GNUPG

HIGH LEVEL CRYPTOGRAPHY

- **NO ONE** will spy on your e-mails anymore
- **MEGA PROTECTION** for home and business
- **100% FREE SOFTWARE**

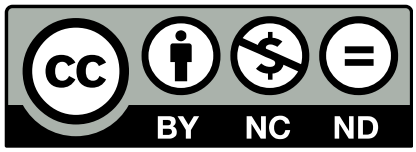
AND
JUST ONE WORD:

PRIVACY

Copyright

Except when otherwise stated, this document is licensed under the following license:

Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License



That means you are free to:

- **Share** – copy and redistribute the material in any medium or format.

Under the following terms:

- **Attribution** – You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

- **NonCommercial** – You may not use the material for commercial purposes.

- **NoDerivatives** – If you remix, transform, or build upon the material, you may not distribute the modified material.

You can find more information about the license in the following link:

https://creativecommons.org/licenses/by-nc-nd/4.0/

Images copyright

Icons and Logos

Below are listed the images used in this book that are licensed under different licenses than this book. If you use any of these images you must follow their respective license terms as well.

The table below contains an identification of the image, the pages they are used, the license they are subjected and a link to their original content and/or license terms.

Logo	Page(s) used	License	Link
Green shield	1	GNU GPL	LINK
Red check mark	9, 10	GNU FDL v2.1+	LINK
NEW seal	13	CC BY ND	LINK
Key	13, 14, 15, 16, 45, 47	Public Domain	LINK
Male avatar	13, 14, 15, 16	CC BY	LINK
Female avatar	13, 14, 15, 16	CC BY	LINK
Verified seal	15	UNKNOWN	LINK
Padlock	16	GNU GPL	LINK
Mushroom	45	CC BY	LINK
EFF logo	47	EFF Copyright's Policy	LINK
GNU Project logo	47	GFDL v1.3 FAL v. 1.3 CC BY SA 2.0	LINK
IETF logo	47	IETF Trade Mark Usage Guidelines	LINK

Screenshots

All screenshots used in this book are licensed under the same terms of the book.

Table of Contents

Introduction

Welcome!

This guide was developed to help people understand what is cryptography, how it works and why they should use it. It deals primarily with e-mail cryptography, but there are also sections covering offline usage for local files.

Most people don't use cryptography simply because they don't know what it is, or they have erroneous ideas about it, such as being extremely complex, expensive and even outlawed. They are also not aware of the risks and dangers they face by not using it.

On the other side, the largest IT companies and e-mail providers also do not provide adequate information on this issue and resist implementing cryptography in their systems because it would increase their costs without giving them direct benefits.

We believe that cryptography is essential and necessary to maintain privacy and security of digital communications, and the more people adopt this technology, the more it becomes an indispensable item which will come together with every service.

This guide is destined to laypersons, so it is easy to understand and there is no need of prior advanced knowledge. You will learn how to install and configure all the necessary programs to have cryptography working in your system, and by the end you will be able to communicate with other people with maximum privacy and security.

We hope you enjoy it. Thanks for choosing this guide!

Best regards,

The Golden Keys Team
https://www.goldenkeys.org

PART 1

BASIC CONCEPTS

In this part you will learn:

- *What is Cryptography*
- *Why use Cryptography*
- *How Cryptography works*
- *The Anatomy of a Key*
- *What is GnuPG*

CHAPTER 1
What is cryptography?

Cryptography is the process of encoding and decoding information, messages and files using secret code with the purpose of offering privacy and security. This can be accomplished through machines, computer programs, or both.

Cryptography is always used when there is a need to transmit information in a secure way between two parts, ensuring that only the sender and the receiver will be able to decipher its original content. Anyone who tries to intercept it without authorization will only see a bunch of symbols and codes that makes no sense, and will not be able to decipher it.

Cryptography has existed for thousands of years, but for most part of its history it was considered a military tool, being used almost exclusively by governments and armies due to its high cost and complexity.

Things started to change with the emergence of personal computers and the internet. With the advent of those technologies, high level cryptography became affordable to the general public at the same time that the need for more secure systems was increasing.

Today cryptography is essential for many areas in our society and it is employed in a variety of systems including personal computing, mobile phones, banking systems, magnetic cards, ATM machines, electronic commerce, data storage, wireless devices, etc. However few users are aware of cryptography's presence in our life, and even fewer know how to use it or how it works internally.

CHAPTER 2

Why use cryptography?

There are several reasons why you should always use cryptography on your personal and professional communications, they all come down to your privacy and security. Below we list 7 points so you can better understand the importance of this technology.

E-mail is extremely insecure

E-mail is one of the most insecure systems ever simply because it was not designed to be secure. Messages travel through many machines, networks and even countries, and they can be intercepted in many different ways by anyone who has access to them. By default their contents (text, images and attachments) are transmitted without any security at all.

You are constantly being monitored

E-mail providers (such as Hotmail, Gmail, Yahoo) store all your sent and received messages for indeterminate time – possibly forever – even after you have erased them from the trash bin or terminated your account. They do it for two reasons: to sell you more services and advertisements, and to collaborate with government surveillance programs.

The registers of your e-mail communications may be – and often are – stored in machines located in countries different than yours, and once they are in another jurisdiction they are subjected to that nation's laws and there is virtually nothing you can do to claim the right to privacy you may have in your country. This may happen even if you haver never been in those countries.

It can be used at home or in business

Cryptography can be used at home or in business and it works with a wide variety of devices such as personal computers, mobile phones, tablet computers, workstations, servers, complex network infrastructures and others.

It can be used for personal communications with family and friends, to store sensitive information, to backup sensitive information, to encrypt the whole disk, to send and receive files, to provide a secure channel to access one's machine, among other uses.

It increases your credibility

When you offer a secure means for people to communicate with you it demonstrates how much you value and worry about their privacy and security. This is especially true in business where there is often a high volume of sensitive information being exchanged, but it also applies to personal relationships.

You convince more people to use it

To send and receive encrypted messages requires that others you communicate with also use cryptography, so if you start using it you will naturally tell other people about it. Given the advantages and benefits of using cryptography, many of them will eventually embrace it, and it is easier to start doing something when others they know are already doing.

Another advantage is that it is possible to use cryptography and still communicate with people who don't use it. The communication will be unencrypted of course, but at least you don't have to limit yourself to only one group of people.

It's free

There are many types of cryptography systems for different needs with varying prices. The system we present to you in this book, GnuPG, is 100% free both in terms of price and in freedom to use it. You can set it up in any environment you want without having to pay for licenses, royalties, fees or require any type of authorization, and the program is powerful enough to be used in a single computer and in the infrastructure of a multinational corporation.

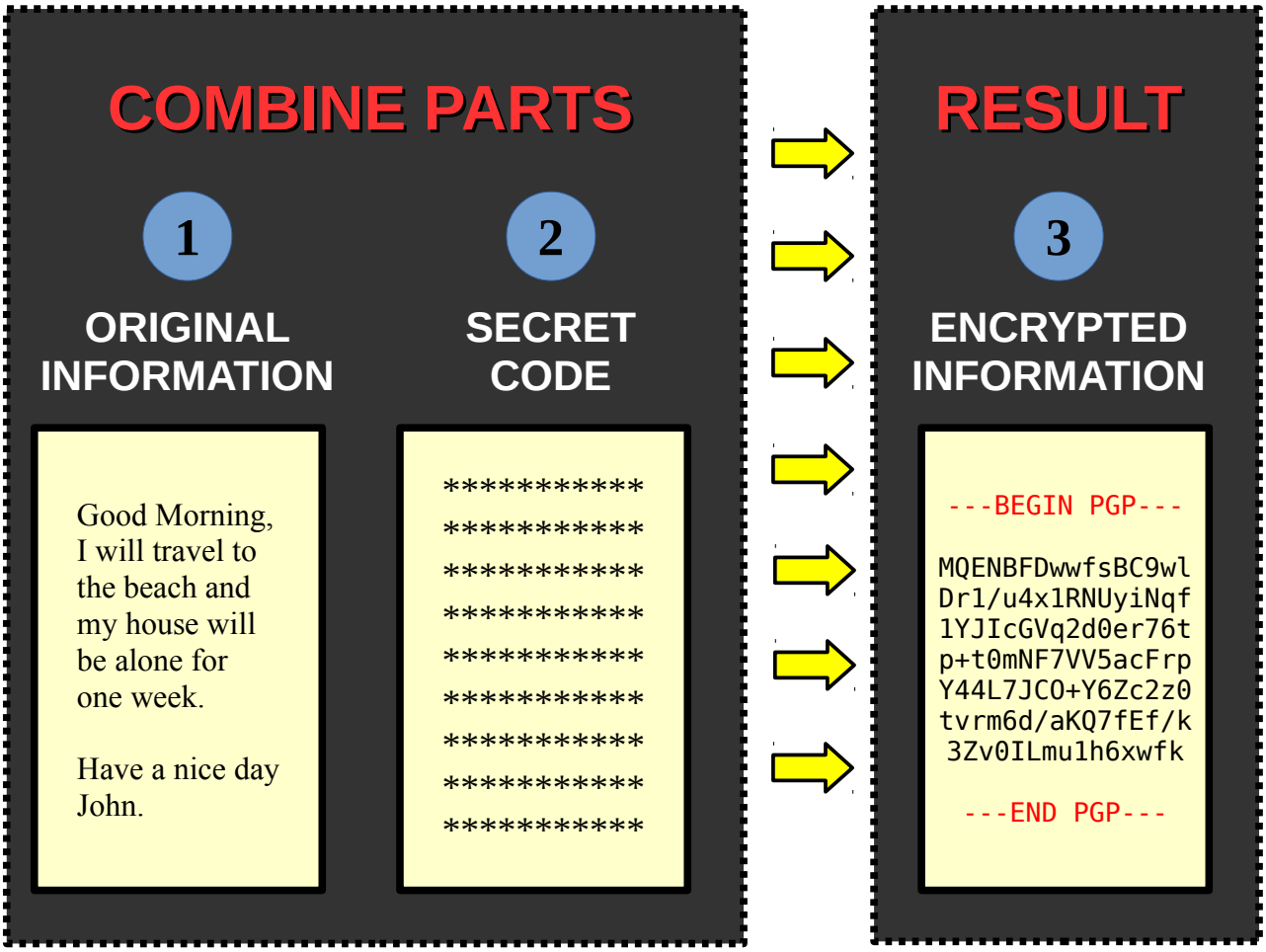
Your privacy

Last but not least, your e-mail communications are private and they should remain private. It doesn't matter if you send a message telling about a new restaurant in the city, your credit card number with the password (yes, people do it), or a picture of yourself naked (yes, people do it A LOT), it is not of anyone else's business and it is up to you to ensure your privacy remains private.

CHAPTER 3

How cryptography works?

The basic idea is to **shuffle** the original information with the secret code, resulting in the encrypted information. The power, strength and security of encryption lies exactly in how these parts are shuffled. The diagram below illustrates this process:



That's it, your message is now encrypted and ready to be sent. For the person be able to decrypt it he will need to posses the secret code, which will be covered in the next section.

Now let's see the two main types of encryption methods: symmetric and asymmetric.

3.1 Symmetric Cryptography

Symmetric cryptography is the simplest of all and you probably have used it many times. The word symmetric means “equal”, which means that to encode and decode a file the password is the same.

The most basic example is when you save a file with password. It doesn't matter if you save it for yourself or for others, the password to open it is always the same.

Symmetric cryptography is faster, simpler and more economic than asymmetric cryptography because it does less mathematical calculations, which in turn uses less machine resources (e.g.: electricity). It is also more compatible with other systems and it is very secure.

However its main problem lies not in strength, but in the *transmission* of the secret code. When you send an encrypted file to another person you also have to send the password so the person can open it, and symmetric cryptography does not provide any means to send the password in a secure way.

You cannot send the encrypted file through e-mail and the password wrote in the message body because that is too obvious and risky. You could send the password by phone, SMS or letter, but these methods are also insecure and could be easily intercepted. You could deliver the password personally, but this is very inconvenient and sometimes inviable.

So how do you do it? As you see the major problem of symmetric cryptography is to transmit the password in a convenient and secure way. If the password is compromised, anyone can access the file and even modify it.

Another disadvantage is that if you use a password, you automatically know it, and others could coerce you to reveal it, as in the customs, through a law order or under interrogation.

It is because of these reasons that symmetric cryptography is recommended for local files that stay stored locally (such as backup copies) or files to be transfered through physical media.

3.2 Asymmetric or public key cryptography

Asymmetric cryptography, also known as public key cryptography, was created to solve the problem of transmitting the secret code that symmetric cryptography poses.

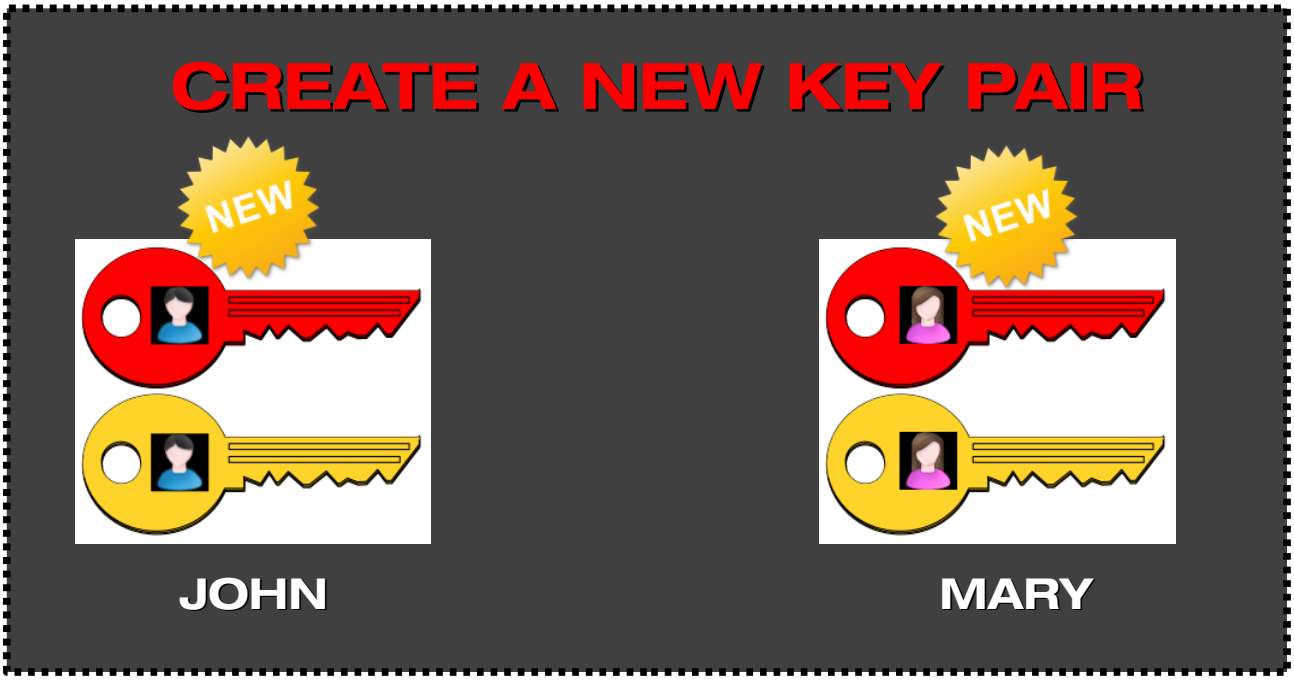
Simply speaking, in public key cryptography instead of using a single code equal for all, it is used **a code with two parts**: your part and the other person's part. This way only her will be able to decrypt the information you send to her. If someone intercept this information along the way he will not be able to do anything because he does not have the necessary part of the code.

These “parts” are actually called keys, which are public and private. The example below illustrates this more easily:

Let's imagine that John wants to send an encrypted file to Mary using public key cryptography. Here are the steps they have to follow to accomplish this:

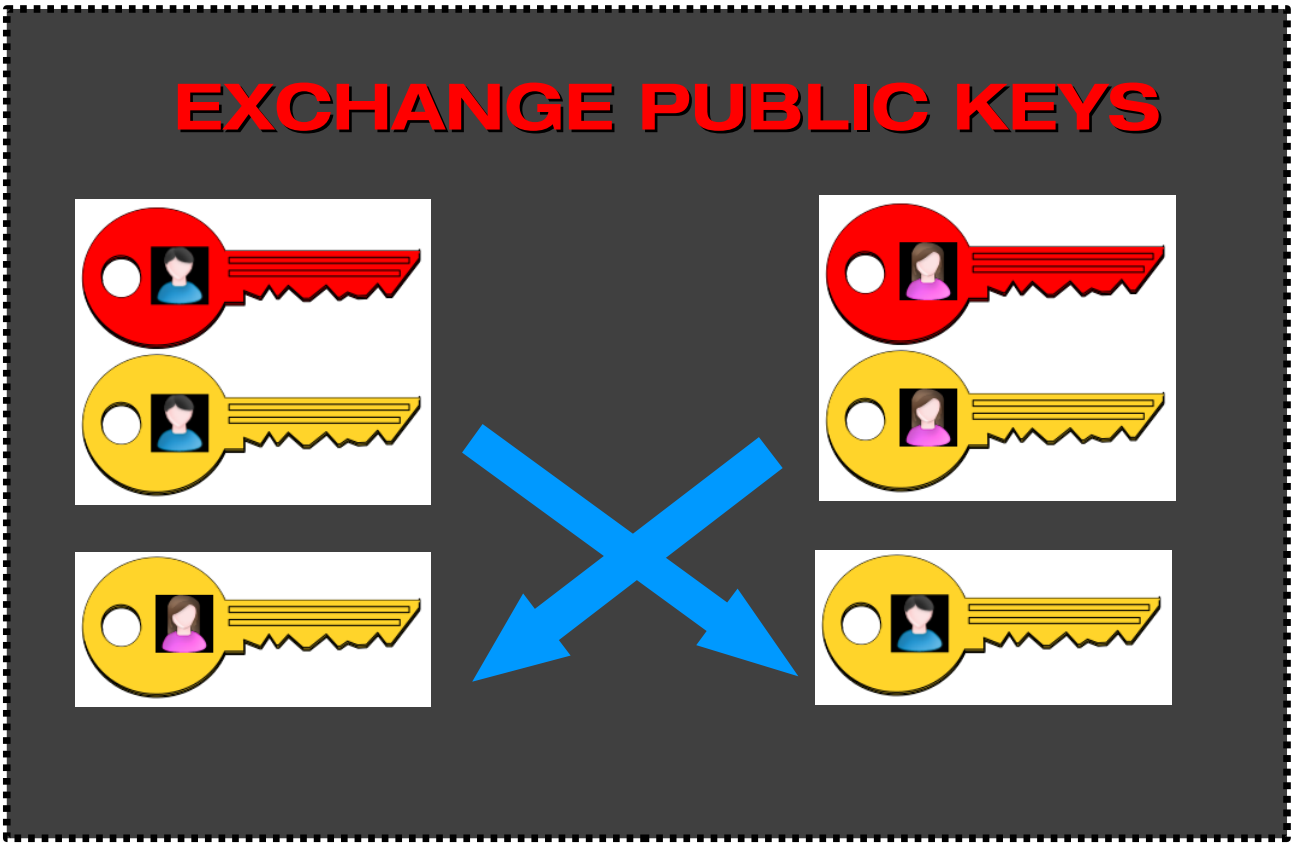
1 – Create a key pair

First each one of them creates a key pair containing a private key (red) and a public key (yellow). This step is covered with details on chapter 6.



2 – Exchange public keys

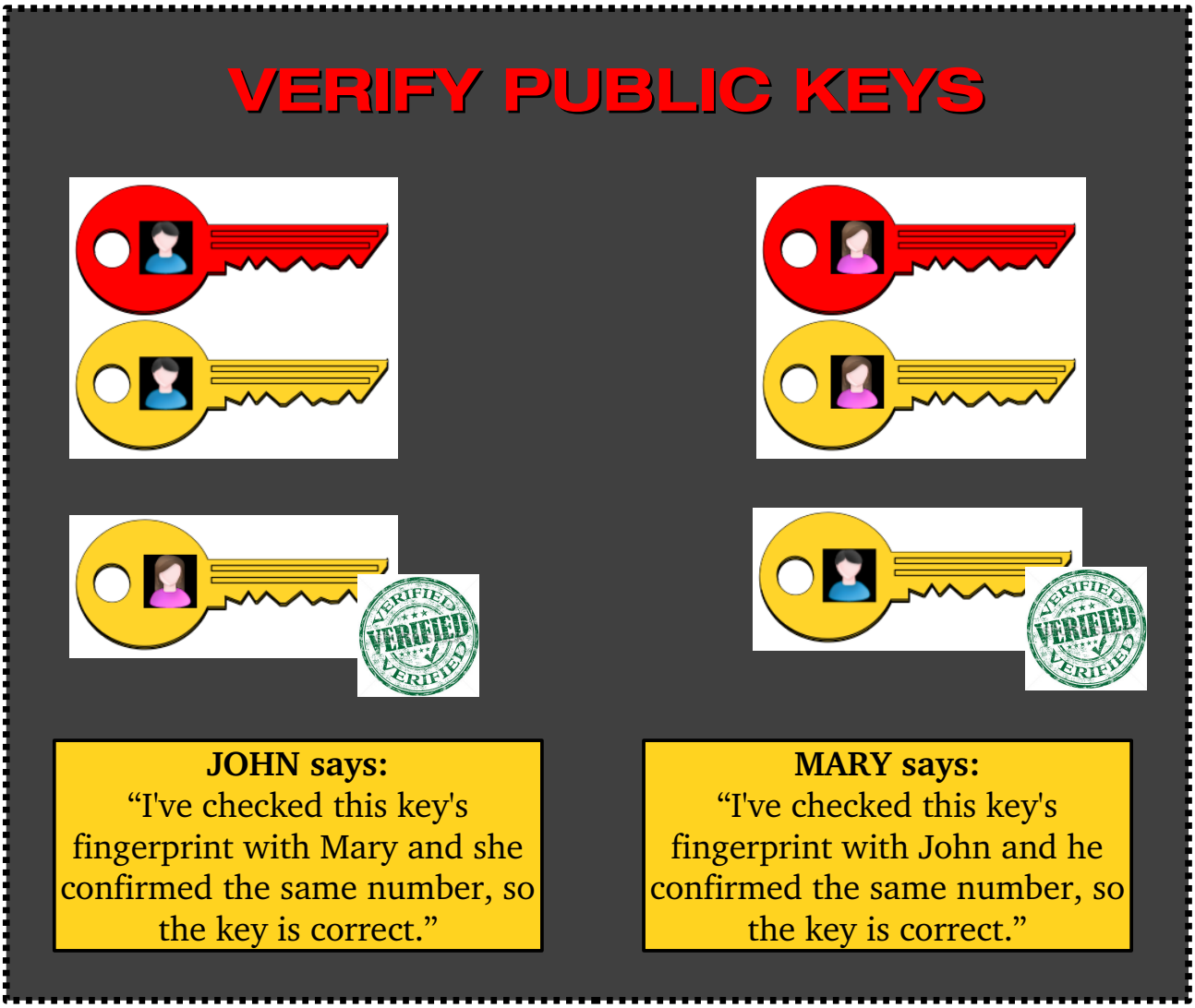
Each one of them sends a copy of their public key to the other, since the purpose of the public key is to give it to others. There are several ways to do it, the most common is to send it via e-mail (discussed in chapter 8.4), but it is also possible to publish the key in a key server, in a personal website, or deliver it through physical media (such as a CD-ROM).



3 – Verify the received key

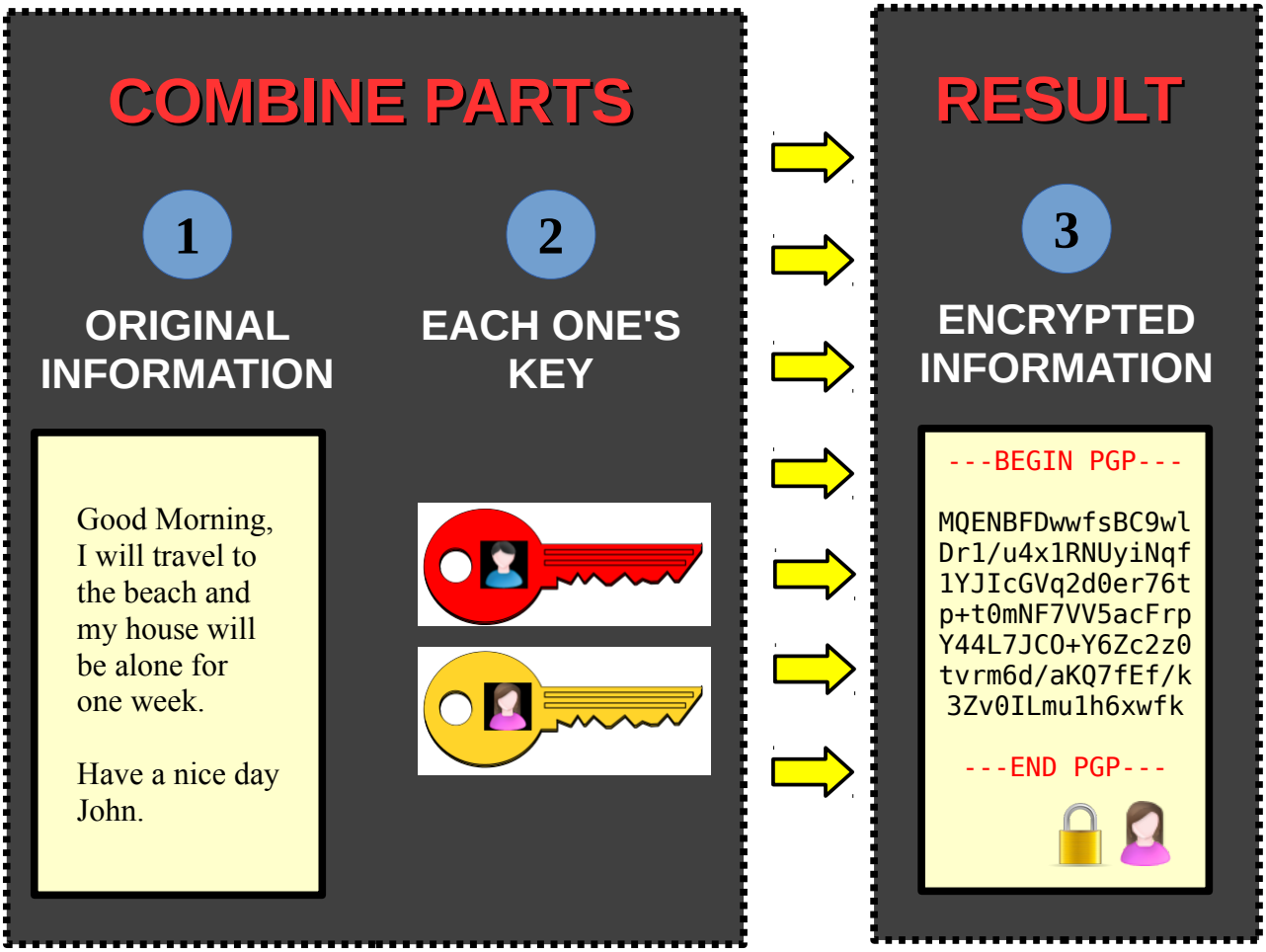
Now each one of them possesses their own key pair and a copy of the other person's public key. They must verify the other's public key to confirm they received it correctly. This is the most important step because it ensures that the key they received was not twisted or modified along the way.

Verifying is a simple process: every key comes with a number (a digital fingerprint) and all they have to do is to check this number with the sender to ensure it is correct.



4 - Encrypt a file and send it

To encrypt a file to another person John just chooses the file he wants to send and the file will be encrypted exclusively to that person.



The original message is combined with the sender's private key (John's) and the receiver's public key (Mary's), resulting in an encrypted file that only the receiver (Mary) can decrypt.

Now the resulting file can be sent to Mary through any means (such as e-mail) because only her can decrypt it, because to do it she needs her private key and the sender's public key.

CHAPTER 4
Anatomy of a key

A key pair consists of a public key and a private key. The public key is the key that you distribute to others, and the private key is the key that you keep with yourself. Keys are basically a stream of text that contains all the necessary information that identify them. Here we provide text and graphical representations of keys. Keys are always stored in key rings.

Keys can realize up to 4 different operations:

- Sign and Verify (**S**)
- Encrypt and Decrypt (**E**)
- Certify (**C**)
- Authenticate (**A**)

In this book we cover the first two operations, which are discussed with more details in their respective chapters.

The example below illustrates the basic information contained in a key pair. The private key is the red one, and the public key is the yellow one.

4.1 – A key pair

Here you can see a key pair containing a private key and a public key.

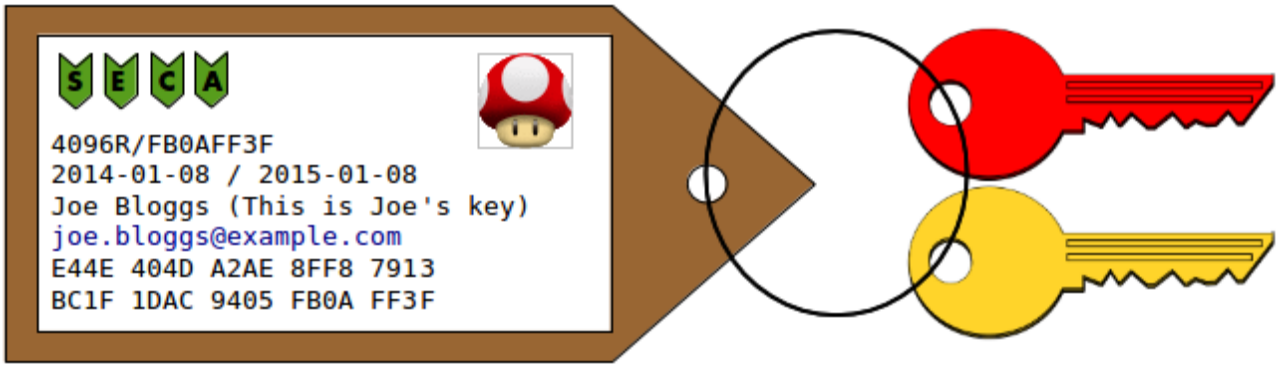


Figure 1: Information contained in a key pair

4.2 – Parts of a key

Below is explained what means each part of the key.

- **S / E / C / A**

  These letters are called 'flags' and they represent the operations the key is able to realize, as described in the beginning of the chapter. In our example the key can realize all the four operations, but not all keys can realize them all.

- **4096R/FB0AFF3F**

  4096 is the length of the key in bits, which normally varies between 1024 and 4096. R is the type of the key, in this case RSA. FB0AFF3F is the key identifier (ID).

- **2014-01-08 / 2015-01-08**

  Those are the creation date and expiry date (if exists) respectively. They are presented in the format YYYY-MM-DD.

- **Joe Bloggs (This is Joe's key) joe.bloggs@example.com**

  The key owner's full name, comment and e-mail address.

- **E44E 404D A2AE 8FF8 7913 BC1F 1DAC 9405 FB0A FF3F**

  This is the key fingerprint. This is a unique hexadecimal number with 40 digits and every key has one. Every time you receive a key you have to confirm it with the key's owner, because it is the only guarantee you have that the key is in fact of the person who claims to be its owner, and was not twisted or modified along the way.

- **Image**

  It is possible to add an image to your key, but this is not recommended for three reasons: it makes your key heavier, some programs have problems to deal with them, and it presents a false sense of security.

All these attributes are present in keys, so you can check them every time you obtain a key, as well as others can check them with your key.

4.3 – Keys in a keyring (graphical visualization)

Keys are always stored in key rings., which are managed by GnuPG. You can easily backup or export your whole keyring to use it in another machine that you own. Below there are 3 keys from different owners in a keyring, which were obtained from the owners' respective websites. Since the keys are not ours, only their public part are available.

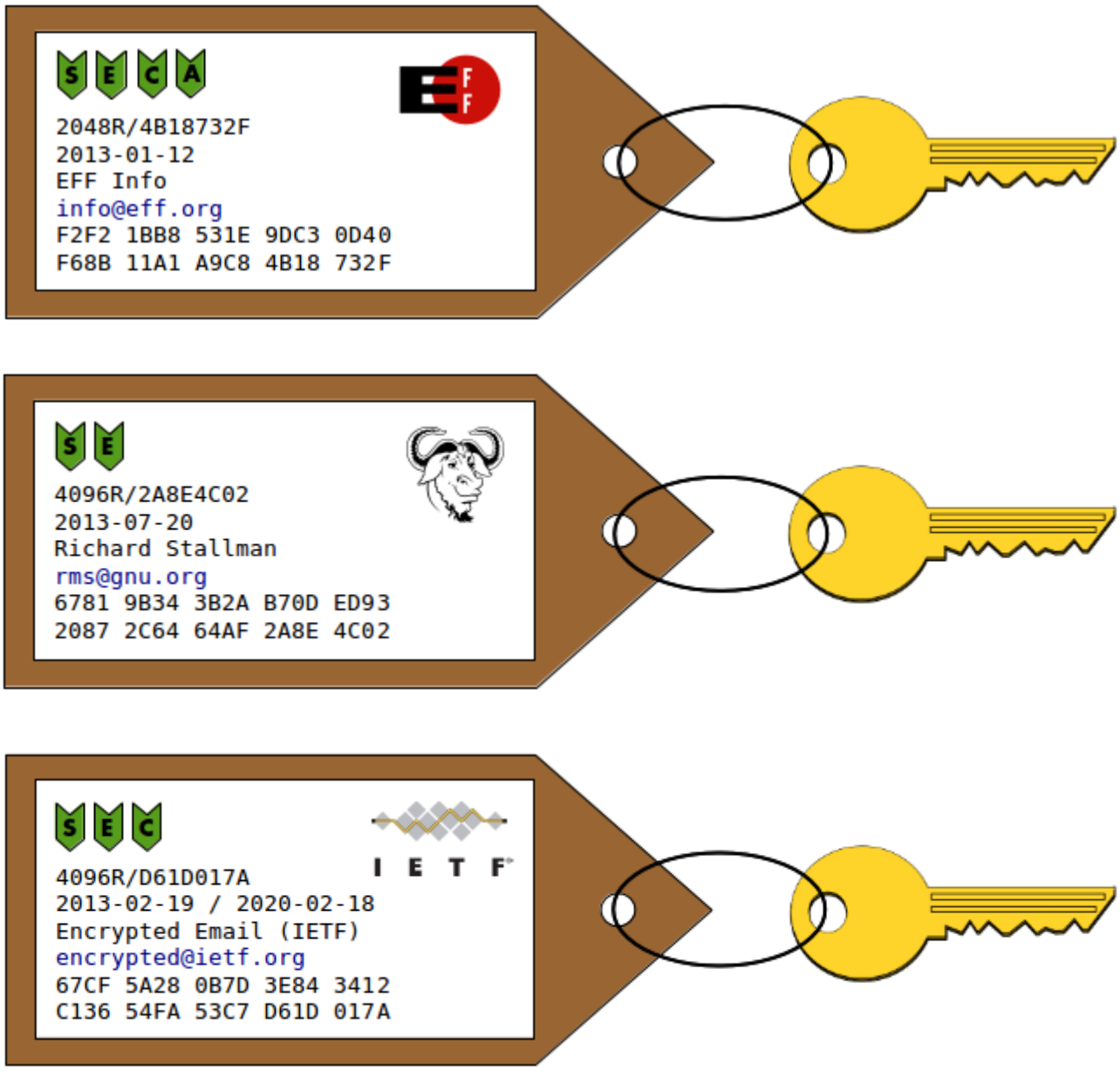


Figure 2: Public keys in a keyring

This is how keys would look graphically in our example. All these keys do not have image, they were added only to make comprehension easier.

