gpgmailencrypt manual

This documentation describes gpgmailencrypt 3.3.x

Table of content

1 What is gpgmailencrypt

gpgmailencrypt is an e-mail gateway that can encrypt and decrypt e-mails.

It supports

- PGP/Inline
- PGP/Mime
- SMime
- encrypted PDF

gpgmailencrypt also can do virus and spam checks and can be a total replacement for amavis. Additionally it can sign your e-mails with a DKIM key.

Supported spamfilters are

- spamassassin
- bogofilter

Supported virus scanners are

- Avast
- Bitdefender
- Clamav
- Drweb
- F-prot
- Sophos

It can be used normally as a script doing everything on command line or in daemon mode, where gpgmailencrypt acts as an encrypting smtp server.

It takes e-mails and if a encryption key exists for this user it will return the e-mail encrypted to another e-mail server.

The encryption method can be selected per user.

2 Installation

2.1 Prerequisites

The following software needs to installed

- python3.x
- gnupg (I recommend version 2)
- openssl
- 7zip (for zipped attachments only)

and the python3 modules

- pypdf2
- beautifulsoup4
- python-magic
- lxml
- requests
- passlib
- bcrypt

For encrypted-PDF you need to install:

- wkhtml2pdf (http://wkhtmltopdf.org)
- pdftk
- 7zip

2.2 General

There are 2 different ways to install gpgmailencrypt. Choose the one that suits you best. In both cases you need to have installed python3 setuptools (on Ubuntu it's 'sudo apt-get install python3-setuptools')

2.2.1 Using pip

Install pip first (on Ubuntu its 'sudo apt-get install python3-pip')

The most easy way to install gpgmailencrypt is via pip:

```
sudo -H pip3 install gpgmailencrypt
```

On Windows you need

```
pip3.exe install --egg gpgmailencrypt
```

2.2.2 Installing the archive

After you've downloaded the package from
https://github.com/gpgmailencrypt/gpgmailencrypt/releases

```
unzip the zip-package with unzip gpgmailencrypt-xxx.zip
or untar it
tar -xzf gpgmailencrypt-xxx.tar.gz
```

Then change to the directory and start installation with

```
sudo python3 ./setup.py install
```

2.3 Daemon

2.3.1 sysvinit based systems

1. Make the gpgmailencrypt init file writeable

   ```
   sudo chmod 755 /etc/init.d/gpgmailencrypt
   ```

2. Create a user (gpg-mailencrypt) under which the daemon should run

   ```
   sudo adduser gpg-mailencrypt
   ```

3. You can set the user in the file /etc/default/gpgmailencrypt. It should contain

   ```
   USER="gpg-mailencrypt"
   DIR="/usr/local/bin"
   ```

2.3.2 Systemd based systems

1. Copy the gpgmailencrypt.service file into /etc/systemd/system

   ```
   [Unit]
   Description=gpgmailencrypt
   After=network.target mysql.service postgresql.service
   Wants=mysql.service postgresql.service

   [Service]
   Type=simple

   User=gpg-mailencrypt
   ExecStart=/usr/local/bin/gme.py --daemon
   StandardOutput=inherit
   TimeoutStartSec=5
   TimeoutStopSec=5

   [Install]
   WantedBy=multi-user.target
   ```

2. Create a user (gpg-mailencrypt) under which the daemon should run (if you use a different user you need to edit the gpgmailencrypt.service file)

```
sudo adduser gpg-mailencrypt
```

3. Start the server

```
sudo systemctl enable gpgmailencrypt.service
sudo systemctl start gpgmailencrypt.service
```

3 Configuration /etc/gpgmailencrypt.conf

To create a default configuration file create the conf file with:

```
gme.py -x >~/gpgmailencrypt.conf
sudo cp ~/gpgmailencrypt.conf /etc
```

```
sudo chown gpg-mailencrypt.root /etc/gpgmailencrypt.conf
sudo chmod 640 /etc/gpgmailencrypt.conf
```

