

ENIGMAIL

OPENPGP EMAIL SECURITY FOR MOZILLA APPLICATIONS

The Handbook

by Daniele Raffo
with Robert J. Hansen and Patrick Brunschwig

v 1.0.0 and earlier

1. TABLE OF CONTENTS

2. Introduction	5
3. Acknowledgements	8
4. The Enigmail team	9
5. Getting started	10
5.1. Installing GnuPG.....	10
5.1.1. Installing GnuPG on Microsoft Windows.....	10
5.1.2. Installing GnuPG on Macintosh OS X.....	10
5.1.3. Installing GnuPG on Linux / UNIX.....	11
5.2. Installing Thunderbird / SeaMonkey.....	11
5.3. Installing Enigmail.....	12
5.3.1. Installing Enigmail on Thunderbird.....	12
5.3.2. Installing Enigmail on SeaMonkey.....	12
5.3.3. Installing a locale for Enigmail.....	13
6. Quick start	14
6.1. The Setup Wizard.....	15
7. Key management	27
7.1. Importing an existing key pair.....	28
7.2. Generating your own key pair.....	29
7.2.1. Tell Enigmail which account to use.....	29
7.2.2. Choose a passphrase.....	30
7.2.3. Choose the time expiry of the key.....	30
7.2.4. Choose the key type and size.....	31
7.2.5. Generate the key.....	31
7.2.6. Generate the revocation certificate.....	31
7.3. Operations on your key pair.....	32
7.3.1. Examining the key properties.....	32
7.3.2. Specifying multiple user IDs.....	33
7.3.3. Adding a PhotoID (via GnuPG command line only).....	34
7.3.4. Changing the passphrase.....	35
7.3.5. Self-signing your key.....	35
7.3.6. Making a backup.....	35
7.4. Distributing your public key.....	36
7.4.1. Share your public key manually.....	36
7.4.2. Publish your public key on a keyserver	37
7.5. Revoking your key pair.....	37
7.6. Importing public keys.....	38
7.7. Validity of public keys.....	39
7.7.1. The Web of Trust.....	39
7.7.2. Trust levels.....	40
7.7.3. Criteria for key validity.....	41

8. Signature and encryption.....	42
8.1. Account settings.....	42
8.2. Signature and verification.....	45
8.2.1. Signing a message.....	45
8.2.2. Verifying a signature.....	47
8.2.3. Retrieving the key that signed the message.....	50
8.3. Encryption and decryption.....	53
8.3.1. Encrypting a message.....	53
8.3.2. Decrypting an encrypted message.....	55
8.4. Handling attachments.....	59
8.5. Notes.....	60
8.6. Per-recipient rules.....	61
8.6.1. Per-Recipient Rules Editor.....	61
8.6.2. Recipient Settings.....	62
8.6.3. Notes.....	63
8.6.4. XML format of per-recipient rules.....	64
9. Preferences.....	66
9.1. Setting the preferences.....	66
9.1.1. Basic.....	66
9.1.2. Sending.....	69
9.1.3. Key Selection.....	71
9.1.4. Advanced.....	72
9.1.5. Keyserver.....	75
9.1.6. Debugging.....	76
9.2. Manually editing the preferences.....	77
10. Troubleshooting.....	86
10.1.1. Thunderbird / SeaMonkey displays a red error message at the bottom of the mail window.....	86
10.1.2. Enigmail fails to install on SeaMonkey.....	86
10.1.3. Enigmail fails to install on Firefox.....	87
10.1.4. The Add-ons Manager shows “This item will be installed after you restart Thunderbird”. Or, there is no Enigmail user interface visible.....	87
10.1.5. I have updated Enigmail on Thunderbird, and now it keeps telling me: “A previous install did not complete correctly. Finishing install.”.....	87
10.1.6. I can't tell whether Enigmail works or not.....	87
10.1.7. I installed a new extension and Enigmail stopped working.....	88
10.1.8. Enigmail icons in the toolbar are misaligned.....	88
10.1.9. Enigmail is unable to access the keyserver.....	88
10.1.10. My own signatures are invalid. Enigmail replaces “>” with “[” and spaces with “~” in quoted messages.....	88
10.1.11. I use a non-English character set, and my own signatures are invalid.....	89
10.1.12. Enigmail sees some emails as broken.....	89
10.1.13. I get an error “Enigmail / Enigmime / IPC failed to initialize”.....	89
10.1.14. I cannot read encrypted messages sent to me! I get an error “Secret key needed to decrypt message”.....	89
10.1.15. I lost my passphrase / my key pair / my private key.....	89
10.1.16. After I reinstalled Enigmail, all keys have disappeared from the	

Key Management window.....	90
10.1.17. I get an error whenever I try to post to a newsgroup.....	90
10.1.18. I have set forwarding rules on Thunderbird, and I get an error "Sending failed, please check your settings".....	90
10.1.19. I get the message "OpenPGP error; Encryption/signing failed; send unencrypted message?".....	91
10.1.20. Key import fails with an error "File name too long".....	91
10.1.21. I have some other problem I can't solve.....	91
11. FAQ.....	92
11.1.1. Can Enigmail be used for webmail? When will this feature be added?	92
11.1.2. Are there known incompatibilities with other Thunderbird or SeaMonkey extensions?.....	92
11.1.3. Why is Enigmail incompatible with my Thunderbird / IceDove / ...?	92
11.1.4. Is it possible to use PGP with Enigmail?.....	93
11.1.5. How do I uninstall Enigmail?.....	93
11.1.6. Which files Enigmail modifies on my system?.....	93
11.1.7. Enigmail seems not to work with Gpg4win. What's wrong?.....	93
11.1.8. Why does Enigmail try to use gpg-agent?.....	94
11.1.9. Which key type/size should I choose for my key pair? Which is best?.....	94
11.1.10. How can I test if I'm using Enigmail correctly?.....	95
11.1.11. How do I encrypt automatically my email messages?.....	95
11.1.12. Is it possible to permanently decrypt email messages?.....	95
11.1.13. Is it possible to use S/MIME and OpenPGP encryption concurrently?.....	96
11.1.14. How do I specify the hash algorithm?.....	96
11.1.15. How do I enable the debug log in Enigmail?.....	97
11.1.16. How do I report a bug?.....	97
11.1.17. How many people use Enigmail?.....	97
11.1.18. It would be great if Enigmail could do this-and-this! Could you please implement it?.....	97
11.1.19. How can I encrypt the Subject?.....	97
11.1.20. Why I can't select some keys for encryption in the Key Selection window?.....	97
12. Notes, Tips & Tricks.....	98
12.1. How to choose a good passphrase.....	98
12.2. Protection of the local machine.....	100
12.2.1. Basic protection.....	101
12.2.2. Increased protection.....	101
12.3. Keeping your key pair in a safer place.....	102
12.3.1. External USB drive.....	102
12.3.2. Encrypted volume.....	102
12.3.3. OpenPGP card.....	103
13. Support.....	106

2. INTRODUCTION

There are two main branches of cryptography: symmetric cryptography and asymmetric cryptography.

Symmetric cryptography is the first type of cryptography invented, dating back to 2000 years ago, and the only one most people know. In symmetric cryptography, a *cipher* (cryptographic algorithm) is used in conjunction with a single *key*, for instance a password, to encrypt a message. The message can then be decrypted using the same key.

Symmetric cryptography poses a problem concerning the delivery of secure messages. The sender can encrypt a message and send it to the recipient, but has to provide the recipient the key to decrypt it. The key cannot obviously be sent with the message, and must be communicated through a secure channel. Encryption provides a secure channel for the delivery of messages but, in order to make it usable, the sender must first deliver the key to the recipient.

This catch-22 problem was solved only thirty years ago with the birth of *asymmetric cryptography*, also called *public key cryptography*.

Public key cryptography is much more interesting and useful. It does not operate with a single key but with a *key pair*, composed of a *public key* and a *private key* (also called *secret key*). Public and secret key are created together at the same time using a special algorithm.

Let's show how public cryptography works by taking as an example two people, Alice and Bob, that want to exchange secure messages.

Alice generates her own key pair in advance. Then makes the public key available to anyone, for instance by publishing the key in a public directory, and carefully keeps for herself the secret key. This is perfectly safe, because it is practically impossible (or, as computer scientists prefer to say, *computationally infeasible*) to derive a private key from its companion public key alone. Bob does the same: generates a key pair, publishes his public key and keeps undisclosed his secret key.

When Bob wants to send a confidential message to Alice, he first retrieves Alice's public key from the directory. Then he encrypts the message with her public key and sends the message. Alice decrypts the message with her private key and is able to read it.

Public key cryptography is not only employed for confidentiality (ensure that the message can be read only by the intended recipient), but also for authentication (ensure that the message really comes from the intended sender) and integrity (ensure that the message has not been altered in transit). Authentication and integrity are enforced by appending a *digital signature* to the message.

A digital signature is generated by an algorithm that uses a *hash function* in conjunction with a key. A hash function is a function that takes in input a message of any length, and outputs a string of fixed small length called *digest* which is a distillate of the message fed in input. Notable features of hash functions include that it is practically impossible to derive the input from the output, and that changing just one bit of the input results in a completely different output.

Hence Bob writes the message, generates the digital signature for the message using a predetermined hash function and his private key, appends the signature to the message, and sends to Alice the whole lot. Alice receives the message and verifies the signature using the same hash function and Bob's public key. If the signature is valid, then the sender is authenticated, because only the owner of the private key, Bob, could have signed the message. This guarantees also the integrity of the message, because had the message been altered in transit, it would resolve to a different digest and the signature would not match.

Public key cryptography was firstly discovered by James Ellis, Clifford Cocks and Malcolm Williamson of the British Government Communication Headquarters in 1975, but the discovery was filed as classified information and never divulged. The following year researchers Whitfield Diffie, Martin Hellman and Ralph Merkle independently made the same discovery and published it on a paper. One year later Ronald Rivest, Adi Shamir and Leonard Adleman provided the first practical implementation of a public key cryptography algorithm by developing the RSA cipher.

Then in 1991 Phil Zimmermann, a free speech activist and anti-nuclear pacifist, developed Pretty Good Privacy (PGP), the first software available to the general public that utilized RSA for email encryption and signing. Zimmermann, after having asked a friend to post the program on the worldwide Usenet, found himself prosecuted by the government and was even charged by the FBI for illegal weapon export. The charges were eventually dropped, and Zimmermann later founded PGP Inc., now acquired by PGP Corporation.

In 1997 PGP Inc. submitted a standardization proposal to the Internet Engineering Task Force. The standard was called OpenPGP and defined in 1998 in the IETF document RFC 2440. The latest version of the OpenPGP standard is described in RFC 4880, published in 2007.

PGP is now a famous commercial product for communication security and privacy in corporate, business and home environment, and is available at <http://www.pgp.com> and <http://www.pgpi.org>.

Nowadays there are many OpenPGP-compliant products: the most widespread is probably GnuPG (GNU Privacy Guard, or GPG for short) which was developed in 1999 by Werner Koch. The GnuPG Project is hosted at <http://www.gnupg.org>.

GnuPG is free, open-source and available for several platforms. It is a command-line only tool, which means that it does not have a graphical interface.

Enigmail, first released in 2001 by Ramalingam Saravanan and maintained by Patrick Brunschwig since 2003, is the GnuPG plug-in for Mozilla email clients (Thunderbird and SeaMonkey). Enigmail interfaces seamlessly with GnuPG and provides a GUI to make easy for everyone to securely encrypt, decrypt, sign, and verify the signature on email messages. The homepage of the Enigmail Project is <http://enigmail.mozdev.org> or <http://www.mozilla-enigmail.org>.

Enigmail, GnuPG, Thunderbird and SeaMonkey are all free and open-source software. They can be downloaded, copied and used for free. As open-source projects, their source code is available for everyone who desires to examine or customize it.

PGP, along with all its variants, is the most famous and widely used public-key-encryption software in the world. Since its creation it has allowed people in totalitarian countries to enjoy privacy, enforce free speech, fight censorship, and protect human rights. It makes use of the strongest ciphers known in the scientific literature, and if utilized properly it is virtually unbreakable.

3. ACKNOWLEDGEMENTS

This Handbook stems from the Quick Start Guide written by Robert J. Hansen, and incorporates technical references written by Patrick Brunschwig (the current developer of Enigmail) and Olav Seyfarth. While writing the handbook I followed the comments, corrections, criticisms, and encouragements from the whole Enigmail team; the handbook also contains many contributions originally posted by the team on the Enigmail forum, newsgroup, and mailing lists. Thanks to them all!

Thanks to Werner Koch for reviewing the part about the OpenPGP smart card, and for giving permission to use his pictures at page 103. Ludwig Hügelschäfer, Christian Marg, and Dave Schaefer also helped with very useful tips and comments.

This document was composed with OpenOffice.org 3 by Sun Microsystems Inc. The icons used throughout this document are part of the Crystal Clear set by Everaldo Coelho, released under GNU LGPL.

The Enigmail logo was designed by Olav Seyfarth. It uses the Good Times font by Ray Larabie, released as freeware.

Thunderbird and SeaMonkey are trademarks of the Mozilla Foundation.

4. THE ENIGMAIL TEAM

Patrick Brunschwig	Project Maintainer and Lead Developer
Ramalingam Saravanan	(no longer active) Original author
John Clizbe	Quality Assurance and User Support
Olav Seyfarth	Website and User Support
Shane M. Coughlan	Website and Testing
John W. Moore	Testing and User Support
Robert J. Hansen	Usability and User Support
Barry Porter	(no longer active) Testing and User Support
Daniele Raffo	Forum Management and Documentation

5. GETTING STARTED

This chapter will illustrate how to get Enigmail up and running.

To use Enigmail, you first need to install GnuPG. GnuPG comes in two flavours: the standard GnuPG (development branch v1.x) and GnuPG 2 (development branch v2.x). The former is the well-known and portable version of GnuPG, while the latter is the enhanced version with additional features, but harder to build and set up. Both flavours offer the same security level. You can safely download and install the latest version (1.x) of the standard GnuPG.

Furthermore, if you don't already use it, you need to install a Mozilla mailclient i.e. either Thunderbird or SeaMonkey, as you prefer. Other derivatives of the Mozilla mailclient, for which Enigmail might work as well, also exist; however, Enigmail is not officially supported for these derivatives.

You may install first GnuPG and then the mailclient, or vice versa.

5.1. Installing GnuPG

5.1.1. Installing GnuPG on Microsoft Windows

GnuPG provides a Windows installer. Download the latest stable version from <http://gnupg.org/download/index.en.html#auto-ref-2>.

Once you have downloaded the installer, just double-click it to begin the installation process. This is very straightforward; you can literally just keep clicking *Next* until it's finished. You don't need to modify anything in the configuration of GnuPG, neither; the default settings will work fine.

Learning how to use the command-line GnuPG is not required; once installed, all operations will be achieved via Enigmail.

Some people prefer to use Gpg4win (<http://www.gpg4win.org>), a sibling project to GnuPG, that in addition offers a GUI for key management, encryption, and decryption. The Gpg4win installer also contains additional optional components such as a certificate manager, plug-ins for MS mailclient and browser, and a mailclient already integrated with the GnuPG plug-in. Enigmail can use Gpg4win instead of GnuPG just as easily.

5.1.2. Installing GnuPG on Macintosh OS X

You have three basic ways to install GnuPG on OS X. Most users will choose the first option.

The first and most popular option is to use MacGPG. The MacGPG Project provides pre-built Universal Binaries of GnuPG 1.4.9 and later for users running OS X 10.4 (Tiger). Just download the package from <http://macgpg.sf.net> and install it.

Please note that if you are running OS X 10.3 or earlier, you will not be able to use the MacGPG packages. Although the MacGPG project provides older versions of GnuPG packaged for older versions of OS X, since Enigmail depends on GnuPG 1.4.9 or later, older versions of GnuPG won't be useful.

The second option is to use Fink. The Fink Project (<http://fink.sf.net>) keeps a very current version of GnuPG in their source tree. If you're using Fink, install GnuPG by typing `fink install gnupg` into Terminal.app.

The third option is the MacPorts Project (<http://www.macports.org>), formerly called "DarwinPorts", that keeps a current version of GnuPG in their source tree. If you're using MacPorts, open up Terminal.app and type `sudo port install gnupg`.

5.1.3. Installing GnuPG on Linux / UNIX

The best thing is to get a pre-compiled version of GnuPG for your UNIX system. Compiling GnuPG from source is not recommended for beginners.

Most Linux distributions today include GnuPG by default. To find out if this is the case, open a command prompt and type `gpg --version`. If it tells you that you've got GnuPG 1.4.9 or some later version, then you don't need to do anything.

The various flavors of BSD UNIX have a common way of installing software. Please see the instructions for MacPorts above.

5.2. Installing Thunderbird / SeaMonkey

Thunderbird is a standalone email client, while SeaMonkey is a suite that incorporates a mailclient, a browser, an HTML composer, and an IRC client. Both stem from the Mozilla project. The mailclient interface and functionalities are similar, and you can choose the one you prefer. You can download Thunderbird from <http://www.mozillamessaging.com/thunderbird> or SeaMonkey from <http://www.seamonkey-project.org>.

Many Linux distributions ship with their own customized version of Thunderbird. If you use your distribution's version of Thunderbird, you must use your distribution's version of Enigmail. If you get Thunderbird from the official site and install it yourself, though, you can use the official Enigmail releases provided by the Enigmail Project.

Thunderbird and SeaMonkey have their excellent documentation and user group support, hence we'll skip the steps to install and configure them.

You should have your mailclient and your email account fully configured before proceeding to the installation of Enigmail.

5.3. Installing Enigmail

Download the latest version of Enigmail suitable for your mailclient and operating system from <http://enigmail.mozdev.org/download>. Enigmail comes as a XPI file which is the standard extension for Mozilla plug-ins. The latest Enigmail version is 1.0.0, which can be installed on Thunderbird 3.0 and SeaMonkey 2.0.1 and higher. If you have an older mailclient, you must install a previous version of Enigmail accordingly: for instance, version 0.97a can be installed on SeaMonkey 2.0, and version 0.96.0 can be installed on Thunderbird 2.0 and SeaMonkey 1.1. The Enigmail download page will help you in automatically identify and download the correct version.

Screenshots throughout this handbook mostly show Enigmail 0.96.0 on SeaMonkey 1.1 (Modern theme) running under Windows XP, and a few show Enigmail 0.97a on SeaMonkey 2.0. If your installation is different concerning the Enigmail version, mailclient, or operating system, there will be minor differences in the GUI but functionalities will be the same.

5.3.1. Installing Enigmail on Thunderbird

You need to right-click on the download link and choose *Save link as...* This is important especially for Firefox users, as Thunderbird and Firefox both use the XPI extension for their plug-ins. If you click on the download link, Firefox will try to install Enigmail as a Firefox plug-in, which will not work. Use *Save link as...* instead and save the XPI file on your machine.

Start Thunderbird. In the menu bar of the main window, select *Tools* → *Add-ons* to bring up the Thunderbird Add-ons Manager window. Select *Extensions*, then click the *Install* button and tell Thunderbird where you saved the Enigmail XPI file. Another window will pop open, warning that you're about to install a plug-in. Confirm your decision. You will then need to restart Thunderbird. Once Thunderbird has restarted, Enigmail will be ready to go.

5.3.2. Installing Enigmail on SeaMonkey

If you use SeaMonkey, clicking on the download link will work fine. As already said, SeaMonkey integrates a browser and an email client, hence you can install directly from the browser any XPI plug-in for the email client or for the browser itself. Just confirm that you want to install the plug-in when you're asked to.

You will then need to restart SeaMonkey. Once SeaMonkey has restarted, Enigmail will be ready to go.

5.3.3. Installing a locale for Enigmail

Enigmail is available in many languages. The following locales are already included in Enigmail 1.0.0:

ar	Arabic	it-IT	Italian
ca	Catalan	ja-JP	Japanese
de-AT	German (Austria)	ko-KR	Korean
de-DE	German (Germany)	nb-NO	Norwegian
el	Greek	pl-PL	Polish
en-US	English (USA)	pt-BR	Portuguese (Brazil)
es-ES	Spanish	pt-PT	Portuguese (Portugal)
fi-FI	Finnish	ru-RU	Russian
fr-FR	French	sl-SI	Slovenian
gl-ES	Galician	sv-SE	Swedish
hu-HU	Hungarian	zh-CN	Chinese

If you want to use Enigmail localised in a language that is not listed above, you need the relevant Language Pack. To do so, download and install Enigmail as previously explained. Then, before restarting the mailclient, go to <http://enigmail.mozdev.org/download/langpack.php> and choose the Language Pack relevant to the preferred language. Download the Language Pack, which is a XPI file, and install it just like you did with Enigmail. Then, restart the mailclient.

If Enigmail still appears in English after the restart, you need to switch once to the default language English (US), restart the mailclient, switch back to your localisation, then restart the mailclient again.

To change the language in SeaMonkey, use the menu command *Edit* → *Preferences...* → *Appearance* → *Languages/Content*. To change the language in Thunderbird, the easiest way is to install Benjamin Smedberg's Locale Switcher extension, available at <https://addons.mozilla.org/enUS/thunderbird/addon/356>.

A note for testers: Enigmail nightly builds are not localized. They come without Language Packs and are meant to be tested in English only.

6. QUICK START

Run the email client you installed (Thunderbird or SeaMonkey). You will notice a new submenu called *OpenPGP* in the menu bar: that's the submenu for all Enigmail operations.

Select *OpenPGP* → *Preferences* to have access to the Enigmail preferences window. There is a toggle button named *Display Expert Settings / Hide Expert Settings* that displays or hides several options to fine tune your installation of Enigmail. Throughout the rest of this handbook we'll assume you stay in Expert Settings mode.

Enigmail is equipped with a Setup Wizard that can guide you through all steps to configure Enigmail and have it ready to use. This is intended for inexperienced users, but is not necessary: you may as well configure Enigmail by hand, which will grant you a deeper knowledge of the mechanisms of Enigmail.

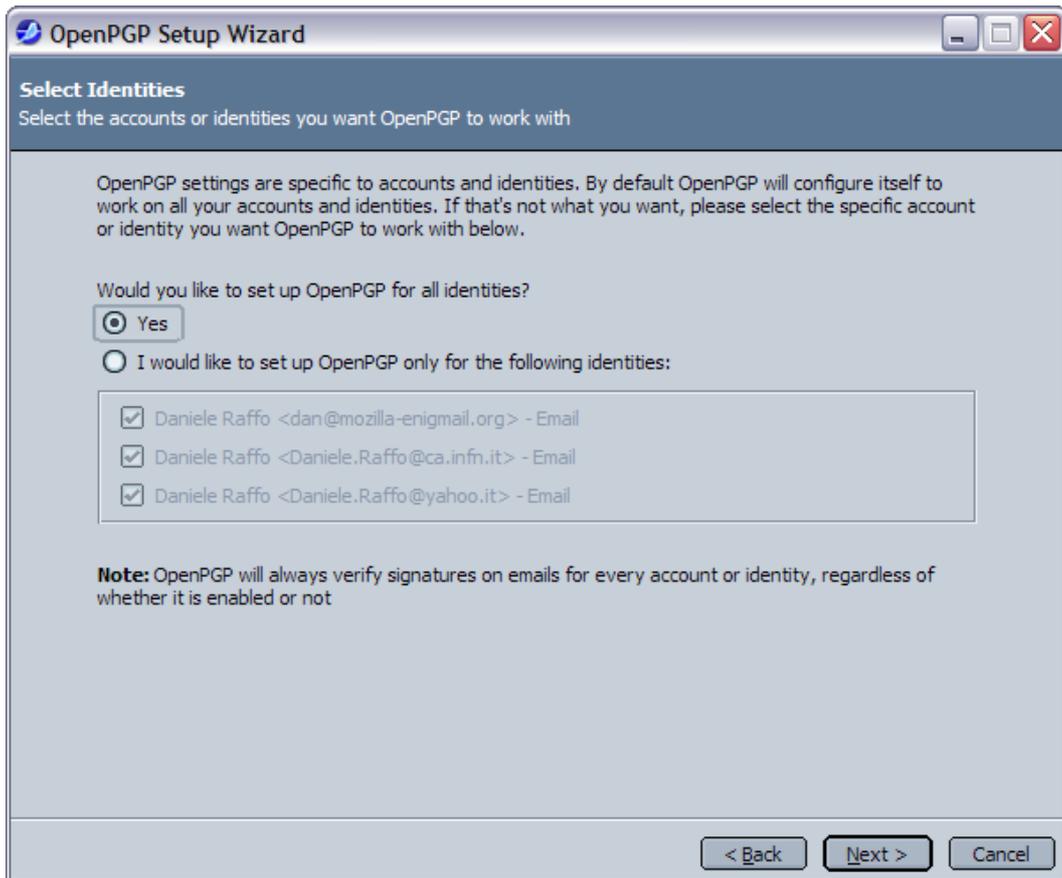
The rest of this chapter is a step-by-step guide to the Setup Wizard followed by a very basic explanation of the signing and encryption functions. If you decide not to use the Setup Wizard, you can go directly to the next Chapter.

6.1. The Setup Wizard

Select *OpenPGP* → *Setup Wizard* and the following window will appear. Remember that you can abort the Setup Wizard at any time, and run it again from the *OpenPGP* menu.

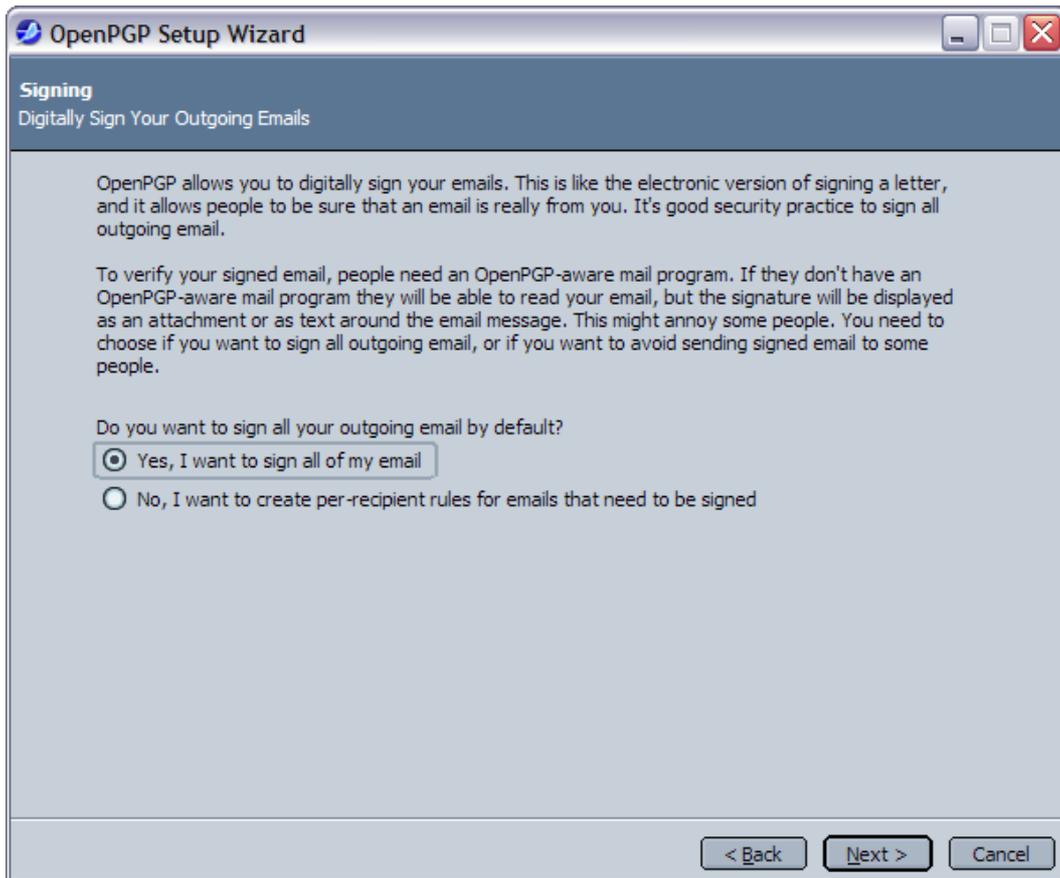


Select *Yes* and click *Next*.



