Understanding PGP and Using GPG

SIPB Cluedump Series / 17 November 2009

Steve Woodrow / `woodrow@mit.edu`

7901 C8DB 4886 EB01 4FC7 EBBA 8A10 C01C **F186 88B8**

Agenda

- Public key crypto
- PGP overview (history & key ideas)
- PGP concepts
- What PGP is good for/where it’s used
- GPG, a PGP implementation
- Setting up & using GPG
- Doing more with PGP

Disclaimer

- Protip: I am not a crypto pro
- This talk has no warranty and is provided "as-is"
- This has come out of much thought on the subject, but this is my first run, and there may be changes
- Tailoring to your situation will require some thought and understanding the risks — think!

Crypto primer

- What is cryptography? Encryption?
  - Confidentiality: $p \Rightarrow E_k \Rightarrow c$
  - Mathematical operations: security comes from secrecy of key
  - Use of “large enough” numbers and specific operations makes reversing the math difficult

Symmetric key

- One key for encryption and decryption
    - What we’ve known as crypto for most of history
- Fast, but the key must be confidentially shared beforehand
- Common Algorithms: AES, DES, Blowfish, etc.

Public (asymmetric) key

- Math allows two keys to be used
  - Encryption by one key is decrypted by the other key
- Called “public key” because one key typically made public and the other kept secret — provides nice properties
- Relatively slow
- Common algorithms: RSA, ElGamal, DSA

Hybrid cryptosystem

- Benefits of both algorithms
- Generation of a one-time *session key*
- Public Key used to exchange session key
- Symmetric Key used to exchange data

Digital signatures

- Provides integrity — assurance a message was not tampered with
- Provides authenticity — assurance a message came from its purported author
  - Non-repudiation
- Implemented as encryption of a compact, unique representation of the message (hash)

Hash/message digest functions

- Generates a fixed-length output statistically unique to the input
    - Difficult to invert
    - Collision resistant
- Common algorithms: SHA, MD5, RIPEMD
- Difference between authenticity & integrity

A bit of PGP history*

- Politics — Gov. bill outlawing strong crypto, clipper chip telephones and FBI key escrow
- Phil Zimmerman (prz)— software developer & anti-nuclear/military policy activist
  - “Pretty Good Privacy”
- Originally “released” in 1991
- Focused mainly on email security

Export Controls

- Crypto > 40 bits was a munition under ITAR and couldn’t leave the US
- First Amendment > ITAR
    - MIT Press published a 900 pp. book of the source for OCRing

MIT’s Involvement

- hal, jis, jdb and others worked with prz to release a non-encumbered version
- MIT was involved in maintaining and releasing new versions
- simsong wrote a pretty good book

Up to the present

- Standardization: OpenPGP
  - RFCs 1991, 2440, 4880
- PGP® Corporation
- New legal and policy environment

PGP’s key ideas

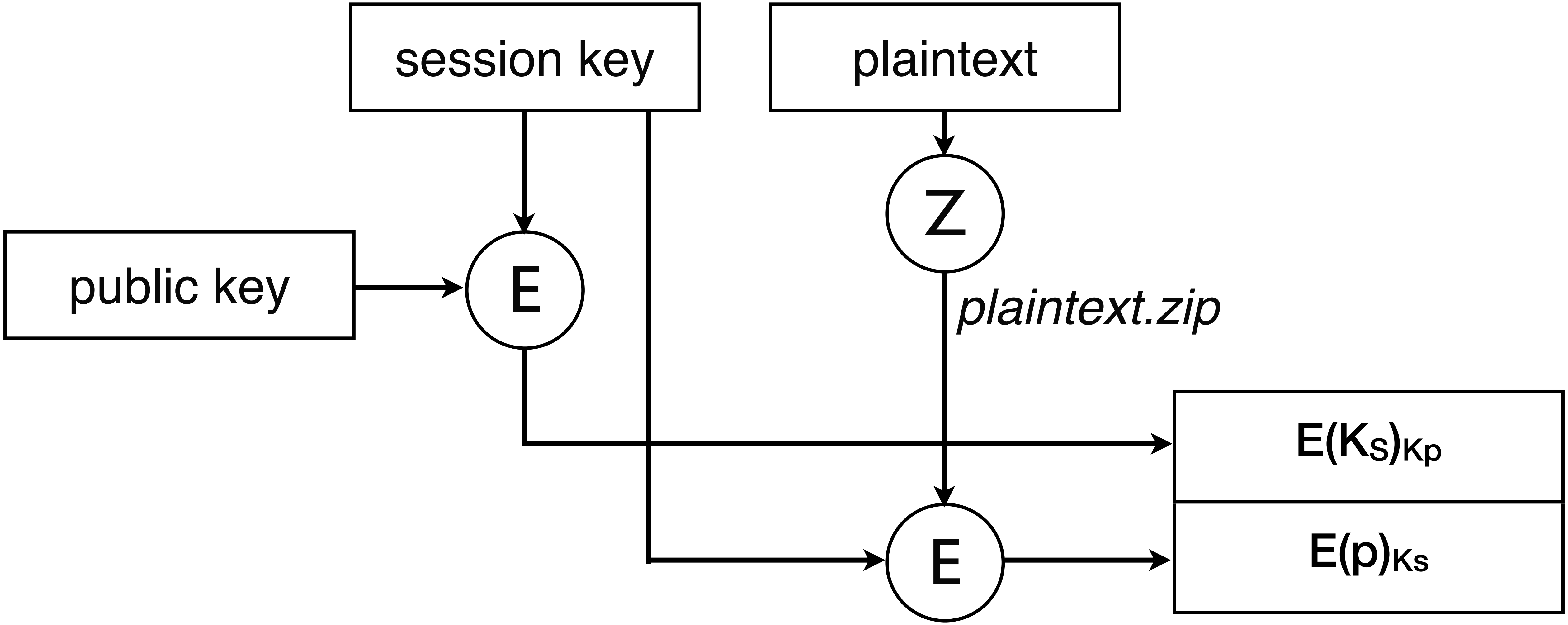
- Public key cryptography
- No central point of control
- Distributed web of trust
- Strong crypto for everyone

