

Cascading Trust Failures in AI Agent Networks: A Trust Architecture for Scalable Revocation and Bounded Containment

Design Principles, Practical Limits and a Path to Implementation

Dr. Carsten Stöcker
Spherity GmbH

Prepared: 10 October 2026



Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). © 2026 Dr. Carsten Stöcker.

Abstract

Credential revocation is a distributed control problem when artificial intelligence (AI) agents act across organisational boundaries. Fine-grained mandates and provenance increase evidence processing, status publication and withdrawal work. This article develops a conditional framework connecting those workloads to protected enforcement and cascading harm in a finite network of industrial engineering agents. Eligibility applies to one complete action and continuation record. The architecture combines domain-local verification, shared authenticated caches, finite leases, reserved containment service and persistent cross-domain operation records. Conditional membership results specify the observation laws required for matched decoys. Exact capacity calculations separate status multiplicity from lost compressibility. Twenty disjoint one-bit statuses per bundle with 50% decoys require forty times the raw capacity. Reproducible GZIP (GNU zip) lossless-compression fixtures give encoded-snapshot ratios of 40.03 for sparse unmasked lists and 1475.44 for random-masked lists, both relative to one compressed single-status sparse baseline. Random masking removes compression savings; these ratios are fixture-specific representation costs. These comparisons do not preserve equivalent incident semantics and do not measure privacy. A hosting model separates resident local reads from authoritative freshness, publication and consensus delay. Conditional offspring and harm bounds connect supported enforcement delays to cascade exposure. The contribution is the explicit set of assumptions at these connections. The companion article develops fixed-cohort decoys, a protected protocol and executed model studies; continuous issuance, implementation refinement and empirical workload calibration remain open. No deployment-level containment or privacy result is claimed.

Keywords: multi-agent systems; AI safety; credential revocation; evidence provenance; zero trust; matched decoys; distributed ledgers; post-quantum cryptography

Contents

1	Research question and contribution	5
1.1	Threat model and claim boundaries	6
2	Eligibility bound to one complete execution record	6
3	Evidence objects and distinct scaling workloads	8
3.1	Required evidence and logical alternatives	8
3.2	Logical demand physical capacity and network work	9
4	Status standards recognition and cryptographic profiles	10
5	Reference architecture and solution patterns	10
5.1	Responsibilities and trust boundaries	11
5.2	Seven complementary solution patterns	13
5.3	Status and privacy representations	13
5.4	Publication hosting and verification	15
5.5	Formal architecture contracts	22
5.6	Composite designs and assessment	23
6	Matched decoys and the privacy objective	24
6.1	Unknown membership and known credential status	24
6.2	Conditional membership proposition	27
6.3	A quantitative target for imperfect matching	28
6.4	Reserve lifetime and continuing issuance	29
7	Net payload and scaling factors	29
7.1	Physical capacity	29
7.2	Compression and transmitted payload	30
7.3	Changes between versions	30
7.4	Ledger storage publication and cover traffic	31
8	Reproducible numerical results	32
8.1	Snapshot codec fixtures	32
8.2	Exact small model checks	35
9	Coupling capacity to withdrawal and cascading harm	35
9.1	Detection publication and protected enforcement	35
9.2	Mean queue delay and guaranteed service	36

9.3	Conditional propagation and expected harm	36
9.4	Persistent pending effects and distinct outcomes	38
9.5	Illustrative containment and harm sensitivity	39
10	Validation interface and companion study	41
10.1	Minimum deployment study	41
10.2	Outcomes and evidence requirements	41
11	Revocation List Propagation in Multi-Agent Systems	42
11.1	Authority across three trust domains	42
11.2	Push pull and lease expiry	44
11.3	Ordering withdrawal and execution	44
11.4	Failures and shared infrastructure	45
11.5	The operational trade-off	45
12	Implementation obligations and research priorities	46
12.1	Standards and experimental profiles	46
12.2	Control selection for software engineers	46
12.3	Telemetry and alert thresholds	46
12.4	Where to start with one protected file release	47
12.5	Research priorities and the companion paper	49
13	Scope and conditions for article claims	49
13.1	Research transparency	50
A	Reproducibility of the numerical fixtures	51
B	Multi-Agent Trust and Revocation Implementation Checklist	51
C	Implementation contracts and research allocation	53
C.1	Contract and remaining implementation refinement	53
C.2	Placement of the remaining questions	54
D	Complete runnable examples and outcome charts	55
D.1	Running the examples	55
D.2	Setup	55
D.3	Example 1: one action name can hide incompatible plans	56
D.4	Example 2: count real status capacity before estimating traffic	56
D.5	Example 3: masking, decoys and compression	58

D.6	Example 4: a readable local list can become too old	61
D.7	Example 5: interpret a conditional cascade bound	63
D.8	Plot the compression outcome	65
D.9	Retain the reproducibility record	66
E	Recommendations for W3C status-list specifications	66
E.1	Applicable documents and the proposal boundary	66
E.2	Proposed changes and required evidence	67
E.3	Why random default wire bits are not recommended	68
E.4	Explicit masked status methods and codec selection	69
E.5	Related specifications and contribution route	70
F	Anonymous status proofs and adoption evidence	70
F.1	Proof binding and receiver freshness	70
F.2	Lifecycle cost and invalidation work	71
F.3	Adoption evidence and limits	71
	References	73

1 Research question and contribution

Industrial AI agents increasingly require authority from several organisations and evidence from several technical sources. A simulation agent may be permitted to read a design, execute a tool and submit a result while lacking authority to release the underlying file to a subcontractor. Its permitted scope can change when a representation credential expires, an issuer loses recognition, an assurance result is withdrawn or a source is compromised. A cryptographically valid credential does not by itself settle any of these questions. The resource operator must determine which current claims justify the specific proposed effect [1–4].

We study a finite network of engineering agents coordinating industrial simulations and cross-organisational file releases. The primary harm is an executed release outside the actual permitted scope. Two separate operational outcomes are legitimate work lost before its deadline and remote operations whose outcome remains unresolved. A rejected action is not an executed confidentiality breach, and a timeout is not proof of cancellation. These distinctions are preserved throughout the model.

The central question is which evidence, freshness and service conditions allow credential withdrawal to bound covered effects across trust domains. Credential multiplicity and privacy representation determine storage and processing demand. That demand affects containment only through the implemented service and enforcement path. Finer credentials may also reduce permitted scope, influence rates and harm severity. A claim about their net effect therefore requires equivalent-policy comparisons or an explicit account of the changed authority.

Table 1: Relationship to established research

Research line	Established work	Contribution of this article
Network cascades [5–8]	Threshold, interdependent and interacting spreading models; next-generation operators.	Typed trust dependencies and enforcement deadlines connected to executed resource effects.
Agent injection [9, 10]	Propagation of adversarial instructions across agents and evaluation environments.	Withdrawal capacity and persistent remote operations under specified matched workloads.
Authorisation and assurance [1–3, 11]	Bound execution records, permission consistency and protected runtime control.	Population-level workloads and conditional containment costs across trust domains.
Credential status [12–15]	Status representations and scoped issuer recognition.	A common model for useful payload, authoritative freshness and hosting failures.
Issuer metadata privacy [16]	Padding, reconstruction and issuer-activity protection in the CRSet credential-revocation scheme.	Full-transcript membership conditions and their capacity and containment implications.

The contribution is a framework connecting complete-record eligibility, status representation and service capacity to protected withdrawal and conditional cascade costs. The propositions use established mathematical methods to state the assumptions at each connection. Exact capacity equations and reproducible codec fixtures establish representation costs; the full connection to operational delay and harm remains a testable hypothesis. Caching, expiry, decoy padding and branching-process bounds are established components. CRSet already addresses issuer-activity

privacy [16]. Table 1 locates the proposed integration within a focused literature comparison, rather than claiming a systematic survey or mathematical priority.

1.1 Threat model and claim boundaries

Agents may propose unauthorised actions, pass adversarial content to peers or present incomplete evidence. Sources may become defective or be withdrawn. Network services may delay, omit, replay or equivocate over status data. The containment results assume that the protected controller, resource gate, required local state and stated cryptographic assumptions remain sound. A bypass of the gate is outside the proved configuration and is included as an omission test. A compromised recognised issuer can make an authentic false assertion; cryptography alone does not resolve its semantic truth.

The operational adversary may replay a valid old view, trigger a refresh flood, exploit inconsistent dependency versions or race invalidation against commit. Independent replicas can expose conflicting signed histories, but cannot establish the truth of an agreed false assertion. Deployment testing must exercise clock rollback, key compromise and correlated infrastructure failures at the declared boundaries. This article does not report those deployment experiments.

The privacy observer sees all public status snapshots and the specified publication transcript. Side information may include issuance times, known credential indices and public incidents. The basic membership proposition concerns observation under a fixed transcript-generating experiment. An active observer requires a specified intervention strategy and matching under that strategy; passive matching does not establish active security. Query privacy is defined separately from membership and issuer-activity privacy.

The results have three evidence levels. Propositions follow from stated assumptions. Numerical results are codec measurements or exact model calculations. Deployment privacy, comparative hosting performance and end-to-end containment remain untested. Section 10 defines the evidence required to connect these levels; an observed codec factor cannot substitute for a measured enforcement or harm result.

Readers can follow two paths. For an architectural reading, start with the file-release example in Section 2, the distinct workloads in Section 3, and the controls in Section 5. Sections 6–8 explain the privacy objective and the cost of representing status. The examples distinguish storage, transmitted bytes and checks performed for each action. Section 9 then connects supported enforcement delays to conditional cascade estimates. Sections 11–12 translate these conditions into cross-domain responsibilities and implementation priorities. Readers checking the mathematical claims should also examine the assumptions immediately before each proposition and the scope stated after its proof. A numerical result belongs to its stated experiment: a compression ratio does not measure privacy, and an expected-harm bound is not a guarantee for every incident. The runnable appendix provides small examples. The companion article develops the protocol and its executed model studies; production evidence remains a separate requirement. Appendices A and B retain the numerical reproduction details and implementation checklist. Appendix C maps the two papers. Appendix D supplies runnable teaching examples; Appendix E sets out proposed standards changes. This two-article study is separate from the earlier Sphery AI Safety Series [1–3].

2 Eligibility bound to one complete execution record

Engineering example. Consider releasing one engineering file to a named subcontractor for a specified simulation. Permission must cover that file, recipient, purpose and duration. The

assurance evidence must address the same operation and its later handling. A permission to process the file internally cannot be combined with an assessment of external processing. The complete record keeps these details together, so the gate evaluates one proposed course of action rather than combining compatible-looking answers to different questions.

The Sphery Executive Brief expresses eligibility as the intersection of mandate and assurance [1, Section 5.1]. The adjacent citation, numbered 6 in that brief, refers to Paper 2, whose equations (10) and (11) distinguish action sets from complete execution records [2]. The action-level rule below is shorthand with a common bound continuation and context. It does not identify the intersection of two arbitrary existential projections:

$$\mathcal{A}_t^{\text{eligible}} = \mathcal{A}_t^{\text{mandate}} \cap \mathcal{A}_t^{\text{assurance}}. \quad (1)$$

The precise implementation evaluates one candidate $g = (a, h, \pi, \nu)$: action a , duration h , continuation π and bound context ν . Context binds principals, resource, audience, episode, configuration, evidence versions, policy, named harms and persistent accounting. Let $\mathcal{P}_g$ be the principals required by the relying policy and $M_{p,t}(g)$ the mandate predicate for principal p . Let $\text{App}_t(g)$ assert that the joint evidence and model apply, required observations and containment exist, bindings are exact and the returned numerical assurance bound is sound.

$$\mathcal{G}_t^{\text{mandate}} = \{g : \forall p \in \mathcal{P}_g, M_{p,t}(g) = 1\}. \quad (2)$$

$$\mathcal{G}_t^{\text{assurance}} = \{g : \text{App}_t(g) = 1, \forall j, r_{t,j}(g) \leq \alpha_{t,j} \leq B_{t,j}\}. \quad (3)$$

For named harm j , $r_{t,j}(g)$ bounds first harm in the remaining joint continuation, conditional on the available information rather than on unobserved survival. It includes delayed containment and pending work through closure, or a separately charged bounded tail [2]. The allowance $\alpha_{t,j}$ is permanently debited before the admitted effect and $B_{t,j}$ is the remaining protected episode budget. The source's episode guarantee additionally requires its model-validity, prefix-replacement and accounting assumptions; it is not a global network expected-harm bound. Rechecking one admitted continuation does not create another debit. Multiplying local scores does not establish the required joint bound.

$$\mathcal{G}_t^{\text{eligible}} = \mathcal{G}_t^{\text{mandate}} \cap \mathcal{G}_t^{\text{assurance}}, \quad \mathcal{A}_t^{\text{eligible}} = \{a(g) : g \in \mathcal{G}_t^{\text{eligible}}\}. \quad (4)$$

Proposition 1 Common-record eligibility. Projection to an action must follow intersection of the complete-record sets. In general, intersecting their separate action projections admits actions for which no common acceptable continuation exists.

$$\text{proj}_a(\mathcal{G}_M \cap \mathcal{G}_A) \subseteq \text{proj}_a(\mathcal{G}_M) \cap \text{proj}_a(\mathcal{G}_A). \quad (5)$$

Proof. Every record in the intersection belongs to each set, so its action lies in both projections. The reverse inclusion fails when $\mathcal{G}_M = \{(a, h, \pi_1, \nu)\}$ and $\mathcal{G}_A = \{(a, h, \pi_2, \nu)\}$ with $\pi_1 \neq \pi_2$. Both projections contain a , but the record intersection is empty. Thus an action permitted only with immediate deletion cannot inherit assurance established only for continued processing. The protected gate retains the selected complete record after projection.