3.1 General

```
[default]
preferred_encryption = pgpinline          #valid values are
                                          #'pgpinline','pgpmime' or 'smime'
add_header = no                           #adds a X-GPGMailencrypt header to
                                          #the mail
domains =                                 #comma separated list of domain
                                          #names, that should be encrypted,
                                          #empty is all
homedomains=localhost                     #a comma separated list of domains,
                                          #for which this server is working
                                          #and users might receive system mails
                                          #and can use pdf encrypt
securitylevel=may                         #valid values are 'may','redirect'
                                          #'script' or 'bounce', see chapter 7.1
bouncehomedomain=true                     #when true and security level is
                                          #'bounce' unencrypted emails to an
                                          #address in 'homedomains' will bounce.
                                          #so e-mails from 'homedomains' to
                                          #'homedomains' and from 'homedomains'
                                          #to external domains will bounce.
                                          #bouncedomain=False only mails from
                                          #'homedomains' to external domains
                                          #will bounce
bouncescript=/path/to/my/bounce.sh        #the script to be executed when
                                          #securitylevel is 'script'
spamsubject =***SPAM                      #Spam recognition string, spam will
                                          #not be encrypted
output=mail                               #valid values are 'mail'or 'stdout'
locale=en                                 #DA|DE|EN|ES|FI|FR|IT|NL|NO|PL|PT|
                                          #RU|SE
mailtemplatedir=/usr/share/gpgmailencrypt/mailtemplates
```

```
                                        #directory where mail
                                        #templates are stored
systemmailfrom=gpgmailencrypt@localhost # e-mail address used when sending
                                        #system mails
alwaysencrypt=False                     #if True e-mails will be sent
                                        #encrypted, even if there is no key.
                                        #Fallback encryption is encrypted pdf
use_sentaddress=False                   #If true a copy of the mail will be
                                        #sent to the sender of the mail.
                                        #the from address will be changed to
                                        #'sent_address'. This can be used
                                        #to store encrypted e-mails in the
                                        #sent folder of the user
                                        #using the sent_address to filter with
                                        #sieve or the e-mail client
                                        # see chapter 7.2
sent_address=SENT                       #the used address looks like
                                        #'sent_address <original@fromaddress>'
decrypt=False                           #if True it will be tried to decrypt
                                        #already encrypted e-mails sent to
                                        #recipients in 'homedomains'

[mailserver]
host = 127.0.0.1                        #smtp host
port = 25                               #smtp port
authenticate = False                    #user must authenticate
usesmtps=False                          #if True, the connection is ssl
                                        #encrypted from the beginning
                                        #don't confuse it with STARTTLS, which
                                        #will be used automatically
smtpcredential =/etc/gpgmailencrypt.cfg #file that keeps user and password
                                        #information
                                        #file format 'user=password'
cacerts=/etc/ssl/ca-certificates.crt    #the ca certificate storage file used
                                        #for verifiying smtp connections
#fingerprints=12345                     #a comma separated list of certificate
                                        #fingerprints used for certificate
                                        #pinning, if list is empty,
                                        #certificate pinning is switched off
host2 = 127.0.0.1                       #used for securitylevel=redirect
port2 = 25                              #used for securitylevel=redirect

[encryptionmap]
user@domain.com = PGPMIME               #PGPMIME|PGPINLINE|SMIME|
                                        #PDF[:zipencryptionmethod]|NONE

[usermap]
#user_nokey@domain.com = user_key@otherdomain.com
```

3.2 PGP specific configuration

```
[gpg]
keyhome = /var/lib/gpgmailencrypt/.gnupg        #home directory of public
                                                #gpgkeyring
gpgcommand = /usr/bin/gpg2
allowgpgcomment = yes                           #allow a comment string in the
                                                #GPG file
extractkey= no                                  #automatically scan emails and
                                                #extract pgp public keys to
                                                #'keyextractdir'
keyextractdir= ~/.gnupg/extract
encryptionkeys=user1,user2                      #comma separated list of
                                                #additional gpg keys, that
                                                #should be used to encrypt each
                                                #email
```

3.3 SMIME specific configuration

```
[smime]
keyhome = ~/.smime                              #home directory of S/MIME public key
                                                #files
opensslcommand = /usr/bin/openssl
defaultcipher = DES3                            #DES3|AES128|AES192|AES256
extractkey= no                                  #automatically scan e-mails and
                                                #extract smime public keys to
                                                #'keyextractdir'
keyextractdir=~/.smime/extract
encryptionkeys=user1.pem,user2.pem              #comma separated list of additional
                                                #smime keys, that should be used to
                                                #encrypt each email

[smimeuser]
smime.user@domain.com = user.pem[,cipher] #public S/MIME key file [,used cipher,
                                                #see defaultcipher]
```

3.4 PDF specific configuration

The e-mail text will be stored in an encrypted pdf, attachments in an encrypted zip-file with the same password.

```
[pdf]
pdfdomains=localhost                            #a comma separated list of sender
                                                #domains, which are allowed to use
                                                #pdf-encrypt
passwordlength=10                               #Length of the automatic created
                                                #password
passwordlifetime=172800                         #lifetime for autocreated passwords in
                                                #seconds. Default is 48 hours
passwordmode=sender                             #valid values are 'sender','none' or
                                                #'script'
```

```
passwordscript=/path/to/script.sh          #script that will be executed to
                                           #transport the password to the users
                                           #when passwordmode is 'script'
```

3.5 Zip-specific configuration

The zip functionality is mostly used with the pdf feature. Nevertheless you can use it independendly to zip all attachments (zipatttachments=True).

```
[zip]
7zipcommand=/usr/bin7za                    #path where to find 7za
use7zarchive=False                         #if True the 7z archive type will be
                                           #used to zip files, if False the zip
                                           #format will be used
defaultcipher=ZipCrypto                    #ZipCrypto|AES128|AES256
                                           #used only when use7zarchive=False
compressionlevel=5                         #1,3,5,7,9  with 1:lowest compression,
                                           #but very fast, 9 is highest
                                           #compression, but very slow,
                                           #default is 5
                                           #used only when use7zarchive=False
securezipcontainer=False                   #attachments will be stored in an
                                           #encrypted zip file. If this option is
                                           #true, the directory will be also
                                           #encrypted
                                           #used only when use7zarchive=False
zipattachments=False                       #if True all attachments will be
                                           #zipped, independent from the
                                           #encryption method
```

Which defaultcipher to use?

