Nick Clark · Published April 25, 2026 · PDF

Sigstore (cosign / Rekor) alternative: enforcing signed policy before an autonomous agent acts

Sigstore, through cosign and the Rekor transparency log, lets teams sign software artifacts and record those signatures in a public, tamper-evident ledger so consumers can verify provenance before they trust a build. That solves supply-chain authenticity, but it does not gate what an autonomous agent is permitted to do at the moment it tries to execute, mutate, delegate, or propagate. This comparison is built on the Cryptographic Governance inventive step, disclosed in United States Patent Application 19/561,229, which turns signed, externally governed policy objects into a deterministic precondition evaluated per action before any execution context is instantiated.

What Sigstore (cosign / Rekor) Does

Sigstore is a widely adopted open-source project for signing, verifying, and proving the provenance of software artifacts. Its cosign component signs container images, binaries, and other blobs, and supports keyless signing in which short-lived certificates are issued against an OpenID Connect identity, removing the burden of managing long-lived signing keys. Its Rekor component is an append-only transparency log that records signing events so that anyone can later verify that a given artifact was signed by a given identity at a given time, and can obtain a cryptographic proof of inclusion in the log.

Sigstore does this job well. It has become a de facto backbone of modern software supply-chain security, is integrated into major registries and CI systems, and gives verifiers a credible, publicly auditable answer to the question, was this artifact produced by who it claims and has the record been tampered with. Keyless signing meaningfully lowered the operational cost of adopting artifact signatures at scale, and the transparency-log model provides strong tamper-evidence for the signing history itself.

The important thing to understand about Sigstore is its scope. It attests to the authenticity and provenance of an artifact, and it makes the signing record auditable. It answers a question about the past state of a thing, established at build or publish time.

The Architectural Axis

The axis this comparison addresses is when and against what a cryptographic check governs behavior. Sigstore verifies an artifact against a signature and a transparency record. That verification is typically performed at admission or deployment time, and it concerns identity and integrity of the artifact itself.

Autonomous and semi-autonomous agents introduce a different problem. An agent that has already been admitted may later propose to execute a task, mutate its own state, delegate to another agent, or propagate a derivative of itself into a new environment. Each of those is a distinct action taken after admission, and each may need to be permitted or refused according to policy that can change, expire, or be revoked over time. A signed-and-admitted artifact says nothing about whether a specific action proposed later is allowed under current constraints. This is a difference in what each system is built to govern, not a defect in Sigstore. Verifying the origin of a build and authorizing an individual runtime action are simply different jobs on different timelines.

How the Disclosed Approach Differs

The Cryptographic Governance inventive step disclosed in United States Patent Application 19/561,229 treats governance as a deterministic cryptographic precondition to each governed action rather than as a one-time check on an artifact. In the disclosed architecture, an agent object carries a policy reference field holding one or more canonical aliases. These aliases do not embed policy content; they are stable references dereferenced at runtime to obtain externally maintained, immutable-by-default policy objects. Because authority lives outside the agent and is immutable absent authorized succession, the agent cannot weaken, reinterpret, or silently downgrade its own constraints through self-modification, replication, or migration.

When an agent proposes an action, the disclosure routes it through a governance gate before any execution context is instantiated. The gate resolves the referenced aliases, filters candidate policy objects on freshness constraints including validity windows, revocation state, and anti-rollback monotonicity, and cryptographically verifies the authenticity of the remaining policy object. Only if the verified policy authorizes the specific proposed action class, under declared scope, validity, and freshness, is performance permitted. Otherwise the action is deterministically denied, and non-execution is treated as a valid, first-class system outcome rather than an error. Crucially, the proposed action itself may be execution, mutation, delegation, or propagation, so the same gate governs an agent trying to spawn a derivative or migrate to a laxer environment, not merely its initial launch.

Several mechanisms in the disclosure address the freshness and continuity gaps that a purely artifact-oriented check does not target. Anti-rollback is enforced through monotonic version indicators and a latest-known-good checkpoint recorded in embedded memory or an append-only audit record, so an older or superseded policy is rejected even under caching or intermittent connectivity. Governance can be changed only by publishing a successor or override policy object; the disclosure describes a quorum-based override in which a plurality of authorized participants co-sign a

replacement policy object, and a continuity reference, such as a signature chain or hash commitment, links the override to the prior authoritative instance so the chain of authority remains verifiable.

The disclosure also includes its own append-only audit ledger whose entries are cryptographically linked into an integrity chain, rendering removal, modification, or reordering detectable, and it can answer audit queries with inclusion proofs, ordering proofs, and integrity-chain validation artifacts without modifying the log. This is architecturally comparable in spirit to a transparency log, but it records governance events, policy resolutions, verification outcomes, authorization decisions, denials, override approvals, and freshness failures, rather than artifact signing events.

Where They Fit Together

These systems compose more naturally than they compete. Sigstore is the right tool for establishing that an artifact, including the agent code or a policy object bundle itself, was produced by a trusted identity and has not been altered, and for giving verifiers a publicly auditable signing record. The disclosed governance layer is the right tool for deciding, per action and at runtime, whether an already-admitted agent may proceed under current, externally governed, freshness-checked policy.