Admission additionally requires a valid operation slot, applicable resource constraints, current status and the freshness contract in Section 5. Ordinary suspension blocks new task effects while preserving separately authorised containment operations. The resource gate rechecks relevant

local state at commit or applies a sound incremental equivalent, because an earlier decision alone does not resolve a later change.

The implementation binds a canonical record identifier to the exact evidence and policy versions. Parsers reject ambiguous encodings and unsupported profiles. A local commit fence compares the admitted versions with the gate’s current dependency generation and expiry. This implements a resource-local order; it does not establish simultaneous revocation across organisations.

3 Evidence objects and distinct scaling workloads

Let $\mathcal{V}_A$ be a finite set of agent instances, $\mathcal{S}_A$ authoritative sources, $\mathcal{C}$ versioned credentials, $\mathcal{L}$ status objects and $\mathcal{R}$ protected resources. Communication, evidence dependency, status delivery and effect execution are distinct relations. A recognised issuer is authorised for specified claims and actions under a relying policy. The credential issuer, status authority, holder, operator and resource owner can be different entities.

3.1 Required evidence and logical alternatives

For candidate g , define selected mandate, assurance and provenance evidence as $C_M(g)$, $C_A(g)$ and $C_P(g)$. The policy’s typed dependency closure includes required ancestors, prohibitions and relevant adverse evidence:

$$C_g = cl_{required}(C_M(g) \cup C_A(g) \cup C_P(g)), \quad k_g = |C_g|. \quad (6)$$

The closure is relative to explicit discovery and completeness assumptions. A hash can detect changes in known evidence; it cannot establish the absence of an undiscovered adverse fact. Policies can contain conjunctions, alternatives and thresholds. An update marks dependent decisions for re-evaluation, while a valid alternative witness can preserve eligibility. Blanket descendant invalidation is justified only when the withdrawn premise is indispensable under the applicable policy.

Bound graph depth, node count, proof size and validation work before admission. Cyclic or unsupported dependency structures need an explicit policy semantics; they cannot inherit the acyclic traversal estimate. If discovery is incomplete or its bounded search cannot establish the required witness, the decision remains unknown.

Deduplication applies to credential versions, not to textual claims. Eight mandate credentials and sixteen assurance or provenance credentials with four shared versions yield twenty distinct credentials when required closure is already included. This is a scenario, not a fixed requirement from the source papers. Immutable historical receipts may require authenticity checking without a live status slot. Revocation of a credential, withdrawal of an issuer’s authority and invalidation of an assurance result have different meanings.

3.2 Logical demand physical capacity and network work

Table 2: Distinct evidence and status workloads

Quantity	Definition	Scaling implication
k_g	Distinct required credential versions for g.	Evidence processing; cached signatures can avoid repeated cryptography.
z_g	Distinct status checks after reference and version deduplication.	Local status inspections; not necessarily remote requests.
ℓ_g	Distinct status objects required by g.	Potential fetches before caching and stapling.
M	Unique allocated status entries across the population.	Persistent capacity; shared credentials count once.
$f_{v\ell}$	Actual completed refreshes per second by cache v for list l.	External byte demand after reuse and freshness policies.

Engineering interpretation. Suppose each proposed file release depends on twenty distinct status entries held in four lists. A gate inspects the twenty entries, but a cache may fetch only four list objects and reuse them for many releases. Shared credentials can also serve several agents without allocating a new status slot each time. The implementation therefore needs separate counters for evidence checks, allocated entries, downloaded objects and transmitted bytes. Improving one counter does not establish the same reduction in the others.

At proposal rate λ_a with the stated candidate distribution, logical demands are $\lambda_a \mathbb{E}[k_g]$ and $\lambda_a \mathbb{E}[z_g]$. These expectations require a specified workload; bursts need separate admission bounds. Traversing a selected acyclic evidence graph is linear in its nodes and edges, while finding a valid witness or computing assurance can be more expensive. For N credential bundles with physical allocation sets $\mathcal{S}_b$:

$$M = \left| \bigcup_{b=1}^N \mathcal{S}_b \right|, \quad k_{pop} = \frac{M}{N}, \quad \dot{B}_{net} = \sum_{v,\ell} f_{v\ell} W_\ell. \quad (7)$$

A physical allocation is identified by list, index, purpose and lifecycle. New snapshots and repeated checks do not allocate new slots. Population multiplicity therefore differs from per-decision checks, issuance rate and renewal rate. The byte equation counts list traffic across specified links; proof, event, ledger synchronization and local processing costs are added through disjoint terms in Section 5.

If z references choose lists independently and uniformly from G lists, an indicator for whether list l is touched has expectation $1 - (1 - 1/G)^z$. Summing these indicators gives:

$$\mathbb{E}[\ell_g] = G \left[1 - \left(1 - \frac{1}{G} \right)^z \right]. \quad (8)$$

For z = 20, the expected counts are 1, 3.99 and 18.21 for G = 1, 4 and 100. The illustration shows why twenty credential statuses need not cause twenty fetches. Real issuer clustering and shared dependencies replace the independent allocation assumption. Freshness must still be checked when an object or its proof is cached.

4 Status standards recognition and cryptographic profiles

The Verifiable Credentials Data Model v2.0 supplies the credential framework [4]. The World Wide Web Consortium (W3C) Bitstring Status List v1.0 is a Recommendation dated 15 May 2025. Its encoded list uses GZIP and Multibase base64url without padding, a Uniform Resource Locator (URL)-safe base64 encoding. The minimum raw size is 16 kibibytes (KiB; 1 KiB = 1024 bytes) and the default width is one bit. Revocation is irreversible; suspension permits reversal. Random index allocation is recommended. The time-to-live (ttl) refresh indication does not extend credential validity [12].

Bitstring Status List v1.1 was published as a First Public Working Draft on 24 September 2026. Its optional terse profile combines a 32-bit index with 2^{26} entries per list and 2^6 addressable lists. A one-bit list contains 8 mebibytes (MiB; 1 MiB = 1,048,576 bytes) raw, 512 times the v1.0 minimum. This is a profile-specific calculation; it does not require a verifier to download every addressable list [13].

Recognized Entities v1.0 is a Working Draft dated 6 September 2026. It represents recognised actions and output-validation constraints, rather than one universal trust flag. The draft explicitly describes itself as experimental and unsuitable for production deployment. The accompanying 8 October 2026 Group Note Draft discusses stale recognition, discovery depth, retrieval privacy and resource exhaustion. We use these documents as emerging model inputs and threat guidance, not as established production conformance profiles [14, 15].

W3C’s October 2026 barcode case study motivates withdrawal after identity-document and signature misuse [17]. It supplies no bound on end-to-end enforcement latency. The model below separates the decision to withdraw, publication, retrieval and resource enforcement. A status record also does not determine whether the underlying real-world qualification remains valid after the credential is withdrawn.

The bitstring specification discusses decoys without defining their generation algorithm. Its normative generation procedure starts with zero values and sets entries associated with issued credentials. Nonzero unissued decoys require standards clarification or an explicitly nonstandard extension. A custom profile cannot override the normative generation rule. Unmasked real entries retain ordinary decoding, while real entries masked with exclusive OR (XOR) require different verifier semantics. Neither extension is claimed to be a conforming producer merely because its output can be decoded [12, 13].

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-207 distinguishes the policy engine, policy administrator and policy enforcement point (PEP). The engine and administrator jointly implement the policy decision point (PDP); the protected resource gate corresponds to enforcement [18]. These roles require authenticated control paths across domains. Neither a local cache nor a ledger is trusted merely because it is inside a domain. OpenID Shared Signals Framework (SSF) and Continuous Access Evaluation Profile (CAEP) carry relevant security events [19, 20]. Their delivery does not establish resource enforcement. Federal Information Processing Standard (FIPS) 204 specifies the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [21]. Object authenticity, credential semantics and transport protection remain separate. Section 12 gives the proposed integration rules.

5 Reference architecture and solution patterns

The proposed architecture implements the eligibility and status model at each protected resource boundary. It combines domain-local evidence verification, shared status caches, authenticated change events, periodic reconciliation, finite execution leases and persistent operation identifiers

(IDs) and records. The recommended design is a hypothesis to evaluate, not a demonstrated deployment result. Its contribution concerns the analytical connection between trust workload, containment deadlines and harm, together with specified empirical tests.

5.1 Responsibilities and trust boundaries

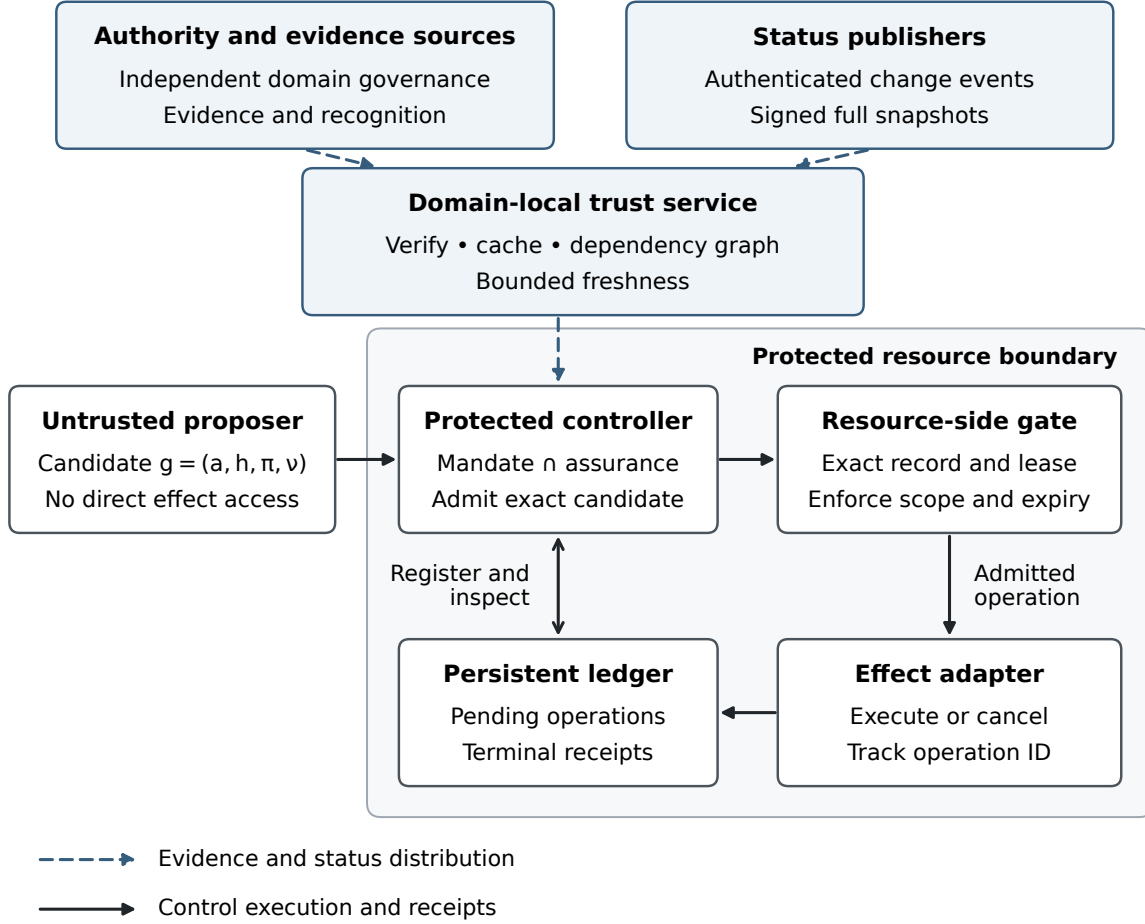
Authority and assurance remain distinct. A possible organisational chain runs from an authentic business register to a legal-entity credential, a representation credential and an agent mandate. Operator, test, evaluation, verification and validation evidence, external conformity-assessment claims and provenance contribute different assurance premises. The relying organisation recognises each issuer only for specified claims and actions. A wallet holds keys and evidence; possession alone does not establish mandate or assurance.

Each resource domain runs a trust service that verifies the required evidence slice, resolves scoped issuer recognition and retains authenticated status versions. It feeds a protected eligibility controller. The controller checks the same complete candidate record against mandate, joint assurance, operation state and resource constraints. The resource-side gate enforces the approved record. This separation follows the policy-decision and enforcement roles in zero trust architecture, with the additional assurance and persistent accounting obligations from the Spherity model [1, 2, 18].

The covered deployment must prevent the proposer from bypassing the gate through an alternative application programming interface (API), credential, queue or tool. The gate and its state are part of the trusted computing base. A remote organisation either performs its own evaluation or explicitly recognises a scoped attestation authority; a remote allow decision is never universally valid. A status authority may differ from the credential issuer, so its right to report status is also checked.

