

Into.the.Void.

Into The Void

02/12/2013

Creating a new GPG key with subkeys

A few weeks ago I created [my new GPG/PGP key](#) with subkeys and a few people asked me why and how. The rationale for creating separate subkeys for signing and encryption is written very nicely in the [subkeys page of the debian wiki](#). The short answer is that having separate subkeys makes key management a lot easier and protects you in certain occasions, for example you can create a new subkey when you need to travel or when your laptop gets stolen, without losing previous signatures. Obviously you need to keep your master key somewhere very very safe and certainly not online or attached to a computer.

You can find many other blog posts on the net on the subject, but most of them are missing a few parts. I'll try to keep this post as complete as possible. If you are to use gpg subkeys you definitely need an encrypted usb to store the master key at the end. So if you don't already have an encrypted USB [go and make one first](#).

When this process is over you will have a gpg keypair on your laptop without the master key, you will be able to use that for everyday encryption and signing of documents but there's a catch. You won't be able to sign other people's keys. To do that you will need the master key. But that is something that does not happen very often so it should not be a problem in your everyday gpg workflow. You can read about signing other people's keys at the end of this post. AFAIK you can't remove your master key using some of the gpg GUIs, so your only hope is the command line. Live with it...

First some basic information that will be needed later.

When listing *secret keys* with **gpg -K** keys are marked with either 'sec' or 'ssb'. When listing (*public*) keys with **gpg -k** keys are marked with 'pub' or 'sub'.

```
sec => 'SECRet key'
ssb => 'Secret SuBkey'
pub => 'PUBlic key'
sub => 'public SUBkey'
```

When editing a key you will see a usage flag on the right. Each key has a role and that is represented by a character. These are the roles and their corresponding characters:

Constant	Character	Explanation
PUBKEY_USAGE_SIG	S	key is good for signing
PUBKEY_USAGE_CERT	C	key is good for certifying other signatures
PUBKEY_USAGE_ENC	E	key is good for encryption
PUBKEY_USAGE_AUTH	A	key is good for authentication

Before doing anything make sure you have a *backup of your .gnupg dir*.

```
$ umask 077; tar -cf $HOME/gnupg-backup.tar -C $HOME .gnupg
```

Secure preferences

Now edit your *.gnupg/gpg.conf* and add or change the following settings (most are stolen from [Riseup: OpenPGP Best Practices](#)):

```
# when outputting certificates, view user IDs distinctly from keys:
fixed-list-mode
# long keyids are more collision-resistant than short keyids (it's trivial to make a key with any desired short keyid)
keyid-format 0xlong
# when multiple digests are supported by all recipients, choose the strongest one:
personal-digest-preferences SHA512 SHA384 SHA256 SHA224
# preferences chosen for new keys should prioritize stronger algorithms:
default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 BZIP2 ZLIB ZIP Uncompressed
# If you use a graphical environment (and even if you don't) you should be using an agent:
# (similar arguments as https://www.debian-administration.org/users/dkg/weblog/64)
use-agent
# You should always know at a glance which User IDs gpg thinks are legitimately bound to the keys in your keyring:
verify-options show-uid-validity
list-options show-uid-validity
# when making an OpenPGP certification, use a stronger digest than the default SHA1:
cert-digest-algo SHA256
# prevent version string from appearing in your signatures/public keys
no-emit-version
```

Create new key

Time to create the new key. I'm marking user input with **bold (↵)** arrows

```
$ gpg --gen-key
gpg (GnuPG) 1.4.12; Copyright (C) 2012 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?

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.
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 Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: foo bar *****

Email address: foobar@void.gr *****

Comment:
You selected this USER-ID:
"foo bar <foobar@void.gr>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0 *****

You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.....+++++
..+++++

gpg: key 0x6F87F32E2234961E 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: 3 signed: 14 trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: depth: 1 valid: 14 signed: 9 trust: 13-, 1q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2014-03-18
pub 4096R/0x6F87F32E2234961E 2013-12-01
    Key fingerprint = 407E 45F0 D914 8277 3D28 CDD8 6F87 F32E 2234 961E
uid [ultimate] foo bar <foobar@void.gr>
sub 4096R/0xD3DCB1F51C37970B 2013-12-01

Optionally, you can add another uid and add it as the default:

$ gpg --edit-key 0x6F87F32E2234961E
gpg (GnuPG) 1.4.12; Copyright (C) 2012 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/0x6F87F32E2234961E created: 2013-12-01 expires: never usage: SC
    trust: ultimate validity: ultimate
sub 4096R/0xD3DCB1F51C37970B created: 2013-12-01 expires: never usage: E
[ultimate] (1). foo bar <foobar@void.gr>

gpg> adduid *****

Real name: foo bar *****

Email address: foobar@riseup.net *****

Comment:
You selected this USER-ID:
"foo bar <foobar@riseup.net>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0 *****

You need a passphrase to unlock the secret key for
user: "foo bar <foobar@void.gr>"
4096-bit RSA key, ID 0x6F87F32E2234961E, created 2013-12-01

