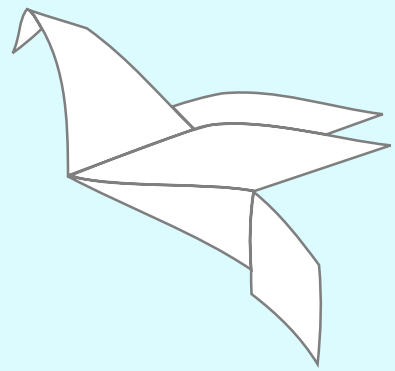
A Practical Introduction to using sq, Sequoia PGP's CLI

Neal H. Walfield <neal@sequoia-pgp.org>
8F17777118A33DDA9BA48E62AACB3243630052D9

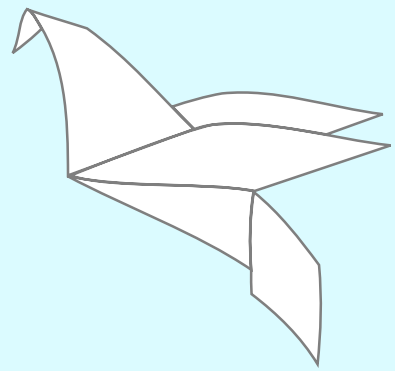
FOSDEM 2025, Security Track

February 1, 2025



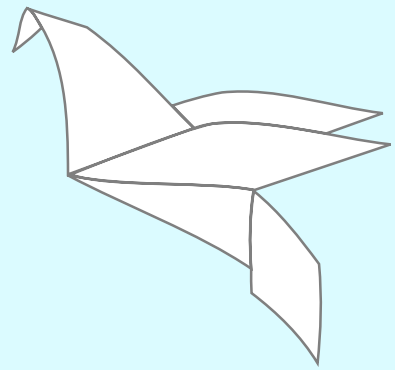
What is Sequoia PGP?

- An OpenPGP implementation
- OpenPGP Infrastructure
- Privacy and Security Tooling
- Privacy and Security Applications



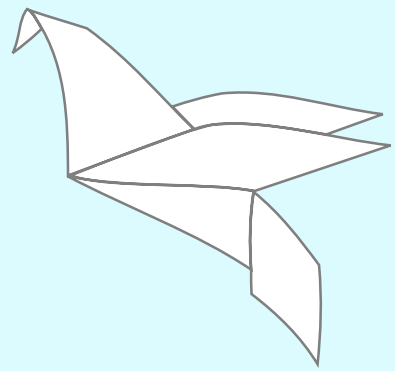
Goals

- Secure
- Robust
- Usable

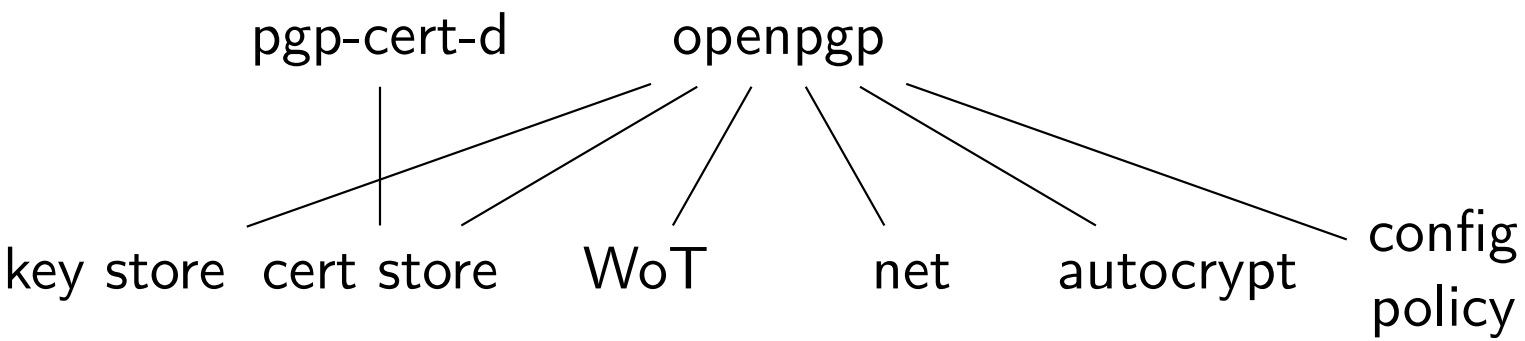


Design

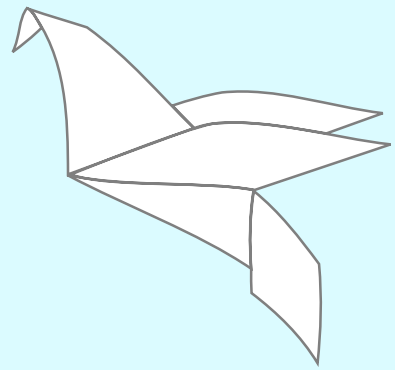
- Library first
- Unopinionated low-level interfaces
- Safe by default
- Opinionated higher-level interfaces
- Optional services
- Reevaluate existing paradigms



