

Modern Keysigning





The WoT

- oldest social network

- 50k keys in the strong set

How to Keysign:

- Verify Fingerprint
- Verify Identity
- Obtain authentic copy of key
 - many traps to be avoided
- Sign and send key
 - currently needs MTA

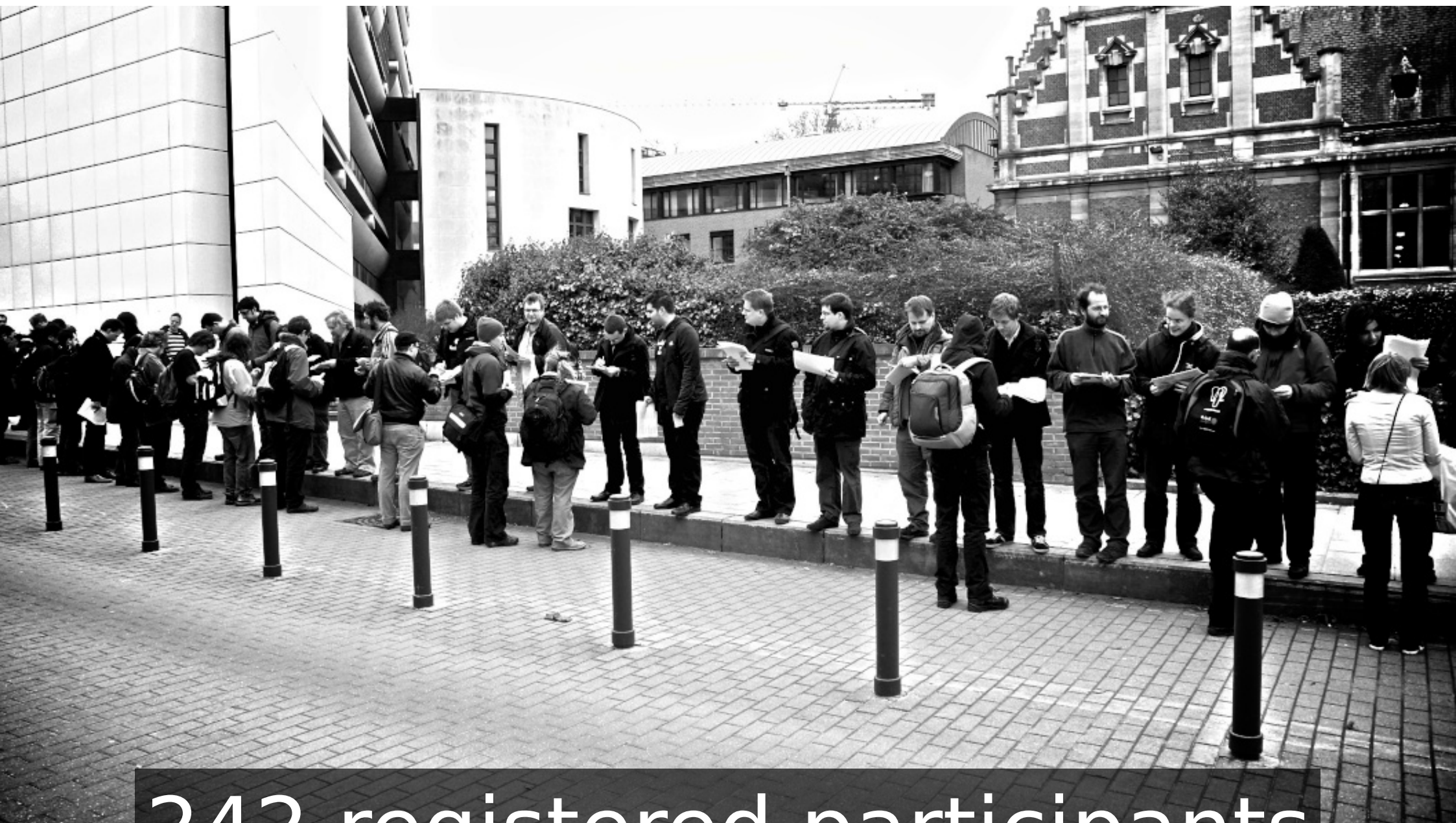
- $O(n)$
- $O(n^2)$

This is a typical keysigning protocol

It is hard to set up



@FOSDEM 2011



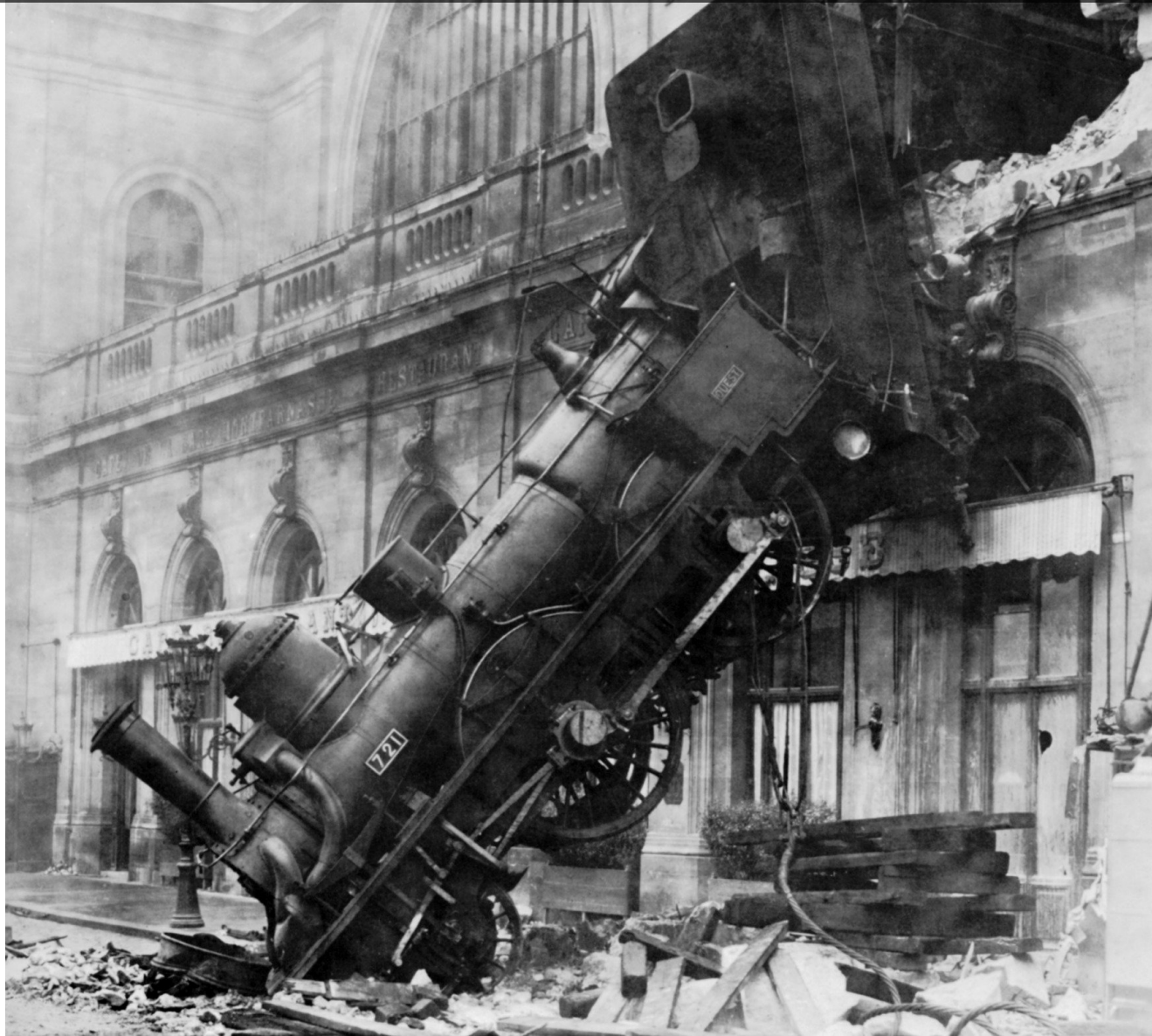
242 registered participants

LET'S DECENTRALISE

Keysigning "parties" suck

ALL THE THINGS!

A party without beer
only to obtain fingerprints



Keysigning "Parties" are not fun
people miss them
They don't print anything
Single point of failure



Base2

01100001000011001011001001010010
00110111101100110111000011101001
11101011001000010000100011101000
10011100111011100001101101101011
00000101100110110101100110001110

Pros: Very accurate, hard to misread

Cons: Very long

WTF

IS THIS SHIT?

