COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Trust and Privacy III: PGP and the Web of Trust

ISTA 488: Privacy Technologies in Context

David Sidi (dsidi@email.arizona.edu)

Administration

- Mejores Dias
- Parcimonie

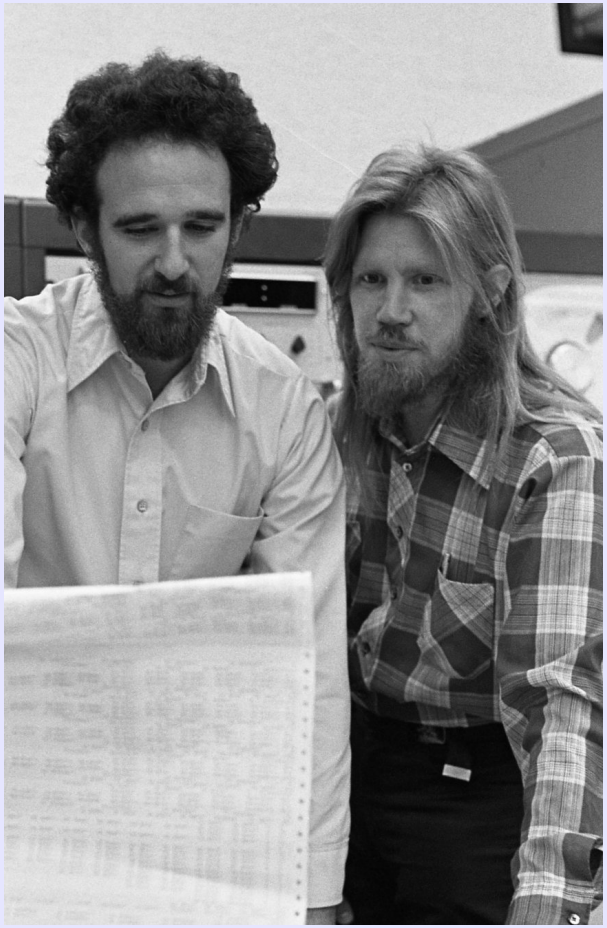
“cryptography can best be thought of as a mechanism for extending the confidentiality and authenticity of one piece of information (the key) to another (the message).” (POTL 34)

Key *management* problems for symmetric cryptosystems

- One way to share a secret: send it via a physically secured channel
- Keep close watch on your infrastructure
- Problem: expensive, hard
- Problem: lots of useful channels aren't physically secure
  - early example: radio, starting in 1905 (secret was in the rotor machines)
  - (old) crypt implements a rotor machine!

Asymmetric encryption

- Conventional symmetric cryptography has a shared secret key: how do you get it to both parties?
- First contribution of asymmetric cryptography was to key management, by providing a secure way to send a secret over an insecure channel



Asymmetric cryptosystems have key *binding* problems

- asymmetric cryptosystems can help with key distribution, but they still face a key binding problem
- public key is associated with a certificate, which gives some human-interpretable identifier ("user ID") to associate with the key
- what's to say that the certificate is right?

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Richard Stallman's Personal Site

If you participate in the commercial ritual of end-of-the-year presents, please avoid the digital products that would mistreat the people you give them to.

fsf.org/givingguide

Political Articles | Political Notes | Travel Experiences | Travel Photos | Fiction | Books | Sayings | Personal Ad | Humor | Non-Political Articles | RMS personal FAQ | GPG Key | Scientific Links | Airlines | Humorous Bio | Comics | Empire of the Megacorporations | There Ought to Be a Law | Media/Press/Bio | Links | Archive | Glossary | Anti-Glossary | Thanks |

Site search: go advanced

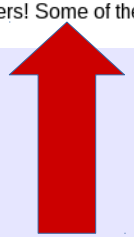
Send comments/questions about the search engine to: rms at gnu dot org

What's bad about: ***Airbnb*** | ***Amazon*** | ***Amtrak*** | ***Ancestry*** | ***Apple*** | ***Discord*** | ***Ebooks*** | ***Eventbrite*** | ***Evernote*** | ***Facebook*** | ***Google*** | ***Intel*** | ***LinkedIn*** | ***Lyft*** | ***Meetup*** | ***Microsoft*** | ***Netflix*** | ***Patreon*** | ***Pay Toilets*** | ***Skype*** | ***Spotify*** | ***Twitter*** | ***Uber*** | ***Wendy's*** |

RSS site feed for the most recent political notes and new material.

This is the personal web site of Richard Stallman.
The views expressed here are my personal views, not those of the Free Software Foundation or the GNU Project.

If you want to send me GPG-encrypted mail, do not trust key servers! Some of them have phony keys under my name and email address, made by someone else as a trick. See gpg.html for my real key.



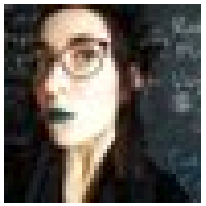
isis agora lovecruft
@isislovecruft

Neat. Someone faked the short keyid for my OpenPGP key. This is your regular reminder to only use full fingerprints. pic.twitter.com/D4iAEQdroN

11:08 AM - 16 Aug 2016

Search results for '0x2cdb8b35'

```
Type bits/keyID     Date       User ID
pub  4096R/2CDB8B35 2014-06-16 *** KEY REVOKED *** [not verified]
                               Isis! <isis@patternsinthevoid.net>
pub  4096R/2CDB8B35 2011-09-09 Isis <isis@torproject.org>
                               Isis! <isis@patternsinthevoid.net>
                               Isis <isis@leap.se>
                               Isis <isis@en.ciph.re>
                               Isis! <isis@riseup.net>
                               Isis! <isis@hackbloc.org>
                               Isis! <isis@globaleaks.org>
                               Isis Grimalkin <isis@patternsinthevoid.net>
```



isis agora lovecruft
@isislovecruft

Neat. Someone faked the short keyid for my OpenPGP key. This is your regular reminder to only use full fingerprints. pic.twitter.com/D4iAEQdroN