4.4 – Keys in a keyring (text visualization)

Here are the same keys displayed in command line:

```
$ gpg2 --list-keys --fingerprint

pub   2048R/4B18732F 2013-01-12
      Key fingerprint = F2F2 1BB8 531E 9DC3 0D40  F68B 11A1 A9C8 4B18 732F
uid                  EFF Info <info@eff.org>
sub   2048R/75DA5789 2013-01-12

pub   4096R/2A8E4C02 2013-07-20
      Key fingerprint = 6781 9B34 3B2A B70D ED93  2087 2C64 64AF 2A8E 4C02
uid                  Richard Stallman <rms@gnu.org>
sub   4096R/62853425 2013-07-20

pub   4096R/D61D017A 2013-02-19 [expires: 2020-02-18]
      Key fingerprint = 67CF 5A28 0B7D 3E84 3412  C136 54FA 53C7 D61D 017A
uid                  Encrypted Email (IETF) <encrypted@ietf.org>
sub   4096R/D4E938B1 2013-02-19 [expires: 2020-02-18]
```

4.5 – How a key looks like

A key is just a file that contains all the attributes mentioned above. Below is an example of how a key looks like in ASCII-armored format:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.19 (GNU/Linux)

mQINBFLNwKoBEADgHEPqhCbz3/hB0sMZUQlERDFgpTl+m5zHBk7XzIHxzG+ijrmY
HGgF4qurzL2RFxjxFHQEvHcAzHxgWnqQNl+lh1QkVtn34ku9o4euGneM+sjbEbcc
S8i7pfBCmjG6dw61xRK64RwKebXYHbmfq4Yx6QVPOHeVsrOY9pFrAgWTWxyWUnQk
LzKfpIxupQIPiVIUE8xQNBfdJSUiK+I/8OIc9fbml/GF3FEp+4BytWsoNFWc4seK
9Y3ybZJPMaKj/bfde4UCH2p9LcpRM87F34uKIzB66so4sbkqNu7kUabdX+skG5tO
rOTBoEddHlBXtVXpG0oGu1gRL8A00CUM519AWukjXyOTPnOHSz2ECsStisFmzBtE
0+Qoop4lVrwHO9QQK3p7aoG+tVqHnhUr6P9f3udKbljWzdXZznANCu5USPpM10JR
tPRn0zQmRPucEwUkdcZ3ieINoJ9vIPJU23o27WbNUMm4WyFwcfawkh7xWXDMoynu
6XIch+10e/EkkSv+/In/HRwxTQhx8RiSQV79HEFsHfwFt8325c23dgZ8UseEsM8M
8KoALtZANBFamNaV7AIf9Xsob9/iLj81bU3qTaj02dsse4WgK+tAzsnGBOsGpNq7
rq2Qii/oEJq3XgyC0fFK69WfqQ+kduV1sJZxVgUgUjLYf5FQlNXnvdvf6QARAQAB
tDdKb2UgQmxvZ2dzIChUaGlzIGlzIEpvZSdzIGtleSkgPGpvZS5ibG9nZ3NAZXhh
bXBsZS5jb20+iQI+BBMBAgAoBQJSzcCqAhsDBQkB4TOABgsJCAcDAgYVCAIJCgsE
FgIDAQIeAQIXgAAKCRAdrJQF+wr/P051D/9mwGEyyFDcJCXXtKjMUK7RMy2K9lb1
J+26qfFUM8zi7wOVC43p+8l43zHPt6X84FP0XFx31FCpUWpiqwBfyTucXfPnA0JC
JoWk2oOGOXJOm9O5n+tr1HllE+T19ANnUC+Z3pvHycfbs+rB+SSW+BjOYsdG1/0x
mMOUExWEw+hQaa3/86Y6Tog0gxJssQhXSDBnImvSLDa6NdK/Nblbl5lOm2zkTlxR
v5AEJuPeUu7p2C7y49SpRflSyeFPV5raC291HqSVagCMZpVVwIdmvzuqlh+fgG73
AUwRYsiIm3i9KVxze8vtEfzze6VuM9iXLuDhkw27N/12YV+VMga20PhuOTtsM169
7v4jAuxgdZlrWMm0hAOuccp0z4vtMsrkb0Ur/ti5GEnGX6bzqv1i5S1B+Uzo4kU1
7+qpw30KofgMRhefAJHMZMzQ98Fe/MxDIIV6XOFrqocTco1+4J4nP/eJ4y+D/rYA
ban0woVzy7GOxBOTPiOFXbFPYGj1vGCd38y2wLjoPD0Sb18kUKZz1JAjkLIevNTH
ZfY9vJSOdKHz24oc3XJpKxTdYtLFuwy5d4fTjxnfCZV2FPDDQvKPiLM6qVBANUbz
xpjJiwEWM/7l0u7CKGmHU1PPJ3KnatqAD7Sc2z5qC60GfQ43Cy6QG4Ha05jB2UbE
CNRN6aseHeOwtrkCDQRSzcCqARAAr9ifvPQeFOpetw/O/+4x37cKIzCRfTlfrSte
wo0y4WJjn58IzBysFQ3EX82w4k4Vb90RR9TUnKP+p3JlIQtiltx26oawJfQLi8bK
1go5f+qS1z/cUFTyK8zlH4XZ0oTAb77zNmzFv613dQdud+H3fbwkUcDJOXBT8yE/
bNeIiKmF/Zuk9fWVAQ8vAUr6TjQxCXhfXxl25yz/FM3/d679Ss6itgfEytXCKQug
BhH0GlkUUZe5sPcGsak4MY62/H73QreEEGXi9CSu6+JKE1p3ONkpGsXiBY9Tod7f
Wm2XjnvbHyV+ZbISiEa/c2LGR5a/7p0shxhKeSHN5y6RaPFDsxW5UQfQuLkKhgOO
cnNGGyLpZN9fs93ZbQLGlE0Ujn6umgU3EWlvHyd9opYh7HOgkZ5KZRnfKXnLiaX+
c0Ao7176AkJJ94O4DQ2I/UQzQunm0RKT7JRsW1s0OlsKR2sRUGdBnnAo4uFMVj+Z
+yANobyMFBlI5FY/L00KmtNnOmXLPfTESHpI6O5jzOLJ3an7ibh9iof7KwWHrk9o
B6bxGU2YWljfP1rcaFBA/Xpm+mzK84Nshx7XKwyQpyRHhSxUAhSNkEBnXstd15mE
ZUDAH43dWikDJthac2vgoFMOPK/Vi/8cTymLFAHQX0WdjNGaIBmnO1y2XIyqTA7P
v4gBfX0AEQEAAYkCJQQYAQIADwUCUs3AqgIbDAUJAeEzgAAKCRAdrJQF+wr/PwLV
D/0bPEJ+h30uvYaR3jeSYFYPfflKMoADpWsexG9cW3zKWznywBkGj6CHJhmZgivn
+61zgT8W08elyz667UMnnFloHWCGQ13NxNzZlCq7w0UZ9Ip7vtD7G986Icy3D8cd
iVdoAYS4NwBSbxV3kxNO1Y6rlqf4MEfNAVfPX0UuRTo9XkjYOEnITzIvDa68lm7U
51a9oMCZm50eENsWkTgx8OYemxlS1NB4tAzZl4q7TjskwTo2NtJA4z9w+8isI2zy
dVOIpVxX0ZBdjN1Ru4gsy7P54eE6QXUV118aKpsBx9YWYEw81H8J0XS5RX5qTYcS
vzz1ukXN58VEq5bP96zpgRRmxSSrsgZlCUov0J7oxHNb/0nuJqm/862DsavFzEIF
O/4c0CAq6aqU4hcMGeHMPWUfqNAJeJZDl3WTe4kok73RFTGE2Zg+yvSZEJBVVeP/
sG+pjehxLTn1HbnY9FoiArknFVBCatwX0BmIbkNm4vhwgpuf2UhVT4uaMSjI1tpq
WOtxP/VDTasG6WKOXIqdZCJ+pCKDIeo4FGb97FXpRX/3jHgcZbWBszpKk5yt5raE
wafwqHSjwNbe+ws7tpv/ADAjcChZx1ToTHRhP4C+3zQbaP9w3EPDpZjrQhcz5ZUB
bsxVMyVeRuL6BnckGoqsL/lEw+qdZPCKsmdwxJ5+3FbbWg==
=4n8Z
-----END PGP PUBLIC KEY BLOCK-----
```

This is how a key looks like in ASCII-armored format. If you open any key with a text editor you will see a similar result (unless the key is in binary format).

The term ASCII, when used throughout this guide, simply means *text* or *in text format* (actually it's more complex than that, you can check this Wikipedia article for more information: https://en.wikipedia.org/wiki/ASCII).

The key in this example is the same key used in section 7.1. If you copy and save it in a text file, you can import it to your keyring as described in chapters 12, 15 and 16. You can also check that the fingerprint is the same. However our example key does not have image, the image was only added to the figure for illustrative purposes.

4.6 – Conclusion

Keys are the basic component of asymmetric. cryptography, also know as public key cryptography. They store information about the key's owner that allows users to identify the owner, as well as other technical aspects regarding its security and capabilities.

Keys are editable, so some of their attributes can be further changed after creation. To be used they must be stored in key rings., which GnuPG creates automatically. They can also be transported to other machines or exported as backup copies.

CHAPTER 5
What is GnuPG?

GnuPG, short of GNU Privacy Guard, is a sofware (computer program) that aims to offer privacy and security to digital communications by encrypting their contents. It is often used together with e-mail to send and receive messages, but it can also be used to protect information that stay stored locally, such as backup copies.

GnuPG is a free (libre) alternative to the original PGP software developed by Philip Zimmerman in 1991, since PGP was – and is still not – free (libre). PGP stands for Pretty Good Privacy and it was incredibly popular since the beginning. As a consequence other softwares started to appear that used the same system. Realizing that a standardized version would be beneficial to all, Mr. Zimmerman proposed a standard called OpenPGP, which is an open, standardized, patent and royalty-free protocol for PGP.

GnuPG is compliant with the OpenPGP protocol, which makes it it compatible with other alternatives available in the market. However the largest advantage of GnuPG is that it is 100% free software, which means it respects your freedom, so you are free to:

1. Use the program in any way you wish.
2. Study how the program works internally, and adapt it to your needs if you wish.
3. Distribute original copies of the program to others.
4. Distribute modified copies of the program to others.

You can do any of those things without asking permission to anyone or any company. Besides, GnuPG also has several other advantages:

- ✔ It is completely free (as in priceless, or costless).
- ✔ It has been in constant development for 15 years.
- ✔ It is free from patents or royalties.
- ✔ It can be used at home, in business, in governments and in public systems.
- ✔ It offers military level cryptography, the highest available today.
- ✔ It is compatible with most popular operating systems, including Microsoft Windows, Apple OS X, Android, iOS, GNU/Linux, BSD, and other *NIX-like distributions.

GnuPG is one of the most powerful cryptography softwares available in the market today, and it is relatively easy for the layperson to obtain, set up and use it. It is also compatible with many popular applications such as e-mail clients and chat programs.

5.1 – What GnuPG does and does not do

Although a very powerful software, there are some things that GnuPG cannot do, so to avoid misconceptions let's see some of the things GnuPG can and cannot do.

GNUPG DOES...

- ✔ **Encrypt and decrypt your messages**
  - Your messages are encrypted, including the attachments, so no one knows their contents and what they are about, only the recipient can decrypt them.
- ✔ **Sign your messages**
  - Your messages are signed to ensure they were sent from yourself and not twisted or modified along the way by an intruder.
- ✔ **Prevent *others* from building a profile of you based on the terms you use**
  - Since they are not able to know the contents of your message, they cannot build a profile of yourself based on the words you use, which they could use to monitor you or offer you intrusive advertising.

GNUPG DOES NOT...

- ✗ **Encrypt the subject of the messages**
  - There is no standard yet that allows e-mail subject to be encrypted.
- ✗ **Prevent *others* from knowing your location and IP address**
  - Your IP address will still show up in the message, which can be used to track your location, and eventually track you down.
- ✗ **Prevent *others* from knowing the e-mail header**
  - Your e-mail header is a bunch of information related to your machine that goes hidden in every e-mail message, such as your IP address, your local time, your e-mail client, your operating system, etc.
- ✗ **Prevent *others* from knowing to whom you contact with and how often**
  - The recipient of the e-mail message is not hidden, and thus they can know to whom you are sending the message.
- ✗ **Prevent *others* from storing your messages**
  - They may store your messages for future decryption. E.g.: they cannot decrypt the message now, but in the future new technologies or systems may emerge that could break today's “unbreakable” cryptography.
- ✗ **Prevent *others* from knowing the size of your messages**
  - Messages size often give a clue about what you are sending. Heavier messages almost certainly mean that there are attachments included.

5.2 – Additional suggestions

Here are listed some simple additional suggestions to improve your security online:

- **Always use cryptography for all messages, not only for the important ones.**
  - Don't use cryptography only for the important messages because it is too obvious you are sending something important – instead use it with all messages.
- **Use cryptography with all your contacts.**
  - Try as much as possible to use cryptography with all your contacts instead of using it with just the ones you consider most important.
- **Do not use revealing subject lines**
  - There's no point in encrypting your message if the subject line is revealing, such as “Pictures of myself naked” or “My credit card number with password” or “My house will be empty for two weeks”. Instead be discreet.
- **Use a free/libre e-mail client**
  - Although cryptography is supported by many e-mail clients, including proprietary ones such as Microsoft Outlook, it is recommended that you use it with a free/libre e-mail clients such as Mozilla Thunderbird, because due to their open nature they are often much more secure.
- **Use a strong password**
  - The best cryptography system in the world won't help you a bit if you use weak, easy-to-break passwords, so always use very strong passwords.
- **Use a powerful antivirus and keep your system clean**
  - You may use the best cryptography system in the world plus very strong passwords but this is completely unuseful if your system is compromised with virus or any other type of malware. So always use original version software and keep your system clean and up to date.

5.3 – Conclusion

GnuPG is a very powerful software that does a lot, but it's not just installing and it magically do everything to secure you. You also have to do your part as well.

PART 2

CONFIGURING AND USING PROGRAMS

In this part you will learn:

- *How to install GnuPG*
- *How to install and configure Mozilla Thunderbird*
- *How to install and configure Enigmail*

GnuPG in six easy steps

Below are listed the six basic steps necessary to the use of e-mail cryptography, which are all covered in this part of the manual. When you are finished you will be able to send and receive e-mails with maximum of security.

CHAPTER 6
Installation

The first step to use GnuPG is to install it. GnuPG is distributed completely free of cost, so you can easily obtain it online. Here we cover GnuPG installation in the most common operating systems, Microsoft Windows and *NIX distributions, but it is also available for other systems as well, such as Apple OS X.

6.1 Microsoft Windows

There is a tool bundle developed for Microsoft Windows called Gpg4win, which includes GnuPG, additional software and documentation.

Fortunately Gpg4win comes with graphical tools and native integration with Windows Explorer file manager, making it easier and more intuitive to use.

1 – Download Ggp4win

Gpg4win can be downloaded in this website: http://www.gpg4win.org/download.html

Click on the first button to download the full version, as indicated in Figure 1.

Figure 3: Gpg4win download button

2 - Choose the language

Choose the language used for setup.

Figure 4: Installer language

3 - Opening screen

This is just the opening screen. Click Next to continue.

Figure 5: Opening screen

4 - License screen

Here is presented a copy of the license. Click Next to continue.

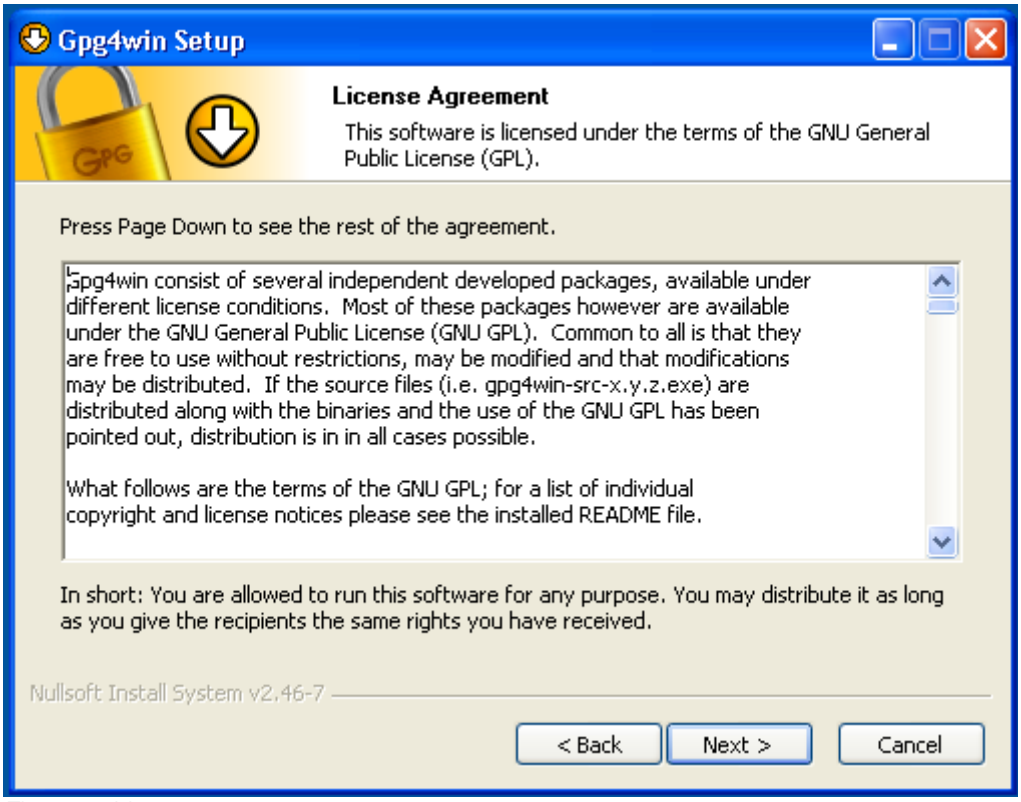


Figure 6: License screen

5 - Choose components

Here you can choose the components that will be installed together with GnuPG. Below there is a description of each component:

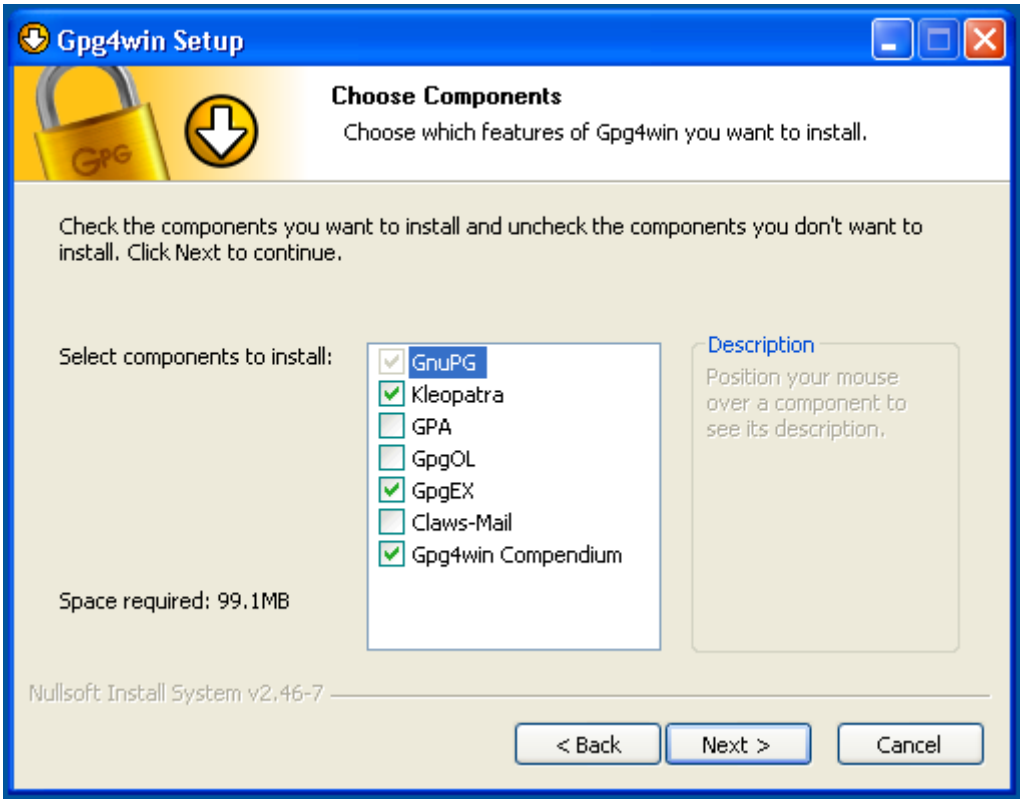


Figure 7: Choose components

GnuPG: The main software of the package, it cannot be deselected.

Kleopatra: A graphical alternative to GnuPG. It is recommended to install it since it is very powerful and simplifies a lot GnuPG usage.

GPA: Another graphical alternative to GnuPG. Although smaller and faster than Kleopatra, it is less powerful and often present many bugs.

GpgOL: GnuPG extension for Microsoft Outlook. Only install it if you use this software. Note that in this tutorial we use Mozilla Thunderbird in our examples, but you are free to use other e-mail clients if you want.

GpgEX: GnuPG extension for Microsoft Explorer. It is recommended to install it.

Claws-Mail: A lightweight e-mail client. You don't have to install it if you use another e-mail client, or if you follow this tutorial since we use Mozilla Thunderbird here.

Gpg4win Compendium: Gpg4win documentation in English and German.

6 - Choose install location

Here you can choose a different install location if you want. Click Next to continue.

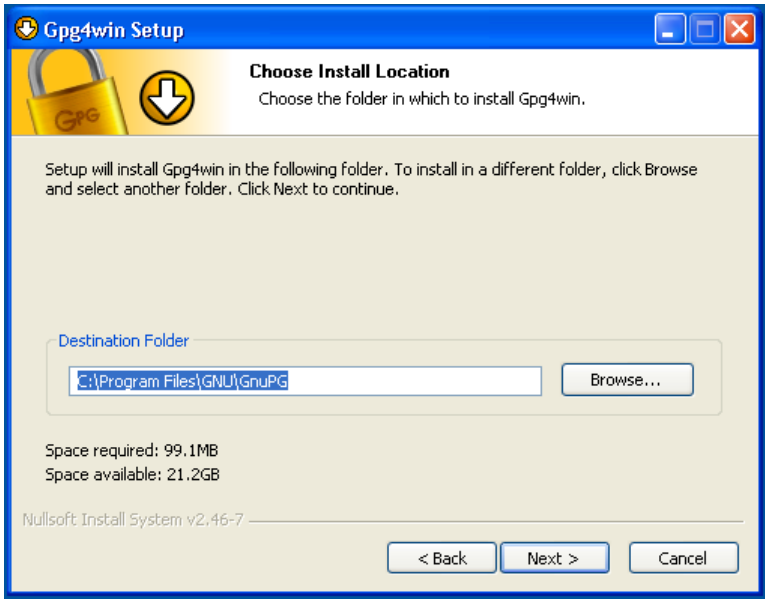


Figure 8: Choose install location

7 - Choose where you want the links to show

Here you can choose where you want the start links to show. Click Next to continue.

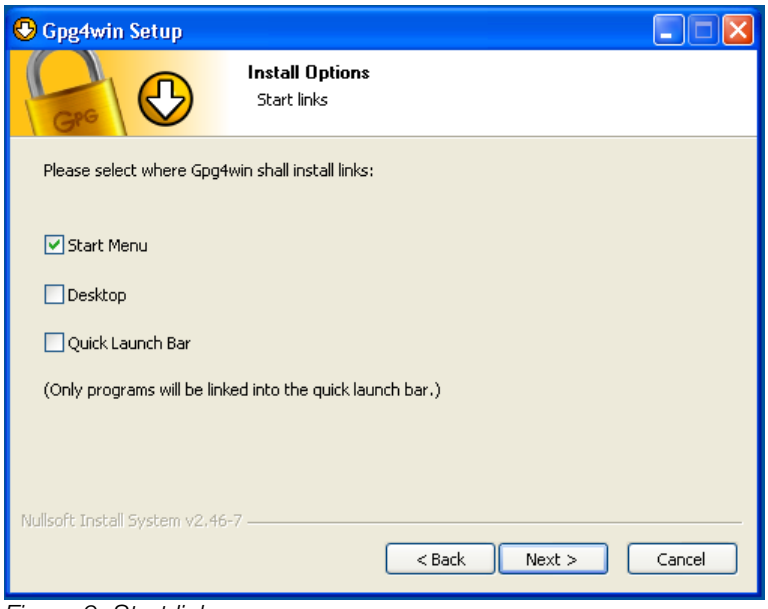


Figure 9: Start links

8 – Choose the folder name in Start Menu

Here you can choose the name GnuPG will have in Start Menu. Click Next to continue.

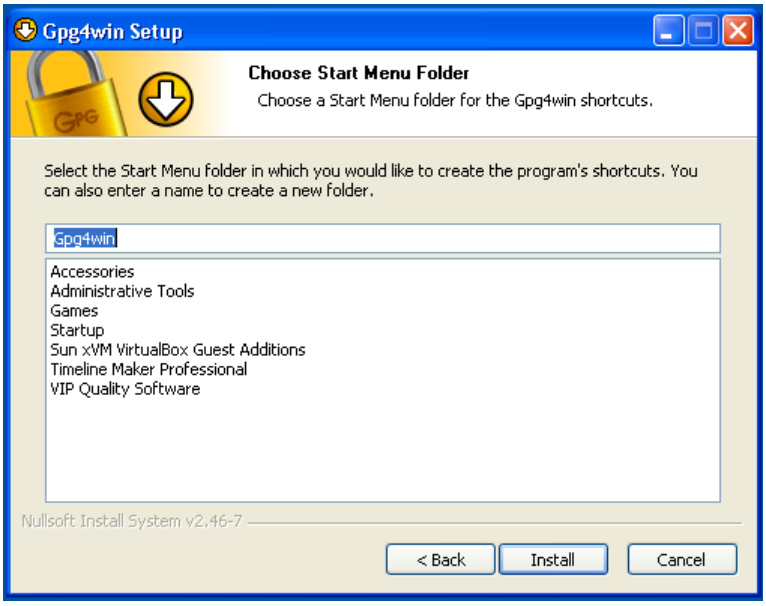


Figure 10: Start Menu name

9 – Wait for the installation to finish

Wait until the installation finishes.

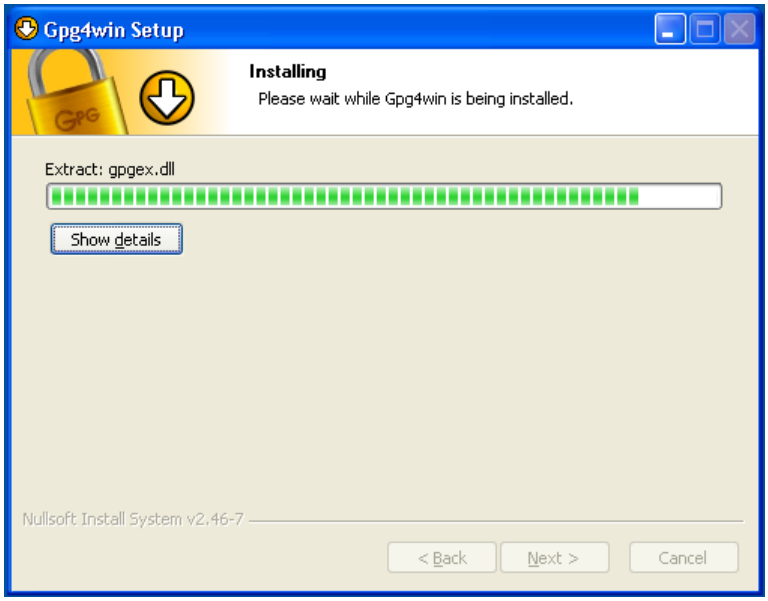


Figure 11: Installation progress

10 – Installation complete

GnuPG is now installed and ready to use. Click on Finish.

Figure 12: Installation complete

6.2 *NIX systems

In *NIX systems most tasks are done through the CLI (Command Line Interface), also known as the Terminal Emulator. If you want to use GnuPG with a graphical interface you need to install a separate software. Here we cover how to install both.

6.2.1 Installing GnuPG

GnuPG usually comes installed by default in most *NIX distributions. To check if it is installed in your system use the commands below:

```
# To check if GnuPG version 1.x is installed
$ gpg --version

# To check if GnuPG version 2.x is installed
$ gpg2 --version
```

We will install GnuPG version 2.x because this is the most recent GnuPG version, and it is the version we use throughout this book, but you can also use version 1.x if you wish, since they are compatible with each other.

Here are the commands to install it in the most common *NIX distributions:

Arch Linux:

```
$ sudo pacman -S gnupg2
```

Debian, Mint, Ubuntu:

```
$ sudo apt-get install gnupg2
```

Fedora, CentOS:

```
$ sudo yum install gnupg2
```

Gentoo, Sabayon:

```
$ sudo emerge gnupg2
```

Mageia:

```
$ sudo urpmi gnupg2
```

FreeBSD, OpenBSD:

```
$ sudo pkg_add -r -v gnupg2
```

6.2.2 Installing Seahorse

Seahorse is a graphical program that can be used as an alternative to GnuPG command line interface for some functions, such as creating and deleting keys, importing and exporting certificates, modifying keys, etc. Seahorse depends on GTK to work properly.

Arch Linux:

```
$ sudo pacman -S seahorse
```

Debian, Mint, Ubuntu:

```
$ sudo apt-get install seahorse
```

Fedora, CentOS:

```
$ sudo yum install seahorse
```

Gentoo, Sabayon:

```
$ sudo emerge seahorse
```

Mageia:

```
$ sudo urpmi seahorse
```

FreeBSD, OpenBSD:

```
$ sudo pkg_add -r -v seahorse
```

CHAPTER 7
Create a key pair

A key pair is the basic element of public key cryptography and it consists of a private key and a public key. They are necessary for you to communicate securely with other users. Here we explain how to create a key pair in three different ways: text mode (works in both systems) and graphical mode (separate versions for Microsoft Windows and *NIX systems).

7.1 Text mode

1 – Start GnuPG key generation wizard

Type the command below to start the GnuPG key generation wizard.

```
$ gpg2 --gen-key
```

2 – Choosing the key type

The first step is to choose the type of key you want. We will choose the first option which is the default option, RSA and RSA. Enter 1 and press ↵.

```
gpg (GnuPG) 2.0.20; 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? 1
```

3 – Choosing the key length

Now you will choose the length of your key. As a general rule, the larger the length of the key, the more secure and harder it is to crack it, so we will choose 4096 bits which is the maximum allowed. Enter 4096 and press ↵.

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

4 – Choosing the expiry of the key

The key may have an expiry that ranges from days until years, or simply not have any expiry at all. To create a key with expiry just follow the example below:

0	The key never expires.
4	The key expires in 4 days.
6w	The key expires in 6 weeks.
2m	The key expires in 2 months.
5y	The key expires in 5 years.

You can choose the period that is more adequate to your needs by following this pattern, just change the values accordingly. It is always possible to change the values later.

In our case we will make a key without expiry, so enter 0 (zero) and then press [Y] to confirm.

```
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) 0
Key does not expire at all
Is this correct? (y/N) y
```

5 – Entering personal data of the key

Here you will enter your data as shown below. They will be used to create your key and will be associated with it. The comment is optional.

If you have more than one e-mail address you can associate them later to your key, instead of having to create a new key pair for each e-mail address. When you finish type O (letter O) and press [↵] to confirm.

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

Real name: John Doe
Email address: john.doe@example.org
Comment: John's key
You selected this USER-ID:
    "John Doe (John's key) <john.doe@example.org>"

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

6 – Entering your password

This is one of the most important steps of the whole process. The strength and security of your key are directly related to your password. There is no point in using the best encryption system in the world if you use a weak password, so choose a VERY STRONG password!

Enter your password twice and press ↵. Depending on how GnuPG is set up in your system you may have to type your password on the terminal or in a new window. If you type it in the terminal it does not show up while you type.

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

7 – Generating a new key

Now that you entered all your data GnuPG will generate a new key. To generate a really secure key it needs unexpected data chains, and the best way to obtain it is realizing diverse activities during this process.

Try opening and closing some heavy programs, move the mouse cursor a lot, or open a text editor and type many random text.

This process takes about 5 minutes, so have patience. During this time GnuPG may show text similar to the image below.

```
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.
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.
```

8 – Key generated

Congratulations, you have just created your first key pair! :)

```
gpg: key 9C08F860 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   4096R/9C08F860 2013-12-23
      Key fingerprint = 5259 EB00 049D 9C06 5D1F  08C7 6A4F 6BF2 9C08 F860
uid                  John Doe (John's key) <john.doe@example.org>
sub   4096R/9677ED61 2013-12-23
```

9 – Verify your key

To verify your key just type the command below:

