Set up verified commits on GitHub

GitHub uses Cryptographic signatures in the form of the GNU Privacy Guard (GPG) key.

Such key comes in a public key which we'll set up in GitHub, and a private key that lives on our computer. Next time we send a commit to GitHub, it will use this key and encrypt our commit and data. On the GitHub site, it will decrypt and make sure it's the right user.

Step 1 Install GPG

I'll be using Homebrew since it;'s the quickest way to install it.

Run the following command in your terminal.

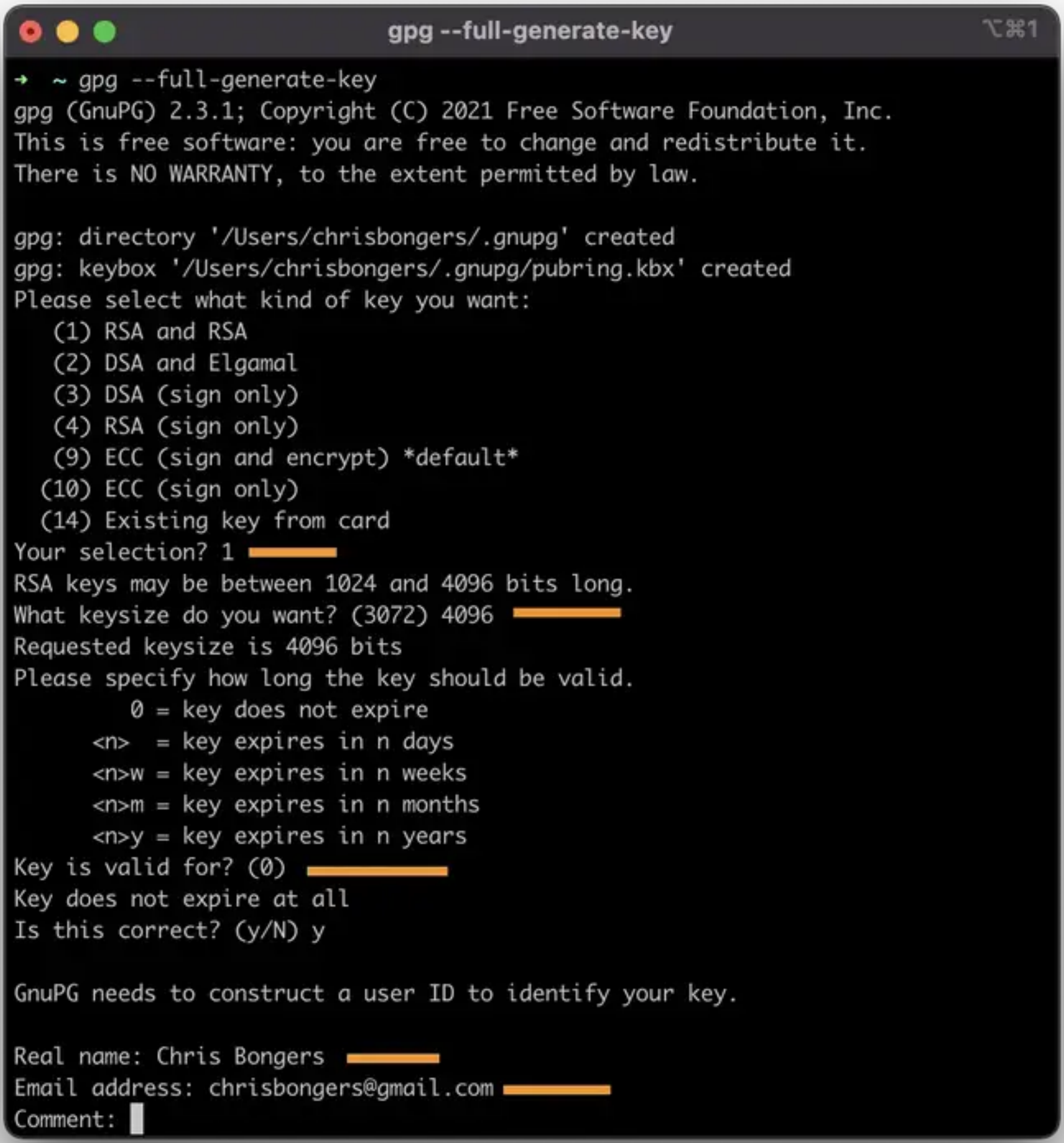
```
brew install gpg
```

Step 2 Generate a new GPG key

We can now use the GPG command to generate a new key. Run the following command in your terminal.

```
gpg --full-generate-key
```

This will prompt a CLI program that will guide you through the process of generation.

