Introduction to Sigstore:
Cryptographic signatures made easier

Maya Costantini
Software Engineer, Red Hat
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](https://blog.yossarian.net) on blog.yossarian.net

"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 25M entries stored since 2021
Project Timeline

- **2020**: Sigstore starts at Red Hat
- **2021**: Project launch with 3 major components: Rekor, Fulcio and Cosign
- **2022**: Sigstore joins the Linux Foundation
- **2023**: GA for Rekor and Fulcio public instances
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
- **fulcio**: Free Certificate Authority
- **cosign**: CLI to sign and verify artifacts

+ ecosystem-specific clients (Python, JavaScript, Rust...)
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
How does it work?

Sigstore’s “keyless” signing workflow
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

[Diagram of Sigstore's keyless signing workflow]

- Generate ephemeral key pair
- Identity proof request
- ID token (JWT)
- Signed Certificate Request + ID token
- Signed Certificate
- Fulcio (Certificate Authority)
- Certificate
- Certificate Transparency Log
- Verifier
- Log entry
- Request inclusion proof
- Verification materials
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
- Append-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/