Web Services and Security in the Fusion Cloud

by Terry Munday

How to generate PGP keys using GPG 1.4.5 on Linux

By Terry Munday-Oracle on May 27, 2015

Overview

This blog describes how to generate a private/public key pair using GPG version 1.4.5. The resulting public key will contain two keys, one key for signing and a subkey for encryption.

This key can be used with HCM Fusion SaaS to encrypt/decrypt files as they are transferred to and from the UCM server.

User input is noted in **RED** text.

System Requirements

- Linux
- GPG version 1.4.5

Step 1 - Confirm GPG version

The GPG version must be version 1.4.5.

Enter the following command to display the version:

```
gpg --help

gpg (GnuPG) 1.4.5
Copyright (C) 2006 Free Software Foundation, Inc.
```

Step 2 - Start the key generation process

Enter the following command to start generating your key:

```
gpg --gen-key
```

Select the type of key

```
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only) -- SELECT THIS OPTION
Your selection? 5
```

Select the key size

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048

Requested keysize is 2048 bits
```

Select the expiration time

```
Please specify how long the key should be valid.
         0 = key does not expire -- SELECT THIS OPTION
      <n>  = key expires in n days
      <n>w = key expires in n weeks
```

fusion gpg hdl hdldi import java jdeveloper linux parameters pgp proxy publisher report saas service subkey ucm web wsdl

Menu

Blogs Home

Weblog

Login

Feeds

RSS

All

/Oracle

/Personal

Comments

Atom

All

/Oracle

/Personal

Comments

```
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0

Key does not expire at all

Is this correct? (y/N) y
```

Enter user name and email

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Your Name
Email address: your.name@somedomain.com
Comment: your comment
You selected this USER-ID:
    "Your Name (your comment) <your.name@somedomain.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

Enter passphrase to protect secret key

```
You need a Passphrase to protect your secret key.

Enter passphrase: *******
Repeat passphrase: *******

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
```

You may see the following message. If you do, follow the instructions and the key generation process will start automatically.

```
Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 284 more bytes)


..+++++
...+++++
gpg: key F709C771 marked as ultimately trusted
public and secret key created and signed

gpg: checking the trustdb
gpg: 3 marginal(s) needed.  1 complete(s) needed, PGP trust model
gpg: depth: 0  valid: 1  signed: 0  trust: 0-, 0q, 0m, 0n, 0f, 1u
pub: 2048R/F709C771 2015-05-27
     key fingerprint = BDC2 5293 DB14 C218 D2DA  711C EB0A 564A F709 C771
uid                  Your Name (your comment) <your.name@somedomain.com>

Note that this key cannot be used for encryption.  You may want to use
the command "--edit-key" to generate a subkey for this purpose.
```

Key generation is complete. At this point, you have generated a private/public key pair with a public key that can be used for signing purposes. The next step is to add a subkey that will be used for encryption.

Step 3 - Add a subkey for encryption

Enter the following command to start generating your key:

```
gpg --edit-key 'Your Name'
```

```
gpg (GnuPG) 1.4.5; Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
```

```
under certain conditions. See the file COPYING for details.

Secret key is available.

pub  2048R/F709C771  created: 2015-05-27  expires: never       usage: SC
                     trust: ultimate      validity: ultimate
[ultimate] (1). Your Name (your comment) <your.name@somedomain.com>
```

Enter the edit-key command

```
Command> addkey

Key is protected.
```

Enter the passphrase you specified in step 2

```
You need a passphrase to unlock the secret key for
user: "Your Name (your comment) <your.name@somedomain.com>"
2048-bit RSA key, ID F709C771, created 2015-05-27

Enter passphrase: *******

user: "Your Name (your comment) <your.name@somedomain.com>"
2048-bit RSA key, ID F709C771, created 2015-05-27
```

Select the type of key

```
Please select what kind of key you want:
   (2) DSA (sign only)
   (4) Elgamal (encrypt only)
   (5) RSA (sign only)
   (6) RSA (encrypt only) -- SELECT THIS OPTION
Your selection? 6
```

Select the key size

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048

Requested keysize is 2048 bits
```

Select the expiration time

```
Please specify how long the key should be valid.
         0 = key does not expire -- SELECT THIS OPTION
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0

Key does not expire at all
Is this correct? (y/N) Y
Really create? (y/N)  Y

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
```

You may see the following message. If you do, follow the instructions and the subkey generation process will start automatically.

```
Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 277 more bytes)


.........+++++
```

```
......+++++

pub  2048R/F709C771  created: 2015-05-27  expires: never       usage: SC
                     trust: ultimate      validity: ultimate
sub  2048R/13DA9D02  created: 2015-05-27  expires: never       usage: E
[ultimate] (1). Your Name (your comment) <your.name@somedomain.com>
```

Note, pub is for signing (SC), sub is for encryption (E)

Exit the edit-key editor and save your changes

```
Command> q
Save changes? (y/N) y
```

Subkey generation for encryption purposes is complete. The next step is to verity and export the keys.

