DEVCONF.cz

An introduction to Sigstore for Pythonistas

DevConf.CZ 2023

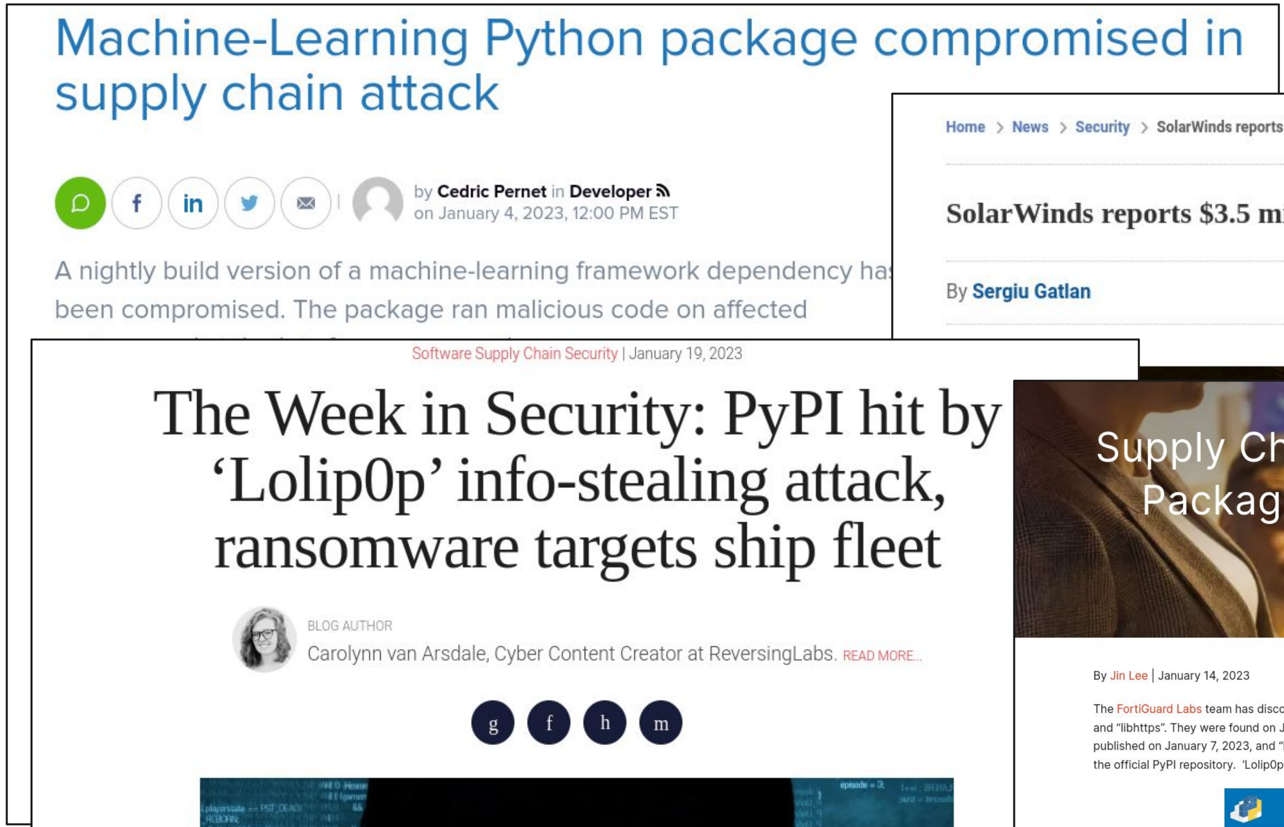
Maya Costantini
Software Engineer, Red Hat

Agenda

- **The importance of digital signatures in supply chain security**
    - ➔ How does signing your code protect you against supply chain attacks?
- **Current practices for signing software**
    - ➔ ... and the need for an alternative
- **What is Sigstore?**
    - ➔ Core principles and philosophy
    - ➔ Adoption in Open Source and by the Python community
- **Demo**
    - ➔ How **you** can use Sigstore to secure your Python project
- **Under the hood: a (quick) dive on Sigstore internals**
- **Q&A**

Supply chain security: what are digital signatures, and why are they important?

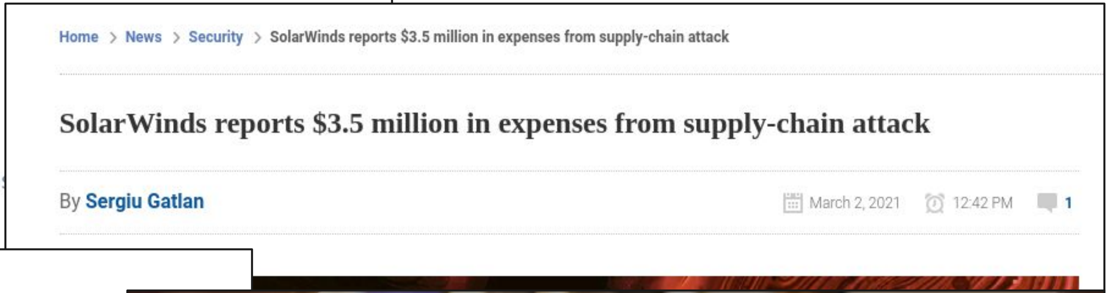
Supply chain security: why are signatures important?



Machine-Learning Python package compromised in supply chain attack

by Cedric Pernet in Developer
on January 4, 2023, 12:00 PM EST

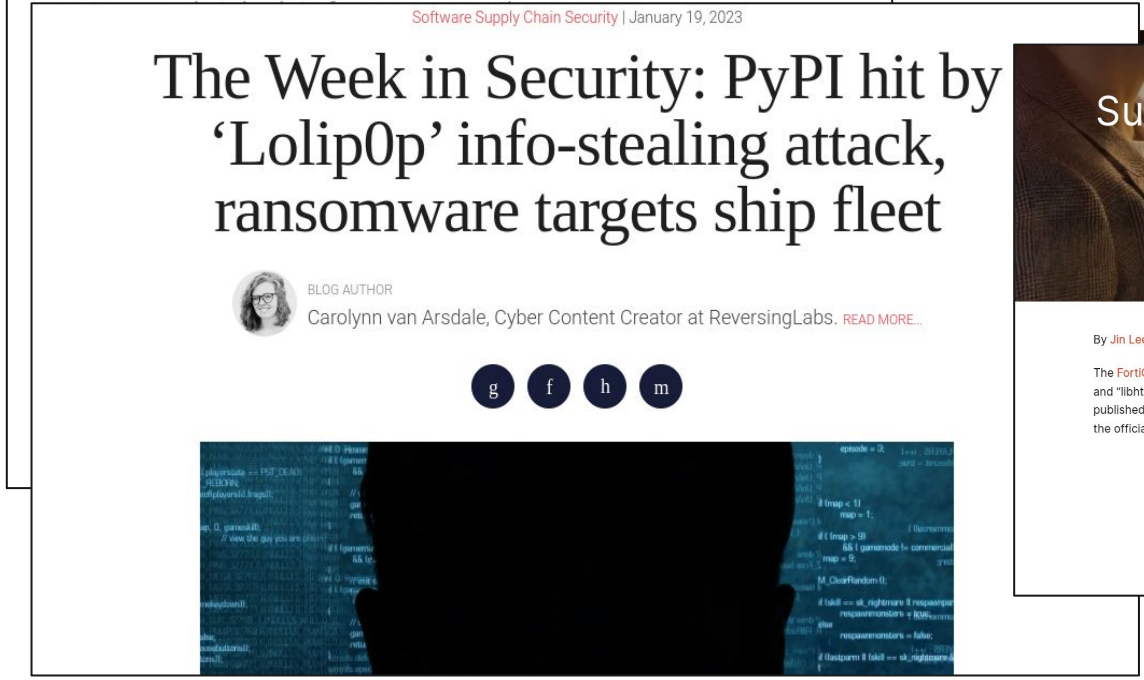
A nightly build version of a machine-learning framework dependency has been compromised. The package ran malicious code on affected



