

Lab Exercise 4- Signed Commits in Git and GitHub

Name – Garvit janu(b3 DevOps)

SAP ID -500119646

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
```

```
manis@Manish MINGW64 ~ (master)
$ gpg --list-secret-keys --keyid-format=long
manis@Manish MINGW64 ~ (master)
```

2. If no key exists, generate a new one

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

```
manis@Manish MINGW64 ~ (master)
$ gpg --full-generate-key
gpg (GnuPG) 2.4.7-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 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) 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: manish133144
Email address: manishkumar133144@gmail.com
Comment: for generating key
You selected this USER-ID:
    "manish133144 (for generating key) <manishkumar133144@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? (O)uit
Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? (O)
Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? 0
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.
manish133144We 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.
```

- o Select **RSA and RSA**

- Key size: **4096** ○

Expiration: **0** (never) or a fixed date

3. Get your key ID

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

```
manis@Manish MINGW64 ~ (master)
$ 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  rsa4096/67FCFD6BE34A25B6 2025-09-07 [SC]
    2433540A387BCE8FA20BBE2E67FCFD6BE34A25B6
uid            [ultimate] manish133144 (for generating key) <manishkumar133144@gmail.com>
ssb  rsa4096/912C2817DD8D68CC 2025-09-07 [E]
```

Example output:

- Enter your **GitHub-**

registered name and email

```
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:

```
nanis@Manish MINGW64 ~ (master)
$ gpg --armor --export 67FCFD6BE34A25B6

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

mQINBGi9XBABEAC1qea04GkDVmpcWmhwZjF0GzAxx8VePg0GuqPsox+Xl4U1qOR1
/sQJgJSJ2pBg0id0T20nZVQqEGN19/bxdjTIQMUXzP/e2UE1oZLtF/hKZqXzKy1A
twTjurCWokI4YDIPdp4lJ4jWdfQVvk9260o4jp1dcCZ19uH6h6gh+Ikt4Ug5eN
NRUHPLaC7b4BVXoVd8nUGdJvo/TyPKGjb1AVTUXjdwiRKwke08H0vPwbqHps06Qb
vhTKPyN8AGM2D+pV2SwMnpLvwssw3p+xk9FDEd6byMpndaTbPJLvvSXNrqxmaJg
JDUFc5XRwu3sk2rLOTpBXLEQ3focMVPt39kN5ZxqZ6dlrxnLDX5wJaPDZMp5wR
9RdkqLvm12g1Sbk+ZChywiWe9wDwxD+P35PLCRXkIRZR750Gp4fIuD793kc70EaF
<kw02g9tChRX7Yut60UIBKsSboKdt0Q4JM1ts8vr7surak8+PVb/QreK2uCgQ04i
RDBeetB6AEZKdwjqZppG13bwMyNPf7cF6XXL/rbj/dNfKYjsFFKwlGbnRXd1+oYm
w1s+fP9RWT/Kn5IRx+iLK2GfNDAAaTLsDzy7AT/sm2jsxipybPCob1dsjoIYSexRs
5jSDNkkZZtMUoT3xk5y8tGmU47qGxGznBUZMgL1NHQhwsRQVVQg7N+nHwWwARAQAB
tD9tYW5pc2gxMzmxNDQgKGZvcibNzW5lcmF0aw5nIGtTeSkgPG1hbm1zaGt1bwFy
MTMzMTQ0QGdtYwlsLmNvbT6JA1EEEwEIADsWIQQkM1QKOHv0j6ILvi5n/P1r40o1
tgUcaL1cEAibAwULCQgHAgiAgYVCgkICwIEFgIDAQIeBwIXgAAKCRBn/P1r40o1
tpd0EACoxMLNHclYEpThNgF3PaGrZDASYYbdkJ+PLPu47KgGiMB1DTFGCzaTmKv
zSGXmcD9fdwYIBAT0V1JooVV+4NQiI/8nZkuo0o0D8jtJGi3vnX1jr3123eci/v0
js+Hv5Pjtge8Amom5Gi8d1Q1tLB1w5Kw2YN5N6MEiWDZ7sMDhLNH5K6yE5v1ZAT0
zBWoe+J1XU+PdFHm5oRwlQeHU715JvtPIkE0YKcmbr3D+WuzBqtr6eGY3iir7Cb
+JmL6K/Q86mMww21QphU79+w90ZndJkvSwvNIhHPNscIrzS1luTPbzRepDwekaj
+1fmhdtXhTh6AniRWUHQ1QRM2WKgifi1RFTdqjsFis7Rtf0q/NBj+sKBGPkd6UB/8
WTywMoVnkG+jEp/J3z51WRAogdBOA1yIiULJPQAsYZ/tASbMNqw3UdAEg/ko1Dm
HRXIwfXqrie65EuK3jZTNB+nmG9SL1GQ0xIfHfM2oMrTrbkoItAHfJzRAtJtgaxh
```