Basic PGP operations

- Symmetric Encryption/Decryption
- Public Key Encryption/Decryption
  - ZIP/etc. compression first
- Sign/Verify
  - canonicalizes text with CRLF line endings
- ASCII armor — base64 + checksum

General Encrypted Message

- Every PGP message is a series of packets



ASCII Armor

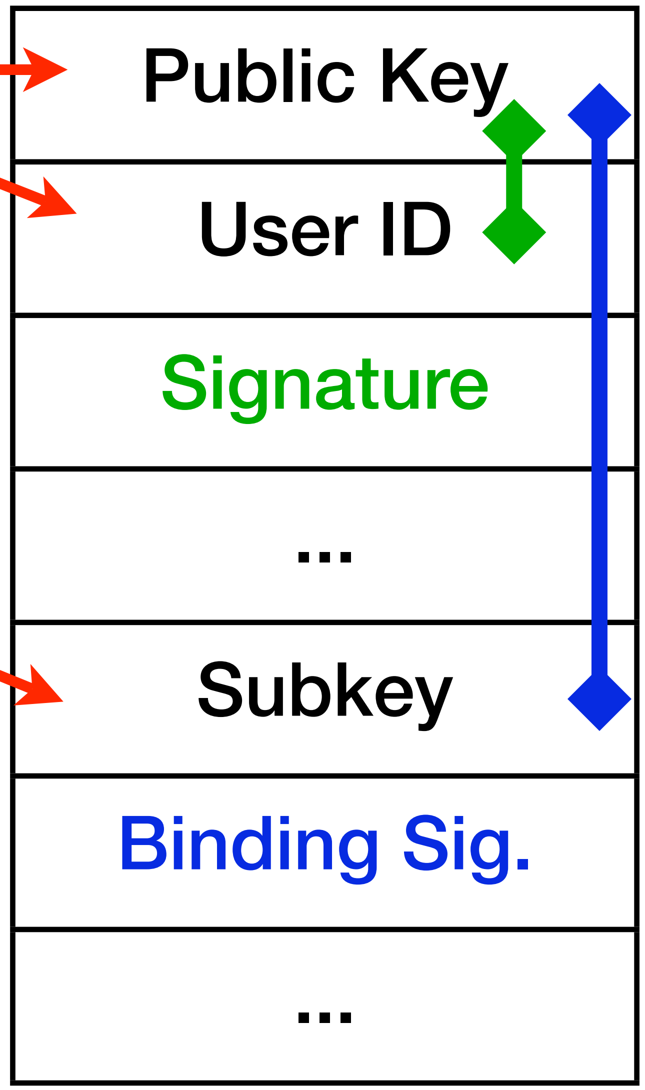
```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.9 (Darwin)

jA0ECQMC6+Nriien1Rhg0koBxK+eAtTlzxAh2Aw9fRw+HR+/Cf59xzwkp2NMYDWS
TU/gD/SH9xKr7lxWAMnYGHnm17O7BEQ2lM6FjQ/+DeaF6Iek9Giu1OXQOg==
=7sWL
-----END PGP MESSAGE-----
```

What about keys?

- Keyring — a collection of keys of interest, at least one public and one secret
- What's in a key? More packets:
  - Key material
  - uid, algorithm preferences, etc.
  - signatures
  - subkeys
- Look inside with `pgpdump`

Inside a PGP key

```
pub   4096R/F18688B8 2009-08-23
uid                  Stephen Woodrow <srwoodrow@gmail.com>
uid                  Stephen Woodrow <woodrow@mit.edu>
uid                  Stephen Woodrow <woodrow@csail.mit.edu>
sub   2048R/7C46749C 2009-08-23
sub   2048g/4899B1CF 2009-08-23
```



