

Python gnupg (GPG) example

Date: 2011 (/blog/2011/)-10-28 | Modified: 2012-04-08 | Tags: python (/blog/tag/python/) | 3 Legacy Comments (/blog/2011/10/python-gnupg-gpg-example/#comments) | 3 Comments (/blog/2011/10/python-gnupg-gpg-example/#disqus_thread)

`python-gnupg` (<http://code.google.com/p/python-gnupg/>) is a Python package for encrypting and decrypting strings or files using GNU Privacy Guard (GnuPG or GPG) (http://en.wikipedia.org/wiki/GNU_Privacy_Guard). GPG is an open source alternative to Pretty Good Privacy (PGP) (http://en.wikipedia.org/wiki/Pretty_Good_Privacy). A popular use of GPG and PGP is encrypting email. For more information, see the `python-gnupg` documentation (<http://packages.python.org/python-gnupg/>). Another option for encrypting data from Python is `keyczar` (<http://code.google.com/p/keyczar/>).

Install

This installs the Ubuntu GPG package, creates a test user, and installs the Python package, `python-gnupg`. This was installed on Ubuntu 10.10 Maverick Meerkat.

```
$ sudo apt-get install gnupg
$ sudo adduser testgpguser
$ sudo su testgpguser
$ cd
$ virtualenv --no-site-packages venv
$ source venv/bin/activate
$ pip install python-gnupg
```

Generate a key

This creates a GPG key. This also creates the `gpghome` directory if it does not exist. You may need to supply random hardware activity during the key generation. See the docs (<http://packages.python.org/python-gnupg/#performance-issues>) for more information. To generate random numbers, you can also install the `rng-tools` package.

```
$ sudo apt-get install rng-tools
```

```
import os
import gnupg

os.system('rm -rf /home/testgpguser/gpghome')
gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
input_data = gpg.gen_key_input(
    name_email='testgpguser@mydomain.com',
    passphrase='my passphrase')
key = gpg.gen_key(input_data)
print key
```

```
B0F4CF530036CE8CD1C064F17D32CEE72C015CD5
```

