Mutt−i, GnuPG and PGP Howto

Table of Contents

Mutt−i, GnuPG and PGP Howto

Andrés Seco AndresSH@ctv.es and J.Horacio M.G. homega@ciberia.es

v1.2, February 2000

This document briefly explains how to configure Mutt−i, PGP and GnuPG in its diferents versions (2.6.x, 5.x and GnuPG), noting the common problems that can occur while sending signed or encrypted mail to be read by mail clients not PGP/MIME compliants as defined in RFC2015 and in other operating systems. It also includes an example of procmail configuration to send the public keys automatically to received e−mails asking for it, as a key servers does.

6.PGP and Mutt integration

7.Interesting Macros for Mutt

8.Procmail notes and tips

9.Interchanging signed/encrypted messages with different MUAs and platforms

10.Programs and versions used

11.More information

1.Introduction

This document explains how to configure *Mutt−i*, *PGP* and *GnuPG* in its diferents versions (2.6.x, 5.x and GnuPG) to quickly start using a mail reader with encryption and digital signing capabilities.

For this purpose, example configuration files will be included to help you starting with it. To obtain maximum performance and to use all the features of the programs that we will be using, it will be necesary to read its documentation and to reconfigure the example files.

Also, some problems derived from not using RFC2015 about PGP/MIME by many mail user agents in Linux and other operating systems will be comented.

An aditional procmail configuration example will be showed to enable our mail client to send a public key on request.

This document has been translated from the Spanish original by Andrés Seco AndresSH@ctv.es, and revised and corrected by Jordi Mallach Pérez jordi−sd@softhome.net and J.Horacio M.G. homega@ciberia.es. It was finished in October 1999. We would like to thanks Roland Rosenfeld roland@spinnaker.de, Christophe Pernod xtof.pernod@wanadoo.fr, Denis Alan Hainsworth denis@cs.brandeis.edu and Angel Carrasco acarrasco@jet.es for their corrections and suggestions.

2.Copyright and discharge of responsability

Information and other contents in this document are the best of our knowledge. However, we may have make errors. So you should determine if you want to follow the instructions given in this document.

Nobody is responsible for any damage in your computers and any other loss derived from the use of the information contained herein.

THE AUTHORS AND MAINTAINERS ARE NOT RESPONSIBLE FOR ANY DAMAGE INCURRED DUE TO ACTIONS TAKEN BASED ON INFORMATION CONTAINED IN THIS DOCUMENT.

Of course, we are open to all type of suggestions and corrections on the content of this document.

3.Sending mail to and receiving mail from the internet

This document does not deal with exchanging mail messages between local machine and other nodes (inside a local area network or over the internet). This exchange should be carried out by messages transfer agents (MTAs) such as `sendmail`http://www.sendmail.org, `qmail`http://www.qmail.org, `exim`http://www.exim.org, `smail`ftp://ftp.planix.com/pub/Smail, etc.

In this document it is presupposed that this method of send/receive messages outside of the local computer is already installed and working in a correct way. If you can send a message and read your mail with the `mail` command from the command line in your computer,

```
$ mail −s <subject> <user@domain.net>
```

```
write here the text, and finish with an alone point in the
next line
.
```

you must have installed any type of MTA that is doing the messages transfer. In other way, you can get documentation about setting it up in the manual pages of *smail*:

```
$ man smail
```

or the MTA that you have, and *fetchmail*:

```
$ man fetchmail
```

or in other similar document that makes reference to those programs.

4.Mutt configuration

Next file is a valid example to start using *Mutt* in a basic way, including paths for alias file, sent messages and postponed messages. You can further personalize it attending to the *Mutt* manual indications and `/usr/doc/mutt/` or `/usr/doc/mutt−i/`.

Simple example of `~/.muttrc`:

```
set folder=~/Mail
set alias_file=.alias
set postponed=.postponed
set record=SendMessages
set signature=.signature
my_hdr From: Name Surname <Name@domain.com>
source =.alias
```

It is necesary that the directory `~/Mail` exists, that is the one that appears as an "equal to" sign in the configuration file `.muttrc` (that is, `=.alias` is to *Mutt* as `~/Mail/.alias`, and `=.postponed` is to *Mutt*`~/Mail/.postponed`). Nevertheless it is possible to have these files in another directory provided we indicate the complete path in `~/.muttrc`, and we have the necesary permissions to work in this directory.