That depends what you can expect from your Windows users. ZipCrypto is the most unsecure one, but the only one the standard Windows unzipper can use. The most secure is AES256, but to open it the user needs either Winzip or 7zip installed.

3.6 Daemon specific configuration

```
[daemon]
host = 127.0.0.1                           #smtp host
port = 10025                               #smtp port
smtps = False                              #use smtps encryption
starttls = False                           #use starttls encryption
forcetls = False                           #communication (e.g. authentication)
                                           #will be only possible after STARTTLS
sslkeyfile = /etc/gpgsmtp.key              #the x509 certificate key file
sslcertfile = /etc/gpgsmtp.crt             #the x509 certificate cert file
```

```
authenticate = False                        #users must authenticate
smtppasswords = '/etc/gpgmailencrypt.pw'  #use smtps encryption
```

The gpgmailencrypt.pw has the following format (for the authenticate option see chapter 6.3.):

```
user1=password1
user2=password2
```

Don't forget to make the file readable only for the gpgmailencrypt user!

How the x509 certificate files can be created see:

```
https://www.e-rave.nl/create-a-self-signed-ssl-key-for-postfix
```

The ssl certificates will be needed for “starttls” and “smtps” mode.

What's the difference between starttls and smtps?

Both create and use a ssl encrypted communication channel. Smtps is the old version, where both sides use ssl from the start. Starttls connections first start unencrypted, then both sides switch to ssl after sending the “startTLS” command.

In doubt use the starttls variant.

If you set starttls and smtps at the same time in your configuration, then SMTPS will be used.

3.7 Virusscanner configuration

```
[virus]
checkviruses=False              #if true,e-mails will be checked for
                                #viruses before being encrypted
quarantinelifetime=2419200      #how long an infected e-mail exists in the
                                #quarantine (in seconds)
                                #(default is 4 weeks). 0 deactivates automatic
                                #deletion
```

Existing virusscanners will be automatically detected.

To scan archives for viruses you should install some archive software, at least **7zip** and **tnef** (this is for MS Windows e-mails).

Other supported archive unpackers are: ace, ar, arj, bzip2, cab, cpio, dar, freeze, gzip, kgb, lha, lzip, lrzip, lzo, rar, ripole, rpm, rzip, shar, snappy, tar, xz, zoo, zpaq.

3.8 Spamscanner configuration

```
[spam]
```

```
checkspam=False                 #if true, e-mails will be checked if they are
                                #spam
spamscanner=spamassassin        #valid values are spamassassin|bogofilter
sa_host=localhost               #server where spamassassin is running
sa_port=783                     #port of the spamassassin server
sa_spamlevel=6.2                #spamassassin threshold for spam, values higher
                                #than that means the mail is spam
sa_spamsuspectlevel=3.0         #spamassassin threshold for spam, values higher
                                #than that means the mail might be spam
                                #(value must be smaller than 'spamlevel')
maxsize=500000                  #maximum size of e-mail,that will be checked if
                                #it is spam
add_spamheader=False            #if True the e-mail gets spam headers
change_subject=False            #if True, the subject of the mail will get a
                                #prefix
spam_subject=***SPAM***         #subject prefix for spam
spamsuspect_subject=***SPAMSUSPICION*** #subject prefix for suspected spam
```

3.9 DKIM configuration

```
[dkim]
use_dkim=False                  #if true, the email will be signed,when the
                                #senders address is in homedomains
dkimdomain=localhost            #the dkim domain name
dkimselector=gpgdkim            #the dkim selector
dkimkey=~/dkim.key              #the private key to be used to sign the mail
```

To create the dkim key simply use

```
opendkim-genkey --domain=YOURDOMAIN  --selector=DKIMSELECTOR
```

You will have to add a TXT entry to your DNS server, so that others can verify your dkim entry. Looking like: DKIMSELECTOR._domainkey.YOURDOMAIN TXT v=DKIM1; k=rsa;p=M...XYZ