Export keys

```
import gnupg

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
ascii_armored_public_keys = gpg.export_keys(key)
ascii_armored_private_keys = gpg.export_keys(key, True)
with open('mykeyfile.asc', 'w') as f:
    f.write(ascii_armored_public_keys)
    f.write(ascii_armored_private_keys)
```

```
(venv)testgpguser@mymachine:~$ cat mykeyfile.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

mI0ETqrVGAEEAP42Xs1vQv40MxA3/g/Le5B0VatnDYaSvAhiYfaub79HY4mjYcCD
FPDo5b54PSzyhlVsz5RL46+RE9NpQ2JdvFofWi7eVzfdmmTtNYEaiUSmzLUq73Vz
qu7P1RhOfwuAyW0otnw/Lw54MVjVZblvp3ln1Fcpleb9ZSrY1h61Y8pHABEBAAAG0
REFldG9nZW51cmF0ZWQgS2V5IChHZW51cmF0ZWQgYnkgZ251cGcucHkpIDx0ZXN0
Z3BndXNlckBteWRvbWVpb5jb20+iLgEEwECACIFAk6q1RgCGy8GCwkIBwMCBhUI
AgkKCWQWAgMBAh4BAheAAAJEH0yzucsAVzVBjwD/1KgTx1y3cpuumu1HF0GtQV0
Wn7l9OaSj98CqQ/f2emHD1l9rrjdt9jmlg7wSsWumpKs57vxz7NXwHw7mI4qZ5m0
cvqg/qRc/BBMP8v2WgzRsmIs97PpLaate1k3QfvDCVs6F1qiIQyELffjxBHbmWPhx
XEwhnpLcvk217NbNnEwA
=exDD
-----END PGP PUBLIC KEY BLOCK-----
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

lQH+BE6q1RgBBAD+Nl7Nb0L+NDMQN/4Py3uQdFWrZw2GkrwIYmH2rm+/R2OJo2HA
gxTw6OW+eD0s8oZVbM+US+OvkRPTaUNiXbxaHlou3lc33Zpk7TWBGolEpsylKu91
c6ruz9UYTn8LgMltKLZ8Py8OeDFY1WW5b6d5Z9RXKZXm/WUq2NYetWPKRwARAQAB
/gMDAQ5W6uxeU2hDYDPZ1Yy+e97ppNXmdAeq1urZHmiPr4+a36nOWd6j0R/HBjG3
ELD8CqYiQ0vx8+F9rY/uwKga2bEkJsQXjvaaZtu97lzPyp2+avsaW2G+3jRAJWNL
5YG4c/XwK1cfEajM23f7zz/t6TRWG+Ve2Dzi7+obA0LuF8czSlpiTTEzLDk8QJCK
y2WmrZ+s+POWv3itVpI26o7PvTQESzwyKXdyCW2W66VnXTm4mQEL6kgyV0oO6xIl
QUVSn2XWvwFMg2iL+02za467rsrlx6Nl8hEQJgFwJCeJD2z+4C4yzEeQGFP9WUps
pbMedAjDHebhC9FzbW7yuQ3H7iTCKlmvidAFw2wTdrkH61ApzmSo/rSTSxXw7hLT
M/ONGYztvr+CpJj+mIu1XvVDiftvMhXlwcvm8c9PB3zv+086K7kJDTnzPgYvL0H/
+V2b9X9BBfAax40MQuXZJWseaLtsxXyl/rhn8jSCFZoqtERBdXRvZ2VuZXJhdGVk
IEtleSAoR2VuZXJhdGVkIGJ5IGdudXBnLnB5KSA8dGVzdGdwZ3VzZXJAbXlk21h
aW4uY29tPoi4BBMBAGAiBQJOqtUYAhsvBgJCAcDAgYVCAIJCgsEFgIDAQIeAQIX
gAAKCRB9Ms7nLAFclQY8A/9SoE8dct3KbrprtRxdBrUFdFp+5fTmko/fAqkP39np
hw9Zfa643bfY5tY08ErFrpqSrOe78c+zV8B8O5iOKmeZtHL4P6kXPwQTD/L9l0M0
bJpbPez6ZWmrXtZN0H7wwlbOhdaoieMhC3348QR251j4cVxMIz6S3L5NpezWzXm
AA==
=v9Z7
-----END PGP PRIVATE KEY BLOCK-----
```

Import keys

```
import gnupg
from pprint import pprint

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
key_data = open('mykeyfile.asc').read()
import_result = gpg.import_keys(key_data)
pprint(import_result.results)
```

```
[{'fingerprint': u'B0F4CF530036CE8CD1C064F17D32CEE72C015CD5',
  'ok': u'0',
  'text': 'Not actually changed\n'},
 {'fingerprint': u'B0F4CF530036CE8CD1C064F17D32CEE72C015CD5',
  'ok': u'16',
  'text': 'Contains private key\nNot actually changed\n'}]
```

List keys

```
import gnupg
from pprint import pprint

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
public_keys = gpg.list_keys()
private_keys = gpg.list_keys(True)
print 'public keys:'
pprint(public_keys)
print 'private keys:'
pprint(private_keys)
```

```
public keys:
[{'algo': u'1',
  'date': u'1319818520',
  'dummy': u'',
  'expires': u'',
  'fingerprint': u'B0F4CF530036CE8CD1C064F17D32CEE72C015CD5',
  'keyid': u'7D32CEE72C015CD5',
  'length': u'1024',
  'ownertrust': u'u',
  'trust': u'u',
  'type': u'pub',
  'uids': [u'Autogenerated Key (Generated by gnupg.py) ']]}

private keys:
[{'algo': u'1',
  'date': u'1319818520',
  'dummy': u'',
  'expires': u'',
  'fingerprint': u'B0F4CF530036CE8CD1C064F17D32CEE72C015CD5',
  'keyid': u'7D32CEE72C015CD5',
  'length': u'1024',
  'ownertrust': u'',
  'trust': u'',
  'type': u'sec',
  'uids': [u'Autogenerated Key (Generated by gnupg.py) ']]}
```