It is also necesary to personalize the `my_hdr` line with the name and electronic mail address you need. In the `~/Mail/.signature` file you caninclude the signature that will appear in all the messages that are sent.

This configuration file can end up being made very big, so it is common to separate some of its commands in different files. For the time being, the *PGP* or *GnuPG* configuration lines are easily detachable, and the keyboard macros that we will personalize. To do that, it will be necesary to add the following lines to the `~/.muttrc` file:

```
source = ~/Mail/.mutt.macros
source = ~/Mail/.gnupgp.mutt
```

and to use the `~/Mail/.mutt.macros` and `~/Mail/.gnupgp.mutt` files to put in them the keyboard macros and the *PGP* or *GnuPG* configuration that are commented forward.

To get a more extensive and complete information over the use and configuration of *Mutt*, and about advanced features, see the Mutt manual http://www.mutt.org.

5.PGP and GnuPG

To use anyone of the versions of *PGP* with *Mutt−i*, first it will be necesary to configure *PGP* properly in the way that the public keys file (public keys ring) and the private keys file (private keys ring) will exist. It is convenient to previously test PGP from the command line to assure that it signs and encrypt correctly.

Remember that the *PGP* versions that exist for *Unix* are `2.6.3(i)` and `5.0(i)`, that we call **PGP2** and **PGP5** respectively forward. **GnuPG** is a new encrypt system, being developed in these days, in an advanced state of development, open source and free, in many aspects better than **PGP** (see GnuPG mini howto http://www.dewinter.com/gnupg_howto).

We will also clarify that *PGP*, as being a program developed in the US, is restricted by certain exporting laws about programs that include cryptographic code; this is the reason for the existance of an international version to almost all binary versions, and it is noted with the "**i**" letter (**pgp − pgpi**).

5.1 PGP2

PGP2 generates keys with the RSA http://www.rsa.com,algorithm and it uses IDEA http://www.ascom.ch as the encryption algorithm. Both are propietary algorithms and its use is restricted by its respectives patents.

To run it correctly, you must have it installed, as well as having a directory called `~/.pgp`, containing the configuration file `pgp−i.conf` and the private and public keys rings files, `pubring.pgp` and `secring.pgp` respectively.

5.2 PGP5

The keys generated by *PGP5* are **DSS/DH** (Digital Signature Standard / Diffie−Helman). PGP5 uses **CAST**, **Triple−DES**, and **IDEA** as encrypt algorithms. PGP5 can work with encrypted or signed data with *RSA* (PGP2), and use that keys to sign or encrypt (with the keys generated with PGP2, because PGP5 can not generate that type of keys). In the other hand, PGP2 can not use the *DSS/DH* keys from PGP5; this creates incompatibility problems, because many users continue using PGP2 with *Unix/Linux*.

To run PGP5 correctly, in the `~/.pgp` directory you will have the public and private key rings (`pubring.pkr` and `secring.skr` respectively), and the configuration file `pgp.cfg`.

In the case that you have installed the both versions of *PGP* (PGP2 installed and configured before PGP5), we will create the configuration file `~/.pgp/pgp.cfg` of PGP5 as a simbolic link to the `~/.pgp/pgp-i.conf` configuration file,

```
~/.pgp$ ln -s pgp-i.conf pgp.cfg
```

adding the following lines at the end of the file `~/.pgp/pgp-i.conf`:

```
PubRing = "~/.pgp/pubring.pkr"
SecRing = "~/.pgp/secring.skr"
RandSeed = "~/.pgp/randseed.bin"
```

The files with the keys rings of the different versions can cohexist without any problem in the same directory.

5.3 GnuPG

GnuPG is a program with the same functions that the previous. The difference with *PGP*, *GnuPG* do not uses algorithms with restrictive patents. *PGP* is free for personal uses but not comercial jobs and its development is closed. *GnuPG* is free to be used in any job and it is open source, as our favorite operating system (also its implementation and development is made mainly in *Linux*).

The keys generated by *GnuPG* are of the type **DSA/ElGamal** (*Digital Signature Algorithm*, also known as *DSS*). Is totaly compatible with *PGP*, except with the use of restricted patents algorithms *RSA* and *IDEA*. Anyway, it is posible to implement certain compatibility with that (see GnuPG mini howto http://www.dewinter.com/gnupg_howto to get it interacting with PGP2 and PGP5).

