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
```

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

- 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

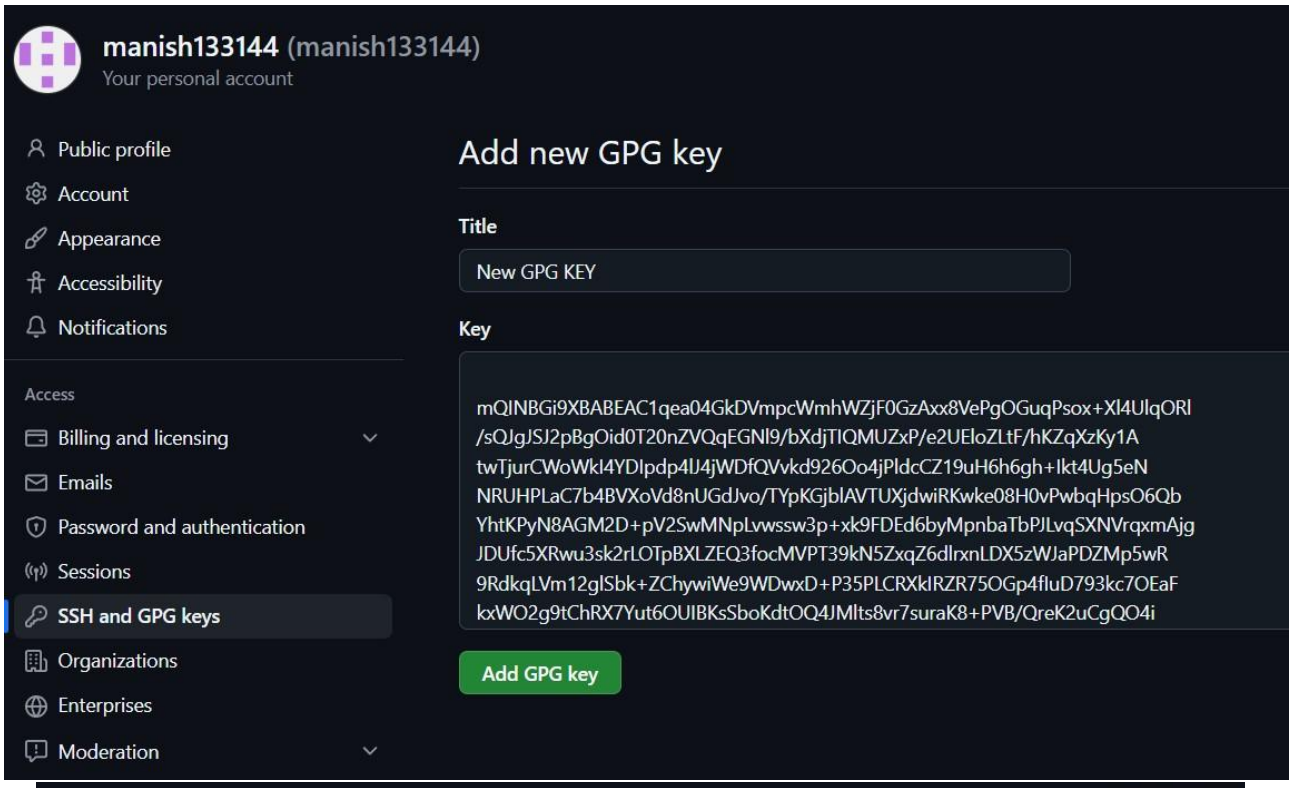
```
sec   rsa4096/3AA5C34371567BD2 2025-08-13 [SC] Here,

3AA5C34371567BD2 is your key ID.
```

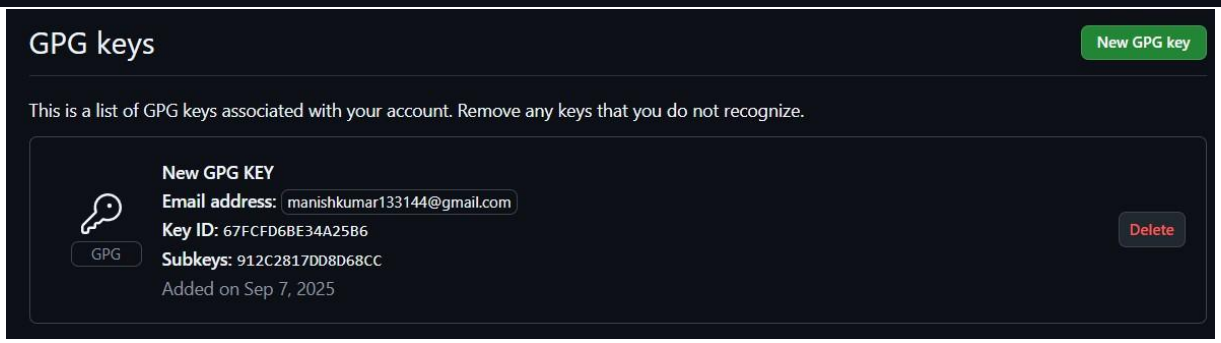
Step 2 – Add GPG Key to GitHub

1. Export your public key:

2. Copy the output.

3. Go to **GitHub → Settings → SSH and GPG Keys → New GPG Key**.


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+Xl4UlqORl
/sQJgJSJ2pBgOid0T20nZVQqEGNl9/bXdjTIQMUZxP/e2UEloZLtF/hKZqXzKy1A
twTjurCWoWkI4YDIpdp4lJ4jWDfQVvkd926Oo4jPldcCZ19uH6h6gh+Ikt4Ug5eN
NRUHPLaC7b4BVXoVd8nUGdJvo/TYpKGjblAVTUXjdwiRKwke08H0vPwbqHpsO6Qb
YhtKPyN8AGM2D+pV2SwMNpLvwssw3p+xk9FDEd6byMpnbaTbPJLvqSXNVrqxmAjg
JDUfc5XRwu3sk2rLOTpBXLZEQ3focMVPT39kN5ZxqZ6dlrxnLDX5zWJaPDZMp5wR
9RdkqLVm12glSbk+ZChywiWe9WDwxD+P35PLCRXkIRZR75OGp4fluD793kc7OEaF
kxWO2g9tChRX7Yut6OUIBKsSboKdtOQ4JMlts8vr7suraK8+PVB/QreK2uCgQO4i
Add GPG key


GPG keys
New GPG key
This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.
New GPG KEY
Email address: manishkumar133144@gmail.com
Key ID: 67FCFD6BE34A25B6
Subkeys: 912C2817DD8D68CC
Added on Sep 7, 2025
GPG
Delete

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
```