11:08 AM - 16 Aug 2016

yan @bcrypt 16 Aug 16
Replying to @isislovecruft
me too, with the same date. they even faked short IDs for some of the certifying keys.
View conversation

Some(@steveklabnik) @steveklabnik 16 Aug 16
Replying to @isislovecruft @torproject
this affected Linux too, Linus's key, for example
View conversation

alienghic@octodon.social 16 Aug 16

Search resul

```
Type bits/keyID     Date       Use
pub   4096R/2CDB8B35 2014-06-16 ***
                                Isis! <isis@patternsinthevoid.net>

pub   4096R/2CDB8B35 2011-09-09 Isis <isis@torproject.org>
                                Isis! <isis@patternsinthevoid.net>
                                Isis <isis@leap.se>
                                Isis <isis@en.ciph.re>
                                Isis! <isis@riseup.net>
                                Isis! <isis@hackbloc.org>
                                Isis! <isis@globaleaks.org>
                                Isis Grimalkin <isis@patternsinthevoid.net>
```

Evil 32: Check Your GPG Fingerprints

GPG usage has grown steadily while the tooling that supports it remains stagnant despite staggering hardware advancement. 32bit key ids were reasonable 15 years ago but are obsolete now. Using modern GPUs, we have found collisions for every 32bit key id in the WOT's (Web of Trust) strong set. Although this does not break GPG's encryption, it further erodes the usability of GPG and increases the chance of human error.

Stop using 32bit key ids

It takes 4 seconds to generate a colliding 32bit key id on a GPU (using scallion). Key servers do little verification of uploaded keys and allow keys with colliding 32bit ids. Further, GPG uses 32bit key ids throughout its interface and does not warn you when an operation might apply to multiple keys.

~~Check your fingerprints~~

~~Key servers do not use transport encryption (e.g. SSL) and GPG does not verify keys received when using `--recv-keys` leaving communicaiton with key servers vulnerable to MITM (man in the middle) or DNS attacks. GPG assumes you have manually checked your keys with `--fingerprint`.~~

Patched in new versions of GPG!

Asymmetric cryptosystems have key *binding* problems

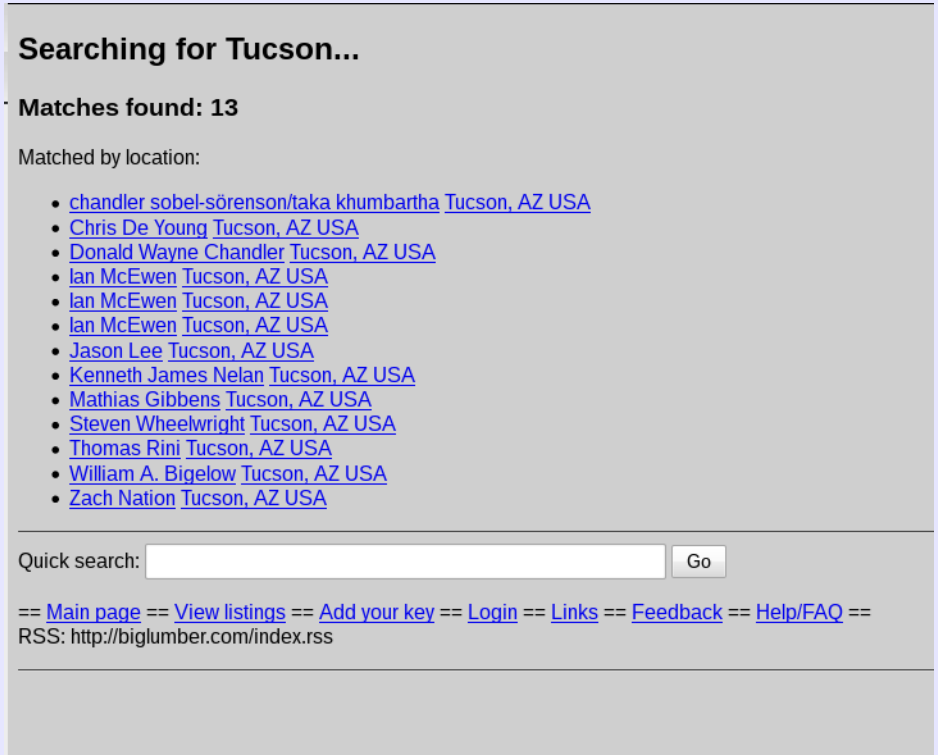
- public key is associated with a certificate, which gives some human-interpretable identifier to associate with the key
- who's to say that the certificate is right?
- PGP's answer: *nothing technical*. What addresses the key binding problem is social: the *web of trust*.

