



[Translation\(s\)](#): [English](#) - [Français](#) - [Italiano](#)

Using OpenPGP subkeys in Debian development

What are keys?

In public key cryptography, a key is actually a pair: a public key, and a private key. You use the private key to digitally sign files, and others use the public key to verify the signature. Or, others use the public key to encrypt something, and you use the private key to decrypt it.

As long as only you have access to the private key, other people can rely on your digital signatures being made by you, and you can rely on nobody else being able to read messages encrypted for you.

GnuPG, the implementation used in Debian, picks the right key at any one time.

What are subkeys?

OpenPGP further supports subkeys, which are like the normal keys, except they're bound to a master key pair. A subkey can be used for signing or for encryption. The really useful part of subkeys is that they can be revoked independently of the master keys, and also stored separately from them.

In other words, subkeys are like a separate key pair, but automatically associated with your main key pair.

GnuPG actually uses a signing-only key as the master key, and creates an encryption subkey automatically. Without a subkey for encryption, you can't have encrypted e-mails with GnuPG at all. Debian requires you to have the encryption subkey so that certain kinds of things can be e-mailed to you safely, such as the initial password for your debian.org shell account.

Why?

Subkeys make key management easier. The master key pair is quite important: it is the best proof of your identity online, at least for Debian, and if you lose it, you'll need to start building your reputation from scratch. If anyone else gets access to your private master key or its private subkey, they can make everyone believe they're you: they can upload packages in your name, vote in your name, and do pretty much anything else you can do. This can be very harmful for Debian. You might dislike it as well. So you should keep all your private keys safe.

You should keep your private master key very, very safe. However, keeping all your keys extremely safe is inconvenient: every time you need to sign a new package upload, you need to copy the packages onto suitable portable media, go into your sub-basement, prove to the armed guards that you're you by using several methods of biometric and other identification, go through a deadly maze, feed the guard dogs the right kind of meat, and then finally open the safe, get out the signing laptop, and sign the packages. Then do the reverse to get back up to your Internet connection for uploading the packages.

Subkeys make this easier: you already have an automatically created encryption subkey and you create another subkey for signing, and you keep those on your main computer. You publish the subkeys on the normal keyserver, and everyone else will use them instead of the master keys for encrypting messages or verifying your message signatures. Likewise, you will use the subkeys for decrypting and signing messages.

You will need to use the master keys only in exceptional circumstances, namely when you want to modify your own or someone else's key. More specifically, you need the master private key:


- when you sign someone else's key or revoke an existing signature,
- when you add a new UID or mark an existing UID as primary,
- when you create a new subkey,
- when you revoke an existing UID or subkey,
- when you change the preferences (e.g., with `setpref`) on a UID,
- when you change the expiration date on your master key or any of its subkey, or
- when you revoke or generate a revocation certificate for the complete key.

(Because each of these operation is done by adding a new self- or revocation signatures from the private master key.)

Since each link of the Web of Trust is an endorsement of the binding between a public key and a user ID, OpenPGP certification signatures (from the signer's private master key) are relative to a UID and are irrelevant for subkeys. In particular, subkey creation or revocation does not affect the reputation of the master key. So in case your subkey gets stolen while your master key remains safe, you can revoke the compromised subkey and replace it with a new subkey without having to rebuild your reputation and without reducing reputation of other people's keys signed with your master key.

How?

Unfortunately, GnuPG's user interface is not entirely fun to use. We'll take you through the necessary steps below.

These instructions assume you use one computer, and keep the master keys on an encrypted USB flash drive, or preferably at least two (you should keep backups of your secret keys). We also assume you already have a key; if not, see  <http://keyring.debian.org/creating-key.html> for instructions.

1. Make backups of your existing GnuPG files (`$HOME/.gnupg`). Keep them safe. If something goes wrong during the following steps, you may need this to return to a known good place.
 - `umask 077; tar -cf $HOME/gnupg-backup.tar -C $HOME .gnupg`(note: `umask 077` will result in restrictive permissions for the backup.)
2. Create a new subkey for signing.
 - Find your key ID: `gpg --list-keys yourname`
 - `gpg --edit-key YOURMASTERKEYID`
 - At the `gpg>` prompt: `addkey`
 - This asks for your passphrase, type it in.
 - Choose the "RSA (sign only)" key type.