3.10 Using database backends

Administring user mapping in the configuration file works well with few users. But when you have hundreds of users you can use a sql backend instead. Simply set 'storagebackend' to the database backend you use and give all credentials in the 'sql' section.

```
[default]
storagebackend=MYSQL            #valid values are TEXT|MSSQL|MYSQL|SQLITE3|
                                #POSTGRESQL

[sql]
database=gpgmailencrypt         #name of database
user=gpgmailencrypt             #database user
password=                       #database password
```

```
host=127.0.0.1                  #sql server
port=3306                       #sql server port
usermapsql=select gpguser from gpgusermap where user=?
                                #SQL command that returns one row with the
                                #alternatve e-mail address
encryptionmapsql=SELECT encrypt FROM encryptionmap WHERE user= ?
                                #SQL command that returns one row with the
                                #preferred encryption method
smimeusersql=SELECT publickey,cipher FROM smimeusers WHERE user= ?
                                #SQL command that returns one row with
                                #information about an SMIME user
smimepublickeysql=SELECT user,publickey,cipher FROM smimeusers
                                #SQL command that returns a list with information
                                #about all SMIME users and their public keys
smimeprivatekeysql=SELECT user,privatekey,cipher FROM smimeusers WHERE
privatekey is not NULL
                                #SQL command that returns a list with information
                                #about all SMIME users and their private keys
use_sqlusermap=True             #if True the usermap will be taken from the sql
                                #database else it will be taken from the config
                                #file, section [usermap]
use_sqlencryptionmap=True       #if True the encryptionmap will be taken from the
                                #sql database else it will be taken from the
                                #config file, section [encryptionmap]
use_sqlsmime=True               #if True the SMIME user definition will be taken
                                #from the sql database else it will be taken from
                                #the config file, section [smimeuser]
use_sqlpdfpasswords=False       #if True the PDF passwords will be stored and
                                #taken from the sql database
sqlpdf_passwordtable=pdfpasswords #table that contains the pdf passwords
sqlpdf_userfield=user           #fieldname that contains the user
sqlpdf_passwordfield=password #fieldname that contains the password
sqlpdf_starttimefield=starttime #fieldname that contains the password creation
                                #time (needed for automatic password deletion)
use_sqlgpgencryptionkeys=False #if True each mail will be encrypted not only
                                #for the receiver, but alsofor additional keys
                                #delivered by 'gpgencryptionkeysql'
gpgencryptionkeysql=SELECT encryptionkey FROM gpgencryptionkeys WHERE user= ?
                                #SQL command that returns a list of gpg keys with
                                #which emails will be additionally encrypted
use_sqlsmimeencryptionkeys=False #if True each mail will be encrypted not only
                                #for the receiver, but alsofor additional keys
                                #delivered by 'smimeencryptionkeysql'
smimeencryptionkeysql=SELECT encryptionkey FROM smimeencryptionkeys WHERE user=
?
                                #SQL command that returns a list of smime keys
                                #with which emails will be additionally encrypted
```

The following is for users of postfixadmin:

with

```
      ALTER TALBLE mailbox add column x_gpg varchar(255)
```

With gpgusermap as

```
CREATE VIEW `gpgusermap` AS select `mailbox`.`username` AS
`user`,`mailbox`.`x_gpg` AS `gpguser` from `mailbox` union select
`alias`.`address` AS `address`,`mailbox`.`x_gpg` AS `x_gpg` from (`alias` join
`mailbox` on((`alias`.`goto` = `mailbox`.`username`))) order by `username`
```

4 Key Management

The following commands have to be used as the user, that is running gpgmailencrypt. Remember that in daemon mode this user is 'gpg-mailencrypt'. So for daemon mode you first have to change the user

```
sudo su - gpg-mailencrypt
```

4.1 PGP

Add a PGP key to the public key ring

```
gpg --import publickey.gpg
```

4.2 SMIME

Smime keys are stored in the directory ~/.smime per default. You have to create it if it does not exist. Each key is stored in a single file in pem-format.

Usually you get the smime.key file in a different format. To convert it use

```
openssl pkcs7 -print_certs -inform DER -in smime.p7s -out smime.pem
```

Let's say you get the smime.p7s from agentj@mib.

Instead of 'smime.pem" you should use a unique name for the file and copy it in ~/.smime/

```
cp smime.pem ~/.smime/agentj@mib.pem
```

For this user you need also an entry in /etc/gpgmailencrypt.conf

```
[smimeuser]
agentj@mib = agentj@mib.pem
```

4.3 Using PDF encryption

Let's say you want to send an e-mail with confidential content to somebody, but you don't have the

gpg or smime key of the receiver. So PDF encryption is for you.

PDF encryption is -in contrary to gpg and smime encryption – a symmetric encryption method. Gpgmailencrypt is converting the e-mail text to a PDF file and puts all attachments in a ZIP file.

Both are encrypted with a password.

How to use PDF encryption?

Simply add **#encrypt** at the beginning of the subject of your e-mail (at least one whitespace after #encrypt).

