GnuPG

The GNU Privacy Guard

Tristan Miller

German Research Center for Artificial Intelligence

Erwin-Schrödinger-Straße 57

67663 Kaiserslautern

tristan.miller@dfki.de

Symmetric ciphers

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

Public-key ciphers

- Uses 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 (1024 bits is standard).
- **Hybrid ciphers** combine elements of both symmetric and public-key encryption.

Digital signatures

- 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.

Web of trust

- 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.
- GnuPG addresses this problem with a mechanism popularly known as the **web of trust**.
- In the web of trust model, responsibility for validating public keys is delegated to people you trust.

Software distribution

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 many popular archive and packaging formats (*e. g.*, RPM)

Authenticating e-mail

- By making it a policy of yours to always sign important e-mails, you can prevent e-mails from being forged in your name.
- By insisting that your colleagues always 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 send it.

Encrypting e-mail

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

- sensitive communications intercepted by or leaked to press
- research results stolen and published by unscrupulous colleagues or students
- corporate espionage on important projects with business research partners
- confidential customer/client information is leaked; customers sue DFKI for invasion of privacy
- private documents accidentally sent to wrong e-mail address

Protecting personal data

GnuPG's symmetric-key encryption can be used to protect sensitive documents stored on your computer. For instance:

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

GnuPG *vs.* PGP *vs.* OpenPGP

- PGP was 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

- download from `http://www.gnupg.org/`
- compile from source or fetch a binary package for a supported system:
  - GNU/Linux
  - Mac OS X
  - Unix (POSIX-compliant)
  - Microsoft Windows
- GUIs are available, but an understanding of the underlying command-line version is important

Generating a new keypair

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

```
[psy@port-3108:~]$ gpg --gen-key
gpg (GnuPG) 1.2.6; Copyright (C) 2004 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)
   (4) RSA (sign only)
Your selection?
```

Generating a new keypair

DSA keys are always 1024 bits. For ElGamal keys, you must specify a key size. The default key size of 1024 bits is appropriate for most users. (2048 bits is too slow and produces overly large signatures; 768 bits is too easy to crack.)

```
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
              minimum keysize is  768 bits
              default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024)
```

Generating a new keypair

Next, you must choose an expiration date. For most users a key that does not expire is adequate.

```
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

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.

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

Real name: Frettchen Rättchen
Email address: frettchen@dfki.de
Comment: Haustier
You are using the 'iso-8859-1' character set.
You selected this USER-ID:
    "Frettchen Rättchen (Haustier) <frettchen@dfki.de>"

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

Generating a new keypair

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

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

Enter passphrase:
```

Because this password 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 passwords by combining five English or German words.

Example: `distel ist landen kammer puffen`

Your public keyring

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

```
[psy@port-3108:~]$ gpg --list-keys
pub  1024D/B935225F 2005-01-27 Frettchen Rättchen (Haustier) <frettchen@dfki.de>
sub  1024g/4DE87B5A 2005-01-27

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

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:

```
[psy@port-3108:~]$ gpg --list-keys Tristan
pub  1024D/EFBF4915 2003-10-24 Tristan Miller (Research scientist) <tristan.miller@dfki.de>
uid                            Tristan Miller <psychonaut@nothingisreal.com>
sub  1024g/B40BE860 2003-10-24
```

Generating a revocation certificate

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.

```
[psy@port-3108:~]$ gpg --output revoke.asc --gen-revoke Frettchen

sec  1024D/B935225F 2005-01-27   Frettchen Rättchen (Haustier) <frettchen@dfki.de>

Create a revocation certificate for this key? 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?
```

The revocation certificate should be printed out and stored in a safe place.

Exporting a public key

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.

