# PGP-BODY: Enigmail companion for servers.

What if your housing ISP will read your mailbox , sited on your mail server? What if someone can get your mail credentials? What if your company's IT employees can read your mail since they have access to the server? What if your laptop was stolen? Pgp/gpg was the answer. It requires anyone is writing you to have GPG/GPG. Then it was *a poor* answer. Is there a way to ALWAYS encrypt the email BEFORE it reaches the mailbox? Is there any Enigmail for servers? Is it opensource?

Actually, yes. I wrote a small piece of code. It will work like anyone writing to you had a working Enigmail. The job is done on your mail server. It happens before of local delivery, so the message goes encrypted in the inbox.

Then, since you have the private key on your client, everything is totaly transparent. If someone has unwanted access to the server (or to your mailbox) , it will read nothing.

PGP-BODY works on postfix , fetchmail, .forward, procmail. (1)

**Pgp-Body works between standard input and standard output.**

I assume to integrate it on *exim, qmail, courier, sendmail* should not be that hard. Anyhow , let me know if you have any problem with it.

What the software is doing?

- It reads the full email from the standard input.
- It keeps the headers without encrypting.
- It takes away any original mime headers, putting them is the body payload. (as per s/mime spec).
- It adds the s/mime headers for the PGP content.
- It encrypts the body with the specified public key.
- It adds the s/mime enclosure to the body.
- It returns a PGP s/mime encrypted message to the standard output.
- The result is totally transparent to Enigmail.

Let's see how to use it.

## Some assumptions:

- GNU gpg installed on /usr/bin . I don't trust your local install. I don't trust your $PATH. Root have to install it properly. Full stop.
- You generated private and public key for your email.
- Public keys are available somehow to gpg (local install or keyserver)
- You have a GNU C++ compiler and automake tools (configure, etc).

## Some easy steps to install (as root):

```
Download the tarball from the link at the bottom of this page
tar -xvzf  gpg_body-XY.tar.gz
cd  gpg_body-XY
```

```
./configure
make
make install
```

## Configure for postfix (easy way , no keyserver)

In your main.cf , you can add:

```
mailbox_command = /some/where/gpg_body -r "$RECIPIENT"          |
/some/where/maildrop -d "$USER"  or

mailbox_command = /some/where/gpg_body -r "$ORIGINAL_RECIPIENT" |
/some/where/maildrop -d "$USER" or

mailbox_command = /some/where/gpg_body -r "$LOCAL"@"$DOMAIN"    |
/some/where/maildrop -d "$USER"
```

pgp_body acts by default as "gpg -ae" . Anything else you put in the command line **is appended** to the "gpg -ae" command, like the -r option.

This means it will need at least one argument: *-r recipient*

How to specify the recipient  depends by your postfix installation, rewriting rules and so.

Please notice pgp_body is not a local delivery agent, **so you still need one**. I assumed maildrop in my example, but you can choose the one you like. Of course, you can make the example a bit more flexible using **mailbox_command_maps**

## Configure for postfix (a bit harder ,  keyserver)

now, maybe you don't want to store all the public keys, for any reason. As I write, pgp_body will attach any command line to the "gpg -ea" command. Then, you can specify any keyserver to retrieve keys from. Any combinations of **--keyserver yourserver.tld , --keyserver-options , --recv-keys , --search-keys** will work.

Just add the parameter as shown in the "easy way" example, invoking gpg_body with the same parameters you whould have used if using gpg itself to retrieve the public key from the server.

Should you do it?

- **PRO:** users can manage their keys by themselves. They are in charge of keeping the keys updated , valid and available. If any problem, they will receive empty mails. You can also specify a default user in the parameters, so when the key is not available, a default one is used. (Hopefully , all users installed all the keys then need).
- **CONS:** when a user stops keeping in order its key, it will only receive empty emails. Also, if you are using a public server, you are introducing more network latency. This could not impact a low traffic server, while harming  high-traffic servers. If you have a local company's keyserver, it is probably ok about latency. You are the sysadmin, you know.

Now, maybe you are not root in your linux server, and  imagine you want it, as a user. You can use .forward files in your home directory. Just download pgp_body, compile it and put somewhere in a folder.

## Configure for local user editing your ~/.forward file:

```
|/some/where/gpg_body -r myname@domain.tld          | /some/where/maildrop
```

 maildrop with no arguments should  deliver the mail to your mailbox anyhow, just checking for your user. For any other local delivery, like procmail or so, act accordingly using the same schema.

## Configure for fetchmail:

Let's imagine your .fetchmailrc looks like this:

```
#### .fetchmailrc

 set daemon 600
 set logfile /home/username/fetchmail.log

 poll the_isp_mail_server proto POP3
  user "remote_username" pass "PASSWORD=" is "local_username"
  ssl
  fetchall
  no keep
  no rewrite
  mda "/usr/bin/maildrop -f %F -d %T";
```

now, just add this:

```
#### .fetchmailrc

 set daemon 600
 set logfile /home/username/fetchmail.log

 poll the_isp_mail_server proto POP3
  user "remote_username" pass "PASSWORD=" is "local_username"
  ssl
  fetchall
  no keep
  no rewrite
  mda "/some/where/gpg_body -r MYKEY | /usr/bin/maildrop -f %F -d %T";
```

and you are done.

