Checking for existing GPG keys

Before you generate a GPG key, you can check to see if you have any existing GPG keys.

Mac Windows Linux

Supported GPG key algorithms

GitHub supports several GPG key algorithms. If you try to add a key generated with an unsupported algorithm, you may encounter an error.

- RSA
- ElGamal
- DSA
- ECDH
- ECDSA
- EdDSA

Note: GPG does not come installed by default on macOS or Windows. To install GPG command line tools, see GnuPG's Download page.

1. Open TerminalTerminalGit Bash.
2. Use the `gpg --list-secret-keys --keyid-format=long` command to list the long form of the GPG keys for which you have both a public and private key. A private key is required for signing commits or tags.

   Shell

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

   Note: Some GPG installations on Linux may require you to use `gpg2 --list-keys --keyid-format LONG` to view a list of your existing keys instead. In this case you will also need to configure Git to use `gpg2` by running `git config --global gpg.program gpg2`.

3. Check the command output to see if you have a GPG key pair.
   - If there are no GPG key pairs or you don't want to use any that are available for signing commits and tags, then generate a new GPG key.
   - If there's an existing GPG key pair and you want to use it to sign commits and tags, you can display the public key using the following command, substituting in the GPG key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

     ```
     $ gpg --armor --export 3AA5C34371567BD2
     # Prints the GPG key ID, in ASCII armor format
     ```

You can then add your GPG key to your GitHub account.

Further reading

- "Generating a new GPG key"
- "Adding a GPG key to your GitHub account"
- "Telling Git about your signing key"
- "Associating an email with your GPG key"
- "Signing commits"
- "Signing tags"