PGP Exercise

We are going to be installing and using an implementation of PGP called GNU Privacy Guard (GPG). There are other implementations we can use. GPG can be a bit weird, but it works pretty well once you're used to it.

1. Install GnuPG

On Debian, this is very simple:

```
  # apt-get install gnupg haveged
```

To verify that you have it installed, you can type "gpg -h" to get a list of options. There are a lot of options. We will use just a few of them.

```
  afnog@pc38:~$ gpg -h
  gpg (GnuPG) 1.4.18
  Copyright (C) 2014 Free Software Foundation, Inc.
  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
licenses/gpl.html>
  This is free software: you are free to change and redistribute it.
  There is NO WARRANTY, to the extent permitted by law.

  Home: ~/.gnupg
  Supported algorithms:
  Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
  Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
          CAMELLIA128, CAMELLIA192, CAMELLIA256
  Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
  Compression: Uncompressed, ZIP, ZLIB, BZIP2

  Syntax: gpg [options] [files]
  Sign, check, encrypt or decrypt
  Default operation depends on the input data

  Commands:

   -s, --sign [file]             make a signature
       --clearsign [file]        make a clear text signature

  ... and so on.
```

Once GnuPG is installed, you should do everything else NOT AS ROOT. You should be typing commands as user "afnog".

If you see the "#" prompt, remember YOU SHOULD NOT BE ROOT. Change to user "afnog".

Things might go weirdly wrong if you are root. Things will definitely become confusing if you switch between root and a regular user ("afnog"). Type everything as user "afnog".

I realise I just gave the same advice three times.

Don't be root.

Four times.

2. Create a new public/private key pair

Making sure that you are not root, use the gpg --gen-key option to create a new public/private key pair for user afnog (not root).

We can choose the default algorithms and key sizes. The key should be identifiable as yours, so use your real name and e-mail address (when it asks). Make the key expire in two days -- this is just a key for playing in a workshop, and we don't want anybody to accidentally think it is useful in the real world. Making it expire helps with that. You can also add a comment to make it clear what the key is for.

Generating a key pair requires the server to generate a random number, and sometimes it can take a while for the system to harvest randomness ("entropy") from paces like network interfaces. You might have to wait a little while for this to finish.

GnuPG will ask you for a password that it will use to encrypt the password when it stores it to disk. You should choose a password you can remember.

```
  afnog@pc38:~$ gpg --gen-key
  gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation,
Inc.
  This is free software: you are free to change and redistribute it.
  There is NO WARRANTY, to the extent permitted by law.

  Please select what kind of key you want:
     (1) RSA and RSA (default)
     (2) DSA and Elgamal
     (3) DSA (sign only)
     (4) RSA (sign only)
  Your selection? 1
  RSA keys may be between 1024 and 4096 bits long.
  What keysize do you want? (2048)
  Requested keysize is 2048 bits
  Please specify how long the key should be valid.
           0 = key does not expire
        <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) 2d
  Key expires at Sun May 31 07:16:33 2015 UTC
  Is this correct? (y/N) y

  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: Joe Abley
  Email address: jabley@nsrc.org
  Comment: temporary for AfNOG 2015 workshop
  You selected this USER-ID:
      "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>"

  Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
  You need a Passphrase to protect your secret key.

  .+++++
  ......+++++
  ...+++++
  ...+++++
  ......+++++
  gpg: /home/afnog/.gnupg/trustdb.gpg: trustdb created
  gpg: key DC2B8C9F 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, 0n, 0m, 0f,
1u
  gpg: next trustdb check due at 2015-05-31
  pub   2048R/DC2B8C9F 2015-05-29 [expires: 2015-05-31]
        Key fingerprint = CB14 3326 B23F 1053 D094  BF16 B9FA 5B46
DC2B 8C9F
  uid                  Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>
  sub   2048R/5DDF749B 2015-05-29 [expires: 2015-05-31]

  afnog@pc38:~$
```

3. List public keys on my keyring

To list the public keys (yours and any others you have imported),
use
the --list-keys command:

```
  afnog@pc38:~$ gpg --list-keys
  /home/afnog/.gnupg/pubring.gpg
  ------------------------------
  pub   2048R/DC2B8C9F 2015-05-29 [expires: 2015-05-31]
  uid                  Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>
```

```
  sub   2048R/5DDF749B 2015-05-29 [expires: 2015-05-31]

  afnog@pc38:~$
```

4. Export a particular key to give to someone else

To export a particular public key as text (e.g. so you can cut and
paste it into an e-mail), use --export with -a. The -a flag makes
the
output text.