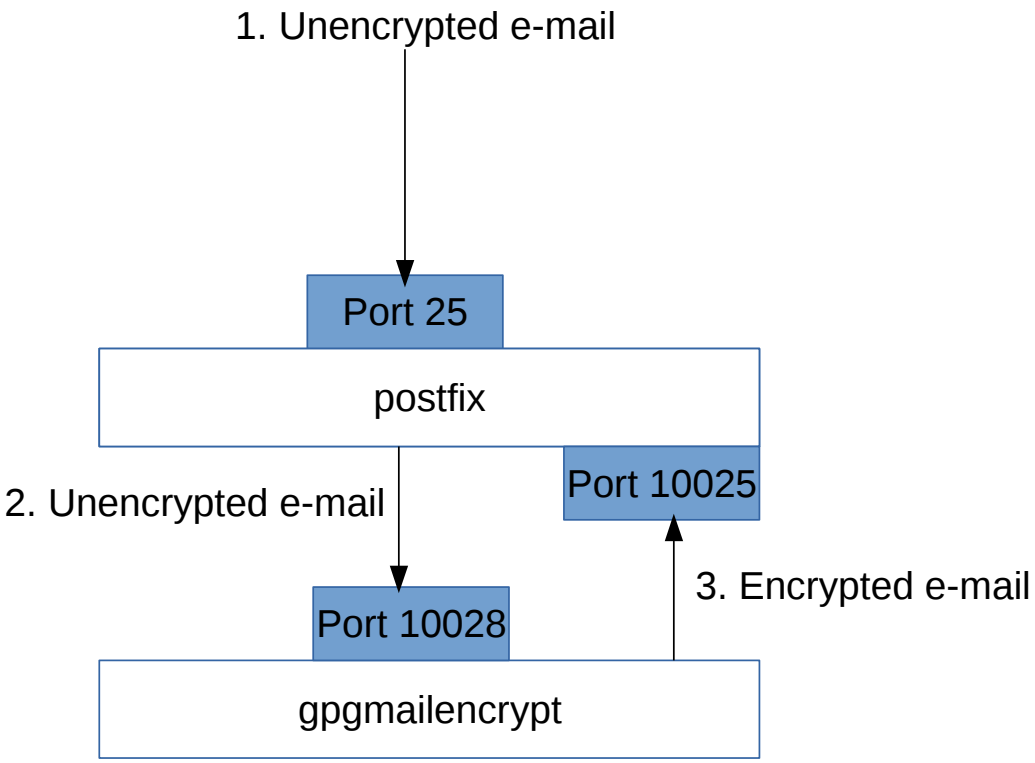
That's all.

When you send your e-mail you will receive an automatic e-mail with the password. Give the password to the e-mail recipient via a different communication channel (e.g. phone). Then he can read the e-mail and the attachments with standard tools like a pdf-reader and any zip-application.

If the recipient has already a gpg or s/mime key, then this will be used.

5 Integrating gpgmailencrypt in postfix

5.1 Direct way from postfix to gpgmailencryt



Install and configure gpgmailencrypt as daemon.

/etc/gpgmailencrypt.conf

```
[mailserver]
host = 127.0.0.1
port = 10025
[daemon]
host = 127.0.0.1
port = 10028
```

/etc/postfix/main.cf

```
content_filter=gpgmailencrypt:[127.0.0.1]:10028
```

/etc/postfix/master.cf

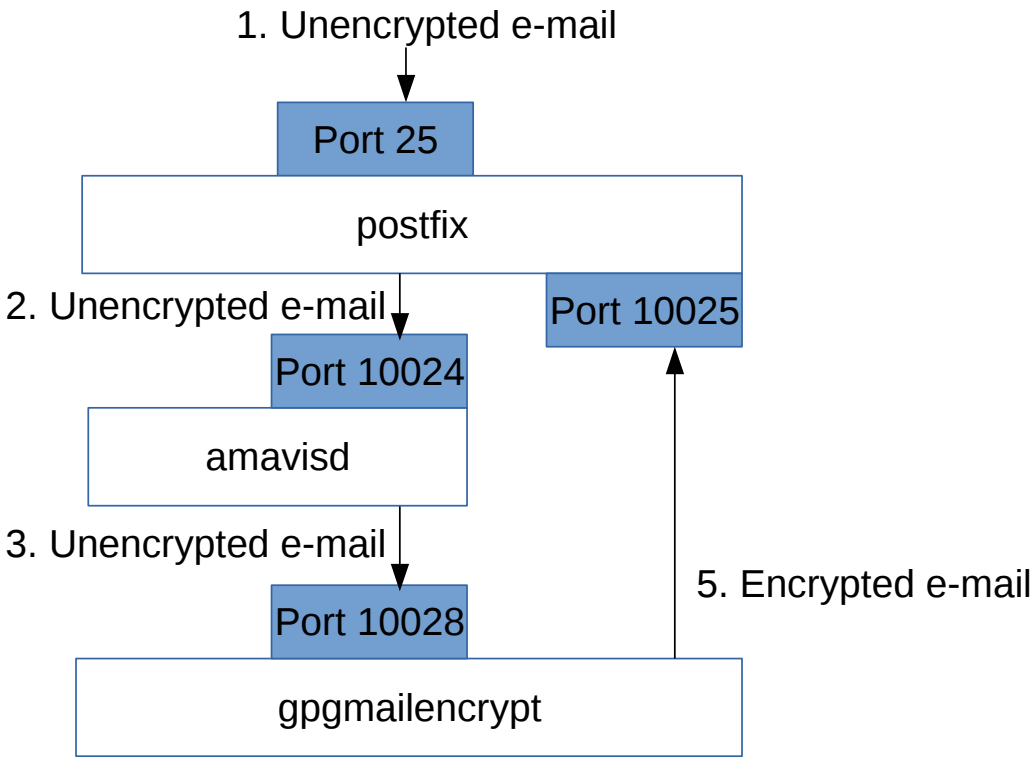
```
localhost:10025 inet n - n - -  smtpd
        -o content_filter=
        -o mynetworks=127.0.0.0/8
        -o receive_override_options=no_unknown_recipient_checks
        -o smtpd_recipient_restrictions=permit_mynetworks,reject_unauth_destination
        -o smtpd_authorized_xforward_hosts=127.0.0.0/8

gpgmailencrypt unix - - n - 2 smtp
        -o smtp_data_done_timeout=1800
```

