SYSTEM AND METHOD FOR CRYPTOGRAPHIC AUTHORITY PRESERVATION ACROSS DELEGATION CHAINS

FIELD OF INVENTION

[0001] The present disclosure relates to post-quantum cryptographic proof systems for distributed authority enforcement, and more particularly to systems and methods for preserving authorized human intent across delegation chains via hash-linked delegation graphs and multi-condition conformance gates.

BACKGROUND

[0002] Modern computing environments increasingly involve distributed execution chains spanning human actors, autonomous software agents, integrated systems, and cross-organizational workflows. In such environments, actions may be delegated through multiple levels of authority, from high-level organizational directives down to specific operational tasks performed by software agents or human workers.

[0003] Existing authorization systems address the question of whether an actor has permission to perform a particular action. Public Key Infrastructure (PKI) systems bind identity to cryptographic keys. Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) systems inherit permissions across organizational roles. Capability systems, OAuth protocols, SPKI, and macaroons delegate authority through attenuated tokens. Workflow engines and business process management systems record approval-step sequences and multi-step authorization chains. However, these existing systems preserve authorization—they answer the question “can you do this?”—but they do not preserve authorized intent—they do not answer the different question “are you still doing what the humans originally meant?” As execution chains become increasingly distributed across human actors, autonomous software agents, integrated systems, and cross-organizational boundaries, the second question increasingly determines the viability of regulated decisions: AI agents act on behalf of humans, regulators audit those actions, insurers underwrite them, and corporate governance bodies are accountable for them.

[0004] However, these existing systems exhibit several technical limitations when applied to distributed delegation chains of arbitrary depth. First, existing delegation mechanisms do not compose constraint sets recursively across arbitrary delegation depth via hash-linked ancestor chains. When authority passes through multiple levels of delegation, existing systems do not produce a cryptographically verifiable proof that the constraints imposed at each level

have been preserved and composed at every descendant level down to the point of action execution. As a result, verifying that a leaf-level action conforms to every ancestor's constraints requires access to each intermediate authorization system rather than verification against a single self-contained cryptographic proof.

[0005] Second, existing authorization systems do not produce independently verifiable cryptographic proof bundles that remain portable across workflow execution engines and verifiable without contact with the issuing system. When an authorization decision is made within a particular workflow engine, the verifiability of that decision is typically coupled to the continued operation of that engine. If the workflow engine is replaced, decommissioned, or migrated, the authorization artifacts produced under the prior engine may not be independently verifiable by a third party without access to the original system. Third, existing systems do not cryptographically bind authority artifacts to the contemporaneous state of external evidence and outcome substrates. Authority decisions are made against a particular system state, but existing authorization artifacts do not carry cryptographic commitments to the evidence and outcome records that existed at the time of authorization, making it difficult to verify after the fact that an authorization decision was consistent with the system state at the time it was made.

[0006] Fourth, existing authorization systems do not produce cryptographic proofs of denial. When an action is refused, existing systems may log the refusal, but they do not produce a portable, independently verifiable cryptographic artifact that identifies the specific condition that caused the denial. Fifth, authorization artifacts that rely on cryptographic algorithms vulnerable to quantum computing advances face potential exposure to future attacks. Authorization artifacts that may need to be verified years or decades after issuance require post-quantum cryptographic protection to maintain long-term verification integrity.

[0007] Accordingly, there exists a need for improved systems and methods for producing cryptographically verifiable, portable, and independently auditable proof bundles across distributed delegation chains using post-quantum cryptographic primitives.

SUMMARY

[0008] It is an object of the present invention to provide a cryptographic authority substrate that preserves authorized human intent across delegation chains of arbitrary depth spanning human actors, autonomous software agents, integrated systems, and cross-organizational workflows.

[0009] It is another object of the present invention to provide a root authority object representing threshold-signed human-authored objective, cryptographically bound by a post-quantum threshold signature meeting an approval set with threshold semantics.

[0010] It is another object of the present invention to provide a delegation graph engine that computes ancestor constraint aggregates for each authority node as a function of every ancestor node from an immediate parent up to and including the root authority object, the ancestor constraint aggregate comprising an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints.

[0011] It is another object of the present invention to provide a pre-execution conformance gate that enforces a four-condition test prior to action execution, comprising an Activated condition, a Current condition, a Within Window condition, and a Lifecycle Compliant condition.

[0012] It is another object of the present invention to provide a proof generation module that produces Portable Authority Proofs upon all conditions passing and Negative Authority Proofs upon any condition failing, each proof being independently verifiable by any third party without contact with the issuing system.

[0013] It is another object of the present invention to provide cross-system cryptographic witness commitments binding authority nodes to the contemporaneous state of evidence substrates and outcome substrates at the time of node issuance.

[0014] It is another object of the present invention to provide an intent-passage registry comprising canonical encodings of human-authored instruction passages, enabling intent citations that cryptographically bind actions to specific human-authored passages.

[0015] The present invention addresses the limitations of the prior art by providing a cryptographic authority preservation architecture that produces independently verifiable proof bundles via hash-linked delegation graphs and post-quantum threshold signatures, thereby achieving measurable technical improvements over prior art authorization systems. The present invention is structurally distinct from prior-art authorization systems in that it produces, simultaneously and cryptographically, three independent forms of inheritance unified by one

substrate: constraint inheritance wherein every node carries constraints from every ancestor, proof inheritance wherein every node produces a portable proof of conformance to those constraints, and intent inheritance wherein every node preserves attribution to the original human intent regardless of delegation depth. By composing constraint sets recursively across arbitrary delegation depth via hash-linked ancestor chains, the present invention enables verification that constraints imposed at each level have been preserved and composed at every descendant level down to the point of action execution. By producing portable cryptographic proof bundles that remain verifiable without contact with the issuing system, the present invention decouples authorization verifiability from the continued operation of any particular workflow engine. By cryptographically binding authority artifacts to the contemporaneous state of external evidence and outcome substrates, the present invention enables verification that authorization decisions were consistent with the system state at the time they were made. By producing cryptographic proofs of denial identifying specific failing conditions, the present invention provides portable evidence of governed refusal. The technical improvements provided by the present invention enable distributed delegation chains spanning human actors, autonomous software agents, integrated systems, and cross-organizational workflows to maintain cryptographic verifiability for the lifetime of the post-quantum signature algorithms through the three-way simultaneous inheritance of constraints, proofs, and intent.

[0016] In order to overcome the limitations of existing authorization systems, wherein delegation mechanisms do not compose constraint sets recursively across arbitrary delegation depth and authorization artifacts are coupled to issuing systems, the present invention provides a system for preserving authorized human intent across delegation chains. The system comprises a processor and a memory coupled to the processor and storing instructions that, when executed by the processor, cause the system to implement a root authority object store, a delegation graph engine, a pre-execution conformance gate, a proof generation module, and a registry complex. The root authority object store maintains threshold-signed human-authored objectives, each root authority object comprising an objective statement, a lifecycle mode indicator selected from PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules, cryptographically bound by a post-quantum threshold signature meeting the approval set.

[0017] The delegation graph engine computes, for each authority node in a delegation graph descended from the root authority object, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object,

the ancestor constraint aggregate comprising an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints. The pre-execution conformance gate enforces, prior to execution of an action by an acting principal, a four-condition test comprising: an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object; a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and a Lifecycle Compliant condition verifying that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp. The proof generation module produces, responsive to the four-condition test passing, a Portable Authority Proof comprising a lineage proof from the action to the root authority object, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a specific human-authored passage from an intent-passage registry, and a structured authority rationale record assembled programmatically from the citations. The proof generation module produces, responsive to any condition of the four-condition test failing, a Negative Authority Proof identifying a failing condition and a denial-category identifier.

[0018] The system may implement authority state portability across heterogeneous workflow execution engines. The authority state may be expressed as a portable serialized authority graph comprising the root authority object and a plurality of delegation nodes, the authority graph being independent of any specific workflow execution engine. The system may migrate the authority state from a first workflow execution engine to a second workflow execution engine by serializing the authority graph from the first workflow execution engine and de-serializing the authority graph into the second workflow execution engine without re-issuance of any signature in the authority graph. The system may further implement root succession by publishing a successor root authority object signed by a post-quantum threshold signature meeting a succession authority field of a predecessor root authority object, binding the successor root authority object to the predecessor root authority object by a signed succession event placed in an append-only succession registry, and maintaining verifiability of authority decisions taken under the predecessor root authority object indefinitely after succession. The system may further implement conflict detection and resolution by detecting that two or more contributing root authority objects reach divergent verdicts on a proposed

action, emitting a conflict detection record, receiving a Conflict Resolution Record, and binding Conflict Precedents to a per-root append-only precedent registry for application to future conflicts. The system may further implement override authorization by receiving an override authorization record from a higher-order authority principal and emitting an Override Portable Authority Proof comprising an indication of what would have been denied, an indication of what was authorized, an identification of the higher-order authority principal, an expiration condition, and a signed trail of an override event.

[0019] The system may further implement historical authority verification via vintage binding. Authority-derived artifacts emitted under a specific root vintage may remain independently verifiable indefinitely after that vintage has been rotated, recovered, retired, or revoked. Each authority-derived artifact may carry a Vintage Binding comprising a root succession identifier, a root vintage identifier, an issuance timestamp, an active transition hash, and a predecessor transition hash. A Vintage Binding verifier may prove that the named vintage existed in the named succession, that the vintage was active at the artifact's issuance timestamp, and that the active-transition and predecessor-transition hashes match the immutable vintage record. The Vintage Binding verifier may not consult mutable fields such as vintage state, vintage successor, or predecessor state for the validity decision. The operational property is that a prior vintage can verify the past but cannot authorize the future, while a current vintage authorizes the future.

[0020] The system may further implement authority transfer with predecessor-delegation lineage closure. An Authority Transfer Record may represent a transfer of an active delegated authority window from one principal to another, distinct from root succession wherein the authority source changes and from principal membership transition wherein the role holder changes. The Authority Transfer Record may comprise a transferring principal identifier, a receiving principal identifier, a predecessor delegation hash identifying the delegation event that established the transferred window, a transfer window that is monotonically no wider than the underlying delegation window, a transfer basis selected from vacation coverage, emergency escalation, legal delegation, M&A integration, and machine-to-machine redundancy, the transferring principal's post-quantum signature, the receiving principal's post-quantum acknowledgment signature, cross-substrate witness commitments, and an explicit revocability field. The predecessor delegation hash is load-bearing: every Authority Transfer must walk back through a prior Authority Delegation event, and a transferred authority cannot originate from the transfer event itself.

[0021] The system may further implement pre-declared recovery authority via a committed Ratification Chain. Every root authority object that admits recovery may commit, as part of its Root Certification Record, a Ratification Chain naming the principals or principal-positions authorized to ratify recovery, the threshold required, and the per-principal signing weights. A recovery transition may be cryptographically valid only if the ratification proof presented at recovery time matches, on signer set and threshold, the Ratification Chain committed at certification time. A root authority object with no committed Ratification Chain may not be recoverable. The Ratification Chain primitive may be general, applying to any transition that requires pre-declared quorum including recovery, rotation under contested circumstances, higher-order governed override, and pre-certification conflict resolution. The operational property is that governance is encoded before failure occurs, not negotiated after failure.

[0022] In one advantageous feature of the present invention, the system substantially eliminates the coupling between authorization verifiability and issuing system availability. The coordinated operation of the root authority object store, the delegation graph engine, the pre-execution conformance gate, and the proof generation module produces portable cryptographic proof bundles that remain independently verifiable by any third party without contact with the system that produced them. This decoupling of verifiability from issuing system availability produces concrete technical improvements including the ability to replace, decommission, or migrate workflow engines while maintaining full verifiability of historical and ongoing authorization decisions.

[0023] In another advantageous feature of the present invention, the ancestor constraint aggregate computation enforces monotonic narrowing across the delegation graph. No cryptographic operation permits a child node to acquire any action class, data class, budget allowance, or temporal allowance exceeding the corresponding constraint of any ancestor node, ensuring that constraints imposed at each level are preserved and composed at every descendant level.

[0024] In another advantageous feature of the present invention, the cross-system cryptographic witness commitments bind authority decisions to contemporaneous external state. By carrying an evidence witness hash, an outcome witness hash, and a prior state hash in each authority node, the system enables verification that authorization decisions were consistent with the system state at the time they were made.

[0025] In another advantageous feature of the present invention, the Negative Authority Proof provides cryptographic proof of governed denial. By identifying the specific failing

condition and denial-category identifier, the Negative Authority Proof serves as portable evidence that the substrate refused an action, naming the specific failing check, and enabling future regulators, auditors, courts, insurers, or counterparties to verify that the substrate refused an action.

[0026] In another advantageous feature of the present invention, the structured authority rationale record provides plain-language explanations derived from cryptographic citations. By assembling the structured authority rationale record programmatically from the lineage proof, Authority Activation Record, intent citation, authority window data, acting principal data, and proposed action data, the system enables any auditor, regulator, court, or counterparty to understand why an action was authorized and verify each statement against underlying cryptographic citations.

[0027] In another advantageous feature of the present invention, the post-quantum threshold signatures provide long-term verification integrity. By using triple-family signatures comprising at least ML-DSA-65, SLH-DSA-SHA2-128f, and FALCON-512 algorithms providing independent mathematical hardness assumptions, the system ensures that proof bundles remain verifiable for the cryptographic lifetime of the post-quantum signature algorithms, with a working horizon measured in decades.

[0028] In another advantageous feature of the present invention, the system supports diverse authority preservation applications. The cryptographic authority substrate may preserve authority across delegation chains including Board to CEO to Department to Workflow to Integrated System to Autonomous Agent to External Counterparty; may enable cross-organizational authority configurations via peer-charter topologies or sovereign-delegation topologies; may enforce lifecycle-mode-specific governance including periodic review for RENEWABLE roots, completion-based retirement for PROJECT roots, and hard-expiry enforcement for EPHEMERAL roots; and may provide defense against poisoned-instruction attacks through author binding mechanisms and instruction source isolation at agent runtimes.

BRIEF DESCRIPTION OF FIGURES

[0029] FIG. 1 illustrates a block diagram of an Authority Preservation System, in accordance with one embodiment of the present invention.

[0030] FIG. 2 illustrates a block diagram of a root authority object, in accordance with one embodiment of the present invention.

[0031] FIG. 3 illustrates a hierarchical tree diagram of a delegation graph, in accordance with one embodiment of the present invention.

[0032] FIG. 4 illustrates a diagram of ancestor constraint aggregate computation, in accordance with one embodiment of the present invention.

[0033] FIG. 5 illustrates a flowchart of a four-condition test, in accordance with one embodiment of the present invention.

[0034] FIG. 6 illustrates a block diagram of a Portable Authority Proof, in accordance with one embodiment of the present invention.

[0035] FIG. 7 illustrates a block diagram of a Negative Authority Proof, in accordance with one embodiment of the present invention.

[0036] FIG. 8 illustrates a block diagram of an Authority Activation Record, in accordance with one embodiment of the present invention.

[0037] FIG. 9 illustrates a block diagram of an instruction tag and agent registry, in accordance with one embodiment of the present invention.

[0038] FIG. 10 illustrates a flowchart of a method for read receipt and instruction citation processing, in accordance with one embodiment of the present invention.

[0039] FIG. 11 illustrates a diagram of cross-system cryptographic witness commitments, in accordance with one embodiment of the present invention.

[0040] FIG. 12 illustrates a block diagram of an intent-passage registry, in accordance with one embodiment of the present invention.

[0041] FIG. 13 illustrates a diagram of authority window composition, in accordance with one embodiment of the present invention.

[0042] FIG. 14 illustrates a flowchart of a method for root succession, in accordance with one embodiment of the present invention.

[0043] FIG. 15 illustrates a diagram of authority state portability, in accordance with one embodiment of the present invention.

[0044] FIG. 16 illustrates a flowchart of a method for conflict detection and resolution, in accordance with one embodiment of the present invention.

[0045] FIG. 17 illustrates a flowchart of a method for root revocation and cascade, in accordance with one embodiment of the present invention.

[0046] FIG. 18 illustrates a flowchart of a method for override authorization, in accordance with one embodiment of the present invention.

[0047] FIG. 19 illustrates a flowchart of a method for RENEWABLE mode enforcement, in accordance with one embodiment of the present invention.

[0048] FIG. 20 illustrates a flowchart of a method for PROJECT and EPHEMERAL retirement, in accordance with one embodiment of the present invention.

[0049] FIG. 21 illustrates a flowchart of a method for a Root Certification Ceremony, in accordance with one embodiment of the present invention.

[0050] FIG. 22 illustrates a flowchart of a method for principal membership transition, in accordance with one embodiment of the present invention.

[0051] FIG. 23 illustrates a hierarchical diagram of a subtree freeze within a delegation graph, in accordance with one embodiment of the present invention.

[0052] FIG. 24 illustrates a diagram of structured authority rationale record assembly, in accordance with one embodiment of the present invention.

[0053] FIG. 25 illustrates a flowchart of a method for authority preservation, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0054] The following description sets forth exemplary aspects of systems and methods for cryptographic authority preservation across delegation chains. It should be recognized, however, that such description is not intended as a limitation on the scope of the present disclosure. Rather, the description also encompasses combinations and modifications to those exemplary aspects described herein, including alternative implementations of hash-linked delegation graphs, post-quantum threshold signatures, and multi-condition conformance gates.