## System footprint

The program stores everything in memory. If your postfix's mail maximum size is 10MB, it will store the whole 10MB in memory in the worst case.

It will store just once, so up to 10MB of more memory per each invocation (no temporary files are used). Of course this can impact your server, depending by the traffic, and the amount of memory your server has.

gpg_body opens a pipe in order to communicate with gpg. So a new pipe is opened for each message, and then closed. This can harm your server if you are out of *pipe file descriptors*.

It takes the CPU to encrypt the message, of course. On high traffic servers you could experience a high cpu usage.

## **Security:**

Well, I kept the program simple, that is 50% of security. I used only std:: C++ functions: no libraries to avoid importing security flaws. Any string the program receives is trimmed to 1024 chars, that means the command line you add to gpg_body cannot exceed 1024 chars.A space counts 1 char. I assume there is a message size limit in your MTA, so input itself should not impact.

Of course... this means nothing. If you find a security flaw , please write me at uriel.fanelli at gmail.com .

## **TROUBLESHOOTING:**

Sometimes gpg_body exits printing, messages in the standard error. They look like "Problem # n". Error numbers n are matching the posix exit() status.

Here the meaning of exit codes:

- "Problem # 1" : command string is too long (>1023)
- "Problem # 2" : chars like "| > < & " passed somehow to the command line.
- "Problem #3" : Message is PGP Encrypted already...
- "Problem #4" : pgp command not found or exited with errors.
- "Problem #5" : No command line.

## **BUGS:**

Well, I am not a pro developer, I am a system architect , so I kept the code really easy, and used std:: C++ features to avoid bugs: less code, less bugs, isn't it? Of course, if you find a bug, please write me at uriel.fanelli at gmail.com

### **KNOWN ISSUES:**

Procmail + mbox format shows some problems with outlook.com's mail. Since maildrop is not having the same problems, I assume it is a procmail issue.Since I am not a procmail expert, feel free to explain me what is going wrong exactly.

## Improvements:

The program is GPL, so you are free to change it. If you improve it, please write me and I will update the source. If you want to join and help writing a better command, just tell me, you are welcome.

## FAQ:

- Q. gpg_body has anything to do with Enigmail? Has something to do with their developers?
- A. No. I just say it is a server side companion for Enigmail, and this is why I am mostly testing it with Enigmail+Thunderbird.

- Q. Why this code sucks?
- A. 'cause I am not a pro developer. I only wrote the program I wanted for my servers.

- Q. Why this code is amazing?
- A. 'cause I am not a pro developer. I only wrote the program I wanted for my servers.

- Q. I am a user and I lost my keys. Is there a way to recover my emails?
- A. No.
- Q. Again me. I lost my keys. What if i produce from scratch new keys for the same email address?
- A. It won't work. You lost your keys, then you lost your past emails.No way.

- Q. I am the only gay in the village, I created my linux distribution, and I have no /usr/bin . How I can change it to /huge/dildo ?
- A. Tons of Linux distros are gay enough for you. They all have /usr/bin. Use them.

- Q. Hello, we are the police department of Somewhere. We got a server to give a look in, but we can't read the email. Is there a way , like a backdoor, to do it?
- A. You could torture the owner until he tells you where are the keys (and the passphrase too). Then pretend stairs were slippery. As you do everyday, btw.

- Q. I am the government of Doomwhere and I have the right of reading other's email. I decided it and I agreed. Now your software is illegal and my Huge God doesn't likes you as well.
- A. Shit happens, isn't it?

- Q. If the attacker becomes root on your machine, it can sniff the incoming email. The same for the legitimate admins of the server.
- A. Yes. pgp_body is an improvement, not the solution for all problems. In any case, even root can't read your past email, the one already on the mailbox.

- Q. We have SSL on our servers. Why to encrypt the body?
- A. To avoid  anyone who illegally gets credentials (social engineering, and so) to read the email from the mailbox, bx example. It can protect stolen laptops also.

- Q. My laptop was stolen, and there I have my private keys as well. What about security?
- A. Still the passphrase is not known to the thief . Much better than *clear text* emails.

- Q. I need to change my key on the keyserver. What happens to my past emails?
- A. Be careful and keep a copy of your **old** keys available. If not, you risk all the past emails to be unreadable. Ask for advices to your network admin.

- Q. Why I have to download the code and compile it by myself? You should produce binaries as well.
- A. If you trust foreign binaries running on your mailserver, then processing all your email, probably your server has a security hole: **you.**

## Where to download the code:

http://code.google.com/p/gpg-body/downloads/list