[ALEX CABAL](https://alexcabal.com/)

* [ABOUT ME](https://alexcabal.com/about)



[CONTACT ME](https://alexcabal.com/contact)

I’m a software developer who runs [Scribophile, an online writing group for serious writers](https://www.scribophile.com/), [Writerfolio, an online writing portfolio service for freelancers](https://writerfolio.com/), and [Standard Ebooks, an open source project that produces liberated ebooks for the true book lover](https://standardebooks.org/).

CREATING THE PERFECT GPG KEYPAIR

March 13th, 2013

Before you start reading, keep in mind that I’m a layperson, just like you. This guide was created by synthesizing a lot of different information and tutorials available online, but I’m not a GPG developer, or even a cryptography professional. I’m just an enthusiast who thinks encryption is a critical part of modern society.

The encryption technology landscape can change rapidly, the PGP standard is bewilderingly complex, and GPG is a stunningly obtuse piece of software on the best of days. There are a lot of ways of creating a functional keypair, and *my* perfect way may not be *your* perfect way. Please make sure to do your own research to ensure the explanations and steps below make sense for you and your situation.

While I’ll do my best to keep this guide updated with what I personally consider to be the latest best practice, I can’t guarantee anything and I can’t answer any GPG-related questions.

There’s a lot of information online on how to create a new GPG keypair. Unfortunately a lot of it is old advice and recommends settings that today might be unsafe.

There also isn’t too much information on how to *protect* your keypair if you use a laptop that might get lost or stolen.

Protecting your keypair on a laptop is tricky.

On one hand, you need your private key with you to decrypt or sign messages.

On the other hand, if your laptop is stolen then you risk losing your entire online identity, perhaps going back years, because the thief would have access to your private key and could then impersonate you.

You’d think that today, where laptops and world travel are commonplace, there’d be a little more information on how to secure a private key you have to travel with. But I could only find one resource: the [Debian Wiki entry on subkeys](http://wiki.debian.org/subkeys). Fortunately it turns out this wiki page has exactly the solution we need.

SUBKEYS HELP PROTECT YOUR IDENTITY IN CASE OF PRIVATE KEY (LAPTOP) THEFT

If a thief gets ahold of the laptop with your private key on it, it’s pretty much game over. The thief can not only decrypt messages intended for you, they can also impersonate you by signing messages with your private key. Your only recourse would be to revoke your key, but that would mean losing years of signatures on that key and basically creating a massive inconvenience for yourself.

Part of the answer to this problem is the concept of subkeys. Subkeys can’t prevent a thief from decrypting messages intended for your private key. But they *can* help mitigate the damage to your identity should your key be lost or stolen.

The concept behind this technique is as follows:

1. Create a regular GPG keypair. By default GPG creates one *signing* subkey (your identity) and one *encryption* subkey (how you receive messages intended for you).
2. Use GPG to add an *additional* signing subkey to your keypair. This new subkey is linked to the first signing key. Now we have three subkeys.
3. This keypair is your **master keypair**. Store it in a protected place like your house or a safe-deposit box. Your master keypair is the one whose loss would be truly catastrophic.
4. Copy your master keypair to your laptop. Then use GPG to *remove the original signing subkey*, leaving only the new signing subkey and the encryption subkey. This transforms your master keypair into your **laptop keypair**.

Your laptop keypair is what you’ll use for day-to-day GPG usage.

What’s the benefit to this setup? Since your master keypair isn’t stored on your traveling laptop, that means you can revoke the subkeys on your laptop should your laptop be stolen. Since you’re not revoking the *original* subkey you created in the master keypair—remember, we removed it from our laptop’s keypair—that means you don’t have to create a new keypair and go through the hassle of getting people to sign it again. You’d still have to revoke the stolen subkey, and the thief could still use the encryption subkey to decrypt any messages you’ve already received, but at least the damage done won’t be as catastrophic.

CREATING THE PERFECT GPG KEYPAIR, STEP-BY-STEP

I’m going to lead you through the steps to create a new keypair using this subkey method. To do this we’ll be using GPG 1.4.11, which is the version currently distributed with Ubuntu 12.04 LTS.

GPG can be pretty noisy in its output. Some of the output below might be cut off due to the fixed-width layout of this blog; what’s cut off isn’t really important, but you can see it by highlighting it with your mouse.

CREATING YOUR INITIAL KEYPAIR

Use the gpg --full-generate-key command to create a new GPG keypair.

Generally you should set your key to expire within a year or less. You can always change the expiration date later, but if you upload a key without an expiration date to a keyserver, and then your key is lost or compromised, the bad key will remain out there forever. Giving it an expiration date is a safeguard against that. For our example key, we’ll set it to not expire to simplify things a little.

When you create your new keypair, use the highest possible values for key length. As computers get more powerful and storage gets cheaper, it’s conceivable that a nasty person could archive a message that’s unbreakable today, then in the future break it using a more powerful computer. Using the highest possible value for key length helps protect you from that scenario. *Don’t use GPG’s default of 2048!*

gpg --gen-key

gpg (GnuPG) 1.4.11; Copyright (C) 2010 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) 4096

Requested keysize is 4096 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) 0

Key does not expire at all

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 E-mail Address in this form:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Bilbo Baggins

E-mail address: bilbo@shire.org

Comment:

You selected this USER-ID:

"Bilbo Baggins <bilbo@shire.org>"

Change (N)ame, (C)omment, (E)-mail or (O)kay/(Q)uit? o

You need a Passphrase to protect your secret key.

<type your passphrase>

gpg: key 488BA441 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

pub 4096R/488BA441 2013-03-13

Key fingerprint = B878 1FB6 B187 B94C 3E52 2AFA EB1D B79A 488B A441

uid Bilbo Baggins <bilbo@shire.org>

sub 4096R/69B0EA85 2013-03-13

When prompted for a passphrase, make sure to pick a long and unique one. If your key gets stolen, this passphrase is the only thing protecting it!

ADDING A PICTURE

You might want to add a picture of yourself for completeness. Since the picture is stored in your public key and your public key gets distributed in a lot of places, including sometimes email, it’s best to use a small image to save space.

Use the gpg --edit-key command. At the gpg> prompt, enter the command addphoto and give GPG the path of the picture you’d like to use. When you’re done, use save at the final gpg> prompt to save your changes:

gpg --edit-key bilbo@shire.org

gpg (GnuPG) 1.4.11; Copyright (C) 2010 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.

Secret key is available.

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

gpg> addphoto

Pick an image to use for your photo ID. The image must be a JPEG file.

Remember that the image is stored within your public key. If you use a

very large picture, your key will become very large as well!

Keeping the image close to 240x288 is a good size to use.

Enter JPEG filename for photo ID: /home/bilbo/me.jpg

Is this photo correct (y/N/q)? y

You need a passphrase to unlock the secret key for

user: "Bilbo Baggins <bilbo@shire.org>"

4096-bit RSA key, ID 488BA441, created 2013-03-13

<type your passphrase>

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

[ unknown] (2) [jpeg image of size 5324]

gpg> save

STRENGTHENING HASH PREFERENCES

Now we set our key to prefer stronger hashes. Use the gpg --edit-key command. At the gpg> prompt, enter the command setpref SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed (note that this will probably be cut off in the example below; highlight it with your mouse to see it), then save.

gpg --edit-key bilbo@shire.org

gpg (GnuPG) 1.4.11; Copyright (C) 2010 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.

Secret key is available.