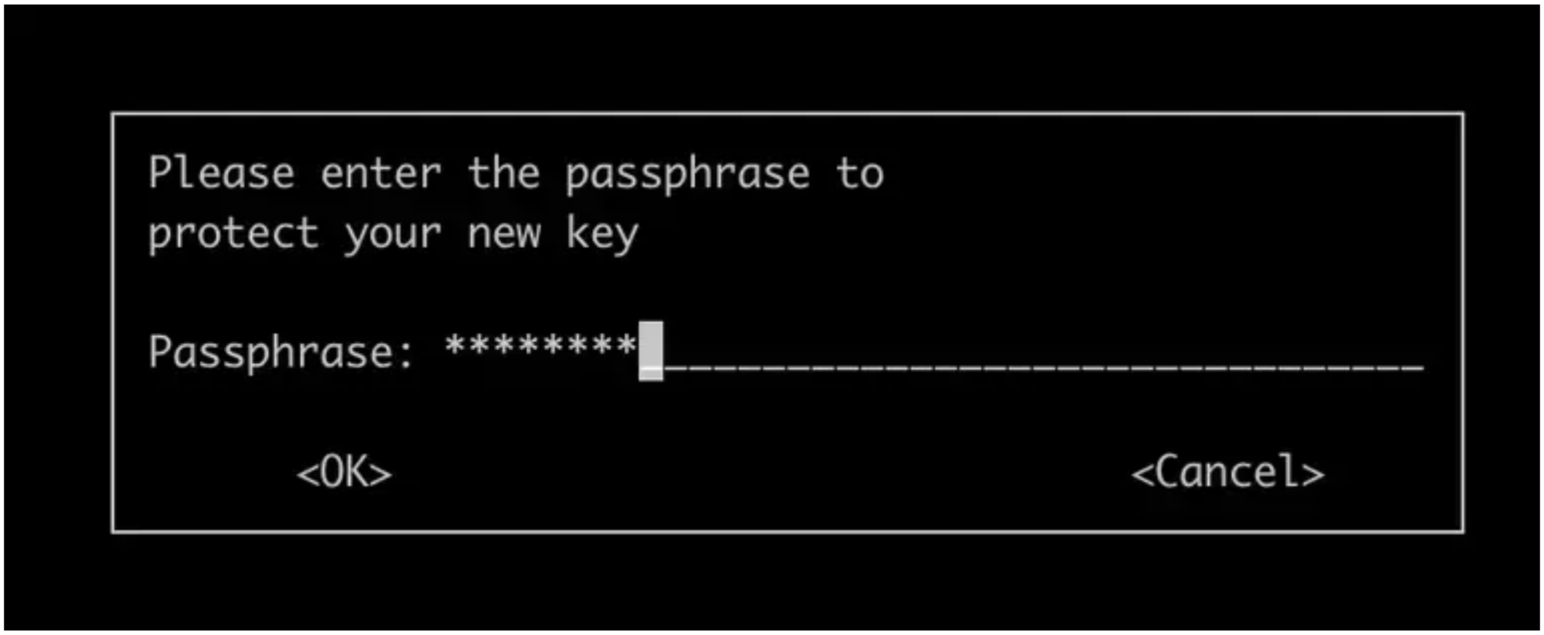


```
➜  ~ gpg --full-generate-key
gpg (GnuPG) 2.3.1; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/Users/chrisbongers/.gnupg' created
gpg: keybox '/Users/chrisbongers/.gnupg/pubring.kbx' created
Please select what kind of key you want:
   (1) RSA and RSA
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (9) ECC (sign and encrypt) *default*
  (10) ECC (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 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)
Key does not expire at all
Is this correct? (y/N) y

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

Real name: Chris Bongers
Email address: chrisbongers@gmail.com
Comment:
```

Keep in mind the following settings:

- I choose an RSA key (Option 1)
- Key should be a MINIMUM of 4096 in size
- I set it never to expire (Option 0)
- Name and email. Use your GitHub email else, it won't work!

After this, it will prompt a password field twice. Use a secure password for this.


```
Please enter the passphrase to
protect your new key

Passphrase: ********_________________________________

     <OK>                                  <Cancel>
```

Step 3 verify the GPG key

Now that we created the key let's verify it's set up correctly.