Sidenote on PGP

- Fashionable to be down on PGP:
  - “Why Johnny can’t encrypt”
  - GPG and Me
  - What’s the matter with PGP
  - Giving up on GPG
- PGP has many uses, though
- A power tool: You do need to know how to use it, but it’s not hard if you exercise some care

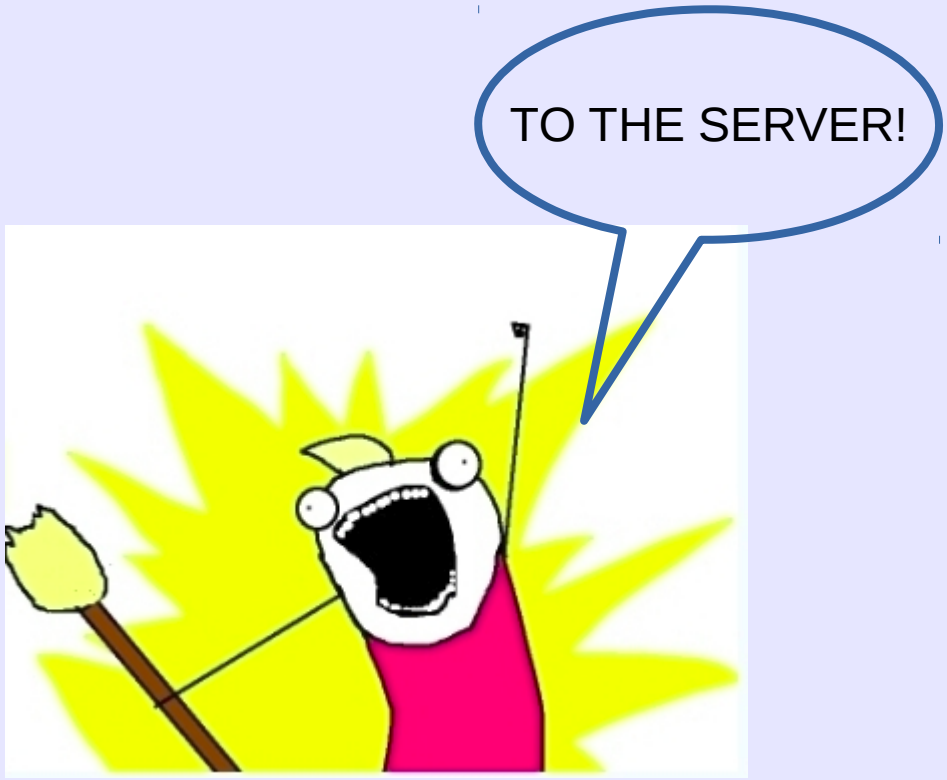
Making the WoT useful

- Meet in person
- keybase
- Multiple sources with archived copies: mailing lists, twitter, etc.
- biglumber



Hands-on: GPG

- GPG is the Gnu Privacy Guard



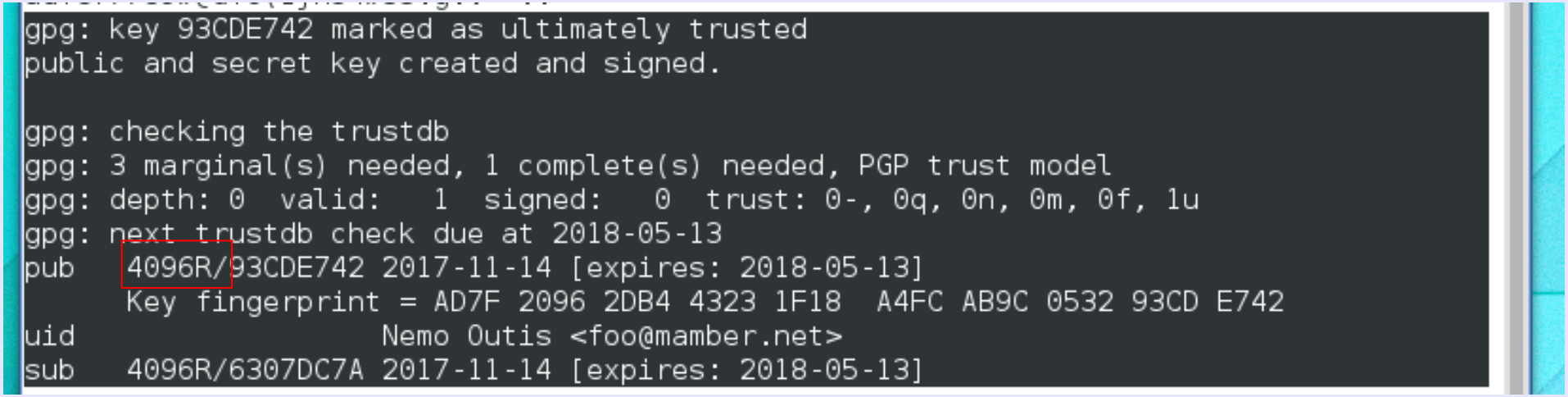
Hands-on: Generate a key

```
fa18-course:dsidi $cat ~/.gnupg/gpg.conf
personal-digest-preferences SHA256
cert-digest-algo SHA256
default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed
```

- Must be 4096, with SHA-2 rather than SHA-1

Hands-on: Generate a key

$ gpg --full-generate-key



WoT via GPG “debugging” output

```
gpg: key 93CDE742 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2018-05-13
pub   4096R/93CDE742 2017-11-14 [expires: 2018-05-13]
      Key fingerprint = AD7F 2096 2DB4 4323 1F18  A4FC AB9C 0532 93CD E742
uid                  Nemo Outis <foo@mamber.net>
sub   4096R/6307DC7A 2017-11-14 [expires: 2018-05-13]
```

