- Partners
- Support
- Community
- Ubuntu.com

- Page History
- Login to edit

Search

GnuPrivacyGuardHowto

"GnuPG uses public-key cryptography so that users may communicate securely. In a public-key system, each user has a pair of keys consisting of a private key and a public key. A user's private key is kept secret; it need never be revealed. The public key may be given to anyone with whom the user wants to communicate." From The GNU Privacy Handbook

Contents

GnuPG, GPG, PGP and OpenPGP

The terms "OpenPGP", "PGP", and "GnuPG / GPG" are often used interchangeably. This is a common mistake, since they are distinctly different.

- **OpenPGP** is technically a *proposed standard*, although it is widely used. OpenPGP is not a program, and shouldn't be referred to as such. PGP and GnuPG are computer programs that implement the OpenPGP standard.

- **PGP** is an acronym for Pretty Good Privacy, a computer program which provides cryptographic privacy and authentication. For more information, see this Wikipedia article.

- **GnuPG** (aka **GPG**) is an acronym for GNU Privacy Guard, another computer program which provides cryptographic privacy and authentication. For further information on GnuPG, see this Wikipedia article.

Generating an OpenPGP Key

The core package required to start using OpenPGP, `gnupg`, is installed by default on Ubuntu systems, as is seahorse, a GNOME application for managing keys. It is called "Passwords and Keys" in Ubuntu.

There are several programs which provide a graphical interface to the GnuPG system.

- Enigmail, an OpenPGP plugin for Mozilla Thunderbird. Enigmail was available in the "Main" repository through Intrepid, but can be found in the "Universe" repository since Jaunty.

  ```
  sudo apt-get install enigmail
  ```

- GNU Privacy Assistant is a graphical user interface for the GnuPG (GNU Privacy Guard). GPA is available in the "Universe" repository. See Repositories for further information on enabling repositories.

  ```
  sudo apt-get install gpa
  ```

- Seahorse is a GNOME application for managing encryption keys. It also integrates with nautilus, gedit, and in other places for encryption operations. Seahorse is available in the "Main" repository.

  ```
  sudo apt-get install seahorse
  ```

- KGPG is a simple, free, open source KDE frontend for gpg. KGPG is available in the "Main" repository since Intrepid, or the "Universe" repository in earlier releases.

  ```
  sudo apt-get install kgpg
  ```

- Kleopatra is another KDE frontend for gpg that is integrated with the KDE PIM (although you need to install it separately for now). Kleopatra is available in the "Universe" repository and it includes S/MIME backend:

  ```
  sudo apt-get install kleopatra
  ```

You can also generate keys using these programs. Use the section below for recommendations on settings.

Using GnuPG to generate a key

- Open a terminal and enter:

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

  - If you are using GnuPG version 1.4.10 or newer, this will lead to a selection screen with the following options:

    ```
    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)
    ```

    Select (1), which will enable both encryption and signing.
  - If you are using an older version, the selection screen will have the following options:

```
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
```

We suggest you select (5). We will generate an encryption subkey later.

- `What keysize do you want? (2048)`

  A keysize of 2048 (which is the default) is also a good choice.

  ```
   * Key is valid for? (0)
  ```

  Most people make their keys valid until infinity, which is the default option. If you do this don't forget to revoke the key when you no longer use it (see below).

- Hit `Y` and proceed.

  ```
  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: Dennis Kaarsemaker
  Email address: dennis@kaarsemaker.net
  Comment: Tutorial key
  You selected this USER-ID:
      "Dennis Kaarsemaker (Tutorial key) <dennis@kaarsemaker.net>"
  ```

- Make sure that the name on the key is not a pseudonym, and that it matches the name in your passport, or other government issued photo-identification! You can add extra e-mail addresses to the key later.

- Type `O` to create your key.

