Introduction to Sigstore: Cryptographic signatures made easier

Maya Costantini
Software Engineer, Red Hat

Pass the SALT 2023

About me

Maya Costantini

Software Engineer,

Red Hat Emerging Technologies
Security team

@MayaCostantini

hachyderm.io/@mayacostantini

@mayaCostantini

Supply chain security: why are signatures important?

- **Software Supply Chain**: the end-to-end journey software takes from development to distribution, involving the tools and people responsible for its delivery
- Developers, version control, build systems, registries, deployment platforms...
- Attackers play on developer expectations of **systematic reproducibility** to find vulnerable links in a Software Supply Chain
- Cryptographic signatures guarantee:
  - Software **integrity**
  - Software **authenticity**

Signing software before Sigstore

Challenges of using OpenPGP/GPG for software signing

- **Public key distribution:** ensure recipients have access to the correct public keys to verify the authenticity of software
- **Private key storage and rotation:**
  - Safeguarding private keys is costly and leaks happen anyway
  - Need to regularly rotate signing keys to protect from key compromise
- **Intricate command line options**
- **Occasional need for cryptography knowledge**

See: *PGP signatures on PyPI: worse than useless* on blog.yossarian.net

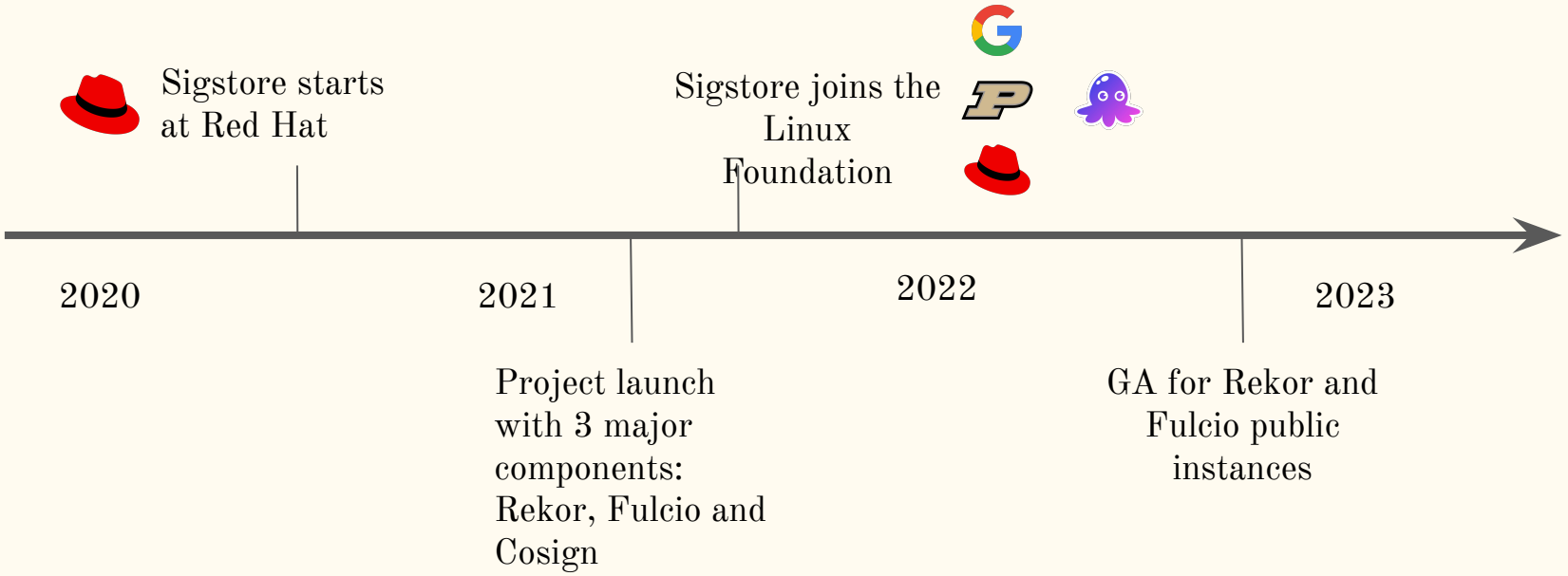
A key signing party in front of FOSDEM 2008, Wikipedia

"Become to digital signatures what Let’s Encrypt is to HTTPS"

- A free and automated Certificate Authority
- Allows any domain owner to obtain a trusted certificate at zero cost
- Over 256M active certificates delivered since 2016 (~3M a day)

- A free service for signing digital artifacts
- Signatures are logged publicly for verification
- Over 25M entries stored since 2021

Project Timeline



What is Sigstore?

Sigstore solves common issues with current signature schemes that prevent developer adoption:

- No knowledge of cryptography or PKI protocols required.
- A simple interface to make signing accessible to everyone
- No more private keys management and rotation
- Easier auditing and revocation in case of compromise
- Signatures are bound to a public **identity** instead of a public key

What is Sigstore?

sigstore rekor

Signature transparency log

sigstore fulcio

Free Certificate Authority

sigstore cosign

CLI to sign and verify artifacts

+ ecosystem-specific clients (Python, JavaScript, Rust...)

Architecture/Spec

OpenSSF Project

Projects

sigstore cosign

sigstore fulcio

sigstore gitsign

sigstore rekor

Deployments

sigstore public service

l.openssf.org

OpenSSF Landscape

OpenSSF

Sigstore is a new standard for signing, verifying, and protecting software. It is an OpenSSF project and this landscape is intended as a map to explore the Sigstore ecosystem.