```
[psy@port-3108:~]$ gpg --armor --export Frettchen
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.6 (GNU/Linux)

eO5hydwuQO0IOr6kQmsXgELS3dc0TC0lNoTrIZv1uUtV3objRktpBL62UwCg/ESR
vKJ5yJ0KnFHrSvkJvq9/41UD/ihE8AOgYR6hFH0xKcZc03mpaqfto2B6PUHLi5yt
CjRDyhtZOq5RdN1+Bqll1uHt3yINClY9l0dIr5zZ6PO1QKJIU2gOtvnLyKrJ9VaZ
AUF7Hl/TZ9UDGNu6yyHI5CJ2Kc4XB0qOlCUd2lHfSq7N+rA3mv4zvFqd/uYcUzqZ
                                ...
PEOgHRvMud9mK0p/KBvffexKxzQlcVLjBQUY7BvU5wUbilNMYevw9m+0H+usBITD
EU1pseZmXp4NYelfFO8h7XdKplRs17LhlYhJBBgRAgAJBQJB+X8WAhsMAAoJEL0B
rqC5NSJfC+0AoIsTwOnzj0EJrx7deCHhM4z5KvUDAKDqTBAF9ZWYgh1wRKtTRZwj
2iptCA==
=9nBI
-----END PGP PUBLIC KEY BLOCK-----
```

Importing a public key

Most 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.

```
[psy@port-3108:~]$ gpg --import walter.gpg
gpg: key 85C62E2D: public key imported
gpg: Total number processed: 1
gpg:               imported: 1

[psy@port-3108:~]$ gpg --list-keys Sommer
pub  1024D/85C62E2D 2000-02-23 Walter Sommer <sommer@dfki.uni-kl.de>
sub  2048g/0F16F686 2000-02-23
```

Validating a key

- 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

- 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.

```
[psy@port-3108:~]$ gpg --fingerprint Walter
pub  1024D/85C62E2D 2000-02-23 Walter Sommer <sommer@dfki.uni-kl.de>
     Key fingerprint = 86F2 9A0D BBE2 89B6 F397  3934 082C 6529 85C6 2E2D
sub  2048g/0F16F686 2000-02-23
```

Signing a key

After checking the fingerprint, you may **sign** the key to validate it. Since key verification is a weak point in public-key cryptography, you should be extremely careful and always check a key's fingerprint with the owner before signing the key.

```
[psy@port-3108:~]$ gpg --sign-key Walter
pub  1024D/85C62E2D  created: 2000-02-23 expires: never      trust: -/-
sub  2048g/0F16F686  created: 2000-02-23 expires: never
(1). Walter Sommer <sommer@dfki.uni-kl.de>

pub  1024D/85C62E2D  created: 2000-02-23 expires: never      trust: -/-
 Primary key fingerprint: 86F2 9A0D BBE2 89B6 F397  3934 082C 6529 85C6 2E2D

     Walter Sommer <sommer@dfki.uni-kl.de>

How carefully have you verified the key you are about to sign actually belongs
to the person named above?  If you don't know what to answer, enter "0".

   (0) I will not answer. (default)
   (1) I have not checked at all.
   (2) I have done casual checking.
   (3) I have done very careful checking.

Your selection? (enter '?' for more information):
```

Listing key signatures

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.

```
[psy@port-3108:~]$ gpg --check-sigs Walter
pub  1024D/85C62E2D 2000-02-23 Walter Sommer <sommer@dfki.uni-kl.de>
sig!        85C62E2D 2000-02-23   Walter Sommer <sommer@dfki.uni-kl.de>
sig!3       B935225F 2005-01-27   Frettchen Rättchen (Haustier) <frettchen@dfki.de>
sub  2048g/0F16F686 2000-02-23
sig!        85C62E2D 2000-02-23   Walter Sommer <sommer@dfki.uni-kl.de>
```

Public key servers

- Most 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

- 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.

```
[psy@port-3108:~]$ gpg --keyserver wwwkeys.eu.pgp.net --send-key Walter
gpg: success sending to 'wwwkeys.eu.pgp.net' (status=200)

[psy@port-3108:~]$ gpg --keyserver wwwkeys.eu.pgp.net --recv-key EFBF4915
gpg: key EFBF4915: "Tristan Miller (Research scientist) <tristan.miller@dfki.de>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
```

Encrypting a document

- 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!

