

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
```

- 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
```

Output:

```
shagu@Shagun MINGW64 ~/git-reset-lab/git-rebase-lab (feature-branch)
$ 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    rsa4096/85AE80ECCB87191B 2025-12-05 [SC]
      AFF02C7DB8577605407AF38085AE80ECCB87191B
uid          [ultimate] shagun (no) <shagungupta5002@gmail.com>
ssb    rsa4096/99EAA2592FAE71CD 2025-12-05 [E]
```

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.
-

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.

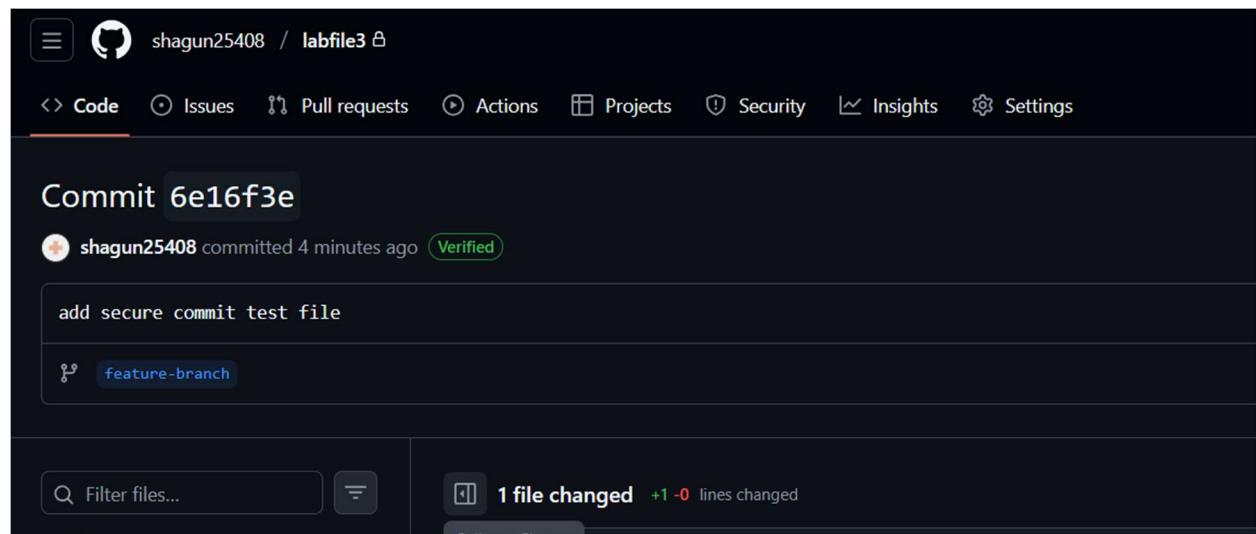
Step 5 – Push and Verify on GitHub

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**.

Output:



Step 6 – Local Verification of Commit

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

This will display the GPG verification details locally.

Output:

```
shagu@Shagun MINGW64 ~/git-reset-lab/git-rebase-lab (feature-branch)
$ git log --show-signature
commit 6e16f3ec57bb4f8a0e3cf850c395e6e872712d7f (HEAD -> feature-branch, origin/feature-branch)
gpg: Signature made Fri Dec 5 16:04:16 2025 IST
gpg:                 using RSA key AFF02C7DB8577605407AF38085AE80ECCB87191B
gpg: Good signature from "shagun (no) <shagungupta5002@gmail.com>" [ultimate]
Author: shagun25408 <shagungupta5002@gmail.com>
Date:   Fri Dec 5 16:04:16 2025 +0530

    add secure commit test file

commit 7d0c2df7469c91cba56dbeba9b268ad5e4564445 (origin/main, main)
Author: shagun25408 <shagungupta5002@gmail.com>
Date:   Fri Dec 5 13:46:22 2025 +0530

    add line 3 from main branch

commit 1b3fbf7188cde588006e617eaa31fa3be79ec7d9
Author: shagun25408 <shagungupta5002@gmail.com>
Date:   Fri Dec 5 13:45:00 2025 +0530

    add line 2
```

Use Case

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