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

- You will be asked for your passphrase twice. Usually, a short sentence or phrase that isn't easy to guess can be used. You would be asked to tap on the keyboard or do any of the things you normally do in order for randomization to take place. This is done so that the encryption algorithm has more human-entered elements, which, combined with the passphrase entered above, will result in the user's private key.

Forgetting your passphrase will result in your key being useless. Carefully memorize your passphrase.

- After you type your passphrase twice, the key will be generated. Please follow the instructions on the screen till you reach a screen similar to the one below.

  ```
  gpg: key D8FC66D2 marked as ultimately trusted
  public and secret key created and signed.

  pub   1024D/D8FC66D2 2005-09-08
        Key fingerprint = 95BD 8377 2644 DD4F 28B5  2C37 0F6E 4CA6 D8FC 66D2
  uid                  Dennis Kaarsemaker (Tutorial key) <dennis@kaarsemaker.net>
  sub   2048g/389AA63E 2005-09-08
  ```

The key-id is `D8FC66D2` (yours will be different).

It is probably a good idea to set this key as default in your *~/.bash_profile* or *~/.profile*. Doing this will allow applications using GPG to automatically use your key.

- Set your key as the default key by entering this line in your *~/.bash_profile* (along with any other environment variables to be exported).

  ```
  export GPGKEY=D8FC66D2
  ```

  - Please note that this will be sourced only during your next login. To use it straight away, you should run the export command in the the current shell.
- Now restart the gpg-agent and set the relevant environment variable.

  ```
  killall -q gpg-agent
  eval $(gpg-agent --daemon)
  export GPGKEY=D8FC66D2
  ```

Encryption

- If you created an "RSA (sign only)" earlier, you will probably want to add encryption capabilities. Assuming you edited *~/.bash_profile* as above, open a terminal again and enter:

  ```
  gpg --cert-digest-algo=SHA256 --edit-key $GPGKEY
  ```

- This will present a dialog like the following:

  ```
  Secret key is available.

  pub   2048R/D8FC66D2  created: 2005-09-08  expires: never       usage: SC
                       trust: ultimate      validity: ultimate
  [ultimate] (1). Dennis Kaarsemaker (Tutorial key) <dennis@kaarsemaker.net>

  Command>
  ```

- To create a subkey, enter 'addkey'. You will have to enter your key's passphrase, and then you'll see a somewhat familiar series of dialogues:

  ```
  Please select what kind of key you want:
     (2) DSA (sign only)
     (4) Elgamal (encrypt only)
     (5) RSA (sign only)
     (6) RSA (encrypt only)
  ```

- Choose 6.

  ```
  What keysize do you want? (2048)
  ```

- Again, 2048 is a sensible default.

  ```
  Key is valid for? (0)
  ```

- Choose whether this encryption subkey is set to expire (default: it doesn't). Then confirm that you want to make this subkey.

  ```
  pub   2048R/D8FC66D2  created: 2005-09-08  expires: never       usage: SC
                       trust: ultimate      validity: ultimate
  sub   2048R/389AA63E created: 2005-09-08  expires: never       usage: E
  [ultimate] (1). Dennis Kaarsemaker (Tutorial key) <dennis@kaarsemaker.net>
  Command>
  ```

- Enter 'save', then 'quit.' Your key is now capable of encryption.

Creating a revocation key/certificate

- A revocation certificate must be generated to revoke your public key if your private key has been compromised in any way.
- It is recommended to create a revocation certificate when you create your key. Keep your revocation certificate on a medium that you can safely secure, like a thumb drive in a locked box.

- You can create a revocation certificate by :

  ```
  gpg --output revoke.asc --gen-revoke $GPGKEY
  ```

- The revocation key may be printed and/or stored as a file. Take care to safeguard your revocation key.

Anybody having access to your revocation certificate can revoke your key, rendering it useless.

Making an ASCII armored version your public key

There are several sites out there that also allow you to paste an ASCII armored version your public key to import it. This method is often preferred, because the key comes directly from the user. The reasoning behind this preference is that a key on a keyserver may be corrupted, or the keyserver unavailable.