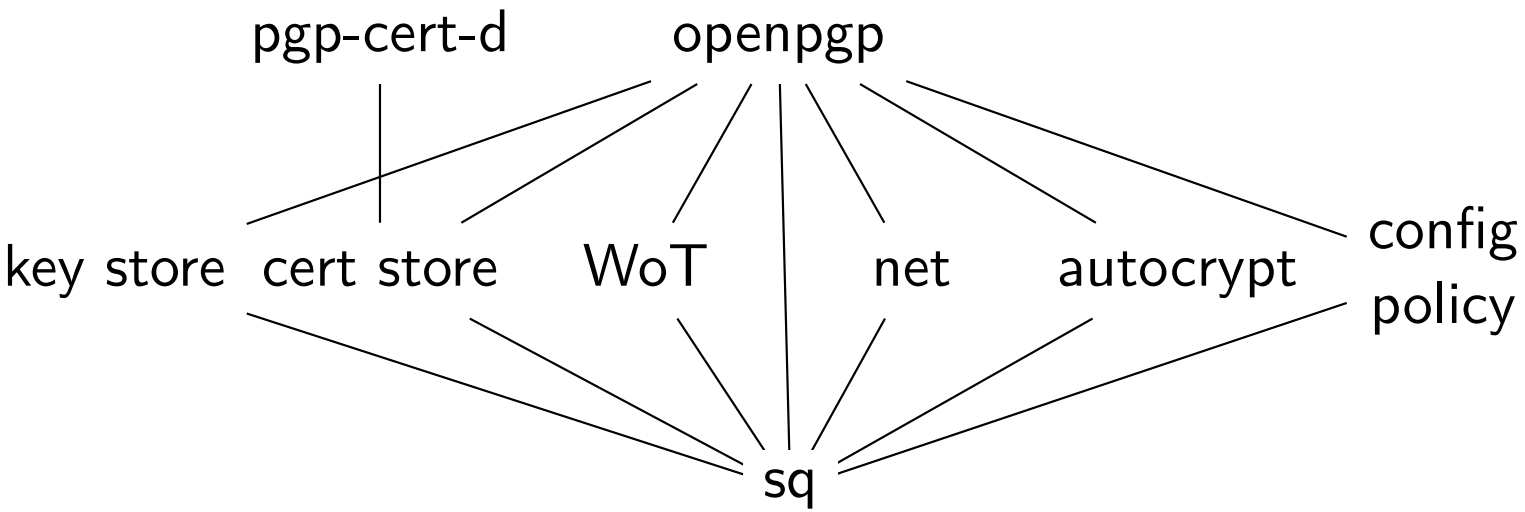
Components



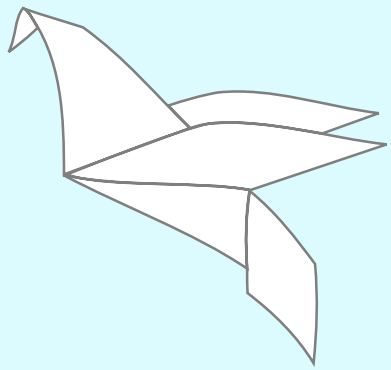
- openpgp: The low-level library
- key store: Private key operations
- pgp-cert-d: On-disk certificate store
- cert store: In-memory certificate store
- WoT: Web of trust engine
- net: Key server, WKD, and DANE support
- autocrypt: Autocrypt functionality
- config policy: Reads and parses a cryptographic policy