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

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

[ultimate] (2) [jpeg image of size 5324]

gpg> setpref SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed

Set preference list to:

Cypher: AES256, AES192, AES, CAST5, 3DES

Digest: SHA512, SHA384, SHA256, SHA224, SHA1

Compression: ZLIB, BZIP2, ZIP, Uncompressed

Features: MDC, Keyserver no-modify

Really update the preferences? (y/N) y

You need a passphrase to unlock the secret key for

user: "Bilbo Baggins <bilbo@shire.org>"

4096-bit RSA key, ID 488BA441, created 2013-03-13

<type your passphrase>

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

[ultimate] (2) [jpeg image of size 5324]

gpg> save

ADDING A NEW SIGNING SUBKEY

Now for the special sauce: let’s add our new signing subkey.

Use the gpg --edit-key command. At the gpg> prompt, enter the command addkey. Select RSA (sign only) and 4096 for the keysize. Don’t forget to save at the last gpg> prompt:

gpg --edit-key bilbo@shire.org

gpg (GnuPG) 1.4.11; Copyright (C) 2010 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.

Secret key is available.

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

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

[ultimate] (2) [jpeg image of size 5324]

gpg> addkey

Key is protected.

You need a passphrase to unlock the secret key for

user: "Bilbo Baggins <bilbo@shire.org>"

4096-bit RSA key, ID 488BA441, created 2013-03-13

<type your passphrase>

Please select what kind of key you want:

(3) DSA (sign only)

(4) RSA (sign only)

(5) Elgamal (encrypt only)

(6) RSA (encrypt only)

Your selection? 4

RSA keys may be between 1024 and 4096 bits long.

What keysize do you want? (2048) 4096

Requested keysize is 4096 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) 0

Key does not expire at all

Is this correct? (y/N) y

Really create? (y/N) y

pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC

trust: ultimate validity: ultimate

sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E

sub 4096R/C24C2CDA created: 2013-03-13 expires: never usage: S

[ultimate] (1). Bilbo Baggins <bilbo@shire.org>

[ultimate] (2) [jpeg image of size 5324]

gpg> save

CREATING A REVOCATION CERTIFICATE

Now we generate a revocation certificate file. If your *master* keypair gets lost or stolen, this certificate file is the only way you’ll be able to tell people to ignore the stolen key. *This is important, don’t skip this step!*

gpg --output \<bilbo@shire.org\>.gpg-revocation-certificate --gen-revoke bilbo@shire.org

Store the revocation certificate file in a different place than your master keypair (which we’ll export in a later step). You’ll use it to revoke your master keypair should you lose access to it. If you only lose access to your laptop keypair, then you’ll revoke those subkeys using the master keypair, not this revocation certificate.

EXPORTING THE FINAL PRODUCT

Now that your keypair has been created, let’s export it so that we can back it up:

gpg --export-secret-keys --armor bilbo@shire.org > \<bilbo@shire.org\>.private.gpg-keygpg --export --armor bilbo@shire.org > \<bilbo@shire.org\>.public.gpg-key

This will create two files: your public key and your private key. Protect these two files, along with the revocation certificate file, as best as you can—*don’t* keep them on your laptop, keep them in your house or in a safe-deposit box. These three files are your **master keypair**.

TRANSFORMING YOUR MASTER KEYPAIR INTO YOUR LAPTOP KEYPAIR

Now we have our master keypair in our keyring, along with three files representing the master keypair plus the keypair’s revocation certificate. To transform our master keypair into our **laptop keypair**, we have to remove the original signing subkey from the master keypair in our keyring.

GPG doesn’t make this easy, but here we go:

1. Export all of the subkeys from our new keypair to a file. We first create a RAM-based ramfs temporary folder to prevent our keys from being written to permanent storage. we use ramfs instead of tmpfs/ or /dev/shm/ because ramfs doesn’t write to swap.

mkdir /tmp/gpgsudo mount -t ramfs -o size=1M ramfs /tmp/gpgsudo chown $(logname):$(logname) /tmp/gpggpg --export-secret-subkeys bilbo@shire.org > /tmp/gpg/subkeys

1. Delete the original signing subkey from the keypair in our keyring:

gpg --delete-secret-key bilbo@shire.org

1. Re-import the keys we exported and clean up our temporary file:

gpg --import /tmp/gpg/subkeyssudo umount /tmp/gpgrmdir /tmp/gpg

That’s all! You can verify it worked by running:

gpg --list-secret-keys

/home/bilbo/.gnupg/secring.gpg

-----------------------------

sec# 4096R/488BA441 2013-03-13

uid Bilbo Baggins <bilbo@shire.org>

ssb 4096R/69B0EA85 2013-03-13

ssb 4096R/C24C2CDA 2013-03-13

See how the third line begins with “sec#”, not “sec”? The pound sign means the signing subkey is not in the keypair located in the keyring.

You’re all done!

WHAT HAVE WE JUST ACCOMPLISHED?

If you followed all the steps in this guide, you:

1. Created a new keypair using the strongest possible settings.
2. Added a new signing subkey to that keypair.
3. Exported the complete keypair to two files plus a revocation certificate, all three of which you’ve stored up in a safe place, *not* on your laptop. This is your **master keypair**.
4. Removed the original signing subkey from the master keypair in your laptop’s keyring, thus transforming your master keypair into your **laptop keypair**. Your life will now be a little easier should your laptop get lost or stolen.

USING YOUR NEW LAPTOP KEYPAIR

You can now use your keypair to encrypt, decrypt, and sign files and messages.

To sign someone else’s key or to create or revoke a subkey on this keypair, you need to use the master keypair that you keep safe—the one that’s not on your laptop.

You should distribute your public key to a keyserver. There are plenty of tutorials online on how to do that.

IN CASE OF EMERGENCY

Should the worst happen and your laptop with your special keypair gets lost or stolen (or your special keypair is otherwise compromised), we need to revoke the subkeys on that keypair.

1. Unlock your safe-deposit box and get your master keypair out.
2. Boot a live USB of Ubuntu or your distro of choice. Then, import your master keypair into the live USB’s keyring:

gpg --import /path/to/\<bilbo@shire.org\>.public.gpg-key /path/to/\<bilbo@shire.org\>.private.gpg-key

1. Now use gpg --edit-key to interactively revoke your subkeys:
2. gpg --edit-key bilbo@shire.org
3. gpg (GnuPG) 1.4.11; Copyright (C) 2010 Free Software Foundation, Inc.
4. This is free software: you are free to change and redistribute it.
5. There is NO WARRANTY, to the extent permitted by law.
6. Secret key is available.
7. pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC
8. trust: ultimate validity: ultimate
9. sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E
10. sub 4096R/C24C2CDA created: 2013-03-13 expires: never usage: S
11. [ultimate] (1). Bilbo Baggins <bilbo@shire.org>
12. [ultimate] (2) [jpeg image of size 5324]
13. gpg> key 1
14. pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC
15. trust: ultimate validity: ultimate
16. sub\* 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E
17. sub 4096R/C24C2CDA created: 2013-03-13 expires: never usage: S
18. [ultimate] (1). Bilbo Baggins <bilbo@shire.org>
19. [ultimate] (2) [jpeg image of size 5324]
20. gpg> key 2
21. pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC
22. trust: ultimate validity: ultimate
23. sub\* 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E
24. sub\* 4096R/C24C2CDA created: 2013-03-13 expires: never usage: S
25. [ultimate] (1). Bilbo Baggins <bilbo@shire.org>
26. [ultimate] (2) [jpeg image of size 5324]
27. gpg> revkey
28. Do you really want to revoke the selected subkeys? (y/N) y
29. Please select the reason for the revocation:
30. 0 = No reason specified
31. 1 = Key has been compromised
32. 2 = Key is superseded
33. 3 = Key is no longer used
34. Q = Cancel
35. Your decision? 1
36. Enter an optional description; end it with an empty line:
37. >
38. Reason for revocation: Key has been compromised
39. (No description given)
40. Is this okay? (y/N) y
41. You need a passphrase to unlock the secret key for
42. user: "Bilbo Baggins <bilbo@shire.org>"
43. 4096-bit RSA key, ID 488BA441, created 2013-03-13
44. <type your passphrase>
45. You need a passphrase to unlock the secret key for
46. user: "Bilbo Baggins <bilbo@shire.org>"
47. 4096-bit RSA key, ID 488BA441, created 2013-03-13
48. <type your passphrase>
49. pub 4096R/488BA441 created: 2013-03-13 expires: never usage: SC
50. trust: ultimate validity: ultimate
51. This key was revoked on 2013-03-13 by RSA key 488BA441 Bilbo Baggins <bilbo@shire.org>
52. sub 4096R/69B0EA85 created: 2013-03-13 expires: never usage: E
53. This key was revoked on 2013-03-13 by RSA key 488BA441 Bilbo Baggins <bilbo@shire.org>
54. sub 4096R/C24C2CDA created: 2013-03-13 expires: never usage: S
55. [ultimate] (1). Bilbo Baggins <bilbo@shire.org>
56. [ultimate] (2) [jpeg image of size 5324]

