PGP SUCKS

“look: if we’re going to be thrown into an authoritarian distopia, people need to know how to protect themselves. ideally not with PGP”. . .

Eric C.“echarlie” Landgraf

VTLUUG

November 3, 2020

Contents

What Is PGP

- “Pretty Good Privacy”
- Piece of software and corresponding open standard for authenticated and confidential messaging
  - Long-lived identity keys
  - “Web of Trust” — the network of people who have verified each others’ identities.
  - Strong symmetric and asymmetric cryptography
- Can be used for local encryption of files, and more often for PGP/MIME, that is encrypted and authenticated messaging.

History of “Pretty Good Privacy”

- First written by Phil Zimmerman in 1991 with symmetric algorithms for anti-nuclear activists. Released 1991-06-05.
- Because it used strong “weapons-grade” crypto, Zimmerman was investigated by the US Gov’t for illegal munitions export in early 1993. (PGP used keys > 40bits)

History of “Pretty Good Privacy”

- PGP 2, later standardized by informational RFC 1991 which was published in 1996, was based on RSA; it was developed by Viacrypt, who had licence for RSA and commercial rights to PGP. PGP 4 was later released by Viacrypt as well.
- PGP 3 was first released in 1996 and contained DSA and ElGamal asymmetric algorithms, as well as CAST-128; all were unencumbered by patents. Later released as PGP 5 in 1997

OpenPGP

- This was formally standardized through the IETF in RFC 2440, based on the PGP 5 implementation, and was published in 1998
- Further, this was revised with several later RFCs, including RFC 4880 in 2007
- Also notable is the development of PGP/MIME with RFC 2015 and then RFC 3156 in 1996 and 2001, respectively

Versions of PGP

- PGP 1 — Completely outdated and irrelevant
- PGP 2 — First PGP incorporating RSA; introduced web of trust. Largely supplanted by PGP 5 using stronger algorithms.
- PGP 3 / PGP 5 (also known as OpenPGP) — introduced DSA and ElGamal keys, which were common for most PGP use until the expiration of RSA patents.
- GnuPG — Most common implementation of the OpenPGP standard, including libraries for integration into other software.

PGP in the Real World

- If you run a common linux distro, you use PGP
- apt, pacman, and yum all rely on PGP to verify package integrity
- *Most* software developers sign their software releases

What’s this “Web of trust” thing?

- Web of Trust (WoT) is a concept pulled from social networks (the sociology type, not Facebook).
- Basically, human trust models don’t reflect machine trust models—the WoT bridges this by putting the onus on the user to verify identities of others.
- No reliance on CAs or centralized authorities.
- This has the benefit that you can choose who you trust, and how much you trust them, but trust can be automatically computed.

What about keysigning?

- For the WoT to work, you have to verify identity of other users.
- This means you hold “Keysigning parties” to do just that.
- For every key you verify, you’re supposed to sign the key, and generally put it on public Keyservers for lookup.

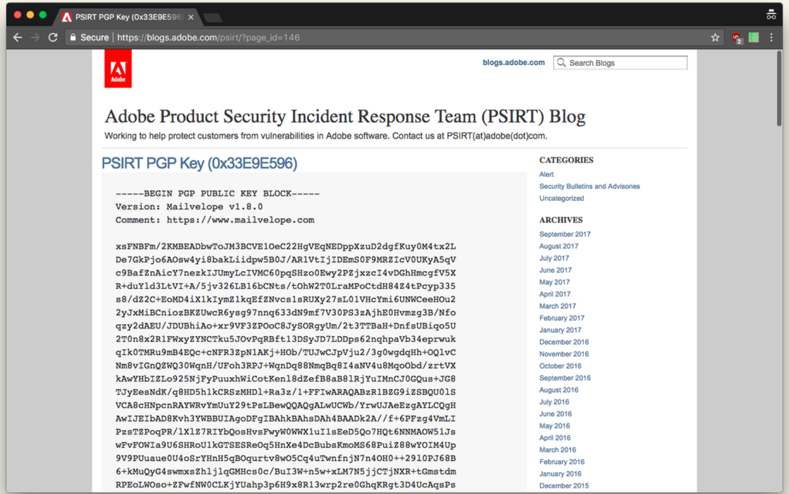
Modes of operation

- PGP has 3 modes of operation for asymmetric keys: Signing, Encrypting, and Authenticating.
- Depending on key algorithm, you need a different type of key for each—RSA supports all these modes, but DSA and ed25519 do not.

