GPG Tutorial

Andreas Hirt

July 12, 2009

1 Introduction

The purpose of this document is to give a *brief* introduction on how to set up and use GPG, the GNU implementation of PGP. The reader must first generate keys, a revocation certificate, set trust levels, and exchange keys as described in Section 2 to 5. Then the user can integrate `GPG` into Evolution to perform signing/encryption and verification/decryption of email as described in Section 6. Lastly, Section 7 describes methods to distribute keys. It is strongly encouraged that [1] be read first before using this document.

2 Creating a signing and encryption keys

To generate a new primary key-pair the command is:

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

Do this command twice: the first time it creates the `.gnupg` and the `options` directories. You are then prompted to select the kind of key you want. The default option of 1 creates both a DSA key-pair and ElGamal key-pair for signing and encryption, respectively. However, this is not recommended because you are then forced to impose the same expiration date on both keys. The roles of the two keys are different. The DSA master signing key should not expire whereas the encryption key should (see Selecting expiration dates and using subkeys in [1] for justification).

You should first create your master signing key (DSA) and then the encryption key (ElGamal) using `gpg --edit-key myKey` where `myKey` is any part of the user ID you just entered. Once you've chosen the type of key to create, you are then prompted to enter the keysize, expiration date, user ID, and passphrase, and to ensure that you chose an encrypt-only ElGamal key. For the DSA key, ensure that the keysize is the maximum (1024 bits) due to the lifetime of the master signing key and the vulnerability of DSA. For the ElGamal key at least 1024 bits are recommended. The user ID cannot be edited, so it should be created carefully (you can only add new user ID's). Lastly, the passphrase should be carefully chosen: it should not use words from a dictionary and should be a mix of both cases of alphabetic characters and non-alphabetic characters (numbers, punctuation, et cetera). The passphrase is used to unlock your private key and is the easiest method for an adversary to break your private key. Once you are done, exit via the command quit.

3 Generating a revocation certificate

You should immediately generate a revocation certificate for your primary public key using the option `--gen-revoke`, for example:

```
gpg --output revoke.asc --gen-revoke myKey
```

where `myKey` is any part of the user ID that identifies your key pair. This allows you to publish the revocation in the event that you forget your passphrase and cannot use your private key, or if your private key is lost or compromised. Ensure that the certificate is not stored where others can use it, otherwise an adversary could publish it and cause others to revoke your key-pair.

4 Exchanging Keys

To exchange keys, use the commands:

```
gpg --list keys
gpg --armor --output keyFile.gpg --export myKey
```

where `myKey` is any part of the user ID (listed in list-keys) for the key-pair that you want to exchange. It creates a file `keyFile.gpg` that contains the public key, et cetera in ASCII-armor so that it can be easily exchanged (default is binary).

5 Importing a public key, signing it, and setting trust

A public key (along with any signatures that have been added to it) can be imported with the command:

```
gpg --import keyFile.gpg
```

Importing a key does not sign it, but before you sign it you should check its fingerprint as follows. Here `keyID` is any part of the user ID of the key; if you don't know it, use the `gpg --list keys` command.

```
gpg --edit-key theKeyID
fpr
```

Because you do not want the web of trust to be corrupted, it is essential that you check the fingerprint to ensure that it is a valid key (i.e., the binding of the user's identity to the key is correct). The fingerprint should be checked with the person who owns the key, as along as you can guarantee that you are communicating with the key's true owner.

Once your are confident in the user/key binding, to sign the key, use the command:

```
sign
```

assuming that you are still in the `--edit-key theKeyID` dialog.

Once you've signed the key, to list the signatures on it and see the signature you've added use the commands:

```
check
exit
```

assuming that you are still in the `--edit-key theKeyID` dialog (note that `exit` exits the `--edit-key` dialog).

Recall that there are two different types of trusts (slightly different terminology in `GPG` and `PGP`):

1. Owner trust — how much you trust the owner to correctly sign another person's key.
2. Validity — calculated trust value using web of trust (based path length, number of fully trusted keys, and number of marginally trusted keys). The path length indicates how long the path length of signed keys leading from K back to your own key.

To set your validity parameters, there appears to be no simple interface (please let me know if you figure it out). The best method is to edit the options file (default location is `/.gnupg/options`) and change the parameters as necessary, for example:

```
#honor-http-proxy
keyserver-options honor-http-proxy

#Trust options
completes-needed 1
```

```
marginals-needed 2
max-cert-depth 4


#Keyserver option
keyserver pgpkeys.mit.edu
```

Note that while you are there, you can comment out honor-http-proxy (at end of default options file) which is deprecated and replace it with keyserver-options honor-http-proxy. You can also set your keyserver option to the keyserver of your choice (see Section 7 for details on keyservers).

To change owner trust in a key, use:

```
gpg --edit-key userID
trust
```

and follow the prompts accordingly to enter the appropriate trust.

