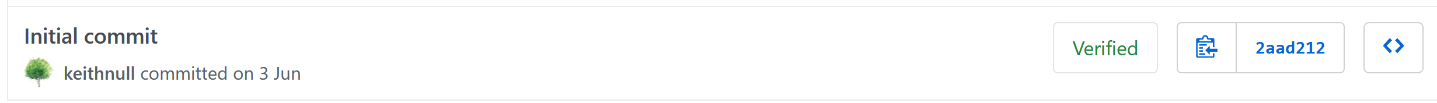
# 在Github上使用GPG的全过程

## 起因

其实在很早之前 Github 就已经充分支持 GPG 密钥了，而在我之前使用 Github 的两年时间内，竟对此一无所知，实在有些“没见过世面”。直至近日，在一次偶然查看仓库的commit历史中，发现某些commit有一个不同寻常的绿色标记（Verified），不仅美观（？）而且看上去舒心，如图所示：



点击这个标记，得知这一次commit是经过签名验证的（signed with a verified signature），因此，我便开始研究如何利用GPG对自己的每次commit进行签名验证。

## 什么是GPG

GnuPG is a complete and free implementation of the OpenPGP standard as defined by [RFC4880](https://link.zhihu.com/?target=https%3A//www.ietf.org/rfc/rfc4880.txt)(also known as *PGP*). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories. GnuPG, also known as *GPG*, is a command line tool with features for easy integration with other applications.

以上是从[GPG网站](https://link.zhihu.com/?target=https%3A//gnupg.org/)上摘取的部分简介，总的来说，GPG的功能十分丰富，然而我这次主要是用它来对Git中的commit进行签名验证，所以需要做的事情也不算太复杂：

1. 生成自己的GPG密钥
2. 关联GPG公钥与Github账户
3. 设置利用GPG私钥对commit进行签名
4. 可选步骤：信任Github的GPG密钥

## 过程

### 安装GPG

由于我的目的是在Git中使用GPG，而Windows版本的Git发行包中，已经包含了可用的GPG命令行。判断方法也很简单，打开Git Bash，输入gpg --version，可以看到类似的GPG版本信息：

$ gpg --version

gpg **(**GnuPG**)** 2.2.16-unknown

libgcrypt 1.8.4

Copyright **(**C**)** 2019 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

Home: /c/Users/---/.gnupg

Supported algorithms:

Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA

Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,

CAMELLIA128, CAMELLIA192, CAMELLIA256

Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224

Compression: Uncompressed, ZIP, ZLIB, BZIP2

不过需要说明的是，如果所安装的Git版本比较久远（比如我一开始所用的Git发行包是2017年的），那么很可能其包含的GPG版本过低，影响后续的操作，建议直接更新所安装的Git发行。

### 生成自己的GPG密钥

打开Git Bash，运行gpg --full-generate-key，根据提示，输入相应的个人信息（**需要注意的是邮箱必须要使用在Github中验证过的邮箱**）、自定义密钥参数、设置私钥密码等等，即可生成自己的GPG密钥。（补充说明，使用gpg --gen-key亦可生成密钥，但是会略去自定义密钥参数的步骤，对于一般场合的使用问题不大。）

输出结果的末尾大致如下：

gpg: key DC3DB5873563E6B2 marked as ultimately trusted

gpg: revocation certificate stored as '/c/Users/---/.gnupg/openpgp-revocs.d/1BA074F113915706D141348CDC3DB5873563E6B2.rev'

public and secret key created and signed.

pub rsa2048 2019-08-04 **[**SC**]** **[**expires: 2021-08-03**]**

1BA074F113915706D141348CDC3DB5873563E6B2

uid fortest <test@test.com>

sub rsa2048 2019-08-04 **[**E**]** **[**expires: 2021-08-03**]**

需要记下的，是上述输出信息中的密钥ID：1BA074F113915706D141348CDC3DB5873563E6B2 或者DC3DB5873563E6B2，后者是前者的简短形式。

当然，如果没有及时将其记下也不要紧，可以运行gpg --list-keys，列出本地存储的所有GPG密钥信息，大致如下：

$ gpg --list-keys

*# some output is omitted here*

pub rsa2048 2019-08-04 **[**SC**]** **[**expires: 2021-08-03**]**

1BA074F113915706D141348CDC3DB5873563E6B2

uid **[**ultimate**]** fortest <test@test.com>

sub rsa2048 2019-08-04 **[**E**]** **[**expires: 2021-08-03**]**

稍微解读一下这些结果：

* pub其后的是该密钥的公钥特征，包括了密钥的参数（加密算法是rsa，长度为2048，生成于2019-08-04，用途是Signing和Certificating，一年之后过期）以及密钥的ID。
* uid其后的是生成密钥时所输入的个人信息。
* sub其后的则是该密钥的子密钥特征，格式和公钥部分大致相同（E表示用途是Encrypting）。