```
# Listing available public keys with fingerprint:
$ gpg2 --list-secret-keys --fingerprint
sec   4096R/9C08F860 2013-12-23
      Key fingerprint = 5259 EB00 049D 9C06 5D1F  08C7 6A4F 6BF2 9C08 F860
uid                  John Doe (John's key) <john.doe@example.org>
ssb   4096R/9677ED61 2013-12-23
```

If you did everything correctly you should see a summary of your key on the screen, including the key's fingerprint, which is a unique code that only **this key** in the world has. When you send your public key to other people, the only way they can be certain that the key they received is yours and was not twisted along the way is by confirming the key's fingerprint with you.

7.2 Microsoft Windows

1 – Open Kleopatra

Open Kleopatra and click on File → New Certificate, or press Ctrl N

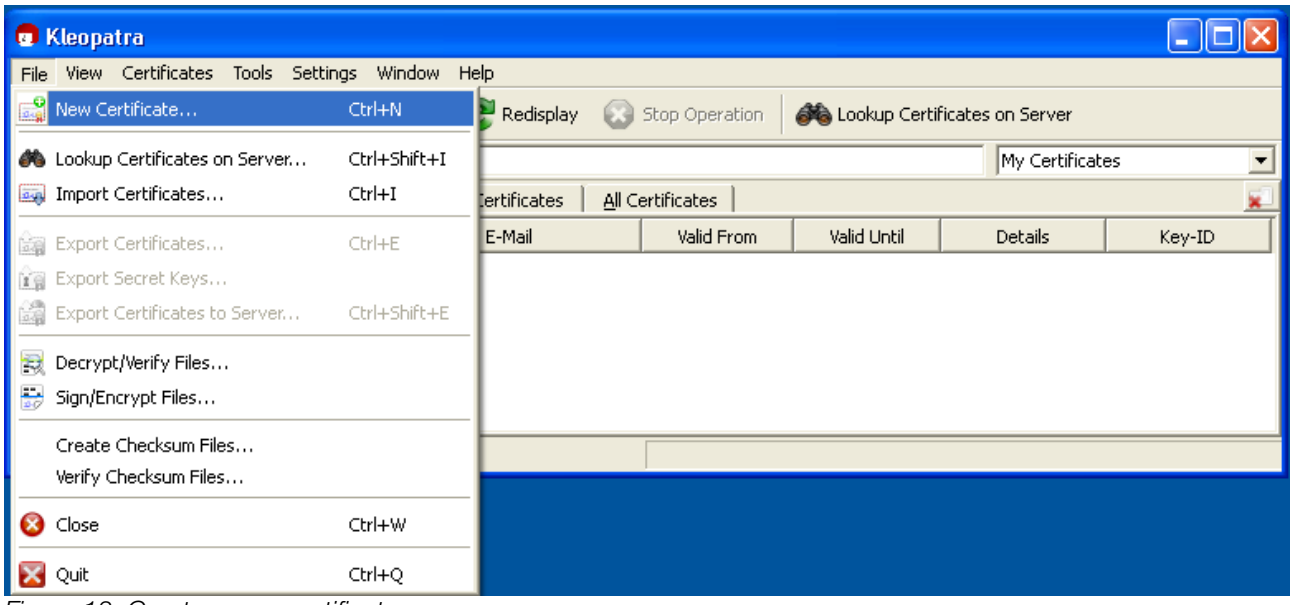


Figure 13: Create a new certificate

2 – Choose the first option

Choose the first option 'Create a personal OpenPGP key pair'.

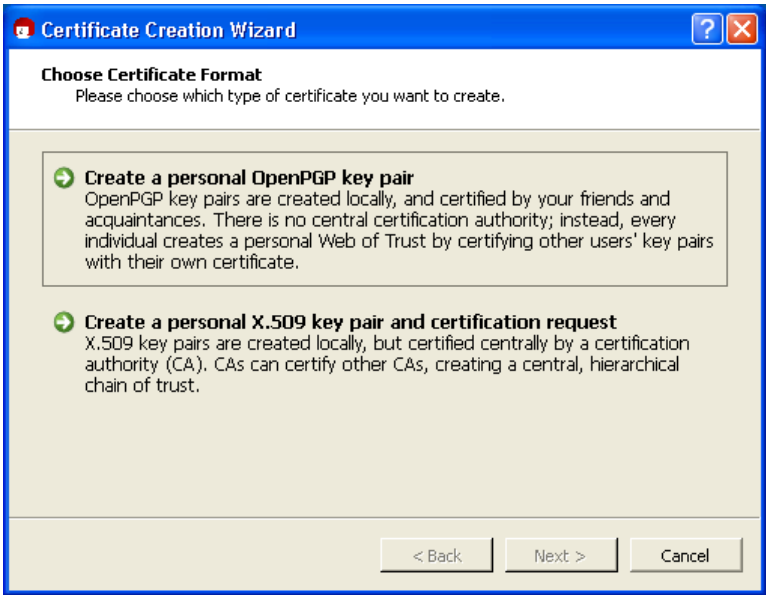


Figure 14: Choose the first option

3 – Enter basic details of your key

Here you will enter your basic personal details which will be part of your key and will be visible to anyone who has your key. The name and e-mail address are required, while the comment is optional. When you are done click on Advanced Settings button.

Figure 15: Enter basic details of your key

4 – Set advanced settings

Here you will set the advanced settings of your key.

Key Material: Select RSA as the key type and set the key length to 4,096 bits.

Certificate Usage: Check options Signing and Encryption.

Valid until: You can define any value you want. Uncheck it if you want no validity.

When you are done click on OK button. You will return to the previous screen. Just click Next to proceed.

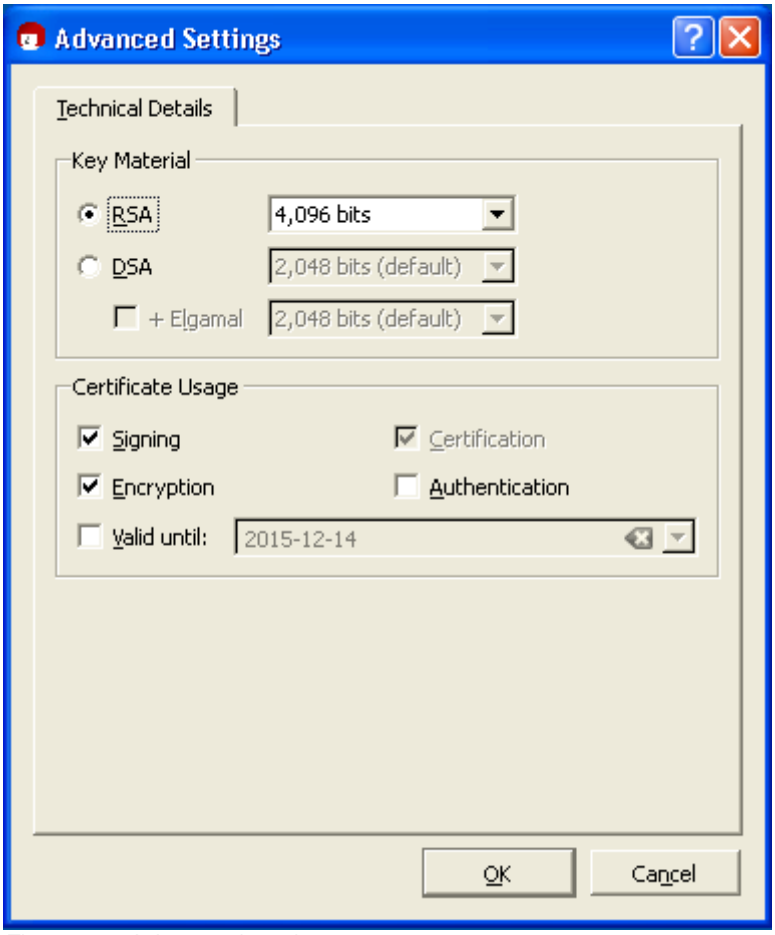


Figure 16: Advanced settings

5 – Review details

Review all details that will be part of your key. If you would you like to change anything just click on the Back button, otherwise click on Create Key button to create your key.

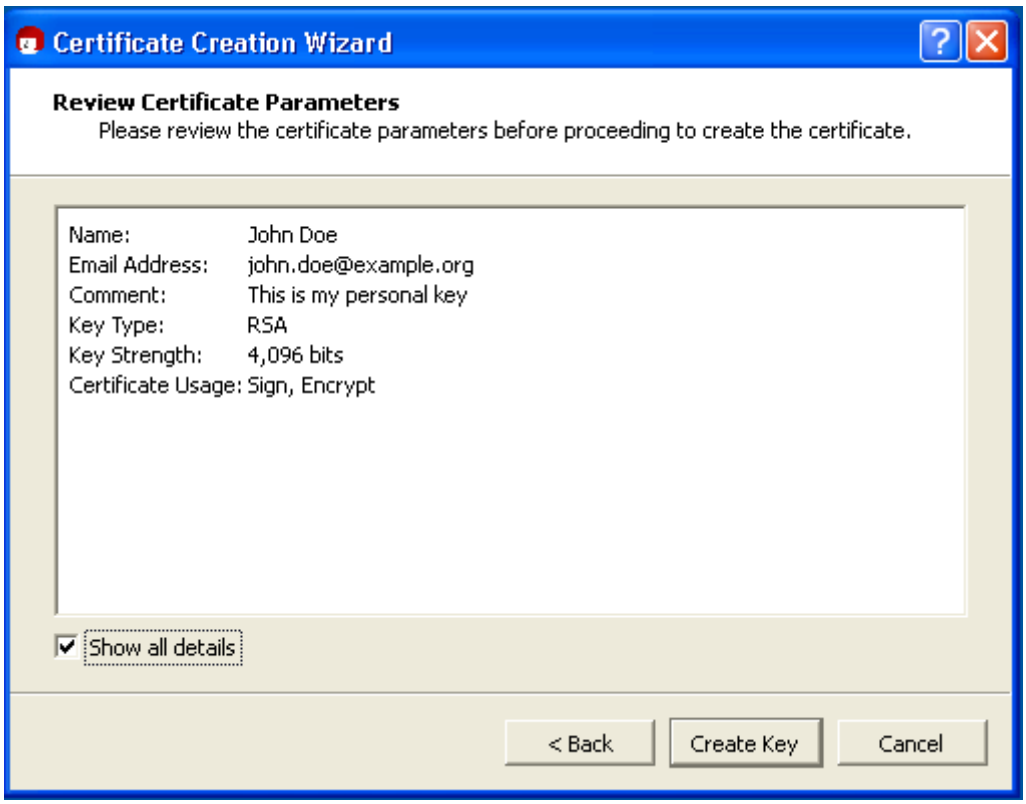


Figure 17: Review details

6 – Choose a password

This is one of the most important steps of the whole process. The strength and security of your key are directly related to your password. There is no point in using the best encryption system in the world if you use a weak password, so choose a VERY STRONG password!

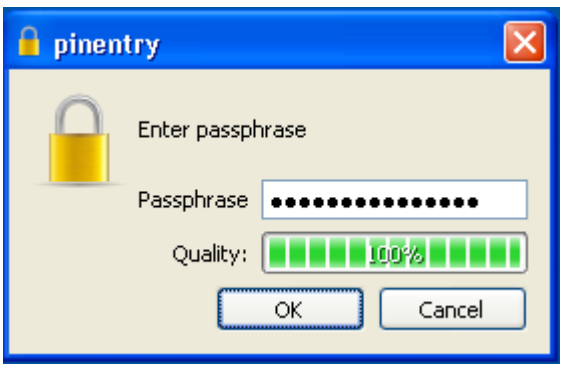


Figure 18: Enter password

7 – Wait for the key creation

During the key creation process it is necessary to generate random numbers, so it is important that you do activities that stimulate this process, such as typing on the keyboard, moving the mouse, opening and closing programs, etc.

Kleopatra offers a white space where you can type whatever you want on it to stimulate this process. It does not matter what you type because it will not be considered on the key creation and will not be part of it.

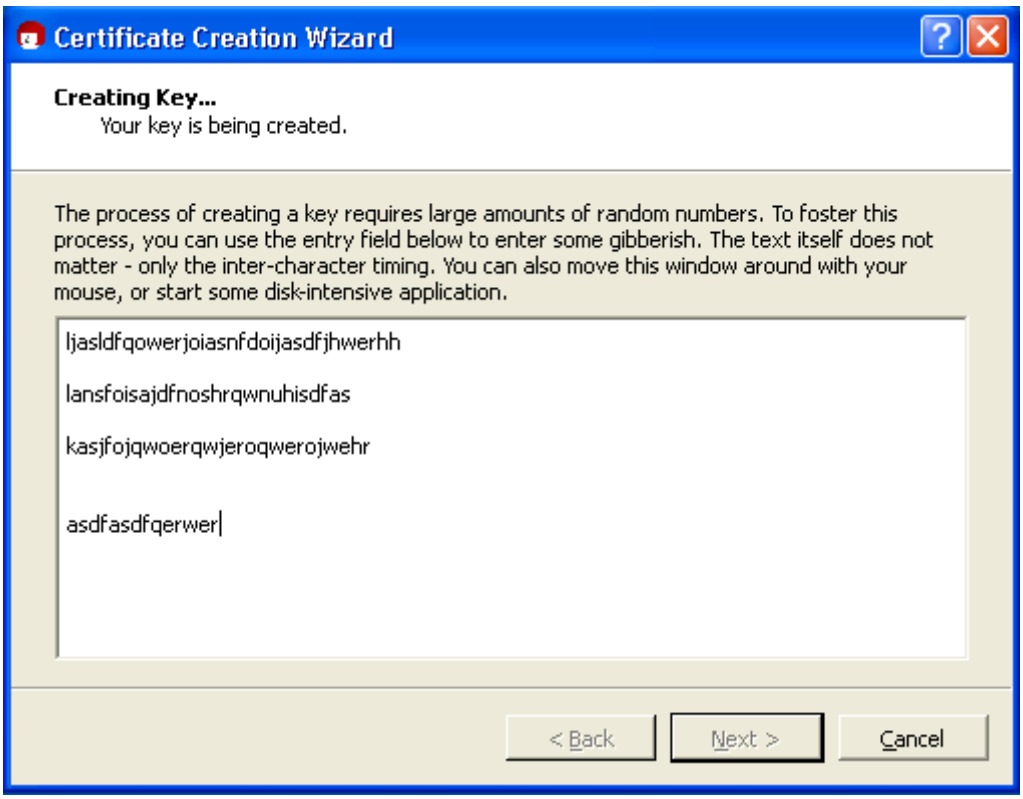


Figure 19: Creating your key. Use the field above to enter some random text.

8 - Confirmation

Congratulations, you have just created your first key pair! :)

A confirmation window will show up showing your key's fingerprint, which is a unique code that only **this key** in the world has. When you send your public key to other people, the only way they can be certain that the key they received is yours and was not twisted along the way is by confirming the key's fingerprint with you.

You can choose any of the three options suggested below, or just finish the process. We will finish the process, so click on Finish button.

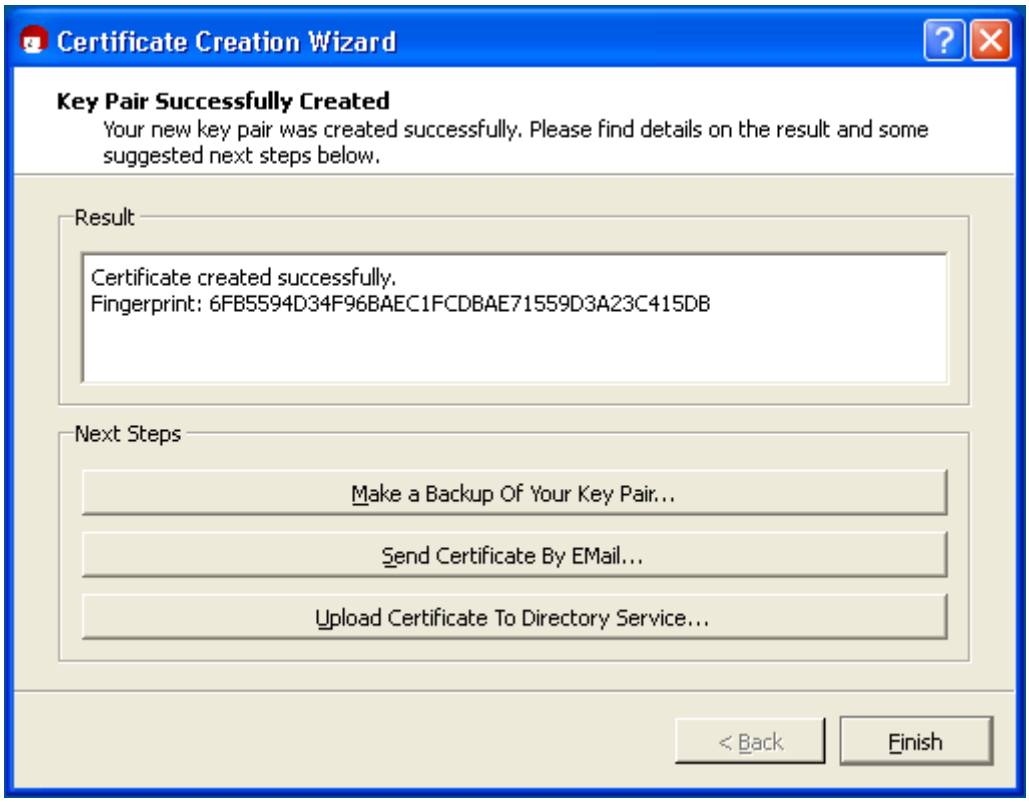


Figure 20: Confirmation window

9 – Verify your key

Now you will notice that your recently created key appears in Kleopatra.

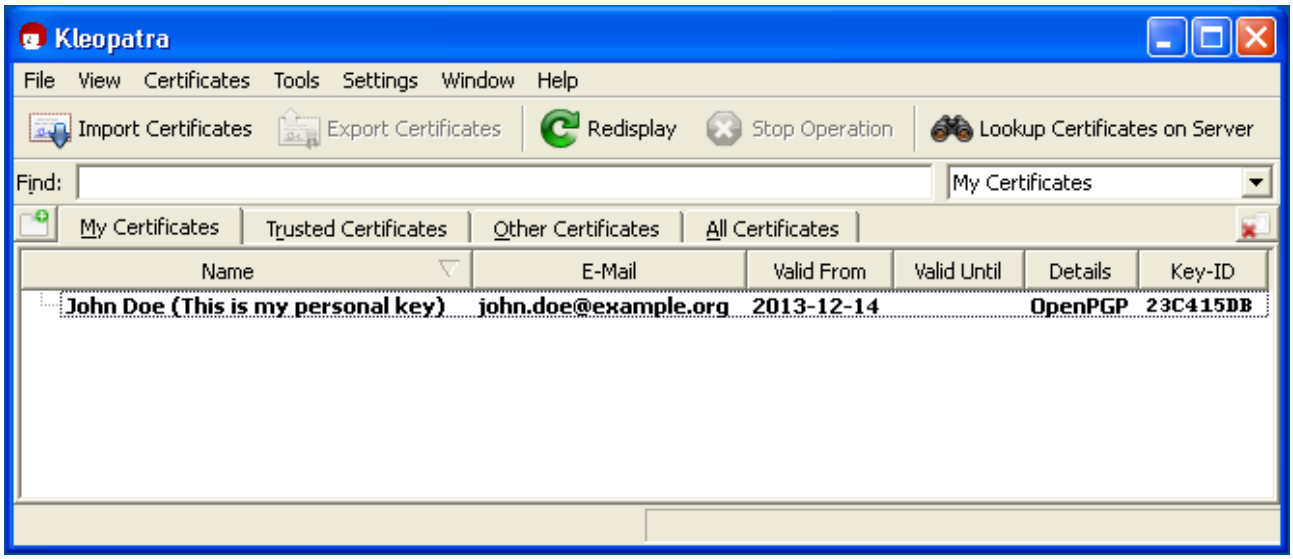


Figure 21: Created key

Every time you want to check details about your key, or make changes in it, just select your key and click with right button of the mouse on top of it and select Properties in the menu.

7.3 *NIX systems

1 - Open Seahorse

Open Seahorse and click on File → New Certificate, or press Ctrl N

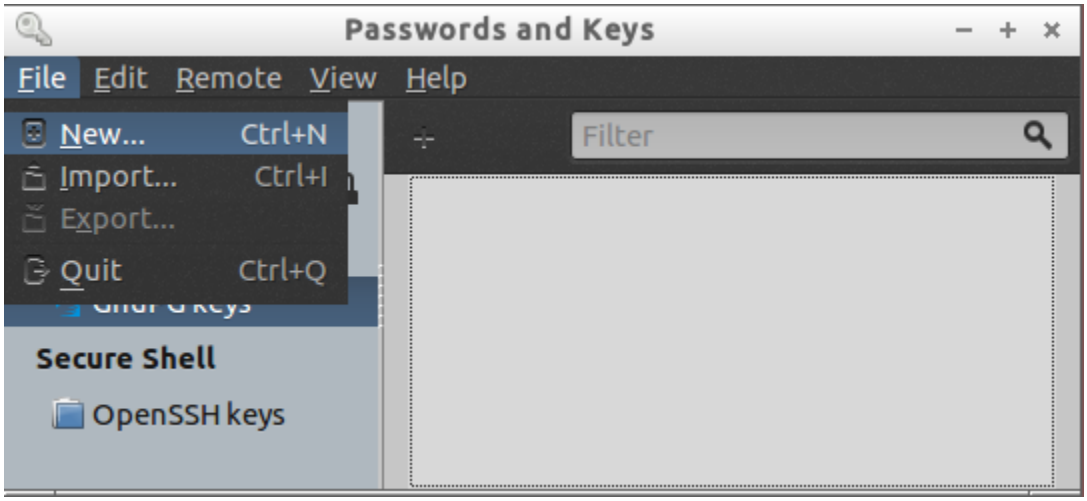


Figure 22: Open Seahorse

2 - Choose PGP Key option

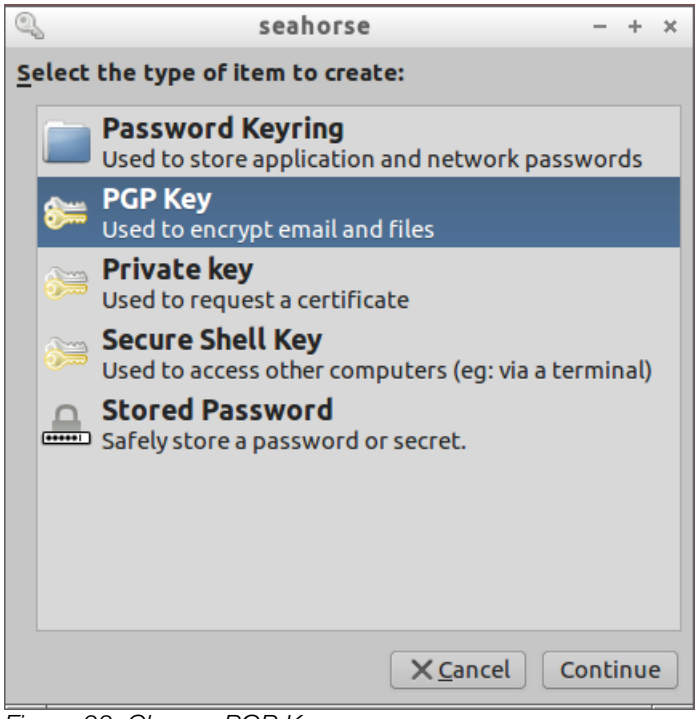


Figure 23: Choose PGP Key

3 - Enter basic details of your key

Here you will enter your basic personal details which will be part of your key and will be visible to anyone who has your key. The name and e-mail address are required, while the comment is optional. When you are done click on Advanced key options.

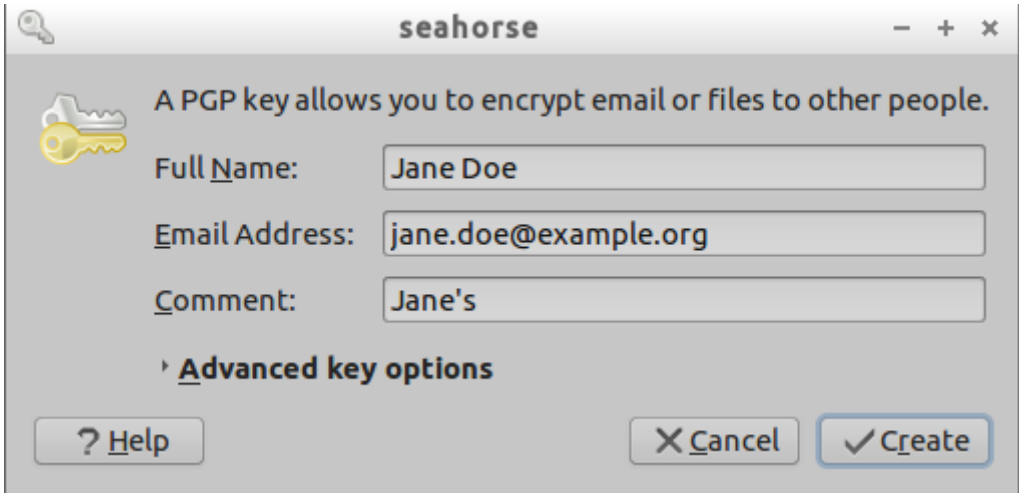


Figure 24: Enter basic details

4 - Set advanced details

Here you will set the advanced settings of your key.

Encryption Type: Select RSA.

Key Strength (bits): Set the key length to 4096.

Valid until: You can define any value you want. Uncheck it if you want no expiry.

When you are done click on Create button to create your key.

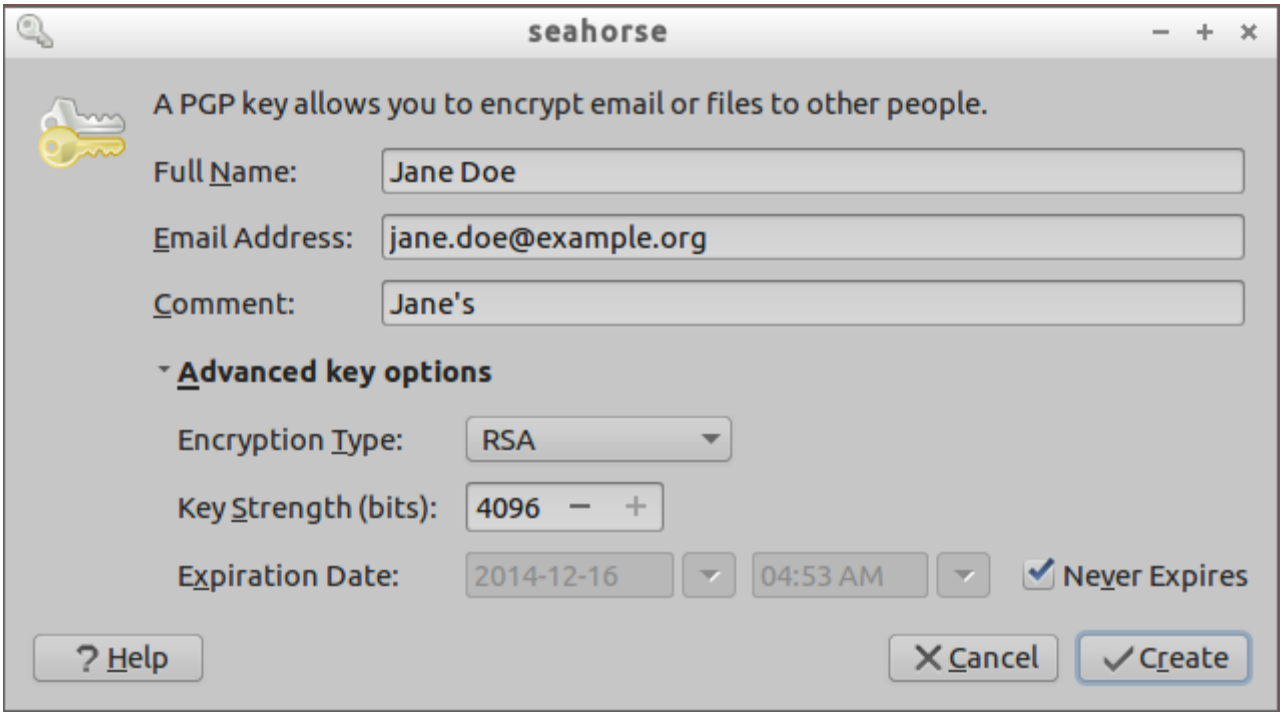


Figure 25: Set advanced options

5 - Choose a password

This is one of the most important steps of the whole process. The strength and security of your key are directly related to your password. There is no point in using the best encryption system in the world if you use a weak password, so choose a VERY STRONG password!

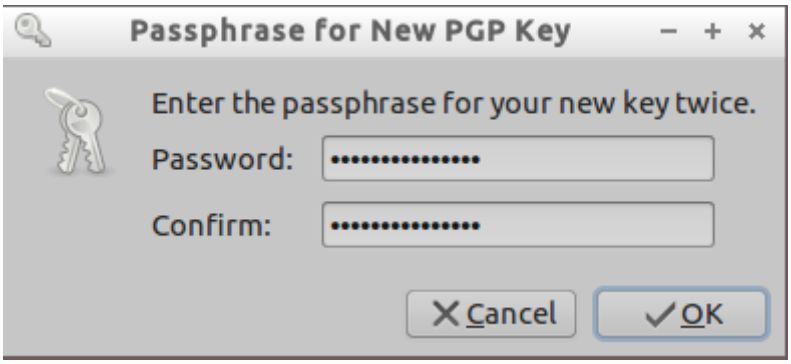


Figure 26: Enter password

7 - Wait for the key creation

During the key creation process it is necessary to generate random numbers, so it is important that you do activities that stimulate this process, such as typing on the keyboard, moving the mouse, opening and closing programs, etc.

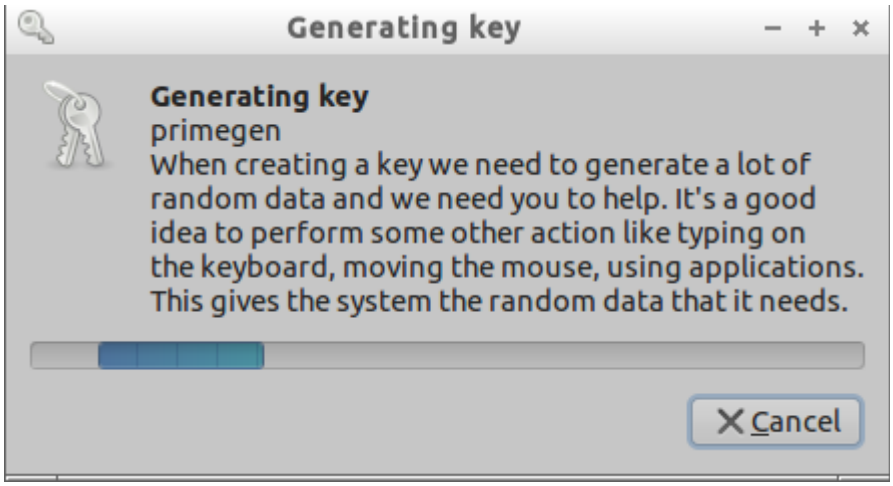


Figure 27: Key being generated

8 - Confirmation

Congratulations, you have just created your first key pair! :)

Now you will notice that your recently created key appears in Seahorse.

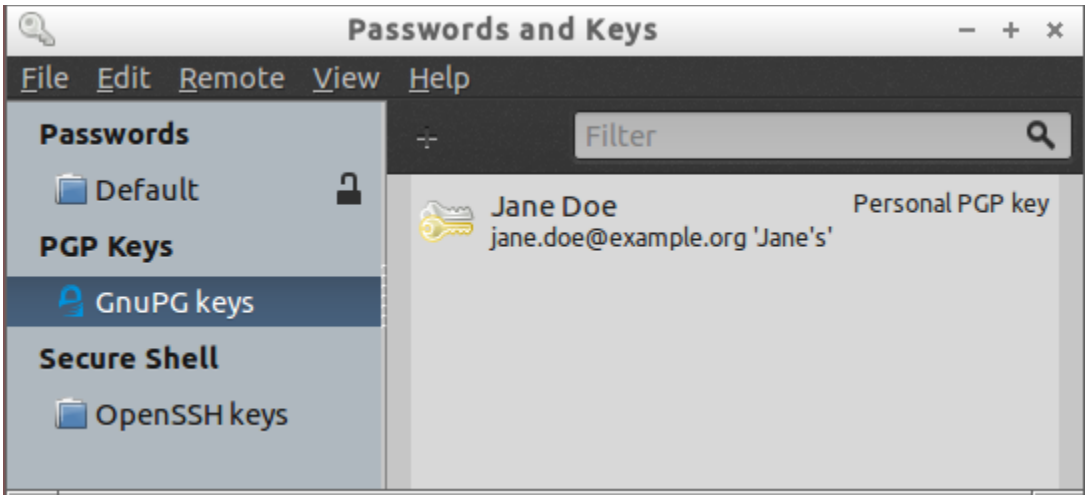


Figure 28: Your recently created key

Every time you want to check details about your key, or make changes in it, just select your key, right-click it and select Properties in the menu.

CHAPTER 8
Thunderbird and Enigmail

Mozilla Thunderbird is an e-mail client similar to Microsoft Outlook, and Enigmail is an add-on of Thunderbird that brings encryption to it. Both softwares are software libre and they are distributed completely free of cost, so you can easily obtain them online.

8.1. Installation

The first step to use both programs is to install them. In this section we cover Mozilla Thunderbird installation in Microsoft Windows and *NIX distributions, but it is also available to other systems such as Apple OS X.

8.1.1 – Windows Installation

1 – Download Thunderbird

Mozilla Thunderbird can be downloaded from the official Mozilla website: https://www.mozilla.org/Thunderbird

Figure 29: Thunderbird download

2 – Install Thunderbird

Thunderbird installation is a very straightforward process, as indicated below:

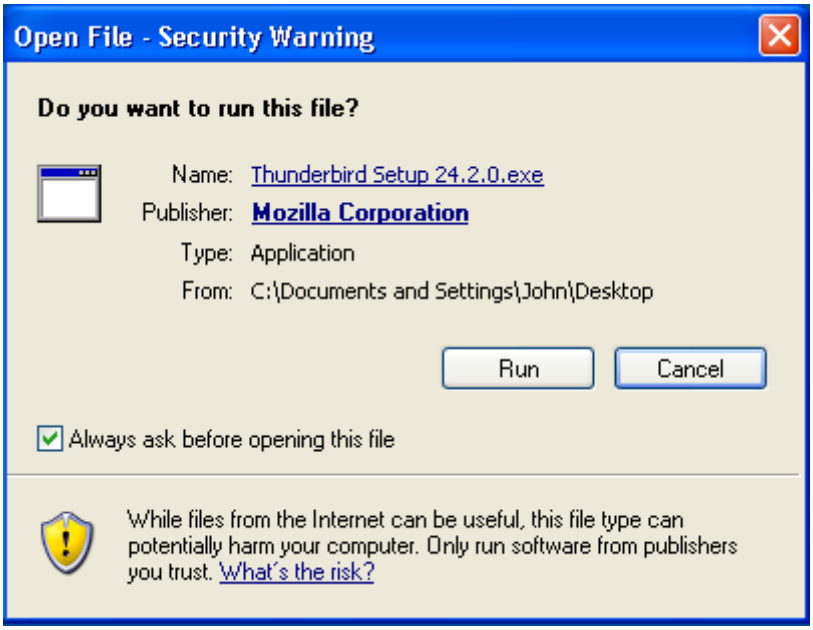


Figure 30: Security warning

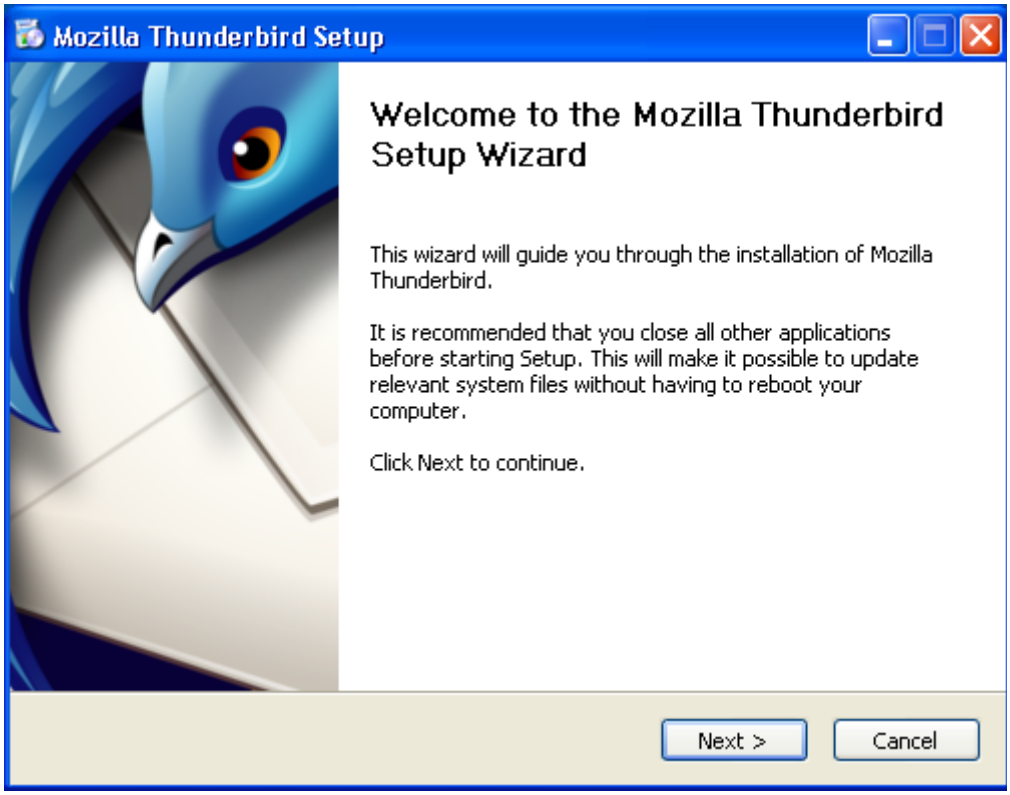


Figure 31: Welcome screen

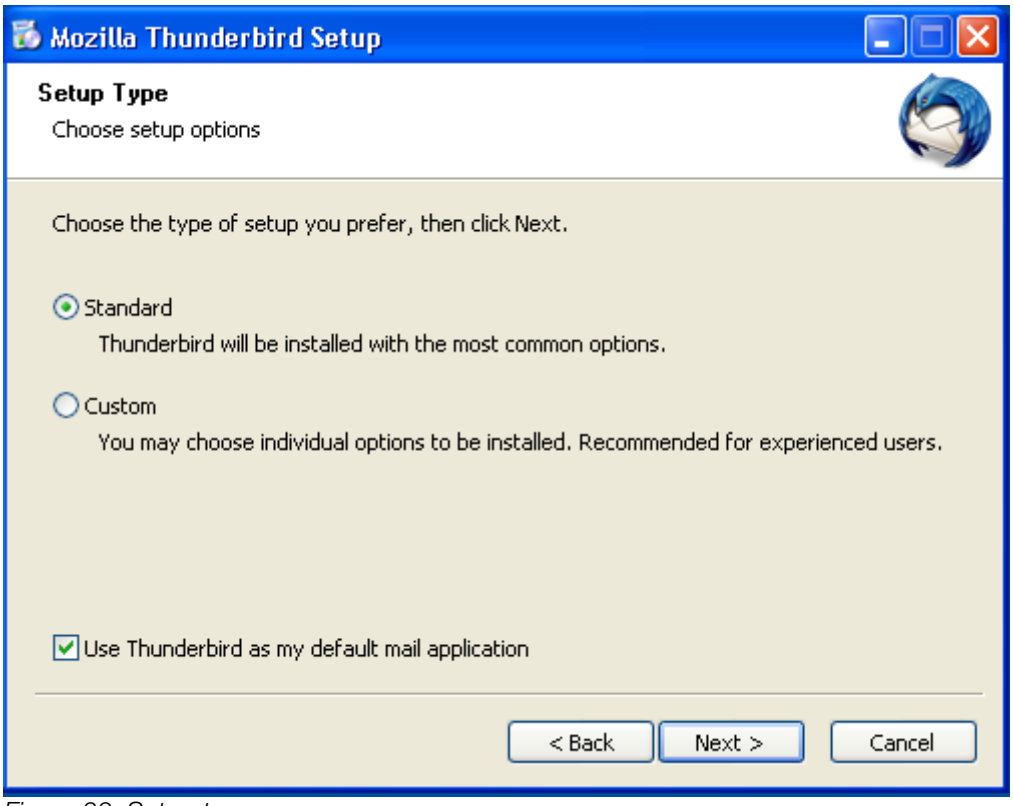


Figure 32: Setup type

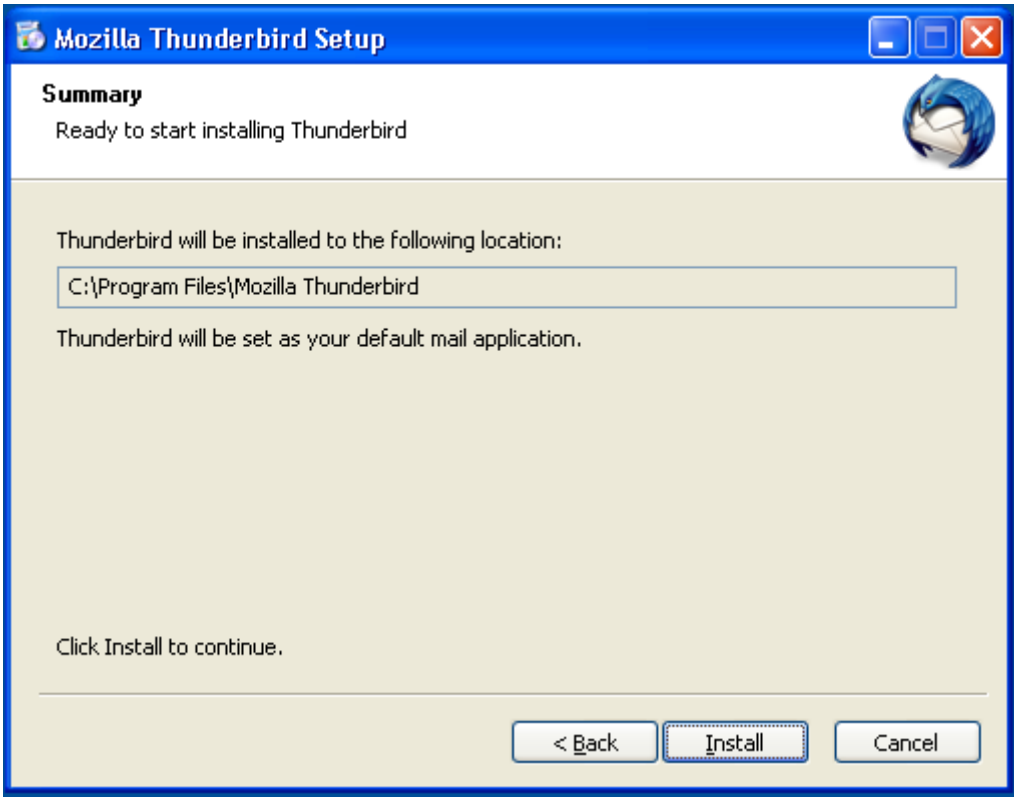


Figure 33: Summary

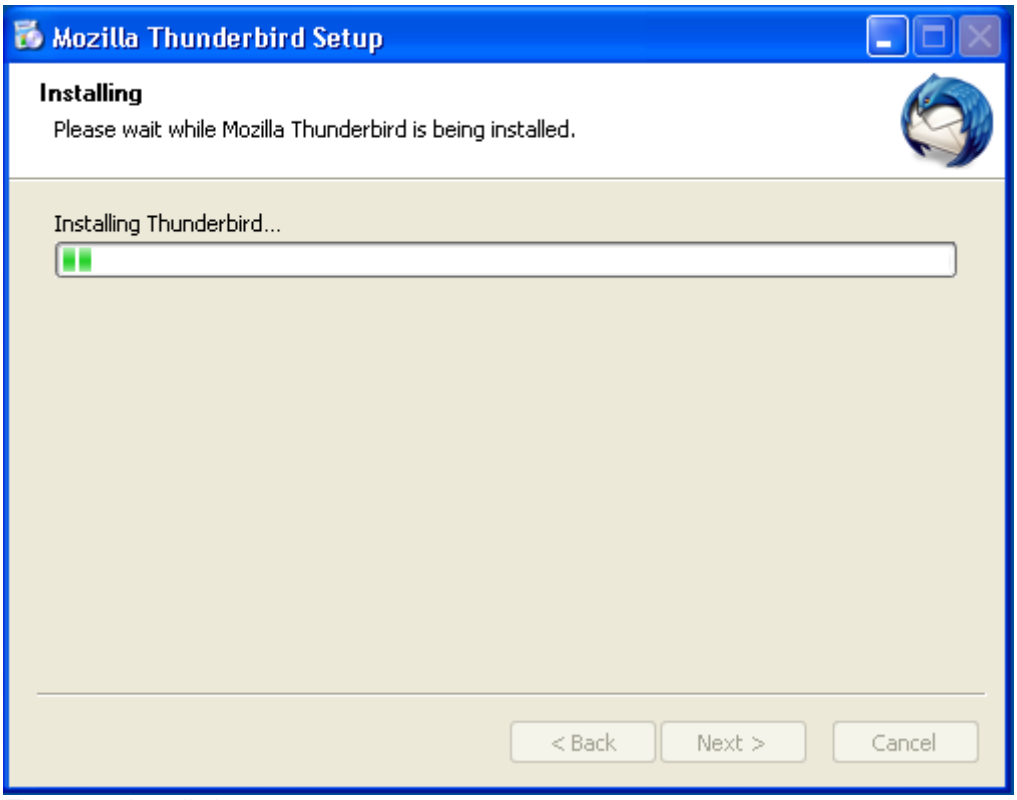


Figure 34: Installation progress

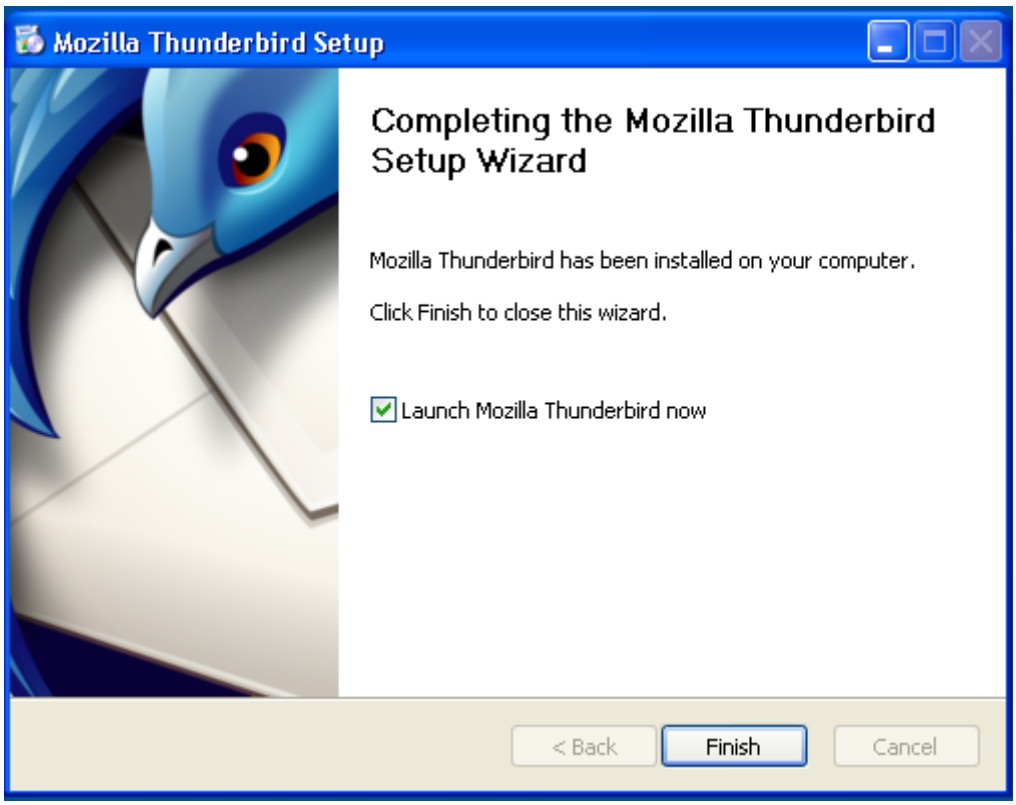


Figure 35: Installation complete

8.1.2 *NIX installation

Below are the commands for the most common *NIX distributions:

Arch Linux:

```
$ sudo pacman -S thunderbird
```

Debian, Mint, Ubuntu:

```
$ sudo apt-get install thunderbird
```

Fedora, CentOS:

```
$ sudo yum install thunderbird
```

Gentoo, Sabayon:

```
$ sudo emerge thunderbird
```

Mageia:

```
$ sudo urpmi thunderbird
```

FreeBSD, OpenBSD:

```
$ sudo pkg_add -r -v thunderbird
```

8.2. Configuring your e-mail account

Open Thunderbird and in the Welcome screen click on button Skip and use my existing email.

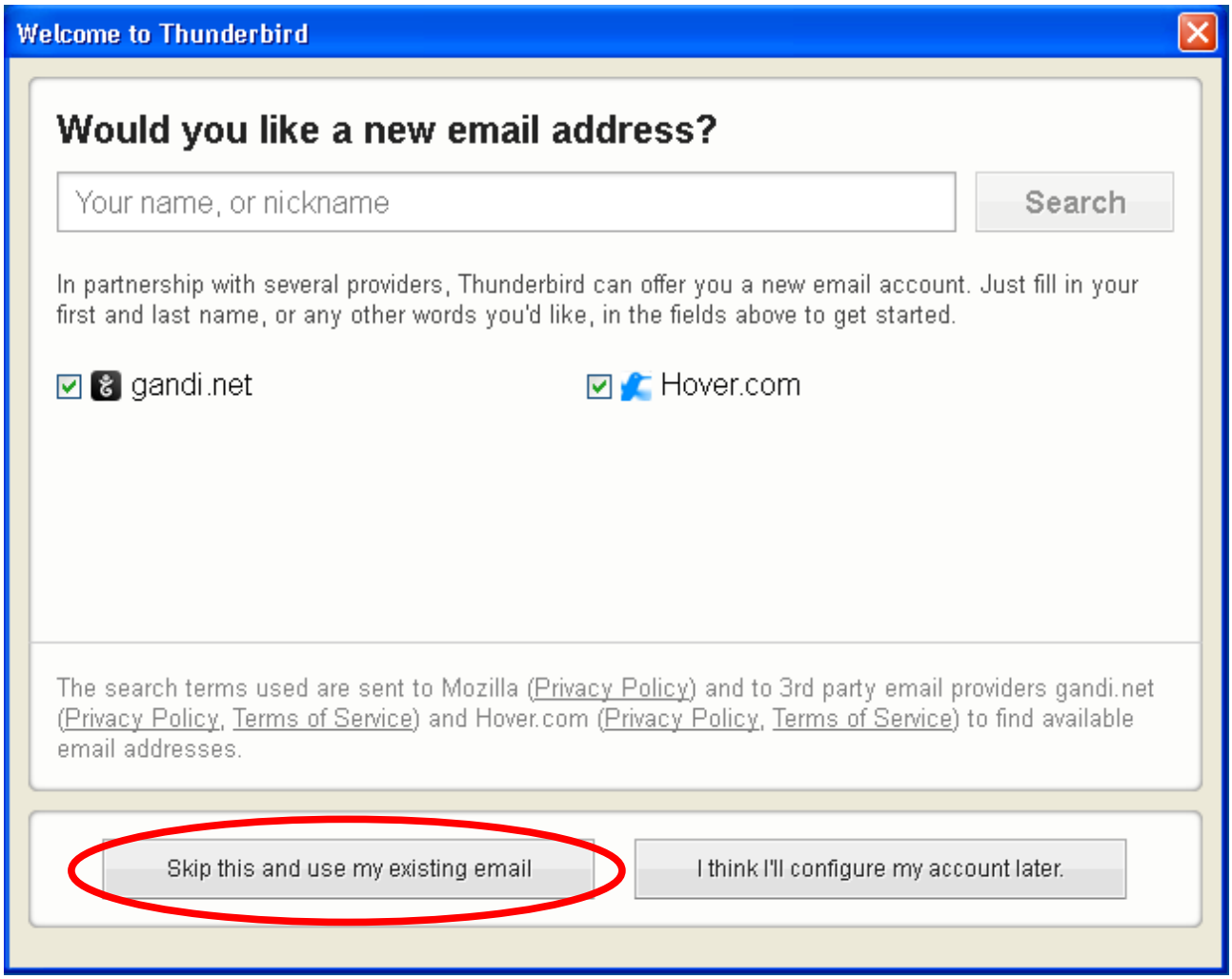


Figure 36: Welcome screen

Enter your name, e-mail address and e-mail password in the fields below. You are configuring an account to be used with the key pair you created in chapter 6, so use the same e-mail address you have created your key for. Check 'Remember password' if you want that Thunderbird automatically remembers your password every time you open it.

When you are done click on Continue button.

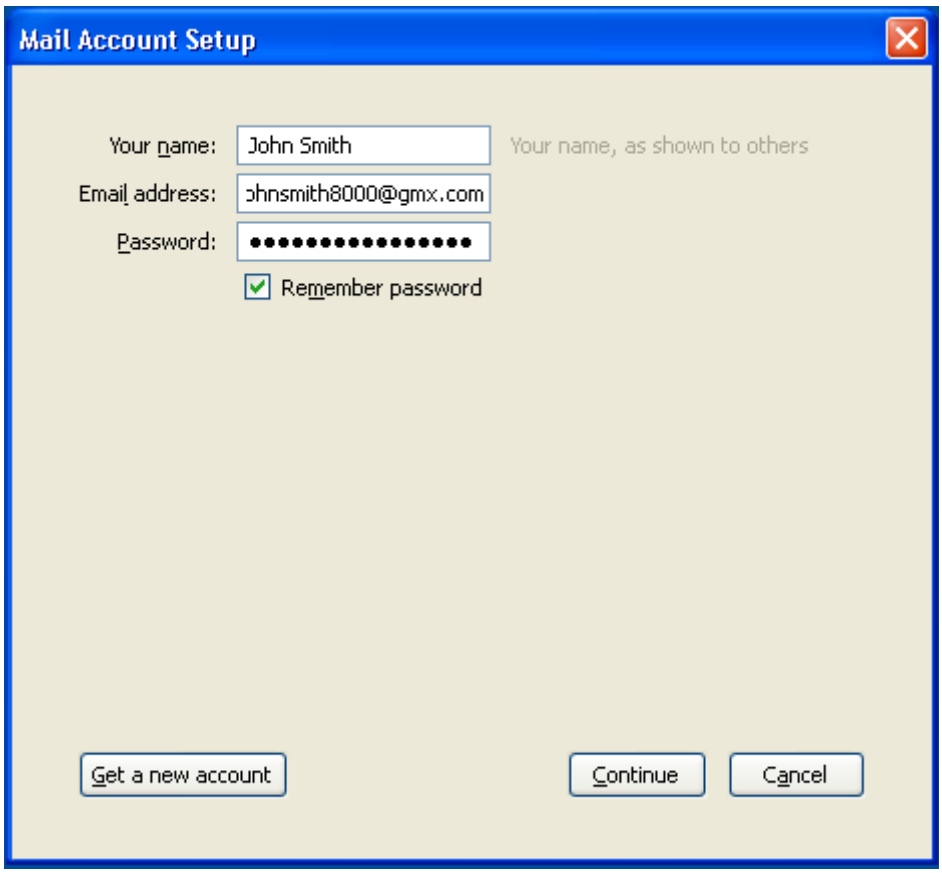


Figure 37: Enter e-mail information

Thunderbird automatically tries to guess the correct configuration for your e-mail account. It usually gives you two choices: IMAP and POP3. We recommend that you use IMAP.

If for any reason Thunderbird cannot set up your account correctly, or you would rather use a different configuration, then click on the button Manual config to manually insert the custom configuration. In this case you can check the help section of your e-mail provider, they usually offer instructions on how to use accounts with other e-mail clients like Thunderbird.

When you are finished click on Done button.

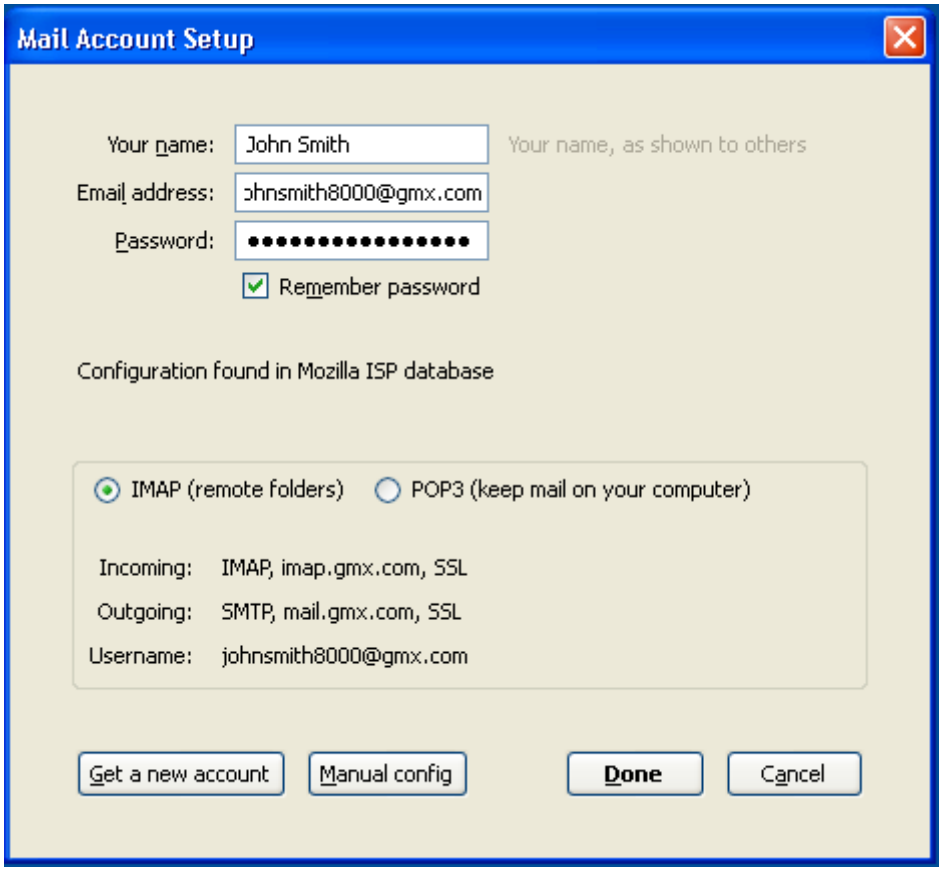


Figure 38: Account configuration

That's it, your account is now created. Now you will be taken to Thunderbird's main screen where your messages will be synchronized with the ones you have in your webmail. If you are using IMAP your messages will remain stored in your e-mail provider, so you can still access them from other computers or from the web browser.

If you wish to have the old menu bar just right-click on menu area in Thunderbird and choose Menu bar, as shown in the images below.

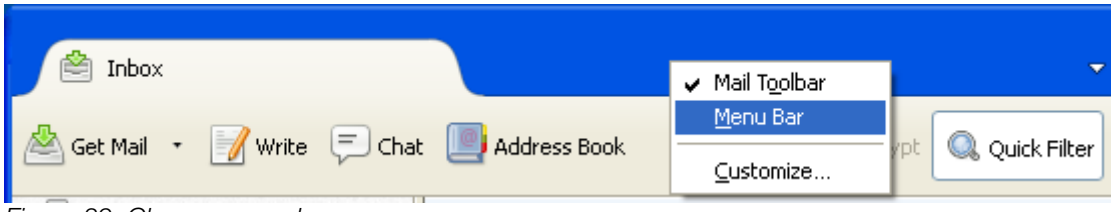


Figure 39: Choose menu bar

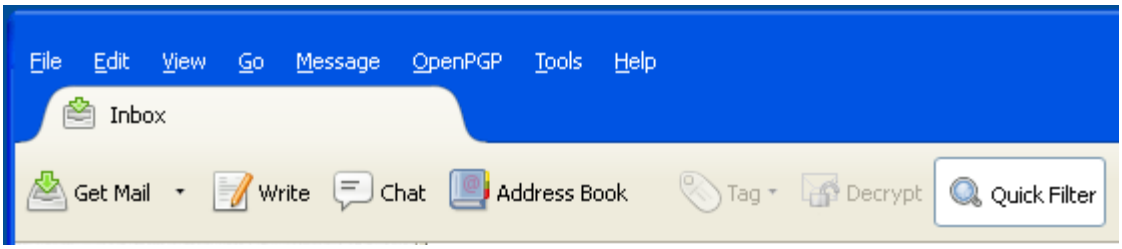


Figure 40: Thunderbird with menu bar

That's it, now you have the old menu bar.

8.3 Configuring Enigmail

1 – Open Add-ons

Click on menu Tools → Add-ons. In *NIX systems this is changed to Edit → Preferences.

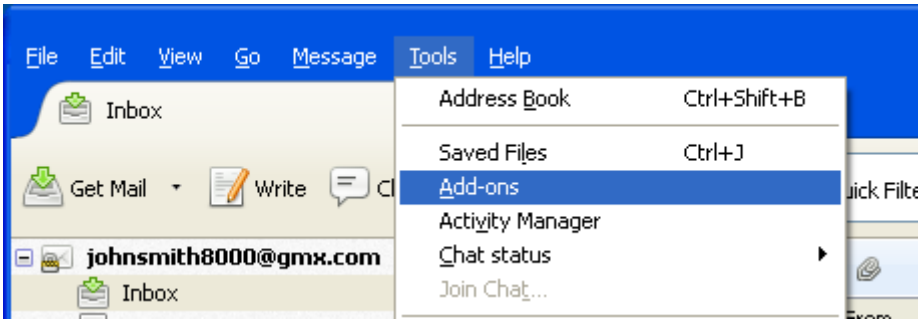


Figure 41: Open Add-ons

2 – Search for Enigmail

In the search field on the upper right corner enter Enigmail and press ↵.

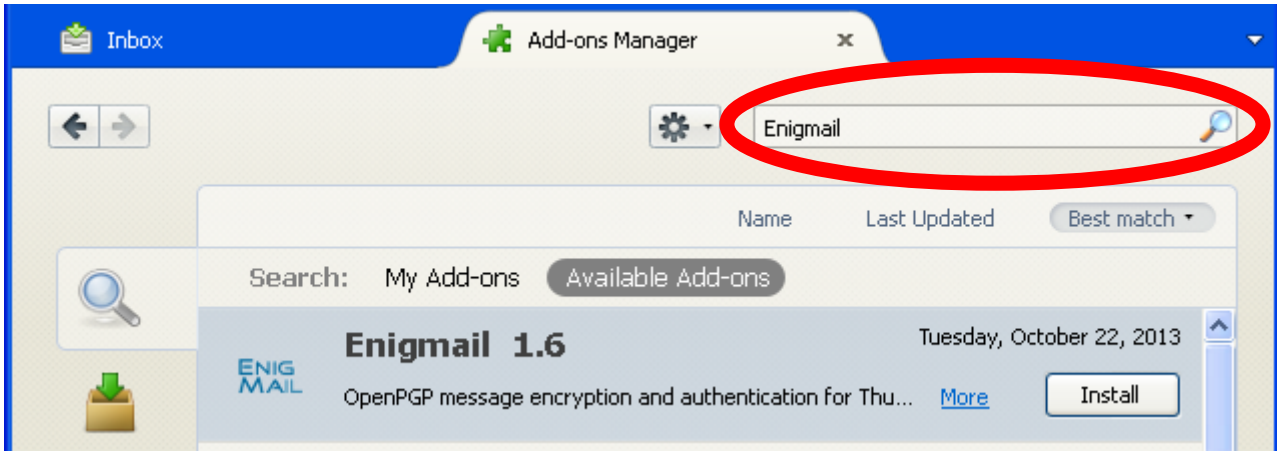


Figure 42: Search for Enigmail

3. Install Enigmail

Click on Install button and wait until the installation finishes.

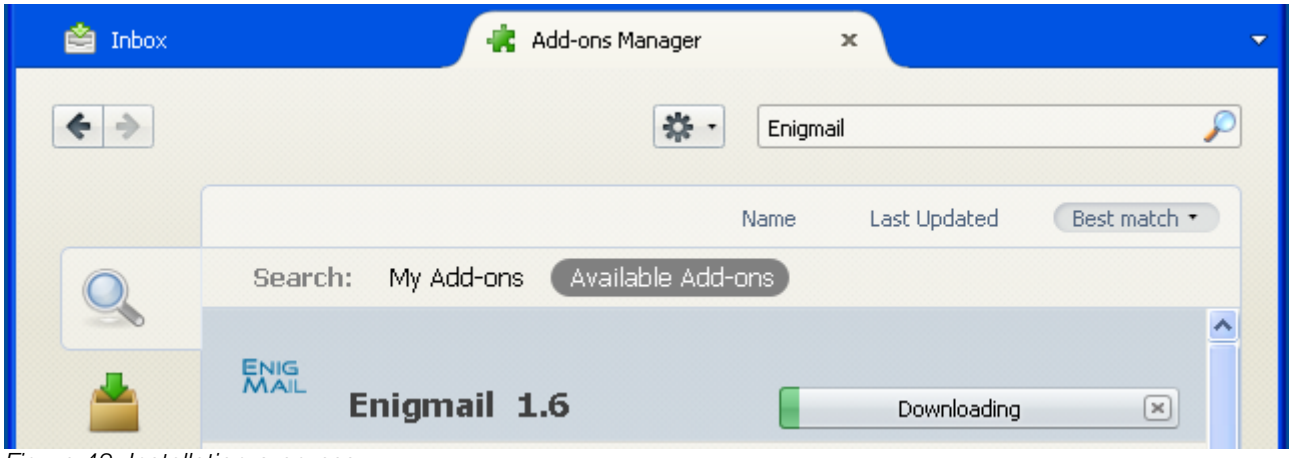


Figure 43: Installation progress

4. Restart Thunderbird

Click on Restart now button, or close and open Thunderbird again.

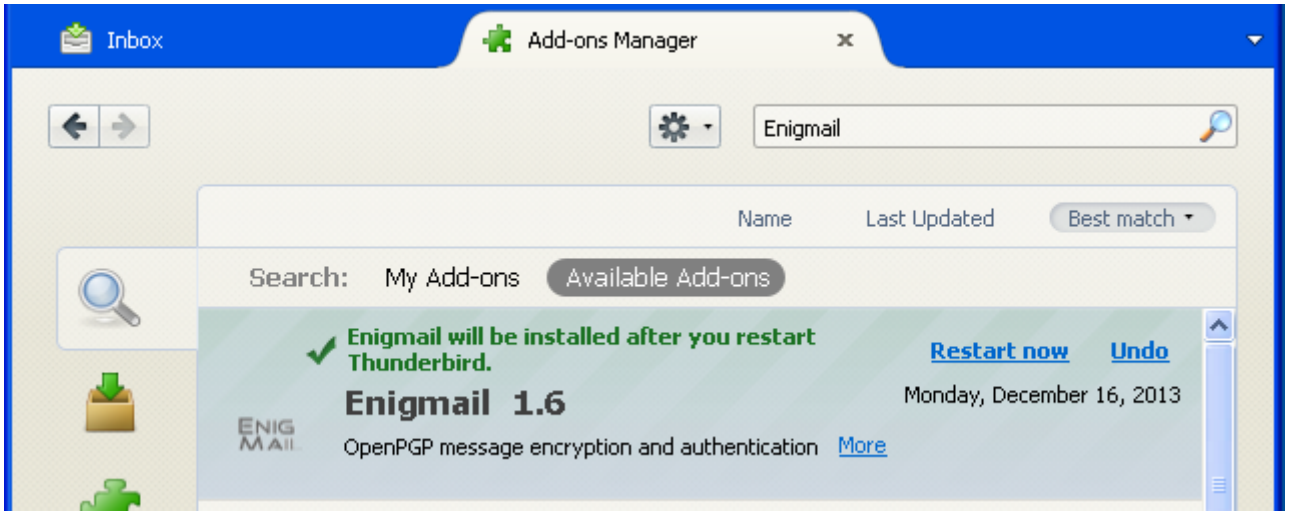


Figure 44: Restart Thunderbird

5. Start the Wizard

Click on OpenPGP menu and choose Setup Wizard. When the Wizard pops up select the first option 'Yes, I would like the Wizard to get me started' and click on Next.

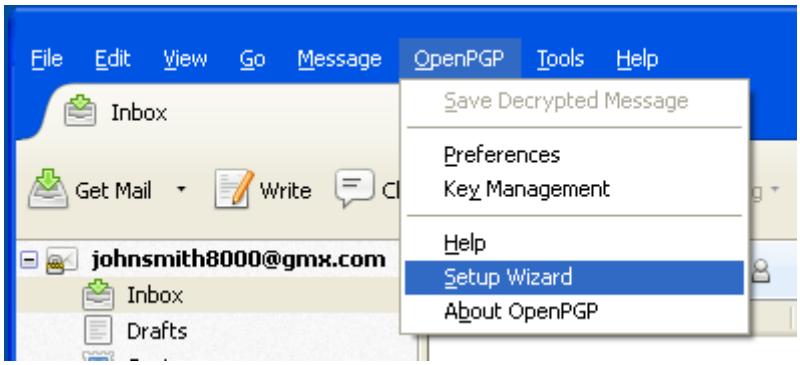


Figure 45: Start Setup Wizard

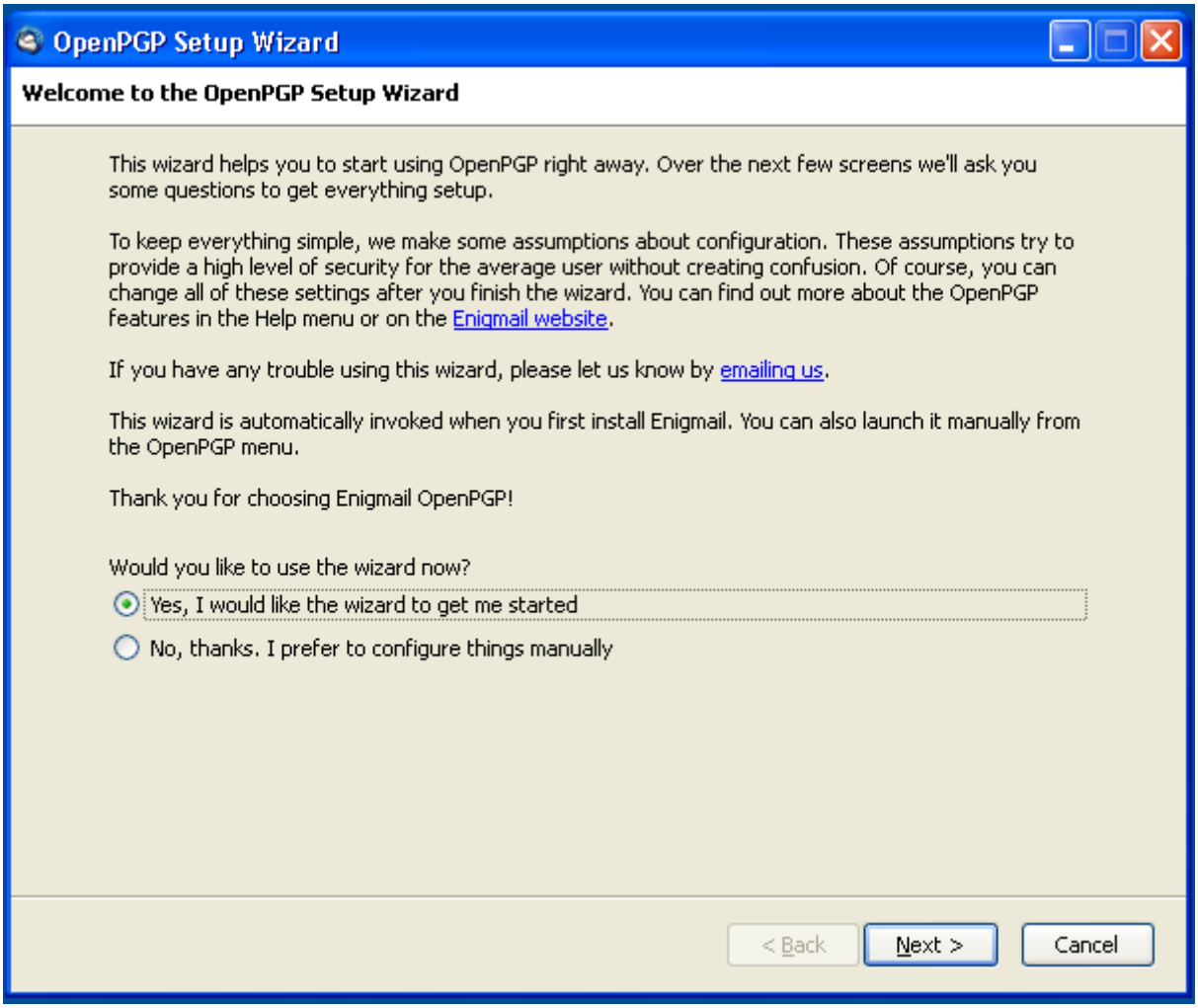


Figure 46: Welcome screen

6. Choose signing behavior

It is a good practice to sign all outgoing e-mails, so we will choose this option.

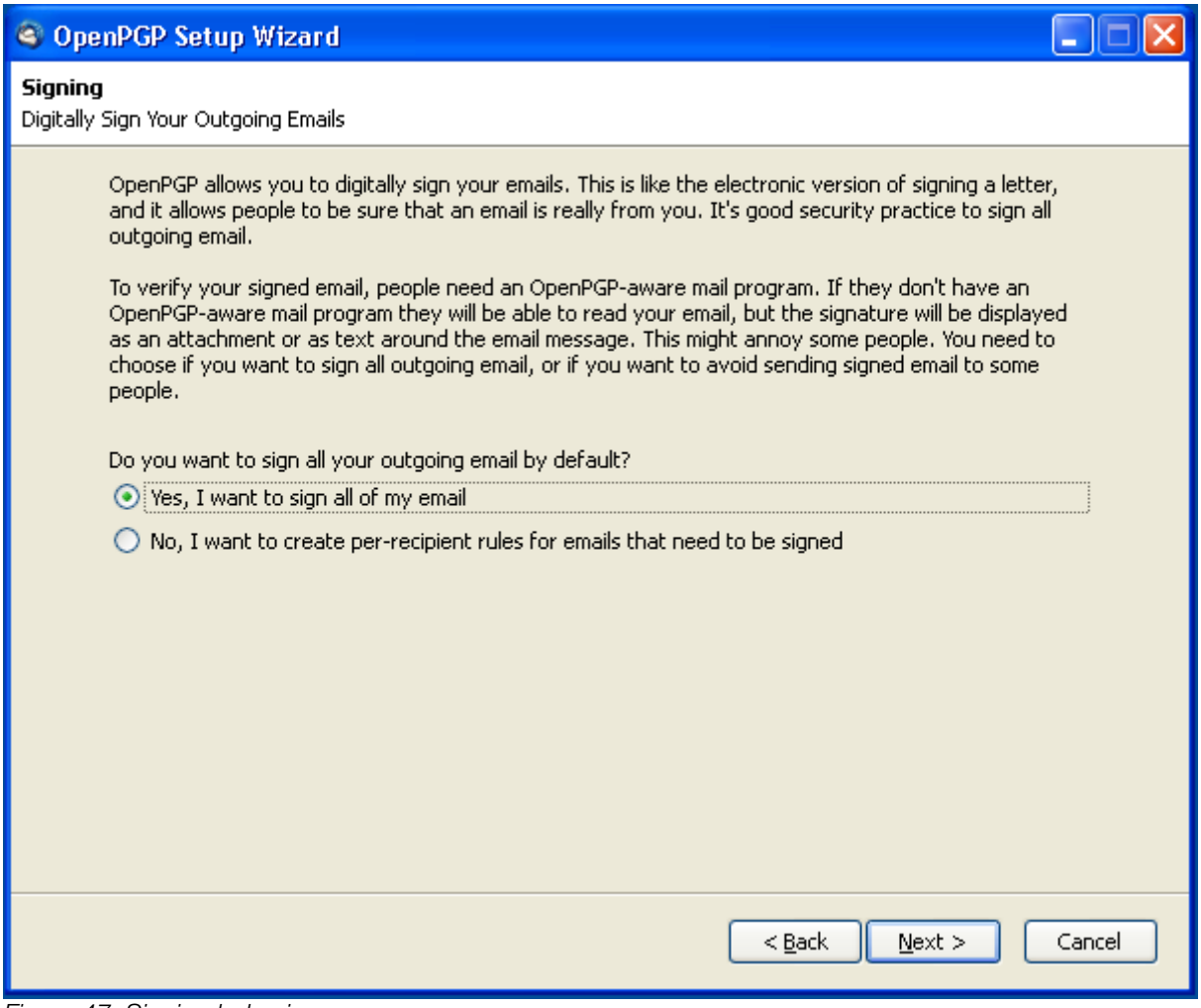


Figure 47: Signing behavior

7. Choose encryption behavior

You can choose between encrypting all outgoing e-mails by default, or creating custom rules for each one of your contacts. You should only choose the first option if you have the public key of all or almost all your contacts, otherwise choose the second option.

Here we will choose the second option because we don't have the public keys of our contacts, and we want to create custom rules for each one of them.

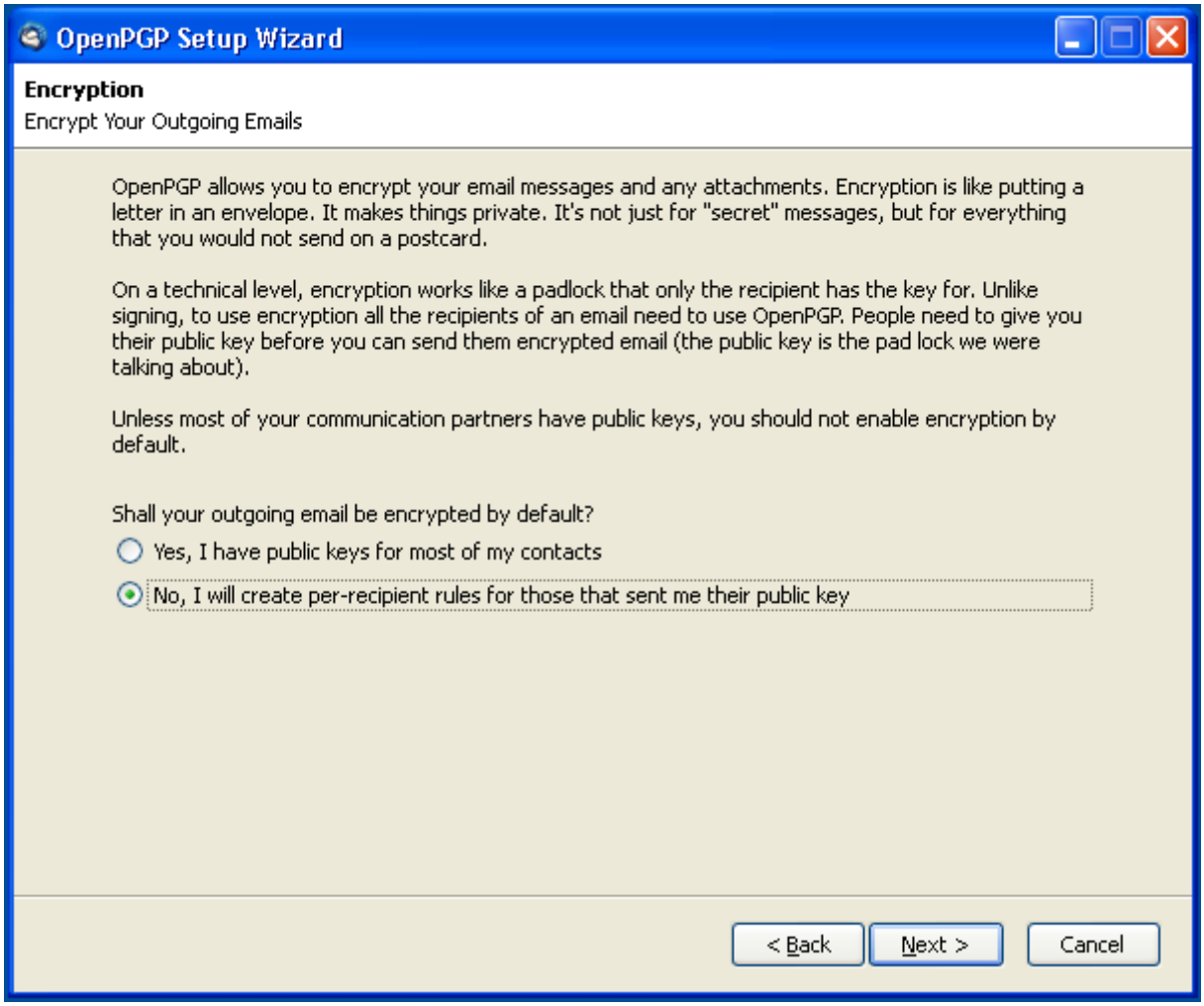


Figure 48: Encryption behavior

8. Preferences

Here you can change advanced settings of encrypting and signing behavior. We will use the default configuration, so leave it the way it is and select the second option.

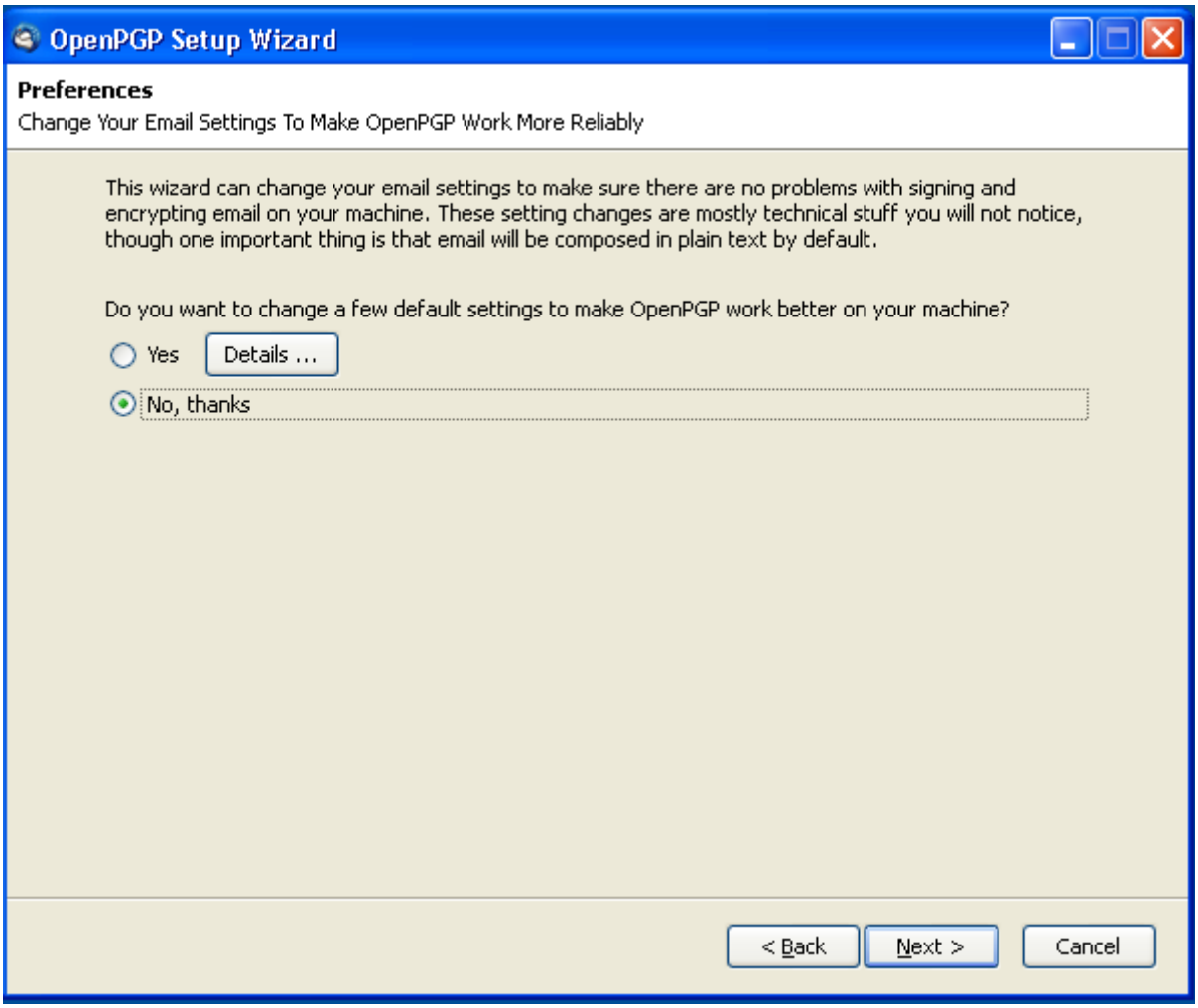


Figure 49: Advanced settings

9. Choosing your private key

Now you will choose the key that you will use with your e-mails. If you have created your key in chapter 6 it should appear here now, so you can choose it. If you have multiple keys, select the one you are configuring your account to.

It is possible to create a key pair through Enigmail instead of using Kleopatra, Seahorse or the command line, but we prefer the other methods because Enigmail may sometimes present bugs in this process.

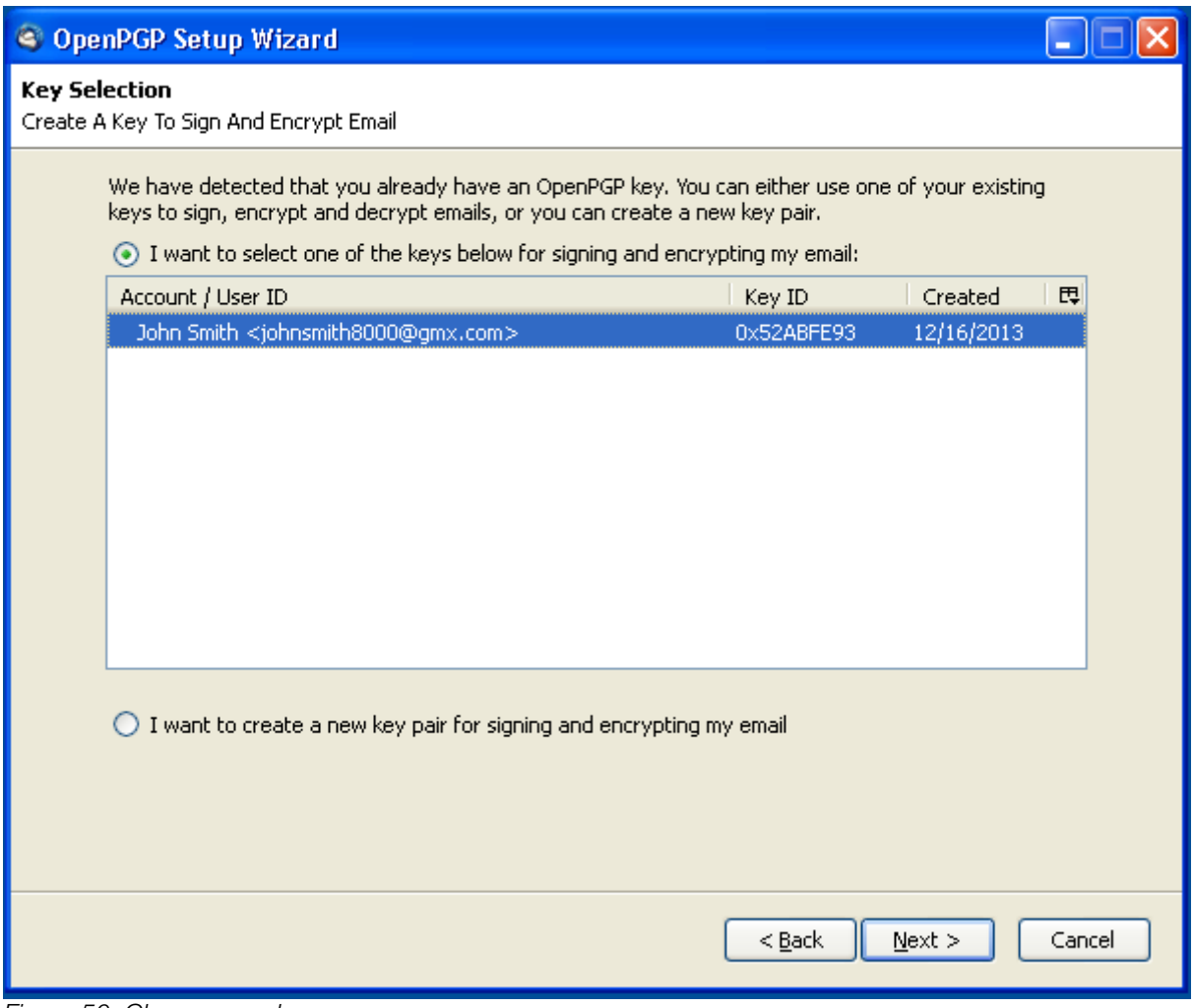


Figure 50: Choose your key

11. Conclusion

Here will be presented a summary. Click Next button.

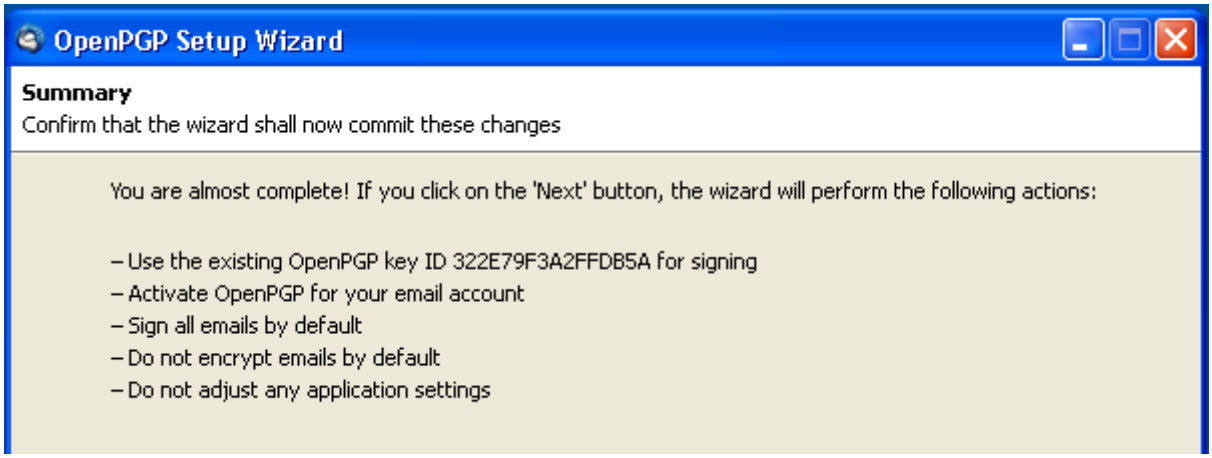


Figure 51: Summary

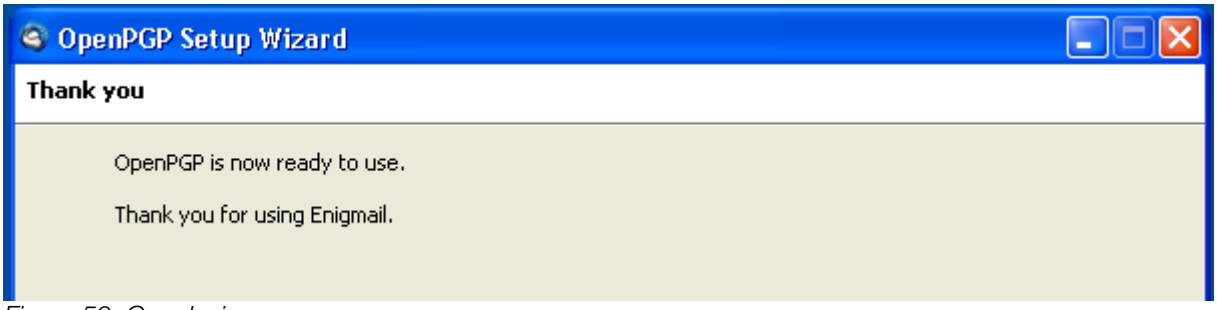


Figure 52: Conclusion

That's it, Enigmail is now installed and ready to be used with encrypted e-mails.

8.4. Testing messages

1 – Write a message

Now let's do a test, you will write a message to one of your contacts and send your public key to him, and request his public key. Your message will not be encrypted because you do not have his public key. Your message will be signed, but he will probably not notice it because he may not use GnuPG, neither Thunderbird.

In Thunderbird go to menu File → New → Message, or press Ctrl N to write a new message.

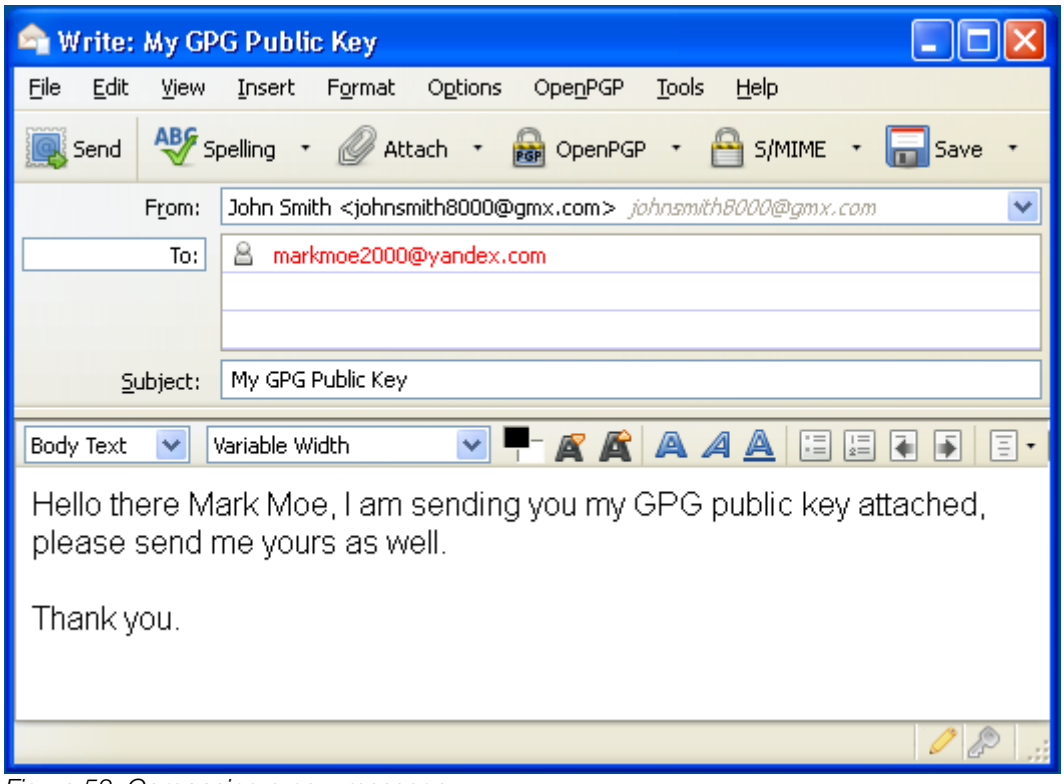


Figure 53: Composing a new message

2 – Attach your public key

To attach your public key just go to OpenPGP menu and select Attach My Public Key. Initially you will notice nothing different on screen, but when you click on Send you will see your key showing as an attachment on the right side of the screen.

To attach another public key, or more than one public key, click on OpenPGP menu and select Attach Public Key. This way the attachment board will appear on the right side.

NOTE: This second option will only be available if Enigmail is set to display advanced configurations, otherwise you will not be able to access it (as in the image below). To do it, in Thunderbird's main window click on OpenPGP menu and select Preferences, and click on button Show Advanced Preferences.

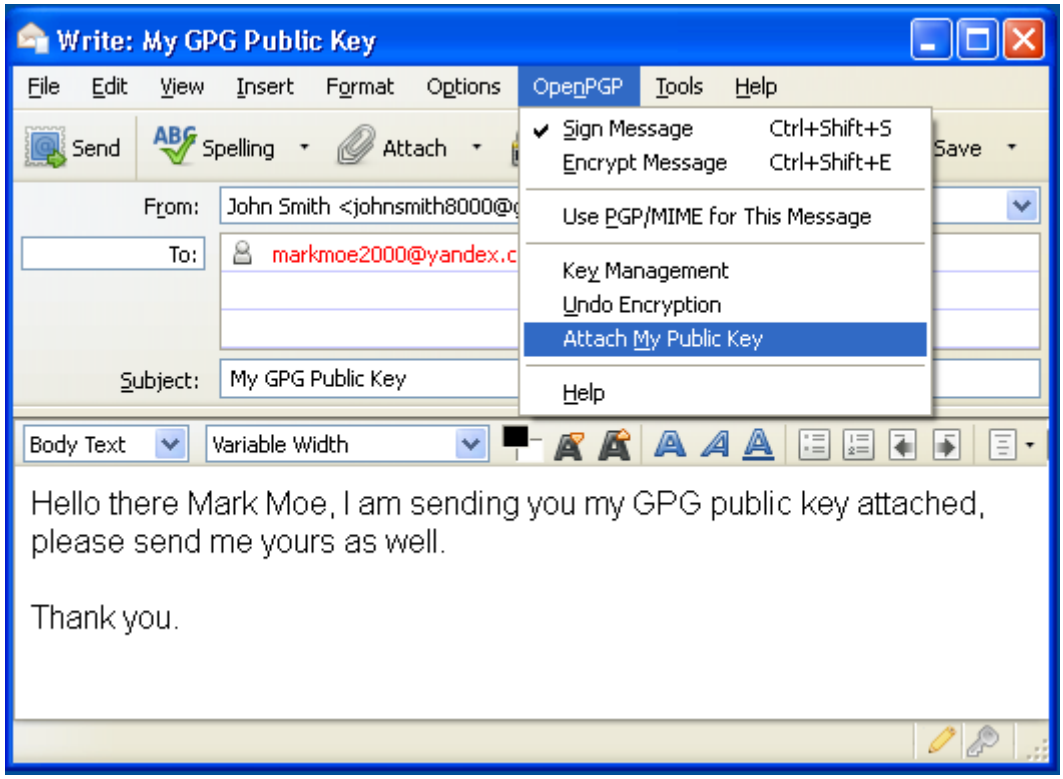


Figure 54: Attaching your public key

3 - Send the message

Just click on the Send button to send the message.

If the OpenPGP Prompt pops up as in the image below, select last option to use PGP/MIME and check the box below to use this method from now on.

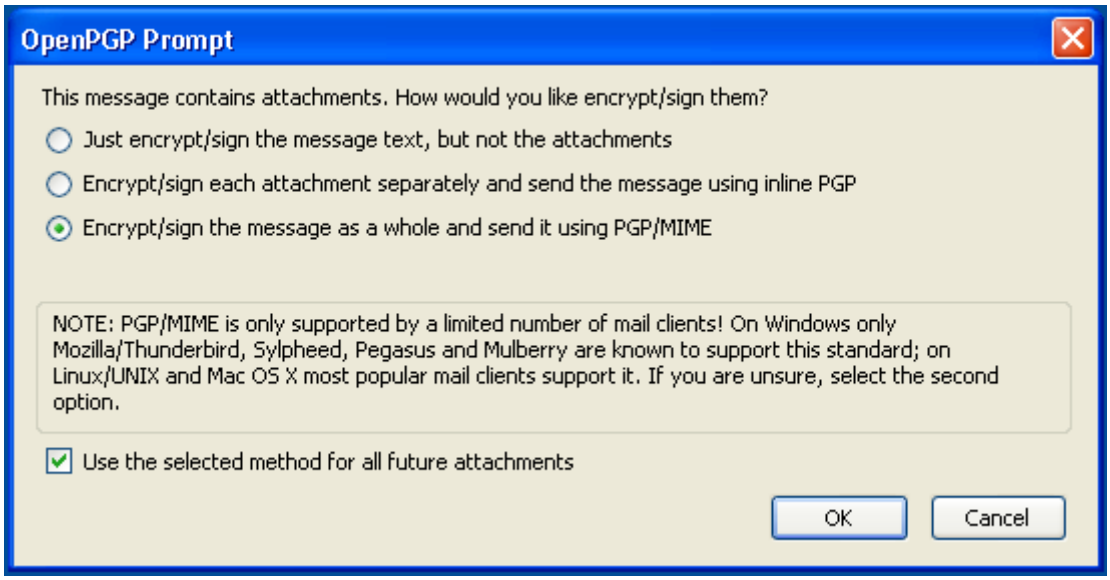


Figure 55: Choose attachment encryption/signing method

Enter the password of your key if requested.

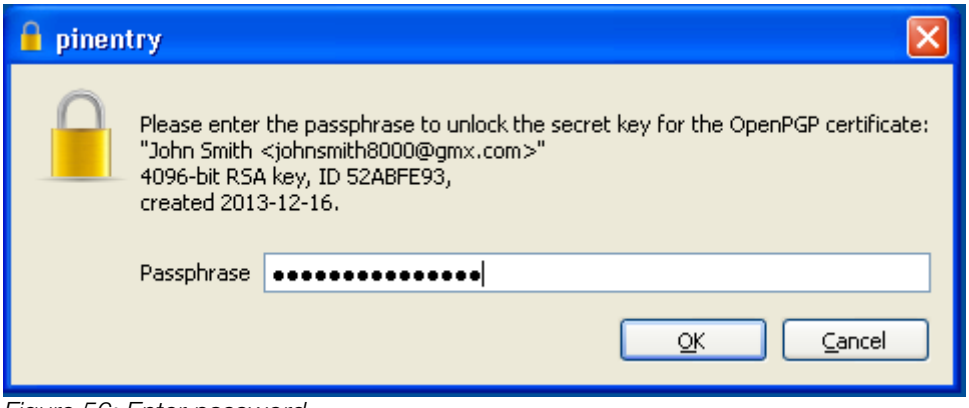


Figure 56: Enter password

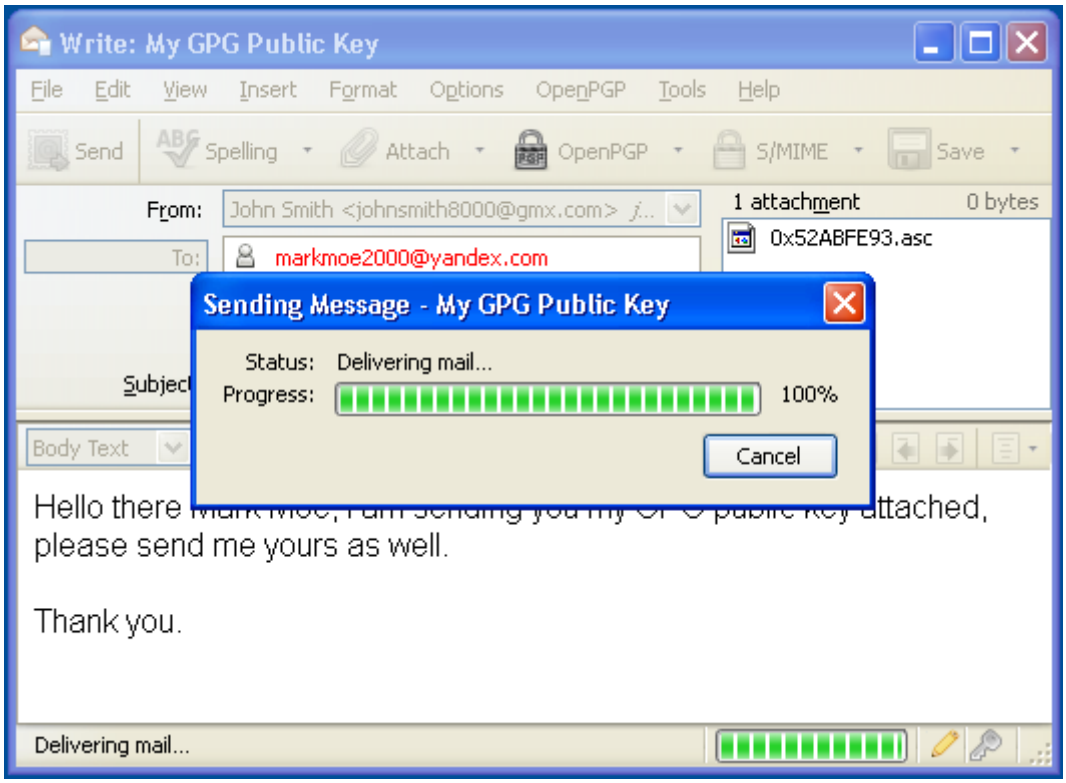


Figure 57: Sending message

That's it, your message has been sent. Now your contact must send his public key as well, which will be done in next step.

8.5 Importing public key

1 – Verify the answer

After you have sent your public key to your contact in the previous step, let's consider that the person decided to use GnuPG too, so he created a key pair for himself, replied your message and attached his public key as well. It would look similar to the image below:

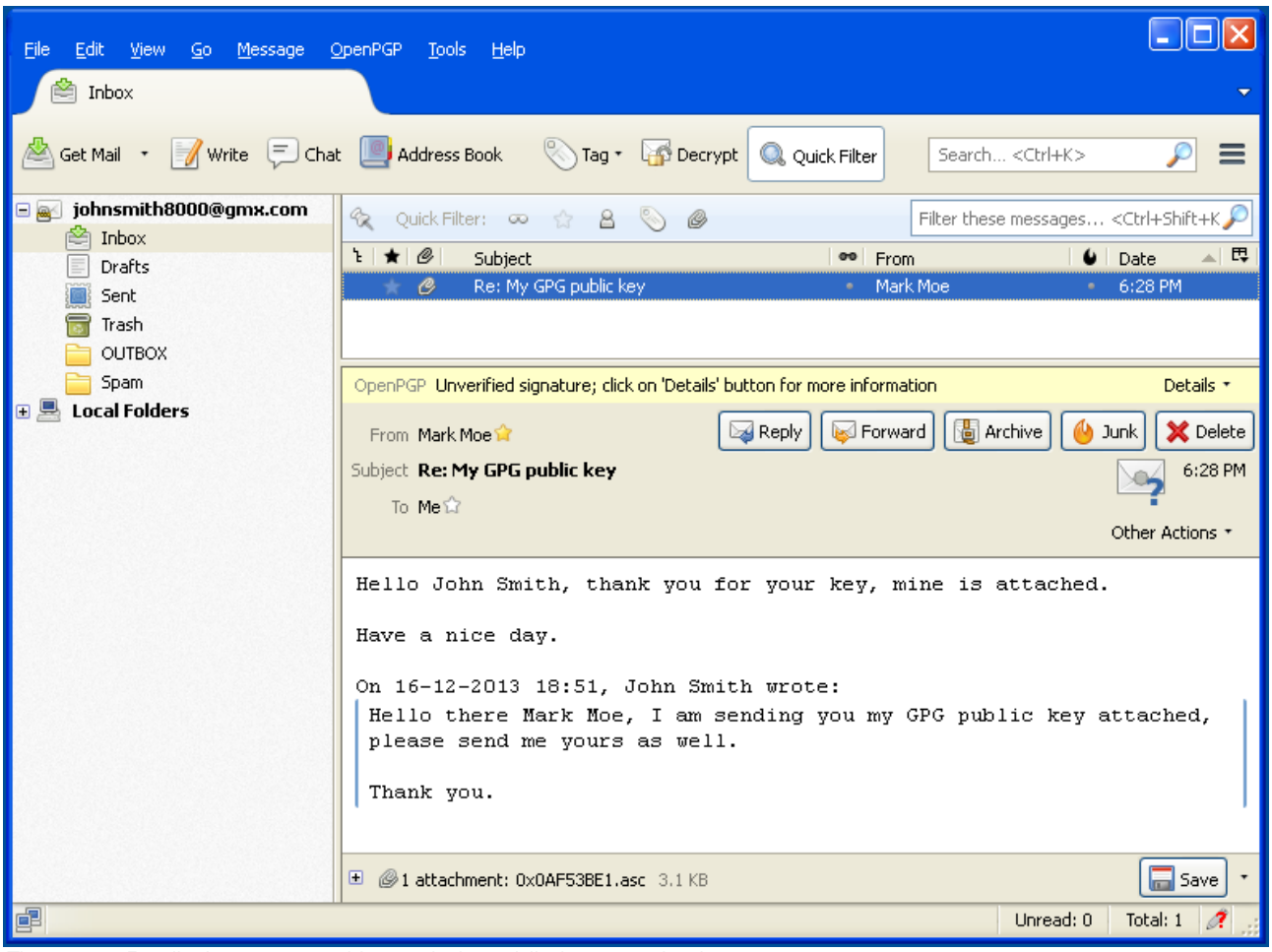


Figure 58: Replied message with sender's public key attached

Your contact also signed the message with his private key, but since you have not imported his public key yet, you see the yellow bar prompting “Unverified signature”.

2 - Import the public key

After receiving the public key attached on the message you have to import it. Right-click on the file's name and select Import OpenPGP Key.

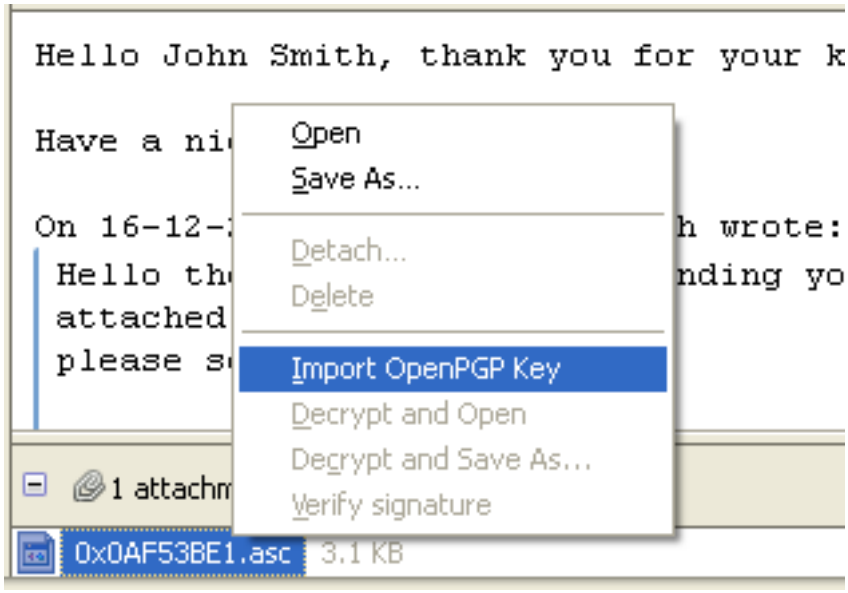


Figure 59: Importing public key

A confirmation message will be prompted showing a summary of the imported key. Just click OK to proceed.

Figure 60: Confirmation message

That's it, your key is now imported.

Now you will notice that the yellow bar turns blue and it says the signature is good and untrusted (if nothing happened and the bar is still yellow, try clicking on another folder or message, and then selecting his message again).

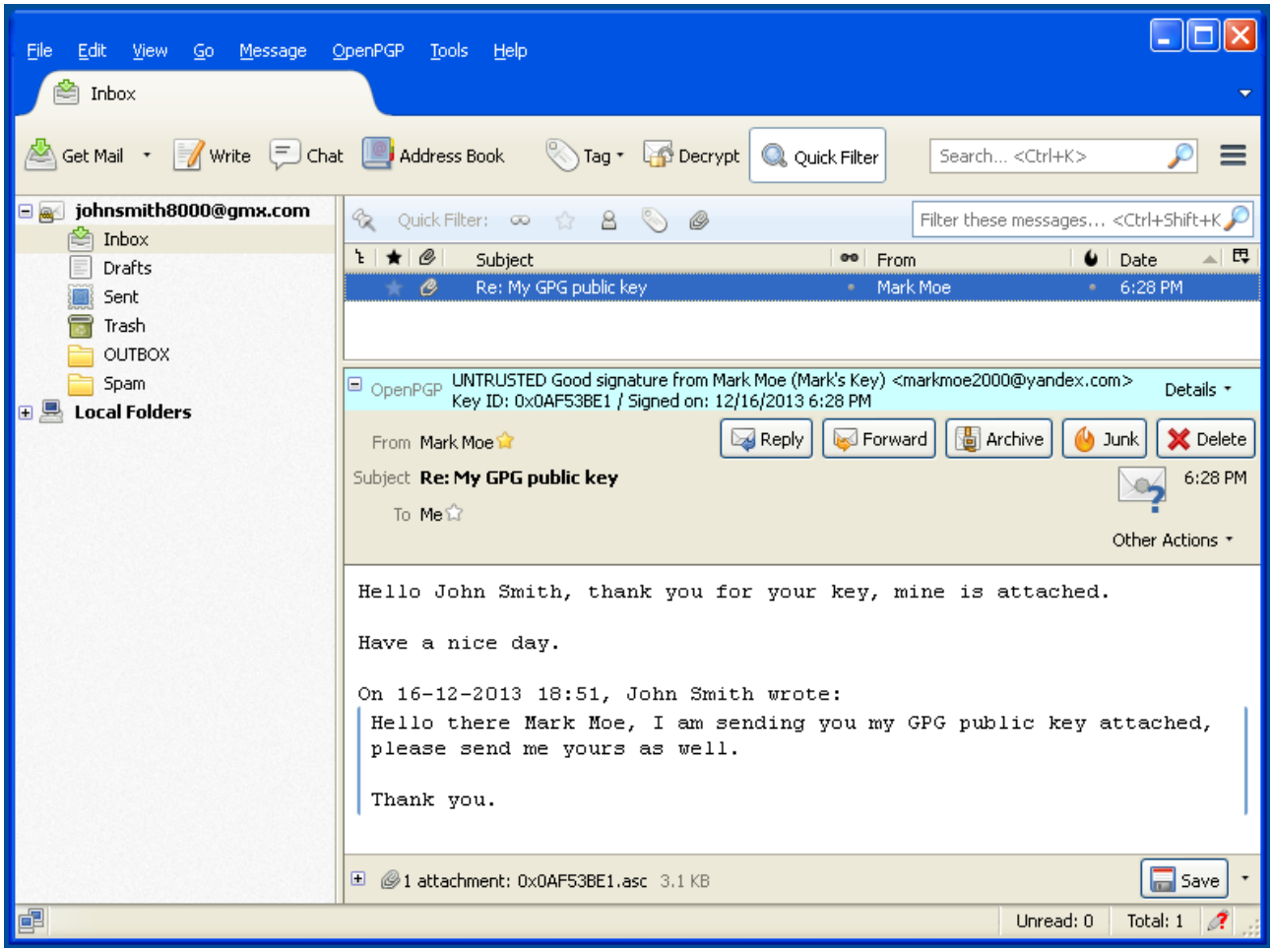


Figure 61: Public key has been imported. Yellow bar turned blue.

It says the signature is good because Thunderbird can now compare the signature in the message with the key you have imported, and it is correct, which means that the message is really from whom it claims it is (your contact).

It says it is untrusted because you have not defined the trust level yet, which you will do in section 8.6, but before doing it you need to verify the key's fingerprint.

3 – Verify the key's fingerprint

Click on the Details button on the blue bar and select OpenPGP Security Info.

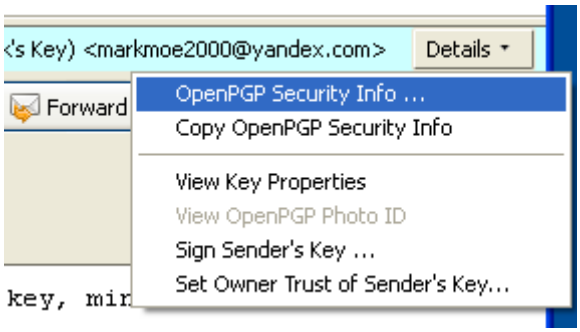


Figure 62: Accessing sender's key info

It will pop up a window similar to the one below, showing details about his key.

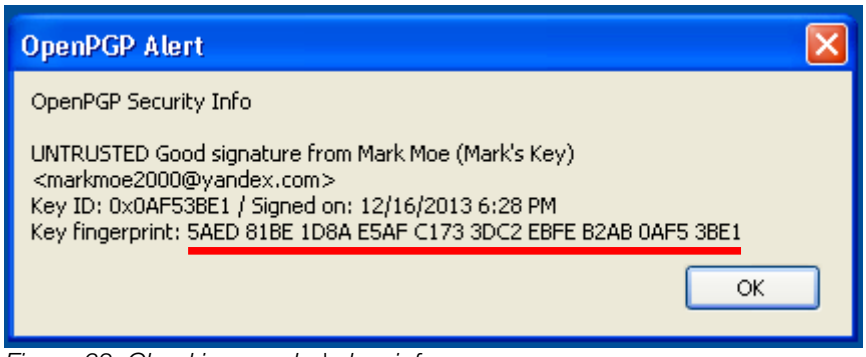


Figure 63: Checking sender's key info

The number highlighted in red is the key fingerprint. You should verify this number with your contact. This number is the only guarantee that you have received the correct key, and it was not modified along the way by an attacker or an intruder. Both of you should have the same number.

When you send your public key to others (as you did in section 8.4), they should also verify your key's fingerprint with you.

8.6 Setting trust level

The trust level is a value you define for each public key you have in your keyring of how much you trust the key's owner. For example: if a work colleague sends you his public key by e-mail, you verify the key's fingerprint with him and it is correct, then you can set the trust level to ultimately.

However if you obtain someone else's key from a dubious website or key server, and you cannot contact the owner to verify the key's fingerprint then you should choose a lower trust level for that key.

The trust level is a local classification and the key's owner will not know the value you have assigned to their key.

Click on Details button on the blue bar and select Set Owner Trust of Sender's Key.

Figure 64: Choose the last option

Now set the trust level you have on the sender's key.

Figure 65: Set the trust level

If you have set the trust level to ultimately, you will notice that the blue bar now becomes green and it says trusted. In our example below the bar is expanded. You can expand it by clicking on the '+' (plus) sign at the left.

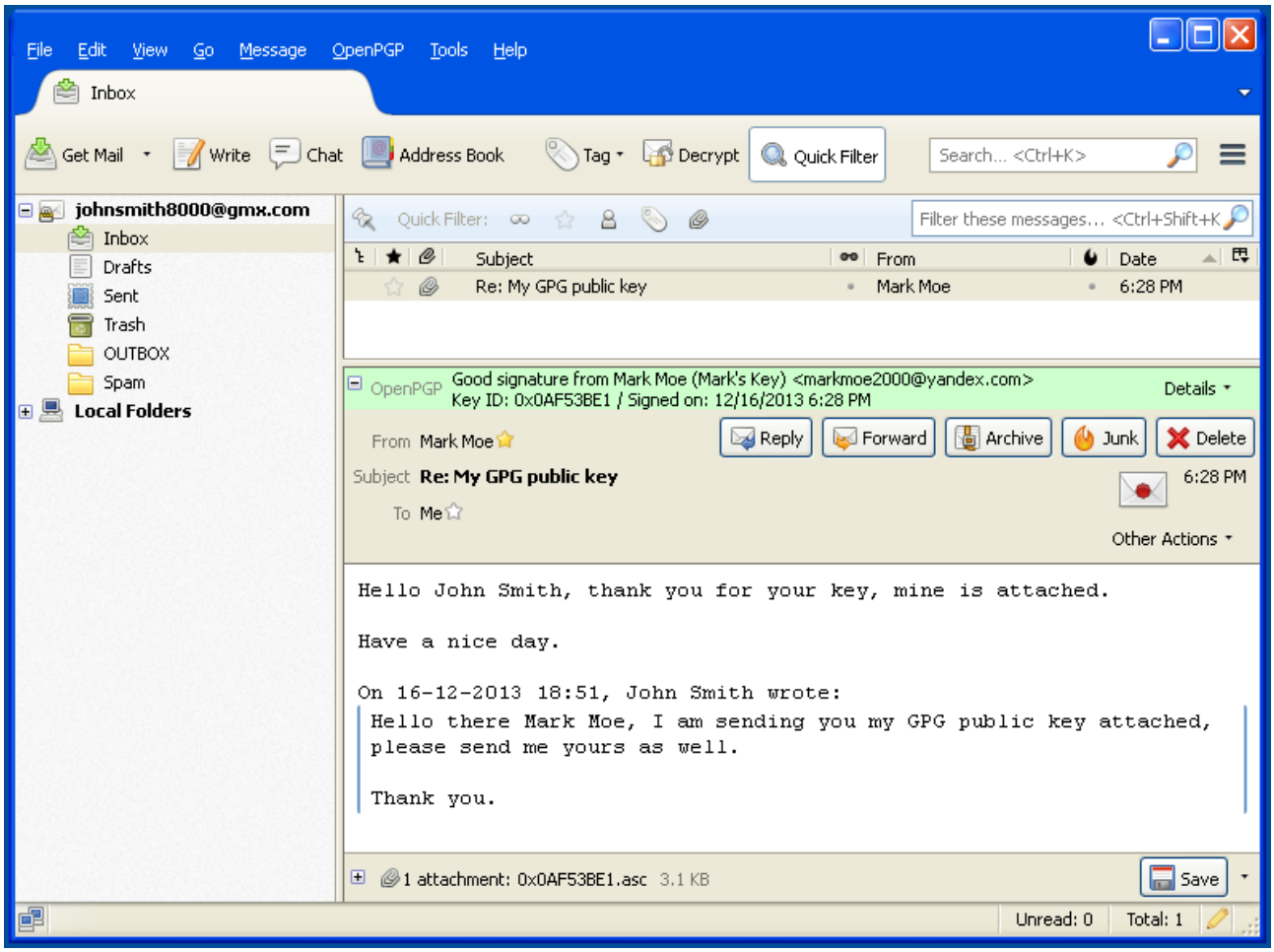


Figure 66: Blue bar turned green because trust level was set to ultimately.

8.7 Setting rules for your contacts

Rules are basically a combination of keys, e-mail addresses and actions (encrypting, signing, attaching) that you set for your contacts (or recipients). It is through rules that Thunderbird and Enigmail know how to behave with the recipient.

Enigmail is flexible and allows you to create very customized rules, but for simplicity sake all our rules will be the same for every recipient. You can modify them if you want.

1 – Open the settings window

Click on the name of your contact and select Create OpenPGP Rule from Address.

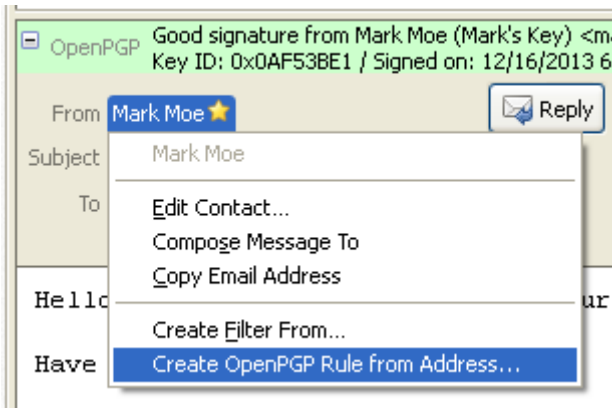


Figure 67: Open the settings window

2 – Select the right key

An advanced configurations window opens up. It shows your contact's e-mail address on top, which is the recipient you are creating a rule for. Ensure the second field is selected 'Is exactly'. You can leave all configurations the way they are for the moment. Click on Select Keys button.

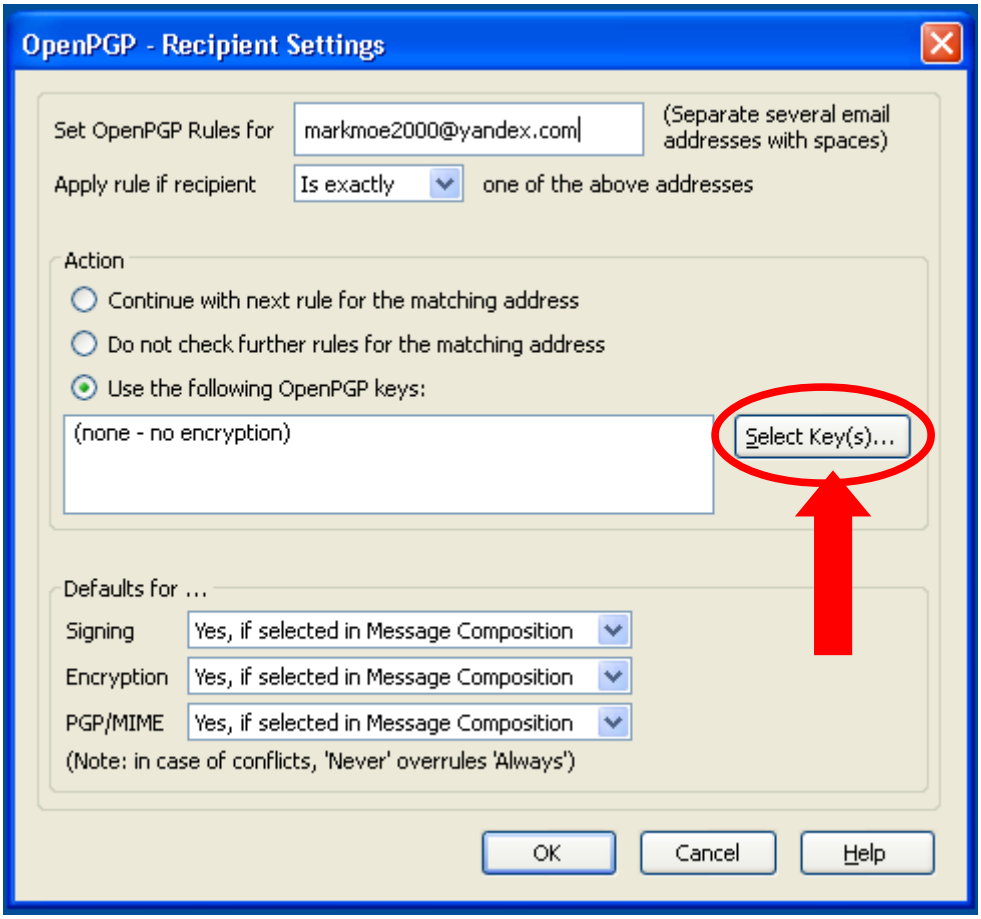


Figure 68: Recipient settings

A new window pops up showing the public keys you have in your keyring. Choose the contact's public key you are setting the rule for, and then click OK.

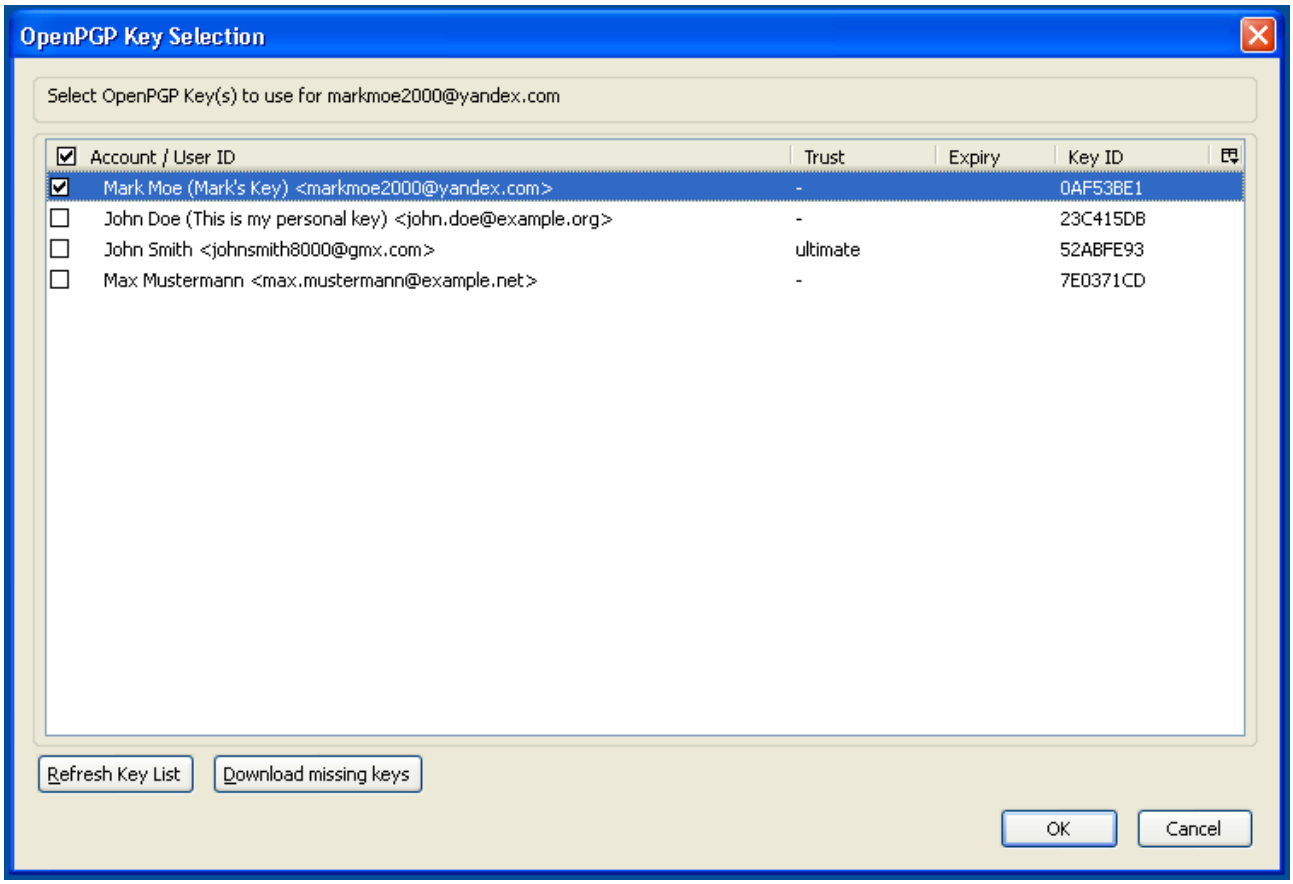


Figure 69: List of keys located in your keyring

3 – Set default behavior

Now you can see the field Action shows the key you have chosen in the previous step.

In the field “Defaults for...” set all fields to Always, as shown in the image below. This means that for this recipient you are setting the rule for, all messages will be sent always signed, always encrypted, and attachments will always be treated as PGP/MIME.

When you are done click on OK button.

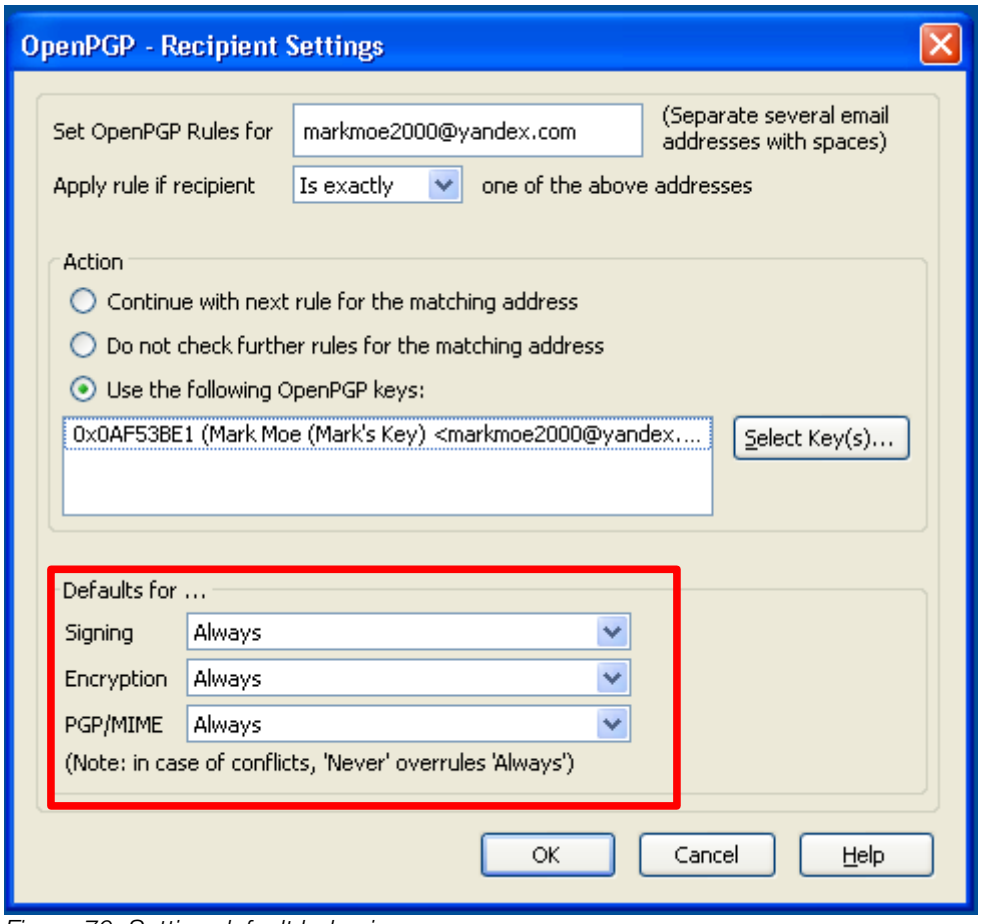


Figure 70: Setting default behavior

Now the rule is set for this recipient. Every time you obtain someone else's public key all you have to do is to repeat the process of this section and make the necessary adjustments.

Although rules can be customized a lot, keep in mind that the current configuration shown in this section is one of the safest possible.

PART 3

OTHER RESOURCES OF GNUPG

In this part you will learn:

- *What is a Revocation Certification*
- *How to create a Revocation Certificate*
- *How to Encrypt and Decrypt Files*
- *How to Sign and Verify Files*
- *How to Import and Export Certificates*
- *What are Key Servers and how to use them*

CHAPTER 9
Revocation certificate

A revocation certificate is a certificate to revocate (invalidate) your key and warn others that they must not trust in your key anymore. It should only be used if your key gets compromised (e.g.: lost, forgotten, erased, destroyed, robbed or violated). Since you are not able to use your key anymore, you have to warn other people about it.

9.1. How a revocation certificate works

Below there is an analogy to help you understand the damage that could happen in case your key is gets compromised, and why a revocation certificate is necessary:

Imagine that your wallet has been robbed with all your documents inside it. The robber might use your documents to impersonate you, commit crimes, sign documents, etc., all using your name, and there is nothing you can do prevent him from doing that. All you can do is to go to a police station and make a notification that your documents have been robbed. You will then be issued new documents, probably with different numbers, codes or dates, and then you will be able to use your new documents normally.

If anything shows up in your name between the time you were robbed and the time you notified the police, you will know it was done by the criminals. In other words, you cannot prevent the criminals from using your documents, but you can minimize the damage by taking these measures. And obviously the faster you notify the police the lesser the damage will be.

A revocation certificate has a similar purpose: if your private key gets compromised and you don't have a backup copy of it, you have to revoke it, warn others that your old key is no longer valid, generate a new key and give to others your new key, valid from now on.

You should create a revocation certificate as soon as possible, preferably right after you create your key pair because it is the only guarantee you have against possible damages.

It is important to note that a revocation certificate is really useful only if you distribute your key in a key server, because there is the place where most people will look for your key and synchronize it, otherwise you would have to warn one by one of them.

9.2. Creating a revocation certificate

1 – Start GnuPG

