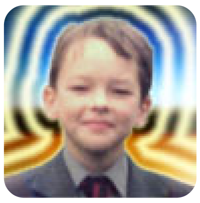
james mckay dot net
because there are few things that are less logical than business logic

Search jamesmckay.net



Signing Git commits with GPG on Windows

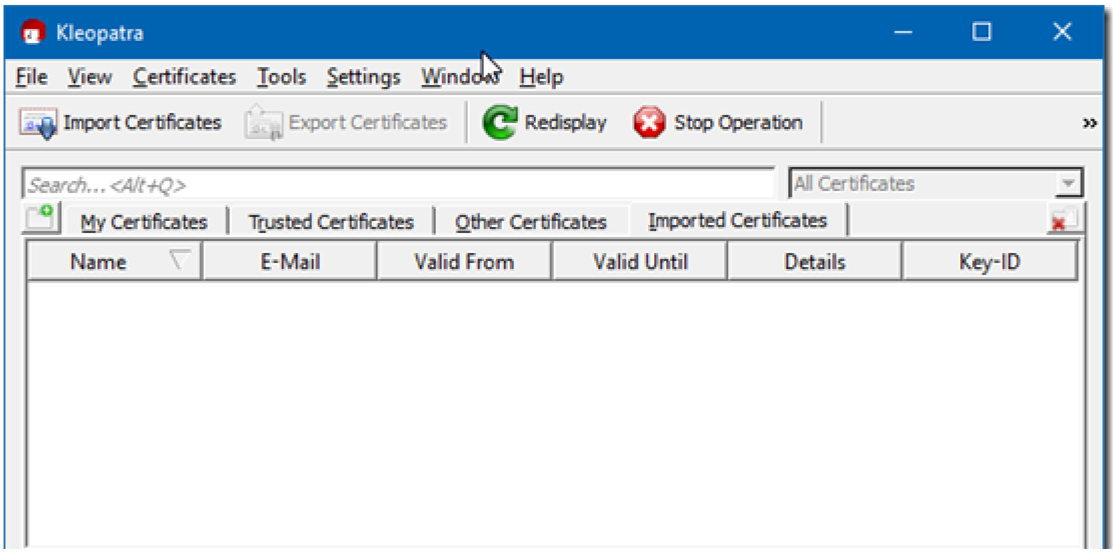
Source control • 6 comments • 24 February 2016

One of the "gotchas" with Git is that it allows you to check in code as anyone. By running `git config user.name` and `git config user.email`, you can put anyone's name to your commits—Stephen Hawking, Linus Torvalds, Henry VIII, or even me. If you want an idea of some of the problems this can cause, Mike Gerwitz's article A Git Horror Story is a cautionary tale.

