OPENPGP - THE ALMOST PERFECT KEY PAIR

October 13, 2017 | 14 mins Thierry Thuon

In this article, I would like to create a small guide on creating a perfect PGP key. For those who do not know, OpenPGP is a standard for encrypting and decrypting messages. Unlike a simple RSA key pair, the protocol OpenPGP allows to create a digital identity that is verified by other people and that is decentralized. There is no authority that will control the identity. It's the users who will check the other people.

Through a set of 4 articles, we will see:

- How to create the PGP key pair
- How to export secrets on a Yubikey smart card
- The storage and backup of master key
- The join in a key signing party

INSTALL THE RIGHT TOOLS

Whether you are on Linux, Mac or Windows, everything can be done in command lines.

First, let's install the tools:

- Windows: **GPG4Win**
- Mac: **GPGtools**
- Linux: **gnupg.org** (already integrated in Ubuntu for example){:rel="nofollow noreferrer"}

For this article, I will run on Ubuntu 16.04 and GnuPG v2.1.11. This is the modern version of gnupg that will replace v1.4 and v2.0.

Before you start creating the key, you need to configure gpg to enhance security. Th Top lines avoid leaking information on how the key was created. Then there is the configuration

there are restrictions on the encryption algorithms in order to use the best and most resistant to date.

Copy this configuration to ~/.gnupg/gpg.conf (Linux and Mac) or C:\Users[username]\AppData\Roaming\gnupg\gpg.conf (Windows).

```
# Avoid information leaked
no-emit-version
export-options export-minimal

# Displays the long format of the ID of the keys and their fingerprints
keyid-format 0xlong
with-fingerprint

# Displays the validity of the keys
list-options show-uid-validity
verify-options show-uid-validity

# Limits the algorithms used
personal-cipher-preferences AES256
personal-digest-preferences SHA512
default-preference-list SHA512 SHA384 SHA256 RIPEMD160 AES256 TWOFISH BLOWFISH ZLI

cipher-algo AES256
digest-algo SHA512
cert-digest-algo SHA512
compress-algo ZLIB

disable-cipher-algo 3DES
weak-digest SHA1

s2k-cipher-algo AES256
s2k-digest-algo SHA512
s2k-mode 3
s2k-count 65011712
```

TAKE ADVANTAGE OF SUBKEYS

When creating an OpenPGP key in its basic mode, gpg will create a key pair that allows you to sign and certify. To increase the security of our key, we will use a special feature of OpenPGP: the subkeys.

OpenPGP allows to create subkeys with a specific use: sign, encrypt and authenticate. Another advantage to the use of the subkeys is that in the event of loss or theft of the secret keys of the subkeys, you only need to revoke the subkey without having to revoke th key (the one that allows to certify other keys).

Top

subkeys to sign, encrypt and authenticate.

CREATING THE MASTER KEY

We will choose to generate our key in a custom way and create the certification key for Wilson. It will allow to certify other keys. It is very important, you must keep it safely. In the event of loss or theft, the person who holds the key would then be able to pretend to be the

```
wilson@spaceship:~$ gpg2 --expert --full-gen-key

gpg (GnuPG) 2.1.11; Copyright (C) 2016 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)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
   (9) ECC and ECC
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
Your selection? 8
```

Then you have to select the attributes of this key. Only the capability **Certify**.

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate
Current allowed actions: Sign Certify Encrypt

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? s

Possible actions for a RSA key: Sign Certify Encrypt Authenticate
Current allowed actions: Certify Encrypt

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? e
```

Top

```
Current allowed actions: Certify

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? q
```

We have configured the capabilities of this first key to allow only certification. Let's move on to the size of the key: it is recommended to have a key of a minimum size of 2048. To this day, this length is still resistant, but it is preferable to take the maximum size: 4096.

For the lifetime of the key, it is always recommended to put one. If this key is lost, and it has been sent to a key server, it will remain there forever valid. Put a duration up to 2 years. Here I will put 1 year. This allows you to practice command lines every year :).

```
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) 1y
Key does not expire at all
Is this correct? (y/N) y
```

Let's add details about Wilson's identity:

```
GnuPG needs to construct a user ID to identify your key.