Public Key

User ID

Signature

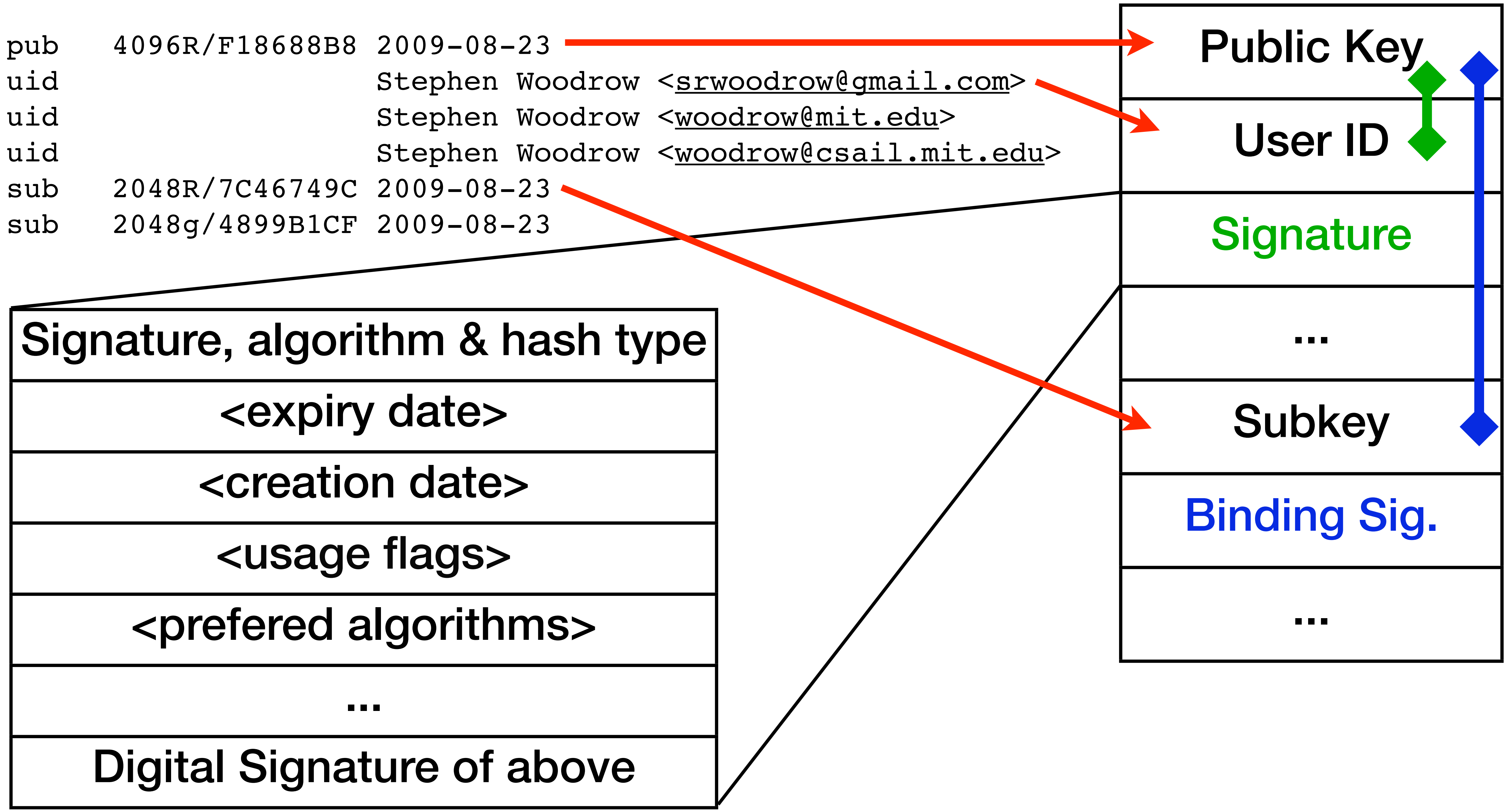
...

Subkey

Binding Sig.

...

Inside a PGP key



```
pub   4096R/F18688B8 2009-08-23
uid                  Stephen Woodrow <srwoodrow@gmail.com>
uid                  Stephen Woodrow <woodrow@mit.edu>
uid                  Stephen Woodrow <woodrow@csail.mit.edu>
sub   2048R/7C46749C 2009-08-23
sub   2048g/4899B1CF 2009-08-23
```

Public Key

User ID

Signature

...

Subkey

Binding Sig.

...

Signature, algorithm & hash type

<expiry date>

<creation date>

<usage flags>

<prefered algorithms>

...

Digital Signature of above

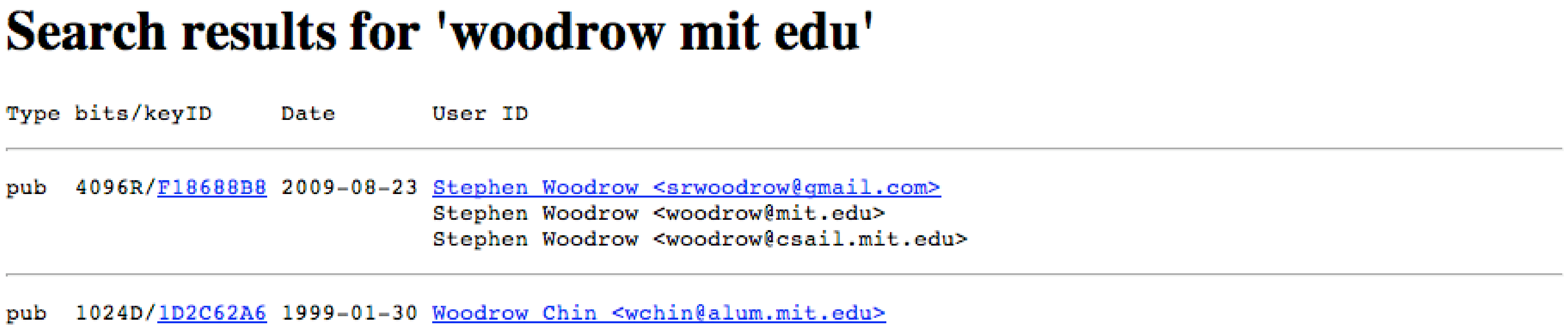
A note about secret keys

- Secret key material is encrypted by a symmetric algorithm (typically CAST5) to keep it safe on disk
- Your hashed passphrase (S2K) as key

Key Distribution

- How do I get other people’s keys?
  - Trading beforehand is inconvenient
- How about a key directory? PGP keyservers
- http://pgp.mit.edu and others (pools)
  - synchronize with each other
  - keys updated, but can’t be deleted
    - must be *revoked*