6.PGP and Mutt integration

The operation to carry out in the outgoing messages (sign, encrypt or both) is chosen exactly before presing "`y`" to send the message, inside the option menu that is visible with the "`p`" option. Once you have choosen the operation to carry out, only the line *PGP* in the message header showed in the screen will change, but until you send the message with "`y`" you won't be asked to insert the pass phrase to activate the sign of the message or the public keys to use to encrypt in the case that no receptors were found in our public keys ring.

NOTE: In the case that the pass phrase was mistyped when it was asked for, *Mutt* seems to be "hung", but that's not true, it is waiting for it to be retyped. To do this, push the `<Enter>` key and delete the pass phrase from memory with `<Ctrl>F`. Next we repeat the message sending with ("`y`") and retype the pass phrase.

Through this procedure, *Mutt* will use *PGP/MIME* to send the message, and one more file will appear in the list of files to be sent with the sign (if we only select to sign) or it will encrypt the complete message (all its *MIME* parts) and it will only leave two MIME parts, the first with the PGP/MIME version and the second with the encrypted message (with all its MIME parts inside) and signed (if we selected to do it).

Note: By some reasons, if the receptor mail user agent can not use *MIME*, we may need that the sign will be included inside the message body. See section about *application/pgp* with PGP5 and with GnuPG.

Mutt will try to verify the sign or decrypt automatically the incoming messages that use *PGP/MIME*. See section Procmail notes and tips, in which it is commented how to change the *MIME* type automatically to the incoming messages that do not set its *MIME* type correctly.

6.1 Optional configuration files

In the next sections you can find modifications to the *Mutt* configuration file to use PGP2, PGP5, and GnuPG easily.

To do that, a new configuration file that we called `.gnupgp.mutt` (that's our name, you can call it any other name setting the name of this file into the main configuration file `~/.muttrc`).

This can be done including the complete path (its location) of the configuration file `.gnupgp.mutt`, in a line at the end of the `~/.muttrc` file. The directory in which we put this and other optional configuration files can be anywhere, if we have correct permissions (in a previous section we included it inside the `~/Mail/`) directory, or any other inside our home directory, with any name:

```
~$ mkdir mutt.varios
```

in which we copy (or create) the optional configuration file `.gnupgp.mutt`, and next we set the origin of this file in the `.muttrc` file with the `source` command, like the following:

```
source ~/mutt.varios/.gnupgp.mutt
```

Now *Mutt* will accept configuration variables in `.gnupgp.mutt` as if it were in `.muttrc` directly.

This method is a good way to avoid having a very big, unsorted configuration file, and can be used to set any other group of configuration variables in other separate file. For example, as before, if we use *vim* as the default editor in *Mutt*, we can tell to `.muttrc` to use a different configuration file `.vimrc` that we use when using *vim* from the command line. First, copy `~/.vimrc` to our optional configuration files directory `~/mutt.varios/` and set it with other name (ex. `vim.mutt`):

```
$ cd /home/user ~$ cp .vimrc mutt.varios/vim.mutt
```

next change the configuration variables that we want to be different in *vim* as the *Mutt* editor, and finally modify `.muttrc` to reflect this change:

```
set editor="/usr/bin/vim −u ~/mutt.varios/vim.mutt"
```

With this last line we are setting `Mutt` to use an external editor, *Vim*, with the needed configuration options.

6.2 General Configuration Variables

There are some variables that we will use globally with the three public key encrypt programs with *Mutt*. These variables are boolean, and can be **set** (activated) or **unset** (deactivated).

In the configuration file (`~/.muttrc`, or `~/mutt.varios/.gnupgp.mutt`, or whatever you use), the sign (**#**) is a comment and will be ignored. So, we will use it from here in advance to comment each variable:

unset pgp_autosign

if this variables is set, *Mutt* will ask to sign all the
outbound messages. (1)

unset pgp_autoencrypt

if this variable is set, *Mutt* will ask to encrypt all the
outbound messages. (1)

set pgp_encryptself

save an encrypted copy of all sent messages that we want to encrypt
(need the general configuration variable `set copy=yes`).

set pgp_replysign

when you answer a signed message, the response message will be

signed too.

set pgp_replyencrypt

when you answer an encrypted message, the response message
will be encrypted too.

set pgp_verify_sig=yes

Do you want to automatically verify incoming signed messages?
Of course!

set pgp_timeout=<n>

delete pass phrase from the memory cache <n> seconds
after typing it. (2)

set pgp_sign_as="0xABC123D4"

what key do you want to use to sign outgoing messages?
Note: it is posible to set it to the user id, but
this can be confuse if you have the same user id with different keys.

set pgp_strict_enc

use "quoted−printable" when PGP requires it.

unset pgp_long_ids

Do not use 64 bits key ids, use 32 bits key ids.

set pgp_sign_micalg=<some>

message integrity check algorithm, where
<some> is something from the next: (3)
pgp−mda5
to RSA keys
pgp−sha1
to DSS (DSA) keys
pgp−rmd160

In the three next sections the configuration variables to each of the PGP versions will be explained. The fourth section will explain how to modify the variables if you use more than one PGP version.

