GNU Privacy Guard

By David Schoen

```
13:37 < kcidau> is that the thing geeks have in their
email signature?
13:38 < lyte> Yes.
```

Why am I doing this talk?

Because I'm loud mouthed and Mick asked me

I'm vaguely interested in cryptography

http://www.flickr.com/photos/11312511@N02/1303166623/

GPG, PGP, WTF?!

PGP - Program created in 1991, some licensing concerns

OpenPGP - RFC2440 updated to RFC4880 - standard for the same, but with only open standards

GPG - GNU Privacy Guard

http://en.wikipedia.org/wiki/Pretty_Good_Privacy
http://tools.ietf.org/html/rfc2440
http://tools.ietf.org/html/rfc4880

What is it good for?

Encrypting - preventing eavesdroppers

- email
- sensitive documents
- passwords

Signing - verifying who created something

- code (yay git!)
- announcements
- email

If it's so awesome, why isn't everyone using it?

You already are!

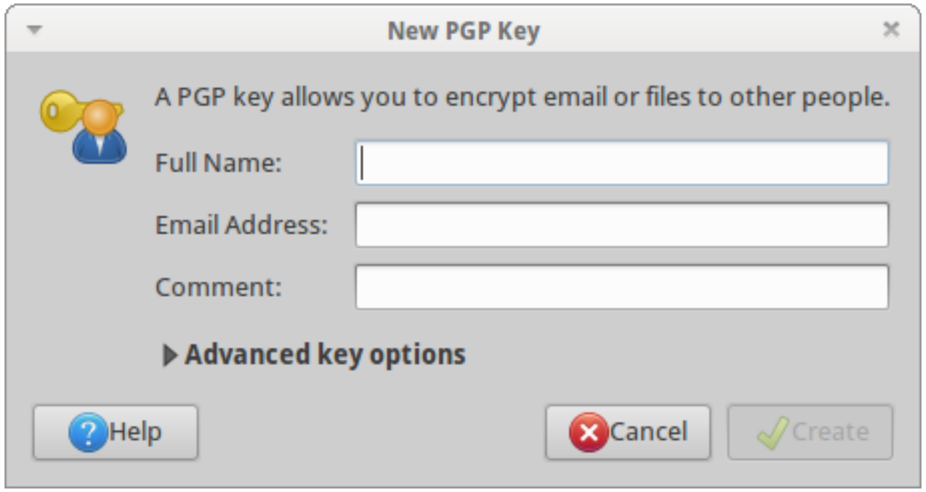
Debian and EL packages are already signed by most repositories using GPG... try it

```
$ apt-key list                              # Yes ... I'm aware Ubuntu is not Debian ...
/etc/apt/trusted.gpg
--------------------
pub   1024D/437D05B5 2004-09-12
uid                  Ubuntu Archive Automatic Signing Key <ftpmaster@ubuntu.com>
sub   2048g/79164387 2004-09-12


# gpg --quiet --with-fingerprint /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5
pub  1024D/E8562897 2007-01-06 CentOS-5 Key (CentOS 5 Official Signing Key) <centos-5-key@centos.org>
      Key fingerprint = 473D 66D5 2122 71FD 51CC  17B1 A8A4 47DC E856 2897
sub  1024g/1E9EA3B6 2007-01-06 [expires: 2017-01-03]
```

Creating a key

GUI: Seahorse > File > New > GPG


New PGP Key

A PGP key allows you to encrypt email or files to other people.

Full Name:

Email Address:

Comment:

CLI:

```
$ gpg --gen-key
```

In most modern distros the defaults are ok...

Use a passphrase!

Backups? (who needs 'em)