Encrypt a string

```
import gnupg

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
unencrypted_string = 'Who are you? How did you get in my house?'
encrypted_data = gpg.encrypt(unencrypted_string, 'testgpguser@mydomain.com')
encrypted_string = str(encrypted_data)
print 'ok: ', encrypted_data.ok
print 'status: ', encrypted_data.status
print 'stderr: ', encrypted_data.stderr
print 'unencrypted_string: ', unencrypted_string
print 'encrypted_string: ', encrypted_string
```

```
ok: True
status: encryption ok
stderr: [GNUPG:] BEGIN_ENCRYPTION 2 9
[GNUPG:] END_ENCRYPTION

unencrypted_string: Who are you? How did you get in my house?
encrypted_string: -----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

hIwDFuhrAS77HYIBBACXqZ66rkGQv8yE61JddEmad3fUNvbfkhBPUI9OSaMO3PbN
Q/6SIDyi3Fmhbm9icOBS7q3xddQpvFhwmrq9e3VLKnV3NSmWo+xJWosQ/GNAA/Hb
cwF1pOtR6bRHFbKqtmptYnBo9rMpokW8lp4WxFxMda+af8TlId8HC0WcRUg4kNJi
AdV1fsd+sD/cGIp0cAltpaVuO4/uwV9lKd39VER6WigLDaeFUHjWhJbcHwTaJYHj
qmy5LRciNSjwsqeMK4zOFZyRPuqPVKwWLiE9kImMni0Nj/K54ElWujgTttZIlBqV
5+c=
=SM4r
-----END PGP MESSAGE-----
```

Decrypt a string

```
import gnupg

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
unencrypted_string = 'Who are you? How did you get in my house?'
encrypted_data = gpg.encrypt(unencrypted_string, 'testgpguser@mydomain.com')
encrypted_string = str(encrypted_data)
decrypted_data = gpg.decrypt(encrypted_string, passphrase='my passphrase')

print 'ok: ', decrypted_data.ok
print 'status: ', decrypted_data.status
print 'stderr: ', decrypted_data.stderr
print 'decrypted string: ', decrypted_data.data
```

```
ok: True
status: decryption ok
stderr: [GNUPG:] ENC_TO 16E86B012EFB1D82 1 0
[GNUPG:] USERID_HINT 16E86B012EFB1D82 Autogenerated Key (Generated by gnupg.
py)
[GNUPG:] NEED_PASSPHRASE 16E86B012EFB1D82 16E86B012EFB1D82 1 0
[GNUPG:] GOOD_PASSPHRASE
gpg: encrypted with 1024-bit RSA key, ID 2EFB1D82, created 2011-11-02
      "Autogenerated Key (Generated by gnupg.py) "
[GNUPG:] BEGIN_DECRYPTION
[GNUPG:] PLAINTEXT 62 1320545729
[GNUPG:] PLAINTEXT_LENGTH 41
[GNUPG:] DECRYPTION_OKAY
[GNUPG:] GOODMDC
[GNUPG:] END_DECRYPTION

decrypted string: Who are you? How did you get in my house?
```

