A simple file encryption tool & format

Filippo Valsorda (@FiloSottile) — Ben Cartwright-Cox (@Benjojo12)
Designed at the Recurse Center during NGW 2019

This is a design for a simple file encryption CLI tool, Go library, and format.

It's meant to replace the use of gpg for encrypting files, backups, streams, etc.

It's called "age", which *might* be an acronym for Actually Good Encryption, and it's pronounced like the Japanese 上げ (with a hard *g*).

```
$ age-keygen > key.txt

$ cat key.txt
# created: 2006-01-02T15:04:05Z07:00
# public key: age1mrmfnwhtlprn4jquex0ukmwcm7y2nxlphuzgsgv8ew2k9mewy3rs8u7su5
AGE-SECRET-KEY-1EKYFFCK627939WTZMTT4ZRS2PM3U2K7PZ3MVGEL2M76W3PYJMSHQMTT6SS

$ echo "_o/" | age -r age1mrmfnwhtlprn4jquex0ukmwcm7y2nxlphuzgsgv8ew2k9mewy3rs8u7su5
-o hello.age

$ age -decrypt -i key.txt hello.age
_o/

$ tar cv ~/xxx | age -r github:Benjojo -r github:FiloSottile | nc 192.0.2.0 1234
```

You can find a **beta** reference implementation at github.com/FiloSottile/age and a beta Rust implementation at github.com/str4d/rage.

Goals

- An extremely simple CLI that composes well with UNIX pipes, and that works well as a backend for other programs
- Small copy-pasteable keys, with optional textual keyrings
- Support for public/private key pairs and passwords, with multiple recipients
- The option to encrypt to SSH keys, with built-in GitHub .keys support
- "Have one joint and keep it well oiled", no configuration or (much) algorithm agility
- A good seekable streaming encryption scheme based on modern chunked AEADs, reusable as a general encryption format

Later

- A password-store backend!
- YubiKey PIV support via PKCS#11 (sigh), maybe TouchBar
- Support for a Pond-style shared secret PAKE server
- Dictionary word encoded mnemonics for keys
- [DONE] An ASCII armored format
- ~~Support for AES-GCM in alternative to ChaCha20-Poly1305~~
- Maybe native support for key wrapping (to implement password-protected keys)
- age-mount(1), a tool to mount encrypted files or archives
  (also satisfying the agent use case by key wrapping)

Out of scope

- Archival (that is, reinventing zips)
- Any kind of signing (which is not a tooling problem, but a trust and key distribution problem, and to the extent that tools matter you should just use signify/minisign, and for keys we should probably use SSH ones)
- git commit signing, in particular (leave that to GitHub to solve) or releases and package signing (which is better solved at scale by transparency)
- Anything about emails (which are a fundamentally unsecurable medium)
- The web of trust, or key distribution really

Command line interface

Key generation

```
$ age-keygen >> ~/.config/age/keys.txt
Public key: age1p4fuklglxqsgg602hu4c4jl4aunu5tynyf4lkg96ezh3jefzpy6swshp5x
```

Encryption to a public key

```
$ echo "_o/" | age -o hello.age -r
age1p4fuklglxqsgg602hu4c4jl4aunu5tynyf4lkg96ezh3jefzpy6swshp5x
```

Encryption to multiple public keys (with default output to stdout)

```
$ echo "_o/" | age -r age1p4fuklglxqsgg602hu4c4jl4aunu5tynyf4lkg96ezh3jefzpy6swshp5x
-r age1t7r9prsqc3w3x4auqq7y8zplrfsddmf8z97hct68gmhea2l34f9q63h2kp > hello.age
```

Encryption with a password (interactive only, use public keys for batch!)

```
$ age -p -o hello.txt.age hello.txt
Type passphrase:
```

Encryption to a list of recipients in a file (not recursive, can't point to other files)

```
$ echo -r age1p4fuklglxqsgg602hu4c4jl4aunu5tynyf4lkg96ezh3jefzpy6swshp5x >>
recipients.txt
$ echo -r age1t7r9prsqc3w3x4auqq7y8zplrfsddmf8z97hct68gmhea2l34f9q63h2kp >>
recipients.txt
$ tar cv ~/xxx | age -r recipients.txt > xxx.tar.age
```

Encryption to an SSH public key

```
$ tar cv ~/xxx | age -r ~/.ssh/id_rsa.pub > xxx.tar.age
```

