Nick Clark · Published April 25, 2026 · PDF

The Update Framework (TUF) / Notary alternative: signing software artifacts vs governing what an agent may do at runtime

The Update Framework (TUF) and the Notary Project (Notation) secure the software supply chain by cryptographically signing artifacts and metadata so that a client can verify what it is about to download and install. That is essential, but it answers a different question than whether a running autonomous agent may execute, mutate, delegate, or propagate a given action right now. This article is built on the Cryptographic Governance, disclosed in United States Patent Application 19/561,229, which enforces signed, externally governed policy as a deterministic precondition to each governed action.

What The Update Framework (TUF) / Notary Does

The Update Framework (TUF) is a specification, originally developed in academic research and now a Cloud Native Computing Foundation project, for securing software update systems against a broad class of repository and network compromises. TUF organizes trust around a small set of signed metadata roles. A root role acts as the trust anchor and signs the public keys of the other roles. A targets role signs metadata describing the artifacts that clients may trust, and can delegate that authority to more specific roles. A snapshot role signs a consistent view of all targets metadata, and a timestamp role is signed frequently and given a short expiration so that clients can

detect when they are being served stale metadata. Through this structure, TUF provides meaningful properties: resistance to rollback and freeze attacks, survivability of individual key compromise, and delegation of signing authority. Longer-lived root and targets metadata typically expire on the order of a year, while frequently changing snapshot and timestamp metadata expire much sooner.

The Notary Project, whose client tool is Notation, is a related CNCF effort focused on signing and verifying artifacts stored in OCI-conformant registries, such as container images. Notation stores signatures as artifacts attached to the signed image in the registry, and is designed to integrate with enterprise X.509 public-key infrastructure and existing certificate workflows. It lets an operator establish that a container image was signed by a trusted identity before that image is admitted.

Both projects are mature, well-reasoned, and widely deployed. They do exactly what they set out to do: give a consuming system cryptographic assurance about the provenance and integrity of a distributable artifact and its metadata before that artifact is retrieved or installed.

The Architectural Axis

The relevant architectural axis is the object and moment that verification governs. TUF and Notation verify artifacts and their metadata at distribution and admission time. The question they answer is: is this file, image, or metadata bundle authentic, current, and signed by an authority I trust, so that I can safely fetch or install it? This is a question about what enters a system.

There is a distinct question that arises once the artifact is running and is itself an autonomous or semi-autonomous actor: is this specific proposed action, by this specific already-admitted agent, permitted right now under the current external policy? A signed and verified container image is a trusted starting point, but the agent inside it may go on to propose many actions over its lifetime, and it may mutate its own state,

spawn derivatives, or migrate to another environment. Artifact signing establishes trust in the thing that was shipped. It does not, by design, adjudicate each governed behavior the thing performs after it is admitted. That is not a defect in TUF or Notation; it is simply outside the scope of supply-chain integrity, which is where those projects deliberately draw their boundary.

How the Disclosed Approach Differs

The disclosed approach operates on that second axis: it treats execution, mutation, delegation, and propagation as governed actions, each of which must be authorized as a deterministic cryptographic precondition before an execution context is instantiated. As described in the specification, an agent object carries a policy reference field containing canonical aliases rather than embedded policy logic. When the agent proposes an action, a governance gate resolves those aliases through a dynamic alias system to obtain external policy objects, verifies their authenticity under an applicable trust model, and permits the action only if the verified authority authorizes that action class. If authorization is absent, the action is deterministically denied and non-execution is returned as a valid system outcome rather than an error to be worked around.

Several mechanisms in the specification address the runtime axis directly. Policy objects are externally maintained and immutable by default, so governance changes occur through issuance of a successor or override policy object under the same canonical alias rather than by in-place modification, which means an agent cannot weaken its own constraints through self-modification or replication. The specification describes freshness, revocation, and anti-rollback controls, including validity windows, revocation state, monotonic version indicators, and rejection of non-current authoritative instances, so that expired or superseded authority is refused even under caching and intermittent connectivity. It describes quorum-based override, in which a plurality of authorized participants co-sign a replacement policy object carrying a continuity reference to the superseded object, and the override is authoritative only if that quorum and signature-chain continuity verify at runtime. It describes lineage as a

verifiable continuity mechanism, so that descendants inherit governance constraints and unauthorized forks are denied. And it describes an append-only audit ledger whose entries are cryptographically linked into an integrity chain and can answer audit queries with inclusion, ordering, and integrity proofs.

The structural difference, scoped to this axis, is that verification here binds to each governed action of a running agent, keyed to externally governed policy that travels with the object across substrates, rather than to a distributable artifact at the point of download or admission.

Where They Fit Together

These are complementary layers, not substitutes. TUF and Notation are well suited to the supply-chain problem: ensuring that the agent image, the policy tooling, and the surrounding software were built and delivered by trusted parties and have not been tampered with in transit. The disclosed governance layer assumes that admitted software is trustworthy at rest and addresses what that software is subsequently allowed to do at runtime.

A layered deployment is natural. Notation or TUF can gate what images and metadata are admitted into an environment. The disclosed architecture can then gate each governed action those admitted agents propose, using externally governed policy objects that are themselves signed artifacts and could plausibly be distributed and integrity-protected using supply-chain tooling. Nothing about the runtime governance model displaces the value of verifying provenance; the two operate at different points in the lifecycle and reinforce each other.

Boundary Conditions

Honesty about scope matters. The disclosed subject matter is an early-stage patent application, not a released product with published benchmarks, and this article makes no performance claims for it. Its guarantees are architectural: they hold to the extent that policy objects can be resolved and verified, that the governance gate is actually in the execution path, and that the trust model behind verification is sound. In environments where an agent can reach a substrate that does not honor the gate, the deterministic precondition property depends on that substrate participating in enforcement, and the specification describes fallback enforcement agents as defense-in-depth rather than as a replacement for gating.

Conversely, TUF and Notation carry their own well-understood operational considerations, such as key management, root key ceremonies, and metadata expiration handling, which their communities document thoroughly. Those are inherent to any signing system and are not shortcomings unique to those projects. The comparison here is strictly about which axis each technology addresses, and neither axis makes the other unnecessary.

Disclosure Scope

The technology attributed to the disclosed approach in this article is described in United States Patent Application 19/561,229. Statements about our side trace to that disclosure; statements about The Update Framework, the Notary Project, and Notation reflect publicly documented, architecture-level facts about those projects and are provided solely as external context to locate the disclosed work within a landscape. Nothing here asserts or implies any defect, vulnerability, or deficiency in TUF, Notation, or any related project; those are capable, widely respected tools that address the software supply chain effectively. The market and comparison framing is context, not a claim of the filing, and the scope of any patent rights is defined by the claims of the application as ultimately allowed, not by this article.

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)