The Sigsum transparency system

2026-04-24

Niels Möller

Slides: `https://git.glasklar.is/nisse/chains-2026`

Digital signatures everywhere

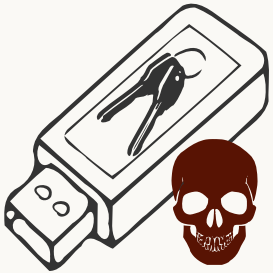
We can imagine supply-chain tooling enforcing:

- Signed source releases.
- Signed binaries and updates.
- Signed repro build claims.
- Signed attestations?

Digital signatures everywhere

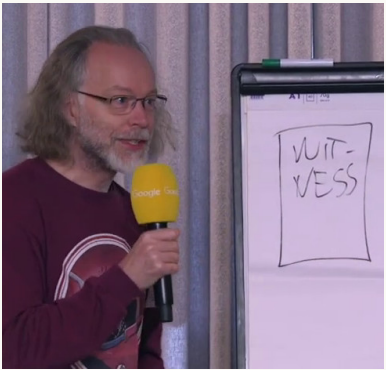
We can imagine supply-chain tooling enforcing:

- Signed source releases.
- Signed binaries and updates.
- Signed repro build claims.
- Signed attestations?



Would you know if a signing key is compromised?

Outline



Transparency logs and Sigsum

What's a transparency log?

- Append-only log.
- Merkle tree inside.

Transparency logs and Sigsum

What's a transparency log?

- Append-only log.
- Merkle tree inside.

Matthew Paul Argall CC-BY
www.flickr.com/photos/79157069@N03/24801150646

Transparency logs and Sigsum

What's a transparency log?

- Append-only log.
- Merkle tree inside.

What's Sigsum?

- Minimalistic transparency log.
- A FOSS project started ca 2020.
- Distributed (but not decentralized).
- Stable specs, pretty good tools.
  `https://www.sigsum.org/docs`

Matthew Paul Argall CC-BY
`www.flickr.com/photos/79157069@N03/24801150646`

Motivation: Rogue software update

Setting:

- Automatic software updates, properly signed.
- Attacker gains control of signing key.
- Attacker delivers rogue updates to targeted users.

Motivation: Rogue software update

Setting:

- Automatic software updates, properly signed.
- Attacker gains control of signing key.
- Attacker delivers rogue updates to targeted users.

Why transparency?

- Prevent? ❌
- Detect? ✅
- Recover? ⚠️

Threat model

The attacker owns:

- Release signing key.
- Distribution infrastructure.
- Sigsum log server.
- Some cosigning witnesses.

Threat model

The attacker owns:

- Release signing key.
- Distribution infrastructure.
- Sigsum log server.
- Some cosigning witnesses.
- Release build infrastructure?

Threat model

The attacker owns:

- Release signing key.
- Distribution infrastructure.
- Sigsum log server.
- Some cosigning witnesses.
- Release build infrastructure?

Aim: Detect attacks, deter attackers.

THE SIGSUM SYSTEM

Sigsum

- Log entries are signed checksums.

Sigsum

- Log entries are signed checksums.

A log entry

Checksum	What?	32
Pubkey hash	By whome?	32
Signature	Binding	64
		bytes

Sigsum

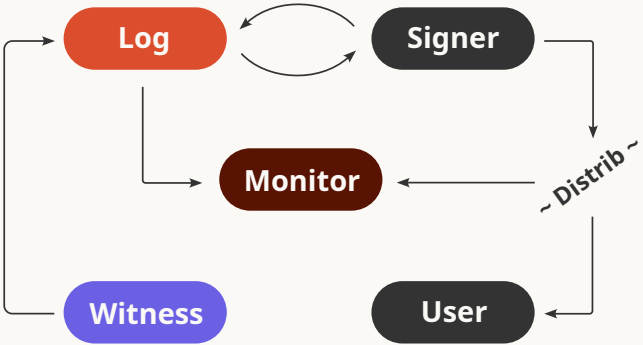
- Log entries are signed checksums.
- Designed for public usage.
- Building block for transparency.

