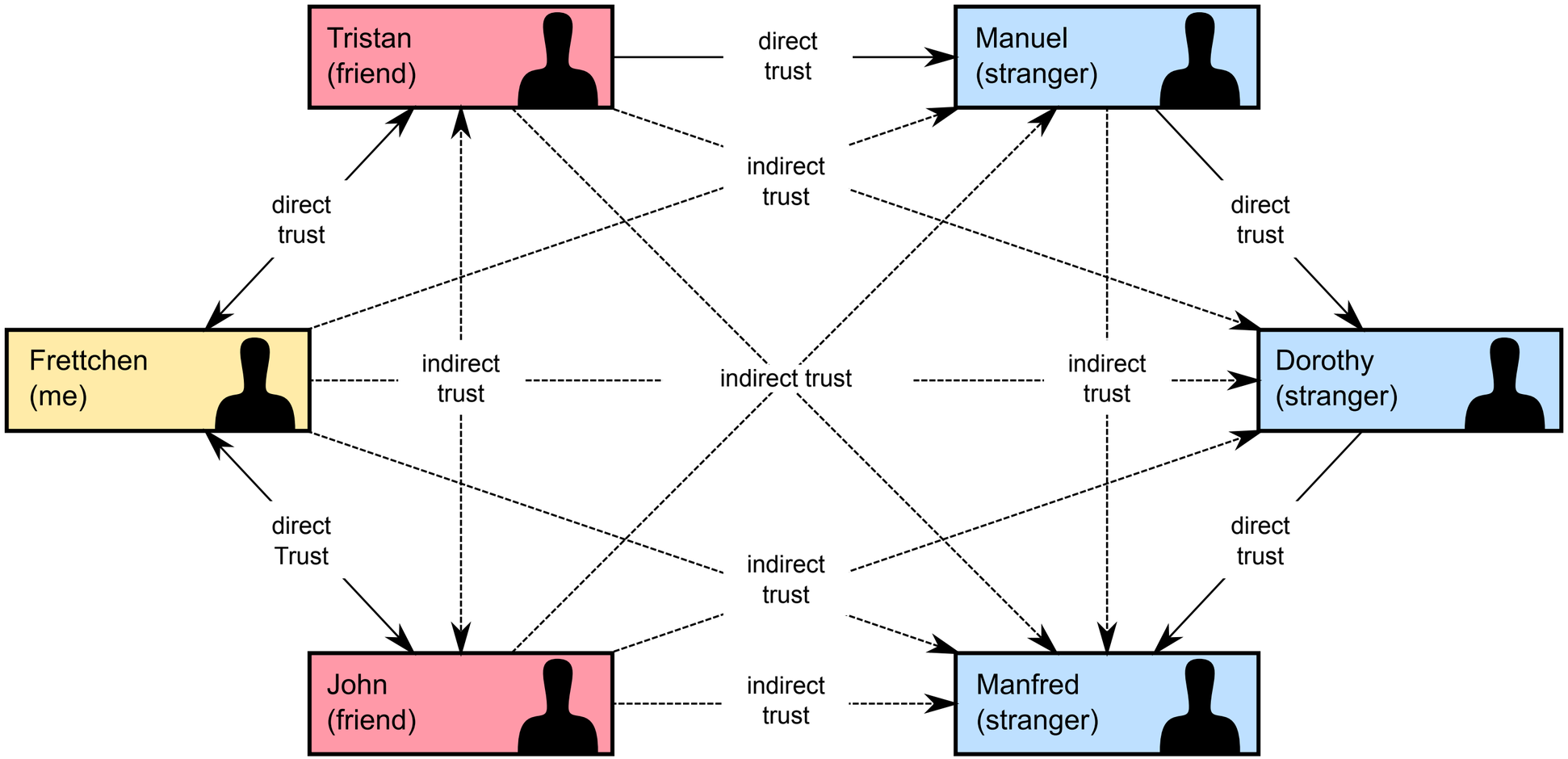
Web of trust

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- When you have faith that a certain public key belongs to a certain person, you can add your digital signature to that public key and then republish it.
- However, it would be awkward for you to have to personally verify and sign every single public key you encounter.
- OpenPGP addresses this problem with a mechanism known as the **web of trust**.
- In the web of trust model, responsibility for validating public keys is delegated to people you trust.

Web of trust

Background
Symmetric ciphers
Public-key ciphers
Digital signatures
Web of trust
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics



Background

Why use GnuPG?

Software distribution

Authenticating e-mail

Encrypting e-mail

Protecting personal data

Acquiring the software

Managing keys

Encryption

Authentication

Trust in a key's owner

Further topics

Why use GnuPG?

Software distribution

Background
Why use GnuPG?
Software distribution
Authenticating e-mail
Encrypting e-mail
Protecting personal data
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

If you distribute software on the Internet, there are many reasons to digitally sign your packages:

- packages cannot be tampered with without breaking the signature
- corrupted downloads will break the signature
- encapsulated signatures are supported and encouraged by major packaging formats: JAR (Maven), RPM, deb, *etc.*

Authenticating e-mail

Background

Why use GnuPG?

Software distribution

Authenticating e-mail

Encrypting e-mail

Protecting personal data

Acquiring the software

Managing keys

Encryption

Authentication

Trust in a key's owner

Further topics

- By making it a policy of yours to always sign important e-mails, you can prevent e-mails from being forged in your name.
- When your correspondents sign their e-mails, you can always be sure you know who you're communicating with.
- Signing e-mails prevents deniability—if you receive a signed document from someone, they cannot later claim that they did not produce it.

Encrypting e-mail

Background
Why use GnuPG?
Software distribution
Authenticating e-mail
Encrypting e-mail
Protecting personal data
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

Encrypting e-mails containing proposals, results, and publication drafts reduces the following risks:

- sensitive communications intercepted by or leaked to press, "Big Brother" governments, *etc.*
- research results copied and published by unscrupulous colleagues or students
- corporate espionage on important projects with business research partners
- private documents accidentally sent to wrong e-mail address

Protecting personal data

Background
Why use GnuPG?
Software distribution
Authenticating e-mail
Encrypting e-mail
Protecting personal data
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

GnuPG's symmetric-key encryption can be used to protect sensitive documents stored on your own (or remote) computers. For instance:

- experiment results
- personal data on experiment volunteers
- password lists
- bank and credit card statements

Background

Why use GnuPG?