Integrations

Artifact HUB

flux

HARBOR

HELM

in-toto

JReleaser

KUBEWARDEN

Kyverno

TEKTON CHAINS

TUF

aqua trivy

Language Clients

GO

Maven

JS

python

Ruby

Signed With

cilium

python

flux

Istio

JENKINS X

Karpenter

KEDA

Knative

kubernetes

KUBEWARDEN

Kyverno

Ockam

TEKTON

urllib3

Case Studies

AUTODESK

EDGELESS SYSTEMS

RANCHER

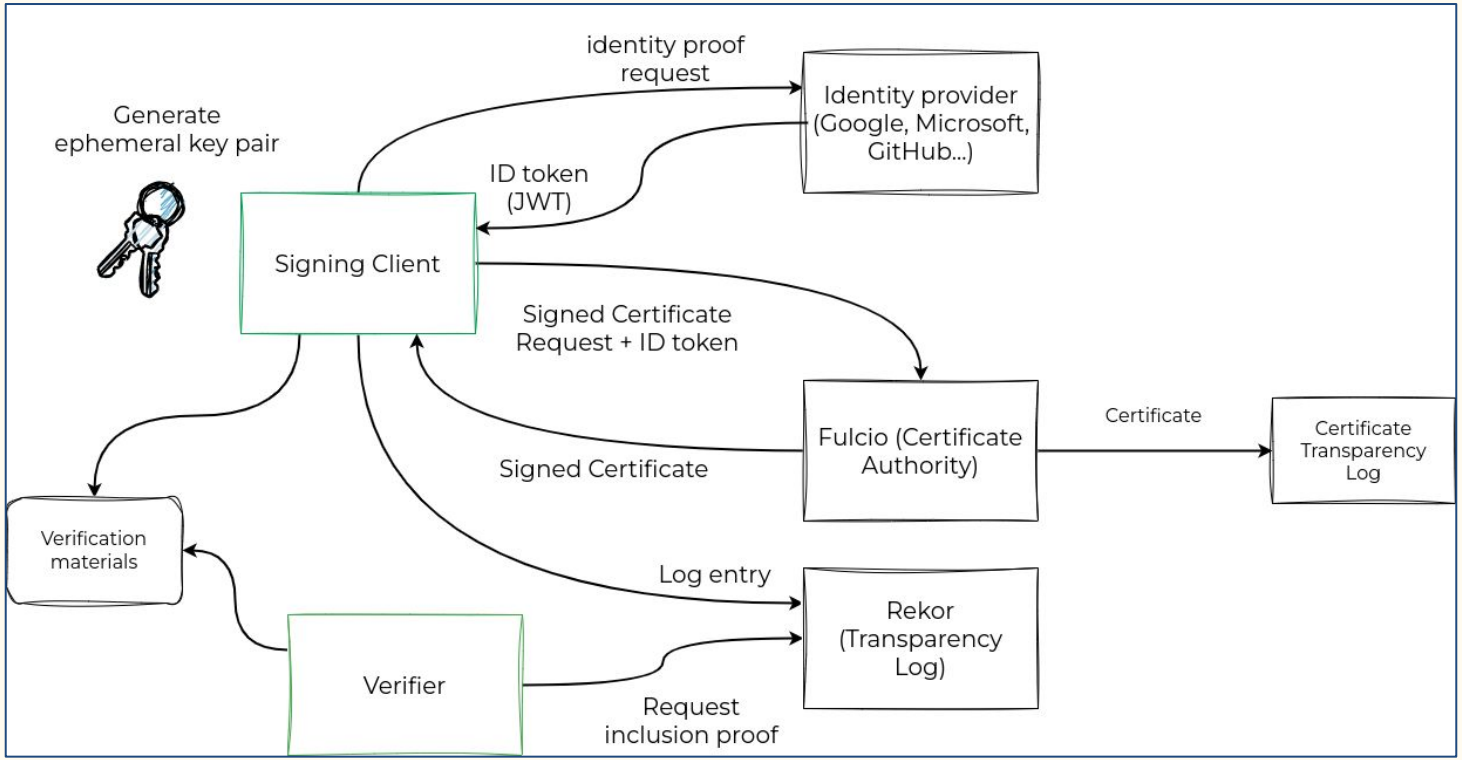
verizon

Demo:
Signing and verifying a file with the Cosign CLI

How does it work?

Sigstore's "keyless" signing workflow

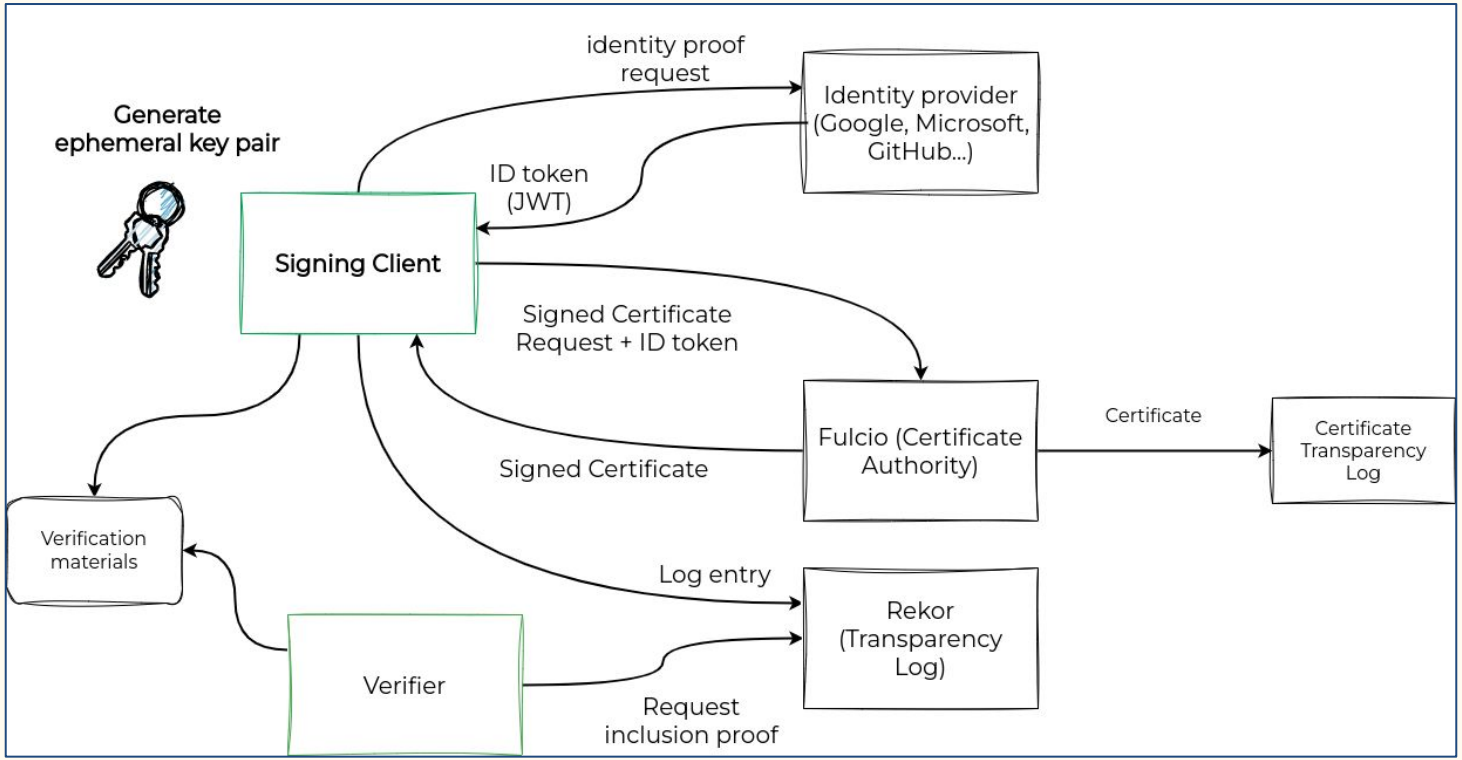
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

