Git Verified Commit Setup & Troubleshooting (Windows + Git Bash)

This document consolidates the **entire conversation** into a **single end-to-end guide** for setting up **verified (GPG-signed) commits**, fixing common errors, and complying with **organization rules that require verified signatures**.

---

1. What “Verified Commit” Means

A commit is marked **Verified** on GitHub when: - The commit is **cryptographically signed** using your **private GPG key** - The **public GPG key** is uploaded to GitHub - The **email in the commit matches the email in the GPG key** - GitHub can validate the signature after push

If your organization enforces this rule, **any unsigned or unverified commit will be rejected**.

---

2. Install GPG on Windows

Install **Gpg4win** (official): - Includes GnuPG + Kleopatra - Restart Git Bash after installation

Verify installation:

```
gpg --version
```

---

3. Generate a GPG Key

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

Recommended options: - Type: RSA and RSA - Size: 4096 - Expiry: 0 (never) - Name: Your real name - Email: **Must match GitHub email** - Passphrase: Required

List secret keys:

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

Example:

```
sec   rsa4096/ABCDEF1234567890
uid   Your Name <you@example.com>
```

Key ID = ABCDEF1234567890

---

4. Configure Git to Use GPG

Set signing key:

```
git config --global user.signingkey ABCDEF1234567890
```

Enable auto-signing:

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

Ensure email matches GPG key:

```
git config --global user.email "you@example.com"
```

---

5. Fix GPG Path Issue (Windows-specific)

Find actual gpg.exe:

```
where gpg
```

Typical correct path:

```
C:\Program Files\GnuPG\bin\gpg.exe
```

Configure Git:

```
git config --global gpg.program "C:/Program Files/GnuPG/bin/gpg.exe"
```

❌Do NOT use Git-bundled gpg.exe

---

6. Fix “No Secret Key” Error

Error:

```
signing failed: No secret key
```

Causes: - Wrong key ID in Git config - GPG key generated under different user - Wrong gpg.exe path

Fix:

```
git config --global --unset user.signingkey
git config --global user.signingkey ABCDEF1234567890
```

---

7. Export & Add Public Key to GitHub

```
gpg --armor --export ABCDEF1234567890
```

Copy full output and add to:

```
GitHub → Settings → SSH and GPG keys → New GPG key
```

---

8. Fix TTY Issue (Mandatory)

```
export GPG_TTY=$(tty)
```

Permanent:

```
echo 'export GPG_TTY=$(tty)' >> ~/.bashrc
```

Restart Git Bash.

---

9. Commit & Verify Locally

Create signed commit:

```
git commit -S -m "message"
```

Check signature:

```
git log --show-signature -1
```

Expected:

```
gpg: Good signature from "Your Name <email>"
```

---

10. Why Push Still Fails After Passphrase

Even if: - You entered passphrase - Git shows “Good signature”

GitHub may still show **Unverified** because: - Commit email ≠ GPG email - Public key not uploaded - Wrong key used - Older commits are unsigned

GitHub verifies **after push**, not locally.

---

11. Organization Error: “Commits must have verified signature”

Meaning: - At least **one commit in your branch is unverified** - Push is rejected

---

12. Re-sign Existing Commits (Most Important Fix)

Re-sign last commit

```
git commit --amend -S --no-edit
```

Re-sign multiple commits

```
git rebase -i HEAD~N
```

Change:

```
pick abc123
```

To:

```
edit abc123
```

Then for each commit:

```
git commit --amend -S --no-edit
git rebase --continue
```

Push rewritten history:

```
git push --force-with-lease
```

---

13. Dropping / Removing Commits

Remove last commit (local only)

```
git reset --soft HEAD~1
```

Remove pushed commit

```
git reset --hard HEAD~1
git push --force-with-lease
```

Remove middle commit

```
git rebase -i <commit-hash>^
# change pick → drop
```

---

14. Viewing Commit Details

```
git log --oneline
git show <commit-hash>
git show --show-signature <commit-hash>
```

Check commit email:

```
git show -s --format='%ae' HEAD
```

---

15. Clone URL Rule (Org vs User)

Repo owner	Clone URL
Personal	username/repo.git
Organization	orgname/repo.git

Always copy URL from **GitHub → Code → Clone**.

---

16. Final Pre-Push Checklist (Use Every Time)

```
git log --show-signature
git show -s --format='%ae' HEAD
git config --global --get user.signingkey
git config --global --get gpg.program
```

Every commit must show:

```
GPG: Good signature
```

---

17. One-Line Guaranteed Fix (Most Users)

```
git config --global user.email "email-in-gpg-key"
git commit --amend -S --no-edit
git push --force-with-lease
```

---

18. Best Practices

- Enable auto-signing globally
- Use feature branches only
- Never force-push shared branches
- Re-sign instead of reverting

✅Result

✔️Commits show **Verified** on GitHub ✔️Organization rule satisfied ✔️Push accepted

---

End of document