- Create an ASCII armored version of your public key using GnuPG with this command:
  - `gpg --output mykey.asc --export -a $GPGKEY`
  - This is the command using our example:

    ```
    gpg --output mykey.asc --export -a D8FC66D2
    ```

Uploading the key to Ubuntu keyserver

This section explains how to upload your **public** key to a keyserver so that anyone can download it. Once you have uploaded it to one keyserver, it will propagate to the other keyservers. Eventually most of the keyservers will have a copy of your key. You can accelerate the process by sending your key to several keyservers.

- Using GnuPG:

  ```
  gpg --send-keys --keyserver keyserver.ubuntu.com $GPGKEY
  ```

  - Using our example, the command would be:

    ```
    gpg --send-keys --keyserver keyserver.ubuntu.com D8FC66D2
    ```

- Using a web browser to submit to Ubuntu key server:
  - Export your key by issuing this command:

    ```
    gpg --export -a "Key-ID" > mykey.asc
    ```

  - Copy the content of *mykey.asc*:
  - Open http://keyserver.ubuntu.com/ in a browser window.
  - Paste the copied content in the box under the label, `Submit a key`
  - Click on `Submit this key to the keyserver!`

Note that keyserver.ubuntu.com is only reachable via IPv4.

Reading OpenPGP E-mail

OpenPGP implementations can be used to digitally sign, encrypt, and decrypt email messages for heightened security. You can register your own personal OpenPGP keys with Launchpad, and under some situations, Launchpad will send you signed or encrypted email. You would then use OpenPGP support in your mail reader to decrypt these messages or verify a message's digital signature. Of course, you can also use the OpenPGP support in your mail reader to trade encrypted messages with your colleagues, or sign your own messages so that others can have better assurances that the email that appears to come from you actually does come from you.

The instructions below are not intended to provide you with detailed information on OpenPGP, its various implementations, or its use. These instructions simply provide links that can help you set up your mail reader to be compatible with OpenPGP signed and/or encrypted email.

We need your help to flesh out these instructions!

Linux mail readers

This section is not all inclusive. Please feel free to add additional mail clients.

Evolution

Evolution has built-in support for OpenPGP. Look under the Security tab when you edit accounts.

- Open Evolution and go to **Edit->Preferences**.
  - Choose your email account, click on it, and then click **Edit**.
  - Click on the **security** tab.
  - In the **PGP/GPG Key ID**: box, paste your **KEY-ID**.
  - Click **OK**. Click **Close**.
- If you want to use your key in any new email, simply click on the **Security** menu item in your new mail message, and then click on **PGP Sign**.

KMail

Kmail / Kontact has built-in support For Gutsy, and later releases, everything required is installed by default. See the Kmail GPG page for details.

Claws Mail

Claws Mail supports OpenPGP through the plugin `claws-mail-pgpinline`

- `claws-mail-pgpinline` is available in the "Universe" repository.
- `sudo apt-get install claws-mail-pgpinline`
- The plugin may have to be loaded manually after installing it. Open Claws Mail and select **Configuration -> Plugins**
  - If **PGP/Core** and **PGP/inline** are in the Plugins dialogue box, the plugins are loaded correctly.

    - Otherwise, click on the **Load Plugin** button towards the bottom of the window. In the file selection dialogue, select *pgpinline.so* and click the **Open** button.

- When Claws Mail tries to open encrypted e-mail, the program will prompt for your key's passphrase and then show the e-mail with the decrypted message.

Thunderbird

- Thunderbird supports OpenPGP through the enigmail plugin.
- Enigmail is available in the "Main" repository.

- `sudo apt-get install enigmail`

- Configure OpenPGP support in Thunderbird under **Enigmail->Preferences** and add under **GnuPG executable path**. The path for GnuPG is */usr/bin/gpg*.