- It would be wise to choose 4096 (or 2048) bit key size.
 - Choose an expiry date (you can rotate your subkeys more frequently than the master keys, or keep them for the life of the master key, with no expiry).
 - GnuPG will (eventually) create a key, but you may have to wait for it to get enough entropy to do so.
 - Save the key: `save`
3. You can repeat this, and create an "RSA (encrypt only)" subkey as well, if you like or if you need to. As mentioned above, keep in mind that the default option when initially creating a new keypair is to create an encryption subkey, so you probably have one already. In any case, for Debian, just the signing key is sufficient.
4. Now copy `$HOME/.gnupg` to your USB drives.
5. Here comes the tricky part: you need to remove the private master key.
- If you are using GnuPG 2.1 or later, all you have to do is to delete the file `$HOME/.gnupg/private-keys-v1.d/KEYGRIP.key`, where KEYGRIP is the "keygrip" of the master key which can be found by running `gpg2 --with-keygrip --list-key YOURMASTERKEYID`. (The private part of each key pair has a keygrip, hence this command lists one keygrip for the master key and one for each subkey.) Note however that if the keyring has just been migrated to the new format, then the now obsolete `$HOME/.gnupg/secring.gpg` file might still contain the private master key: thus be sure to delete that file too if it is not empty.
 - Unfortunately GnuPG <2.1 does not provide a convenient way to do remove the private master key. Instead, we need to export the subkey, remove the private key, and import the subkey back.
 - Export the subkeys:
`gpg --output secret-subkeys --export-secret-subkeys YOURMASTERKEYID`. Alternatively, specify the subkey IDs each followed with an exclamation mark to choose which subkeys to export:
`gpg --output secret-subkeys --export-secret-subkeys SUBKEYID! [SUBKEYID! ...]`
 - Remove your master secret key: `gpg --delete-secret-keys YOURMASTERKEYID`
 - Import the subkeys back: `gpg --import secret-subkeys`
 - Remove the file containing the private subkeys: `rm secret-subkeys`

Verify that `gpg -K` shows a `sec#` instead of just `sec` for your private key. That means the secret key is not really there. (See the also the presence of a dummy OpenPGP packet in the output of `gpg --export-secret-keys YOURMASTERKEYID | gpg --list-packets`.)

6. Change the passphrase protecting the subkeys: `gpg --edit-key YOURMASTERKEYID passwd`. This way if your everyday passphrase is compromised, the private master key will remain safe from someone with access to the backup: the private key material on the backup, including the private master key, are protected by the old passphrase.

Your computer is now ready for normal use.

When you need to use the master keys, mount the encrypted USB drive, and set the `GNUPGHOME` environment variable:

```
export GNUPGHOME=/media/something
gpg -K
```

or use the `--homedir` command-line argument:

```
gpg --homedir=/media/something -K
```

The latter command should now list your private key with `sec` and not `sec#`.

Then what?

At this point, you have a subkey, and you need to send it to the Debian keyserver, if your key is already in the Debian keyring, and the general keyserver network:

```
gpg --keyserver hkp://keyring.debian.org --send-key YOURMASTERKEYID  
gpg --keyserver hkp://pool.sks-keyservers.net --send-key YOURMASTERKEYID
```

The upload to the Debian key server only works if your master public key is in the DD or DM keyrings already: the Debian key server accepts updates to existing keys, but not new keys. New keys are added by the keyring maintainers manually. Updates to keys further need a manual update to be added to the actual keyring used by Debian's servers, which usually happens about once a month. (See <https://anonscm.debian.org/cgiit/keyring/keyring.git/log/> to see if your subkey has been added.)

So it may take 1 month, to make the new ?SubKeys get updated to Debian online servers.

(First time your key gets added to the Debian keyrings: manual, when you get accepted as DD or DM. After that, uploading subkeys to key server: automatic. Copying updates from key server to the Debian keyrings: manual, once a month.)

After this, you should be able to upload packages to Debian using the subkey, rather than the master key. The subkey will inherit the web-of-trust status of the master key pair.

Eek

If disaster strikes, and you need to revoke the subkey for whatever reason, do the following:

1. Mount the encrypted USB drive.
2. `export GNUPGHOME=/media/yourdrive`
3. `gpg --edit-key YOURMASTERKEYID`
4. At the `gpg>` prompt, list the keys (`list`), select the unwanted one (key 123), and generate a revocation certificate (`revkey`), then save.
5. Send the updated key to the keyservers as above.

Caveats

Multiple Subkeys per Machine vs. One Single Subkey for All Machines

One might be tempted to have one subkey per machine so that you only need to exchange the potentially compromised subkey of that machine. In case of a single subkey used on all machines, it needs to be exchanged on all machines in case of a compromising.

But this only works for signing subkeys. If you have multiple encryption subkeys, gpg is said to encrypt only for the most recent encryption subkey and not for all known and not revoked encryption subkeys.