To see the trust level for a key:

```
gpg --edit-key userID
```

The listing shows you the key with its secondary keys and all user IDs. Selected keys or user IDs are indicated by an asterisk. The trust value is displayed with the primary key: the first is the assigned owner trust and the second is the calculated trust value. Letters are used for the values:

1. - — no owner trust assigned / not yet calculated,
2. e — trust calculation has failed; probably due to an expired key,
3. q — not enough information for calculation,
4. n — never trust this key,
5. m — marginally trusted,
6. f — fully trusted,
7. u — ultimately trusted.

6 Incorporating GPG with Evolution

6.1 Setup

To run evolution, just type `evolution&` in a terminal. If you have not ran it before, it will ask you if you want to import information from Pine or Netscape Mail, it will then proceed to ask you for your e-mail information. After you enter your identification, it will ask for the e-mail server information for receiving mail, please see the following URL for the required information:

`http://www.cpsc.ucalgary.ca/Help/internet/email/emailsystinfo.php`

In addition, make sure that you change your server type for receiving mail to `POP` (`IMAP` will not work). Then it will ask you for your e-mail server information for sending mail, please see the following URL for the required information:

`http://www.cpsc.ucalgary.ca/Help/internet/email/emailsystinfo.php`

After you are done entering this information, Evolution will start. You then need to make a few more changes:

1. Left click tools
2. Left click mail settings
3. Left click the account you just created to highlight it, and then left click edit.
4. Left click the tab receiving options.
5. Left click the button leave messages on server.
6. Left click the tab security.
7. Enter your PGP Key user ID.
8. Left click the bullet "Always Encrypt to myself when sending encrypted messages." This is essential to maintain a record of the session key (encrypted with your public key) used to encrypt an email, in case you want to read an encrypted email that you sent previously.
9. Optionally, left click the bullet "Always sign outgoing messages when using this account."

You are now done the set up and can use Evolution to send/receive encrypted mail.

6.2 Sending signed and/or encrypted e-mail

To send a message, choose new message from the top tool bar on the left hand side. Write your message and then left click "Security" on the top right hand side. Choose "PGP sign" and "PGP encrypt" as needed. When you send the mail, it will ask you for your secret pass-phrase to unlock your private key.

6.3 Receiving signed and/or encrypted e-mail

When receiving an e-mail that is only signed, there will be a lock icon on the bottom of your screen. You can left click it to see the signature information. When receiving an e-mail that is signed and encrypted or just encrypted you will be prompted for your secret pass-phrase to unlock your private key in order to perform decryption.

6.4 Maintaining your key ring

Unfortunately, there is no GUI in Evolution for maintaining your keyring, and this must be done in a terminal as described in Section 2 to 5.

7 Distributing keys

In order to send someone an e-mail with PGP, they must also be using PGP and you must know their keys. Common methods are to publish keys on personal websites, include them in one's `.plan` file, exchange them via e-mail, exchange them in person, or store/retrieve them on a GPG keyserver. However, one *MUST* validate the fingerprint on the key as described in Section 5. Another reason to distribute keys is to collect signatures on your key. Have the other user send you your updated and newly signed key, update your key by importing it and checking the signatures, and then re-publish it on a key server.

There are server popular keyservers around the world, and they synchronize themselves. Just pick a keyserver that is close to you on the Internet. Note that these keyservers can also be used to look up keys. For a list of keyservers see

`http://openpksd.org/kslist.html`

For example, you can go to:

http://www.keyserver.net/en/

and find my key by searching for hirt@cpsc.ucalgary.ca.

Another method to send and receive keys efficiently is:

```
gpg --send-keys --keyserver pgpkeys.mit.edu IDs
gpg --recv-keys --keyserver pgpkeys.mit.edu IDs
```

8 Common Questions

Key IDs are found in the second column output by --list-keys. For example, in the short sample output:

```
---------------------------------
pub  1024D/E05E7378 2004-03-02 Andreas Hirt (Student) <hirt@cpsc.ucalgary.ca>
sub  1024g/9C685721 2004-03-02
...
```

the key IDs are E05E7378 and 9C685721. Note that ID is not the same as user ID.

One question will certainly come up is: "Why do I get the error gpg: Warning: using insecure memory!?" The reason you get this is because the executable (to find it use whereis gpg) does not have locked memory pages (instead of executable x permission you want s permission). Locked memory pages prevent the OS from writing them to disk, i.e., you don't want the gpg executable to page information temporarily such as your unencrypted secret key to disk. To change this requires root permission and can have a negative impact upon performance. If you think that the gpg executable permissions should be changed to use locked memory pages, please e-mail bugzilla@cpsc.ucalgary.ca and let them know. It is possible to supress this warning in the options file, but this is not recommended, lest your forget about the vulnerability.

For more information see [1]. In addition, an excellent FAQ for GPG can be found at

http://www.gnupg.org/documentation

References

[1] M. Copeland, J. Grahn, and D. Wheeler. "The GNU Privacy Handbook", http://www.gnupg,orf/gph/en/manual.html, 1999.