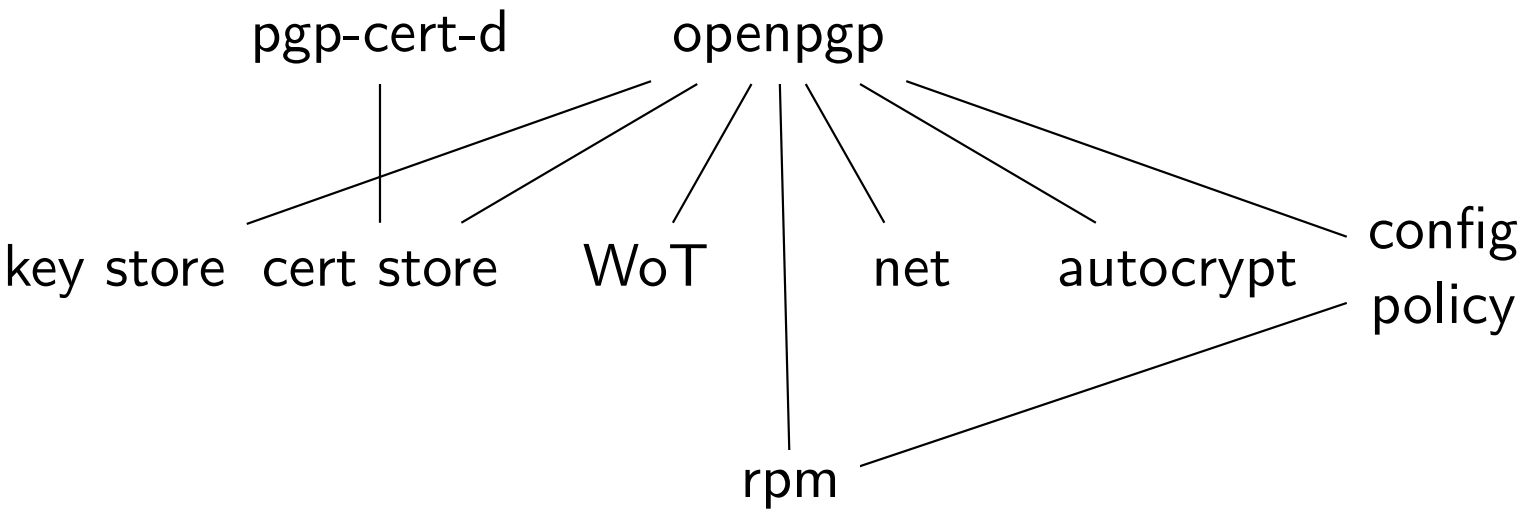
Components



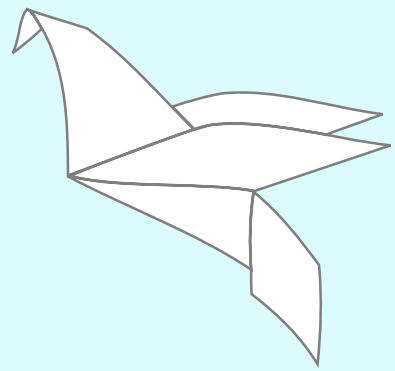
- sq
  - Uses all high-level libraries and services



Components

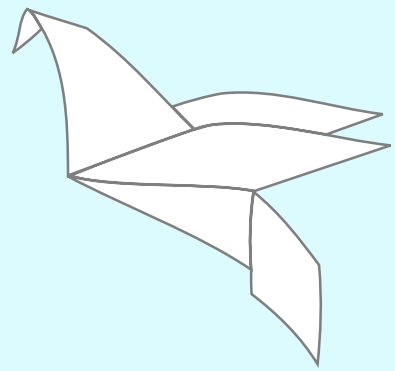


- RPM Package Manager (`rpm`)
  - Doesn't use secret key material
  - Has its own certificate store implementation
  - Has its own trust model
  - Uses the common policy configuration



What is sq

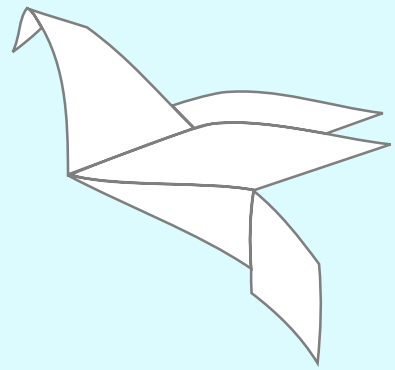
- Command-line tool
- Subcommand style interface
  - `sq key generate`
  - `sq network search`
- High-level interface
  - Focus on end user workflows
  - More control via libraries
- Stable CLI
  - Version 1.0 released December 16, 2024
  - Can be used for scripting



Demo Time!

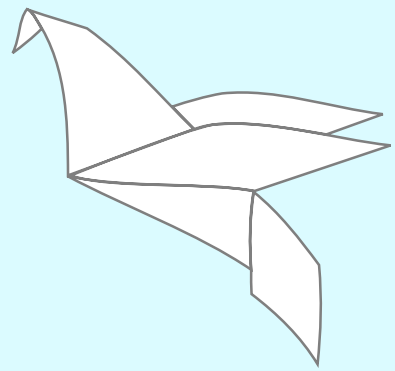
```
$ sq version
sq 1.2.0
using sequoia-openpgp 1.22.0
with cryptographic backend OpenSSL
```

- Due to space constraints some output has been trimmed!