Home > News > Security > SolarWinds reports $3.5 million in expenses from supply-chain attack

SolarWinds reports $3.5 million in expenses from supply-chain attack

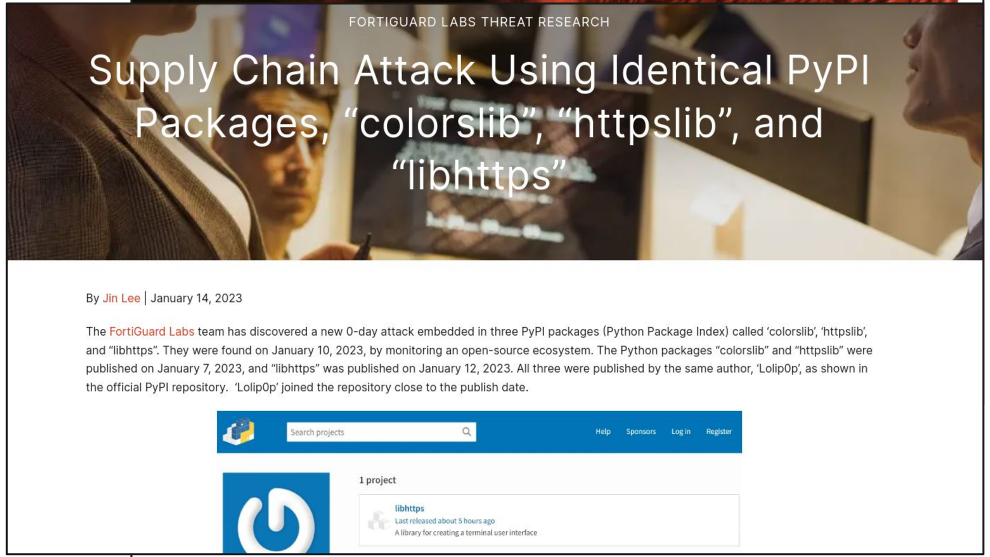
By Sergiu Gatlan March 2, 2021 12:42 PM 1



Software Supply Chain Security | January 19, 2023

The Week in Security: PyPI hit by ‘Lolip0p’ info-stealing attack, ransomware targets ship fleet

BLOG AUTHOR
Carolynn van Arsdale, Cyber Content Creator at ReversingLabs. READ MORE...



FORTIGUARD LABS THREAT RESEARCH

Supply Chain Attack Using Identical PyPI Packages, “colorslib”, “httpslib”, and “libhttps”

By Jin Lee | January 14, 2023

The FortiGuard Labs team has discovered a new 0-day attack embedded in three PyPI packages (Python Package Index) called ‘colorslib’, ‘httpslib’, and “libhttps”. They were found on January 10, 2023, by monitoring an open-source ecosystem. The Python packages “colorslib” and “httpslib” were published on January 7, 2023, and “libhttps” was published on January 12, 2023. All three were published by the same author, ‘Lolip0p’, as shown in the official PyPI repository. ‘Lolip0p’ joined the repository close to the publish date.

Search projects
Help
Sponsors
Log in
Register
1 project
libhttps
Last released about 5 hours ago
A library for creating a terminal user interface

Supply chain security: why are signatures important?

…In an era of new security challenges faced by the Python community

- Software supply chain attacks are on the rise (**742%** in the past 3 years)
- Recent attacks targeting Python software users on PyPI:
  - Typosquatting well-known library names (`beautifulsoup4` ➔ `beauitfulsoup4`, `requests` ➔ `requesys`…)
  - Dependency confusion attacks (see the PyTorch-nightly dependency compromise)
  - Packages running arbitrary source code on installation
  - Malware injection by attackers taking over a project maintainer account
- **PyPI temporarily suspended new user registrations and project uploads** as the package index faced a surge of malicious package uploads

Supply chain security: why are signatures important?

As a key component of a secure Software Supply Chain

- ***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

Introducing Sigstore: code signing made easy and accessible

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

Let's Encrypt

- 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)

sigstore

- A free service for signing digital artifacts
- Signatures are logged publicly for verification
- Over **20M** entries stored since 2021

