

# Lab Exercise 4- Signed Commits in Git and GitHub

## Objective:

To configure Git to sign commits with GPG, push them to GitHub, and verify commit authenticity for secure code contribution.

---

## Prerequisites:

- Git installed on your system
  - GPG (GNU Privacy Guard) installed and configured
  - GitHub account with a repository (you own or have write access to)
  - Basic knowledge of Git commands
- 

## Step 1 – Generate or Use an Existing GPG Key

### 1. Check for existing keys

```
gpg --list-secret-keys --keyid-format=long
```

## 2. If no key exists, generate a new one

**gpg --full-generate-key**

```
jayde@ASUS-16x MINGW64 ~
gpg --list-secret-keys --keyid-format=long
gpg: directory '/c/Users/jayde/.gnupg' created
gpg: /c/Users/jayde/.gnupg/trustdb.gpg: trustdb created

jayde@ASUS-16x MINGW64 ~
gpg --full-generate-key
gpg (GnuPG) 2.4.5-unknown; Copyright (C) 2024 g10 Code GmbH
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
  (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 keysizes do you want? (3072) 4096
Requested keysizes 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

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

Real name: JAYDEV-GIT
Email address: jaydevswain44@gmail.com
Comment: first-gpg-key
You selected this USER-ID:
    "JAYDEV-GIT (first-gpg-key) <jaydevswain44@gmail.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
risks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
Warning: We need to generate a lot of random bytes. It is a good idea to perf
orm
some other action (type on the keyboard, move the mouse, utilize the
risks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: directory '/c/Users/jayde/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/c/Users/jayde/.gnupg/openpgp-revocs.d/1C
60CF300043DC6BFC41B25441DE1C2C4CB9345.rev'
Public and secret key created and signed.

pub   rsa4096 2025-08-20 [SC]
      1CD60CF300043DC6BFC41B25441DE1C2C4CB9345
uid           JAYDEV-GIT (first-gpg-key) <jaydevswain44@gmail.com>
sub   rsa4096 2025-08-20 [E]
```

- Select **RSA and RSA**
- Key size: **4096**
- Expiration: **0** (never) or a fixed date
- Enter your **GitHub-registered name and email**

### 3. Get your key ID

```
gpg --list-secret-keys --keyid-format=long
```

```
jayde@ASUS-16x MINGW64 ~  
$ gpg --list-secret-keys --keyid-format=long  
gpg: checking the trustdb  
gpg: marginals needed: 3 completes needed: 1 trust model: pgp  
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u  
[keyboxd]  
-----  
sec   rsa4096/441DE1C2C4CB9345 2025-08-20 [SC]  
      1CD60CF300043DC6BFC41B25441DE1C2C4CB9345  
uid           [ultimate] JAYDEV-GIT (first-gpg-key) <jaydevswain44@gmail.com>  
ssb   rsa4096/A0AFE2F6FA631423 2025-08-20 [E]  
-----  
j...@ASUS-16x MINGW64 ~
```

Example output:

```
sec rsa4096/3AA5C34371567BD2 2025-08-13 [SC]
```

Here, 3AA5C34371567BD2 is your key ID.