You can usually refer to a particular public key using the e-mail
address associated with it, or its hexadecimal key id (shown with
--list-keys).

```
  afnog@pc38:~$ gpg -a --export jabley@nsrc.org
  -----BEGIN PGP PUBLIC KEY BLOCK-----
  Version: GnuPG v1

  mQENBFVoEngBCACggggNHvnoEXg6qRbsUYDrGI5vyKviVMQBN4PQC6Hoz4Yt3ETZ
  JJ9w0qVyRkbg+65FzDgsAt5JVH7v+I+P8Ekt64ZT8iYOce2BDxVlx2kuGLZvACrt
  LK77gK/+vxvbxKOl689otzJosFNAUfMGNHQIJPPT7S9yvA5L0NuI4I+7o6tDFgiy
  MV+iToNHpGjkSwPXonVZ+oDBPhj0Uk64gfQqOdJae8mTsgX6Lzb8LKgBPhc204RZ
  F1d7UFYcC2VeIwqfisiLXWtUKwHnfzQjpbnymo8zK47LHZ9/XpKIQ2nGnMFbGnsr
  GBadAd18d+5Sum629J5e+8/dGe1/D8c8pf2RABEBAAG0P0pvZSBBYmxleSAodGVt
  cG9yYXJ5IGZvciBBZk5PRyAyMDE1IHdvcmtzaG9wKSA8amFibGV5QG5zcmMub3Jn
  PokBPgQTAQIAKAUCVWgSeAIbAwUJAAKjAAYLCQgHAwIGFQgCCQoLBBYCAwECHgEC
  F4AACgkQufpbRtwrjJ+S6gf/bSx9w/V1f8AaeJ8VjPE3Ow1WgHqCRtPdI79Xk1tn
  hiPDsp44JlOejlHQH0gucKF/AVNHmHHaOw0JudtjuAlDAZFZgH0+1Nja2kUjt3Je
  yzX5cE+Xpq2UZ6EcvvZRnZM4LS4LZlPhFp45aerXWPxV8/VWcbb0X++YwJZJLCuy
  yqY5ufB9I84SAYgBF+O28qnYsLSp989CUyqHb8AluwSw0npLQK4s2uHxLwlclzwG
  8Y1sfPsjXycj50GkVTdZ+Vu6GErNcd7/t8gnNR2CY3l0smyfnUYPuZGLTSgWA/n+
  Zx7ptOuXSvR0iDreK1k1gSVrb4ZRkOdbjLATy8D1/M28GbkBDQRVaBJ4AQgA4WMB
  rPgO7aaIK8sMNCaP6Qnm4vYXELqc1ZmhPgiFcyNjf3yAICtl2SbV91ZA5AWLZp7D
  ExiYE/DFhgBNGroo5CjFGc4tpH5mYi22LIeldKP8mPk9+Kp1wmCLXlGjTBbGOqZ8
  kE8vV3kmm1RUrX0Y+P4ntii0zfBo3Y79AwLfrkh4E2cVtFUKQZ7XClnS2jspYSeu
  KIhsM51HQTmRbr0csBztgtU6l1jYdpuXNuYO63elB7QiUKy7IIhoPjYl3LFcJrxx
  cYo4QnfsxKvf8nH3COzLb9+SSm0riwoPE0eD4WxhkKK66tG3fID65/F7z9Zv8aIj
  0mMFoQb7XUJ1WteuTwARAQABiQElBBgBAgAPBQJVaBJ4AhsMBQkAAqMAAAoJELn6
  W0bcK4yfNyIIAIjX2cKbgWgUyMPfupYW/epYVo/w3EErSUIdSqefD16d352Ft5Dp
  wh5P9ADEXKE4zoFQgCsBGlweYNjVxw7CpL5a6B53HmF/JG1jFWWd/pbQPCfxXIPa
  Bw393K5DU0YkUPAXZNlYXuVEF6njTEa4rmER6F2l3XwKnTAD74+Yf01Pgd9mm0N9
  CJt6GIBf8CeBVhMfoMxrwXvvIBk9REFMoKGFRblOeLL75JtoB7xhuZEFMlYDUYt2
  tIgCUekACaV/gm3LEsPTWGutKlL0c21orkWAm+V0e8mig9lr3g2eP26zYxX7f7ZI
  88FiMuZxakhSdFndoyH6cpLdK0/4dnx5W5w=
  =kijt
  -----END PGP PUBLIC KEY BLOCK-----
  afnog@pc38:~$
```