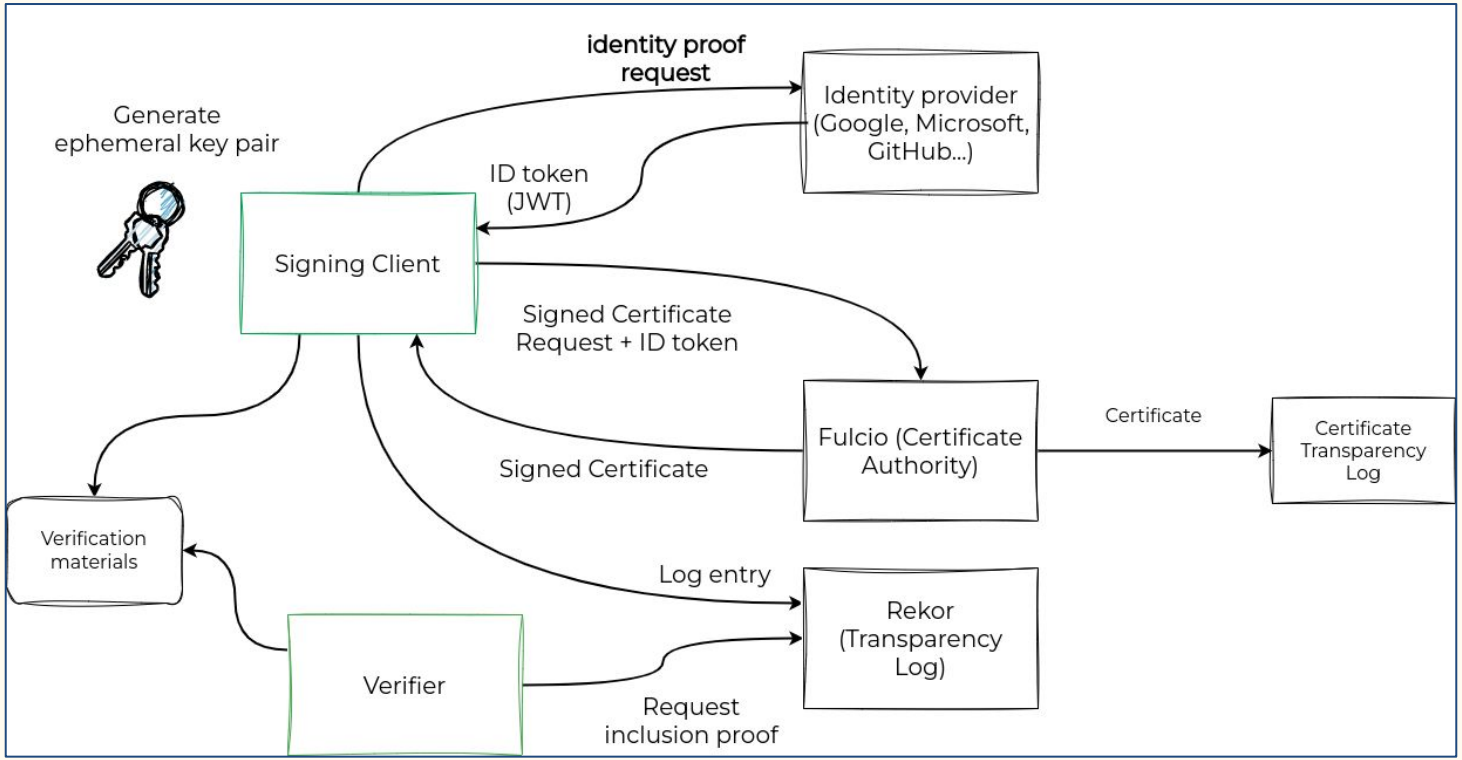
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

