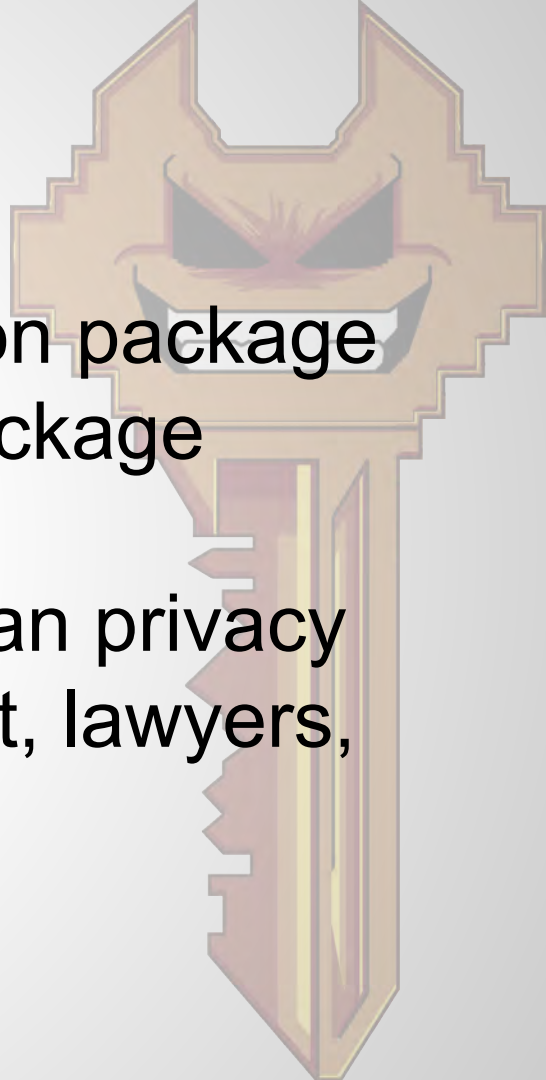
Check Your Fingerprints: Cloning the Strong Set

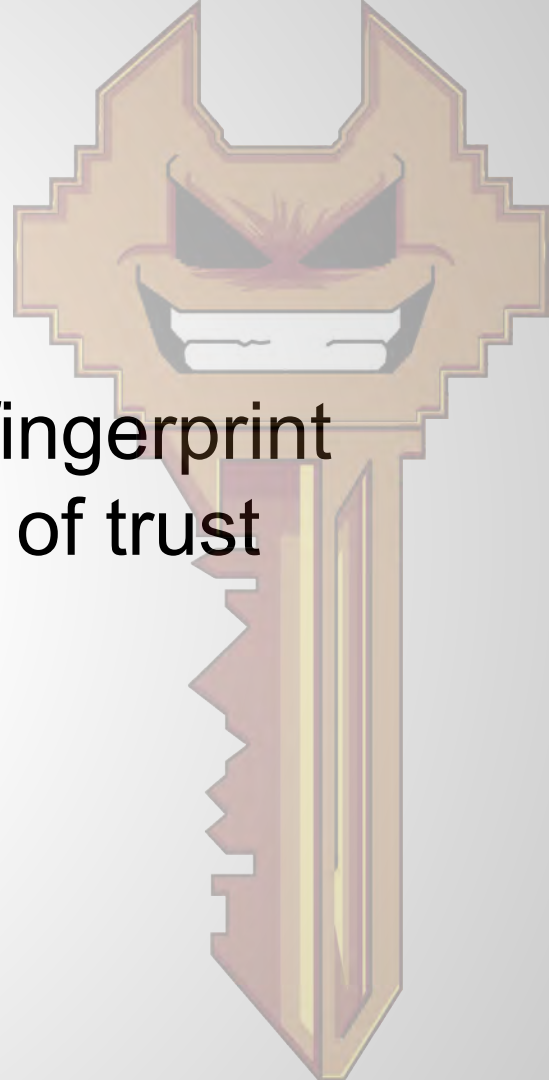
Richard Klafter
Eric Swanson

GPG is Used Everywhere



- Most widely used email encryption package
- Used extensively for software package verification
- Lots of people use GPG other than privacy conscious crypto nerds: journalist, lawyers, software maintainers