---

### Step 2 – Add GPG Key to GitHub

1. Export your public key:

```
gpg --armor --export YOUR_KEY_ID
```

```
jayde@ASUS-16x MINGW64 ~
$ gpg --armor --export YOUR_KEY_ID
gpg: WARNING: nothing exported

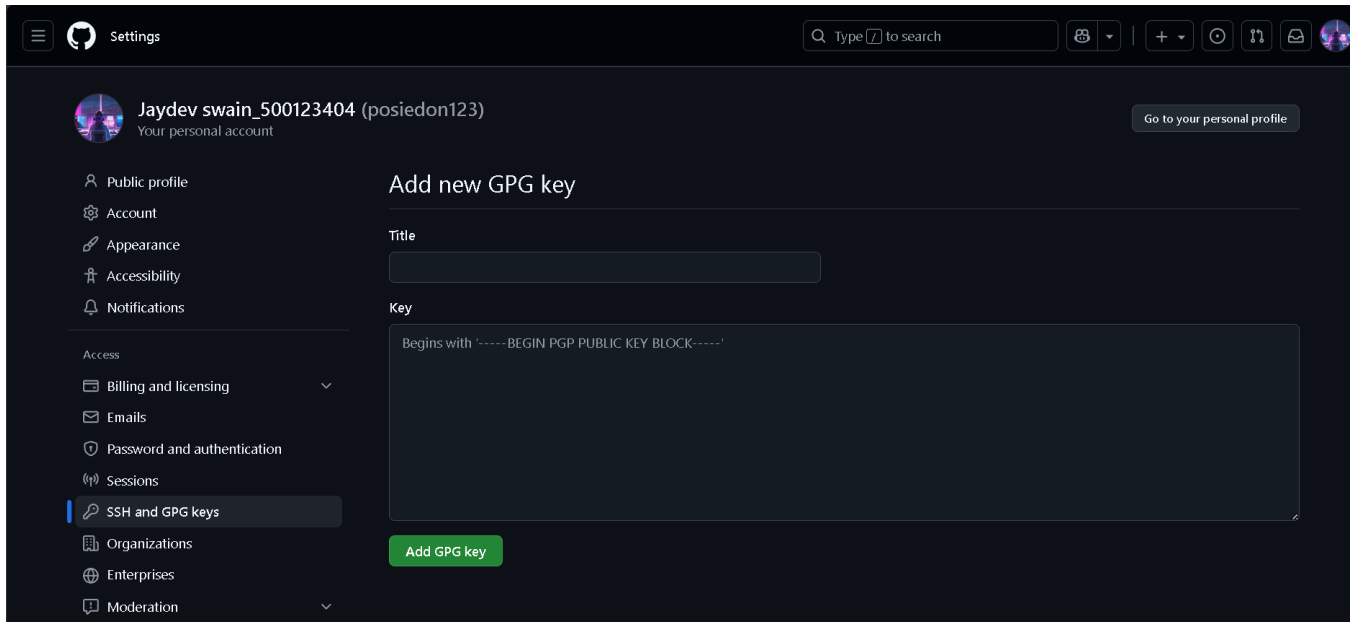
jayde@ASUS-16x MINGW64 ~
$ gpg --armor --export 441DE1C2C4CB9345
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBGimMHsBEADdNQ0dkKeZyT2oAi/Yz/yxQ22m09yyKkbXfe77Tvy9h13nWHEH
lNjsOZ3Zof7IxUq6TaavQ2XGyUcghzDYn8XvtRIXK3j+6sEQEQoz6/khiDjUfUBg
dsf4zL3gtrkctxAbNiDu14NF6cb3eBVlz/KMQRznf+T3/o4qcCaNiftgIgjtpCFH
gm7MEW7TBj9NK08KK5Le3gewjctee1j+wwqjjNf6w5MTsv65XT/vRayQK89zvU6h
vsFaeROP1rir4OLn12Vt3o/vFbqzGEOQ39Yr4isNaPzbdwGPe+qsd5vJPsa4agUN
eEYHwh7xxN2r0zQe6a9ADi7Mm5r37UbbyqL41RstPlYvFP0p3DnsJYnfaX4+3cIe
LSwPj6a53VC90QpKBgviA857DSmxObEGs1AIt35mxztuQlPXbbsEajm2CHc+eHMw
Mxl68Zwrzj5eq+eUSYJhgwUwi fMoJ5v4430Xgb4upENURwZYm0AmJLhu/wozLsX
TTi00pL1iair+kC2I4a8udKIx7LUGn6w3oS1NsgrIIBCSE5hAGjXzz/sFH60mMjg
A3HFpuYovXLVI/RL/oe/U2XxIqHNXJwpqYSMxQyC1UvQ0mgfjghIaCO2k0WgpdNj
eFBPLjmZ2H9GFMjiVF33hiubVLu0Ww0r7/Jyclp9bwRU9UhZSaGOapEzwARAQAB
tDRKQVlERVYtR0lUIChmaXJzdC1ncGcta2V5KSA8amF5ZGV2c3dhaw40NEBnbWpFp
bc5jb20+iQJRBMBCAA7FiEEHNYM8wAEPca/xBs1RB3hwsTLk0UFamiMHsCGwMF
CwkIBwICIGIGFQoJCAsCBBYCAwECHgcCF4AACgkQRB3hwsTLk0WCTxAAnpJfyhYE
E00+yBayfBjUxmavrnTjGggGkRdbzZgrnx3CEXv1Gn8QBEomePw9Nr0ypXwRkPqo
2bBIZdwSKIqfPOMjpgNlU2ibiVvva08LxUoUiAtTGzb5zLjIZv8St01ORoREkqlq
rUZqnKvARvNl+lJmua2ZqrCpDxlC+WnoPbN4KUd7bbgjbciMR8ZlF1bXsUj8wX0y
G1Dl09iA6cSRwMXM8kL/1wdxe+/ezkpdua+rDLwjKPNeymEk9fRB/zU4a+vJdBIP
qz8y3Pw6qR+cKSy7cZa2Q6/qBDD0CSULAVCrpB3ay0xDcxbOGSjWms32GwlR4xiY
4+M63EepiCgh9EFTGSKp+skZnAeOSZoAP0/31VS9EEXUm4TNAwq7LiH9fk/U8qpo
hyAktJEXCFYimaAjQMh0sF0+3ybsrudciTvWxtBm8jYcVLK8gUZ87gAvgFwqf9CBE