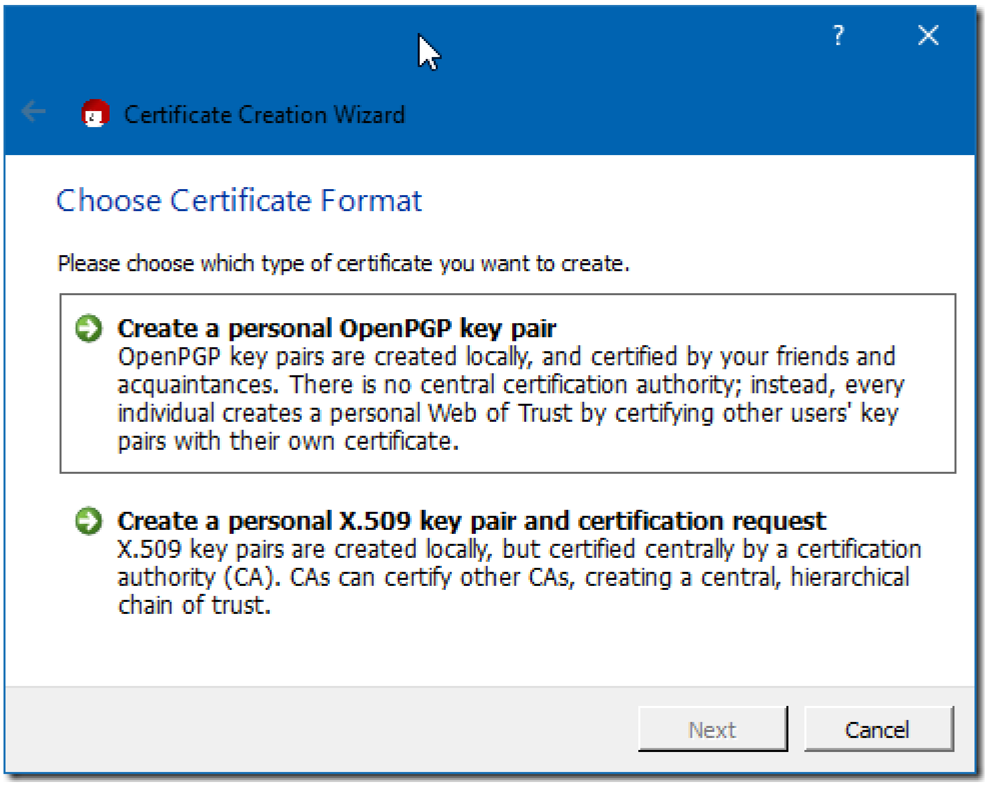
To resolve this problem, Git allows you to sign commits using GPG (GNU Privacy Guard, the GNU implementation of PGP), and in fact, Git includes the command-line version of GPG out of the box. You can run it within a Git Bash console.

However, most Windows users would prefer a GUI-based version, and gpg4win (GPG for Windows) is your go-to option here. You can install it either using the downloadable installer or else via Chocolatey.

To use gpg4win with Git needs a little bit of configuration, but first we'll generate a new certificate. Go to your Start menu and start up Kleopatra, the gpg4win key manager:



Now click on the “File” menu and choose “New certificate…”


Certificate Creation Wizard

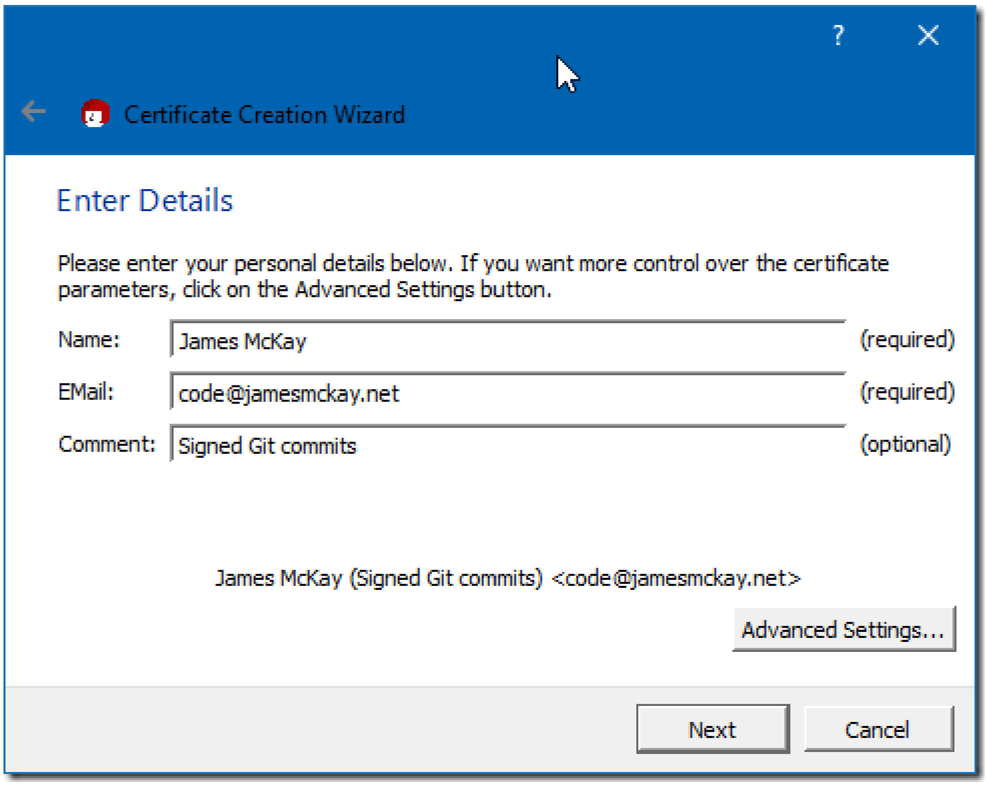
Choose Certificate Format

Please choose which type of certificate you want to create.