PGP Sucks

- The standard is complicated—in crypto, this is a bad thing!
  - 65 pages for the original standard, RFC 2440
  - Current draft revision (RFC 4880bis-10) is 102
  - The 5 current standards comprise 130 pages, and only partially cover what is needed to implement PGP
- The tools are more complicated than the standard! GnuPG's primary man page is 35 (80x100 char) pages alone, just to document the command line flags. And it has texinfo documentation, too!

Looks okay, right?


PSIRT PGP Key (0x33E9E596
Secure | https://blogs.adobe.com/psirt/?page_id=146
blogs.adobe.com
Search Blogs

Adobe Product Security Incident Response Team (PSIRT) Blog

Working to help protect customers from vulnerabilities in Adobe software. Contact us at PSIRT(at)adobe(dot)com.

PSIRT PGP Key (0x33E9E596)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: Mailvelope v1.8.0
Comment: https://www.mailvelope.com

xsFNBFm/2KMBEADbwToJM3BCVE1OeC22HgVEqNEDppXzuD2dgfKuy0M4tx2L
De7GkPjo6AOsw4yi8bakLiidpw5B0J/AR1VtIjIDEmS0F9MRZIcV0UKyA5qV
c9BafZnAicY7nezkIJUmyLcIVMC60pqSHzo0Ewy2PZjxzcI4vDGhHmcgfV5X
R+duYld3LtVI+A/5jv326LB16bCNts/tOhW2T0LraMPoCtdH84Z4tPcyp335
s8/dZ2C+EoMD4iX1kIymZ1kqEfZNvcs1sRUXy27sL01VHcYmi6UNWCeeHOu2
2yJxMiBCniozBKZUwcR6ysg97nnq633dN9mf7V30PS3zAjhE0Hvmzg3B/Nfo
qzy2dAEU/JDUBhiAo+xr9VF3ZPOoC8JySORgyUm/2t3TTBaH+DnfsUBiqo5U
2T0n8x2R1FWxyZYNCTku5JOvPqRBft13DSyJD7LDDps62nqhpaVb34eprwuk
qIk0TMRu9mB4EQc+cNFR3ZpN1AKj+HOb/TUJwCJpVju2/3g0wgdqHh+OQlvC
Nm8vIGnQZWQ30WqnH/UFoh3RPJ+WqnDq88NmqBq8I4aNV4u8MqoObd/zrtVX
kAwYHbIZLo925NjFyPuuxhWiCotKenl8dZefB8aB8lRjYuIMnCJ0GQus+JG8
TJyEesNdK/q8HD5h1kCRSzMHDl+Ra3z/1+FFIwARAQABzR1BZG9iZSBQU0lS
VCA8cHNpcnRAYWRvYmUuY29tPsLBewQQAQgALwUCWb/YrwUJAeEzgAYLCQgH
AwIJEIbAD8Kvh3YWBBUIAgoDFgIBAhkBAhsDAh4BAADk2A//f+6PFzg4VmLI
PzsTZPoqPR/lXlZ7RIYbQosHvsFwyW0WWX1uI1sEeD5Qo7HQt6NNMAOW51Js
wFvFOWIa9U6SHRoU1kGTSESReOq5HnXe4DcBubsKmoMS68PuiZ88wYOIM4Up
9V9PUuaue0U4oSrYHnH5qBOqurtv8wO5Cq4uTwnfnjN7n4OH0++29l0PJ68B
6+kMuQyG4swmxsZhljlqGMHcs0c/BuI3W+n5w+xLM7N5jjCTjNXR+tGmstdm
RPEoLWOso+ZFwfNW0CLKjYUahp3p6H9x8R13wrp2re0GhqKRgt3D4UcAqsPs
```

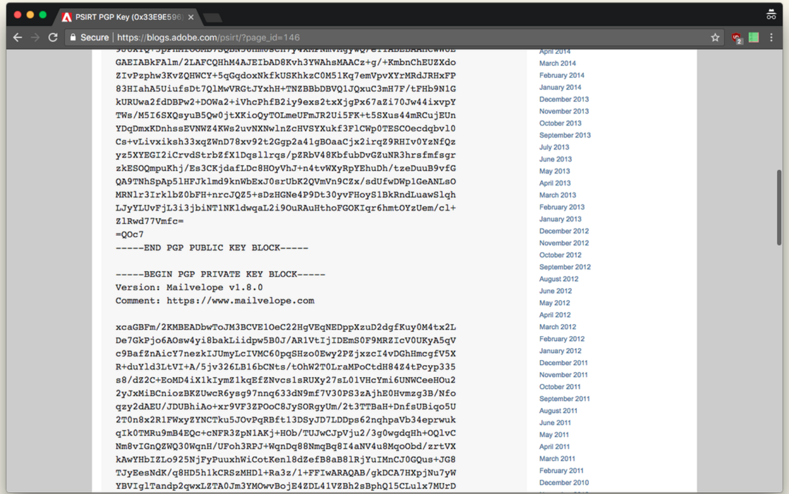
CATEGORIES

- Alert
- Security Bulletins and Advisories
- Uncategorized

ARCHIVES

- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015

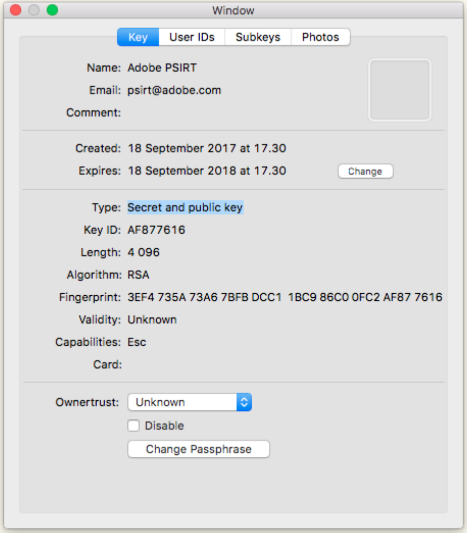
Hmm. . .


PSIRT PGP Key (0x33E9E596)
Secure | https://blogs.adobe.com/psirt/?page_id=146