5.2 From Postfix to amavisd to gpgmailencrypt

This is not necessary any more, as gpgmailencrypt can replace amavisd. If for any reason you still want to use amavis, here is the solution:

This is to check for viruses and spam before encrypting the mail



This has to be configured as before, except:

/etc/postfix/main.cf

```
content_filter=amavis:[127.0.0.1]:10024
```

/etc/postfix/master.cf

```
amavis  unix - - n - 2 smtp
        -o smtp_data_done_timeout=1800
```

/etc/amavis/conf.d/50-user

```
[...]
$notify_method  = 'smtp:[127.0.0.1]:10028';
$forward_method = 'smtp:[127.0.0.1]:10028';
```

Of course you have to configure amavisd to suite your needs.

Amavis and SSL

Amavis does not support SSL encryption of any kind or authentication. So you can't use these features of gpgmailencrypt in combination with amavis.

5.3 Using authentication

For using the authentication add the following to gpgmailencrypt section in /etc/postfix/master.cf

```
        -o smtp_sasl_auth_enable=yes
        -o smtp_sasl_password_maps=hash:/etc/postfix/gpgmailencrypt_passwd
```

With /etc/postfix/gpgmailencrypt_passwd having the following structure

```
localhost user:password
```

Then use the following commands

```
sudo chmod 640 /etc/postfix/gpgmailencrypt_passwd
sudo postmap /etc/postfix/gpgmailencrypt_passwd
```

5.4 Using starttls

Add the following to gpgmailencrypt section in /etc/postfix/master.cf:

```
      -o smtp_use_tls = yes
      -o smtp_tls_security_level = encrypt
```

5.5 Using smtps

To use the gpgmailencrypt smtps feature with postfix 2.x you need to install stunnel (in Ubuntu the package is called stunnel4)

Create the file /etc/stunnel/gpgmailencrypt.conf

```
[gpgmailencrypt-smtps]
    accept = 10000
    client = yes
    connect = localhost:10028
```