### 关联GPG公钥与Github账户

还记得在上一步中记下的密钥ID吗？现在，我们需要根据这个ID来导出对应GPG密钥的公钥字符串。继续在Git Bash中，运行命令gpg --armor --export {key\_id}:

$ gpg --armor --export 1BA074F113915706D141348CDC3DB5873563E6B2

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF1GT3wBCADC9Hb3HtDc69XzYlohVKvdL1KnK0FslJISRuF6S0sdoOiWo2wJ

OiYVplWguTSkrMytjnMsoysZVolkYluY1wk67NT8YuYfnu6LSuF/doihrRldnKmz

9NZWw+15MLnENKsWCtwNwcCGDeZNJACyyUMYk7nJeIiM72k3/rnsyEpHqB25W/Zf

1VBkwf/ShePZ2W+rUktJ8j1TZuxe2bQpJdHQ9EKWG50D8O3xk+N+xEg4pcXLMfwT

vnVpf2wINGLA6+3ypVMDipC0fgAnINBrrjiKsq2Sskv0O73D3sZlkOi0jgAhx+21

5dI2xHbcs3DrcZbWAF1xEA8wGsoyYQWoSCBrABEBAAG0F2ZvcnRlc3QgPHRlc3RA

dGVzdC5jb20+iQFUBBMBCAA+FiEEG6B08RORVwbRQTSM3D21hzVj5rIFAl1GT3wC

GwMFCQPCZwAFCwkIBwIGFQoJCAsCBBYCAwECHgECF4AACgkQ3D21hzVj5rJ3BQgA

nUusNKaf8SIWq1w4ZR6CKhZP+kz+5kOEBs3+qIXJV++9nbjs4jnqOnXJUUdpLS9E

HGYnd6XSeyqWmBAuFCcmld4VGIajYxgDbF11/ql5Gnbu26/jV7hnrBBK6Xn/6oV9

bBmLoT9xget5xFC6g2VE0EvneRqacUgMBCkvrMzcVnHmpkSOpjfXRAItnyK/bhia

8k/+5URO8v7Ao2+QO0zk8XzgGc5B8H/yItzDiKe7gpzdUyCviG8m/tkDUURzloY4

09wCmQWWzerbBHJT4RdpPqdTEtC6f4jTuT32zp5NtLpJ740WmSJly/8nAJ/0x3Vf

pVkzhsg9gVHe/JSFa6/hXbkBDQRdRk98AQgAyjXZ98VOgftRThuGuYxKhqahonLf