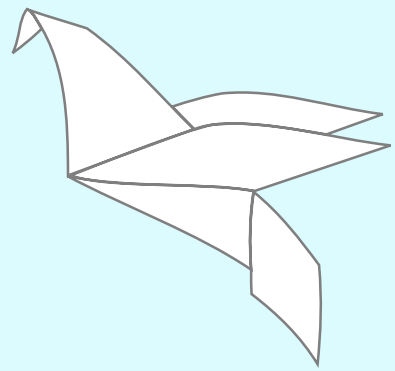
Let's Verify a Download

- Goal:
  - Download the latest version of Qubes
  - Verify the signature



sq download

- Downloads the signature and the file
- Verifies the signature
- Only releases the data if the signature is valid



Try #1

```
$ sq download \
  --signature-url \
    https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso.asc \
  --url https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso \
  --output qubes.iso
We can't verify the signature, because we don't have certificates for any of the
alleged signers:

 - 9C884DF3F81064A569A4A9FAE022E58F8E34D89F (missing certificate)

Hint: Try searching public directories:

  $ sq network search 9C884DF3F81064A569A4A9FAE022E58F8E34D89F

Aborting download.

  Error: Can't authenticate any of the alleged signers
```

Getting the Certificate

```
$ sq network search 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
Found 1 certificate related to the query:

 - 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
   - Qubes OS Release 4.2 Signing Key (UNAUTHENTICATED)
   - created 2022-10-04 14:10:01 UTC
   - found via: hkps://keys.openpgp.org, hkps://keyserver.ubuntu.com,
     hkps://sks.pod01.fleetstreetops.com
Imported 1 new certificate, updated 0 certificates, 0 certificates unchanged,
0 errors.

Hint: After checking that the certificate 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
      really belongs to the stated owner, you can mark the certificate as
      authenticated.  Each stated user ID can be marked individually using:

  $ sq pki link add --cert=9C884DF3F81064A569A4A9FAE022E58F8E34D89F \
    --userid="Qubes OS Release 4.2 Signing Key"
```

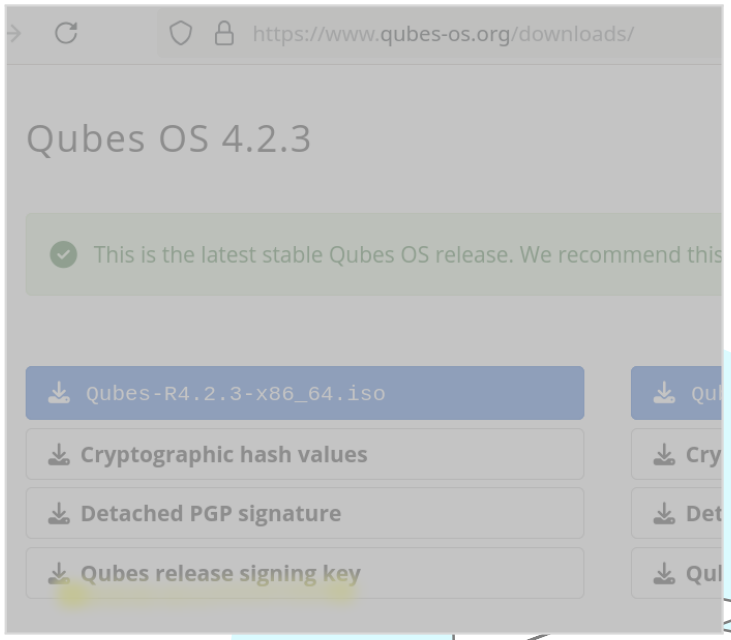
Try #2

```
$ sq download --signature-url \
    https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso.asc \
  --url https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso \
  --output qubes.iso
Can't authenticate the alleged signer:
 - ┌ 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
   └ Qubes OS Release 4.2 Signing Key (UNAUTHENTICATED)
 - No bindings matching the query could be authenticated.

We can't verify the signature, because we can't authenticate any of the alleged
signers:

 - 9C884DF3F81064A569A4A9FAE022E58F8E34D89F Qubes OS Release 4.2 Signing Key
   (UNAUTHENTICATED)

Hint: Verify that one of the certificates is authentic, and then link it:

  $ sq pki link add --cert=FINGERPRINT

Aborting download.
```

Authenticating the Certificate

- Authentication: Figuring out what certificate belongs to a given entity

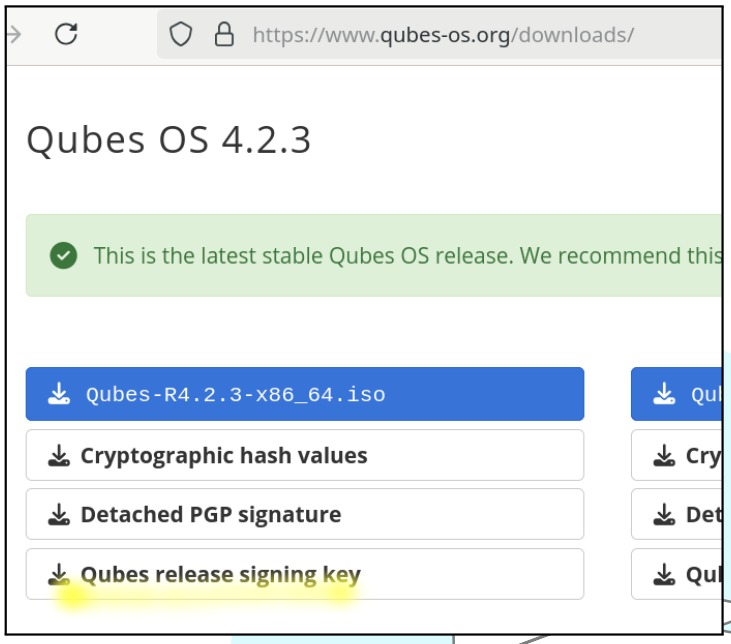
- Ask a Qubes community member
- Ask a friend
- Go to the Qubes stand
- Go to the website