Step 4 - List your keys

Enter the following command to list the key on your keyring:

```
gpg -k
```

```
/home/yourname/.gnupg/pubring.gpg
---------------------------------
pub   2048R/F709C771 2015-05-27
uid                  Your Name(your comment) <your.name@somedomain.com>
sub   2048R/13DA9D02 2015-05-27
```

Step 5 - Export the public key (including subkey) in ASCII format

Enter the following command to list the keys:

```
gpg --armor --output yourname-pub-sub.asc --export 'Your Name'

cat yourname-pub-sub.asc

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.5 (GNU/Linux)

mQENBFVl4UwBCADmaSjdDpEjBtuKhb/m2W4W1I9WD8xiU0Kkg2wXeL4QbCL1RylE
hiCqJuPsZd/0zRKPGDpT5XeJrYPVGlaYsPuB+zppY29bhuFLQqZlYSHHCd3Fd0au
EkchOjT5YyXNXgYdyBFA+g8WX/Tn+Ju5ROIyXzkBo4QOdfCoU0NbF1kHXpM6J+XU
M99FEguiPd8VwOOqbWMMjxvgmiOFxSRWtrIcktdh0IOBJ6d5Saj5uthOViJi/qQs
SWvEkxXN74rj1j0kdrly64DLF6JczXJfKVVsCL6cLBtpp2Otbh74mTj8ikMEueP1
pEguBTpckRYyE7j7S71+rOeu5tYrQWmM6PydABEBAAG0M1RlcnJ5IE11bmRheSAo
S2V5IHRlc3QgMSkgPHRlcnJ5Lm11bmRheUBvcmFjbGUuY29tPokBNgQTAQIAIAUC
VWXhTAIbAwYLCQgHAwIEFQIIAwQWAgMBAh4BAheAAAoJEOsKVkr3CcdxTJUIALJq
rWTzEBZh3dC2FVxA2TuaEGdj0yYkbgqm4BxSDd0YFVlAqqBANG6KLQCKTU2BBzZ+
lAd3Z35CYSPjJsWFAzuuQX1ZCgvA4K0G1ym353qsFRWgMySzZu9xpnPJM2GyHQX1
4or4TtAD+WTnoJ3TDxqDEY2httmjeCj4/d3XnxzWH16XbEhPIsiBQ/jxWbupnq5J
oAC9R/L0vBKD4YuYDsvyVlbEwG/T4RM9PZITZPM7pE9fbAIC8y2jSrD9v7cv++bA
jCCYZl/lniRQs0BXL9kQGOwLEEd8q/N8tdUJh6T+7flo/4TXS/atNPCm5XibUEgi
5TID0BxkcfnM3peoSsa5AQ0EVWX1UAEIAOBuTMQ9P6fPVDaqro4QhXsHvgaU429n
hRDxAat0IQRQasS0FcJeyIPnoH4h1QUcyQSSQ5yo/COpukpjW1zC9m4VFajIcjsX
G1XCMozr/CCsJUkoOMDQA/nunXrBw9AK26yqL57apOQb7pSitU0VXxjEsEtp8Yv9
m7Cy7upgr/NvDD80YBsLQDXU6Zykd4oJc1cZ9IrMQOOghCvtjcz/FET3Y4yrmSHb
a2+/5MmAuKpSsSXOwJKASkFXCPRqbcji2bmOPtbHGtmKfVWLJtXSPW5eMqsuaVIl
DQKRV9IvD9rd4bZHOHKJ6fFPH4Dl2iUZ6iEmLhx2MWo/6qVO+qJ7cWsAEQEAAYkB
HwQYAQIACQUCVWX1UAIbDAAKCRDrClZK9wnHcRtOB/9TvvZrjtBJEPMbC87eFRXJ
Rf4TxOGB1qFbu+19t7r+pEeS/pOt53S0XSLo5ial0itlmaGJfK+HI8ohpmd5MqjH
WhjyvVc9XPOc0YX+E2DqpBHUqvjoVdW3YsBPKpBtAzgdkXRKDqfGWL8I6F9WUbTE
6Ap+f8jFO15eLOg9q5htnMk5L5Q28pdHDlZibJ5KfMrOyJo1ZI72+M0wVUl6TO2d
neOEyvY/XH8IrPOKB0dV4gfUI9eY4RW4I4AoSFlvAwTZh3FItHkWX4yzY0QzJTlA
Trmlhy13e2Ghj/2zWaw+0ONffUZ3tjGhumZwM9tVeEwJ7DCcRXgB76XEPkduCdh9
=fqFm
-----END PGP PUBLIC KEY BLOCK-----
```

Thats it! You now have a public key that contains a key for signing and a subkey for encryption.

Category: Oracle ::: Tags: encryption gpg linux pgp subkey :::
Permanent link to this entry :::

Comments:

Post a Comment:

Name:

guest

E-Mail:

URL:

☐ **Notify me by email of new comments**

☐ **Remember Information?**

Your Comment:

HTML Syntax: ***NOT allowed***

Please answer this simple math question

2 + 87 =

Preview Post