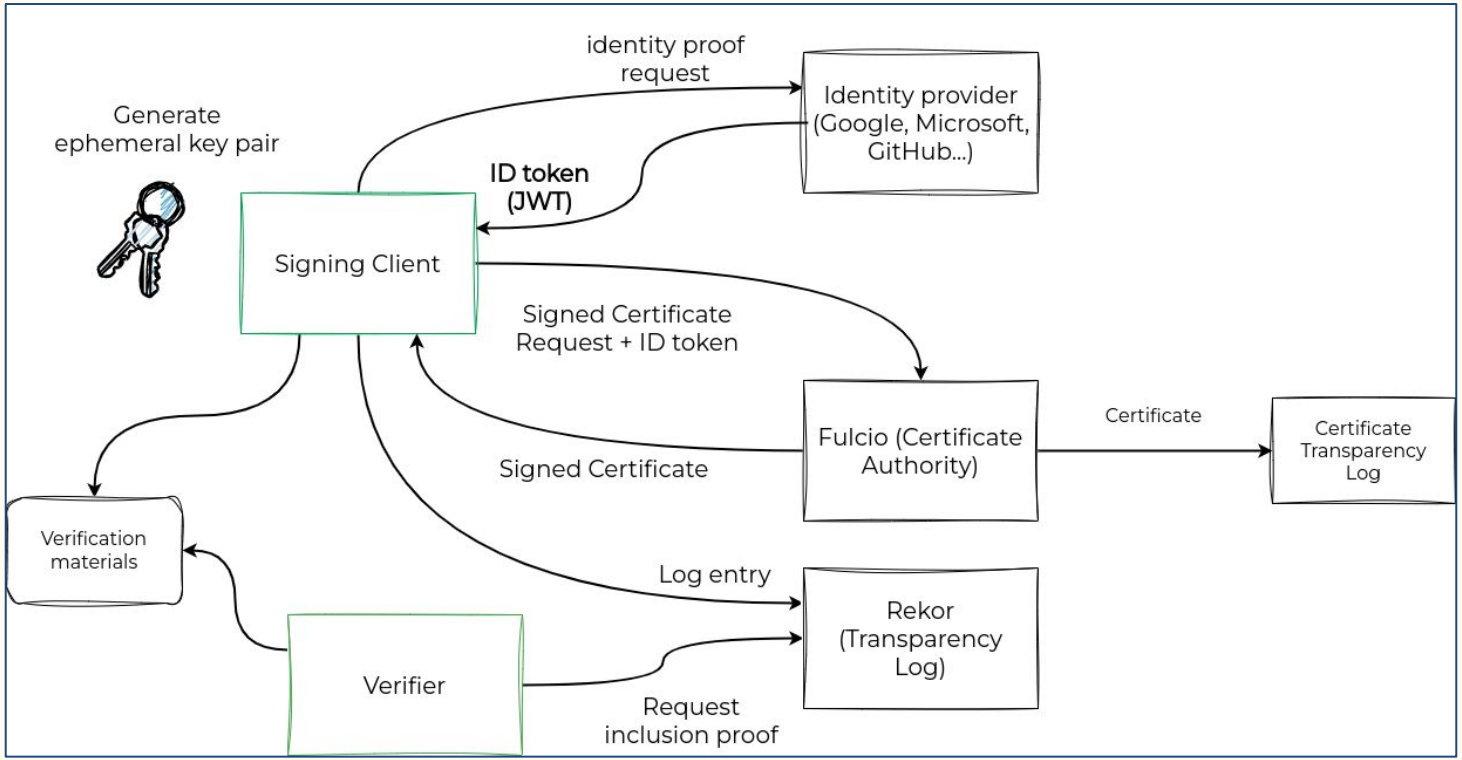
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

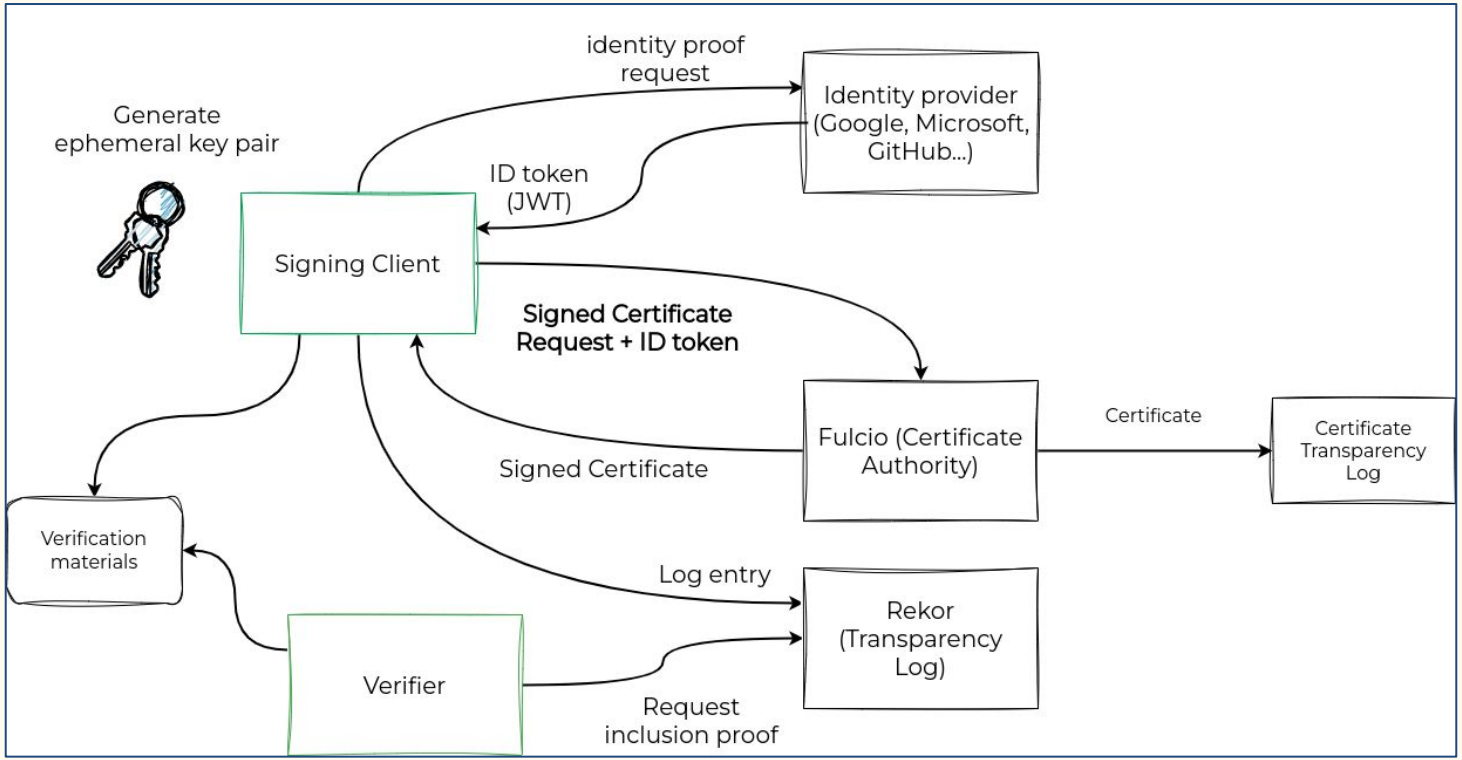
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