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

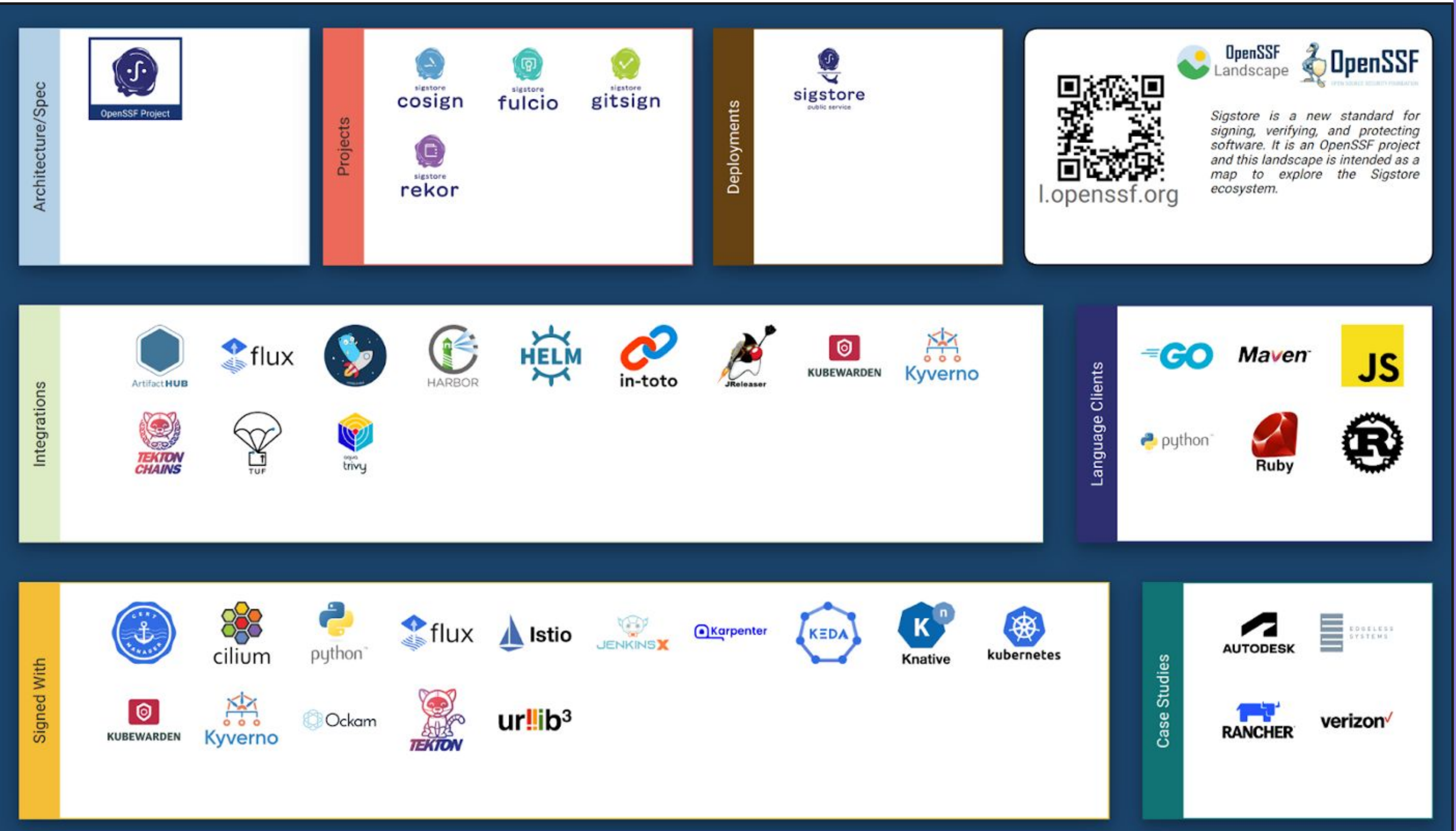
Free Certificate Authority

sigstore cosign

CLI to sign and verify artifacts

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

Sigstore Open Source adoption



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.

Sigstore in the Python ecosystem

A Python language client: sigstore-python to integrate Sigstore into your Python project

- Sign files and blobs from the command line using a **“keyless”** workflow, interactively or with ambient credentials
- Sign artifacts in a GitHub CI workflow with the sigstore-python GitHub Action
- Integrate sigstore-python natively into a Python project using the library public API, stable since v1.0.0

```
$ pip install sigstore
```

Sigstore in the Python ecosystem

Sigstore adoption in Python packaging

- PEP 480: *Surviving a Compromise of PyPI: End-to-end signing of packages*
- Extends PEP 458: *Secure PyPI downloads with signed repository metadata* with a “maximum security model” protecting end users against a PyPI compromise
- Package managers will support verifying signatures
- Similar user experience, with guarantees of integrity

Sigstore in the Python ecosystem

Sigstore's Python client is now used to sign releases of CPython

```
$ python -m sigstore verify identity \
  --certificate Python-3.11.0.tgz.crt \
  --signature Python-3.11.0.tgz.sig \
  --cert-identity pablogsal@python.org \
  --cert-oidc-issuer https://accounts.google.com \
  Python-3.11.0.tgz
```

python.org/download/sigstore/

Demo time:

Signing and verifying a Python file with sigstore-python

How does it work?