Real name: Wilson Eleven
Email address: wilson.eleven@labs.com
Comment:
You selected this USER-ID:
    "Wilson Eleven <wilson.eleven@labs.com>"

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

Top

Choose one long enough that you can memorize easily.

```
gpg: key 1A8132B1 marked as ultimately trusted
gpg: directory '/home/wilson/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/wilson/.gnupg/openpgp-revocs.d/5EA44F
public and secret key created and signed.

gpg: checking the trustdb
eeded: 3  completes needed: 1  trust model: PGP
alid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
pub   rsa4096/1A8132B1 2017-10-05 [] [expires: 2018-10-10]
      Key fingerprint = 5EA4 4FF5 3CEB 240F D3F1  A6E4 DCEE 216E 1A81 32B1
uid         [ultimate] Wilson Eleven <wilson.eleven@labs.com>
```

The master key pair is created. Now create the subkeys.

CREATING SUBKEYS

As we saw in the introduction on the subkeys, it is important to have one dedicated to each task:

- Authenticate (A)
- Sign (S)
- Encrypt (E)

Let's create them now.

We will first list the available keys:

```
wilson@spaceship:~$ gpg2 --list-keys

/home/wilson/.gnupg/pubring.gpg
--------------------------------
pub   rsa4096/1A8132B1 2017-10-05 [C] [expires: 2018-10-10]
uid         [ultimate] Wilson Eleven <wilson.eleven@labs.com>
```

Edit it to add subkeys. To do this, you will need to switch to expert mode.

```
wilson@spaceship:~$ gpg2 --expert --edit-key 1A8132B1
gpg (GnuPG) 2.1.11; Copyright (C) 2016 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.
```

Top

```
sec  rsa4096/1A8132B1
     created: 2017-10-05  expires: 2018-10-05  usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1). Wilson Eleven <wilson.eleven@labs.com>

gpg>
```

You are now in edit mode.

Add the encryption key with the `addkey` command.

```
gpg> addkey
Please select what kind of key you want:
   (3) DSA (sign only)
   (4) RSA (sign only)
   (5) Elgamal (encrypt only)
   (6) RSA (encrypt only)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
  (12) ECC (encrypt only)
  (13) Existing key
Your selection? 8

Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions: Sign Encrypt

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? s

Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions: Encrypt

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? q
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
```

Top

```
Key expires at ven. 05 oct. 2018 13:37:19 CEST
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.

sec  rsa4096/1A8132B1
     17-10-05  expires: 2018-10-05  usage: C
     mate      validity: ultimate
ssb  rsa4096/B73A9C79
     created: 2017-10-05  expires: 2018-10-05  usage: E
[ultimate] (1). Wilson Eleven <wilson.eleven@labs.com>

gpg>
```


The key with fingerprint B73A9C79 has been created. Repeat for *Signing* and *Authentication* key.

In the end, you must have these keys:

```
sec  rsa4096/1A8132B1
     created: 2017-10-05  expires: 2018-10-05  usage: C
     trust: ultimate      validity: ultimate
ssb  rsa4096/B73A9C79
     created: 2017-10-05  expires: 2018-10-05  usage: E
ssb  rsa4096/9CC8B2FB
     created: 2017-10-05  expires: 2018-10-05  usage: S
ssb  rsa4096/8047B454
     created: 2017-10-05  expires: 2018-10-05  usage: A
[ultimate] (1). Wilson Eleven <wilson.eleven@labs.com>

gpg> save
gpg> quit
```

Enter `save` then `quit`, and you're done. Wilson now has an OpenPGP key pair with its identity and subkeys with each a capability. Before you can fully use this key, let's backup it.

EXPORT THE MASTER KEY

The PGP key must not be used as it is. Remember, in the event of theft of the master key and the password, the robber can spoof the digital identity and sign messages instead of the real person. It is therefore essential to separate the master key from the subkeys. The r

from the network.

First, create a revocation certificate in the event of theft of the master key.

```
wilson@spaceship:~$ gpg2 --output 1A8132B1.rev --gen-revoke 1A8132B1
```

The revocation certificate is created in `1A8132B1.rev`. It must be preserved in a safe place (USB).