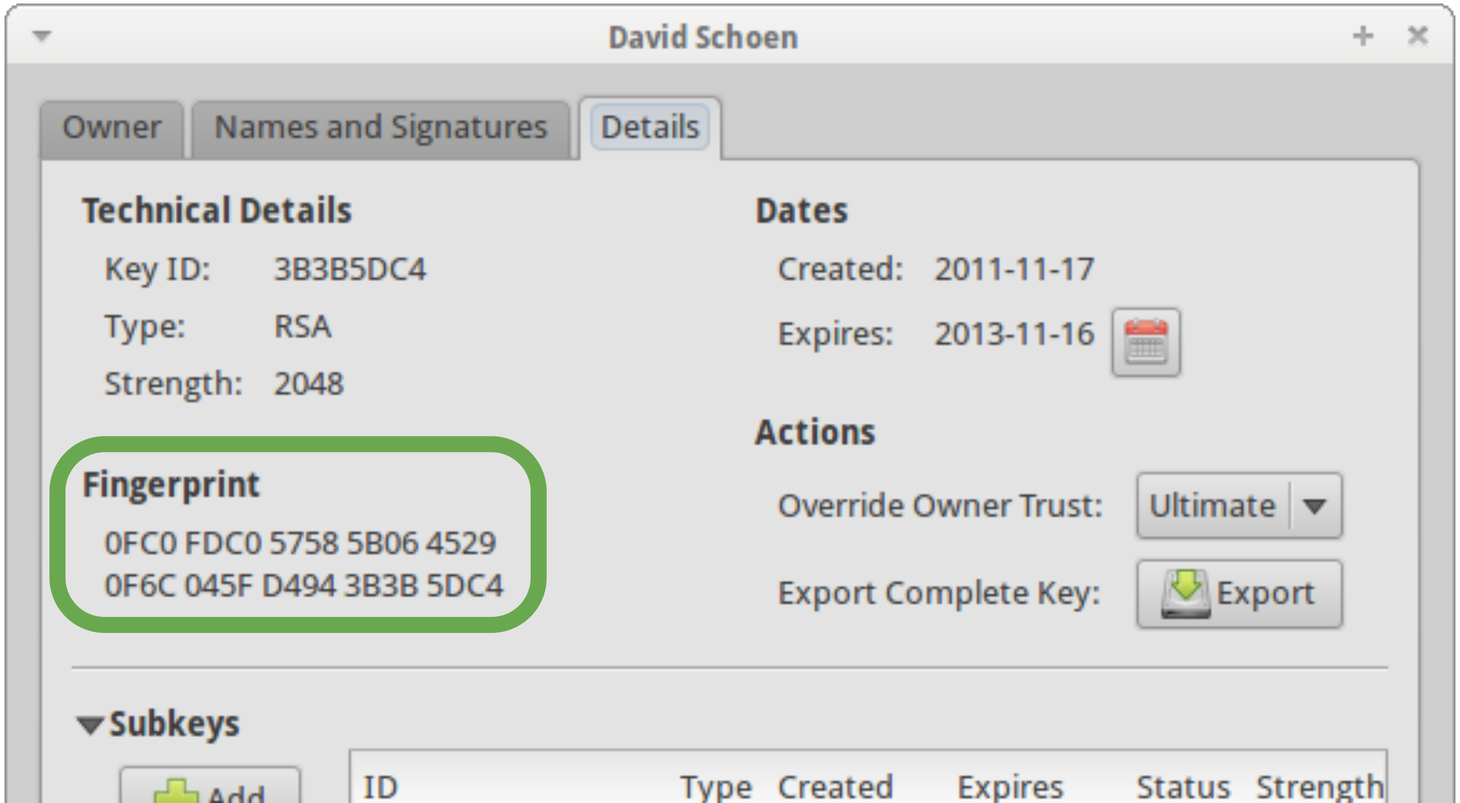
Backup contents of ~/.gnupg

Store it somewhere ***<u>safe</u>***

The private key may be encrypted, but the passphrase can still be brute forced

Fingerprints

```
$ gpg --fingerprint dave@lyte.id.au
pub   2048R/3B3B5DC4 2011-11-17 [expires: 2013-11-16]
      Key fingerprint = 0FC0 FDC0 5758 5B06 4529  0F6C 045F D494 3B3B 5DC4
uid                  David Schoen <dave@lyte.id.au>
sub   2048R/BD9063CE 2011-11-17 [expires: 2013-11-16]
```