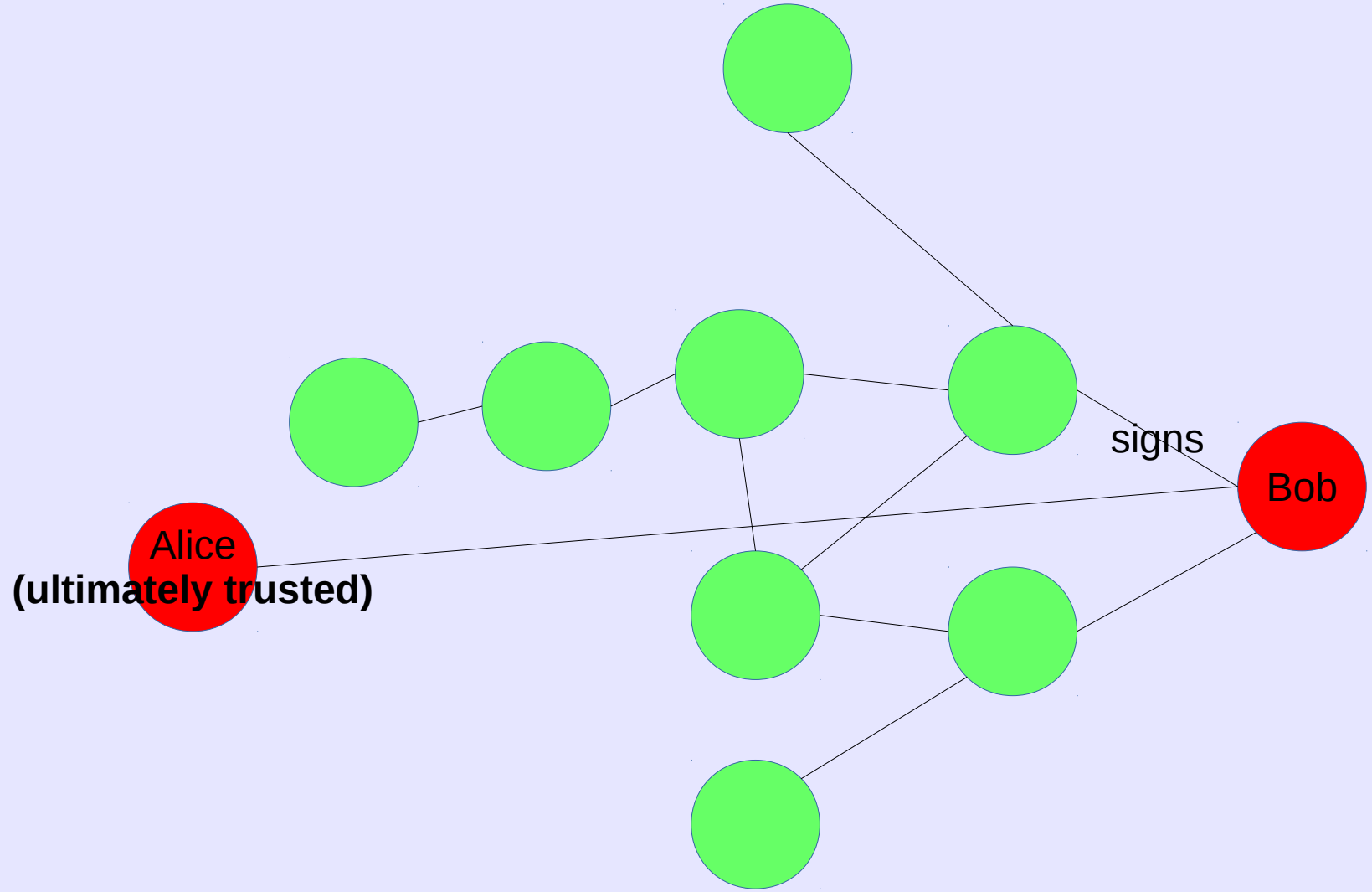
```
gpg: depth: 0  valid:   1  signed:  16  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:  16  signed: 115  trust: 1-, 1q, 1n, 1m, 12f, 0u
gpg: depth: 2  valid: 105  signed: 189  trust: 81-, 11q, 0n, 4m, 9f, 0u
gpg: depth: 3  valid:  29  signed: 120  trust: 19-, 9q, 0n, 0m, 1f, 0u
```

What is a fully valid key?