(1) as *Mutt* requires to type the passphrase every time you want to sign or select the receipts if you want to encrypt, it may be unconvenient to set this variable. Possibly you may want to unset this variable. This is specially true encrypting messages, as you don't have all the public keys of the message receipts.

(2) depending on the number of messages that we sign or decrypt, we would like to maintain the pass phrase in cache memory more or less time. This option avoid you from type the pass phrase each time you sign a new message or decrypt an incoming message. **Warning:** maintaining the pass phrase in cache memory is

not secure, specially in network connected systems.

(3) this is only necesary with the key that we use to sign. When the key is selected from the compose menu, *Mutt* will calculate the algoritm.

6.3 PGP2 configuration variables

To use PGP2 with *Mutt−i* you need to add the following lines to the `~/mutt.varios/.gnupgp.mutt` file:

```
set pgp_default_version=pgp2
set pgp_key_version=default
set pgp_receive_version=default
set pgp_send_version=default
set pgp_sign_micalg=pgp-md5
set pgp_v2=/usr/bin/pgp
set pgp_v2_pubring=~/.pgp/pubring.pgp
set pgp_v2_secring=~/.pgp/secring.pgp
```

As you know, the `~/.pgp/pubring.pgp` and `secring.pgp` files must exist. More information on PGP2 with the `man pgp` command.

6.4 PGP5 configuration variables

To use PGP5 with *Mutt−i* you need to add the following lines to the `~/mutt.varios/.gnupgp.mutt` file:

```
set pgp_default_version=pgp5
set pgp_key_version=default
set pgp_receive_version=default
set pgp_send_version=default
set pgp_sign_micalg=pgp-sha1
set pgp_v5=/usr/bin/pgp
set pgp_v5_pubring=~/.pgp/pubring.pkr
set pgp_v5_secring=~/.pgp/secring.skr
```

As you know, the `~/.pgp/pubring.pkr` and `secring.pkr` files must exist. More information on PGP 5 with the `man pgp5` command.

6.5 GnuPG configuration variables

To use *GnuPG* with *Mutt−i* you need to add the following lines to the `~/mutt.varios/.gnupgp.mutt` file:

```
set pgp_default_version=gpg
set pgp_key_version=default
set pgp_receive_version=default
set pgp_send_version=default
set pgp_sign_micalg=pgp-sha1
set pgp_gpg=/usr/bin/gpg
set pgp_gpg_pubring=~/.gnupg/pubring.gpg
set pgp_gpg_secring=~/.gnupg/secring.gpg
```

As you know, the `~/.gnupg/pubring.gpg` and `secring.gpg` files must exist. More information on GnuPG with the `man gpg.gnupg`, `man gpgm`, and `man gpg` commands.

6.6 Mixed configuration variables

If you want to use more than one PGP software you need to modify some of the variables that we have commented previously. Really, it is only to remove the redundant version variables.

If, for example, you want to use GnuPG as the default signing tool, all menu commands in *Mutt* to use GnuPG/PGP would call to this program to the signing, decrypting, encrypting, verifying, etc... operations To do that you must set the configuration variable `$set_pgp_default`**once**, so:

```
set pgp_default_version=gpg
```

now, to use the all three programs, the `~/mutt.varios/.gnupgp.mutt` file could be like this:

```
set pgp_default_version=gpg     # default version to use

set pgp_key_version=default     # default key to use
                                # in this case, gnupg defines it

set pgp_receive_version=default # default version to decrypt will be the default
set pgp_send_version=default    # version defined in the first line (gpg)

set pgp_gpg=/usr/bin/gpg        # where to find the GnuPG binary
set pgp_gpg_pubring=~/.gnupg/pubring.gpg        # public key file to GnuPG
set pgp_gpg_secring=~/.gnupg/secring.gpg        # secret key file to GnuPG

set pgp_v2=/usr/bin/pgp         # where to find the PGP2 binary
set pgp_v2_pubring=~/.pgp/pubring.pgp           # public key file to PGP2
set pgp_v2_secring=~/.pgp/secring.pgp           # secret key file to PGP2
```

```
set pgp_v5=/usr/bin/pgp         # where to find the PGP5 binary
set pgp_v5_pubring=~/.pgp/pubring.pkr           # public key file to PGP5
set pgp_v5_secring=~/.pgp/secring.skr           # secret key file to PGP5
```

