Lab Exercise 4- Signed Commits in Git and GitHub

Name – Pratik Agrawal

Sap ID – 500123601

Batch – Devops B2

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

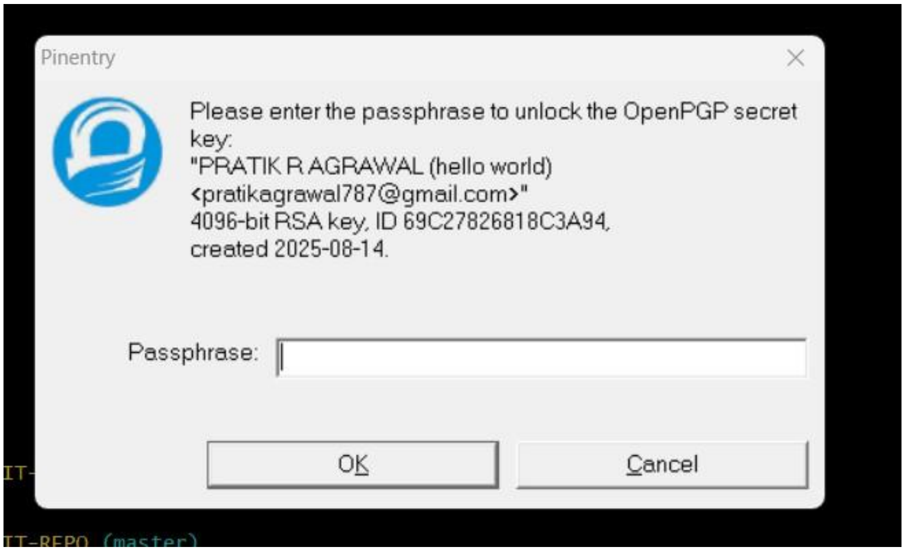
1. **Check for existing keys**

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

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

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

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



Please enter the passphrase to unlock the OpenPGP secret key:
"PRATIK R AGRAWAL (hello world)
<pratikagrawal787@gmail.com>"
4096-bit RSA key, ID 69C27826818C3A94,
created 2025-08-14.

Passphrase:

3. **Get your key ID**

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

Example output:

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

```
Here, 3AA5C34371567BD2 is your key ID.
```

Step 2 – Add GPG Key to GitHub

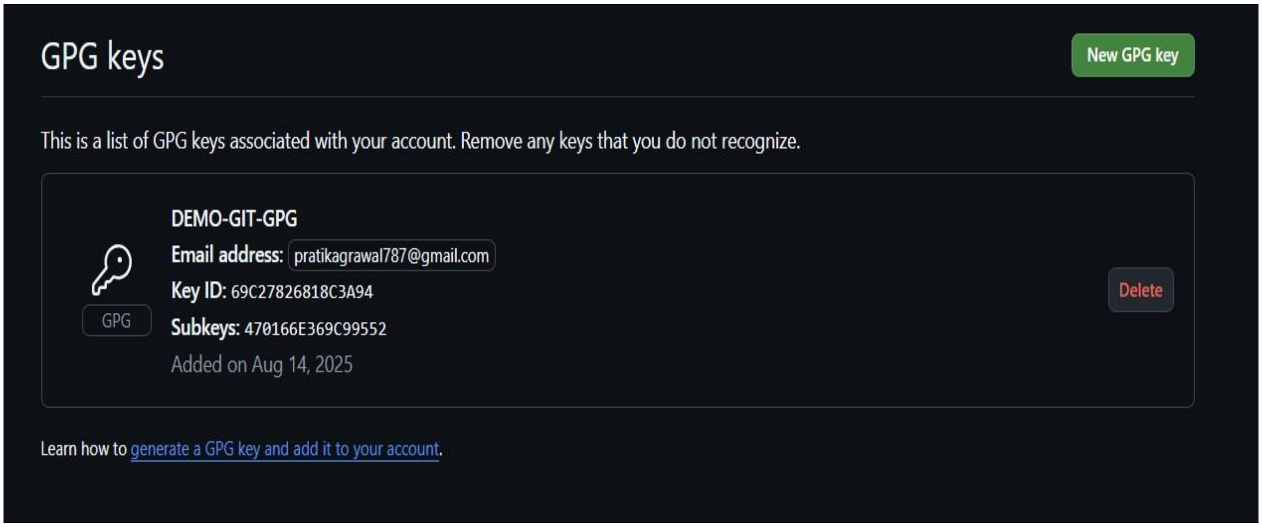
1. Export your public key:

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

2. Copy the output.

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

4. Paste your key and save.



GPG keys

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

DEMO-GIT-GPG

Email address: pratikagrawal787@gmail.com

Key ID: 69C27826818C3A94

Subkeys: 470166E369C99552

Added on Aug 14, 2025

Learn how to generate a GPG key and add it to your account.