gpg> save

1. Now that your subkey has been revoked, you have to tell the world about it by distributing your key to a keyserver.

FURTHER READING

* [The GNU Privacy Handbook](http://www.gnupg.org/gph/en/manual.html), a very detailed explanation of how public-key cryptography works and how to use GPG.
* [Subkeys](http://wiki.debian.org/subkeys) at the Debian Wiki, an explanation of why using subkeys is a good idea and a step-by-step guide to setting them up.
* [Creating GPG Keys](http://fedoraproject.org/wiki/Creating_GPG_Keys) at the Fedora project, a step-by-step guide to creating a new GPG keypair.
* [GPG How-to](https://help.ubuntu.com/community/GnuPrivacyGuardHowto) at the Ubuntu Community Help Wiki.

1. ****[**Vadim**](http://vadimszenins.blogspot.com/)**June 12th, 2013**

Thanks for info about “sec#” (the signing subkey is not in the keypair located in keyring).

1. ****[**Byron Sanchez**](http://hackbytes.com/)**June 29th, 2013**

Great writeup. I managed to get through the whole guide and setup my local keyring. Now I just have to find a secure location for the master keypair.

1. **Boris Ent  July 9th, 2013**

How do I solve this problem?  
gpg —import “..\DEV\SecretKey.gpg”  
gpg: no valid OpenPGP data found.  
gpg: Total number processed: 0

The same happens with:  
gpg —import “..\DEV\SecretSubKey.gpg”  
gpg —import “..\DEV\PublicKey.gpg”

Followed these instructions “Creating the perfect GPG keypair — Alex Cabal”.  
<https://alexcabal.com/creating-the-perfect-gpg-keypair/>

Everything worked as described.

Encryption. Decryption. Signing. Perfect.

Then I wanted to add a photo:  
gpg: secret key parts are not available  
gpg: signing failed: general error

Oops. I needed to import my keys.

Import worked! I added my photo. Done.

That was 2 months ago.

Now, I want to change my photo to colour.

First, I deleted the old photo…

No password. No “secret key parts” warning.

But, the image is gone. Good.

So, I run:  
gpg —import “..\DEV\SecretKey.gpg”

And, I’m rewarded with:  
gpg: no valid OpenPGP data found.  
gpg: Total number processed: 0

I’m running gpg (GnuPG) 1.4.13 on Win8, Win7 and Vista.  
\_\_\_\_  
¯¯¯¯

1. ****[**Alex Cabal**](https://alexcabal.com/)**July 9th, 2013**

That warning suggests that the SecretKey.gpg file does not contain a valid GPG secret key. Have you tried opening the file in a text editor to confirm it has the entire key?

1. **Boris Ent  July 12th, 2013**

Yes.

I did back up my exported keys.

There isn’t anything missing from the keys I tried to import.

I think the problem is in deleting the image without the need of the secret key present.

Running the scenario on a new keyring in the “correct order” of:  
• import the secret key  
• delete the image  
• add a new image  
• export the secret key  
• etc.  
didn’t produce this error.

When I find the time I’ll experiment further.

1. **Facundo  July 16th, 2013**

Great, thank you!

1. **Alex  August 15th, 2013**

Thanks for the great writeup. I have a practical question regarding “To sign someone else’s key or to create or revoke a subkey on this keypair, you need to use the master keypair that you keep safe—the one that’s not on your laptop.”

How exactly would I do that to sign Fred’s key? Please note that the “safe” environment that stores my master keypair is not networked.

My guess:  
1. Get Fred’s public key on a USB key  
2. Get my master keypair from where it’s stored  
3. Boot a trusted environment (e.g. TAILS) with no networking  
4. Import my master keypair to the trusted environment  
5. Import Fred’s public key to the trusted environment (careful that Fred’s key is not trojaned etc)  
6. Sign Fred’s public key  
7. Now what? How do I make Fred’s signed key available to the world (keyservers) and to my “laptop” environment for day-to-day use? Do I just export Fred’s newly-signed key to a USB key and then import it in my “laptop” environment & upload to keyservers?

1. ****[**Alex Cabal**](https://alexcabal.com/)**August 16th, 2013**

Yes Alex that’s correct. Once you’ve signed the other person’s public key in a trusted environment, you can send that signature back to them via email at your convenience. See here for more details (specifically the “After the Party” section): <http://www.phillylinux.org/keys/terminal.html>

1. **Alex  August 16th, 2013**

Thanks for confirmation — one more question I’ve been mulling over:

You state that “Your life will now be a little easier should your laptop get lost or stolen.” — but it’s unclear to me what the practical difference between the following two scenarios is:  
(a) revoking my signing and encryption subkey, therefore having to publish new ones to the world and  
(b) revoking all of my keys

Even if I had heavily invested in the Web Of Trust and I had a public key signed by hundreds of people, wouldn’t that be lost with this setup?

Not sure what the practical advantage of keeping that offline key is… could you elaborate?

1. ****[**Alex Cabal**](https://alexcabal.com/)**August 17th, 2013**

From the Debian Subkeys wiki: “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.”

1. **Stefan  September 7th, 2013**

Great guide!

One question: why use gpg 1.4? Gpg2 is intended for desktop use for quite a while now, it supersedes the gpg 1.4 series. Gpg 1.4 is still maintained but intended for servers and embedded systems.

1. ****[**Alex Cabal**](https://alexcabal.com/)**September 8th, 2013**

gpg 1.4 is the version packaged in Ubuntu 12.04 LTS, so it’s the version most likely to be in use for the next few years (for Ubuntu users at least).

1. **Stefan  September 9th, 2013**

I see.

I just completed the guide using gpg2 (2.0.19 to be exact), works fine.  
Also referenced some other sources, but yours was the main one I used. I’m probably gonna publish a blog about it as well, I’ll reference you of course 🙂  
(unless you don’t want to, let me know here, I’ll check back, or email me)

1. ****[**Alex Cabal**](https://alexcabal.com/)**September 9th, 2013**

Sure, no need to ask!

1. ****[**Scott Rubin**](http://frontrowcrew.com/)**September 14th, 2013**

Can the same concept of subkeys be used for SSH? Is there a way to have one SSH master key, put subkeys on all of my SSH client computers, then just put my public key on the server I am SSH-ing to?

1. **zokier  September 14th, 2013**

@Scott Rubin

SSH supports certificates, which can be signed by a CA. It kinda creates similar master/sub structure.

Here is a blog post (not mine) explaining how to create such setup:

<https://blog.habets.se/2011/07/OpenSSH-certificates>

1. **Stanislav Sedov  September 15th, 2013**

gpg-agent can be used in-place of ssh-agent (it has a compatibility mode). So you can create the authentication subkey in a similar manner and use it with ssh. gpgkey2ssh(1) tool can convert the specified GPG key to a format suitable for authorized\_keys.