A red locomotive, numbered 9771, is shown pushing a blue railcar through a dense pile of shipping containers. The containers are in various colors, including red, green, and blue. The scene is set outdoors on a gravel surface. A semi-transparent black box is overlaid on the image, containing white text.

Base16

610CB25237B370E9EB21
08E89CEE1B6B059B598E

Pros: looks familiar to nerds

Cons: Hard to distinguish characters

NOT SURE IF



B IS 8

Verifying fingerprints is hard



Base64

YQyyUjezcOnrlQjonO4bawWbWY4=

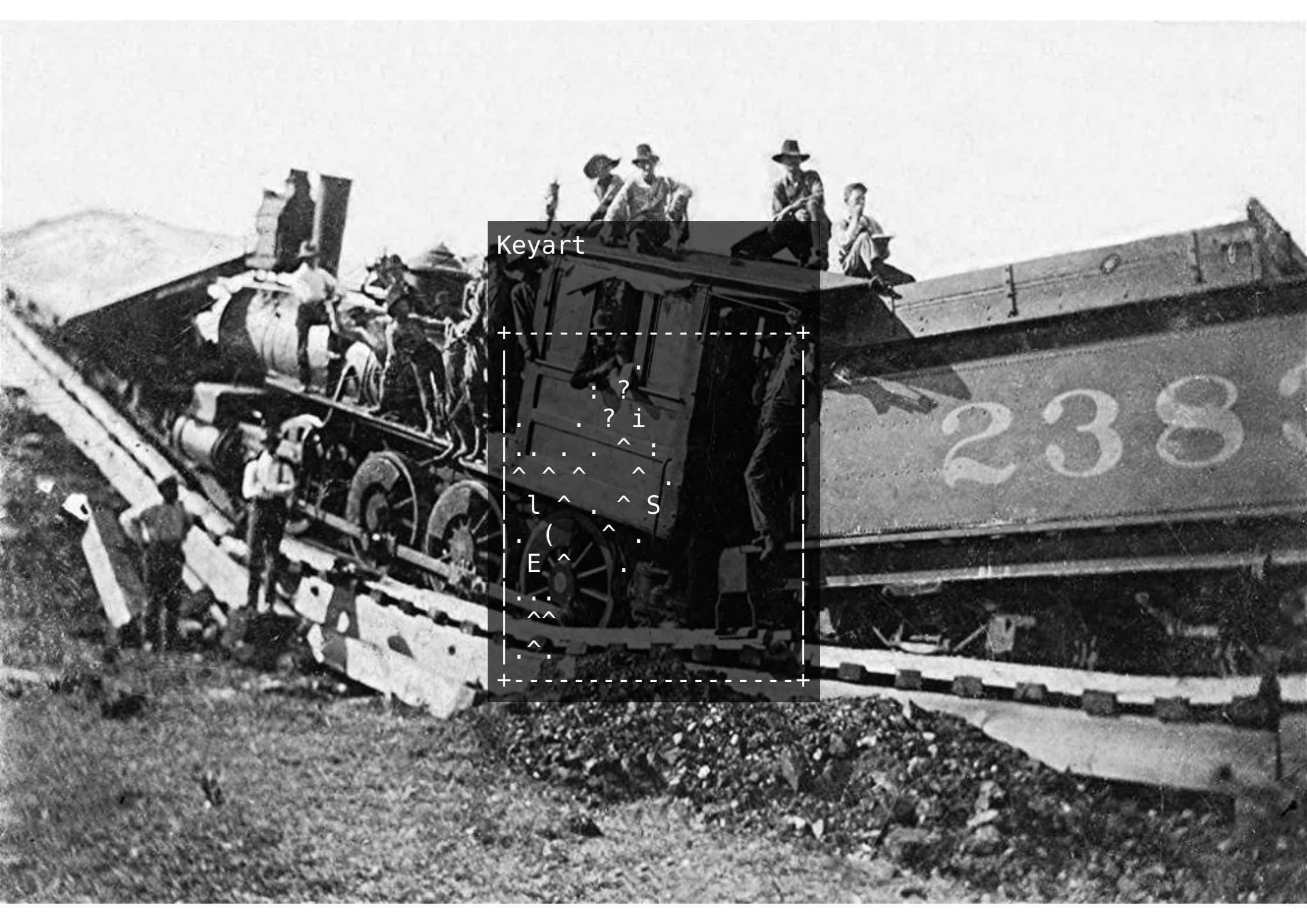
Pros: Shorter than other things

Cons: Probably too big of an alphabet



Base58Check

9r9knGannSDvojyUoGbgyWDUeWGdx7rUC



Keyart

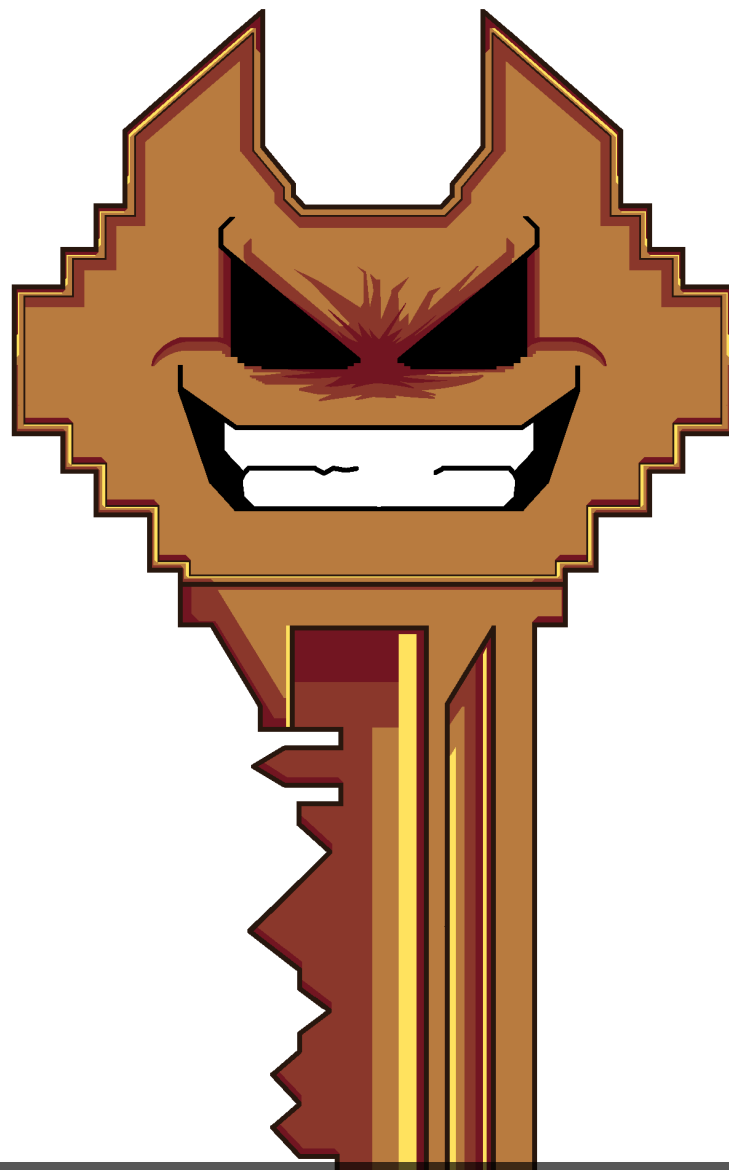
+-----+
| |
| : ? |
| . . ? i |
| . . . ^ : |
| ^ ^ ^ ^ ^ |
| l ^ . ^ S |
| . (^ . |
| E ^ . |
| . ^ |
| ^ ^ |
| . ^ |
+-----+

Pros: Probably simpler to compare

Cons: Easy to have collisions



Eventually, you've verified the fingerprint and the identity.
You try to obtain an authentic copy of the key.



short key ids
evil32.com

Of course you don't use short key ids. Do you..?

Browser address bar: <https://wiki.debian.org/gpghowto>

Wiki navigation: [WIKI](#) [Login](#) [FrontPage](#) [RecentChanges](#) [FindPage](#) [HelpContents](#)

Search: [Titles](#) [Text](#)

debian / Wiki / [Login](#) [Info](#) [Attachments](#) [More Actions:](#)

gpghowto

This page is about howto get ride of the error which says "W: GPG error: <http://debian.cites.uiuc.edu> testing Release: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 010908312D230C5F". You may want to run apt-get update to correct these problems" while updating your system with apt-get. Most of the time you get this error if the pubkey is not available on the debian server from were you are downloading your packages.

