Lab Exercise 4- Signed Commits in Git and GitHub

Name – Misha (B2 DevOps)

SAP ID -500119679

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
```

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (master)
$ gpg --list-secret-keys --keyid-format=long
gpg: directory '/c/Users/Misha/.gnupg' created
gpg: /c/Users/Misha/.gnupg/trustdb.gpg: trustdb created
```

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

gpg --full-generate-key

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (master)
$ 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 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: Misha1207-code
Email address: mishu5705@gmail.com
Comment: for generating key
You selected this USER-ID:
    "Misha1207-code (for generating key) <mishu5705@gmail.com>"
```

- 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

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (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   rsa4096/9D4C24ECE449E2A6 2025-08-14 [SC]
      BCF6D1AF9D83DD2C0247AE5D9D4C24ECE449E2A6
uid                 [ultimate] Misha1207-code (for generating key) <mishu5705@gmail.c
om>
ssb   rsa4096/530CFBF0E24E819C 2025-08-14 [E]
```

Example output:

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

Here, 3AA5C34371567BD2 is your key ID.

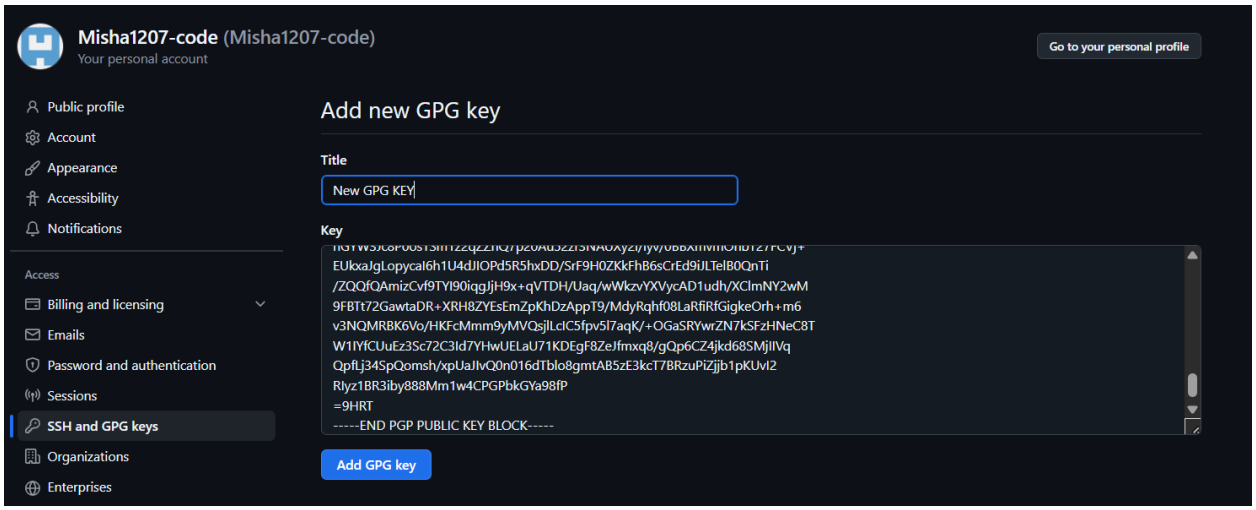
Step 2 – Add GPG Key to GitHub

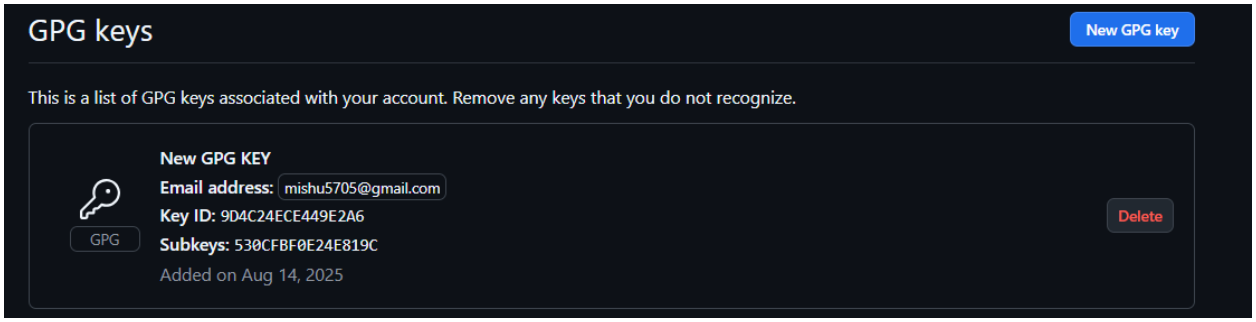
1. Export your public key:

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (master)