[0055] A detailed description of systems, devices, and methods for producing independently verifiable cryptographic proof bundles via hash-linked delegation graphs and post-quantum cryptographic primitives is provided below. While several embodiments are described, it should be understood that disclosure is not limited to any one embodiment, but instead encompasses numerous alternatives, modifications, and equivalents. In addition, while numerous specific details are set forth in the following description in order to provide a thorough understanding of the authority preservation mechanisms, pre-execution conformance gates, and proof generation modules disclosed herein, some embodiments can be practiced without some or all of these details. Moreover, for the purpose of clarity, certain technical material relating to conventional cryptographic systems and authorization frameworks that is known in the related art has not been described in detail in order to avoid unnecessarily obscuring the disclosure.

[0056] The following terminology definitions apply throughout this disclosure and are provided to support the claims and ensure clarity of the terms used herein in connection with root authority objects, delegation graphs, ancestor constraint aggregates, pre-execution conformance gates, Authority Activation Records, Portable Authority Proofs, Negative Authority Proofs, and related cryptographic authority preservation constructs.

[0057] As used herein, the term “root authority object” refers to a cryptographically signed data structure representing a top-level statement of human-authored objective for an authority substrate. A root authority object may comprise at least an objective statement, a lifecycle mode indicator, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules. The root authority object may be cryptographically bound by a post-quantum threshold signature meeting the approval set. In some aspects, the root authority object may serve as the origin from which all downstream delegation and action authorization derives.

[0058] As used herein, the term “delegation graph” refers to a directed graph of authority nodes descended from a root authority object, wherein each authority node carries at least a

root hash, a parent hash, and a node hash that together establish a cryptographic lineage from the node back to the root authority object. In some cases, the delegation graph may span arbitrary delegation depths across heterogeneous principal categories including autonomous software agents, human actors, integrated systems, and cross-organizational workflows.

[0059] As used herein, the term "ancestor constraint aggregate" refers to a composed constraint set computed at any authority node as a function of every ancestor node from the immediate parent up to and including the root authority object. The ancestor constraint aggregate may comprise at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints. In some aspects, the ancestor constraint aggregate may be recomputed at every pre-execution conformance gate invocation rather than cached.

[0060] As used herein, the term "pre-execution conformance gate" refers to an enforcement mechanism that evaluates a set of conditions before any action is permitted to execute. The pre-execution conformance gate may enforce a four-condition test comprising an Activated condition, a Current condition, a Within Window condition, and a Lifecycle Compliant condition. In some cases, the pre-execution conformance gate may produce a Portable Authority Proof upon all conditions passing, or a Negative Authority Proof upon any condition failing.

[0061] As used herein, the term "Authority Activation Record" refers to a signed non-repudiation artifact produced by a designated principal acknowledging receipt of a governing authority object. An Authority Activation Record may comprise at least a predecessor artifact hash, a successor artifact hash being acknowledged, a reconstructed lineage view, cross-system cryptographic witness commitments, an acknowledgment timestamp, and a post-quantum signature by the acknowledging principal. In some aspects, the Authority Activation Record may be persisted in a per-principal append-only activation registry.

[0062] As used herein, the term "Portable Authority Proof" refers to a cryptographic proof bundle produced responsive to a four-condition test passing at the pre-execution conformance gate. A Portable Authority Proof may comprise a lineage proof from an action to a root authority object, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a specific human-authored passage from an intent-passage registry, and a structured authority rationale record. In some cases, a Portable Authority Proof may remain independently verifiable by any third party without contact with the system that produced it.

[0063] As used herein, the term "Negative Authority Proof" refers to a cryptographic proof bundle produced responsive to any condition of the four-condition test failing at the pre-execution conformance gate. A Negative Authority Proof may identify a failing condition and a denial-category identifier. In some aspects, a Negative Authority Proof may serve as portable evidence that the substrate refused an action, naming the specific failing check.

[0064] As used herein, the term "structured authority rationale record" refers to a data structure assembled programmatically from cryptographic citations carried by a Portable Authority Proof. A structured authority rationale record may comprise at least an acting principal identifier, an action summary, a cited intent passage in natural language, a delegation chain description, an authority window description, and an activation state description. In some cases, the structured authority rationale record may be derived from cited cryptographic artifacts via a signed mapping such that any party may verify the natural-language rendering corresponds to the cited cryptographic artifacts. The structured authority rationale record is not a free-text annotation but rather a programmatic assembly from cryptographic citations.

[0065] As used herein, the term "intent-passage registry" refers to a per-root-authority-object append-only data structure comprising canonical encodings of human-authored instruction passages. Each passage in the intent-passage registry may be signed by a registered author principal and assigned an intent passage identifier. In some aspects, each passage may be bound by inclusion proof to an intent-passage registry root hash signed by a publishing principal.

[0066] As used herein, the term "Root Certification Record" refers to a signed data structure produced during a Root Certification Ceremony at root creation. A Root Certification Record may comprise a full intent hash, a summary hash, a proposed root hash, display proof hashes, and a post-quantum threshold signature from approving principals. In some cases, the Root Certification Record may be bound by inclusion proof to a Root Certification Registry, and the pre-execution conformance gate may deny any action whose authority chain terminates at a root authority object lacking a valid Root Certification Record. As used herein, the term "display proof" refers to a cryptographic commitment to an exact rendering shown to an approving principal during a Root Certification Ceremony. A display proof may commit to the summary text shown, the navigation context, the action affordances offered, and the time of display.

[0067] As used herein, the term "cross-system cryptographic witness commitments" refers to a set of post-quantum digests carried by each authority node in a delegation graph, binding the authority node to the contemporaneous state of an evidence substrate and an outcome

substrate. Cross-system cryptographic witness commitments may comprise at least an evidence witness hash over a cycle root of an evidence substrate, an outcome witness hash over a cycle root of an outcome substrate, and a prior state hash over a system state observed at a time of node issuance.

[0068] As used herein, the term "Root Refresh Record" refers to a signed data structure emitted for a root authority object having a lifecycle mode indicator of RENEWABLE when the responsible certification authority elects to continue enforcement without material modification. A Root Refresh Record may comprise an unchanged root hash, a differential summary confirming no material modification, a re-displayed intent summary with display proofs, certifying principal signatures, a refresh effective timestamp, and a next review due timestamp.

[0069] As used herein, the term "lifecycle mode indicator" refers to a field carried by each root authority object that identifies which of a set of lifecycle modes governs the root authority object. The lifecycle mode indicator may be selected from a closed set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL. In some cases, each lifecycle mode may carry mode-specific schema fields and may trigger mode-specific enforcement at the pre-execution conformance gate. PERMANENT mode may represent constitution-level authority carrying no expiration. RENEWABLE mode may represent operating-policy authority carrying a renewal period and a next review due timestamp. PROJECT mode may represent mission-order authority carrying a project binding with completion criteria. EPHEMERAL mode may represent emergency authority carrying a hard expiry timestamp.

[0070] As used herein, the term "denial-category identifier" refers to a classifier included in a Negative Authority Proof that identifies the specific type of failure that caused the pre-execution conformance gate to deny an action. In some aspects, the denial-category identifier may indicate which of the four conditions of the four-condition test failed, or may indicate a specific contingency condition such as a stale authority condition, a revoked authority condition, a governance review missed condition, or a retired authority condition.

[0071] Referring to **FIG. 1**, an Authority Preservation System **100** for cryptographic authority preservation across delegation chains may comprise a processor **102**, a memory **104** storing executable instructions, and a bus **106** interconnecting the processor **102** and the memory **104**. The Authority Preservation System **100** may also be referred to herein as system **100**. The processor **102** may execute instructions stored in memory **104** to implement functional modules of the system **100**. The system **100** may implement a cryptographic authority substrate for preserving authorized human intent across delegation chains of arbitrary

depth spanning human actors, autonomous software agents, integrated systems, and cross-organizational workflows.

[0072] The system **100** may further comprise a non-transitory computer-readable medium **108** that stores instructions that, when executed by the processor **102**, cause the processor **102** to perform authority preservation operations including creating root authority objects, computing ancestor constraint aggregates, enforcing four-condition tests at a pre-execution conformance gate, and producing Portable Authority Proofs or Negative Authority Proofs. The memory **104** may store instructions that, when executed by the at least one processor **102**, cause the system **100** to perform operations for authority preservation across delegation chains.

[0073] With continued reference to **FIG. 1**, the system **100** may include a root authority object store **110** configured to maintain threshold-signed human-authored objectives. The root authority object store **110** may store root authority objects comprising objective statements, lifecycle mode indicators selected from a set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, approval sets with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules. Each root authority object stored in the root authority object store **110** may be cryptographically bound by a post-quantum threshold signature meeting the approval set of the root authority object.

[0074] The system **100** may include a delegation graph engine **112** configured to compute ancestor constraint aggregates for each authority node descended from root authority objects stored in the root authority object store **110**. The delegation graph engine **112** may compute, for each authority node in a delegation graph descended from a root authority object, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object. The ancestor constraint aggregate may comprise an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints.

[0075] As further shown in **FIG. 1**, the system **100** may include a pre-execution conformance gate **114** configured to enforce a four-condition test prior to action execution. The pre-execution conformance gate **114** may enforce, prior to execution of an action by an acting principal, a four-condition test comprising: (i) an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object; (ii) a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; (iii) a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and (iv) a Lifecycle Compliant condition

verifying that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp.

[0076] With continued reference to **FIG. 1**, the system **100** may include a proof generation module **116** configured to produce Portable Authority Proofs upon all conditions of the four-condition test passing, or Negative Authority Proofs upon any condition of the four-condition test failing. The proof generation module **116** may produce, responsive to the four-condition test passing, a Portable Authority Proof comprising a lineage proof, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a human-authored passage, and a structured authority rationale record. The proof generation module **116** may produce, responsive to any condition of the four-condition test failing, a Negative Authority Proof identifying a failing condition and a denial-category identifier. In some cases, the proof generation module **116** may include, in each proof bundle, a portable verifier identifier referencing an open-source verifier binary licensed under Apache-2.0 or a similar license for offline verification. The open-source verifier binary, when executed offline against the proof bundle, may produce a verdict.

[0077] The system **100** may include a registry complex **118** configured to maintain append-only registries. The registry complex **118** may maintain activation registries for storing Authority Activation Records, succession registries for storing succession events binding successor root authority objects to predecessor root authority objects, revocation registries for storing Root Revocation Records, refresh registries for storing Root Refresh Records, retirement registries for storing Root Retirement Records, intent-passage registries for storing canonical encodings of human-authored instruction passages, agent registries for binding candidate recipient agents to post-quantum signing public keys and post-quantum key-encapsulation public keys, and precedent registries for storing conflict precedents.

[0078] As further shown in **FIG. 1**, the system **100** may include a human principal interface **120** configured to enable human actors to interact with the cryptographic authority substrate. The human principal interface **120** may enable human actors to create root authority objects, approve delegations, participate in Root Certification Ceremonies, and resolve conflicts through conflict resolution records.

[0079] The system **100** may include an agent runtime interface **122** configured to enable autonomous software agents to interact with the cryptographic authority substrate. The agent runtime interface **122** may enable autonomous software agents to receive instruction tags, produce read receipts upon decryption of instruction bodies, and submit proposed actions for evaluation by the pre-execution conformance gate **114**.

[0080] With continued reference to **FIG. 1**, the system **100** may include a workflow engine interface **124** configured to enable integration with heterogeneous workflow execution engines while maintaining authority state portability. The workflow engine interface **124** may enable migration of authority state from a first workflow execution engine to a second workflow execution engine by serializing an authority graph from the first workflow execution engine and de-serializing the authority graph into the second workflow execution engine without re-issuance of any signature in the authority graph.

[0081] The system **100** may include an external substrate interface **126** configured to enable cross-system cryptographic witness commitments binding authority nodes to evidence substrates and outcome substrates. The external substrate interface **126** may enable each authority node in a delegation graph to carry witness commitments comprising an evidence witness hash over a cycle root of an evidence substrate, an outcome witness hash over a cycle root of an outcome substrate, and a prior state hash over a system state observed at a time of node issuance.

[0082] The root authority object stored in the root authority object store **110** is described in greater detail with reference to **FIG. 2**. Referring to **FIG. 2**, a root authority object **200** represents a cryptographically signed top-level statement of human-authored objective from which all downstream delegation and action authorization derives. The root authority object **200** may serve as the origin for an authority substrate, and every action within the authority substrate may descend from the root authority object **200** via a delegation graph.

[0083] The root authority object **200** may comprise an objective statement field **202** containing a typed policy expression in a deterministic policy language. In some cases, the deterministic policy language may comprise Cedar, Open Policy Agent Rego, Datalog, or another typed policy language. The objective statement field **202** may ensure that any evaluation of the objective against a proposed action is fully deterministic and reproducible by any third party.

[0084] The root authority object **200** may comprise a lifecycle mode indicator **204** selected from a closed set consisting of four modes: PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL. PERMANENT mode may represent constitution-level authority carrying no expiration and requiring a succession quorum exceeding the approval set threshold for supersession. RENEWABLE mode may represent operating-policy authority carrying a renewal period and a next review due timestamp and requiring periodic Root Refresh Records. PROJECT mode may represent mission-order authority carrying a project binding with completion criteria and auto-retiring upon completion. EPHEMERAL mode may represent

emergency authority carrying a hard expiry timestamp and requiring human recertification for extension.

[0085] The root authority object **200** may comprise an approval set field **206** specifying threshold semantics for authorization. In some cases, the approval set field **206** may comprise at least three signers with a threshold of two required for authorization. The approval set field **206** may define which principals and how many of those principals are required to sign the root authority object **200** for the root authority object **200** to become valid.

[0086] The root authority object **200** may comprise an allowed action classes field **208** defining permissible actions that may be authorized under the root authority object **200**. The root authority object **200** may comprise a prohibited action classes field **210** defining forbidden actions that may not be authorized under the root authority object **200** regardless of other constraints. The root authority object **200** may comprise a temporal constraints field **212** specifying validity windows during which the root authority object **200** may authorize actions.

[0087] The root authority object **200** may comprise a revocation rules field **214** containing authorities for various revocation and override operations. The revocation rules field **214** may contain a subtree freeze authority **214a** naming principals who may freeze subtrees within a delegation graph descended from the root authority object **200**. The revocation rules field **214** may contain a succession authority **214b** naming principals who may authorize root succession from the root authority object **200** to a successor root authority object. The revocation rules field **214** may contain a revocation authority **214c** naming principals who may revoke authority artifacts descended from the root authority object **200**. The revocation rules field **214** may contain an override authority **214d** naming principals who may issue overrides for actions that would otherwise be denied.

[0088] A post-quantum threshold signature **216** may cryptographically bind the entire root authority object **200**. The post-quantum threshold signature **216** may comprise a triple-family signature providing independent mathematical hardness assumptions. In some cases, the post-quantum threshold signature **216** may comprise at least ML-DSA-65, SLH-DSA-SHA2-128f, and FALCON-512 algorithms. In some aspects, the post-quantum threshold signature **216** at the root authority object **200** may use highest NIST security category algorithms comprising at least ML-DSA-87, SLH-DSA-SHA2-256s, and FALCON-1024. The triple-family signature approach may provide cryptographic resilience against potential vulnerabilities in any single algorithm family.

[0089] A root hash **218** may be derived from the bound structure of the root authority object **200**. The root hash **218** may serve as a unique identifier for the root authority object **200**

and may be carried by every descendant authority node in a delegation graph to establish cryptographic lineage back to the root authority object **200**.

[0090] Authority propagates from the root authority object **200** through a delegation graph, as shown in **FIG. 3**.

[0091] Referring to **FIG. 3**, a delegation graph **300** represents a directed graph of authority nodes descended from the root authority object **200** described with reference to **FIG. 2**. The delegation graph **300** may span arbitrary delegation depths across heterogeneous principal categories including human actors, autonomous software agents, integrated systems, and cross-organizational workflows.

[0092] A root node **302** at the top of the delegation graph **300** represents the root authority object **200**. The root node **302** may serve as the origin from which all downstream delegation and action authorization derives within the delegation graph **300**.

[0093] Level-one authority nodes **304a**, **304b**, and **304c** descend from the root node **302**, representing first-level delegations. In some cases, the level-one authority nodes **304a**, **304b**, and **304c** may represent delegations such as Board to CEO, Board to CFO, and Board to General Counsel, respectively. Level-two authority nodes **306a** and **306b** descend from authority node **304b**, representing second-level delegations. In some aspects, the level-two authority nodes **306a** and **306b** may represent delegations such as CFO to Finance Manager and CFO to Treasury, respectively. A level-three authority node **308** descends from authority node **306a**, representing a third-level delegation. In some cases, the level-three authority node **308** may represent a delegation such as Finance Manager to Invoice Approval Agent.

[0094] Each authority node in the delegation graph **300** may carry three hash fields: a root hash linking back to the originating root authority object **200**, a parent hash linking to the immediate parent node, and a node hash uniquely identifying the current node. The root hash carried by each authority node may establish cryptographic lineage from the authority node back to the root authority object **200**.