Mutt

- Create a *~/.mutt* directory and copy this file into it: */usr/share/doc/mutt/examples/gpg.rc*

- Append this line to the muttrc configuration file.

  ```
  source ~/.mutt/gpg.rc                           # Use GPG
  ```

- If you're using Mutt 1.5.13, you'll need to fix the paths to pgpewrap as detailed in this post

Miscellaneous/all platforms (web mail)

This section in need of expansion. Please feel free to add any additional plugins for Firefox or other browsers.

Gmail

- If you use the Chrome or the Chromium browser you can use **Goopg** to sign and verify emails directly from the Gmail web interface. For details click here).

It's All Text!

- It's All Text! is a ***Firefox*** extension which allows you to edit your mail in your preferred local text editor.

- If your editor supports it, this can make handling of encrypted mail easier.

- For example, you could use gnupg.vim and a local Vim instance.

Validation on Launchpad

You need to tell Launchpad about your OpenPGP key(s) to be able to sign the Ubuntu Code of Conduct (and thus become an Ubuntero) and to build packages using HCT.

OpenPGP keys and Launchpad

- Visit the OpenPGP Keys page once logged into Launchpad. Paste your key fingerprint into the textbox:

```
  gpg --fingerprint
  ```

  - The key fingerprint would be something like:

    ```
    95BD 8377 2644 DD4F 28B5  2C37 0F6E 4CA6 D8FC 66D2
    ```

    Launchpad will send you an email which you will have to decrypt. You can save the text to a file: (Sample message- make sure not to alter format)

    ```
    -----BEGIN PGP MESSAGE-----
    Version: GnuPG v1.4.3 (GNU/Linux)

    hQIOA0THhKozD+K5EAf9F3PcOL2iU6onH2YsvB6IKDXNxbK0NBVy6ppxcNq8hoTe
    cuHvzWLFfh1ehhSNe1V6xpuFnt5sJoeA4qEEOxez3HmY80tKIKMPLyhC/8JiSIW9
    fwuxj4C0F6pdyrpvGbQAzfPEFk/P1AtIHXm4WLXduhBT7YEpmUk/I4A/KlSrKoiP
    J5vBtbroUyp2jvIhDUmY7ToU+ifrDe3+VP1ZzSEJzOOXec9oPbcbvf5NptXA7Hbp
    S0ElBAcLjKpAu7VKotCwFZIsVXDHT/mxf2qm88bGIrlXS5uTzvmyhQps1KmyNiCz
    I0i5kSVvHZWyVZ+8FrROLqYAqqnEIMg9hUnbFAervgf/YiYs0xxWLYf9e14eoMZA
    ranGT72q/JHmBNBYenOijaquFNi1TH5J8Udtt2RfdyRUlmGilxRvtIYL8gpnuNpS
    +GHOoBWUN2f4nawaDeqgrf6Nt3qQWWLO4iJPgieejFP2FP6zkLme1t7dXo+z1ary
    EZuxSLtKIWkOFEZ8Gcn02hBgOhJZucnkF6BmVW9dr1C4QEAmGM631uqfsp5PapAn
    yjHbEU1L2R9i7vPtJNRr6ubFLWg1Yhfv63ByxSx/WQHMMqlrbL+moXBGED3L2hM8
    7FP9eapBRgmS+Bda9ArcGMUElTOkWoUYIOPyLOYmo15LvbxHOVaXjn7+fDgr2S1J
    R9LArwHycmdKKelRww+ZvylHIfq8xy10atRQIYawchh9A1myXD1TlWbrrIkodQJF
    iEpO2i1LKvqwZHOx3szT4hF+44tNFzQIL1j+zF5Hrt2WOTnS5WXGgGRtfEd8F7fN
    khQZOAdhwrnlY+yknruC8Y8Jm8vM57+KnPgBfvxuxzLX1XFTfTZCHXeUmwwu3mga
    m+6WzckeBGBDHKK6GqwFoOAykTwjyqOZaty7DPHeoINc0tLMVr9Ks64DScf8bgh4
    MkNonA0YhMQbkmwRc33APw441+/iLw5gqndQdX44kKqC71dG6LqanAOjD29Xj3JV
    ZBsjg95Jrx7Sx+i/V0PUeaU9QjCT0Q1jEy1Bcs8NYtTJnpG+4oHYJ0pyiGxIquQH
    V9E+hW6Qehx5DbsIXEvfeaBBHOfAHHOhUH14WK4bsJWm8wZ50XiYBZrNFOqzsm13
    2STcY4VIoJp3Uw2qNyvZXQUhpndlfgQGO14CMSadzDn6Vts=
    =hTe6
    -----END PGP MESSAGE-----
    ```