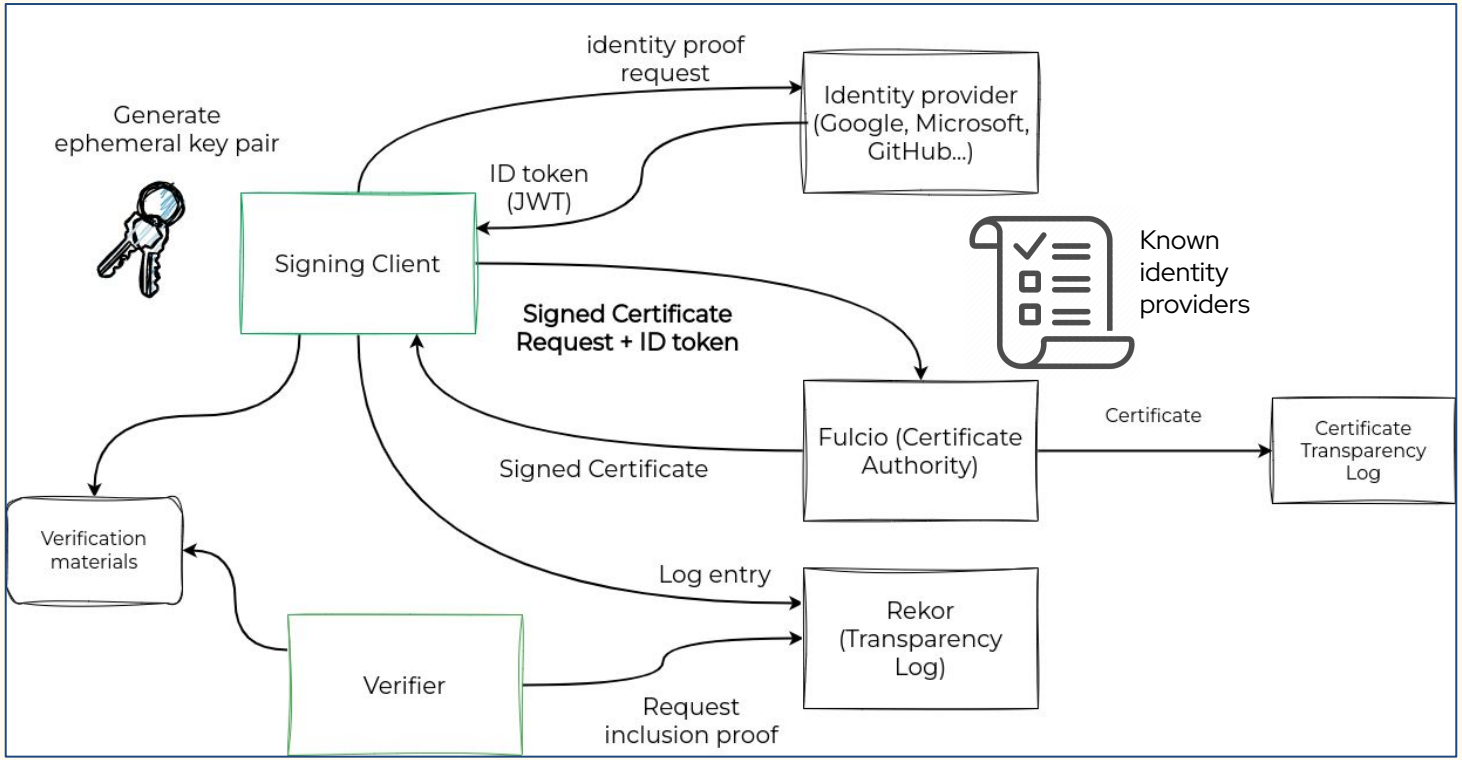
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