1. **Klaas  September 15th, 2013**

Please do not advise (or give an example) people to not set a key expiration date, that’s bad practice. Instead, set a reasonable key expiration length in addition to generating the revocation certificate. This protects against loss of the certificate/master key/etc. It’s easy to extend the expiration date, even after it elapsed.

1. **clutton  September 17th, 2013**

Ok, nice topic, but you’d better encrypt your laptop hard/solid drive…

1. **Lars  September 20th, 2013**

What’s the point of creating a revocation certificate and then storing it in the same place as the private key? If you need to revoke the key, why not just get the private key out and revoke it?

1. ****[**Alex Cabal**](https://alexcabal.com/)**September 20th, 2013**

Lars, good point, I’ll revise that line. The revocation certificate should be used to revoke the entire master keypair should you lose it, not the individual subkeys of the master or laptop keypair.

1. **John  September 24th, 2013**

This is an excellent guide.

I do have one question: if one wanted to add new UIDs to the key would you need to generate a new master key pair (revocation, public, and private certificates) to replace the backup, or a new laptop key pair? The reason I ask is that I have done just that, and obviously my “laptop” secret key doesn’t have the new UIDs within it.

Ta

1. ****[**Alex Cabal**](https://alexcabal.com/)**September 24th, 2013**

If you added a new UID to your master keypair, then I imagine what you’d have to do is re-generate the laptop keypair using the above instructions. That would probably be the most failsafe way of doing it.

1. ****[**Nikolay Kolev**](https://nikolay.com/)**October 14th, 2013**

SHA1 and 3DES cannot be removed though, so, setpref sneaks them in.

1. **Toolow  October 14th, 2013**

“you can revoke the compromised subkey and replace it with a new subkey without having to rebuild your reputation” => I don’t understand how. My reputation is build by letting my friends signing the keys I published (the subkeys).  
If I have to revoke those subkeys, how can my reputation still be ok ?

1. **Miro  October 15th, 2013**

I’ve noticed that the public key generated before keyring transformation is different then afterwards.

The content of the key is same … inspecting by (gpg —with-fingerprint ), but the byte content is different. Inspecting by (pgpdump ) there is one clause “Old: Signature Packet” on top there.

Why this difference?  
And what is the public key to be distributed? send to keyserver?

1. ****[**Alex Cabal**](https://alexcabal.com/)**October 15th, 2013**

@Toolow: You only have others sign your master keypair. Thus, if your laptop keypair is lost, you still maintain the original signatures on your master keypair, and you can use it to issue a new laptop keypair. That means that likewise you must use your master keypair to sign the keys of others.

Now if you lost your master keypair, then you’re right—you’re in big trouble!

1. ****[**Alex Cabal**](https://alexcabal.com/)**October 15th, 2013**

@Miro: I don’t know. That’s a question for someone with more in-depth PGP knowledge than me 🙂

1. ****[**Toolow**](http://twolaw@free.fr/)**October 16th, 2013**

Ok, thanks.  
So what i finally release on keyservers are :  
— the master public key (then signed by my friends)  
— the uid (signed by the private master key)  
— the laptop public signing subkey (signed by the private master key)  
— the laptop public encription subkey (signed by the private master key)

1. ****[**Alex Cabal**](https://alexcabal.com/)**October 16th, 2013**

@Toolow: All you have to do is gpg —send-keys to have gpg sort out what keys need to be sent where.

1. **PleaseHelpMeMystie72  November 20th, 2013**

Hi. Thank YOU, Alex, for sharing your knowledge. I am a mere mortal, non-geek and am doing the ONLY REASONABLE thing for EVERYONE to do: LEARN to encrypt EVERYTHING — even millions of pictures of adorable kitty-cats. Please allow me to take this moment to ask ALL of the GOOD geeks EVERYWHERE to — please — make it EASY for everyone to teach their grandparents to [both] ENCRYPT everything AND to EASILY use GNUGPG. This time next year? Okay A.S.A.P. This is important. You ALL KNOW this. Thanks.

Now, Alex, could you please help me? I would be so very grateful. Because I followed this tutorial “to-the-‘t'” I now find myself in a bit of a pickle. I now think that we missed an important step — that you probably assumed that anyone who would follow your guidance would know to do. But, alas, I am a non-geek.

I was so impressed with myself, after adding a picture. Then I was even more impressed with You, after You had me strengthen the hash preferences. And I felt very satisfied from choosing to make new signing sub-key. And of course, I made a revocation key — and properly saved each key, per your instructions. I also proudly use full-disk encryption — and have copied these keys, that I made per your instructions, onto several small and fully encrypted usb-sticks, which I intend to distribute to my beautiful friends and family, to keep my keys safely and redundantly, off-site.

Now — uh-oh! And oh-crap. I just read a couple pages of advice, on different sites — pointing out that it must be the ORIGINAL MASTER signing key, that signs the keys of others — or revoking the second signing key would revoke your web-of-trust.

[Now I have just ALSO realized, that even if I never have a theft or damage or compromise, that this very old hard drive will eventually fail — and I will need to transfer my set-up or duplicate it somehow, eventually.]

A couple of sites say to copy the .gnupg directory and paste it somewhere or make a tar[?] backup. But I just securely deleted all of my [2] backups — of my home-files — and replaced them with copies of my new home files — right after doing your tutorial — so that I would have these new and impressive changes INCLUDED in my backed-up important files — changes made per your tutorial. So I NOW HAVE — NO .gnupg directory — that contains the Original Master — and with keys extracted — per your tutorial — I now have multiple copies of the keys — each produced separately.

Now, mounting one of the usb-sticks, in order to sign someone else’s key, with the per-other-advice, like this:

export GNUPGHOME=/media/MyMediaName/My FolderName/AnotherFolderName

…which is where my key files are sitting — and then this:

gpg -K

I get gpg-ish blah-styled-output along these lines: lock not made: link() failed: Operation not permited gpg: can’t lock ‘[blah-my media — media/MyMediaName/My FolderName/AnotherFolderName] /secring.gpg’ — blah — DGB: oops — blah — is not locked blah keyblock resource — general blah-error… fatal: can’t acquire lock — giving up — secmem usage — blah-blocks of blah-pool…

And previously, I was so happy when the “gpg -K” command gave me the sec# — not the sec…

So I assume that gpg is looking for a .gnupg directory on my usb — which I now do NOT now have — because it’s not in your tutorial to put it there — previous to your laptop-keypair step — and I assume now — too late — that you assume that anyone attempting to follow a gpg tutorial would know to do this. But, sadly — no — not me. I am a mere idiot — trying, valiantly, to reach my genius mentality. So I assume that I must now do part of your tutorial — in reverse — to recover my previous .gnupg directory files — and then do your laptop-keypair step — again — except that I cannot figure out how — after 21 hours of looking for the command-line answers to this dilemma. And now I am wondering why you have us backup each key separately — when they apparently must all be together in the securing-file…

Can you Please help me? I don’t know any commands except what I have encountered in beginner-ish tutorials. Don’t I need to now re-create the .gnupg directory — by putting my Original Master back in? and then copy this directory into each usb-stick? and then re-do your laptop-keypair step? and then mount a usb with my keys and with my Masterkey included in the .gnupg directory — for Masterkey-signing — and set the GNUPGHOME environment variable [?] like this:

export GNUPGHOME=/media/TheNameOfMyMedia/.gnupg -> ? yes?

…Is this [above] EXACTLY the command? with NO “/” trailing-slash at the end of the command?

…And might you PLEASE HELP me — and hold my hand during it? please? I promise to teach others after setting A GOOD-ENCRYPTION-EXAMPLE for them.

1. ****[**Alex Cabal**](https://alexcabal.com/)**November 20th, 2013**

Your situation sounds a little too complex to sort out in blog comments. If you’ve deleted files you need or otherwise messed around with directories you shouldn’t have, I suggest reinstalling GPG and simply creating a new keypair.

