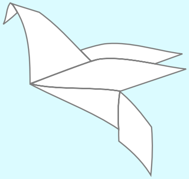
Sequoia PGP

Neal H. Walfield
Wiktor Kwapisiewicz
Lars Wirzenius
Justus Winter
Heiko Schaefer

November 23, 2021

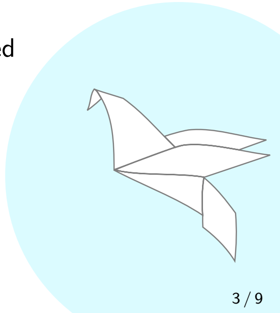


Genesis

- Sequoia project started in Fall 2017
- Founders: Neal, Justus & Kai Michaelis
- Sponsored by p≡p Foundation

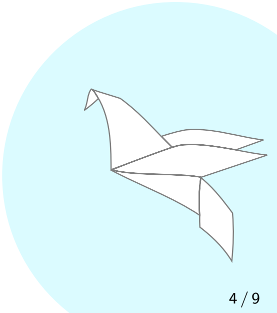
Prehistory

- 2015: Werner Koch hires Neal, Justus & Kai
- Formative period
  - Worked on GnuPG
  - Worked with developers integrating GnuPG
  - Worked with GnuPG users
  - Identified problems
  - Disagreements with Werner about how to proceed
  - Parted ways in Summer 2017



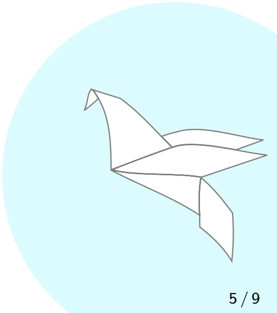
An OpenPGP Implementation for p≡p?

- Not a point solution
- Not an OpenPGP implementation
- A project to improve the OpenPGP ecosystem
  - Yes, a new OpenPGP library
  - But also:
    - Improve existing tools
    - Develop new tools
    - Rethink UX paradigms



Approach

- Bottom up
- Avoid technical debt
- Unopinionated, policy-free interfaces
- . . . but, secure by default
- Documentation, documentation, documentation



What We've Done

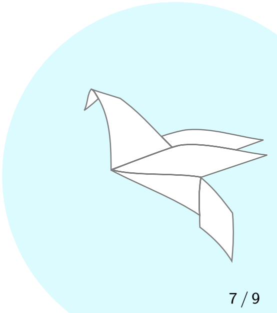
- Low-level OpenPGP library
  - Version 1.0 released December 2020
  - Lots of API documentation with examples[1]
  - Stable API
  - Few bugs. . . so far

[1] `https://docs.sequoia-pgp.org/sequoia_openpgp/`

What We're Doing (1/2)

Towards day-to-day use

- Public key store
- Private key store
  - Support for hardware security modules (HSMs)
- Command-line tool
  - `sq` - subcommand-style, REST-like interface, scriptable
  - `gpg chameleon` - gpg CLI that uses sequoia
- Strong, easy authentication with the web of trust
  - Beyond keysigning parties
  - Tools for using, creating, and managing CAs
    - OpenPGP CA

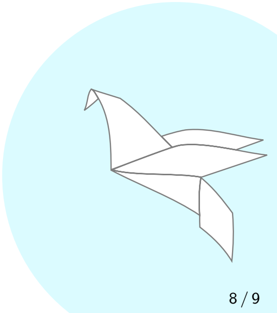


What We're Doing (2/2)

Standardization Work

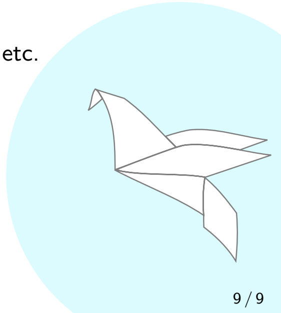
- IETF OpenPGP Design Team
- Public key store[2]
- Web of Trust[3]

[2] https://sequoia-pgp.gitlab.io/pgp-cert-d/
[3] https://sequoia-pgp.gitlab.io/sequoia-wot/



Where We're Going

- Continue up the stack
- Improve authentication, integrity, and confidentiality in existing tools
- Develop new tools to make authentication, integrity, and confidentiality easier
  - email
  - File Exchange
  - Authentication
    - Login to services
    - A cross-domain web of trust: ssh, matrix, age, etc.
  - Federated services
  - Server side infrastructure
  - Framework integration