Encryption to a list of recipients at an HTTPS URL (not recursive, can't point to files or other HTTPS addresses)

```
$ echo "_o/" | age -o hello.age -r https://github.com/FiloSottile.keys
$ echo "_o/" | age -r https://filippo.io/.well-known/age.keys
```

Encryption to a GitHub user (equivalent to `https://github.com/FiloSottile.keys`)

```
$ echo "_o/" | age -r github:FiloSottile | nc 192.0.2.0 1234
```

Encryption to an alias (stored at `~/.config/age/aliases.txt`, change with `-aliases`)

```
$ cat ~/.config/age/aliases.txt
filippo: pubkey:jqmfMHBjlb7HoIjjTsCQ9NHIk_q53Uy_ZxmXBhdIpx4
ben: pubkey:ZAE2ZnRdItykp0ncAZJ2FAzIIfTvmGcgIx/759QhnQw github:Benjojo
$ tar cv ~/xxx | age -r alias:filippo > xxx.tar.age
```

Decryption with keys at `~/.config/age/keys.txt` and `~/.ssh/id_*` (no agent support)

```
$ age -decrypt hello.age
_o/
```

Decryption with custom keys

```
$ age -d -o hello -i keyA.txt -i keyB.txt hello.age
```

Encryption refuses to print to stdout if it is bound to a TTY, and so does decryption unless the payload is short and printable. Password input is only supported if a TTY is available. Duplicated aliases are both ignored and a warning is printed. Key generation checks the permissions of the output and prints a warning if world readable.

Format

The file starts with a textual header that declares the version of the age format, and encapsulates the 128-bit master file key for each recipient.

```
age-encryption.org/v1
-> X25519 SVrzdFfkPxf0LPHOUGB1gNb9E5Vr8EUDa9kxk04iQ0o
0OrTkKHpE7klNLd0k+9Uam5hkQkzMxaqKcIPRIO1sNE
-> X25519 8hWaIUmk67IuRZ41zMk2V9f/w3f5qUnXLL7MGPA+zE8
tXgpAxKgqyu1jl9I/ATwFgV42ZbNgeAlvCTJ0WgvfEo
-> scrypt GixTkc7+InSPLzPNGU6cFw 18
kC4zjzi7LRutdBfOlGHCgox8SXgfYxRYhWM1qPs0ca8
-> ssh-rsa SkdmSg
SW+xNSybDWTCkWx20FnCcxlfGC889s2hRxT8+giPH2DQMMFV6DyZpveqXtNwI3ts
5rVkW/7hCBSqEPQwabC6O5ls75uNjeSURwHAaIwtQ6riL9arjVpHMl8O7GWSRnx3
NltQt08ZpBAUkBqq5JKAr20t46ZinEIsD1LsDa2EnJrn0t8Truo2beGwZGkwkE2Y
j8mC2GaqR0gUcpGwIk6QZMxOdxNSOO7jhIC32nt1w2Ep1ftk9wV1sFyQo+YYrzOx
yCDdUwQAu9oM3Ez6AWkmFyG6AvKIny8I4xgJcBt1DEYZcD5PIAt51nRJQcs2/ANP
+Y1rKeTsskMHnlRpOnMlXqoeN6A3xS+EWxFTyg1GREQeaVztuhaL6DVBB22sLskw
XBHq/XlkLWkqoLrQtNOPvLoDO80TKUORVsP1y7OyUPHqUumxj9Mn/QtsZjNCPyKN
ds7P2OLD/Jxq1o1ckzG3uzv8Vb6sqYUPmRvlXyD7/s/FURA1GetBiQEdRM34xbrB
-> ssh-ed25519 Xyg06A rH24zuz7XHFc1lRyQmMrekpLrcKrJupohEh/YjvQCxs
Bbtnl6veSZhZmG7uXGQUX0hJbrC8mxDkL3zW06tqlWY
--- gxhoSa5BciRDt8lOpYNcx4EYtKpS0CJ06F3ZwN82VaM
[BINARY ENCRYPTED PAYLOAD]
```

The first line of the header is `age-encryption.org/` followed by an arbitrary version string. Here and below, an arbitrary string is a sequence of one or more ASCII characters with values 33 to 126. We describe version `v1`, other versions can change anything after the first line.