Authenticating the Certificate

- Authentication: Figuring out what certificate belongs to a given entity

- Ask a Qubes community member
- Ask a friend
- Go to the Qubes stand
- Go to the website



Linking the Certificate

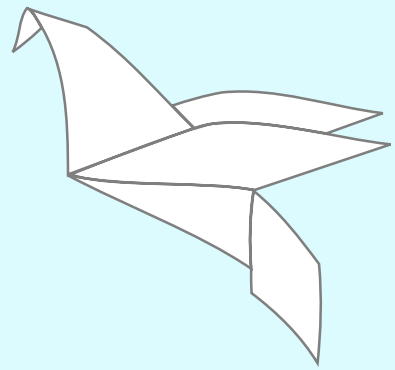
```
$ sq pki link add --cert 9C884DF3F81064A569A4A9FAE022E58F8E34D89F --all
 - ┌ 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
   └ Qubes OS Release 4.2 Signing Key
   - certification created

$ sq cert list 9C884DF3F81064A569A4A9FAE022E58F8E34D89F --show-paths
 - 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
   - created 2022-10-04 14:10:01 UTC

   - [    ✓     ] Qubes OS Release 4.2 Signing Key

     ○─┐ 40D9475A75203F505995BE2F9B45738A699AD8AA
       │ └ (Local Trust Root)
       │
       │  certified the following binding on 2025-01-31
       │
       └─┐ 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
         └ Qubes OS Release 4.2 Signing Key
```

Try #3

```
$ sq download --signature-url \
    https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso.asc \
  --url https://mirrors.edge.kernel.org/qubes/iso/Qubes-R4.2.3-x86_64.iso \
  --output qubes.iso
...
[                                                                    ] 6.91 GB/6.91 GB
Finished downloading data.  Authenticating data.

Authenticating 9C884DF3F81064A569A4A9FAE022E58F8E34D89F (Qubes OS Release 4.2
Signing Key (UNAUTHENTICATED)) using the web of trust:
  Fully authenticated (120 of 120) 9C884DF3F81064A569A4A9FAE022E58F8E34D89F,
  Qubes OS Release 4.2 Signing Key
    ○─┬ 40D9475A75203F505995BE2F9B45738A699AD8AA
      │ └ (Local Trust Root)
      │  certified the following binding on 2025-01-31
      └─┬ 9C884DF3F81064A569A4A9FAE022E58F8E34D89F
        └ Qubes OS Release 4.2 Signing Key
  Authenticated signature made by 9C884DF3F81064A569A4A9FAE022E58F8E34D89F (Qubes
  OS Release 4.2 Signing Key (authenticated))
1 authenticated signature.
```

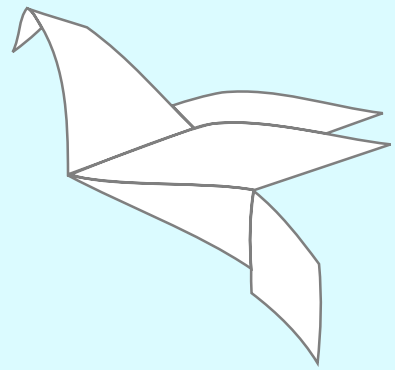
Summary

- `sq download` combines
  - Fetching the file
  - Fetching the detacted signature
  - Performing the verification
- Have to authenticate the signer
  - Slight burden the first time
- Data only released if it is authenticated



Let's Sign a Message

- Need a certificate
- Need to make it available to the recipient



Generating a Certificate

```
$ sq key generate --name 'Ada Lovelace' --email 'ada@example.org' --own-key
Please enter the password to protect key (press enter to not use a password):
                                                  Please repeat the password:
 - ┌ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   └ Ada Lovelace
   - certification created

 - ┌ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   └ <ada@example.org>
   - certification created

Transferable Secret Key.

      Fingerprint: 56A1EC32A41F1E7A512852378B070EF8CB7B193C
...
Hint: You can export your certificate as follows:
  $ sq cert export --cert=56A1EC32A41F1E7A512852378B070EF8CB7B193C

Hint: Once you are happy you can upload it to public directories using:
  $ sq network keyserver publish --cert=56A1EC32A41F1E7A512852378B070EF8CB7B193C
```