```
[psy@port-3108:~]$ gpg --output doc.gpg --encrypt --recipient Walter
```

Decrypting a document

- 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.

```
[sommer@serv-3100:~]$ gpg --output doc.txt --decrypt doc.gpg
```

Symmetric encryption

- Documents may also be encrypted with a symmetric cipher instead of public-key cryptography.
- The symmetric cipher offers higher security, but should only be used 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`.

```
[psy@port-3108:~]$ gpg --output doc.gpg --symmetric doc.txt
Enter passphrase:
```

Signing a document

- 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 distributions are signed so that users who download them can verify that they have not been modified since they were packaged.
- E-mails are 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

- The option `--clearsign` causes a text document to be wrapped in an ASCII-armored signature.
- Clearsigning is used most often for e-mail messages and Usenet postings.

```
[psy@port-3108:~]$ echo "Hello, world!" >hello.txt
[psy@port-3108:~]$ gpg --clearsign hello.txt

You need a passphrase to unlock the secret key for
user: "Frettchen Rättchen (Haustier) <frettchen@dfki.de>"
1024-bit DSA key, ID B935225F, created 2005-01-27

[psy@port-3108:~]$ cat hello.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hello, world!
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.6 (GNU/Linux)

iD8DBQFB+cUmvQGuoLk1Il8RAiw5AJ46quj41qP0prQVv8Zpyeki6Z/WrQCgljYB
xUYHD/FazJNPyluZwOyjGCM=
=VHb7
-----END PGP SIGNATURE-----
```

Detached signatures

- 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.

```
[psy@port-3108:~]$ gpg --armor --output hello.sig --detach-sig hello.txt
```

Verifying signatures

- 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.

```
[psy@port-3108:~]$ gpg --verify hello.sig hello.txt
gpg: Signature made Fri 28 Jan 2005 06:06:47 AM CET using DSA key ID B935225F
gpg: Good signature from "Frettchen Rättchen (Haustier) <frettchen@dfki.de>"
```

Trust model

- 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
- 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

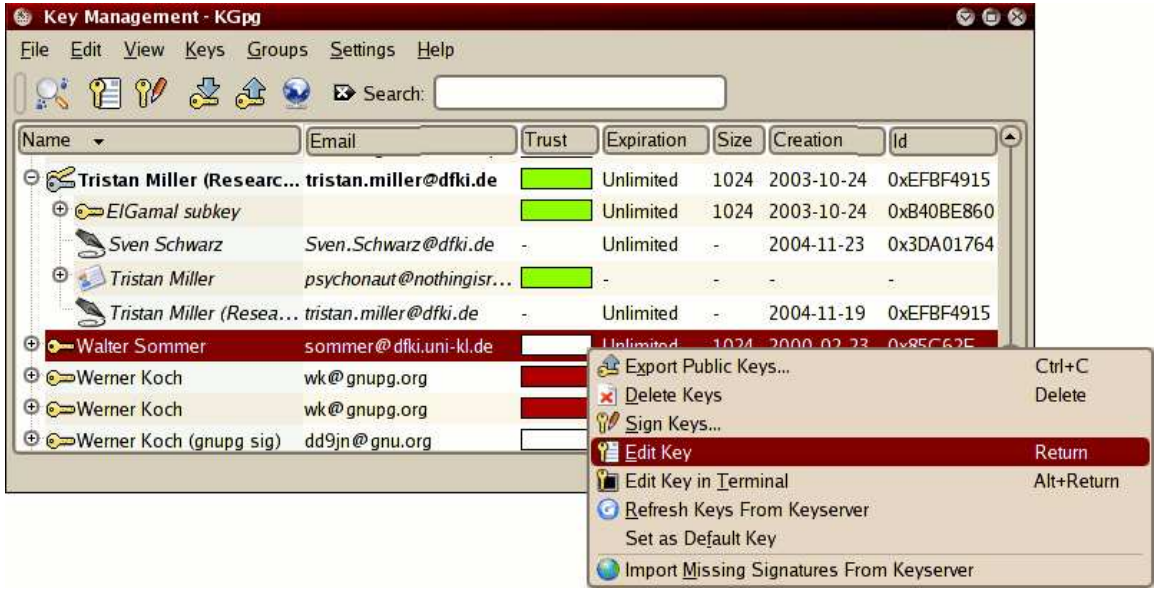