Adding TPM Support to Sequoia PGP

Wiktor Kwapisiewicz <wiktor@sequoia-pgp.org>

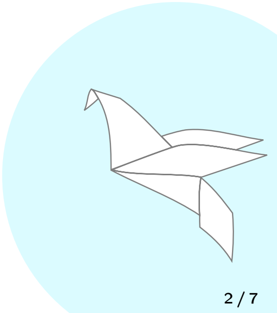
November 23, 2021

`https://wiktor.gitlab.io/tpm-openpgp/`

`https://gitlab.com/wiktor/tpm-openpgp`

Secure private key storage

- Problem: it's hard and expensive to protect cryptographic keys
- TPM to the rescue! Secure storage for your private keys that you already have!

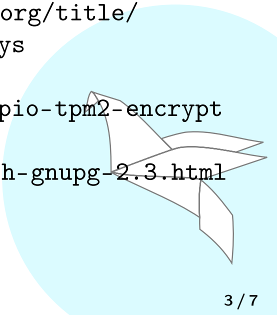


TPM in free/open-source software

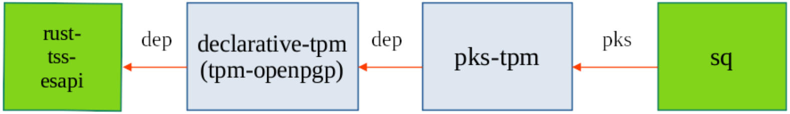
- TPMs as "treacherous computing" used for DRM

  (...) we conclude that the 'Trusted Platform Modules' available for PCs are not dangerous, and there is no reason not to include one in a computer or support it in system software. – Richard Stallman, https://www.gnu.org/philosophy/can-you-trust.html

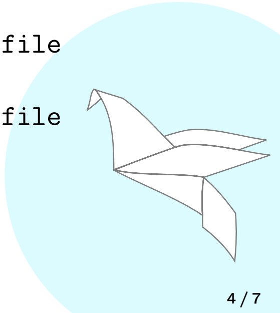
- TPM in open-source projects:
  - SSH authentication `https://wiki.archlinux.org/title/Trusted_Platform_Module#Securing_SSH_keys`
  - LUKS volume decryption `https://github.com/electrickite/mkinitcpio-tpm2-encrypt`
  - And even... GnuPG! `https://gnupg.org/blog/20210315-using-tpm-with-gnupg-2.3.html`



TPM in Sequoia PGP: Project structure



```
$ sq sign --signer-key wiktor.asc
  --private-key-store http://localhost:3000 file
$ sq decrypt --recipient-key wiktor.asc
  --private-key-store http://localhost:3000 file
```



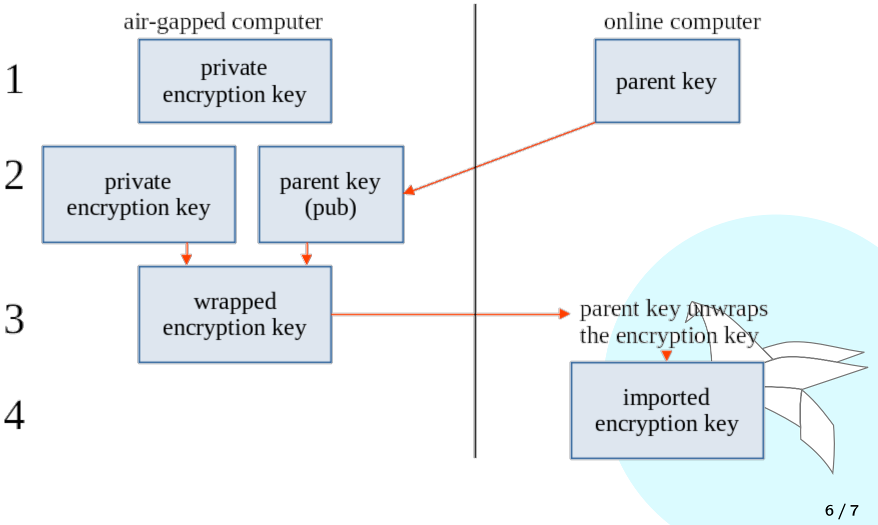
Declarative TPM

```
$ cat key.yml
spec:
  provider:
    tpm:
      tcti: "device:/dev/tpmrm0"
      handle: 0x81000027
  algo:
    RSA:
      bits: 2048
  capabilities:
    - sign

$ create-key -f key.yml
$ echo -n foo |
    openssl dgst -binary -sha256 |
    sign-digest -f key.yml > signature
```

TPM: Key migration

- Problem: Secure provisioning of decryption keys
- Solution: TPM Duplicate + TPM Import



Thank you NLnet!