Here you can choose whether to have Enigmail configured to work on all your email accounts and identities, or for some only. If you are a beginner user, you should select *Yes* to avoid confusion when switching accounts. How to set up an account for use with OpenPGP/Enigmail is explained in Section 8.1.

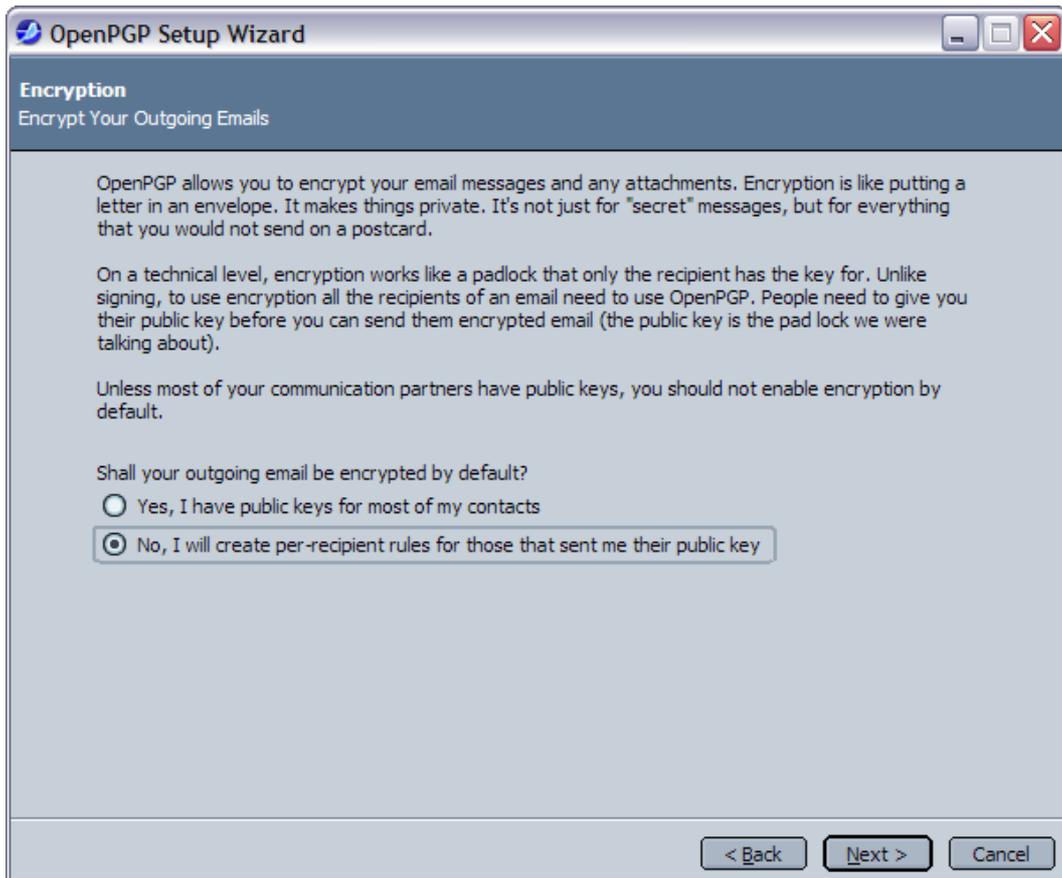
Click *Next*.



Here you can choose whether to sign all mail you send, or to pre-select recipients (through more complex per-recipient rules) to whom send signed messages. Per-recipient rules are explained in Section 8.6.

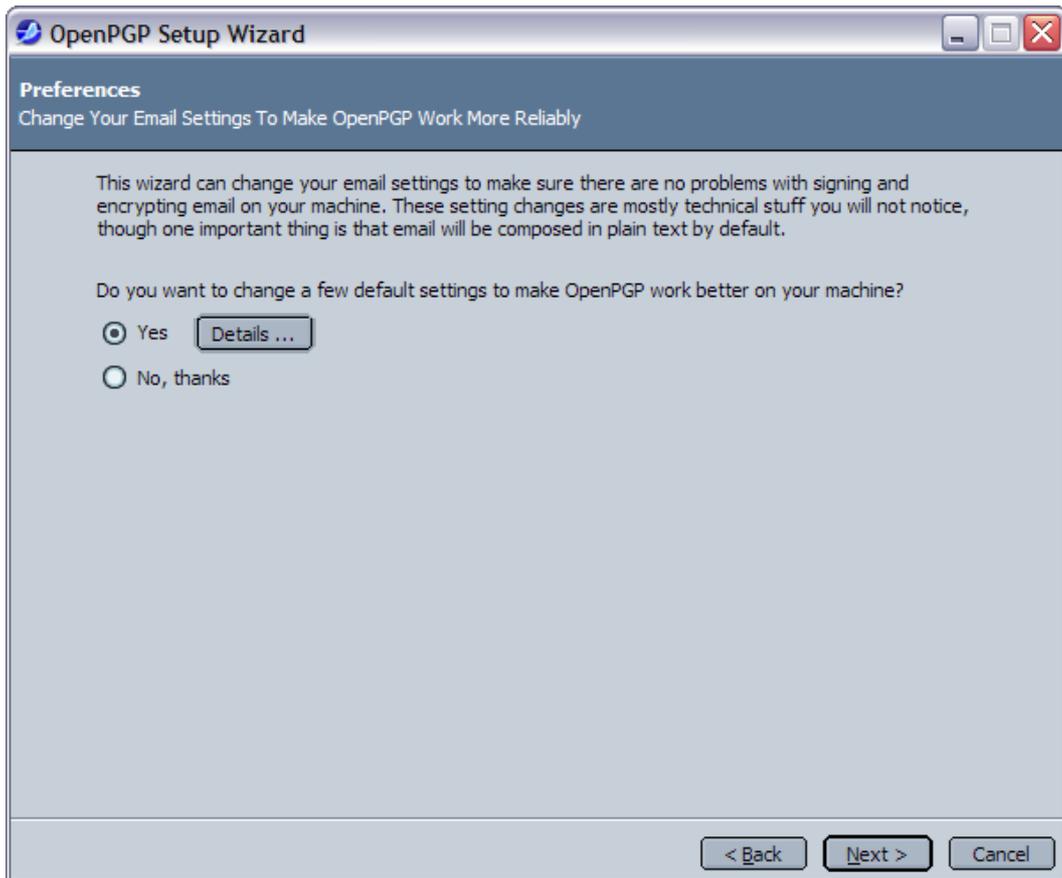
You can safely select **Yes** here. In this way, signing will be enabled by default for all your outgoing messages, but you can easily turn it off in the Message Composing window.

Click *Next*.



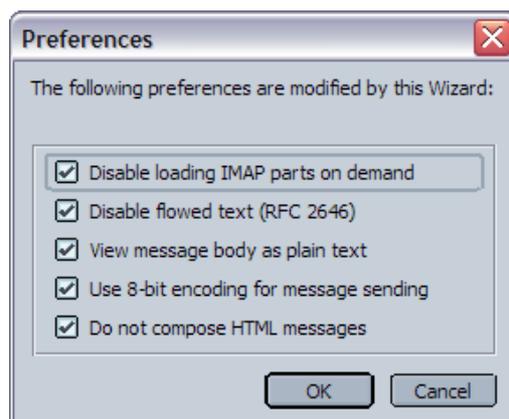
Here you can choose whether to enable encryption by default for all your outgoing mail. To encrypt a message, you need to have the public key of the recipient(s). We recommend you select *No* so that you can enable encryption only if you really mean to.

Click *Next*.



The Setup Wizard here asks you permission to modify some email settings to make sure Enigmail works seamlessly on your machine. You can safely select **Yes**.

If you want to see what's going to be changed, click on *Details...* and the following window will appear:

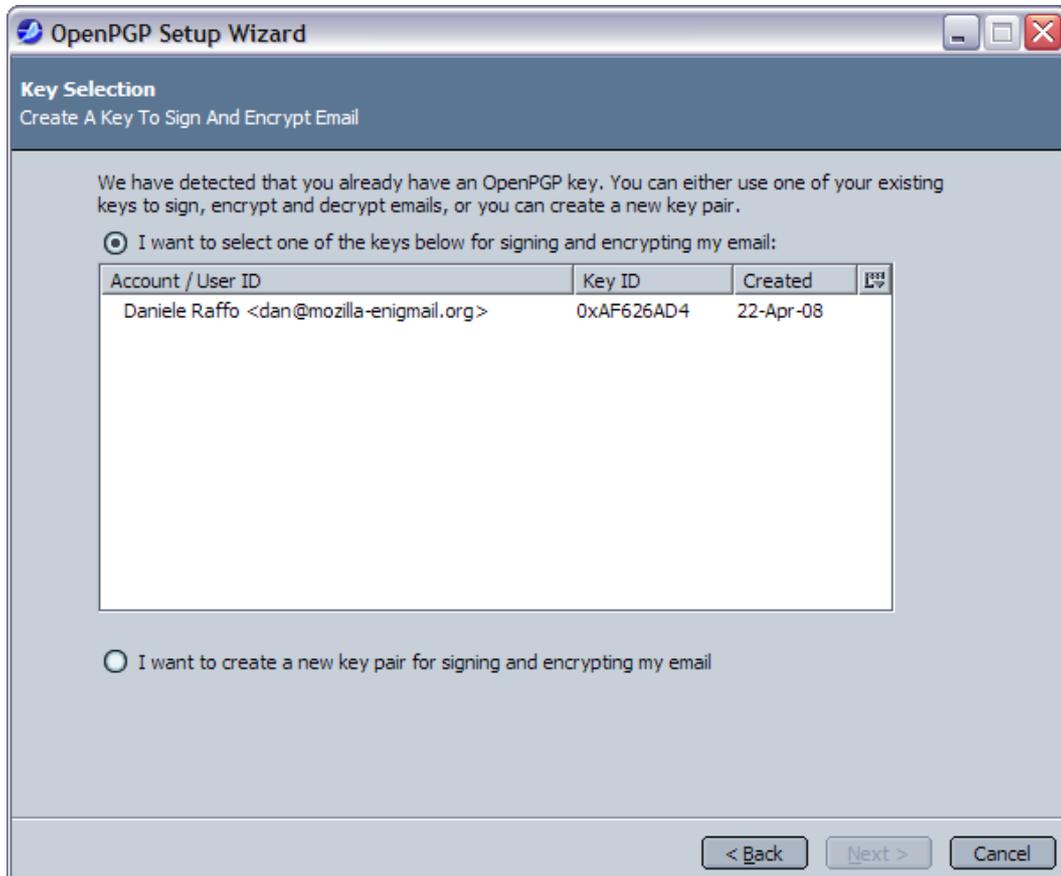


Don't worry if you don't understand what that means: these are perfectly reasonable settings, and we advise to leave all options checked. The most noticeable change will be that, from now on, email messages will be composed and viewed in plain text instead of HTML. This is necessary because HTML may cause problems when using signing and/or encryption.

Note that, as a good rule of netiquette, you should refrain from using HTML also when writing normal (unsigned, unencrypted) mail. The other settings are relevant to more technical issues such as message downloading from IMAP servers, text formatting, and character encoding. These will not improve your mailclient experience in any way.

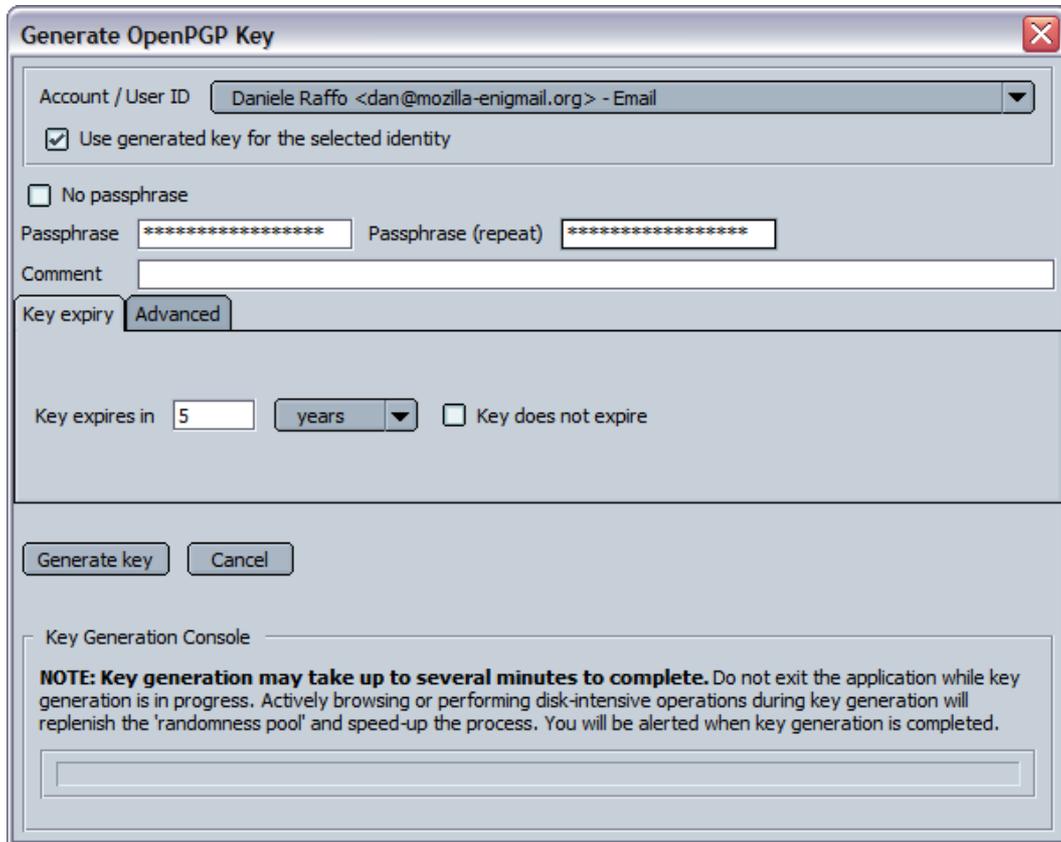
Click *Next*.

Perhaps you already used Enigmail (or GnuPG, or any other OpenPGP software) in the past before installing this version of Enigmail, and have created a GnuPG key pair that is still on your machine. In this case, Enigmail will find it in the GnuPG directory and will offer to use it for your identity. Otherwise, you must generate a new key pair.

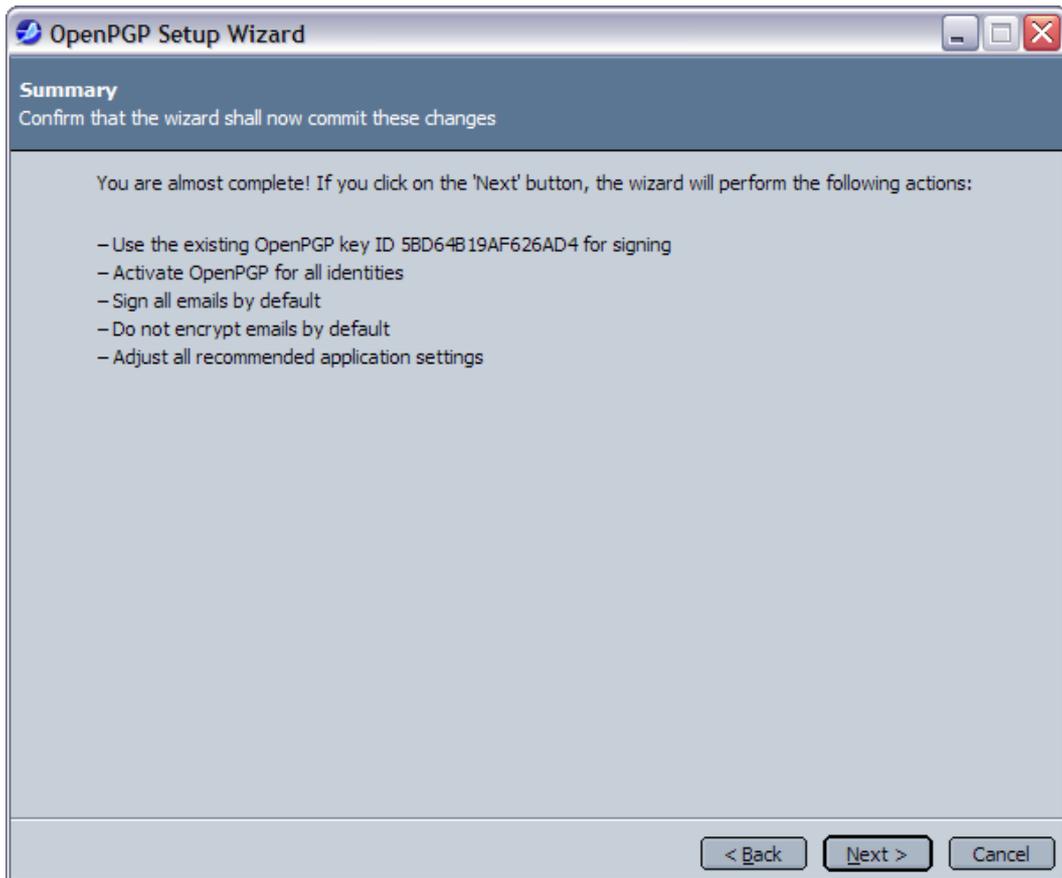


In this figure, Enigmail has detected a previous OpenPGP key that can hence be used. Select the key and click *Next*.

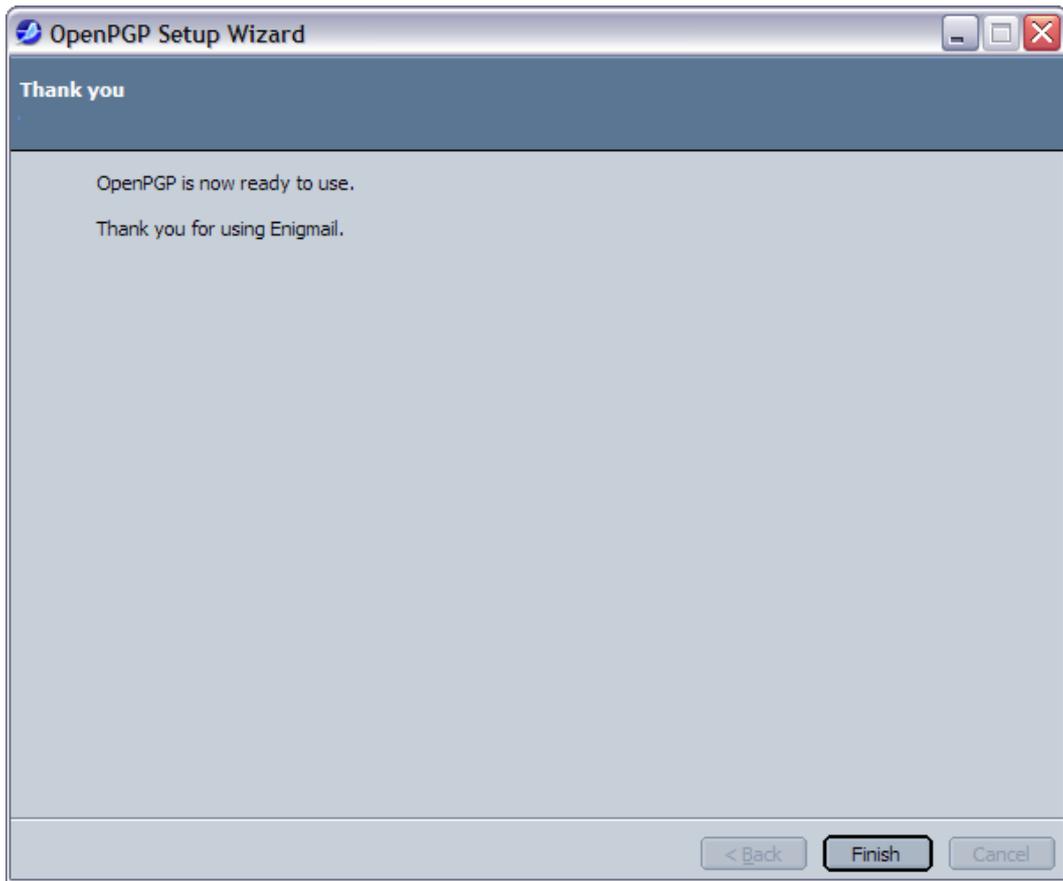
More likely, this is the first time you use OpenPGP, so you will need to generate a new key pair. Therefore, select *I want to create a new key pair for signing and encrypting my email* and click *Next*. You will be brought to the Generate OpenPGP Key window:



Choose a passphrase to protect your key pair: you will need to type that passphrase every time you sign or decrypt a message. You don't need to change anything else, as the default settings will work fine. Just make sure *Use generated key for the selected identity* is checked, then click on *Generate key* and wait. You will be alerted once the key has been generated. If you want to know more about key generation, read Section 7.2.

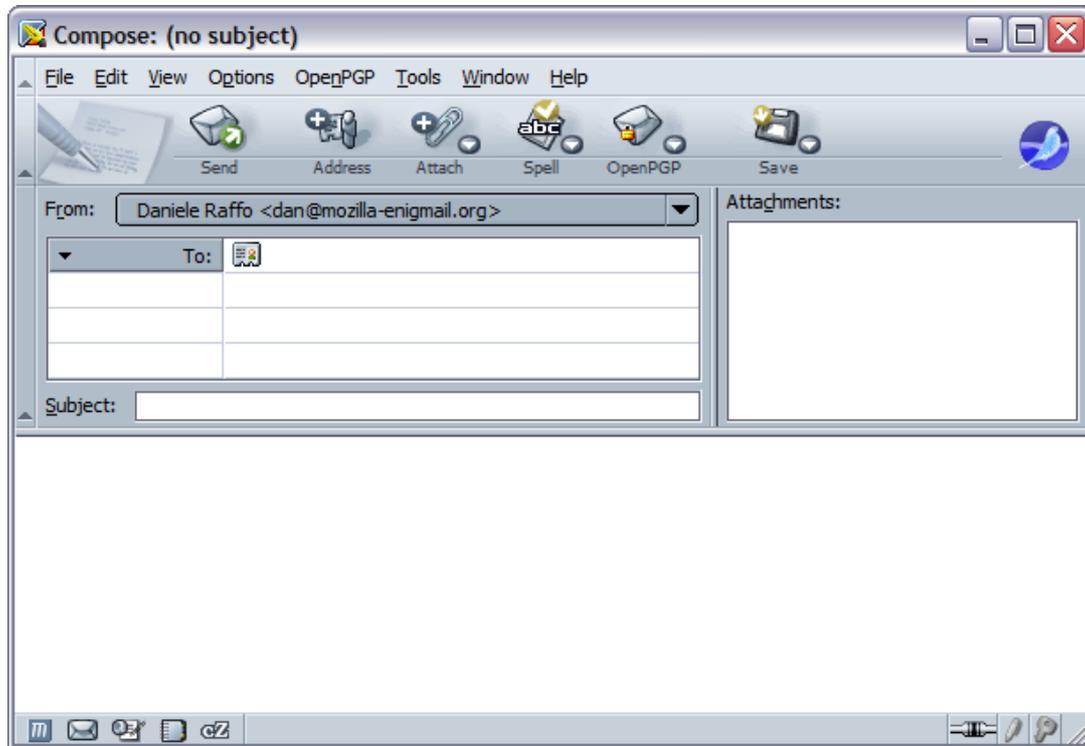


The last window of the Setup Wizard allows you to review the choices you've made and confirm. Click *Next* to commit the changes and finish.



Enigmail is now configured and ready to use.

When you start writing a mail, you will now notice a new *OpenPGP* button in the toolbar of the Compose window. This button allows you to sign and/or encrypt the message. In the bottom right corner, a pen and/or a key icon will be green lit to signal that, respectively, signing and/or encryption is enabled.



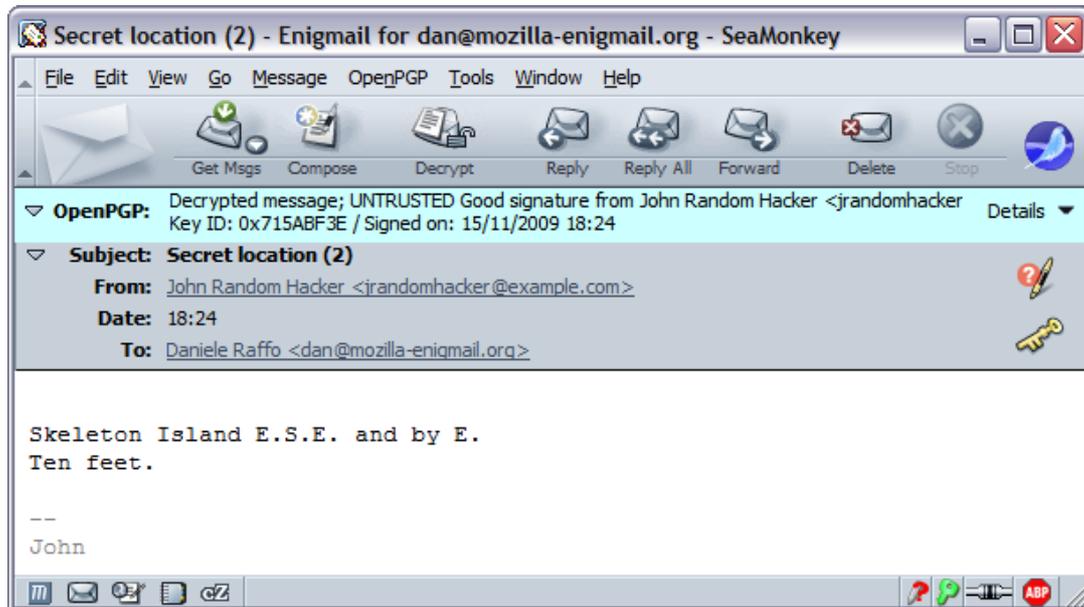
You can immediately send signed mail to anyone. However, in order to allow someone to verify your signature or to send you an encrypted message, you must provide him with your public key. You can send your public key as an attachment by choosing *OpenPGP* → *Attach Public key...* in the Compose window, and then by selecting your key in the Key Selection window that will appear.

All your stored keys (your own key pair, and other people's public keys you have acquired) can be seen in the Key Management, via the menu command *OpenPGP* → *Key Management*.

To send encrypted mail, you need to have the public key of the recipient. You can acquire it in one of the following ways:

- ask him to email you his public key as an attachment; then right-click on the attachment and choose *Import OpenPGP Key*;
- download his public key from his web site as an ASC file, then import it via *File* → *Import Keys From File* from Key Management;
- retrieve his public key from a keyserver via *Keyserver* → *Search for keys* from Key Management.

When you receive a mail message that has been OpenPGP-secured (signed and/or encrypted), it will appear as such:



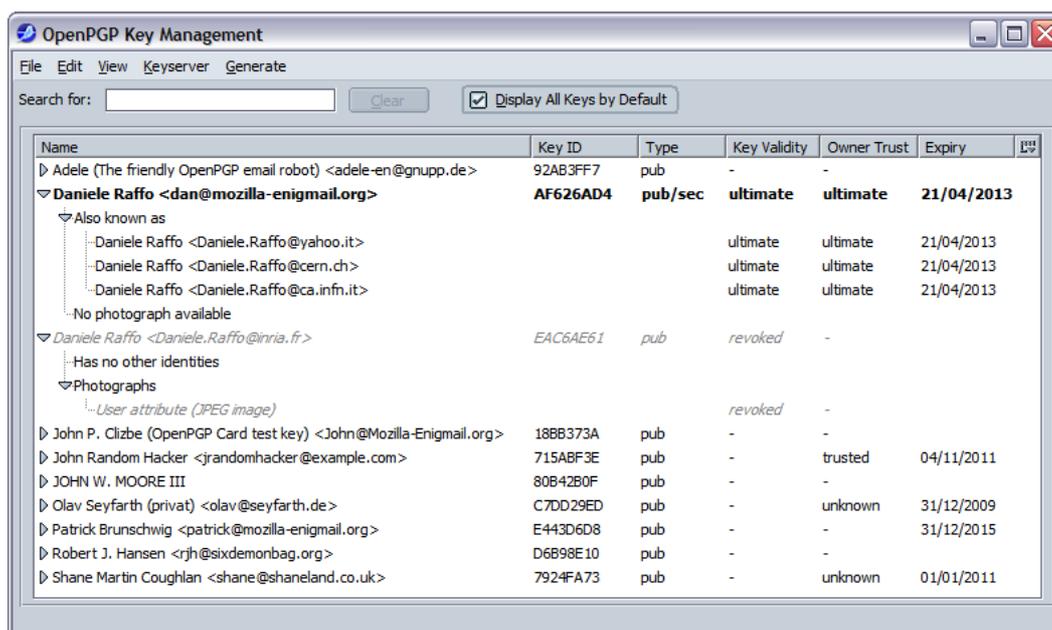
The message in the figure has been both signed and encrypted, as shown in the OpenPGP status bar.