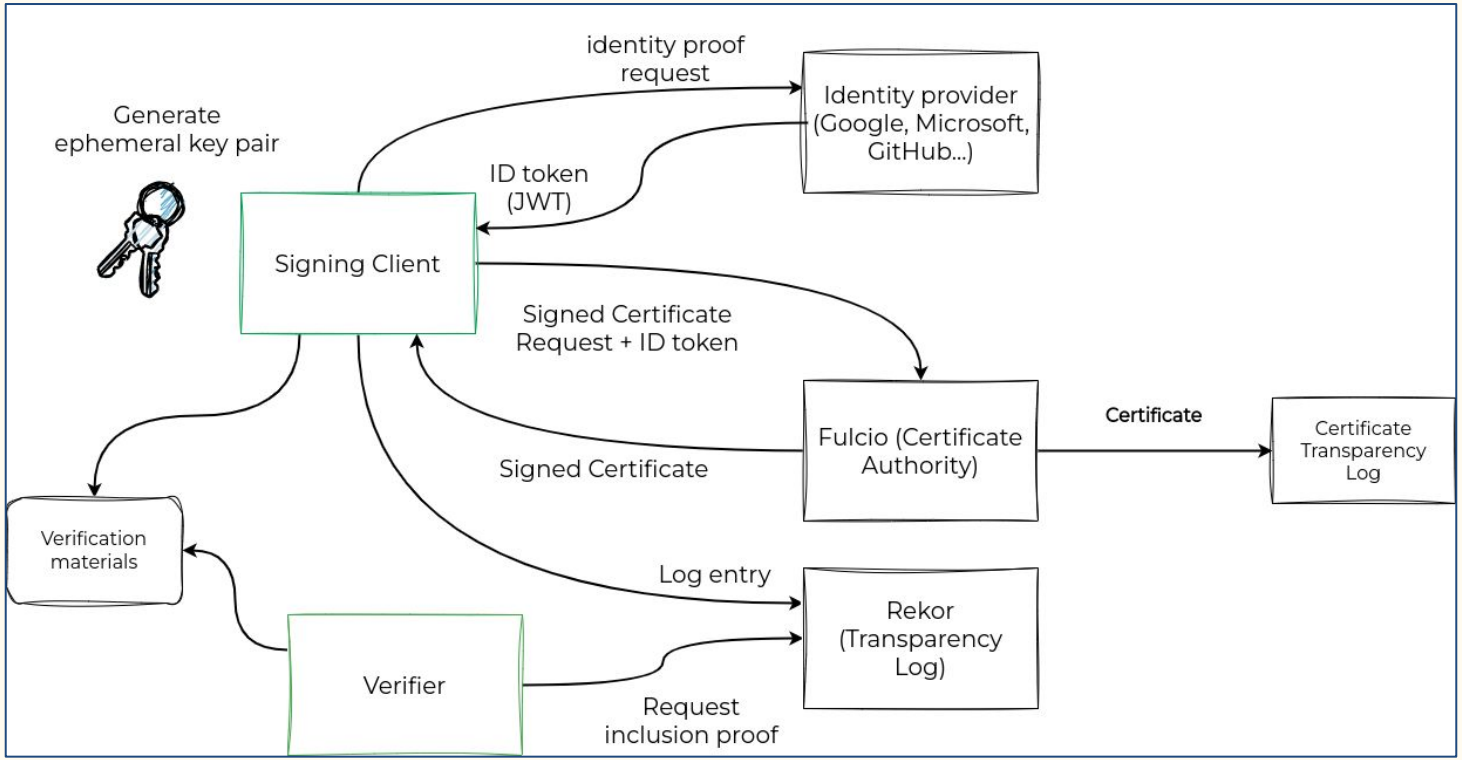
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

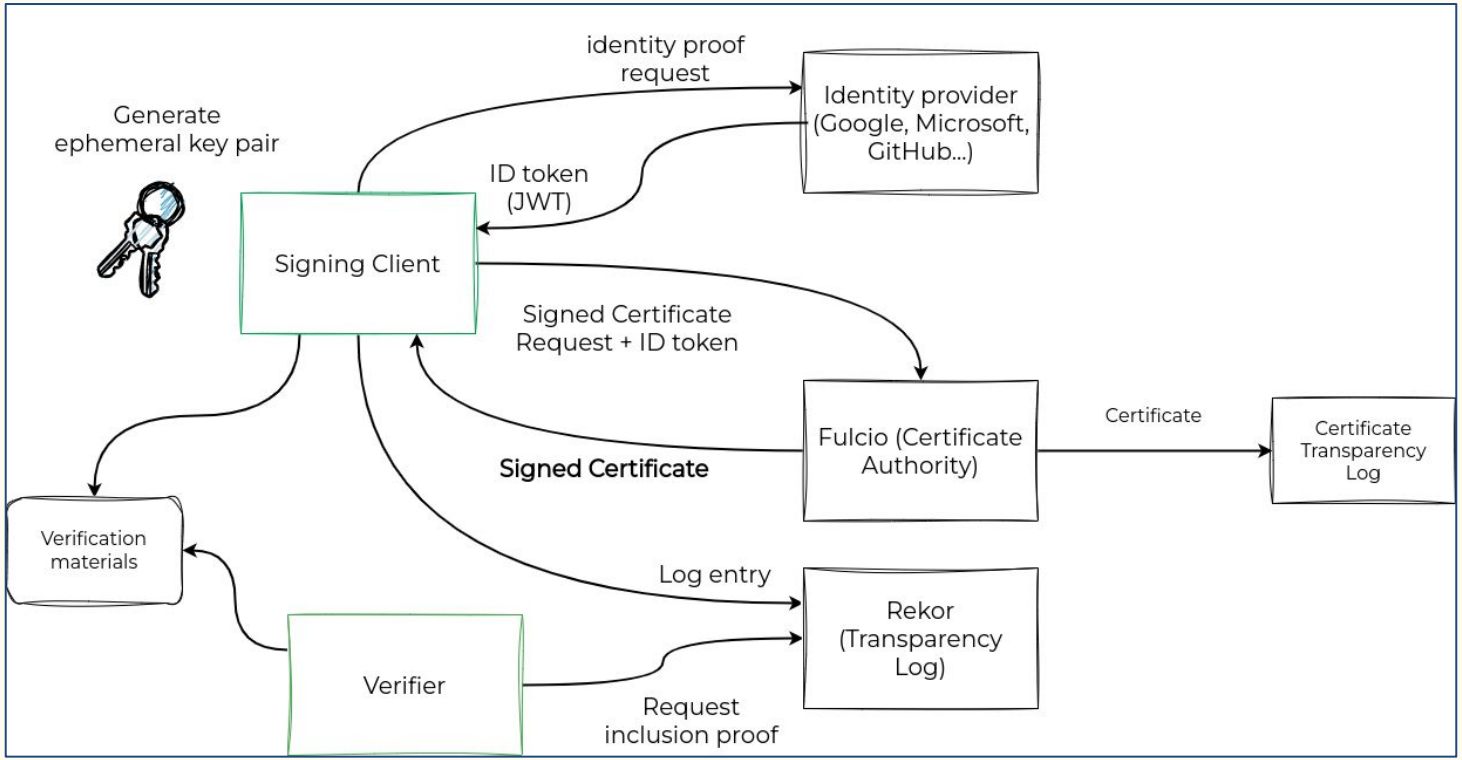
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