Encrypt a file

```
import gnupg

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
open('my-unencrypted.txt', 'w').write('You need to Google Venn diagram.')
with open('my-unencrypted.txt', 'rb') as f:
    status = gpg.encrypt_file(
        f, recipients=['testgpguser@mydomain.com'],
        output='my-encrypted.txt.gpg')

print 'ok: ', status.ok
print 'status: ', status.status
print 'stderr: ', status.stderr
```

```
ok: True
status: encryption ok
stderr: [GNUPG:] BEGIN_ENCRYPTION 2 9
[GNUPG:] END_ENCRYPTION
```

```
(venv)testgpguser@mymachine:~$ cat my-encrypted.txt.gpg
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.10 (GNU/Linux)

hIwDfTLO5ywBXNUBBADo7trFZUD6Ir1vPRAJsoQXDiiw32N1m9/PXWCnQqX0nyzW
LfluNMfLFQRclNPVEg+o91qhS71apKvagp8DW7SCDE2SdCYk8nAS3bwAg5+GUyDs
XY2E6BQ1cLA1eK1v6D15ih6cq0laRzWuFkehH9PQ5Yp4ZZOmCbopw7dufnYPjdJb
AVGLpZRq64SuN1BUWIIHbO7vqQGFq7qhGQwuegbleMm4vyr6FBW6JA/x4G/PMfImZ
1cH6KBrWGWrlCTiU/FKG9JvOm8mg8NXzd/TVjPs6rHRaKPFln37T7cLUwA==
=FSQP
-----END PGP MESSAGE-----
```

Decrypt a file

```
import gnupg

gpg = gnupg.GPG(gnupghome='/home/testgpguser/gpghome')
with open('my-encrypted.txt.gpg', 'rb') as f:
    status = gpg.decrypt_file(f, passphrase='my passphrase', output='my-decr
rypted.txt')

print 'ok: ', status.ok
print 'status: ', status.status
print 'stderr: ', status.stderr
```

```
ok: True
status: decryption ok
stderr: [GNUPG:] ENC_TO 16E86B012EFB1D82 1 0
[GNUPG:] USERID_HINT 16E86B012EFB1D82 Autogenerated Key (Generated by gnupg.
py)
[GNUPG:] NEED_PASSPHRASE 16E86B012EFB1D82 16E86B012EFB1D82 1 0
[GNUPG:] GOOD_PASSPHRASE
gpg: encrypted with 1024-bit RSA key, ID 2EFB1D82, created 2011-11-02
      "Autogenerated Key (Generated by gnupg.py) "
[GNUPG:] BEGIN_DECRYPTION
[GNUPG:] PLAINTEXT 62 1320546031
[GNUPG:] PLAINTEXT_LENGTH 32
[GNUPG:] DECRYPTION_OKAY
[GNUPG:] GOODMDC
[GNUPG:] END_DECRYPTION
```

```
(venv)testgpguser@mymachine:~$ cat my-decrypted.txt
You need to Google Venn diagram.
```


Comments

#1 shackra (<http://swt.encyclomundi.org/>) commented on 2012-01-23:

and Can you encrypt data with your private key and send it to a friend? then Can he decrypt the data with your public key?

#2 GPG (<http://www.hotelsutruptibhubaneswar.com/>) commented on 2012-04-30:

Although the basic GnuPG program has a command line interface, there exist various front-ends that provide it with a graphical user interface. For example, GnuPG encryption support has been integrated into KMail and Evolution, the graphical e-mail clients found in KDE and GNOME, the most popular Linux desktops. There are also graphical GnuPG front-ends (Seahorse for GNOME, KGPG for KDE). For Mac OS X, the Mac GPG project provides a number of Aqua front-ends for OS integration of encryption and key management as well as GnuPG installations via Installer packages

#3 Nemesis Fixx commented on 2012-12-04:

Was struggling with PyCrypto's DES3 CBC stuff, this just makes my task cake!

Thanks for illustrating almost every use-case!

3 Comments**SaltyCrane Blog****1 Login** ▾

♥ Recommend

↗ Share

Sort by Oldest ▾



Join the discussion...

**sadry** • 9 months ago

If you run this code you may have a small error :

execv() arg 2 must contain only strings

Then you just have to convert the key in string with str(key)

^ | ▾ • Reply • Share ›

**Kevin Jones** ↗ sadry • 6 months ago

I had the same issue.

^ | ▾ • Reply • Share ›

**Genesis** • 3 months ago

Nice article.

Thanks.

<http://computersecuritypgp.blog...>

^ | ▾ • Reply • Share ›

ALSO ON SALTYCRANE BLOG

WHAT'S THIS?