Thank you for using Enigmail! These are the basics of it. You can read about all topics in detail by perusing the rest of this Handbook.

7. KEY MANAGEMENT

Once you have Enigmail on your system, you need to populate it with keys: it's pretty useless without them. You need to have your own key pair in order to sign messages and allow anybody to send encrypted messages to you. Furthermore, you will need the public key of a person in order to send that person an encrypted message, or to verify the signature in a message sent by that person.

Select *OpenPGP* → *Key Management* to open the Key Management window. You can also open Key Management as a standalone application by appending `-pgpkeyman` to the command that runs Thunderbird / SeaMonkey, e.g. on Windows: "`C:\Program Files\mozilla.org\SeaMonkey\SeaMonkey.exe`" `-pgpkeyman`.



The Key Management window shows all keys (yours and other people's) you have stored in your machine; this is called your *keyring*. The set of all public keys that you have collected is often called your *public keyring*. Key pairs (i.e. a bundle of public key and its companion secret key) are shown in bold. Revoked or expired keys, which are invalid keys, are shown in italics. All others are valid public keys.

If you run the Setup Wizard and generated a new key pair, or imported an existing one, it will be shown here. (If it doesn't, tick the option *Display All Keys by Default*.) Otherwise, the window will be empty.

By clicking the expand gadget at the left of each key, you can see the key's additional user IDs and PhotoID, if present. The columns (*Key ID*, *Type*, *Key Validity*, *Owner Trust*, *Expiry*, and *Fingerprint*) show a number of other key properties: you can choose which columns you want to see by selecting them in the rightmost gadget in the column header bar. In the rest of this chapter we'll explain what these properties mean.

The menu bar of the Key Management window allows you to operate on the keys of your keyring. To do that, select a key, and then choose a menu item. You also have most of these menu items in a pop-up menu that shows up when you right-click on a key. Some menu items will be disabled (greyed out) if the operation is not permitted on the key you selected. Some operations on a key require that you have the companion private key, so you can accomplish them on your key pair but not on other people's public keys.

7.1. Importing an existing key pair

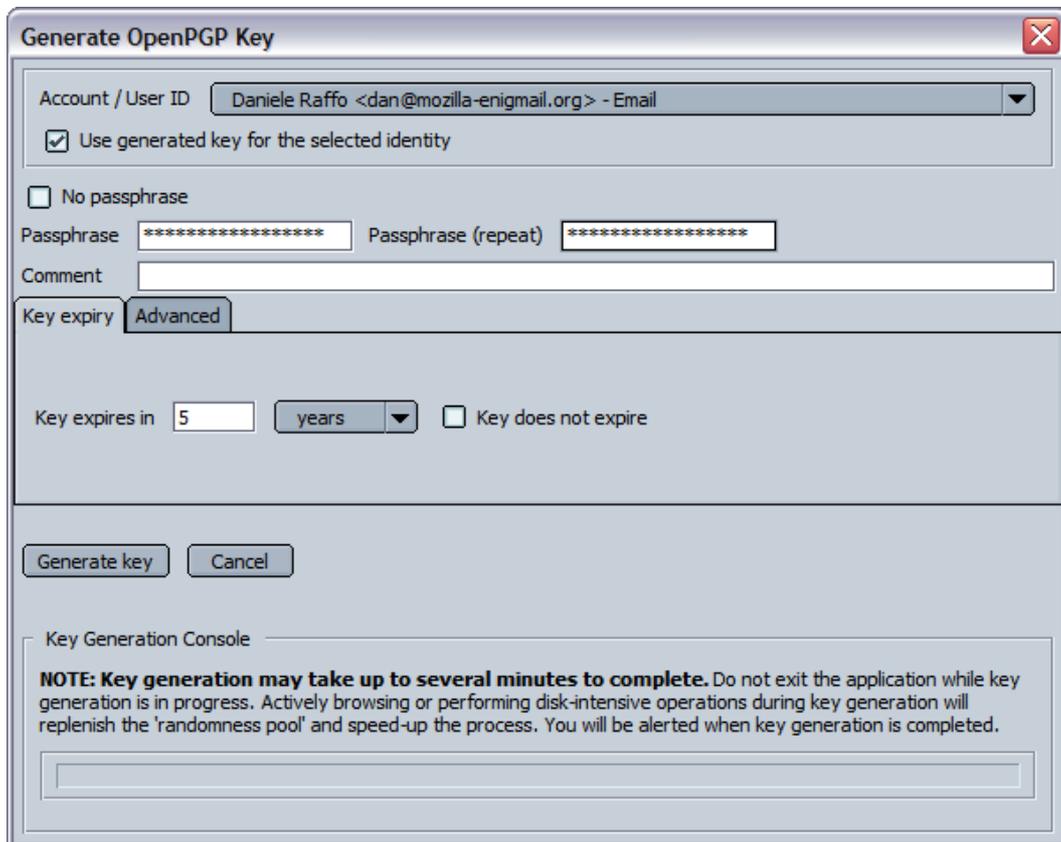
Perhaps you have already used GnuPG, Enigmail, or any other OpenPGP software in the past, and you generated a key pair in that occasion. In this case, you might want to use your existing key pair in this installation of Enigmail. Make sure you remember the passphrase of your old key pair, otherwise you will be unable to use it.

To import your key pair, choose *File* → *Import Keys From File* and select the file containing your key pair. Your key pair will be imported for you to use it.

If you never had a key pair (which would be the usual case for an inexperienced users) you need to generate one, and it's time to do it now. Don't worry; this is easier than it sounds, and how to do that is explained in the next Section.

7.2. Generating your own key pair

You need to own a key pair to join the elite that communicates securely using GnuPG. You can create one at any moment by selecting *Generate* → *New Key Pair*. A window will open:



To successfully generate a key pair, you must proceed through the following steps.

7.2.1. Tell Enigmail which account to use

The *Account/User ID* drop-down menu shows all your email accounts and identities configured on your mailclient: you must select one to associate to this key pair. Select whichever account will be receiving encrypted mail, and make sure the option *Use generated key for the selected identity* is ticked.

Should you decide later that you want to change the email address associated to the key pair, or use the same key pair for multiple email addresses, you can do this easily; how to do that is explained in Section 7.3.2.

In the *Comment* field you can write something about this key pair. This will help you distinguish keys for different purposes. The comment is optional and, if set, it will appear next to the user ID. Note that the comment will also appear in the exported public key that you distribute to other people.

7.2.2. Choose a passphrase

Your private key is all that you need to send signed messages and decrypt messages that you receive on your selected email account. Should the private key fall in enemy hands, this would allow someone else to sign messages on your behalf and decrypt messages that were supposed for your eyes only.

Luckily, GnuPG uses an additional layer of protection: the private key is protected with a secret passphrase. You're being asked here what the secret passphrase should be for your new key pair: choose something that is easy to remember but very hard for someone to guess. Read Section 12.1. for some suggestions on how to choose a good passphrase.

Enter your passphrase in the *Passphrase* field, then repeat it in the *Passphrase (repeat)* field.



If you forget your passphrase, you will be unable to use your private key. There is no way to recover the passphrase: your only hope is to try to remember what the passphrase was. This is a security feature of GnuPG and cannot be circumvented.

You can also choose not to protect the key with a passphrase by ticking the option *No passphrase*, although we strongly recommend you to not do that.

7.2.3. Choose the time expiry of the key

It may happen that some day you lose your private key or your passphrase, and therefore are unable to use your key pair. It may also happen that your private key gets compromised e.g. an intruder manages to have access to your computer and steal your key pair, or you could send someone your private key by mistake.

Furthermore, there might be some breakthrough in cryptanalysis or simply the discover of a weakness in a cryptographic algorithm, although this possibility is more remote. In this case, the cipher and key size you chose for your key pair many years ago could be unfit for offering valuable security in today's standards.

In all these unfortunate cases, it is a good idea to have defined a key pair that is valid only for a limited period of time.

The *Key Expiry* tab allows you to limit the lifespan of your key pair. After the allotted time, the public key will automatically be marked as invalid (expired), and people will be prevented to use it. The messages previously encrypted and signed with that key pair will still be decryptable and verifiable, though. Once your key pair has expired you will need to generate another one, and distribute around your new public key.

A good time lapse for key expiration is between 3 and 5 years. You may also choose to have a key that never expires, although we do not recommend to do so.

It is also possible, and recommended, to create a revocation certificate that you can use at any time to mark the key as invalid; this will be explained further on.

7.2.4. Choose the key type and size

By clicking the *Advanced* tab you can choose some properties used for the generation of your key pair: the *Key size* (1024, 2048 or 4096 bits) and the *Key type*, which defines which cipher will be used (RSA or DSA & ElGamal).

Basically, the larger the key size, the stronger the key, the greater the processing power requested for encryption/decryption. If you are going to send a lot of messages that will be decrypted on old machines or on PDAs, you should probably choose a 1024-bit key. Otherwise, you don't really need to ponder your choice: the default option (2048-bit RSA key) offers excellent security coupled with a good usability, and will work fine.

See also FAQ entry 11.1.9.

7.2.5. Generate the key

Once you made all these choices, simply click on the button *Generate key* and wait.

Your computer makes use of a great quantity of random numbers in order to create the key. You may speed up the process by using the computer in the meanwhile, or simply by randomly typing on the keyboard or wiggling the mouse around to generate more randomness. On a modern computer, a standard 2048-bit RSA key pair takes no more than a dozen seconds to generate.

Congratulations! You just created your first key pair.

7.2.6. Generate the revocation certificate

As we just said, it may happen that your private key gets lost or compromised. In this case it is of crucial importance to have a revocation certificate for that specific key pair.

Once it finishes generating your key, Enigmail will offer you to also generate such a certificate. If you accept (which we recommend), click on *Generate certificate*. You will be asked where to save the revocation certificate file. The revocation certificate file has an ASC extension to indicate it's in ASCII-armored format, similar to uuencoded documents. This is the common file format for exported certificates, key pairs, public keys, whole keyrings, and all exported GnuPG output in general.

Keep the revocation certificate in a safe place and not on the same machine you installed Enigmail, so if you even lose your key pair (e.g. due to a hard drive crash) you don't lose the revocation certificate with it. An external hard drive, where you also put a backup copy of your key pair, will be fine.

The revocation certificate can be used to invalidate your key pair at any time. You do not need to type your passphrase or physically have in your keyring the key pair you are revoking, so this will work great even if you forgot your passphrase or accidentally deleted your key pair. However, keep in mind that anyone having access to the revocation certificate can revoke your key pair!

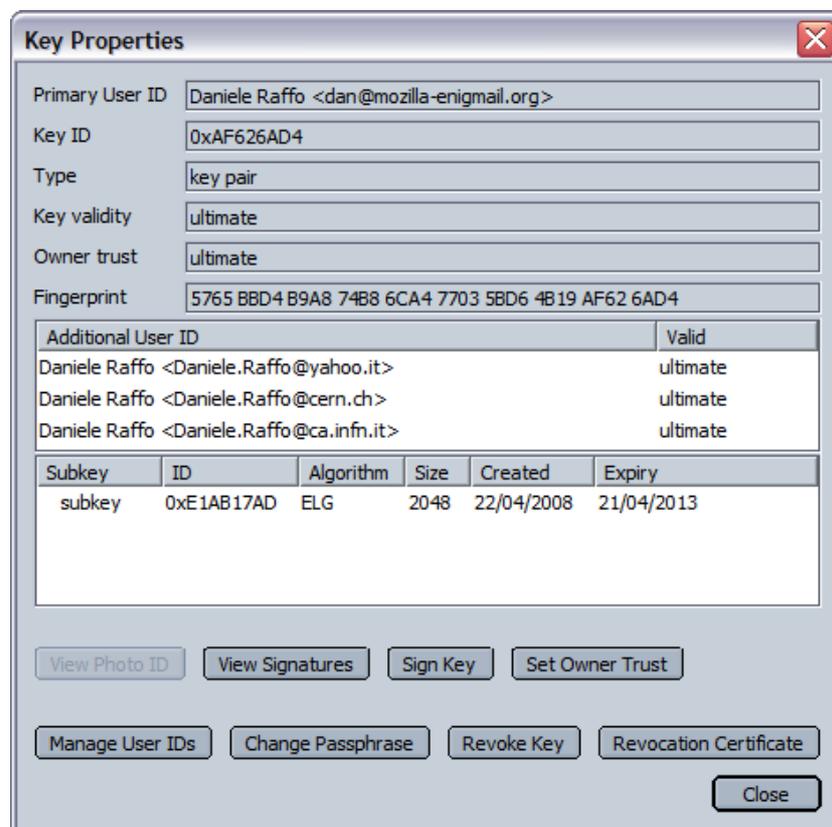
You may also generate the revocation certificate at any later time by selecting your key pair and choosing *Generate* → *Revocation certificate*.

7.3. Operations on your key pair

Once generated, your key pair should show up in the Key Management window. (If it doesn't, tick the option *Display all keys by default*.)

7.3.1. Examining the key properties

Select your key pair and choose *View* → *Key Properties*. A new window will pop up, showing all the key's properties. As you've seen, most of these properties can also be viewed directly from the Key Management window.



- The *Primary User ID* is the name and email address associated with the key i.e. the key owner's. These match the name and address of the email account you generated the key for.
- The *Additional User ID* field show the other user IDs (name and email address) associated with the key, if any. By now your key pair has no additional user ID: should you want to add an user ID, how to do that is explained in Section 7.3.2.
- The *Type* of the key shows “key pair” (“pub/sec” in Key Management window) to mean that this is a key pair. For other people's public keys that you have imported, this field will show “public” (“pub” in Key Management window).

- *Key validity* and *Owner trust* indicate respectively the validity of the key and the trust in the key's owner. Key validity will show you whether a key has expired or has been revoked (and is hence invalid). For the valid key pair that you just created, validity and owner trust should be set to “ultimate”. These fields are used in the Web of Trust model, explained in Section 7.7. , and are related to public keys purporting to other people.
- The *Fingerprint* is a unique 40-digit hexadecimal number automatically generated by applying a hash algorithm to the key. The fingerprint unambiguously identifies the key worldwide. This means that no key will have the same fingerprint of another key ever created in the whole world.
- The last 8 digits of the fingerprint make the *Key ID* which, being much shorter, is used to locally identify the key for all practical purposes. While the fingerprint is unique for each key ever created worldwide, you can safely assume the key ID unique for each key in your keyring. (You would need to collect dozens of thousands of public keys in your keyring before there is a good possibility that two keys share the same key ID.)
- The bottom field shows all key components i.e. primary key and subkey(s) along with their properties: the key ID, the algorithm used for the key, the key size in bits, the creation date, and the expiration date (if any). This field is called *Subkey* in Enigmail 0.96.0 and *Key Parts* in later versions; the primary key is not shown in Enigmail 0.96.0.

Key ID and fingerprint are often prepended by the characters 0x, which is the prefix indicating a hexadecimal number.

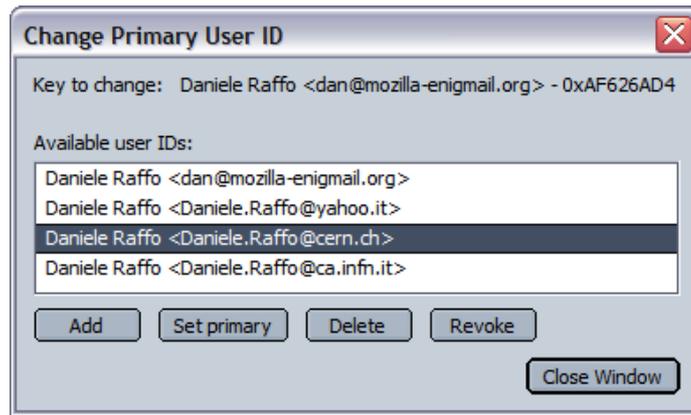
One or two rows of buttons at the bottom of the Key Properties window allow you to perform different operations on the key. (In Enigmail 1.0.0, these buttons are grouped as menu items inside a drop-down menu called *Select action...*) The operations you can perform on a key depend whether you have or not the companion private key, so you can accomplish some operations on your key pair only and not on other people's public keys; Enigmail shows only the buttons (or, in v1.0.0, menu items) relevant to the permitted operations. The same operations are available from the Key Management window as well.

Now hit the *Close* button to close the Key Properties window and go back to Key Management.

7.3.2. Specifying multiple user IDs

You might desire to use more than one email address to send secure email from. In this case, you do not need to generate one key pair for each address: you may simply associate multiple email addresses to your key pair. This will save you from the burden of managing multiple key pairs.

Select your key pair and choose *Edit* → *Manage User IDs* from Key Management, or choose *Manage User IDs* from the Key Properties of your key. A window will pop up to show you a list of all user IDs (primary user ID and all additional user IDs) that are currently associated with the key. If this is the first time you perform this operation, your key will have only a primary user ID.



The *Add* and *Delete* buttons add and delete other user IDs. An user ID is composed of a name and email address; it is also possible to put an optional comment.

The *Set primary* button sets the selected user ID as primary, and as a consequence the previous primary user ID is relegated to the role of additional user ID.

The *Revoke* button revokes the selected user ID, which is then greyed out and deactivated. The difference with deletion is that a revoked user ID is still associated to the key pair but no longer usable.

Click on *Close window* to save the changes you have made to the key and to return to Key Management.

All user IDs are included in the public key when it is sent or exported. There is no mechanism in OpenPGP for selecting which user IDs are included and which not. This also applies to the PhotoID, which is a special type of user ID.

7.3.3. Adding a PhotoID (via GnuPG command line only)

Public keys may have an embedded PhotoID i.e. the owner's photograph. To view the PhotoID, select the key and choose *View* → *PhotoID* from Key Management, or click the *View PhotoID* button from the Key Properties. If the menu item or button are greyed out, the key carries no PhotoID.



It is not currently possible to add a PhotoID from Enigmail, but you can do so from GnuPG command line. Assuming that 0x89ABCDEF is your key ID, type at the command prompt:

```
gpg --edit-key 0x89ABCDEF addphoto
```

Then enter the path of the image file. Suggested image sizes are 240x288 pixels (GnuPG) or 120x144 pixels (PGP).

Another possibility is to use Windows Privacy Tray, or WinPT for short, to manage your key pair and add a PhotoID from there. WinPT is a Windows front-end that provides a graphical interface for GnuPG; it is free and open-source, and you can download it from <http://winpt.gnupt.de>.

The image will be ASCII-armored and included in your public key as an additional user ID. For this reason, GnuPG and PGP recommend to use JPEG files of maximum 7 Kb, otherwise your public key may become very big in size.

7.3.4. Changing the passphrase

To change the passphrase that protects your private key, click on your key and choose *Edit* → *Change Passphrase* from Key Management, or click the *Change Passphrase* button from Key Properties.

7.3.5. Self-signing your key

You should self-sign your key after you've finished changing its properties and are happy with it. Self-signing ensures that nobody can tamper with the properties of your public key. To self-sign your key, click on your key and choose *Edit* → *Sign Key* from Key Management, or click the *Sign Key* button from Key Properties.

You may also sign other people's public keys in order to build a Web of Trust, although this is not required for basic Enigmail operations. This will be explained in Section 7.7.1.

7.3.6. Making a backup

Once you've generated your key pair, verified its properties, and self-signed it, it's a very good idea to make a backup copy of it.

Select your key pair from Key Management and click on *File* → *Export Keys to File*. You will be asked whether you want to include your secret key in the saved OpenPGP key file:



If you now click on *Export Secret Keys*, the exported file will contain your whole key pair (secret key and public key). If you click on *Cancel* instead, the exported file will contain your public key only.



Be careful on how you export your key pair!

When backing it up, you **must** include the secret key, or the backup will be useless. When exporting in order to distribute your public key (as will be explained in Section 7.4.1.), make sure you do **not** include the secret key, or you'll compromise your key pair.

Since you're making a backup, click on *Export Secret Keys*, then choose where to save the file. Your key pair will be saved as an ASC file named like that:

```
Firstname Lastname you@domain.com (0x89ABCDEF) pub-sec.asc.
```

As you see, the filename is composed of the primary user ID of the key, the key ID, and the word “pub-sec” indicating that this is a key pair. Store this file in an external hard disk, keep it in a safe place and don't pass it to anybody.

7.4. Distributing your public key

Now that you have a key pair, you must distribute your public key around. This is necessary so people can use it to send you encrypted messages and verify the signature on messages you send. You can do this manually and/or by the means of a public keyserver.

7.4.1. Share your public key manually

On the Key Management window, select your key pair and click on *File* → *Export Keys to File*. Again, you will be asked whether you want to include your secret key in the saved OpenPGP key file: make sure you click on *Cancel* this time. Save the ASC file, which will have a name like

```
Firstname Lastname you@domain.com (0x89ABCDEF) pub.asc.
```

This is a copy of your public key; notice the “pub” word at the end of the filename.

You can now put the file on your website for people to download it, or carry it around in a USB drive to distribute it to people, or send it via email as an attachment.

Concerning this last option, there is a simple Enigmail shortcut for it: when writing your mail in the Message Composition window, simply choose *OpenPGP* → *Attach my Public key* and you will see your public key appear as an attachment of the message you're composing.

Note: if you use Enigmail 0.96.0, this command does not work well due to a bug. Instead, choose *OpenPGP* → *Attach Public key...* (which allows you to attach to the message any key of your public keyring), then in the Key Selection window select your own public key and click *Ok*.

7.4.2. Publish your public key on a keyserver

By far, the easiest way to let the world know your public key is to publish it on the public keyserver network, a global database of keys.

Some people believe it's a bad idea to use a keyserver, because spammers search them for email addresses. While this is true, the amount of spam you risk to receive by doing this is quite small: you receive roughly ten times more spam when you put your email address on a web page or post a message on the Usenet.

In order to publish your key, click on your key and select *Keyserver* → *Upload public keys*. You will be given a choice of keyservers; *pool.sks-keyservers.net* is the one Enigmail uses by default. Major keyservers periodically synchronize themselves.



Click on *Ok* and wait for your key to be uploaded. Your key is now published on the Internet for anyone to find.



Once a key is uploaded to a keyserver it will stay there forever.

There is no way to delete a key from a keyserver. You can disable the key by revoking it, but anyone in the world will still be able to view all your user IDs (your name and email addresses), **including your PhotoID**. If you have privacy concerns about this, don't upload your key in the first place.

7.5. Revoking your key pair

In the unfortunate event that you lose your key pair or feel it has been compromised, you should revoke it.

You can use the revocation certificate that you generated in advance (or at least you should have done so) to invalidate the key pair. Select *File* → *Import keys from file* from the Key Management window and choose the ASC file containing your revocation certificate.

If you did not generate the revocation certificate in advance, you can revoke the key on the fly, provided that you still have your key pair and you remember your passphrase. To do so, select the key pair you want to revoke and click on *Edit*

→ *Revoke key*. This effectively creates a revocation certificate and imports it in one shot. Note that this command does not work in Enigmail 0.96.0 due to a bug.

Send the revoked key to your contacts to warn them not to use it any more. If you published your public key on a keyserver, remember to upload again the revoked key to it.

7.6. Importing public keys

You need someone's public key to accomplish two things: encrypt a message destined to this person, or verify the signature of a message coming from this person.

You might obtain the public key as an ASC file from the person himself, either as an email attachment or by hand. In this case save the file on your machine, then from Key Management select *File* → *Import keys from file*. Choose the ASC file you just saved. Enigmail will add this public key to your keyring.

Another way to find someone's public key is to download it from a keyserver. Select *Keyserver* → *Search for keys* and insert as the search term a part of the name or email address of this person. You may also search for key IDs. The keyserver will return a list of public keys that match. Tick the checkboxes on the left of any key you would like to import and Enigmail automatically will do it for you.

Later, you can also refresh your public keyring in part or as a whole by the menu commands *Keyserver* → *Refresh Selected Public Keys* and *Keyserver* → *Refresh All Public Keys*.