Key Exchange



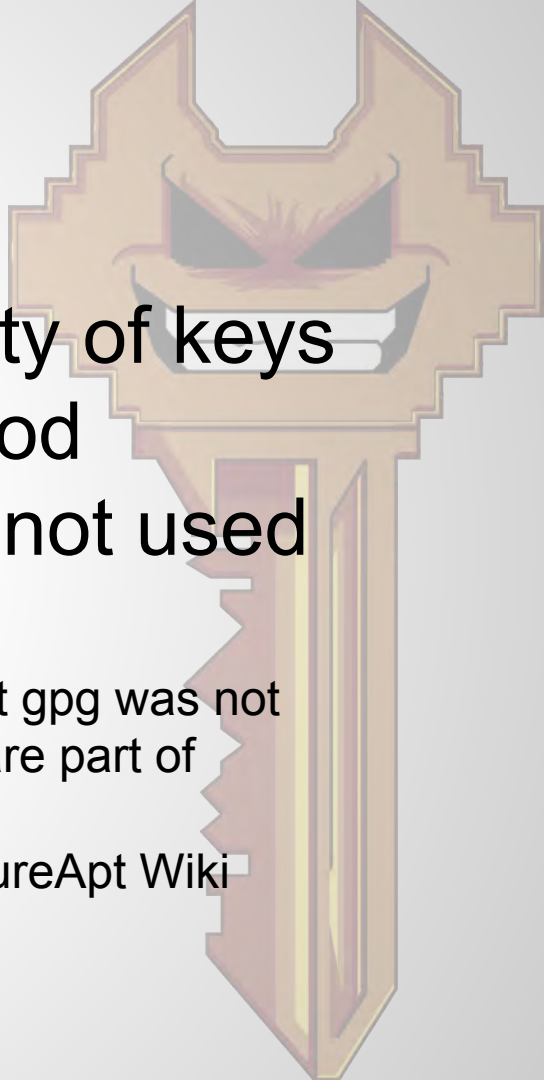
- Usually relies on key servers
- Retrieved with 32bits, 64bits, or fingerprint
- Verify key with fingerprint or web of trust
- Mistakes are fatal
- Key exchange is hard

State of the Web of Trust

- Method of establishing authenticity of keys
- Most popular decentralized method
- Usage difficulty means it is often not used

The Warning: "no ultimately trusted keys found" means that gpg was not configured to ultimately trust a specific key. Trust settings are part of OpenPGPs Web-of-Trust which does not apply here.

- Debian SecureApt Wiki



DEMO - Install Puppet

- Download Puppet and its signature from our mirror
  - http://mirror.evil32.com/puppet.tar.gz
  - http://mirror.evil32.com/puppet.tar.gz.asc
- Verify signature (because you don't trust evil32.com :)
  - http://docs.puppetlabs.com/guides/puppetlabs_package_verification.html

Humans are Broken (Not Encryption)

- Humans make mistakes
- Not good at comparing large strings
- GPG does not help protect us from ourselves

```
eswanson@turing ~$ gpg --keyserver pgp.mit.edu --recv-keys 10000001

gpg: requesting key 10000001 from hkp server pgp.mit.edu
gpg: key 10000001: public key "John Doe <john@doe.com>" imported
gpg: key 10000001: public key "Jane Doe <jane@doe.com>" imported
gpg: Total number processed: 2
gpg:               imported: 2   (RSA: 2)
```