[0095] A leaf action **310** descends from authority node **308** and represents a specific action to be executed. In some aspects, the leaf action **310** may represent an action such as approving invoice #8421 for $14,200 to ACME Corp. The leaf action **310** may be subject to evaluation by the pre-execution conformance gate **114** prior to execution.

[0096] A lineage proof path **312**, shown as a dashed upward arrow in **FIG. 3**, represents the cryptographic lineage from the leaf action **310** to the root node **302**. The lineage proof path **312** may be verifiable offline by any third party without contact with the system that produced the delegation graph **300**.

[0097] The delegation graph **300** may support arbitrary delegation depth across heterogeneous principal categories. In some cases, the delegation graph **300** may support delegation chains including Board to CEO to Department to Workflow to Integrated System to Autonomous Agent to External Counterparty. The portable authority artifact produced from the delegation graph **300** may remain valid and verifiable for at least the cryptographic lifetime of the post-quantum signature algorithm used at the root authority object **200**, with a working horizon measured in decades.

[0098] As described with reference to **FIG. 1**, the delegation graph engine **112** may compute, for each authority node in the delegation graph **300** descended from the root authority object **200**, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object **200**. **FIG. 4** illustrates the computation of the ancestor constraint aggregate in greater detail.

[0099] Referring to **FIG. 4**, an ancestor constraint aggregate computation **400** illustrates how constraints from all ancestors may be composed at any authority node in a delegation graph. A root node **408** at the top of the ancestor chain may carry a set of constraints comprising allowed action classes, prohibited action classes, temporal validity windows, and policy constraints. A grandparent node **406** may descend from the root node **408** and may carry a set of constraints. A parent node **404** may descend from the grandparent node **406** and may carry a set of constraints. A target node **402** may descend from the parent node **404** and may represent the authority node for which an ancestor constraint aggregate is computed.

[0100] The constraints of all ancestor nodes may flow downward from the root node **408** through the grandparent node **406** and the parent node **404** to the target node **402**. A composition block **410** may receive the constraints from every ancestor node and may perform four composition operations to compute an ancestor constraint aggregate **412**.

[0101] The ancestor constraint aggregate **412** may comprise at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints. The composition block **410** may compute the intersection of allowed action classes such that the target node **402** can perform actions permitted by every ancestor in the ancestor chain. The composition block **410** may compute the union of prohibited action classes such that the target node **402** is prohibited from any action forbidden by any ancestor in the ancestor chain. The composition block **410** may compute the intersection of temporal validity windows such that the validity window of the target node **402** falls within every ancestor's validity window. The composition block **410** may compose policy constraints following type-specific composition rules.

[0102] In some cases, the root node **408** may allow action classes comprising invoice approval, payment initiation, and vendor management. The parent node **404** may allow action classes comprising invoice approval and payment initiation. The intersection of allowed action classes computed by the composition block **410** may yield allowed action classes comprising invoice approval and payment initiation at the target node **402**. In some aspects, the grandparent node **406** may prohibit international wire transfers and the parent node **404** may prohibit cryptocurrency payments. The union of prohibited action classes computed by the composition block **410** may yield prohibited action classes comprising international wire transfers and cryptocurrency payments at the target node **402**.

[0103] The ancestor constraint aggregate computation **400** may enforce monotonic narrowing across the delegation graph. No cryptographic operation permits a child node to acquire any action class, data class, budget allowance, or temporal allowance exceeding the corresponding constraint of any ancestor node. The ancestor constraint aggregate **412** may be recomputed deterministically at every pre-execution conformance gate invocation rather than cached, ensuring fresh evaluation against current state at the time of action evaluation.

[0104] The deterministic recomputation of the ancestor constraint aggregate at every gate invocation from the same inputs allows two independent verifiers evaluating the same authority chain against the same proposed action to produce identical ancestor constraint aggregates without coordination.

[0105] Referring to **FIG. 5**, a four-condition test **500** illustrates the evaluation sequence at the pre-execution conformance gate **114** prior to execution of any action. The four-condition test **500** may determine whether a proposed action by an acting principal is permitted or denied based on evaluation of four sequential conditions.

[0106] The method begins at step 502 when a proposed action is received from an acting principal. The proposed action may comprise a canonical encoding of the action to be performed, an identifier of the acting principal, and an action timestamp.

[0107] The proposed action is then evaluated against four sequential conditions. An activated condition **504** verifies that the acting principal has produced an Authority Activation Record acknowledging a governing authority object. The governing authority object may be one of a root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact. The activated condition **504** may ensure the acting principal has demonstrably acknowledged the authority under which the acting principal acts. In some cases, the activated condition **504** may verify that the Authority Activation Record is persisted in a per-principal append-only activation registry.

[0108] A current condition **506** verifies that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal. The current condition **506** implements freshness verification at every layer. In some aspects, the pre-execution conformance gate **114** may enforce freshness at every layer including root freshness, delegation freshness via an acknowledged delegation chain hash, and instruction-set freshness. An action by a principal whose acknowledged root hash differs from the authority chain's current active root hash, or whose acknowledged delegation chain hash differs from the authority chain's current delegation chain hash, or whose acknowledged instruction-set commitment root differs from the current commitment root, may be denied by the pre-execution conformance gate **114**. The current condition **506** may prevent stale-intent attacks where an agent acts under a superseded root that has since been replaced by a successor root.

[0109] A within-window condition **508** verifies that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain. The within-window condition **508** enforces temporal constraints across the delegation depth. In some cases, the authority windows may comprise clock-window scopes specifying permitted times of day, calendar-window scopes specifying permitted calendar periods, event-window scopes conditional on substrate-observable events, and condition-window scopes conditional on substrate-observable predicates. The effective authority window at a leaf action may be the intersection of authority window fields of every ancestor from the leaf action to the root authority object.

[0110] A lifecycle-compliant condition **510** verifies that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp. The lifecycle-compliant condition **510** may dispatch on the lifecycle mode indicator and apply mode-specific checks. For PERMANENT mode, the lifecycle-compliant condition **510** may require no additional check beyond the standard authority window. For RENEWABLE mode, the lifecycle-compliant condition **510** may verify that the most recent Root Refresh Record's next review due timestamp is in the future at the action timestamp. For PROJECT mode, the lifecycle-compliant condition **510** may verify that a completion criteria predicate is FALSE at the action timestamp. For EPHEMERAL mode, the lifecycle-compliant condition **510** may verify that the action timestamp is at or before a hard expiry timestamp.

[0111] In some aspects, the pre-execution conformance gate **114** may enforce a five-stage activation lifecycle comprising delivered, read, acknowledged, activated, and operated under stages. The first four stages bind the principal to the capability to act under current intent. The

fifth stage binds each individual action to the specific acknowledgment under which the action operates. The delivered stage may comprise delivery via post-quantum key-encapsulation mechanism wrapping to the recipient. The read stage may comprise production of a signed read receipt over a decrypted instruction body. The acknowledged stage may comprise production of an Authority Activation Record signed over the governing artifact. The activated stage may comprise registration of the acknowledgment in the per-principal activation registry. The operated under stage may comprise citation of the activated artifact and Authority Activation Record in every subsequent action proof.

[0112] In some cases, the pre-execution conformance gate **114** may comprise 30 total checks including 12 primary checks and 18 extension checks contributed by specific independent claims. The 12 primary checks may verify root existence, parent existence, parent authorization of delegation scope, action within scope, temporal validity window validity, policy constraint satisfaction, required human approvals, sensitive-data access rules, revocation registry exclusion, threshold post-quantum signature validity, lineage proof validity, and root conformance. The 18 extension checks may be contributed by claims addressing instruction citation, preservation trilogy binding, anti-poisoning, freshness verification, activated-authority citation, revocation registry exclusion along the authority chain, authority window intersection, intent citation, root certification, principal role continuity, lifecycle mode compliance, retirement compliance, and authority transfer lineage closure.

[0113] If all four conditions are satisfied, the method proceeds to a permit output **512**. The permit output **512** produces a Portable Authority Proof. The Portable Authority Proof may comprise a lineage proof from the action to the root authority object, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a specific human-authored passage from an intent-passage registry, and a structured authority rationale record assembled programmatically from the citations.

[0114] If any one of the four conditions fails, the method proceeds to a deny output **514**. The deny output **514** produces a Negative Authority Proof with a denial-category identifier. The denial-category identifier may indicate which of the four conditions failed. In some cases, the denial-category identifier may indicate a specific contingency condition such as stale authority, revoked authority, governance review missed, retired authority, off window, or intent citation missing.

[0115] Producing a deterministic binary outcome, either a Portable Authority Proof or a Negative Authority Proof, from the same inputs, the four-condition test yields a gate output

reproducible by any verifier holding the same authority chain state, without requiring contact with the original system.

[0116] Referring to **FIG. 6**, a Portable Authority Proof **600** represents a cryptographic proof bundle produced responsive to the four-condition test passing at the pre-execution conformance gate **114**. The Portable Authority Proof **600** may be independently verifiable by any third party without contact with the system that produced the Portable Authority Proof **600**.

[0117] The Portable Authority Proof **600** may comprise a lineage proof **602** from the action to the root authority object **200**. The lineage proof **602** may comprise an ordered chain of node hash, parent hash, and root hash values from the leaf action to the root authority object **200**. Each link in the lineage proof **602** may bear the threshold signatures of the issuing principal. The lineage proof **602** may establish cryptographic descent from the action through every intermediate authority node to the root authority object **200**.

[0118] The Portable Authority Proof **600** may comprise cross-system cryptographic witness commitments **604**. The cross-system cryptographic witness commitments **604** may bind the authority decision to the contemporaneous state of external substrates at the time of action authorization. The cross-system cryptographic witness commitments **604** may comprise an evidence witness hash **604a**, an outcome witness hash **604b**, and a prior state hash **604c**. The evidence witness hash **604a** may comprise a post-quantum digest over a cycle root of an evidence substrate. The outcome witness hash **604b** may comprise a post-quantum digest over a cycle root of an outcome substrate. The prior state hash **604c** may comprise a post-quantum digest over the system state observed at the time of node issuance.

[0119] The Portable Authority Proof **600** may comprise an instruction citation **606**. The instruction citation **606** may comprise an instruction identifier **606a**, an inclusion proof **606b**, and a signed read receipt **606c**. The instruction identifier **606a** may uniquely identify the cited instruction tag. The inclusion proof **606b** may bind the cited instruction tag to the instruction-set commitment root in force at action time. The signed read receipt **606c** may comprise a digest of the decrypted instruction body signed by the recipient agent. The signed read receipt **606c** may demonstrate that the recipient agent received and acknowledged the instruction prior to action execution.

[0120] The Portable Authority Proof **600** may comprise an activated-authority citation **608**. The activated-authority citation **608** may comprise an activation record hash **608a**, an activated governing authority object hash **608b**, and a binding proof **608c**. The activation record hash **608a** may reference the Authority Activation Record produced by the acting principal. The activated governing authority object hash **608b** may identify the acknowledged

artifact under which the acting principal operates. The binding proof **608c** may demonstrate that the constraints evaluated by the pre-execution conformance gate **114** correspond to the activated governing authority object.

[0121] The Portable Authority Proof **600** may comprise an intent citation **610** referencing a specific human-authored passage from an intent-passage registry. The intent citation **610** may comprise an intent passage identifier **610a**, a canonical hash of cited passage text **610b**, and a scope-conformance verification binding **610c**. The intent passage identifier **610a** may be assigned to the human-authored passage that created the authority under which the action operates. The canonical hash of cited passage text **610b** may represent the passage text at the cited epoch. The scope-conformance verification binding **610c** may demonstrate that the action's scope is consistent with the cited passage.

[0122] The Portable Authority Proof **600** may comprise a structured authority rationale record **612** assembled programmatically from the citations. The structured authority rationale record **612** may comprise an acting principal identifier **612a**, an action summary **612b**, a cited intent passage **612c** in natural language, a delegation chain description **612d**, an authority window description **612e**, and an activation state description **612f**. The acting principal identifier **612a** may identify the principal performing the action, such as "Invoice Approval Agent." The action summary **612b** may describe the action performed, such as "Approved Invoice #8421 for $14,200 to ACME Corp." The cited intent passage **612c** may contain the human-authored text that authorized the action, such as "Review invoices up to $25,000 and escalate anything larger." The delegation chain description **612d** may describe the authority path, such as "Finance Manager → AP Agent." The authority window description **612e** may describe the temporal scope, such as "Q3 FY26, weekdays 09:00-17:00 EST." The activation state description **612f** may describe the activation status, such as "Activated 2026-07-01; acknowledged root matches current active root."

[0123] In some aspects, the structured authority rationale record **612** may include a lifecycle mode description in plain-language form. The lifecycle mode description may indicate the lifecycle mode of the root authority object **200** and the review status, such as "Renewable; reviewed 2026-Q3; next review due 2026-Q4." The structured authority rationale record **612** may answer the question "Why did this agent think it was allowed?" in natural language suitable for auditors, courts, regulators, insurers, boards, and counterparties.

[0124] A post-quantum signature **614** may bind the entire Portable Authority Proof **600**. The post-quantum signature **614** may comprise a triple-family signature providing independent mathematical hardness assumptions. The post-quantum signature **614** may ensure that the

Portable Authority Proof **600** remains verifiable for the cryptographic lifetime of the post-quantum signature algorithms.

[0125] Carrying the lineage proof, witness commitments, instruction citation, activated-authority citation, intent citation, and structured authority rationale record in a single self-contained bundle, the Portable Authority Proof enables a verifier to confirm the complete authority chain from the human-authored intent passage to the executed action without access to any intermediate system or registry.

[0126] Referring to **FIG. 7**, a Negative Authority Proof **700** represents a cryptographic proof bundle produced responsive to any condition of the four-condition test failing at the pre-execution conformance gate **114**. The Negative Authority Proof **700** may serve as portable evidence that the substrate refused an action, naming the specific failing check. The Negative Authority Proof **700** may be independently verifiable by any future regulator, auditor, court, insurer, or counterparty as evidence that the substrate refused an action. The Negative Authority Proof **700** is not an audit log but rather a portable proof of governed denial.

[0127] The Negative Authority Proof **700** may comprise a proposed action canonical encoding **702**. The proposed action canonical encoding **702** may contain a deterministic encoding of the action that was denied. The proposed action canonical encoding **702** may enable any verifier to reconstruct the exact action that was evaluated by the pre-execution conformance gate **114**.

[0128] The Negative Authority Proof **700** may comprise a failing condition identifier **704**. The failing condition identifier **704** may identify which of conditions (i), (ii), (iii), or (iv) failed. Condition (i) corresponds to the Activated condition. Condition (ii) corresponds to the Current condition. Condition (iii) corresponds to the Within Window condition. Condition (iv) corresponds to the Lifecycle Compliant condition. The failing condition identifier **704** may enable precise identification of the specific check that caused the denial.

[0129] The Negative Authority Proof **700** may comprise a denial-category identifier **706**. The denial-category identifier **706** may be selected from enumerated categories providing specific failure classification. In some cases, the denial-category identifier **706** may include an escalation classifier. The escalation classifier may include stale authority escalation, activation citation missing, activation citation invalid, conflict pending resolution, override required, override expired escalation, revoked authority escalation, off window escalation, intent citation missing, intent citation invalid, certification missing, certification invalid, principal role orphan, governance review missed, refresh drift exceeded, project completed, ephemeral expired, and retired authority escalation.

[0130] In some aspects, a governance review missed Negative Authority Proof **700** may be a cryptographic proof of the absence of a governance event that was due. The governance review missed Negative Authority Proof **700** may be distinct from audit logs that record events that occurred. The governance review missed Negative Authority Proof **700** may represent a proof primitive for absences, providing proof that a required periodic review did not occur.

[0131] The Negative Authority Proof **700** may comprise a contingency artifact **708**. The contingency artifact **708** may be an optional attachment. In some cases, the contingency artifact **708** may include a Conflict Capsule when conflict_pending_resolution is indicated by the denial-category identifier **706**. In some cases, the contingency artifact **708** may include an Override Capsule when override_required is indicated by the denial-category identifier **706**. The contingency artifact **708** may include other relevant artifacts for escalation paths.

[0132] A post-quantum signature **710** may bind the entire Negative Authority Proof **700**. The post-quantum signature **710** may ensure that the Negative Authority Proof **700** remains verifiable for the cryptographic lifetime of the post-quantum signature algorithms. The post-quantum signature **710** may prevent tampering with any field of the Negative Authority Proof **700** after issuance.

[0133] The inclusion of the failing condition identifier and denial-category identifier in a signed portable artifact allows a third party to verify that the system refused a specific action for a specific reason without access to the system that produced the denial.

[0134] Referring to **FIG. 8**, an Authority Activation Record **800** represents a signed non-repudiation artifact produced by a designated principal acknowledging receipt of a governing authority object. The Authority Activation Record **800** may be required prior to any action by a recipient principal under a governing authority object. The governing authority object may be one of a root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact. No action by the recipient principal under the governing authority object may be emitted prior to the production and persistence of the Authority Activation Record **800**.

[0135] The Authority Activation Record **800** may comprise a predecessor artifact hash **802**. The predecessor artifact hash **802** may identify the prior authority artifact the principal was operating under. The predecessor artifact hash **802** may enable lineage tracking across authority transitions by linking the current acknowledgment to the principal's previous authority state. In some cases, the predecessor artifact hash **802** may be omitted where no prior artifact exists, such as when a principal is first activated under an initial root authority object.