UziVonn7RRWGV6UHK5uV+NCECwYoe7PqKHhmY29WnRv1kGx6gymLXTiDoKNvt5H
ZovJAdn/8LihRbXK0v4eGiSkhVW6CGCEZuKPiXURoig3vcu2wOCLG77KK7dB0Z41
NR3KyfCjTrbg2kvZIZqyM2Z1IbpnQIV7kcwBRYCj5pCGxongNFhKCUXFrRI8k8q3
OLZVZNNwz1NM70e1UAjctPhaxf/hRFehD3e5Ag0EaKYwewEQAkkMiYKX0ibc/If6
LTn8l9b+tlgbsYaut/9yQtCkJOqa40y7hcYBVFKw2fvqNdQdJ1P0iezv1/yeDrTG
XAlOsqu5LrdTtm4j11VRdJc9GbUwF6iil03wKU7Cow1OwofNySGLf1qPULHi+q+
ItNoi4N4pL9482Vuc6KlTYV/X2o5DpLvrd8p3JI6YzwXEdjD+x7fiynkouXt2Kud
PLNmFtj4Z2VGIVd1lKUCRQY5t/BUoTsGdNw2Z1KVs8Kjba+47ORJpAC+61ueEv2B
Ags+CdU3fhwrN5j7Tq4Wh1gXdSxt3FRo6LwvO3xIXmJkgP+IZn+E+NyVCQn5kbCE
6thKybzcxqC65pSHNljw9ZTwsqVIpdaqpZTt6N+h92BSOLFuwB4NqE7P1mqeXyDI
c2Vwbr4MTV0axYFP36SLGIqVsbgrYrifiNqjnZcYv4NqdoCqvSy10v1UHyNWpOuU
wePa3/AODIxOL9u78cKzLmx2Yg39KbPXQL35HAfdwKmrifFHcOYq3LX4YAjlFckM
aTd2lpKLOQu5IzVwmmKBKvU0gYmzM+U0GhJIuLkSTivNOM1Mlt8TmxKM3H1wtwDOQ
A/qaiH1G2azHSsAsuHZZE45pIhtj+9J0CknRk+lOajcHr2mZnUGNUMqYgpjDMYyi
v1+rNoAXy00kUfjkTpA5zGVtOVFFABEBAAGJAjYEGAETACAWIQQc1gzZAAQ9xr/E
GyVEHEHCxMuTRQUCAKYwewIbDAACKRBEHEHCxMuTRRx8D/wMBff5RxxH8LpCA1+UT
ETogWebPzuGfDeIB4R7zdJbgxmA6xikJ8yceJjy97oXeqQNqgome12/1H5Byw0BN
mzIJWB7KLhj187u+KKNSE3Ibqx5O9ewvTAKG9d0T5jPgH7o1JLwOlde2ow10tJhC
UFMIZN+XadLsisiz1lRxxg7vu+MKEPEdFkuLpyu9sxC6DY+Aw9wm3HuoJt2+p0St
x4Vw3pR/jHNlft9SNouAQEKIQHiEBFFLNIzGfYyZfc7RFLJSPxdqTfpvXXHXGxyU
6aNd1JRY0wNJDY+up3s7am7LuJcJCbQtEZ78zv1Wh1fFPRbcsGsdvOXigtr6mHS
DcaV7ZWwH81BvrX/NS/z8buJoNa8ihwgvSWBR+KHN4JlDxJv1oAX6xrNR0yYEvKk
qPDLeEGZYtT6E0FRpR/EPNv5JaiMUfAcgqdxfgNiX60hYXL6rn+3LmA/S84XAxsi
xzWft9I+1aj7L01Kw1fv5HS1Pwfl/MCrDaLJHnu/US9LaEaii1i4awjrvc9qhhBy
dw4GSV3Y1gzGCOlIvRQZY7nFTeWwryBvI5D1Caqt0SP0qaXk7Eego1lHxyDJ2ogq
07SCnY5zqhZPBhyx43ATLZWUXvQdT70r/TBfix3r9/ZFTY3YJXVLab5n17pps40z
8PwLdG3bFYLR0Y7iZvkTywDwug==
=Kcy9
```