Nobody uses short keyids. Except Debian

More detailed info are present on [SecureApt](#) (interesting things starting from [How apt uses Release.gpg](#)) . Please understand what you are doing first.

These are the commands which you need to run.

```
$ gpg --recv-keys 2D230C5F
$ gpg --export -a 2D230C5F | sudo apt-key add -
$ apt-get update
```


I don't always use short key ids

private key, you need to follow these guidelines when signing people's keys:

During the Event

1. Keysigning is always done after meeting in person
2. During this meeting you hand each other your OpenPGP key fingerprint and at least one government issued key fingerprints are usually distributed as key fingerprint slips, created by a script such as `gpg-key2ps` (pack
3. You check whether the name on the key corresponds with the name on the ID and whether the person in front of he is.

After the Event

You now have the printed public key information from the other participants.

Example key IDs for the other participants will be E4758D1D, C27659A2, and 09026E7B. Replace these IDs with the other participants.

1. retrieve the keys:
 1. `gpg --recv-keys E4758D1D C27659A2 09026E7B`
2. sign the keys:
 1. `gpg --sign-key E4758D1D`
 2. `gpg --sign-key C27659A2`
 3. `gpg --sign-key 09026E7B`
3. export the keys
 1. `gpg --armor --export E4758D1D --output E4758D1D.signed-by.01234567.asc`
 2. `gpg --armor --export C27659A2 --output C27659A2.signed-by.01234567.asc`
 3. `gpg --armor --export 09026E7B --output 09026E7B.signed-by.01234567.asc`
4. Email the key users (use the email address that was part of the key's user ID) and attach the corresponding signed key to the key server:
 1. `gpg --send-keys --keyserver keyserver.ubuntu.com E4758D1D`

And Ubuntu



For the party, you will need these strips and an official photo ID, such as a driver's license or passport.

After the Party

Step 1: Get other people's keys

You now have the printed public key information from the other participants.



Example key IDs for the other participants will be [E4758D1D](#), [C27659A2](#), and [09026E7B](#). Replace these IDs with the key IDs you received from the other participants.

And pretty much everybody else

1. Find the key ID numbers on each printout and get the public keys from the keyservers:

```
gpg --recv-keys E4758D1D C27659A2 09026E7B
```

Step 2: Sign the keys

1. Sign a key:

```
gpg --sign-key E4758D1D
```

- i. If a key has multiple user IDs, GPG will ask if you want to sign all of them.

share improve this question

edited Jun 22 '14 at 11:23

asked Jun 22 '14 at 8:54



Elena

385 2 14

add a comment

1 Answer

active oldest vot



As error message, you haven't configured gpg server.

Try this:

```
gpg --keyserver subkeys.pgp.net --recv-keys 6092693E && gpg --export --armor 6092693E | sudo apt-key add -
```

on the Internet

Updated

It seems that you can not connect to server:

```
gpg: keyserver timed out
```

Do you have a firewall block port 11371 of hkp service.

You can use port 80 instead of 17371:

```
gpg --keyserver subkeys.pgp.net:80 --recv-keys 6092693E
```

share improve this answer

edited Jun 22 '14 at 11:00

answered Jun 22 '14 at 10:00



cuonglm

31.2k 3 30 73



issue1579: GnuPG ignores the fingerprint

Also: v3 keys still accepted

So you use the fingerprint instead of short key ids
however, currently shipped gnupg version do not
check for the fingerprint of the key to be imported

A photograph of a train accident. Several freight cars are derailed and overturned on a set of railroad tracks. The cars are various colors, including brown, green, and yellow. Some have text on them, like "RUNG TO N" and "RT HE RN". There are people standing around the wreckage, and the scene is outdoors with some vegetation visible in the background.

Let's not use Keyservers

- leaks data (plain HTTP)
- trivial MITM attacks (issue1579)
- packet forgery
 - drop UIDs
 - signatures
 - revocations
- OCaml... srsly.

<http://bugs.gnupg.org/gnupg/issue1579>

**I DON'T ALWAYS
TARGET USERS**



**BUT WHEN I
DO, IT'S ME**

Let's define our target users.
It's my mom!