Create a personal OpenPGP key pair
OpenPGP key pairs are created locally, and certified by your friends and acquaintances. There is no central certification authority; instead, every individual creates a personal Web of Trust by certifying other users' key pairs with their own certificate.

Create a personal X.509 key pair and certification request
X.509 key pairs are created locally, but certified centrally by a certification authority (CA). CAs can certify other CAs, creating a central, hierarchical chain of trust.

Choose the first option here—a personal OpenPGP key pair. Enter your name and e-mail address, and optionally a comment:


Certificate Creation Wizard

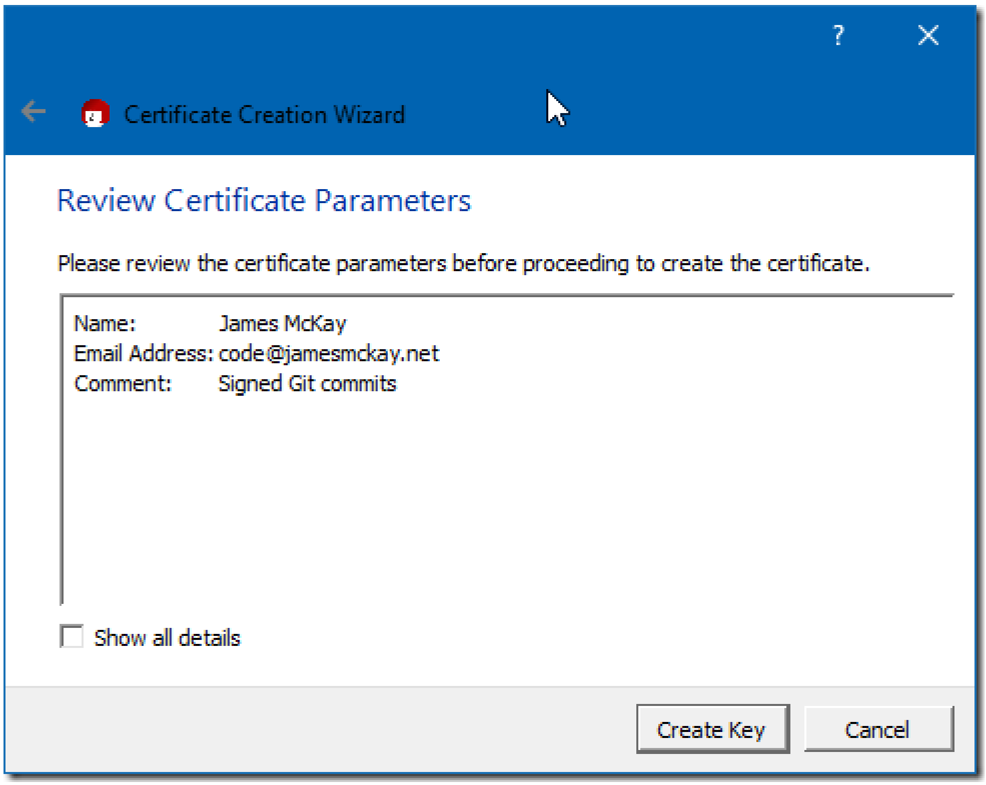
Enter Details

Please enter your personal details below. If you want more control over the certificate parameters, click on the Advanced Settings button.