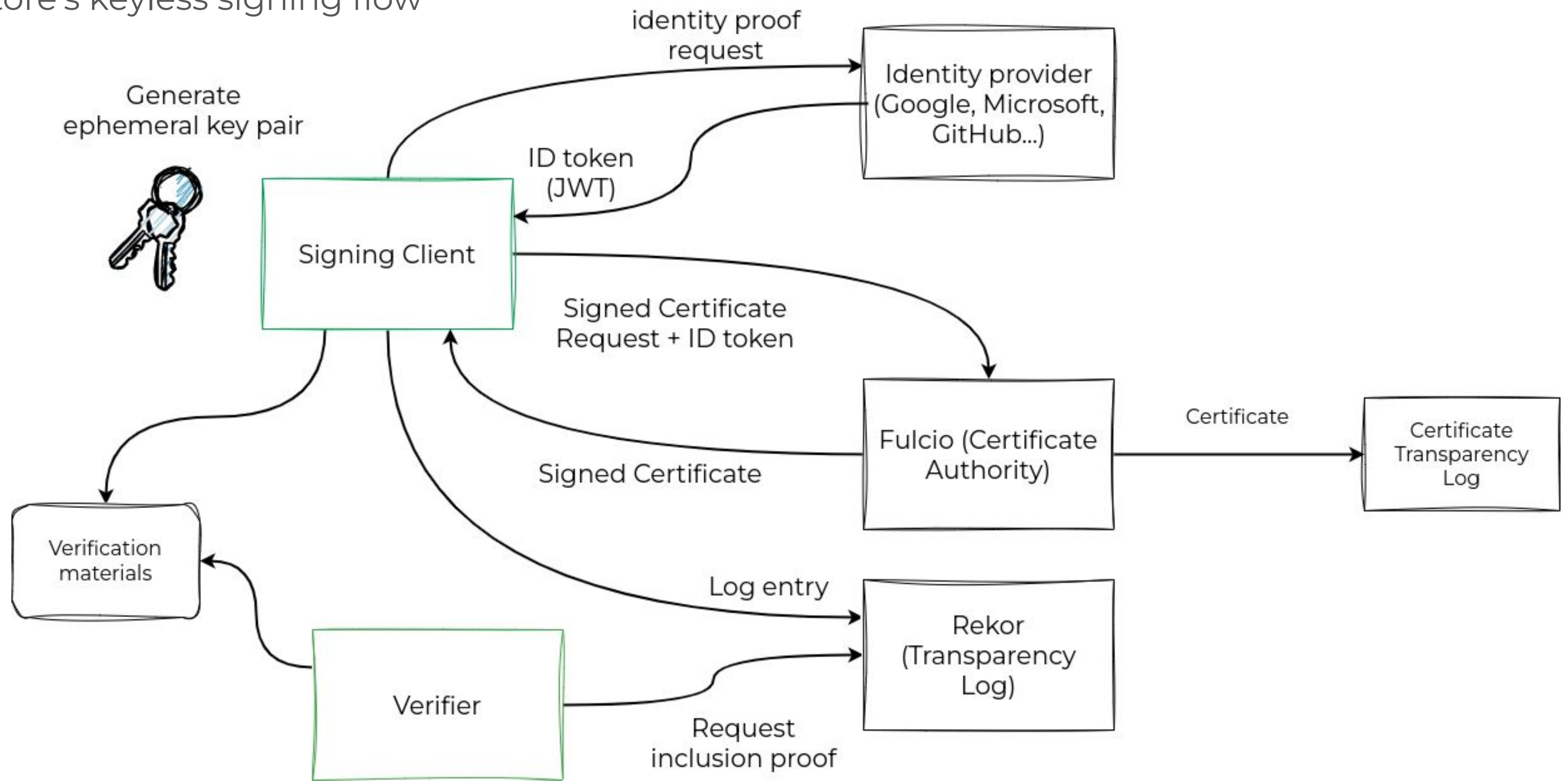
Sigstore’s keyless signing flow

- **Ephemeral keys:** no self-managed private keys are used, artifacts are signed using an ephemeral private key that stays in **memory**
- **Ephemeral signing certificates:** Fulcio generates X509 signing certificates valid for 10min only. The Fulcio **Certificate Transparency Log** keeps track of all certificates issued
- **Permanently logged signatures:** Signatures and their metadata are stored in the Rekor Transparency Log in an **immutable, append-only** way
- **Publicly verifiable signatures:** The Rekor and Fulcio public instances **publicly log** signatures and certificates, which are verifiable by all using Cosign (or a language-specific client)

How does it work?

Sigstore's keyless signing flow

Signing an artifact

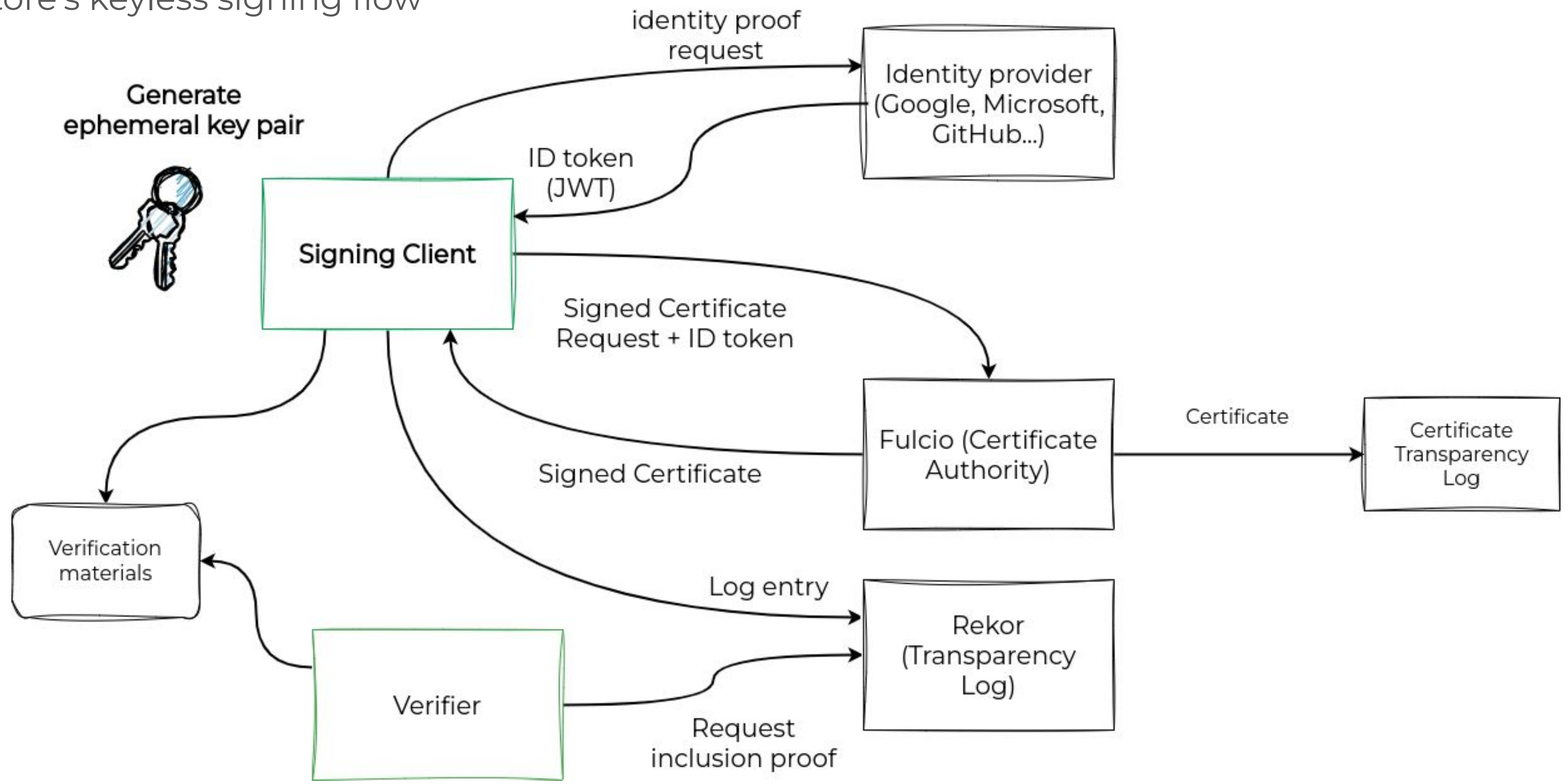


Generate ephemeral key pair
identity proof request
Identity provider (Google, Microsoft, GitHub...)
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