Technical Details

Key ID: 3B3B5DC4

Type: RSA

Strength: 2048

Fingerprint

0FC0 FDC0 5758 5B06 4529
0F6C 045F D494 3B3B 5DC4

Dates

Created: 2011-11-17

Expires: 2013-11-16

Actions

Override Owner Trust: Ultimate

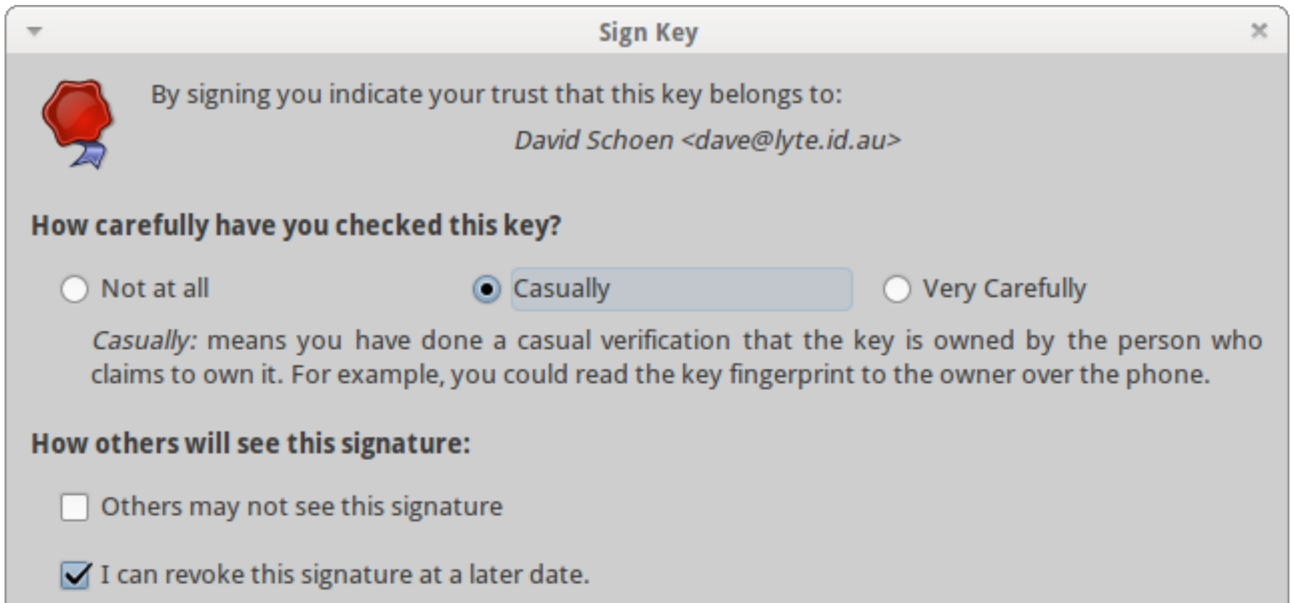
Export Complete Key: Export

Subkeys

Signing someone else's key

Don't sign a key you haven't independently verified!

```
$ gpg --recv-key <key ID>
$ gpg --sign-key <anything that matches>
$ gpg --send-keys <key IDs>
```



By signing you indicate your trust that this key belongs to:

David Schoen <dave@lyte.id.au>

How carefully have you checked this key?

Not at all [✓ Casually] Very Carefully

Casually: means you have done a casual verification that the key is owned by the person who claims to own it. For example, you could read the key fingerprint to the owner over the phone.

How others will see this signature:

☐ Others may not see this signature

☑ I can revoke this signature at a later date.

Thunderbird on Ubuntu

```
$ sudo apt-get install enigmail
```

Restart Thunderbird...