Proposed resource-domain architecture

Independent source governance and local enforcement



Repeat this pattern at each resource boundary. No single global trust root is assumed.

Figure 1: Proposed trust and execution architecture. Each resource domain applies its own relying policy. Signed evidence and status enter through validation. The design requires ordinary effects to pass through the protected gate. Authenticated terminal evidence satisfying the operation contract permits closure. Trust-service and ledger messages do not independently authorise execution. A separate covered-message control is required if the cascade bound includes communication containment.

5.2 Seven complementary solution patterns

The patterns in Table 3 distinguish required enforcement controls from delivery optimisations and experimental privacy variants. P1 and P6 bind local effects and remote operations to the complete record. P4 supplies the bounded-work conditions needed under load. P2 and P3 can reduce repeated retrieval and reevaluation, but their benefit depends on sound version and dependency handling.

P5 applies only under the observation law in Section 6. A public event path that identifies genuine withdrawals can defeat that privacy objective. P7 governs algorithm and key migration; its costs enter the byte and service accounting without changing the truth of the signed claim. Appendix B lists test obligations for every pattern. The companion paper specifies protocol transitions and proposed profile obligations, then evaluates executable models [22]; production conformance remains to be established.

Table 3: Solution patterns and their conditions

Pattern	Expected gain	Decisive limitation	Assessment
P1 Gate and lease	Exact effects and bounded stale reliance.	Complete mediation, current renewal and coherent state.	Core control
P2 Cache and events	Shared downloads and quicker typical withdrawal.	Stale cache, lost events and concentrated failure.	Preferred delivery
P3 Dependency index	Recheck the affected evidence slice.	Incomplete dependencies can miss invalidation.	Conditional optimisation
P4 Capacity reservation	Preserve enforcement during ordinary overload.	Finite work bounds and an availability cost.	Core under load
P5 Matched privacy cohorts	Limit specified membership inference.	Joint histories and all public channels must match.	Experimental branch
P6 Persistent closure	Account for remote effects and retries.	Receiver enforcement and authoritative receipts.	Core across domains
P7 Algorithm agility	Migrate authenticity assumptions.	Proof cost, downgrade rules and key compromise.	Lifecycle control

These assessments are reasoned design judgements. No numerical maturity or security score is assigned. Evaluate each pattern against exact-record correctness, deadline behaviour, computation, bytes, stated privacy, failure concentration, closure correctness and standards compatibility.

5.3 Status and privacy representations

Sparse bitstrings provide the reference representation. Matched decoys and masking are separate experimental variants. The companion study compares two hosting paths in an executable workload model of the same protected architecture; accumulators, private query certification and delegated design D remain assessed alternatives.

Origin-signed evidence stapling is a delivery option for cross-domain jobs. The submitting party carries fresh signed lists and recognition evidence; the receiver verifies authority, age and version independently. Reuse a cached identical version and deduplicate lists within the submission. Ordinary W3C bitstring stapling carries a signed list, not a standard compact proof of a single

bit. A new Merkle proof, accumulator proof or status attestation needs a separate method and trust profile [12].

Table 4: Status representations and privacy objectives

Representation	Privacy objective and scaling	Unresolved cost or assumption
Ordinary sparse bitstring	Group retrieval with inexpensive local bit checks and strong sparse compression.	Public histories remain visible; the known index is linkable.
Unmasked matched decoys	Unknown-membership protection under matched histories; keeps sparse compression in the evaluated regime.	Unissued decoy profile and joint-history matching remain experimental.
Masked matched decoys	Same conditional membership target, with random initial states.	Loss of compression; any additional benefit needs a separate threat model.
CRSet	Published research targets issuer-activity privacy through padding and regular reconstruction.	Full rebuild and distribution costs; capacity rollover and retrieval leakage [16].
Accumulator proofs (Indy/AnonCreds; ALLOSAUR)	Hide the presented status index under construction-specific proof assumptions. Compact verifier state.	Witness and publication work; hidden-index invalidation; no general PQC guarantee [23–25]. See Section 5.3.1 and Appendix F.
Private query certification	Private information retrieval (PIR) conceals a queried index; certification can bind the answer for a verifier.	Online service, update and integrity assumptions; query privacy alone does not prove correct status [26, 27].

Oblivious HTTP, a privacy mechanism for Hypertext Transfer Protocol (HTTP) requests, can separate the client identity seen by a relay from the request seen by a gateway. It does not hide the queried resource from that gateway and depends on the separation of those roles [28]. Likewise, a short-lived mandate reduces its valid lifetime but raises renewal work; it does not hide presentation metadata. Private Record Certification is a useful research comparator, but its published construction separates strong unlinkability from an integrity assumption that revocation servers follow the protocol. Its application-specific timestamp window remains part of freshness [27].

5.3.1 Anonymous status proofs and Indy AnonCreds

The relevant primitive is a cryptographic accumulator, rather than a cryptographic aggregator. Indy combines Camenisch–Lysyanskaya (CL) credential signatures with a Type-3 pairing-based revocation accumulator, identified as CL_ACCUM in its application interface [24]. The revocation accumulator should not be described as Rivest–Shamir–Adleman (RSA)-based merely because the credential-signature construction uses an RSA group. AnonCreds (Anonymous Credentials) now specifies a ledger-independent interface; Indy is one registry option [25].

A holder constructs a zero-knowledge proof (ZKP) that its credential is unrevoked in a selected authenticated state without exposing the credential’s revocation index. For example, a file-release

agent can prove a current mandate while withholding that stable status locator from the receiver. Public tails files supply auxiliary values for the holder’s cryptographic witness; the verifier needs the accumulator and verification material, not that tails file. Compact verifier state therefore shifts work to state distribution, witness maintenance and proof processing. The cryptographic witness is distinct from the policy-level evidence witness used by the protected controller [24, 25].

This property addresses presentation linkability. It does not conceal every public update, registry identifier, timestamp or disclosed attribute. Historical Indy updates expose issued and revoked index deltas; issuers retain their own issuance mappings [24]. Thus anonymous presentation, private retrieval and issuer-activity privacy require separate tests. Formal analysis establishes correctness, unforgeability and anonymity under its stated cryptographic model; it does not establish the source-age or receiver-containment bounds of this article [29].

Two related choices need explicit separation. A plain Merkle path ordinarily exposes its selected position. A zero-knowledge wrapper can conceal it; zk-creds provides a researched example, with path maintenance and proof-system assumptions of its own [30]. BBS, a pairing-based selective-disclosure signature scheme, supports derived proofs but does not by itself define a complete non-revocation protocol [31]. Neither classic CL/pairing proofs nor a pairing-based wrapper becomes post-quantum merely because its published root is signed with ML-DSA.

For P3, a hidden status index prevents direct per-index invalidation unless an explicitly authorised, scope-limited dependency handle and an authenticated, complete withdrawal-to-handle mapping are available and bound to the same credential proof. Otherwise invalidate all live decisions dependent on the changed registry epoch and require a new admissible proof or lease termination. This broadens P4’s service workload. Keep operation identifiers local to their contract rather than adding a globally correlatable credential identifier. Appendix F states the interface and accounting obligations; the companion’s Section 5.10 and Appendix F define the planned adapter tests. No accumulator benchmark is reported here.

5.3.2 Adoption and selection

Public evidence checked on 10 October 2026 supports coexistence. British Columbia uses BC Wallet for access to court materials, and its maintained wallet includes AnonCreds [32, 33]. W3C Bitstring Status List has Recommendation status and a multi-vendor test report; cheqd documents both status families [12, 34, 35]. These sources do not supply comparable counts of revocable credentials or checks. AnonCreds’ own project review identifies incomplete adoption data [36]. There is no supported market-wide winner. Appendix F separates deployed use, implementation support and research proposals.

5.4 Publication hosting and verification

The previous subsection selects a status representation. This subsection asks how an authoritative version reaches a verifier, how the verifier authenticates it, and how long it may be used. Representation, publication ordering, data availability, verification mode and delivery are separate design choices. A bitstring can use signed HTTP snapshots, replicated ledger state or an off-chain object committed on a ledger. Caching, events and stapling can apply to each. These choices feed the evidence and status input of Figure 1; they do not replace its protected controller, gate or operation ledger.

5.4.1 Central services federations and ledger replicas

A central HTTP service offers simple publication and inexpensive replication through domain caches. Its operator can observe requests that reach it, including time, network address and requested list. An authenticated client may also reveal an explicit verifier identity; an Internet Protocol (IP) address alone need not identify one. Fetching a whole list does not ordinarily disclose the requested bit index. Signed snapshots permit independent mirrors without delegating the issuer’s authority. A federation should specify version ordering, anti-rollback rules, equivocation handling, independent administration and correlated Domain Name System (DNS) or cloud failures. Mirroring is not itself a consensus protocol.

A synchronized local distributed ledger technology (DLT) node can read already validated state without a per-check external request. This preserves access during failure of the issuer’s HTTP service or domain name. Fresh updates can also continue if the node retains working peer connectivity and the issuer can still publish. A total network partition preserves only last-known state. Local signed-snapshot caches provide the same temporary offline-read property. DLT adds replicated ordering and independent state validation, at the cost of consensus and node operation [37–39].

Pin the chain identifier, contract address, namespace, list, encoding version and authority bindings locally. Cache required schemas, issuer keys and recognition evidence under their own freshness rules. Otherwise remote procedure call (RPC) access, Ethereum Name Service (ENS) or DNS resolution, a wallet service or an off-chain dependency can remain on the critical path. ENS discovery is optional in Ethereum Request for Comments (ERC) 7506 [37]. Protect the local client and its RPC access, retain rollback-resistant checkpoints, monitor synchronization and finality, and use independent peers. A locally compromised or eclipsed node is not a trustworthy freshness source.

Table 5: Verification modes and external dependencies

Mode	Integrity and local availability	Lookup exposure and cost
Local full node	Validates chain state under the selected consensus policy; retained data remain readable offline.	No required external per-check query; sustained synchronization, disk, central processing unit (CPU) resources and operations.
Remote RPC	Availability and answers depend on the provider unless independently checked.	Provider sees requested state and timing; moving an endpoint onto a ledger does not remove this observer.
Light client with proofs	Authenticates headers and storage proofs; old proofs remain valid only for their old root.	Less local state, but on-demand proof requests can leak indices or word keys and depend on providers.
Signed local mirror cache	Authenticates issuer snapshots independently of mirror operator; finite offline reuse.	Scheduled bulk refresh can also remove per-decision requests; freshness and anti-equivocation need explicit rules.

Ethereum read calls do not create transactions. The JavaScript Object Notation (JSON) remote procedure call interface (JSON-RPC) distinguishes latest, safe and finalized state; the relying policy must select one and bind all reads to a single accepted block hash. Ethereum Improvement Proposal (EIP) 1186 specifies account and storage proofs, which must be checked against an independently authenticated root [39, 40]. Constant-time bit extraction applies to an already

materialized array or word. Disk access, trie traversal, proof validation, synchronization and whole-list decompression are separate costs. Checking z distinct entries still requires their corresponding local inspections, before dependency and assurance evaluation.

5.4.2 A proposed packed bitstring profile over ERC 7506

The cited specification is Draft ERC-7506, Trusted Hint Registry. It is not a standard named Ethereum-based Status List. It supplies address namespaces, list and key identifiers, 32-byte hint values, access management and public change events. It leaves application governance to implementers. The related ERC-5539 Revocation List Registry is marked Stagnant and uses Boolean revocation values per key [37, 41]. The following packed representation is this article’s proposed profile; it is not supplied by either ERC or by W3C Bitstring Status List.

For a list of L one-bit positions, allocate $J_L = \text{ceil}(L/256)$ hint words. At index i , use a profile-defined key for word $j = \lfloor i/256 \rfloor$ and offset $o = i \bmod 256$. Fix bit order, padding and serialization. If H_j is the unsigned integer represented by that word, decode:

$$B_i = (H_j \gg o) \& 1. \quad (9)$$

Bind the credential to the chain, contract, namespace, list, epoch, index, purpose and profile. Bind writer authority to scoped issuer recognition. Require an initialized list, authenticated length and lifecycle, valid contract version and valid list-wide status. An unset zero mapping must not be interpreted as an initialized active credential. A bare zero bit proves neither issuance nor truth; the credential and its signature supply separate evidence. If an existing W3C credential and signed snapshot are transported unchanged, preserve their standard decoding. A new on-chain status reference or masked decoder needs a separately specified method.

The displayed extraction rule uses a custom least-significant-bit offset. W3C lists place index zero at the left-most bit [12]. Directly importing 32 consecutive W3C bytes into a big-endian integer instead uses offset $255 - (i \bmod 256)$. A profile must declare either this mapping or a verified conversion. Test indices 0, 7, 8, 255 and 256 to expose byte and word boundary mistakes.

Whole-word replacement can lose concurrent changes when two delegates update different bits from the same old word. The primary unmasked revocation implementation should expose atomic bitwise OR updates, which set selected bits without clearing others, through a controlled writer, with every alternative setter constrained by the same policy. Suspension requires a separate reversible-state rule and version checks. A masked implementation needs equivalent semantic monotonicity or a one-transition record; OR on masked bits is incorrect. Nonces prevent particular replays but do not solve lost updates. Epoch changes must not silently reset a live credential’s revocation state.