Keys in a keypair can be exported to different files, as per the tutorial. They don’t have to be in a single file. (And I don’t think they can be outside of GPG internal machinery, which you shouldn’t really mess with.)

1. **PleaseHelpMeMystie72  November 21st, 2013**

Hi — Thank you! The reason that my situation is so perfectly-simple — is that YOUR TUTORIAL WAS SIMPLY-PERFECT — and — it went just as PERFECTLY as I could have possibly imagined. Please note my Main Question [implied and also blatant in the main-text of Paragraph-5 of my first post:] HOW does one transfer this/your Beautiful setup — to a second computer? ? ? This is the main crisis that Everyone who lives long enough is GUARANTEED to face — as ALL hard drives are guaranteed to fail eventually…

I think this should be super-simple — as I have only done ONLY and EXACTLY as you have instructed — and never messed with any directory — and — never deleted any files, in this process. Could you re-read my plea for help ’cause I think you must have added/assumed something that I wouldn’t even know how to do… And I think you missed Paragraph-3 — the point of which — is that following your tutorial went PERFECTLY. 100%, Perfectly. The only thing that I “deleted” was NOT ON my computer — it was a backup-usb’s [Spare/Backup] copy of my personal home files — yes — from an external usb stick — really — nothing to do with this/your process, except that it contained a copy of the .gnupg directory as it was — before I ever found your tutorial. Then I did exactly as you said to — throughout your tutorial — and then I just replaced that tiny backup [on-a-tiny-usb-stick] with what happened TO my linuxbox home files — as a result of PERFECTLY following your very easy-to-follow [compared-with-some-others] tutorial — which [very pleasantly] turned out to incur — for EVERY STEP of the way — the EXACT RESULTS that you said it would occur. Perfect, so far…

Note in Paragragh-6 that it is BACKUPS that were deleted… My main system has ONLY the PERFECTLY-CREATED files that are EXACTLY what you guided me to create.

Your tutorial does not include how to use my Masterkey to sign someone else’s key — I cannot find a tutorial on that — and yet your tutorial — that turned out PERFECTLY — and exactly as you said it should — for every step of the way — leaves me — Perfectly — exactly as you said it Would — with no Masterkey in my “laptop” — which now has a “Laptop-Keypair” — which numerous other tutorials point out — that I cannot use for signing keys in a way that will preserve the web-of-trust if disaster strikes the “Laptop-Keypair.” And — also — your tutorial does not help me set up my other computers with THE SAME perfectly-beautiful “Laptop-Keypair,” so that I can use my second computer for gpg, as well. The most likely — and the ONLY GUARANTEED CRISIS — that everyone who lives long enough will face — is a hard drive dying.

So my main two question [and — please — help me] are:

1. How do I set up my second computer with these Perfectly-Beautiful keys that you have helped me to create?

2. Also — specifically — HOW does Anyone — who follows your tutorial — and gets YOUR PERFECTLY-SPECIFIED RESULTS — fulfill the very-important function of mounting the Masterkey — on a usb-stick — in order to use the Masterkey — to sign the keys of others with?

1. **PleaseHelpMeMystie72  November 21st, 2013**

P.S. Please forgive my wordiness. I’m aware — it’s a problem and I’m working on it. Please help with my [above] dilemmas resulting from the first tutorial that has ever gone perfectly for me. I feel in — almost — over my head, but I will set a good example and then teach others. Thanks.

1. **Jim Hickstein  November 22nd, 2013**

This is great! At first, though, signatures made with my new signing-only sub-key were not trusted, even though the master key (still in my keyring) was set to “ultimate”. gpg —edit-key showed [unknown] next to my UID. Somehow I managed to clear that up, though it’s very obscure.

1. ****[**d33tah**](http://deetah.jogger.pl/)**December 1st, 2013**

Awesome, thanks! The only thing I’d change is replacing the “rm” command with “shred”, otherwise the private key remains on the hard drive.

I’ve got a question — I already have a GPG key that I uploaded to the MIT keyserver. Knowing that its public part is, well, public, can I still use it as the master key and generate a subkey? Or would it leave a nasty confusing trace in the keyserver and it’s better to revoke it and generate two new pairs?

BTW, @Klaas, thanks for the comment about expiration — can I change the expiration date if I already uploaded the key to keyservers?

1. ****[**d33tah**](http://deetah.jogger.pl/)**December 1st, 2013**

Also, how risky is it to temporarily import the private key onto an online machine? Something tells I’d be too lazy to use an offline computer to sign others’ keys/generate new subkeys (and I don’t really have a spare computer).

1. ****[**Alex Cabal**](https://alexcabal.com/)**December 1st, 2013**

@d33tah: When uploading to a keyserver only your public key gets sent, so you should be OK to mess around with the corresponding private key as much as you like.

If you’re serious (Schneier , Greenwald, or Snowden serious) about security, you should never move your private key to an online machine at any time. If you’re lazy, you can certainly take the risk—the chances of something peeking at it are low to zero, but you never know.

1. **3  January 17th, 2014**

Is it possible to map your subkey-procedure to two ordinary/regular gpg-keys (keypairs)? (Create one ordinary gpg key named secret and one named public. The secret one can sign the public one and the public one is used every day. The secret one plays the role of your subkey.)

What is the benefit of your subkey-procedure instead of doing this by two or more ordinary/regular gpg-keys?

1. ****[**Alex Cabal**](https://alexcabal.com/)**January 19th, 2014**

@3 Sure, I suppose that would work, but then you’d have to manage and sign two entirely different keypairs. I suppose it’s more of a semantic difference than a practical one.

1. **das-g  February 12th, 2014**

@PleaseHelpMeMystie72: What did you do with the files from <https://alexcabal.com/creating-the-perfect-gpg-keypair/#exporting-the-final-product> ? The first one (\*.private.gpg-key) is the “ORIGINAL MASTER signing key” you are looking for. Before signing others’ keys, just (re-)import that file.

1. **Anonymous  February 17th, 2014**

Very nice tutorial. I am not an expert but I also found the following on the debian wiki.

They state <https://wiki.debian.org/subkeys>

“Export the subkeys:

gpg —export-secret-subkeys SUBKEYID1! .. SUBKEYIDn! > subkeys

(NOTE: The exclamation marks ! are significant)”

In your tutorial you don’t have the exclamation marks. Is that important or not.

Thanks

1. ****[**Alex Cabal**](https://alexcabal.com/)**February 18th, 2014**

@anonymous: I *think* that the method you highlighted in the Debian wiki is asking GPG to export a list of individual subkeys you specify one by one. To export all the subkeys of a certain master keypair, you would have to individually list each subkey by ID.

The method used in this post exports all of the subkeys of a master keypair at once, without having to specify them one by one.

The final product should be the same. You might want to use the method used by the Debian wiki if you’re doing this process on a more complex key you created with a different tutorial.

Please correct me if I’m wrong!

1. **Anson Ng  February 22nd, 2014**

After I deleted the key and re-imported the subkeys, should I still use the master keyid for default-key in the gpg.conf or should I change it to the keyid of the signing subkey? For example:

sec# 4096R/0x1234567890ABCDEF 2014-02-21 [expires: 2016-02-29]  
Key fingerprint = AB9F 87E6 03AC 8BC3 DA13 B71D A8E3 FADC CCAD 23A4  
uid Test Key  
ssb 4096R/0x234567890ABCDEF1 2014-02-21  
ssb 4096R/0x34567890ABCDEF12 2014-02-21

In gpg.conf, which one should I set:  
default-key 0x1234567890ABCDEF  
or  
default-key 0x34567890ABCDEF12

Thanks.