- Now run:

  ```
  gpg --decrypt file.txt
  ```

  - You will need to enter your passphrase.
- The message will be displayed along with the link you must follow to confirm your key in Launchpad.
- Follow the link, enter your Launchpad password as asked, and you are done!

Validating using Firefox and FireGPG

- If you are on gmail, using the FireGPG addon, simply scroll down and click "decrypt this mail". You will now see the decrypted message with a link and a token. Copy that URL:

  ```
  https://launchpad.net/token/somealphanumerictoken
  ```

  - Follow the link and click on "Confirm". Please note that validation does take some time. If you run into an internal 500 server, simply try again with the same token.
- A confirming page should appear once the validation is successfully completed.

Signing Data

Signing data is helpful in verifying if the data from a person is indeed from that person. A typical scenario is described below.

Launchpad Key Signing

When you've set up GnuPG and have a key in the strong set, it is time to sign the Ubuntu Code Of Conduct if you want to become an Ubuntu member or Ubuntero. Signing is done in 3 easy steps:

1. Download the code of conduct from https://launchpad.net/codeofconduct/2.0/+download.

2. Run the command

   ```
   gpg --clearsign UbuntuCodeofConduct-2.0.txt
   ```

3. Upload the contents of UbuntuCodeofConduct-2.0.txt.asc on https://launchpad.net/codeofconduct/2.0/+sign

Getting your key signed

The whole point of all this is to create a web of trust. By signing someone's public key, you state that you have checked that the person that uses a certain keypair, is who they says they are and really is in control of the private key. This way a complete network of people who trust each other can be created. This network is called the *Strongly connected set*. Information about it can be found at http://pgp.cs.uu.nl/

In summary,

1. Locate someone that lives near you and can meet with you to verify your ID. Sites like http://www.biglumber.com/ are useful for this purpose

2. Arrange for a meeting. Bring at least one ID with photo and printed fingerprint of your OpenPGP key, ask the same from the person you will be meeting with.
3. Print copies of your public key
   - get the last eight digits of your fingerprint: 0995 ECD6 3843 CBB3 C050 28CA E103 6EED **0123 4567**
   - terminal: gpg --fingerprint 01234567 >> key.txt
   - print the resulting key.txt file and bring as many copies to the meeting as you expect to have people sign
4. Meet, verify your IDs and exchange OpenPGP key fingerprints
5. Sign the key of the person you've just met. Send him/her the key you've just signed.
6. Update your keys on the keyserver, the signature you've just created will be uploaded.

Keysigning Guidelines

Since a signature means that you checked and verified that a certain public key belongs to a certain person who is in control of the accompanying private key, you need to follow these guidelines when signing peoples keys:

During the Event

1. Keysigning is always done after meeting in person

2. During this meeting you hand each other your OpenPGP key fingerprint and at least one government issued ID **with a photograph**. These key fingerprints are usually distributed as key fingerprint slips, created by a script such as gpg-key2ps (package: signing-party)

3. You check whether the name on the key corresponds with the name on the ID and whether the person in front of you is indeed who they say they are.

After the Event

You now have the printed public key information from the other participants.

Example key IDs for the other participants will be E4758D1D, C27659A2, and 09026E7B. Replace these IDs with the key IDs you received from the other participants.