Name:	James McKay	(required)
EMail:	code@jamesmckay.net	(required)
Comment:	Signed Git commits	(optional)

James McKay (Signed Git commits) <code@jamesmckay.net>

Review the certificate parameters and click “Create Key”:

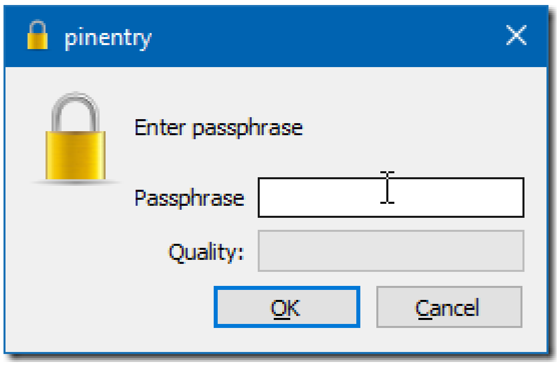


Review Certificate Parameters

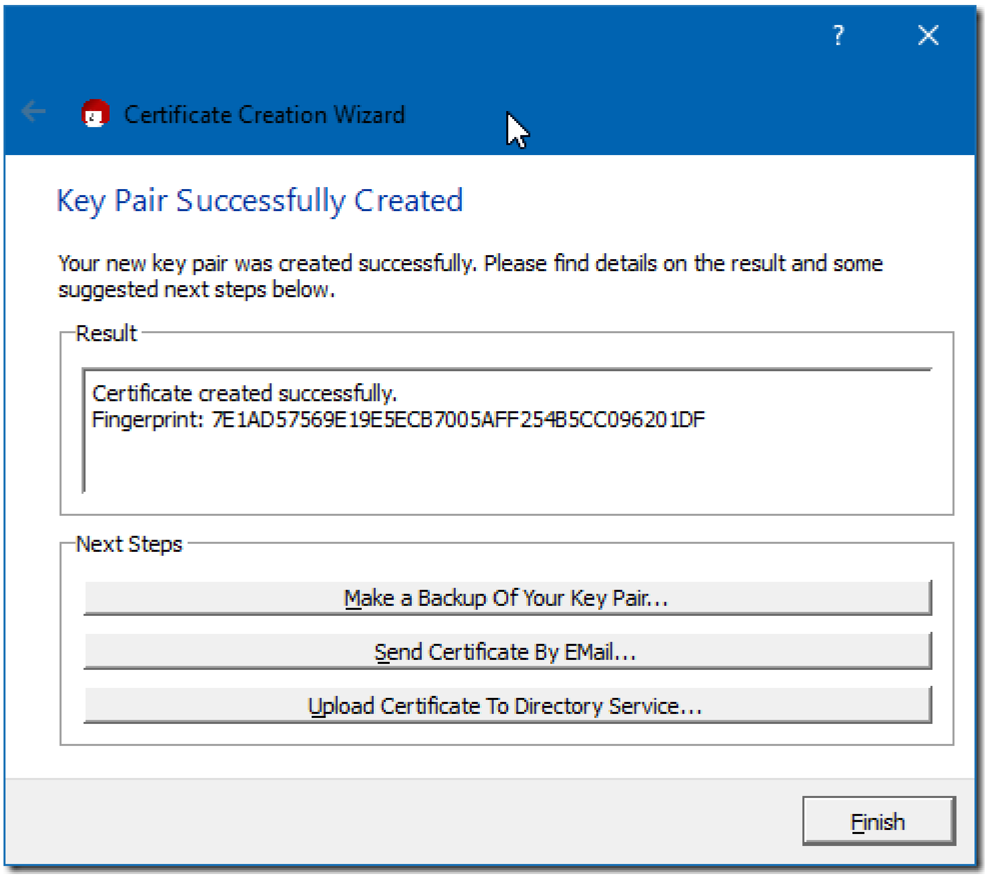
Please review the certificate parameters before proceeding to create the certificate.