```

```
pub 4096R/0x6F87F32E2234961E created: 2013-12-01 expires: never      usage: SC
                                trust: ultimate   validity: ultimate
sub 4096R/0xD3DCB1F51C37970B created: 2013-12-01 expires: never      usage: E
[ultimate] (1). foo bar <foobar@void.gr>
[unknown] (2). foo bar <foobar@riseup.net>

gpg> uid 2 *****
gpg> primary *****
gpg> save *****
```

Let's see what we've got until now, 0x6F87F32E2234961E is the master key (SC flags) and 0xD3DCB1F51C37970B (E flag) is a separate subkey for encryption.

Add new signing subkey

Since we already have a separate encryption subkey, it's time for a new signing subkey. Expiration dates for keys is a very hot topic. IMHO there's no point in having an encryption subkey with an expiration date, expired keys are working just fine for decryption anyways, so I'll leave it without one, but I want the signing key that I'm regularly using to have an expiration date. You can read more about this topic on the [gnupg manual \(Selecting expiration dates and using subkeys\)](#).

```
$ gpg --edit-key 0x6F87F32E2234961E
gpg (GnuPG) 1.4.12; Copyright (C) 2012 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/0x6F87F32E2234961E created: 2013-12-01 expires: never      usage: SC
                                trust: ultimate   validity: ultimate
sub 4096R/0xD3DCB1F51C37970B created: 2013-12-01 expires: never      usage: E
[ultimate] (1). foo bar <foobar@riseup.net>
[ultimate] (2). foo bar <foobar@void.gr>

gpg> addkey *****
```

Key is protected.

You need a passphrase to unlock the secret key for
 user: "foo bar <foobar@riseup.net>"
 4096-bit RSA key, ID 0x6F87F32E2234961E, created 2013-12-01

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
  = key expires in n days
  w = key expires in n weeks
  m = key expires in n months
  y = key expires in n years
```

Key is valid for? (0) 5y *****

Key expires at Fri 30 Nov 2018 03:36:47 PM EET

Is this correct? (y/N) y *****

Really create? (y/N) y *****

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

```
+++++
.....+++++
```

```
pub 4096R/0x6F87F32E2234961E created: 2013-12-01 expires: never      usage: SC
                                trust: ultimate   validity: ultimate
sub 4096R/0xD3DCB1F51C37970B created: 2013-12-01 expires: never      usage: E
sub 4096R/0x296B12D067F65B03 created: 2013-12-01 expires: 2018-11-30 usage: S
[ultimate] (1). foo bar <foobar@riseup.net>
[ultimate] (2). foo bar <foobar@void.gr>

gpg> save *****
```

As you can see there's a new subkey 0x296B12D067F65B03 with just the S flag, that the signing subkey. Before moving forward it's wise to create a revocation certificate:

```
$ gpg --output 0x6F87F32E2234961E.gpg-revocation-certificate --armor --gen-revoke 0x6F87F32E2234961E
sec 4096R/0x6F87F32E2234961E 2013-12-01 foo bar <foobar@riseup.net>
```

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? 1 *****

Enter an optional description; end it with an empty line:

> **This revocation certificate was generated when the key was created. *******
>

Reason for revocation: Key has been compromised

This revocation certificate was generated when the key was created.

Is this okay? (y/N) y *****

You need a passphrase to unlock the secret key for

user: "foo bar <foobar@riseup.net>"

4096-bit RSA key, ID 0x6F87F32E2234961E, created 2013-12-01

Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets access to this certificate he can use it to make your key unusable. It is smart to print this certificate and store it away, just in case your media become unreadable. But have some caution: The print system of your machine might store the data and make it available to others!

Encrypt this file and store it someplace safe, eg your encrypted USB. You should definitely not leave it at your laptop's hard disk. You can even print it and keep it in this form, it's small enough so one could type it if needed.

Remove Master key

And now the interesting part, it's time to remove the master key from your laptops's keychain and just leave the subkeys. You will store the master key in the encrypted usb so it stays safe.

First go and backup your .gnupg dir on your encrypted USB. Don't move forward until you do that. DON'T!