$ gpg --armor --export 9D4C24ECE449E2A6
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGidblABEACxWvhisWbjSMgpOl3L0cXznoSXqkoEagjDGcmMGqqdlu8Mq8n/
3UgWAbGiu3COtDVmDfnKCRB2yYjzlMAcPI2YI4OUtW5TyiGC4+Nmp/mrjylGmG/L
KD9b+pwRiWtVVttnem2y4HK6vXFhsLw2xdCZPf1utMz96QYofa1ESJdKfdqB6IPz
U5hGovqdtZVPhYAln7I/hmgenydkYEqIp/xoiqSOeDYr9f5TAVhsHB5hVaGkaVM3
Azj5uNN2pQi3vPN6IN27meycTA2Z4SlEvuRjTGi4fvfnmNEnLI3IP7+rFifmcU+X
MkO65A9VWWJLm6NLWK5JmHUvmHtOkh+x54HskaWFFCOQYdGZf6kEGK9kWBX4XevY
LL6p98E1GVEZKEWDlVQXyRJrFYy7kez9GJLCtPMma1Q2JzkkKHynr8ABTNWRQk5i
mB/PVWsg9FmWun49EcWGtJ+No/EbOUaBYscxLVAU4uxL+FiBYNILSX3bw2uycm2f
vRS9LqvQGLk97nbxYUXpgV4qVagse3a5mv2/lQUVSV37m8wfcDi4CqiG5zSChqOc
F4r1xp5/FC3Bsq0z9+9OcAh6/EO93kYKlvnaKU3V3IB1FATUUQ7abVIp9//PXiwG
3+c4e/ffSo7pKEOJfY8Nx8nANgZyWtzrd0Uo5zIZHGenZAAYE4WRyl5pvQARAQAB
tDlNaXNoYTEyMDctY29kZSAoZm9yIGdlbmVyYXRpbmcga2V5KSA8bWlzaHU1NzA1
QGdtYWlsLmNvbT6JAlEEEwEIADsWIQS89tGvnYPdLAJHrl2dTCTs5EnipgUCaJ1u
UAIbAwULCQgHAgIiAgYVCgkICwIEFgIDAQIeBwIXgAAKCRCdTCTs5EnippASEACL
```

2. Copy the output.

3. Go to **GitHub → Settings → SSH and 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
```

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (master)
$ git config --global user.signingkey 9D4C24ECE449E2A6
```

2. Enable signing for all commits:

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

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg (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
```