```
GAEIABkFAlm/2LAFCQHhM4AJEIbAD8Kvh3YWAhsMAACz+g/+KmbnChEUZXdo
ZIvPzphw3KvZQHWCY+5qGqdoxNkfkUSKhkzC0M51Kq7emVpvXYrMRdJRHxFP
83HIahA5UiufsDt7QlMwVRGtJYxhH+TNZBBbDBVQ1JQxuC3mH7F/tFHb9N1G
kURUwa2fdDBPw2+DOWa2+iVhcPhfB2iy9exs2txXjgPx67aZi70Jw44ixvpY
TWs/M5I6SXQsyuB5Qw0jtXKioQyTOLmeUFmJR2Ui5FK+t5SXus44mRCujEUn
YDqDmxKDnhssEVNWZ4KWs2uvNXNwlnZcHVSYXukf3FlCWp0TESCOecdqbvl0
Cs+vLivxiksh33xqZWnD78xv92t2Ggp2a41gBOaaCjx2irqZ9RHIv0YzNfQz
yz5XYEGI2iCrvdStrbZfX1Dqsllrqs/pZRbV48KbfubDvGZuNR3hrsfmfsgr
zkESOQmpuKhj/Es3CKjdafLDc8HOyVhJ+n4tvWXyRpYEhuDh/tzeDuuB9vfG
QA9TNhSpAp5lHFJklmd9knWbExJ0srUbK2QVmVn9CZx/sdUfwDWp1GeANLsO
MRNlr3IrklbZ0bFH+nrcJQZ5+sDzHGNe4P9Dt30yvFHoyS1BkRndLuawSlqh
LJyYLUvFjL3i3jbiNT1NKldwqaL2i9OuRAuHthoFGOKIqr6hmtOYzUem/cl+
ZlRwd77Vmfc=
=QOc7
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: Mailvelope v1.8.0
Comment: https://www.mailvelope.com