Simplistic Python Thread example

2 comments • 9 months ago

http://at.at.ua — Wow! Cool!**Saving a Python dict to a file using pickle**

1 comment • 9 months ago

Nathaniel — can you type a value e.g 1,2,3, and display it under read using pickle? displaying the value in numbers and not in binary?**PyQt: How to pass arguments while emitting a signal**

1 comment • 9 months ago

Szabolcs Illés — Should it work with PySide too ? Because when the last line (self.do_something()) is commented it brings up**Configuring the VxWorks Full Simulator on Windows**

1 comment • 9 months ago

Trevor — Here I am. In 2015... reading old posts for old software from 2007. :(



(/blog/)

ABOUT (/ABOUT/)

I'm Eliot and this is my notepad for programming topics such as Javascript, Python, Emacs, etc... **more**
(/about/)

LINKS



Twitter

(http://twitter.com/saltycrane/)



Github

(http://github.com/saltycrane/)



Stack Overflow

(http://stackoverflow.com/users/101911/saltycrane/)



Atom Feed

(/feeds/latest/)

TAGS

algorithms

(/blog/tag/algorithms/) (6)

android

(/blog/tag/android/) (2)

aws (/blog/tag/aws/) (10)

c_plusplus

(/blog/tag/c_plusplus/)
(12)

blogproject

(/blog/tag/blogproject/) (20)

cardstore

(/blog/tag/cardstore/) (8)

concurrency

(/blog/tag/concurrency/)
(13)colinux (/blog/tag/colinux/)
(2)

conkeror

(/blog/tag/conkeror/) (2)

datastructures

(/blog/tag/datastructures/)
(17)

core (/blog/tag/core/) (2)

cygwin (/blog/tag/cygwin/)
(17)

datetime

(/blog/tag/datetime/) (4)

decorators

(/blog/tag/decorators/) (4)

django

(/blog/tag/django/) (41)

files_directories

(/blog/tag/files_directories/)
(12)emacs (/blog/tag/emacs/)
(22)

frontend

(/blog/tag/frontend/) (3)

install_setup

(/blog/tag/install_setup/)

git (/blog/tag/git/) (7)

hardware

(/blog/tag/hardware/) (8)

(8)

javascript

(/blog/tag/javascript/) (7)

keyboard

(/blog/tag/keyboard/) (9)

matplotlib

(/blog/tag/matplotlib/) (6)

mercurial

(/blog/tag/mercurial/) (4)

nginx (/blog/tag/nginx/) (2)

persistence

(/blog/tag/persistence/) (6)

[preferences](/blog/tag/preferences/) (7) [processes](/blog/tag/processes/) (4) [pyqt](/blog/tag/pyqt/) (18)
[python](/blog/tag/python/) (166)
[ratpoison](/blog/tag/ratpoison/) (3) [regexes](/blog/tag/regexes/) (6)
[softwaretools](/blog/tag/softwaretools/) (17)
[rsync](/blog/tag/rsync/) (3) [sql](/blog/tag/sql/) (14)
[ssh](/blog/tag/ssh/) (12) [subversion](/blog/tag/subversion/) (6) [twisted](/blog/tag/twisted/) (7)
[ubuntu](/blog/tag/ubuntu/) (68) [urxvt](/blog/tag/urxvt/) (5) [vxworks](/blog/tag/vxworks/) (25)
[webdev](/blog/tag/webdev/) (13) [wmii](/blog/tag/wmii/) (7)

ARCHIVE

[2015](/blog/2015/) (3) [2014](/blog/2014/) (7) [2013](/blog/2013/) (4)
[2012](/blog/2012/) (10) [2011](/blog/2011/) (15) [2010](/blog/2010/) (26)
[2009](/blog/2009/) (26) [2008](/blog/2008/) (90) [2007](/blog/2007/) (107)
[2006](/blog/2006/) (7)

Created with Django (<http://www.djangoproject.com/>) and Bootstrap
 (<http://getbootstrap.com/>) | Hosted by Linode (<http://www.linode.com/>)