Name: James McKay
Email Address: code@jamesmckay.net
Comment: Signed Git commits

You will be prompted to enter a passphrase:



Finally, your key pair will be successfully created:


Certificate Creation Wizard

Key Pair Successfully Created

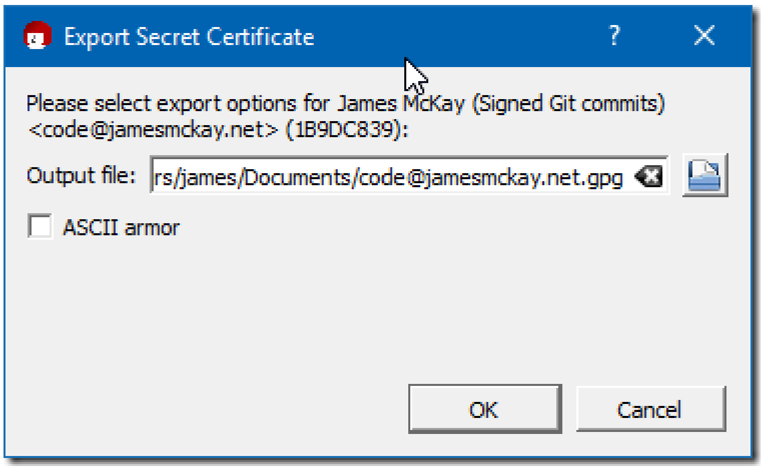
Your new key pair was created successfully. Please find details on the result and some suggested next steps below.

Result

```
Certificate created successfully.
Fingerprint: 7E1AD57569E19E5ECB7005AFF254B5CC096201DF
```

Next Steps
Make a Backup Of Your Key Pair...
Send Certificate By EMail...
Upload Certificate To Directory Service...
Finish

Click on “Make a Backup of Your Key Pair” to back it up to your hard disk.



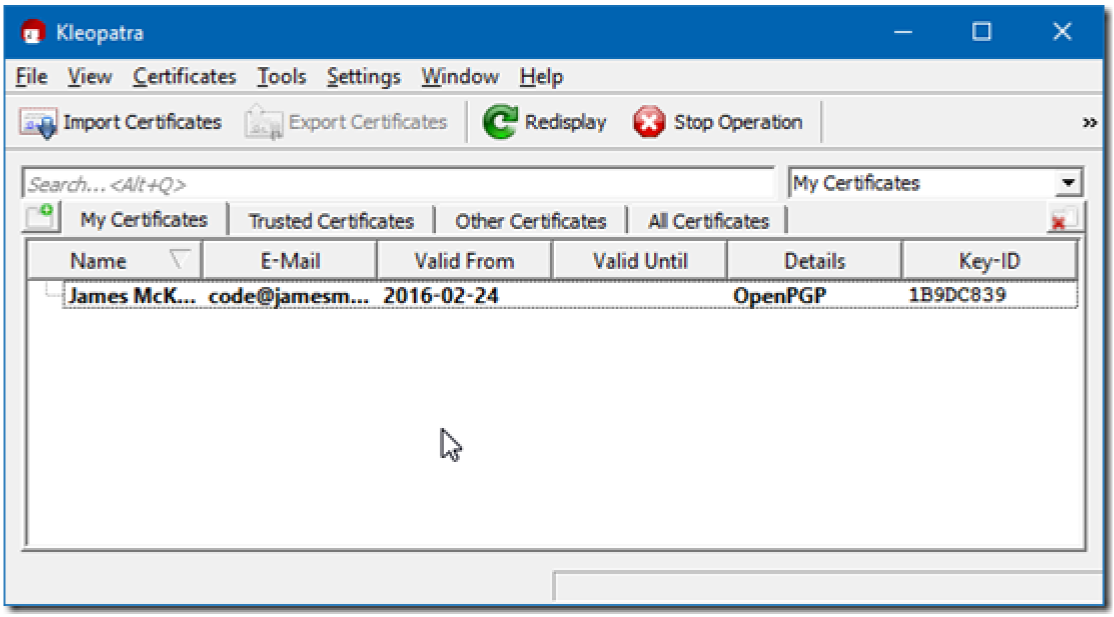
Please select export options for James McKay (Signed Git commits) <code@jamesmckay.net> (1B9DC839):

Output file: rs/james/Documents/code@jamesmckay.net.gpg

☐ ASCII armor

Save your private key somewhere safe. Your password manager database is as good a place as any. (You *are* using a password manager, aren’t you?)