Acquiring the software

GnuPG *vs.* PGP *vs.* OpenPGP

Installing GnuPG

Managing keys

Encryption

Authentication

Trust in a key's owner

Further topics

Acquiring the software

GnuPG vs. PGP vs. OpenPGP

Background
Why use GnuPG?
Acquiring the software
GnuPG *vs.* PGP *vs.* OpenPGP
Installing GnuPG
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- **PGP**: first developed and freely released by Phil Zimmerman
- PGP later commercialized; now a proprietary system
- encryption method standardized as **OpenPGP**
- **GnuPG** is GNU's free implementation of the OpenPGP standard
- other implementations of OpenPGP exist, but GnuPG is free and popular

Installing GnuPG

Background
Why use GnuPG?
Acquiring the software
GnuPG *vs.* PGP *vs.* OpenPGP
Installing GnuPG
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics

- Download from `https://gnupg.org/` or from your OS's package manager.
- Compile from source or fetch a binary package for a supported system:
  - GNU/Linux
  - Mac OS X
  - Android
  - Microsoft Windows
- Third-party GUIs are available, but an understanding of the underlying command-line version is helpful.

Background

Why use GnuPG?

Acquiring the software

Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers

Encryption

Authentication

Trust in a key's owner

Further topics

Managing keys

Generating keypairs

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- All GnuPG functions are invoked through the gpg command.
- The command-line option `--gen-key` is used to create a new primary keypair.

```
$ gpg --gen-key
gpg (GnuPG) 2.0.24; Copyright (C) 2013 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.

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?
```

Generating a new keypair

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- You will be prompted for a key type and key size.
- The defaults (2048-bit RSA keys, in recent GnuPG versions) are generally sensible.
- Larger key sizes provide extra security at the cost of speed; if future-proofing is important for you, use the maximum key size of 4096 bits.

```
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

Generating a new keypair

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Next, you must choose an expiration date.
- For most users, a key that does not expire is adequate.
- Some people prefer to set an expiry date (say, two years) as a safeguard. You can periodically extend the expiry date if your key is still in use.

```
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)
```

Generating a new keypair

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- You must now provide a user ID.
- It is possible to add additional user IDs later in case you want to use the key in two or more contexts.
- A user ID should be created carefully since it cannot be edited after it is created!
- Avoid using the "Comment" field without good reason.

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

Real name: Frettchen Rättchen
Email address: frettchen@dfki.de
Comment:
You are using the 'utf-8' character set.
You selected this USER-ID:
    "Frettchen Rättchen <frettchen@dfki.de>"

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

Generating a new keypair

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Finally, you must enter a **passphrase** to protect your private key.

```
You need a Passphrase to protect your secret key.

Enter passphrase:
```

- Because this passphrase protects access to your PGP identity, it should be carefully chosen. It must be long enough to be secure, but also easy for you to remember and type.
- At `http://www.diceware.com/` you will find a method of generating long but easy-to-remember passphrases by combining five English or German words.
  Example: `distel ist landen kammer puffen`

Your public keyring

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Your **keyring** is a list of all public keys you have generated or imported. You can view it with the `--list-keys` option:

```
$ gpg --list-keys
/home/frettchen/.gnupg/pubring.gpg
----------------------------------
pub   2048R/4116CBB0 2016-08-15
uid        [ultimate] Frettchen Rättchen <frettchen@dfki.de>
sub   2048R/FA659AB0 2016-08-15

pub  1024D/EFBF4915 2003-10-24 Tristan Miller (Research scientist) <tristan.mi
uid                            Tristan Miller <psychonaut@nothingisreal.com>
sub  1024g/B40BE860 2003-10-24
```

Your public keyring

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Most command-line arguments dealing with keys let you specify a particular key or set of keys. You can use the key's ID or any part of the user ID. For example:

```
$ gpg --list-keys Tristan
pub  1024D/EFBF4915 2003-10-24 Tristan Miller (Research scientist) <tristan.mi
uid                            Tristan Miller <psychonaut@nothingisreal.com>
sub  1024g/B40BE860 2003-10-24
```

Revocation certificates

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- If you forget your passphrase or if your private key is compromised or lost, a **revocation certificate** may be published to notify others that the public key should no longer be used.

```
$ gpg --output revoke.asc --gen-revoke Frettchen

sec  2048R/4116CBB0 2016-08-15 Frettchen Rättchen <frettchen@dfki.de>

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?
```

Exporting a public key

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- To communicate with others you must exchange public keys.
- To export a public key on your keyring, use the `--export` option.
- By default, keys are exported as binary data, but you can specify an ASCII encoding using the `--armor` option.

```
$ gpg --armor --export Frettchen
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBFex0P4BCADR8+CMx/j8mIMQEMWjS/2GOSth1tR5mSwuk31oPMn0xtPTY+oK
BshOMDJjujJew/+aJBu2rTBGQdfza4W/jBLy/uwU8C92+Rp6SBsPw9P+f0g297pm
                                ...
WWKwsmkDsqlJzdDVsFHYYJTHy+MUBRzrVbuG8lscCeOVUnUf8oIkiydrCwprl249
iz5kKxk4VcCdEmfEluj9AZxhONKKnH4X+cyVrk9tIsup9mjMCqCv
=9bBk
-----END PGP PUBLIC KEY BLOCK-----
```

Importing a public key

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Many people publish their public key on their web page.
- A public key may be added to your public keyring with the `--import` option. You can either specify a filename or paste from the clipboard into stdin.