The rest of the header is a sequence of one or more recipient stanzas. Each recipient stanza starts with a line beginning with `->` and its type name, followed by zero or more SP-separated arguments. The type name and the arguments are arbitrary strings. Unknown recipient types are ignored. The rest of the recipient stanza is a body of canonical base64 from RFC 4648 without padding wrapped at exactly 64 columns.

`encode(data)` is canonical base64 from RFC 4648 without padding.
`encrypt[key](plaintext)` is ChaCha20-Poly1305 from RFC 7539 with a zero nonce.
`X25519(secret, point)` is from RFC 7748, including the all-zeroes output check.
`HKDF[salt, label](key)` is 32 bytes of HKDF from RFC 5869 with SHA-256.
`HMAC[key](message)` is HMAC from RFC 2104 with SHA-256.
`scrypt[salt, N](password)` is 32 bytes of scrypt from RFC 7914 with r = 8 and P = 1.

`RSAES-OAEP[key, label](plaintext)` is from RFC 8017 with SHA-256 and MGF1.
`random(n)` is a string of `n` bytes read from a CSPRNG like `/dev/urandom`.

An **X25519** recipient line is

```
-> X25519 encode(X25519(ephemeral secret, basepoint))
encrypt[HKDF[salt, label](X25519(ephemeral secret, public key))](file key)
```

where `ephemeral secret` is `random(32)` and MUST be new for every new file key,
`salt` is `X25519(ephemeral secret, basepoint) || public key`,
and `label` is `"age-encryption.org/v1/X25519"`.

An **scrypt** recipient line is

```
-> scrypt encode(salt) log2(N)
encrypt[scrypt["age-encryption.org/v1/scrypt" + salt, N](password)](file key)
```

where `salt` is `random(16)`, and `log2(N)` is the base-2 logarithm of the scrypt cost parameter in decimal. A new salt MUST be generated for every new file key.

Note that if an scrypt recipient is present it SHOULD be the only recipient: every recipient can tamper with the message, but with passwords there might be a stronger expectation of authentication.

An **ssh-rsa** recipient line is

```
-> ssh-rsa encode(SHA-256(SSH key)[:4])
RSAES-OAEP[public key, "age-encryption.org/v1/ssh-rsa"](file key)
```

where `SSH key` is the binary encoding of the SSH public key from RFC 8332. (Note that OpenSSH public key lines are `"ssh-rsa " || base64(SSH key)` in this notation.)

An **ssh-ed25519** recipient line is

```
-> ssh-ed25519 tag encode(X25519(ephemeral secret, basepoint))
encrypt[HKDF[salt, label](X25519(ephemeral secret, tweaked key))](file key)
```

where `tag` is `encode(SHA-256(SSH key)[:4])`,
`ephemeral secret` is `random(32)` and MUST be new for every new file key,
`salt` is `X25519(ephemeral secret, basepoint) || converted key`,
`label` is `"age-encryption.org/v1/ssh-ed25519"`, and `SSH key` is the binary encoding of the SSH public key from draft-ietf-curdle-ssh-ed25519-ed448-08.

The `tweaked key` for an ssh-ed25519 recipient is `X25519(tweak, converted key)` where `tweak` is `HKDF[SSH key, "age-encryption.org/v1/ssh-ed25519"]("")` and `converted key` is the Ed25519 public key converted to the Montgomery curve.

On the receiving side, the recipient needs to apply `X25519` with both the Ed25519 private scalar `SHA-512(private key)[:32]` and with `tweak`.

(I know I am using signing keys for encryption, which is unholy. I'm sorry? It would be nice to check further for cross-protocol attacks but it looks like we'll be ok. The X25519 with the tweak is meant to generate a derived key for some domain separation.)

The header ends with the following line

```
--- encode(HMAC[HKDF["", "header"](file key)](header))
```

where `header` is the whole header up to the `---` mark included.

(To add a recipient, the master key needs to be available anyway, so it can be used to regenerate the HMAC. Removing a recipient without access to the key is not possible.)

After the header the binary payload is

```
nonce || STREAM[HKDF[nonce, "payload"](file key)](plaintext)
```

where `nonce` is `random(16)` and `STREAM` is from Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance with ChaCha20-Poly1305 in 64KiB chunks and a nonce structure of 11 bytes of big endian counter, and 1 byte of last block flag (`0x00` / `0x01`).