5.4.3 Formal state transition and verification model

Define $\mathcal{M} = (\mathcal{S}, \delta, \mathcal{V}, \mathcal{P})$, where $\mathcal{P}$ is the observation and privacy-leakage model specified in Section 5.4.5. Fix a relying policy for recognized authorities, transition rules, accepted chain state, freshness and failure handling. The state space $\mathcal{S}$ contains contract states, pending operations and verifier replicas. L denotes status positions, whereas N denotes credential bundles. A committed state at block k is $\sigma_k = (\mathbf{B}_k, \iota_k, a_k, \nu_k, R_k, \omega_k)$. This tuple is an authenticated analytical view, not a self-referential object stored in the contract; it projects the full state and the contract also retains nonce and replay-protection state. Here $\mathbf{B}_k \in \{0, 1\}^L$ is the decoded semantic status vector, ι_k binds the initialized list and profile, a_k records writer and list authority, ν_k is the epoch and version, and R_k is the authenticated chain state root. The completeness

watermark ω_k means that all authoritative status decisions through that time are reflected in the view. It needs an explicit assertion signed by the recognized authoritative status source or a justified publication bound; a block timestamp alone does not supply it.

Engineering interpretation. A completeness watermark is an “all decisions through this time” statement. If the file-release service downloads a list at 10:04 whose authoritative watermark is 10:00, the relevant information is four minutes old. A new download, mirror timestamp or ledger block cannot make those decisions newer. The gate needs evidence that the authority has incorporated later decisions. If that evidence is unavailable, the service must apply its existing age limit to the old watermark.

The local state includes a validated copy $\hat{\sigma}_v(t)$, its block hash, selected finality evidence, accepted watermark ω_v , monotonic checkpoint and local clock-error bound κ_v . Pending transactions and local replicas are observation states, not immediately committed updates. For an authorized operation with nonce and expected version, the abstract transition is:

$$\delta(\sigma, (i, v, auth, nonce, \nu)) = \sigma'. \quad (10)$$

The transition preserves all other status positions and updates the specified bit only after authorization and profile checks. In irreversible revocation, $v = 1$ and $B'_i \geq B_i$; invalid operations leave the state unchanged. Consensus orders accepted transactions into successive states, subject to its stated fault and liveness assumptions. An issuer’s decision, transaction submission, inclusion, finality and application at a verifier are distinct events. A finalized transition does not establish that the issuer was entitled to make the underlying real-world claim. For an unchanged list, advance its completeness watermark only through an authenticated authoritative heartbeat or the declared bounded-publication inference. Count heartbeat and commitment-refresh traffic; an advancing chain does not automatically refresh the list.

Let OK_i include credential authenticity and binding, initialized-list membership, status-authority recognition, accepted chain evidence, anti-rollback checks and all relevant validity limits. Let $Fresh_v(t)$ require both the accepted chain-view policy and $t - \omega_v + \kappa_v \leq \Delta t_{\max}$. The binary status-admission predicate is:

$$\mathcal{V}(C_i, \hat{\sigma}_v, t) = \mathbf{1}\{OK_i \wedge Fresh_v(t) \wedge \hat{B}_{v,i} = 0\}. \quad (11)$$

An implementation should retain three diagnostic states: active, withdrawn and unknown. Missing, stale positive, uninitialized or unauthenticated evidence is unknown; it maps to zero admission, not to a factual assertion of revocation. An authenticated irreversible withdrawal can remain known despite age, provided its authority and epoch remain established. For a masked wire vector Y , obtain the semantic bit from the authenticated initial value using $B = Y \text{ XOR } r$. Passing this predicate is only one premise in the complete-record mandate and assurance intersection in Section 2. A ledger cannot replace that intersection.

5.4.4 Freshness and availability under delayed consensus

Measure from an authoritative withdrawal decision. Let p be decision-to-submission delay, b inclusion delay, τ the selected finality delay after inclusion, s propagation and local validation delay, and a a cache-to-gate application delay. With supported upper bounds on every term, the ledger path satisfies:

$$d_{DLT} \leq p_{\max} + b_{\max} + \tau_{\max} + s_{\max} + a_{\max}. \quad (12)$$

A sufficient deadline condition for this path is $p_{\max} + b_{\max} + \tau_{\max} + s_{\max} + a_{\max} + \kappa_v \leq \Delta t_{\max}$. This is stronger than comparing finality alone with the deadline. Ethereum finality depends on validator votes and can stall; inclusion can also be delayed. Without the required publication, network and consensus assumptions there is no finite deterministic bound [38]. Model empirical percentiles as probabilities, not worst-case guarantees. The probability of any stage exceeding its allocated budget bounds deadline failure by the union bound, without assuming independent stages.

Suppose connectivity is lost at t_0 with a still-acceptable watermark ω . The protected timing convention advances the enforced conservative age by at least the elapsed physical time. The expiry scheduler must realize this envelope, or stop earlier, with drift and restart accounted for. Without new evidence, the largest remaining freshness budget under this convention is:

$$H_{fresh} = \max\{0, \Delta t_{\max} - (t_0 - \omega) - \kappa_v\}. \quad (13)$$

The local copy remains readable after this interval, but new ordinary approvals must stop. Credential validity, lease expiry or other dependencies can shorten the interval further. For an illustrative 90-second accepted age, a 20-second-old watermark and a 2-second clock margin leave at most 68 seconds. If stipulated publication, inclusion, finality, synchronization and application bounds are 2, 3, 30, 4 and 1 seconds, their sum plus the margin is 42 seconds. These are test parameters, not Ethereum mainnet measurements. A 120-second upper bound on finality cannot certify that 90-second end-to-end target, although some actual updates may arrive sooner.

Proposition 2 Partition freshness. Let ω be the last accepted authoritative completeness watermark at partition onset t_0 . Assume the protected gate enforces the accepted-age rule under the timing convention above, its clock margin is κ_v , and no new authoritative evidence becomes available. Then the maximum remaining age budget is $H_{fresh} = \max\{0, \Delta t_{\max} - (t_0 - \omega) - \kappa_v\}$.

Proof. At elapsed physical time u , acceptance requires $t_0 + u - \omega + \kappa_v \leq \Delta t_{\max}$. Solving for u gives the stated bound. Other validity limits can only reduce it. A repeated fetch or local reread of the same state changes neither ω nor the bound. To show the information limit, couple two executions with the same observed history: after the partition, one authority withdraws and the other does not. No local algorithm can distinguish those executions from that history alone. An algorithm that accepts indefinitely in the second execution also accepts an unseen withdrawal in the first. Exact current-status knowledge and uninterrupted admission are therefore not jointly guaranteed under an unbounded partition.

The clock margin must remain valid over the complete offline interval, including drift and restart. A residual timestamp difference alone is not a certified physical-time duration; the expiry scheduler must enforce the stated elapsed-time envelope. For several mandatory objects, evaluate the minimum residual age budget across their accepted watermarks and validity ends. If the initial age rule already fails, zero remaining budget means that no new ordinary admission is permitted. At equality there is no positive remaining interval; implementations can require a strict pre-expiry margin.

Engineering interpretation. During a network outage, the file-release service may still read its local status copy. This preserves the ability to evaluate information already received. It does not reveal a withdrawal made elsewhere after contact was lost. Before the permitted age expires, the service can operate under its declared stale-information allowance. After expiry, it stops the affected ordinary releases. A local ledger and an authenticated snapshot cache face this same information limit; their update and recovery paths differ.

A useful hybrid permits an independently authorized signed withdrawal event to suspend reliance before ledger finality. It must not permit an unfinalized grant to create authority under the same

shortcut. Publish and reconcile the durable ledger version later. Reorganization, conflicting authority and recovery require explicit rules; a mempool observation alone is not an authoritative suspension. The supported DLT path enters the minimum over supported push, reconciliation and expiry paths. Withdrawal can still be fast when finality is slow, but the fast path has its own trust and availability assumptions.

5.4.5 Lookup privacy and the public update transcript

Let Q be a sequence of verifier lookups and Z the observer’s prior information. Define the central observation for a request as $m = (IP, t, list, headers, identity)$, retaining only fields actually visible. Hypertext Transfer Protocol Secure (HTTPS) protects the path from many intermediaries but the endpoint sees its request. A remote RPC call can additionally expose word keys or contract arguments. Define lookup leakage using conditional mutual information:

$$\mathcal{P}_X = I\left(Q; O_X^{lookup} \mid Z\right), \quad \Delta\mathcal{P} = \mathcal{P}_{central} - \mathcal{P}_{local}. \quad (14)$$

If every check uses already resident data, synchronization and prefetch are independent of Q , the local interface is trusted, and no query-dependent telemetry or observable side channel escapes, then $O_{local}^{lookup} = \emptyset$ and $\mathcal{P}_{local} = 0$. Thus the incremental remote lookup channel is eliminated under those conditions. This is not zero total privacy leakage: peers can see node participation, public updates expose issuer activity, and credential presentations can reveal identities. Conditioning on Z separates this claim from information already available elsewhere.

For q uncached direct HTTP checks, the endpoint obtains q request records; local checks obtain zero such records at that endpoint, a 100% reduction in this specific count. A cache with m scheduled bulk refreshes exposes m fetch records instead, and may already reveal no per-decision timing. Consequently there is no universal number of privacy bits saved or unconditional ranking over a well-designed federated cache. Measure request counts, timing correlation, inferred verifier identity and queried-list or index recovery separately. On-demand light-client proofs and remote RPC restore a lookup channel unless separately protected.

ERC-7506 publicly emits the changed key and new value [37]. The proposed packed profile therefore exposes changed words, and consecutive values identify changed bits. Include senders, calldata, function selectors, event topics, fees, batching and timing in Section 6’s observation O . Cover updates must follow the same observable path as genuine updates. A separate decoy writer or real-only event identifies labels regardless of matching bit frequencies. Fixed masks cancel in differences; if an observer sees initialization when all semantic bits are known active, that snapshot reveals the initial mask itself. Unknown-membership protection then rests on hidden allocation and matched cover histories.

5.4.6 Committed objects authenticated trees and alternative publication

Table 6 includes content-addressed storage through the InterPlanetary File System (IPFS), where a content identifier (CID) addresses an object. It also compares RSA-based accumulators and layer 2 (L2) systems that settle to a layer 1 (L1) base ledger.

Table 6: Alternative publication and proof patterns

Pattern	Useful property	Cost and limitation
Complete packed state on DLT	Local authenticated status after synchronization; shared update ordering.	Writes, consensus delay, public update history and replicated state; no automatic issuer truth or freshness.
On-chain digest with signed mirrors or IPFS	Small commitment; large snapshots can use caches and independent replicas.	Both an accepted commitment and available matching bytes are needed. A CID alone does not retain data.
Merkle status map	A bit or word plus a logarithmic authentication path can prove its value under a root.	Current root, data and proof availability remain necessary. Index-revealing proofs are not anonymous proofs.
RSA or other accumulator	Some constructions offer compact membership or non-membership witnesses.	Witness maintenance, setup and manager assumptions vary. RSA and pairing constructions are not post-quantum.
L2 publication	Can amortize transactions and reduce some publication costs.	Sequencer acceptance, data availability, proof or dispute completion and L1 settlement are different conditions.
Participant state channel	Can amortize repeated updates among a fixed group.	Co-signature and dispute rules complicate public revocation; counterparties must not veto unilateral withdrawal.

For committed off-chain snapshots, bind the digest to the issuer, list, profile, monotonic version, authoritative as-of time and validity. A 32-byte digest fits one hint; an arbitrary CID need not fit and requires a defined codec or manifest. Retain independent pins or mirrors and locally prefetch the working set. IPFS pinning prevents local garbage collection, while public content routing can expose retrieval and provider relationships [42, 43]. A valid old commitment does not establish the latest status. Content addressing solves object integrity, not confidentiality or continued availability.

A balanced Merkle status map with J leaves has authentication paths of order $\log J$; its root may be signed or ledger-anchored. It must define what an active leaf means and prove the relevant value. An append-only transparency log can audit signed versions, but inclusion alone is not a proof of current non-revocation; detecting split views additionally requires witnesses, gossip or another consistency mechanism. Request for Comments (RFC) 9162 supplies inclusion and consistency-proof techniques, not a credential-status protocol [44].

RSA accumulators and Merkle trees are distinct constructions. Dynamic accumulator research supports compact proofs with construction-specific update and trust assumptions [23, 45]. Neither a small accumulator value nor a small Merkle root makes witness or data distribution free. Local witness verification can avoid per-check requests only while the witness and authenticated root meet the same freshness rule. A plain Merkle path may identify its leaf; anonymity requires an additional proof protocol. Hash-based authentication also needs appropriate security parameters and quantum-resistant signatures before an end-to-end PQC claim is justified.

For L2, state exactly which acceptance stage counts toward the deadline. Optimistic dispute

windows and validity-proof generation have different implications; an early sequencer promise cannot be relabelled settled finality. Include sequencer outage, censorship, escape-path delay, data availability and bridge assumptions [46, 47]. State channels normally require participants to sign updates and support disputes [48]. Their suitability is limited for status that arbitrary new verifiers must consume. A public fallback for unilateral withdrawal is essential. Ephemeral caching remains an independent layer across every option, with startup, loss and expiry costs.