Let's also save all keys.

```
wilson@spaceship:~$ gpg2 --export --armor 1A8132B1 > 1A8132B1.pub.asc
wilson@spaceship:~$ gpg2 --export-secret-keys --armor 1A8132B1 > 1A8132B1.priv.asc
wilson@spaceship:~$ gpg2 --export-secret-subkeys --armor 1A8132B1 > 1A8132B1.sub_p
```

`1A8132B1.pub.asc` will contain all public keys and `1A8132B1.priv.asc` the private keys of the master key. `1A8132B1.sub_priv.asc` contains only the private keys of the subkeys.

As mentioned above, we will only use the subkeys for daily use.

Let's delete all private keys.

```
wilson@spaceship:~$ gpg2 --delete-secret-key 1A8132B1
```

Then, we import only the private keys of the subkeys.

```
wilson@spaceship:~$ gpg2 --import 1A8132B1.sub_priv.asc
```

Let's check that we have only the private keys of the subkeys:

```
wilson@spaceship:~$ gpg2 --list-secret-keys
/home/wilson/.gnupg/secring.gpg
sec#  rsa4096/1A8132B1 2017-10-05 [C] [expires: 2018-10-05]
uid         [ultimate] Wilson Eleven <wilson.eleven@labs.com>
ssb   rsa4096/B73A9C79 2017-10-05 [E] [expires: 2018-10-05]
ssb   rsa4096/9CC8B2FB 2017-10-05 [S] [expires: 2018-10-05]
ssb   rsa4096/8047B454 2017-10-05 [A] [expires: 2018-10-05]
```

Top

a stub instead.

All the files we have created will have to be kept offline (CD, USB stick, magnetic tape, paper sheet, ...).

CONCLUSION

we have created a PGP key with a set of subkeys dedicated to a particular task. The advantage of using OpenPGP against a simple asymmetric key is the subkeys. If one of the keys is compromised, you only need to revoke it and regenerate a new one. It will not be necessary to revoke the master key, the one that holds our digital identity. This strategy offers a much higher level of security.

You can now sign your emails and get them encrypted, sign your commit git, use keybase.io and even authenticate yourself to a server in SSH.

In addition, on November 2, there will be a **key signing party** (key signing party) {:rel="nofollow noreferrer"} at Mozilla France. This event is an opportunity to meet other OpenPGP enthusiasts and, above all, it will make it possible to have your newly created key certified.

ARTICLE EN RELATION

- **OpenPGP - The almost perfect key pair (part 1)**
- **OpenPGP - Export Secret Keys to a Yubikey (part 2)**
- **OpenPGP - Long term storage (part 3)**
- **OpenPGP - I was in a Key Signing Party (part 4)**

RESOURCES

- **GPG : création de votre première paire de clefs et chiffrement d'un fichier**
- **GPG : comment créer une paire de clefs presque parfaite**
- **Creating the perfect GPG keypair**
- **Ma nouvelle clé PGP**
- **OpenPGP Best Practices**
- **Using OpenPGP subkeys in Debian development**
- **PGP Key signing party**

Top

AUTHOR(S)

Thierry Thuon

Astronaute adepte de Symfony.

YOU CAN ALSO READ

MANAGING TRANSLATIONS WITH LOCALISE.BIZ

May 9, 2019 | 3 mins Marie Minasyan

In this article I am going to introduce localise.biz - a SaaS tool to manage translations.

PRESENTATION OF THE PHP LIBRARY XPRESSION

September 10, 2018 | 12 mins Anthony MOUTTE

As developers, all of us already had to filter dataset (array, collection, API, etc...). Let's focus on the Xpression library, wich allows us to filter different content sources with a simplified query syntax.

SYMFONY 4

September 5, 2018 | 3 mins Anthony MOUTTE

Let's take a look at Symfony 4 innovations and changes.

sakis • 2 months ago

As **@Alexandre Franc** said, the link to Part 2 is broken. The English version of the article is here: https://blog.eleven-labs.co...

Reply • Share ›

Alexandre Franc • a year ago

thnaks for sharing your knowledge. your link "OpenPGP - Export Secret Keys to a Yubikey (part 2)" is broken but I could find the french version of your article and will try. thanks again you're a time saver :)

Reply • Share ›

Site is Ready for Offline Use

+33 1 82 83 11 75

15, avenue de la Grande Armée 75016 PARIS

Top