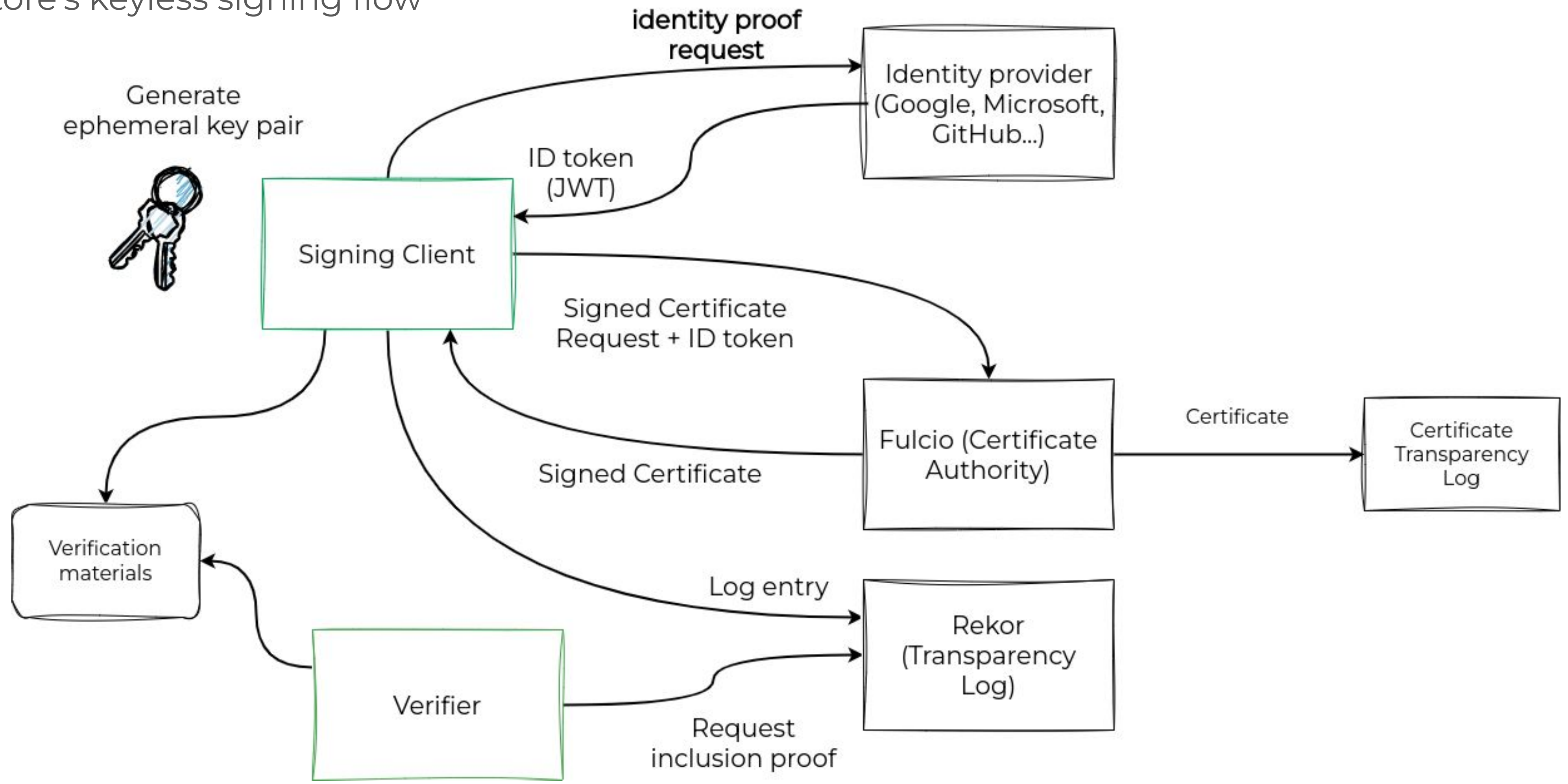
Signing an artifact


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

Signing an artifact

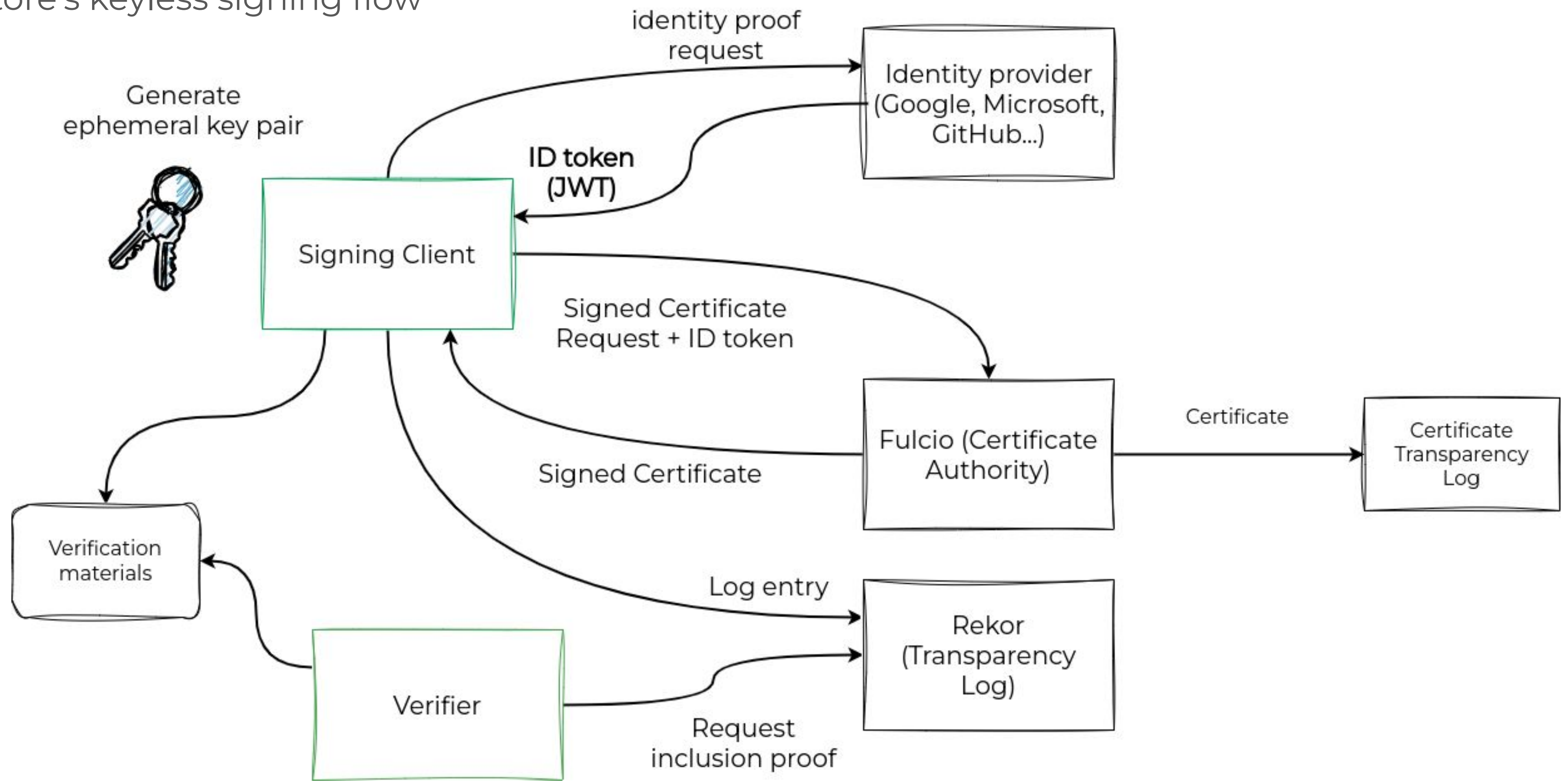


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

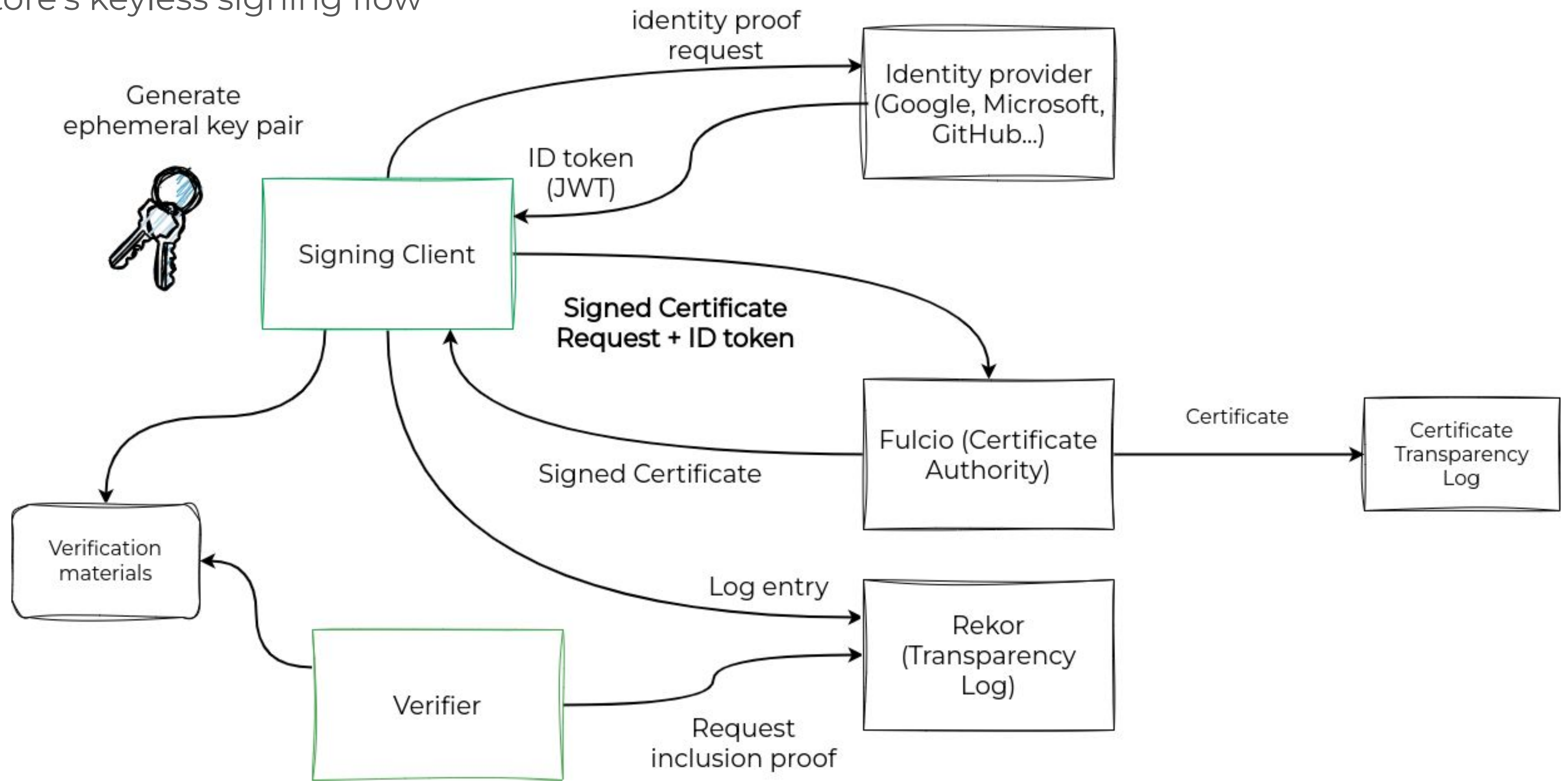
Signing an artifact


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

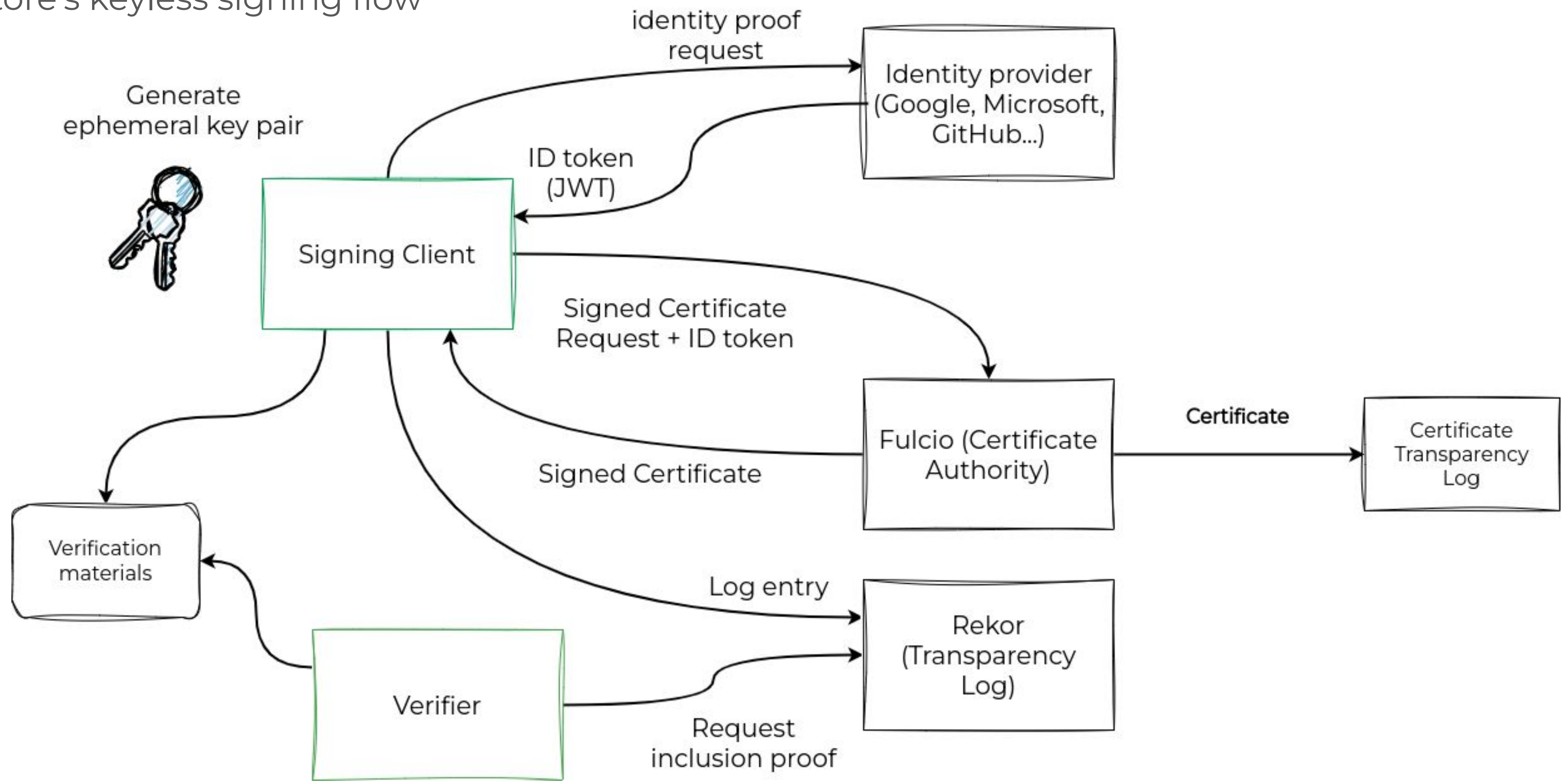
Signing an artifact


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

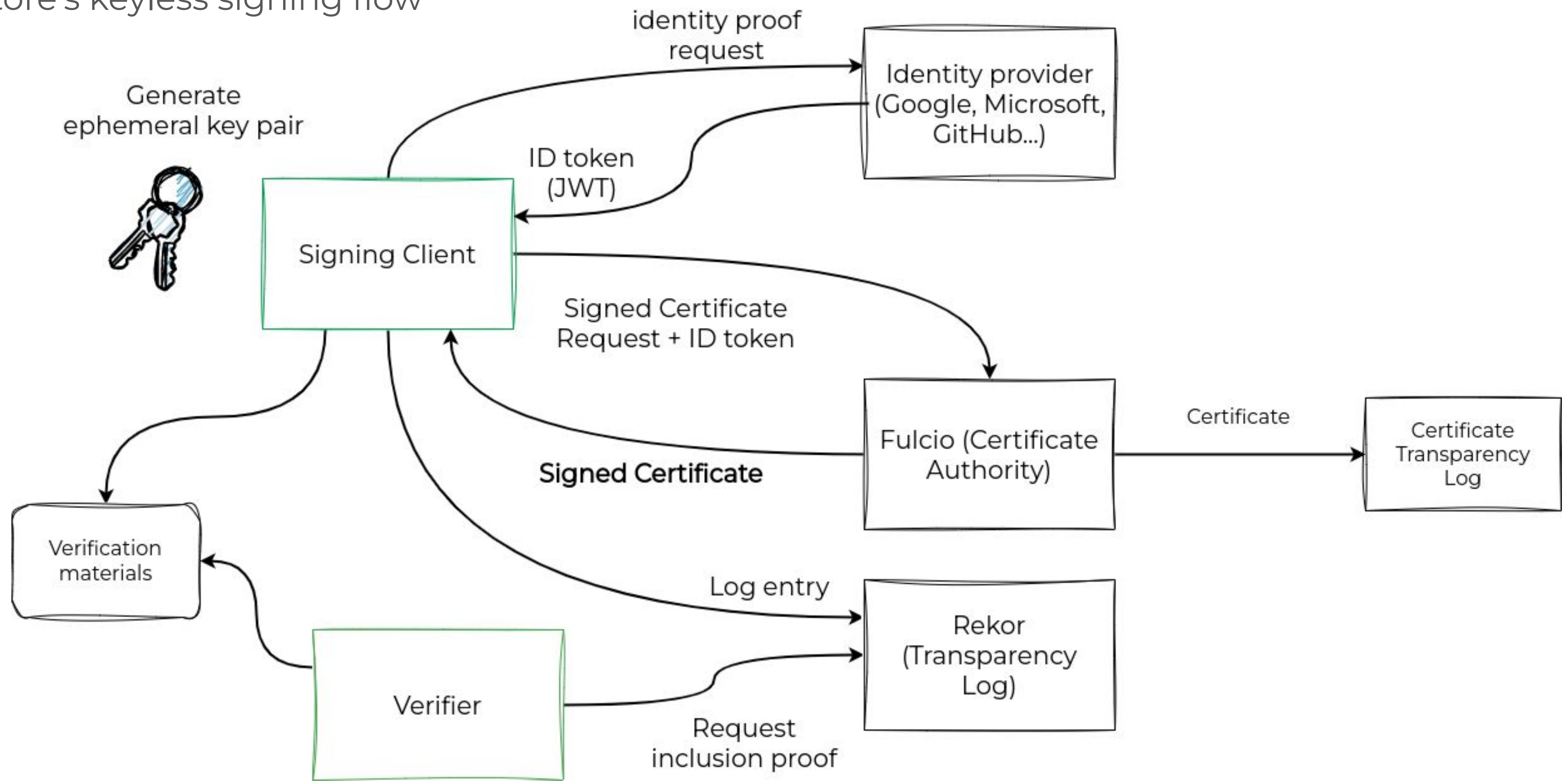
Signing an artifact


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