5. Accept a public key from someone else

Find someone else in the room and ask them to give you their public

key. You can ask more than one person. They could e-mail you their
key, or put it on a USB thumb drive, or copy it over the network.
The way in which they transmit the public key to you does not need
to be secure. It could be printed in a newspaper and read by
thousands
of people.

In this case, I have a file called "joes-key.txt" that looks like
this:

```
  afnog@pc38:~$ more joes-key.txt
  -----BEGIN PGP PUBLIC KEY BLOCK-----
  Comment: GPGTools - https://gpgtools.org

  mQGiBEd74JoRBADBnwbFbanrMjMaq3fMMGWnZoEAoYFhB/UuBXxkFhmdTgVB45u4
  rylJlkquYoh/+KGKZVQAfqfH2kEZ9M9XIYiapGAdWKNVMjhMzanJjkvJfwFBdtp/
  gOQxjWume5ma+8FasSCAWL9BsgEew1FUdqF9OqRxiCtWzWSiTHeoUPYRQwCgzhl/
  zoYjNOXy5U7wtfQsRfz/6skEAIY/XUquYPUUmuNjz3WlMLzhUkuho8xe531fk0wS
  vHUnAEZUyBXa7IOJSnqP/nOldu46/EMEpVjchveMJY1BaynfBc3laGU9YyiSEJY+
  84YlrLZee343yY/KlhWY6yDtDgZDd4HOWVYidNnELw1TQOKB+VF0NVGUjvaTyel3
  JTXCA/45I+58pqN2mQxHmXlJz7R9tQJytfA2lgcUnx9IZpS0/gagm+clRhabWxq8

  ... and so on
```

I can import that key into my local keyring like this:

```
  afnog@pc38:~$ gpg --import joes-key.txt
  gpg: key 86523A2C: public key "Joe Abley <jabley@hopcount.ca>"
imported
  gpg: Total number processed: 1
  gpg:               imported: 1
  gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
  gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f,
1u
  gpg: next trustdb check due at 2015-05-31
  afnog@pc38:~$
```

6. Check that the public key I just imported is authentic

If I generate a fingerprint of the key I just imported, and I ask
the person whose key it is to tell me what they think the
fingerprint
of the same key is, they should match. If they don't match, the
keys must be different (and I should not trust the copy I have).

Each person can calculate the fingerprint using the --fingerprint
command:

```
  [scallop:~]% gpg --fingerprint jabley@hopcount.ca
  pub   1024D/86523A2C 2008-01-02 [expires: 2017-03-20]
      Key fingerprint = 80B7 8D10 922C ED01 CBE9  85D8 348F 0CBD
8652 3A2C
  uid       [ unknown] Joe Abley <jabley@hopcount.ca>
  sub   4096g/3D9B1B97 2008-01-02 [expires: 2017-03-20]

  [scallop:~]%
```

The "Key Fingerprint" above is short enough that you could read it
over a phone (but be sure you know who you are talking to) or,
better, compared face-to-face in a meeting like AfNOG.

When you have compared fingerprints and found them to be the same,
you know that you have a trusted copy of the other person's public
key. You can record this trust by signing your copy of the public
key you just checked. This will provide a convenient record that
you have taken the time to check that the public key is genuine.

```
  afnog@pc38:~$ gpg --sign-key jabley@hopcount.ca

  pub  1024D/86523A2C  created: 2008-01-02  expires: 2017-03-20
usage: SC
                       trust: unknown       validity: unknown
  sub  4096g/3D9B1B97  created: 2008-01-02  expires: 2017-03-20
usage: E
  [ unknown] (1). Joe Abley <jabley@hopcount.ca>


  pub  1024D/86523A2C  created: 2008-01-02  expires: 2017-03-20
usage: SC
                       trust: unknown       validity: unknown
   Primary key fingerprint: 80B7 8D10 922C ED01 CBE9  85D8 348F 0CBD
8652 3A2C

       Joe Abley <jabley@hopcount.ca>

  This key is due to expire on 2017-03-20.
  Are you sure that you want to sign this key with your
  key "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>" (DC2B8C9F)

  Really sign? (y/N) y

  You need a passphrase to unlock the secret key for
  user: "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>"
  2048-bit RSA key, ID DC2B8C9F, created 2015-05-29

  afnog@pc38:~$
```