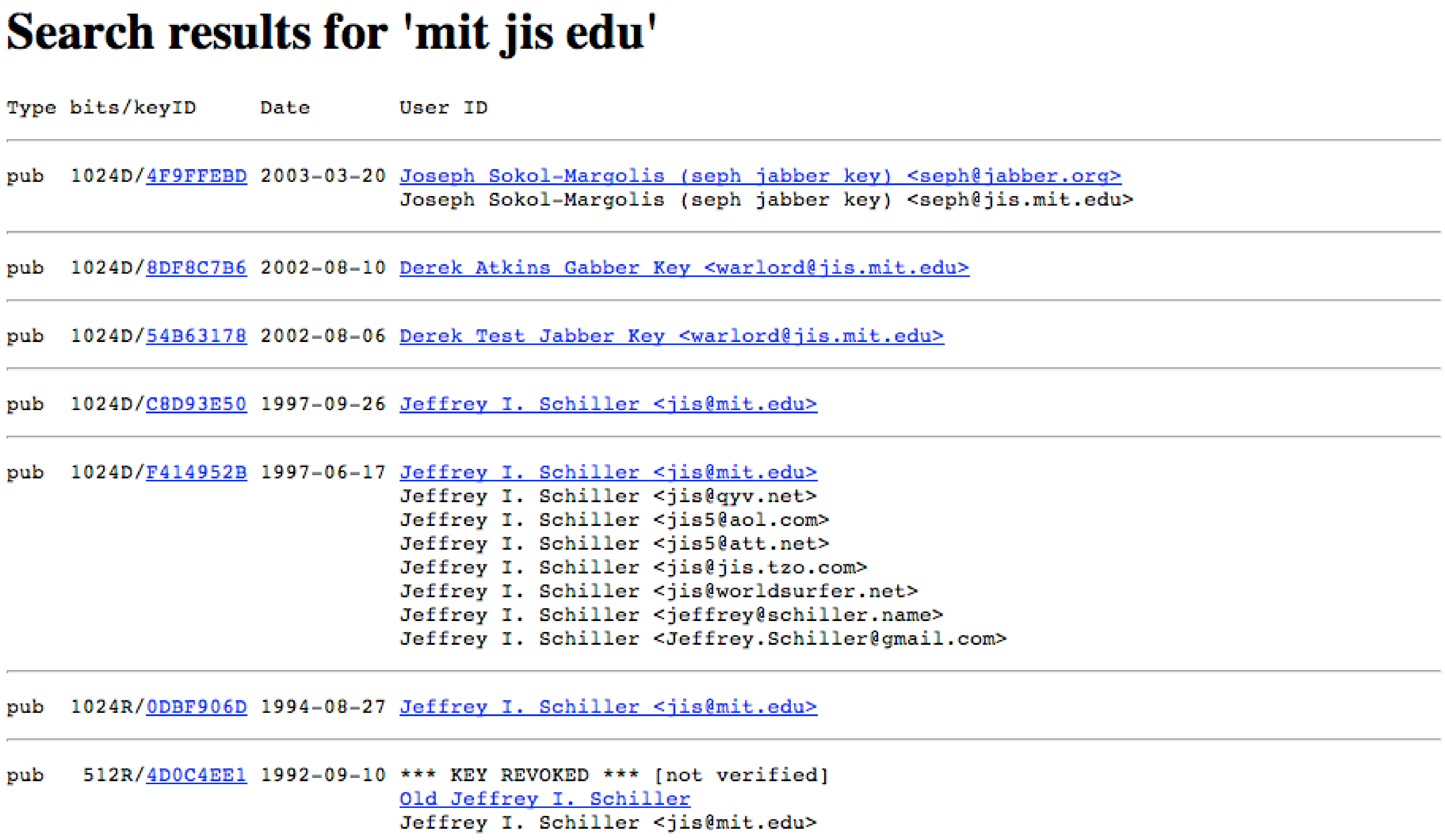
http://pgp.mit.edu



Search results for 'woodrow mit edu'

```
Type bits/keyID     Date       User ID
---------------------------------------------------------------------------------
pub  4096R/F18688B8 2009-08-23 Stephen Woodrow <srwoodrow@gmail.com>
                               Stephen Woodrow <woodrow@mit.edu>
                               Stephen Woodrow <woodrow@csail.mit.edu>
---------------------------------------------------------------------------------
pub  1024D/1D2C62A6 1999-01-30 Woodrow Chin <wchin@alum.mit.edu>
```

http://pgp.mit.edu



Search results for 'mit jis edu'

```
Type bits/keyID     Date       User ID

pub  1024D/4F9FFEBD 2003-03-20 Joseph Sokol-Margolis (seph jabber key) <seph@jabber.org>
                               Joseph Sokol-Margolis (seph jabber key) <seph@jis.mit.edu>

pub  1024D/8DF8C7B6 2002-08-10 Derek Atkins Gabber Key <warlord@jis.mit.edu>

pub  1024D/54B63178 2002-08-06 Derek Test Jabber Key <warlord@jis.mit.edu>

pub  1024D/C8D93E50 1997-09-26 Jeffrey I. Schiller <jis@mit.edu>

pub  1024D/F414952B 1997-06-17 Jeffrey I. Schiller <jis@mit.edu>
                               Jeffrey I. Schiller <jis@qyv.net>
                               Jeffrey I. Schiller <jis5@aol.com>
                               Jeffrey I. Schiller <jis5@att.net>
                               Jeffrey I. Schiller <jis@jis.tzo.com>
                               Jeffrey I. Schiller <jis@worldsurfer.net>
                               Jeffrey I. Schiller <jeffrey@schiller.name>
                               Jeffrey I. Schiller <Jeffrey.Schiller@gmail.com>

pub  1024R/0DBF906D 1994-08-27 Jeffrey I. Schiller <jis@mit.edu>

pub   512R/4D0C4EE1 1992-09-10 *** KEY REVOKED *** [not verified]
                               Old Jeffrey I. Schiller
                               Jeffrey I. Schiller <jis@mit.edu>
```

What about trust?