1. ****[**Alex Cabal**](https://alexcabal.com/)**February 22nd, 2014**

@Anson Ng You can use your laptop keypair just like you would a regular keypair, so I imagine the settings would remain the same.

1. **Anonymous  February 23rd, 2014**

Thank you for the very well laid out tutorial it was very straight forward and everything worked as outlined. The only thing I am confused about is, even thought when I edit my key and do showpref I get

Cipher: AES256, AES192, AES, CAST5, 3DES  
Digest: SHA512, SHA384, SHA256, SHA224, SHA1  
Compression: ZLIB, BZIP2, ZIP, Uncompressed  
Features: MDC, Keyserver no-modify

So I am assuming I set the preferences correctly, but when I sign an email it still shows SHA1. The only way I was able to get that to change was by editing the gpg.conf file ad add

personal-digest-preferences SHA256  
cert-digest-algo SHA256

Is that the way it is supposed to be or did I miss something. I thought that gpg should grab the preferences from the key.?

Also is there a way to verify that encrypted files and/or signed files are using the preferred preferences?

Thanks

1. **Martin  March 6th, 2014**

Hi there, I have the following problem after going through all these steps.  
I tried to import the secret MasterKey with  
gpg —import name.priv.gpg-key name.pub.gpg-key

I get the following message (I have to translate to english):  
>gpg —import name.private.gpg-key name.public.gpg-key  
gpg: Key 23B16B91: Is already in the secret keylist  
gpg: Key 23B16B91: $NAME not changed  
gpg: Number of changed Keys : 2  
gpg: unchanged : 1  
gpg: imported secret Keys : 1  
gpg: unchanged secret Keys : 1

And now, if I want to add a userid or sign a different Key I always get the message:  
gpg: Parts of the secret Key are missing  
gpg: signing failed: General Error

Also if I do gpg -K after the import I still have the # after the sec.  
What did I do wrong or maybe missunderstood?

One thing that is working correctly tho, if I delete the Key completly and than import the files, gpg -K gives an output without the #.

I tested this with Win7 (gpg version 1.4.16) and Debian Wheezy (gpg version 1.4.12)

1. **Gert van Dijk  March 14th, 2014**

Great writeup, especially about the key preferences. I see a lot of people having RSA 4096 bit keys nowadays, but haven’t changed any of the default key preferences.

I’m still missing the smartcard/token private key storage here. It would be even more secure to have the smartcard/token generate the keypair, without the secret ever have to leave the token.

1. **Anonymous  May 1st, 2014**

Hi, I don’t understand why the laptop keypair ends up with not only the subkey made manually, but also the other subkey that was created when making the master key.

If that “lesser” keyring gets compromised, one would have to generate two new subkeys from the master key, no ? Why is this not done here in the first place ?

1. ****[**Zachary Lym**](http://indolering.com/)**May 3rd, 2014**

This worked well for me, however, I chose a slightly different option and setup a key management system using Tails. This means that the private keys are never exposed to an uncontrolled environment and it enables me to simply reboot into the key management server if I want to make any changes.

1: Create a persistent Tails USB live disk (boot into Tails and install Tails onto another USB).  
2: Boot into the persistent Tails image.  
3: Follow the above directions but do NOT delete your master key pair. Instead, just save the subkeys file onto a different USB disk.  
4: Reboot into normal OS and import the sub keys.

It’s still a very good idea to save the exported GPG keys onto another, non Tails USB disk for safe keeping.

1. ****[**Zachary Lym**](http://indolering.com/)**May 3rd, 2014**

Why do you bother with the algorithm preference change? AFAICT, your alternative matches the default save for the compression algorithms. Removing SHA1 would improve things, but the order of the compression algorithm has zero impact on security.

1. ****[**Alex Cabal**](https://alexcabal.com/)**May 21st, 2014**

@Anonymous, May 1: That’s right, if your laptop keypair is compromised then you’ll have to generate new subkeys from your master keypair. You can do that at any time though, so if you want to do it all at once then that’s OK.

The difference between the laptop keypair and the master keypair is that the laptop keypair is essentially a set of different subkeys signed by your master keypair. So if your laptop keypair is compromised, you can create a new laptop keypair, sign it with your master keypair again, and carry on with less inconvenience than if you just had a single keypair everywhere.

1. ****[**Alex Cabal**](https://alexcabal.com/)**May 21st, 2014**

@Zachary on Ubuntu 12.04 the algorithm preferences are not entirely the same as what we set them too. Furthermore users of older versions of GPG may benefit from defaulting to newer algorithms.

1. **Andreas Rohner  June 6th, 2014**

Really great tutorial. It helped me tremendously in understanding the whole concept of subkey management. Instead of deleting the primary key I just moved it to a smartcard. <https://en.wikipedia.org/wiki/OpenPGP_card> I can sign and encrypt emails normally, but when I try to sign other keys or change my own key GPG asks me to insert my card and enter my pin. Works like a charm!

1. **Alex B.  June 24th, 2014**

Awesome write-up. It might be worth mentioning that GPG self-signs newly created keys by itself. This may be verified using gpg —edit-key followed by the “check” command.

1. **Kal  August 16th, 2014**

Thanks for the great write-up.

I followed this tutorial, so I now have the following:  
— my master key pair on an SD card (locked away in a drawer);  
— the revocation certificate for the master key pair (locked away in my dry cabinet);  
— the subkey on my laptop.

How do I add more identities though? I tried this:  
$ gpg2 —edit-key [kju@theparty.kp](mailto:kju@theparty.kp)  
gpg (GnuPG) 2.0.25; Copyright (C) 2013 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.

Secret key is available.

pub 4096R/C5CC4D95 created: 2014-08-16 expires: never usage: SC  
trust: ultimate validity: ultimate  
sub 4096R/7E72487D created: 2014-08-16 expires: never usage: E  
sub 4096R/5B5B8F8C created: 2014-08-16 expires: never usage: S  
[ultimate] (1). Kim Jong Un

gpg> adduid  
Real name: Kim Jong Un  
Email address: [kju@gov.kp](mailto:kju@gov.kp)  
Comment:  
You selected this USER-ID:  
“Kim Jong Un ”

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o  
gpg: secret key parts are not available  
gpg: signing failed: Unusable secret key

So I thought maybe I should reimport the master key pair first:  
$ gpg2 —import /Volumes/GPG\_Keys/\.public.gpg-key /Volumes/GPG\_Keys/\.private.gpg-key  
gpg: key C5CC4D95: “kju ” not changed  
gpg: key C5CC4D95: already in secret keyring  
gpg: Total number processed: 2  
gpg: unchanged: 1  
gpg: secret keys read: 1  
gpg: secret keys unchanged: 1

And then I tried `gpg2 —edit-key [kju@theparty.kp](mailto:kju@theparty.kp)` and `adduid` again. But I get the same error messages:  
gpg: secret key parts are not available  
gpg: signing failed: Unusable secret key

What am I missing?

1. **G  September 8th, 2014**

Thanks a lot for your writeup, this is much easier to understand than pulling together bits and pieces from different sources. 🙂

1. **Ultimate-GPG-Settings  September 11th, 2014**

Here is a link to a bash script that increases the GnuPG key size limit beyond 4096 bits.  
The page also provides an ideal GnuPG .conf file.  
<https://gist.github.com/anonymous/3d928a0bcbb3ed92c454>  
<https://tinyurl.com/ultgpgset>  
Please provide input and recommended changes.

Ultimate-GPG-Settings

1. **Marcus Cornus  October 24th, 2014**

Martin  
I had the same problem. After importing private key I got “#sec”. I just deleted private key once more and imported it. Then, using gpg —edit-key your\_key\_id I was able to revoke all subkeys belonging to the key.