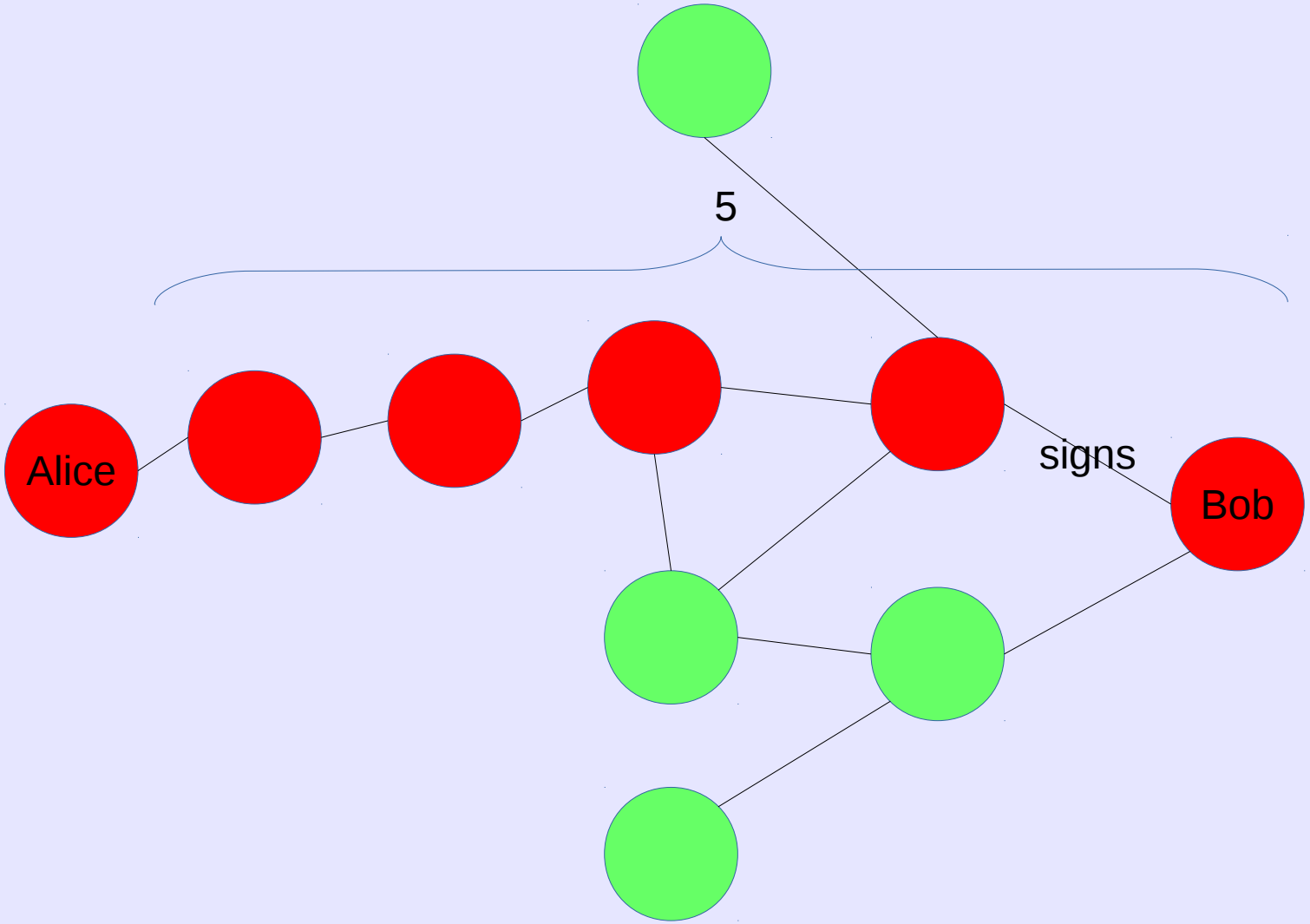
Introducers vs. Signed keys

- Essential to the PGP web of trust
- An **introducer** is trusted to some extent to verify other keys (not shared with keyservers)
- “Extent” here has two dimensions: **full** or **marginal**, and **depth**
- A signed key is one that has been certified to belong to the person identified by the user ID (shared with keyservers, unless you specify)

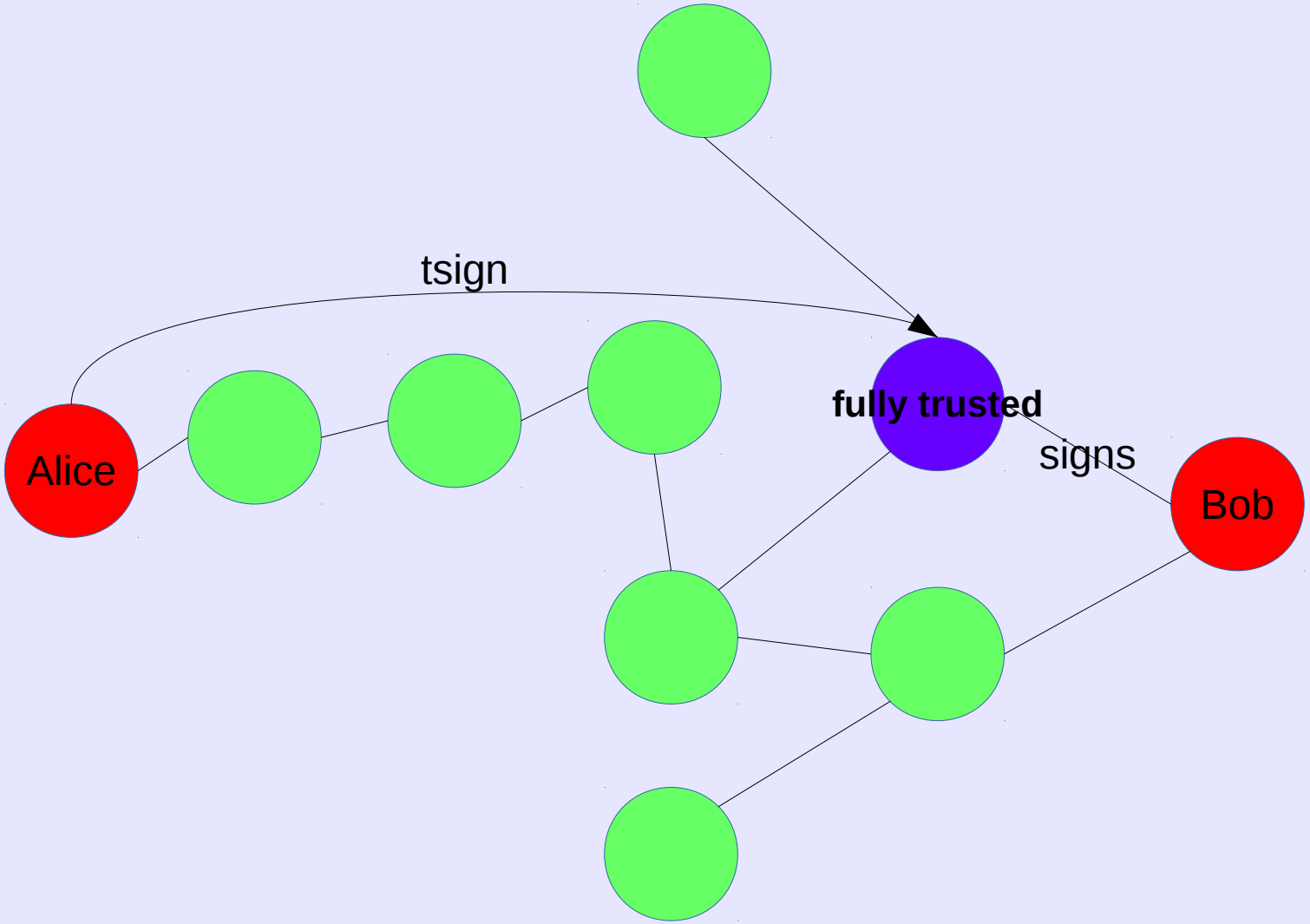
A fully valid key is signed with valid keys, and has a short signed path