A plausible combined deployment would use Sigstore to sign and transparency-log the artifacts and policy-object payloads, and the disclosed governance gate to enforce, at each proposed execution, mutation, delegation, or propagation, whether that action is authorized under a freshly resolved and verified policy object. The signing step answers provenance; the gate answers present authorization. Neither replaces the other.

Boundary Conditions

Several honest limits apply. The disclosed subject matter is an early-stage patent application, not a shipped product with independent benchmarks; claims here about what it does trace to the specification of United States Patent Application 19/561,229 and describe disclosed mechanisms and embodiments, not measured performance. The governance model depends on agents being expressed as structured objects carrying resolvable policy references and governance-relevant state, and on a resolution substrate being reachable or on cached authority being validated under freshness and anti-rollback rules; environments that cannot meet those preconditions fall outside the described enforcement.

On the Sigstore side, the observations here are limited to its well-known, architecture-level purpose as an artifact signing and transparency-log system. Sigstore is not designed as a per-action runtime authorization gate for autonomous agents, and describing that scope boundary is not a criticism of a tool that does its intended job well. Where a deployment already relies on Sigstore for provenance, nothing in this comparison suggests replacing it.

Disclosure Scope

The invention described here is disclosed in United States Patent Application 19/561,229, and statements about what the disclosed approach does are grounded in that specification. References to Sigstore, cosign, Rekor, and the software supply-chain signing market are provided solely as external context to locate the disclosed subject matter along one architectural axis; they are not characterizations, claims, or elements of the filing, and nothing here asserts that Sigstore or its maintainers have any defect, that Sigstore fails to perform its intended function, or that the disclosed approach and Sigstore address the same problem. Any comparison is limited to the structural

difference between artifact-level signing and provenance and per-action, pre-execution policy enforcement for autonomous agents, and should not be read as legal advice or as defining the scope of any claim.

Cryptographic Governance (/cryptographic-governance)

All 40 steps → (/inventive-steps)

Policy that binds cryptographically — not by convention.

U.S. 19/561,229 (/patents/19-561229)

PRIMARY TECHNICAL DISCLOSURE

- Ethical Enforcement as Infrastructure: Cryptographic Governance for Autonomous Systems (/articles/ethical-enforcement-as-infrastructure-cryptographic-governance-for-autonomous-systems)

SECONDARY TECHNICAL

- Governance Gate as Deterministic Precondition: No Verification, No Execution (/articles/cryptographic-governance/governance-gate)
- Canonical Alias to External Policy Indirection: Policy Evolution Without Agent Mutation (/articles/cryptographic-governance/policy-indirection)
- Immutable-by-Default Policy Objects: Governance Changes Through Successor Issuance (/articles/cryptographic-governance/immutable-policies)
- Runtime Policy Resolution Pipeline: Mandatory Verification Before Every Execution (/articles/cryptographic-governance/policy-resolution)
- Freshness, Revocation, and Anti-Rollback Controls: Preventing Stale Authority (/articles/cryptographic-governance/freshness-revocation)
- Memory-Derived Eligibility Conditioning: Past Violations Constrain Future Authorization (/articles/cryptographic-governance/memory-eligibility)
- Intent-Independent Authorization: Governance Without Alignment Scoring (/articles/cryptographic-governance/intent-independent-auth)
- Execution Feedback as Enforcement Signals: Operational Outcomes Shaping Future Authorization (/articles/cryptographic-governance/enforcement-feedback)
- Trust Degradation as State Transition: Policy-Defined Narrowing of Permitted Actions (/articles/cryptographic-governance/trust-degradation)

- Structural Quarantine: Execution Prevention Until Authorized Remediation (/articles/cryptographic-governance/structural-quarantine)
- Lineage-Constrained Governance Inheritance: Constraints That Persist Across Generations (/articles/cryptographic-governance/governance-inheritance)
- Unauthorized Fork Prevention: Lineage Continuity as Anti-Cloning Mechanism (/articles/cryptographic-governance/fork-prevention)
- Meta-Policy Objects: Higher-Order Constraints Across System Behavior Categories (/articles/cryptographic-governance/meta-policy)
- Quorum-Based Governance Override: Multi-Party Approval With Signature-Chain Continuity (/articles/cryptographic-governance/quorum-override)
- Distributed Alias Publication: Policy Dissemination Through Federated Registries (/articles/cryptographic-governance/alias-publication)
- Fallback Enforcement Agents: Distributed Monitors as Defense-in-Depth (/articles/cryptographic-governance/fallback-enforcement)
- Append-Only Governance Audit Ledger: Tamper-Evident Records of Every Authorization (/articles/cryptographic-governance/audit-ledger)
- Governance Without Persistent Keypairs: Trust-Slope Authorization Replacing Static Keys (/articles/cryptographic-governance/keyless-governance)
- Execution Eligibility Indicator: Dynamic Computation From Policy, Memory, and Lineage (/articles/cryptographic-governance/eligibility-indicator)
- Cross-Domain Spatial-Temporal Escalation (/articles/cryptographic-governance/cross-domain-spatial-temporal-escalation)
- Cross-Authority Handoff Governance (/articles/cryptographic-governance/cross-authority-handoff-governance)
- The Guardrail an Agent Can't Remove: Gating an Agent's Mutation of Its Own Policy, Role, Memory, and Lineage (/articles/cryptographic-governance/self-modification-governance)