Signing a Message

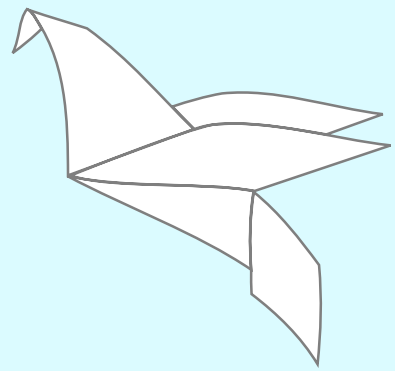
```
$ echo hello \
  | sq sign --signer 56A1EC32A41F1E7A512852378B070EF8CB7B193C --message \
  | tee message.pgp
-----BEGIN PGP MESSAGE-----
...
$ sq verify --message message.pgp
Authenticating 56A1EC32A41F1E7A512852378B070EF8CB7B193C (Ada Lovelace
(authenticated)) using the web of trust:
  Fully authenticated (120 of 120) 56A1EC32A41F1E7A512852378B070EF8CB7B193C,
  <ada@example.org>
    ○─┬ 2AE84205DF0CF8C02B5C654754FBD00C74E76D66
      │ └ (Local Trust Root)
      │
      │  certified the following binding on 2025-01-31 as a meta-introducer
      │  (depth: unconstrained)
      │
      └─┬ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
        └ <ada@example.org>

1 authenticated signature.
```

Making a Detaching Signature

```
$ sq sign --signer 56A1EC32A41F1E7A512852378B070EF8CB7B193C \
  --signature-file release-1.2.0.tar.gz.sig \
  release-1.2.0.tar.gz
$ sq verify --signature-file release-1.2.0.tar.gz.sig release-1.2.0.tar.gz
Authenticating 56A1EC32A41F1E7A512852378B070EF8CB7B193C (Ada Lovelace
(authenticated)) using the web of trust:
  Fully authenticated (120 of 120) 56A1EC32A41F1E7A512852378B070EF8CB7B193C,
  <ada@example.org>
    ○─┬ 2AE84205DF0CF8C02B5C654754FBD00C74E76D66
      │ └ (Local Trust Root)
      │
      │   certified the following binding on 2025-01-31 as a meta-introducer (depth:
      │   unconstrained)
      │
      └─┬ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
        └ Ada Lovelace

  Authenticated signature made by 56A1EC32A41F1E7A512852378B070EF8CB7B193C
  (<ada@example.org> (authenticated))
1 authenticated signature.
```

Let's Sign a Certificate

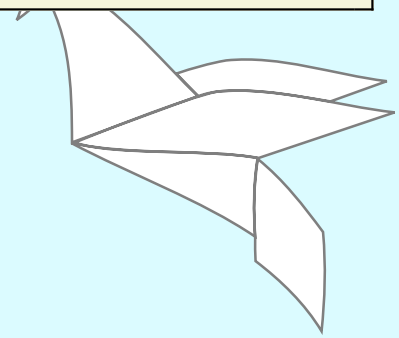
- Alice vouches that Bob has a certain certificate
  - Vouch: public assertion
  - Link: local assertion



Signing Bob's Certificate

```
$ sq network search 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
...
$ sq pki vouch add \
  --certifier 56A1EC32A41F1E7A512852378B070EF8CB7B193C \
  --cert 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717 --all
 - ┌ 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
   └ <bob@example.org>
   - certification created

 - ┌ 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
   └ Bob
   - certification created
```

- Can also use `--certifier-self` once configured

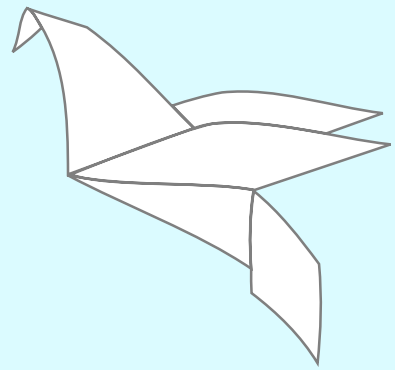


Examining Bob's Certificate

```
$ sq cert list 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717 --show-paths
 - 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
   - created 2025-01-31 22:00:49 UTC
   - will expire 2028-02-01T15:27:10Z

   - [    ✓     ] <bob@example.org>

     ○─┬ 2AE84205DF0CF8C02B5C654754FBD00C74E76D66
       └ (Local Trust Root)

       certified the following certificate on 2025-01-31 as a meta-introducer
       (depth: unconstrained)

       ├┬ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
       │└ (<ada@example.org>)

       certified the following binding on 2025-01-31 (expiry: 2035-02-01)

       └┬ 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
        └ <bob@example.org>
```