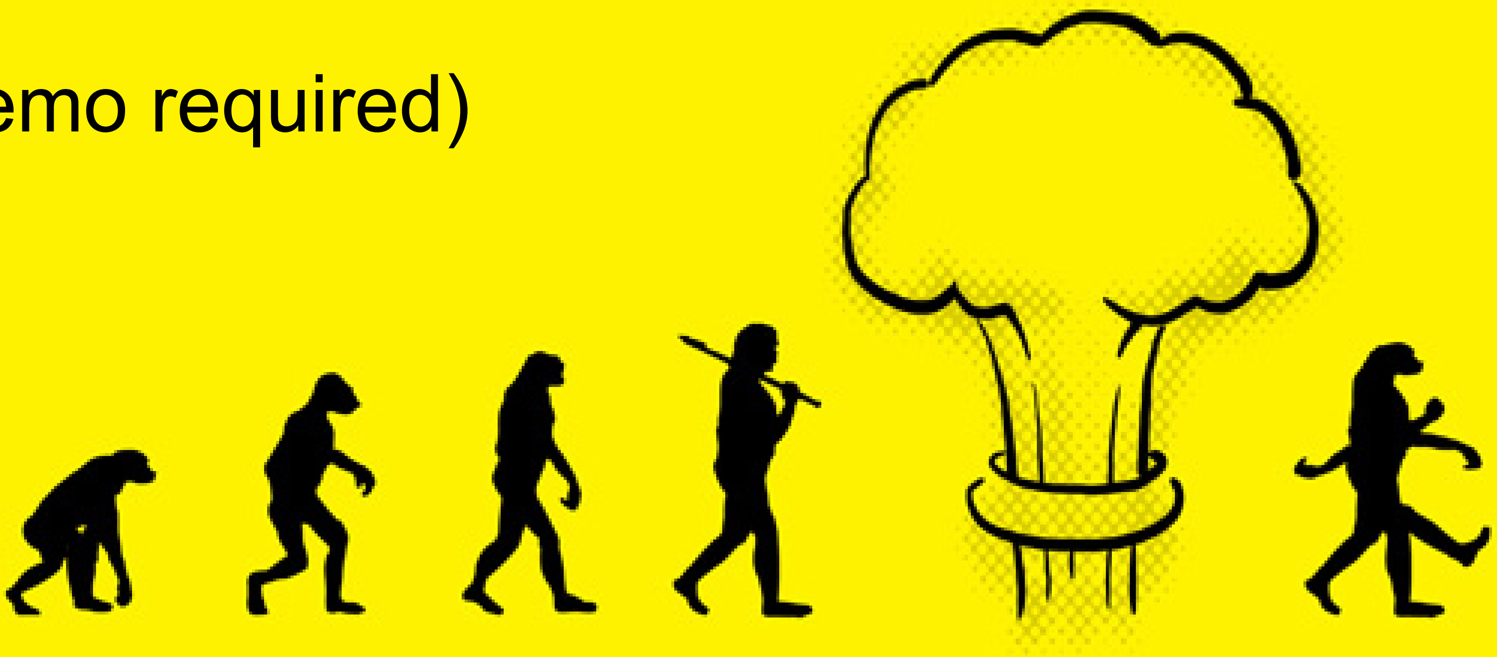
Edit > Account Settings > %account% > OpenPGP Security

Signatures attached by default, alter "Use PGP/Mime"

Evolution

Just works!

(Demo required)



http://h4nd.deviantart.com/art/Nuclear-Evolution-122996365

Random email related stuff

Attachment based (.asc) email sigs allow for HTML

Not this:

```
-----BEGIN PGP SIGNED MESSAGE-----
```

Gmail and other web based emails

There have been a few attempts

None are either still supported or mature

http://howto.cnet.com/8301-11310_39-10434684-285/want-really-secure-gmail-try-gpg-encryption/

http://getfiregpg.org/

https://github.com/RC1140/cr-gpg

Manual signing

Sign a file (binary):

```
$ gpg --sign ...
```

Plain text signature:

```
$ gpg --clearsign ...
```

Detached (separate file) binary sign:

```
$ gpg --detach-sign ...
```

Revocation

Revocation certs are special signed docs that tell everyone to stop trusting your key

Hard to spread around

http://www.flickr.com/photos/annamatic3000/2519852967/

Editors store decrypted caches...