The Gold Standard:

K-3048

File Edit View Search Tools Documents



*caffrc x

```
# .caffrc -- vim:ft=perl:
# This file is in perl(1) format - see caff(1) for details.

$CONFIG{'owner'} = 'Username';
#$CONFIG{'email'} = '[user]@[domain]';
#$CONFIG{'reply-to'} = 'foo@bla.org';

# You can get your long keyid from
#   gpg --with-colons --list-key <yourkeyid|name|emailaddress...>
#
# If you have a v4 key, it will simply be the last 16 digits of
# your fingerprint.
#
# Example:
#   $CONFIG{'keyid'} = [ qw{FEDCBA9876543210} ];
# or, if you have more than one key:
#   $CONFIG{'keyid'} = [ qw{0123456789ABCDEF 89ABCDEF76543210} ];
#$CONFIG{'keyid'} = [ qw{0123456789abcdef 89abcdef76543210} ];

# Select this/these keys to sign with
#$CONFIG{'local-user'} = [ qw{123456789abcdef 89abcdef76543210} ];
# Additionally encrypt messages for these keyids
```

CAFF... PERL... srsly..?

That's the pinnacle, the gold-standard of contemporary keysigning

**LET'S MAKE THEM USE
BASE16, OCAML, AND PERL**

FOR THEIR CRYPTO

AM I THE ONLY ONE AROUND HERE



WHO IS SICK OF FINGERPRINTS?

A man with dark, curly hair and glasses is seated at a desk in a cluttered office. He is wearing a plaid shirt and is looking down at his work. The desk is covered with various items, including a blue pen holder with several pens, a yellow folder, and a brown paper bag. In the background, there are shelves filled with books and other office supplies. A sign on the wall reads "Fast forward 20 years".