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

```
jayde@ASUS-16x MINGW64 ~
$ |
```

2. Copy the output.
3. Go to **GitHub** → **Settings** → **SSH and GPG Keys** → **New GPG Key**.
4. Paste your key and save.



---

## Step 3 – Configure Git for Signed Commits

1. Tell Git which key to use:

```
git config --global user.signingkey YOUR_KEY_ID
```

2. Enable signing for all commits:

```
git config --global commit.gpgsign true
```

```
jayde@ASUS-16x MINGW64 ~
$ git config --global user.signingkey 441DE1C2C4CB9345

jayde@ASUS-16x MINGW64 ~
$ git config --global commit.gpgsign true

jayde@ASUS-16x MINGW64 ~
$ |
```

---

## Step 4 – Make a Signed Commit

1. Clone your repo (or use an existing one):

```
git clone https://github.com/<username>/<repository>.git
```

```
cd <repository>
```

```
jayde@ASUS-16x MINGW64 ~
$ git clone https://github.com/posiedon123/GPG-KEY
Cloning into 'GPG-KEY'...
warning: You appear to have cloned an empty repository.

jayde@ASUS-16x MINGW64 ~
$ |
```

2. Edit or create a file:

```
jayde@ASUS-16x MINGW64 ~
$ cd GPG-KEY

jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ echo "Secure commit test" >> secure.txt

jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ git add secure.txt
warning: in the working copy of 'secure.txt', LF will be replaced by CRLF the next time Git touches it

jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ git commit -S -m "Add secure commit test file"
[main (root-commit) 73able9] Add secure commit test file
1 file changed, 1 insertion(+)
create mode 100644 secure.txt

jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ |
```

3. Commit with signing:

```
git commit -S -m "Add secure commit test file"
```

4. Enter your GPG passphrase when prompted.

---

## **Step 5 – Push and Verify on GitHub**

1. Push the commit:

```
git push origin main
```

2. Go to your repository on GitHub → Click the commit → You should see a **green** “Verified” badge.

---

## Step 6 – Local Verification of Commit

```
git log --show-signature
```



```
jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ git log --show-signature
commit 73ab1e97594940d5d576a001397e278089b5f496 (HEAD -> main, origin/main)
gpg: Signature made Thu Aug 21 02:24:47 2025 IST
gpg:                using RSA key 1CD60CF300043DC6BFC41B25441DE1C2C4CB9345
gpg: Good signature from "JAYDEV-GIT (first-gpg-key) <jaydevswain44@gmail.com>" [ultimate]
Author: posiedon123 <jaydevswain33@gmail.com>
Date:   Thu Aug 21 02:24:47 2025 +0530

    Add secure commit test file

jayde@ASUS-16x MINGW64 ~/GPG-KEY (main)
$ |
```

This will display the GPG verification details locally.

---

## Use Case

Signed commits prevent identity spoofing in collaborative projects, ensuring only verified authors can make trusted changes in critical codebases.