Signing an artifact

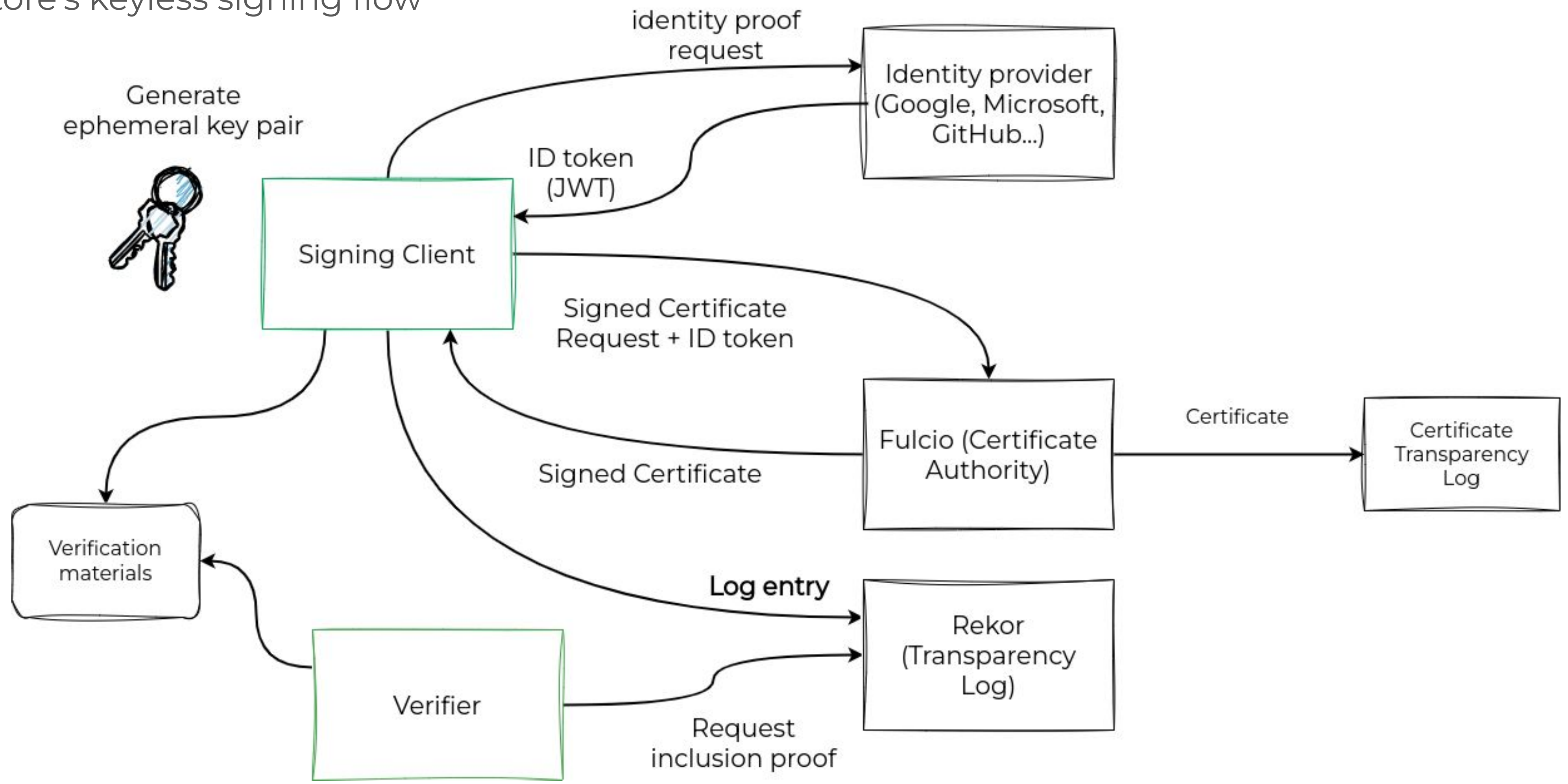


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

Signing an artifact

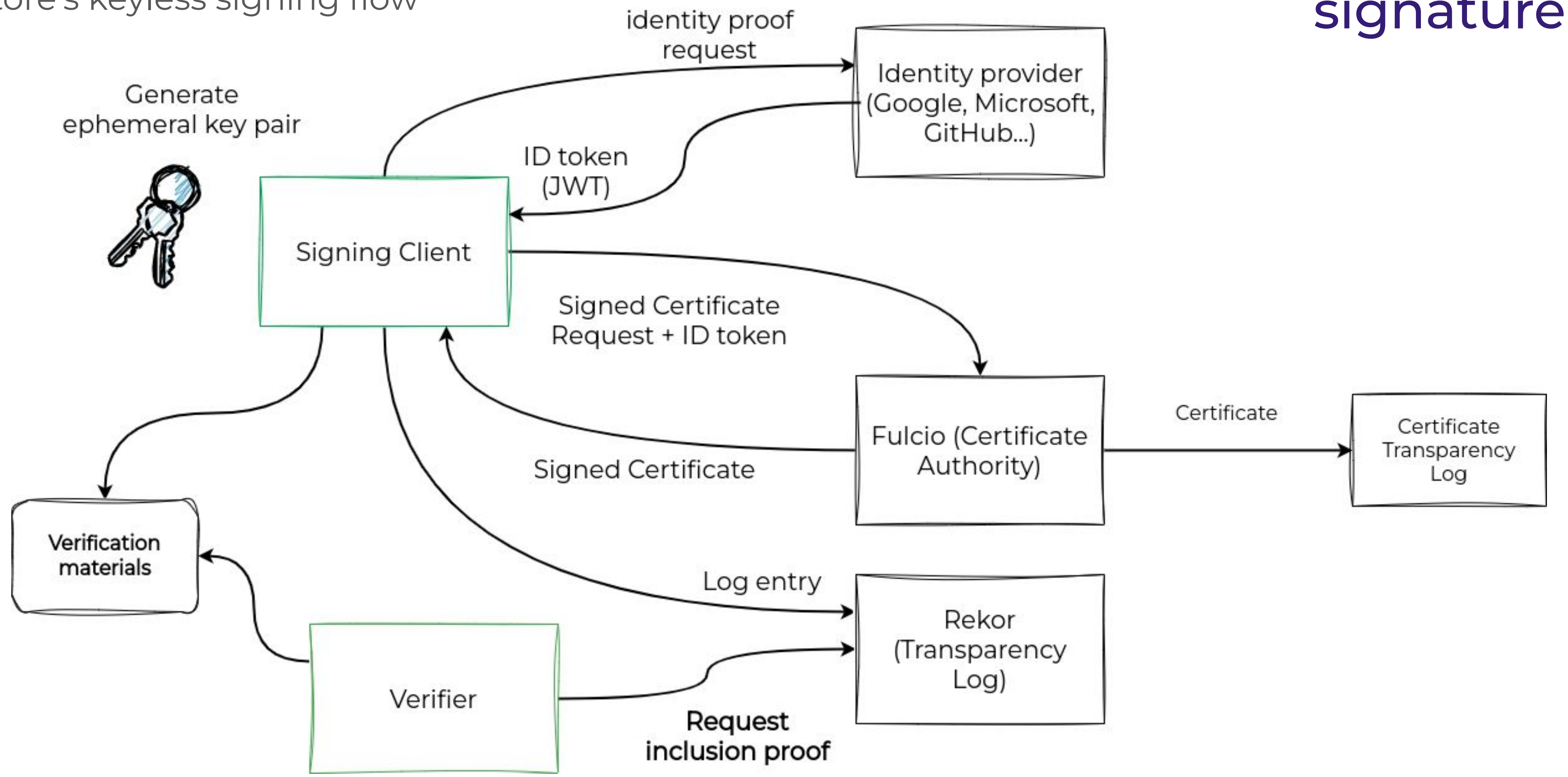


identity proof request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Fulcio (Certificate Authority)
Certificate
Certificate Transparency Log
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

How does it work?

Sigstore's keyless signing flow

Verifying a Sigstore signature


identity proof
request
Identity provider (Google, Microsoft, GitHub...)
Generate ephemeral key pair
ID token (JWT)
Signing Client
Signed Certificate Request + ID token
Certificate
Certificate Transparency Log
Fulcio (Certificate Authority)
Signed Certificate
Verification materials
Log entry
Rekor (Transparency Log)
Verifier
Request inclusion proof

Takeaways

- **Ensure Software authenticity:** Provide a strong evidence of authenticity and establish trust with users
- **Protect against supply chain attacks:** Avoid supply chain compromises by verifying dependencies provenance and integrity
- **Eliminate private key management:** No need to store permanent private keys, delegate security considerations to the OpenID Connect identity provider with secure logging like MFA, OTP, biometric authentication...
- **Enhance Transparency:** Allow everyone to verify your software via a centralized and trusted log and facilitate audits in case of compromise

Join the Sigstore community and get involved

sigstore.dev/community

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

Sigstore YouTube channel

https://blog.sigstore.dev/

Thank you!

Q&A

@MayaCostantini

hachyderm.io/@mayacostantini

@mayaCostantini