Nick Clark · Published April 25, 2026 · PDF

Enforcing Build Provenance Before Artifacts Ship: Cryptographic Governance for Software Supply-Chain Integrity

Software supply-chain attacks succeed because build systems, package registries, and deployment pipelines trust artifacts whose provenance is checked late, inconsistently, or not at all across the many substrates a build traverses. Attestations get logged after promotion, policy lives in mutable CI configuration an attacker can edit, and stale allow-lists linger long after a signing key is rotated or revoked. This application is built on Cryptographic Governance, disclosed in United States Patent Application 19/561,229, which makes authorization a signed, externally governed, pre-execution precondition for every governed action rather than an after-the-fact audit.

What This Application Specifies

The application specifies a way to make authorization a deterministic cryptographic precondition to a governed action, evaluated before that action is allowed to take effect. In the terms of the disclosure, a proposed action is submitted to a governance gate that resolves one or more policy references (canonical aliases) to external policy objects, filters candidates against freshness constraints (a validity window, a revocation state, and an anti-rollback monotonicity constraint), cryptographically verifies the surviving policy object, and determines whether the action is authorized under that verified

authority. The action proceeds only if authorized; otherwise it is deterministically denied, and non-execution is treated as a valid, first-class system outcome rather than an error to be worked around.

Applied to a software supply chain, the "governed action" is any step that would otherwise commit an artifact forward: admitting a dependency into a build, signing or promoting a built artifact, publishing to a registry, or deploying an image into a runtime. The "policy object" is a signed, immutable authority that declares which provenance conditions must hold. Because the policy is referenced by a stable alias and resolved at runtime rather than embedded in the pipeline configuration, the same authority governs the build controller, the registry admission check, and the deployment gate without copying rules into each place where they could drift or be edited.

Several disclosed properties matter directly here. Policy content is immutable by default; governance changes happen by publishing a successor under the same alias, not by mutating rules in place. Freshness is enforced at authorization time, so an expired, revoked, or superseded provenance policy is rejected even when a cached copy is still present. Anti-rollback is enforced through monotonic version indicators and rejection of non-current authoritative instances, which structurally resists downgrade to a weaker prior policy. And every resolution, verification, permit, denial, and freshness determination can be written to an append-only, integrity-chained audit ledger that answers inclusion and ordering queries with cryptographic proofs.

Why It Matters

Modern software is assembled, not written. A single release pulls transitive dependencies from public registries, runs them through build and test stages on heterogeneous infrastructure, and lands in production through a chain of automated promotions. The industry response has converged on provenance: standards such as SLSA describe build-provenance levels, in-toto describes signed attestations linking a

subject artifact to the steps that produced it, and Sigstore-style signing describes how to attach verifiable signatures. Regulators and buyers increasingly expect a software bill of materials, reflected in guidance such as United States Executive Order 14028 and the associated NIST Secure Software Development Framework (SP 800-218).

The gap is not the existence of attestations; it is where and when they are checked. Provenance data that is generated but verified late, or verified in one stage but not another, does not prevent a compromised artifact from advancing. Policy that lives in mutable pipeline configuration can be edited by whoever can edit the pipeline, which is exactly the surface a supply-chain attacker targets. Allow-lists that are not freshness-aware keep honoring a signing identity after its key should have been retired. The disclosed architecture speaks to each of these: authorization is evaluated before instantiation of the next step, the governing authority is external and immutable absent authorized succession, and freshness and revocation are checked at the moment of the decision.

How It Composes With the Domain

Map the disclosure onto a pipeline. Each artifact-advancing step carries, or is evaluated against, one or more canonical policy aliases such as a provenance-admission alias and a signing-authority alias. When a step is proposed, the governance gate issues a resolution request for those aliases. Resolution can be scope-aware, so a production-deployment substrate can be routed to a stricter authoritative instance than a development substrate, and the resolver returns candidate policy objects together with the provenance sufficient to verify them.

The gate then applies the disclosed freshness filtering before verification: candidates outside their validity window, marked revoked, or failing the anti-rollback monotonicity constraint are discarded. Surviving policy is verified cryptographically. The policy body declares the permitted and prohibited conditions in domain terms, for example that a build attestation must be present and signed by an authority within the current

authorized set, that the declared build steps match, and that no dependency carries a revoked provenance reference. Only if the proposed step satisfies the verified, applicable policy does the gate permit it to proceed; otherwise it emits a deterministic denial, and no promotion, publish, or deploy is instantiated.