Best regard

1. ****[**iain**](http://iainbarnett.me.uk/)**October 25th, 2014**

Thanks for this tutorial, Alex, it’s really helpful. I have a question about creating the revocation certificate ahead of time. When I run the revocation command I get:

Create a revocation certificate for this key? (y/N) y  
Please select the reason for the revocation:  
0 = No reason specified  
1 = Key has been compromised  
2 = Key is superseded  
3 = Key is no longer used  
Q = Cancel  
(Probably you want to select 1 here)  
Your decision?

It’s right, I \*probably\* want to select 1, that’s why I’ve created it ahead of time — but might 0 be a better choice, and does it affect things a lot?

Regards, iain

1. ****[**Alex Cabal**](https://alexcabal.com/)**October 26th, 2014**

You can create multiple reovcation certificates, each with a different reason, if you like. The reason is only there to inform others as to why you revoked it; it’s not strictly necessary, and the key is always marked as revoked regardless of the reason.

1. ****[**iain**](http://iainbarnett.me.uk/)**October 27th, 2014**

Thanks Alex, much appreciated.

1. **crane  January 5th, 2015**

This is one of the best and up-to-date how-to for pgp. Well done! 🙂

1. **iStudent  January 9th, 2015**

Great tutorial, Alex, it’s really helpful.  
I have a question: if I didn’t explicitly upload my public key or anything, can I delete (shred) my local keys and redo your guide from scratch? would “rm -r ~/.gnupg/” do the trick?

Thanks.

1. ****[**Alex Cabal**](https://alexcabal.com/)**January 9th, 2015**

@iStudent yes that should work, but you’d want to shred everything in there instead of using rm.

1. **me  January 11th, 2015**

Excellent tutorial.

One question: Could you do a tutorial or explanation why a master key can be unlimited in time or should be limited in time?

There are quite a few opinions out there if a key should expire or not.

Thanks 🙂

1. ****[**Eddie**](https://samurai.io/)**January 18th, 2015**

Well done. Very helpful and I have several people that I will be sharing it with.

1. **Jason  March 1st, 2015**

This is a great tutorial — thanks for putting it together. However: If this is a best practice, why doesn’t GPG do this by default?

1. **RPM  March 12th, 2015**

Alex,  
Fantastic guide, I really love the attention to detail and instruction method.

One related question — there is always the talk of “…if they get your private key it’s game over”. My question is, if someone steals my private key — won’t they need my passphrase too in order to do anything with it? Is the assumption that the associated password to the private key COULD be cracked (given enough time / cryptanalysis)?

Follow up, if you change your passphrase for your key, what affect does that have on your private/public key as well as messages and files you’ve encrypted? If you start out with a weak passphrase and update to a strong one, are new files any better off? Do you need to update the key server?

Best.

1. **Kevin G.  March 15th, 2015**

Let’s see the OpenPGP smartcard approach to the same problem.

1. **anton  March 30th, 2015**

hi

I have got three email addresses. I would like to use gpg subkeys system. The main goal is to create three different subkey pairs. How can I do? Should I repeat five times the add new singing key step?

1. **Edd Barrett  August 9th, 2015**

Thanks very much for this guide. It clarified many of the questions I had about gpg.

After reading around, I think you could revise the part about making the primary key. If you use `gpg2 —full-gen-key —expert`, you can make the primary key with certify-only capabilities. This avoid having two signing keys (right?).

I also wondered why people don’t (and why gpg doesn’t allow) ridiculous key sizes (8192 and up). The answer seems to be that the larger the keys involved, the longer it takes to compute stuff.

1. **John Miller  September 17th, 2015**

Thx for your work and this guide. I’ve a question about the “Using your new laptop keypair” part.

How can i extract the encryption and certification subkeys to use them in another computer ?

1. ****[**Darren**](https://www.encryptomatic.com/openpgp)**September 18th, 2015**

This is a great article for us. We’re completing a project to simplify installing and using OpenPGP for MS Outlook. This was helpful, thanks.

1. **pratfall  September 21st, 2015**

Hi,

I set this up, and someone tried to send me an encrypted file with:

gpg -e -r my-email@address file

When I tried to decrypt the file on my laptop, gpg said:

gpg: decryption failed: No secret key

I had to have my fried specify a subkey with -r instead of my email address. Is there something I can do to make this easier for people trying to encrypt things for me?

1. **nwt  September 23rd, 2015**

Really liked your guide, thanks. Might use an update for ECC, though.

1. ****[**Thomas Vander Stichele**](http://thomas.apestaart.org/)**October 15th, 2015**

Thanks for this write-up.

1) at which point do you send keys to the keyserver? I assume you do it before you remove the master key from your kerying?

2) Why create a signing subkey, but not an encryption subkey? The latter is suggested also in <https://we.riseup.net/riseuplabs+paow/openpgp-best-practices>

Thanks,  
Thomas

1. **Ben  December 9th, 2015**

Hey Alex, thanks for the guide, I have used this guide every time I needed to create a subkey, though, I am not sure if this is still working. Yesterday, I tried to make a new subkey (following your guide), but I cannot decrypt any files encrypted with the other subkeys I have made a few months ago. Can you confirm this? Am I missing anything? Also I have tried making a encryption key, but still not working. It seems that my problem is similar to @pratfall’s comment.

1. ****[**Kaspars**](http://kaspars.net/)**December 13th, 2015**

Alex, what do you think of creating and storing keys on secure elements such as smartcards, Yubikey, etc.?

1. ****[**Stephen Wood**](http://www.heystephenwood.com/)**December 14th, 2015**

Alex thanks for taking the time for the writeup. I didn’t read through all of the comments so forgive me if my question has already been answered, but which public key do I upload to key servers? The master signing pub key or the sub pub key or both?

Also, what are the best ways to get a gpg key trusted?

1. **Eric Bates  December 29th, 2015**

On the laptop with no master key, how do I specify that the subkey should be used for signing? (Specifically, on Arch, I’m attempting to retrieve and —lsign a developer’s key so I can build enigmail via makepkg.)

Everytime I attempt to sign the developer’s key, gpg specifies (by ID) that it is going to use my master key and then fails, of course, because there is no copy of the secret key on the laptop.

I tried to push gpg to use the subkey by specifying it’s id as either —default-key or —local-user (both on command line and in gpg.conf); but it always insists on using the master.

Thanks.

1. ****[**Alex Cabal**](https://alexcabal.com/)**December 29th, 2015**

From the article:

To sign someone else’s key or to create or revoke a subkey on this keypair, you need to use the master keypair that you keep safe—the one that’s not on your laptop.

1. **Laurence  January 14th, 2016**

“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.”  
—from debian wiki subkeys (<https://wiki.debian.org/Subkeys>)

1. **Radu  March 8th, 2016**

Hey Alex, thanks for the tutorial. Is it possible to use keybase.io with this setup? You kind of need a signing key there right?

1. **Bertrand  March 17th, 2016**

I created a key and gpg told me:

“gpg: revocation certificate stored as ‘/home//.gnupg/openpgp-revocs.d/.rev”

Perhaps it’s specific to “—full-gen-key”. Anyway, I thought it would be worth mentioning because someone could forget it’s in their laptop.

Thanks for the great tutorial!

1. **Mohan  April 7th, 2016**

Hi Alex,  
I need some help regarding gpg encryption on cygwin..encrypton is not working getting error: encryption failed no public key.. it working beofore..due to system(windows server 2008 r2) changes or not sure what happend to cygwin too..but the keyring path modified or changed..in the log i can see the following notes  
gpg: keyring `C:/Users/test/AppData/Roaming/gnupg/secring.gpg’ created  
gpg: keyring `C:/Users/test/AppData/Roaming/gnupg/pubring.gpg’ created with 0kb file size for all( pubring,secring)  
FYI ..key(vendor) is good, am able to manually encrypt the files only it is not working on cygwin associated to cron job…Could you please let me know ..if you have any idea, Thanks

1. **Pushpak  August 18th, 2016**

Thnx a lot Alex. Wonderful and very helpful documentation.