You might also find a public key published as raw text, for instance in a personal web page. It will appear like this:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.9 (MingW32)

mQGIEgN3cERBAC+2os2hoACip4EiKxEzv+iVHOWaznOJIGZY9zY4y8C0BhUP++q
ccgO9vgN01vIXApYvJctwX9HFJieNdlsBrWOR69hPBAAbDo+3BbOKwJFYgg8akYnv
tBCOdCNfOFwQs/8XdOH25/Oig+UjKhgxwKjkd1UCj7shdGioXOvj013xwCgZrGa
k2oA9Bne3hW+jUPjJlU4UbkD/i7mbQfFwTgxcXfRfsVDnkmPc+QvKe00ajfRP31o
(... 30 lines omitted ...)
3bRLiE8EGBECAA8FAkgN3cECGwwFCQlmAYAACgkQW9ZLGA9iatRGlACeIyyIBTGj
uhrZRubL8gSdI3HyPzEAnAqWD1g+DvEHyEOZ0/2rUcsj7CT2
=3RyK
-----END PGP PUBLIC KEY BLOCK-----
```

This is also what you see if you open a public key ASC file with a text editor.

Select the whole block of text, including the lines `-----BEGIN PGP PUBLIC KEY BLOCK-----` and `-----END PGP PUBLIC KEY BLOCK-----` and copy

it to the clipboard (*Ctrl+C* under Windows). Then choose *Edit* → *Import keys from Clipboard* to import this public key into your keyring.

You can search and add public keys to your keyring at any time. Enigmail will also offer to do so automatically when you receive a signed message from someone and you do not have his public key; we'll see this in Section 8.2.3. If you followed all instructions up to here, by now your keyring should contain your own key pair and a number of public keys purporting to other people.

7.7. Validity of public keys

Importing a public key from a keyserver is quick and easy, but it does not guarantee that the key really purports to the person specified as the user ID. After all, anybody could have uploaded that key.

Furthermore, if you received someone's public key via email, you should reflect on the fact that there is an inherent security problem in using the same channel (e-mail) both for key distribution and for the exchange of messages secured by that key. Theoretically, an attacker that is able to compromise the channel can replace the public key in transit with a rogue public key of a key pair he created himself (*man-in-the-middle attack*). The attacker can now intercept the message that was encrypted with the rogue public key, and decrypt it since he owns the companion private key.

A solution to this problem is to check the public key's fingerprint with the key owner through a different channel. You may phone the key owner and have him read the key fingerprint to you. If the fingerprint does not match, you both know that the key was replaced in transit.

This procedure is safe but cumbersome whenever, as is almost always the case, you do not know personally the key owner or if you have several keys in your public keyring. This problem was therefore firstly addressed in PGP by developing a trust delegation model called *Web of Trust*.

7.7.1. The Web of Trust

In the Web of Trust model, responsibility for key validation is delegated to other people you trust. The trust is expressed in signing other people's public keys. For instance, Alice would use her key 0xAAAAAAAAA to sign Bob's public key 0xBBBBBBBB to certify that particular public key belonging to the individual called Bob. Bob has signed Carol's public key 0xCCCCCCCC. From this, Alice can infer that Carol's public key is valid (i.e. public key 0xCCCCCCCC purports to the individual called Carol) because there is a path of valid signatures from her public key to Carol's.

The *View* → *Signatures* menu item in Key Management, or the *View Signatures* button in Key Properties, allow you to view the signatures attached on a key i.e. by whom this key has been signed.

Participation to the Web of Trust is completely voluntary: you do not need to

sign other people's keys to successfully use GnuPG or Enigmail. To participate, when you receive a public key and have verified both its fingerprint and the identity of the key owner (either because you know him/her or by means of a ID card, passport, driving license...), you sign the key to endorse the ownership of that public key to that person.

You can sign a key by selecting it and choosing *Edit* → *Sign Key* from Key Management, or by clicking the *Sign Key* button from Key Properties. A window will pop up, asking you how carefully have you verified the identity of the key owner. Choose an answer and click on *OK*. Your signature will then be attached to that public key; if the key was already signed by other people, your signature will be added to the list. When a key is exported, the list of signatures is exported with it.

Once you have signed a public key, you should return it (for instance in a signed email message) to the owner so he can redistribute it and upload it again on a keyserver.

Note that you can upload your public keyring to a keyserver by the means of the menu command *Keyserver* → *Upload Public Keys*, but this is not considered good PGP netiquette: only the owner of a key pair should upload his public key.

In many cases you will want to perform a local signature only to mark keys on your keyring as valid, without having them checked carefully. This is done by ticking the option *Local signature (cannot be exported)* in the Sign Key window. In fact, you should only sign keys as non-local (exportable) if you have carefully checked the identity of the owner and the ownership of the key, as already said, and that you intend to send the key back to the owner once you have signed it.

7.7.2. Trust levels

In addition to this, it is possible to subjectively decide the level of trust assigned to a particular key signer. In the previous example, Alice could decide that she does not trust Bob, because he is known for happily signing any public key he gets his hands on without caring to verify the owner's identity. In this case she sets the trust level of key 0xBBBBBBBB (the key Bob uses to sign other people's public keys) to None.

Here, “trust” refers solely as Bob's capacity to properly validate public keys. It does not infer anything else concerning Bob as a person, such as his trustworthiness, his being a law-abiding citizen, or any of his moral qualities. It does not concern, neither, the content of Bob's messages being truthful or not.

There are five levels of trust:

- **Unknown.** Nothing can be said about the owner's judgement in key signing. This is the trust level initially associated to other people's public keys in your keyring.
- **None.** The owner is known to improperly sign keys.
- **Marginal.** The owner is known to properly sign keys.
- **Full.** The owner is known to put great care in key signing.
- **Ultimate.** The owner is known to put great care in key signing, and is allowed to make trust decisions for you.

You can set the level of trust of a particular key by selecting that key and choosing the option *Set Owner Trust* from Key Management, or from Key Properties itself.

You alone decide which level of trust to assign to a key, and the trust is assigned only locally. This is considered private information: it is not included with the key when it is exported, and is stored in a separate place from your keyring.

You should set the trust level of your own key pair to the maximum (*I trust ultimately*).

7.7.3. Criteria for key validity

The trust level is used to calculate the validity of a key. Being the trust level only locally significant, as a consequence the calculated validity of a key is only relative to your local installation of OpenPGP. That is, a key you consider valid might be considered invalid by someone else.

A key is considered valid if it has been signed by a fully trusted key, or by at least 3 marginally trusted keys. In addition to that, the path leading from that key back from your own key must be 5 steps or shorter.

Note that, as a consequence, keys signed personally by your own key are always considered valid.

Ultimate trust bypasses any restrictions on how many fully trusted signatures are needed to make a key valid: any key signed by an ultimately trusted key is always valid. You can set ultimate trust to a particular key you want to allow to make trust decisions for you.

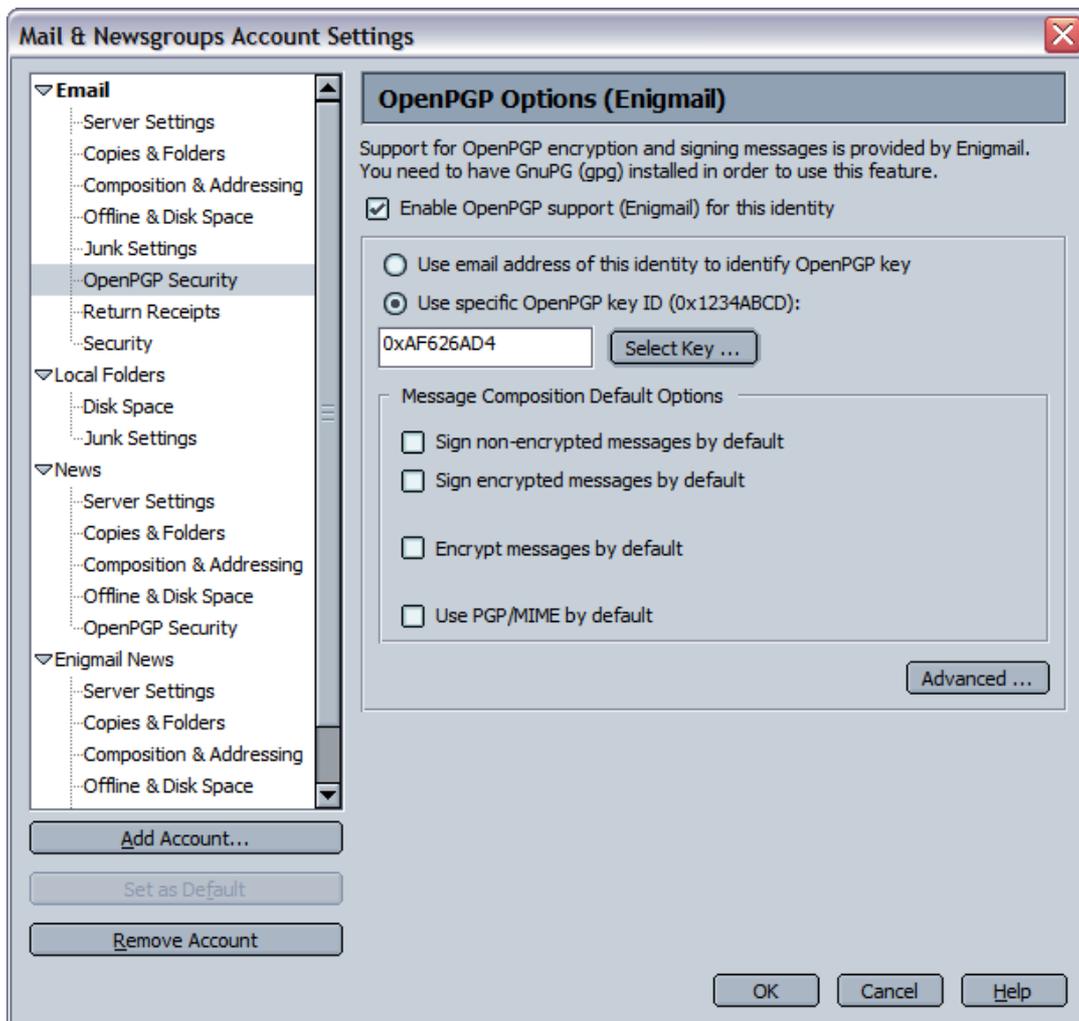
You can see the calculated validity and trust value in the Key Management or Key Properties window, in the fields *Key validity* and *Owner trust*.

8. SIGNATURE AND ENCRYPTION

You have generated your own key pair and have imported other people's public keys, so you are now able to exchange secure emails with them. But first, you must ensure that your account is correctly set up to use Enigmail capabilities.

8.1. Account settings

Launch your email client and choose *Edit* → *Mail & Newsgroups Account Settings...*, and then click on *OpenPGP Security* from the email account settings:



If you have multiple identities enabled, you can (and should) set these OpenPGP options independently for each identity. You will do this from the Identity Settings window, which after the Enigmail installation will have a new *OpenPGP Security* tab with the same options as above.

When configuring the OpenPGP options for your email account, first make sure the option *Enable OpenPGP support (Enigmail) for this identity* is turned on. This is necessary in order to send signed or encrypted email on behalf of this account, and to configure Enigmail. This does not disable signature verification and decryption, which is account-independent.

You need to let Enigmail know which key to use with this account. By choosing *Use email address of this identity to identify OpenPGP key*, Enigmail will automatically select the key pair which lists amongst its user IDs the email address associated with this account. Do not select this option if you have more than one key pair with the same user ID.

Alternatively, you could explicitly specify a key pair by choosing *Use specific OpenPGP key ID (0x1234ABCD)*: then enter in the field the ID (prepended by 0x) of the key pair you want to use, or simply click on *Select Key...* and select the desired key pair.

You can then set some default options when composing a message from this email account:

- *Sign non-encrypted messages by default*: enables signing automatically if encryption is not enabled at the same time.
- *Sign encrypted messages by default*: enables signing automatically if encryption is enabled at the same time.
- *Encrypt messages by default*: always tries to encrypt messages.

These will be the default options, unless modified manually. If you change the identity while composing a message, signing/encryption will be activated or deactivated according to the above options for the chosen identity, unless you have modified the signing or encryption status manually.

By default, Enigmail uses a standard called inline PGP for signed and/or encrypted messages. Checking the option *Use PGP/MIME by default* tells Enigmail to use the PGP/MIME standard instead of inline PGP. PGP/MIME is defined in RFC 3156 and offers additional features such as attachments encryption together with the message body, and support for encryption of messages that use HTML format and special character sets. Unfortunately PGP/MIME is not supported by all mailclients; those that currently are compatible with it are Enigmail, Apple Mail, Becky, Evolution, KMail, Mulberry, Sylpheed, and The Bat!.

You can click on the *Advanced...* button to set some advanced options. The first two options add a new mail header in the message, containing OpenPGP information about your key:

- *Send OpenPGP key ID* adds the mail header `OpenPGP: id=key_id` which mentions the key ID specified in *Use specific OpenPGP key ID*.

- *Send URL for key retrieval* adds the mail header `OpenPGP: url=url` which mentions the URL from where your public key can be retrieved. If you enable this option, you must also specify the URL by typing it into the field. This header actually is not used for any purpose of key retrieval, e.g. even if it is set Enigmail won't fetch the public key at the specified URL.

Finally, *Attach my public key to messages* automatically attaches your public key to any message you send.

(Versions of Enigmail prior to 1.0.0 are affected by a cosmetic bug which makes the advanced options window transparent and its font incoherent with the rest of the GUI. This does not improve its functionality in any way.)

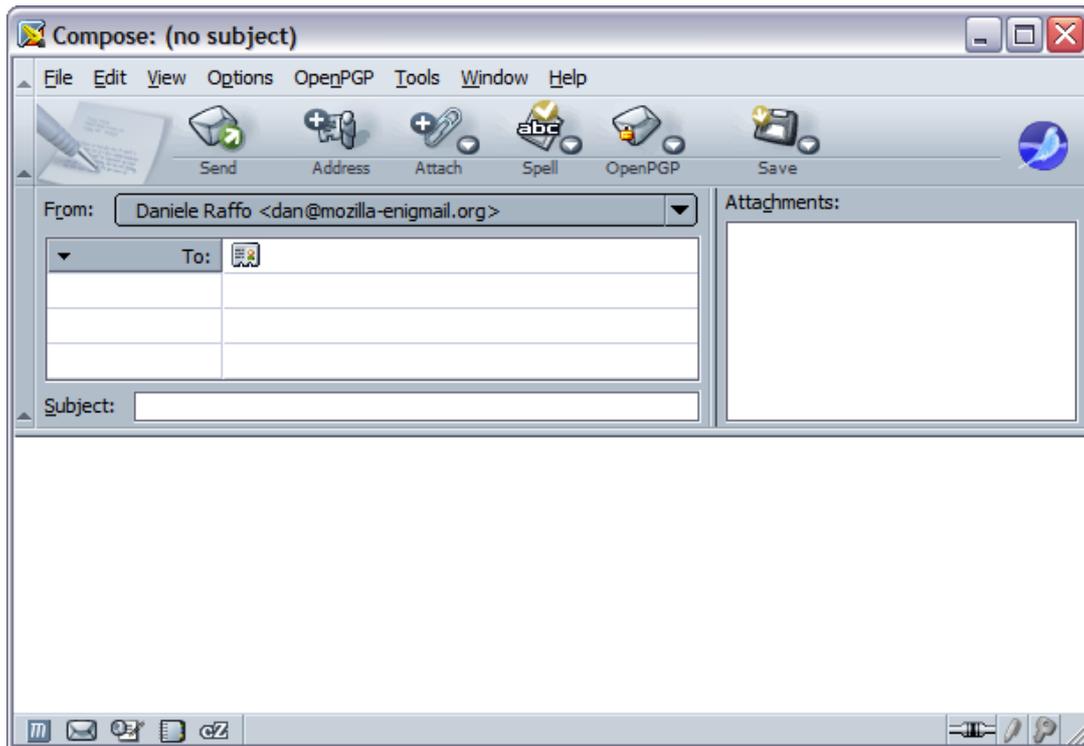
While you're in Account Settings, it is a good idea to ensure that the HTML format for outgoing messages is disabled. You can check that by choosing *Composition & Addressing* and make sure *Compose messages in HTML format* is unchecked. This because Enigmail does not handle HTML messages well, and should be used with plain text messages only. In fact, even netiquette recommends using the plain text format for email messages.

If you insist in leaving HTML as default, remember to disable it by hand every time you want to sign or encrypt the message you're writing. The *Compose* button in SeaMonkey 2.0 works as a drop-down menu from which you can choose on-the-fly whether to compose your mail in HTML or plain text. In Thunderbird and previous versions of SeaMonkey, if you have HTML enabled by default you can disable it for the current message by holding down the Shift key when clicking on the button used for message composition (*Compose* in SeaMonkey, and *Write* in Thunderbird).

8.2. Signature and verification

8.2.1. Signing a message

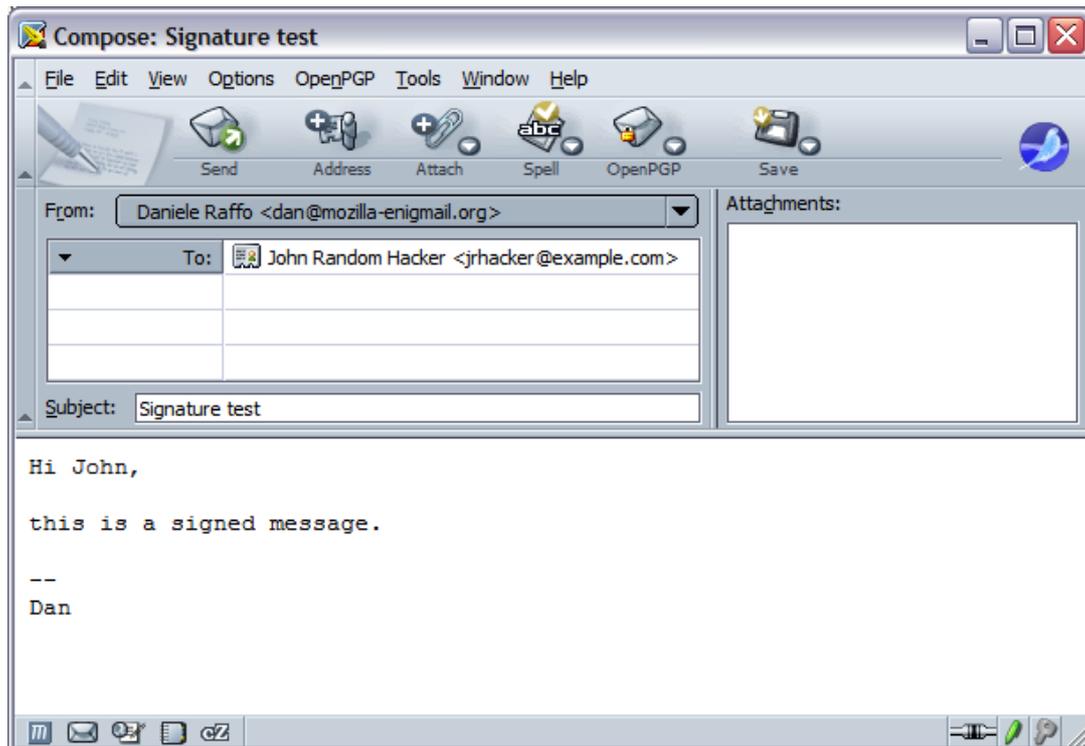
You are now ready to write your first digitally signed email message. From your email client, click on the *Compose* button as you normally would do. You will notice two new gadgets on the Compose window: a *OpenPGP* icon on the toolbar, and small icons of a pen and a key on the bottom right corner.



The *OpenPGP* icon behaves both like a button and a drop-down menu. From there, you can select the options *Sign Message* and/or *Encrypt Message*. You can select the same options from the *OpenPGP* menu item. As you already know by now, you can send a message signed, encrypted, or both encrypted and signed.

The pen icon (which means signature) and/or the key icon (which means encryption) lights up to signal that the relevant option is on. The icons work also as toggle buttons, that is, you can also click directly on the pen and the key icons to respectively toggle signature and encryption.

The following figure shows the composition of a signed message:



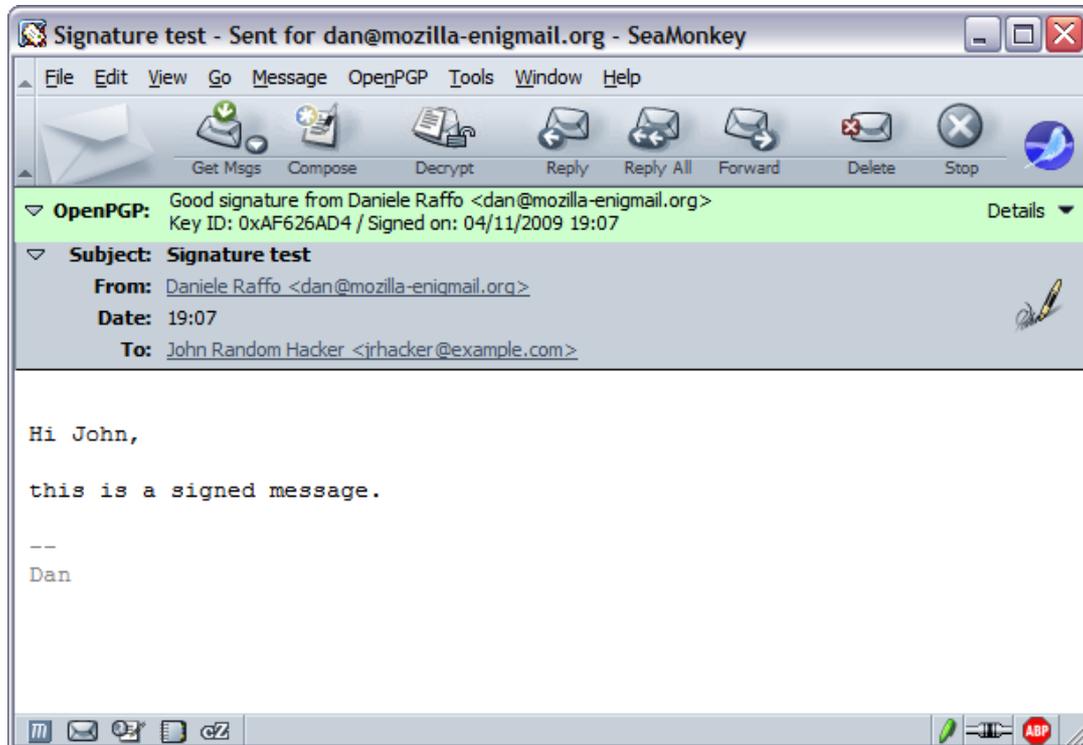
Select the option *Sign Message* and click *Send*. The message will be signed with the key specified in the Account Settings for the account you're currently using (the one shown in the *From:* drop-down menu).

You will be asked for your passphrase, which is necessary for all operations concerning your private key such as signing messages, decrypting messages, and revoking or modifying properties of your key pair. It is also possible to cache your passphrase for a chosen amount of minutes so you won't have to type it every time: this can be set from *OpenPGP* → *Preferences* → *Passphrase settings* and is explained in detail in Section 9.1.1.

At the time of sending, your mailclient might complain that the message you composed contains characters not found in the selected Character Encoding, and ask you to choose between different options. In this case, the best choice is *Send in UTF-8*.

8.2.2. Verifying a signature

Now, if your mailclient is set up so that a copy of outgoing emails is automatically saved in the Sent folder, it is possible to have a look at how the signed message looks like:



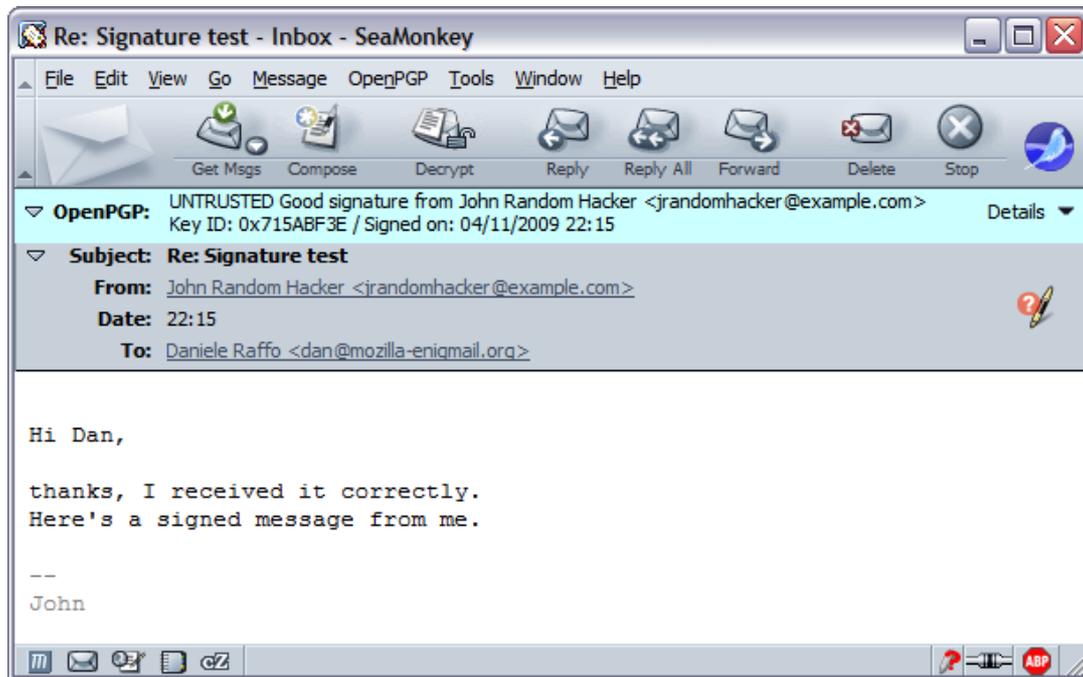
In this Message window, the OpenPGP status bar shows up to indicate that the message is secured with OpenPGP.

The status bar says that the message was correctly signed, and gives information about the sender's key, i.e. the key that was used to sign. This information is the user ID (the identity of the signer i.e. name and email address) and the key ID. The status bar also reports the date and time of the signature. You can expand or shrink the status bar by clicking on the expand button on the left.

The OpenPGP status bar is green to indicate that the sender's key is trusted. In fact, in this case the sender's key is my own key, which has ultimate trust in my own Enigmail environment. Accordingly, a picture of a signing pen is shown near the email headers, and the pen icon is green.

You can have more details about the signing process (in this case, the fingerprint of the signing key) by selecting *Details* → *OpenPGP Security Info...* or simply by clicking on the picture of the pen. *Details* → *Copy OpenPGP Security Info* copies the security information to the Clipboard instead.

Now let's have a look at a signed message I received from `jr hacker@example.com`, assuming I have his public key:



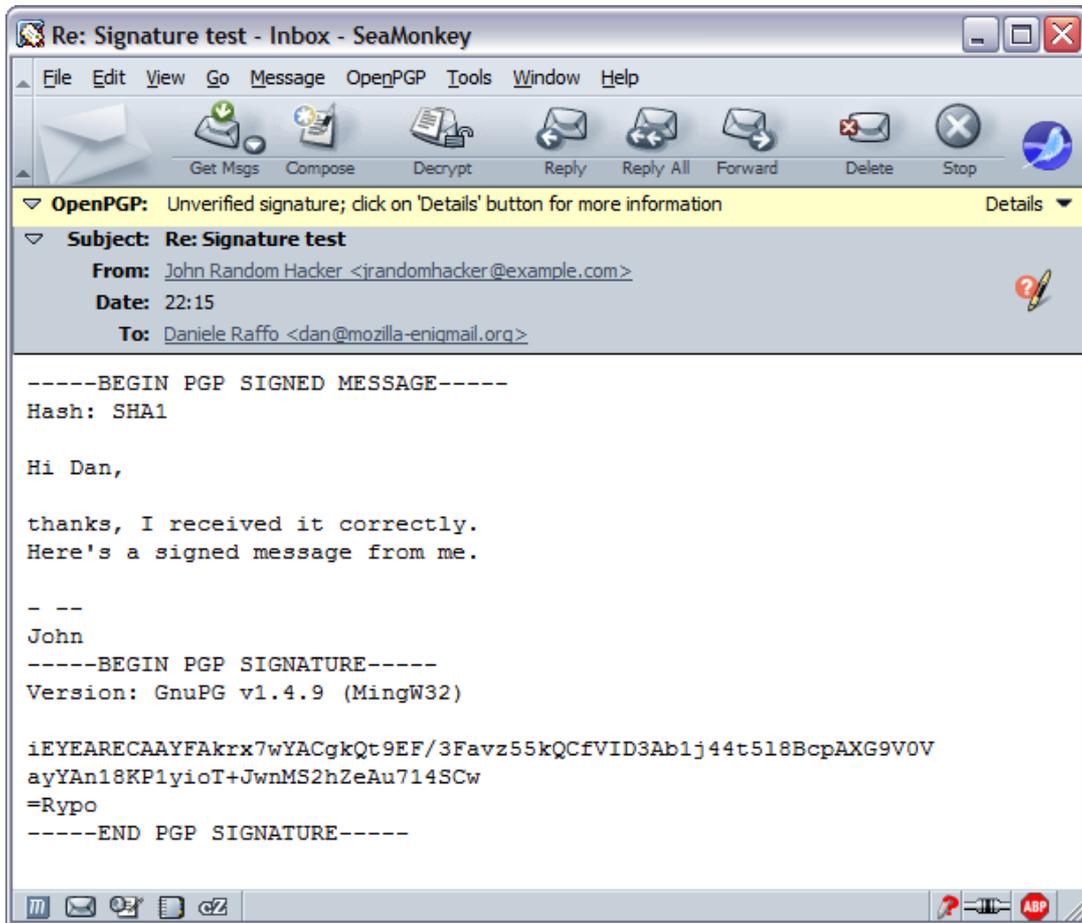
The OpenPGP status bar tells that the signature is valid, although untrusted. This is mostly normal, and just means that John Random Hacker's public key has a trust lower than ultimate in my public keyring, which is the default for freshly imported keys. As we explained earlier in Section 7.7.2. , trust is subjective and related to how we obtained someone's public key. The most important point here is that OpenPGP verified that the message has been correctly signed with the specified public key. Note that the colour of the OpenPGP status bar is now blue. Accordingly, a pen with a question mark is shown near the email headers and in the bottom right corner.

From the *Details* menu you can operate directly on the sender's key:

- *View OpenPGP PhotoID* allows you to see the PhotoID, if any;
- *Sign Sender's Key...* allows you to sign the sender's key;
- *Set Owner Trust of Sender's Key...* allows you to set the sender's key to the desired trust.

These are just shortcuts, you can do the same operations from Key Management as well.

What if I haven't had John Random Hacker's public key? In this case, the message would appear as such:



The message is signed, but the signature cannot be verified at all. This is also how a recipient that does not use Enigmail, nor any other OpenPGP software, will see the message. As expected, the original text is still readable: signature ensures authentication of the sender, not confidentiality of the message.

Note how OpenPGP manipulates the mail when signing. The original message is prepended by a line

```
-----BEGIN PGP SIGNED MESSAGE-----
```

and then it is specified which hash function has been used. Then there is the original message. Finally, the digital signature, embedded within two lines