[0136] The Authority Activation Record **800** may comprise a successor artifact hash **804**. The successor artifact hash **804** may identify the governing authority object being acknowledged. The successor artifact hash **804** may reference a root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact. The successor artifact hash **804** may establish the authority under which the principal operates for subsequent actions.

[0137] The Authority Activation Record **800** may comprise a reconstructed lineage view **806**. The reconstructed lineage view **806** may comprise the principal's reconstruction of the lineage from the principal's local authority node to the activated artifact. The reconstructed lineage view **806** may enable verification that the principal understood the principal's position in the delegation hierarchy at the time of acknowledgment. In some aspects, the reconstructed lineage view **806** may demonstrate that the principal has correctly identified the chain of authority from the principal's position in the delegation graph up to the governing authority object.

[0138] The Authority Activation Record **800** may comprise cross-system cryptographic witness commitments **808**. The cross-system cryptographic witness commitments **808** may bind the acknowledgment to the contemporaneous state of evidence and outcome substrates at acknowledgment time. As described with reference to **FIG. 6**, cross-system cryptographic witness commitments may comprise an evidence witness hash, an outcome witness hash, and a prior state hash. The cross-system cryptographic witness commitments **808** in the Authority Activation Record **800** may capture the substrate state at the moment of acknowledgment, enabling verification that the principal acknowledged the governing authority object against a specific system state.

[0139] The Authority Activation Record **800** may comprise an acknowledgment timestamp **810**. The acknowledgment timestamp **810** may record when the acknowledgment occurred. The acknowledgment timestamp **810** may enable temporal ordering of acknowledgments and may be used to verify that the acknowledgment preceded any actions taken under the acknowledged authority.

[0140] The Authority Activation Record **800** may comprise a post-quantum signature **812** by the acknowledging principal. The post-quantum signature **812** may provide non-repudiation using the principal's registered signing key. In some cases, the post-quantum signature **812** may comprise a triple-family signature providing independent mathematical hardness assumptions. The post-quantum signature **812** may ensure that the Authority Activation Record **800** is

producible only by a principal in possession of the corresponding private key whose registry entry was valid at acknowledgment time.

[0141] The Authority Activation Record **800** may be persisted in an activation registry **814**. The activation registry **814** may be a per-principal append-only activation registry. The append-only nature of the activation registry **814** may ensure that acknowledgment records cannot be modified or deleted after creation. The pre-execution conformance gate **114** may access the activation registry **814** to verify that an acting principal has produced an Authority Activation Record **800** acknowledging the governing authority object.

[0142] The pre-execution conformance gate **114** may verify, at the pre-execution conformance gate **114**, that the acknowledged root hash of the acting principal equals the current active root hash for the authority chain. The verification may implement a freshness check that prevents stale-intent attacks where a principal acts under a superseded root that has since been replaced by a successor root. An action by a principal whose acknowledged root hash differs from the authority chain's current active root hash may be denied by the pre-execution conformance gate **114**.

[0143] Every action proof may require an activated-authority citation. The activated-authority citation may comprise a hash of the Authority Activation Record **800**, a hash of an activated governing authority object, and a binding proof that constraints evaluated by the pre-execution conformance gate **114** correspond to the activated governing authority object. The activated-authority citation may establish that the acting principal acknowledged the authority under which the acting principal operates and that the constraints evaluated at the pre-execution conformance gate **114** match the acknowledged authority.

[0144] Referring to **FIG. 9**, an instruction tag and agent registry diagram **900** illustrates structures enabling secure multi-recipient instruction distribution with post-quantum cryptographic protection. The instruction tag and agent registry diagram **900** shows the relationship between an agent registry **902** and an instruction tag **912**.

[0145] The agent registry **902** may bind each candidate recipient agent to cryptographic keys. The agent registry **902** may be published as described with reference to **FIG. 1**. Each entry in the agent registry **902** may comprise an agent identifier **904**, a signing public key **906**, a KEM public key **908**, and a parent node reference **910**. The agent identifier **904** may uniquely identify the agent within the authority substrate. The signing public key **906** may be a post-quantum signing public key. In some cases, the signing public key **906** may comprise an ML-DSA-87 public key. The KEM public key **908** may be a post-quantum key-encapsulation public key. In some aspects, the KEM public key **908** may comprise an ML-KEM-1024 public key at

NIST security category 5. The parent node reference **910** may commit the agent's provisioning lineage in the delegation graph **300** described with reference to **FIG. 3**.

[0146] The instruction tag **912** may comprise an instruction identifier **914**, an encrypted body **916**, a wrapped-keys field **918**, an author signature field **920**, and an intent passage identifier field **922**. The instruction identifier **914** may uniquely identify the instruction tag **912** within an instruction set. The encrypted body **916** may contain natural-language content encrypted under a per-instruction data encryption key. The wrapped-keys field **918** may contain the per-instruction data encryption key separately encapsulated under each intended recipient's KEM public key **908** by a post-quantum key-encapsulation mechanism. In some cases, the post-quantum key-encapsulation mechanism may be ML-KEM-1024 at NIST security category 5.

[0147] The author signature field **920** may carry a post-quantum signature by a registered author principal over a canonical encoding of a decrypted instruction body. The author principal may be distinct from a publishing principal. The distinction between the author principal and the publishing principal may enable separation of content authorship from publication authority. The intent passage identifier field **922** may identify the human-authored passage that authored the instruction body. The intent passage identifier field **922** may reference an entry in the intent-passage registry described with reference to **FIG. 12**.

[0148] An instruction set may comprise a plurality of instruction tags **912**. The instruction-set commitment may be a Merkle accumulator with logarithmic inclusion proof size per instruction tag **912**. The Merkle accumulator structure may enable efficient verification of instruction tag membership without requiring access to the complete instruction set.

[0149] Referring to **FIG. 10**, a read receipt and instruction citation flowchart **1000** illustrates the processing sequence for establishing non-repudiation of instruction reading. The read receipt and instruction citation flowchart **1000** may enable verification that a recipient agent demonstrably received and acknowledged an instruction prior to action execution.

[0150] At step 1002, a recipient agent receives an instruction tag from an instruction set. The instruction tag may be the instruction tag 912 described with reference to FIG. 9, comprising an instruction identifier 914, an encrypted body 916, a wrapped-keys field 918, an author signature field 920, and an intent passage identifier field 922.

[0151] At step 1004, the recipient agent unwraps a data encryption key from the wrapped-keys field 918 using the recipient's key-encapsulation private key. The key-encapsulation private key may correspond to the KEM public key 908 registered in the agent registry 902

described with reference to FIG. 9. In some cases, the unwrapping operation may use a post-quantum key-encapsulation mechanism such as ML-KEM-1024.

[0152] At step 1006, the recipient agent decrypts the encrypted body 916 using the unwrapped data encryption key to obtain natural-language instruction content. The decryption operation may yield the instruction body that was encrypted by the publishing principal under the per-instruction data encryption key.

[0153] At step 1008, the recipient agent computes a canonical digest of the decrypted instruction body. The canonical digest may be computed using a post-quantum hash function. The canonical digest may serve as a unique identifier for the specific instruction content received by the recipient agent.

[0154] At step **1010**, the recipient agent produces a signed read receipt comprising the computed digest and a post-quantum signature using the agent's registered signing key. The signed read receipt may comprise a digest of the decrypted instruction body and a post-quantum signature by the recipient agent.

[0155] At step **1012**, the read receipt is persisted. The persistence of the read receipt may create non-repudiation evidence that the specific agent demonstrably read the specific instruction. In some cases, the read receipt may be producible only by an agent in possession of the corresponding key-encapsulation private key whose registry entry was valid at decryption time. The read receipt may be persisted in a per-agent append-only receipt registry.

[0156] At step **1014**, at action time, an instruction citation is included in an action proof. The instruction citation may comprise the instruction identifier **914**, an inclusion proof against an instruction-set commitment root, and the signed read receipt. The instruction citation may enable verification that the acting agent received and acknowledged the cited instruction prior to action execution.

[0157] The pre-execution conformance gate **114** may verify the instruction citation by confirming that the proposed action cites at least one instruction tag identifier, that the inclusion proof verifies against the instruction-set commitment root, that the read receipt verifies under the agent's registered signing key and the registry entry was valid at decryption time, and that the action scope lies within an intersection of cited instruction envelopes.

[0158] Referring to **FIG. 11**, a cross-system cryptographic witness commitments diagram **1100** illustrates how each authority node in a delegation graph is cryptographically bound to the contemporaneous state of external evidence and outcome substrates. The cross-system cryptographic witness commitments diagram **1100** may form a preservation trilogy binding authority decisions to external substrate state at the time of authorization.

[0159] An authority node **1102** represents any node in the delegation graph **300** described with reference to **FIG. 3**. The authority node **1102** may carry witness commitments that bind the authority node **1102** to external substrates at the time of node issuance.

[0160] An evidence witness hash **1104** may comprise a post-quantum digest over a cycle root of an evidence substrate **1106**. The evidence substrate **1106** may be an evidence preservation substrate. The evidence witness hash **1104** may cryptographically commit the authority node **1102** to the state of evidence records that existed at the time of node issuance.

[0161] An outcome witness hash **1108** may comprise a post-quantum digest over a cycle root of an outcome substrate **1110**. The outcome substrate **1110** may be an outcome preservation substrate. The outcome witness hash **1108** may cryptographically commit the authority node **1102** to the state of outcome records that existed at the time of node issuance.

[0162] A prior state hash **1112** may comprise a post-quantum digest over a system state at time of node issuance **1114**. The system state at time of node issuance **1114** may capture a runtime model fingerprint, a corpus snapshot identifier, and a policy version identifier in force at issuance time. The prior state hash **1112** may bind the authority node **1102** to the computational context that existed when the authority node **1102** was created.

[0163] The three witness commitments—the evidence witness hash **1104**, the outcome witness hash **1108**, and the prior state hash **1112**—may feed into a node hash **1116**. The node hash **1116** may cryptographically bind the authority node **1102** to the contemporaneous external state. The witness commitments may be bound into the node hash **1116** such that modification of any witness commitment invalidates a signature of the authority node **1102** and every descendant lineage proof.

[0164] A note **1118** in **FIG. 11** explains that modification of any witness commitment invalidates the signature of the authority node **1102** and every descendant lineage proof. The invalidation may provide tamper-evidence across the entire descendant subtree of the authority node **1102**. In some cases, any attempt to alter the evidence witness hash **1104**, the outcome witness hash **1108**, or the prior state hash **1112** after node issuance may cause verification failure for the authority node **1102** and for every authority node descended from the authority node **1102** in the delegation graph.

[0165] Binding the witness commitments into the node hash means that any alteration of the evidence substrate state, outcome substrate state, or prior system state after node issuance causes the node hash to change, which in turn invalidates the post-quantum threshold signature of the authority node and propagates verification failure through every descendant in the lineage proof path.

[0166] In some aspects, the pre-execution conformance gate **114** may enforce a witness commitment check. The witness commitment check may verify that a proposed action carries witness commitments that match the most recent valid commitments at an immediate parent node, or that the proposed action carries a signed authority-issued update event explaining any divergence. The witness commitment check may ensure continuity of substrate binding across the delegation graph.

[0167] The trilateral binding provided by the evidence witness hash **1104**, the outcome witness hash **1108**, and the prior state hash **1112** may ensure that authority decisions are cryptographically linked to the evidence that existed and outcomes that were computed at the time of authorization. In some cases, any future verifier may simultaneously confirm authority lineage and substrate integrity from a single portable artifact. The cross-system cryptographic witness commitments may enable verification that an authorization decision was consistent with the system state at the time the authorization decision was made.

[0168] Referring to **FIG. 12**, an intent-passage registry **1200** represents an append-only data structure maintained per root authority object **200** for cryptographically binding actions to specific human-authored instruction passages. The intent-passage registry **1200** may comprise canonical encodings of human-authored instruction passages, each passage signed by a registered author principal and assigned an intent passage identifier.

[0169] The intent-passage registry **1200** may comprise entry rows, each entry comprising an intent passage identifier **1202**, a canonical encoding of human-authored passage **1204**, an author principal signature **1206**, and a publication timestamp **1208**. The intent passage identifier **1202** may uniquely identify the passage within the intent-passage registry **1200**. The canonical encoding of human-authored passage **1204** may contain the exact natural-language instruction text authored by a human. In some cases, an entry in the intent-passage registry **1200** may contain a passage such as "Review invoices up to $25,000 and escalate anything larger."

[0170] The author principal signature **1206** may be a post-quantum signature by a registered author principal authenticating the passage content. The author principal signature **1206** may ensure that the passage content was authored by a principal authorized to create instruction passages under the root authority object **200**. The publication timestamp **1208** may record when the passage was added to the intent-passage registry **1200**.

[0171] An inclusion proof **1210** may bind each entry to an intent-passage registry root hash **1212**. The inclusion proof **1210** may enable verification that a specific passage was present in the intent-passage registry **1200** at a specific time. The intent-passage registry root

hash **1212** may be signed by a publishing principal. In some aspects, the intent-passage registry **1200** may be maintained as a Merkle accumulator such that addition of new passages produces a deterministic registry root hash verifiable at logarithmic cost.

[0172] As described with reference to **FIG. 9**, each instruction tag **912** may include an intent passage identifier field **922** identifying a human-authored passage that authored an instruction body. The intent passage identifier field **922** may reference an entry in the intent-passage registry **1200**, binding the instruction tag **912** to the specific human-authored passage that created the instruction content.

[0173] An intent citation **1214** may be included in every action proof. The intent citation **1214** may comprise an intent passage identifier **1216** of each governing passage, a canonical hash of cited passage text **1218**, and a scope-conformance verification binding **1220**. The intent passage identifier **1216** may reference the human-authored passage that authorized the action. The canonical hash of cited passage text **1218** may represent the passage text at the cited epoch, enabling detection of any text drift between the time of passage publication and the time of action execution. The scope-conformance verification binding **1220** may demonstrate that the action's claimed scope is consistent with the plain semantic content of the cited passage.

[0174] In some cases, the intent citation **1214** may further comprise an inclusion proof against the intent-passage registry root hash **1212** in force at action time. The inclusion proof may enable verification that the cited passage was present in the intent-passage registry **1200** when the action was authorized.

[0175] As used herein, the term “scope-conformance verification binding” may refer to a cryptographic data structure that demonstrates an action’s scope is consistent with the plain semantic content of a cited human-authored passage. The scope-conformance verification binding may comprise at least three components: (a) a semantic scope extraction from the cited passage, wherein the semantic scope extraction may be a structured representation of the action types, parameters, limits, and conditions expressed in the human-authored passage; (b) a scope envelope of the proposed action, wherein the scope envelope may be a structured representation of the action type, parameters, and effects of the proposed action; and (c) a cryptographic binding demonstrating that the action scope envelope is a subset of the passage-derived semantic scope. In some cases, the semantic scope extraction may be computed by parsing the cited passage against a scope grammar committed at root certification time. In some aspects, the subset relationship may be verified by evaluating whether every action type in the action scope envelope appears in the semantic scope extraction, whether every parameter value falls within ranges specified in the semantic scope extraction, and whether every effect of the

proposed action is permitted by the semantic scope extraction. The scope-conformance verification binding may be signed by the substrate such that any verifier may confirm the binding was computed correctly from the cited passage and the proposed action.

[0176] The pre-execution conformance gate **114** may deny any action whose intent citation **1214** is missing, references a missing or stale passage, references a passage whose text hash has drifted from the canonical hash of cited passage text **1218**, or fails a scope-consistency rubric. The denial may produce a Negative Authority Proof **700** with a denial-category identifier indicating intent citation missing or intent citation invalid.

[0177] Referring to **FIG. 13**, an authority window composition **1300** illustrates how multi-dimensional temporal constraints are composed at a single authority node and inherited across an ancestor chain in a delegation graph. Each authority node may carry an authority window field expressing a conjunction of temporal scopes. The authority window composition **1300** may enable complex temporal authorization policies governing when actions may be executed.

[0178] At a single authority node, four window-type scopes may be combined by conjunction in a conjunction block **1310**. A clock-window scope **1302** may specify permitted times of day. In some cases, the clock-window scope **1302** may specify a time range such as "09:00-17:00 EST weekdays" with explicit time-zone and daylight-saving disambiguation. A calendar-window scope **1304** may specify permitted calendar periods. In some aspects, the calendar-window scope **1304** may specify a period such as "Q3 FY26" or "fiscal year 2026" as defined in a calendar registry committed at root creation.

[0179] An event-window scope **1306** may be conditional on substrate-observable events. In some cases, the event-window scope **1306** may specify conditions such as “until board approval expires” or “while audit period active.” A condition-window scope **1308** may be conditional on substrate-observable predicates. In some aspects, the condition-window scope **1308** may specify predicates such as “while risk assessment current” or “if compliance certification valid.”

[0180] The conjunction block **1310** may combine the clock-window scope **1302**, the calendar-window scope **1304**, the event-window scope **1306**, and the condition-window scope **1308** to produce a composite authority window at a single node. The authority window field may support composition-window scopes formed by intersection, union, or relative complement of clock, calendar, event, and condition windows, enabling complex temporal authorization policies.