A log entry

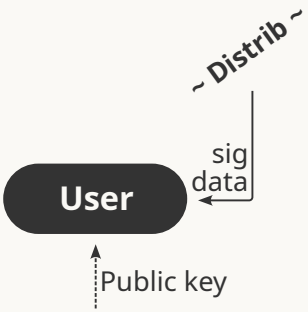
Checksum	What?	32
Pubkey hash	By whome?	32
Signature	Binding	64
		bytes

System overview

The entities in the Sigsum system:

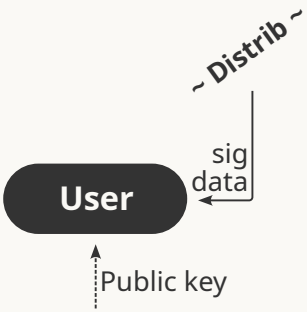


User perspective

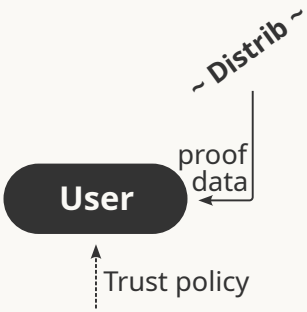


Without Sigsum

User perspective

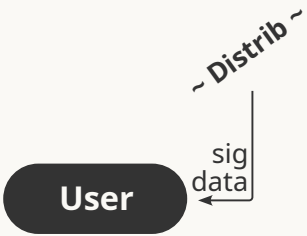


Without Sigsum

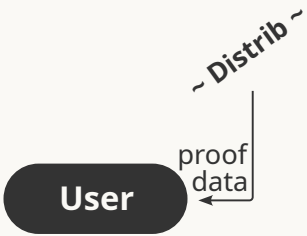


With Sigsum

User perspective



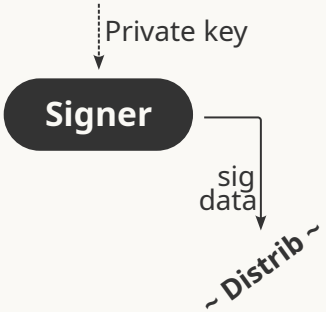
Without Sigsum



With Sigsum

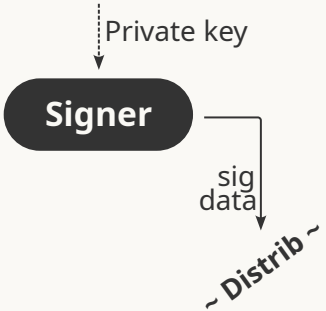
Offline verification.

Signer perspective

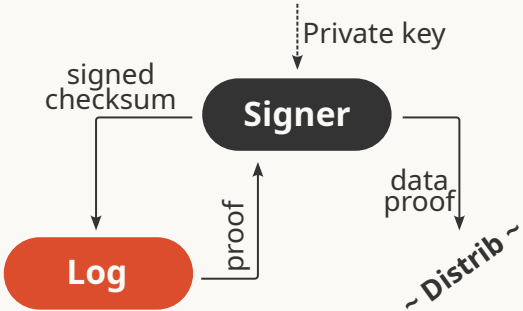


Without Sigsum

Signer perspective

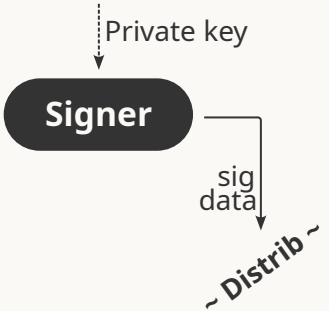


Without Sigsum

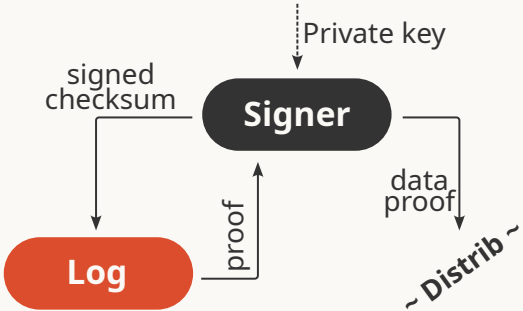


With Sigsum

Signer perspective



Without Sigsum



With Sigsum

Depends on the log being available at signing time.