```
-----BEGIN PGP SIGNATURE-----
```

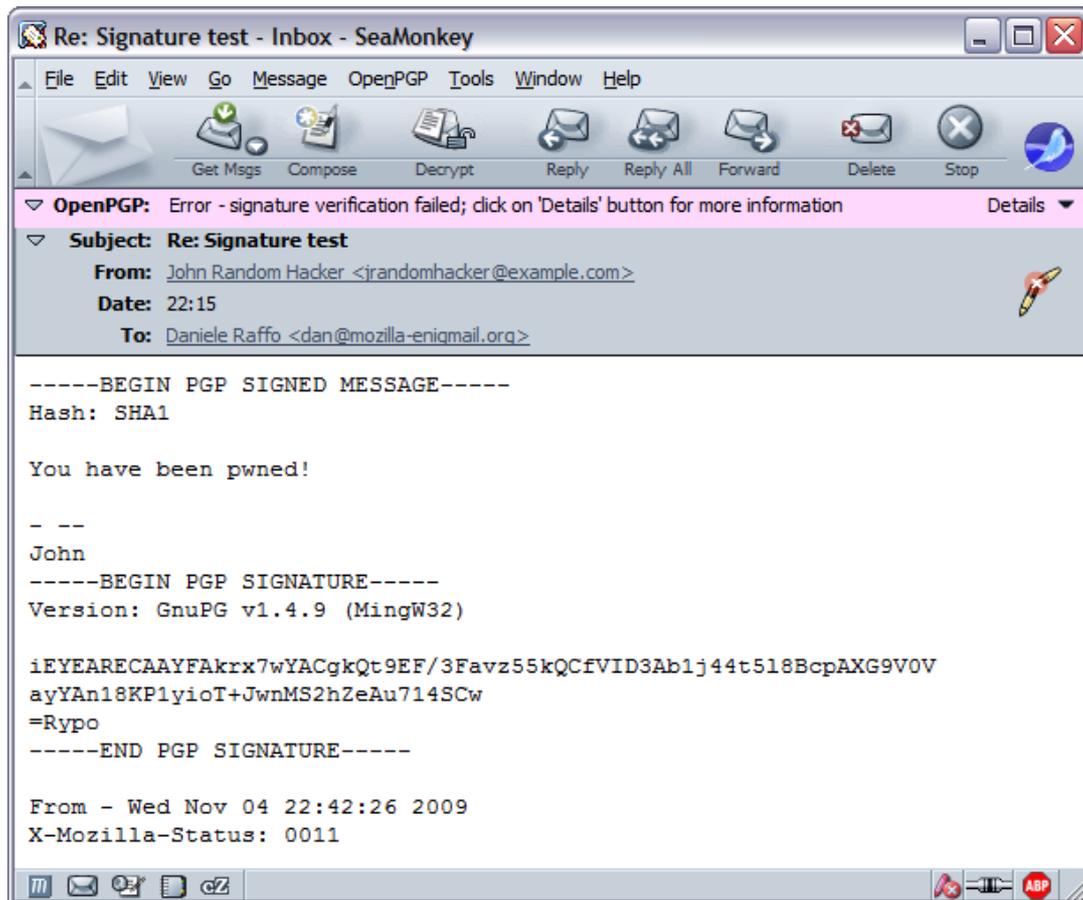
and

```
-----END PGP SIGNATURE-----
```

is appended to the message. Within the signature there is a line that specifies the version of GnuPG used. It is also possible to put an additional GnuPG comment line after the version, by passing additional parameters to the GnuPG command line; this will be explained in Section 9.1.4.

The lines starting with ----- are called *PGP headers*.

Finally, you might receive a mail that Enigmail shows as such:



The signature is invalid, which means that the message has been altered in transit, or the key used for signing is not that of the intended sender. You must therefore not believe it. Notice the purple status bar, the image of a broken pen, and that the pen icon is marked with an X.

8.2.3. Retrieving the key that signed the message

A nice feature of Enigmail is that it can import automatically the public key needed to verify a message. If you received a message for which you don't have the sender's public key, as shown in the figure at page 49, select from the OpenPGP status bar the menu command *Details* → *Import Public Key*, and Enigmail will offer to try to download from a keyserver the key that was used for signing:

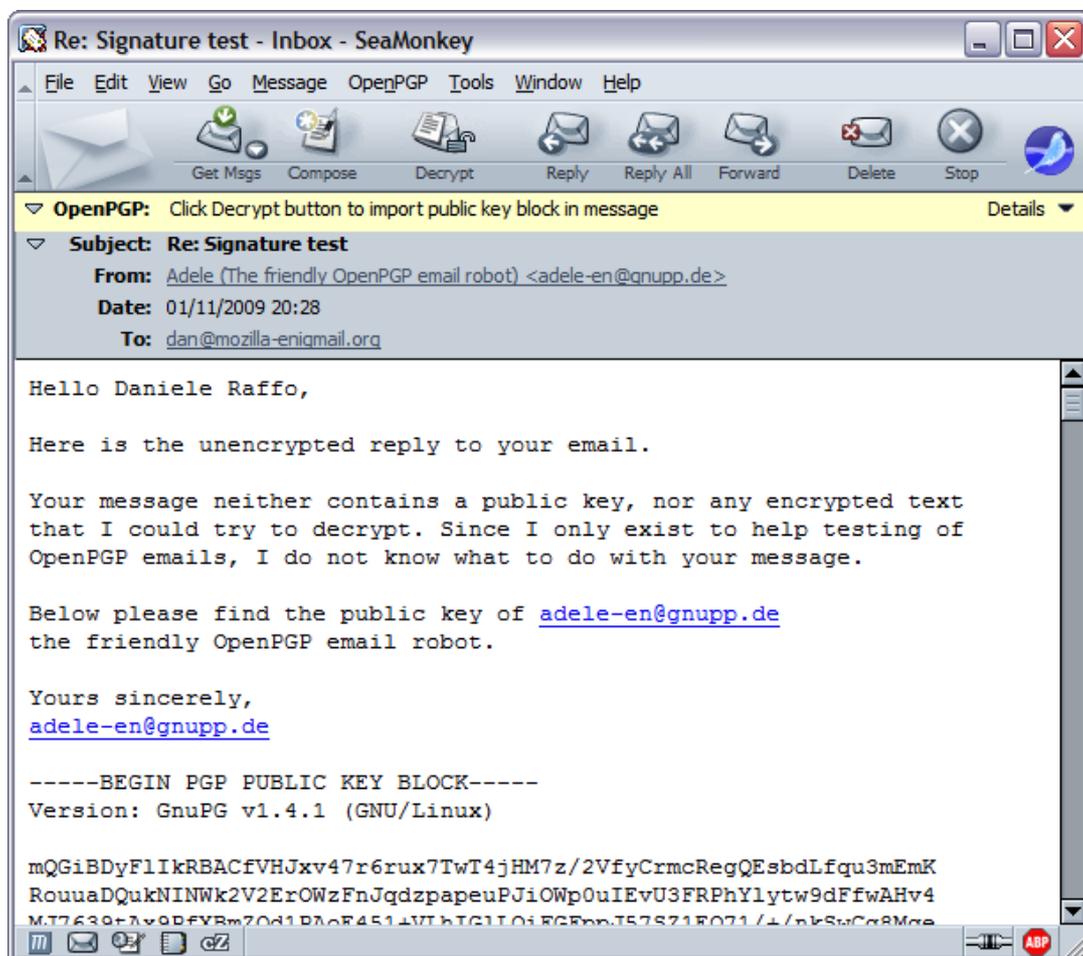


Just click on *Import* and Enigmail will do that for you. The imported key will be added to your keyring.

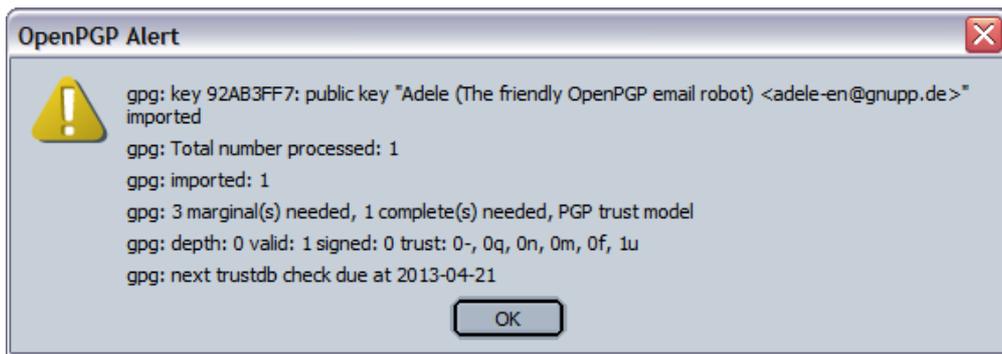
More often, you will receive someone's public key as an ASC file attached to the email message. In this case, importing the public key is just as easy: you only have to right-click on the attachment and choose *Import OpenPGP Key*.

Someone might also send you his public key embedded in the message text. If you want to do some signature (and encryption) tests, then you'll find a very patient correspondent in Adele, "The Friendly OpenPGP Email Robot". Adele can be contacted at adele-en@gnupp.de and is an automated program that is able to receive and understand OpenPGP messages, and to reply to them accordingly in a very short time.

I sent a simple cleartext mail (unsigned, unencrypted) to Adele, and here's how she replied to it:



Here Adele complains that there was no public key attached to my message, so she doesn't know what to do with it. However, she provided me with her public key embedded in the message: note the OpenPGP block in the mail body, and the yellow OpenPGP status bar. Clicking on the *Decrypt* button, and then confirming, will import the public key into my public keyring:



Adele's public key is now in my public keyring.

8.3. Encryption and decryption

Here comes the fun part – exchanging secret messages.

8.3.1. Encrypting a message

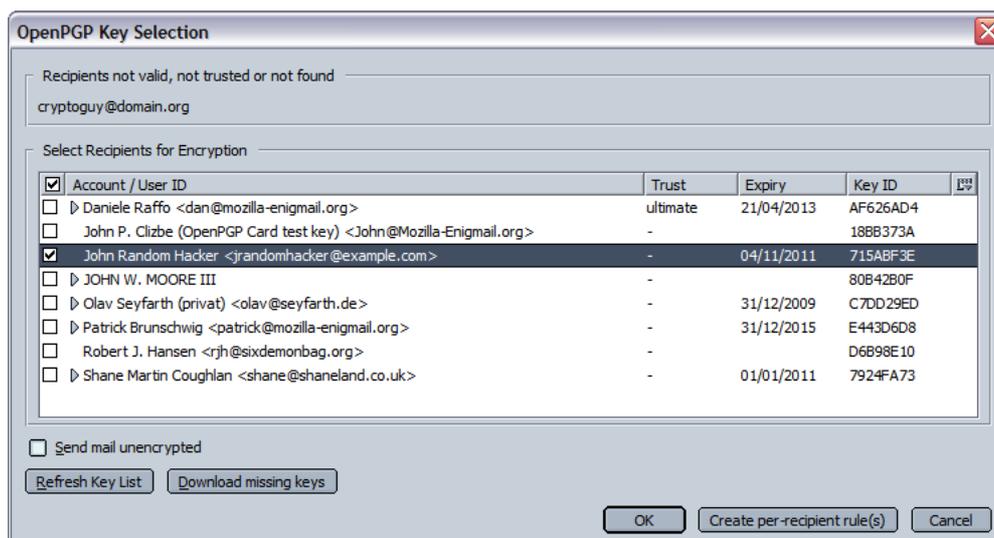
To encrypt a message, select the option *Encrypt Message* before sending, and make sure the key icon in the lower right corner is lit. It is common practice to also sign a message you're encrypting.

To send an encrypted message to someone, you need to have his public key. If you have it, the key is automatically selected: Enigmail searches your keyring and selects the public key that has an user ID that matches the recipient's address. (Note: If you have set per-recipient rules, these will be looked up first. You will learn about per-recipient rules in Section 8.6.)

This is done for each recipient. Recipient addresses are all those specified in the mail headers To:, Cc:, and Bcc:.

Additionally, the message is also automatically encrypted with your own public key, to allow you to read (from the Sent folder) the messages you sent. This setting is regulated by the option *OpenPGP* → *Preferences* → *Sending* → *Add my own key to the recipients list*, and we recommend you leave this option checked.

As you see, this is pretty straightforward. But what happens if Enigmail is unable to select a public key for a recipient, for instance because you don't have it? In this case, Enigmail pops up the Key Selection window to ask you to select the key(s) by hand:



In the figure, I was trying to send an encrypted email to `cryptoguy@domain.org`, which let's imagine is set as an alias and forwards all mails to `jrandomhacker@example.com`. In this case, I would select John

Random Hacker's public key, as shown in the figure, and click *Ok*. The message would then be sent to `cryptoguy@domain.org` encrypted with John Random Hacker's public key.

If I had to send mail to `cryptoguy@domain.org` often, it would be worth creating a per-recipient rule that says "Encrypt all mail that is sent to the address `cryptoguy@domain.org` with the public key associated with address `jrandomhacker@example.com`". This can be done directly from the Key Selection window by clicking the *Create per-recipient rule(s)* button. Alternatively, if John Random Hacker intends to use often his alias address, he should add the user ID `cryptoguy@domain.org` to his public key, and redistribute the updated key.

As you have learnt, a message can be encrypted with more than one public key. In fact, it is usually encrypted with at least two public keys: yours (to let you be able to read a copy of the message) and the recipient's.

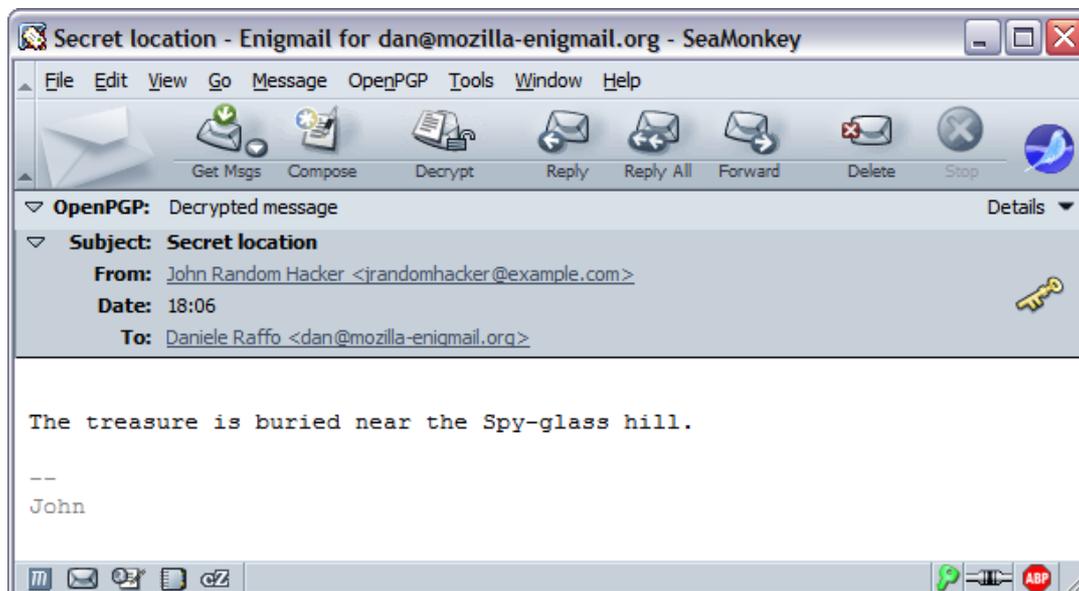
To be more precise, OpenPGP uses hybrid encryption. First it generates a random session key, and encrypts the message with the session key using a symmetric algorithm; then, for each intended recipient, it encrypts the session key with the recipient's public key and adds each encrypted session key to the encrypted message. It then builds an OpenPGP block, which includes an header containing the key IDs and user IDs of any public key the message has been encrypted with. Each recipient then receives the same OpenPGP block.

As a consequence, it is not possible to send to multiple recipients a message that is encrypted for some recipients and unencrypted for others. The message is sent out either encrypted or unencrypted for the whole list of recipients.

That being stated, you should not send encrypted messages to Bcc: recipients, because from the OpenPGP block each recipient is able to tell the identities of the others – hence thwarting the purpose of the Bcc: field. While Enigmail is able to do some workaround to hide the Bcc: recipients from the header, as a side-effect this could block users of other products (e.g. PGP Corp.) from being able to decrypt the message.

8.3.2. Decrypting an encrypted message

This is a message that John Random Hacker sent encrypted to me:



The status of the OpenPGP bar, the key in the headers bar, and the key icon lit green in the corner, indicate that the message was correctly decrypted.

By default, the message is automatically decrypted as it is opened. If you ever want to change this setting, deselect the option *OpenPGP → Automatically Decrypt/Verify Messages*; then you can decrypt messages by hand by clicking the *Decrypt* button in the toolbar.

If I look at the message source (command *View → Message Source* from the mailclient menu) I can see the raw message as it was transferred through the network, and as anyone who is not the intended recipient would see the message – and get a confirmation that the secret has been preserved from prying eyes:

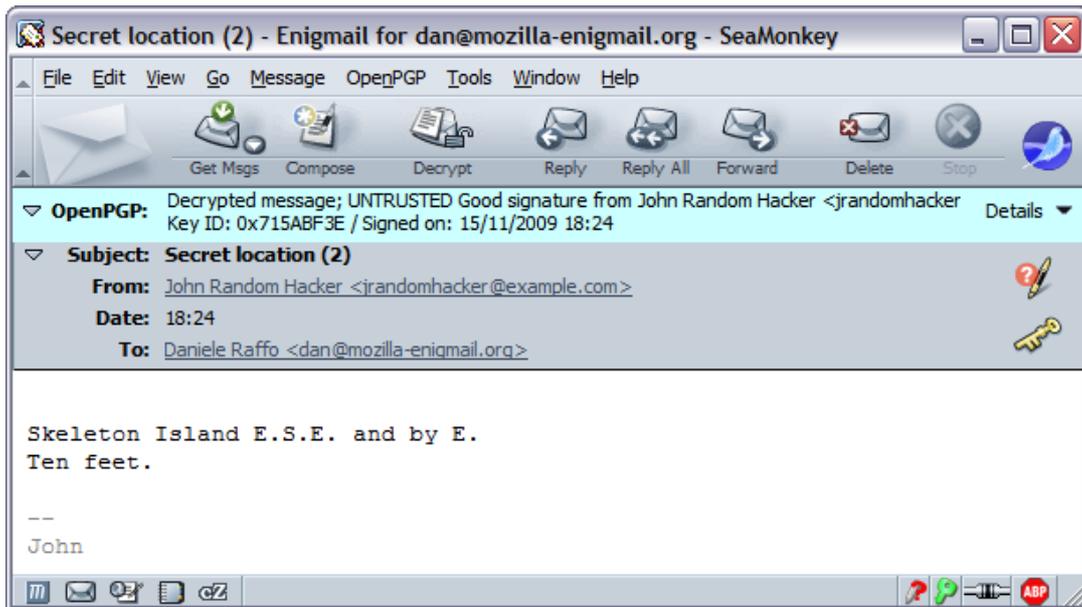
```
(... several mail headers omitted ...)  
Date: Sun, 15 Nov 2009 18:06:10 +0100  
From: John Random Hacker <jrandomhacker@example.com>  
To: Daniele Raffo <dan@mozilla-enigmail.org>  
Subject: Secret location  
X-Enigmail-Version: 0.96.0  
Content-Type: text/plain; charset=ISO-8859-1  
Content-Transfer-Encoding: quoted-printable
```