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
```

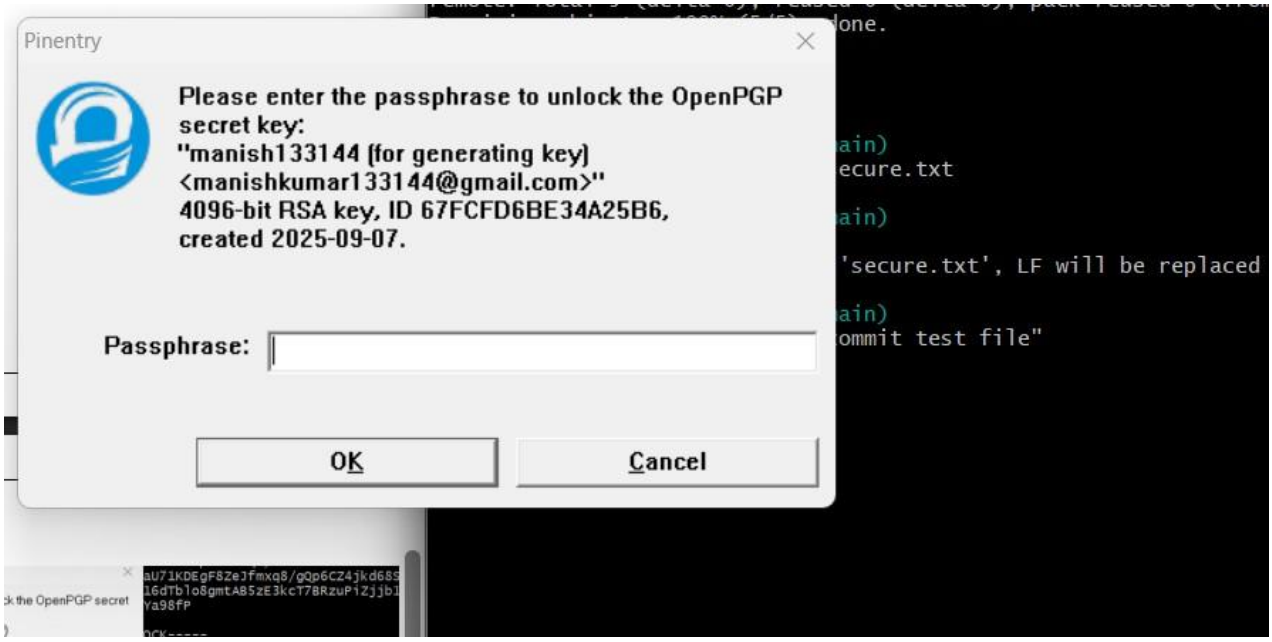
```
manis@Manish MINGW64 ~/task-7 (main)
$ echo "Secure commit test" >> secure.txt

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.



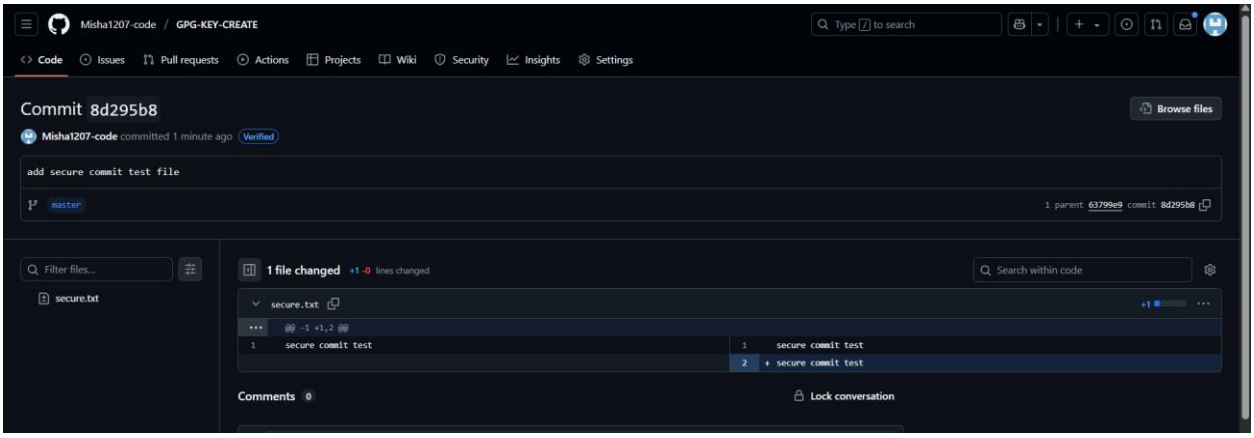
Step 5 – Push and Verify on GitHub

1. Push the commit:

git push origin main

```
anis@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

This will display the GPG verification details locally.