```
$ rsync -avp $HOME/.gnupg /media/encrypted-usb
or
$ umask 077; tar -cf /media/encrypted-usb/gnupg-backup-new.tar -C $HOME .gnupg
```

Did you backup your key? Are you sure ?

Then it's time to remove the master key!

```
$ gpg --export-secret-subkeys 0x6F87F32E2234961E > /media/encrypted-usb/subkeys
$ gpg --delete-secret-key 0x6F87F32E2234961E
$ gpg --import /media/encrypted-usb/subkeys
$ shred -u /media/encrypted-usb/subkeys
```

What you've accomplished with this process is export the subkeys to */media/encrypted-usb/subkeys* then *delete the master key* and *re-import just the subkeys*. **Master key resides only on the encrypted USB key now. Don't lose that USB key.** USB keys are extremely cheap, make multiple copies of the encrypted key and place them in safe places, you can give one such key to your parents or your closest friend in case of emergency. For safety, make sure there's at least one copy outside of your residence.

You can see the difference of the deleted master key by comparing the listing of the secret keys in your .gnupg and your /media/encrypted-usb/.gnupg/ dir.

```
$ gpg -K 0x6F87F32E2234961E
sec# 4096R/0x6F87F32E2234961E 2013-12-01
uid                               foo bar <foobar@riseup.net>
uid                               foo bar <foobar@void.gr>
ssb 4096R/0xD3DCB1F51C37970B 2013-12-01
ssb 4096R/0x296B12D067F65B03 2013-12-01 [expires: 2018-11-30]

$ gpg --home=/media/encrypted-usb/.gnupg/ -K 0x6F87F32E2234961E
sec 4096R/0x6F87F32E2234961E 2013-12-01
uid                               foo bar <foobar@riseup.net>
uid                               foo bar <foobar@void.gr>
ssb 4096R/0xD3DCB1F51C37970B 2013-12-01
ssb 4096R/0x296B12D067F65B03 2013-12-01 [expires: 2018-11-30]
```

Notice the **pound (#)** in the 'sec' line from your ~/.gnupg/. That means that the master key is missing.

Upload your new key to the keyserver if you want to...

Key Migration

In case you're migrating from an older key you need to sign your new key with the old one (not the other way around!)

```
$ gpg --default-key 0x0LD_KEY --sign-key 0x6F87F32E2234961E
```

Write a [transition statement](#) and sign it with both the old and the new key:

```
$ gpg --armor -b -u 0x0LD_KEY -o sig1.txt gpg-transition.txt
$ gpg --armor -b -u 0x6F87F32E2234961E -o sig2.txt gpg-transition.txt
```

That's about it...upload the transition statement and your signatures to some public space (or mail it to your web of trust).

Signing other people's keys

Because your laptop's keypair does not have the master key anymore and the master key is the only one with the 'C' flag, when you want to sign someone else's key, you will need to mount your encrypted USB and then issue a command that's using that encrypted directory:

```
$ gpg --home=/media/encrypted-usb/.gnupg/ --sign-key 0xSomeones_keyid
```

Export your signature and send it back to people whose key you just signed..

Things to play with in the future

Next stop ? An [OpenPGP Smartcard!](#) ([eshop](#)) or a [yubikey NEO](#), ([related blogpost](#)). Any Greeks want to join me for a mass (5+) order?

References

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

<https://help.riseup.net/en/security/message-security/openpgp/best-practices>

<https://alexcabal.com/creating-the-perfect-gpg-keypair/>

P.S. 0x6F87F32E2234961E is obviously just a demo key. You can find my real key [here](#).

P.S.2 The above commands were executed on gpg 1.4.12 on Debian Wheezy. In the future the output of the commands will probably differ.

Filed by kargig at 13:07 under [Encryption, Internet, Linux, Networking, Privacy](#).



Tags: [debian](#), [encrypted usb](#), [Encryption](#), [gpg](#), [keyring](#), [master key](#), [pgp](#), [private](#), [public](#), [signing](#), [subkeys](#), [wheezy](#).

14 Comments | 134,487 views

14 Responses to “Creating a new GPG key with subkeys”

1. Óscar


March 5th, 2014 | [17:15](#)

Using  Safari 537.74.9 on  Mac OS X 10.9.2

Thank you for your article. Very helpful! But why do you generate the master key with signing capabilities?

2. [Transitioning to a new more secure GnuPG key.](#) | [Torsten's Thoughtcrimes](#)



March 31st, 2014 | [20:00](#)

Using  WordPress 3.6.1

[...] any questions about this transition. If you want to transition to a new key as well, you might find this guide [...]

3. ds

April 2nd, 2014 | [02:53](#)



Using  Debian IceWeasel 30.0a2 on  Linux

is hash, not pound.

£ is pound.

4. djmwj

April 22nd, 2014 | [09:53](#)

Using  Mozilla Firefox 20.0 on  Ubuntu Linux

How do you receive signatures back to the master key on the usb when someone else signs your key?

5. a



July 16th, 2014 | [19:21](#)

Using  Debian IceWeasel 30.0 on  Linux

Reference link has been updated: <https://help.riseup.net/en/security/message-security/openpgp/best-practices>

6. gpgman819

September 11th, 2014 | [06:17](#)



Using  Mozilla Firefox 24.0 on  Windows 7

Here is a link to a bash script that increases the GnuPG key size limit beyond 4096 bits. The page also gives a sample GnuPG .conf file.

<https://gist.github.com/anonymous/3d928a0bcb3ed92c454> Please provide input and recommended changes.

7. lastuwfpwa

September 11th, 2014 | [06:29](#)

Using  Mozilla Firefox 24.0 on  Windows 7

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/3d928a0bcb3ed92c454>



<https://tinyurl.com/ultgpgset>

Please provide input and recommended changes.

Ultimate-GPG-Settings

8. David



October 24th, 2014 | [15:24](#)

Using  Mozilla Firefox 24.0 on  Windows 7

I have found very little online on the construction of the passphrase itself. Natural language is ill-advised, I understand, but I'm particularly curious what is considered to be the minimum number of characters to make it truly effective and secure. Thanks!

9. paradox

January 11th, 2015 | [12:30](#)

Using  Mozilla Firefox 34.0 on  Linux

you could also write:

```
gpg -u oldkeyID -u newkeyID --clearsign gpg-transition.txt
```

to sign the statement and have a gpg-transition.txt.asc version



which then issuing:

```
gpg --verify gpg-transition.txt.asc
```

will show everyone who signed the transition (your old key and your new)

10. jdeo

September 22nd, 2015 | [17:27](#)

Using  Google Chrome 44.0.2403.155 on  Linux



“Then add another uid and add it as the default:”

Why on earth would I need that?

Also, any updates for gpg 2.1.8?

11. Kristian



June 12th, 2016 | [13:10](#)

Using  Mozilla Firefox 45.0 on  Linux

Thanks for the nice tutorial!

12. matt



April 21st, 2017 | [00:41](#)

Using  Google Chrome 57.0.2987.133 on  Windows 7

nice! and yet no explanation on how to export the public half of any subordinate encryption keys. I want to hand out different encryption keys to different organizations. They are sending me files and I want the public key they each use to be different from the other.

13. matt

April 21st, 2017 | [06:14](#)

Using  Google Chrome 57.0.2987.133 on  Windows 7

sorry to answer my own question – my Google'foo stinks, nobody does this, or the solution is obvious to practitioners that it doesn't need to be asked.

export just the public half of a particular subkey



```
$ gpg --export --armor !
```

encrypt using a specific key

```
$ gpg --encrypt -r ! ...
```

14. Antyradek

June 11th, 2017 | [07:31](#)

Using  Google Chrome 42.0.2311.135 on  Windows 7

Instead of removing the whole key and then reimporting subkeys, you can just delete key file.