And change /etc/default/stunnel4

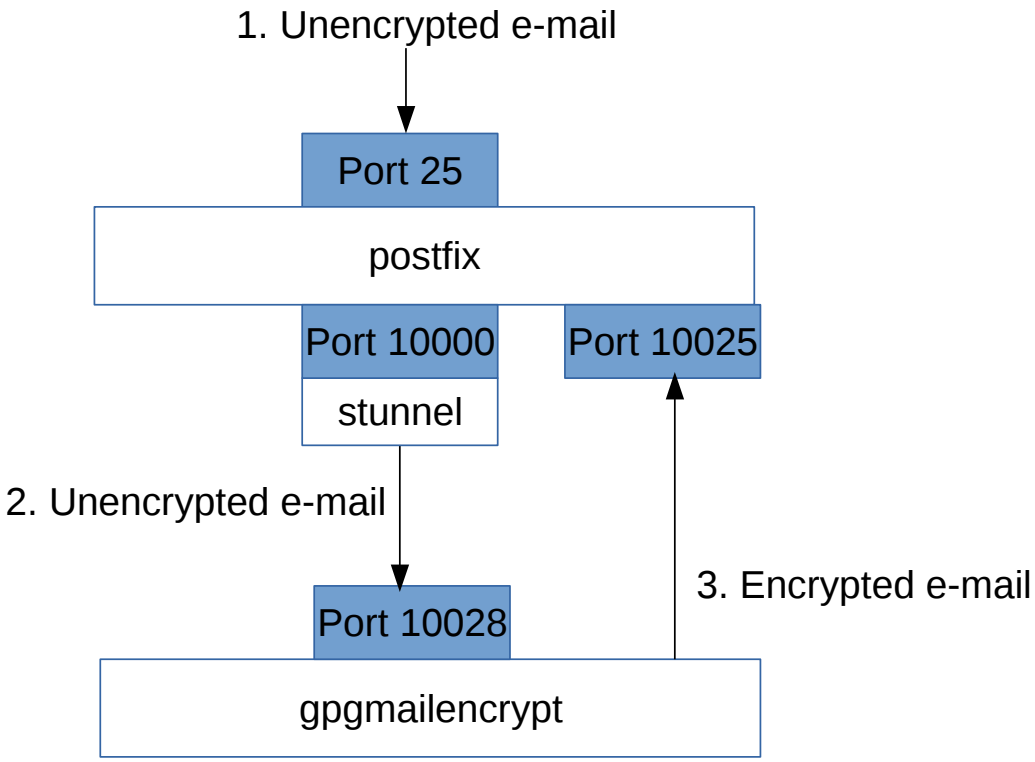
```
ENABLED=1
```

Then start stunnel with

```
/etc/init.d/stunnel4 start
```

/etc/postfix/main.cf should be changed to

```
content_filter=gpgmailencrypt:[127.0.0.1]:10000
```



6 The admin console

6.1 The admin user – the hen and egg problem

First you have to create a normal user (which only the admin user can do), then you can give him administration rights.

So for the very first time you have to start the following python program as the user, that runs the gpgmailencrypt daemon. Replace “admin1” and “secreet” with the values you like

```
#!/usr/bin/python3
```

```
import gpgmailencrypt
user="admin1"
password="secret"
with gpgmailencrypt.gme() as g:
      g.adm_set_user(user,password)
```

Now the user is created.

Now change the /etc/gpgmailencrypt.conf to the following value

```
[daemon]
admins=admin1                          #comma separated list of admins, that can
                                       #use the admin console
```

Now restart the daemon, that's it.

6.2 Using the admin console

Start

```
gme_admin.py localhost 10025
```

Replace 'localhost' and the port number with the values of your daemon

Now login as an admin user. You will see the following screen

```
Try to connect to localhost:10025 ...
gpgmailencrypt admin console
============================
User: admin1
Password:
OK
Authentication successful.
Welcome. Enter 'HELP' for a list of commands
> _
```

The following commands are available:

flush	tries to re-send deferred e-mails
debug true/false	sets the debug mode valid options are TRUE|FALSE, ON|OFF, YES|NO
deluser	deletes a user example: 'deluser john'
help	Shows all available commands
messages	shows all systemwarnings and -errors
quit	leave the console

reload	*reloads the configuration file*
resetstatistics	*sets all statistic values to 0*
setuser	*adds a new user or changes the password for an existing user* *example: 'setuser john johnspassword'*
statistics	*print statistic information*
users	*print users*

6.3 Non-admin users

Users, that don't have the admin privilege can be used to login to the server, when the daemon need authentication, set with

/etc/gpgmailencrypt.conf

```
[daemon]
authenticate = True                    #users must authenticate
```

7 Tips and Tricks

7.1 How to handle e-mails that can’t be encrypted?

Usually mails that can’t be encrypted will be sent unencrypted. If you want to change this behaviour you can use the config following gpgmailencrypt.conf options

```
[default]
securitylevel=may        #valid values are 'may',’script’,'redirect' or 'bounce'
bouncehomedomain=true    #when true and security level is 'bounce' unencrypted
                         #emails to an address in 'homedomains' will bounce.
```

The different securitylevels are:

may	*delivers unencryptable e-mails any way*
script	*A script will be executed which ist defined in:* **[default]** **bouncescript=/path/to/my/bounce.sh** *The script gets the following 3 command line parameters:* *from_emailaddress to_emailaddress filename_with_mailcontent* *so the script will be called as follows:* **/path/to/my/bounce.sh from_emailaddress to_emailaddress filename_with_mailcontent**
redirect	*Sends the unencrypted e-mail to a different smtp server. Will be used in combination with the config settings:* **[mailserver]** **host2** *= 127.0.0.1*

	port2 = *25*
bounce	*Does not deliver the e-mail. The sender gets a bounce information e-mail*

7.2 How to encrypt e-mails in the “Sent”-folder

The idea is to let gpgmailencrypt encrypt a copy of your e-mail and then let the mailserver/sieve or your e-mail client move this e-mail to your “SENT” folder. Gpgmailencrypt does this only for domains which are in the “homedomains” variable in the “default” section.

The gpgmailencrypt.conf configuration

```
[default]
use_sentaddress=True
sent_address=SENT
```

How does it work?

Gpgmailencrypt creates a copy of your e-mail and changes the “From” field to “SENT <your@email.address>” and delivers it back to the mail server. This e-mail will be delivered to you. Then you simply have to create a filter to store this e-mail in your Sent folder.

The rule for sieve:

```
require ["fileinto","regex","imap4flags"];

if header :regex ["From"] "SENT <.*@mydoma.in>" {
    setflag "\\seen";
      fileinto "Sent";
      }
```

Tip:

Don't forget to configure all your e-mail clients **not** to store a copy in the Sent folder!

In Thunderbird deactivate “**Account setting->Copies&Folders->Place a copy in**”

7.3 How to use your own design for system mails

The templates for system e-mails are ordinary html files and stored in “/usr/share/gpgmailencrypt/mailtemplates” (or whereever your mailtemplatedir variable in the config file refers to).