- Thank you for supporting open-source!
- Statistics:
  - declarative-tpm (tpm-openpgp): 128 commits, 37 end-to-end tests, usage documentation
  - pks-tpm: 25 commits, 8 end-to-end tests, usage doc
  - sequoia-pgp+sq: 3 commits, API docs, man page,
  - rust-tss-esapi: 22 commits, 10 merged PRs,

`https://wiktor.gitlab.io/tpm-openpgp/`

`https://gitlab.com/wiktor/tpm-openpgp`

Making sq better

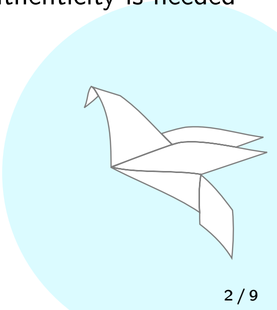
Lars Wirzenius <liw@sequoia-pgp.org>

November 23, 2021

`https://sequoia-pgp.org`

Our dream

- Sequoia is a great OpenPGP implementation
  - sq the obvious choice for command line use
- people like using sq
- Sequoia is considered when privacy, integrity or authenticity is needed

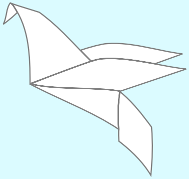


Project goals

- Add missing functionality to sq
- Add a programmatic API using JSON
- Document and verify acceptance criteria

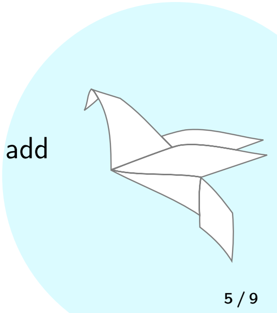
Missing functionality

- sq currently implements basic functionality
  - example: sq can't extend expiration
  - the Sequoia library supports everything (really)
- no good list of what's missing
- build list of missing features
  - compare sq to GnuPG
  - interview stakeholders
- add as many features as there's time for
  - document what will remain missing for now
  - implement only what's actually important for users
  - it is <u>not</u> important to have feature parity with gpg



JSON API

- <u>ideally</u> other programs will use the Sequoia library, not sq
- <u>realistically</u> people are going to use sq from scripts
- <u>therefore</u> make it easy to use sq from scripts
  - safe
  - secure
  - convenient
  - future-proofed
- JSON seems like a fairly obvious choice
  - JSON is supported by basically every language
  - easy to use in programs
  - no need to parse free form text
- after JSON support, other formats are easy-ish to add
  - YAML? TOML? S-exp?



JSON API example

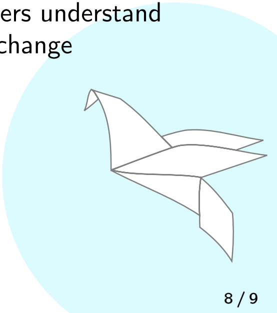
```
$ sq inspect --format=json foo.pgp
{
  "_sq_schema": [1,0,0],
  "filename": "foo.pgp",
  "file_type": "transferable-public-key",
  "key_type": "ed25519",
  "valid_until": 1637593318,
  "valid_until_iso8601": "2021-11-22T17:02:27+02:00",
  "user_ids": [
    "Lars Wirzenius",
    "<liw@sequoia-pgp.org>",
  ],
}
$ sq inspect --format=json foo.pgp | jq -r .valid_until
1637593318
$
```

JSON API, possible uses

- a password manager a la `pass` but uses Sequoia
  - port `pass` to support sq?
- cron job to find keys that will expire soon
- Debian keyring maintainers could script technical checks for keys
- script to find keys that have not been certified by a CA
- script to find keys that aren't strong enough anymore: too short, weak algorithm, . . .