- How do I know which one “is” Jeff Schiller?
    - Nothing stops Mallory from creating a key with uid “Jeff Schiller <jis@mit.edu>”
- Each key has a unique “fingerprint” — the hash of the public
    - I could ask jis (in person) which fingerprint is correct
    - I could sign a copy of his key to note this to me and to others who trust me

Trust-related concepts

- Validity: does the key really belong to the person named on it?
- Trust: Do I trust the honesty and judgement of the keyholder to claim that others' keys are valid?
  - "ownertrust"

Trust in PGP is an individual decision

- MIT’s X.509 Certs:
  - absolute trust of the root CA
  - root certifies all certs as valid
- PGP:
  - You certify some keys as valid
  - You control who you trust to verify the keys of others

The Web of Trust

- Authentication is difficult if you only trust verified friends
- By trusting some friends, you can treat their certification of other keys as “probably” valid
- You may find a chain of trust from to you a desired recipient

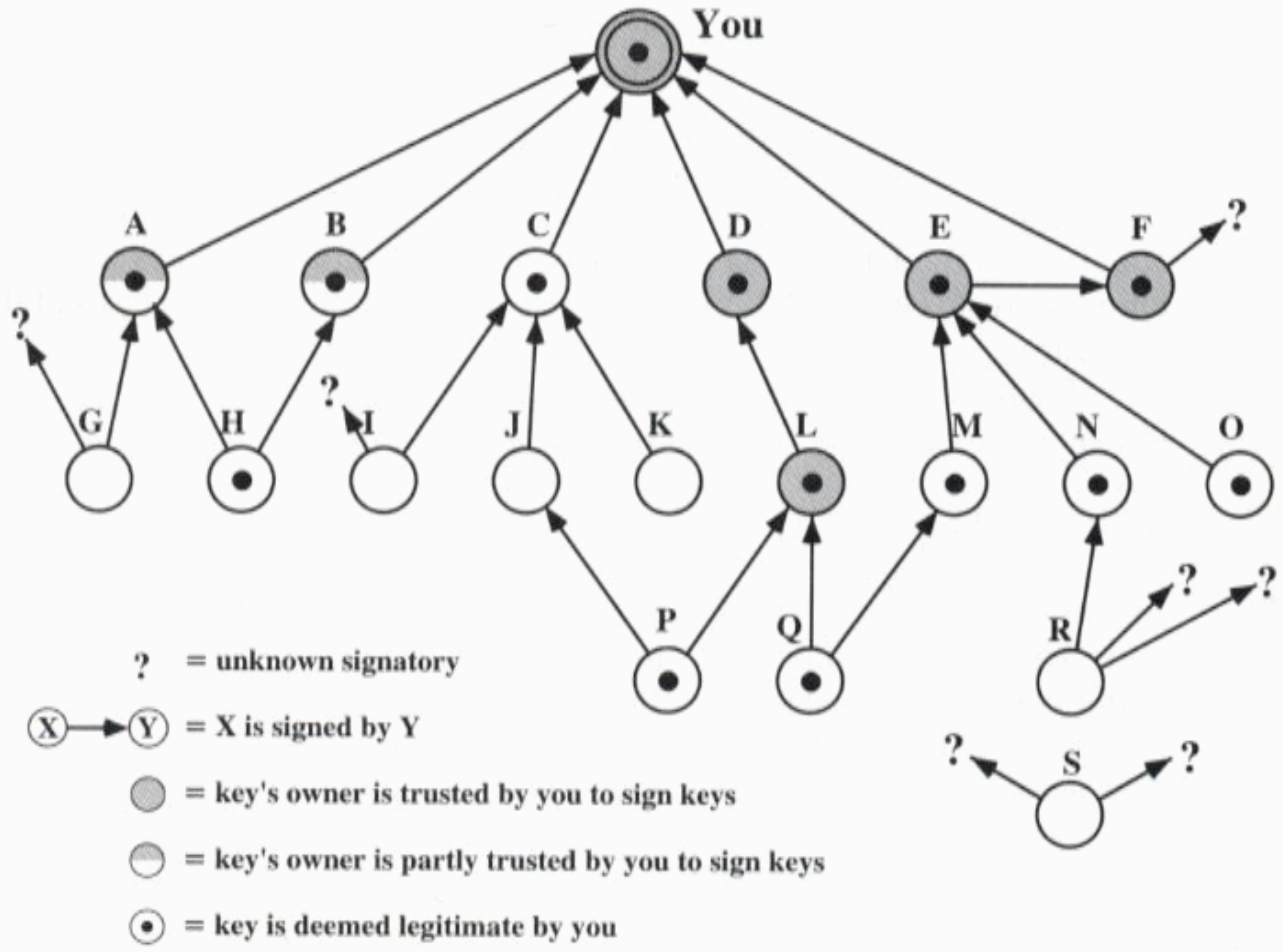
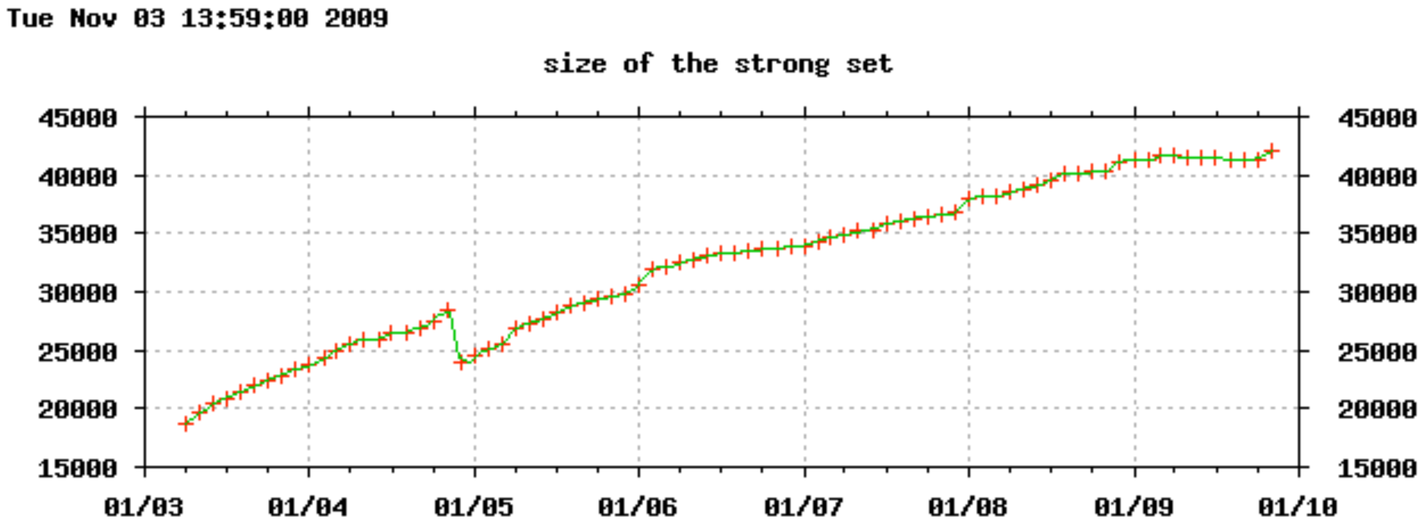