1. retrieve the keys:
   - gpg --recv-keys E4758D1D C27659A2 09026E7B
2. sign the keys:
   - gpg --sign-key E4758D1D
   - gpg --sign-key C27659A2
   - gpg --sign-key 09026E7B
3. export the keys
   - gpg --armor --export E4758D1D --output E4758D1D.signed-by.01234567.asc
   - gpg --armor --export C27659A2 --output C27659A2.signed-by.01234567.asc
   - gpg --armor --export 09026E7B --output 09026E7B.signed-by.01234567.asc
4. Email the key users (use the email address that was part of the key's user ID) and attach the corresponding signature file - or - send their signed key to the key server:
   - gpg --send-keys --keyserver keyserver.ubuntu.com E4758D1D
5. Once you receive your signed key import them to your keyring:
   - gpg --import 01234567.signed-by.E4758D1D.asc
   - gpg --import 01234567.signed-by.C27659A2.asc
   - gpg --import 01234567.signed-by.09026E7B.asc
6. You should see your keys:
   - gpg --list-sigs 01234567
7. Send your keys to the keyserver:
   - gpg --send-keys 01234567

Congrats you have now entered a web of trust or enlarged an existing one.

Backing up and restoring your key pair

Why should you back up your key pair? If you lose your key pair:

- Any files encrypted with the lost key pair will be unrecoverable.
- You will not be able to decrypt mails sent to you.
  - Decrypting emails sent to you requires your **private key**, this key is not stored on the keyservers.

If you lose your keypair you should revoke your key. ***This cannot be done without a revocation key.***

Backing up your public key

- List your public keys:

  ```
  gpg --list-keys
  ```

- Look for the line that starts something like "pub 1024D/". The part after the 1024D is the key_id. To export the key:

  ```
  gpg -ao _something_-public.key --export key_id
  ```

Backing up your private key

- List your secret keys:

  ```
  gpg --list-secret-keys
  ```

- Look for the line that starts something like "sec 1024D/". The part after the 1024D is the key_id. To export the secret key:

  ```
  gpg -ao _something_-private.key --export-secret-keys key_id
  ```

Restoring your keys

- To restore your keys - copy the two files created above to the machine and type:

  ```
  gpg --import _something_-public.key
  gpg --import _something_-private.key
  ```

Make sure you protect these files!

Revoking a keypair

In the event your keys are lost or compromised, you should revoke your keypair. This tells other users that your key is no longer reliable.

❗ For security purposes, there is no mechanism in place to revoke a key without a revocation key. As much as you might want to revoke a key, the revocation key prevents malicious revocations. Guard your revocation key with the same care you would use for your private key.

- To revoke your key you need to first create a revocation key. Indicate the key to be revoked and direct the output to a file. Continuing with the example used previously:

```
gpg --gen-revoke D8FC66D2 >revoke.asc
```

- Import your revocation key:

```
gpg --import revoke.asc
```

- Upload the revocation key to your keyserver of choice. Here, the example uses Ubuntu's key server:

```
gpg --keyserver keyserver.ubuntu.com --send-key D8FC66D2
```

Un-revoking a keypair

If you unintentionally revoke a key, or find that your key has in fact not been lost or compromised, it is possible to un-revoke your key. First and foremost, ensure that you do not distribute the key, or send it to the keyserver.

- Export the key

```
gpg --export <key> > key.gpg
```

- Split the key into multiple parts. This breaks the key down into multiple parts.

```
gpgsplit key.gpg
```

- Find which file contains the revocation key. In most cases, it is 000002-002.sig, however you should make sure by using the following. If the sigclass is 0x20, you have the right file. Delete it.

```
gpg --list-packets 000002-002.sig
```

- Put the key back together

```
cat 0000* > fixedkey.gpg
```

- Remove the old key

```
gpg --expert --delete-key <key>
```

- Import the new key

```
gpg --import fixedkey.gpg
```

GPG 2.0

GPG 2.0 is not installed as a default application on Ubuntu.