The companion workload model keeps design C’s protected enforcement and compares two publication paths: issuer-signed snapshots from independently operated mirrors and the proposed packed local-ledger profile. Ledger-committed off-chain snapshots remain an optional extension for testing reduced on-chain payload and ordered publication. Their root authenticity, object availability and issuer authority require separate checks. The next subsection defines the common freshness and enforcement contract.

ShadowPath provides a related holder-side approach: it reconstructs status witnesses locally and proves non-revocation against a selected authenticated root without revealing the index to the verifier. Its stated guarantees exclude issuer–verifier collusion and synchronization traffic [49]. This separates private presentation from the local verifier-read architecture studied here. It does not replace the need to establish current authoritative state or to analyse publication histories.

5.5 Formal architecture contracts

Let $\mathcal{L}_g$ be the status and recognition objects required for candidate g . Object l has an authenticated authoritative as-of time s_l and a relying-policy maximum accepted age τ_l . Let v_g be the earliest applicable evidence, mandate, continuation or policy validity end, b the lease issue time and ℓ_g^{lease} its permitted duration. A proposed freshness rule is:

$$e_g \leq \min \left\{ b + \ell_g^{lease}, v_g, \min_{l \in \mathcal{L}_g} (s_l + \tau_l) \right\}. \quad (15)$$

The gate accepts the record only before its authenticated expiry, allowing conservatively for the specified clock error. Re-signing or re-packaging old evidence without authoritative revalidation does not reset its as-of time; a new authoritative assertion of current status can. Any parent dependency missing a supported validity or freshness rule prevents an ordinary approval. This is a proposed deployment contract, not an additional field semantics supplied by W3C.

For a dependency withdrawn at its authoritative source, let d_i^{push} , d_i^{recon} and d_i^{expiry} be supported upper bounds from withdrawal to protected enforcement by each path. Include delivery, validation, state application and scheduling. Then:

$$d_i^{enforce} \leq \min \{ d_i^{push}, d_i^{recon}, d_i^{expiry} \}. \quad (16)$$

An unavailable or unbounded path contributes infinity. For periodic reconciliation, the bound includes the maximum poll gap and completed fetch, validation and application. The expiry path requires bounded clock error, trustworthy as-of times and a publisher that does not issue fresh active assertions after authoritative withdrawal. A short child lease renewed from stale evidence does not independently supply the claimed withdrawal bound. Detection before authoritative withdrawal remains separate.

Let $\mathcal{C}_{cache}$ denote actual cache domains, $\mathcal{L}_c$ the lists used by cache c , f_{cl} completed full-refresh rates and W_l complete signed snapshot bytes, or the event-weighted mean size of versions actually fetched when size varies. Let $\dot{B}_{event}$, $\dot{B}_{staple}$ and $\dot{B}_{other}$ count their actual transmitted bytes over declared links. The architecture-level accounting is:

$$\dot{B}_{total} = \sum_{c \in \mathcal{C}_{cache}} \sum_{l \in \mathcal{L}_c} f_{cl} W_l + \dot{B}_{event} + \dot{B}_{staple} + \dot{B}_{other}. \quad (17)$$

Count each byte on each reported link once. Origin egress, internal replication, cache-to-gate traffic and job attachments are separate measurements. A refresh interval may substitute its reciprocal for f only under the corresponding periodic-fetch assumption. Conditional HTTP responses and event-coalesced refreshes need their actual sizes and rates. Logical status checks still scale with the evidence slice even when external download work is shared.

For CPU work, let c_g be residual per-proposal work, ν_{cl} the rate of applied list updates at cache c , and c_{cl}^{update} the work of decoding and invalidation per update there. Let f_{cl}^{verify} count actual proof verifications and c_l^{proof} their measured cost. Use disjoint accounting categories:

$$\dot{C}_{CPU} = \lambda_a \mathbb{E}[c_g] + \sum_{c,l} \nu_{cl} \mathbb{E}[c_{cl}^{update}] + \sum_{c,l} f_{cl}^{verify} c_l^{proof}. \quad (18)$$

The update term includes changed-position analysis and affected-decision work. Proof verification must not also be charged in that term if it is counted separately. Post-quantum signatures change measured proof costs and object sizes. Cryptographic batch verification requires a sound scheme-specific algorithm, including invalid-batch isolation and fallback costs.

For a changed dependency x , define $I_x = \{g : x \in C_g\}$, with an extended dependency relation for status and policies. A reverse-index lookup can locate affected records in output-sensitive time. Re-evaluation work is $\sum_{g \in I_x} cost(g)$, rather than all active records, only when that dependency index is sound. A root shared by every decision gives no such saving.

For continuously live leases of equal duration ℓ , steady renewal demand is approximately N_{live}/ℓ before retries, early renewal and churn. Let Q be bounded queued work ahead of a containment job, w_{job} its own bounded work, and R_{srv} a guaranteed reserved service rate in the same work units per second. A dedicated first-in, first-out (FIFO) service with bounded scheduling latency θ gives:

$$d^{service} \leq \theta + \frac{Q + w_{job}}{R_{srv}}. \quad (19)$$

This last inequality requires a real service guarantee, bounded work and no overtaking, or Q must also bound all later higher-priority interference. It is not obtained from average CPU utilisation. Network transmission, source detection and receiver execution remain outside it unless explicitly included. The same contract must be checked for each relevant containment path.

A finite R_{srv} is insufficient under unbounded backlog. Admission must preserve a supported Q_{max} , including non-preemptible blocking and all admitted interference. CPU, memory, input/output (I/O) and network bottlenecks each need their own service assumptions. Section 12 gives a backlog trigger derived from the deadline rather than from average utilisation.

5.6 Composite designs and assessment

Compare the following configurations at equal agent population, evidence graph, list encoding, privacy observer and maximum accepted status age. Keep complete-record gates, operation accounting and pending-effect semantics in every main comparison. Removing these controls is a separate omission counterexample.

Table 7: Composite enforcement architectures

Design	Configuration	Expected result to test
A Local periodic verification	Each verifier retains its own origin-signed snapshots and checks finite approvals.	Simple baseline; duplicated fetches and proof work, with fewer shared-cache failure dependencies.
B Shared periodic verification	Domain caches share authenticated versions; jobs may staple origin evidence.	Lower external egress and repeated signature work; larger cache compromise or outage impact.
C Event assisted bounded enforcement	B plus event-triggered refresh, reconciliation, scoped dependency re-evaluation and reserved enforcement capacity.	Preferred reference architecture; quicker typical withdrawal and controlled overload, under explicit timing contracts.
D Delegated cross-domain decisions	C plus reliance on a recognised remote attester’s scoped decision manifest.	Potentially less repeated evaluation, but adds delegated semantic trust and shared-failure exposure.

Design C is the reference architecture for the companion study. Its executable workload comparison varies a hosting model between independently operated signed mirrors and local ledger state while retaining the same model of a protected receiver. Designs A and B provide optional cache or event ablations for deployment study. Design D adds explicit trust in a remote attester’s evaluation and is not equivalent to forwarding origin-signed evidence.

A concrete workload calculation illustrates the separation of costs. Suppose 1000 verifiers share 10 domain caches. Every verifier or cache refreshes the same forty lists once per minute. Using the Section 8 twenty-status and 50%-decoy fixture, unmasked snapshot bytes are 23743.1 per full population set and masked bytes are 875083.8, excluding envelopes. Origin-to-cache snapshot traffic falls by a factor of 100 when 1000 independent fetchers become 10 shared fetchers: from 395718.3 to 3957.2 bytes/s unmasked, or from 14584730 to 145847.3 bytes/s masked. These are workload calculations, not measured deployment rates. Internal forwarding, events and stapled attachments add traffic, and logical credential checks remain.

For a timing scenario, assume completed enforced push has a supported 2-second bound, reconciliation a 65-second bound, and protected freshness expiry a 90-second bound including clock error. The resulting withdrawal-to-enforcement bound is 2 seconds while all premises hold, 65 seconds if push fails, and 90 seconds if both communication paths fail. Without the expiry premise there is no finite partition bound. These are stipulated test parameters. They say nothing about the earlier detection interval or an already accepted irreversible effect.

6 Matched decoys and the privacy objective

6.1 Unknown membership and known credential status

The proposed privacy objective is to limit an observer’s ability to identify which unknown bit positions represent real credentials, or to infer business activity from public status histories. A recipient who already knows a credential’s index can still follow that credential’s status. Membership privacy and issuer-activity privacy need separate experiments. The first does not establish the second.

Study a fixed cohort first. Secretly assign genuine and decoy labels to positions. A decoy belongs

to no credential. For irreversible revocation, both kinds make at most one semantic transition during the modeled lifetime. A random initial mask is optional:

$$x_i(t) = \mathbf{1}\{T_i \leq t\}, \quad Y_i(t) = r_i \oplus x_i(t), \quad r_i \sim \text{Bernoulli}(1/2). \quad (20)$$

For a genuine position, T_i is its revocation time; for a decoy it is a sampled cover-event time. The credential carries its authenticated list reference, position and initial mask in the masked profile. A production mask requires a cryptographic random source. In the unmasked comparison, set every $r_i = 0$. Reversible suspension needs matched state-transition and dwell-time laws instead of a one-transition model.

The full allocation vector $J \in \{0, 1\}^L$ and mask vector r are maintained by the status operator. A holder and its verifier receive the credential's authenticated index and r_i ; the full vectors are not disclosed through that interface. Thus only the master allocation is intended to remain operator-private, and even that claim excludes leaked issuance mappings. Masks are independent across positions, labels and semantic event histories. These assumptions support the incompressibility calculation; Bernoulli marginals alone do not.

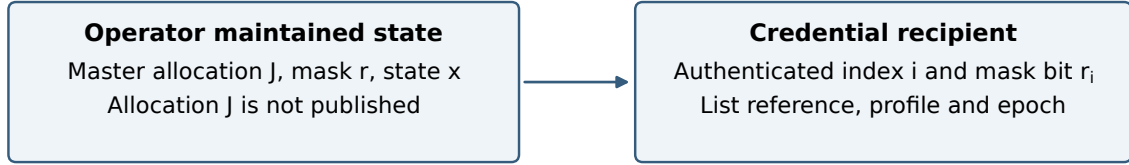
Snapshots reveal $Y_i(t) \oplus Y_i(s) = x_i(t) \oplus x_i(s)$. Matched decoys address the resulting uncertainty about which observed changes are genuine. They do not prevent detection that some positions changed. Arbitrary repeated decoy toggles are a negative control: repeated reversals can identify decoys when genuine revocations are absorbing.

For ledger publication, extend the observation to all public transactions, senders, calldata, events and word-level updates described in Section 5.4.5. Matching bit histories alone is insufficient. The publisher must also account for initialization, allocation and rotation disclosures; fixed masks provide no status concealment from an observer who sees a known-active initialization.

At initialization, $x_i(t_0) = 0$ for all displayed positions. The unmasked vector is all zero and does not itself disclose J . The masked vector equals r . If that known-active initialization is public, an observer learns r from the snapshot and can decode subsequent semantic transitions. With an unobserved initialization, independent masks can hide a single snapshot's semantic values; fixed masks still reveal transition locations across snapshots. Figure 2 separates these properties from unknown-membership protection.

Initial masking and matched cover events

Experimental profile • Twelve illustrative positions • Irreversible status



Only the relevant per-entry fields are disclosed in the credential

Public snapshot transcript

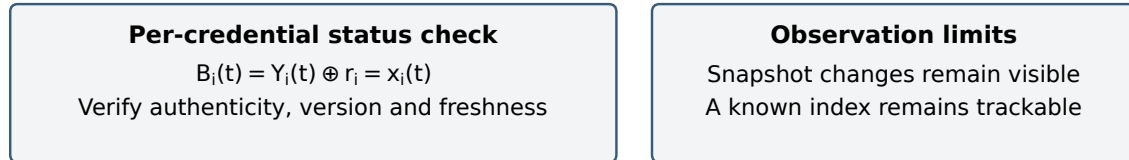
	0	1	2	3	4	5	6	7	8	9	10	11
t_0 : initial	0	1	1	0	1	0	0	1	0	1	1	0
t_1 : position 2 changes	0	1	0	0	1	0	0	1	0	1	1	0
t_2 : position 7 changes	0	1	0	0	1	0	0	0	0	1	1	0

Operator events: genuine $x_2 : 0 \rightarrow 1$ at t_1 ; matched cover $x_7 : 0 \rightarrow 1$ at t_2 .

The illustrated times show encoding mechanics, not evidence that event laws match.

Unmasked initial $r_i = 0$	0	0	0	0	0	0	0	0	0	0	0	0
-------------------------------	---	---	---	---	---	---	---	---	---	---	---	---

Zeros reveal initial semantic state, not the genuine versus decoy allocation.



If $x(t_0) = 0$ is known, then $Y(t_0) = r$: initial masks are observable.

Unknown-membership protection requires matched complete histories and stated side information.

Figure 2: Decoy lifecycle and observation boundaries. The operator maintains the full allocation and mask arrays; a credential discloses its own index and initial bit to its holder and verifier. The known-active initial masked snapshot reveals the mask. An all-zero unmasked snapshot does not identify genuine positions. Genuine and cover events both make one irreversible semantic transition. Fixed-mask differences reveal transition positions; membership protection requires matched complete observation histories.