You
A
B
C
D
E
F
?
?
G
H
?
I
J
K
L
M
N
O
P
Q
R
?
?
?
S
?
? = unknown signatory
X → Y = X is signed by Y
= key's owner is trusted by you to sign keys
= key's owner is partly trusted by you to sign keys
= key is deemed legitimate by you

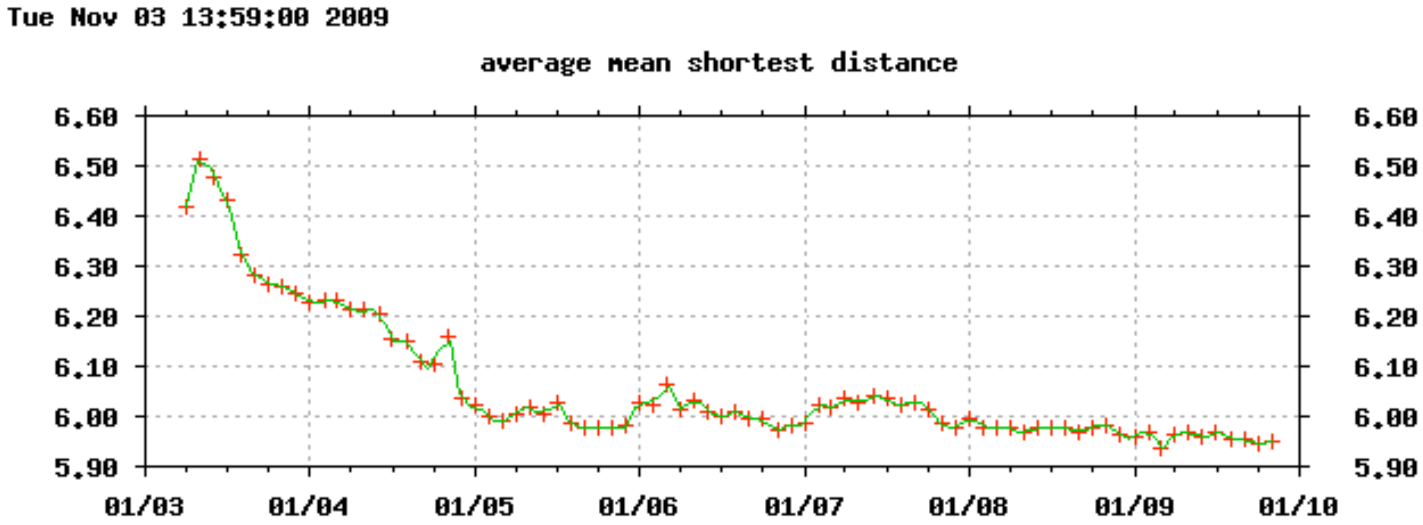
Figure 15.7 PGP Trust Model Example

From http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/PGP.html, from William Stallings, Cryptography and Network Security, Principles and Practice, Prentice Hall, 1999. Copyright 2003 by Pearson Education Inc

The “Strong Set”

- The largest strongly connected graph on public keyservers
  - Bidirectional path between every pair of keys





The “Strong Set”

- Find a path through the strong set: http://people.cs.uu.nl/henkp/henkp/pgp/

- Publicity of signing others keys allows the strong set to be constructed, but it also allows inference about associations
    - potential privacy violation

What does it mean to sign a key?

- Certifying the key is valid
- Steps to verify validity vary greatly
    - Some people have policies, though it’s pretty intense: http://www.nieveler.org/PGP/pgp.htm
- Degrees of verification are allowed to be specified, but not always used and are a bit vague

0x10: Generic certification of a User ID and Public-Key packet. The issuer of this certification does not make any particular assertion as to how well the certifier has checked that the owner of the key is in fact the person described by the User ID.

0x11: Persona certification of a User ID and Public-Key packet. The issuer of this certification has not done any verification of the claim that the owner of this key is the User ID specified.