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:

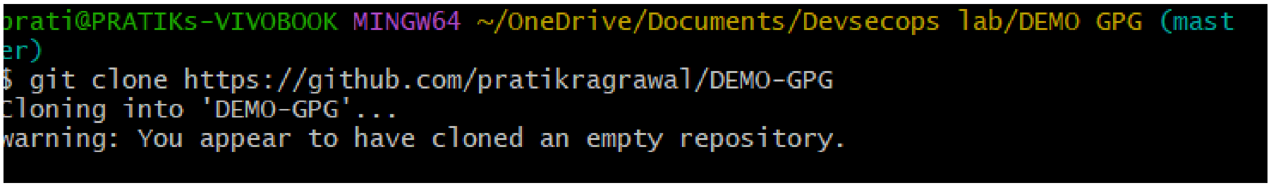
```
echo "Secure commit test" >> secure.txt

git add secure.txt
```

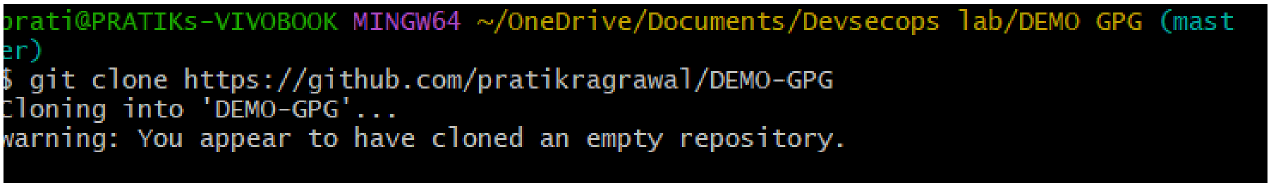
3. Commit with signing:

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

4. Enter your GPG passphrase when prompted.



```
prati@PRATIKs-VIVOBOOK MINGW64 ~/OneDrive/Documents/Devsecops lab/DEMO GPG (mast
er)
$ git clone https://github.com/pratikragrawal/DEMO-GPG
Cloning into 'DEMO-GPG'...
warning: You appear to have cloned an empty repository.
```



```
prati@PRATIKs-VIVOBOOK MINGW64 ~/OneDrive/Documents/Devsecops lab/DEMO GPG (mast
er)
$ git clone https://github.com/pratikragrawal/DEMO-GPG
Cloning into 'DEMO-GPG'...
warning: You appear to have cloned an empty repository.
```

Step 5 – Push and Verify on GitHub

1. Push the commit:

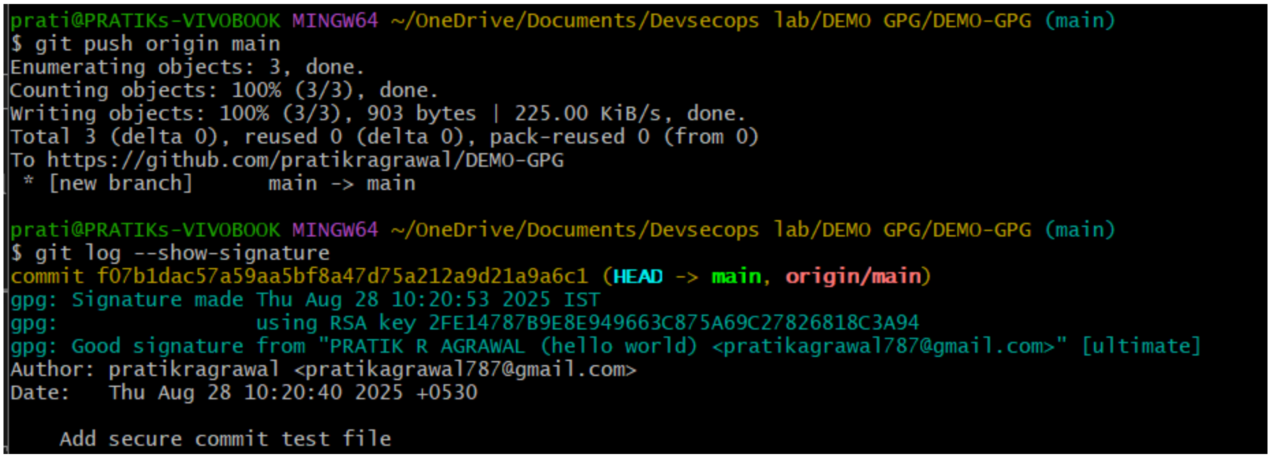
```
git push origin main
```

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
```

This will display the GPG verification details locally.



```
prati@PRATIKs-VIVOBOOK MINGW64 ~/OneDrive/Documents/Devsecops lab/DEMO GPG/DEMO-GPG (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 903 bytes | 225.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/pratikragrawal/DEMO-GPG
 * [new branch]      main -> main

prati@PRATIKs-VIVOBOOK MINGW64 ~/OneDrive/Documents/Devsecops lab/DEMO GPG/DEMO-GPG (main)
$ git log --show-signature
commit f07b1dac57a59aa5bf8a47d75a212a9d21a9a6c1 (HEAD -> main, origin/main)
gpg: Signature made Thu Aug 28 10:20:53 2025 IST
gpg:                using RSA key 2FE14787B9E8E949663C875A69C27826818C3A94
gpg: Good signature from "PRATIK R AGRAWAL (hello world) <pratikagrawal787@gmail.com>" [ultimate]
Author: pratikragrawal <pratikagrawal787@gmail.com>
Date:   Thu Aug 28 10:20:40 2025 +0530

    Add secure commit test file
```

Use Case

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