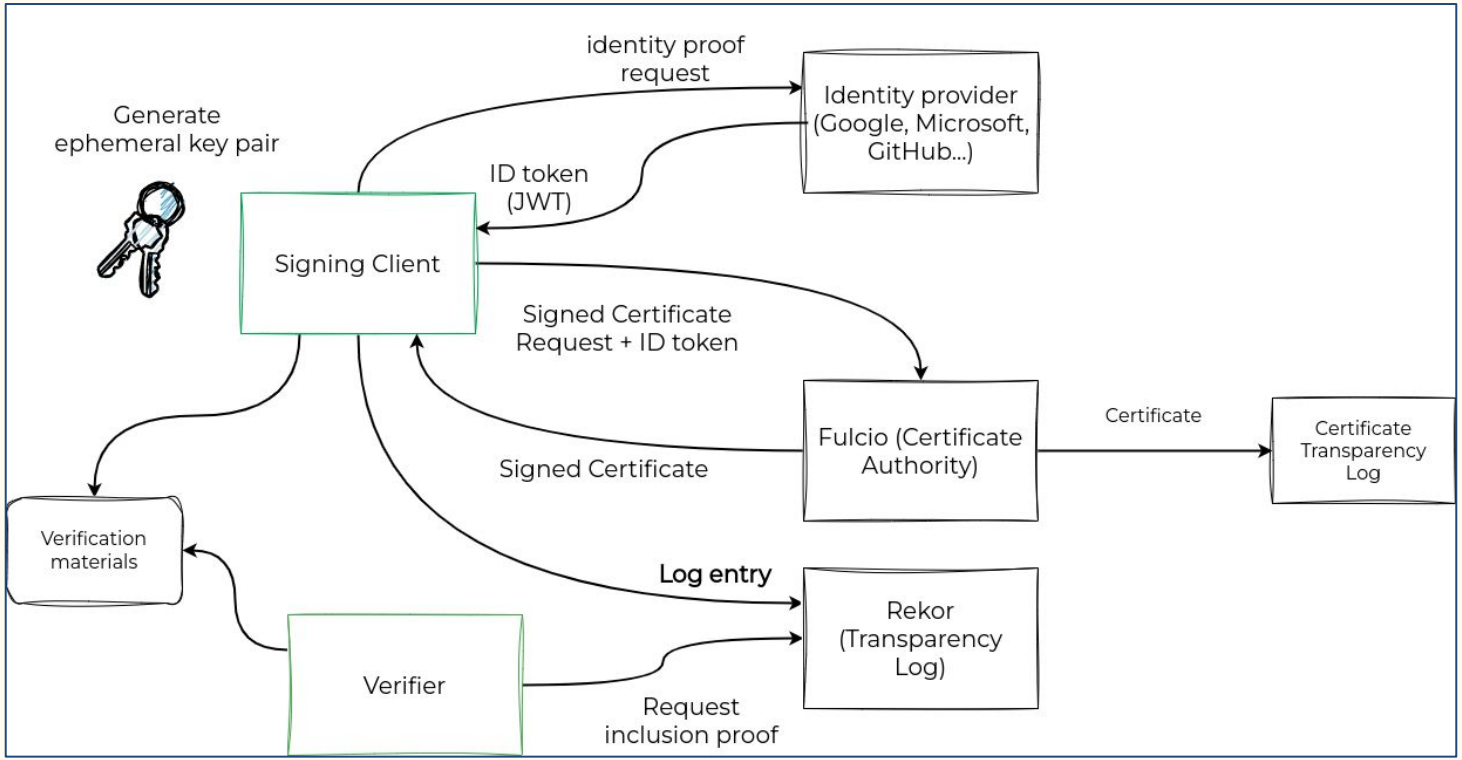
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