2. Copy the output.
3. Go to **GitHub → Settings → SSH and GPG Keys → New GPG Key.**

The screenshot shows the GitHub account settings interface for the user 'manish133144'. On the left, there's a sidebar with various account management options like Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and licensing, Emails, Password and authentication, Sessions, SSH and GPG keys (which is selected and highlighted in blue), Organizations, Enterprises, and Moderation. The main area is titled 'Add new GPG key'. It has fields for 'Title' (set to 'New GPG KEY') and 'Key' (containing a long string of characters representing the public key). A green 'Add GPG key' button is at the bottom.

manish133144 (manish133144)
Your personal account

Public profile
Account
Appearance
Accessibility
Notifications

Access
Billing and licensing
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation

Add new GPG key

Title
New GPG KEY

Key

```
mQINBGi9XBABEAC1qea04GkDVmpcWmhWZjF0GzAxx8VePgOGuqPsox+Xl4UlqORI  
/sQJg/SJ2pBgOid0T20nZVQqEGNI9/bXdjTIQMUZxP/e2UEloZLtf/hKZqXzKy1A  
twTjurCWoWkl4YDlpdp4IJ4jWDfQVvk926Oo4jPldcCZ19uH6h6gh+lkt4Ug5eN  
NRUHPLaC7b4BVXoVd8nUGdJvo/TYpKGjblAVTUXjdwiRKwke08H0vPwbqHpsO6Qb  
YhtKPyn8AGM2D+pV2SwMNplvwssw3p+xk9FDEd6byMpnbabTbPJlvqSXNvrqxmAjg  
JDUFc5XRwu3sk2rlOTpBXLZEQ3focMVPT39kN5ZxqZ6dlrxnLDX5zWJaPDZMp5wR  
9RdkqLVm12glSbk+ZChywiWe9WDwxD+P35PLCRXkIRZRT5OGp4fluD793kc7OEaF  
kxWO2g9tChRX7Yut6OUIBKsSboKdtOQ4JMIts8vr7suraK8+PVB/QreK2uCgQO4i
```

Add GPG key

The screenshot shows the 'GPG keys' section of the GitHub account settings. It lists a single key entry:

New GPG KEY
Email address: manishkumar133144@gmail.com
Key ID: 67FCFD6BE34A25B6
Subkeys: 912C2817DD8D68CC
Added on Sep 7, 2025

A green 'New GPG key' button is located at the top right of this section.

GPG keys

New GPG key

Step 3 – Configure Git for Signed Commits

1. Tell Git which key to use:

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

```
manis@Manish MINGW64 ~ (master)
$ git config --global user.signingkey 67FCFD6BE34A25B6
```

2. Enable signing for all commits:

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

```
manis@Manish MINGW64 ~ (master)
$ git config --global commit.gpgsign true
```

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>
```

2. Edit or create a file:

```
echo "Secure commit test" >> secure.txt
```

```
git add secure.txt
```

```
manis@Manish MINGW64 ~/task-7 (main)
$ echo "Secure commit test" >> secure.txt