[0181] The distinction between event-window scopes and condition-window scopes may be understood as follows. An event-window scope may be conditional on substrate-observable

events, where an event is a discrete occurrence having a defined start time and end time. In some cases, an event may be a named occurrence such as “audit_window_open,” “board_recess,” or “incident_response_active.” The event may transition between active and inactive states at specific timestamps recorded in the substrate. A condition-window scope may be conditional on substrate-observable predicates, where a predicate is a continuous boolean condition that evaluates to TRUE or FALSE at any given timestamp based on substrate state. In some aspects, a predicate may be a named condition such as “risk_assessment_current,” “compliance_certification_valid,” “board_approval_not_expired,” or “regulatory_filing_complete.” The predicate may be evaluated against substrate state committed in the action proof bundle at action time. The operational distinction is that events are externally signaled state transitions with explicit start and end markers, whereas predicates are computed boolean expressions evaluated against substrate-observable data at the moment of gate evaluation. Both event-window and condition-window evaluation may occur against substrate-observable state committed in the action’s proof bundle, such that a future verifier may independently reproduce the gate’s evaluation without contact with the original substrate.

[0182] The authority window composition **1300** may also illustrate inheritance of authority windows across an ancestor chain. A root node **1312** may contribute an authority window at the top of the delegation chain. A first ancestor **1314** descending from the root node **1312** may contribute an authority window. A second ancestor **1316** descending from the first ancestor **1314** may contribute an authority window. A leaf node **1318** descending from the second ancestor **1316** may represent the point at which an action is evaluated.

[0183] An intersection block **1320** may compute the intersection of authority window fields from every ancestor in the chain. The intersection block **1320** may receive the authority windows from the root node **1312**, the first ancestor **1314**, the second ancestor **1316**, and the leaf node **1318**. The intersection block **1320** may produce an effective authority window **1322**.

[0184] The effective authority window **1322** may represent the intersection of authority window fields of every ancestor from the leaf node **1318** to the root node **1312**. In some cases, if the root node **1312** permits “calendar year 2026,” the first ancestor **1314** permits “Q3-Q4 2026 business hours,” and the second ancestor **1316** permits “September 2026 only,” the effective authority window **1322** may be “September 2026 business hours.”

[0185] As described with reference to **FIG. 5**, the Within Window condition of the four-condition test **500** may verify that an action timestamp falls within the effective authority window **1322**. An action whose timestamp falls outside any ancestor's authority window may

be denied by the pre-execution conformance gate **114** with a Negative Authority Proof **700** carrying a denial-category identifier indicating off window.

[0186] Referring to **FIG. 14**, a root succession **1400** illustrates a method for transitioning authority from a predecessor root authority object to a successor root authority object while maintaining cryptographic continuity. The root succession **1400** may enable organizations to transition authority across events such as CEO transitions, board reconstitutions, mergers and acquisitions, or regulatory changes without breaking cryptographic lineage or losing historical verifiability.

[0187] At step **1402**, a predecessor root authority object exists with a succession authority field. The succession authority field may name principals authorized to sign succession events. In some cases, the succession authority field may be contained within the revocation rules field **214** of the root authority object **200** described with reference to **FIG. 2**. The succession authority field may specify a threshold of principals and a quorum requirement for authorizing succession events.

[0188] At step **1404**, a successor root authority object is created and signed by a post-quantum threshold signature meeting the succession authority field of the predecessor root authority object. The post-quantum threshold signature may comprise a triple-family signature providing independent mathematical hardness assumptions. In some aspects, the post-quantum threshold signature may comprise at least ML-DSA-65, SLH-DSA-SHA2-128f, and FALCON-512 algorithms. The successor root authority object may not become valid until the threshold signature requirement of the predecessor root authority object's succession authority field is satisfied.

[0189] At step **1406**, the successor root authority object carries a predecessor-root-identifier field, a succession event descriptor field, and a succession effective timestamp. The predecessor-root-identifier field may identify the predecessor root authority object by hash. The succession event descriptor field may describe the type of succession event. In some cases, the succession event descriptor field may indicate a succession type such as board change, CEO transition, M&A, regulator change, or organizational restructuring. The succession effective timestamp may indicate when the succession takes effect.

[0190] At step **1408**, the successor root authority object is bound to the predecessor root authority object by a signed succession event placed in an append-only succession registry. The signed succession event may bear a threshold post-quantum signature meeting the predecessor root authority object's succession authority field. The append-only succession registry may create an auditable chain of root transitions. In some aspects, the succession

registry may be mirrored by external parties and may survive the cessation of the original issuer.

[0191] At step **1410**, new delegations made after succession inherit unexpired constraints from the predecessor root authority object via the succession event. The inheritance of unexpired constraints may ensure continuity of policy constraints across the transition. In some cases, new delegations descending from the successor root authority object may be bound by any temporal validity windows, allowed action classes, prohibited action classes, or policy constraints from the predecessor root authority object that remain unexpired at the time of delegation.

[0192] At step **1412**, authority decisions taken under the predecessor root authority object remain verifiable indefinitely after succession. The verifiability of authority decisions under the predecessor root authority object may proceed against the predecessor root authority object's hash regardless of whether the successor root authority object is currently in force. The preservation of historical verifiability may maintain the evidentiary value of historical authorizations across root transitions.

[0193] In some cases, an external party may reconstruct the complete authority lineage of an enterprise across an arbitrary number of root successions by traversing the succession registry. The succession registry may provide a complete audit trail of root transitions from an initial root authority object through every subsequent successor root authority object.

[0194] In some aspects, a successor root authority object may not become operational for a delegated principal until the delegated principal produces an Authority Activation Record acknowledging the successor root authority object. The Authority Activation Record may acknowledge the successor root hash, the predecessor root hash being superseded, and the principal's reconstructed lineage view from the principal's local authority node to the successor root authority object. The pre-execution conformance gate **114** may deny any action by a principal whose acknowledged root hash does not equal the current active root hash for the principal's authority chain, ensuring that principals explicitly acknowledge each root transition before continuing to act under the successor root authority object.

[0195] Referring to **FIG. 15**, an authority state portability diagram **1500** illustrates how authority state may be migrated between heterogeneous workflow execution engines without re-issuance of signatures. The authority state portability diagram **1500** may address the technical problem of authorization artifacts coupled to issuing systems by enabling organizations to replace, decommission, or migrate workflow engines while maintaining full verifiability of historical and ongoing authorization decisions.

[0196] A first workflow execution engine **1502** contains a delegation graph **1504**. The delegation graph **1504** may represent an authority state deployed in a first workflow execution environment. In some cases, the first workflow execution engine **1502** may be an enterprise ticketing system, a business process management platform, or another workflow automation system. The delegation graph **1504** may comprise the root authority object **200** described with reference to **FIG. 2** and a plurality of authority nodes as described with reference to **FIG. 3**.

[0197] A serialize operation **1512** extracts the authority state from the first workflow execution engine **1502** as a portable serialized authority graph **1506**. The portable serialized authority graph **1506** may comprise the root authority object and a plurality of delegation nodes in a canonical encoding. The portable serialized authority graph **1506** may be independent of any specific workflow execution engine. The canonical encoding may preserve all cryptographic bindings including root hash values, parent hash values, node hash values, and post-quantum threshold signatures carried by each authority node in the delegation graph **1504**.

[0198] A de-serialize operation **1514** imports the portable serialized authority graph **1506** into a second workflow execution engine **1508**. The second workflow execution engine **1508** may be a structurally different system from the first workflow execution engine **1502**. In some cases, the second workflow execution engine **1508** may be a governance-risk-compliance platform, a different enterprise ticketing system, or another workflow automation system. The second workflow execution engine **1508** now contains a delegation graph **1510**.

[0199] The delegation graph **1510** in the second workflow execution engine **1508** may be identical to the delegation graph **1504** in the first workflow execution engine **1502**. The delegation graph **1510** may carry identical root hash, parent hash, and node hash values as the delegation graph **1504**. A note **1516** in **FIG. 15** confirms that migration occurs without re-issuance of any signature in the authority graph. The note **1516** further confirms that all cryptographic bindings are preserved and the root hash remains identical before and after migration.

[0200] Preserving all cryptographic bindings during serialization—including root hash values, parent hash values, node hash values, and post-quantum threshold signatures—results in the root hash computed from the delegation graph in the second workflow execution engine being identical to the root hash computed from the delegation graph in the first workflow execution engine, and no principal whose signature appears in the authority graph is required to re-sign.

[0201] The authority state may be expressed as the portable serialized authority graph **1506** comprising the root authority object and the plurality of delegation nodes. The authority

graph may be independent of any specific workflow execution engine. The method may comprise migrating the authority state from the first workflow execution engine **1502** to the second workflow execution engine **1508** by serializing the authority graph from the first workflow execution engine **1502** via the serialize operation **1512** and de-serializing the authority graph into the second workflow execution engine **1508** via the de-serialize operation **1514** without re-issuance of any signature in the authority graph.

[0202] After migration, every authorization decision in the second workflow execution engine **1508** may continue to descend from the root authority object and may satisfy the ancestor constraint aggregate at every level. The ancestor constraint aggregate, as described with reference to **FIG. 4**, may comprise an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints. The pre-execution conformance gate **114** operating within the second workflow execution engine **1508** may verify that every authorization decision continues to descend from the same root authority object that governed the delegation graph **1504** in the first workflow execution engine **1502**.

[0203] In some cases, the migration may be performed in-place as a live cutover. In some aspects, the migration may be performed in a staged manner. In either case, the migration may proceed without re-authorization by any principal whose signature appears in the authority graph. The preservation of signatures across migration may ensure that historical authorization decisions remain verifiable by any third party without contact with the first workflow execution engine **1502** after the first workflow execution engine **1502** has been replaced or decommissioned.

[0204] Referring to **FIG. 16**, a conflict detection and resolution **1600** illustrates a method for handling situations where multiple root authority objects reach divergent verdicts on a proposed action. The conflict detection and resolution **1600** may address scenarios where an action requires authorization from multiple independent authority sources that may not agree on whether the action is permitted.

[0205] At step **1602**, a proposed action touches multiple root authority objects. In some cases, the proposed action may require authorization from both a Finance root and a Legal root. In some aspects, the proposed action may require authorization from both organization-internal roots and organization-external roots. Organization-external roots may include a regulator authority root, an industry consortium root, or an auditor authority root. A wire transfer may require authorization from Finance, Compliance, and Legal roots. A vendor onboarding may require authorization from Procurement, Privacy, Security, and Legal roots. A clinical trial

enrollment may require authorization from Medical, Regulatory, Privacy, Compliance, and Insurance roots. The named roots in multi-root consensus may include both organization-internal roots such as Security and Finance roots and organization-external roots such as regulator authority roots, industry consortium roots, or auditor authority roots.

[0206] At step **1604**, a determination is made whether the contributing root authority objects reach divergent verdicts on the proposed action. As described with reference to **FIG. 1**, the pre-execution conformance gate **114** may detect, at the pre-execution conformance gate **114**, that two or more contributing root authority objects reach divergent verdicts on a proposed action. If the contributing root authority objects agree on the proposed action, the process continues to step **1606** for normal gate processing.

[0207] If divergent verdicts are detected at step **1604**, the process proceeds to step **1608**. At step **1608**, a conflict detection record is emitted. The conflict detection record may comprise identifiers of the contributing root authority objects, conflicting constraints, a canonical encoding of the proposed action, a conflict type classifier, and the evidence supporting each authority's verdict including lineage proofs and cross-substrate witnesses. The conflict type classifier may be selected from verdict divergence, scope envelope intersection empty, temporal window disjoint, constraint irreconcilability, and precedence unset. Verdict divergence may indicate that one root permits and another root denies the proposed action. Scope envelope intersection empty may indicate that the intersection of contributing roots' scope envelopes does not admit the proposed action. Temporal window disjoint may indicate that no time window admits the action under every contributing root's authority window. Constraint irreconcilability may indicate that per-root constraints whose conjunction is unsatisfiable for the proposed action. Precedence unset may indicate that no Conflict Precedent governs the same conflict type involving the same set of roots.

[0208] At step **1610**, a Conflict Resolution Record is received from a resolving authority principal. The Conflict Resolution Record may comprise a hash of the conflict detection record, a resolution decision, an authority basis citation, and a precedence disposition field. The resolution decision may be selected from PERMIT under specified roots, DENY, ESCALATE, or DEFER. The precedence disposition field may be selected from one-time and canonical.

[0209] At step **1612**, the precedence disposition field is evaluated. If the precedence disposition field is one-time, the process proceeds to step **1614**. At step **1614**, the resolution applies to the current conflict and does not establish precedent for future conflicts.

[0210] If the precedence disposition field is canonical at step **1612**, the process proceeds to step **1616**. At step **1616**, a Conflict Precedent is bound to a per-root append-only precedent

registry for application to future conflicts of a same conflict type classifier. The Conflict Precedent may include a precedent predicate comprising substrate-evaluable conditions over risk class, amount, or scope envelope. The Conflict Precedent may also include a supersession_authority field naming principals who may supersede the Conflict Precedent. The precedent predicate may enable the pre-execution conformance gate **114** to apply the Conflict Precedent automatically to future conflicts matching the substrate-evaluable conditions without requiring re-escalation to a human-resolution principal.

[0211] In some cases, a Conflict Precedent may be superseded by a principal meeting the precedent's supersession authority field. The supersession event may be bound as an append-only record. The supersession mechanism may enable organizations to evolve conflict resolution policies over time while maintaining an auditable history of precedent changes.

[0212] In some aspects, a consensus policy for multi-root threshold authority may specify an advisory rather than required role for at least one named root. Where a named root has an advisory role, withholding approval by the advisory root may be recorded in the proof bundle without blocking the action. The advisory role may provide an auditable record of the advisory root's position while permitting the action to proceed based on the remaining required roots.

[0213] In some cases, a meta-consensus root that authorizes evolution of the consensus policy may be federated across the named roots. The federation of the meta-consensus root may enable the named roots to collectively govern changes to the consensus policy itself.

[0214] The process ends at a step **1618**.

[0215] Referring to **FIG. 17**, a root revocation and cascade **1700** illustrates a method for immediate revocation of authority artifacts with cascade through descendant chains. The root revocation and cascade **1700** may address scenarios where an authority artifact requires immediate invalidation due to compromise, mandate withdrawal, or other revocation events.

[0216] At step **1702**, a revoking principal meeting a revocation authority of an authority artifact emits a Root Revocation Record. The Root Revocation Record may comprise a canonical hash of the authority artifact being revoked, a revocation event descriptor, a revocation effective timestamp, and a post-quantum signature. The revocation event descriptor may be selected from compromise detected, mandate withdrawn, regulator directive, internal audit finding, legal injunction, and principal terminated. In some cases, the Root Revocation Record may include an optional re-certification path describing conditions under which a descendant capsule may be re-validated under a successor authority.

[0217] At step **1704**, the Root Revocation Record is bound by inclusion proof to a per-chain revocation registry. The per-chain revocation registry may be implemented as a sparse

Merkle tree providing logarithmic-time inclusion proofs for revoked artifacts and logarithmic-time exclusion proofs for non-revoked artifacts. The sparse Merkle tree structure may enable efficient verification of revocation status without requiring access to the complete revocation registry.

[0218] At step **1706**, at the pre-execution conformance gate **114** for every action, exclusion of every authority artifact along an authority chain of the action from the per-chain revocation registry is verified. The verification may occur in logarithmic time due to the sparse Merkle tree structure of the per-chain revocation registry.

[0219] At step **1708**, a determination is made whether any artifact in the authority chain has a revocation effective timestamp preceding an action timestamp. The determination may evaluate each authority artifact along the authority chain from the leaf action to the root authority object **200**.

[0220] If no revoked artifact is found at step **1708**, the process continues to step **1710** for normal gate processing. The normal gate processing may proceed with evaluation of the remaining conditions of the four-condition test **500** described with reference to **FIG. 5**.

[0221] If a revoked artifact is found at step **1708**, the process proceeds to step **1712**. At step **1712**, the action is denied regardless of whether descendant authority nodes have unexpired validity windows. The revocation may cascade immediately through all descendants of the revoked authority artifact. The immediate cascade may ensure that revocation of an ancestor authority artifact invalidates all descendant authority regardless of the descendants' own temporal validity windows.

[0222] At step **1714**, a Negative Authority Proof **700** is emitted with a denial-category identifier indicating revoked authority. The Negative Authority Proof **700** may serve as portable evidence that the pre-execution conformance gate **114** refused the action due to a revoked authority artifact in the authority chain.

[0223] Referring to **FIG. 18**, an override authorization **1800** illustrates a method for higher-order authority principals to override denied actions through governed channels. The override authorization **1800** may address scenarios where an action denied by the pre-execution conformance gate **114** requires authorization from a principal with higher-order authority committed by an ancestor root authority object.

[0224] At step **1802**, an original action is denied by the pre-execution conformance gate **114**. The denial may produce a Negative Authority Proof **700** as described with reference to **FIG. 7**. The Negative Authority Proof **700** may identify the failing condition and the denial-category identifier that caused the original denial.