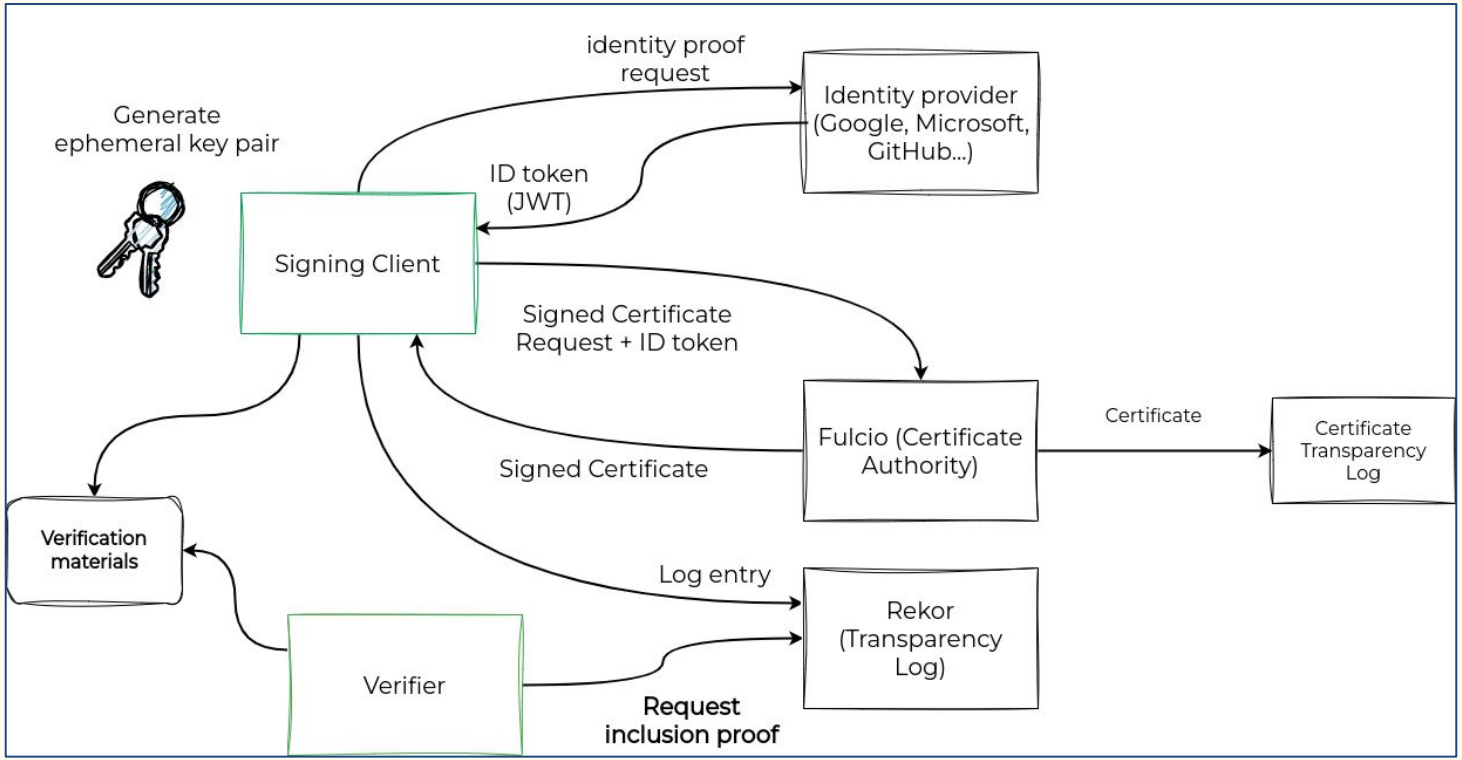
Signing an artifact



How does it work?

Sigstore's "keyless" signing workflow

Verifying a Sigstore signature

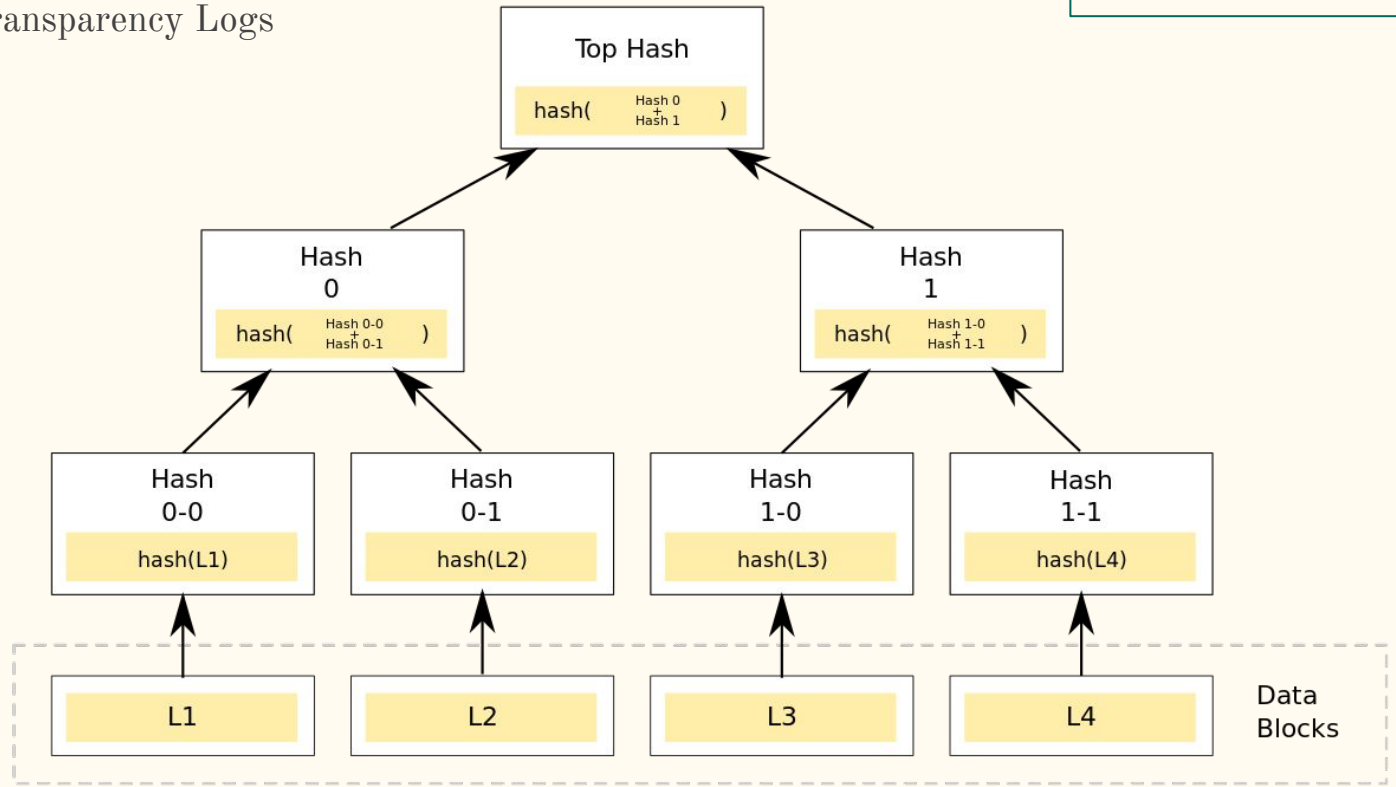


How does it work?

Behind Sigstore's Transparency Logs

- Immutable
- Read-only

Merkle Tree



Join the community and get involved

sigstore.dev/community

https://links.sigstore.dev/slack-invite

https://www.youtube.com/@projectsigstore

https://blog.sigstore.dev/

Thank you!

Q&A