6.2 Conditional membership proposition

Let $J_i = 1$ mark a genuine position. Let O contain the full public transcript: all list trajectories, timestamps, joint changes, lengths, placement, publication schedule and rotations. Let Z contain attacker side information. Write $\alpha_Z = \Pr(J_i = 1 \mid Z)$, and f_R and f_D for conditional transcript likelihoods given a real or decoy position. Bayes' rule gives:

$$\Pr(J_i = 1 \mid O = o, Z) = \frac{\alpha_Z f_R(o \mid Z)}{\alpha_Z f_R(o \mid Z) + (1 - \alpha_Z) f_D(o \mid Z)}. \quad (21)$$

Proposition 3 Conditional membership. Fix side information $Z = z$ and a protected position i . Assume $0 < \alpha_z = \Pr(J_i = 1 \mid Z = z) < 1$. Let the real and decoy transcript distributions have densities $f_R(\cdot \mid z)$ and $f_D(\cdot \mid z)$ with respect to a common dominating measure. If these densities agree almost everywhere, then $\Pr(J_i = 1 \mid O, Z = z) = \alpha_z$ almost surely.

Proof. The mixture density is the prior-weighted sum in the denominator of Bayes' formula. Substitution of equal densities cancels their common factor wherever the mixture has positive measure. Zero-measure transcripts do not affect the almost-sure statement. Priors zero and one remain unchanged trivially. The argument applies to the complete transcript and stated side information, rather than to one bit's marginal event frequency.

For a selection rule $I = s(O, Z)$, the same statement holds with the corresponding conditional prior if every selectable coordinate has the asserted posterior. A uniform posterior bound γ is preserved because $\Pr(J_I = 1 \mid O, Z) = \sum_i \mathbf{1}\{I = i\} \Pr(J_i = 1 \mid O, Z) \leq \gamma$. This does not permit additional undeclared side information or interventions after observing the data.

A mathematical construction chooses the hidden real/decoy allocation uniformly at a fixed count, then generates all public histories from a joint law independent of that allocation. It proves feasibility of the probabilistic premise in an artificial experiment. It is not a deployment algorithm: genuine withdrawals are driven by external incidents that the publisher cannot choose freely. Practical cover generation must match the observable conditional law under those incidents. A proof of average-rate matching is therefore insufficient.

Engineering example. Imagine one hundred hidden positions, fifty genuine and fifty decoys. Under the proposition's complete matching assumptions and without distinguishing side information, observing their histories does not improve a position's initial one-in-two chance of being genuine. This does not hide the status of a credential whose index is already known. If the subcontractor knows its own credential's position, it can still follow that status. The proposed cover protects uncertainty about unknown membership, not secrecy of every revocation.

This is a conditional membership statement, not a new cryptographic theorem. It holds for masked and unmasked bits under the same assumptions. Random initial masks are therefore not necessary for this particular objective. Any extra benefit from masking needs a distinct threat model, such as a limited single-snapshot observer, and its own evaluation.

Matching only average rates is insufficient. For example, genuine credentials can revoke in simultaneous pairs while independent decoys have the same marginal lifetime distribution. Exact pair coincidences then reveal genuine positions in a continuous-observation model. Match batch sizes, shared-source shocks, birth cohorts, expiry, retention and correlations with public incidents. Known indices, leaked mappings and issuer metadata can change the prior to one.

6.3 A quantitative target for imperfect matching

Suppose every transcript possible for a real position satisfies $f_R/f_D \leq e^\varepsilon$, conditional on the declared side information. Then a posterior target γ is supported by the following sufficient bound:

$$\Pr(J_i = 1 \mid O, Z) \leq \frac{e^\varepsilon \alpha_Z}{1 - \alpha_Z + e^\varepsilon \alpha_Z}, \quad \alpha_Z \leq \frac{\gamma}{e^\varepsilon(1 - \gamma) + \gamma}. \quad (22)$$

With a real/decoy pool that is exchangeable conditional on Z , let R and D denote its real and decoy slot counts, so $\alpha_Z = R/(R + D)$. A sufficient reserve ratio for the stated likelihood-ratio envelope is:

$$\frac{D}{R} \geq e^\varepsilon \frac{1 - \gamma}{\gamma}. \quad (23)$$

Proposition 4 Approximate matching. Suppose $f_R \leq e^\varepsilon f_D$ almost everywhere for the full conditional transcript, with $\varepsilon \geq 0$. Posterior odds equal prior odds times the likelihood ratio:

$$\frac{\Pr(J_i = 1 \mid O, Z)}{1 - \Pr(J_i = 1 \mid O, Z)} = \frac{\alpha_Z}{1 - \alpha_Z} \frac{f_R(O \mid Z)}{f_D(O \mid Z)}. \quad (24)$$

Proof. Divide Bayes' posterior by its complement, apply the likelihood-ratio upper bound and solve the resulting inequality for α_Z . This gives the bound displayed above. For an exchangeable pool conditional on Z , substituting $\alpha_Z = R/(R + D)$ yields $D/R \geq e^\varepsilon(1 - \gamma)/\gamma$ as a sufficient reserve condition. It is tight as a worst-case guarantee based only on the stated likelihood-ratio bound; a particular process may need less cover when its attainable ratios are smaller.

This is a conditional, one-sided likelihood-ratio guarantee. It is not differential privacy, nor a guarantee about all issuer-level statistics. Known positions can have prior one. An event possible only for real entries violates every finite likelihood-ratio bound. A separate activity-privacy experiment must define neighbouring issuer histories and the observations or decision problem to be protected.

The likelihood-ratio envelope is a full-history assumption. Conditional on one fixed initial side-information value Z , let each successive observation increment include all newly disclosed side information as well as the release itself. If its conditional likelihood ratio is bounded by e^{ε_j} after every permitted observation history, the probability chain rule bounds the finite complete-history ratio by $e^{\sum_j \varepsilon_j}$. Omitting newly disclosed side information or using unconditional one-release bounds does not establish that composition. An indefinite publication stream needs a separate bounded complete-history argument or provision for its accumulating privacy cost.

Exact matching gives $\varepsilon = 0$ and decoy fraction $d \geq 1 - \gamma$. The likelihood-ratio condition must cover whole histories, including rare events; similar-looking plots do not prove it. This is not a differential-privacy claim. An unmatched event with positive real likelihood and zero decoy likelihood rules out any finite bound.

For a single interval among unflipped positions, the posterior after a flip is $\frac{\alpha_t q_R}{\alpha_t q_R + (1 - \alpha_t) q_D}$. Equal conditional hazards preserve the survivor prior. Raising the decoy rate can reduce that event's posterior while making non-flips or longer histories easier to classify. The full-history condition remains the research target.

6.4 Reserve lifetime and continuing issuance

Let F_D be an absolutely continuous cover-event lifetime distribution and λ_D its hazard. For an initial reserve D_0 , expected unflipped decoys and their instantaneous event rate, while survival is positive, are:

$$\mathbb{E}[D_{\text{surv}}(t)] = D_0 [1 - F_D(t)], \quad \eta_D(t) = D_0 [1 - F_D(t)] \lambda_D(t). \quad (25)$$

If the lifetime law has point masses, the survivor identity still holds, but expected event counts at each such time use the initial reserve multiplied by that probability mass. For a fixed cohort with matched lifetimes, real and decoy survivor expectations decline in the same ratio. Depletion alone does not increase the relative factor. Continuing real issuance requires matched cover births and cohort metadata. In the simplest equal-hazard model, maintaining proportional survivor expectations from matching initial conditions and a common age-independent hazard requires compatible birth rates $b_D = r b_R$. Spent positions remain occupied until the profile permits retirement. Reuse must not change the meaning of any live credential reference. List rotation or replenishment can itself leak activity and belongs in O.

7 Net payload and scaling factors

7.1 Physical capacity

Let L be physical list bits, w the width of a status value, d the fraction of slots reserved for decoys, and u the occupancy of the remaining genuine slots. Let m_L be the number of allocated genuine status entries. Then:

$$P_L = \lfloor L/w \rfloor, \quad m_L = \lfloor u \lfloor (1-d)P_L \rfloor \rfloor, \quad \eta_{\text{raw}} = w m_L / L \approx u(1-d). \quad (26)$$

Unused genuine capacity does not automatically provide privacy cover. It can count as cover only when its observable history follows the same decoy model; otherwise an observer may distinguish it.

The useful raw status payload is $w m_L$ bits. If independent bundles require k distinct entries and must fit wholly within a list, capacity is $\lfloor m_L/k \rfloor$ complete bundles. If bundles may span lists, total population capacity uses pooled entries and list rounding. These alternatives must not be mixed.

$$G = \text{ceil} \left(\frac{M}{m_L} \right), \quad F_{\text{raw}} \approx \frac{k_{\text{pop}} w}{u(1-d)}. \quad (27)$$

The list-count expression requires positive genuine capacity; with none, no finite number of lists can serve a positive demand. The asymptotic factor compares to one occupied one-bit status per bundle, equal populations and equal minimum list sizes, ignoring rounding. It counts status storage, not validation traffic. For exact matched histories, full occupancy and one-bit statuses, the asymptotic privacy requirement is $F_{\text{raw}} \geq k_{\text{pop}}/\gamma$. Finite-population comparisons must instead use the integer list counts: both designs can fit in one minimum-size list, giving an exact ratio of one. For the imperfect-matching envelope, a sufficient asymptotic factor is $k_{\text{pop}} [1 + e^\epsilon (1 - \gamma)/\gamma]$.

Table 8: Conditional privacy targets and physical capacity

Posterior target	Decoys	Genuine slots per 16 KiB list	Raw factor for k = 20	Lists for 131072 bundles
50%	50%	65,536	40×	40
10%	90%	13,107	200× asymptotic	201
5%	95%	6,553	400× asymptotic	401

This table assumes hidden labels, exact full-history matching, one status per distinct credential, no credential sharing, full genuine occupancy and a separate posterior cap in every list. The finite list counts use $\lfloor \gamma L \rfloor$ genuine slots. A likelihood ratio as large as two changes the asymptotic factors to 60×, 380× and 780×. These are design calculations, not measured privacy guarantees.

Engineering interpretation. Twenty distinct statuses need twenty genuine entries per file-release evidence bundle in this scenario. With half the list reserved for decoys, those entries occupy forty physical positions per bundle on average. That is the source of the forty-fold raw factor. The factor falls if bundles share status entries, and finite-list rounding can increase allocated capacity. None of these counts requires forty network requests per release: caching and list grouping determine the request workload separately.

7.2 Compression and transmitted payload

Let B_G be the GZIP byte count and E all additional signed-list envelope bytes, excluding the encoded bitstring. For the evaluated JSON-compatible encoding, the transmitted object size is:

$$W = E + 1 + \text{ceil} \left(\frac{4B_G}{3} \right). \quad (28)$$

The one-byte term is the multibase prefix. Transport headers, fetch cadence and proof verification are separate. A fully random initial mask across the entire list makes snapshots approximately incompressible, so $B_G \approx L/8$ plus framing. The known position is already required by ordinary status references; incremental mask information is w bits per genuine entry before serialization, plus explicit profile and epoch bindings. Do not count the ordinary index twice.

For an ideal independent and identically distributed (i.i.d.) sparse baseline with revoked fraction p , define $H_2(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$. Ignoring envelopes, rounding and finite compression overhead, the masked comparison is:

$$F_{\text{entropy}} \approx \frac{k_{\text{pop}} w}{u(1 - d)H_2(p)}. \quad (29)$$

At $p = 0.001$, $H_2(p) = 0.01140776$. With $k = 20$, $d = 0.5$ and $w = u = 1$, this ideal coding ratio is 3506.39×. It is not a GZIP prediction. An unmasked matched-decoy pool can retain the same sparse one-fraction as genuine entries, giving approximately the raw multiplicity factor when occupancy, coding and event laws are matched. This makes it a necessary comparison.

7.3 Changes between versions

For a one-bit, fully occupied real/decoy pool in one interval, let q and q_D be marginal transition probabilities. The expected changed fraction is $f = (1 - d)q + dq_D$. Under independent homogeneous Bernoulli transitions, $LH_2(f)$ is the entropy of the change vector; more generally it is an

upper bound obtained from the marginal entropies. A known partition with independent transitions has entropy $L[(1-d)H_2(q) + dH_2(q_D)]$. Correlations and authenticated delta framing change actual bytes.

Matched monotone decoys keep histories plausible and add cover transitions. In an already random, fixed-length full snapshot, further flips do not add snapshot bytes. They may add publication events, delta bytes, CPU work and retained versions. Count those costs separately. A delta protocol requires authenticated base versions, ordering, recovery and rollback handling; it is not supplied by these entropy equations.