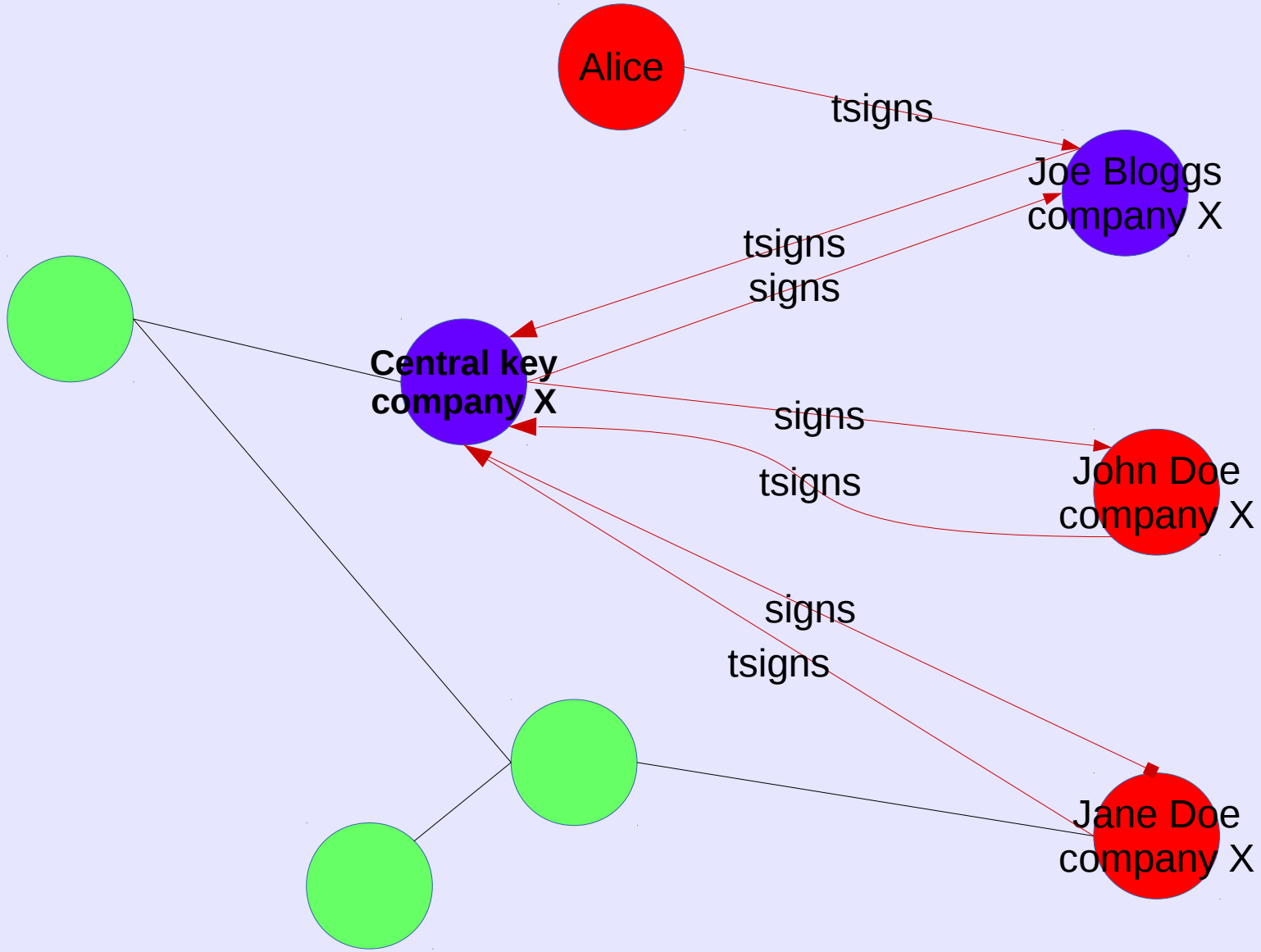
A fully valid key is signed with valid keys, and has a short signed path