xcaGBFm/2KMBEADbwToJM3BCVE1OeC22HgVEqNEDppXzuD2dgfKuy0M4tx2L
De7GkPjo6AOsw4yi8bakLiidpw5B0J/AR1VtIjIDEmS0F9MRZIcV0UKyA5qV
c9BafZnAicY7nezkIJUmyLcIVMC60pqSHzo0Ewy2PZjxzcI4vDGhHmcgfV5X
R+duYld3LtVI+A/5jv326LB16bCNts/tOhW2T0LraMPoCtdH84Z4tPcyp335
s8/dZ2C+EoMD4iX1kIymZ1kqEfZNvcs1sRUXy27sL01VHcYmi6UNWCeeHOu2
2yJxMiBCniozBKZUwcR6ysg97nnq633dN9mf7V30PS3zAjhE0Hvmzg3B/Nfo
qzy2dAEU/JDUBhiAo+xr9VF3ZPOoC8JySORgyUm/2t3TTBaH+DnfsUBiqo5U
2T0n8x2R1FWxyZYNCTku5JOvPqRBft13DSyJD7LDDps62nqhpaVb34eprwuk
qIk0TMRu9mB4EQc+cNFR3ZpN1AKj+HOb/TUJwCJpVju2/3g0wgdqHh+OQlvC
Nm8vIGnQZWQ30WqnH/UFoh3RPJ+WqnDq88NmqBq8I4aNV4u8MqoObd/zrtVX
kAwYHbIZLo925NjFyPuuxhWiCotKenl8dZefB8aB8lRjYuIMnCJ0GQus+JG8
TJyEesNdK/q8HD5h1kCRSzMHDl+Ra3z/1+FFIwARAQAB/gkDCA7HXpjNu7yW
YBVIglTandp2qwxLZTA0Jm3YMOwvBojE4ZDL41VZBh2sBphQ15CLulx7MUrD
```

April 2014
March 2014
February 2014
January 2014
December 2013
November 2013
October 2013
September 2013
July 2013
June 2013
May 2013
April 2013
March 2013
February 2013
January 2013
December 2012
November 2012
October 2012
September 2012
August 2012
June 2012
May 2012
April 2012
March 2012
February 2012
January 2012
December 2011
November 2011
October 2011
September 2011
August 2011
June 2011
May 2011
April 2011
March 2011
February 2011
December 2010

Well, Shit



Name: Adobe PSIRT
Email: psirt@adobe.com
Comment:

Created: 18 September 2017 at 17.30
Expires: 18 September 2018 at 17.30

Type: Secret and public key
Key ID: AF877616
Length: 4 096
Algorithm: RSA
Fingerprint: 3EF4 735A 73A6 7BFB DCC1 1BC9 86C0 0FC2 AF87 7616
Validity: Unknown
Capabilities: Esc
Card:

Ownertrust: Unknown
Disable

Keyservers have problems

- Can’t delete keys from the keyserver
- Keyservers leak lots of data to make PGP easier
- Most of the keyservers are broken these days, b/c of persistent attacks (fill up the disk).
- in 2015, there were over 100 keyservers active in the SKS pool; there are now 22.
- It is hard to make sane decisions when automatically downloading keys; “short” key IDs are the root of this problem.

SKS software is a mess

- The standard keyserver software is called SKS, for "Synchronizing Key Server".
- Yaron Minsky devised algorithm that could do reconciliations very quickly. SKS is *proof of concept* of his idea for his PhD Thesis.
- Written in OCaml, and an idiosyncratic (i.e. PhD’s) dialect.

SKS software is a mess

- Because of this, SKS is unmaintained
- Design goals are the reason SKS is vulnerable!
- Public editing with no central authority makes tainting data easy
- Lack of central authority also makes design changes impossible.

GnuPG is complicated

- People fuck this up all the time
- Hell; in GPG 2.1, the devs couldn't even write dirmngr—one of many components of gpg—to do key lookups over IPv6
- gpg 2.x have at least 5 different components that have to work: dirmngr, gpg-agent, a pinentry program, a management tool for gpg-agent, and then the program itself.
- integration of GPG with smartcards, using it as your ssh keyring, and using it for x.509 certificates and s/mime all add complexity

“Secure keys” are hard in GnuPG

Personal story:

- I wanted seperate PGP keys on my laptop from my desktop; multiple key pairs with one master