For every language there is a subdirectory. If there is no translation for a file, it will be taken from the “EN” subdirectory. So now let's have a look in this directory.

The general design is defined in the file “00-template.html”

```
<html>
<head>
<meta charset="utf-8"/>
<style>
#maintable {
    border-top-style: solid; border-width: 5px;
}
</style>
</head>
<body>
%EMAILTEXT%
<br><br>
<table width=100% ; id="maintable">
<td style="font-size:0.8em">
Powered by <span style="font-size:1.0em ;background-color: #b0c4de;font-
style:italic">gpgmailencrypt %VERSIONDATE%</span>
</td>
<td style="text-align: right;font-size:0.8em ;">%COPYRIGHT%
</td>
</tr>
</table>
</body>
</html>
```

Variables are delimited with '%' and are in captial letters (e.g. %EMAILTEXT%).

The variable %EMAILTEXT% in particular includes the content of the other -topic specific- templates.

How to localize gpgmailencrypt?

Let's say you want to have gpgmailencrypt localized for greek. The country code for this language is “GR”

1.: Simply copy the “EN” directory in your new “GR”directory

2.: Translate the files in this directory

3.: Search for the gpgmailencrypt.py file and in this file for the variable “_LOCALEDB”. Create a new entry in this variable and translate the values

```
"GR":("appointment","file","content","attachment"),
```

4.: Set the variable “locale”in the configuration file to “GR”

5.: Restart gpgmailencrypt

7.4 How to decrypt e-mails?

Warning: Storing the private keys on a server is a tremendous security risk, so if in doubt, don’t do it.

The config entry is:

```
[default]
decrypt=True                                #if True it will be tried to decrypt
                                            #already encrypted e-mails sent to
                                            #recipients in 'homedomains'
```

Gpgmailencrypt tries to decrypt any kind of pgp/smime/pdf/zip based encryption.

For symmetric based encryption formats (pdf and zip) gpgmailencrypt uses all available passwords of the sender and the receiver of the e-mail to decrypt.

For assymetric encryption formats (pgp and smime) you need to store the private keys without password protection in the keyring. Again: **this is a security risk!**

Gpg2 and sudo
Gpg2 can behave quite nasty in sudo environments, when importing/exporting private keys. The reason is, that the gpg-agent tries to ask for a password (with the help of pinentry). And this pinentry does not change the user when using sudo, so it has the wrong permissions and this will lead to a “permission denied” error.

```
The only way I found to circumvent this, is the following  trick:
start the command with 'script'
e.g. script "gpg2 --edit-key mykey@gpgmailencry.pt"
and delete the "typescript" file in the home directory afterwards
```

8 Using gpgmailencrypt from the command line

Start

```
gme.py
```

The command line options are

```
Usage:

gme.py [options] receiver@email.address < Inputfile_from_stdin

Options:

-a --addheader:   adds X-GPGMailencrypt header to the mail
-c f --config f:  use configfile 'f'. Default is /etc/gpgmailencrypt.conf
-d --daemon :     start gpgmailencrypt as smtpserver
-e pgpinline :    preferred encryption method, either 'pgpinline','pgpmime' or
                  'smime'
-f mail :         reads email file 'mail', otherwise from stdin
-h --help :       print this help
-k f --keyhome f: sets gpg key directory to 'f'
-l t --log t:     print information into _logfile, with valid types 't'
                  'none','stderr','syslog','file'
-n domainnames:   sets the used domain names (comma separated lists, no space),
                  which should be encrypted, empty is all
-m mailfile :     write email file to 'mailfile', otherwise email will be sent
                  via smtp
-o p --output p:  valid values for p are 'mail' or 'stdout', alternatively you
                  can set an outputfile with -m
--spamcheck=true: if true, check if the e-mail is spam
-x --example:     print example config file
-v --verbose:     print debugging information into _logfile
--viruscheck=true:if true, check if the e-mail contains a virus
-z --zip:         zip attachments
```

9 Using gpgmailencrypt as a module

```
import gpgmailencrypt
help(gpgmailencrypt)
```

The most important function is

```
gpgmailencrypt.encrypt_mails(self,mailtext,receiver)
```

An example for encryption:

```
import gpgmailencrypt

#reads the e-mail from a file
f=open("myemail.eml)
mail=f.read()
f.close()

#sends an encrypted e-mail
with gpgmailencrypt.gme() as g:
      g.encrypt_mails(mailtext=mail,recipient="agentj@mib")
      #the following sends the e-mail to 2 recipients
      g.encrypt_mails(mailtext=mail,recipient=["agentj@mib","agentk@mib"])
```

Decrypting e-mails is also simple:

```
import gpgmailencrypt

#reads the encrypted e-mail from a file
f=open("myemail.eml)
mail=f.read()
f.close()

#decrypts an encrypted e-mail
with gpgmailencrypt.gme() as g:

g.decrypt_mail(mailtext=mail,from_addr="external@somewhere",to_addr="agentj@mib
")
```