Run the following command in a terminal.

```
gpg --list-secret-keys --keyid-format LONG
```

This should return something like this:

```
[SCRAMBLING NOT REALLY IMPORTANT]
------------------------------------
sec   rsa4096/[THIS_KEY_ID] 2021-07-07 [SC]
      2B18EEB732D15480D40A60D605AE1785E201CE95
uid                 [ultimate] Chris Bongers <chrisbongers@gmail.com>
ssb   rsa4096/C98A99F6B0202433 2021-07-07 [E]
```

Noted, I added the [THIS_KEY_ID], which will contain a key that is important to do the commits with.

Step 4 Export the key to GitHub

Now we should be able to create an export of this key to use in GitHub.

With the [THIS_KEY_ID] you got from step 3, run the following command:

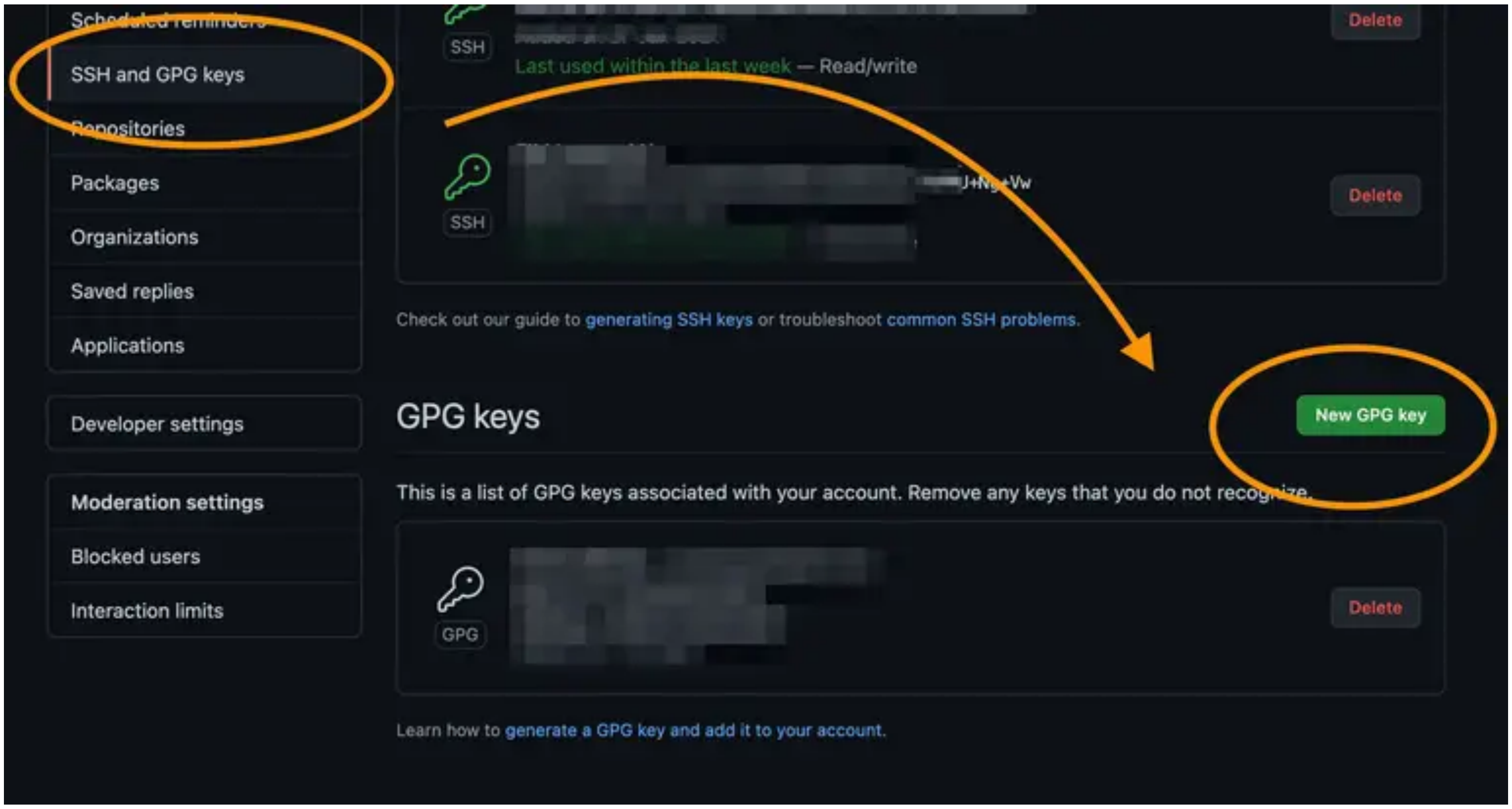
```
gpg --armor --export [THIS_KEY_ID]
```