```
$ gpg --import /tmp/smith.gpg
gpg: key 65B46947: public key "John Smith <smith@example.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)

$ gpg --list-keys Smith
pub   2048R/65B46947 2016-08-15
uid       [ unknown] John Smith <smith@example.com>
sub   2048R/4D993DDE 2016-08-15
```

Validating a key

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Once a key is imported, it should be **validated**.
- Sometimes a key may be automatically validated by virtue of a chain of trust.
- You may need to personally validate some keys. This entails the following:
  1. Verify the key's fingerprint with the owner.
  2. Sign the key to certify it as valid.

Verifying a key

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- A key's **fingerprint** is verified with the key's owner.
- This may be done in person or over the phone or through any other means as long as you can *guarantee* that you are communicating with the key's true owner.
- If the fingerprint you get is the same as the fingerprint the key's owner gets, then you can be sure that you have a correct copy of the key.
- Use the `--fingerprint` option to retrieve a key's fingerprint.

```
$ gpg --fingerprint Smith
pub   2048R/65B46947 2016-08-15
      Key fingerprint = 42A4 5686 FFC8 B7B1 0FCF  F3BD 397F 0C60 65B4 6947
uid       [ unknown] John Smith <smith@example.com>
sub   2048R/4D993DDE 2016-08-15
```

Signing a key

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- After checking the fingerprint, you may **sign** the key to validate it.

```
$ gpg --sign-key Smith

pub  2048R/65B46947  created: 2016-08-15  expires: never       usage: SC
                     trust: unknown       validity: full
sub  2048R/4D993DDE  created: 2016-08-15  expires: never       usage: E
[  full  ] (1). John Smith <smith@example.com>
```

```
pub  2048R/65B46947  created: 2016-08-15  expires: never       usage: SC
                     trust: unknown       validity: full
 Primary key fingerprint: 42A4 5686 FFC8 B7B1 0FCF  F3BD 397F 0C60 65B4 6947

     John Smith <smith@example.com>

Are you sure that you want to sign this key with your
key "Frettchen Rättchen <frettchen@dfki.de>" (4116CBB0)

Really sign? (y/N)
```

Listing key signatures

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Signatures are incorporated into a public key, and are distributed with it.
- Once signed you can check the key to list the signatures on it and see the signature that you have added.
- Every user ID on the key will have one or more self-signatures as well as a signature for each user that has validated the key.

```
$ gpg --check-sigs Smith
gpg: checking the 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:   0  trust: 1-, 0q, 0n, 0m, 0f, 0u
pub   2048R/65B46947 2016-08-15
uid       [  full  ] John Smith <smith@example.com>
sig!3        65B46947 2016-08-15  John Smith <smith@example.com>
sig!2        4116CBB0 2016-08-15  Frettchen Rättchen <frettchen@dfki.de>
sub   2048R/4D993DDE 2016-08-15
sig!         65B46947 2016-08-15  John Smith <smith@example.com>
```

Public key servers

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- Many people publish their public key on their web page.
- However, not everyone has a web page, or knows where to find yours.
- To solve this problem, **public key servers** are used to collect and distribute public keys.
- A public key received by the server is either added to the server's database or merged with the existing key if already present.
- When a key request comes to the server, the server consults its database and returns the requested public key if found.
- There are several popular keyservers in use around the world. The major ones synchronize themselves regularly, so you can just pick one for your general use.

Public key servers

Background
Why use GnuPG?
Acquiring the software
Managing keys
Generating keypairs
Your public keyring
Revocation certificates
Exporting a public key
Importing a public key
Validating a key
Verifying a key
Signing a key
Listing key signatures
Public key servers
Encryption
Authentication
Trust in a key's owner
Further topics

- You can send and receive keys to/from keyservers with the `--send-key` and `--recv-key` options. You also need to specify which keyserver using the `--keyserver` option.

```
$ gpg --keyserver hkps://hkps.pool.sks-keyservers.net --send-key BF8A2EE4
gpg: sending key BF8A2EE4 to hkps server hkps.pool.sks-keyservers.net
```

```
$ gpg --keyserver hkps://hkps.pool.sks-keyservers.net --recv-key BF8A2EE4
gpg: requesting key BF8A2EE4 from hkps server hkps.pool.sks-keyservers.net
gpg: key BF8A2EE4: "Tristan Miller <tristan@logological.org>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
```

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Encrypting a document

Decrypting a document

Symmetric encryption

Authentication

Trust in a key's owner

Further topics

Encryption

Encrypting a document

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Encrypting a document
Decrypting a document
Symmetric encryption
Authentication
Trust in a key's owner
Further topics

- To **encrypt** a document the option `--encrypt` is used.
- You must have the public keys of the intended recipients, whom you specify with the `--recipient` option.
- GnuPG expects the name of the document to encrypt as input; if omitted, it reads standard input.
- The encrypted result is placed on standard output or as specified using the option `--output`.
- The document is automatically compressed before encryption.
- Remember to include *yourself* as a recipient if you want to be able to decrypt and view the document!

```
$ gpg --output doc.gpg --encrypt --recipient Smith
```

Decrypting a document

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Encrypting a document
Decrypting a document
Symmetric encryption
Authentication
Trust in a key's owner
Further topics

- To **decrypt** a message the option `--decrypt` is used.
- You need the private key to which the message was encrypted.
- The document to decrypt is input, and the decrypted result is output.

```
$ gpg --output doc.txt --decrypt doc.gpg
```

Symmetric encryption

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Encrypting a document
Decrypting a document
Symmetric encryption
Authentication
Trust in a key's owner
Further topics

- Documents may also be encrypted with a symmetric cipher instead of public-key cryptography.
- The symmetric cipher offers higher security, but should be used only when the passphrase does not need to be communicated to others.
- Documents can be encrypted with the `--symmetric` option and decrypted as usual with `--decrypt`.