```
-----BEGIN PGP MESSAGE-----  
Charset: ISO-8859-1  
Version: GnuPG v1.4.9 (MingW32)
```

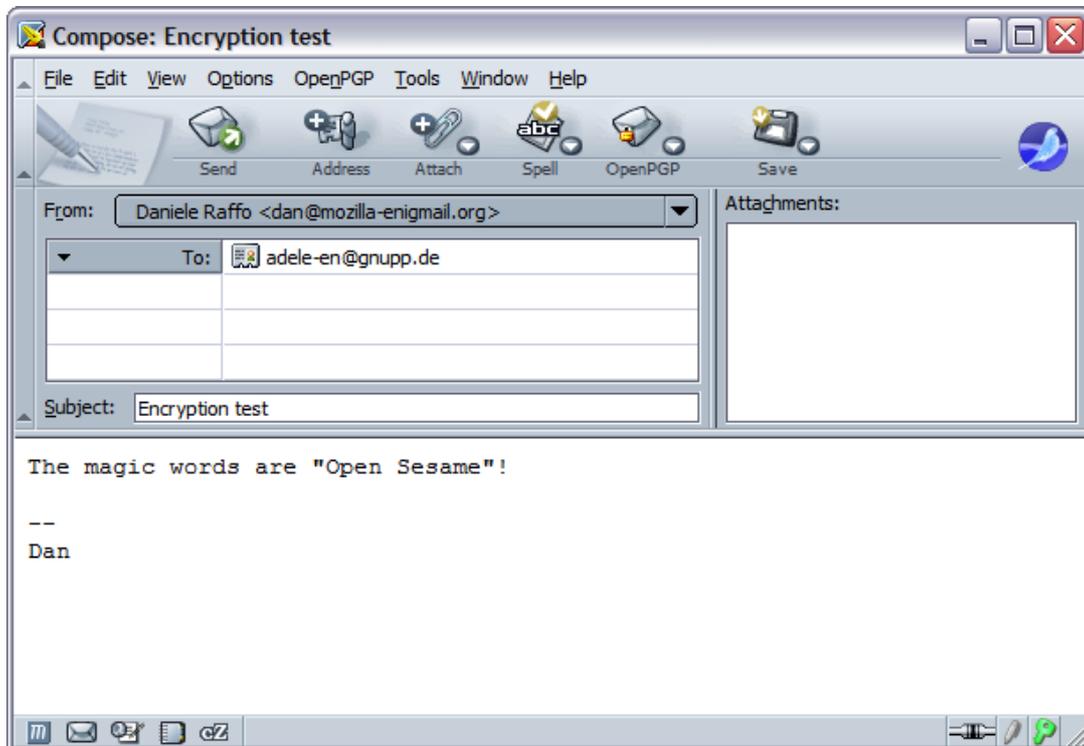
```
hQIOA4xGg4nhqxetEAF/b4Kv/WJyC1MsFkhqgyqm7b1VGXQgN8u03G7PSe5IyIPQ  
zKnEehaz989yofvkLcq3wImHhj+YaSIZ8FJulxyulvQwTwxrmDoRdV+93vjGVbt
```

HTKV+knUvwzBUkLCRW06GaAjOBrV+t0RnJ3yAzEgo/UX+7+wZqng/LIFUVLCCr8
z/cN7CkLBVB2d/qyOXcU7gLq3/EdgHxIe8tqOwnYEugfqDtJp8oQtMUwXiw71X+d
(... 18 lines omitted ...)
TNJ3AYBiN1euvmfekgrZhXKLXGMonBT9tebjIoNqj4Jh3AI91Eg/EUsiufbsqXaQ
IzM2ZBrzjhG5hczgcQUtDfDpRtDAdYhpSAWGKdj/Vpvcj9KXn6pAsS47L22UxO4N
1rMzrWOQVWSURhZtSO8xHKg4/eLAHPatDLM=3D
=3DktOd
-----END PGP MESSAGE-----

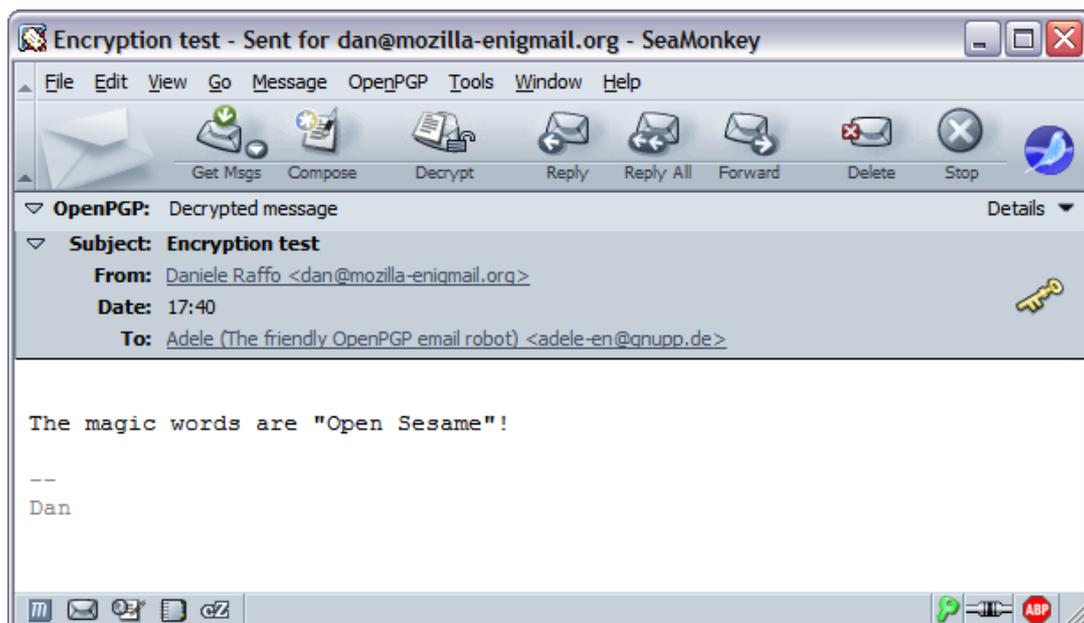
The previous message was encrypted but not signed. Here's how it looks a message that is both signed and encrypted:



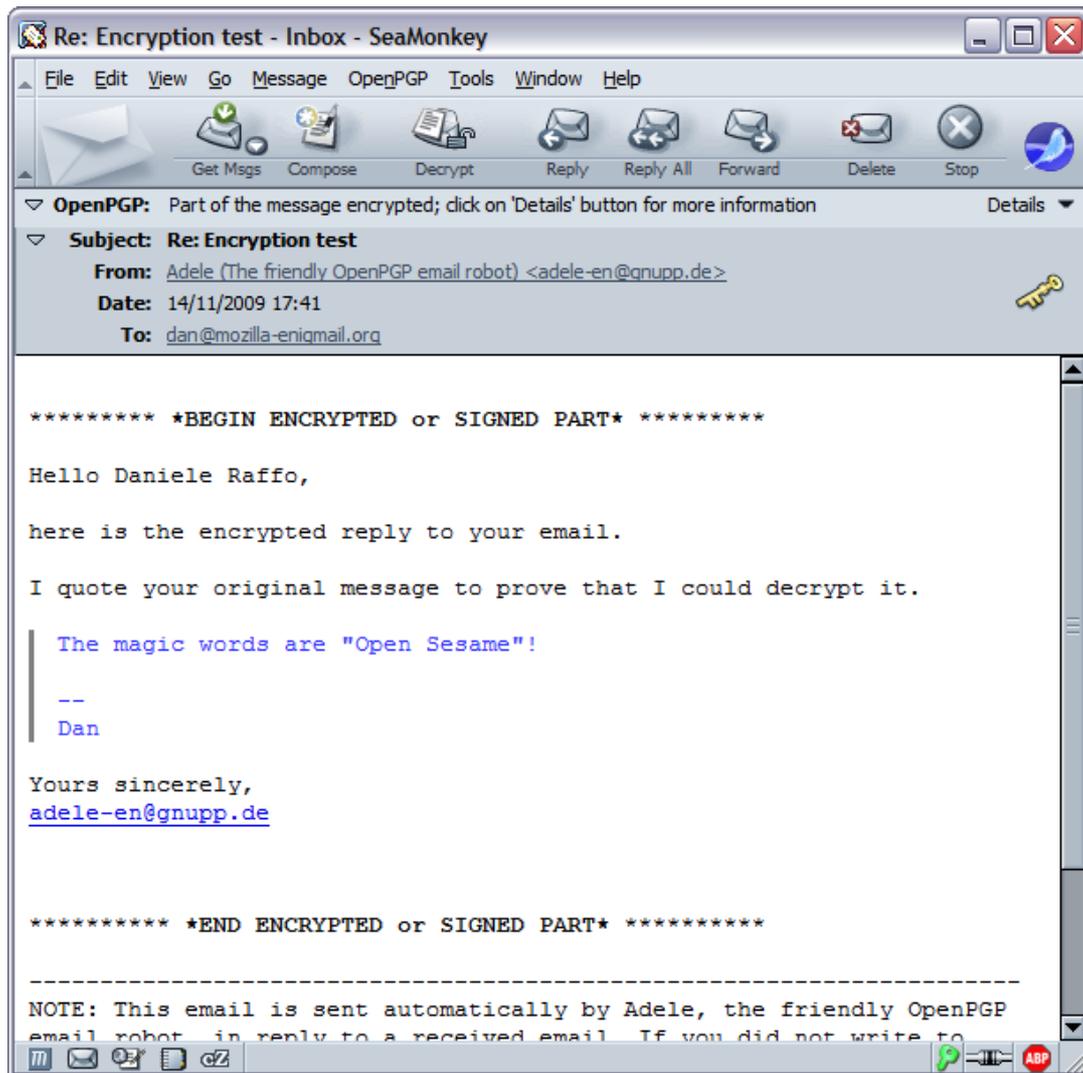
I can use Adele's services to test that my messages are encrypted and decrypted correctly. As you remember, I have imported Adele's public key in my keyring, and I am therefore able to send her an encrypted message:



If I look in my Sent folder there is my message, automatically decrypted as I open it. I am able to read it only because Enigmail, by default, encrypts any outgoing message with the sender's public key too. Shouldn't Enigmail do that, the message would look gibberish to me – even if I was the creator of the message. The next figure shows my own message, correctly decrypted:



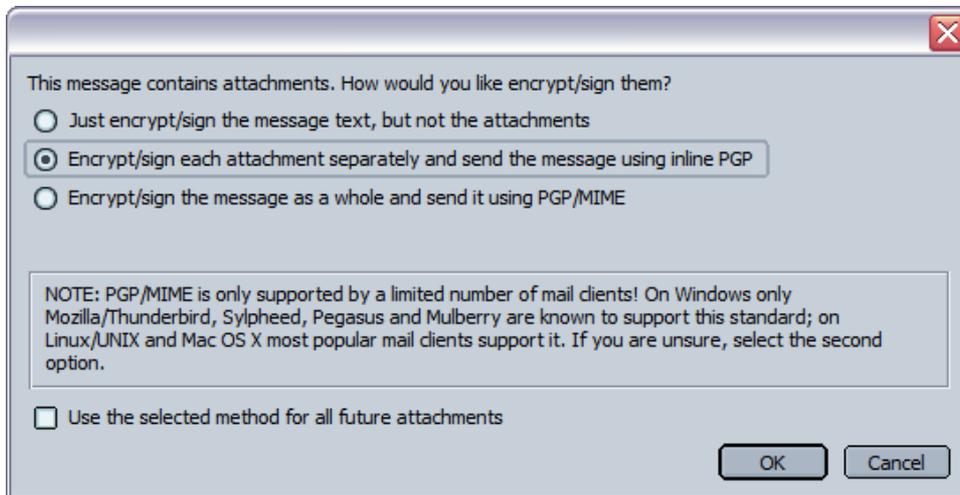
A short time later, I receive Adele's reply:



Notice that the OpenPGP status bar warns that the mail body is partly encrypted: Adele's message is, while the note at the bottom isn't.

8.4. Handling attachments

When sending an encrypted or signed email message that has attachments, you will be given the choice how to encrypt/sign the attachments:



There are three mutually exclusive options:

- *Just encrypt/sign the message text, but not the attachments.* This will only protect the message text.
- *Encrypt/sign each attachment separately and send the message using inline PGP.* In this case, the signature for each attachment `filename.ext` will be stored in an additional attached file `filename.ext.sig`. If the attachment must be encrypted, it will be renamed as `filename.ext.pgp`. This is the default option, which is also the safest and most recommended.
- *Encrypt/sign the message as a whole and send it using PGP/MIME.* As said previously, the PGP/MIME standard is not supported by all mail clients, so the risk here is that the recipient could be unable to read the message.

If you tick the option *Use the selected method for all future attachments*, Enigmail will remember your choice and won't ask you any more. To have Enigmail show you the options again, go to *OpenPGP* → *Preferences* → *Advanced* and click on the *Reset Warnings* button.

When you receive an encrypted attachment, you can view it or save it simply by right-clicking on it and selecting, respectively, *Decrypt and Open* or *Decrypt and Save As...*

8.5. Notes



Mail headers cannot be encrypted, nor included in the signature computation. Do not write any sensitive information in the Subject when sending an encrypted message.

One important point concerns mail header security. Signature and encryption applies to the mail body only – and also to attachments, if you chose so. That is, when you sign a message, no mail header (such as the Subject, Date, all Received headers, etc.) can be included in the signature. In a similar way, when you encrypt a message, mail headers are not encrypted.

Most mail headers (e.g. the Date) are added by the Mail User Agent (mailclient) only after that Enigmail processes the payload; other mail headers (e.g. the Received) are subsequently added by the Mail Transfer Agents at each hop of the path from sender to receiver; finally, other mail headers are added by the Mail Delivery Agent at the endpoint. Even mail headers that are user-set at the time the message is composed (e.g. the Subject) may be legitimately modified by antispam or antivirus applications on the destination server.

This is to say that mail headers change in transit (therefore they cannot be signed), and they must be in cleartext (therefore they cannot be encrypted) in order for the mail message to be processed by intermediary routers and delivered.

This is not a limitation of Enigmail, neither of GnuPG. The OpenPGP standard does not describe a way to sign or encrypt the Subject or, for this purpose, any other mail header.

Thus, please remember that mail headers aren't and cannot be secured, and they could be snooped and forged in transit like any cleartext mail message.

Note that, for the same reason, Enigmail will not sign and/or encrypt a blank message: the message body is empty, so there's nothing Enigmail can process.

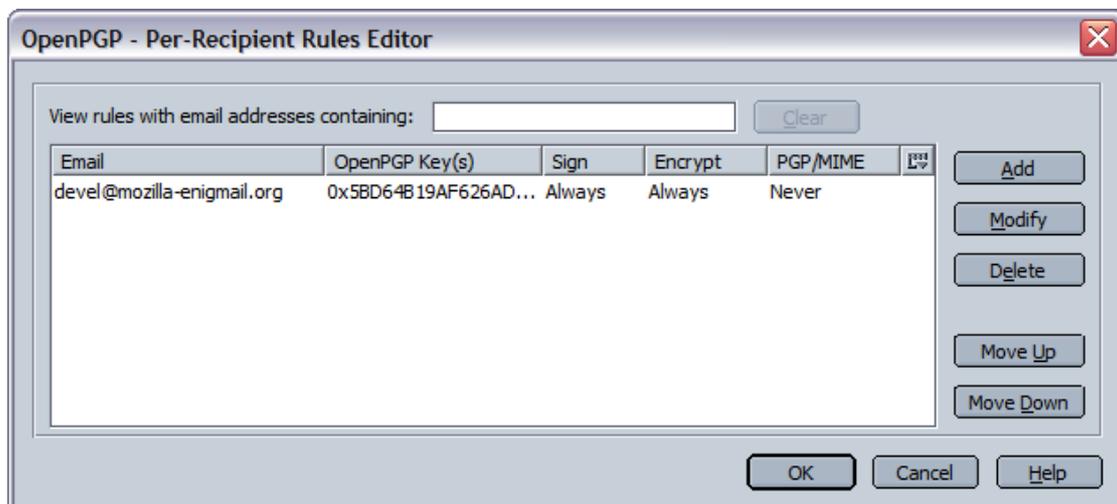
8.6. Per-recipient rules

Enigmail features an advanced per-recipient rule editor that, for any recipient, allows you to specify in advance whether to sign, encrypt, or use the PGP/MIME standard instead of inline PGP.

Per-recipient rules also allow you to specify which key to use for an intended recipient of an encrypted message. By default, Enigmail first searches the per-recipient rules and looks up for a rule matching the recipient; if no rule is specified (as it is the case after a fresh install of Enigmail), Enigmail selects the key with an user ID matching the recipient.

8.6.1. Per-Recipient Rules Editor

To edit per-recipient rules, select *OpenPGP* → *Edit Per-Recipient Rules*. The picture below shows the Per-Recipient Rules Editor window with one created rule.



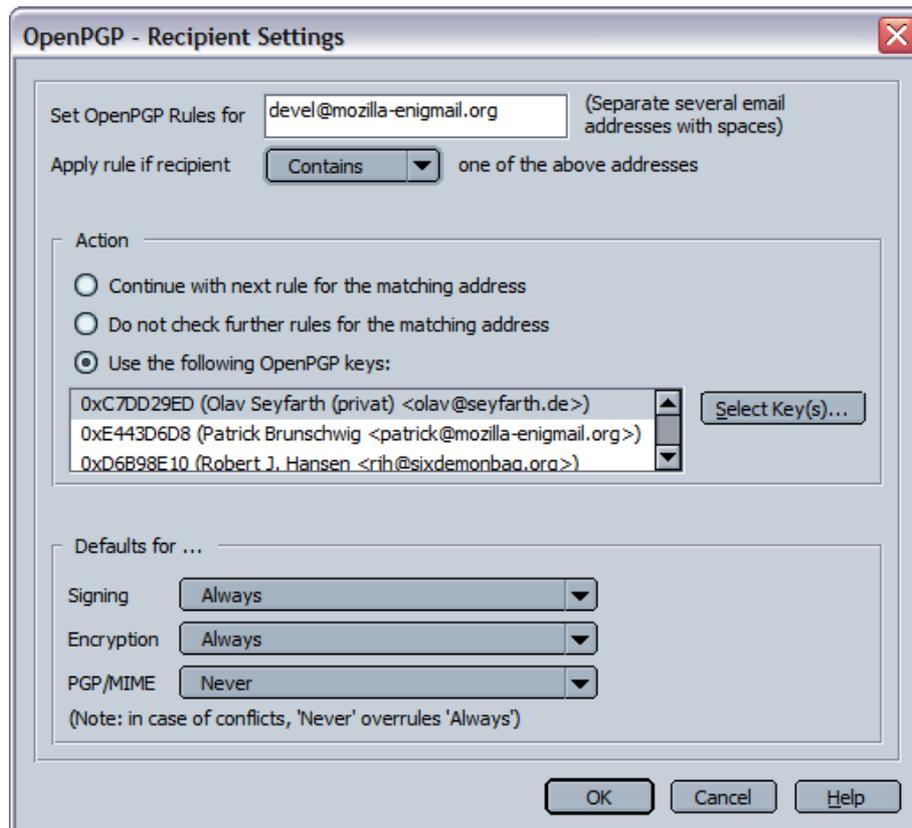
I participate in the mailing list for the developers of Enigmail and, as a habit, all subscribers always sign and encrypt all messages to the mailing list. A mailing list is just a list of physical recipients and as such does not own a key pair, as this would made no sense. Therefore, to encrypt a message for a mailing list, I have to encrypt it with the public keys of all people that subscribed to the mailing list (of course, I need to know who the subscribers are, and I need to have their public keys).

For this purpose, I wrote the above rule, which says: “When sending a message to the address specified in *Email*, always sign it, always encrypt it with all the public keys specified in *OpenPGP Key(s)*, and never use PGP/MIME”.

If you create more than one rule, they are evaluated in order from top to bottom. You can change the rules order by using the buttons *Move Up* and *Move Down*, while *Delete* will delete a rule.

The *Add* button adds a new rule, and *Modify* modifies an existing rule. Clicking these buttons will bring you to the Recipient Settings window.

8.6.2. Recipient Settings



In the *Set OpenPGP Rules for* field you must enter the recipient email address you're writing the rule for. Recipients are the addresses specified in the fields To:, Cc:, and Bcc: of the email message, without distinction. If you want to have a rule for multiple email addresses, enter them all in the field, separated by spaces. Then choose the pattern matching criteria from the drop-down menu (*Is exactly, Contains, Starts with, Ends with*).

In the *Action* zone you specify the Enigmail behaviour if there is a match with the specified recipient email address(es):

- *Continue with next rule for the matching address* allows you to define a rule but to not have to specify a key ID in the *Use the following OpenPGP keys:* field. In this way, the email address is used to check for a key when sending the message. Also, further rules for the same address will be processed as well.
- *Do not check further rules for the matching address* stops processing any other rules for the matching address if this rule is matched. Rule processing restarts with the next recipient.
- *Use the following OpenPGP keys:* allows you to specify which recipient keys will be used for encryption. Use the *Select Key(s)...* button to choose the keys.

In the *Defaults for...* zone you decide whether to activate signing, encryption and PGP/MIME if the rule is matched. Each function can be independently set

to three options:

- *Never* specifies that the function will be off.
- *Yes, if selected in Message Composition* allows you to set the option at the time of message composition.
- *Always* specifies that the function will be on.

When sending a message to multiple recipients, in case of conflicts between rules, *Never* overrules *Always*. For instance, if you created two rules for the following two recipients:

```
alice@example.com
Signing:    Always
Encryption: Always
PGP/MIME:  Yes, if selected in Message Composition
```

```
bob@domain.org
Signing:    Always
Encryption: Never
PGP/MIME:  Never
```

and you try to send a signed and encrypted message to `alice@example.com` and `bob@domain.org`, the message will be signed only. Also, should you have turned on PGP/MIME when composing the message, this setting would have been ignored and the message won't be encrypted with PGP/MIME.

8.6.3. Notes

If you wish to send a message to somebody for whom you did not create a rule, and you wish to manually turn on signing, encryption, or PGP/MIME, this will be overridden by the OpenPGP global settings (*OpenPGP* → *Preferences*), OpenPGP account settings (*Edit* → *Mail & Newsgroups Account Settings...* → *OpenPGP Security*), and per-recipient rules.

A possible workaround is to add a new rule as follows:

- In the *Set OpenPGP Rules for* field, enter @ (this matches any email address)
- Set *Apply rule if recipient* to *Contains*
- Set *Continue with next rule for the matching address*
- Do not add any keys
- Set *Signing, Encryption, PGP/MIME* to *Yes, if selected in Message Composition*
- Save the rule and ensure that it is at the bottom of the rules list.

This rule will always be executed.

You can jump directly to the Recipient Settings, and create a rule for a particular email address, by right-clicking on that address from the Message window or the Address Book and selecting *Create OpenPGP Rule from Address...* from the pop-up menu.

The rules are processed sequentially in the order displayed in the rules editor. If a rule contains an OpenPGP key, the rule is applied, but the address that triggered the match will not be rechecked in any following rules.

In order to minimize the number of rules you have to create, you should set carefully your OpenPGP global and account settings.

How Enigmail chooses the recipient keys for an encrypted message is controlled by the settings in *OpenPGP* → *Preferences* → *Key Selection*. By default, Enigmail first checks per-recipient rules, then the user IDs in your public keyring.

You can disable on-the-fly the lookup of per-recipient rules for a message you're composing by checking the option *OpenPGP* → *Ignore Per-Recipient Rules* in the Message Composition window.

You can enable the option *OpenPGP* → *Preferences* → *Sending* → *Always confirm before sending* in order to check the status for signing, encryption, and PGP/MIME before a message is sent.

Per-recipient rules are stored in a file called `pgprules.xml` located in your profile directory. If you backup/restore your profile manually, you should make sure to include this file.

8.6.4. XML format of per-recipient rules

This Section details the format of `pgprules.xml`, which is a XML file generated by Enigmail and containing the per-recipient rules. This is intended as a technical reference for developers only; normal users should never edit the XML file manually, and can skip this Section.

The `pgprules.xml` file has the following structure:

```
<pgpRuleList>
<pgpRule email='{alice@example.com}' keyId='0x1234ABCD'
sign='1' encrypt='1' pgpMime='1' />
<pgpRule email='{bob@ {user@domain}' keyId='0xCDEF6789'
sign='2' encrypt='1' pgpMime='0' />
<pgpRule email='{mailinglist@domain.org}'
keyId='0x11111111 0x22222222 0x33333333' sign='2'
encrypt='2' pgpMime='0' />
...
</pgpRuleList>
```

Each `<pgpRule ... />` line is a per-recipient rule stating how Enigmail should enable or disable encryption, signing and PGP/MIME and which key to use. The file is processed sequentially; if a rule contains a key ID attribute with some value, the rule is applied, but the address that matched will not be rechecked in any following rule. The attributes are defined as follows.

`email` defines the recipient address(es) to match. Multiple email addresses are separated by spaces. The matching is done on substrings, with curly brackets (`{ }`) defining substring boundaries:

- `{user@domain}` matches exactly and only `user@domain`
- `user@domain}` matches anything that ends with `user@domain`
- `{user@domain` matches anything that starts with `user@domain`
- `user@domain` matches anything containing `user@domain`

`keyId` is the list of key IDs to use for the recipient. The key ID is specified in the 8-byte format (e.g. `0x1234ABCD`) or in the 16-byte format (e.g. `0x1234567890ABCDEF`). Multiple keys are separated by spaces. If a dot (`.`) is the only value in the field, Enigmail does not use a specific key ID and finds the correct key using the email address. Any further rule for this recipient will be ignored.

`sign` specifies message signing, `encrypt` specifies message encryption, and `pgpMime` specifies PGP/MIME use. All these attributes must have one of the following values:

- `0` – Disables the action even if it was enabled in the Message Composition window. This is equivalent to the *Never* option in the GUI.
- `1` – Uses the setting specified in the Message Composition window. This is equivalent to the *Yes, if selected in Message Composition* option in the GUI.
- `2` – Enables the action even if it was not enabled in the Message Composition window. This is equivalent to the *Always* option in the GUI.

When a message is sent to multiple recipients, and multiple rules are applied, the value `0` overrides the value `2`: if one of the rules disables the action, the action will not be applied for the message, regardless of any other rule with value `2`.

9. PREFERENCES

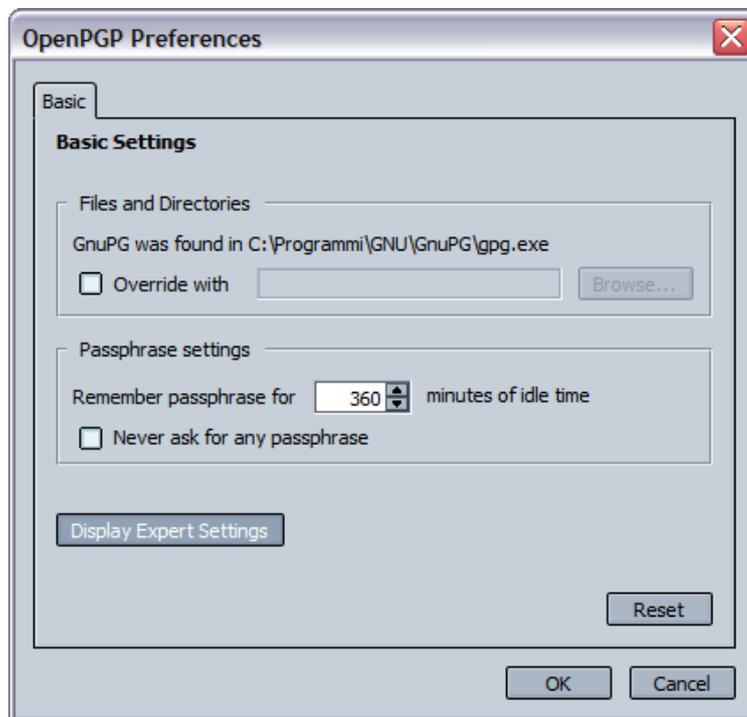
Enigmail can be fine-tuned to tailor your needs. This chapter illustrates the many configuration options of Enigmail.

If you use GnuPG and configured it manually, please note that these preferences will override any similar entry in `gpg.conf`.

9.1. Setting the preferences

9.1.1. Basic

To access the Enigmail preferences, select *OpenPGP* → *Preferences* from the menu bar. (If you use SeaMonkey 1.1 you can also select *Edit* → *Preferences*, which brings up the mailclient general preferences, and then select the *OpenPGP* submenu.) This will initially bring up the Basic preferences, which control the basic functioning of Enigmail.



In *Files and Directories*, it is shown where GnuPG was found. Enigmail tries to locate automatically the GnuPG executable file upon its start. Typical locations are `C:\Program Files\GnuPG\gpg.exe` (absolute path) or `..\GnuPG\gpg.exe` (path relative to the mailclient) for Windows, and

`/usr/local/bin/gpg` for Linux.

If however Enigmail can't manage to find GnuPG, or you want to specify that location manually, tick *Override with* and enter the path to the GnuPG executable file.

Enigmail asks for your passphrase every time it needs to access your private key, for instance whenever you sign, decrypt, or change your key pair properties. It is often cumbersome to have to type the passphrase all the time, and you might be tempted to choose a passphrase that's short and simple to type, which is a bad idea. Instead, you should set Enigmail to save your passphrase in the cache for a fixed time.

You can do this by entering the desired number of minutes in the field *Remember passphrase for [] minutes of idle time*. In the figure, Enigmail will not ask for the passphrase for 6 hours. Entering the value 0 in the field disables password caching.

You will be asked again for the passphrase when one of these events occurs:

- The specified caching time has expired;
- You quit and restarted the mailclient;
- You flushed the passphrase cache via the command *OpenPGP* → *Clear Saved Passphrase*.



Do not leave your computer unattended while the passphrase is stored in the cache.

The passphrase caching mechanism is implemented in a safe way so it is not accessible by other users or processes. However, if the mailclient is swapped to disk, the passphrase is swapped to disk too. If you do not want this to happen, do not use passphrase caching.

If your key pair is not protected by a passphrase, enable the option *Never ask for any passphrase* to prevent Enigmail from asking your passphrase and passing it to GnuPG.

The *Reset* button resets all Enigmail settings to their default value.

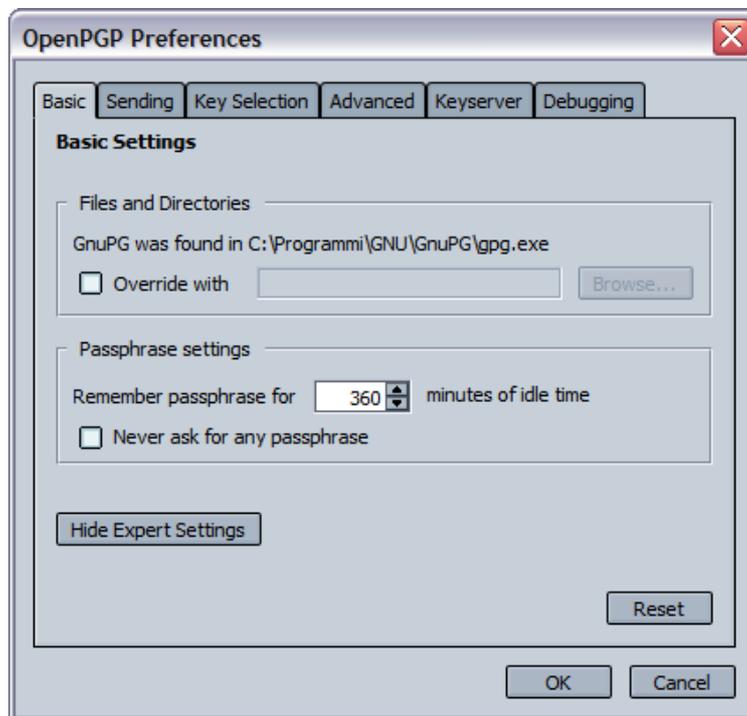
If you're using SeaMonkey 1.x, there will be also a button named *Uninstall Enigmail* which uninstalls the Enigmail application altogether.

SeaMonkey 2.0 and Thunderbird do not show this button, as concerning these mailclients Enigmail can be uninstalled like any other extension from *Tools* → *Add-ons* → *Extensions*.

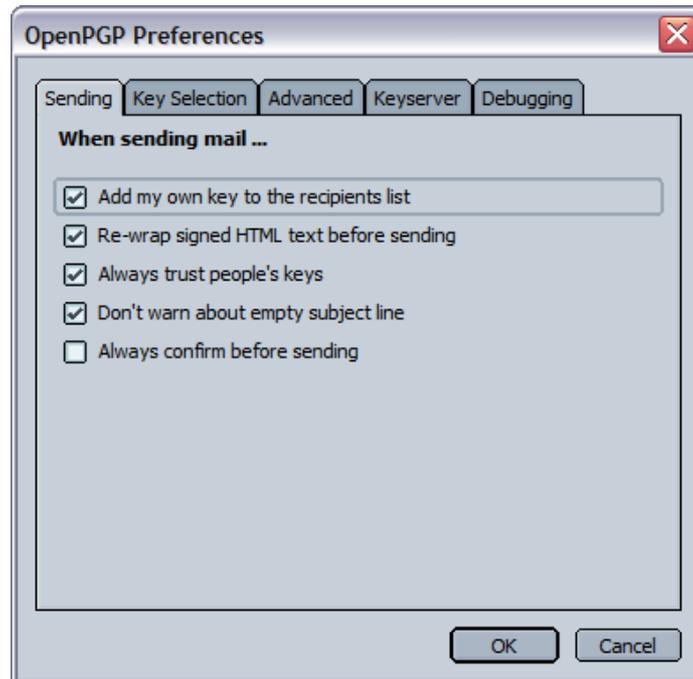
Finally, the *Display Expert Settings / Hide Expert Settings* toggle button permits to access the expert preferences.

In Enigmail 0.96.0 and earlier, this button makes another button named *Advanced...* appear (and disappear). Clicking on the *Advanced...* button opens a separate window with the expert preferences, divided into five tabs: Sending, Key Selection, Advanced, Keyserver, and Debugging.

In newer versions of Enigmail, the *Display Expert Settings* button activates instead the five tabs with the expert settings directly in the same window. Expert preferences settings are explained in the rest of this Section.



9.1.2. Sending



These settings define how Enigmail must behave when sending secured mail. You can jump to this settings window also by selecting the menu command *OpenPGP* → *Default Composition Options* → *Send Options...* in the Message Composition window.

Add my own key to the recipients list tells Enigmail, when sending encrypted messages, to encrypt with your own public key (as specified in the current account settings) as well as with the recipients' public keys. This is required to allow you to decrypt your own messages you have encrypted. Therefore, you should leave this option enabled.

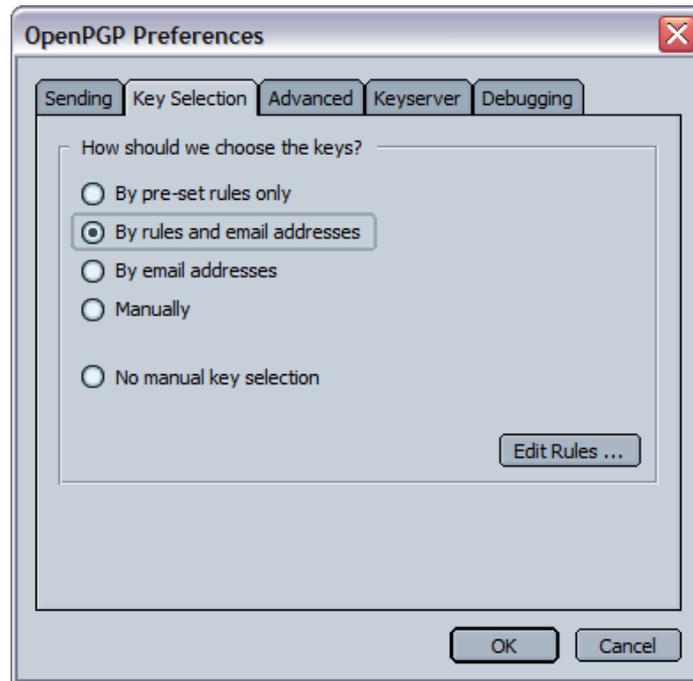
If you use HTML to compose email messages, messages signed with the inline PGP standard (the default in Enigmail) need to be re-wrapped before they can be sent in order to avoid invalid signatures. However, re-wrapping can cause the text to look differently from what you have typed. We recommend you leave enabled the option *Re-wrap signed HTML text before sending*, unless you have problems caused by re-wrapping.

Usually, OpenPGP does not allow you to encrypt message with keys that you do not trust. The option *Always trust people's keys* overrides the trust level of other people's public keys so that you are able to encrypt with untrusted keys. You should leave this option enabled.

Don't warn about empty subject line prevents Thunderbird / SeaMonkey from complaining about an empty subject line when composing a message.

Always confirm before sending prompts you a confirmation dialog before sending any message, so that you can check the signing, encryption, and S/MIME status. It is advisable to turn off this option if you send S/MIME signed or encrypted messages from time to time.

9.1.3. Key Selection



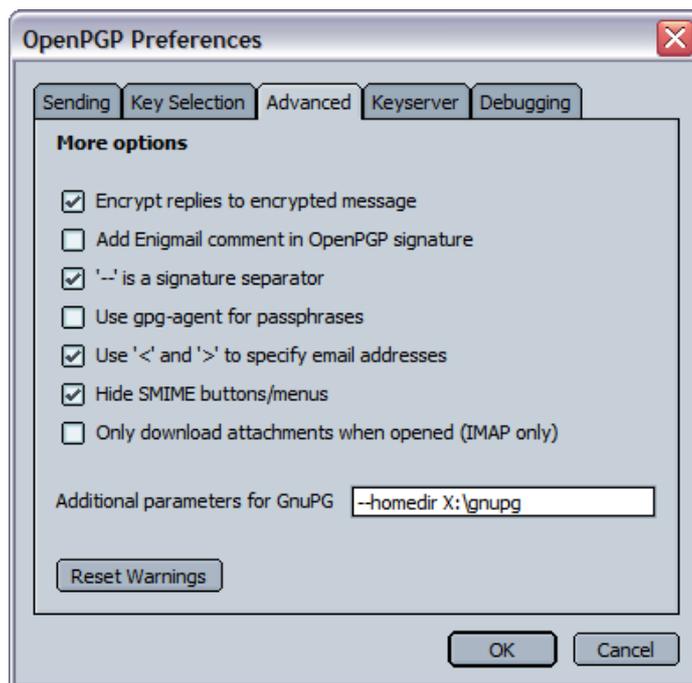
This setting defines how Enigmail should select, for each recipient, the public keys to encrypt a message with. You can jump to this settings window also by selecting the menu command *OpenPGP* → *Default Composition Options* → *Key Selection Options...* in the Message Composition window.

There are five possibilities:

- *By pre-set rules only* chooses the key depending exclusively on per-recipient rules. If for an intended recipient there is not a per-recipient rule, you will be prompted to create one.
- *By rules and email addresses* first searches for a per-recipient rule matching the recipient; if there is not such a rule, is selected the key with an user ID matching the recipient. This is the default setting.
- *By email addresses* selects the key with an user ID matching the recipient. No per-recipient rule is enforced.
- *Manually* instructs Enigmail to never select a key automatically. Whenever message encryption is applied, Enigmail pops up the Key Selection window to let you choose the keys manually.
- *No manual key selection* tells Enigmail to encrypt the mail if a valid key for all recipient is found. The mail is sent unencrypted if no valid key can be determined for all recipients, i.e. if there is at least one recipient for which there is not public key in the public keyring. Per-recipient rules are enforced.

The *Edit Rules...* button opens the Per-Recipient Rules Editor window.

9.1.4. Advanced



These settings define miscellaneous OpenPGP and Enigmail options.

Enable *Encrypt replies to encrypted message* if you want Enigmail to automatically switch on encryption when composing a reply to an encrypted message. This is a smart thing to do, especially if you quote the original message.

Add Enigmail comment in OpenPGP signature adds the comment line `Comment: Using GnuPG with Mozilla -`
`http://enigmail.mozdev.org/` to the OpenPGP signature block. Note that you can add any comment to the OpenPGP signature by calling GnuPG with the parameter `--comment your_comment` (see below to learn how to specify additional parameters to the GnuPG executable).

When signing, lines starting with a dash (-) are replaced with two dashes separated by a space (- -) according to the OpenPGP standard. This however makes a double-dash line (--) no longer appear as a separator between the message body and a personal signature, usually displayed in grey. By enabling the option *'--' is a signature separator*, Enigmail makes some workaround to correctly handle the signature separator when reading and composing messages.

Use gpg-agent for passphrases tells Enigmail to use gpg-agent, the GnuPG passphrase agent, to cache the passphrase; this tool is especially useful if you use several passphrases. Do not activate this option if you want Enigmail to ask you for your passphrase.

GnuPG version 2.0.x is distributed with gpg-agent. Enabling this option makes Enigmail use gpg-agent also for GnuPG version 1.4.x (this requires the gpg-agent and pinentry tools to be installed).

Note that some distributions install Seahorse instead of gpg-agent, and this may cause problems when using OpenPGP smart cards. If you use a smart card for your key, then either use gpg-agent and enable this option, or unset it and make sure the environment variable `GPG_AGENT_INFO` is unset prior to starting Enigmail (since GnuPG expects gpg-agent to be running once it detects that this variable is set). See also FAQ entry 11.1.8.

Use '<' and '>' to specify email addresses. Usually, email addresses are surrounded by angle brackets (< >) to separate the full name part from the email part, e.g. John Random Hacker <jrhacker@example.com>.

Deactivating this option removes the brackets from email addresses. This is necessary to ensure compatibility with some provider service, like Hushmail, that does not support brackets in email addresses. Hushmail is a provider for OpenPGP encryption over the web, but key generated with Hushmail are not fully compatible to OpenPGP.

This option should be normally turned on when encrypting, as Enigmail relies on it to avoid potential confusions and hence security problems, but needs to be turned off for Hushmail keys.

Hide SMIME buttons/menus hides the S/MIME button from the toolbar of the mailclient interface. This option exists only on SeaMonkey 1.1, because Thunderbird and SeaMonkey 2.0 and higher allow the user to customize the whole toolbar. This option, if exists, is enabled by default in order to avoid to confuse users.

Only download attachments when opened (IMAP only) enables an IMAP feature that makes Thunderbird / SeaMonkey download only the first 35-40 Kb of a message, downloading attachments only on demand. However, if an encrypted message is larger than this size, it may happen that it is downloaded only in part, its end will be missing, and hence Enigmail will fail to decrypt it.

If you use an IMAP inbox, and notice that Thunderbird / SeaMonkey sees some of your mails as broken or reports an error when trying to decrypt them, disable this option. The mailclient will then download the complete message at once. Alternatively, you can click on the broken lock to download the message in full. This problem often happens with Thunderbird 2. It should not appear with Thunderbird 3, which by default downloads the whole message.

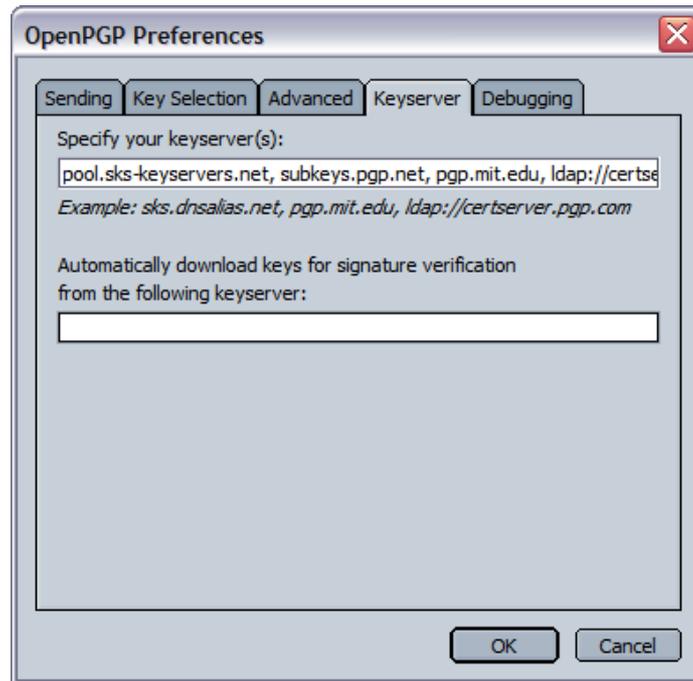
The text field *Additional parameters for GnuPG* allows you to have Enigmail call the GnuPG executable with the parameters you prefer.

For instance, the figure at page 72 shows the additional parameters `--homedir X:\gnupg` being passed to GnuPG; this tells GnuPG to go look for the keyrings in the `X:\gnupg` directory instead of the default directory.

Finally, the *Reset Warnings* button controls the way Enigmail pops up the interactive dialogs asking you to make a choice. If you ever asked Enigmail to

remember your choice for the future (for instance when choosing how Enigmail should sign/encrypt attachments), clicking this button will have Enigmail prompt you the dialog again when needed.

9.1.5. Keyserver



These are the options related to keyservers used to search public keys from.

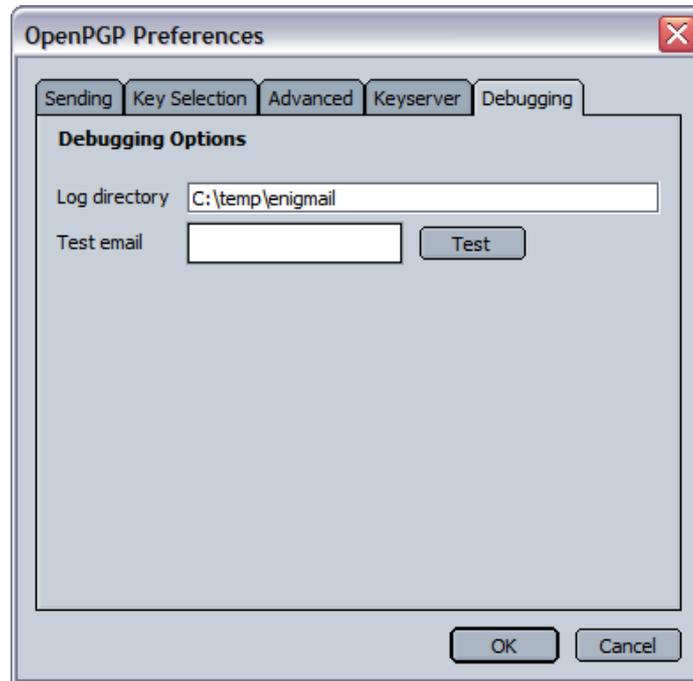
The text field *Specify your keyserver(s)* allows you to specify a list of OpenPGP keyservers. These keyservers will be proposed to you next time you launch a search for a person's public key on a keyserver.

You may prepend a protocol to the name of a keyserver, e.g.

`hkp://keyserver.example.com` or `ldap://certserver.pgp.com`.

If you want, you may enter a keyserver name in the field *Automatically download keys for signature verification from the following keyserver*. Enigmail will then automatically try to download from the specified keyserver any public key needed to verify signed messages. If you use this option, specify only one name.

9.1.6. Debugging



These options can help to track down why Enigmail doesn't work as expected.

In the field *Log directory* you can type the name of a valid directory path, e.g. `C:\Windows\temp` (Windows) or `/data/tmp` (Linux). Once you enter a value in the field, debugging will automatically be turned on and log files will be stored in the directory.

Close the Preferences window and restart the mailclient. After restart, Enigmail will create a file named `enigdebug.txt`, containing the trace log, in the specified directory. You can view this file via the menu command *OpenPGP* → *Debugging Options* → *View Logfile*.

Changing the value of the log directory requires Thunderbird / SeaMonkey to be restarted.

Test email allows you to check an email address that is a user ID of a key in your keyring. Enter the email address in the field and press the Test button. Enigmail will then perform a couple of tests and inform you about the result. You can check the details of these tests in the Enigmail Console, which is available via the menu command *OpenPGP* → *Debugging Options* → *View Console*.

9.2. Manually editing the preferences

Manual editing of preferences are intended for advanced users only. Enigmail preferences are stored together with Mozilla global preferences inside the `user.js` and `prefs.js` files in your profile directory. Mozilla first reads the preferences in `user.js`, if this file exists, and copies them in `prefs.js`. You can create the file `user.js` in order to keep your custom preferences that you do not wish to be ever changed; this file, unlike `prefs.js`, won't be modified by Mozilla.

The preferences are accessible in SeaMonkey by opening the URL `about:config`, and in Thunderbird under *Tools* → *Preferences* → *Advanced* → *General* → *Config Editor*.

From there, you can change the preferences values, which will then be written in `prefs.js` under the following format:

```
user_pref("preference_name", value);
```

This section provides a reference in alphabetical order to the various Enigmail preferences. First is printed the preference name and its default value; then follows its meaning, and the relevant menu command if the preference can also be set from the GUI.

You can review the default preferences settings in the `enigmail.js` file.

extensions.enigmail.addHeaders ***true***

Adds custom X-Enigmail mail headers to all outgoing messages. These headers are not currently used for any function, but may be used by Enigmail in the future.

Currently the added headers are:

```
X-Enigmail-Version: 0.96.0.0 (or whatever)
X-Enigmail-Supports: pgp-inline, pgp-mime
```

extensions.enigmail.advancedUser ***false***

Shows the advanced settings.

OpenPGP → *Preferences* → *Display Expert Settings / Hide Expert Settings*

extensions.enigmail.agentAdditionalParam ***""***

Additional parameters passed to the GnuPG executable. Default value is blank.

OpenPGP → *Preferences* → *Advanced* → *Additional parameters for GnuPG*

extensions.enigmail.agentPath

""

The path to the GnuPG executable. If it is already in the PATH, this setting can be left blank.

OpenPGP → Preferences → Files and Directories

extensions.enigmail.allowEmptySubject

false

Allows the user to send a mail with no subject.

OpenPGP → Preferences → Sending → Don't warn about empty subject line

extensions.enigmail.alwaysTrustSend

true

Always trust other people's keys when sending. Default is on.

OpenPGP → Preferences → Sending → Always trust people's keys

extensions.enigmail.autoCrypto

false

Reserved for future use; do not change this setting.

extensions.enigmail.autoDecrypt

true

Automatically decrypt/verify received messages.

OpenPGP → Automatically Decrypt/Verify Messages

extensions.enigmail.autoKeyRetrieve

""

Keyserver to automatically download public keys from. As default, no keyserver is specified.

OpenPGP → Preferences → Keyserver → Automatically download keys for signature verification from the following keyserver:

extensions.enigmail.composeHtmlAlertCount

3

Sets the number of times a warning message is shown when composing in HTML and attempting to send a signed message using inline PGP.

extensions.enigmail.configuredVersion

""

The last configured Enigmail version. This setting should not be changed.

extensions.enigmail.confirmBeforeSend ***false***

Pops up the confirmation dialog before sending a message.
OpenPGP → *Preferences* → *Sending* → *Always confirm before sending*

extensions.enigmail.disableSMIMEui ***false***

Disables the *S/MIME* button (only for SeaMonkey).
OpenPGP → *Preferences* → *Advanced* → *Hide SMIME buttons/menus*

extensions.enigmail.displayPartiallySigned ***true***

Handles the display of the OpenPGP status bar if only part of the message is signed (this may happen for instance when a person replies quoting a signed message). An example of a partly signed message is shown in the figure at page 58.

If set to false, PGP headers that appear within the message body will be ignored and displayed literally. This does not apply for the PGP headers at the beginning and the end of the message body, which are present there when the message is normally signed in full; in this case, the OpenPGP status bar will appear.

If set to true (default), Enigmail even removes the quote prefix character (>) from signed text embedded in the rest of the body if the message itself as a whole is not signed. This only works if the embedded quote has not been modified at all.

extensions.enigmail.displaySecondaryUid ***true***

Forces Enigmail to search your keyring for secondary IDs in order to match the sender address of a received message, and displays it instead of the default key ID.

extensions.enigmail.displaySignWarn ***true***

Warns when you change the signing status by clicking on the sign icon in the bottom right corner of the message composition window.

extensions.enigmail.doubleDashSeparator ***true***

Handles the double dash (--) as a signature separator..
OpenPGP → *Preferences* → *Advanced* → *'--' is a signature separator*

extensions.enigmail.encryptAttachments **1**

This setting stores the value of the last encryption method used to send a message with attachment.

extensions.enigmail.encryptAttachmentsSkipDlg **0**

Controls whether to skip the attachment dialog, where the user is asked how attachments should be encrypted/signed.

extensions.enigmail.encryptToNews **false**

If set to false (default), won't allow user to send an encrypted message to a newsgroup.

extensions.enigmail.encryptToSelf **true**

Automatically encrypts outgoing messages with your public key too.
OpenPGP → *Preferences* → *Sending* → *Add my own key to the recipients list*

extensions.enigmail.gpgVersionWarnCount **1**

Warns if the installed version of GnuPG is less than v1.0.6, which is the minimum version Enigmail can work with.

extensions.enigmail.handleDoubleClick **true**

Normally, to decrypt and open an encrypted attachment you right-click on it and select *Decrypt and Open* from the pop-up menu. This option enables automatic decryption and opening if you double-click on the attachment.

extensions.enigmail.hideHeaders ***x-enigmail-version***
openpgp
content-transfer-encoding
x-mimeole
x-bugzilla-reason
x-php-bug

Specifies the mail headers that must be retrieved from the mail backend, but hidden to users. This preference exists only in Enigmail v0.96.0 and earlier.

extensions.enigmail.hushMailSupport ***false***

Enables support for Hushmail.

OpenPGP → *Preferences* → *Advanced* → *Use '<' and '>' to specify email addresses*

extensions.enigmail.initAlert ***true***

Displays a warning if Enigmime fails to initialize.

extensions.enigmail.inlineAttachAsciiArmor ***false***

Encrypts attachments in ASCII-armored format (GnuPG parameter -a) when sending inline PGP encrypted messages.

extensions.enigmail.inlineAttachExt ***".pgp"***

Sets the extension to be appended when creating attachments to inline PGP encrypted messages. Default is `.pgp` (i.e. a file `filename.ext` will be renamed to `filename.ext.pgp` when encrypted).

extensions.enigmail.inlineSigAttachExt ***".sig"***

Sets the extension to be appended when creating attachments to inline PGP signed messages. Default is `.sig` (i.e. a file `filename.ext` will be accompanied by an additional file named `filename.ext.sig` containing its signature).

extensions.enigmail.keepSettingsForReply ***true***

Enables encryption of replies to encrypted messages.

OpenPGP → *Preferences* → *Advanced* → *Encrypt replies to encrypted message*

extensions.enigmail.keyManShowAllKeys ***false***

If set to false (default), the Key Management window shows only those keys that match the search terms entered in the field *Search for*. If set to true, all keys are displayed.

OpenPGP → *Key Management* → *Display All Keys by Default*

extensions.enigmail.keyserver ***"pool.sks-keyservers.net,
subkeys.pgp.net,
pgp.mit.edu,
ldap://certserver.pgp.com"***

The list of keyserver(s) to use.

OpenPGP → *Preferences* → *Keyserver* → *Specify your keyserver(s)*

extensions.enigmail.logDirectory ***""***

Specifies the log directory. If set, also enables debugging. Default is empty (unset).

OpenPGP → *Preferences* → *Debugging* → *Log directory*

extensions.enigmail.maxIdleMinutes ***5***

Caches the passphrase for the specified time. Default is 5 minutes.

OpenPGP → *Preferences* → *Remember passphrase for [] minutes of idle time*

extensions.enigmail.mimeHashAlgorithm ***0***

GnuPG hash algorithm to use. Available values are:

- 0 – automatic selection, let GnuPG choose (default)
- 1 – SHA1
- 2 – RIPEMD160
- 3 – SHA256
- 4 – SHA384
- 5 – SHA512

extensions.enigmail.noPassphrase ***false***

Tells Enigmail to never ask for a passphrase.

OpenPGP → *Preferences* → *Basic* → *Never ask for any passphrase*

extensions.enigmail.parseAllHeaders ***true***

Tells Enigmail to parse all MIME headers in the message. This value is for testing purposes only, and must be left enabled.

extensions.enigmail.quotedPrintableWarn **0**

Issues a warning when Enigmail detects that a message going to be sent contains 8-bit characters and will use Quoted Printable encoding. Default is off. This setting remembers the selected state.

extensions.enigmail.recipientsSelection **2**

Controls how Enigmail should select the keys for unknown recipients of encrypted messages. Available values are:

- 1 – by per-recipient rules only
- 2 – by per-recipient rules and email address (default)
- 3 – by email address only
- 4 – manually, by prompting the user
- 5 – no key selection

OpenPGP → *Preferences* → *Key Selection*

extensions.enigmail.respectHttpProxy **true**

Uses the same HTTP proxy settings that were defined in Thunderbird / SeaMonkey to retrieve keys from keyservers.

extensions.enigmail.saveEncrypted **0**

If this setting is on, when composing a message that is going to be sent encrypted, if you save it as a draft Enigmail asks if you would like to encrypt the draft. Default is on. This setting remembers the selected state.

extensions.enigmail.supportMultiPass **false**

Support for multiple passphrase on the same key pair. Reserved for future use; do not change this setting.

extensions.enigmail.useDefaultComment **false**

If set to false (default), adds an Enigmail comment in OpenPGP signature.
OpenPGP → *Preferences* → *Advanced* → *Add Enigmail comment in OpenPGP signature*

extensions.enigmail.useGpgAgent ***false***

Use gpg-agent to handle passphrases.
OpenPGP → *Preferences* → *Advanced* → *Use gpg-agent for passphrases*

extensions.enigmail.useGpgKeysTool ***true***

Enables Enigmail to use the gpgkeys_hkp, gpgkeys_ldap, and gpgkeys_http tools to retrieve keys from key servers without using gpg itself.

extensions.enigmail.warnClearPassphrase ***true***

Has Enigmail pop up a confirmation message informing you that the passphrase has been cleared.

extensions.enigmail.warnGpgAgentAndIdleTime ***true***

Warns if gpg-agent is found and the setting *Remember passphrase for [] minutes of idle time* is active.

extensions.enigmail.warnIso2022jp ***true***

Displays a warning if the broken character set ISO-2022-JP is used. This setting remembers the selected state.

extensions.enigmail.warnOnRulesConflict ***0***

Issues a warning when sending a message to multiple addresses with conflicting per-recipient rules. Default is off. This setting remembers the selected state.

extensions.enigmail.warnOnSendingNewsgroups ***true***

Issues a warning when trying to send an encrypted message to a newsgroup.

extensions.enigmail.warnRefreshAll ***true***

Displays a warning when all keys are going to be refreshed.

extensions.enigmail.wrapHtmlBeforeSend ***true***

Re-wrap HTML text in signed messages before sending. Default is on.
OpenPGP → *Preferences* → *Sending* → *Re-wrap signed HTML text before sending*

mail.identity.default.enablePgp ***false***
mail.identity.default.pgpkeyId ***""***
mail.identity.default.pgpKeyMode ***0***
mail.identity.default.pgpSignPlain ***false***
mail.identity.default.pgpSignEncrypted ***false***
mail.identity.default.defaultEncryptionPolicy ***0***
mail.identity.default.openPgpHeaderMode ***0***
mail.identity.default.openPgpUrlName ***""***
mail.identity.default.pgpMimeType ***false***
mail.identity.default.attachPgpKey ***false***

Preferences for per-identity settings.

10. TROUBLESHOOTING

This chapter contains several tips to troubleshoot any problem you may encounter when installing or using Enigmail.

10.1.1. Thunderbird / SeaMonkey displays a red error message at the bottom of the mail window.

After having installed Enigmail, Thunderbird / SeaMonkey may displays errors like this at the bottom of the mail window:

```
<menu id="menu_Enigmail"  
-----^
```

This problem usually arises when using a translated version of Thunderbird / SeaMonkey or a translation pack. In this case, you need to have a translated version of Enigmail, and the language pack must match the Enigmail version. Several languages are available from the Language Packs page on the Enigmail website.

If your language is not available, you are encouraged to translate Enigmail yourself. A description for how to do it is provided on the same page.

10.1.2. Enigmail fails to install on SeaMonkey.

If you cannot manage to install SeaMonkey, try these tips:

- Currently, only users with write access to the directory where SeaMonkey is installed can install additional components like Enigmail. On a multi-user Unix system where SeaMonkey is installed in `/usr`, only the root user (the system administrator) can install Enigmail. On Linux systems, try using the RPMs from the download page.
- If you are unable to use the RPM/dpkg method, you will need to run SeaMonkey as root and install Enigmail as root via the XPI file. You should then restart the browser as root to initialize Enigmail and update SeaMonkey's component registry.
- Of course, you may always install a copy of SeaMonkey in your own (non-root) subdirectory and then install Enigmail.
- Installation requires access to a temporary directory. If that is full, then installation may fail.
- If an installation failure message displays on screen, try to locate the installation logfile, which is a file named `install.log` located in the same directory as the SeaMonkey executable. (If you installed a Mozilla tarball, this would correspond to the `mozilla` directory.) Perusing the last few entries in the logfile should give you some information on the reasons for the installation failure.

10.1.3. Enigmail fails to install on Firefox.

Enigmail is an extension for Thunderbird and the SeaMonkey mailclient. It is not supposed to, and hence cannot, be installed in Firefox.

If you use Firefox to download Enigmail, you need to right-click on the download link, select *Save as...*, and save the XPI file on your machine. Then open Thunderbird, go to *Tools* → *Add-ons* → *Extensions*, click the *Install* button and select the XPI Enigmail file. Restart Thunderbird afterwards.

If you're looking for a GnuPG extension for Firefox, for instance to encrypt/decrypt/sign messages using webmail, then have a look at the FireGPG extension at <http://getfiregpg.org>.

10.1.4. The Add-ons Manager shows “This item will be installed after you restart Thunderbird”. Or, there is no Enigmail user interface visible.

The Add-ons Manager checks whether an extension is validated to work on your version of Thunderbird. If you have updated Thunderbird and are trying to install a previous version of Enigmail, it may be that Enigmail has not been validated against the new version.

Hence, you will need to remove Enigmail or use a version that is validated for the version of Thunderbird you're using. The Add-ons Manager currently does not remove extensions in the correct way, therefore in order to remove Enigmail you will need to shut down Thunderbird and delete the `Chrome` and `Extensions` directories from your profile.

This problem may also happen on SeaMonkey.

10.1.5. I have updated Enigmail on Thunderbird, and now it keeps telling me: “A previous install did not complete correctly. Finishing install.”

Follow these steps:

Go to your Thunderbird installation directory and delete the file

`xpicleanup.dat`.

Run Thunderbird, go to *Tools* → *Add-ons* → *Extensions*, select the Enigmail extension and click on *Disable*. Restart Thunderbird.

Install the new version of Enigmail, then enable Enigmail again from the Add-ons Manager.

Restart Thunderbird and this problem should be fixed.

10.1.6. I can't tell whether Enigmail works or not.

If installation was successful, restart Thunderbird / SeaMonkey. Then look on the Mail/News window, which should have an *OpenPGP* menu on the menubar. Selecting *OpenPGP* → *About OpenPGP* will display the Enigmail version number and GnuPG executable details.

Remember that Enigmail has only been tested with milestone releases of Thunderbird and SeaMonkey. If you use a nightly build, or your own build, Enigmail might not work and might even crash.

10.1.7. I installed a new extension and Enigmail stopped working.

Some extension causes conflicts with Enigmail, preventing it to successfully sign/encrypt outgoing mail or verify/decrypt incoming mail. Check the FAQ entry 11.1.2. for a list of supported and unsupported extensions.

10.1.8. Enigmail icons in the toolbar are misaligned.

If you're using SeaMonkey and your toolbar looks like that:



then try changing your theme twice. If you are using the Modern theme as in the picture, switch to the Classic theme and quit SeaMonkey. Restart, change back to the Modern theme, quit, and restart again.

If you're using Thunderbird, then your theme does not contain the necessary icons for Enigmail. Please report the problem to the author of your theme.

10.1.9. Enigmail is unable to access the keyserver.

Keyservers use the Horowitz Keyserver Protocol (HKP) to exchange keys through TCP port 11371. If you are behind a firewall, you must ensure that this port is open for outgoing connections. Alternatively, many keyservers allow access to clients also on HTTP (TCP port 80), which is normally open.

If you are using HTTP proxy behind a firewall, you must add the following line to your `gnupg.conf` file:

```
keyserver-options http-proxy=proxy_host_name
```

10.1.10. My own signatures are invalid. Enigmail replaces “>” with “|” and spaces with “~” in quoted messages.

This is not a bug, but a workaround that Enigmail uses to handle the *text=flowed* format (RFC 2646), which causes problems with OpenPGP signatures.

Do not use this format in your mailclient; please set the option `mailnews.send_plaintext_flowed` to false. In Thunderbird, this option is under *Tools* → *Preferences* → *Advanced* → *General* → *Config Editor*. In SeaMonkey, this option is found by opening the URL `about:config`.

10.1.11. I use a non-English character set, and my own signatures are invalid.

When sending signed emails containing non-English characters (e.g. å or ð), the signature may fail to verify.

To fix this problem, make sure the option *For messages that contain 8-bit characters, use 'quoted printable' MIME encoding* is unchecked in your mail client. In SeaMonkey, this option is under *Edit → Preferences → Mail & Newsgroups → Composition*. In Thunderbird, this option is under *Tools → Options → Composition*.

10.1.12. Enigmail sees some emails as broken.

This problem occurs often when using an IMAP mailserver and is due to Thunderbird / SeaMonkey not downloading the message as a whole. To fix this problem, go to *OpenPGP → Preferences → Advanced* and disable the option *Only download attachments when opened (IMAP only)*. See also the explanation of this option in Section 9.1.4.

10.1.13. I get an error “Enigmail / Enigmime / IPC failed to initialize”.

Enigmail works only if it is built using the same build environment as Thunderbird or SeaMonkey was built. This means that you can use the official Enigmail releases only if you use the official releases of Thunderbird or SeaMonkey provided by mozilla.org.

If you use a Thunderbird or SeaMonkey version coming from some other source (e.g. the provider of your Linux distribution), or if you build Thunderbird or SeaMonkey yourself, you should either use an Enigmail version built by the same source or build Enigmail yourself. For building Enigmail from source code, refer to <http://enigmail.mozdev.org/download/source.php>.

Please don't file any bug report concerning this problem: it is not solvable.

10.1.14. I cannot read encrypted messages sent to me! I get an error “Secret key needed to decrypt message”.

Unless you accidentally deleted your key pair (for which there is no remedy, unless you have a backup – see below), the message you received was not encrypted with your public key. The sender most likely encrypted it with his public key instead of yours.

Make sure the sender has your public key, and tell him to encrypt the message with it.

10.1.15. I lost my passphrase / my key pair / my private key.

A note: Your private key is bundled with your public key in your key pair, hence losing your private key and losing your key pair means exactly the same.

There is no way to recover your passphrase: your only hope is to try to remember what it was. If you don't succeed, you lose the use of your private

key, and hence your whole key pair is now useless. There is no way to recover your private key, either. It cannot be obtained from your public key or from any message that was signed/encrypted by that private key. You can only recover it if you made a backup in the past.

Hence, losing the passphrase or the key is definitive. If you generated a revocation certificate (and you should have), use it to revoke the key pair. You must also generate a new key pair, send the new public key to your contacts and warn them not to use the old public key any more.

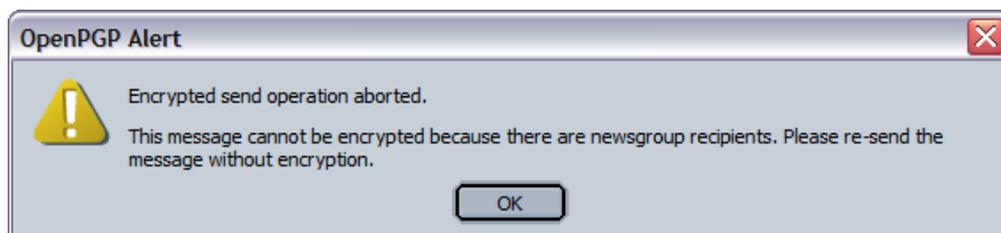
Messages that were sent to you encrypted with the old key cannot be decrypted any more. Messages that were signed by you with the old key can still be verified by the recipients by using the old (revoked) key.

To avoid this disaster, it is recommended that you backup in advance your key pair: from Key Management, select *File* → *Export Keys to File*, make sure you included the secret key, then store the file in a safe place. Make sure you chose a passphrase you can remember, too.

10.1.16. After I reinstalled Enigmail, all keys have disappeared from the Key Management window.

The keys are still there, but are displayed only the keys that match the search criteria entered in the *Search for* field. If you want to see all keys, tick the checkbox *Display all keys by default*.

10.1.17. I get an error whenever I try to post to a newsgroup.



You are trying to post an encrypted message to a newsgroup. This doesn't make any sense as a newsgroup, like a mailing list, is a public space and not an entity that could own a key pair. (Just ask yourself: who is supposed to own the private key? What would be the trust associated with this entity?)

Send your message unencrypted. If you just want to obfuscate information, like spoilers, you can as well use ROT13.

10.1.18. I have set forwarding rules on Thunderbird, and I get an error “Sending failed, please check your settings”.

Unfortunately forwarding rules do not work in Thunderbird with Enigmail installed. Currently there is no solution or workaround for this issue.

10.1.19. I get the message “OpenPGP error; Encryption/signing failed; send unencrypted message?”.

This happens when you're writing a mail and you have enabled the option *OpenPGP → Attach my Public key* to send your public key with the mail. This option causes some problem due to a bug. As a workaround, choose *OpenPGP → Attach Public key...* instead, then in the Key Selection window select your key and click *Ok*.

10.1.20. Key import fails with an error “File name too long”.

If, when you try to import a key, you get the following error:

```
Importing the keys failed
gpg: can't open (very long string of characters running off screen)
File name too long
gpg: Total number processed: 0
```

then try to import the key through GnuPG. Open a command line and issue the following command: `gpg --import key_file_to_import.asc`.

10.1.21. I have some other problem I can't solve.

In this case, if you know how to use GnuPG, you can try to achieve the same result through the GnuPG command line. For instance, if you cannot remove public key 0xABCDEF01 from Key Management, open a shell prompt and issue the following command: `gpg --delete-keys 0xABCDEF01`.

If the above doesn't work or you don't feel yourself enough experienced to use GnuPG, ask the friendly Enigmail/GnuPG community for support. References for obtaining support are listed in Chapter 13.

11. FAQ

This chapter contains the Frequently Asked Question about Enigmail and around.

11.1.1. Can Enigmail be used for webmail? When will this feature be added?

Enigmail is developed for the Mozilla mailclient (Thunderbird and SeaMonkey). There is no intention from our team to extend Enigmail to support web based mail, or web applications in general.

The FireGPG extension (<http://getfiregpg.org>) is targeted to supporting web mail for Firefox.

11.1.2. Are there known incompatibilities with other Thunderbird or SeaMonkey extensions?

The following extension causes problems with Enigmail:

- Allow Empty Subject (Does not work. Use the following Enigmail option instead: *OpenPGP* → *Preferences* → *Advanced* → *Don't warn about empty subject line.*)

The following extensions have been tested and are compatible with Enigmail:

- Buttons
- Calendar
- ForumZilla
- Get All Messages
- JSlib
- Mnenhy
- Preferential

11.1.3. Why is Enigmail incompatible with my Thunderbird / IceDove / ...?

Enigmail is only compatible to official releases of Thunderbird and SeaMonkey as provided by the Mozilla Foundation and its subsidiaries.

In particular, releases shipped with Linux distributions (such as Ubuntu, Red Hat, OpenSuSE or Debian) cannot be supported. If you want to use Thunderbird / SeaMonkey / IceDove / IceApe created and installed by your distribution, then you also need an Enigmail release created by the same distribution.

11.1.4. Is it possible to use PGP with Enigmail?

No. PGP is not supported with Enigmail. PGP does not provide a command line capability that Enigmail can use, unless you purchase PGP Command Line which costs over €1000.

11.1.5. How do I uninstall Enigmail?

In SeaMonkey 1.x: go to *OpenPGP* → *Preferences*, click the *Uninstall* button, then restart SeaMonkey.

In SeaMonkey 2.0 and Thunderbird: go to *Tools* → *Add-ons* → *Extensions*, right-click on Enigmail, choose *Uninstall* from the context menu, then restart the mailclient.

11.1.6. Which files Enigmail modifies on my system?

The following information is valid for SeaMonkey 1.x only.

Assuming that `mozilla/` is the directory where the SeaMonkey executable resides, Enigmail creates the following files as a result of its installation:

```
mozilla/components/enigmail.js
enigmail.xpt
libenigmime.so on Linux, enigmime.dll on Windows
enigmime.xpt
ipc.xpt
enigprefs-service.js
mozilla/chrome/enigmail.jar
```

These are the files to delete if, for any reason, you want to do a manual uninstall.

Enigmail also modifies the following two overlay files by inserting Enigmail-related lines:

```
mozilla/chrome/overlayinfo/communicator/content/
overlays.rdf
mozilla/chrome/overlayinfo/messenger/content/
overlays.rdf
```

Do not manually delete `mozilla/chrome/enigmail.jar` without removing the Enigmail entries from the `overlays.rdf` files; this could cause SeaMonkey to hang.

11.1.7. Enigmail seems not to work with Gpg4win. What's wrong?

Gpg4win versions 1.1.0 and 1.1.1 contain an error that prevents Enigmail to work with it. However, there is an easy workaround.

Assuming that you choose the default installation directory for Gpg4win, go to *OpenPGP* → *Preferences*, tick the checkbox *Override with*, and enter the

following path: `C:\Program Files\GNU\GnuPG\gpg.exe`.
(Depending on your localisation of Windows, your Program Files folder may be called Programmi, Programmes, or other instead.)

The problem is that Gpg4win creates a wrong entry in the PATH environment variable. Alternatively, instead of the workaround above, it is also possible to change the PATH such that `\pub` is removed from `C:\Program Files\GNU\GnuPG\pub`.

11.1.8. Why does Enigmail try to use gpg-agent?

Enigmail will use the gpg-agent for passphrase handling if any of the following conditions are fulfilled:

- if GnuPG version 2.0 or higher is detected;
- if the environment variable `GPG_AGENT_INFO` is set;
- if the option *Use gpg-agent for passphrase* is active (this option is located in *OpenPGP* → *Preferences* → *Advanced* tab)

If you are using GnuPG version 1.4.x, and call the GnuPG executable with the parameter `--use-agent`, the usual cause is that the environment variable `GPG_AGENT_INFO` is set.

If the variable is set by your Linux/Unix distribution, but you don't want to use gpg-agent, you can unset the variable e.g. in the file `.xsession` or `.bashrc`, or you can launch your mailclient through a wrapper shell script like this one:

```
#!/bin/sh
unset GPG_AGENT_INFO
exec /path/to/your/mailclient "$@"
```

An alternative solution if you don't want gpg-agent is to correctly install all helper utilities.

gpg-agent needs the pinentry tool to query for the password; if it's not available, then you need to install it. For Linux, you have the choice between `pinentry-qt` and `pinentry-gtk`. Install either of them, and create a link from it to `pinentry`.

11.1.9. Which key type/size should I choose for my key pair? Which is best?

There is no such thing as “the best key.” All choices have consequences and trade-offs. You might feel that a 4096-bit RSA key is safer, but the person you're sending email to might be trying to read it on a PDA which takes over a minute to decrypt each message. You might decide to use SHA-1 because it's widely supported in OpenPGP implementations, but SHA-1 has some mathematical flaw and does not offer long-term security. Finding precisely the optimal set of consequences and trade-offs is a very subtle thing, and the perfect set for you will probably not be the same for anyone else.

The IETF OpenPGP Working Group has spent over a decade looking at which

choices offer an excellent balance of speed, safety, and compatibility for the vast majority of users. Their opinions have evolved over time to take into account the technology and threats of the day. The people of the GnuPG project are active participants in the Working Group, and as such GnuPG implements the Working Group's recommendations.

Therefore, the best advice we can give is to stick to Enigmail's defaults. They are not perfect, because no two people have the same definition of perfection. However, the defaults are excellent for the overwhelming majority of users.

11.1.10. How can I test if I'm using Enigmail correctly?

First, you can try to send yourself some message signed/encrypted, and check if you are able to verify/decrypt them correctly. Then you can send messages to Adele, “the Friendly OpenPGP email robot”, at `adele-en@gnupg.de`. Adele is an automated program that is able to receive and understand OpenPGP messages and reply accordingly. Remember to send her your public key so she can encrypt test messages to you. Some examples of communications with Adele are shown in Chapter 8.

11.1.11. How do I encrypt automatically my email messages?

Select *Encrypt messages by default* in your account settings. Then enable *OpenPGP* → *Preferences* → *Key Selection* → *No manual key selection*. This way, outgoing messages are encrypted automatically if a public key is available.

This method of default encryption works only for trusted public keys. So either you enable *OpenPGP* → *Preferences* → *Always trust people's keys* (which is slightly insecure), or you have to check every public key and sign it to make the key trustworthy (in this case you should only sign them locally, non-exportable).

There is one problem with this method: you never know if a mail will be encrypted when you just push the *Send Now* button. The workaround for this is to click on *Send Later* and look in your Unsent Messages folder.

11.1.12. Is it possible to permanently decrypt email messages?

This feature has been requested periodically by many people, primarily because the Email Search feature (in Thunderbird / SeaMonkey) cannot be performed on encrypted emails.

However, decrypting messages permanently is potentially a bad security practice. Once a message is stored unencrypted, it is vulnerable to compromise by anyone that has access to the computer or by any malware application that could steal it and forward it to anybody. If a message was confidential enough to be encrypted, it should stay encrypted.

The matter is even worse when the email message is not stored on the local machine but on a POP/IMAP server instead. Not only this could potentially disclose the message to the whole world, but makes more difficult to implement the feature. Permanently decrypting the email message would translate to delete the encrypted message on the server and upload a new message containing the decrypted information from the original mail. This involves a lot of email access issues – and email access is the email client's duty, not Enigmail's. Hence, implementing this feature would greatly increase the complexity in Enigmail, unless Mozilla provides someday an API to manipulate email data.

In short: permanent message decryption is not supported in Enigmail, and is not going to be in the near future.

However, you can export the decrypted message as a text file. To do this, select *OpenPGP* → *Save Decrypted Message* from the menu.

11.1.13. Is it possible to use S/MIME and OpenPGP encryption concurrently?

No, you cannot mix S/MIME and OpenPGP in the same message as the two standards, and their implementation in Mozilla, interfere with each other. If you want to use S/MIME you should not enable the Enigmail option *Encrypt messages by default* in your account settings (nor the one from S/MIME).

The S/MIME button is hidden by default in SeaMonkey 1.1. You can unhide it by unchecking the option *OpenPGP* → *Preferences* → *Advanced* → *Hide SMIME buttons/menus*.

11.1.14. How do I specify the hash algorithm?

As of version 0.95.0, Enigmail relies by default on GnuPG for selecting the hash (digest) algorithm. From GnuPG, the hash algorithm can be specified in the file `gpg.conf` using the parameter `digest-algo hash_algorithm`.

If you want to select the hash algorithm from within Enigmail, you can do so by modifying the preference `extensions.enigmail.mimeHashAlgorithm` and assigning to it one of the following values:

- 0 – automatic selection, let GnuPG choose
- 1 – SHA1
- 2 – RIPEMD160
- 3 – SHA256
- 4 – SHA384
- 5 – SHA512

To know more about modifying the preference values manually, also read Section 9.2.

11.1.15. How do I enable the debug log in Enigmail?

Select *OpenPGP* → *Preferences* → *Advanced* → *Debugging* and type a valid directory path in the *Log Directory* field. After you restart the mailclient, Enigmail will create there a trace log file named `enigdebug.txt`. See also Section 9.1.6.

11.1.16. How do I report a bug?

Visit <http://enigmail.mozdev.org/support/bugs.php>. Please check first the list of already known bugs so that a bug doesn't get submitted twice. If you spotted a new bug, you can file a bug report using Bugzilla.

11.1.17. How many people use Enigmail?

You can view the Enigmail download and usage statistics at <https://addons.mozilla.org/en-US/thunderbird/statistics/addon/71>.

11.1.18. It would be great if Enigmail could do this-and-this! Could you please implement it?

You can submit feature requests in the Enigmail Forum, Feature Requests thread. To know how to reach the Forum, see Chapter 13.

But please first consider that Enigmail follows the OpenPGP standard. It is not its purpose to innovate or invent new protocols. If the feature you propose is not included in or not compliant to the standard, the feature is not going to be included in Enigmail, no matter how many users ask for it.

The Enigmail source code is freely available, though. If you really need such a feature, you can download the code and modify it to suit your needs. Please consider first that breaking standards is generally not a wise idea, and will result in incompatible products.

11.1.19. How can I encrypt the Subject?

It is not possible to encrypt or sign the Subject of a mail message, nor any other mail header. See Section 8.5.

11.1.20. Why I can't select some keys for encryption in the Key Selection window?

Keys that have a red checkbox next to them are revoked or expired keys, and you cannot use them to encrypt. Download a valid public key from a keyserver, or contact your recipient and have him mail you his new, valid public key.

12. NOTES, TIPS & TRICKS

12.1. How to choose a good passphrase

The passphrase is the last line of defence to your private key, should your key pair fall in enemy hands. This might happen more easily than you think, by means of someone stealing your laptop, a malware uploading your private documents from your infected machine to a rogue server, or simply by your momentary thoughtlessness when you distribute your whole key pair instead of your public key.

With your secret key and your passphrase, anyone can impersonate you by signing messages on your behalf, and decrypt messages that were intended for your eyes only.

Luckily, the passphrase provides a quite good protection, since it encrypts the private key with a strong cipher. It is important that you choose a strong passphrase that could not be easily cracked by password guessing or brute-force programs. In this section we illustrate some criterion to do so.

GnuPG/Enigmail also allow you to not set a passphrase on your key pair. This is absolutely not recommended, and should be done only in exceptional circumstances, for instance when non-interactive processing is needed.

Do not use the following as your passphrase:

-  Your name, address, age, date or place of birth, car license plate, the name of your spouse, children, parents, pets, or any other information related to you;
-  Words in any language/dialect, past or present, real or imaginary, e.g. French, Cockney, Latin, Elven, and Klingon;
-  Names of real or fictitious people or places;
-  Names of movies, songs, music bands, groups, and such;
-  Obvious sequences of letters and/or numbers e.g. abc123, qwertyu, YYYYYYYY

-  Numerical constants e.g. 2.718281828459 (it's the mathematical constant e)
-  Any of the above written in all uppercase, all lowercase, or with alternated case e.g. CaLiFoRnIa
-  Any of the above prefixed or suffixed by a single character e.g. +California, California3
-  Any of the above with obvious replacements e.g. C4l1f0rn14
-  Anything that's less than 8 characters long (Enigmail will not even let you choose a passphrase that's shorter than that)
-  A password that you already use (e.g. on web sites or for your email account)

Instead, use these criteria to create a passphrase:

-  Use always a mix of at least 3 of the following characters in your passphrase: uppercase letters, lowercase letters, numbers, symbols like # * ! ? + - (& /
-  Insert two characters or more inside a word or name e.g. Ch7op8in, Debus!Z*sy
-  Join two words or names by two or more characters e.g. Bach#+Strauss
-  Nest one word or name inside another e.g. BeLudwigethoven
-  Condensate a proverb, a quote, a verse from a poem, a phrase from a movie, or any sentence you could have fixed in your mind:

Iw20yat/SPttbtp/thbgiaaos/btagtras.

This might seem impossible to remember but is in fact quite easy, once you think about the lyrics of *Sgt. Pepper's Lonely Hearts Club Band* by Lennon/McCartney:

“It was twenty years ago today
Sgt. Pepper taught the band to play
They've been going in and out of style
But they're guaranteed to raise a smile.”

Each letter of the passphrase is the first letter of each word. In the first line, the number is written in figures instead of being spelt out. In the second line, the name of the protagonist of the song is in uppercase letters. Each verse is separated by a slash, and a final dot is added. You can make up the rules as you prefer.

Another possible passphrase would be

HwmyrsmtBeyuclhm?

This one comes from Bob Dylan's *Blowin' In The Wind*, and is derived from the first and last letter of each word, considering only the first four of each verse:

“How many roads must a man walk down
Before you call him a man?”

These are the strongest passphrases, as they look like random sequences of letters.

You can use an existing quote, and make up the rules to transform it in a passphrase; should you ever forget the quote, a quick look on a book will solve the problem. You may also invent your own quote, although in this case forgetting it would be fatal.

12.2. Protection of the local machine

You should be aware of the truth that your encrypted mails are as safe as allowed by the computer you use Enigmail on. This point can never be stressed enough.

If your machine is infected with a keylogger and a malware that grants an intruder remote access on your files, all the cryptographic robustness of OpenPGP and the strongest passphrase won't protect your messages from being snooped or falsified. In a similar way, if you leave your computer unattended with your passphrase cached on, prepare yourself for nasty surprises. In fact, even using cryptography, your communications cannot be secure if your machine isn't. Even worse, cryptography could lure you into a false sense of security, making you more prone to share sensitive information via email.

The ciphers OpenPGP uses are the strongest known, and OpenPGP encryption is virtually unbreakable if done in the right way. However, there are a lot of other things that can go wrong.

The well-established fact that OpenPGP is the strongest link in the chain of security simply means that an attacker wanting to read your encrypted messages won't try to brute-force the encryption (which would take millions of years), but will focus on other weaknesses instead.

He might break into your computer and steal your secret key. Then, infect your computer with a spyware to record your passphrase, or directly record your

secret messages as you're typing them. For the purpose of recording, he might as well use a hardware keylogger installed between keyboard and machine. Or simply a hidden camera pointed towards your screen. Or even a TEMPEST device, hoping that you still use a CRT screen.

Once he gets his hands on the contents of your computer, either physically or from a remote location over the network, he may search for any plaintext remnants in nonvolatile storage devices or RAM.

From where did you get your copy of GnuPG and Enigmail? You should only trust software downloaded from the official web sites. Copies obtained from other sources might have been tampered with, and as such contain viruses, backdoors or trojans.

Finally, an attacker might persuade, force, or delude you (e.g. by impersonation) to surrender your passphrase, your secret key, or your messages. And all these attacks can be carried over your correspondents, too. The possibilities are endless.

12.2.1. Basic protection

You must follow these golden rules in order to keep your machine reasonably safe:

- Don't install, run, or open software of dubious origin (e.g. warez found on peer-to-peer networks, or programs hosted on untrusted web sites). This includes suspicious email attachments and macros on word processing programs.
- Use an antivirus software, updated daily. Make frequent scans of your machine and external hard drives.
- Use one or more antimalware programs.
- Filter network traffic with a personal firewall and deny all unknown connections.
- Install OS vendor patches. Keep all your software up-to-date, and keep yourself informed of the latest vulnerabilities.
- Use a screen lock when you are not physically in front of your computer and there are strangers around.
- Use strong passwords. Don't write them down in easy-to-find places.
- If you use a Wi-Fi connection, enable WPA on your access point.

12.2.2. Increased protection

If your communications involve critically sensitive information, you should not leave your computer physically accessible at all – even when turned off. If stolen, the thief would have access to all your files, including your secret key. The private key will still be protected by the passphrase but, by performing analysis and forensics on the filesystem, the thief will have access to a lot of plaintext data (temporary files, memory swap files, and such) that could include information you thought was encrypted. Windows leaves a lot of data around, and other OSes aren't much better with respect to this.

You might consider using whole-disk encryption at this point. Section 12.3.2. mentions some disk encryption software for additional protection of your key pair; most of this software can also be used to encrypt the whole OS.

It is also worth noting that a technically skilled intruder having physical access to a turned-off computer could infect it, leaving no traces, by replacing the bootloader with an infected one (*evil maid attack*).

12.3. Keeping your key pair in a safer place

To increase the security of your secret key you may decide to store your key pair in a different location than the default directory chosen by GnuPG, which for Windows is `C:\Documents and Settings\your_username\Application Data\GnuPG` in the local machine.

The easier solution is to keep the GnuPG files in an external USB drive, or an encrypted volume in the local hard disk. A more complex solution involves the use of a smart card.

12.3.1. External USB drive

First, mount the external drive and move there all GnuPG files (your keyring, the random seed file, and configuration files) that were contained in the default directory. The mailclient must not be running while you move the files.

Then, you must tell GnuPG where the new location is, by passing the additional parameter `--homedir new_location` to the GnuPG executable. This is done directly inside the OpenPGP Preferences, via the menu command *OpenPGP* → *Preferences* → *Advanced*, in the field *Additional parameters for GnuPG*. How to do this was explained in Section 9.1.4.

For instance, the figure at page 72 shows Enigmail configured as to have GnuPG look for the keyrings in the `X:\gnupg` directory.

Once you have done this, you can use Enigmail in the usual way. Remember to have your external drive mounted before running Enigmail or GnuPG.

12.3.2. Encrypted volume

You may store the GnuPG files on, instead of an external drive, an encrypted virtual volume in the local hard disk (or even, for extra protection, an encrypted virtual volume on an external drive itself).

There are many on-the-fly encryption programs available, some of the most known ones being:

- PGP Whole Disk Encryption (commercial), <http://www.pgp.com>
- TrueCrypt (open-source and free), <http://www.truecrypt.org>
- FreeOTFE (open-source and free), <http://www.freeotfe.org>

The encrypted virtual volume will behave just like an external drive. Once you have installed the encryption program of your choice, created the encrypted virtual volume, and mounted it, do the necessary setup by following the same steps explained in Section 12.3.1.

12.3.3. OpenPGP card

Enigmail supports the OpenPGP card, a smart card compatible with ISO standards 7816-4 and 7816-8; see <http://g10code.com/p-card.html>. The figures below show front and back of an OpenPGP card:



OpenPGP cards are distributed by Kernel Concepts, <http://www.kernelconcepts.de>. It is also possible to obtain a OpenPGP card by becoming a Fellow of the Free Software Foundation Europe; please read http://wiki.fsfe.org/Crypto_Card for more information.

OpenPGP v2.0 cards feature three independent RSA keys, for signing, encryption, and authentication, of up to 3072 bits each. The card is used to store the actual secret key. A secret key stub remains in the secret keyring to permit standard key operations. The purpose of using a smart card is that the secrets it contains cannot be copied from the card. Therefore, as long as the card stays physically in your possession, you know that your secret key is safe.

There are two methods to initialize a card.

Following the first method, the key is generated on-card, i.e. the card calculates the key using its built-in random generator; in this way the secret key never leaves the card.

Otherwise, a standard RSA key can be generated in a safe environment, e.g. a

clean Linux workstation not connected to any network and booted from a CD-ROM. The secret key is then moved to the card.

Enigmail only supports on-card key generation. If a key is generated on-card, it is possible to create a copy of the secret encryption key (only). This key can later be restored to another OpenPGP card if the original card gets lost or broken. However, the new card will have new signing and authentication keys.

For advanced users: the method that guarantees the maximum availability of the keys, at the expense of secrecy, is to create a compatible key. This is done by creating via the GnuPG command line (use the `--expert` flag) keys with distinct functionalities (1024-3072 bit, RSA only).

These keys allow you to backup a fully functional key, for which no card is needed, which is helpful in case you revoke your card key but still want your mail archive to be readable.

You can also create a full clone of that key on another card if availability is vital. As long as you protect your original backup key appropriately, this allows you to leave your card in a system managed by someone else without the fear that your secret key could be stolen unnoticed. In fact, since the secret key cannot be copied from the card, the only way to pick up the key is to physically steal the card – which you'll notice.

From the menu item *OpenPGP → Manage SmartCard...* you can access all smart card operations:

- manage the user data (name, sex, language, login ID, URL of the public key) stored on the OpenPGP card;
- generate a new key on-card;
- change your PIN (123456 by default) and Admin-PIN (12345678 by default).



Generating a new key on-card will overwrite the pre-existing key.

Remember to change your PIN and Admin-PIN before generating a new key. The PIN is not restricted to digits only but can be any combination of characters; choose strong PINs since they are the only protection to the secret key if the card is lost or stolen. However, bear in mind that non-numeric PINs cannot be entered on PIN-pad readers.

It is strongly recommended that you test to recover your secret keys (both your card and the key on your local machine) from a backup key and a blank card. If you have only one card available, you may still simulate the recover (v2.0 cards only) by resetting the card via the command

```
gpg-connect-agent < resetfile
```

where *resetfile* is an ASCII text file composed of the following lines:

```
/hex
scd serialno
scd apdu 00 20 00 81 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 81 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 81 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 81 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 83 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 83 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 83 08 40 40 40 40 40 40 40 40
scd apdu 00 20 00 83 08 40 40 40 40 40 40 40 40
scd apdu 00 e6 00 00
scd apdu 00 44 00 00
/echo card has been reset to factory defaults
```

13. SUPPORT

This handbook, once read in full, should answer all questions you might have about Enigmail and give you a thorough understanding of it. You can always find the most up-to-date information (and Enigmail executable) directly on the Enigmail official web site:

`http://enigmail.mozdev.org`

Should you experience problems or want to ask further questions, you can obtain support (provided by volunteers) from several places: there is a Forum, a mailing list, and a newsgroup dedicated to Enigmail. For information about how to access them, visit

`http://enigmail.mozdev.org/support`

If your question is not strictly related to Enigmail but rather to broad OpenPGP topics (e.g. signing, encryption, key management, trust...), you may also address yourself to the GnuPG resources. GnuPG documentation and support is available at

`http://www.gnupg.org/documentation/index.en.html`