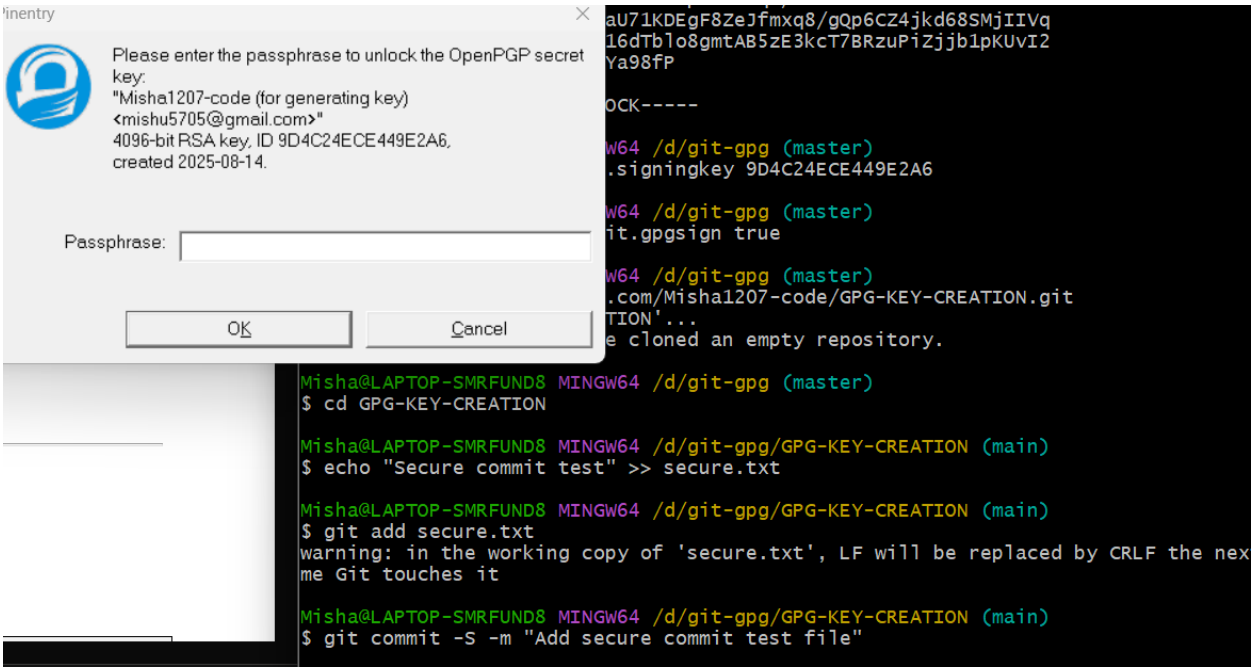
```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg-creation (master)
$ echo "secure commit test" >> secure.txt

Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg-creation (master)
$ 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"
```

```
Misha@LAPTOP-SMRFUND8 MINGW64 /d/git-gpg-creation (master)
$ git commit -S -m "add secure commit test file"
[master 8d295b8] add secure commit test file
 1 file changed, 1 insertion(+)
```

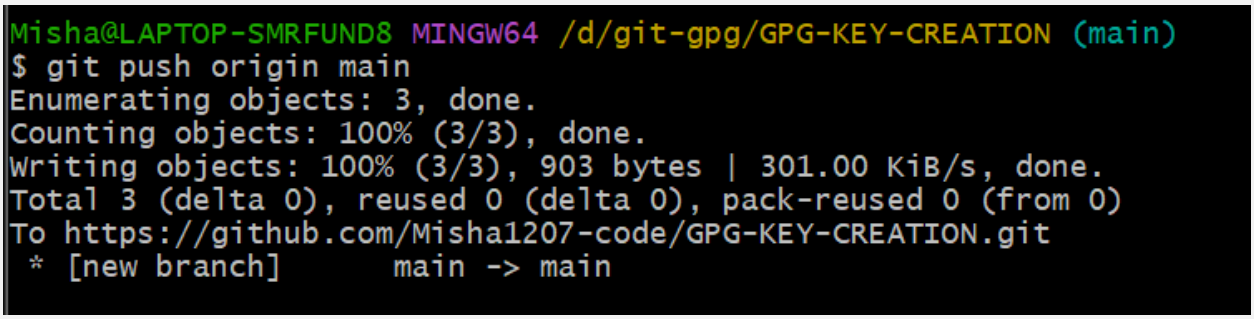
4. Enter your GPG passphrase when prompted.