```
[psy@port-3108:~]$ gpg --edit-key Walter
pub  1024D/85C62E2D  created: 2000-02-23 expires: never      trust: -/f
sub  2048g/0F16F686  created: 2000-02-23 expires: never
(1). Walter Sommer <sommer@dfki.uni-kl.de>

Command> trust
pub  1024D/85C62E2D  created: 2000-02-23 expires: never      trust: -/f
sub  2048g/0F16F686  created: 2000-02-23 expires: never
(1). Walter Sommer <sommer@dfki.uni-kl.de>

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

 1 = Don't know
 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

- Formerly, a key was considered valid only if you signed it personally.
- 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.

Key management

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

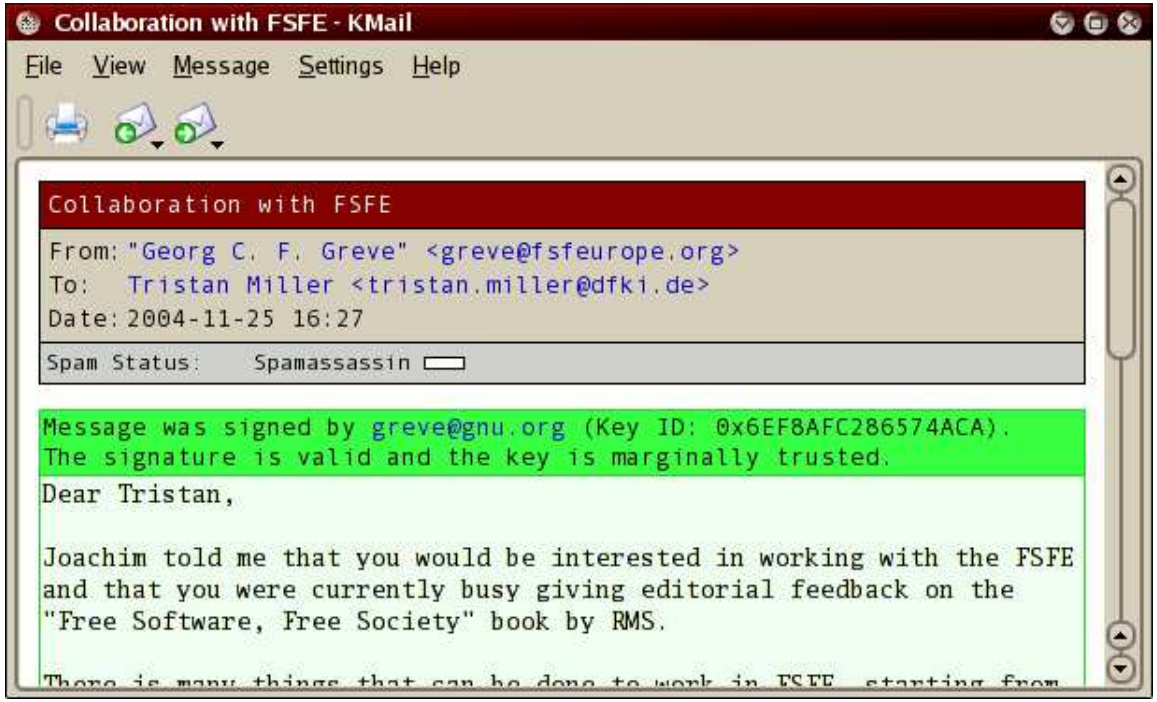


E-mail integration

Many e-mail clients now support digital signatures. For each e-mail account, you can associate a public key for signing and encryption.

E-mail integration

Signatures on messages are automatically checked.



E-mail integration

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