```
$ gpg -K --with-keygrip
```

Will list also keygrips, which are filenames in ~/.gnupg/private-keys-v1.d

Delete the master key file or better, change it to soft link to the backup key stored on your USB (will work as long as it is always mounted under the same name).

Leave a reply

name (required)

email (will not be shown) (required)

website

 wipe right and hold to unlock

Submit Comment

Search

Related Posts

- [keys.void.gr – A GPG Keyserver in Greece](#)
- [Anonymize headers in postfix](#)
- [New gpg key](#)
- [Greek PGP Web of Trust 2012 edition](#)
- [Scaling a small streaming system from 50 to 4000+ users](#)

Links

- [Home](#)
- [RSS](#)

December 2013

M	T	W	T	F	S	S
					1	
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					
« Nov						May »

Archives

- [2018](#)
- [2017](#)
- [2016](#)
- [2015](#)
- [2014](#)
- [2013](#)
- [2012](#)
- [2011](#)
- [2010](#)
- [2009](#)
- [2008](#)
- [2007](#)
- [2006](#)
- [2005](#)
- [2004](#)

Recent Comments:

- [ferongr](#): Υπάρχει κάποιο update σχετικά με αυτά που αναφέρονται στην παρούσα...
- [method77](#): Λειτουργει ακομα η λιστα; Στο ad away android μου λεει “not...

- [forcedalias](#): As people mentioned, if you just want a simple, secure...
- [popo](#): This is great, thank you. This worked out very well. In addition to the...
- [Kade](#): Thanks for the helpful tutorial! Worked great to replace server private...

Most Viewed Posts:

- [Some statistics on linux-greek-users mailing list and forum.hellug.gr](#) - 174,307 views
- [Creating a new GPG key with subkeys](#) - 134,487 views
- [Openvpn – MULTI: bad source address from client – solution](#) - 105,308 views
- [There's a rootkit in the closet!](#) - 99,148 views
- [Greek adblock plus filter](#) - 89,592 views
- [SSH Escape Characters](#) - 85,137 views
- [Linux SSD partition alignment tips](#) - 71,019 views
- [Rate limit outgoing emails from PHP web applications using postfix and policyd](#) - 65,331 views
- [Anonymize headers in postfix](#) - 60,345 views
- [Bypassing censorship devices by obfuscating your traffic using obfsproxy](#) - 59,786 views

Categories

- [Encryption](#) (28)
- [General](#) (179)
- [Gentoo](#) (37)
- [Greek](#) (42)
- [Internet](#) (112)
- [IPv6](#) (16)
- [Linux](#) (244)
- [MacOSX](#) (8)
- [maths](#) (3)
- [Networking](#) (73)
- [Privacy](#) (51)
- [Private](#) (12)
- [slogans](#) (6)
- [VoIP](#) (5)

Tag Cloud

[adblock](#) [adblock plus](#) [ads](#) [antispam](#) [athens](#) [bash](#) [blacklist](#) [bug](#) [bugzilla](#) [debian](#) [DHCPv6](#) [dns](#) [email](#) [Encryption](#) [firefox](#) [fluxbox](#) [fosscomm](#) [Gentoo](#) [gpg](#) [Greek](#)
[grrbl](#) [https](#) [iceweasel](#) [Internet](#) [ipv6](#) [Linux](#) [mysql](#) [Networking](#) [ossec](#) [perl](#) [php](#) [plesk](#) [portage](#) [postfix](#) [presentation](#) [Privacy](#) [python](#) [security](#) [slaac](#) [spam](#)
[swiftfox](#) [tls](#) [tor](#) [vulnerability](#) [wordpress](#)

◦ Pages

- [Contact](#)
- [Digital rights](#)
- [Greek adblock plus filter](#)
- [GrRBL](#)
- [My PGP/GPG key](#)
- [Presentations/Articles](#)

◦ Blogs

- [modal echoes of the void](#)
- [∴ autoverse ∴](#)
- [~mperedim/weblog](#)
- [0entropy](#)
- [About life, the universe and everything](#)
- [argp's blog](#)
- [dimitris kalamaras](#)
- [dmesg | less](#)
- [Don't fear the penguin](#)
- [e-Προβάρια](#)
- [vortex of false deceit](#)

◦ Links

- [Census Labs](#)
- [Ioannina Linux Users Group](#)
- [IPv6 material](#)

◦ Projects

- [Greek AdblockPlus Filter](#)
- [HTTPS-Everywhere Greek Rules](#)
- [ilooq](#)
- [Tormap](#)

◦ Meta

- [Log in](#)



This work is licensed under a [Creative Commons Attribution-Noncommercial-Share Alike 3.0 Greece License](#).

Visits

Using a modified [Gentle Calm theme](#).

W o r d P r e s s took 0.573 seconds to generate this [XHTML](#) page.