GPG NOT Verifying Received Key

```
eswanson@turing ~$ gpg --keyserver pgp.mit.edu --recv-keys
80615870F5BAD690333686D0F2AD85AC1E42B367

gpg: requesting key 1E42B367 from hkp server pgp.mit.edu
gpg: key 0BADBEEF: public key "Evil32" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

?

- GPG does verify what the server sent you
- Key server can tell you to import anything
- Key servers do not use SSL so MITM or DNS break

Installing Docker with apt-key adv

```
root@gpgevil:~# sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9

Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --
homedir /tmp/tmp.GBHIRGOWXe --no-auto-check-trustdb --trust-model always --
keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --
keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
36A1D7869245C8950F966E92D8576A8BA88D21E9
gpg: requesting key A88D21E9 from hkp server keyserver.ubuntu.com
gpg: key 0BADBEEF: public key "Evil32" imported
gpg: Total number processed: 1
gpg:               imported: 1   (RSA: 1)
```

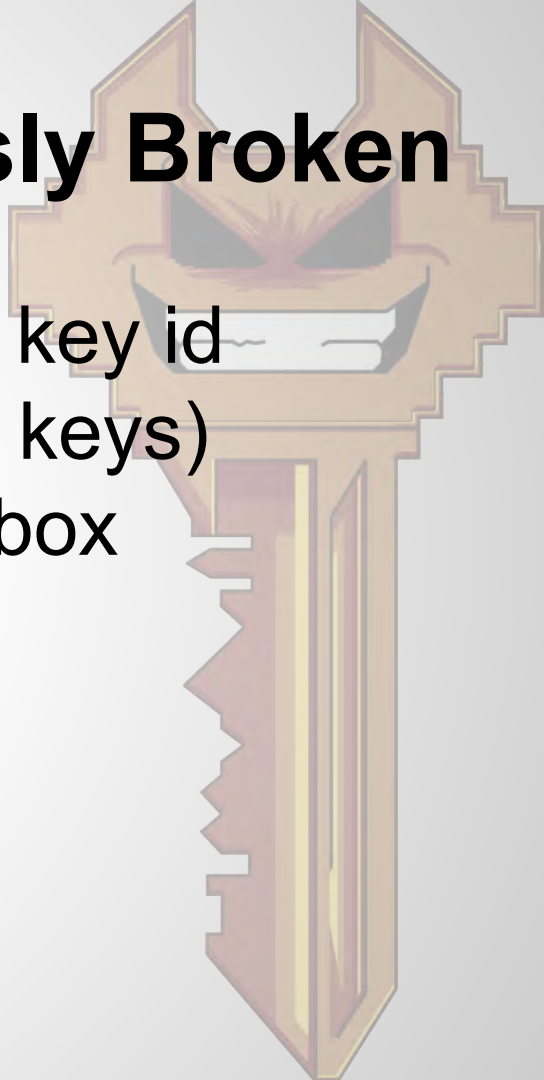
How the Tool Works

1. Generates 500 million GPG keys a second