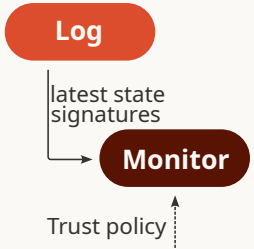
Monitor perspective

:(

Without Sigsum

Monitor perspective

:(

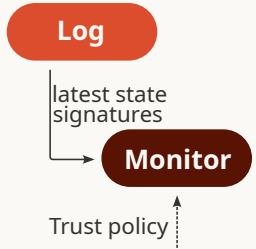


Without Sigsum

With Sigsum

Monitor perspective

:(



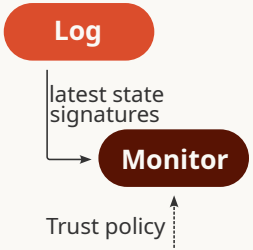
Without Sigsum

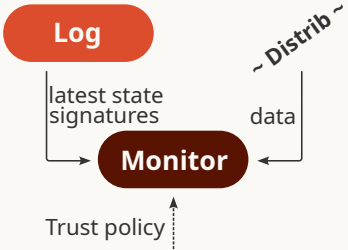
With Sigsum

All user-accepted signatures can be discovered.

Monitor perspective

:(





Without Sigsum | **With Sigsum** | **Also with Sigsum**

All user-accepted signatures can be discovered.

Split view?

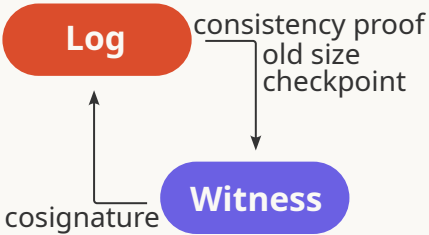
- Critical that monitor and user see the **same** log.
- What if the log looks different depending on who's asking?

Split view?

- Critical that monitor and user see the **same** log.
- What if the log looks different depending on who's asking?

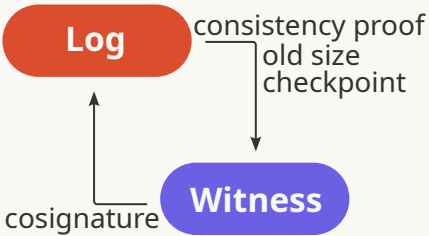
Don't want a single point of trust.

Enter witnesses



- Witness: “All older entries still there.”

Enter witnesses

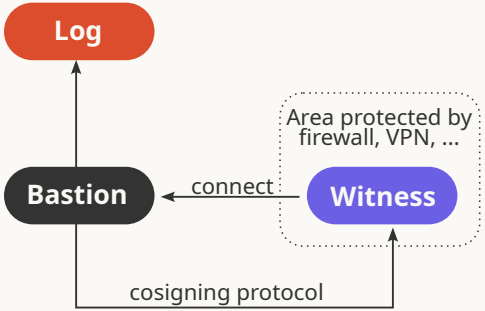


- Witness: "All older entries still there."
- Witness cosignatures are included in proofs.
- Require cosignatures from m of n witnesses.
- User and monitor agree on trusted witnesses.