Ihu+NuNMFG6sGGzkm2T+1i4uKyM8T/kGdcTzTXE/SMHmrCMz94FNcQ77/OFLz5HY

8hjaz5Sun7iNmz5HGct8OrsP6gQeJ5ucqm3vDZmnwU/+J+wcTosv5mgWoBVob7jb

PBnoNVBQSVhD2ek4CDljn3PdReqYfe+ee8yn+6K1t9c1HHHMco3WpdgofUABd+7l

Q1LF8IpBRDvWgdMciAPaSthIqFT6R6xLQhXV8SUm0mr2/GXbYqIptjvy1JmUwNk3

jE3LOLYulZChRdvVg3Y+xgkVlMYLy3SBQ1EaTnUUsGYbhGQnOwDwVAlxgwARAQAB

iQE8BBgBCAAmFiEEG6B08RORVwbRQTSM3D21hzVj5rIFAl1GT3wCGwwFCQPCZwAA

CgkQ3D21hzVj5rLzvAf/QzfDOrhRz9AVLiAqus3Z/WfZY81sUiewNM+YdV9aODht

q4VE92SYHeR/b72+Fl62SRbDqxw7qG5FJGByuqo6nJjHEpnFzqB/pepTVDzlwvdn

JO46tmepFAChPBpeTTjTs2CF/BG0As0KxXQCpdFw4m8UdkZ7Olt1/LKnXrFmr1BA

jp2MvmAo38j2RyPTyXKWmJW+vC8DwmOGMoHCL6fM0TeaWey3rNxST7bbxPdRVc4Z

/26k450FEW5D+VInb9NuFYSoE2UXs6DgI1OWuuGvWePrtXHeQvuNbGdEdUwU14mf

msQ78G2MjX4AAYR5iNnQ/IWDBKbOWt3ajIoJuebArw**==**

**=**oHpZ

-----END PGP PUBLIC KEY BLOCK-----

然后，在Github的SSH and GPG keys中，新增一个GPG key，内容即是上述命令的输出结果。

再次提醒，GPG密钥中个人信息的邮箱部分，必须使用在Github中验证过的邮箱，否则添加GPG key会提示未经验证。

### 利用GPG私钥对Git commit进行签名

首先，需要让Git知道签名所用的GPG密钥ID：

git config --global user.signingkey **{**key\_id**}**

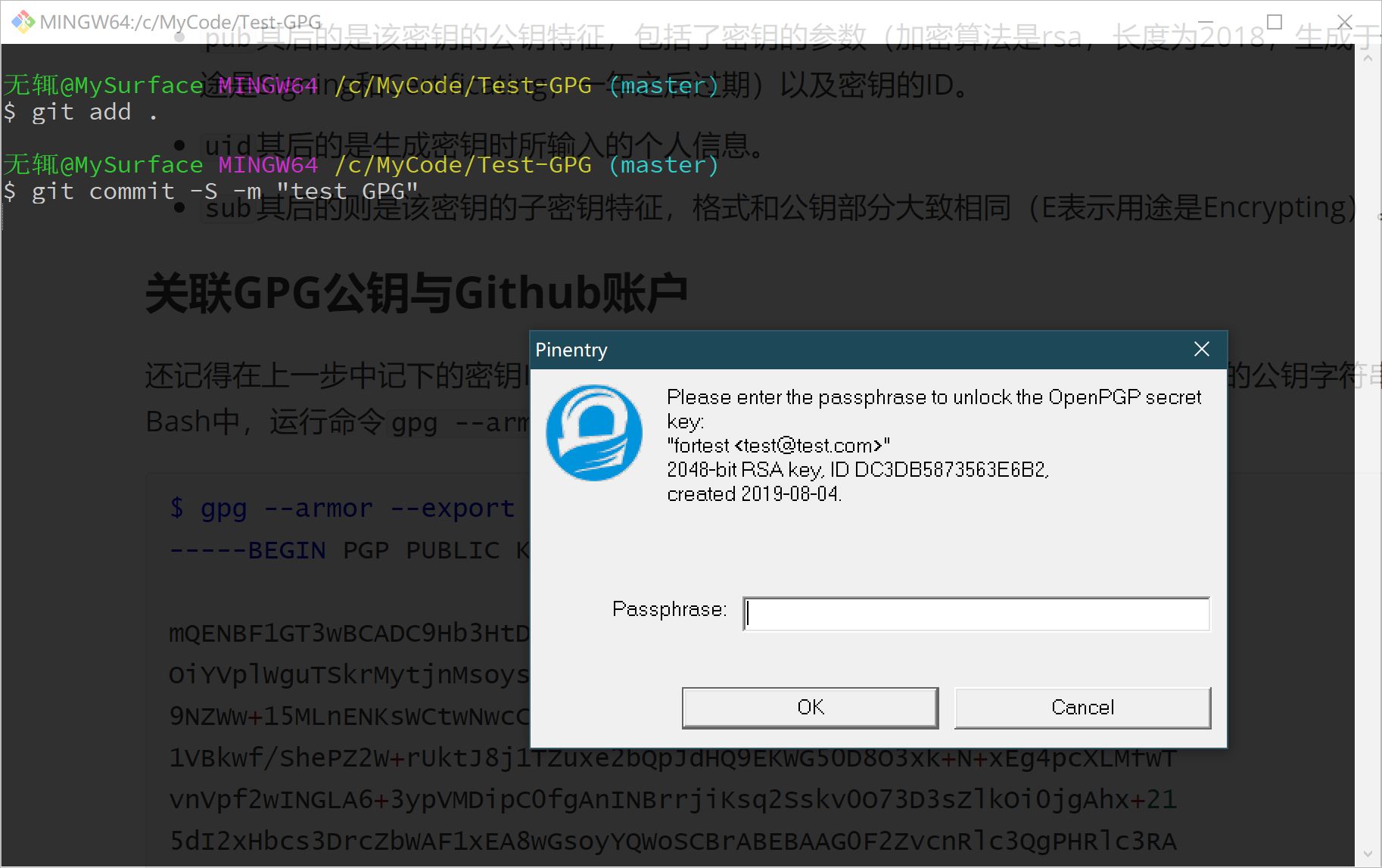
然后，在每次commit的时候，加上-S参数，表示这次提交需要用GPG密钥进行签名：

git commit -S -m "..."