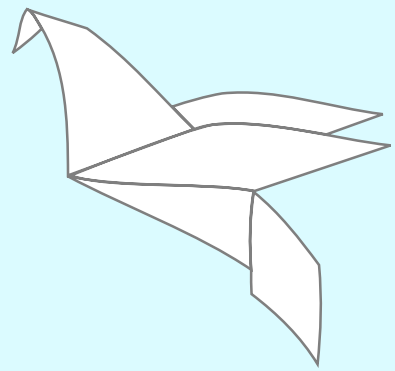
Let's Extend a Certificate's Expiration

```
$ sq key expire \
  --cert 56A1EC32A41F1E7A512852378B070EF8CB7B193C \
  --expiration 3y

Hint: Imported updated cert into the cert store.  To make the update effective,
      it has to be published so that others can find it, for example using:

  $ sq network keyserver publish \
    --cert=56A1EC32A41F1E7A512852378B070EF8CB7B193C
```



Let's Rotate a Certificate

- Generate a new certificate
  - Same user IDs
  - Similar structure
- Link the new certificate
- Cross sign the old and new certificate
- Replay certifications made by old certificate
- Retire old certificate

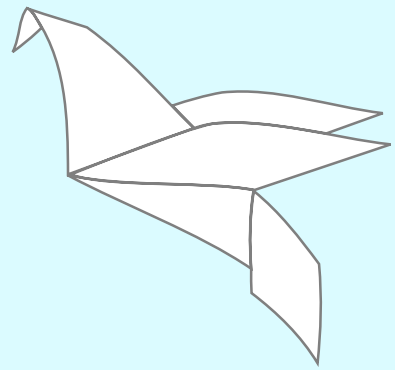


Rotating a Certificate

```
$ sq key rotate --cert 56A1EC32A41F1E7A512852378B070EF8CB7B193C
Cross signing the old and new certificates.
Replaying the old certificate's links:
  Copying link for <ada@example.org>:
   - created at 2025-01-31 21:16:40
   - linked as a fully trusted CA
  ...
Replaying certifications made by the old certificate:
  Considering the source certificate's certification of the binding:
   - ┌ 0AA62EDB7A5DF253FC4518470BBB8243CD2C3717
     └ <bob@example.org>
    Source certificate's active certification:
     - created at 2025-01-31 22:09:36
     - expiration: 2035-02-01
    Replayed certification.
  ...
Retiring the old certificate as of 2025-08-01 22:58:32.
Transferable Secret Key.
      Fingerprint: 29A4E06EB551927DA3F5DDE7FD6EF8DE8FFA83EC
...
```

Let's Create a Certification Authority

- Create a CA for `example.org`
- Create vouches for members of the organisation
- Publish certificates and vouches in a WKD
- Keep them up to date

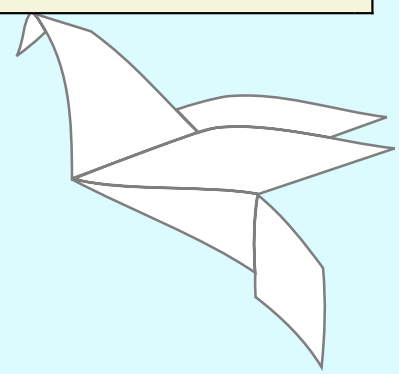


Creating a CA key

```
$ sq key generate --email ca@example.org --own-key \
  --cannot-sign --cannot-authenticate --cannot-encrypt
 - ┌ 916802D57EE60DC856A5538D73CEA1FD4C2F0D38
   └ <ca@example.org>
   - certification created

Transferable Secret Key.

      Fingerprint: 916802D57EE60DC856A5538D73CEA1FD4C2F0D38
  Public-key algo: EdDSA
  Public-key size: 256 bits
       Secret key: Unencrypted
    Creation time: 2025-01-31 23:11:54 UTC
  Expiration time: 2028-02-01 16:38:15 UTC (creation time + 2years 11months 30days)
        Key flags: certification

           UserID: <ca@example.org>
   Certifications: 1, use --certifications to list
...
```

Creating Vouches

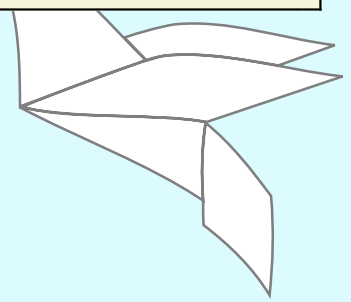
```
$ sq pki vouch add \
  --certifier 916802D57EE60DC856A5538D73CEA1FD4C2F0D38 \
  --cert 56A1EC32A41F1E7A512852378B070EF8CB7B193C --all
 - ┌ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   └ <ada@example.org>
    - certification created

 - ┌ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   └ Ada Lovelace
    - certification created
```