APPLICATIONS · GENERAL

- Cryptographically Enforced Governance for SCADA and OT: Gating Autonomous Control Actions in Power, Water, and Industrial Control Systems (/articles/cryptographic-governance/critical-infrastructure-ics)
- How to Make High-Risk AI Agents EU AI Act Compliant by Architecture (/articles/cryptographic-governance/eu-ai-compliance)
- Self-Verifying Financial Audit Trails Without Trusted Intermediaries (/articles/cryptographic-governance/financial-audit-trails)
- Enforcing HIPAA at Every Data Operation: Structural Healthcare Compliance (/articles/cryptographic-governance/healthcare-compliance)

- Preventing Classified Data Spillage: Cryptographic Classification Enforcement for Defense (/articles/cryptographic-governance/defense-classification)
- Tamper-Evident Environmental Monitoring: Cryptographic Governance for Emissions and Compliance Data (/articles/cryptographic-governance/environmental-monitoring)
- Pharmaceutical Supply Chain Governance: DSCSA, FMD, and Cold-Chain Compliance Bound to the Product (/articles/cryptographic-governance/pharmaceutical-supply)
- Cryptographic Governance for Nuclear Facility Operations: Structural Enforcement of Technical Specifications (/articles/cryptographic-governance/nuclear-facility-governance)
- Preventing CSAM Distribution at the Source: Cryptographic Governance for Child Safety Content Enforcement (/articles/cryptographic-governance/child-safety-enforcement)
- Coalition Policy Distribution Without Shared Authority (/articles/cryptographic-governance/coalition-policy-distribution)
- EU AI Act Recital 73 and Article 14: How to Build AI That Cannot Disable Its Own Oversight (/articles/cryptographic-governance/eu-ai-act-self-constraint)
- Enforcing Build Provenance Before Artifacts Ship: Cryptographic Governance for Software Supply-Chain Integrity (/articles/cryptographic-governance/software-supply-chain-provenance)

APPLICATIONS · SPECIFIC

- HashiCorp Vault Alternative for Governed Agent Execution: Binding Policy to Action (/articles/cryptographic-governance/hashicorp-vault)
- AWS KMS Manages Encryption Keys. The Keys Do Not Carry Governance. (/articles/cryptographic-governance/aws-kms)
- Open Policy Agent Decoupled Policy From Code. The Policy Is Not Cryptographically Bound. (/articles/cryptographic-governance/open-policy-agent)
- Styra vs Cryptographically Governed Agent Execution: Beyond Advisory Policy (/articles/cryptographic-governance/styra)
- Snyk vs Cryptographic Governance: Vulnerability Scanning Is Not Runtime Enforcement (/articles/cryptographic-governance/snyk)
- Palo Alto Networks Inspects Traffic. It Does Not Govern the Operations That Generate It. (/articles/cryptographic-governance/palo-alto)
- SPIFFE/SPIRE vs Governed Agent Execution: Workload Identity Without a Cryptographic Policy Binding (/articles/cryptographic-governance/spiffe-spire)
- cert-manager vs Cryptographic Governance: Certificates Authenticate Identity, They Do Not Gate Execution (/articles/cryptographic-governance/cert-manager)
- Keycloak vs Cryptographically Governed Agent Execution: Beyond Identity Tokens (/articles/cryptographic-governance/keycloak)

- HashiCorp Boundary Alternative for Governed Session Operations: Zero-Trust Access vs Cryptographic Governance (/articles/cryptographic-governance/boundary)
- Teleport Alternative for Governed Operations: Access Control Is Not Cryptographic Governance (/articles/cryptographic-governance/teleport)
- BeyondTrust vs Cryptographic Governance: PAM Manages Privilege, It Does Not Bind Operations to Signed Policy (/articles/cryptographic-governance/beyondtrust)
- CyberArk vs Cryptographically Governed Agent Execution: PAM Protects the Credential, Not the Operation (/articles/cryptographic-governance/cyberark)
- 1Password vs Cryptographically Governed Agent Execution: Credential Custody Is Not Bound Governance (/articles/cryptographic-governance/1password)
- The Update Framework (TUF) / Notary alternative: signing software artifacts vs governing what an agent may do at runtime (/articles/cryptographic-governance/tuf-notary)
- **Sigstore (cosign / Rekor) alternative: enforcing signed policy before an autonomous agent acts (/articles/cryptographic-governance/sigstore)**

Cryptographic Governance overview → (/cryptographic-governance)