Witnesses are light

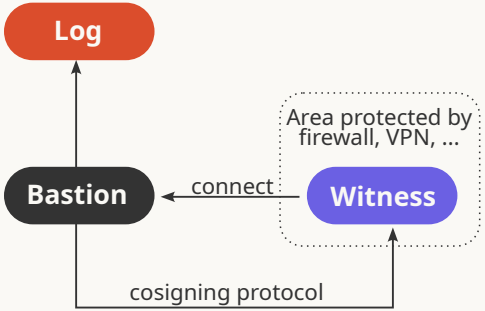
- Witnesses care about consistency, not contents.
- Modest storage, processing and bandwidth.
- Configuration via `https://witness-network.org/`

Witnesses are light



- Witnesses care about consistency, not contents.
- Modest storage, processing and bandwidth.
- Configuration via
  `https://witness-network.org/`
- Does not need any publicly known IP address.

Witnesses are light



- Witnesses care about consistency, not contents.
- Modest storage, processing and bandwidth.
- Configuration via `https://witness-network.org/`
- Does not need any publicly known IP address.

Does need secure and reliable operation.

Trust policy

The sigsum-generic-2025-1 policy:

```
log 0ec7e16843119b120377...1f25 https://seasalp.glasklar.is
log f00c159663d09bbda613...8314 https://ginkgo.tlog.mullvad.net

witness glasklar   b2106db9065ec97f25e0...8536
witness mullvad    15d6d0141543247b74ba...61ad
witness tillitis   076be8c9ee7ea60916f0...770c

group quorum-rule 2 glasklar mullvad tillitis
quorum quorum-rule
```

Trust policy

The sigsum-generic-2025-1 policy:

```
log 0ec7e16843119b120377...1f25 https://seasalp.glasklar.is
log f00c159663d09bbda613...8314 https://ginkgo.tlog.mullvad.net

witness glasklar   b2106db9065ec97f25e0...8536
witness mullvad    15d6d0141543247b74ba...61ad
witness tillitis   076be8c9ee7ea60916f0...770c

group quorum-rule 2 glasklar mullvad tillitis
quorum quorum-rule
```

Wanted: Additional independent witness operators.

CASE STUDY

Case study: Reproducible builds in System Transparency

Question: What software is running on that server?
Prototype: `https://git.glasklar.is/nisse/st-complete-poc`

Case study: Reproducible builds in System Transparency

Question: What software is running on that server?
Prototype: `https://git.glasklar.is/nisse/st-complete-poc`

System transparency combines:

- Transparency logging.
- Reproducible builds.
- Remote attestation.

Case study: Reproducible builds in System Transparency

Question: What software is running on that server?
Prototype: `https://git.glasklar.is/nisse/st-complete-poc`

System transparency combines:

- Transparency logging.
- Reproducible builds.
- Remote attestation.

Aim: Detect if server's supply-chain is compromised.

Remote attestation TL;DR

Components:

- Firmware (UEFI/BIOS/coreboot/...).
- Trusted Platform Module (TPM).
- Client/Server protocol.

Remote attestation TL;DR

Components:

- Firmware (UEFI/BIOS/coreboot/...).
- Trusted Platform Module (TPM).
- Client/Server protocol.

Output: Hash of software loaded at boot, signed by the TPM.

Remote attestation TL;DR

Components:

- Firmware (UEFI/BIOS/coreboot/...).
- Trusted Platform Module (TPM).
- Client/Server protocol.

Output: Hash of software loaded at boot, signed by the TPM.

```
PCR#4=43929c6dacb28003d46aa24d45cf73c7d4000a33fc08903bb4cf26602dc85a9d
```

Now what?

A repro build claim

```
$ GET https://tee.sigsum.org/~nisse/pcr/43/43...9d

claim-type=st-poc-pcr-claim-v0
pcr-index=4
pcr-sha256=43929c6dacb28003d46a...5a9d
commit=c07bcf80bde72d6627007ae9f934b6a77c2b8073

version=2
log=4e89cc51651f0d95f3c6...bb4d
leaf=ddce2439d2b04384adc5...13ea 3b22691a57d125d67126...770d

size=4371
root_hash=e8cb842eebcd5c671838...3e42
...
```

A repro build claim

```
$ GET https://tee.sigsum.org/~nisse/pcr/43/43...9d

claim-type=st-poc-pcr-claim-v0
pcr-index=4
pcr-sha256=43929c6dacb28003d46a...5a9d
commit=c07bcf80bde72d6627007ae9f934b6a77c2b8073

version=2
log=4e89cc51651f0d95f3c6...bb4d
leaf=ddce2439d2b04384adc5...13ea 3b22691a57d125d67126...770d

size=4371
root_hash=e8cb842eebcd5c671838...3e42
...
```

A Sigsum logged claim that the PCR#4 is valid and reproducible from source.