(The STREAM scheme is similar to the one Tink and Miscreant use, but without nonce prefix as we use HKDF, and with ChaCha20-Poly1305 instead of AES-GCM because the latter is unreasonably hard to do well or fast without hardware support.)

X25519 keys

X25519 private keys are 32 random bytes sourced from a CSPRNG. They are encoded as Bech32 with HRP "`AGE-SECRET-KEY-`".

X25519 public keys are `X25519(private key, basepoint)`. They are encoded as Bech32 with HRP "`age`".

(Note that Bech32 strings can only be all uppercase or all lowercase, but the checksum is always computed over the lowercase string.)

This is the encoding of a keypair where the private key is a buffer of 32 `0x42` bytes:

```
age1zvkyg2lqzraa2lnjvqej32nkuu0ues2s82hzrye869xeexvn73equnujwj
AGE-SECRET-KEY-1GFPYYSJZGFPYYSJZGFPYYSJZGFPYYSJZGFPYYSJZGFPYYSJZGFPQ4EGAEX
```

ASCII armor

age files can be encoded as PEM with a block type of `AGE ENCRYPTED FILE`.

PEM is a catastrophically malleable format; implementations are encouraged to be as strict as workable. The reference implementation requires canonical Base64, rejects garbage before and after the message, and doesn't support headers. Note that regular age files are not malleable.

Changes

2019-05-16: added "created" comment to generated keys. Via @BenLaurie.

2019-05-16: added RSA-OAEP label. Via @feministPLT.

2019-05-16: moved `~/.config/age.keys` to `~/.config/age/keys.txt` and added aliases. Via @BenLaurie and @__agwa.

2019-05-19: added Ed25519 tweak and switched to SHA-512 everywhere for consistency. Via kwantam.

2019-05-19: removed public key hash from header to get recipient privacy like gpg's `--throw-keyid`. Via private DM.

2019-05-19: replaced egocentric GitHub link with `age-tool.com`.

2019-05-26: reintroduced public key hash for SSH keys to identify encrypted and hardware keys. Via private DM. (For better privacy, use native keys.)

2019-05-26: included X25519 shares in derived key according to RFC 7748, Section 6.1 by using HKDF as suggested in RFC 5869, Section 3.1.

2019-05-26: documented that aliases can expand to multiple keys.

2019-05-26: swapped scrypt for Argon2 in the name of implementation ubiquity. Switched back to SHA-256 to match the scrypt core hash.

2019-05-26: rewrote the Format section in terms of RFCs. Made minor changes to accommodate that, most importantly now using X25519 to apply the ssh-ed25519 tweak scalar.

2019-06-06: added “Maybe in v2” section, moved PKCS#11 to it.

2019-06-06: added header HMAC. Via @lasagnasec.

2019-06-12: added a nonce to the HKDF payload key derivation, making the file key reusable. (Mostly for misuse resistance.)

2019-06-12: introduced requirement for an scrypt recipient to be the only one.

2019-06-24: settled the important question, the pronunciation. It’s “g” like in “gif”.

2019-07-11: made the ssh-ed25519 tweak 64 bytes to reduce bias. (Which is free because the reduction doesn’t have to be constant time.) Pointed out at a Bar Pitti table, chose to donate £50 to ProPublica.

2019-07-20: added AEAD field to the closing of the header.

2019-10-06: removed AEAD field.

2019-10-06: made the ssh-ed25519 tweak 32 bytes again, so we can use X25519 to apply it, and there is no need for a scalar field implementation anywhere.

2019-10-08: changed the scrypt work factor field to log(N). See #10.

2019-10-13: made ssh-rsa body wrap at 56 columns, so it cuts along byte boundaries.

2019-11-24: specified the ASCII armored format. See #17.

2019-11-27: updated the CLI to use options for recipients and identities, and an optional argument for the input. See #22.

2019-12-27: switched keys to Bech32, armor to PEM, base64 encoding to the standard alphabet, and ssh-rsa body columns to 64. See discussion.

2019-12-28: switched intro and labels to `age-encryption.org/v1`. Added a label prefix to the scrypt salt. Recipients are now all version scoped.

2019-12-28: clarified how ssh-ed25519 differs from X25519. See discussion.

2019-12-29: documented the key format and generation.

2020-01-08: specified the generic recipient stanza format. See #9.

2020-03-25: clarified that arbitrary strings can’t be empty.