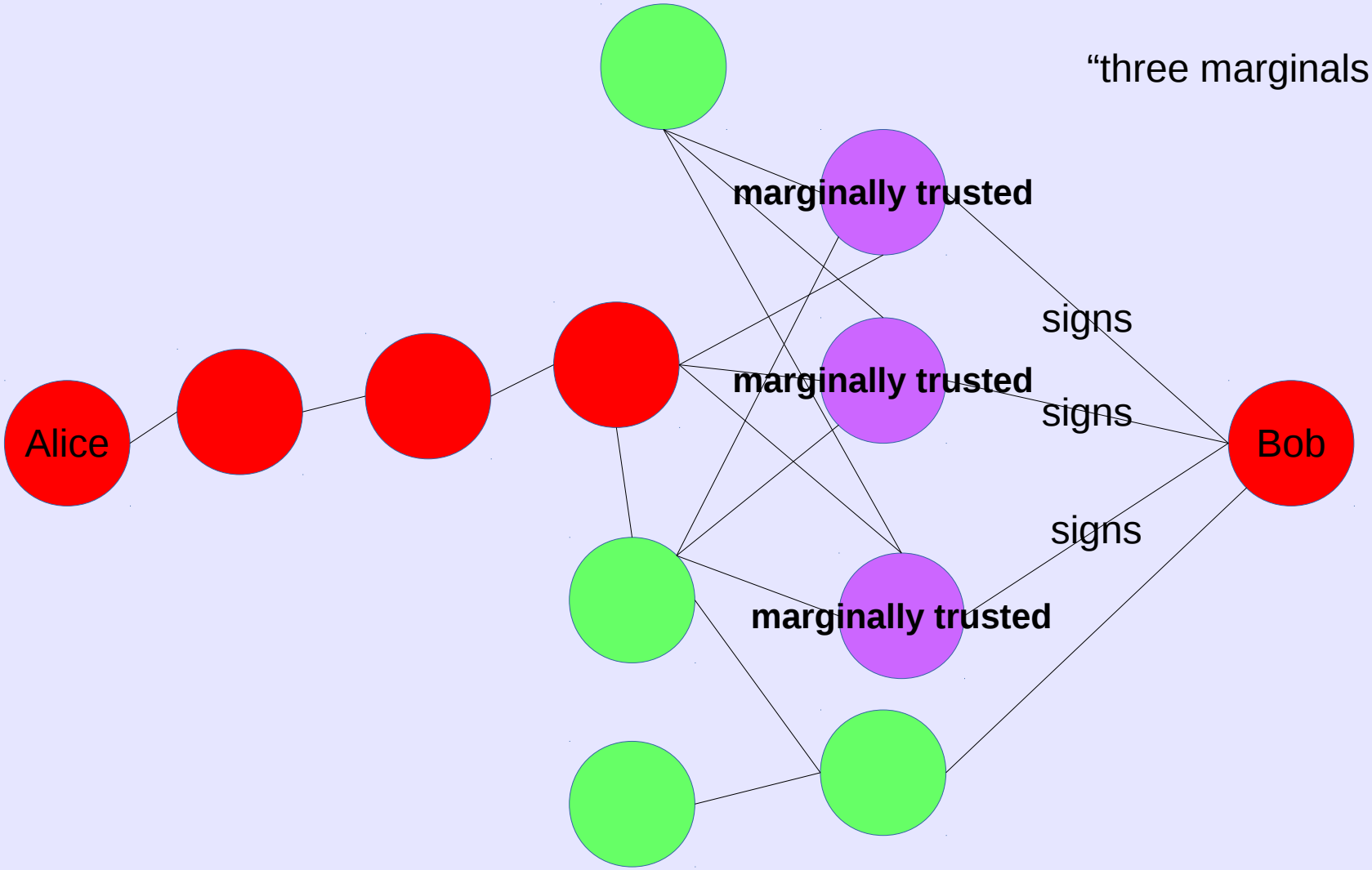
A fully valid key is signed with valid keys, and has a short signed path



What happens when you sign a key that has tsigned another key



A fully valid key is signed with valid keys, and has a short signed path



Trust models

--trust-model pgp|classic|direct|always|auto
Set what trust model GnuPG should follow. The models are:

pgp This is the Web of Trust combined with trust signatures as used in PGP 5.x and later. This is the default trust model when creating a new trust database.

classic
This is the standard Web of Trust as introduced by PGP 2.

direct Key validity is set directly by the user and not calculated via the Web of Trust.

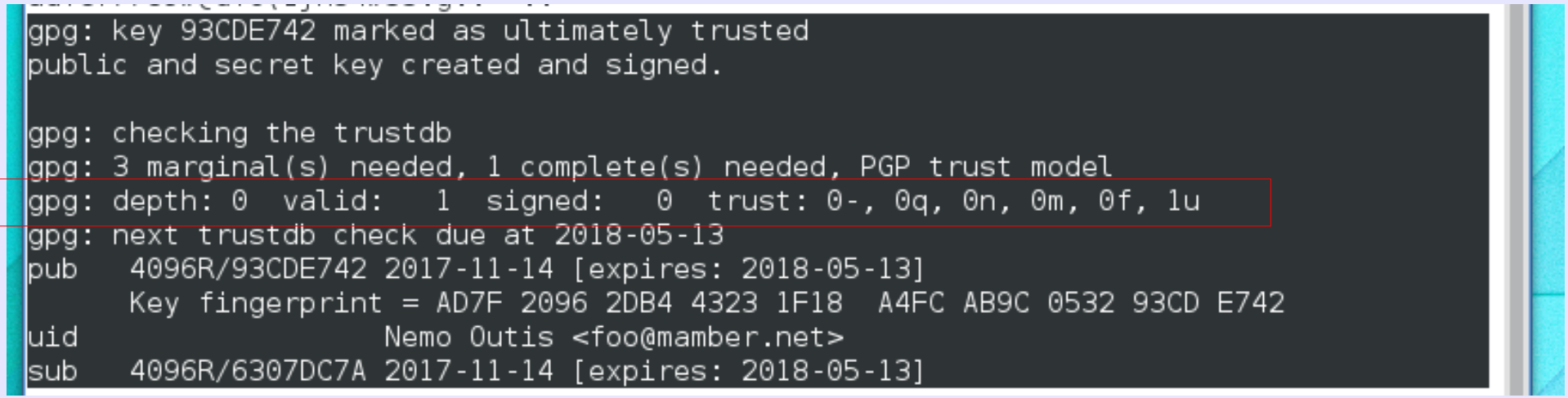
always Skip key validation and assume that used keys are always fully valid. You generally won't use this unless you are using some external validation scheme. This option also suppresses the "[uncertain]" tag printed with signature checks when there is no evidence that the user ID is bound to the key. Note that this trust model still does not allow the use of expired, revoked, or disabled keys.

auto Select the trust model depending on whatever the internal trust database says. This is the default model if such a database already exists.