Fast forward 20 years

We see the fire burning

We need to do something about the situation

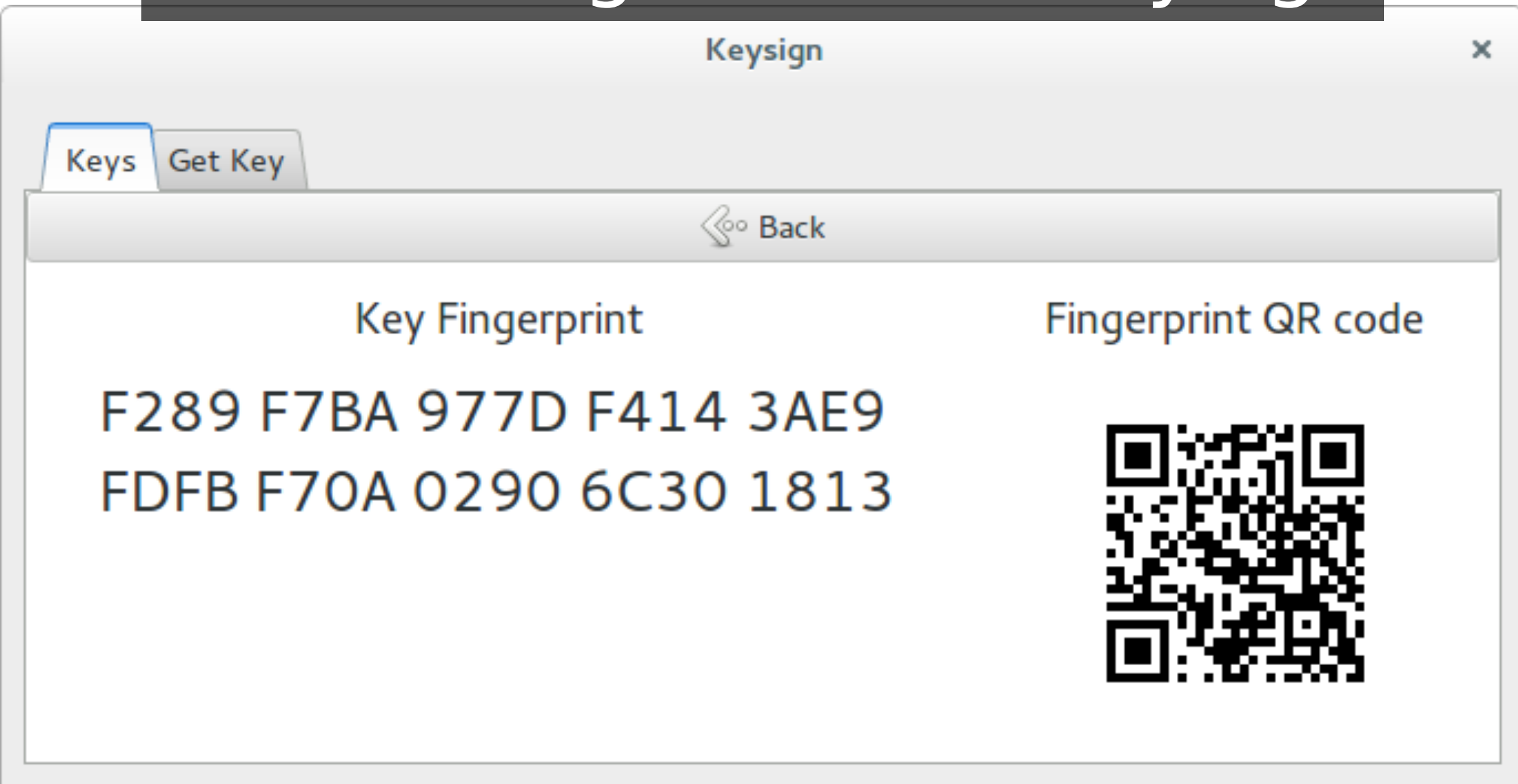


Two decades later:

- mobile computing
- WiFi
- QR Codes

Yet, our machines cannot talk to each other

Introducing: GNOME Keysign



Leveraging 2000s technologies

Keysign


Keys

Get Key


Type fingerprint


F289 F7BA 977D F414 3AE9
FDFB F70A 0290 6C30 1813

... or scan QR code

 Open Image

Step 1: Scan QR Code or type fingerprint and click on 'Download' button

 Back

 Next



Demo

./geysign.sh /home/muelli/vcs/geysigning

File Edit View Search Terminal Tabs Help

nano /home/muelli/vcs/ge... nano /home/muelli/vcs/ge... ipython /home/muelli/vcs/... ./geysign.sh /home/muelli... fish /home/muelli/vcs/gey... python /home/muelli/vcs/...

→ geysigning git:(tobi_tmp_email) X> ./geysign.sh

/usr/lib/python2.7/dist-packages/gobject/constants.py:24: Warning: g_boxed_type_register_static: assertion 'g_type_from_name (name) == 0' failed

import gobject.gobject

root (INFO): Startup

root (INFO): Activate!

Found service 'HTTP Keyserver' type '_geysign_tcp' domain 'local'

Found service 'HTTP Keyserver' type '_geysign_tcp' domain 'local'

Found service 'HTTP Keyserver' type '_geysign_tcp' domain 'local'

service resolved

name: HTTP Keyserver

address: fe80::a64e:31ff:fedc:e2a4

port: 9001

root (INFO): Probably discovered something, let me check; HTTP Keyserver fe80::a64e:31ff:fedc:e2a4:9001

emitted None

service resolved

name: HTTP Keyserver

address: fe80::3e97:eff:fed8:f7f7

port: 9001

root (INFO): Probably discovered something, let me check; HTTP Keyserver fe80::3e97:eff:fed8:f7f7:9001

emitted None

service resolved

name: HTTP Keyserver

address: 10.183.252.44

port: 9001

root (INFO): Probably discovered something, let me check; HTTP Keyserver 10.183.252.44:9001

emitted None

Geysign.sh 'geysign.sh' is ready



GNOME Keysign

- Directly transfers keys
- Sends encrypted email
- No MTA needed

STILL WAITING

<https://wiki.gnome.org/GnomeKeysign>

FOR PATCHES

memegenerator.net

Still waiting for patches



```
virtualenv /tmp/gnome-keysign  
/tmp/gnome-keysign/bin/pip install  
  'git+https://github.com/  
    muelli/geysigning.git  
    #egg=gnome-keysign'
```



```
sudo apt-get install python avahi-daemon python-avahi  
python-gi gir1.2-glib-2.0 gir1.2-gtk-3.0 python-dbus  
python-requests  
monkeysign  
python-qrcode  
gir1.2-gstreamer-1.0 gir1.2-gst-plugins-base-1.0 gstreamer1.0-plugins-bad
```

A historical black and white photograph showing a steam locomotive that has derailed and is tilted precariously against the side of a large, ornate building. The building has arched windows and signs that include "RESTAURANT", "CARE OF THE", and "DE LA GARE MONTMARTRE". A long ladder is leaning against the building. Several people in period clothing are standing on the street, observing the scene. The locomotive is numbered "221".

Thank you