7.Interesting Macros for Mutt

Mutt is highly configurable and its working mode can be modified in a very flexible manner if the configuration variables inside `.muttrc` are well configured.

Here you can see some macros that help you to generate signed messages avoiding the *PGP/MIME* standard, to send it to receipts that don't support this type of signed messages following the *PGP/MIME* standard, and to edit the alias file and reload it without exiting *Mutt* (this last macro is not related to *PGP/GnuPG*, it is presented only as an example to show the macro power in *Mutt*).

It is possible to tell Mutt the key bindings you want to use with *PGP/GnuPG*. Even when some of this options are yet configured, we can change it or add others easily modifiying the configuration file.

7.1 Signing on the message body without using PGP/MIME with PGP5

Before existing *PGP/MIME*, the signature in a message was included in the message body. This is a very common form of sending signed messages in many mail user agents.

If we want to sign like this, we have two options, leave the *MIME* type of the message or modify it as `application/pgp`.

To implement this two forms of signing in *Mutt*, we will add the following lines to the `~/mutt.varios/mutt.macros` file. Previously, we have to set this option file path in the `.muttrc` main configuration file (see Optional configuration files):

```
macro   compose \Cp     "F/usr/bin/pgps\ny"
macro   compose S       "F/usr/bin/pgps\ny^T^Uapplication/pgp; format=text; x-action=sign\n"
```

and now, pressing `<Ctrl>p` or `S` we can include the signature into the message part that has the cursor on it, just before send the message.

7.2 Signing on the message body without using PGP/MIME with GnuPG

As in the previous case, but with GnuPG. The macros are:

```
macro   compose \CP     "Fgpg --clearsign\ny"
macro   compose \CS     "Fgpg --clearsign\ny^T^Uapplication/pgp; format=text; x-action=sign\n"
```

7.3 Modifying the alias file and reloading it

With this macro included in `~/mutt.varios/macros.mutt` you can edit with *vi* (changing the line you can use other editor) the alias file without exiting *Mutt* pressing `<Alt>a`.

```
macro   index   \ea     "!vi ~/Mail/.alias\n:source =.alias\n"
```