This will generate a large code block between:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
[SCRAMBLE]
-----END PGP PUBLIC KEY BLOCK-----
```

Copy that whole section, including the comments.

Now head over to GitHub, click on your profile image -> Settings. Choose SSH and GPG Keys from the left menu, scroll down and add a new GPG Key.


Scheduled reminders
SSH and GPG keys
Repositories
Packages
Organizations
Saved replies
Applications
Developer settings
Moderation settings
Blocked users
Interaction limits
SSH
Last used within the last week — Read/write
Delete
SSH
Delete
Check out our guide to generating SSH keys or troubleshoot common SSH problems.
GPG keys
New GPG key
This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.
GPG
Delete
Learn how to generate a GPG key and add it to your account.

Copy that code block in the editor, and press save. You should now have your GPG key setup.

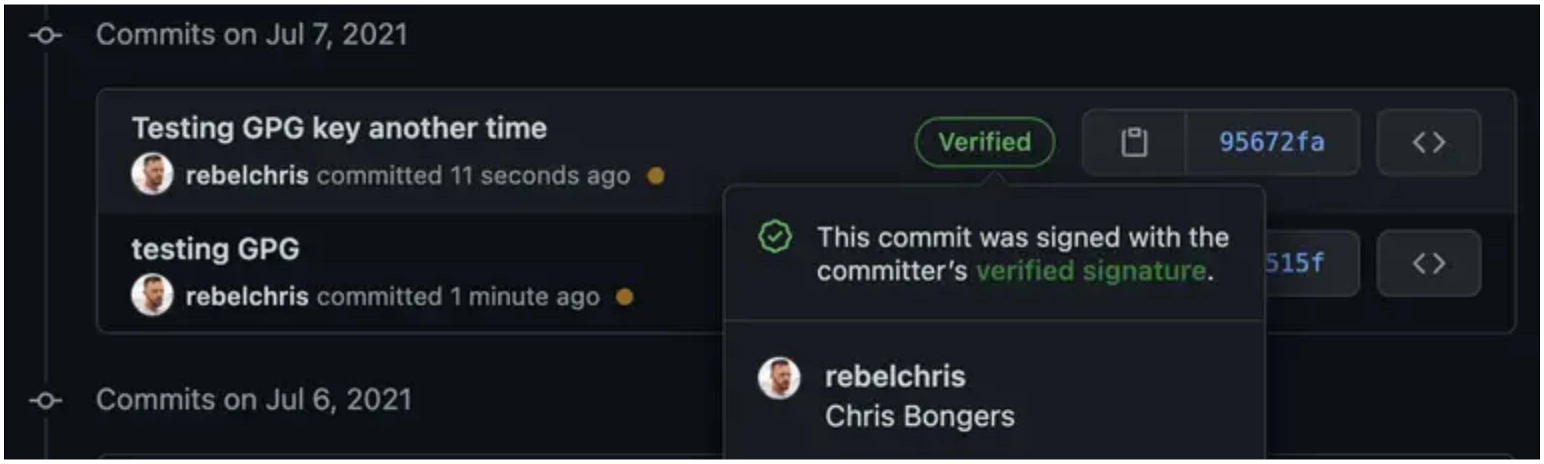
Step 5 Configure git always to sign commits

Let's enable the Git client always to sign commits with our new key.

Run the following commands in a terminal.

```
git config --global user.signingkey [THIS_KEY_ID]
git config --global commit.gpgsign true
```

Now try and commit to one of your projects. And it should show a verified commit like this:


Commits on Jul 7, 2021

Testing GPG key another time
rebelchris committed 11 seconds ago

Verified 95672fa

testing GPG
rebelchris committed 1 minute ago

515f

This commit was signed with the committer's verified signature.

rebelchris
Chris Bongers

Commits on Jul 6, 2021

Troubleshooting

If you have issues in the last part and the response is saying the commit can't be verified, you can try the following:

```
echo "test" | gpg --clearsign
```

If that is showing it failed, use the following command:

```
export GPG_TTY=$(tty)
```

Now rerun the command, and you should see a PGP signature.

It might also help kill the gpg client so that it will ask for the password the first time!

```
gpgconf --kill all
gpg-agent --daemon
```

Thank you for reading, and let's connect!

Thank you for reading my blog. Feel free to subscribe to my email newsletter and connect on Facebook or Twitter

🛠️ Edit on GitHub

Spread the knowledge with fellow developers on Twitter

Tweet this tip

+ 28 devs liked it

Powered by Webmentions -Learn more