```
$ gpg2 --output certrevoc.asc --gen-revoke mykey
```

In the above command `certrevoc.asc` is the name of the file that will contain your revocation certificate. Change `mykey` by the identifier of your key.

2 – Choosing the revocation reason

```
sec  4096R/9C08F860 2013-12-23 John Doe (John's key) <john.doe@example.org>

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
```

The first step to create a revocation certificate is to choose a reason for it. By default GnuPG always suggests the second option (number 1), but you can choose between any of them. When you are done press ↵.

3 – Entering a description

```
Enter an optional description; end it with an empty line:
> I forgot the password
>
Reason for revocation: Key has been compromised
I forgot the password
Is this okay? (y/N) y
```

Here you can enter a description to complement the revocation reason chosen previously. This step is optional. After choosing the reason (or not), finish by leaving a blank line, confirm by entering y and press ↵.

4 - Entering your password

```
You need a passphrase to unlock the secret key for
user: "John Doe (John's key) <john.doe@example.org>"
4096-bit RSA key, ID 9C08F860, created 2013-12-23
```

Enter your password to finish the process. If you are using the command line then your password does not show up while you type.

5 - Conclusion

```
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!
```

After conclusion, GnuPG shows a message suggesting how to protect your certificate (see more instructions below).

To verify your certificate just type the command below in the terminal (if you are using Windows change `cat` by `type`):

```
$ cat certrevoc.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.20 (GNU/Linux)
Comment: A revocation certificate should follow

iQI0BCABAgAeBQJSuIduFx0CSSBmb3Jnb3QgdGhlIHBhc3N3b3JkAAoJEGpPa/Kc
CPhgt30P/0YNKJnvA5+zn1vHgE3CamfVoa8UkiYXxcS8wK1aF/ceJYZXsKN/GhYV
a9OZoQ/vmTyAj9dxvLHp7+32vtDG7xNHmURpfRqHmG4xafY4FD9ceKpFB3DT4NX9
CJslx/lEsFdb4mV1IOMaSvazm5qLEtwTqFhBj7AY84tFEkQT70Cax4PgE7iPQef9
BdH7DmDBCsLkJ8qV6SmWVaEtDhTAOFOWdmnVEDp+gvugjEOGcPQlDPFRASSikL+c
kNK2E+6kWmtaHvLHpIBxcjWL6zfuppI2+MqR+rfAQLCureIqoY62x176Tk/4tKEC
vkeCAtSoyNP+lprHadK916ed0l9ywsEAcmzcRG8f8gjA5bm94B6GbXr9QavkYTRV
CdTtj70Y7I0H3rHJ5E4NL7aZfMU49OYAAer/LNlgcfUFsFHeGNSrlf2ivfWTenrB
1bbkEnQGj4Ln9tNPNRtoKH606HQPvByPXElCth8Xfv9IG7AIFd1RrSe84hZlGN3V
xsK74cPxn+B8QxgYtgbR7OxnZikMsBlgPaYGAklUnmfIoy+O9rie79vFN81MqipZ
wRVbu+Ki2mh1qPHdLnGP79rbgTQJVVDkFCHWHkorFYoCoB4NYIrafnLrb2qwR2C+
2kSzE2H4K1ZXBB84tfkvDmTk5hVqrsxDxRRjO21uQxskoJ76DFCP
=R+j7
-----END PGP PUBLIC KEY BLOCK-----
```

9.3. Storing your certificate

Your revocation certificate is your only guarantee in case your private key gets compromised, so it is very important that you protect it carefully.

One idea might be to print a copy of your certificate, or store it in a CD/DVD-ROM disc or USB drive, and store it in a safe or another safe place in your house.

9.4. Revoking your key

Revoking a key is an easy process, however it is recommended that you read chapter 17 before doing this so you may better understand the implications of using a key server.

The basic process is to revoke the key locally and then upload it to a key server.

1 – Import your revocation certificate

First you import your revocation certificate (generated in step 9.2) into your keyring:

```
$ gpg --import revoked.asc
```

Now your key is unusable because it has already been revoked.

2 – Send it to a key server

Send your revoked key to a key server (change `key_ID` by your actual key ID):

```
$ gpg --keyserver pgp.mit.edu --send-keys key_ID
```

Now your key is publicly revoked. Next time someone searches for your key or refreshes their keys database they will know that your key has been revoked. It is also important to generate a new key and publish it so people can still contact you.

For more information on how to use key servers check out chapter 17.

CHAPTER 10
Encrypting and decrypting

Encrypting and decrypting files is the main purpose of GnuPG, you can do it for yourself or for others. There are two ways to do it: using symmetric and asymmetric. encryption.

10.1 – Encrypting files

In GnuPG you can encrypt files for yourself and for others. There are two ways to do this: using symmetric encryption and asymmetric encryption. For more information about these methods check out chapter 3.

10.1.1 – Through asymmetric encryption

This is the most common method of encrypting files for others. You need the other person's public key to do it. You can also use it to encrypt files for yourself.

Syntax:

```
$ gpg2 --encrypt --recipient recipient_ID file_name
```

The recipient's ID can be any identifier of the key, such as the ID, fingerprint, e-mail address or name. It is a good practice to enclose it in single quotation mark.

Usage example #1:

```
# Encrypting a file using e-mail address as recipient:
$ gpg2 --encrypt --recipient myfriend@example.org Document.pdf
```

Usage example #2

```
# Encrypting a file using the key ID number as recipient:
$ gpg2 --encrypt --recipient A1B2C3D4 Document.pdf
```

Usage example #3:

```
# Encrypting a file using name as recipient
$ gpg2 --encrypt --recipient 'John Doe' Document.pdf
```

In our examples the resulting file is called `Document.pdf.gpg`.

10.1.2 – Through symmetric encryption

This method is recommended to encrypt files for yourself only, since it uses a single password and does not specify a receiver.

Syntax:

```
$ gpg2 --symmetric file_name
```

Usage example:

```
# Encrypting a file through symmetric encryption:
$ gpg2 --symmetric FamilyPictures.zip
```

In our example the resulting file is called `FamilyPictures.zip.gpg`.

10.2 – Decrypting files

You may need to decrypt files from others or the ones you encrypted yourself. The syntax to do it is the same. It is necessary to have the sender's public key to decrypt files.

Syntax:

```
$ gpg2 --output output_file --decrypt file_to_be_decrypted
```

The recipient's ID can be any identifier of the key, such as the ID, fingerprint, e-mail address or name. It is a good practice to enclose it in single quotation mark.

Usage example #1:

```
# Decrypting a file to a file:
$ gpg2 --output Book.pdf --decrypt Book.pdf.gpg
```

In this example the file is output to another file. It is the preferred method to decrypt files. In our example the resulting file is called Book.pdf.

Usage example #2:

```
# Decrypting a file to the screen:
$ gpg2 --decrypt Message.txt
```

In this example the file is output to the screen. This method should only be used for short text files, or when combined with more advanced piping commands.

10.3 – Changing the output filename

By default output files from GnuPG are named according to the original file, adding the adequate extension. For example:

```
file.txt → file.txt.gpg (binaries)
file.txt → file.txt.asc (encoded text)
file.txt → file.txt.sig (signatures)
```

You can easily change this behavior and choose the name you desire for the output file, as indicated in the examples below:

Syntax:

```
$ gpg2 --output desired_name --encrypt --recipient recipient_ID file_name
```

The recipient's ID can be any identifier of the key, such as the ID, fingerprint, e-mail address or name. It is a good practice to enclose it in single quotation mark.

Usage example #1:

```
# Encrypting a file and changing its output name:
$ gpg2 --output MSG.gpg --encrypt --recipient myfriend@example.org Message.txt
```

In this example the file `Message.txt` after being encrypted will be named `MSG.gpg`.

The output filename change also work with other GnuPG operations, such as signing, which is covered in the next chapter but can be seen in the example below:

Usage example #2:

```
# Signing a file and changing its output name:
$ gpg2 --output SignedMessage.sig --detach-sig Message.txt
```

In this example it is generated a detached signature of the file `Message.txt` which is called `SignedMessage.sig`.

10.4 – Choosing between multiple keys

If you have multiple private keys in your keyring you will have to choose between them depending on the operation and the recipient you are working with, otherwise GnuPG will use the key that is set as default.

To choose a key between multiple private keys use the option `--local-user` after the desired operation, as shown in the examples below:

Syntax:

```
# Encrypting a file:
$ gpg2 --encrypt --local-user user_ID --recipient recipient_ID file_name

# Signing a file:
$ gpg2 --output output_file --sign --local-user user_ID file_name
```

Usage example #1:

```
# Encrypting a file:
$ gpg2 --encrypt --local-user Fred --recipient Mary Message.txt
```

Usage example #2:

```
# Signing a file:
$ gpg2 --output SignedMessage.sig --sign --local-user Peter Message.txt
```

As you could notice, this step also works with other operations such as signing, covered in the next chapter.

CHAPTER 11
Signing and Verifying Files

A digital signature has two purposes: to ensure the authenticity of the sender (and not someone impersonating him/her), and to ensure that the information is original and was not twisted along the way. In a way it is similar to a physical signature in a cheque or in a contract, but despite marking the sender's identity, it also marks the time the information was signed, thus offering double security.

As a good practice you should sign files every time you encrypt them.

11.1 – Making signatures

There are three ways to make signature with GnuPG: generating an unreadable signed file, generating a readable signature, and generating a detached signature. Each one has different uses and purposes:

11.1.1 – Binary signature (unreadable)

This method generates a new file in binary format containing the original file (now compressed) plus the signature. This method is recommended to be used with non-text files.

Usage example #1:

```
# Sign and generate a binary file:
$ gpg2 --sign file.txt
```

A file named `file.txt.gpg` is generated.

Usage example #2:

```
# Sign and generate an ASCII-armored text file:
$ gpg2 --sign --armor file.txt
```

A file named `file.txt.asc` is generated.

11.1.2 – Clear signature

This method generates a new file in text format containing the original file plus the clear signature in the end. This method is recommended to be used with e-mail messages, online

forum posts and discussion lists, since it does not compress or modify the original file, only the signature is added in the end.

```
# Sign and generate a readable signed file:
$ gpg2 --clearsign file.txt
```

A file named `file.txt.asc` is generated, containing the original file plus the signature.

11.1.3 – Detached signature

This method generates a new file containing the signature only. This method is recommended to be used when the original file may be distributed through several different ways, such as for download on different websites, since the signature may be obtained apart.

Usage example #1:

```
# Generate a file containing the signature in binary format:
$ gpg2 --detach-sig file.txt
```

A file named `file.txt.sig` is generated containing the signture only.

Usage example #2:

```
# Generate a file containing the signature in ASCII-armored format:
$ gpg2 --detach-sig --armor file.txt
```

A file named `file.txt.asc` is generated containing the signature only.

11.2 – Verifying signatures

This process is used to verify if the signature corresponds to the author of the original file. It can be done either for attached or detached signatures.

Usage example #1:

```
# Verify signature attached to a file:
$ gpg2 --verify file.sig
```

This example is used to verify binary and clear signatures.

Usage example #2:

```
# Verify signature detached from a file:
$ gpg2 --verify file.sig file.txt
```

This example is used to verify detached signatures.

11.3 – Extracting files from signed files

After you verify the file's signature you may want to extract the original

When you obtain a signed file and verify its signature you may want to extract the original file from it. Another reason for that is that signed files are often given encrypted. You can extract it using the `--decrypt` command, as shown below:

Syntax:

```
$ gpg2 --output output_name --decrypt signed_file
```

Example:

```
$ gpg2 --output file.txt --decrypt file.txt.sig
```

This way the file will be extracted to a file named `file.txt`.

11.4 – Choosing between multiple keys

Check out chapter 10.4 for more information on this.

CHAPTER 12
Importing and Exporting Certificates

To export a certificate means to generate a copy of a certificate located in your keyring to a file where it could then be moved or sent to others. To import a certificate means to insert a certificate from a file or from the internet into your keyring where it can then be used.

To sign, verify, encrypt, decrypt and certify, you often need to import others' certificates, and export yours to them.

12.1 – Exporting certificates

12.1.1 – Exporting your public key

The public key is the key you make available for others to communicate with you. It is only through this key that others can contact you privately.

As it is located in your keyring, you first need to export it to a file, and then make this file available to others. To export your public key use the command below:

Syntax:

```
$ gpg2 --export --armor --output output_file key_ID
```

Example:

```
$ gpg2 --export --armor --output mykey.asc joe.bloggs@example.com
```

Your public key has now been exported to the file `mykey.asc`.

You can give this file to other people by any means you wish: through CD/DVD-ROM disc, USB drive, send by e-mail, you can publish it in a key server in the internet or make it available for download in your website, blog or social network.

12.1.2 – Exporting your private key

The private key is your unique, personal and untransferable key, so you must never give it or send it to anyone. Ideally you should only export your private key to make a backup copy or to use it in another computer that you own.

To export your private key use the command below:

```
$ gpg2 --export-secret-keys --armor --output mykey.asc myself@example.com
```

Your key has now been exported to file `mykey.asc`

12.1.3 – Exporting your whole keyring

Normally your whole keyring would only be exported to transfer it to another machine or to do a backup copy. We will present you two different ways to do it:

Using a single file

This way you will first export your public keys to a file and then export the private keys to the same file by appending to it.

```
$ gpg2 --export --armor > keyring.asc
$ gpg2 --export-secret-keys --armor >> keyring.asc
```

Your keyring has now been exported to file `keyring.asc`.

Using two files

This way you will export your keyring to two different files, one containing the public keys and the other containing the private keys. It is recommended that you do it this way.

```
$ gpg2 --export --armor --output pub_keyring.asc
$ gpg2 --export-secret-keys --armor --output sec_keyring.asc
```

Your keyring has now been exported to files `pub_keyring.asc` and `sec_keyring.asc`. Now when you want to import your keyring first import `pub_keyring.asc` and then `sec_keyring.asc`.

12.2 – Importing keys and certificates

12.2.1 – Importing certificates from a file

To import public keys, private keys, whole key rings. or certificates of any kind use the command below:

```
$ gpg2 --import certificate.asc
```

Now your certificate is imported and ready to be used.

12.2.2 – Importing certificates from key servers

Check out chapter 17 for more information on this.

CHAPTER 13
Encrypting and Signing Files

A digital signature has two purposes: to ensure that who sent the information is really who he claims to be (and not someone impersonating him), and to guarantee that the information is original and was not modified along the way.

In a way it is similar to a physical signature in a cheque or in a contract, but it also stamps the time the signature was made, thus offering double security for the receiver.

Digital signatures are often used together with encryption.

1. Choosing the file

Right-click on the file and choose Sign and encrypt, as shown in Figure 1.

It is also possible to realize the same process through Kleopatra's main window by clicking on menu File → Sign/Encrypt files.

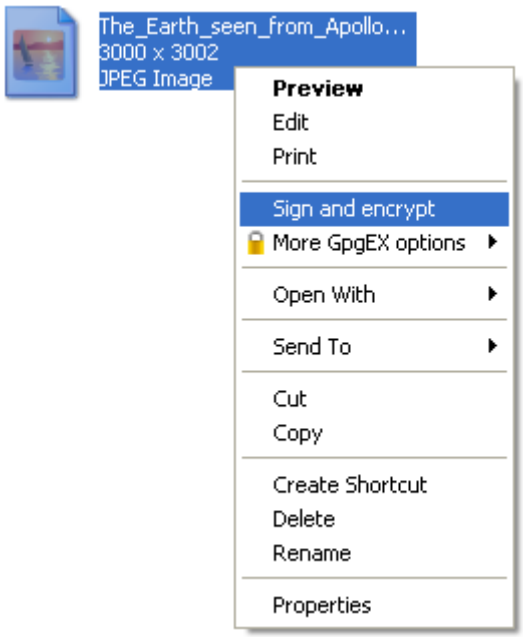


Figure 71: Choose Sign and encrypt

2. Choosing actions

You have the option to Sign, to Encrypt or to do both actions. It is a good practice to always encrypt and sign, so we will do both. Choose Sign and Encrypt (OpenPGP only).

If you want to send the file by e-mail, or you would like to have it available in pure text, choose the option Text output (ASCII armor).

You can also change other options if you want, but we will leave them as default.

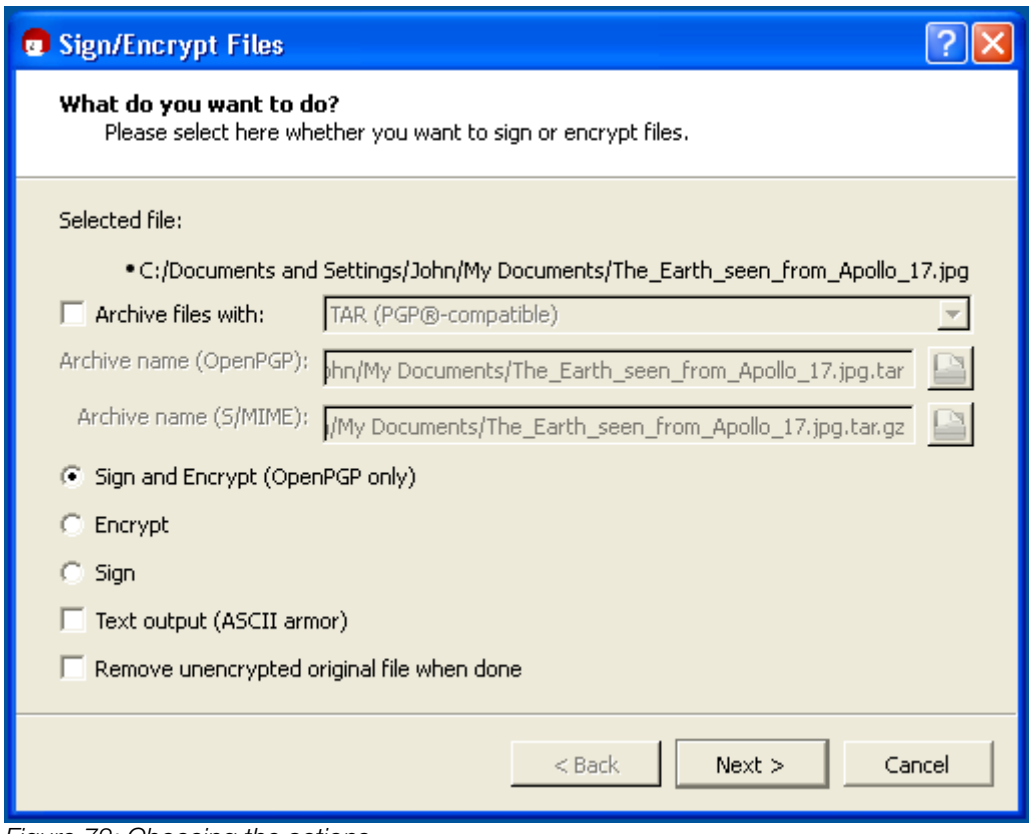


Figure 72: Choosing the actions

3. Choosing the receiver(s)

In the above field choose the person to whom you are sending the file by clicking on the person's name, and then click on Add button. The person's name and address will be shown on the field below which is the field of the receivers.

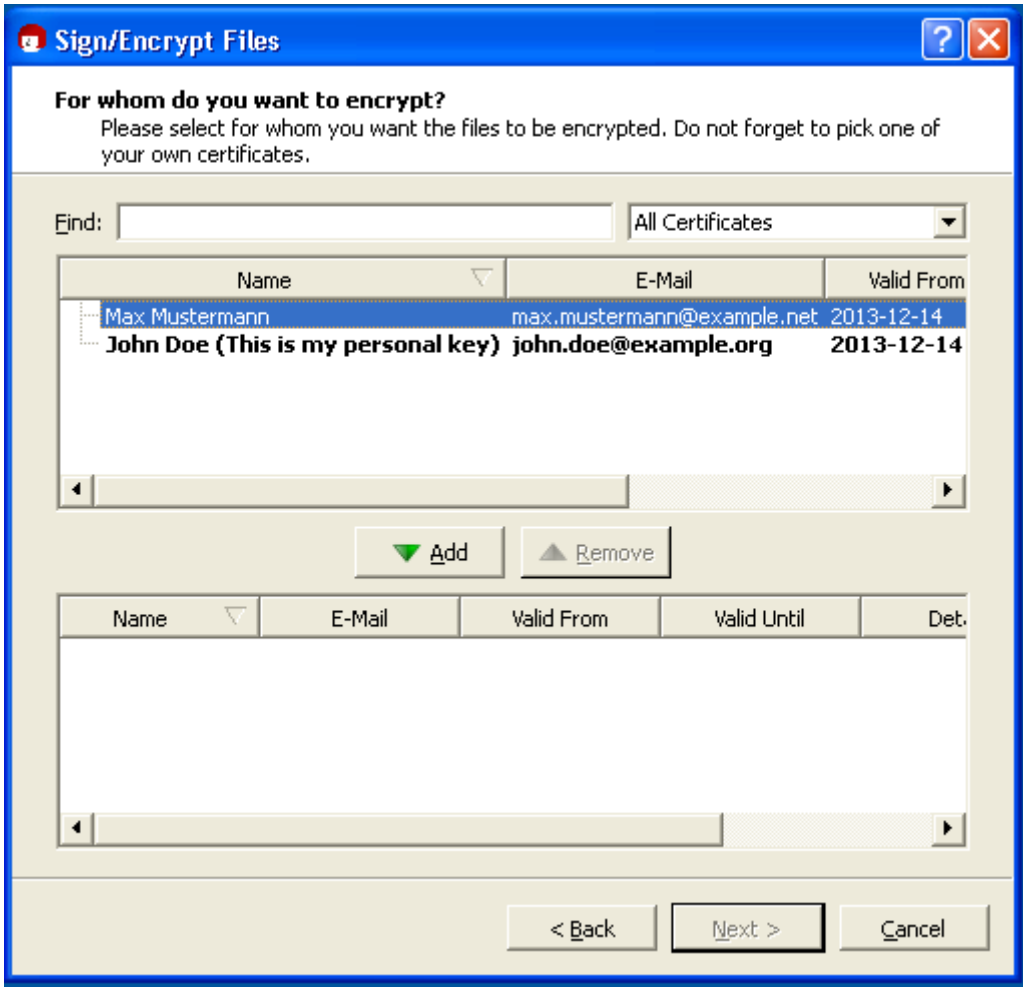


Figure 73: Choosing the recipient(s)

You can choose as many recipients as you want, including yourself. In our example we chose only Max Mustermann, as shown in Figure 4. If you want to remove a person from the receiver's field just select the person and click on the Remove button.

When you are done click on the Next button.

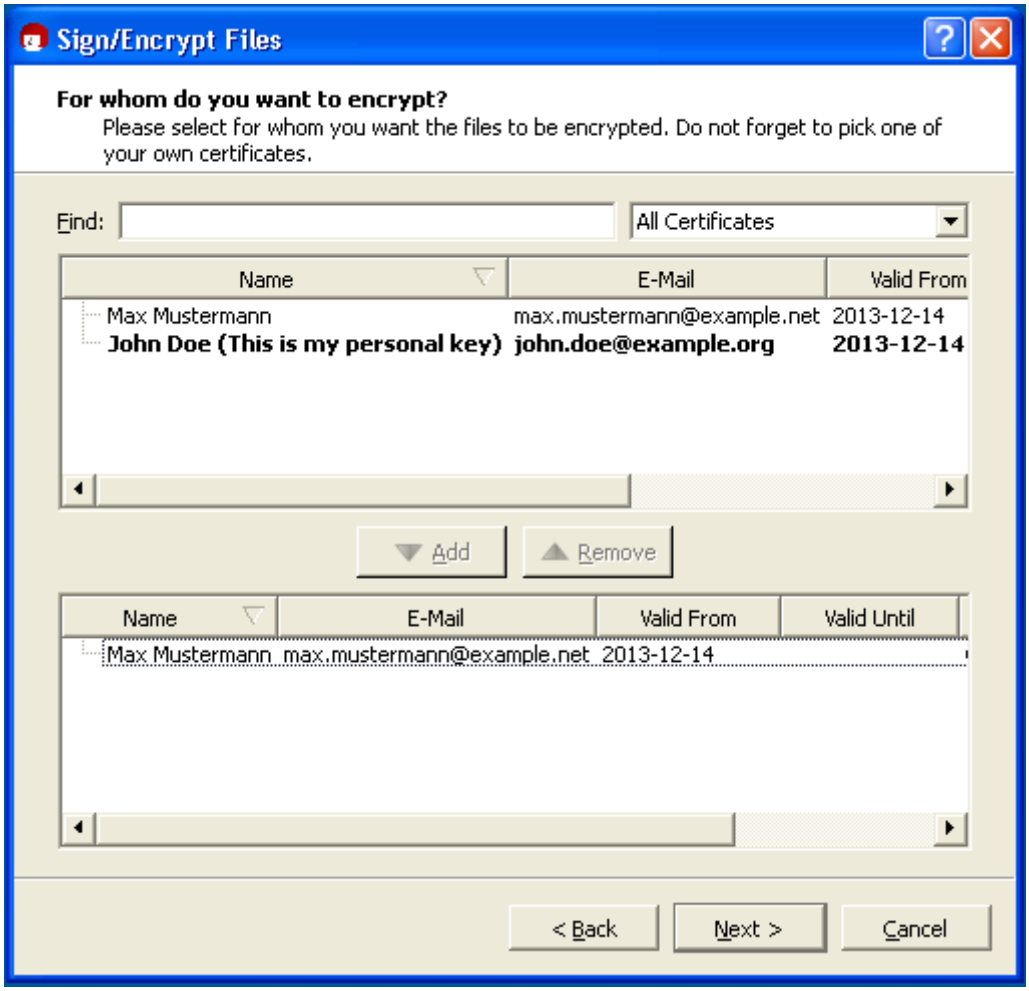


Figure 74: List of persons chosen

4. Warning

If you did not add yourself to the receiver's field, Kleopatra will issue a warning informing you that you will not be able to decrypt the file you are sending to another person. It is often a good practice to add yourself too, so you can go back and add yourself if you want.

However if you keep a copy of the original file, or you do not mind not being able to decrypt it, you may click on Continue button to proceed. and/or check Do not ask again so Kleopatra will not issue this warning again in future.

We will click on Continue button.

Figure 75: Warning message

5. Choosing the private key

Now you have to choose which private key you want to use to encrypt and/or sign the file. If you have more than one key – or intend to have more than one key in the future – you may just choose the one you want to use now and leave the check box unchecked.

However if you only use a single key you may choose it and check the check box below so Kleopatra will not prompt you about it anymore.

When you are done click on Sign & Encrypt button.

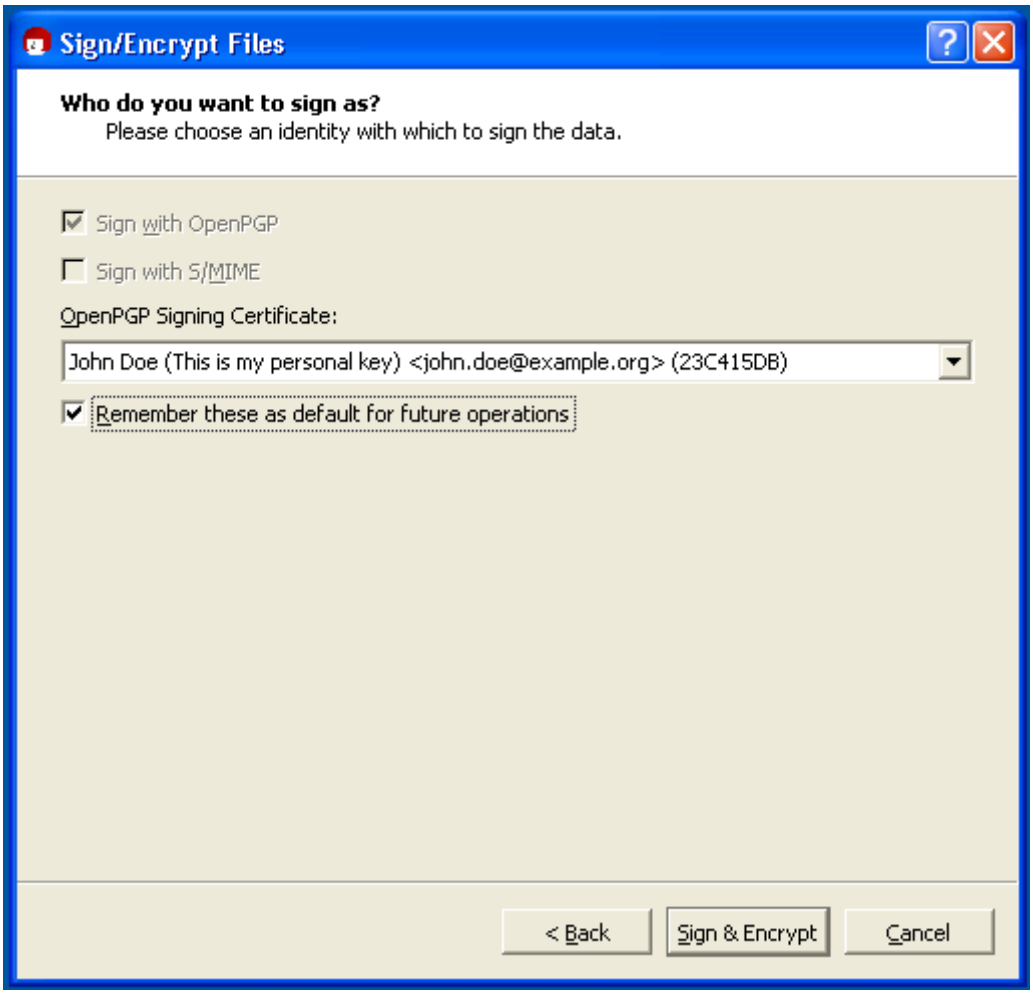


Figure 76: Choosing the private key you want to use

6. Password

Enter your private key password if requested.

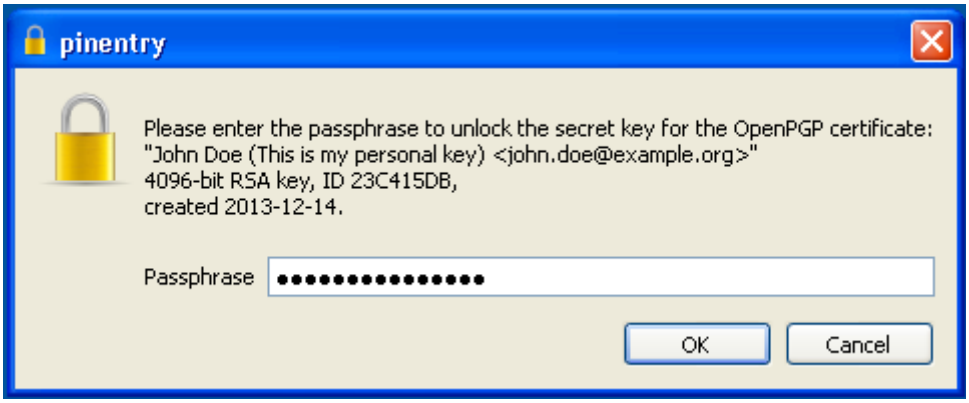


Figure 77: Enter password

7. Wait for the operation to finish

Wait for the signing and/or encryption operation to finish.

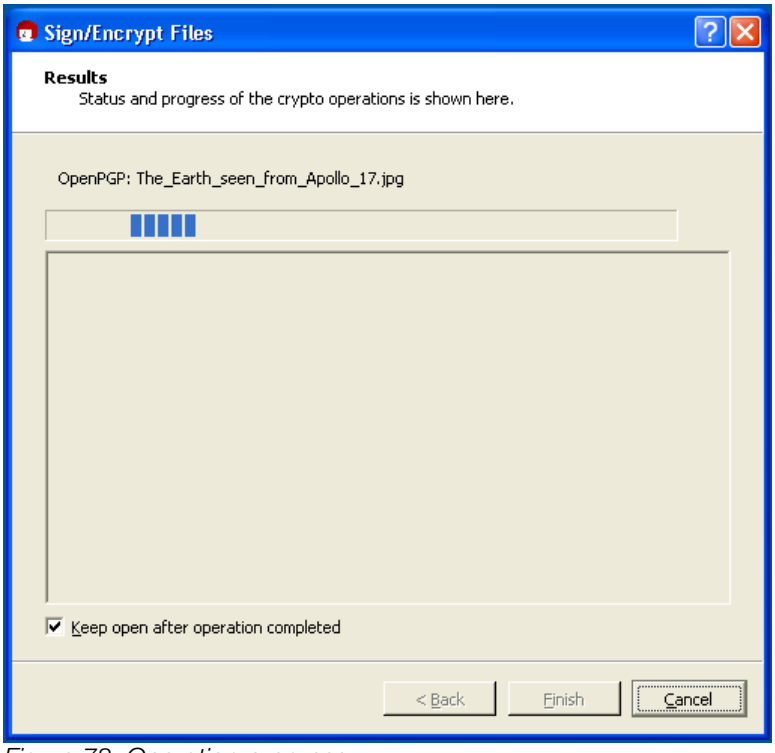


Figure 78: Operation progress

7. Conclusion

After the operation is finished, it will be created an encrypted file in the same folder of the original file, or in a different place if you have chosen one.

Just click on the Finish button or close the window.

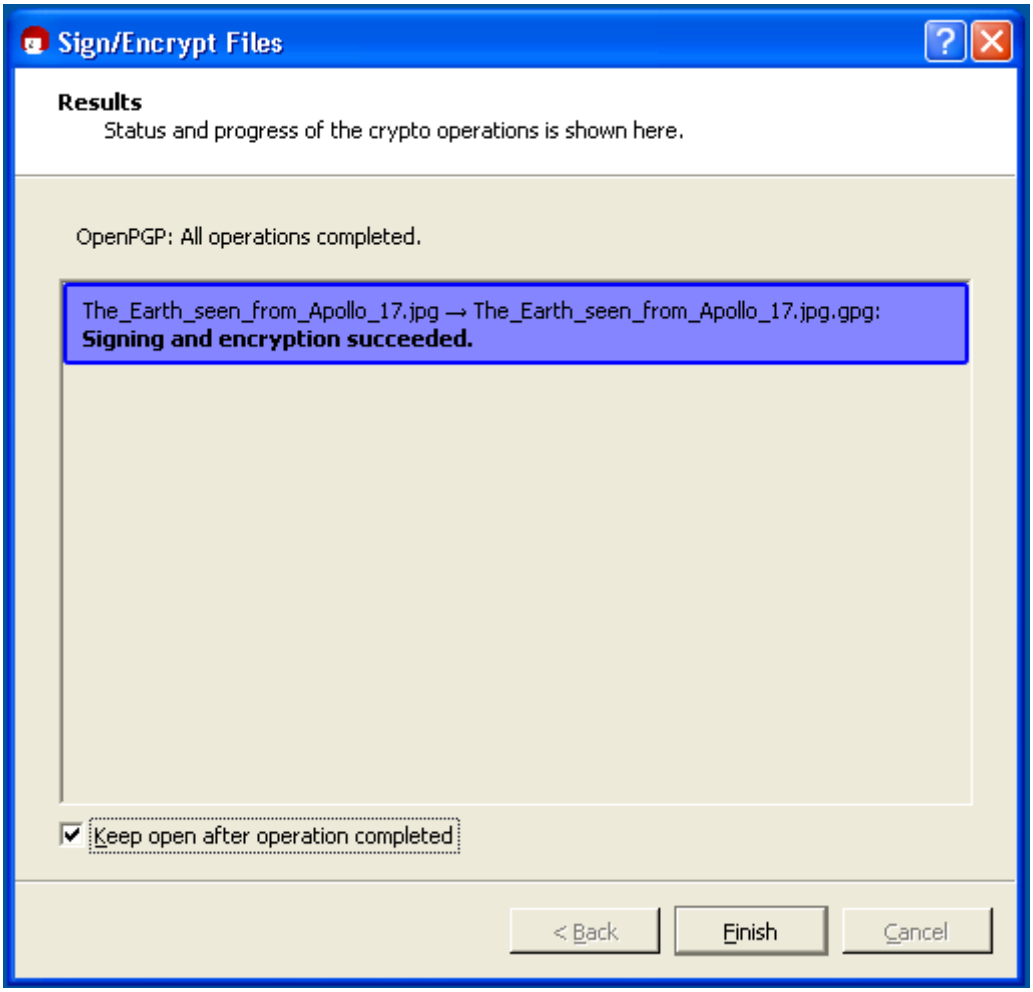


Figure 79: Conclusion

CHAPTER 14
Decrypting and Verifying

To decrypt a file you must have the sender's public key in your keyring, and to verify the signature of a file you must have the original file and/or the signature.

1. Choose the file

Right-click on the file and choose Decrypt and verify, as shown in Figure 1.

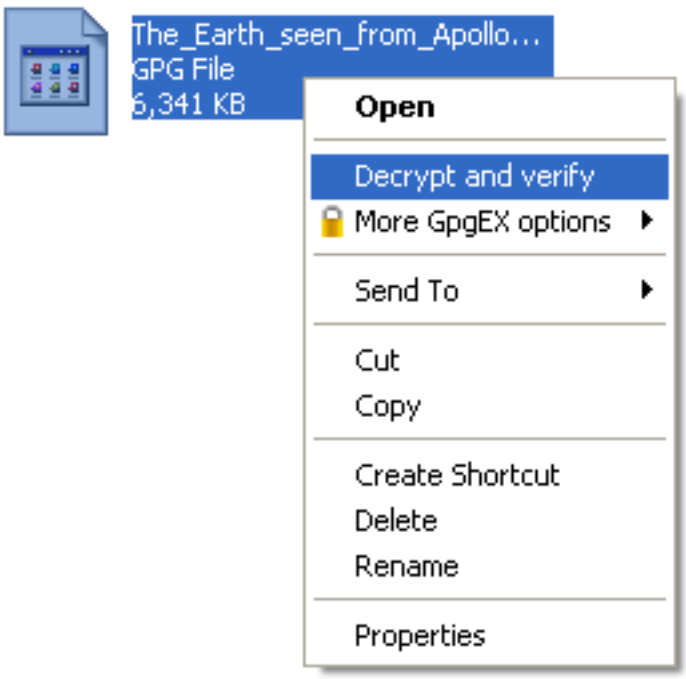


Figure 80: Choose Decrypt and verify

2. Perform the action

If you are verifying a file with a detached signature, check the checkmark 'Input file is a detached signature' and click on the folder icon button to choose the detached signature file.

When you are ready to decrypt and/or verify the file click on Decrypt/Verify button.

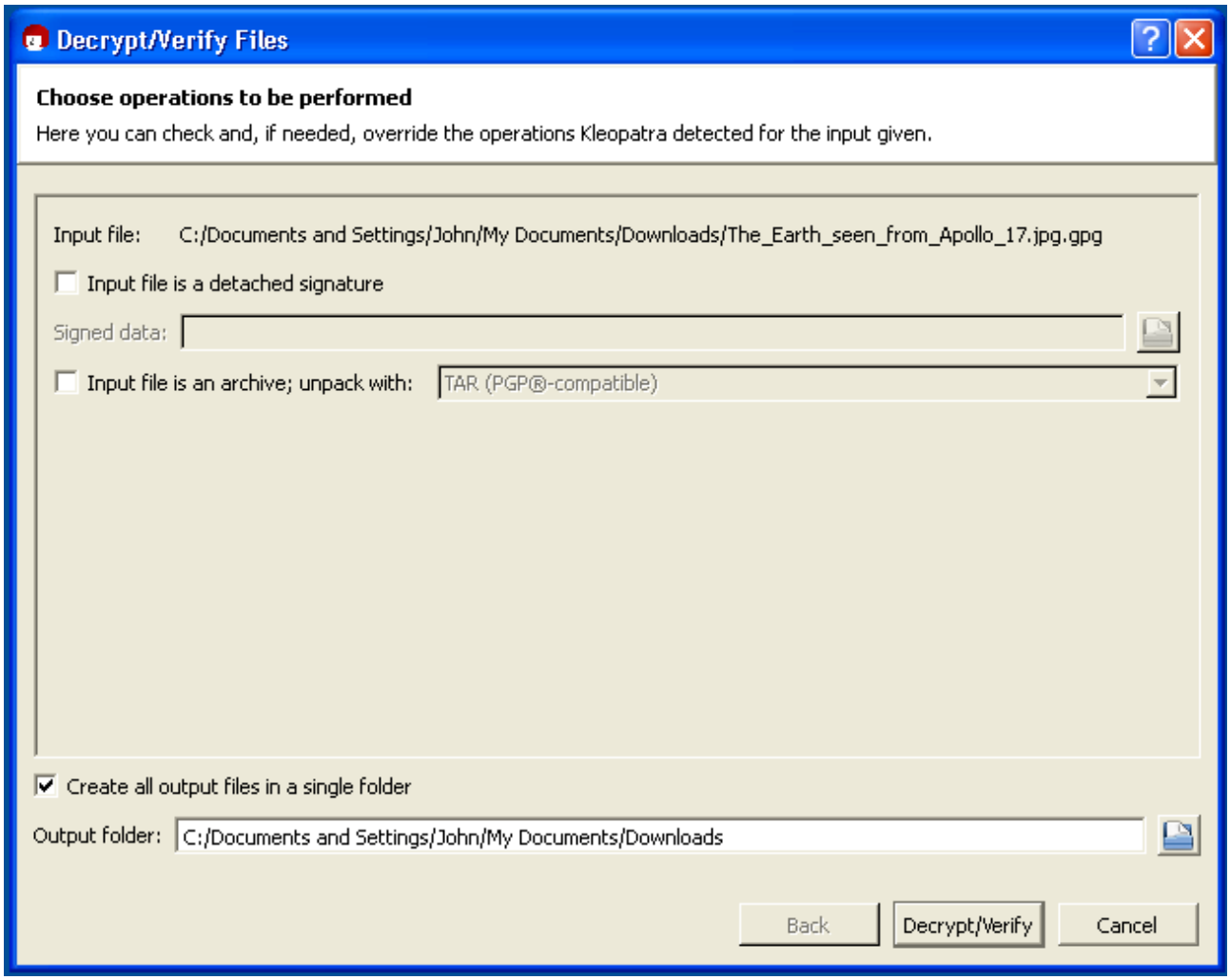


Figure 81: Perform the action

3 – Enter password

Enter your key password if requested.

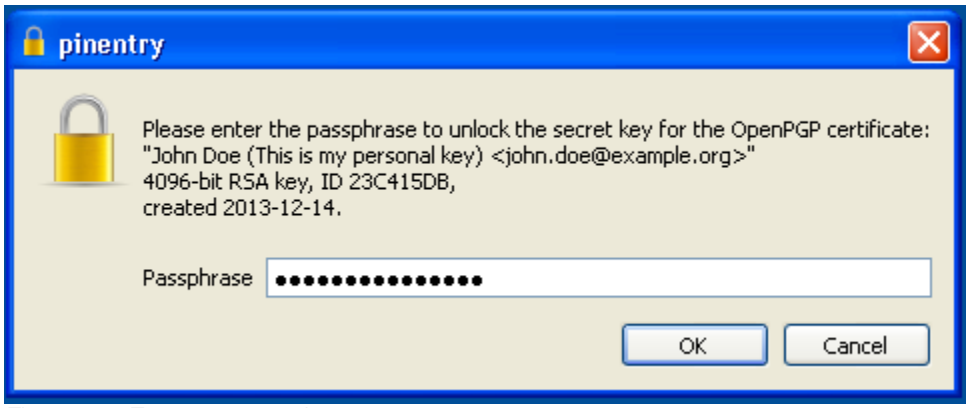


Figure 82: Enter password

4 – Wait for the operation completion

Wait the operation completion.

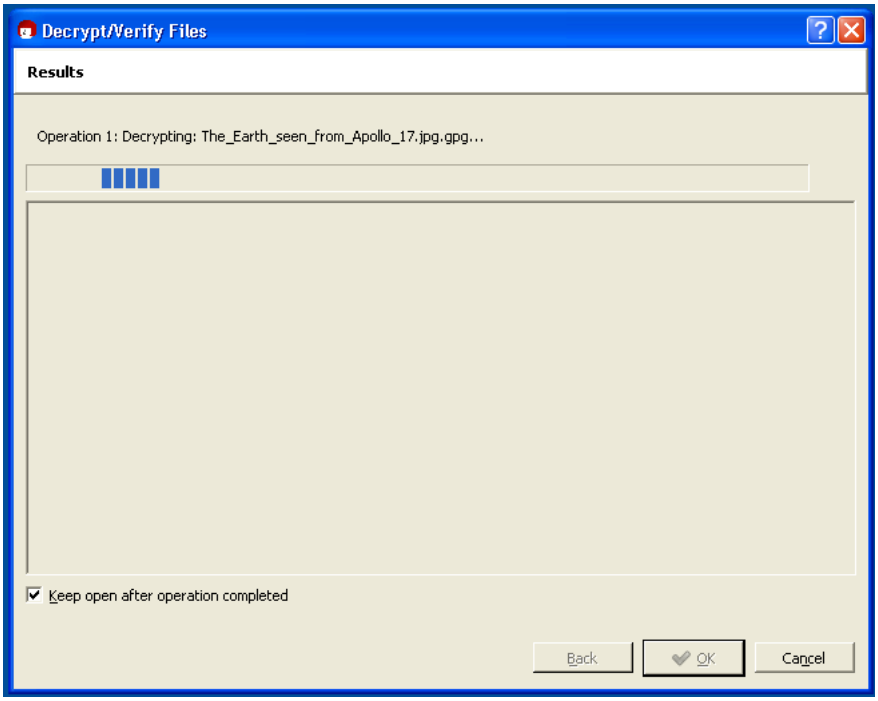


Figure 83: Wait the operation completion

5 – Operation completed

The operation is now completed. If Kleopatra could validate the sender's signature you should see a green bar as in Figure 5, otherwise it would show a yellow bar.

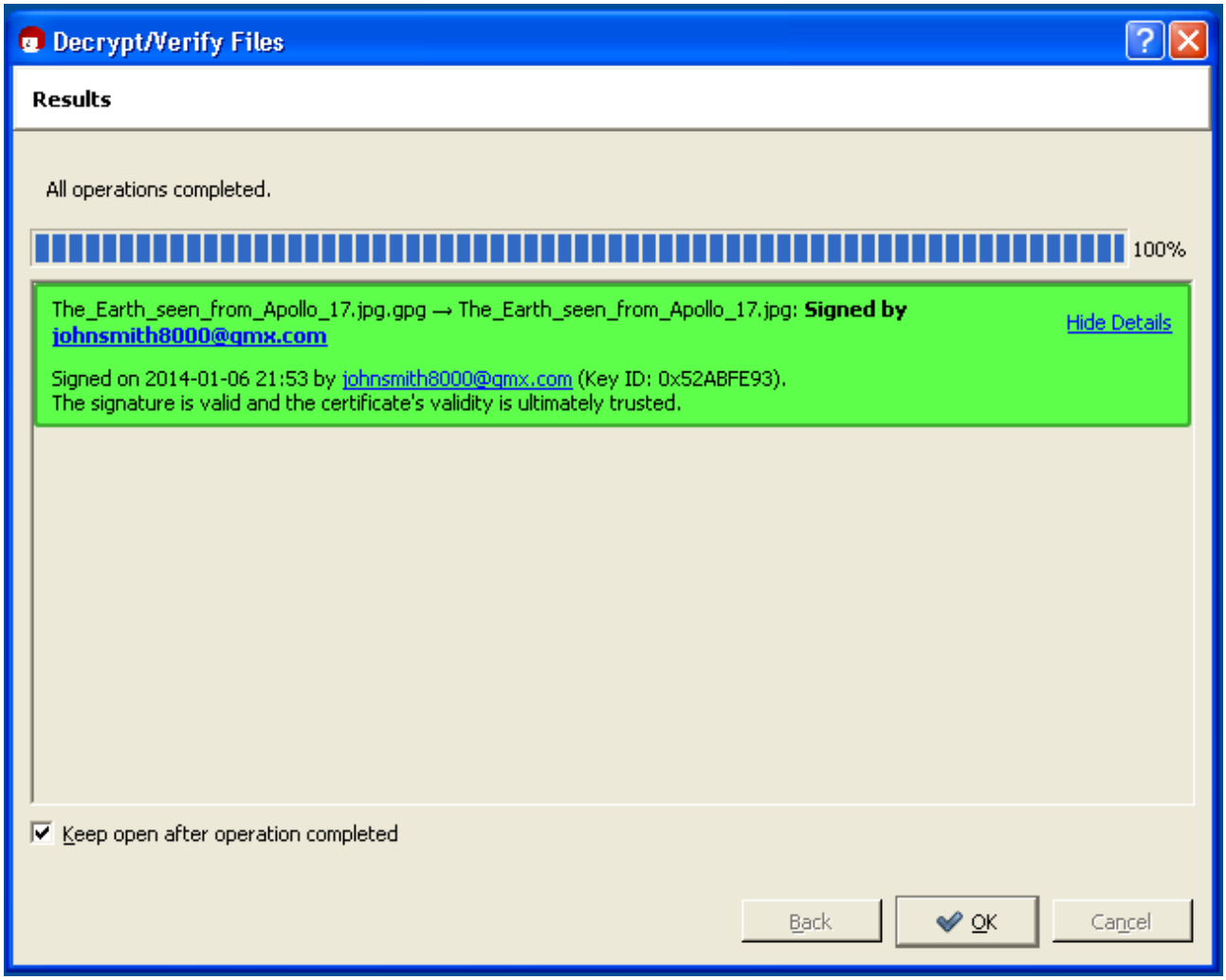


Figure 84: Operation completed

CHAPTER 15
Importing and Exporting Certificates

To export a certificate means to generate a copy of a certificate that is in your keyring to a file where it could then be moved or sent to others. To import a certificate means to insert a certificate from a file into your keyring where it can then be used.

To sign, verify, encrypt, decrypt and certify, you often need to import others' certificates, and export yours to them.

15.1. Exporting your public key

The public key is the key you make available for others to communicate with you. It is only through this key that others can contact you privately.

To export your public key open Kleopatra, right-click on your key and select Export Certificates, or press Ctrl E:

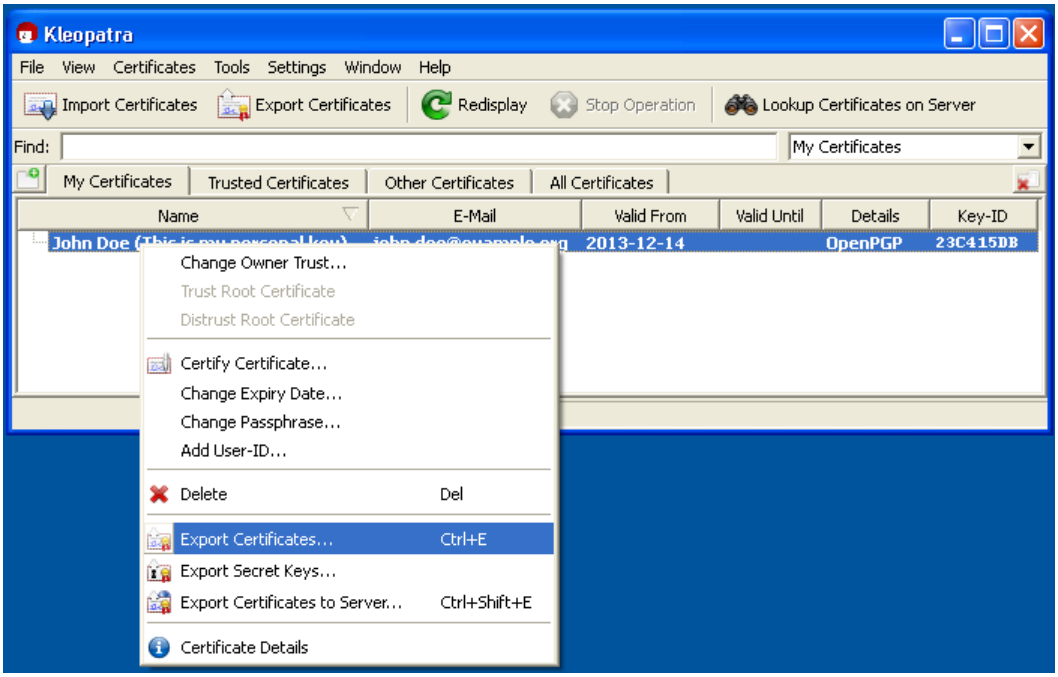


Figure 85: Exporting public key

Now choose where you want to save your public key. By default Kleopatra suggests the key's fingerprint as a name, but you can change that to any name you want.

When you are done click on Save button.

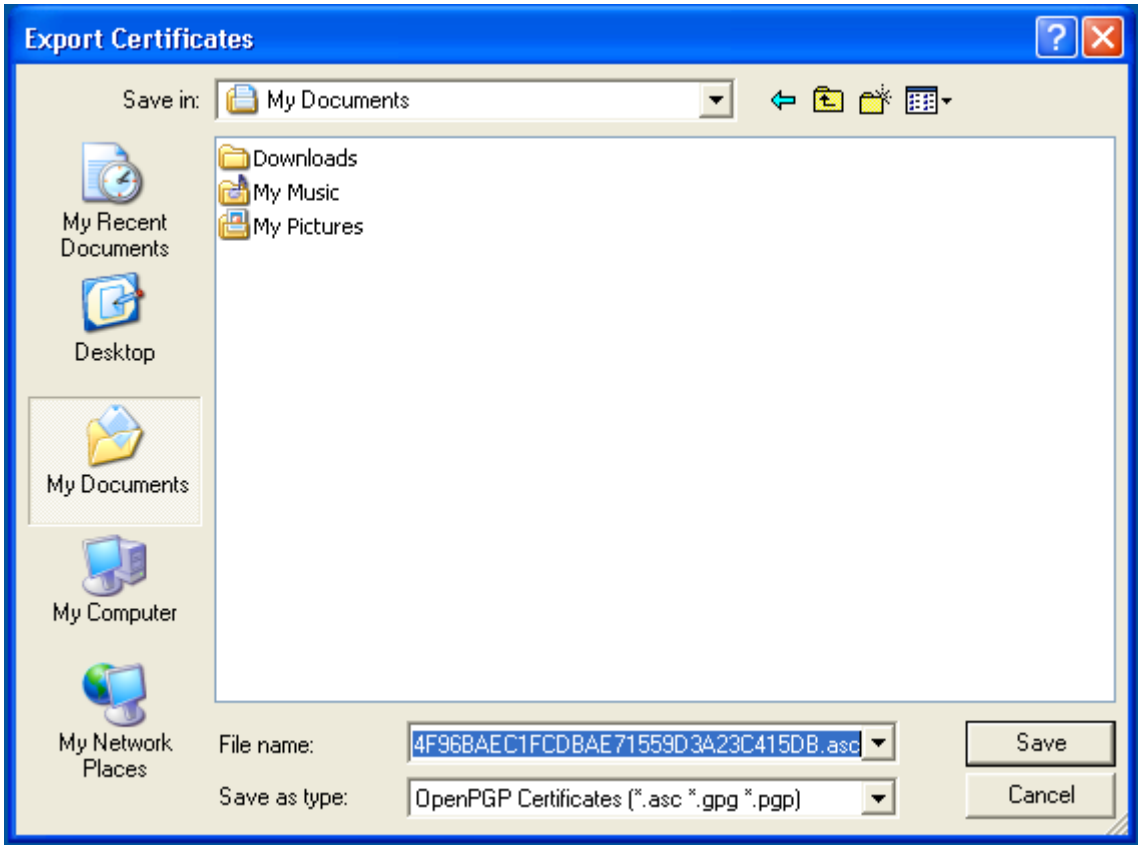


Figure 86: Choosing export directory

That's it, now your public key has already been exported to the directory you chose. This operation does not show a confirmation message.

15.2. Exporting your private key

The private key is your unique, personal and untransferable key, so you must never give it or send it to anyone. Ideally you should only export your private key to make a backup copy or to use it in another computer that you own.

1 – Select the certificate

Right-click on your key and select Export Secret Keys.

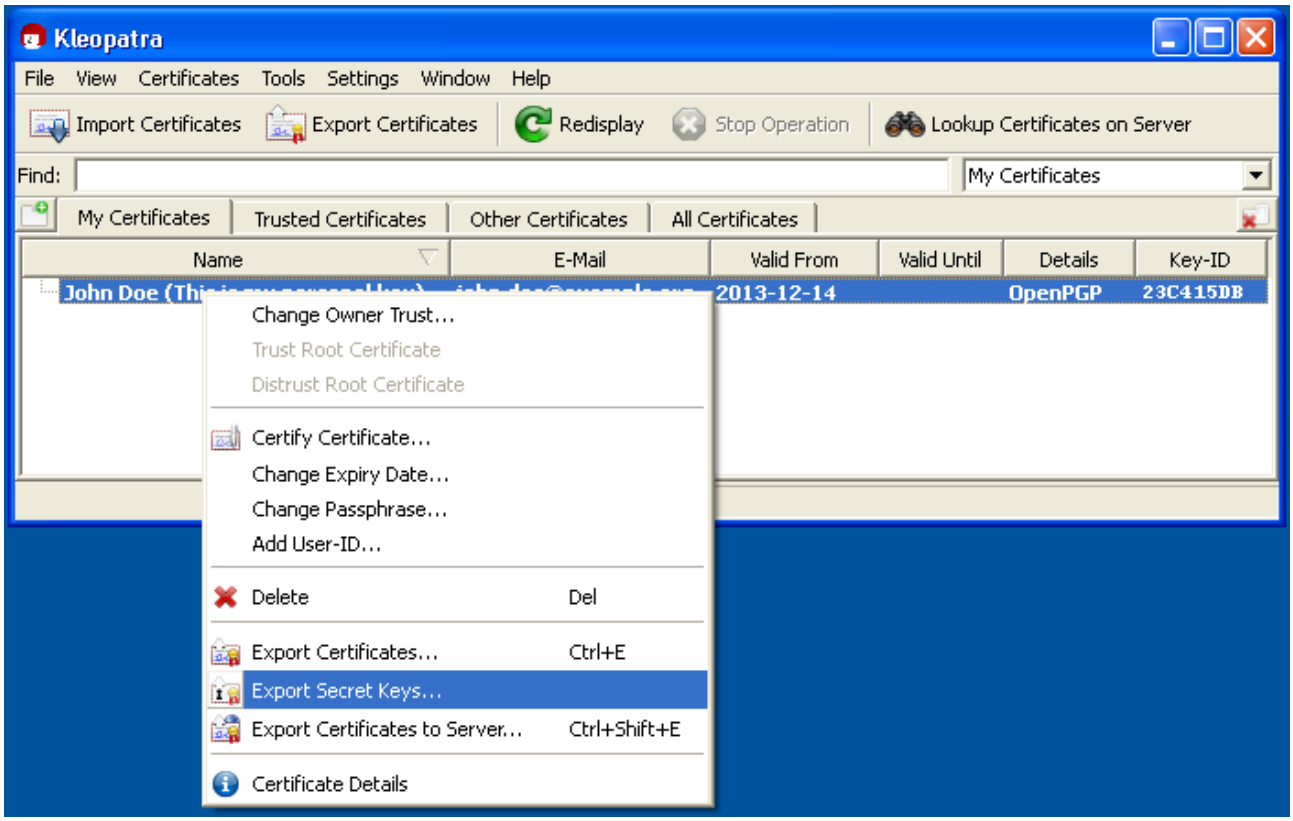


Figure 87: Exporting private key

2 - Choose the output location