7.4 More macro examples

The next listing has been obtained from Roland Rosenfeld and it shows macros to change the default signing/encrypting software and to sign without PGP/MIME with GnuPG:

```
# ~/Mail/.muttrc.macros
# keyboard configuration file for Mutt-i
# copied, modified and translated from the original:
#
################################################################
# The ultimative Key-Bindings for Mutt                         #
#                                                              #
# (c) 1997-1999 Roland Rosenfeld <roland@spinnaker.rhein.de>   #
#                                                              #
# $ Id: keybind,v 1.36 1999/02/20 19:36:28 roland Exp roland $ #
################################################################
#
# To use it, add the next line to ~/.muttrc:
# source ~/Mail/.muttrc.macros
#

# Generic keybindings
# (for all the Mutt menus, except the pager!)
# With the next three we can change the encrypting default selected software:

# <ESC>1 to use GnuPG
macro   generic \e1     ":set pgp_default_version=gpg ?pgp_default_version\n"\
"Switch to GNU-PG"
```

```
# <ESC>2 to use PGP2
macro   generic \e2     ":set pgp_default_version=pgp2 ?pgp_default_version\n"\
"Switch to PGP 2.*"

# <ESC>5 to use PGP5
macro   generic \e5     ":set pgp_default_version=pgp5 ?pgp_default_version\n"\
"Switch to PGP 5.*"

#NOTE: Be careful with the last backspace at the end of the previous
macros. If you write that line and the next in the same line, do not write
it.

# index, OpMain, MENU_MAIN
# (Main menu)
# The next macro only runs from the main menu (the one that appears when
# you starts Mutt). The keys <CTRL>K permit us to extract the public keys
# from a message if it has (this is known because it has the K letter in
# the message line):

macro  pager   \Ck     ":set pipe_decode pgp_key_version=pgp2\n\e\ek:set pgp_key_version=pgp5\n\e\ek:set pgp_key_version=gpg\n\e\ek:set pgp_key_version=default nopipe_decode\n"\ "Extract PGP keys to PGP2, PGP 5, and GnuPG keyrings"


# pager, OpPager, MENU_PAGER
# (Pager menu)
# It permits the same operations that previous, with the same key combinations,
# but in this case from the pager menu:

macro   pager   \e1     ":set pgp_default_version=gpg ?pgp_default_version\n"\
"switch to GNUPG"

macro   pager   \e2     ":set pgp_default_version=pgp2 ?pgp_default_version\n"\
"switch to PGP 2.*"

macro   pager   \e5     ":set pgp_default_version=pgp5 ?pgp_default_version\n"\
"switch to PGP 5.*"


# compose, OpCompose+OpGerneric, MENU_COMPOSE
# (Compose menu)
# The next operations are used from the compose menu.
# That is, after you have composed your message and you close it to send it,
# just before pressing the "Y" key that allows us to send it to the MTA.

# In this case, we create a menu that appears when you press "P".
# The options in this menu are going to be bound to MENU_PGP. This are the
# main use options (encryption and signing).

bind    compose p       pgp-menu

# As many programs can't use PGP/MIME (especially from M$), the <CTRL>P key
# will allow us to sign "as in the old times" (Application/PGP):

macro   compose \CP     "Fgpg --clearsign\ny"

# The next, <CTRL>S will allow us to sign using PGP/MIME with the private key
# that we have defined as default. This macro is not necesary, as we can
# do the same from the "P" menu:
macro   compose \CS     "Fgpg --clearsign\ny^T^Uapplication/pgp; format=text; x-action=sign\n"
```

You can add more macros, and some other are yet configured as default in newer versions of Mutt. Some other options include:

- <CTRL>K (extract public keys from a message)
- <ESC>K (adjunt a public key to a message)
- <CTRL>F (when using the key phrase to sign or decrypt a message, it is still in memory. With this you can delete it from memory)
- etc...

To see what other options are activated, you must go to the help menu (?) from the menu where you were.

8.Procmail notes and tips

8.1 Configuring Procmail to send automatically your public keys

As this is not the objetive of this Howto, we will comment that the securest way to get the public key from anybody is that he gives it to us directly by hand.

As many times this is not an easy method (how long they are) the people can send the public key by electronic mail, or searching it in a key server, but none of those methods assure that the obtained key is really from whom it seems to be. If you use other communication media considered "secure" (searching the owner in the phone listing and asking him to read his key "fingerprint" to contrast with the fingerprint from the key we have obtained from the non−secure path).

What we are going to see is a "tip" to put into the `.procmailrc` from the Procmail mail processor to get back automatically your publick key to the remitent when you get a message with a determined text in the `Subject` line:

```
:0 h
* ^Subject:[ 	]+\/(|send)[ 	]+key pub\>.*
| mutt −s "Re: $MATCH" `formail −rtzxTo:` </clau/mykey.asc
```

What it is said in the previous paragraph is: we have a copy in ASCII of our public key, in any directory (in this case the `/clau` directory) in a file named `mykey.asc`; when procmail gets a message that include "send key pub" in the `Subject:` line, send the file to the remitent.

IMPORTANT: what you have between the brackets is **an space** and **a tab**.

8.2 Verify and decrypt automatically messages without PGP/MIME

When you receive a signed message that uses PGP/MIME and you open it with your preferred MUA (Mutt, isn't it?), it recognizes the message as PGP/MIME and checks the signature if you have the remitent public key. These messages are the ones that have the "S" in the first part of the message line in Mutt:

```
36  S  05/09 Andres Seco Her ( 12K) Al fin
```

while the encrypted messages have the "P":

```
12  P  03/24 Andres Seco Her (6,3K) Re: FW: Re: Mutt − pgp/gnupg
```

But if the message is signed and has the "application/pgp" MIME type, when you open it Mutt doesn't check its sign, and this sign is into the message body, as here:

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

Date: Tue, 25 May 1999 13:04:26 +0200
From: La Corporación <bill@reboot.com>
Subject: Actualización S.O.
To: Sufrido Usuario <pepe@casa.es>


Sufrido usuario:

le comunicamos que puede usted adquirir la última actualización del
programa O.E. con la adquisición de nuestro sistema operativo reboot99
por el módico precio de ... etc.