```
$ gpg --output doc.gpg --symmetric doc.txt
Enter passphrase:
```

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Authentication

Signing a document

Clearsigned documents

Detached signatures

Verifying signatures

Trust in a key's owner

Further topics

Authentication

Signing a document

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Signing a document
Clearsigned documents
Detached signatures
Verifying signatures
Trust in a key's owner
Further topics

- A digital signature certifies and timestamps a document.
- If the document is subsequently modified in any way, a verification of the signature will fail.
- A digital signature can serve the same purpose as a hand-written signature with the additional benefit of being tamper-resistant.
- Software packages can be signed so that users who download them can verify that they have not been modified since they were packaged.
- E-mails can be signed so that the recipient can verify that the message has not been forged or altered.
- There are two common ways of producing a signature: clearsigning and detatched signatures

Clearsigned documents

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Signing a document
Clearsigned documents
Detached signatures
Verifying signatures
Trust in a key's owner
Further topics

- The option `--clearsign` causes a text document to be wrapped in an ASCII-armored signature.

```
$ echo "Hello, world\!" >hello.txt
$ gpg --clearsign hello.txt

You need a passphrase to unlock the secret key for
user: "Frettchen Rättchen <frettchen@dfki.de>"
2048-bit RSA key, ID 4116CBB0, created 2016-08-15

$ cat hello.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hello, world\!
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iQEcBAEBAgAGBQJXsdvKAAoJEFNJFOhBFsuwn/oH/1Io2fYEC/zOIgwEMZqlg2Qi
SfGiP8mu8l5nIjWBZtvrxltF9oocx8YJBF4Xt0TqAW6XN82dnAaw6+cu3xkcayqh
BJqFdXsxJ8cqG2kNDjkeO3fkQvVhiUquYI9kRIP1xBH78GdijvRG9pQQl4Ty1hiw
NkH9GW1eYRtzfckJMOj6DpVzWJNLJg8Jt4oyvKQ28W4zhTIy34ke8McGel9h57dj
luPU7q38IH6N/9VEoyTLk0tv6DTIId8dqkXz7TJfkMVHIGcsPbHomvx+lPCOn+5F
WukuStzm7sJLjHl9ootfuxCvOrshmuZVNjxoAhLW1UdPR/b8Wu8gaFmOfSURcTY=
=tTMY
-----END PGP SIGNATURE-----
```

Detached signatures

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Signing a document
Clearsigned documents
Detached signatures
Verifying signatures
Trust in a key's owner
Further topics

- Clearsigned documents have two limitations:
  - Clearsigning is appropriate only for text documents.
  - To obtain the original version, the document must be edited to remove the signature.
- It is therefore possible to output a signature to a separate file, leaving the original document intact.
- For this the `--detach-sig` option is used.

```
$ gpg --armor --output hello.sig --detach-sig hello.txt
```

Verifying signatures

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Signing a document
Clearsigned documents
Detached signatures
Verifying signatures
Trust in a key's owner
Further topics

- Given a signed document and a public key, you can check the signature with the `--verify` option.
- If the document has a detached signature, you need to specify both the signature and document filenames on the command line.

```
$ gpg --verify hello.sig hello.txt
gpg: Signature made Mon 15 Aug 2016 05:14:28 PM CEST using RSA key ID 4116CBBC
gpg: Good signature from "Frettchen Rättchen <frettchen@dfki.de>" [ultimate]
```

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Authentication

Trust in a key's owner

Trust model

Assigning trust

Using trust to validate keys

Further topics

Trust in a key's owner

Web of trust

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Authentication

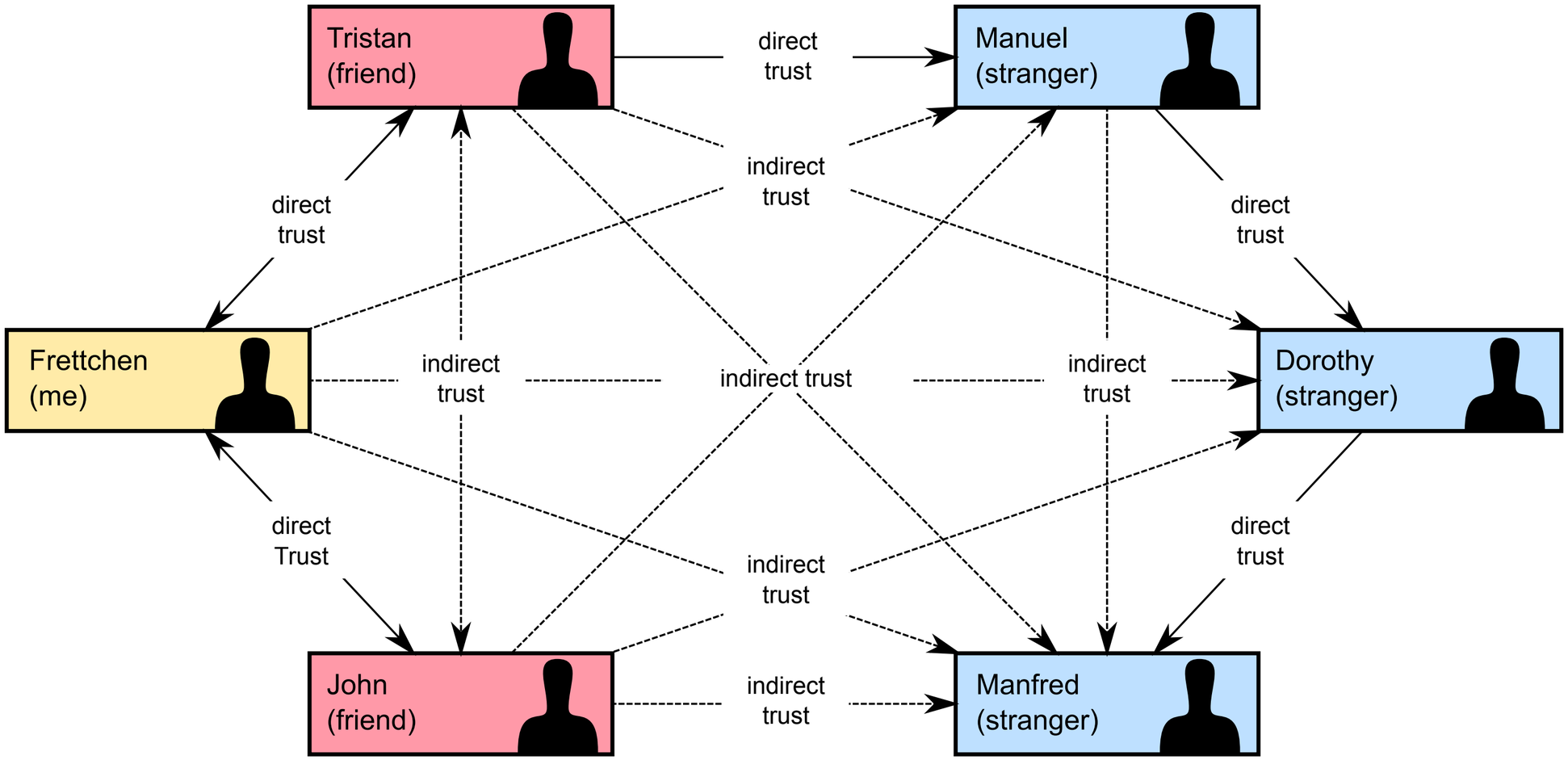
Trust in a key's owner

Trust model

Assigning trust

Using trust to validate keys

Further topics



Trust model

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Trust model
Assigning trust
Using trust to validate keys
Further topics

- In practice trust is subjective.
- For example, Blake's key is valid to Alice since she signed it, but she may not trust Blake to properly validate keys that he signs.
- The web of trust model accounts for this by associating with each public key on your keyring an indication of how much you trust the key's owner:
  - unknown, none, marginal, full, ultimate
- A key's trust level is something that you alone assign to the key, and it is considered private information.
- It is not packaged with the key when it is exported; it is even stored separately from your keyrings in a separate database.

Assigning trust

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Trust model
Assigning trust
Using trust to validate keys
Further topics