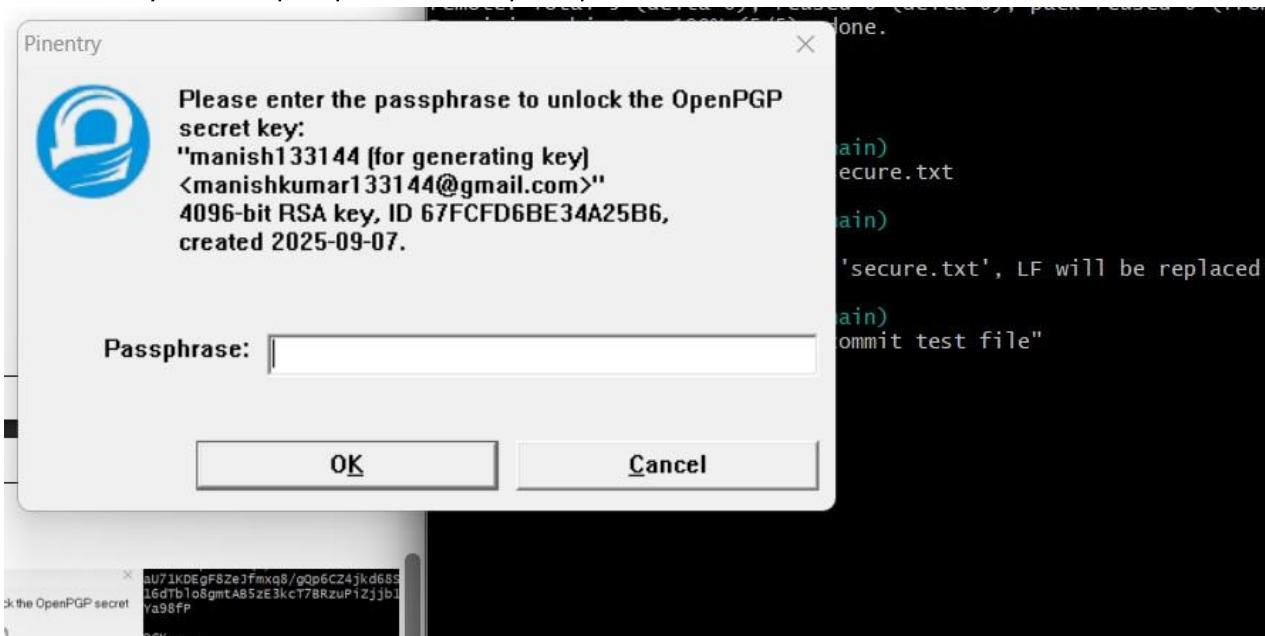
manis@Manish MINGW64 ~/task-7 (main)
$ git add secure.txt
warning: in the working copy of 'secure.txt', LF will be replaced by CRLF the next time Git touches it
```

3. Commit with signing:

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

```
manis@Manish MINGW64 ~/task-7 (main)
$ git commit -S -m "Add secure commit test file"
[main 57fa649] Add secure commit test file
 1 file changed, 1 insertion(+)
 create mode 100644 secure.txt
```

4. Enter your GPG passphrase when prompted.



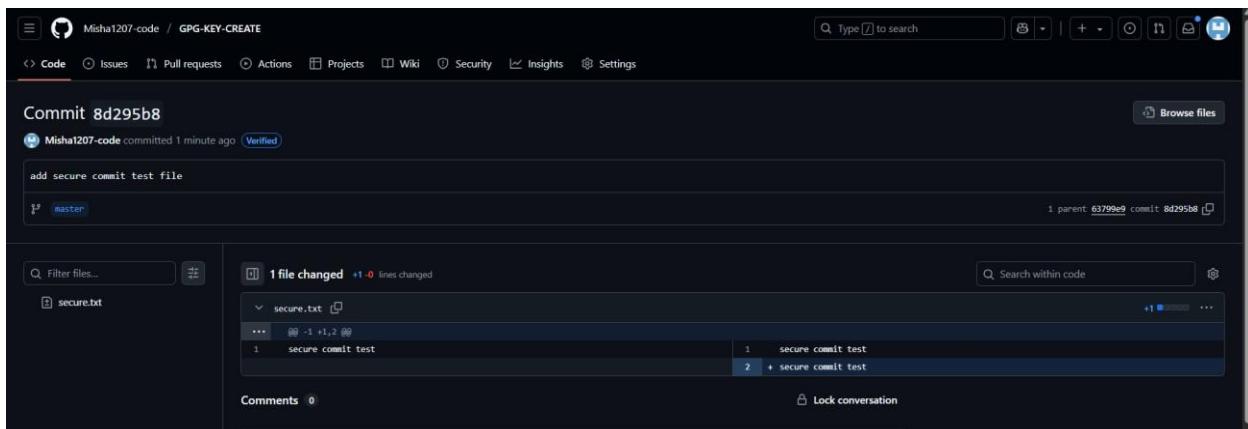
Step 5 – Push and Verify on GitHub

1. Push the commit:

```
git push origin main
```

```
mhanis@Manish MINGW64 ~/task-7 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 984 bytes | 492.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/manish133144/task-7.git
  93f2cee..57fa649  main -> 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
```

```
manis@Manish MINGW64 ~/task-7 (main)
$ git log --show-signature
commit 57fa64940c3b20a3aa12fa5e9953588f91de46fa (HEAD -> main, origin/main, origin/HEAD)
gpg: Signature made Sun Sep 7 16:28:18 2025 IST
gpg:           using RSA key 2433540A387BCE8FA20BBE2E67FCFD6BE34A25B6
gpg: Good signature from "manish133144 (for generating key) <manishkumar133144@gmail.com>" [ultimate]
Author: manish133144 <manishkumar133144@gmail.com>
Date:  Sun Sep 7 16:28:18 2025 +0530
```

This will display the GPG verification details locally.