7.4 Ledger storage publication and cover traffic

Hosting does not change the useful fraction $u(1-d)$ or raw capacity factor for an identical bitstring. For m useful one-bit statuses and a direct packed profile, let $J = \text{ceil}(L/256)$. The logical value payload and useful fraction are:

$$S_{\text{value}} = 32J \text{ bytes}, \quad \eta_{DLT, \text{value}} = \frac{m}{256J}. \quad (30)$$

Keys, initialization records, authority data, trie and database overhead, transaction receipts and client state are additional. Direct Ethereum Virtual Machine (EVM) words do not receive the GZIP savings of Section 8. Conversely, storing a compressed blob sacrifices direct extraction until it is decoded locally. Current state, retained transaction history and archive-state storage are distinct measurements; updating a fixed word does not allocate a fresh status word, though it adds publication history. Ordinary current-state verification does not require retaining every historical state [50].

One 131072-bit list contains 512 words and 16384 logical value bytes. Twenty disjoint statuses per bundle with 50% decoys use forty such lists: 20480 words and 655360 bytes, still 40 times the one-list raw baseline. This is neither the 1475.44-fold encoded snapshot ratio nor a gas ratio. Replicating these values at r maintained nodes accounts for r times the payload as a logical full-replica total, before each node's infrastructure costs. Count dedicated-node costs separately from marginal use of an existing node.

For an illustrative i.i.d. revoked fraction p in an unmasked one-bit list, the expected number of nonzero words is $J[1 - (1-p)^{256}]$. For i.i.d. uniformly masked words it is $J(1 - 2^{-256})$, essentially J . At $p = 0.001$ and $J = 512$ these are approximately 115.69 and 512, a 4.43-fold difference in nonzero value slots. This is a separate sparse-state illustration, not the fixed-count GZIP fixture. Allocation, database bytes and gas must be measured; slots and gas are not proportional in general.

Within a batch of n independent updates whose word choices are uniform over J words, coalescing touches on average:

$$\mathbb{E}[U(n)] = J \left[1 - \left(1 - \frac{1}{J} \right)^n \right]. \quad (31)$$

Actual unique indices, correlated withdrawals and cover batches require their actual placement law. If genuine and decoy positions have identical hazards and proportional at-risk populations, adding decoys multiplies expected transition volume by $1/(1-d)$. It need not multiply transactions or touched words by that factor, because updates can share batches and words. Batching amortizes overhead but adds waiting time that belongs in the withdrawal deadline. Reserve capacity for urgent genuine updates without publishing a distinguishing real-only path.

Use a measured gas model rather than a fixed currency forecast. For batch b , let U_b be touched words and $g_{b,j}$ the storage cost under the chosen fork and original/current/new values. Then:

$$G_b = G_b^{base} + G_b^{auth} + G_b^{data} + G_b^{events} + \sum_{j \in U_b} g_{b,j}. \quad (32)$$

Storage metering depends on the transition and transaction context [51]. Report initialization separately from steady updates; record fork, contract code, batch size, calldata, receipt bytes, fee schedule and executed gas. If the sustainable publication budget is C gas per second, the batch-arrival rate is λ batches per second and mean batch work is $E[G]$ gas, then $\lambda E[G] < C$ is a subcritical mean-load condition, not a deterministic delay certificate. It is not a worst-case inclusion guarantee on a shared public ledger. Add node synchronization, proof traffic and local CPU once in the disjoint accounting categories of Section 5.5.

An anchored digest makes the on-chain commitment small but leaves snapshot distribution, signatures, masks and decoys off-chain. Update frequency, retained versions and replications remain costs. Report state bytes, history bytes, verifier traffic, gas and synchronization separately; multiplying the raw factor, GZIP ratio, replica count and gas overhead into one universal scaling factor would be misleading.

8 Reproducible numerical results

8.1 Snapshot codec fixtures

Purpose of the codec comparison. GZIP provides a standards-based baseline for measuring the representation cost of random masking. Section 2.2 of W3C Bitstring Status List v1.0 requires credentialSubject.encodedList to contain a GZIP-compressed bitstring represented as Multibase-encoded base64url without padding; Section 3.3 specifies the compression step [12]. This encoding is part of the status-list credential, independent of HTTP transport compression. Section 2.2 of the v1.1 First Public Working Draft retains this requirement [13].

The fixture holds this codec fixed to measure the loss of compressibility under random masking. It does not establish standards conformance of the experimental masked status semantics. Table 9's random snapshot grows from 16,384 raw bytes to 16,407 GZIP bytes: 23 bytes (0.14%) overhead, with no size saving.

A reproducible compression fixture uses $L = 131072$ bits, 30 trials per case, seed 20261008, Python 3.12.14, the zlib compression library version 1.3.2, GZIP level 6 and mtime zero. Sparse trials choose a fixed number of uniformly placed ones: 131 for nominal $p = 0.001$ and 1311 for nominal $p = 0.01$. Random trials use independent reproducible byte fixtures. This generator is for measurement, not production cryptographic masking.

Table 9: Measured sizes of one physical snapshot

One list	Mean GZIP bytes	Mean encoded bytes with prefix
Sparse 0.1%	443.83	593.10
Sparse 1%	2,166.60	2,890.13
Random initial mask	16,407.00	21,877.00

These fixtures compare equal bundle populations and equal marginal status densities. They do not

instantiate temporal privacy histories. If all twenty independent statuses were required, a marginal withdrawal probability of 0.001 would imply bundle invalidity $1 - 0.999^{20}$, approximately 0.019811, rather than the single-status baseline’s 0.001. Therefore the codec factors do not compare equivalent authority or incident semantics. The architecture experiments must control those semantics separately.

For equal populations of 131072 bundles, the baseline has one status per bundle and one sparse list. A bundle with twenty disjoint one-bit statuses and 50% decoys needs forty lists. The measured aggregate comparison is:

Table 10: Equal-population snapshot codec comparison

Representation	Lists	Raw bytes	Encoded bytes	Ratio to baseline
Single-status sparse baseline	1	16,384	593.10	1×
Unmasked sparse proxy	40	655,360	23,743.10	40.03×
Masked random proxy	40	655,360	875,083.80	1475.44×

Reading the comparison. Both forty-list variants have the same 655,360-byte raw capacity. Their encoded sizes differ by a factor of 36.86 (875,083.80 divided by 23,743.10), because random masking removes the sparse representation’s compression advantage. The 40.03-fold and 1475.44-fold factors share the same 593.10-byte compressed baseline. Thus the latter combines capacity expansion and lost compressibility; it is not an additional multiplier to apply to forty-fold raw capacity. Encoded sizes include base64url and its prefix. They exclude signatures and transport envelopes and do not measure bytes per verification. Random decoy allocation or event timing does not itself make a snapshot incompressible: an unmasked pool can remain sparse. A 50% decoy share is not a 50% one-bit density. The comparison concerns full snapshots; Section 7.3 treats changes between versions and the additional requirements for delta delivery.

Adding an illustrative 1024-byte envelope to every list changes aggregate sizes to 1617.10, 64703.10 and 916043.80 bytes. The corresponding ratios are 1×, 40.01× and 566.47×. The envelope is a sensitivity assumption, not a W3C size or a PQC profile. All ratios use sample means. They exclude transport headers and caching and do not represent bytes per action.

At 50% decoys, one 16 KiB list holds 65536 genuine one-bit entries: 8 KiB useful raw status payload. It can hold 3276 complete disjoint twenty-entry bundles if bundles must remain within one list. A random encoded snapshot provides about 3.00 genuine entries per transmitted byte before the envelope. Its useful raw status payload occupies about 37.45% of its encoded byte count. Compression can make this representation ratio exceed 100% for sparse data, so it is not a channel-capacity measure.

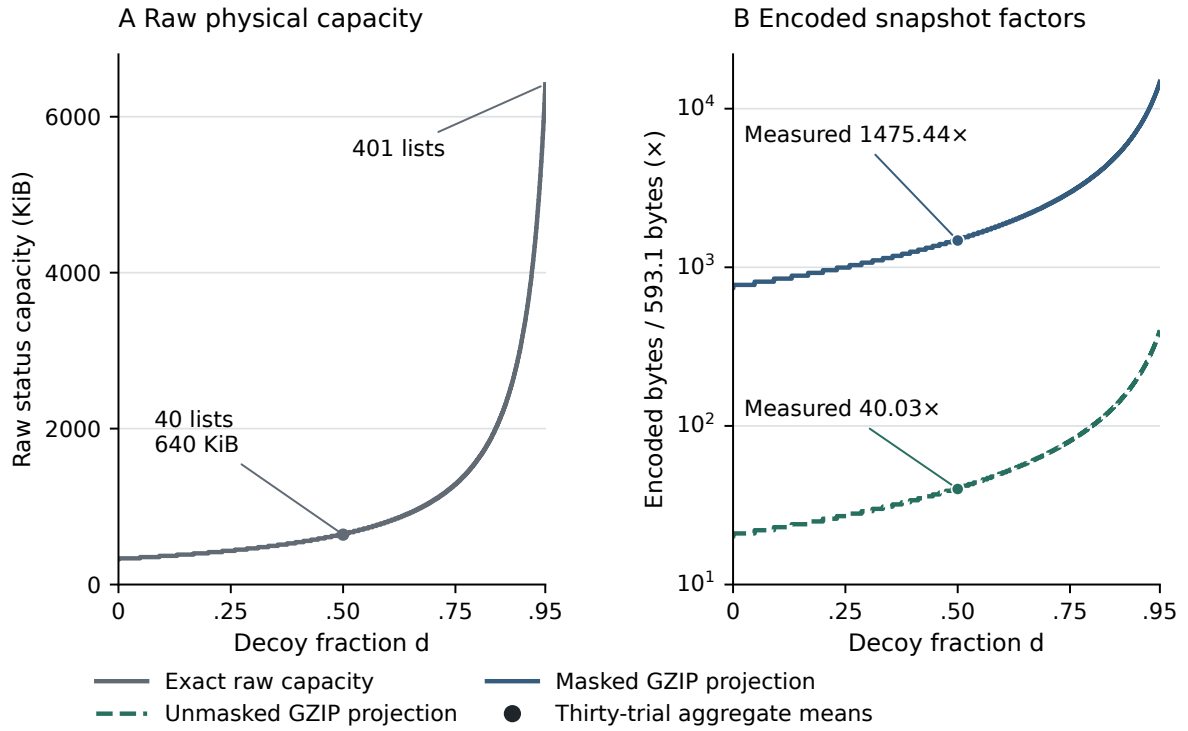
One random 8 MiB fixture, relevant to the draft terse-profile size, produced 8391186 GZIP bytes and 11188249 encoded bytes, about 10.67 MiB. This shows the cost of one random full snapshot. It does not mean every credential check downloads that snapshot. The ordinary sparse representation must be measured at the same population, status pattern and refresh policy.

The matched-history propositions are analytic. The codec fixture verifies byte counts only. A repeated-toggle decoy test is a negative control, not evidence against matched monotone decoys. Future privacy evaluation must estimate adversary advantage under side information and joint events, with uncertainty and explicit coverage limits.

Figure 3 projects the measured snapshot sizes across decoy shares. It uses exact finite-list rounding and separates representation costs from temporal privacy and incident outcomes.

Capacity cost of matched decoy fractions

$L = 131072$ bits; $N = 131072$ bundles; $k_{\text{pop}} = 20$; one-bit statuses



Panel B applies GZIP then unpadded base64url and a one-byte prefix to every list.

Projection curves use measured one-list means; they are not new privacy or network experiments.

Figure 3: Capacity and encoded snapshot payload across decoy share. $L = 131072$ bits and 131072 bundles with twenty unique statuses per bundle. Curves use exact finite-list counts and measured one-list codec means; markers at $d = 0.5$ use the aggregate fixture definitions in Appendix A. Encoded payload includes GZIP, unpadded base64url and one multibase prefix per list, but excludes signatures and transport envelopes. Ratios use the 593.1-byte one-list sparse encoded baseline. The 40.03-fold and 1475.44-fold results are codec workload comparisons, not privacy or safety equivalence.

8.2 Exact small model checks

Three exact examples check the interpretation of the conditional cascade framework. They are mathematical models, separate from the codec measurements and from the proposed industrial deployment.

First, consider a continuous-time susceptible–infectious–removed (SIR) model on six agents. Every ordered susceptible–infectious pair has infection rate 0.1 and each infectious agent has independent removal rate 1. Start with one infectious agent and permit no reinfection. Let $f(s, i)$ be expected additional infections. Set $f(s, 0) = f(0, i) = 0$. Conditioning on the next event gives:

$$f(s, i) = \frac{s}{s+10} [1 + f(s-1, i+1)] + \frac{10}{s+10} f(s, i-1). \quad (33)$$

Exact rational evaluation yields $1 + f(5, 1) = 1.6175452191$ expected ever-infected agents. A six-type branching matrix with off-diagonal entries 0.1 and zero diagonal has spectral radius 0.5 and a resolvent total of 2 for one root. The larger result is consistent with a dominating process that does not receive all the finite-population reductions of the SIR model.

Second, let a twenty-type deterministic chain have $K_{j+1, j} = 1$ for $j = 1$ through 19 and zero elsewhere. Then $K^{20} = 0$ and $\rho(K) = 0$, but one root in the first type reaches all twenty types. The finite sum $I + K + \dots + K^{19}$ records this reach. A spectral radius below one therefore does not by itself mean a small finite cascade.

Third, take ten agents with no secondary propagation and marginal root probability 0.1. Independent Bernoulli roots give probability 0.0016349374 of at least five roots. A common shock affecting all ten with probability 0.1 gives probability 0.1 for the same event. Both models have expected root count one, but their tail probabilities differ by a factor of 61.1644. The example shows why matching mean root exposure does not match tail risk.