Once you’ve gone through the wizard, you will see your new key pair in the Kleopatra main window. Click on the “My Certificates” tab if you don’t see it at first:



Name	E-Mail	Valid From	Valid Until	Details	Key-ID
James McK...	code@jamesm...	2016-02-24		OpenPGP	1B9DC839

The number in the right hand column, in this case 1B9DC839, is your key ID. You now need to configure Git to use it. Type this in a Git shell, replacing "1B9DC839" with your own GPG key:

```
git config --global user.signingkey 1B9DC839
```

Prior to version 2.0, you had to instruct Git to sign each commit one at a time by specifying the `-S` parameter to `git commit`. However, Git 2.0 introduced a configuration option that instructs it to sign every commit automatically. Type this at the console:

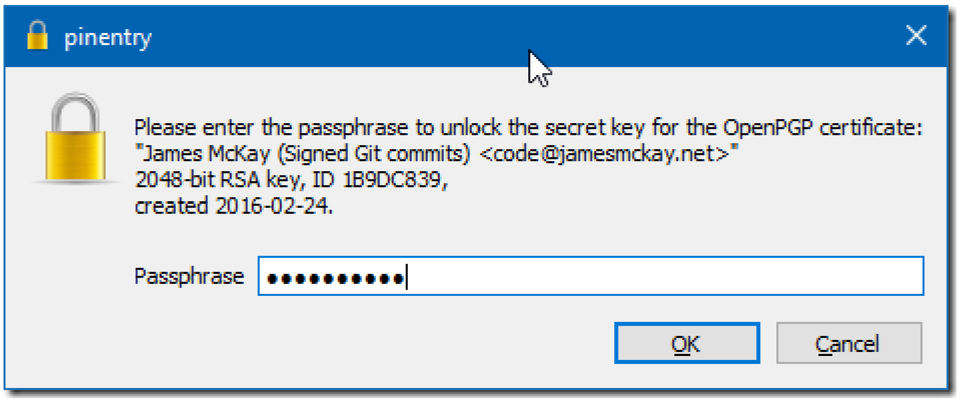
```
git config --global commit.gpgsign true
```

Finally, you need to tell Git to use the gpg4win version of `gpg.exe`. Git comes with its own version of `gpg.exe`, but it is the MinGW version—a direct port of the Linux version, which saves your keychain in the `~/.gnupg` folder in your home directory. The gpg4win port, on the other hand, saves your keychain in `~/AppData/Roaming/GnuPG`. Certificates managed by one won't be seen by the other. You will also need to use the gpg4win version if you want to use a GUI such as SourceTree, since the MinGW version of `gpg.exe` is entirely command line based and doesn't play nicely with Git GUIs. By contrast, the gpg4win version brings up a dialog box to prompt for your password.

```
git config --global gpg.program "C:\Program Files (x86)\GnuPG\bin\gpg.exe"
```

If you are using 32-bit Windows, or if you have installed gpg4win into a custom location, you will need to tweak the location of the program. *(Update February 2018: the path to gpg.exe has changed with the release of gpg4win 3.0. I have updated the path here to point to the new location. Note also that if you are using a Cygwin shell, you may need to specify the path in a different format — see the comments below.)*

To check that it works, commit some code to a repository somewhere. You should be prompted for the passphrase that you entered earlier:



Please enter the passphrase to unlock the secret key for the OpenPGP certificate:
"James McKay (Signed Git commits) <code@jamesmckay.net>"
2048-bit RSA key, ID 1B9DC839,
created 2016-02-24.