- GPG perfectly allows me to strip subkeys out of a keyring, s.t. I only have one signing and one encryption key.
- *However*, GPG only allows me to encrypt to the most recent key!
- (There are hacks around this, but they aren’t general)

Key trust is complicated

- Because of the web of trust, the onus is on you, the user, to verify keys with other people.
- You have to be painfully aware of cryptography to understand this—otherwise, verifying identities of others, and then *signing keys correctly*, is nearly impossible
- And then other people have different opinions on "sufficient verification"—some verify emails, while some only verify identity.
- It only approximates human trust models loosely—human trust is ephemeral, where PGP trust cannot be.

Long-lived identity keys suck

- Ever lose your phone or lose your house keys?

Long-lived identity keys suck

- Ever lose your phone or lose your house keys?
  It’s kind of painful to have to deal with, right?

Long-lived identity keys suck

- Ever lose your phone or lose your house keys?
  It's kind of painful to have to deal with, right?
- Losing PGP keys sucks so much worse. You potentially lock yourself out of everything, and there's no way to get it back. If you can't revoke your keys, it's even worse—there's no way to validate that you can't use your key.

Long-lived identity keys suck

- Ever lose your phone or lose your house keys?
  It's kind of painful to have to deal with, right?
- Losing PGP keys sucks so much worse. You potentially lock yourself out of everything, and there's no way to get it back. If you can't revoke your keys, it's even worse—there's no way to validate that you can't use your key.
- Trust can't be migrated to a new key—if your master key uses weak crypto that can be compromised, there's nothing you can do about it.
- Change your name? Want to use a new algorithm? Time for a new key and a complete loss of your trust!

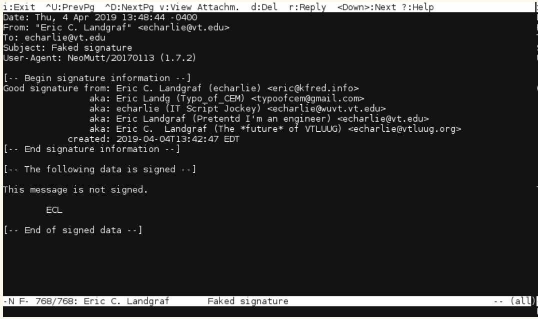
Implementation Flaws

- Public keys are *huge* when they have a lot of signatures.
- The tools don’t convey the gravity of actions, and are often quite unintuitive.
- There are lots of opinions on the *Right Way*™ to use PGP, rather than some standard being enforced by the tooling.
- Short key IDs are default in most tools; collisions can be made—indeed, the entire strongset was once duplicated.
- Trust is largely external to tooling—there aren’t many good ways to verify people without meeting them in-person

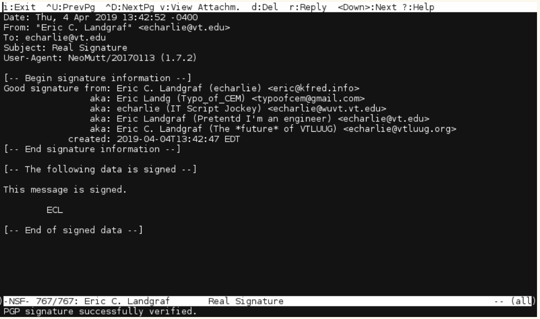
Email Signing and encryption issues

- "EFail", a CVE wherein broken MIME parsers are exploited to render html wrapping an encrypted blob (potentially sending the decrypted text to a malicious URL)
- Signature info can be forged; most MUAs render plain text for signature info.
- Message mangling: gmail (and others) mangle the body of messages sent to mail lists, rather than appending a text MIME attachment, causing signature validation to fail.
- Outlook and Plain text: a bug in outlook caused messages to not be sent encrypted if they weren't multipart.

Example of faked signatures in mutt:



```
i:Exit  ^U:PrevPg  ^D:NextPg v:View Attachm.  d:Del  r:Reply  <Down>:Next ?:Help
Date: Thu, 4 Apr 2019 13:48:44 -0400
From: "Eric C. Landgraf" <echarlie@vt.edu>
To: echarlie@vt.edu
Subject: Faked signature
User-Agent: NeoMutt/20170113 (1.7.2)

[-- Begin signature information --]
Good signature from: Eric C. Landgraf (echarlie) <eric@kfred.info>
                aka: Eric Landg (Typo_of_CEM) <typoofcem@gmail.com>
                aka: echarlie (IT Script Jockey) <echarlie@wuvt.vt.edu>
                aka: Eric Landgraf (Pretentd I'm an engineer) <echarlie@vt.edu>
                aka: Eric C.  Landgraf (The *future* of VTLUUG) <echarlie@vtluug.org>
            created: 2019-04-04T13:42:47 EDT
[-- End signature information --]

[-- The following data is signed --]

This message is not signed.

        ECL

[-- End of signed data --]




-N F- 768/768: Eric C. Landgraf       Faked signature                        -- (all)
```



```
i:Exit  ^U:PrevPg  ^D:NextPg v:View Attachm.  d:Del  r:Reply  <Down>:Next ?:Help
Date: Thu, 4 Apr 2019 13:42:52 -0400
From: "Eric C. Landgraf" <echarlie@vt.edu>
To: echarlie@vt.edu
Subject: Real Signature
User-Agent: NeoMutt/20170113 (1.7.2)

[-- Begin signature information --]
Good signature from: Eric C. Landgraf (echarlie) <eric@kfred.info>
                aka: Eric Landg (Typo_of_CEM) <typoofcem@gmail.com>
                aka: echarlie (IT Script Jockey) <echarlie@wuvt.vt.edu>
                aka: Eric Landgraf (Pretentd I'm an engineer) <echarlie@vt.edu>
                aka: Eric C. Landgraf (The *future* of VTLUUG) <echarlie@vtluug.org>
            created: 2019-04-04T13:42:47 EDT
[-- End signature information --]

[-- The following data is signed --]

This message is signed.

        ECL

[-- End of signed data --]