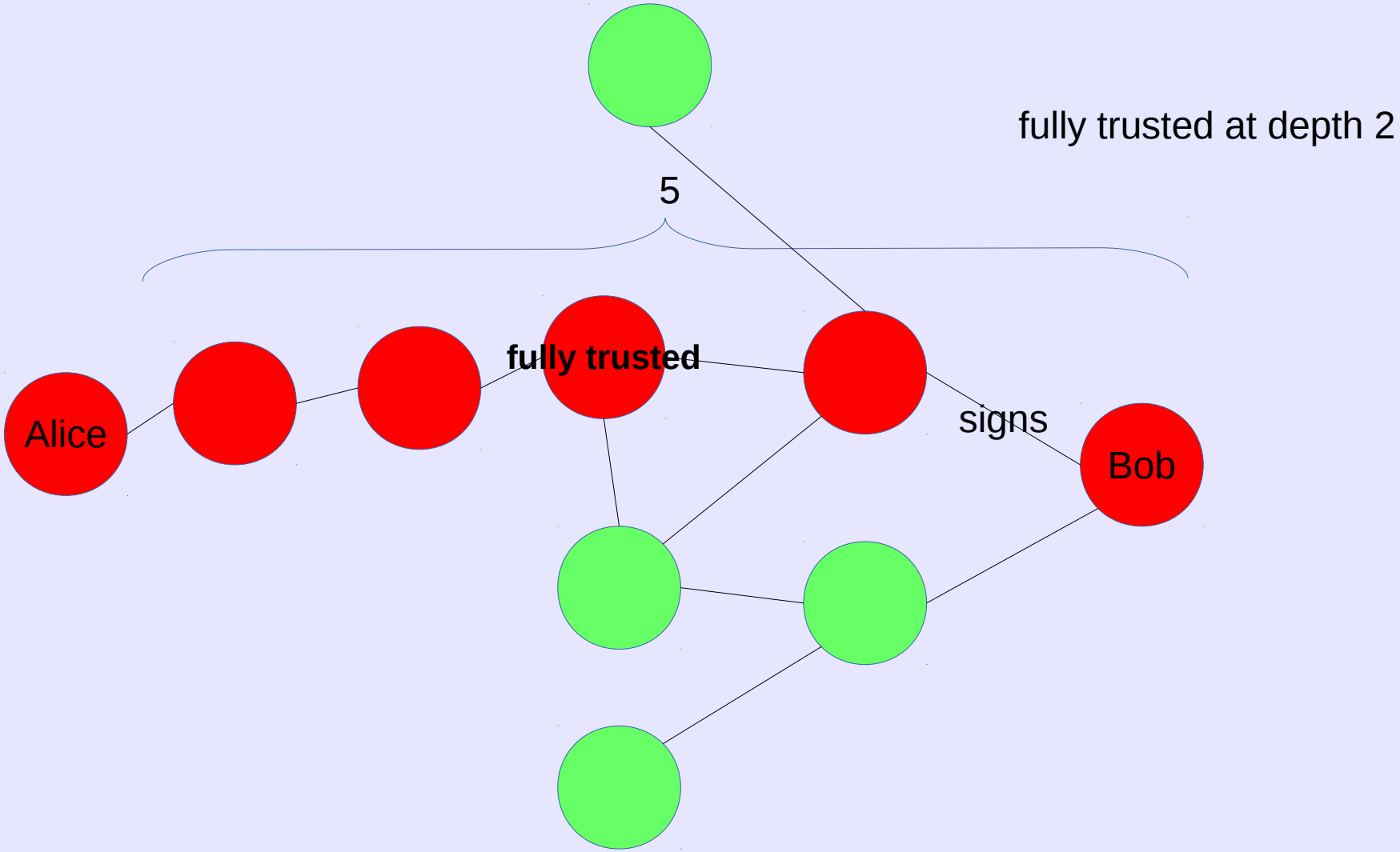
```
gpg: key 93CDE742 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2018-05-13
pub   4096R/93CDE742 2017-11-14 [expires: 2018-05-13]
      Key fingerprint = AD7F 2096 2DB4 4323 1F18  A4FC AB9C 0532 93CD E742
uid                  Nemo Outis <foo@mamber.net>
sub   4096R/6307DC7A 2017-11-14 [expires: 2018-05-13]
```

Hands-on: Generate a key

- An **introducer** is trusted to some extent to verify other keys
- Extent has two dimensions: **full** or **marginal**, and **depth**



A fully valid key is signed with valid keys, and has a short signed path



Hands on: sign my key

- Verify with government-issued identification
- `$ gpg --recv-key EEBA8245`
- `$ gpg --edit-key david@sidiprojects.us`

  (Prompt changes from ‘$’ to ‘gpg>’)
- `gpg>sign`
- or

  `gpg>tsign`

Hands on: Locally sign a key

```
$ gpg --edit-key <KEY-ID>
gpg>lsign
```

- Question: why do this instead of sign?

```
gpg> tsign
Really sign all user IDs? (y/N) y

pub  4096R/EEBA8245  created: 2014-03-14  expires: 2018-03-14  usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: E622 43FC 0F47 135B 28F0  02C4 8496 9123 EEBA 8245

     David Sidi <david@sidiprojects.us>
     David Sidi <davidsidi@gmail.com>
     David Sidi <dsidi@email.arizona.edu>

This key is due to expire on 2018-03-14.
Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I trust marginally
  2 = I trust fully

Your selection? 2

Please enter the depth of this trust signature.
A depth greater than 1 allows the key you are signing to make
trust signatures on your behalf.

Your selection? 1

Please enter a domain to restrict this signature, or enter for none.

Your selection?

Are you sure that you want to sign this key with your
key "Nemo Outis <foo@mamber.net>" (93CDE742)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Nemo Outis <foo@mamber.net>"
4096-bit RSA key, ID 93CDE742, created 2017-11-14
```