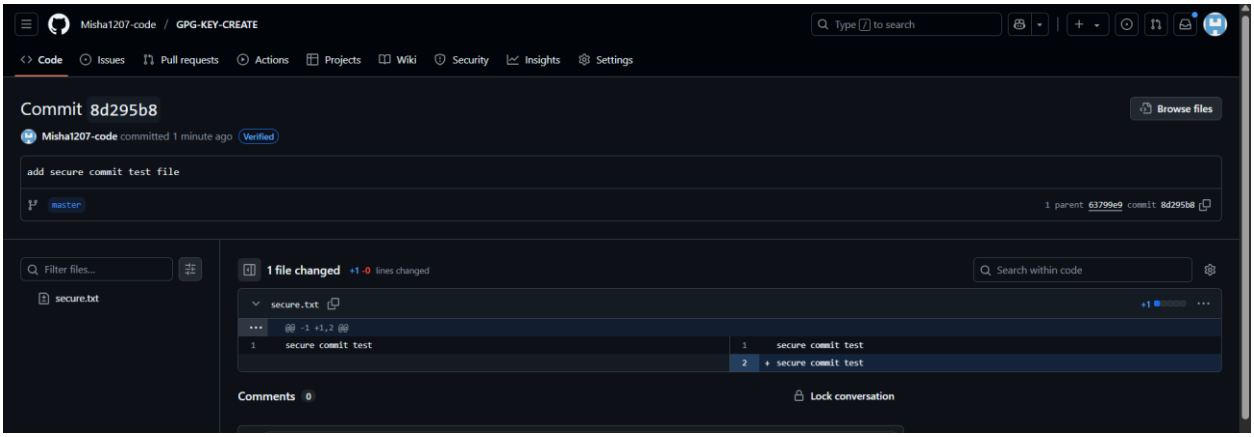
Step 5 – Push and Verify on GitHub

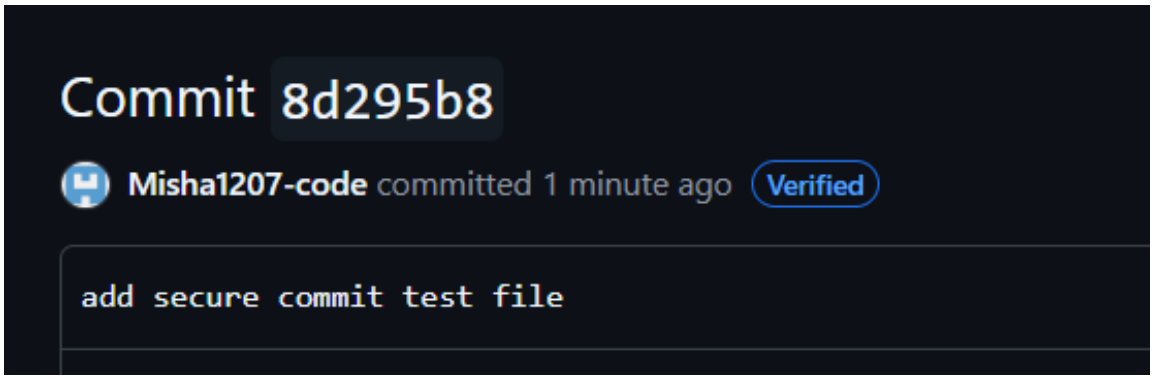
1. Push the commit:

```
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
```

```
$ git log --show-signature
commit 8d295b84ecff9727a07fee53e75164dbe52dcdd9 (HEAD -> master, origin/master)
gpg: Signature made Thu Aug 14 18:51:07 2025 IST
gpg:                using RSA key 1AC9E782CC1FD00B740964CEE7E03B9ED8AACEAB
gpg: Good signature from "Misha1207-code (NEWEST KEY) <mishu5705@gmail.com>" [ultimate]
Author: Misha1207-code <mishu5705@gmail.com>
Date:   Thu Aug 14 18:51:06 2025 +0530

    add secure commit test file
```

This will display the GPG verification details locally.