-NSF- 767/767: Eric C. Landgraf       Real Signature                         -- (all)
PGP signature successfully verified.
```

Certificate Spamming attack

- OpenPGP *doesn't limit* how many signatures can be attached to a certificate.
- SKS handles certificates with up to about 150,000 signatures.

Certificate Spamming attack

- OpenPGP *doesn't limit* how many signatures can be attached to a certificate.
- SKS handles certificates with up to about 150,000 signatures.
- GnuPG doesn't. *Any time* GnuPG has to deal with such a spammed certificate, GnuPG grinds to a halt.

Certificate Spamming attack

Consequences:

- If you fetch a poisoned certificate from the keyserver network, you will break your GnuPG installation.
- Poisoned certificates cannot be deleted from the keyserver network.
- The number of deliberately poisoned certificates, currently at only a few, will only rise over time.
- We do not know whether the attackers are intent on poisoning other certificates.
- We do not even know the scope of the damage.

Certificate Spamming attack

More info:

- `https://dkg.fifthhorseman.net/blog/openpgp-certificate-flooding.html`
- `https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f`

Potential Improvements and mitigations

- Double opt-in `keys.openpgp.org`
- Stop using Keyservers
- more diversity in keyserver implementations (e.g. Hockeypuck)

Double-opt-in keys.openpgp.org

- Public single-node keyserver running “Hagrid” software
- Only distributes identity information for keys which have verified email addresses
- Non-identity (i.e. fingerprint and key, but not UIDs) info distributed for all keys uploaded
- Do not distribute third-party signatures. Must use caff or pius to distribute signatures
- does not distribute revokations!

PGP needs to die

- To be clear, the crypto is not the weak part of PGP; it's the tooling and the model itself.
- better alternatives exist

PGP needs to die

- To be clear, the crypto is not the weak part of PGP; it's the tooling and the model itself.
- better alternatives exist
  - signify (openbsd) — signing only
  - reop (tedunangst) — basic encryption only (NaCl-based)
  - OTR — ephemeral message keys and simple trust
  - signal — easy-to-use, ephemeral message keys, easy-to-establish out-of-band trust
  - plain old s/mime with ca-certs — better supported for email
  - age — like reop, but supports x25519 keys for ed25519 operations (e.g. ssh keys)

Pragmatic PGP

Still want to use PGP?

- you’re insane, but fine

Trust models

- Most useful feature of PGP is its sense of “trust”, if you’re willing to understand it.
  - Note: trust is often used interchangeably with “authenticity”.
- Download a key off a keyserver and look at it; PGP will say whether it is trusted or not; this is configurable.
  - TOFU
  - pgp (web of trust)
  - direct

Key Validity

- How can you tell if a key is valid?

Key Validity

- How can you tell if a key is valid?
- Trivial case: unexpired key that you’ve signed
- Or an expired or revoked key—clearly you *shouldn’t* trust it (and PGP won’t let you)

Key Validity

- How can you tell if a key is valid?
- Trivial case: unexpired key that you've signed
- Or an expired or revoked key—clearly you *shouldn't* trust it (and PGP won't let you)
- Less trivial: key is signed by people you "trust"

Establishing trust

- Establishing trust is hard, because trust is hard. PGP simplifies it. “Your identity has been validated by someone I ‘trust’ to verify identities”.
- In-person, a glance at hard-to-forge identification may do, but what about online?
- PGP takes a lot of implicit trust and forces you to make it explicit, for it to work.
- Some tools, like `https://keybase.io` help establish online, persona-based authenticity.

Threat models

- PGP does not defend you against all known attacks! The crypto is secure, but only if you know what it does!
- Your data and PGP key are encrypted at rest, which is great unless someone installs a keylogger.
- pew has copies of my (encrypted) PGP subkeys—should I revoke? (at time of updating, these have been revoked, for unrelated reasons)

Threat models

- Expiring and revoking keys help manage threats and trust
- If my key is compromised, I can revoke it; if I lose it, I can revoke or wait for it to expire
- On the other hand, I lose what trust I have—would I rather trust an expired key with strong signatures and a long life, or a newly generated one?

Threat models

- PGP IS NOT A PANACEA!!!
- If I haven’t made it clear, PGP is powerful, but only if you understand basic opsec (topic of a very different talk), and what the tool does.
- PGP is useless unless you and the party you’re communicating with:
  - Have PGP keys
  - Have a “Secure” or “Trusted” channel to establish communications and verify each others’ identity and key
  - Can use the tools available.
- Sophisticated enemies will *not* target the crypto; they’ll target the machines you use, the network, or the people around you.

When things fall apart

- PGP on mobile sucks.
- PGP with webmail sucks.
- PGP with large non-text files sucks.
- PGP provides no anonymity: it is designed for the opposite!
- PGP provides no forward secrecy: if your encryption private key is compromised, all of your encrypted data using that key is readable.

Actually using it

You'll need an implementation of the OpenPGP tools:

- GNU Privacy Guard (gnupg or GPG) is most common; *nix and windows distributions are available
  - `https://gpgtools.org/` is available for OS X
  - `https://www.gpg4win.org/` is a port of GnuPG to windows
- I have no experience with GPG4Win, or any non-GPG tools; many of them provide GUIs, though.
- If you're a masochist, I suppose you could use Symantec PGP. Tell me how that goes.

I strictly use GnuPG 2.x; 1.4 (shipped by default through Ubuntu 16.04 and Fedora 28) has a lot of flaws and is not actively developed. Use the gpg2 binary on these distros.

A note on PGP GUIs

There are a lot of PGP GUIs; both gpgtools and gpg4win ship them. GNOME and KDE also have their own. These do not make it much easier to manage your key, and make messing up much easier!

GNOME Keyring actively sucks when using GPG and *will* get in the way eventually!

Demo

Here's where I switch over to a terminal and generate a key, then do some things with it. I'll also talk about the "Perfect" PGP key.

Keeping your key secure

- The security of your key is paramount to PGP’s effectiveness!
- You should obviously use a strong password for your key, but there is a lot more you can do:
  - back up your key onto an *encrypted* drive stored somewhere safe. Check it regularly.
  - strip out your creation key from your keyring on devices
  - print out a copy of your revocation certificate; you can OCR it later.
  - put your key on a yubikey for use on “insecure” machines

Email tools

- mutt has built-in support
- thunderbird through enigmail
- mail.app on OS X through a plugin in gpgtools

password management

- pass `https://www.passwordstore.org/`
  - pass has numerous wrappers and plugins for use on non-unix platforms
  - use of asymmetric crypto is also a boon when using with teams
- Implementing your own is pretty easy (there are plenty of people who do)

SSH

It’s also possible to use your gpg-agent as an ssh agent, and a gpg key as an ssh key. This works really nicely with a yubikey and shared workstations!