```
$ gpg --edit-key John
...

pub  2048R/65B46947  created: 2016-08-15  expires: never       usage: SC
                     trust: unknown       validity: full
sub  2048R/4D993DDE  created: 2016-08-15  expires: never       usage: E
[  full  ] (1). John Smith <smith@example.com>

gpg> trust
pub  2048R/65B46947  created: 2016-08-15  expires: never       usage: SC
                     trust: unknown       validity: full
sub  2048R/4D993DDE  created: 2016-08-15  expires: never       usage: E
[  full  ] (1). John Smith <smith@example.com>

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?
```

Using trust to validate keys

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Trust model
Assigning trust
Using trust to validate keys
Further topics

- Formerly, a key was considered valid only if you signed it personally (after verifying it).
- Now we have a revised model. A key K is considered valid if it meets two conditions:
  1. It is signed by enough valid keys, meaning
     - you have signed it personally, or
     - it has been signed by one fully trusted key, or
     - it has been signed by three marginally trusted keys; and
  2. the path of signed keys leading from K back to your own key is five steps or shorter.

Background

Why use GnuPG?

Acquiring the software

Managing keys

Encryption

Authentication

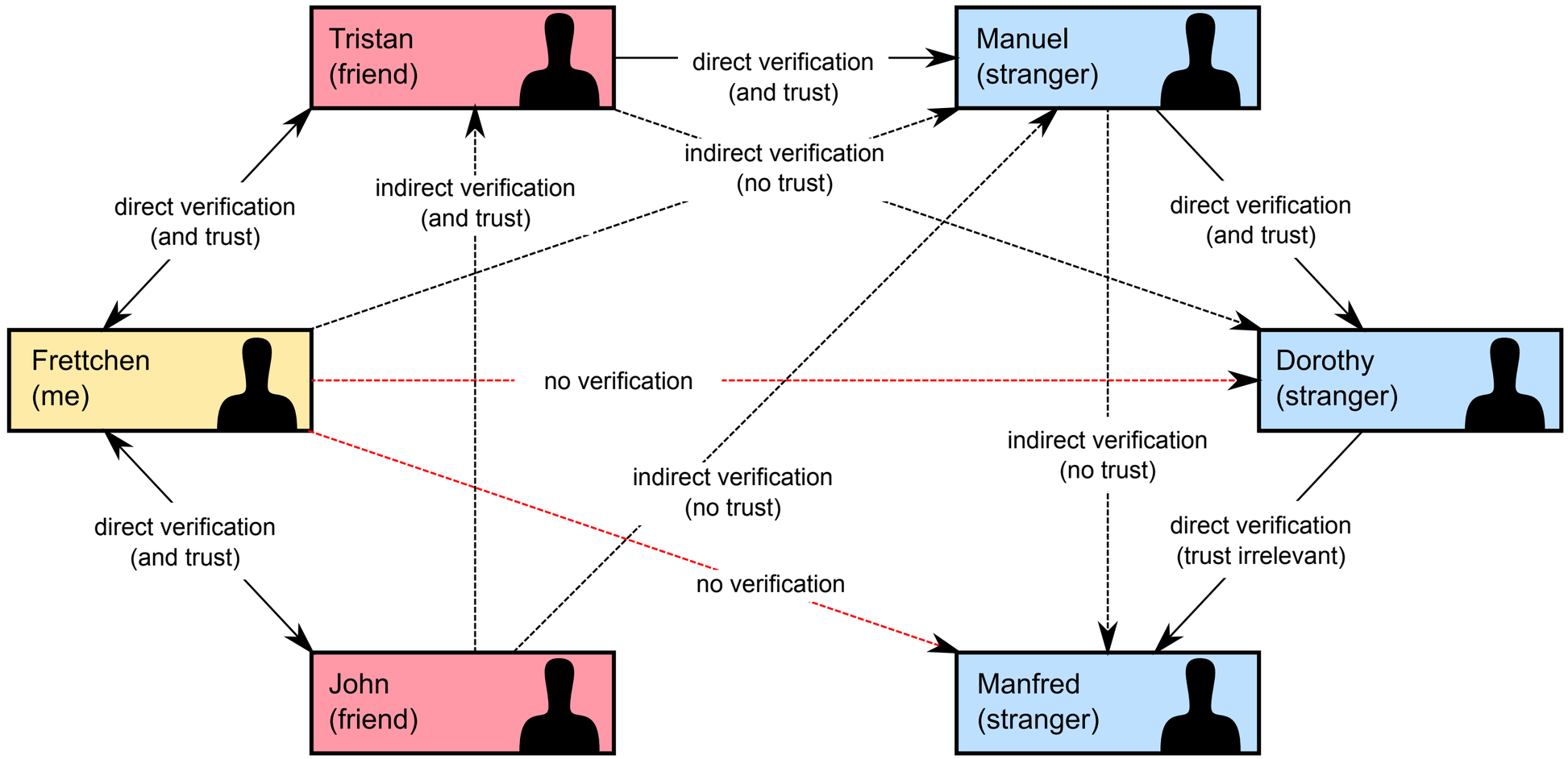
Trust in a key's owner

Trust model

Assigning trust

Using trust to validate keys

Further topics



Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

Further topics

Conveniences

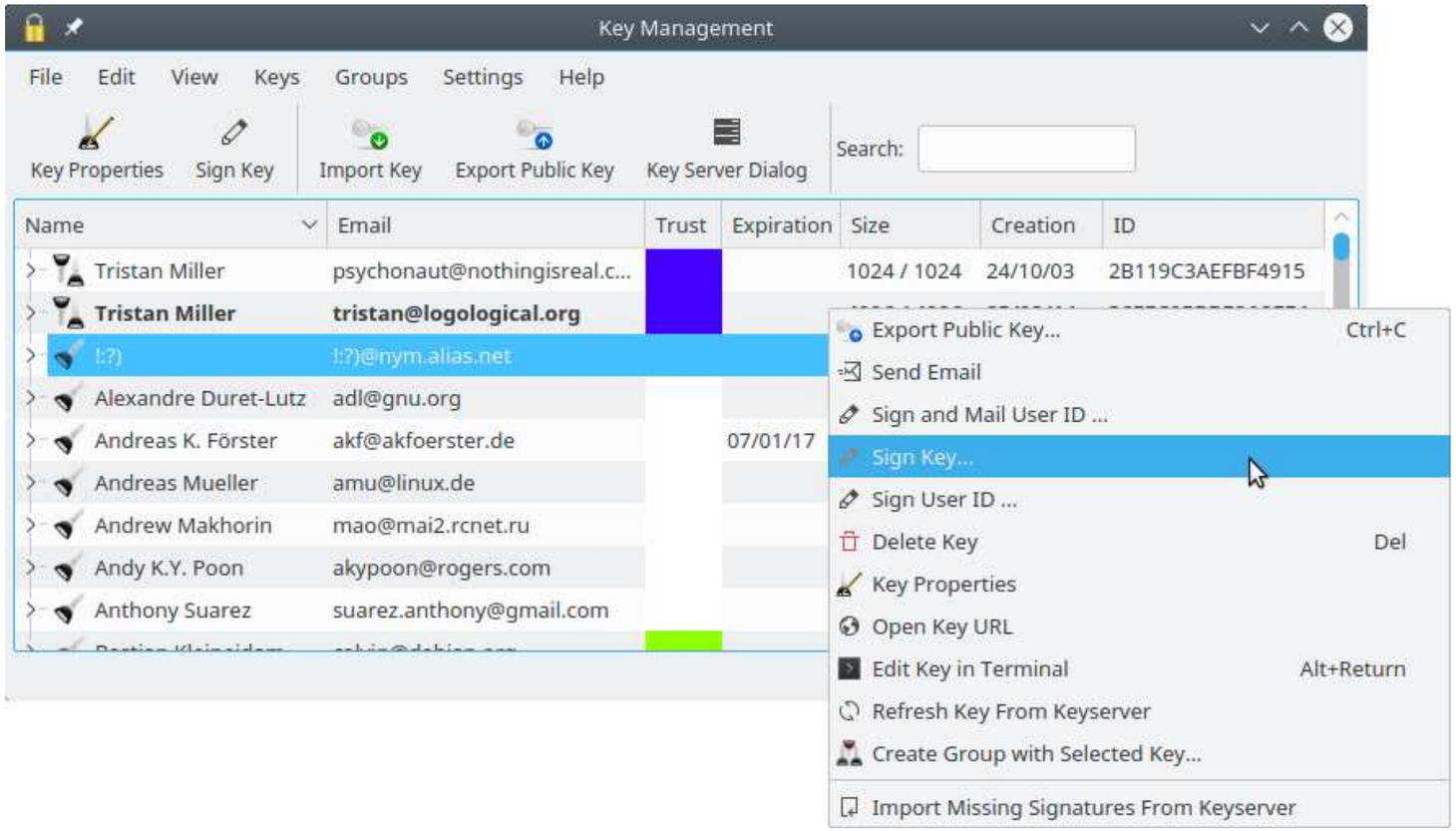
Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- You can use a local configuration file (`~/.gnupg/gpg.conf`) to set your default key, preferred key server, *etc.*
- You can use `gpg-agent`, a daemon that securely caches your keys in memory. This way you don't need to type your passphrase every time you want to use your key.
- `pinentry` is a collection of programs that will prompt for your passphrase using your desktop environment's standard dialogs rather than on the terminal.

GUIs

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