Passphrase ●●●●●●●●●●●

You can then verify that your commit has been signed as follows:

```
$ git log c05ddaa8 --show-signature -1
commit c05ddaa8e9da289fa5148d370b8ba9e5c419df9a
gpg: Signature made 02/24/16 08:08:39 GMT Standard Time using RSA key ID
1B9DC839^M
gpg: Good signature from "James McKay (Signed Git commits)
<code@JAMESMCKAY.NET>" [ultimate]^M
Author: James McKay <code@JAMESMCKAY.NET>
Date:   Wed Feb 24 08:07:47 2016 +0000
```

You won't be asked for your passphrase every time. Once you've entered it once, gpg spins up a process called gpg-agent.exe, which caches it in memory for a while.

Posted at 13:00 on 24 February 2016.
Tagged: git, gpg, windows

« "I've never had a problem with it." Fuzzy dates aren't as good an idea as you think »

6 comments:

Reply from **Dash** at 23:43 on 20 Aug 2016 #

git config –global gpg.program "c:/Program Files (x86)/GNU/GnuPG/gpg2.exe"

Exactly what I was looking for! Thank you

Reply from **Myy** at 11:19 on 6 Oct 2017 #

From a Cygwin shell, this worked for me :
git config –global gpg.program "/cygdrive/c/Program Files (x86)/GnuPG/bin/gpg.exe"

Still, very good tutorial !

Reply from **Laurent Mazuel** at 17:48 on 18 Oct 2017 #

With latest GPg4WIN 3.0.0:
git config –global gpg.program 'C:\Program Files (x86)\GnuPG\bin\gpg.exe'

#

@Laurent

Thanks, but it's: git config –global gpg.program "C:\Program Files (x86)\GnuPG\bin\gpg.exe"

Reply from **Ahamed** at 15:51 on 23 Nov 2017 #

Is it possible to use separately installed gpg (C:\Program Files (x86)\GnuPG\bin\gpg.exe) from git bash? By default it's only using older gpg, mingw version that came with git.
I have put git config –global gpg.program "C:\Program Files (x86)\GnuPG\bin\gpg.exe" in git configuration

Reply from **james** at 07:50 on 7 Feb 2018 #

Thanks for all your feedback folks. I've not been doing much coding on Windows for a while now and only just recently got back into it, so this post has ended up getting a bit out of date: the path to gpg changed without me noticing.

As you've all correctly pointed out, the path that you're looking for with the latest gpg is:

C:\Program Files (x86)\GnuPG\bin\gpg.exe

I've updated the post to provide the correct path.

Comments are closed.

Search

Services I use and recommend

- FastMail
- DigitalOcean

Recent Posts

- To throw or not to throw? Doing something that has already been done
- Positive, negative, or error?
- Supergiant stars are hot vacuums
- What is xkcd "Time" all about?
- Error handling is the one thing that puts me off learning Go

- Sharing data between Terraform configurations
- You probably only need a t2.nano instance for that
- Query Objects: a better approach than your BLL/repository
- Sorting out the confusion that is OWIN, Katana and IAppBuilder

Recent Comments

- David on Sharing data between Terraform configurations
- james on Signing Git commits with GPG on Windows
- Arun on Programmatically starting an AWS instance with an encrypted EBS volume attached
- YEC Best Evidence 9: not enough salt in the sea, or not enough precision in the measurements? – How old is the earth? on What is xkcd "Time" all about?
- Ahamed on Signing Git commits with GPG on Windows

Archives

Select Month

Categories

- .NET
- Agile
- AWS
- Blog
- Commuting
- DevOps
- Humour
- Lab notebooking
- Miscellaneous
- N-tier deconstructed
- Open source
- Patterns and practices
- Personal
- Photography
- Programming

- Science and faith
- Security
- Side projects
- Source control
- TDD
- Technology
- Web development
- Work

Tags

.net agile asp.net aws aws lambda bcrypt BDD blog BS build scripts C# code reviews communication configuration continuous integration creation css design dvcs ec2 evolution exceptions feature branches feature switches git golang javascript lab notes less logging mercurial n-tier deconstructed passwords pet projects photography preprocessors programming languages recruiting sass/scss science security surveys tdd vim visual studio

Meta

- Log in
- Entries RSS
- Comments RSS
- WordPress.org