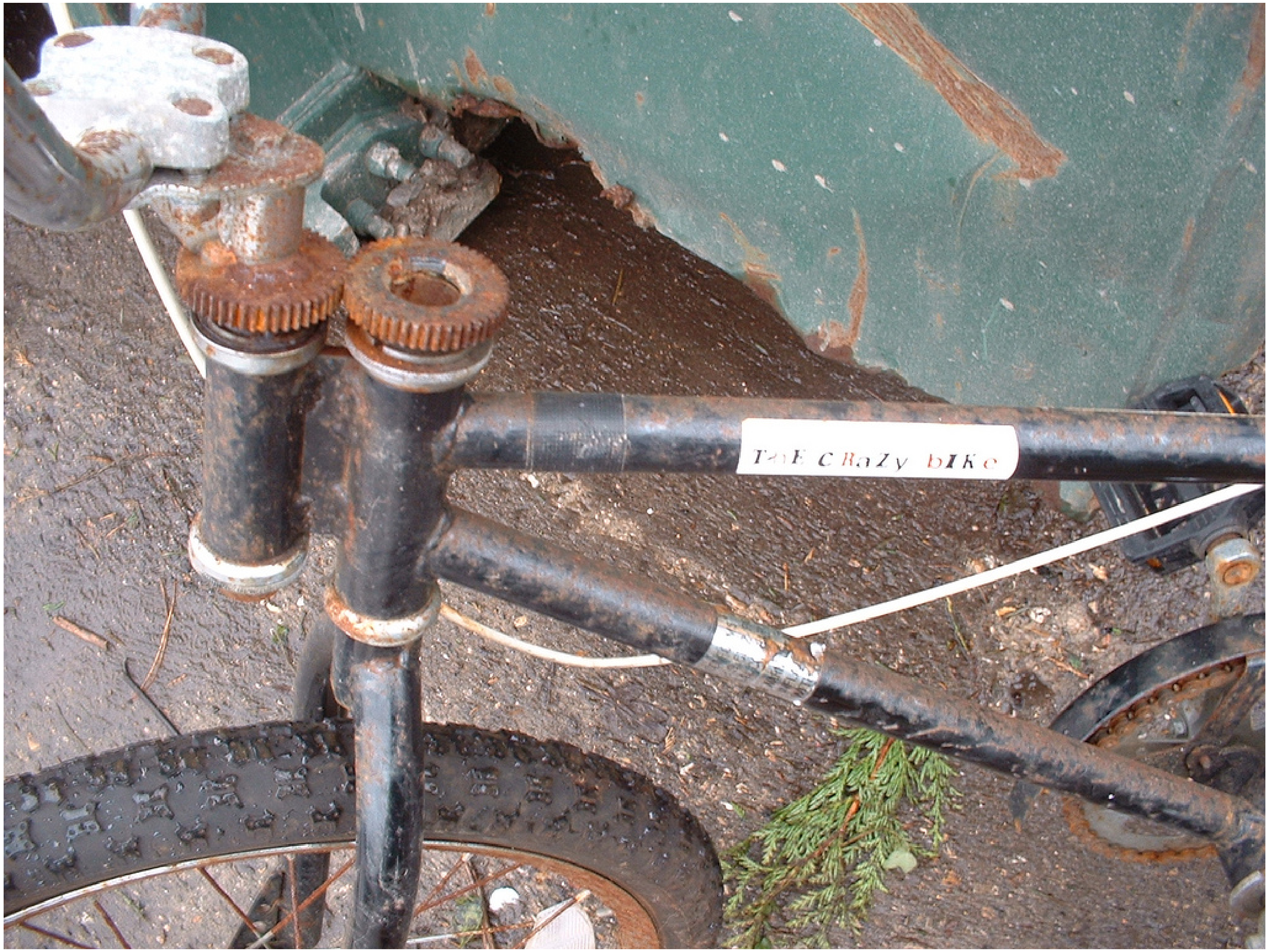
```
$ gpg -d <sensitive>.txt.gpg | vi -n -
```

From inside vi saving gets a little harder:

```
:1,$ w !gpg -e -r <key|group ID> [-r
<key|group ID> ...] > <encrypted file>
```