[0225] At step **1804**, a higher-order authority principal whose authority is committed by an ancestor root authority object produces an override authorization record. The override authorization record may comprise a hash of an original denial identifying exactly what was denied, an override decision, an override reason field, a higher-order authority basis citation, and an expiration condition. The hash of the original denial may reference the Negative Authority Proof **700** produced at step **1802**. The override decision may indicate whether the higher-order authority principal authorizes the originally denied action. The override reason field may comprise free-text or a citation into a root-published override-rationale taxonomy. The higher-order authority basis citation may reference the ancestor root artifact authorizing the higher-order authority principal to perform overrides. The expiration condition may be selected from one-time, time-bounded, and condition-bounded. The one-time expiration condition may indicate that the override applies to the single action identified by the hash of the original denial. The time-bounded expiration condition may include an end timestamp at which the override expires. The condition-bounded expiration condition may include a substrate-evaluable predicate, and the override may expire when the predicate becomes false.

[0226] At step **1806**, the pre-execution conformance gate **114** verifies the higher-order authority basis citation. The verification may confirm that the overriding principal is authorized by an ancestor root authority object to perform overrides of the type indicated by the override authorization record.

[0227] At step **1808**, a determination is made whether the higher-order authority basis is valid. If the higher-order authority basis is invalid, the process proceeds to step **1810**. At step **1810**, the override is denied and the original denial stands. The originally denied action remains denied, and no Override Portable Authority Proof is emitted.

[0228] If the higher-order authority basis is valid at step **1808**, the process proceeds to step **1812**. At step **1812**, an Override Portable Authority Proof is emitted. The Override Portable Authority Proof may comprise five components. A first component (α) may comprise an indication of what would have been denied. A second component (β) may comprise an indication of what was authorized in place of the denied action. A third component (γ) may comprise an identification of the higher-order authority principal who authorized the override. A fourth component (δ) may comprise the expiration condition governing the duration of the override. A fifth component (ε) may comprise a signed trail of an override event creating a complete audit record. The five-component format of the Override Portable Authority Proof may distinguish the override authorization **1800** from override-logging approaches that capture

that an override occurred without capturing the complete authorization basis and expiration conditions.

[0229] Referring to **FIG. 19**, a RENEWABLE mode enforcement **1900** illustrates a method for enforcing periodic governance review on operating-policy authority. The RENEWABLE mode enforcement **1900** may address scenarios where root authority objects representing operating-policy authority such as invoice approval policies, vendor onboarding procedures, or spending limits require periodic review to maintain validity.

[0230] At step **1902**, a root authority object has a lifecycle mode indicator **204** of RENEWABLE. As described with reference to **FIG. 2**, the lifecycle mode indicator **204** may be selected from a closed set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL. The RENEWABLE mode may represent operating-policy authority carrying a renewal period and a next review due timestamp and requiring periodic Root Refresh Records.

[0231] At step **1904**, a certification authority emits a Root Refresh Record. The Root Refresh Record may comprise an unchanged root hash, a differential summary confirming no material modification, a re-displayed intent summary with display proofs, certifying principal signatures, a refresh effective timestamp, and a next review due timestamp. The unchanged root hash may be preserved rather than changed, distinguishing the Refresh operation from the Succession operation described with reference to **FIG. 14**. The differential summary may comprise a machine-checked differential summary signed by the substrate confirming no material modification relative to a prior certification or refresh record. The re-displayed intent summary with display proofs may confirm that the intent was re-shown to certifying principals during the refresh process. The certifying principal signatures may authenticate the refresh event. The refresh effective timestamp may indicate when the refresh takes effect. The next review due timestamp may indicate when the next periodic review is due. In some cases, the Root Refresh Record may commit, at original Certification time, two anti-drift caps: a max_refresh_count specifying the maximum total number of Refresh Records the substrate will accept for the root before requiring a new Certification Ceremony, and a max_refresh_age specifying the maximum total elapsed time from original certification under which Refresh Records may be bound before requiring a new Certification Ceremony. In some cases, the pre-execution conformance gate **114** may deny any action under a RENEWABLE root authority object whose most recent Root Refresh Record has a next review due timestamp in the past, or whose refresh count has exceeded max_refresh_count, or whose elapsed time since certification has exceeded max_refresh_age, with a denial-category identifier indicating governance review missed or refresh drift exceeded.

[0232] At step **1906**, the Root Refresh Record is bound to a per-root refresh registry. The per-root refresh registry may be maintained by the registry complex **118** described with reference to **FIG. 1**. The binding of the Root Refresh Record to the per-root refresh registry may create an auditable chain of periodic governance reviews for the RENEWABLE root authority object.

[0233] As used herein, the term "material modification" may refer to any change to a root authority object that affects the substrate's evaluation semantics. Material modification may include any change to the root's objective statement, approval set, allowed action classes, prohibited action classes, allowed data classes, policy references, revocation rules, certification authority, lifecycle mode indicator, or any other schema field that changes how the pre-execution conformance gate evaluates proposed actions against the root authority object. In contrast, non-material changes may include cosmetic changes such as display formatting adjustments or summary text re-rendering that hash to the same canonical content as the prior version. The differential summary may be computed by a substrate-signed deterministic procedure that compares the current root authority object against the most recent prior Root Certification Record or prior Root Refresh Record. The differential summary may comprise a machine-checked comparison of each schema field, with the substrate attesting that no material modification has occurred when all evaluation-affecting fields remain unchanged. Where the certification authority elects to materially modify the root, the substrate may fall through to Root Succession rather than emitting a Root Refresh Record, thereby producing a successor root with a new root hash and requiring downstream Authority Activation Records.

[0234] At step **1908**, at the pre-execution conformance gate **114**, for every action under a RENEWABLE root authority object, the pre-execution conformance gate **114** verifies the most recent Root Refresh Record. The verification may occur as part of the Lifecycle Compliant condition of the four-condition test **500** described with reference to **FIG. 5**.

[0235] At step **1910**, a determination is made whether the next review due timestamp in the most recent Root Refresh Record is in the past relative to an action timestamp. The determination may compare the next review due timestamp from the most recent Root Refresh Record against the action timestamp of the proposed action.

[0236] If the review is current such that the next review due timestamp is in the future relative to the action timestamp, the process continues to step **1912** for normal gate processing. The normal gate processing may proceed with evaluation of the remaining conditions of the four-condition test **500**.

[0237] If the review is overdue such that the next review due timestamp is in the past relative to the action timestamp, the process proceeds to step **1914**. At step **1914**, the action is denied and a governance review missed Negative Authority Proof **700** is emitted. The governance review missed Negative Authority Proof **700** may be a cryptographic proof of the absence of a governance event that was due. The governance review missed Negative Authority Proof **700** may serve as proof that a required periodic review did not occur. The governance review missed Negative Authority Proof **700** may be distinct from audit logs that record events that did occur. The governance review missed Negative Authority Proof **700** may represent a proof primitive for absences.

[0238] The Refresh primitive may be distinct from the Succession primitive described with reference to **FIG. 14**. The Refresh primitive may preserve the root hash and may require no downstream re-acknowledgment by recipient principals. The Succession primitive may change the root hash and may require recipient re-activation via Authority Activation Records **800** as described with reference to **FIG. 8**.

[0239] By preserving the root hash rather than changing it, the Root Refresh Record leaves all dependent delegation nodes, instruction publications, Authority Activation Records, and cross-system cryptographic witness commitments that reference the root hash valid across the refresh without requiring downstream re-acknowledgment.

[0240] Referring to **FIG. 20**, a PROJECT and EPHEMERAL retirement **2000** illustrates a method for planned retirement of mission-order authority and emergency authority. The PROJECT and EPHEMERAL retirement **2000** may address scenarios where root authority objects representing time-bounded or goal-bounded authority reach a natural end of lifecycle and require governed retirement with closure artifacts and residual obligation tracking.

[0241] At step **2002**, a root authority object has a lifecycle mode indicator **204** of PROJECT or EPHEMERAL. As described with reference to **FIG. 2**, the lifecycle mode indicator **204** may be selected from a closed set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL. PROJECT mode may represent mission-order authority carrying a project binding with completion criteria. In some cases, the completion criteria may specify conditions such as “complete mobile app build” or “conduct Q3 audit.” EPHEMERAL mode may represent emergency authority carrying a hard expiry timestamp. In some aspects, the hard expiry timestamp may specify a duration such as “incident response for 72 hours.”

[0242] At step **2004**, a determination is made whether completion criteria have been met for PROJECT mode or a hard expiry timestamp has elapsed for EPHEMERAL mode. For PROJECT mode, the determination may evaluate whether a completion criteria predicate has

become TRUE. For EPHEMERAL mode, the determination may evaluate whether the current timestamp exceeds the hard expiry timestamp.

[0243] If neither retirement condition is met at step **2004**, the process continues to step **2006** for normal gate processing. The normal gate processing may proceed with evaluation of the four-condition test **500** described with reference to **FIG. 5**.

[0244] When retirement conditions are met at step **2004**, the process proceeds to step **2008**. At step **2008**, a Root Retirement Record is emitted. The Root Retirement Record may comprise a retired root hash, a retirement event descriptor, a retirement effective timestamp, a closure artifact summarizing lifetime activity, and a residual obligations marker naming obligations persisting beyond retirement.

[0245] The retired root hash may identify the root authority object being retired by hash. The retirement event descriptor may be selected from project completed, project cancelled, ephemeral expired, ephemeral purpose satisfied, and certification authority initiated. The project completed descriptor may indicate that a PROJECT root reached completion criteria. The project cancelled descriptor may indicate that a PROJECT root was terminated prior to completion. The ephemeral expired descriptor may indicate that an EPHEMERAL root reached hard expiry timestamp. The ephemeral purpose satisfied descriptor may indicate that an EPHEMERAL root's purpose was fulfilled prior to hard expiry. The certification authority initiated descriptor may indicate that a certification authority initiated retirement.

[0246] The retirement effective timestamp may indicate when the retirement takes effect. The closure artifact may summarize lifetime activity under the retiring root authority object. The closure artifact summarizing lifetime activity may comprise a structured data record including at least: (a) a count of Portable Authority Proofs emitted under the root authority object over its lifetime; (b) a count of Negative Authority Proofs emitted under the root authority object; (c) identifiers of every Conflict Resolution Record and every Conflict Precedent that governed the root authority object; (d) identifiers of every Override Capsule emitted under the root authority object and the Override Portable Authority Proofs they produced; (e) derived delegation hashes identifying every delegation node descended from the root authority object; (f) identifiers of every cited intent-passage registry passage referenced by actions under the root authority object; (g) summary statistics of cross-substrate witness commitments accumulated under the root authority object, including counts of evidence witness hashes and outcome witness hashes bound during the root’s lifetime; and (h) a substrate integrity attestation comprising a triple-family post-quantum signature over the foregoing elements. The closure artifact may be structured as a canonical encoding with deterministic

serialization such that any verifier may independently compute the same closure artifact hash from the underlying data.

[0247] The residual obligations marker may name obligations created under the retired root authority object that persist beyond retirement. In some cases, the residual obligations marker may name audit-retention duties. In some aspects, the residual obligations marker may name regulatory reporting obligations. The residual obligations marker may name evidence-preservation duties under H33-74. The residual obligations marker may name outcome-preservation duties under Cachee. The residual obligations marker may name continuing third-party rights whose interests were established under the root authority object. The residual obligations marker may also name in-flight Conflict, Override, or Refresh markers that were pending at retirement time.

[0248] At step **2010**, the Root Retirement Record is bound to a retirement registry. The retirement registry may be maintained by the registry complex **118** described with reference to **FIG. 1**. The binding of the Root Retirement Record to the retirement registry may create an auditable record of root authority object retirement.

[0249] At step **2012**, at the pre-execution conformance gate **114**, for every action after retirement, the pre-execution conformance gate **114** checks whether the root authority object is retired. The check may occur as part of the Lifecycle Compliant condition of the four-condition test **500** described with reference to **FIG. 5**.

[0250] At step **2014**, a determination is made whether the action invokes a listed residual obligation. The determination may evaluate whether the proposed action corresponds to an obligation named in the residual obligations marker of the Root Retirement Record.

[0251] If the action invokes a listed residual obligation at step **2014**, the process proceeds to step **2016**. At step **2016**, the action is permitted for the residual obligation. The permission may be limited to the specific residual obligation invoked by the action.

[0252] If the action does not invoke a listed residual obligation at step **2014**, the process proceeds to step **2018**. At step **2018**, the action is denied and a retired authority escalation Negative Authority Proof **700** is emitted. The retired authority escalation Negative Authority Proof **700** may identify the retired root authority object, the retirement event descriptor, and the residual obligations marker. The retired authority escalation Negative Authority Proof **700** may enable a verifier to distinguish a post-retirement action attempting to use retired authority from a post-retirement action correctly invoking a residual obligation.

[0253] The Retirement primitive may be distinct from the Revocation primitive described with reference to **FIG. 17**. Retirement may be a planned end of lifecycle carrying a closure

artifact and residual obligations. Revocation may be an unplanned termination carrying a revocation event descriptor for compromise or mandate withdrawal. The distinction between Retirement and Revocation may enable different handling of planned lifecycle completion versus unplanned authority invalidation.

[0254] Referring to **FIG. 21**, a Root Certification Ceremony **2100** illustrates a method for establishing a cryptographic genesis-layer legal bridge between human intent and a root authority object. The Root Certification Ceremony **2100** may be performed at root creation to ensure that no root authority object becomes operational without cryptographic evidence that human principals reviewed and approved the root authority object.

[0255] At step **2102**, a human intent statement is received from a proposing principal. The human intent statement may contain a full canonical statement of authorized intent. The human intent statement may serve as the substrate-canonical reference for the root authority object being certified.

[0256] In some aspects, the Root Certification Ceremony **2100** may include a Pre-Certification Conflict Scan prior to accepting threshold approval signatures. The Pre-Certification Conflict Scan may compare a proposed root envelope against PERMANENT roots, active roots with intersecting envelopes, and active Conflict Precedents. The proposed root envelope may be composed from allowed action classes, prohibited action classes, allowed data classes, authority windows, lifecycle mode, and mode-specific fields. The Pre-Certification Conflict Scan may produce a Pre-Cert Conflict Scan Record comprising three categories of findings. A first category may comprise FATAL constitution-level conflicts that block certification. A second category may comprise NOTICE precedent matches indicating that future runtime conflicts will be auto-resolved per matched precedents. A third category may comprise ADVISORY potential-conflict intersections indicating that runtime conflicts will produce Conflict Capsules. The Pre-Cert Conflict Scan Record may be bound to the Root Certification Record such that no root authority object may be certified absent a present, valid, signed Pre-Cert Conflict Scan Record and such that no FATAL conflicts are present. The Pre-Certification Conflict Scan is architecturally distinct from runtime conflict resolution: runtime conflict resolution fires at the pre-execution gate after both roots are already valid, whereas Pre-Certification Conflict Scan fires at the Root Certification Ceremony and prevents contradictory authority structures from ever becoming valid. The two security models are complementary—Pre-Certification Conflict Scan prevents contradictory authority structures from ever existing, while runtime conflict resolution stops contradictory actions when both roots are already valid.

[0257] At step **2104**, a human-authored summary distinct from the human intent statement is received. The human-authored summary may provide a concise rendering for display to approving principals. The human-authored summary may be distinct from the full human intent statement, with the summary being displayed to approving principals and the full text serving as the substrate-canonical reference.

[0258] At step **2106**, for each approving principal, a display proof is generated. The display proof may comprise a cryptographic commitment to an exact rendering shown to the approving principal. In some cases, the display proof may comprise a Merkle commitment over a canonical encoding of a displayed UI state. The displayed UI state may include the summary text shown, the navigation context, the action affordances offered, and the time of display. The display proof may be bound to the approving principal's session such that no approving principal may later claim to have been shown a different summary or rendering. The display proof may be a portable artifact independently verifiable by any third party at any future time.

[0259] At step **2108**, a post-quantum threshold signature is obtained from approving principals. The post-quantum threshold signature may be obtained over a full intent hash, a summary hash, a proposed root hash, and display proof hashes. In some aspects, the post-quantum threshold signature may comprise a triple-family signature providing independent mathematical hardness assumptions. The approving principals may be members of a certification authority threshold set named in a certification authority field of the proposed root authority object.

[0260] At step **2110**, a Root Certification Record is emitted. The Root Certification Record may be bound by inclusion proof to a Root Certification Registry. The Root Certification Registry may be anchored at substrate genesis. The Root Certification Record may comprise the canonical encoding of the human intent statement, the human-authored summary, the full intent hash, the summary hash, the proposed root hash, the display proof hashes, identifiers and registered signing-key hashes of every approving principal, the post-quantum threshold signature, a certification event timestamp, and cross-system cryptographic witness commitments.

[0261] At step **2112**, the pre-execution conformance gate **114** denies any action whose authority chain terminates at a root authority object lacking a valid Root Certification Record. The denial may produce a Negative Authority Proof **700** with a denial-category identifier indicating certification missing or certification invalid. The enforcement at the pre-execution conformance gate **114** may ensure that no action may be authorized under a root authority object that was not certified through the Root Certification Ceremony **2100**.

[0262] In some cases, the substrate may prove that principals were shown the display proof and signed the threshold signature. The substrate may not claim that principals understood the content shown. The discipline of proving operational facts rather than mental states may apply across the authority preservation system **100**.

[0263] Referring to **FIG. 22**, a principal membership transition **2200** illustrates a method for handling personnel changes affecting registered roles while a root authority object, policy constraints, and delegations remain unchanged. The principal membership transition **2200** may address scenarios where personnel changes such as resignations, terminations, retirements, role changes, or reorganizations require updating the binding between principal identities and roles without modifying the underlying authority structure.