7. Create an encrypted message to someone else

Create a text file that contains something that you can recognise.
It could be a few sentences, or a poem, or some lyrics from a song.
Personalise it, so that nobody else could reasonably guess what it
is. Put this in a file called "plain.txt".

Sign and encrypt that file, and put the results in a new file called "cypher.txt" like this. When you encrypt, you have to specify who you want to be able to decrypt the text. You need to have the public key of each recipient available.

It's a good idea also to encrypt towards yourself. Why?

```
  afnog@pc38:~$ gpg -a -se -r jabley@hopcount.ca -r jabley@nsrc.org
plain.txt

  You need a passphrase to unlock the secret key for
  user: "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>"
  2048-bit RSA key, ID DC2B8C9F, created 2015-05-29

  afnog@pc38:~$
```

The result of this is a file called "plain.txt.asc", which contains encrypted data:

```
  afnog@pc38:~$ more plain.txt.asc
  -----BEGIN PGP MESSAGE-----
  Version: GnuPG v1

  hQQOA30ShNM9mxuXEA/7Buy7tJjEOFM/B7BmT2jhiqyiSnH2k6ja67ZXcbGyMoyF
  J7qWaoKkD9dSHBbHMXXW9snIo4+p9vsv81e0icqxCmXJxpyHurKK3iXJXmjo8kTI
  /DlBSatgKJq41Yme5VfxIs+SWljqsPSwLKzQcTJVVx/T74Q9VFt0pRIIJbQUsvBN
  BZenpP7FOc9o+2LJDpZYHcjwkho5UDJjjrZmo0BJ2L3c93ESbxvf2tt0gw44KYLL
  oKyp8i4TZe9Euftke1o8ZNjhGDXCEpjsi0d6kgQKqbLoKcC5C/tNhJXbaWMdFBeR
  wk9qkfSpb2hawPVzddMrkGtRZ5PXFyj2R5FY8f4Lbn/DeySmXtDazDefvL+voCIy
  a4AvL9dobhH8DQVWvisX4kjeCgH6rPHx4b2XEwfs8R/DksnrEB40KZNprVaP88zy

  ... and so on
```

Delete the original plain-text file, so that all you have left is the encrypted copy.

```
  afnog@pc38:~$ rm plain.txt
  afnog@pc38:~$ ls plain*
  plain.txt.asc
  afnog@pc38:~$
```

Now give the encrypted text (e.g. by e-mail, by USB stick) to the people you think should be able to encrypt it, and see if they can tell you what your secret text was.

If you give the encrypted text to someone else, they should not be able to decrypt it. Your secret is safe from them.

You can also check that you can decrypt it yourself, using:

```
  afnog@pc38:~$ gpg plain.txt.asc

  You need a passphrase to unlock the secret key for
  user: "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>"
  2048-bit RSA key, ID 5DDF749B, created 2015-05-29 (main key ID
DC2B8C9F)

  gpg: encrypted with 4096-bit ELG-E key, ID 3D9B1B97, created
2008-01-02
        "Joe Abley <jabley@hopcount.ca>"
  gpg: encrypted with 2048-bit RSA key, ID 5DDF749B, created
2015-05-29
        "Joe Abley (temporary for AfNOG 2015 workshop)
<jabley@nsrc.org>"
  gpg: Signature made Fri May 29 07:50:51 2015 UTC using RSA key ID
DC2B8C9F
  gpg: Good signature from "Joe Abley (temporary for AfNOG 2015
workshop) <jabley@nsrc.org>"
  afnog@pc38:~$ ls plain.txt
  plain.txt
  afnog@pc38:~$
```

(being able to decrypt it yourself is useful, in case you forget
what it
was!)