In the window below click on the button with the symbol of a folder.

Figure 88: Output window

Choose a place to save your certificate and choose a name for it if you wish.

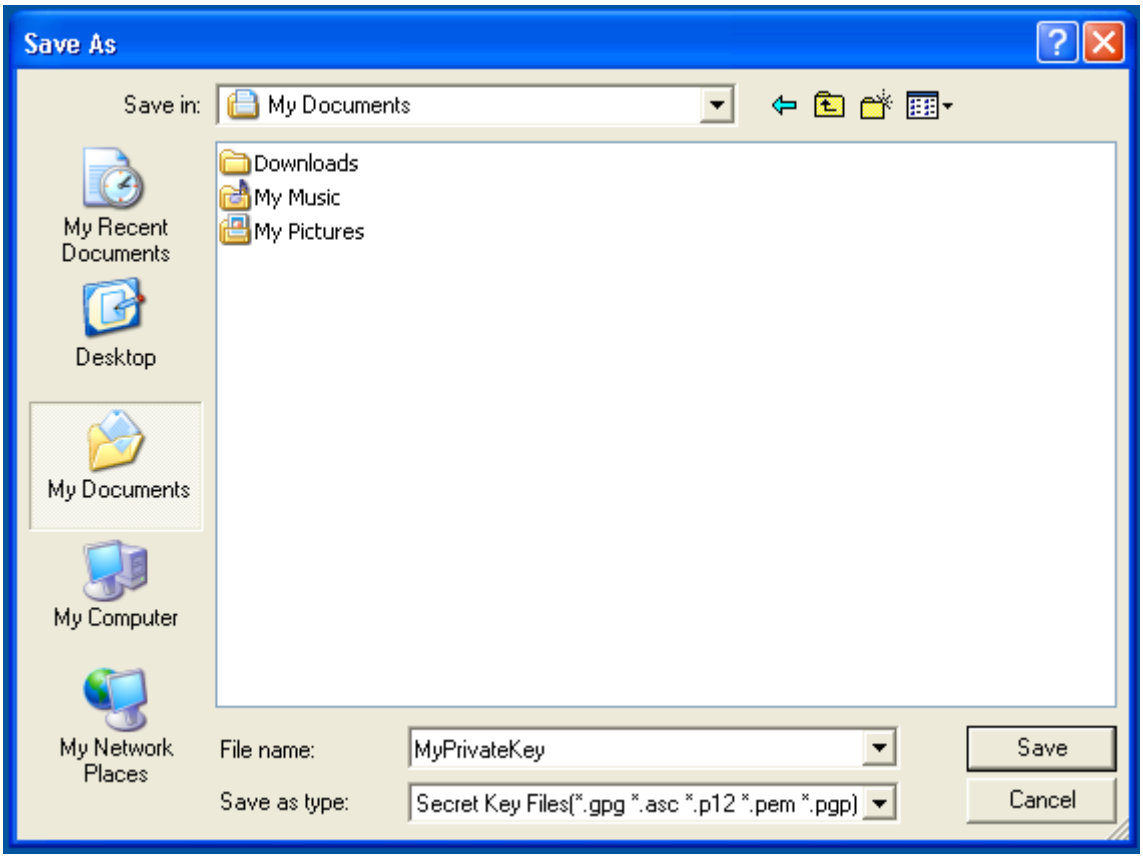


Figure 89: Choose output location

3 - Confirmation

The window below shows the output location you have chosen to export your certificate. Since you are exporting a certificate, it is recommended that you check the ASCII Armor checkbox.

When you are done click on OK button.

Figure 90: Output window

Figure 91: Confirmation message

That's it, now your certificate is exported to the directory you have chosen.

15.3 – Importing Certificates

There are two different ways to import certificates: through the Kleopatra main interface or right-clicking on the file directly.

15.3.1 – Importing through Kleopatra

Open Kleopatra and click on File → Import Certificates, or press Ctrl I, or click on the Import Certificates button on the toolbar.

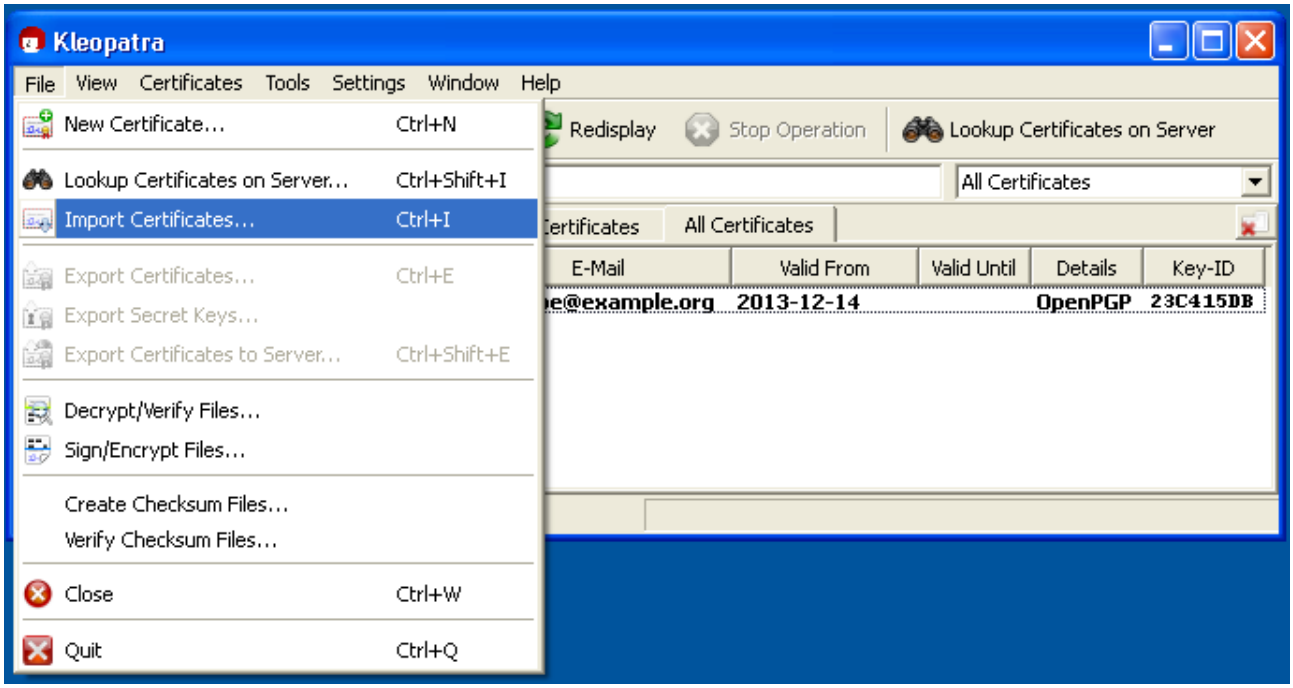


Figure 92: Importing certificates

Choose the place where the certificate is located, select it and click on Open button.

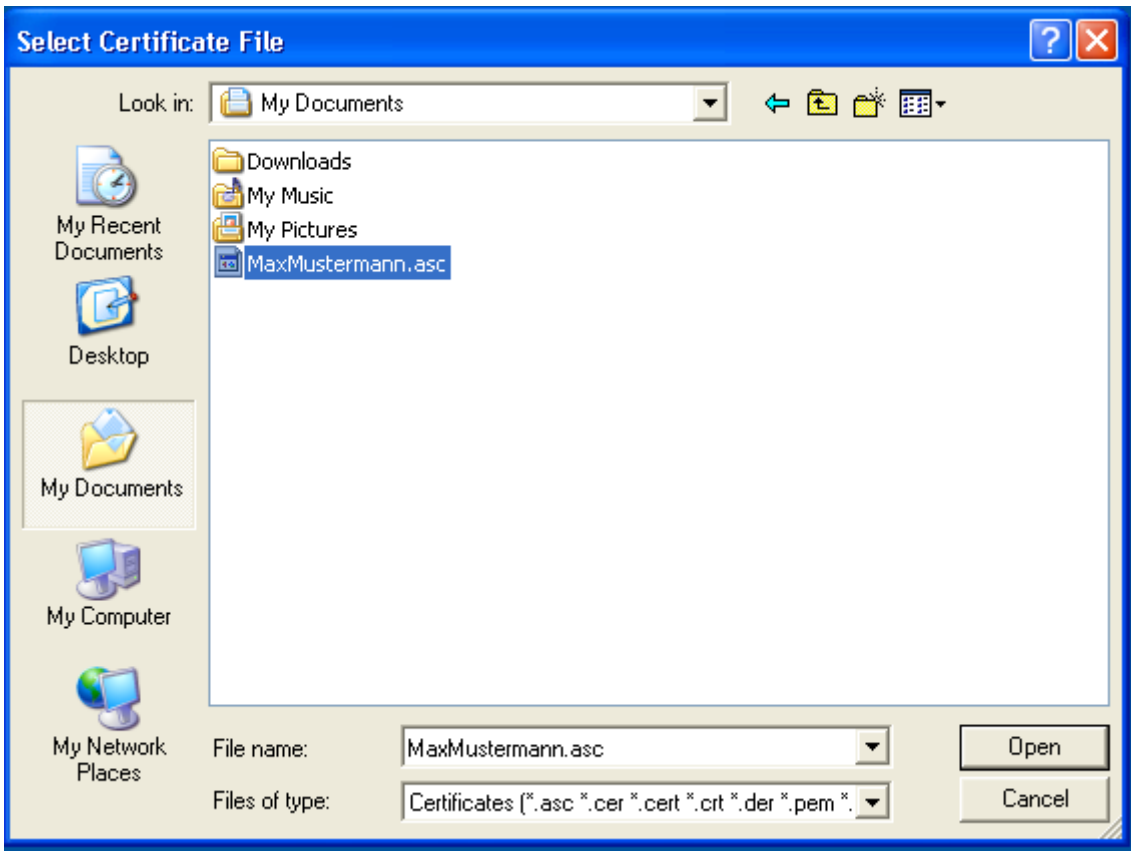


Figure 93: Open the certificate

A confirmation message will show up informing you that the operation was successful. You can now see the certificate in your list.

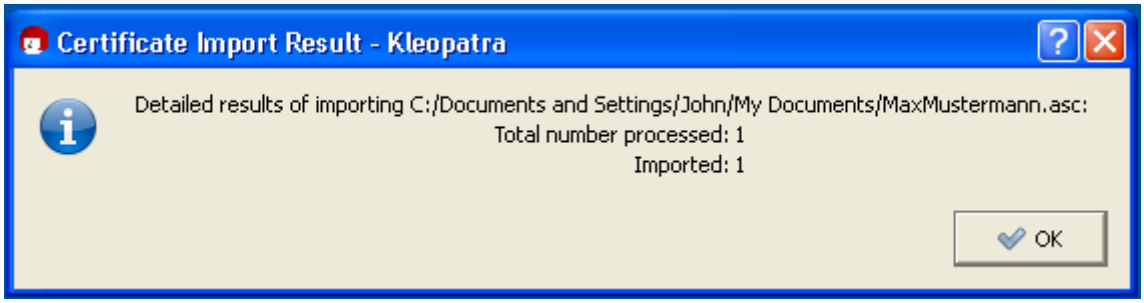


Figure 94: Confirmation message

15.3.2 – Importing through the file manager

Right-click on the file and select More GpgEX options → Import keys.

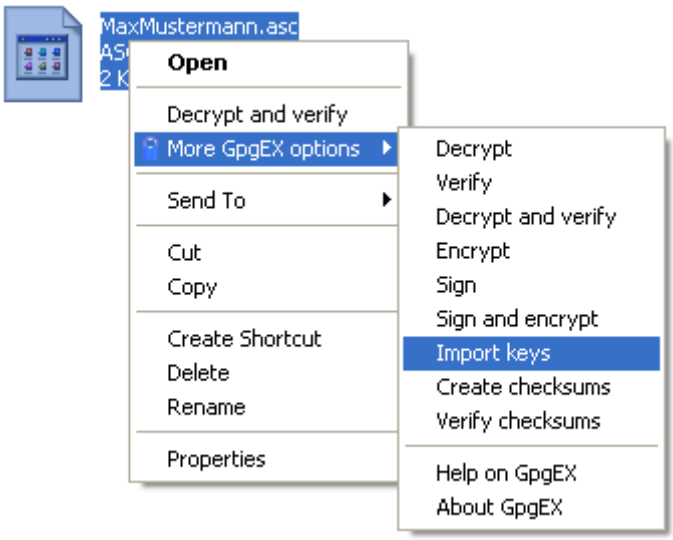


Figure 95: Choosing file

A confirmation message will show up informing you that the operation was successful. You can now see the certificate in your list.

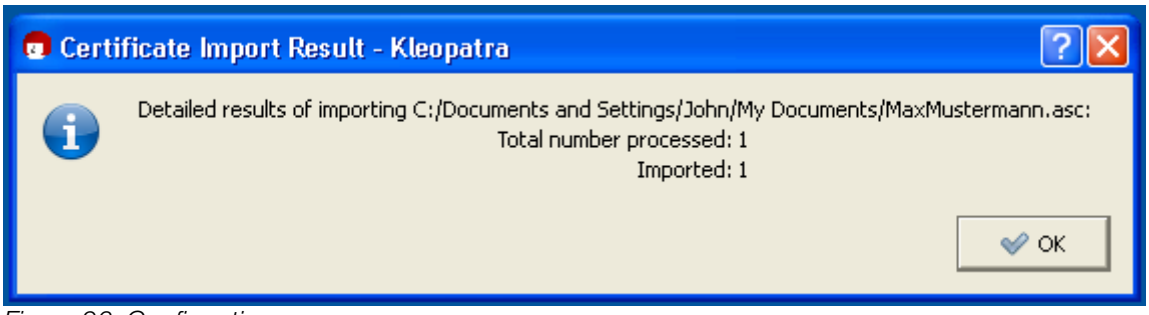


Figure 96: Confirmation message

CHAPTER 16

Importing and Exporting Certificates

To export a certificate means to generate a copy of a certificate that is in your keyring to a file where it could then be moved or sent to others. To import a certificate means to insert a certificate from a file into your keyring where it can then be used.

To sign, verify, encrypt, decrypt and certify, you often need to import others' certificates, and export yours to them.

1. Exporting your public key

The public key is the key you make available for others to communicate with you. It is only through this key that others can contact you privately.

To export your public key open Seahorse, select your key and click on menu File → Export.

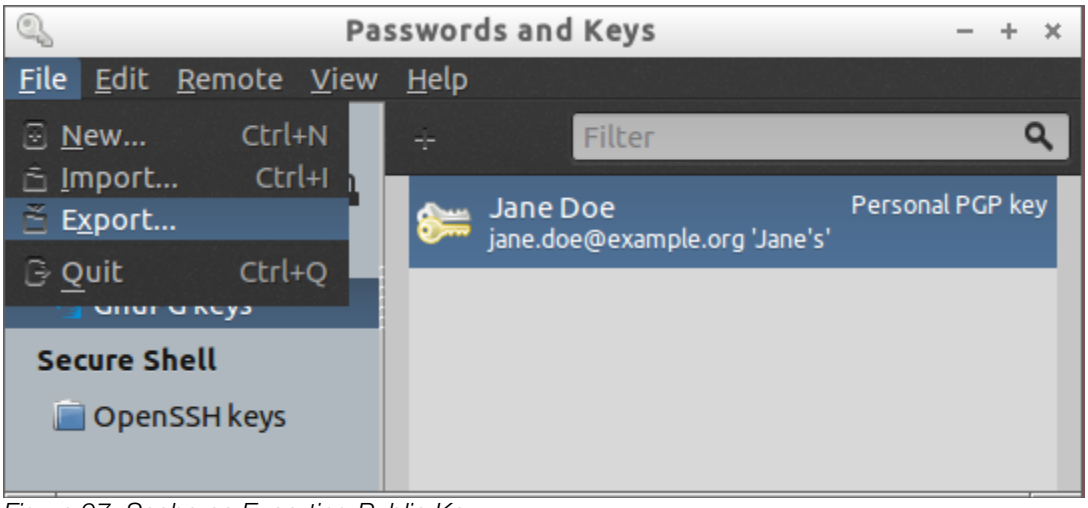


Figure 97: Seahorse Exporting Public Key

Now choose the place where you want to save it and choose a name for the file if you wish. You can also select Armored PGP keys in the lower right side if you want your key to be exported as encoded text.

When you are finished click on Export button.

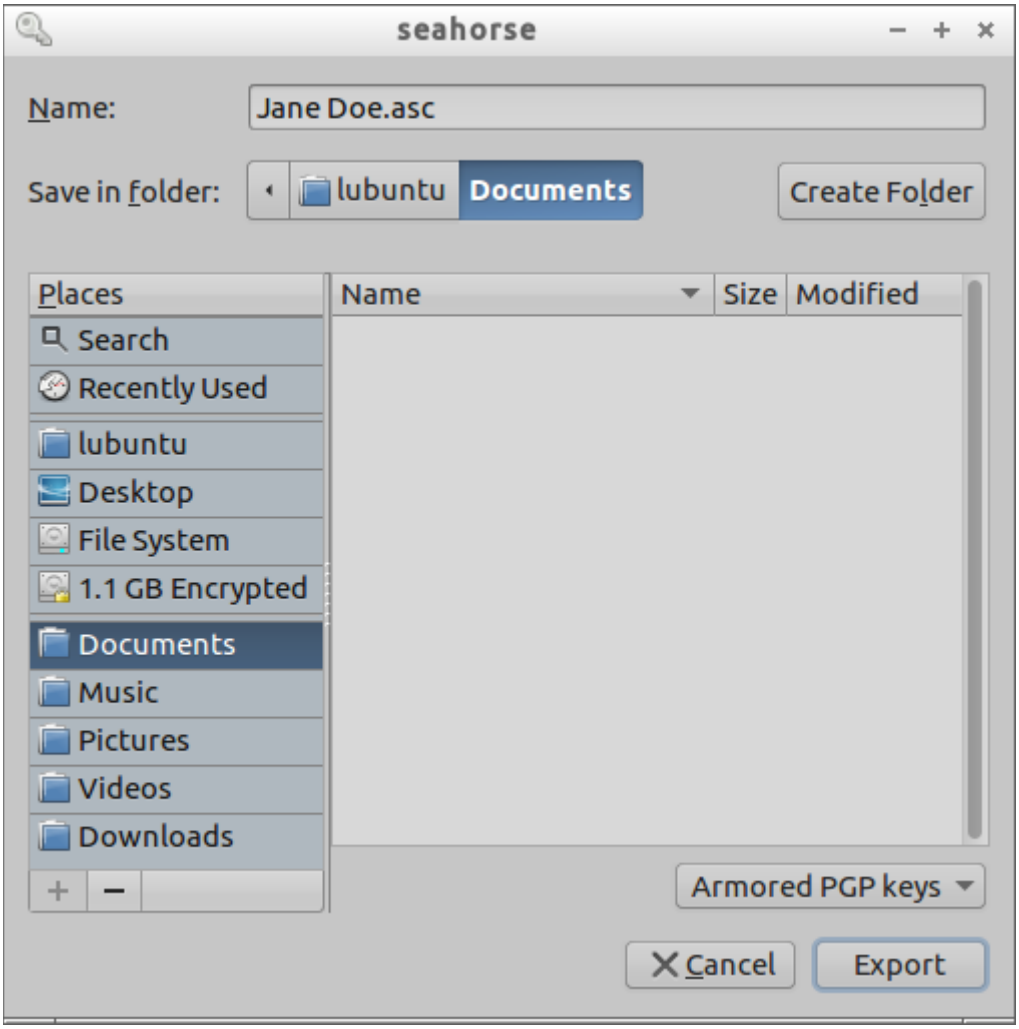


Figure 98: Seahorse Saving Exported Public Key

That's it, now your public key is exported to the directory you chose. This operation does not show confirmation message.

2. Exporting your private key

The private key is your unique, personal and untransferable key, so you must never give it or send it to anyone. Ideally you should only export your private key to make a backup copy or to use it in another computer that you own.

Right-click on your key and select Properties.

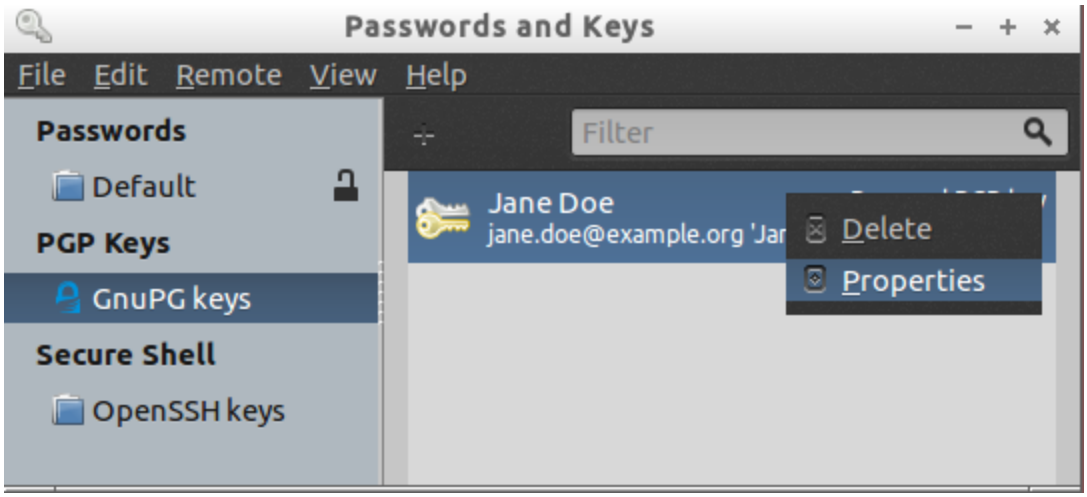


Figure 99: Access your key's properties

In the window that opens it shows a summary of your key. Click on the last tab Details.

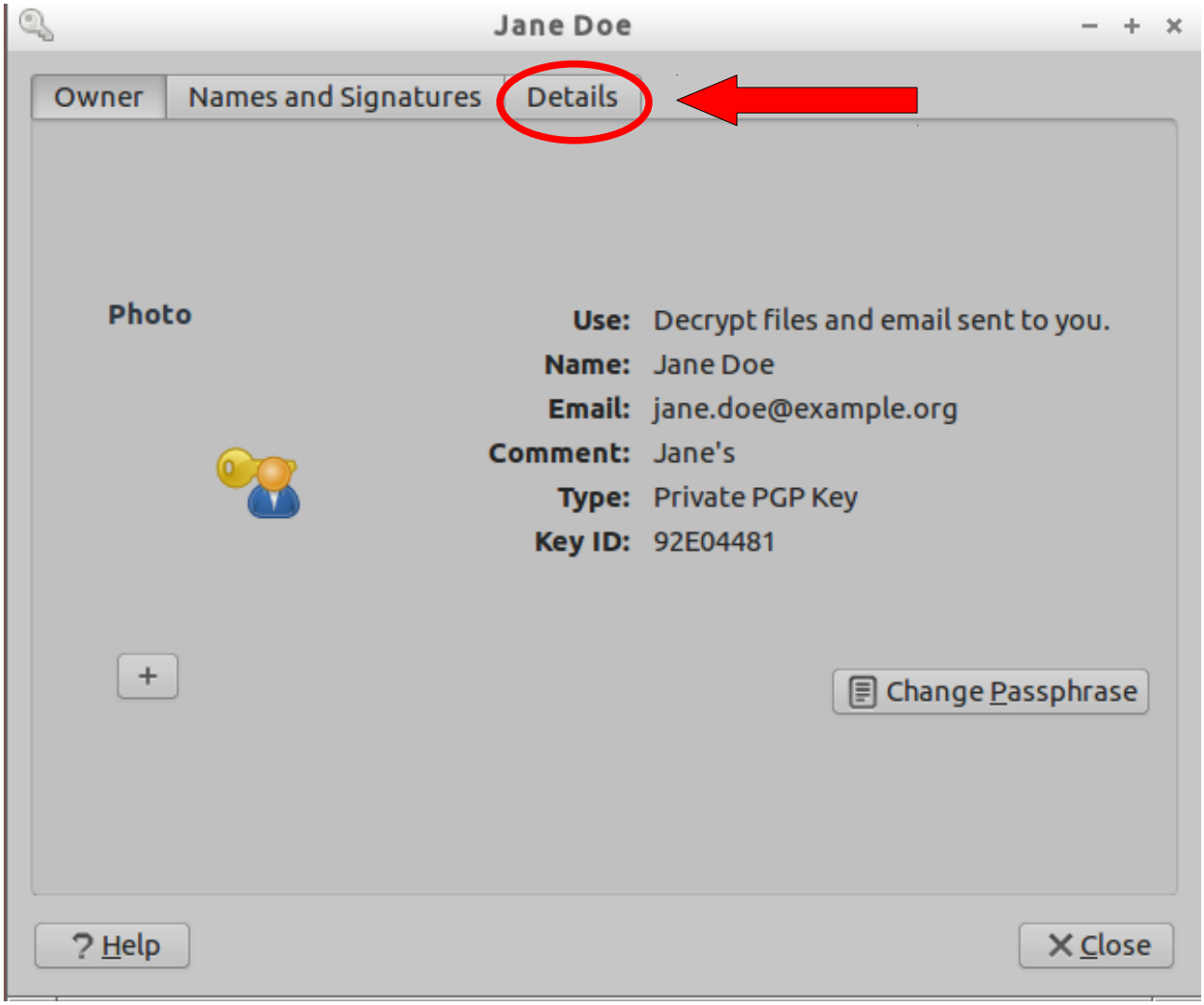


Figure 100: Summary of your key

Now you can see the Details tab showing advanced details of your key. Click on the Export button to export your private key.

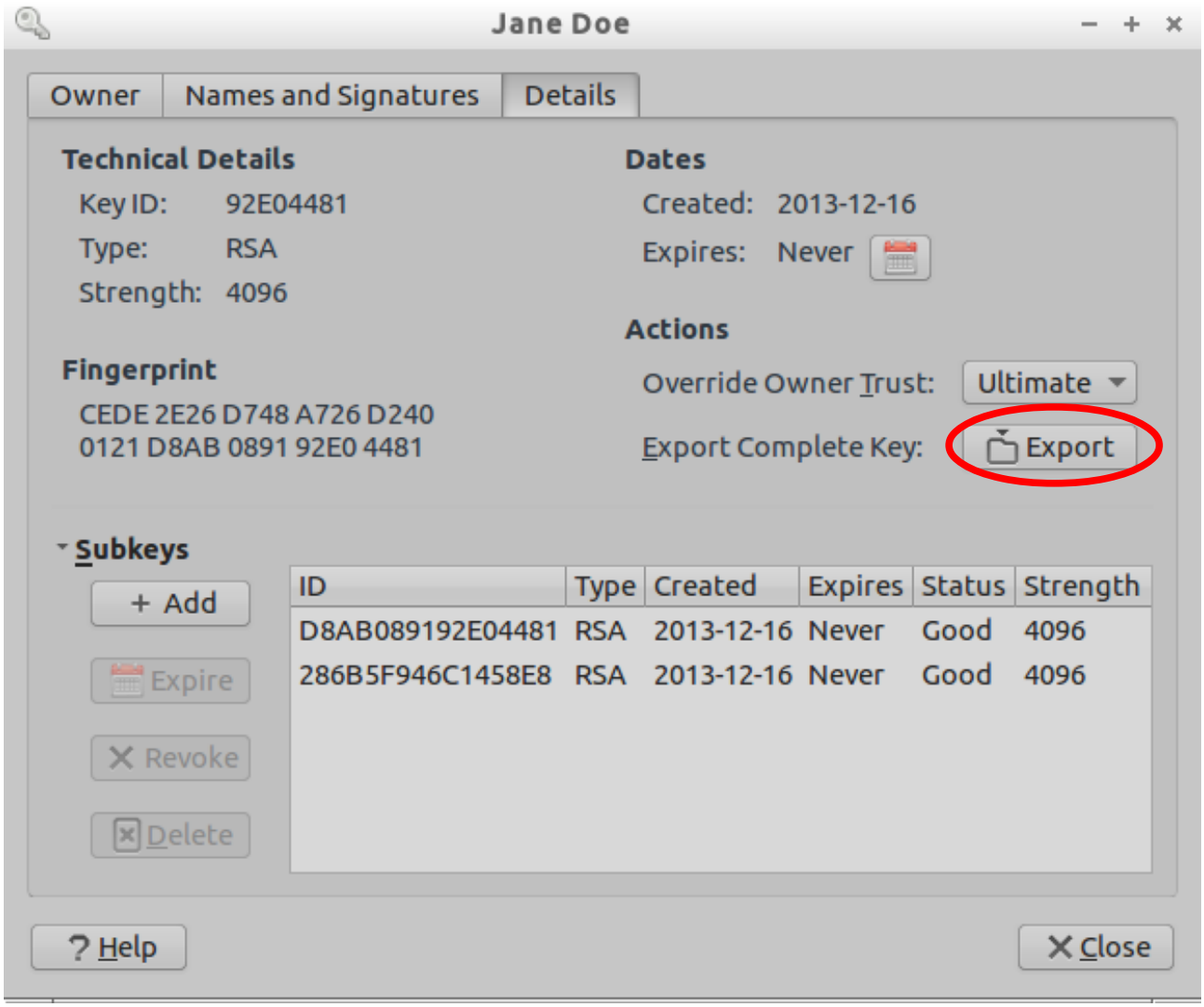


Figure 101: Advanced details of your key

Now choose the place where you want to save it and choose a name for the file if you wish. Private keys can only be exported as Armored PGP (encoded text).

When you are finished click on Export button.

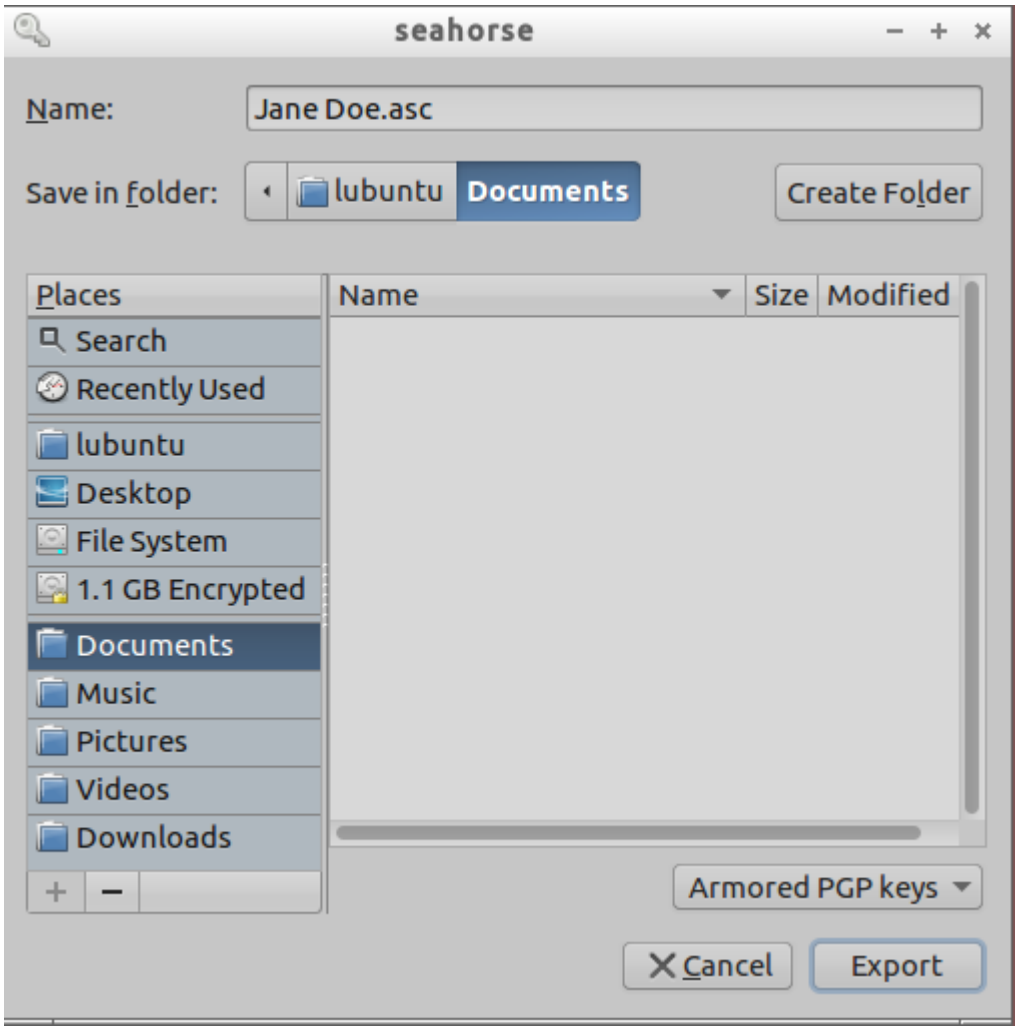


Figure 102: Choosing saving location

That's it, now your public key is exported to the directory you chose.

3. Importing keys and certificates

Open Seahorse, click on menu File → Import, or press Ctrl I.

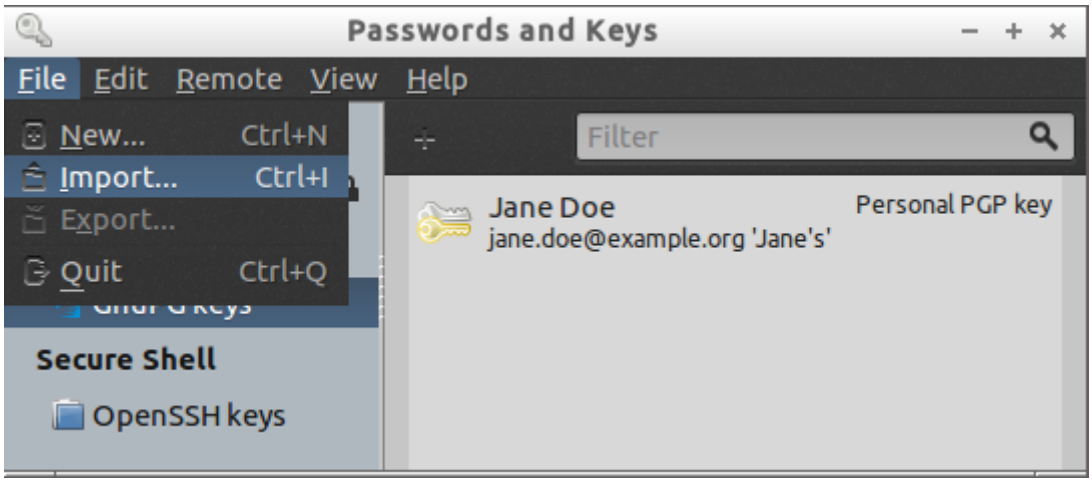


Figure 103: Importing certificates

Choose the certificate you want to import and click on Open button.

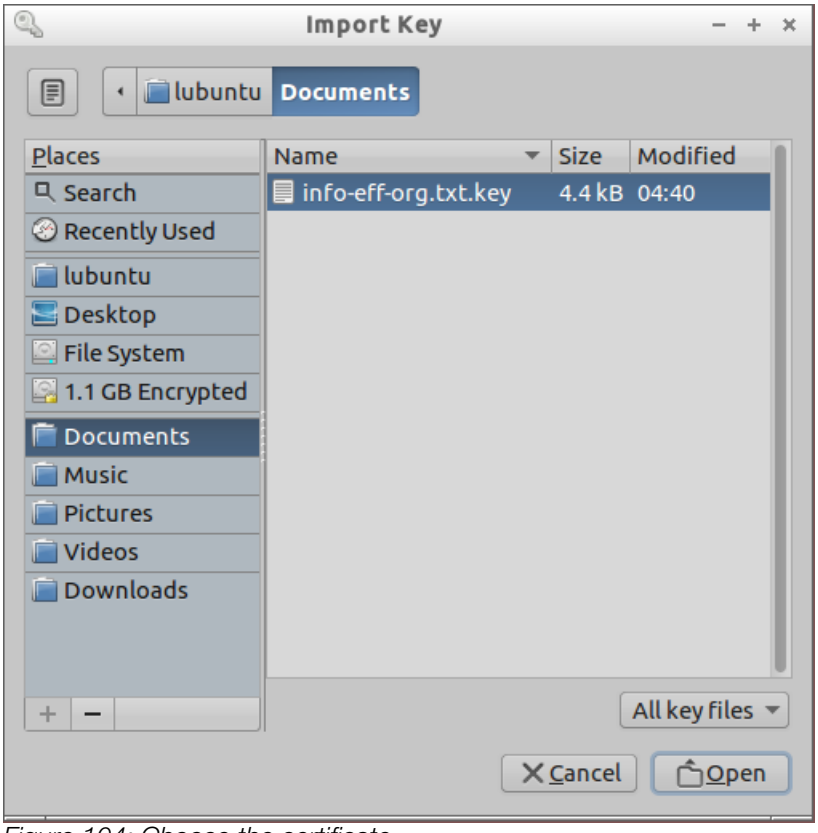


Figure 104: Choose the certificate

Seahorse will show a message informing you that the certificate has not been verified yet. You can click in Details if you want to check additional information about the certificate.

To import the certificate just click on Import button.

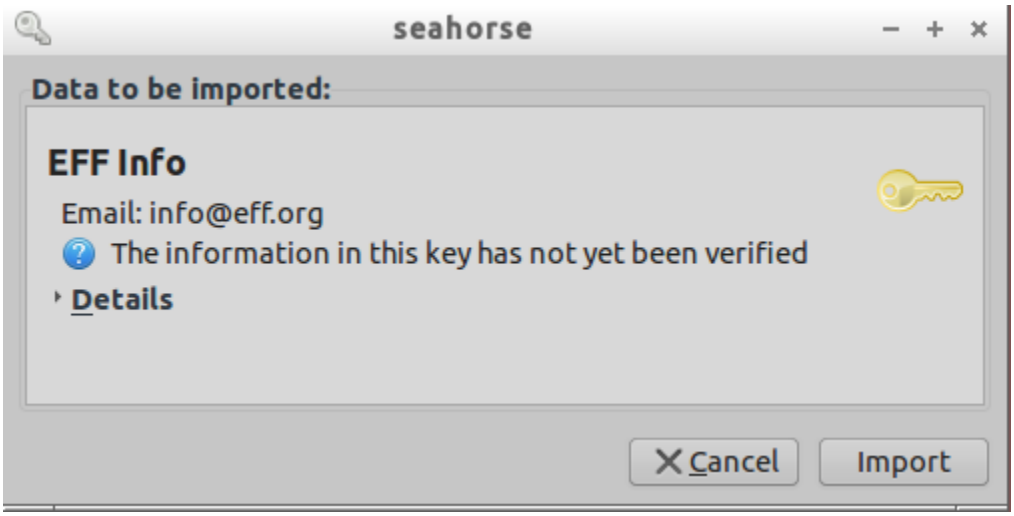


Figure 105: Certificate to be imported

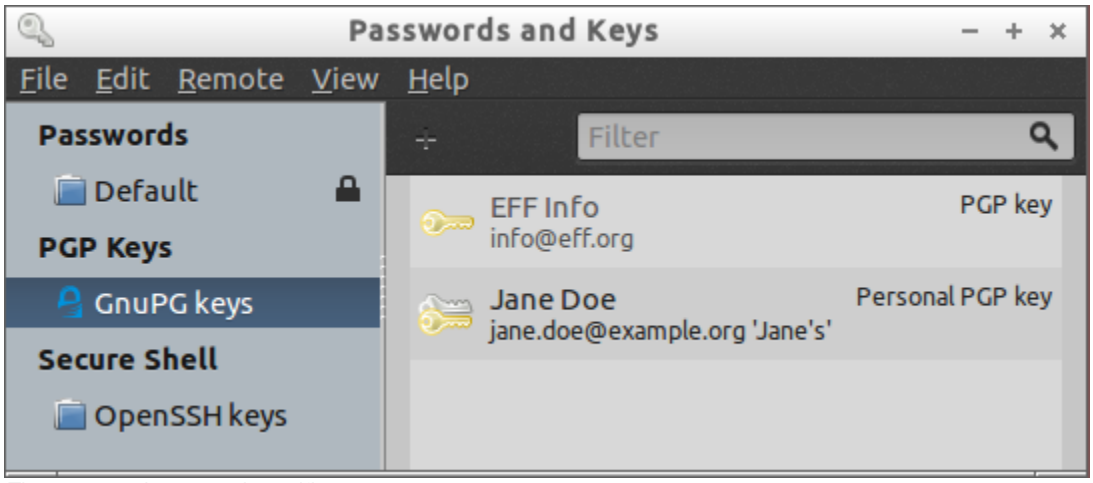


Figure 106: Imported certificate

That's it, now your certificate is imported and is already located in your keyring.

CHAPTER 17
Key servers

Key servers are computers that store public keys and serve them to users, allowing them to upload, retrieve and revoke keys. There are many key servers available and the basic idea is that they all synchronize their databases so they always have the same keys and they are always updated, although each key server is free to set its own rules regarding any of the operations mentioned before.

17.1 – Why use key servers?

The biggest advantage of using a key server is that if someone wants your key to contact you securely they don't have to request it to you, instead they can simply search for it in a key server and download your key. This is very useful if you own a blog or a website, or if you often expect strangers to contact you.

However once a key is uploaded to a key server it is publicly available and cannot be deleted, it will supposedly remain there forever. If you don't want to use that key anymore all your can do is to revoke it, but even then it will still remain there marked as revoked.

Also keep in mind that anyone could upload your key without asking your permission. This is because key servers are highly unregulated, so anyone can upload anything without verification or validation. Consequently there is a a large amount of fake keys stored on them.

Key servers used to be more popular in the beginning of the 1990s, before the internet became commercial, because back at that time very few people had internet access and an updated place where other people's keys could be easily found was very convenient. Today most GnuPG users have websites, blogs and/or social networks and they can publish their keys in those platforms, which anyone can access directly.

Nonetheless, key servers are still very popular and most – if not all – OpenPGP implementations have support for key servers.

17.2 – Using Key Servers via Command Line

17.2.1 – Searching and Importing keys

To import a key from a key server you have to search for it using the command below:

Syntax:

```
$ gpg2 --search-keys KEY_ID
```

Below we will search for a key named 'Bill Gates' and then import it into our keyring.

```
# Searching for a key named 'Bill Gates'
$ gpg2 --search-keys 'Bill Gates'
gpg: searching for "Bill Gates" from hkp server keys.gnupg.net
(1)   bill gates (claves) <billgates@gmail.com>
     2048 bit DSA key E2DDE443, created: 2013-10-17
(2)   Bill Gates <bill@gates.net>
     2048 bit RSA key DA6782E0, created: 2013-08-28 (revoked)
(3)   Bill Gates <2648778@gmail.com>
     2048 bit RSA key AC260CFB, created: 2013-08-20
(4)   Bill Gates <billg@microsoft.com>
     2048 bit RSA key D42DA1AA, created: 2013-07-12
(5)   Bill Gates <dedalus@mail.is>
     2048 bit RSA key 500FF66A, created: 2013-07-11
(6)   yoyo50 <bill@gates>
     2048 bit RSA key 20F8D5EF, created: 2013-06-01
(7)   Bill Gates (stupid) <billgates@microsoft.com>
     2048 bit RSA key 0BAB5FA5, created: 2011-12-21
(8)   Bill Gates <jerry.sych@gmail.com>
     2048 bit RSA key 61CDB1EB, created: 2011-12-04
(9)   bill gates <jerry.sych@gmail.com>
     2048 bit RSA key 1B385AE3, created: 2011-12-04
(10)  bill gates <jerry.sych@gmail.com>
     2048 bit RSA key A88D8F59, created: 2011-12-03
(11)  Bill Gates <fila.andr@gmail.com>
     2048 bit RSA key 9EA412C7, created: 2011-03-19
Keys 1-11 of 91 for "Bill Gates".  Enter number(s), N)ext, or Q)uit > 4
```

91 keys were found. You can enter the corresponding number of the key you are looking for and press ↵ or press N and see the next 11 results, and so on. You can choose more than one key if you want, just separate them with a comma. We will choose the 4th key.

```
gpg: requesting key D42DA1AA from hkp server keys.gnupg.net
gpg: key D42DA1AA: public key "Bill Gates <billg@microsoft.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Now the key is already imported and you can check it with the listing command.

17.2.2 – Sending your key to a key server

To publish your key you need to choose the key server you are going to use and use one of the commands below:

Syntax:

```
# Sending a key without specifying the key server
$ gpg2 --send-keys KEY_ID

# Sending a key specifying the key server
$ gpg2 --keyserver KEYSERVER_NAME --send-keys KEY_ID
```

Which command you choose depends on which key server you would like to use and how GnuPG configurations files are configured in your computer.

We will choose the second command because it allows us to specify they key server we want to use, which in our example is the same used by GnuPG by default:

```
$ gpg2 --keyserver keys.gnupg.net --send-keys A1B2C3D4
gpg: sending key A1B2C3D4 to hkp server keys.gnupg.net
```

That's it, your key has been sent to the key server and now it is available to the public.

You can check your key now following the instructions shown in section 17.2.1.

17.3 – Using Key Servers via Web Interface

17.3.2 – Searching and importing keys

Enter the name of the key you are looking for in the field indicated in the image below and press Search button. You can also customize the following options:

- **Index:** is cleaner and shows less information of the keys. You have to click on the keys links to see additional information.
- **Verbose index:** shows additional information of the keys in the main window.
- **Show OpenPGP "fingerprints" for keys:** shows the keys fingerprints.
- **Only return exact matches:** tries to return exact matches of the string(s) entered.

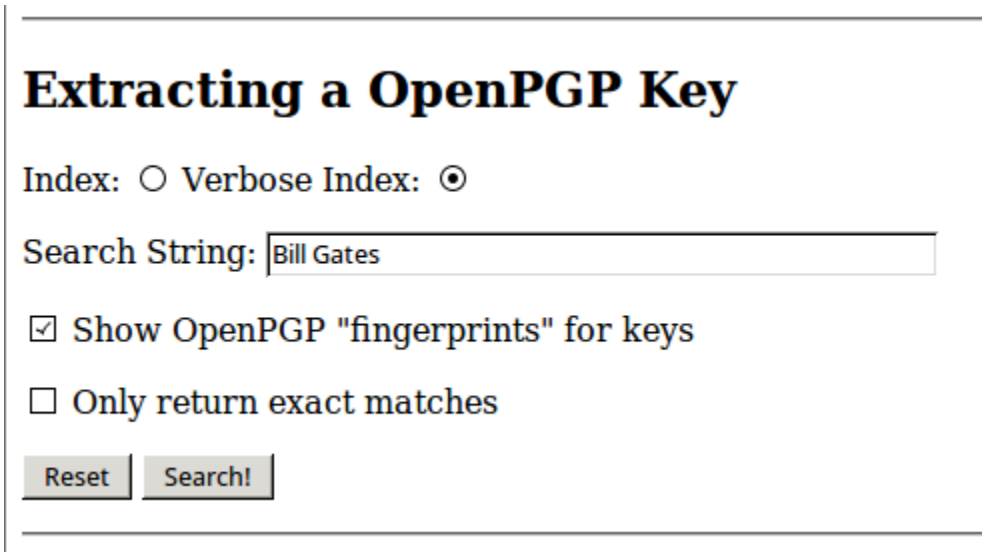


Figure 107: Searching key string

The key server will show a list of keys containing the strings you entered. Find the correct key you want to import. As we did in our previous example, we will choose the 4th key again, which is shown in the image below. Click on the link indicated to see the key.

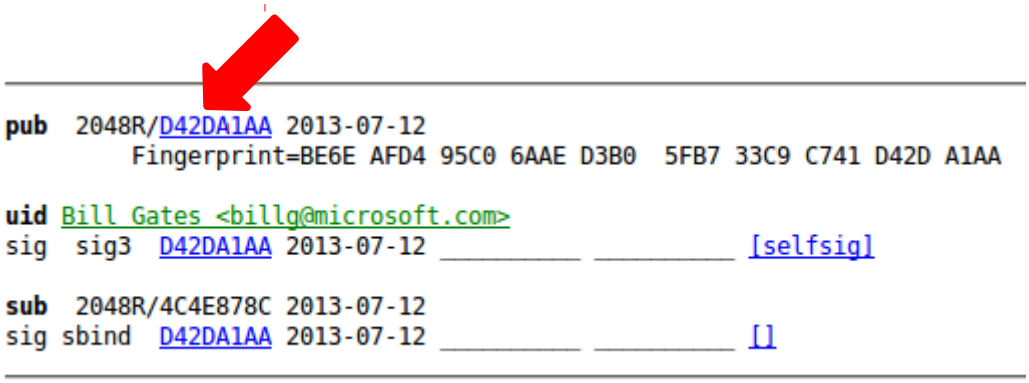


Figure 108: Search key result

Now you will see the key on screen. Select it, including the beginning and end tags, copy it and save it in a text file. You can use any extension you want, preferably .asc or .key.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: SKS 1.1.4
Comment: Hostname: keyserver.ubuntu.com

mQENBFHgOD4BCADJDolugN5Cmri6oKG9IQr1Xv3zArySCtXPa82zdh1m6qYdEm6m8vvbgnDv
14PoqRTLiy5WqxJSWrWRJN/P6O3qJ7gSjTu2Psd8pL6+yww7q2gWfNOPzbyE6e6mU6Q3phe3
TiRJaZFnHsJd6ikd/1KamcAM/gsKDIg/IvrZW+4AU2DXPg2ZtibxdUmGtIIysJM6dxrccq0x
7UVGqam08wvNEUfxi7O4vHaGG+5c/PyM92ZpTYwUZaNExJZjRicc1/tIGucdn215g/Pq1ROd
Nv9+PjDgYVOsaz9kva7uhQxmKqgMGEnJqNr8hcfkmfGlTO+JFEowAr9F0+EUwqmICAvhABEB
AAG0IEJpbGwgR2F0ZXMgPGJpbGxnQG1pY3Jvc29mdC5jb20+iQE5BBMBAgAjBQJR4Dg+AhsD
BwsJCAcDAgEGFQgCCQoLBBYCAwECHgECF4AACgkQM8nHQdQtoarJ7Af/T0MDpDCcYaGOGdyX
4pcfqBtpGBtTrJpadtdTq/EUE8Y96bbJizYt5zMWa0l90gbAcSXsD7RupDimadJelThdZbZ3
toIq0dXEzgj3tyXRAQX5mR7EmgK6zg0VBPDphk9xDH0NNzluyjCVaBvjgYi+KTdcI9jr+78m
/0LXzpDNeaNCA/6oOQ/n/6DJ5SYcx1mxeqd4waRGg3bez0u8vDuapzAmjAnaOTn6yWf89cwv
3ozswSrUnn67oFnGCaQvO2SMKYMcP8zSFUDgdM93xNpa3D+VYVYYU2cxSlBgGFbYtl4yNxQn
7MTYvbnjp2oEw+f3l65+Oq54Nunam56B5D+14LkBDQRR4Dg+AQgAu+6AXMJfHFx08Bj6QNVO
qxEiaDuzYl6E7RggcWfHW7No6kYGy0+yRwgu+8BKPsPc+VlZ1xweOBqLSm4NDQQINRiLbC7G
TnKLmZWqSID4IWvhzRX9kDV2KWi3jfPKD8cCBI6z4afsfLPBM6KFtW2TpfIwZ5jf54RbWozf
IxPGotzh8NqwKSfsUq+P+6YqtfqHIu8QEUhdSNB1Fx3vYQaY4qrx7eX24ZI57zSnOG0e+vvq
cX7ShfI1EVgtJr4e5q6X38bX02We4eE4jduduo4WypGXH4eAJUsM/TdJ6MLRpYXitI/TpG33
e8NP77nExZFSLdpT0+/xtNEDBOlYNQwt6wARAQABiQEfBBgBAgAJBQJR4Dg+AhsMAAoJEDPJ
x0HULaGq6GYH/1IZ8fx5nuoWR/TQQTl/L+OMpS350ZZ0AdWn7ScihpRLPoFCzYtDNv7yOwSc
Zmj/kyokVpYWb5KgaIDRFw8DJzowBSzG/q8CkVMvZoKVo8TmUh5RCix/LD6FxEk93MfThme1
ofHYEbcMPhrPcATFOpE7aSC2vSI6HD8/+Gjk63kfUr71WYc7jFEDmAbkirTTtO8IaqgZWTuP
LAS6/S3tBBv6lVE2QDYcCJ4jwxEgRZX9knXXw6tePWj8iQatLn7lACUavgDKN5iqCjqSGU0s
c2nGpEECquftcyWHfKKFxvv8A2IPWi0+aAqo+sNB+V0jG1iuuHKNHryffu0WUsVjzD0=
=2YWm
-----END PGP PUBLIC KEY BLOCK-----
```

Now you have a copy of the key. All you have to do is to import the file to your keyring, check out chapters 12, 15 and 16 for more information.

17.3.1 – Sending your key to a key server

First you have to access the website of the key server you want to use. Below are listed some of the most popular key servers that can be accessed via web interface:

- keys.gnupg.net
- keyserver.ubuntu.com
- pgp.mit.edu

1 – Access the website

Access the website of the key server you have chosen. We are using keyserver.ubuntu.com.

2 – Paste your key on the field

Scroll down and paste your key on the field 'Submitting a new OpenPGP Key', as shown below. You must have already exported your public key to do it. For more information on how to do it see chapters 12, 15 and 16.

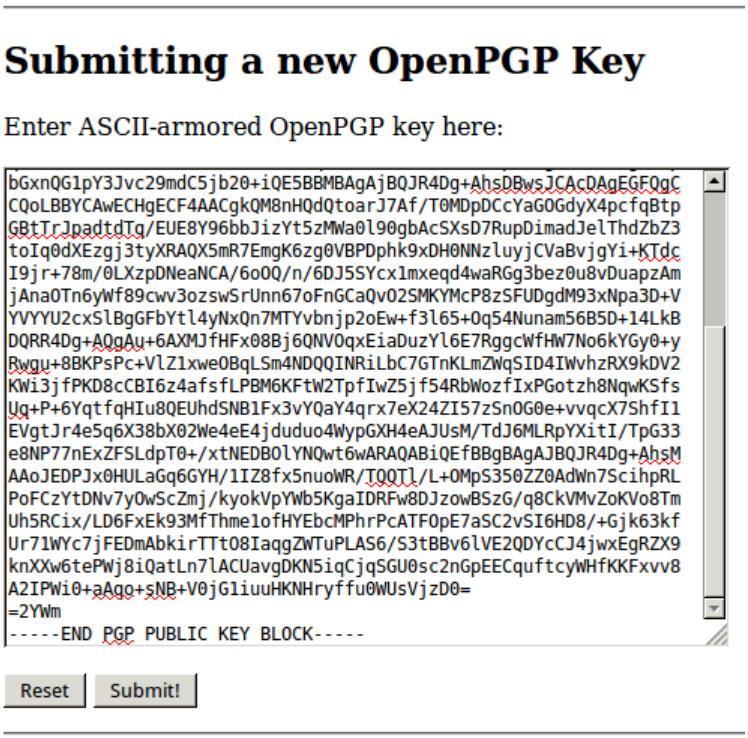


Figure 109: Sending public key

3 – Send your key

Click on Submit button. You will see the following confirmation message:

Key block added to key server database. New public keys added:
1 key(s) added successfully.

Now your key has been successfully published.

PART 4

FINAL CONSIDERATIONS

In this part you will learn:

- *List of Commands*
- *Bringing more People to GnuPG*
- *Conclusion*

CHAPTER 18
Commands Reference List

Here you will find a reference list with the most common GnuPG commands. This is not a complete list, but it contains all commands used throughout this manual, including some variations not previously used, such as shortening and combining.

Some commands were broken into two lines to facilitate comprehension.

Encryption

Symmetric Encryption

Encrypting in binary format (GnuPG will ask you to enter a password):

```
$ gpg2 --symmetric file.txt
$ gpg2 -c file.txt
```

Encrypting in ASCII-armored format:

```
$ gpg2 --symmetric --armor file.txt
$ gpg2 -c -a file.txt
$ gpg2 -ca file.txt
```

Asymmetric Encryption

Encrypting in binary format without recipient (GnuPG will ask you to provide one):

```
$ gpg2 --encrypt file.txt
$ gpg2 -e file.txt
```

Encrypting in binary format with recipient:

```
$ gpg2 --recipient Mary
       --encrypt file.txt
$ gpg2 -r Mary -e file.txt
```

Encrypting in ASCII-armored format with recipient:

```
$ gpg2 --recipient Mary --encrypt
       --armor file.txt
$ gpg2 -r Mary -e -a file.txt
$ gpg2 -r Mary -ea file.txt
```

Decryption

Decrypting without output file (GnuPG will show its content on the screen):

```
$ gpg2 --decrypt file.txt.gpg2
$ gpg2 -d file.txt.gpg2
```

Decrypting with output file:

```
$ gpg2 --output list.txt
       --decrypt file.txt.gpg2
$ gpg2 -o list.txt -d file.txt.gpg2
```

Signing

Signing in binary format:

```
$ gpg2 --sign file.txt
$ gpg2 -s file.txt
```

Signing in ASCII-armored format:

```
$ gpg2 --sign --armor file.txt
$ gpg2 -s -a file.txt
$ gpg2 -sa file.txt
```

Clearsigning:

```
$ gpg2 --clearsign file.txt
```

Detached signature in binary format:

```
$ gpg2 --detach-sign file.zip
```

Detached signature in ASCII-armored format:

```
$ gpg2 --armor --detach-sign file.zip
```

NOTE: To extract files from binary or clear signed files check Decryption section.

Verifying

Verifying Signed and Clearsigned files:

```
$ gpg2 --decrypt file.gpg2
```

Verifying Detached Signatures:

```
$ gpg2 --verify file.sig file.txt
```

Key Management

Generate a key pair

```
$ gpg2 --gen-key
```

Import keys

```
$ gpg2 --import key.asc
```

Export Keys

Export public keys in binary format:

```
$ gpg2 --export mykey@example.org
```

Export public keys in ASCII-armored format:

```
$ gpg2 --export
       --armor mykey@example.org
```

Export private keys in ASCII-armored format:

```
$ gpg2 --armor --output mykey.asc
       --export-secret-keys john
```

Export whole keyring into a single file:

```
$ gpg2 --export --armor > keyring.asc
$ gpg2 --export-secret-keys
       --armor >> keyring.asc
```

Export whole keyring into two files:

```
$ gpg2 --export --armor
       --output pub_keyring.asc
$ gpg2 --export-secret-keys --armor
       --output sec_keyring.asc
```

Listing keys

Listing public keys:

```
$ gpg2 --list-keys
$ gpg2 -k
```

Listing private keys:

```
$ gpg2 --list-secret-keys
$ gpg2 -K
```

Showing fingerprint

```
$ gpg2 --fingerprint
```

NOTE: You can combine the `--fingerprint` option with the listing commands

Deleting keys

Deleting public keys:

```
$ gpg2 --delete-key mark
```

Deleting private keys:

```
$ gpg2 --delete-secret-and-public-key
       mark
```

Revoking keys:

```
$ gpg2 --output revoke.asc
       --gen-revoke mykey
```

CHAPTER 19

Bringing more people to GnuPG

There's no point in using cryptography to communicate with others in a secure way if they don't use it and they don't know how to use it. The reality is that very few people will look for and start using cryptography by themselves. They are more likely to do it if they become aware of the benefits of this technology and others they know already use it.

To address this problem we made a short text message so you can send it to other people and invite them to know more about e-mail cryptography.

Hello **CONTACT NAME**,

what would you think if all e-mails you've ever sent and received, including the ones you deleted years ago, were made public so the entire world could read them? Your personal messages, documents, attachments and so on, all available for everybody to see?

You think this is impossible and it could never happen to you, don't you? Well, keep reading because I have bad news for you.

Are you aware that every time you use e-mail you have absolutely no security and privacy?

Are you aware that all your messages, sent and received, are being stored – possibly forever – by the same companies that provide you that nice free e-mail account, such as Gmail, Hotmail, Yahoo, and others?

Do you realize that once those informations are stored they can be leaked at any time, and they WILL PROBABLY BE LEAKED?

Yes, unfortunately this is all true. And do you know why they do it? For two reasons: to sell you garbage in form of advertisement and to spy on you.

Do you like garbage? Of course not!

Do you like others spying your life? Of course not!

So what do you do then if you have no choice and you have to use e-mail?

Fortunately there is a solution, it is called GNU Privacy Guard, or GnuPG. GnuPG is a com-

puter program that allows you to send and receive e-mails with total privacy and security by encoding your messages so no one can read them, except the persons you communicate with. And the best part is that it is 100% free and open source, so you don't have to pay anything and it doesn't come with viruses or any other type of malware.

Are you interested? Check out this website https://www.goldenkeys.org and get a copy of the GnuPG guide that teaches you everything you need to know to use GnuPG and set it up in minutes.

Check it out now and get your security and privacy back!

Send this message to all your contacts because most people are unaware of the risks they face by not using cryptography in their communications, and they will not take any steps to use it until someone they know presents it to them.

Conclusion

By now you are already aware of how cryptography works and the importance of using this technology. With GnuPG it is possible for anyone to reach high levels of security and privacy relatively easy and practically for free.

Although more and more people are using cryptography these days, the reality is that for most of them it is still a complicated issue and they don't understand the risks they face by not using it. Also, most companies still don't use cryptography in their services, including services that work with important information such as e-mail.

We believe that the more people and companies use cryptography, the more they will be aware of the dangers of insecure communications, and the more cryptography will become a default requirement rather than a mere convenience.

In this guide we offered some basic advice on computer security so you can start using cryptography right now and secure your digital communications. This guide is not complete and it does not cover all the resources of GnuPG, but in terms of e-mail privacy you are now ahead of the majority of computer users, including many so called experts.

Feel free to send this guide to as many people as you want.

We hope you have liked this guide!

If you would like to make any comments please write to contact@goldenkeys.org.

Thank You!

The Golden Keys Team

https://www.goldenkeys.org