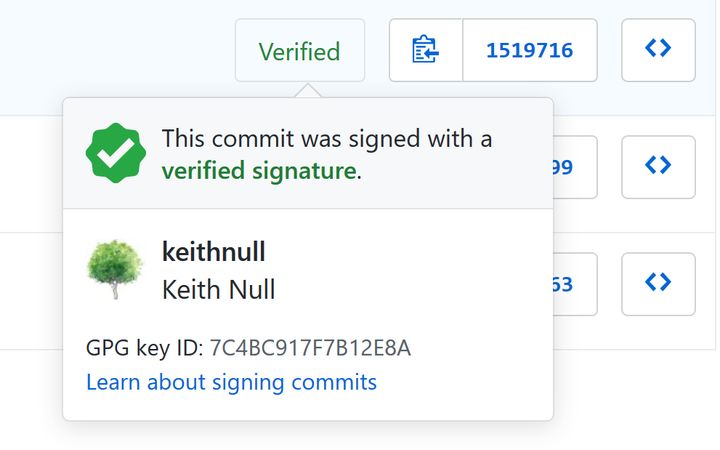
如果觉得每次都需要手动加上-S有些麻烦，可以设置Git为每次commit自动要求签名：

git config --global commit.gpgsign true

但不论是否需要手动加上-S，commit时皆会弹出对话框，需要输入该密钥的密码，以确保是密钥拥有者本人操作，如图所示：



输入正确密码后，本次commit便被签名验证，push到Github远程仓库后，即可显示出Verified绿色标记（由于fortest <test@test.com>密钥的邮箱未经验证，所以此处实际用的是我本人的密钥进行签名）：



## 可选步骤：信任Github的GPG密钥

事实上，在完成上述步骤后，已经可以**基本**完全正常地同时使用Github和GPG了，那为什么还需要这一步骤呢？很简单，不妨用git log --show-signature试试查看本地的某个Git仓库的commit记录和签名信息：

$ git log --show-signature

*# some output is omitted*

commit ec37d4af120a69dafa077052cfdf4f5e33fa1ef3 **(**HEAD -> master**)**

gpg: Signature made 2019年08月 4日 12:52:29

gpg: using RSA key 1BA074F113915706D141348CDC3DB5873563E6B2

gpg: Good signature from "fortest <test@test.com>" **[**ultimate**]**

Author: keithnull <keith1126@126.com>

Date: Sun Aug 4 12:52:29 2019 +0800

test GPG

commit 6937d638d950362f73bfbf28bc4a39d1700bf26b

gpg: Signature made 2019年07月24日 15:58:46

gpg: using RSA key 4AEE18F83AFDEB23

gpg: Can't check signature: No public key

Author: Keith Null <20233656+keithnull@users.noreply.github.com>

Date: Wed Jul 24 15:58:46 2019 +0800

Initial commit

可以发现，虽然所有的commit在Github中查看都是Verified，但是有一些比较特殊：在Github网页端进行的操作，比如创建仓库。这些commit并没有用我们之前生成的密钥进行签名，而是由Github代为签名了。这样的结果就是，我们本地无法确认这些签名的真实性。

为了解决这个问题，我们需要导入并信任[Github所用的GPG密钥](https://link.zhihu.com/?target=https%3A//github.com/web-flow.gpg)。

先是导入：

$ curl https://github.com/web-flow.gpg | gpg --import

*# curl's output is omitted*

gpg: key 4AEE18F83AFDEB23: public key "GitHub (web-flow commit signing) <noreply@github.com>" imported

gpg: Total number processed: 1

gpg: imported: 1

然后是信任（用自己的密钥为其签名验证，需要输入密码）：

$ gpg --sign-key 4AEE18F83AFDEB23

pub rsa2048/4AEE18F83AFDEB23

created: 2017-08-16 expires: never usage: SC

trust: unknown validity: full

**[** full **]** **(**1**)**. GitHub **(**web-flow commit signing**)** <noreply@github.com>

pub rsa2048/4AEE18F83AFDEB23

created: 2017-08-16 expires: never usage: SC

trust: unknown validity: full

Primary key fingerprint: 5DE3 E050 9C47 EA3C F04A 42D3 4AEE 18F8 3AFD EB23

GitHub **(**web-flow commit signing**)** <noreply@github.com>

Are you sure that you want to sign this key with your

key "Keith Null <keith1126@126.com>" **(**7C4BC917F7B12E8A**)**

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

至此，再尝试查看本地仓库的commit签名信息，则会发现所有的commit签名都已得到验证：

$ git log --show-signature

*# some output is omitted*

commit 6937d638d950362f73bfbf28bc4a39d1700bf26b

gpg: Signature made 2019年07月24日 15:58:46

gpg: using RSA key 4AEE18F83AFDEB23

gpg: Good signature from "GitHub (web-flow commit signing) <noreply@github.com>" **[**full**]**

Author: Keith Null <20233656+keithnull@users.noreply.github.com>

Date: Wed Jul 24 15:58:46 2019 +0800

Initial commit

## 结束

经过这一番操作，Github和GPG圆满结合在了一起，而我也得到了我想要的Verified标记。不过，GPG的功能远非止于此，它还可以用来对文件、邮件等进行加密，还可以进行身份验证等等，都有待我去学习研究。