Groups can be defined in ~/.gnupg/gpg.conf

Creative use cases



http://www.flickr.com/photos/joimson/3242849101/

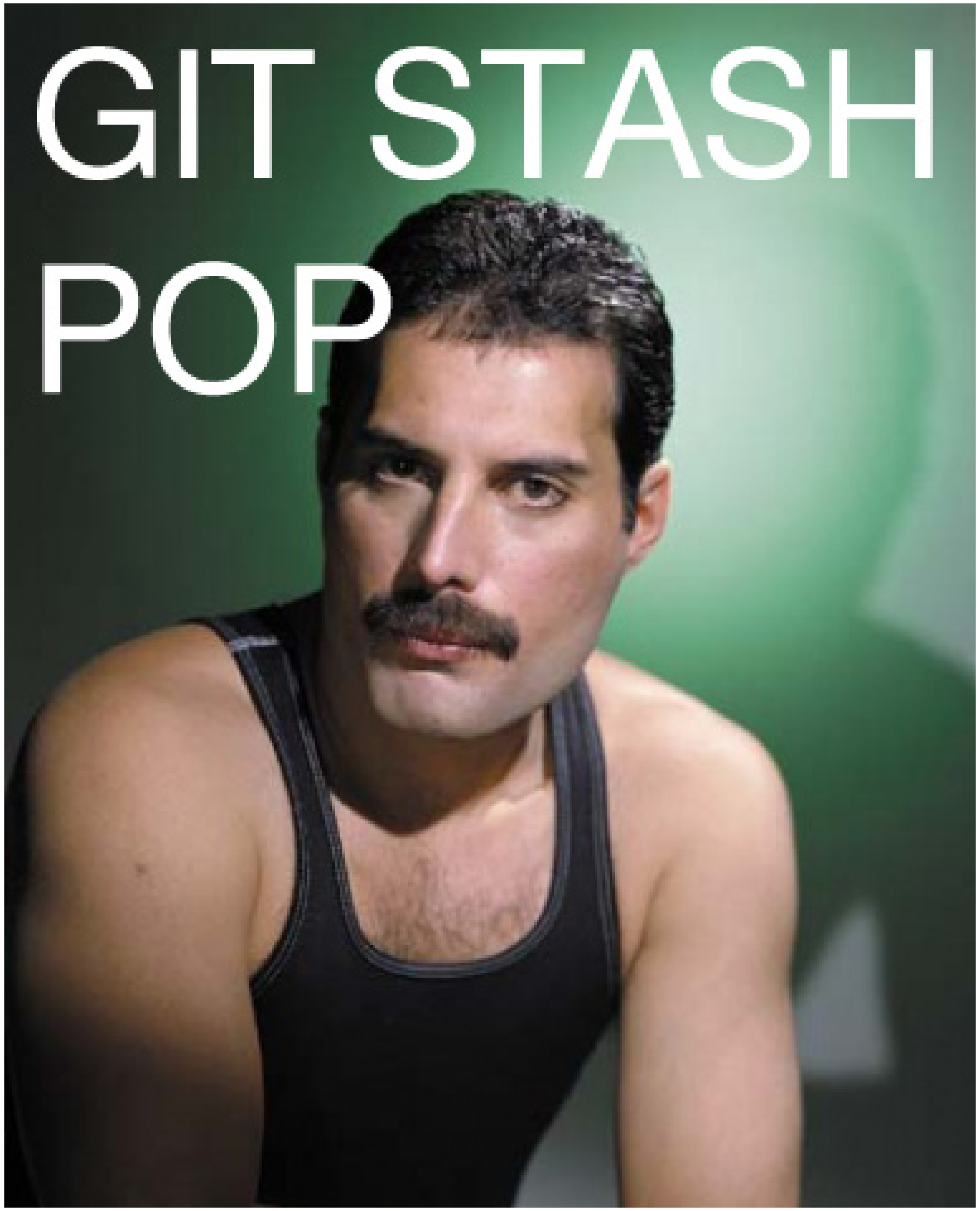
git + gpg

Create a signed tag:

```
$ git tag -s <name>
```

Verify a signed tag:

```
$ git tag -v <name>
```



http://www.flickr.com/photos/dealingwith/4556711396/

Creative Use Cases - Puppet

Puppet without a Master

Worried about a central server getting hacked?