2. Checks each key for a partial fingerprint collision

- Runs using OpenCL on a modern GPU
- Uses simplified regexes to find multiple collisions simultaneously
- Source and more info on our github project: **https://evil32.com**

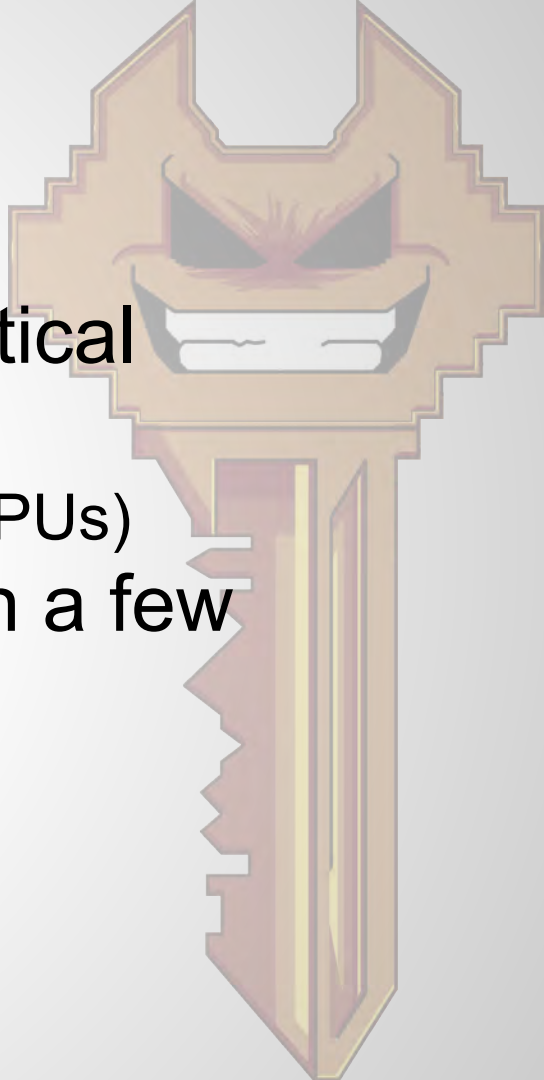
32bit Key IDs are Ridiculously Broken

- Few seconds to generate a 32bit key id
- Cloned the web of trust (~50,000 keys)
- Took a day on a 4 year old linux box



64bit Key Ids

- Finding a specific key id not practical
- Finding a key id relatively easy
  (107 days looking for 100 keys with 20 GPUs)
- Likely easy to find 64bit key ids in a few years



Vulnerabilities

Network is owned:

- Send arbitrary keys in response to --recv-keys
- Tamper with http fingerprints

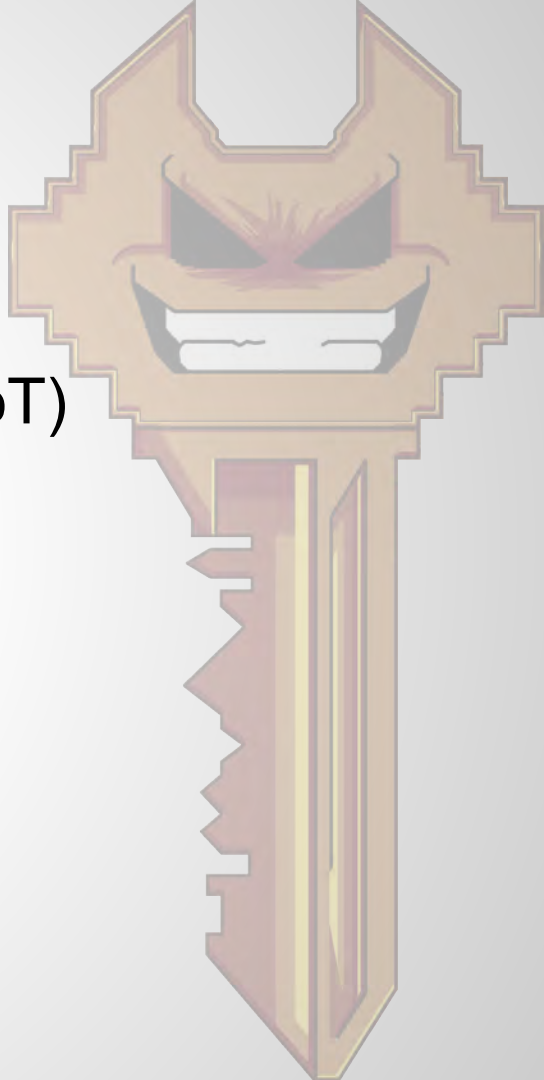
Network is secure:

- Exploit 32-bit key collisions
- Upload arbitrary data to keyserver

Takeaways

- Three rules of GPG
  1. Verify your fingerprints (or use the WoT)
  2. Don’t trust the keyserver
  3. Never use 32-bit key IDs

- GPG UI is broken (who knew?)



Go to our project page to get the source code and see other fun GPG UI exploits

https://evil32.com