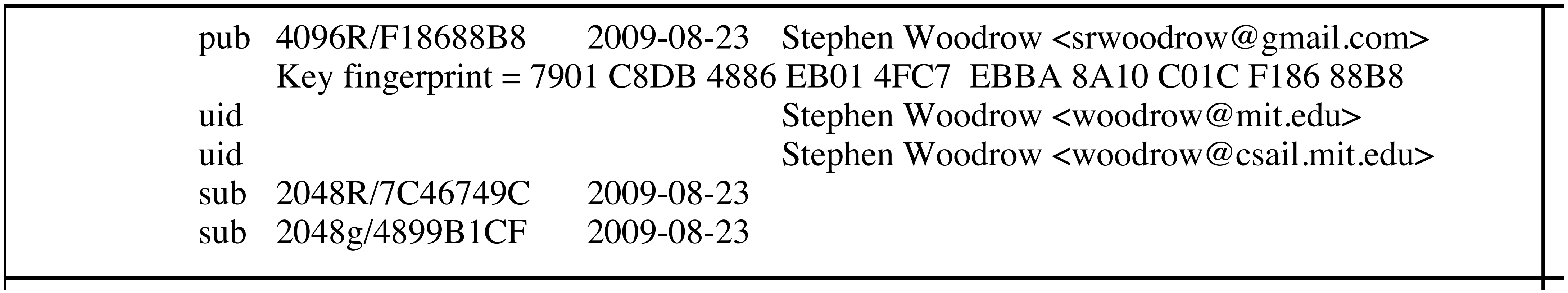
0x12: Casual certification of a User ID and Public-Key packet. The issuer of this certification has done some casual verification of the claim of identity.

0x13: Positive certification of a User ID and Public-Key packet. The issuer of this certification has done substantial verification of the claim of identity.

Most OpenPGP implementations make their "key signatures" as 0x10 certifications. Some implementations can issue 0x11-0x13 certifications, but few differentiate between the types.

How to sign a key

- Bring paper with a double-checked key fingerprint and UID — probably not a laptop
- Demand whatever you feel you need to verify someone’s identity — typically a couple of pieces of ID (one photo)
- If you’re satisfied with identity, go home and pull their public key from a keyserver
- Does fingerprint match? Sign their key.
- Export their signed public key and encrypt it with their key to verify their control of the associated private key



```
pub   4096R/F18688B8       2009-08-23   Stephen Woodrow <srwoodrow@gmail.com>
      Key fingerprint = 7901 C8DB 4886 EB01 4FC7  EBBA 8A10 C01C F186 88B8
uid                                     Stephen Woodrow <woodrow@mit.edu>
uid                                     Stephen Woodrow <woodrow@csail.mit.edu>
sub   2048R/7C46749C       2009-08-23
sub   2048g/4899B1CF       2009-08-23
```

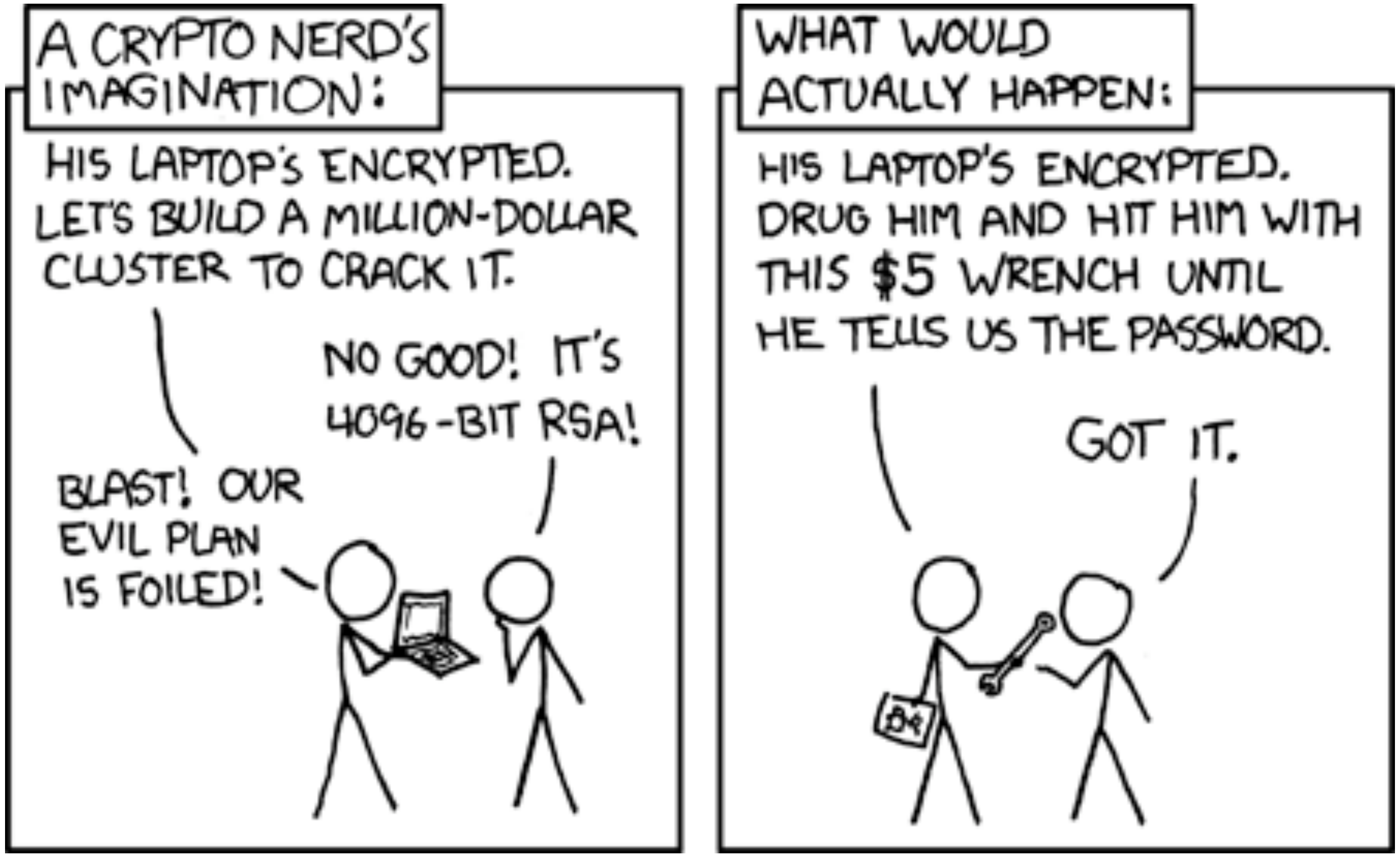
Other signing thoughts

- When to sign keys?
  - Individually
  - Keysigning parties are efficient (stay tuned)
- Not everyone has the same standards
  - This is where trust and judgement come in
- Made a bad decision?
  - Revoke signature or adjust trust

What is PGP good for?