Monitoring

Work for a repro build monitor:

- Retrieve all log entries, filter by operator's key hash.
- Use checksum to retrieve corresponding claim from archive.

Monitoring

Work for a repro build monitor:

- Retrieve all log entries, filter by operator's key hash.
- Use checksum to retrieve corresponding claim from archive.
- Checkout corresponding git revision.
- Build the bootloader image.
- Compute expected hash value, check that it matches claim.
- Raise an alarm if any of these steps fail.

Monitoring

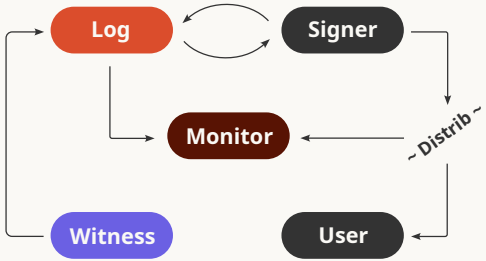
Work for a repro build monitor:

- Retrieve all log entries, filter by operator's key hash.
- Use checksum to retrieve corresponding claim from archive.
- Checkout corresponding git revision.
- Build the bootloader image.
- Compute expected hash value, check that it matches claim.
- Raise an alarm if any of these steps fail.

Sigsum does not prevent false claims, but enables detection.

SUMMARY

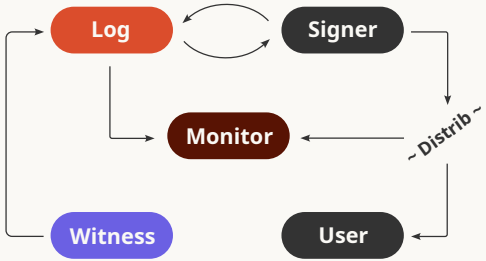
What's Sigsum?



Abstractly:

- Public record of signatures.
- Signatures have meaning to applications.
- Trusted when cosigned by witnesses.

What's Sigsum?



Abstractly:

- Public record of signatures.
- Signatures have meaning to applications.
- Trusted when cosigned by witnesses.

Concretely:

- Submit and collect proof when signing.
- Bring your own distribution.
- Offline verification.
- Monitor logs, for keys of interest.

What are your Sigsum applications?

Basic key usage transparency:

- Log all signatures.
- Enforce by verifiers.
- Key owner monitors log for unexpected signatures and key misuse.

What are your Sigsum applications?

Basic key usage transparency:

- Log all signatures.
- Enforce by verifiers.
- Key owner monitors log for unexpected signatures and key misuse.

Transparent software distribution:

- Alarm if a signed release is unavailable or unannounced.
- Enable third-party monitoring, no hidden releases.

What are your Sigsum applications?

Basic key usage transparency:

- Log all signatures.
- Enforce by verifiers.
- Key owner monitors log for unexpected signatures and key misuse.

Transparent software distribution:

- Alarm if a signed release is unavailable or unannounced.
- Enable third-party monitoring, no hidden releases.

Reproducible builds:

- Falsifiable claims on supply-chain for distributed binaries.
- Rebuilders as independent log monitors.

What are your Sigsum applications?

Basic key usage transparency:

- Log all signatures.
- Enforce by verifiers.
- Key owner monitors log for unexpected signatures and key misuse.

Transparent software distribution:

- Alarm if a signed release is unavailable or unannounced.
- Enable third-party monitoring, no hidden releases.

Reproducible builds:

- Falsifiable claims on supply-chain for distributed binaries.
- Rebuilders as independent log monitors.

Do you want to be a witness operator?