Two disclosed mechanisms extend this cleanly across a real pipeline. Lineage provides verifiable continuity: the disclosure treats a current state as authorized only when it is a valid successor of a prior authorized state, which maps to requiring that a deployed artifact trace back through signed build and promotion events rather than appearing from an untracked branch or a replayed snapshot. And quorum-based override lets a genuine governance change, such as rotating the authorized signing set or tightening an admission rule, take effect only when a plurality of authorized participants co-sign a successor policy object that carries a continuity reference (a signature chain or monotonic version link) to the instance it supersedes. Published under the same alias, that successor governs subsequent decisions without editing any pipeline.

Underneath, the append-only audit ledger records what authority was resolved, what verification occurred, and what outcome resulted for each governed step, linked into an integrity chain so that removal, modification, or reordering is detectable. Auditors and compliance systems can query it and receive inclusion and ordering proofs, which turns "show that this release was admitted only under a current, verified provenance policy" into a verifiable statement rather than a trust exercise.

What This Enables

Because authorization is a pre-execution gate rather than a post-hoc log, a build or deployment step that lacks verified provenance simply does not happen, and that non-execution is a recorded, intended outcome. Concretely, the architecture as disclosed enables:

- Uniform enforcement across substrates. The same externally governed policy governs the build controller, the registry admission check, and the deployment gate, because authority is resolved from a shared alias rather than duplicated into each stage's configuration.
- Revocation and rotation that take effect immediately at the decision point. Publishing a successor policy or marking a signing authority revoked causes subsequent authorizations to reject the stale authority even where a cached copy remains, per the disclosed freshness and anti-rollback handling.
- Downgrade resistance. Monotonic versioning and rejection of superseded instances structurally resist an attacker who tries to reintroduce a weaker prior admission policy.
- Deliberate, multi-party governance change. Tightening or loosening provenance rules requires a co-signed successor with continuity linkage, so no single compromised account can silently weaken the gate.
- Provable compliance posture. The integrity-chained ledger supplies inclusion and ordering proofs about which policy governed each release, supporting attestation-heavy regimes such as SLSA-aligned provenance and SBOM expectations.

Boundary Conditions

The application does not describe a scanner, a vulnerability database, or a way to decide whether a dependency is "safe." It enforces whether a proposed step is authorized under verified policy; the substance of provenance conditions is expressed in the policy body by whoever governs it. Its guarantees are precondition guarantees: a step is not instantiated absent verified authorization, which is distinct from proving that authorized code is free of defects.

The architecture also assumes that the resolution and verification path is present in the pipeline. The disclosure is explicit that a governance gate can be implemented within a substrate, as middleware, or as a distributed validation function, and that fallback

enforcement agents provide cross-substrate consistency checking as defense in depth rather than as the primary path. A stage that never routes proposed actions through a gate is outside the enforced boundary. Trust in signing keys, quorum participants, and the resolver's provenance is assumed, not created, by the mechanism; the disclosure supports both public-key verification and continuity-based identity, but the choice and custody of those trust roots remain the deployer's responsibility. The performance and operational characteristics of a specific pipeline are not claimed here, and this article introduces none.

Disclosure Scope

The invention is disclosed in United States Patent Application 19/561,229, "Cryptographically Enforced Governance for Autonomous Agents and Distributed Execution Environments." The claims about what the invention does, namely pre-execution resolution of externally governed policy references, cryptographic verification, freshness, revocation, and anti-rollback filtering, deterministic permit-or-deny with non-execution as a valid outcome, quorum-based override with continuity linkage, and an append-only integrity-chained audit ledger with inclusion proofs, trace to that disclosure. The software supply-chain framing here, including references to SLSA, in-toto, Sigstore-style signing, SBOM practice, United States Executive Order 14028, and NIST SP 800-218, is external domain and regulatory context provided to describe one faithful, enabling implementation; those standards and regulations are not part of the disclosure and are named only as accurate real-world context. Nothing here should be read as a benchmark, a performance representation, or an assertion of compliance with any particular standard.

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)