



[Translation\(s\)](#): [English](#) - [Español](#) - [Italiano](#) - [Português \(Brasil\)](#)

Inhaltsverzeichnis

1. [Introduction](#)
2. [How to](#)
 1. [Step 1: Create a RSA keypair](#)
 2. [Step 2: Generate a revocation certificate](#)
 3. [Step 3: Make your public key public](#)
 4. [Step 4: Print your key](#)
 5. [Step 5: Hand out your key's fingerprint](#)
 6. [Step 6: Get your key digitally signed](#)
 7. [Step 7: Send your signed key to the server](#)
3. [Beyond Debian](#)
4. [See also](#)



Introduction

The intent of this page is to explain how you can create and sign an OpenPGP key.

Then, to get connected to the web of trust, go to the keysigning [coordination page](#).

How to

Tutorials explaining how to use GnuPG:




-  [Official GPG documentation](#)
-  [GnuPG for daily use \(a mini How-To...\)](#)

If you want your OpenPGP key signed by at least one (but ideally more than one) [Debian Developer](#), you have to follow the below steps.

Step 1: Create a RSA keypair

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

See also  [creating a keypair](#).

 Note that due to weaknesses found with the SHA1 hashing algorithm  [Debian wants stronger RSA keys that are at least 4096 bits and preferring SHA2](#) although  [Ed25519](#) is even better.

Also see  [OpenPGP Best Practices](#), [documentation about subkeys](#) and  [migration off of SHA-1 key](#).

Step 2: Generate a revocation certificate





Generate also a revocation certificate if you already have one!

```
gpg --gen-revoke [KEY_ID] > ~/.gnupg/revocation-[KEY_ID].cr
```

Step 3: Make your public key public

```
gpg --send-key 1A2B3C4D5E6F7G8H
```

Some public keyservers:

-  <http://keyring.debian.org> (only Debian Developers can upload or send key updates to this server)
-  <https://keys.openpgp.org/>
-  <http://pgp.surfnet.nl>
-  <http://pgp.mit.edu>

Step 4: Print your key

The printout of your fingerprint must contain the following information:

- Your first name
- Your last name
- Your e-mail addresses (the ones you use with the key)
- The encryption method and the ID of the key (e.g. 4096R/1A2B3C4D5E6F7G8H)
- The fingerprint itself

You can use this function :

```
gpg -v --fingerprint 1A2B3C4D5E6F7G8H
```

Usually, you make several printouts on a sheet of paper. It can for example be the size of a business card. You can also use the `gpg-key2ps` which is part of the [DebPkg: signing-party](#) package to create these printouts as:

```
gpg-key2ps -p a4 1A2B3C4D5E6F7G8H > out.ps
```

Alternatively, you can print in one column only to avoid printing issues (for extra wide keys):

```
gpg-key2ps -1 -p a4 1A2B3C4D5E6F7G8H > out.ps
```

If you go to a key signing party, you will have to send this information beforehand, and they will then print a list for each participant.


TIP: to read the `out.ps` file, you can use `evince`, `okular`, `ghostscript` or other ? PostScript viewer.

TIP2: Some websites also can be used to generate the PDF of OpenPGP fingerprint, such as: <http://openpgp.quelltextlich.at/slip.html> or <http://keysheet.net>

TIP3: if you would like a pdf instead of a postscript file you can pipe the `gpg2ps` output through `ghostscript` e.g.

```
gpg-key2ps -p a4 1A2B3C4D5E6F7G8H | gs -sDEVICE=pdfwrite -  
sOutputFile=out.pdf
```

Alternatively you can show your fingerprint(s) on your laptop screen like this:



```
LANG=C gpg --list-secret-keys --fingerprint --keyid-format long | grep
```

Step 5: Hand out your key's fingerprint

The people who will sign your key will need to see some form of government issued ID (passport or similar).

You have to give the printout to at least one [Debian Developer](#).

Read the official Debian  [keysigning](#) page.

A  [CAcert](#) member will need to see two IDs.

Step 6: Get your key digitally signed

The [Debian Developer](#) will

- retrieve your key from the server

```
gpg --recv-keys 00AA11BB22CC33DD
```

- verify that the information is correct (the fingerprint)

```
gpg --fingerprint 00AA11BB22CC33DD
```

- sign it.

```
gpg --sign-key 00AA11BB22CC33DD
```

- send it back to the key owner as an encrypted email (Do not send it directly to a server). Sending it encrypted is preferred as you can verify the person can decrypt the messages they receive.

```
gpg --armor --export 00AA11BB22CC33DD | gpg --encrypt -r  
00AA11BB22CC33DD --armor --output 00AA11BB22CC33DD-signedBy-  
1A2B3C4D5E6F7G8H.asc
```

Alternatively, the `caff` tool from the [DebPkg: signing-party](#) package automates all this process:

```
caff 00AA11BB22CC33DD
```

Step 7: Send your signed key to the server

Some time after having participated in a keysigning, you will perhaps receive your signed key as an e-mail attachment. Import the signatures:



```
gpg -d 1A2B3C4D5E6F7G8H-signedBy-00AA11BB22CC33DD.asc | gpg --import
```

Afterwards you will have to send your updated key to the server:

```
gpg --send-key 1A2B3C4D5E6F7G8H
```

Beyond Debian

Those interested in expanding the web of trust beyond Debian should visit:

-  [Biglumber](#).
-  <http://www.cacert.org/>

See also

- [/Coordination](#)
- [/DMOffers](#)
- [/Need](#)
- [/Offers](#)

[CategoryCommunity](#). [CategoryDeveloper](#)