Acceptance criteria for sq

- how do we know we're building the right tool?
- how do we know what we've built works after it's changed?
- communication and automation are the answer
    - talk to all stakeholders
    - agree on actual, verifiable requirements
    - agree on how to automate verification, when that's possible
    - document agreements in a way that all stakeholders understand
    - in CI, verify that requirements are met for every change
- we're using the Subplot tool for this
    - Lars is biased, having co-authored Subplot

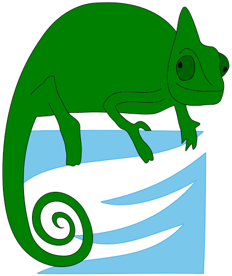


Stakeholders? User testing volunteers?

- Lars will be looking for volunteers
- Stakeholders
  - intend to use sq
  - want to use sq
- User testing
  - use sq to achieve specific goals under observation, over video chat, using screen sharing
  - is the sq command line interface OK to use?
  - what parts are hard or could be improved?
- Look for announcements on the Sequoia project blog
  - `https://sequoia-pgp.org/blog/`

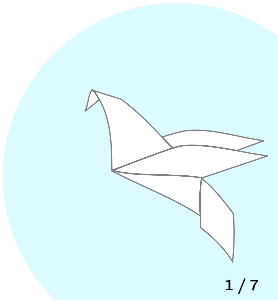
Sequoia's GnuPG Chameleon

Justus Winter <justus@sequoia-pgp.org>



2021-11-23

https://sequoia-pgp.org/



Why reimplement GnuPG

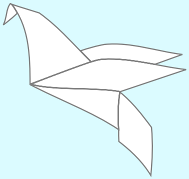
- many existing programs use GnuPG
  - direct invocation
  - GPGME
  - third-party libraries like GMime
  - $\rightarrow$ reimplement the ‘gpg‘ CLI
- infeasible to port them all
- no migration path for users and developers

Benefits for Users

- migration path to Sequoia
- implemented in Rust
- uses modern algorithm policies
- EFAIL defense
- scalable and well-documented Web of Trust implementation
- scalable certificate store certd

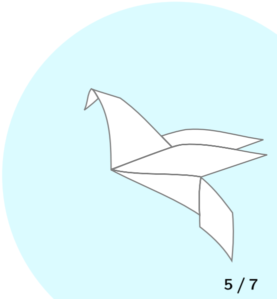
In Scope

- *gpg* and *gpgv*
- signing, verifying, encryption, decryption, key management
- the human-readable interface
- the machine-readable interface
  - status-fd
  - with-colons
- reading configuration files
- reading keyrings and keyboxes
- using *gpg-agent*
- reading trust-anchors from *trustdb.gpg*
- Keyserver and WKD



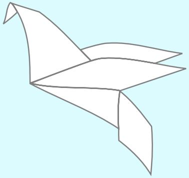
Out of Scope

- bug-to-bug compatibility
- translations of messages and program arguments
- using or reimplementing *dirmngr*
- reimplementing *gpg-agent*
- reimplementing *scdaemon*
- updating the *trustdb.gpg*
- esoteric features like server mode



Evaluation

- black box testing
  - using *gpg* and *gpgv* as oracle
  - comparing machine-readable output
    - modulo normalization
  - comparing human-readable output
    - edit distance $<\sim 20$
- large-scale integration testing
  - rebuilding Debian packages
  - problems:
    - too coarse, e.g. boolean verification $\rightarrow$ bisimulation
    - old artifacts, e.g. v3 signatures in GnuPG's test suite



How to migrate

- apt install sequoia-gpg-chameleon
  - dpkg-diverts /bin/gpg away
  - update-alternatives(1) provides /bin/gpg using chameleon
  - $\rightarrow$ every program magically uses Sequoia

- apt remove sequoia-gpg-chameleon
  - update-alternatives(1) provides /bin/gpg using GnuPG
  - dpkg-divert of /bin/gpg is undone
  - $\rightarrow$ every program uses GnuPG again

OpenPGP CA - a certification authority for trust management in groups

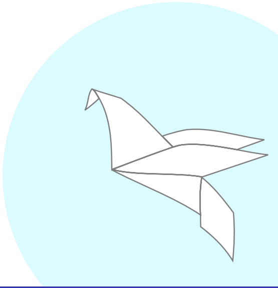
Heiko Schäfer <heiko@schaefer.name>

November 23, 2021

What is OpenPGP CA?