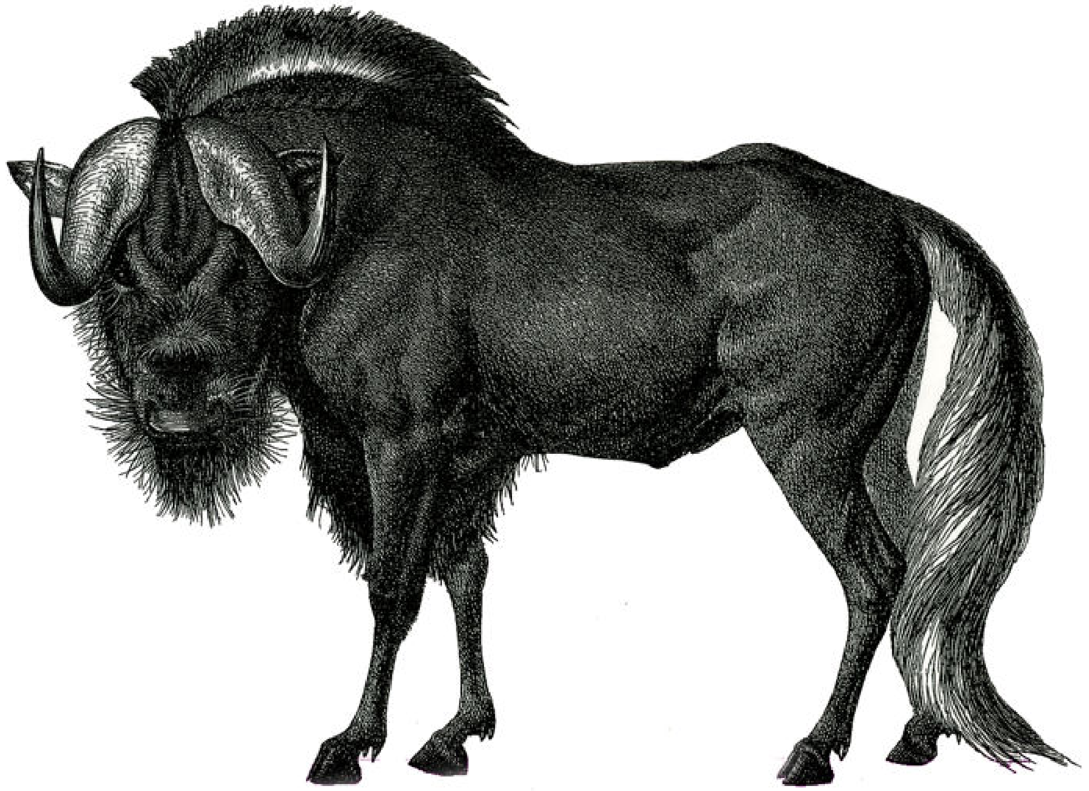
Want to verify changes are from trusted humans?

Use GPG.

http://www.flickr.com/photos/bulldogsrule/99318756/
http://current.workingdirectory.net/posts/2011/puppet-without-masters/

Password management using Org-Mode

Edit a file with .org.gpg extension, emacs does the rest!



Idea verbally "borrowed" from Ben Sturmfels - http://stumbles.id.au/

http://www.flickr.com/photos/thefangmonster/3872158703/

Duplicity - Backups

Space efficient - Secure backups

http://duplicity.nongnu.org/
http://www.flickr.com/photos/rbrwr/149453029/

Creative Use Cases - Anonymous

Currently have big problems proving or dismissing an announcement is theirs

If they use GPG, they only need to solve the problem once

Steve Gibson - http://www.grc.com/sn/sn-341.txt

http://www.flickr.com/photos/mattw/2336507468/

Monkeysphere (slide not present at actual talk)