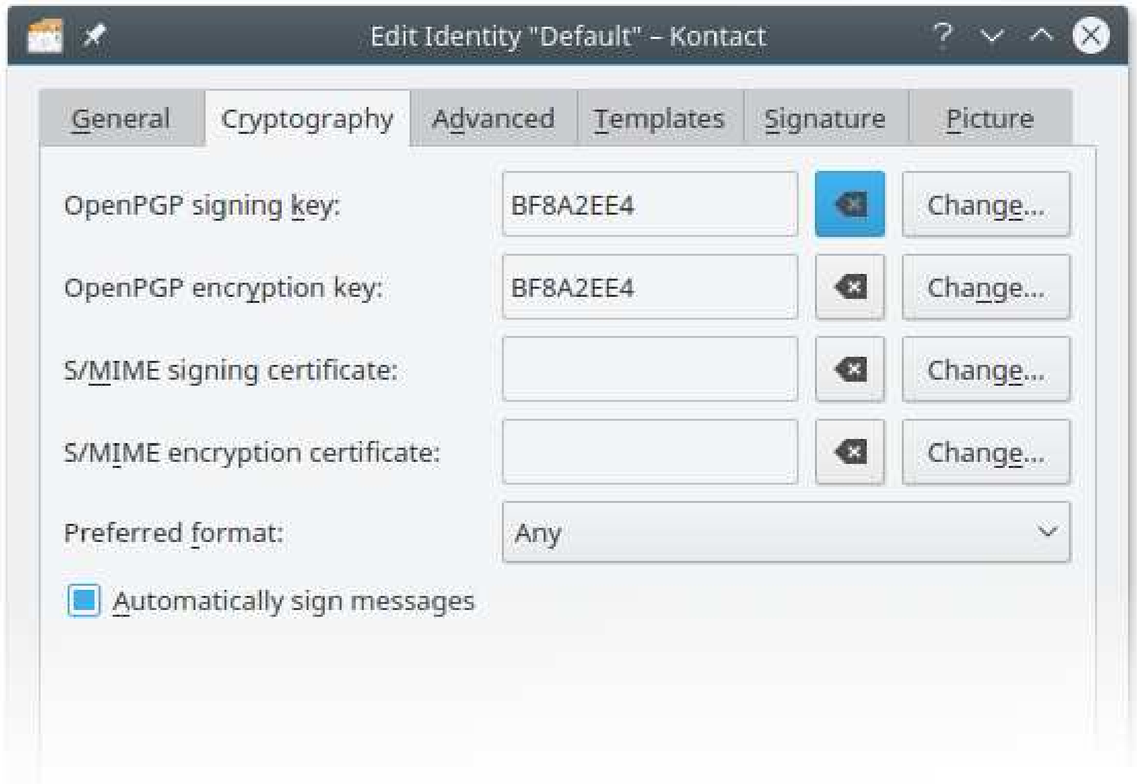
There are also a number of full-fledged key management tools which let you generate, list, edit, import, export, sign, and publish the keys on your keyring.



E-mail integration

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

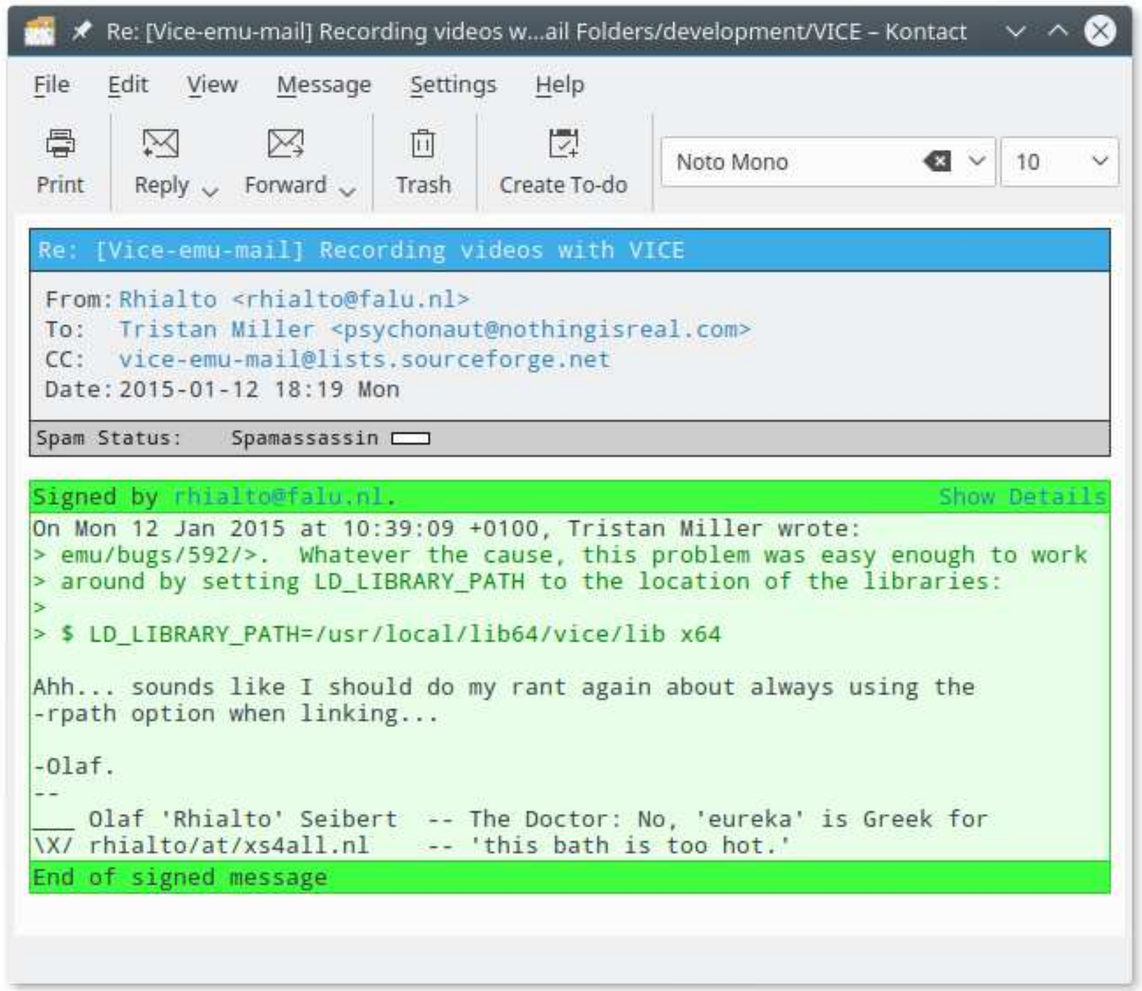
Many e-mail clients support GnuPG, natively or via plugins. For each e-mail account or identity, you can associate a public key for signing and encryption.



E-mail integration

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

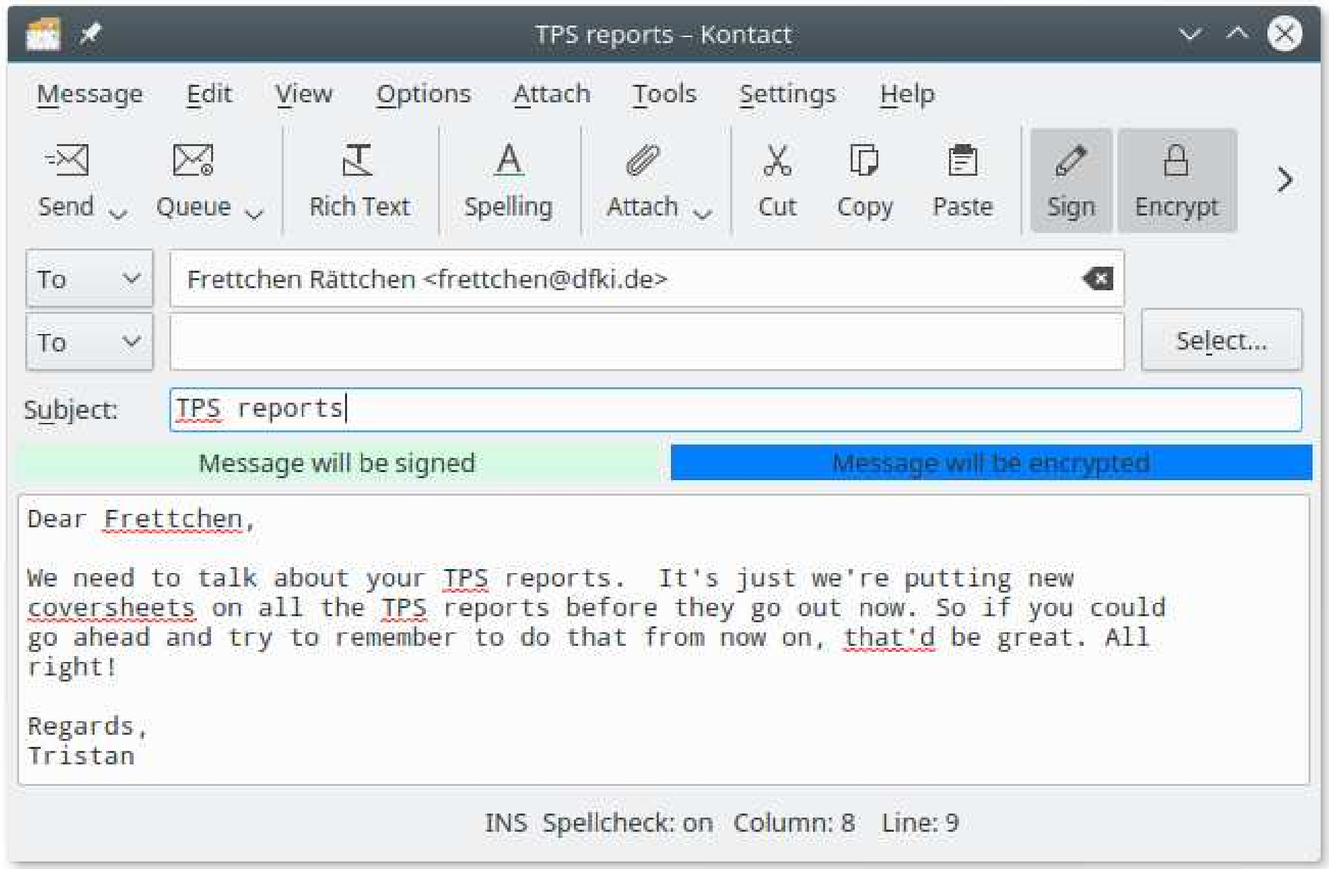
Signatures on messages are automatically checked.



E-mail integration

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

In the message composer, you are given the choice of signing and/or encrypting the message.



Git integration

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- Signing contributions protects your reputation from impostors.
- Verifying signed commits before pulling or merging can help protect your repository's integrity against impostors.

Setting a default signing key:

```
$ git config --global user.signingkey 4116CBB0
```

Signing and verifying tags:

```
$ git tag -s v1.5 -m 'my signed 1.5 tag'
$ git tag -v v1.5
```

Signing and verifying commits:

```
$ git commit -S -m 'signed commit'
$ git log --show-signature
$ git merge --verify-signatures signed-branch
```

Packaging integration

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- Many software packaging and release tools support GnuPG, natively or via plugins.
- The Maven release plugin handles this automatically (at least for binary releases).
- For manually posting source or binary releases on the Web or on GitHub, produce a detached signature of the release and upload it alongside the release.

Identity management

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- Your keypair can include several user IDs (name, e-mail, and comment) for each of your identities
  - Advantages: convenient (only one keypair to worry about)
  - Disadvantage: all your identities are publically linked to each other
- If you have an identity you want to keep separate, then you must generate a separate keypair for it.

Key-signing parties

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- A key-signing party is an event at which people present their public keys fingerprints to each other, so that they can later sign the keys and extend the web of trust:
  - Before the party, participants send their key fingerprints to the organizer, who prepares a master list.
  - At the party, the organizer posts the master list and distributes a printed copy to each participant.
  - Each participant declares whether their identity and fingerprint on the lists is correct.
  - Each participant then verifies the identity of every other participant, and marks the entry on the list.
  - After the party, participants sign the keys that passed the identity check, and then upload them to the key server.

Online resources

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

- GnuPG home page and FAQ
- InfoSec Stack Exchange questions on GnuPG
- OpenPGP Best Practices from Riseup
- Keybase
- OpenPGP courses and lobbying
- The Keysigning Party HOWTO
- Creating signed GitHub releases
- A Git Horror Story: Repository Integrity with Signed Commits

These tutorial slides, on GitHub:
https://github.com/logological/GnuPGforBeginners

Background
Why use GnuPG?
Acquiring the software
Managing keys
Encryption
Authentication
Trust in a key's owner
Further topics
Conveniences
GUIs
E-mail integration
Git integration
Packaging integration
Identity management
Key-signing parties
Online resources

Thank you!