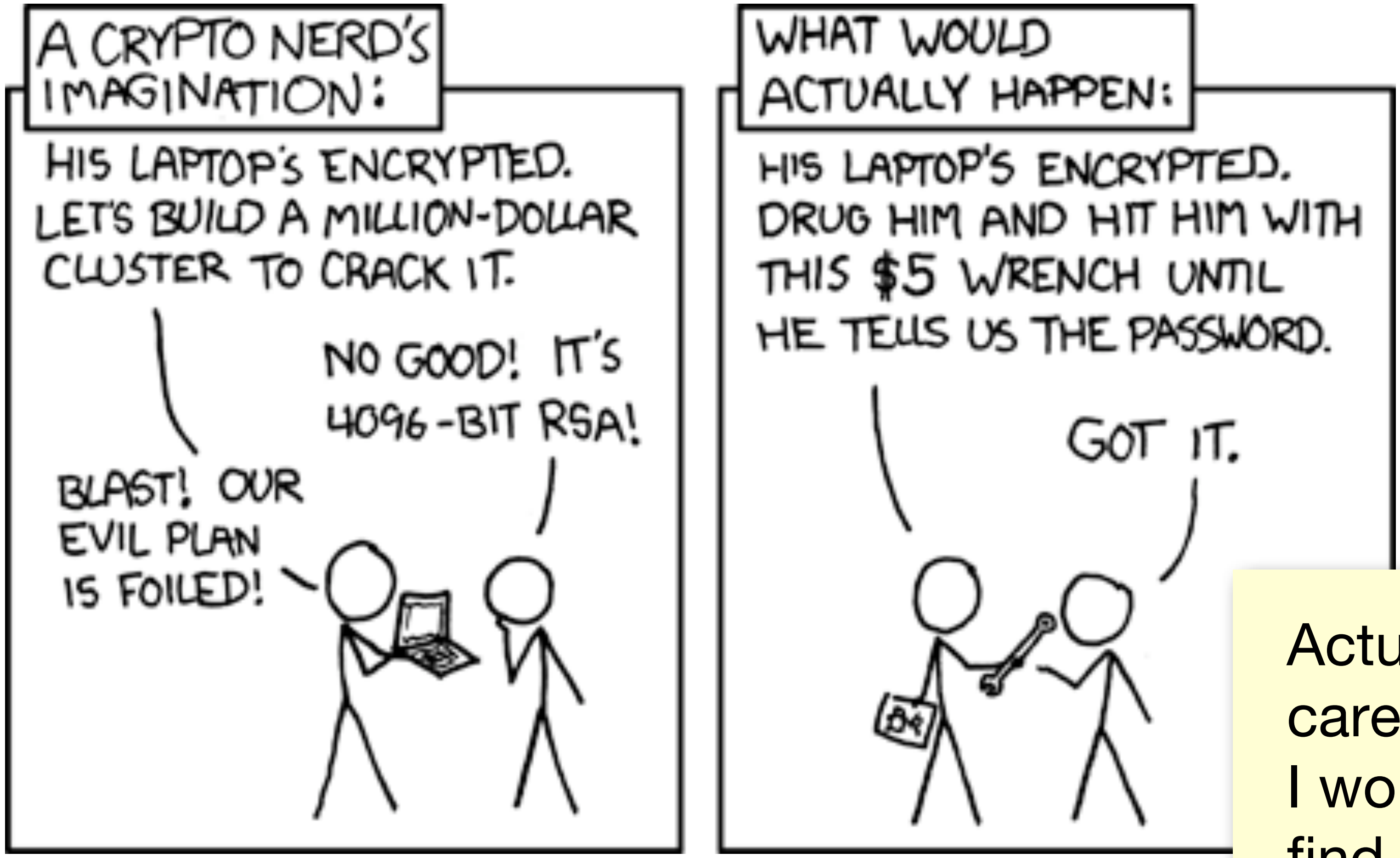
- Encrypting/decrypting & signing files and mail
  - passwords & other sensitive data
  - sign things to CYA?
- Debian/Ubuntu/Debathena
  - uploading packages & securing APT
- Signing code in repository: git-tag -s
- Signing software you release (better than MD5 or SHA1)

Threat Model


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

CC-BY-NC Randall Munroe (http://xkcd.com/538/)

Threat Model


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

CC-BY-NC Randall Munroe (http://xkcd.com/538/)

Actual actual reality: nobody cares about his secrets. (Also, I would be hard-pressed to find that wrench for $5.)

Assumptions & Threats

- Assume source wasn’t tampered with
  - You can inspect source of major implementations
- Assume GPG binaries haven’t been tampered with
- Assume your machine is trusted & files you depend on can’t be modified
- Other threats?
- Only you know your threat model: think hard and act accordingly

Implementations

- GPG: popular, free, OpenPGP compliant (1.4 vs. 2)
- PGP Corp: non-free, desktop software, business-oriented
- Graphical frontends for gpg: kgpg, gnome-gpg, firegpg
- MUA/Mail clients: Enigmail, GPGMail, etc.
- `apt-cache search "gpg|pgp|gnupg"` (though there's some overlap)

References

- S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly, 1995.
- P. Zimmerman. *The Official PGP User's Guide.* MIT Press, 1995.