Publishing Vouches

```
$ sq network wkd publish \
  --trust-root 916802D57EE60DC856A5538D73CEA1FD4C2F0D38 \
  --domain=example.org --all \
  --create --rsync user@example.org:public_html
Updating:
 - 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   - Ada Lovelace (UNAUTHENTICATED)
   - inserted

 - 916802D57EE60DC856A5538D73CEA1FD4C2F0D38
   - <ca@example.org> (UNAUTHENTICATED)
   - inserted

2 updates applied.
```

- Supports rsync (--rsync) and local directories
- Can be used from a cron job



Let's Use a CA

```
$ sq encrypt --for-email ada@example.org --without-signature
  Error: Failed to resolve --for-email "ada@example.org"
because: 56A1EC32A41F1E7A512852378B070EF8CB7B193C, <ada@example.org>
         cannot be authenticated at the required level (0 of 120).

$ sq pki link authorize --cert 916802D57EE60DC856A5538D73CEA1FD4C2F0D38 --all \
  --domain example.org
 - ┌ 916802D57EE60DC856A5538D73CEA1FD4C2F0D38
   └ <ca@example.org>
   - certification created

$ sq encrypt --for-email ada@example.org --without-signature
Composing a message...

 - encrypted for <ada@example.org> (authenticated)
   - using 56A1EC32A41F1E7A512852378B070EF8CB7B193C
```

- CA is *scoped*: only used for the domain `example.org`

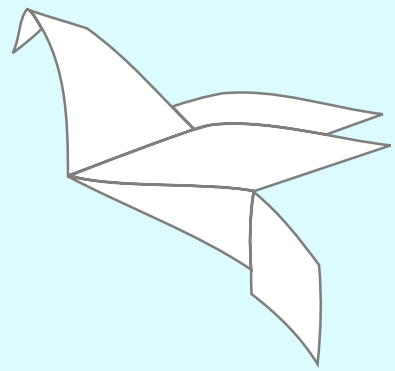
What Happened?

```
$ sq cert list ada@example.org --show-paths
 - 56A1EC32A41F1E7A512852378B070EF8CB7B193C
   - created 2025-01-31 21:16:40 UTC
   - will expire 2028-02-01T15:55:48Z

   - [    ✓     ] <ada@example.org>

     ○─┬ A0616BFB7A8D2FEF703570B2B601A5239ADF5782
       └ (Local Trust Root)

       certified the following certificate on 2025-02-01 as a meta-introducer
       (depth: unconstrained)

       ├─┬ 916802D57EE60DC856A5538D73CEA1FD4C2F0D38
       │ └ (<ca@example.org>)

       certified the following binding on 2025-01-31 (expiry: 2035-02-01)

       └─┬ 56A1EC32A41F1E7A512852378B070EF8CB7B193C
         └ <ada@example.org>
```

Summary

- sq supports high-level workflows
- sq guides the user with hints
- sq has some new concepts
  - Local trust root and links
  - Mandatory authentication
- Learn more:
  - User manual: `https://book.sequoia-pgp.org`
  - Man pages:
    `https://sequoia-pgp.gitlab.io/sequoia-sq/man/sq.1.html`

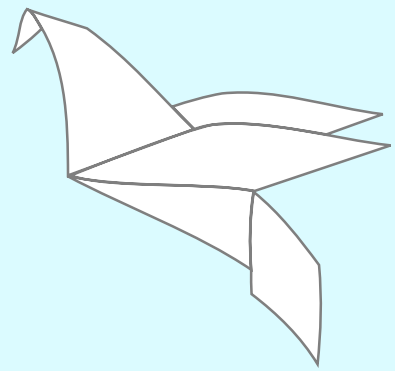
Questions?



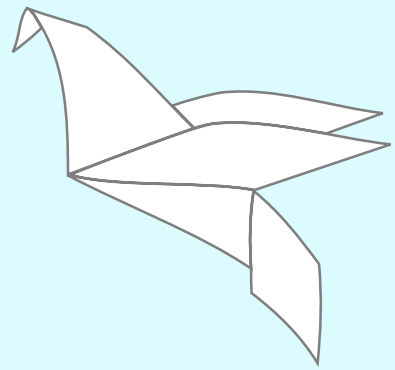
Summary

- sq supports high-level workflows
- sq guides the user with hints
- sq has some new concepts
  - Local trust root and links
  - Mandatory authentication
- Learn more:
  - User manual: `https://book.sequoia-pgp.org`
  - Man pages:
    `https://sequoia-pgp.gitlab.io/sequoia-sq/man/sq.1.html`

Questions?



Extra Slides



Migrating from GnuPG

- sq automatically imports gpg's certificate store
- sq uses gpg-agent
- sq does *not* import gpg's owner trust
- Chameleon is an (almost) drop-in replacement for gpg