SSL via Web of Trust instead of CA based checks

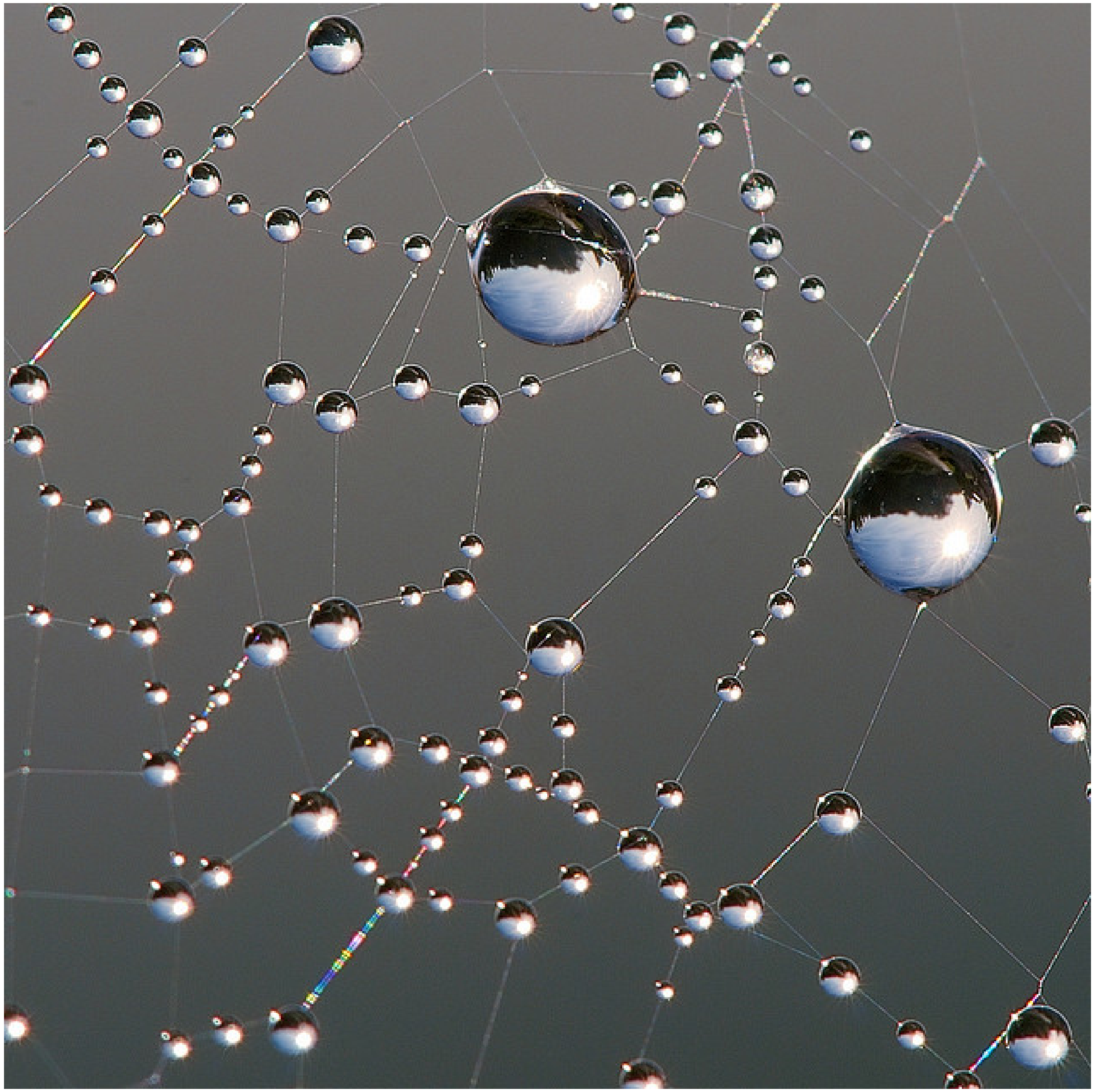
SSH host key verification no more "The authenticity of host '...' can't be established." when someone you trust has verified the host

SSH authorized_keys management via Web of Trust

http://web.monkeysphere.info/

Web of Trust

Almost the opposite of CA based trust



http://en.wikipedia.org/wiki/Web_of_trust - http://www.flickr.com/photos/luc_viatour/4247957432/

So how does it actually work?

http://www.flickr.com/photos/life-is-a-photo/3811903312/

So how does it actually work?

Hashing

Asymmetric encryption ciphers

Symmetric encryption ciphers

So what is a hash?

http://www.flickr.com/photos/drome/2928600995/

Not that kind of hash.

Function that takes arbitrary input and returns fixed size output

$f(x) = x \bmod 10$

So what is a ***good*** hash?

- Easy to compute
- Infeasible to generate a message matching a given hash output
- Infeasible to modify a message without changing the hash
- Infeasible to find two different messages with the same hash

http://en.wikipedia.org/wiki/Cryptographic_hash_function

Symmetric Cryptography

Both parties know the key

Difficult to initiate communication

Secure with small key size once initiated

Fast!

http://en.wikipedia.org/wiki/Symmetric-key_algorithm

Asymmetric Cryptography

AKA Public key crypto

Good for transmitting symmetric keys

Vulnerable to MITM due to lack of authentication (where WoT comes in)

http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange

GPG encryption in practice

Message encrypted with random "session key" (symmetric cipher)

"session key" encrypted with all recipients public keys

encrypted "session key" and symmetrically encrypted message stored together

http://www.ietf.org/rfc/rfc4880.txt

GPG signing in practice

Hash of document calculated

Hash encrypted with private key

http://www.flickr.com/photos/oberazzi/318947873/

Other References

Matt Guica's Free Software Melbourne talk contained great inspiration for this talk

See:

- http://lists.softwarefreedom.com.au/pipermail/free-software-melb/2011-November/000281.html
- http://lists.softwarefreedom.com.au/pipermail/free-software-melb/2011-November/000277.html