1. ****[**Explorare**](https://hencolle.com/)**September 5th, 2016**

Thanks for this guide. If I revoke my sub-keys and push the change to key server, do I need to tell my friend to update from key server or the gpg client will do it automatically?

1. **Peter  September 5th, 2016**

Fantastic guide. Many thanks!

What do I do to export the laptop keypair to another laptop (or IPG Mail app or the like)? Could I simply use the “export keys to file” in e.g. Enigmail?

1. **Yann  September 12th, 2016**

Hey Alex,

This is a great tutorial!

One thing is still unclear for me. If I want to generate further keys (let’s say for every device I use), what are the steps to reproduce every time?

1. Reimport the master keypair from the encrypted device  
2. Add a new subkey to the master keypair  
3. Reexporting the master keypair (public and private) and save it on the encrypted device (?)  
4. Regenerating a new revocation certificate for the master keypair (?)  
5. Then removing signing subkeys of other devices from the master keypair  
6. Export subkeys (private and public) and use those on the specific device

I’m not sure steps 3 and 4 are really needed, and I might have forgotten some steps. What do you think?

Thanks a lot for your input 🙂

1. **spencer  September 23rd, 2016**

@Boris Ent did you ever figure out what is causing the  
gpg: secret key parts are not available  
gpg: signing failed: Unusable secret key  
error? I am seeing the same issue.

1. **spencer  September 23rd, 2016**

For anyone who encounters the

gpg: secret key parts are not available  
gpg: signing failed: Unusable secret key  
error

I solved it by deleting my master key with

$ gpg —delete-secret-and-public-key [KEY ID]

Then I imported from the private and public back up

$ gpg —import [private key file] [public key file]

after you finish your signing work or add user work (which requires the master private key)  
make sure to follow the securing process again:

$ gpg —export-secret-subkeys [KEY ID] > subways  
$ gpg —delete-secret-key [KEY ID]  
$ gpg —import subkeys  
$ rm subkeys

1. **nothingmuch  October 11th, 2016**

shred is not appropriate in this age of WAL and SSD remapping, tmpfs is a more reliable way of ensuring there is no record of the secret keys on disk.

1. **Guest  November 14th, 2016**

Nota Bene: I had to perform the “Transforming your master keypair into your laptop keypair” for both versions of GPG – gpg and gpg2. This is my solution for the “gpg: decryption failed: No secret key” error.

1. **AMILTON JUSTINO  December 3rd, 2016**

Alex,

Thank you very much for the great tutorial!

1. **anon anonsen  December 24th, 2016**

gpg2 —full-gen-key

is required for setting the changes you want in gpg (GnuPG) 2.1.16 or later, or you will only be able to fill in your name address and name.

Bit of a fingerprint issues too atm, you have to manually pick out the last 8 digits to get the fingerprint. Also really got some photo issues atm, does not show in thunderbird. Also it refuses to show your signatures photo, even in the viewer.

A lot have changed over night here

Great tutorial though.

1. **anon anonsen  December 24th, 2016**

errr that would ofc be mail address and name there

1. **anon anonsen  December 26th, 2016**

I am spamming you here Sir, but is there really noway to get that pesky 3DES shit away, it isn’t good at all. Sweat32 etc.

1. **someone  January 6th, 2017**

Hello.  
Does any tried the above solution with thunderbird and enigmail? i cant sign my mails.  
Thank you

1. **Dustin Krysak  January 12th, 2017**

When publishing to the keyservers — would you push the master public key?

1. **Michael  January 28th, 2017**

Thanks for the nice tutorial!

Even though I was aware of the existence of gpg and gpg2 I first was puzzled that some stuff looked different until I realized my mistake.  
Maybe you make a hint that this tutorial is for gpg1. I think gpg2 has some changes how it stores the keys in the keyring, so this might be relevant anyway?

1. **Mika  May 28th, 2017**

Hi Alex, it is a very useful tutorial. For a newbie like myself, there is still some confusion left. When the second sub key is created and then we deleted the original signing subkey (or the “pub” key) from the keyring, and we have this:  
sec# 4096R/488BA441 2013-03-13  
uid Bilbo Baggins  
ssb 4096R/69B0EA85 2013-03-13  
ssb 4096R/C24C2CDA 2013-03-13

then which key is considered ‘public’ to be uploaded to a keyserver? I guess it is one of the two, either 4096R/69B0EA85 or 4096R/C24C2CDA?  
Thank you.

1. **nmhd  May 28th, 2017**

generate periodic key & export n,p,q (for RSA) in gnupg

hello  
I have 2 questions.

1) I wanna know how does I generate or create one arbitrary key in gnupg?

for ex: I wanna create this key: 10100 10100 10100 10100 … 10100 10100

the key is 2048\_bit number obtained by repeating the pattern 10100

2) how does I see in rsa (generated in gnupg) , publickey n (pq) and p & q?  
I mean, I should use what instruction that show n & p & q for the key that I know his privatekey, it means I generate it.

[parinaz.mahdion@yahoo.com](mailto:parinaz.mahdion@yahoo.com)  
tnx

1. **Robert  September 23rd, 2017**

@Kal: this worked for me to add a user id:

1. I followed the instructions on <https://incenp.org/notes/2015/using-an-offline-gnupg-master-key.html> to “use the offline master key” (basically, importing the key into a temporary directory using something like “gpg2 —homedir ~/gpgtmp —import /run/media/alice/mystick/alice-private-keys.asc”)  
2. In addition to the “—keyring” option given there I also added the “—no-default-keyring” option (something like “gpg —homedir ~/gpgtmp —keyring ~/.gnupg/pubring.gpg —no-default-keyring —edit-key KEYIDHERE”) which then allowed me to add a user id with adduid.

Hope this helps

1. **Laszlo  December 10th, 2017**

Consider using ramfs instead of tmpfs. Swap might be an issue.

1. ****[**Alex Cabal**](https://alexcabal.com/)**December 10th, 2017**

Thanks, I’ve updated the guide!

1. **Brett Randall  March 8th, 2018**

Wonderful write-up and great advice here, thanks!

I do have a query though about the use of the phase “signing subkey” throughout the article. It appears 11 times in the main article — 5 of those mentions refer to the new, signing subkey which is added along the way. The other 6 mentions refer in one way or another to the “original” signing subkey. That naming is what I am unsure about.

According to <https://wiki.debian.org/Subkeys> , “GnuPG actually uses a signing-only key as the master key”. Right after I run —gen-key, I have one master keypair with usage [SC] (signing, certification), and one subkey with usage [E] (encryption). What I’m wondering is — isn’t the original signing key the master key, and not a subkey at all? Or am I missing something and the master keypair is actually multiple entities, and hides a signing subkey?

If not, then I think all (6) references to “signing subkey” in the context of the original master key should be reworded, perhaps to “original signing master key”. I don’t see any obvious signing subkey with a new key-pair, only a master key with signing/certification and an encryption subkey.

Thanks again,  
Brett

1. **Carolyn J. Merrill  April 24th, 2018**

Brett, by default GPG creates a master key that can be used for signing (S) and certifying (C). But it’s possible (and advised!) to add a separate signing subkey and avoid using master key (removing secret part of master key from your daily laptop etc.). When I setup my master key I explicitly disabled signing capability (so it’s C only now).

1

The writing in this blog is dedicated to the [public domain](https://en.wikipedia.org/wiki/Public_domain) via the [CC0 1.0 Universal Public Domain Dedication](https://creativecommons.org/publicdomain/zero/1.0/), and is thus free of copyright restrictions worldwide. [Why?](https://alexcabal.com/why-i-release-things-into-the-public-domain)