[0264] At step **2202**, a per-root principal-position registry is maintained binding principal identities to roles. Each entry in the per-root principal-position registry may comprise a principal identifier, a signing-key hash, a role descriptor, and effective timestamps. In some cases, the role descriptor may indicate a role such as “Finance Manager,” “Accounts Payable Agent,” or “Treasury Analyst.” The effective timestamps may indicate when the principal's binding to the role became active and when the binding expires or was terminated.

[0265] At step **2204**, a personnel change occurs. The personnel change may be selected from resignation, termination, retirement, role change, and reorganization. In some cases, the personnel change may involve a first principal leaving a role and a second principal replacing the first principal in the role. For example, a first principal named John may leave the Finance Manager role and a second principal named Sarah may replace John in the Finance Manager role.

[0266] At step **2206**, a transition authority for the role emits a Principal Membership Transition Record. The Principal Membership Transition Record may comprise a predecessor entry hash, a successor entry hash, an unchanged root hash, an authority basis citation, a transition event descriptor, and a transition effective timestamp. The predecessor entry hash may identify the registry entry of the departing principal, such as John's registry entry. The successor entry hash may identify the registry entry of the incoming principal, such as Sarah's registry entry. The unchanged root hash may confirm that the root authority object is not changed by the transition. The authority basis citation may reference the authority under which the transition is performed. The transition event descriptor may describe the type of personnel change. The transition effective timestamp may indicate when the transition takes effect.

[0267] At step **2208**, at the pre-execution conformance gate **114**, for every action, the pre-execution conformance gate **114** verifies role-continuity. The role-continuity verification may occur as part of the four-condition test **500** described with reference to **FIG. 5**.

[0268] At step **2210**, a determination is made whether the acting principal corresponds to a valid principal-position registry entry at an action timestamp and whether no Principal Membership Transition Record bearing the acting principal as predecessor exists at or before the action timestamp. The determination may evaluate two conditions: first, that the acting principal has a valid registry entry at the action timestamp; and second, that no Principal Membership Transition Record has been filed naming the acting principal as the predecessor at or before the action timestamp.

[0269] If the determination at step **2210** is affirmative, the process continues to step **2212** for normal gate processing. The normal gate processing may proceed with evaluation of the remaining conditions of the four-condition test **500**.

[0270] If the determination at step **2210** is negative, the process proceeds to step **2214**. At step **2214**, the action is denied. In some cases, the action may be denied when a former principal attempts to act under a role after a Principal Membership Transition Record was filed naming the former principal as predecessor. For example, if John attempts to act under the Finance Manager role after the Principal Membership Transition Record was filed naming John as predecessor, the action may be denied at step **2214**. The denial may produce a Negative Authority Proof **700** with a denial-category identifier indicating principal role orphan.

[0271] The principal membership transition **2200** may be distinct from root succession described with reference to **FIG. 14**, authority revocation described with reference to **FIG. 17**, and authority activation described with reference to **FIG. 8**. Root succession may change the root hash, whereas the principal membership transition **2200** preserves the unchanged root hash. Authority revocation may represent an unplanned termination of authority, whereas the principal membership transition **2200** may represent a planned personnel change. Authority activation may address freshness of acknowledgment, whereas the principal membership transition **2200** may address continuity of role binding across personnel changes.

[0272] Referring to **FIG. 23**, a subtree freeze diagram **2300** illustrates selective revocation of a subtree within the delegation graph **300** without cascading to sibling subtrees. The subtree freeze diagram **2300** may provide granular revocation capability enabling targeted invalidation of a specific delegation path while permitting the remainder of an organization to continue operating.

[0273] A root node **2302** at the top of the subtree freeze diagram **2300** is marked ACTIVE and has three level-one authority nodes. An authority node **2304** descends from the root node **2302** and is active. An authority node **2306** descends from the root node **2302** and is the target node marked TARGET—FROZEN. An authority node **2308** descends from the root node **2302** and is active.

[0274] The authority node **2304** has active descendant nodes comprising an authority node **2310** and an authority node **2312**. The authority node **2306** has frozen descendant nodes comprising an authority node **2314** and an authority node **2316**. The authority node **2308** has active descendant nodes comprising an authority node **2318** and an authority node **2320**.

[0275] A subtree freeze instruction **2322**, shown as a dashed arrow in **FIG. 23**, is directed at the authority node **2306** from a freeze authority principal **2324**. The freeze authority principal **2324** may be a principal listed in the subtree freeze authority **214a** of the root authority object **200** described with reference to **FIG. 2**. The subtree freeze instruction **2322** may identify a target node within the delegation graph from a principal listed in a subtree freeze authority of the root authority object.

[0276] Upon receiving the subtree freeze instruction **2322**, the freeze authority principal **2324** produces a signed freeze event **2326**. The signed freeze event **2326** may identify the target node, which in the subtree freeze diagram **2300** is the authority node **2306**. The signed freeze event **2326** is published to a revocation registry **2328**. The revocation registry **2328** may be an append-only revocation registry. Publishing the signed freeze event **2326** to the append-only revocation registry **2328** may create an auditable record of the subtree freeze operation.

[0277] A note **2330** in **FIG. 23** confirms that revocation of the subtree at the authority node **2306** does not cascade to sibling subtrees at the authority node **2304** or the authority node **2308**. The sibling subtrees at the authority node **2304** and the authority node **2308** may continue normal operation. The selective nature of the subtree freeze may enable targeted revocation without disrupting unaffected portions of the delegation graph **300**.

[0278] The pre-execution conformance gate **114** may enforce, for any action involving a descendant of the target node, rejection with a revocation-blocked artifact. In the subtree freeze diagram **2300**, any action involving the authority node **2314** or the authority node **2316** may be rejected with a revocation-blocked artifact because the authority node **2314** and the authority node **2316** are descendants of the frozen authority node **2306**.

[0279] The pre-execution conformance gate **114** may permit, for any action not involving a descendant of the target node, normal continuation of authority such that revocation of one subtree does not cascade to sibling subtrees. In the subtree freeze diagram **2300**, actions

involving the authority node **2310**, the authority node **2312**, the authority node **2318**, or the authority node **2320** may proceed with normal gate processing because those authority nodes are not descendants of the frozen authority node **2306**.

[0280] In some cases, the subtree freeze may address scenarios where a department is compromised or a delegation path is discovered to be improper. The specific subtree associated with the compromised or improper delegation path may be frozen while the rest of the organization continues operating under the remaining active subtrees.

[0281] In some aspects, the freeze event may be reversible via a signed reinstate event meeting the original root's can revoke quorum. The reversibility of the freeze event may enable restoration of authority to a previously frozen subtree when the conditions that prompted the freeze have been resolved.

[0282] Referring to **FIG. 24**, a structured authority rationale record **2400** illustrates the programmatic derivation of plain-language explanations from cryptographic citations. The structured authority rationale record **2400** may enable any auditor, regulator, court, or counterparty to understand why an action was authorized by examining a plain-language record and then verifying each statement against underlying cryptographic citations.

[0283] A lineage proof **2402** from the Portable Authority Proof **600** described with reference to **FIG. 6** may be processed to extract a delegation chain description **2404**. The delegation chain description **2404** may produce text describing the authority path. In some cases, the delegation chain description **2404** may produce text such as "Finance Manager → AP Agent" describing the chain of delegation from a delegating principal to an acting principal.

[0284] An Authority Activation Record **2406** may be processed to extract an activation state description **2408**. The activation state description **2408** may produce text confirming freshness of the acting principal's acknowledgment. In some aspects, the activation state description **2408** may produce text such as "Activated 2026-07-01; acknowledged root matches current active root" confirming that the acting principal acknowledged the governing authority object and that the acknowledged root hash matches the current active root hash.

[0285] An intent citation **2410** from the intent-passage registry **1200** described with reference to **FIG. 12** may be processed to extract a cited intent passage **2412**. The cited intent passage **2412** may produce the literal human-authored sentence that authorized the action. In some cases, the cited intent passage **2412** may produce text such as "Review invoices up to $25,000 and escalate anything larger" representing the exact natural-language instruction authored by a human principal.

[0286] Authority window data **2414** from the ancestor chain may be processed to extract an authority window description **2416**. The authority window description **2416** may produce text describing the temporal scope under which the action was authorized. In some aspects, the authority window description **2416** may produce text such as “Q3 FY26, weekdays 09:00-17:00 EST, until board approval expires” describing the intersection of clock-window scopes, calendar-window scopes, and event-window scopes from the ancestor chain.

[0287] Acting principal data **2418** may be processed to extract an acting principal identifier **2420**. The acting principal identifier **2420** may produce text identifying the principal performing the action. In some cases, the acting principal identifier **2420** may produce text such as "Invoice Approval Agent" identifying the specific agent or human actor that executed the action.

[0288] Proposed action data **2422** may be processed to extract an action summary **2424**. The action summary **2424** may produce text describing the action performed. In some aspects, the action summary **2424** may produce text such as "Approved Invoice #8421 for $14,200 to ACME Corp" summarizing the specific action that was authorized and executed.

[0289] All six output fields—the delegation chain description **2404**, the activation state description **2408**, the cited intent passage **2412**, the authority window description **2416**, the acting principal identifier **2420**, and the action summary **2424**—may converge into a structured authority rationale record **2426**. The structured authority rationale record **2426** may answer the question "Why did this agent think it was allowed?" in natural language suitable for auditors, courts, regulators, insurers, boards, and counterparties.

[0290] A note **2428** in **FIG. 24** confirms that the structured authority rationale record **2426** is assembled programmatically from cryptographic citations rather than being free-form text. The structured authority rationale record **2426** is not a free-text annotation but rather a deterministic assembly from cited cryptographic artifacts via a signed mapping. Any party may verify that the natural-language rendering corresponds to the cited cryptographic artifacts by checking each field of the structured authority rationale record **2426** against the underlying cryptographic citation from which the field was derived.

[0291] In some cases, the programmatic assembly of the structured authority rationale record **2426** may enable verification that the plain-language explanation accurately reflects the cryptographic state at the time of action authorization. The signed mapping between cryptographic citations and natural-language descriptions may prevent discrepancies between what the cryptographic artifacts attest and what the plain-language record states.

[0292] Referring to **FIG. 25**, an authority preservation method **2500** illustrates a method integrating the components described with reference to **FIG. 1** through **FIG. 24**. The authority preservation method **2500** may produce independently verifiable proof bundles via hash-linked delegation graphs and post-quantum threshold signatures.

[0293] The authority preservation method **2500** begins at step **2502**. At step **2504**, a root authority object is created representing threshold-signed human-authored objective. The root authority object may be the root authority object **200** described with reference to **FIG. 2**. The root authority object may be created and stored in the root authority object store **110** described with reference to **FIG. 1**. The root authority object may comprise at least an objective statement, a lifecycle mode indicator selected from a set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules. The root authority object may be cryptographically bound by a post-quantum threshold signature meeting the approval set. In some cases, the post-quantum threshold signature may comprise a triple-family signature providing independent mathematical hardness assumptions. In some aspects, the root authority object may be certified through a Root Certification Ceremony as described with reference to **FIG. 21** prior to becoming operational.

[0294] At step **2506**, for each authority node in a delegation graph descended from the root authority object, an ancestor constraint aggregate is computed as a function of every ancestor node from an immediate parent up to and including the root authority object. The delegation graph may be the delegation graph **300** described with reference to **FIG. 3**. The ancestor constraint aggregate may be computed by the delegation graph engine **112** described with reference to **FIG. 1**. The ancestor constraint aggregate may comprise at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints. The ancestor constraint aggregate computation may follow the process described with reference to **FIG. 4**.

[0295] At step **2508**, a proposed action is received from an acting principal. The proposed action may be received through one of the interfaces described with reference to **FIG. 1**. In some cases, the proposed action may be received through the human principal interface **120** when the acting principal is a human actor. In some aspects, the proposed action may be received through the agent runtime interface **122** when the acting principal is an autonomous software agent. In some cases, the proposed action may be received through the workflow engine interface **124** when the acting principal is an integrated system or workflow component.

The proposed action may comprise a canonical encoding of the action to be performed, an identifier of the acting principal, and an action timestamp.

[0296] At step **2510**, at the pre-execution conformance gate **114**, a four-condition test is enforced prior to execution of the proposed action. The four-condition test may be the four-condition test **500** described with reference to **FIG. 5**. The four-condition test may comprise: (i) an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object, the governing authority object being one of a root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact; (ii) a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; (iii) a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and (iv) a Lifecycle Compliant condition verifying that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp.

[0297] At step **2512**, a determination is made whether all four conditions of the four-condition test pass.

[0298] If all four conditions pass at step **2512**, the authority preservation method **2500** proceeds to step **2514**. At step **2514**, the proof generation module **116** produces a Portable Authority Proof. The Portable Authority Proof may be the Portable Authority Proof **600** described with reference to **FIG. 6**. The Portable Authority Proof may comprise a lineage proof from the action to the root authority object, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a human-authored passage from the intent-passage registry **1200** described with reference to **FIG. 12**, and a structured authority rationale record assembled programmatically from the citations. The structured authority rationale record may be assembled as described with reference to **FIG. 24**. The Portable Authority Proof may be independently verifiable by any third party without contact with the system that produced the Portable Authority Proof.

[0299] If any condition of the four-condition test fails at step **2512**, the authority preservation method **2500** proceeds to step **2516**. At step **2516**, the proof generation module **116** produces a Negative Authority Proof. The Negative Authority Proof may be the Negative Authority Proof **700** described with reference to **FIG. 7**. The Negative Authority Proof may identify the failing condition and a denial-category identifier. The denial-category identifier may indicate which of the four conditions failed. In some cases, the denial-category identifier

may indicate a specific contingency condition such as stale authority, revoked authority, governance review missed, retired authority, off window, or intent citation missing. The Negative Authority Proof may serve as portable evidence that the pre-execution conformance gate **114** refused the proposed action, naming the specific failing check.

[0300] The authority preservation method **2500** ends at step **2518**.

[0301] In some aspects, the authority preservation method **2500** may support arbitrary delegation depth across heterogeneous principal categories including human actors, autonomous software agents, integrated systems, and cross-organizational workflows. The proof bundles produced by the authority preservation method **2500** may remain independently verifiable for at least the cryptographic lifetime of the post-quantum signature algorithms used at the root authority object.

[0302] The author binding mechanism and instruction source isolation may together provide defense against poisoned-instruction attacks. The author binding mechanism may require that each instruction tag include an author signature field carrying a post-quantum signature by a registered author principal over a canonical encoding of a decrypted instruction body. The author principal may be distinct from a publishing principal. The separation between the author principal and the publishing principal may provide that an attacker who compromises the publishing principal cannot author instruction content, and an attacker who compromises an author principal cannot distribute instructions.

[0303] In some cases, a Policy Author principal may author an instruction such as “Review invoices up to $25,000 and escalate anything larger” while a separate Publishing Principal authorizes distribution to specific agents. The pre-execution conformance gate may verify both signatures. The pre-execution conformance gate may verify that an amendment lineage of cited tags was Q-Sign-evaluated. The pre-execution conformance gate may verify that the author signature verifies under a registered author whose registry entry was valid at content-signing time. The pre-execution conformance gate may verify that the action derives authority solely from cited instruction tags.

[0304] Instruction source isolation at an agent runtime may require that the agent runtime treat as authoritative only text carrying a verifiable instruction-tag citation with a valid read receipt and traceable lineage to an authorized publishing principal. All other text observed by the agent runtime from tool call responses, retrieved corpus documents, user messages, other agent messages, or environment artifacts may be treated as non-authoritative input data that is excluded from action authorization. The quarantine of non-authoritative text may ensure that

an attacker cannot inject instructions via side channels such as prompt injection in tool responses or poisoned retrieval-augmented-generation corpora.

[0305] In some aspects, the agent runtime may produce a signed source-isolation-violation artifact upon detection of a side-channel injection attempt. The signed source-isolation-violation artifact may provide evidence of the attack for audit and remediation. The signed source-isolation-violation artifact may comprise an identifier of the detected injection attempt, a timestamp of detection, a classification of the side-channel type, and a post-quantum signature by the agent runtime.

[0306] Machine readable storage including machine-readable instructions, when executed, to implement a method or realize an apparatus for cryptographic authority preservation across delegation chains in any of the examples of the present application.

[0307] Various techniques, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, a non-transitory computer readable storage medium, or any other machine-readable storage medium wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the various techniques. In the case of program code execution on programmable computers, the computing device may include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. The volatile and non-volatile memory and/or storage elements may be a RAM, an EPROM, a flash drive, an optical drive, a magnetic hard drive, or another medium for storing electronic data. The Authority Preservation System and associated components may also include a transceiver component, a counter component, a processing component, and/or a clock component or timer component for coordinating temporal constraints and authority window evaluations. One or more programs that may implement or utilize the various techniques described herein may use an application programming interface (API), reusable controls, and the like. Such programs may be implemented in a high-level procedural or an object-oriented programming language to communicate with a computer system. However, the program(s) may be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or an interpreted language, and combined with hardware implementations.