GPG 2.0 is the new kid on the block. GPG 2.0 is aimed or done for the desktops rather than embedded or server applications.

- GnuPG2 is available in the "Main" repository since Intrepid, or in the "Universe" repository in earlier releases.
    - If you want to use gnupg2 with the **firegpg** firefox extension, you need to install gnupg2 first.
- More information of GnuPG2 can be found here
- If you are going to use gpg2 for the same purposes as outlined above then you just need to add `2` to the gpg command.

  ```
  gpg2 --gen-key
  ```

Tips and Tricks

- Add your key to *~/.bash_profile* by adding a line similiar to `export GPGKEY=YOUR-KEY-ID`
- gnupg-agent and pinentry-gtk2 are packages that facilitate not having to enter the password for your key every time you want to use it. Open the file `~/.gnupg/gpg.conf` in your favorite editor. Browse through it and change what you like. A few useful things to change are:
    - keyserver-options auto-key-retrieve
    - use-agent (the Ubuntu default for Gutsy and later releases.)

The former makes gpg automatically retrieve GPG keys when verifying signatures. The latter makes you use gpg-agent, which is very useful if you use GPG a lot but don't like typing your password all the time. It is also required for some programs (such a Kmail) to sign or encrypt messages). Gnupg-agent and pinentry are in Main for Gutsy and automatically installed/configured in Kubuntu. If you are upgrading from Ubuntu 7.04 (Fiesty), the file ~/.gnupg/gpg.conf may have failed to be created by default in your home directory due to a bug in the gnupg package. In that case, GPG agent integration will not be enabled by default. If you have not created your own gpg.conf, you can correct this issue by running `cp /usr/share/gnupg/options.skel ~/.gnupg/gpg.conf`. If you do have a gpg.conf and are affected by this issue, that command would overwrite it with Ubuntu's default options and wipe any customizations you have made; you can still correct the issue by running `echo use-agent >> ~/.gnupg/gpg.conf` instead.

Now create the file ~/.gnupg/gpg-agent.conf with the following content:

```
pinentry-program /usr/bin/pinentry-gtk-2
default-cache-ttl 86400
max-cache-ttl 86400
```

This will make gpg-agent use pinentry-gtk2 and it will remember your password for 24 hours (please consider the security implications for doing this - anyone gaining access to your computer for 24 hours would then be able to sign anything with your key). For Kubuntu, use pinentry-qt4 instead.

* Changing your password. If you wish to change the password of a key, you can use

```
    gpg --edit-key userid
```

(the 'real name' part of the userid suffices). Choose `passwd` in the menu and enter the new password twice. You can leave the menu using `quit` .

Related Articles

- GPGKeyOnUSBDrive
- UnsignedGpgKey
- GPGsigningforSSHHowTo

Resources

- GNUPG Manual
- Using GnuPG, on Linux Gazette
- A short history of PGP / OpenPGP / GnuPG
- UbuntuForums Howto, thanks to Kassetra
- Beginners Guide to GnuPG -- Ubuntu Forums
- http://www.biglumber.com
- http://wiki.openskills.org/OpenSkills/OpenPGP+Key+Backup -- Backing up and restoring your keys
- http://en.wikipedia.org/wiki/Gnupg -- Wikipedia article

- http://moser.cm.nctu.edu.tw/gpg.html -- GnuPG for everyday use
- Creating & Utilizing PGP Forum Tutorial
- The Keysigning Party HOWTO
- HOWTO prep for migration off of SHA-1 in OpenPGP

CategorySoftware CategorySecurity CategoryGNU

GnuPrivacyGuardHowto (last edited 2015-12-15 21:38:50 by anthony-geoghegan @ ip-84-203-37-102.broadband.digiweb.ie[84.203.37.102]:anthony-geoghegan)