-----BEGIN PGP SIGNATURE-----
Version: 2.6.3ia
Charset: noconv

iKBGNpUBX0235VapRBUy1KklAQGl9wQA3SBMio0bbbajHAnyKMOlx3tcgNG7/UVC
AbqXcUnyGGOo13Nbas95G34Fee3wsXIFo1obEfgiRzqPzZPLWoZdAnyTlZyTwCHe
6ifVpLTuaXvcn9/76rXoI6u9svN2cqHCgHuNASKHaK9034uq81PSdW4QdGLgLoeB
vnGmxE+tGg32=
=Xidf
-----END PGP SIGNATURE-----
```

To verify it, you must save it and use the command line. But, it is possible to convert this MIME messages type with *Procmail* to allow *Mutt* to recognize it as *PGP/MIME*. You only need to add this to `.procmailrc`:

```
:0
* !^Content-Type: message/
* !^Content-Type: multipart/
* !^Content-Type: application/pgp
{
    :0 fBw
    * ^-----BEGIN PGP MESSAGE-----
    * ^-----END PGP MESSAGE-----
    | formail \
        -i "Content-Type: application/pgp; format=text; x-action=encrypt"

    :0 fBw
    * ^-----BEGIN PGP SIGNED MESSAGE-----
    * ^-----BEGIN PGP SIGNATURE-----
    * ^-----END PGP SIGNATURE-----
    | formail \
        -i "Content-Type: application/pgp; format=text; x-action=sign"
}
```

As you can see, this is valid to signed messages and to encrypted messages with application/pgp.

8.3 Change MIME type for messages with keys inside without PGP/MIME

When you receive a public key block from a non *PGP/MIME* compliant MUA, you must save the message body in your disk and then insert it into your public key ring, but, including this lines into your `.procmailrc` file, you can include it directly from mutt.

```
    :0 fBw
    * ^-----BEGIN PGP PUBLIC KEY BLOCK-----
    * ^-----END PGP PUBLIC KEY BLOCK-----
    | formail -i "Content-Type: application/pgp-keys; format=text;"
```

Thanks to Denis Alan for this procmail note.

9.Interchanging signed/encrypted messages with different MUAs and platforms

In the first days, the PGP sign was included inside the text to sign. Later, it was included the `application/pgp` MIME type to show that the next attach was the sign or the encrypted PGP message, and finally, with the PGP/MIME specification, it was possible to isolate the sign from the original affected, to not modify absolutelly and somebody that didn't have PGP could view the message as it was originally (only for signed messages), without any added text in the beginning or in the end from PGP.

The actual situation is that only a few mail user agents (MUAs) are capable to integrate PGP to use the PGP/MIME standard, and it is necesary to send messages using the old time PGP sign when you know that the recipient doesn't recognize PGP/MIME.

In Linux, the available mail user agents that are PGP/MIME compliant are mutt−i and pine. In Windows, only the Eudora mail client versions 3.x and 4.x can use PGP/MIME. If you know any other mail user agent that supports it, tell us by mail, to include it here.

10.Programs and versions used

To write this document we have used the next Mutt versions:

- Mutt 0.93i − you can not use GnuPG with this version.
- Mutt 0.95.3i − all PGP and GnuPG versions can be used.

And the next PGP and GnuPG versions:

- PGPi 5.0
- GnuPG 0.4.3
- GnuPG 0.9.4

11.More information

The original documentation from where this document has been obtained can be found in the man pages from "mutt", "pgp", "pgp5", "gnupg", "procmail", in the respectives directories in /usr/doc and in the world wide web sites:

- Mutt Official Home Page − http://www.mutt.org
- GnuPG Main Page − http://www.gnupg.org
- PGP International Page − http://www.pgpi.com
- Procmail Official Home Page − http://www.procmail.org

The recommendations (request for comments, RFC) that are referenced in this document are:

- 1847 − Security Multiparts for MIME: Multipart/signed and Multipart/encripted
- 1848 − MIME Object Security Services
- 1991 − PGP Message Exchange Formats
- 2015 − MIME Security with Pretty Good Privacy (PGP)
- 2440 − OpenPGP Message Format

and can be found in /usr/doc/doc−rfc and in various sites in the world wide web, like http://metalab.unc.edu and http://nic.mil. You can get information from RFCs in RFC−INFO@ISI.EDU