[0308] It should be understood that many of the functional units described in this specification may be implemented as one or more components, which is a term used to more particularly emphasize their implementation independence. For example, a component such as the root authority object store, the delegation graph engine, the pre-execution conformance

gate, or the proof generation module may be implemented as a hardware circuit comprising custom very large scale integration (VLSI) circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A component may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices, or the like.

[0309] Components may also be implemented in software for execution by various types of processors. An identified component of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions, which may, for instance, be organized as an object, a procedure, or a function. Nevertheless, the executables of an identified component need not be physically located together, but may comprise disparate instructions stored in different locations that, when joined logically together, comprise the component and achieve the stated purpose for the component.

[0310] Indeed, a component of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data such as root authority objects, delegation graphs, ancestor constraint aggregates, Authority Activation Records, Portable Authority Proofs, and Negative Authority Proofs may be identified and illustrated herein within components, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network. The components may be passive or active, including agents operable to perform desired functions such as autonomous software agents operating under the cryptographic authority substrate.

[0311] Reference throughout this specification to “an example” or “some aspects” means that a particular feature, structure, or characteristic described in connection with the example is included in at least one embodiment of the present invention. Thus, appearances of the phrase “in an example” or “in some aspects” in various places throughout this specification are not necessarily all referring to the same embodiment.

[0312] As used herein, a plurality of items, structural elements, compositional elements, and/or materials may be presented in a common list for convenience. However, these lists should be construed as though each member of the list is individually identified as a separate and unique member. Thus, no individual member of such list should be construed as a de facto equivalent of any other member of the same list solely based on its presentation in a common group without indications to the contrary. In addition, various embodiments and examples of

the present invention may be referred to herein along with alternatives for the various components thereof. It is understood that such embodiments, examples, and alternatives are not to be construed as de facto equivalents of one another, but are to be considered as separate and autonomous representations of the present invention.

[0313] Although the foregoing has been described in some detail for purposes of clarity, it will be apparent that certain changes and modifications may be made without departing from the principles thereof. It should be noted that there are many alternative ways of implementing both the processes and apparatuses described herein, including alternative implementations of hash-linked delegation graphs, post-quantum threshold signatures, multi-condition conformance gates, and proof generation modules. Accordingly, the present embodiments are to be considered illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

[0314] Those having skill in the art will appreciate that many changes may be made to the details of the above-described embodiments without departing from the underlying principles of the invention. The scope of the present invention should, therefore, be determined only by the following claims.

CLAIMS

WHAT IS CLAIMED IS:

1. A method for preserving authorized human intent across delegation chains via a cryptographic authority substrate, the method comprising:

creating, by one or more processors, a root authority object representing threshold-signed human-authored objective, the root authority object comprising at least an objective statement, a lifecycle mode indicator selected from a set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules, wherein the root authority object is cryptographically bound by a post-quantum threshold signature meeting the approval set;

computing, for each authority node in a delegation graph descended from the root authority object, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object, the ancestor constraint aggregate comprising at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints;

enforcing, at a pre-execution conformance gate prior to execution of an action by an acting principal, a four-condition test comprising: (i) an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object, the governing authority object being one of the root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact; (ii) a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; (iii) a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and (iv) a Lifecycle Compliant condition verifying that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp;

producing, responsive to the four-condition test passing, a Portable Authority Proof comprising: a lineage proof from the action to the root authority object, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a specific human-authored passage from an intent-passage registry, and a structured authority rationale record assembled programmatically from cryptographic

citations including the lineage proof, the instruction citation, the activated-authority citation, and the intent citation; and

producing, responsive to any condition of the four-condition test failing, a Negative Authority Proof identifying a failing condition and a denial-category identifier.

2. The method of claim 1, wherein the method further comprises:

expressing an authority state as a portable serialized authority graph comprising the root authority object and a plurality of delegation nodes, the portable serialized authority graph being independent of any specific workflow execution engine;

migrating the authority state from a first workflow execution engine to a second workflow execution engine by serializing the portable serialized authority graph from the first workflow execution engine and de-serializing the portable serialized authority graph into the second workflow execution engine without re-issuance of any signature in the portable serialized authority graph; and

verifying, after migration, that every authorization decision in the second workflow execution engine continues to descend from the root authority object and satisfies the ancestor constraint aggregate at every level.

3. The method of claim 1, wherein the method further comprises:

publishing a successor root authority object signed by a post-quantum threshold signature meeting a succession authority field of a predecessor root authority object, wherein the successor root authority object carries a predecessor-root-identifier field identifying the predecessor root authority object by hash, a succession event descriptor field, and a succession effective timestamp;

binding the successor root authority object to the predecessor root authority object by a signed succession event placed in an append-only succession registry;

inheriting, by new delegations made after succession, unexpired constraints from the predecessor root authority object via the succession event; and

maintaining verifiability of authority decisions taken under the predecessor root authority object indefinitely after succession.

4. The method of claim 1, wherein each authority node in the delegation graph carries witness commitments comprising: an evidence witness hash comprising a post-quantum digest over a cycle root of an evidence substrate, an outcome witness hash comprising a post-quantum

digest over a cycle root of an outcome substrate, and a prior state hash comprising a post-quantum digest over a system state observed at a time of node issuance, wherein the witness commitments are bound into a node hash such that modification of any witness commitment invalidates a signature of a respective authority node and every descendant lineage proof.

5. The method of claim 1, wherein the method further comprises:

publishing an agent registry binding each candidate recipient agent to a post-quantum signing public key and a post-quantum key-encapsulation public key;

publishing an instruction set comprising a plurality of instruction tags, each instruction tag comprising an encrypted body containing natural-language content encrypted under a per-instruction data encryption key, and a wrapped-keys field containing the per-instruction data encryption key separately encapsulated under each intended recipient's key-encapsulation public key by a post-quantum key-encapsulation mechanism;

requiring, upon decryption of an instruction body by a recipient agent, that the recipient agent produce a signed read receipt comprising a digest of a decrypted instruction body and a post-quantum signature by the recipient agent; and

requiring, in every action proof, an instruction citation comprising an instruction identifier, an inclusion proof, and a signed read receipt.

6. The method of claim 5, wherein the method further comprises:

including, in each instruction tag, an author signature field carrying a post-quantum signature by a registered author principal over a canonical encoding of a decrypted instruction body, wherein the author principal is distinct from a publishing principal;

requiring, at an agent runtime, that the agent runtime treat as authoritative only text carrying a verifiable instruction-tag citation with a valid read receipt and traceable lineage to an authorized publishing principal; and

treating all other text observed by the agent runtime from tool call responses, retrieved corpus documents, user messages, other agent messages, or environment artifacts as non-authoritative input data that is excluded from action authorization.

7. The method of claim 1, wherein the method further comprises:

requiring, prior to any action by the acting principal under the governing authority object, that the acting principal produce an Authority Activation Record comprising: a predecessor artifact hash, a successor artifact hash being acknowledged, a reconstructed

lineage view, cross-system cryptographic witness commitments, an acknowledgment timestamp, and a post-quantum signature by the acting principal;

persisting the Authority Activation Record in a per-principal append-only activation registry;

verifying, at the pre-execution conformance gate, that the acknowledged root hash of the acting principal equals the current active root hash for the authority chain; and

requiring, in every action proof, an activated-authority citation comprising a hash of the Authority Activation Record, a hash of an activated governing authority object, and a binding proof that constraints evaluated by the pre-execution conformance gate correspond to the activated governing authority object.

8. The method of claim 1, wherein the method further comprises:

detecting, at the pre-execution conformance gate, that two or more contributing root authority objects reach divergent verdicts on a proposed action;

emitting a conflict detection record comprising identifiers of the contributing root authority objects, conflicting constraints, a canonical encoding of the proposed action, and a conflict type classifier;

receiving a Conflict Resolution Record comprising a hash of the conflict detection record, a resolution decision, an authority basis citation, and a precedence disposition field selected from one-time and canonical; and

binding, where the precedence disposition field is canonical, a Conflict Precedent to a per-root append-only precedent registry for application to future conflicts of the same conflict type classifier.

9. The method of claim 1, wherein the method further comprises:

receiving, from a revoking principal meeting a revocation authority of an authority artifact, a Root Revocation Record comprising a canonical hash of the authority artifact being revoked, a revocation event descriptor, a revocation effective timestamp, and a post-quantum signature;

binding the Root Revocation Record by inclusion proof to a per-chain revocation registry;

verifying, at the pre-execution conformance gate for every action, exclusion of every authority artifact along the authority chain of the action from the per-chain revocation registry; and

denying any action whose authority chain includes any artifact having a revocation effective timestamp preceding the action timestamp, regardless of whether descendant authority nodes have unexpired validity windows.

10. The method of claim 1, wherein the method further comprises:

receiving, from a higher-order authority principal whose authority is committed by an ancestor root authority object, an override authorization record comprising: a hash of an original denial, an override decision, an override reason field, a higher-order authority basis citation, and an expiration condition selected from one-time, time-bounded, and condition-bounded;

verifying, at the pre-execution conformance gate, the higher-order authority basis; and

emitting an Override Portable Authority Proof comprising: an indication of what would have been denied, an indication of what was authorized, an identification of the higher-order authority principal, the expiration condition, and a signed trail of an override event.

11. The method of claim 1, wherein the method further comprises:

requiring, at root creation, a Root Certification Ceremony comprising: receiving a human intent statement and a human-authored summary distinct from the human intent statement; generating, for each approving principal, a display proof comprising a cryptographic commitment to an exact rendering shown to the approving principal; obtaining a post-quantum threshold signature from approving principals over a full intent hash, a summary hash, a proposed root hash, and display proof hashes; and

emitting a Root Certification Record bound by inclusion proof to a Root Certification Registry, wherein the pre-execution conformance gate denies any action whose authority chain terminates at the root authority object lacking a valid Root Certification Record.

12. A system for preserving authorized human intent across delegation chains, the system comprising:

at least one processor; and

a memory storing instructions that, when executed by the at least one processor, cause the system to implement a root authority object store, a delegation graph engine, a pre-execution conformance gate, and a proof generation module, and to perform operations comprising:

creating a root authority object representing threshold-signed human-authored objective, the root authority object comprising at least an objective statement, a lifecycle mode indicator selected from a set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules, wherein the root authority object is cryptographically bound by a post-quantum threshold signature meeting the approval set;

computing, for each authority node in a delegation graph descended from the root authority object, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object, the ancestor constraint aggregate comprising at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints;

enforcing, at a pre-execution conformance gate prior to execution of an action by an acting principal, a four-condition test comprising: (i) an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object, the governing authority object being one of the root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact; (ii) a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; (iii) a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and (iv) a Lifecycle Compliant condition verifying that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp;

producing, responsive to the four-condition test passing, a Portable Authority Proof comprising a lineage proof, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a specific human-authored passage from an intent-passage registry, and a structured authority rationale record; and

producing, responsive to any condition of the four-condition test failing, a Negative Authority Proof identifying a failing condition and a denial-category identifier.

13. The system of claim 12, wherein each authority node carries an authority window field expressing a conjunction of temporal scopes comprising at least: clock-window scopes specifying permitted times of day; calendar-window scopes specifying permitted calendar periods; event-window scopes conditional on substrate-observable events; and condition-window scopes conditional on substrate-observable predicates, wherein an effective authority window at the action is an intersection of authority window fields of every ancestor from the action to the root authority object.

14. The system of claim 12, wherein the operations further comprise:

maintaining a per-root principal-position registry binding principal identities to roles, each entry comprising a principal identifier, a signing-key hash, a role descriptor, and effective timestamps;

emitting, upon a personnel change, a Principal Membership Transition Record comprising a predecessor entry hash, a successor entry hash, an unchanged root hash, an authority basis citation, a transition event descriptor, and a transition effective timestamp; and

verifying, at the pre-execution conformance gate for every action, that the acting principal corresponds to a valid principal-position registry entry at the action timestamp and that no Principal Membership Transition Record bearing the acting principal as predecessor exists at or before the action timestamp.

15. The system of claim 12, wherein the lifecycle mode indicator comprises:

PERMANENT mode for constitution-level authority carrying no expiration and requiring a succession quorum exceeding a threshold of the approval set for supersession;

RENEWABLE mode for operating-policy authority carrying a renewal period and a next review due timestamp and requiring periodic Root Refresh Records;

PROJECT mode for mission-order authority carrying a project binding with completion criteria and auto-retiring upon completion; and

EPHEMERAL mode for emergency authority carrying a hard expiry timestamp and requiring human recertification for extension.

16. The system of claim 12, wherein the operations further comprise:

emitting, for the root authority object having the lifecycle mode indicator of RENEWABLE, a Root Refresh Record comprising: an unchanged root hash, a differential summary confirming no material modification, a re-displayed intent summary with display

proofs, certifying principal signatures, a refresh effective timestamp, and a next review due timestamp;

binding the Root Refresh Record to a per-root refresh registry; and

denying, at the pre-execution conformance gate, any action under the root authority object whose most recent Root Refresh Record has a next review due timestamp in the past, and emitting a Negative Authority Proof with a denial-category identifier indicating governance review missed.

17. The system of claim 12, wherein the operations further comprise:

emitting, for the root authority object having the lifecycle mode indicator of PROJECT or EPHEMERAL upon reaching a completion criteria or a hard expiry timestamp, a Root Retirement Record comprising: a retired root hash, a retirement event descriptor, a retirement effective timestamp, a closure artifact summarizing lifetime activity, and a residual obligations marker naming obligations persisting beyond retirement;

binding the Root Retirement Record to a retirement registry; and

denying, at the pre-execution conformance gate, any action under the retired root authority object whose action timestamp follows the retirement effective timestamp and does not invoke a listed residual obligation.

18. The system of claim 12, wherein the operations further comprise:

receiving a subtree freeze instruction identifying a target node within the delegation graph from a principal listed in the revocation rules of the root authority object;

publishing to an append-only revocation registry a signed freeze event identifying the target node;

enforcing, at the pre-execution conformance gate for any action involving a descendant of the target node, rejection with a revocation-blocked artifact; and

permitting, at the pre-execution conformance gate for any action not involving a descendant of the target node, normal continuation of authority such that revocation of one subtree does not cascade to sibling subtrees.

19. The system of claim 12, wherein the operations further comprise:

maintaining, per root authority object, an append-only intent-passage registry comprising canonical encodings of human-authored instruction passages, each passage signed by a registered author principal and assigned an intent passage identifier;

including, in each of a plurality of instruction tags, an intent passage identifier field identifying a human-authored passage that authored an instruction body;

requiring, in every action proof, an intent citation comprising: an intent passage identifier of each governing passage, an inclusion proof against an intent-passage registry root, a canonical hash of cited passage text, and a scope-conformance verification binding; and

assembling, with each Portable Authority Proof, a structured authority rationale record comprising: an acting principal identifier, an action summary, a cited intent passage in natural language, a delegation chain description, an authority window description, and an activation state description.

20. A non-transitory computer-readable medium storing instructions that, when executed by a processor, cause the processor to perform operations comprising:

creating a root authority object representing threshold-signed human-authored objective, the root authority object comprising at least an objective statement, a lifecycle mode indicator selected from a set consisting of PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules, wherein the root authority object is cryptographically bound by a post-quantum threshold signature;

computing, for each authority node in a delegation graph descended from the root authority object, an ancestor constraint aggregate as a function of every ancestor node from an immediate parent up to and including the root authority object, the ancestor constraint aggregate comprising at least an intersection of allowed action classes, a union of prohibited action classes, an intersection of temporal validity windows, and a composition of policy constraints;

enforcing, at a pre-execution conformance gate prior to execution of an action by an acting principal, a four-condition test comprising: (i) an Activated condition verifying that the acting principal has produced an Authority Activation Record acknowledging a governing authority object, the governing authority object being one of the root authority object, a successor root authority object, an instruction-set publication, or a delegation artifact; (ii) a Current condition verifying that an acknowledged root hash of the acting principal equals a current active root hash for an authority chain of the acting principal; (iii) a Within Window condition verifying that an action timestamp falls within an intersection of authority windows of every ancestor in the authority chain; and (iv) a Lifecycle Compliant condition verifying

that the lifecycle mode indicator of the root authority object permits execution of the action based on mode-specific temporal enforcement at the action timestamp;

producing, responsive to the four-condition test passing, a Portable Authority Proof comprising a lineage proof, cross-system cryptographic witness commitments, an instruction citation, an activated-authority citation, an intent citation referencing a human-authored passage, and a structured authority rationale record; and

producing, responsive to any condition of the four-condition test failing, a Negative Authority Proof identifying a failing condition and a denial-category identifier.

ABSTRACT

A method for preserving authorized human intent across delegation chains comprises creating a root authority object representing threshold-signed human-authored objective, the root authority object comprising an objective statement, a lifecycle mode indicator selected from PERMANENT, RENEWABLE, PROJECT, and EPHEMERAL, an approval set with threshold semantics, allowed action classes, prohibited action classes, temporal constraints, and revocation rules, cryptographically bound by a post-quantum threshold signature. The method comprises computing, for each authority node in a delegation graph, an ancestor constraint aggregate as a function of every ancestor node. The method comprises enforcing, at a pre-execution conformance gate, a four-condition test comprising an Activated condition, a Current condition, a Within Window condition, and a Lifecycle Compliant condition. The method produces a Portable Authority Proof responsive to the test passing, and a Negative Authority Proof responsive to any condition failing.