- A paradigm for how to use OpenPGP in groups/organizations
- Tooling to implement this paradigm

Goal: make PGP both easier and safer for users.



Main objective: strong authentication, but easy

Authentication, aka: being sure you're using 'the right key'.

Alice
(doesn't particularly enjoy checking fingerprints)

Bob
`0x12345678`

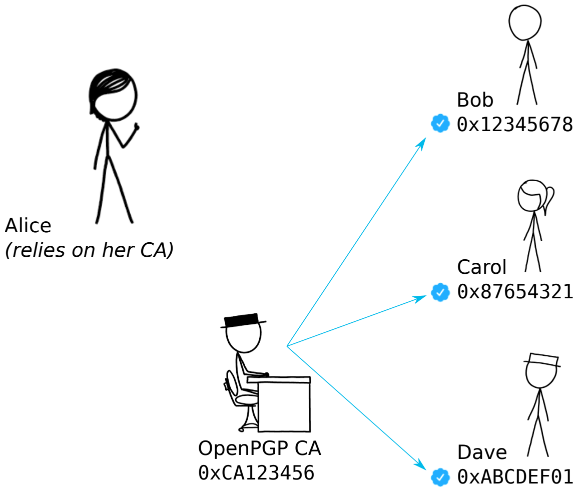
Carol
`0x87654321`

Dave
`0xABCDEF01`

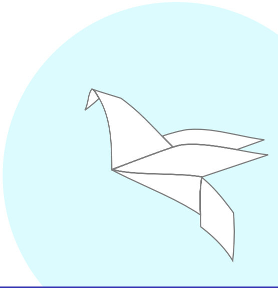
Stick figures from xkcd (CC BY-NC 2.5)

Authentication relying on a CA

Manual work performed by CA admin on behalf of their users.

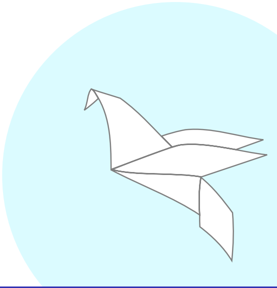


Stick figures from xkcd (CC BY-NC 2.5)



Common schemes for authentication

- Central, mandatory trust anchors (e.g. TLS)
- Variations of TOFU or YOLO (e.g. ssh, e2ee messengers)



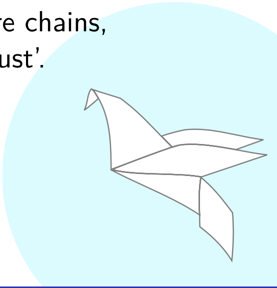
Striking a balance

Decentralization is great.

Strong authentication is great.

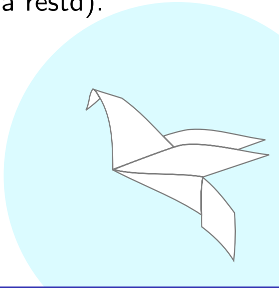
Asking each user to manually authenticate all other parties, not so much.

$\rightarrow$ OpenPGP CA uses (machine readable) signature chains, but keeps the decentral approach of the 'web of trust'.



There's more to OpenPGP CA, including . . .

- Publishing keys, especially via WKD.
- Trust for a CA can be scoped (by domain).
- Federation (*"bridging"*) between organizations.
- Integration into existing infrastructure (e.g. via restd).

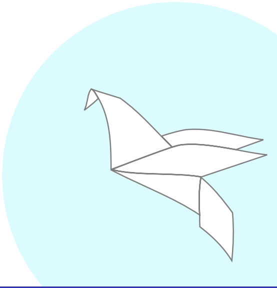


Advantages from user perspective

- Strong authentication that doesn't require ongoing effort.
- Can rely on CA(s) whose incentives are aligned with their own.
- No additional software needed (OpenPGP CA populates the "web of trust", relying on established features of the OpenPGP standard).

Next steps: hardened CA instances

- Hardware backed CA private keys (OpenPGP card)
- Split mode: online CA + protected CA



Thank you

In conclusion:

- Maybe your organization should run an OpenPGP CA instance?
- Let's talk! (e.g. on IRC, OFTC `#sequoia` or `#openpgp-ca`)

Thanks to NLnet and pep foundation for funding these projects