9 Coupling capacity to withdrawal and cascading harm

9.1 Detection publication and protected enforcement

For source s and resource i , let D_s end at the authoritative withdrawal decision. It includes detection and assessment. Let Δ_{is}^X measure subsequent resource-gate enforcement delay and Δ_{is}^M the delay for covered outgoing influence. Publication, inclusion, selected finality, retrieval, validation and scheduling belong to the latter delays. The exposure window for ordinary resource effects is:

$$W_{is}^X = D_s + \Delta_{is}^X, \quad \Delta_{is}^X \leq \min\{T_{is}^{postwithdrawal}, F_{is} + \kappa_i\}. \quad (34)$$

Here F_{is} is the maximum authoritative status age enforced by an expiry cap, not merely the duration of a newly issued child lease. The term $T_{is}^{postwithdrawal}$ includes publication and all delivery and application stages for a supported path. Unsupported paths contribute infinity. The post-withdrawal path must end at the effect-enforcing receiver, rather than at an event broker acknowledgement. The clock margin κ_i includes any stated gate-clock and enforcement tolerance; a stricter conservative clock check may reduce it.

To justify the expiry path, let withdrawal occur at r and suppose every active status assertion used afterward has authoritative watermark $s_l \leq r$. The rule $e_g \leq s_l + F_{is}$ implies $e_g \leq r + F_{is}$.

Therefore continued ordinary admission ends no later than r plus the accepted-age cap and the supported clock tolerance. An assertion freshly claiming active status after r violates this premise. Renewing a short child lease from the same stale parent does not shorten that parent's allowed age.

A counterexample separates lease duration from withdrawal delay. Suppose a ten-second child lease is renewed every nine seconds from evidence allowed to remain sixty seconds old. Withdrawal just after the parent watermark can remain unseen across several child renewals. Ten seconds is therefore not a valid withdrawal bound. With the parent-age cap, sixty seconds plus tolerance is supported. If only an age-at-issuance bound A and an uncapped child duration are enforced, the general bound is A plus that duration and tolerance. This distinction is required before a lease duration is substituted into a cascade matrix.

In Section 5's notation, expiry is capped by both the lease duration and each required dependency's authoritative age. Detection before withdrawal remains outside that bound. A timeout at the submitting organisation also does not establish that a remote receiver stopped an already queued effect. Such effects remain in the operation state of Section 9 4.

9.2 Mean queue delay and guaranteed service

A descriptive single-server $M/M/1$ queue has Poisson arrivals at rate λ_Q and i.i.d. exponential service at rate μ_0 . If every service duration is scaled by F at fixed arrivals, the new rate is μ_0/F . For $\rho_0 = \lambda_Q/\mu_0$ and $F\rho_0 < 1$, the stationary mean-sojourn ratio is:

$$\frac{T(F)}{T(1)} = \frac{1/(\mu_0/F - \lambda_Q)}{1/(\mu_0 - \lambda_Q)} = \frac{F(1 - \rho_0)}{1 - F\rho_0}. \quad (35)$$

For baseline utilisation 0.1, work factors 5 and 9 give mean-delay factors 9 and 81. Factor 10 reaches the stability boundary. The calculation holds for the declared queue, not for arbitrary batching, bursty revocation traffic or public-ledger scheduling. Applying a byte multiplier to CPU service time requires measured evidence that service work scales accordingly. Multiplying both arrival rate and service cost by the same credential factor would require two independent workload changes.

A stationary mean does not supply the hard lifetime bound used below. That bound must come from a protected deadline or an enforced bounded-work service contract such as Section 5 5's reserved FIFO service. Under fail-closed freshness, congestion may stop legitimate work. Under stale permissive operation, it can extend harmful exposure. Both outcomes must be measured separately. A publication-budget inequality is likewise a load condition, not a deterministic consensus guarantee.

Engineering interpretation. A gate can be fast on average yet miss a withdrawal deadline during a burst. The bounded-service model instead asks how much work is already waiting, how much the withdrawal adds, and what processing capacity remains guaranteed. For the file-release service, an overload rule must restrict new work before that budget is exhausted. Stopping an otherwise legitimate release is an availability cost. Allowing a release on stale evidence is a different outcome and needs separate reporting.

9.3 Conditional propagation and expected harm

For descriptive network dynamics, let $X_i(t)$ indicate task-level taint, $U_{ij}(t)$ indicate acceptance of covered influence from j by i , $Z_s(t)$ indicate a defective source and $F_{is}(t)$ indicate continued reliance. A baseline taint intensity is:

$$\lambda_i(t) = (1 - X_i(t)) \left[\sum_j m_{ij}(t) p_{ij}(t) X_j(t) U_{ij}(t) + \sum_s a_{is}(t) Z_s(t) F_{is}(t) \right]. \quad (36)$$

The symbols m and p denote contact intensity and conditional adoption probability; a is exposure intensity. These are model parameters, not quantities determined by graph edges alone. Shared defective sources create correlated root exposures. The finite-state intensity model is a descriptive starting point. The expected-cascade theorem instead requires the explicit conditional offspring premises below, which can dominate a suitable model without assuming independent roots or offspring.

$$K_{ij}^{upper} = \beta_{ij} \left(\delta_j^{-1} + d_j^M \right). \quad (37)$$

Proposition 5 Expected cascade size. Consider a countable genealogy of taint episodes with types in a finite set. Give each non-root episode one causal parent using a declared rule based only on the history available at its birth. The parent, generation and type must be fixed at birth and cannot depend on subsequently observed offspring or harm. Let $\mathbf{Z}^{(n)}$ count episodes in generation n and $\mathbb{E}[\mathbf{Z}^{(0)}] = \mathbf{b} < \infty$. Suppose each type- j episode, conditional on its admissible birth history, produces in expectation at most K_{ij} type- i children. Birth histories must reveal the episode's type and generation, and the bound must hold uniformly for those histories. If K is nonnegative and $\rho(K) < 1$, then:

$$\mathbb{E} \left[\sum_{n \geq 0} \mathbf{Z}^{(n)} \right] \leq \sum_{n \geq 0} K^n \mathbf{b} = (I - K)^{-1} \mathbf{b}. \quad (38)$$

Proof. For each potential episode, multiply its conditional offspring bound by the birth-measurable indicator that it exists with the specified generation and type. Take expectations and sum over episodes. Tonelli's theorem applies because all counts are nonnegative. This yields the componentwise recurrence $\mathbb{E}[\mathbf{Z}^{(n+1)}] \leq K \mathbb{E}[\mathbf{Z}^{(n)}]$, without requiring independent episodes. Induction gives $\mathbb{E}[\mathbf{Z}^{(n)}] \leq K^n \mathbf{b}$. Summing and applying monotone convergence yields the series. For a finite-dimensional nonnegative matrix with spectral radius below one, the Neumann series converges to $(I - K)^{-1}$.

Proposition 6 Expected harm. Under the assumptions of Proposition 5, every counted harmful operation belongs to exactly one episode or the pre-existing pending category, so $H_T \leq H_{pre} + \sum_e H_e$ pathwise. Assign each episode its attributable nonnegative harm without double counting. Assume its conditional expected cost at birth is at most c_j for type j . Let h_{pre} bound expected harm from previously pending work outside this genealogy. Then for every horizon T :

$$\mathbb{E}[H_T] \leq \mathbf{c}^\top (I - K)^{-1} \mathbf{b} + h_{pre}. \quad (39)$$

Proof. Apply the same conditional-expectation and nonnegative-summation argument to episode costs, then use Proposition 5. Later queued effects attributable to an episode belong to its cost bound. The separate pending term includes only effects not already charged. This is an expected-harm upper bound, not a pathwise maximum or a probability bound for every named harm. Markov's inequality can give a tail bound for a specified nonnegative harm count, but may be loose. Indefinite external seeding requires a separate immigration model.

Reading the bound. The matrix describes how many further affected episodes one episode can produce, under stated conditional limits. A spectral radius below one makes the mathematical

upper-bound series converge. It does not ensure that an incident is small or that no file is released improperly. An expected-harm bound averages across the model's possible incidents. Engineers should first establish which release and communication paths are controlled, then justify their timing limits. A low value computed from unsupported average rates is not a safety certificate.

Let L_j be total covered-influence lifetime, bounded by detection-and-assessment duration plus d_j^M , with $\delta_j > 0$ and $\beta_{ij} \geq 0$. To justify the particular matrix $K_{ij} = \beta_{ij} (\delta_j^{-1} + d_j^M)$, require that type- j episodes have conditional mean time to authoritative withdrawal at most δ_j^{-1} at every admissible birth history. The covered outgoing-influence lifetime after withdrawal is bounded by d_j^M . Under a common history filtration, the enabled-influence indicator is predictable and the potential type- i offspring counting process has predictable conditional intensity at most β_{ij} while that influence remains enabled. Compensator integration, or an equivalent conditional rate construction, gives expected offspring at most $\beta_{ij} \mathbb{E}[L_j \mid \mathcal{F}_{birth}]$, which gives the proposed entry. Observed average contact rates alone do not meet this premise.

The communication deadline must bound the last accepted covered influence, including queued and in-flight messages. Sender shutdown alone is insufficient. Receiver-side status, version, expiry or cancellation checks must enforce that deadline. Resource-side release containment and message containment have different lifetimes. A file gate does not justify a short outgoing-influence bound if the agent can still transmit harmful instructions elsewhere. The model therefore requires the exact covered propagation channel, its enforcing control and any bypass assumptions. A finite network can contain repeated episodes; saturation and repeated contacts may make the branching upper bound loose. These conditions also prevent the spectral radius alone from being presented as a practical safety guarantee.

The spectral criterion is sufficient for the stated upper bound. If $\rho(K) \geq 1$, that certificate is unavailable; the real process may still be finite, subcritical under a tighter matrix, or unable to reach the supercritical types. At criticality, even an exact scalar branching model has constant generation means and linearly growing cumulative expectation. In a finite network with no reinfection, the number ever affected is bounded by its population.

For fixed rates, root vector and harm costs, let $R_K = (I - K)^{-1}$ and $\Phi = \mathbf{c}^\top R_K \mathbf{b}$. With $K_{ij} = \beta_{ij} (\delta_j^{-1} + d_j^M)$, differentiating $(I - K)R_K = I$ gives $dR_K = R_K(dK)R_K$. Hence:

$$\frac{\partial \Phi}{\partial d_j^M} = (\mathbf{c}^\top R_K \beta_{:j}) (R_K \mathbf{b})_j \geq 0. \quad (40)$$

This local sensitivity holds where the spectral radius remains below one. It shows how a supported change in a covered-influence deadline affects the conditional expected-harm bound. It does not imply that more credentials increase harm: credential granularity can also reduce influence rates, roots and per-episode cost. Those changes require their own terms and matched semantic workloads. Figure 4 in Section 9.5 illustrates this sensitivity with a declared scalar model. It also distinguishes declining per-generation incidence from cumulative harm, which cannot decline with the observation horizon.

9.4 Persistent pending effects and distinct outcomes

Let $P(t)$ contain logical operations that remain unresolved under their agreed contract. An operation is removed only after authoritative evidence establishes execution, cancellation or another specified terminal state. If nonnegative expected-harm bounds $q_o(t)$ are valid conditional on the current information $\mathcal{F}_t$, linearity and a non-overlapping attribution give:

$$\mathbb{E}[H_{pending} \mid \mathcal{F}_t] \leq \sum_{o \in P(t)} q_o(t). \quad (41)$$

No independence assumption is needed. If q is a probability that an operation produces at least one harmful release, the expression bounds an expected count of harmful operations. It also bounds a release count only when each operation permits at most one counted release. Multiple releases require a conditional expected-count bound; severity requires appropriately weighted cost bounds. Previously pending costs are included once, either in the episode cost or the separate pre-existing term. A sender's timeout does not remove an operation. An effect accepted at time zero can still execute at time ten after new requests are blocked at time one unless the receiver enforces the corresponding cancellation or expiry rule.

The reported outcomes are agents ever tainted, executed prohibited releases, legitimate jobs completed by deadline and unresolved-operation duration. Each named harm retains its own interpretation. A weighted aggregate that hides confidentiality loss behind a higher legitimate throughput would not evaluate the stated primary harm.

9.5 Illustrative containment and harm sensitivity

Figure 4 illustrates the conditional expected-harm bound as the supported outgoing-influence delay changes. Outside the stated premises and subcritical region, this certificate is unavailable. Reserved service affects the bound only through a supported deadline for the covered receiver-enforced channel.

For the scalar illustration, take one root, unit conditional harm per episode, no previously pending harm, conditional detection mean bound $\bar{D} = 10$ seconds and offspring intensity bound $\beta = 0.04$ per second. Let d now denote a supported outgoing-influence delay in seconds, distinct from the decoy fraction in Figure 3. The scalar matrix has spectral radius $R_{cas}(d)$:

$$R_{cas}(d) = \beta (\bar{D} + d), \quad \mathbb{E}[H_T] \leq \mathcal{B}(d) = \frac{1}{1 - R_{cas}(d)}, \quad d < 15 \text{ s}. \quad (42)$$

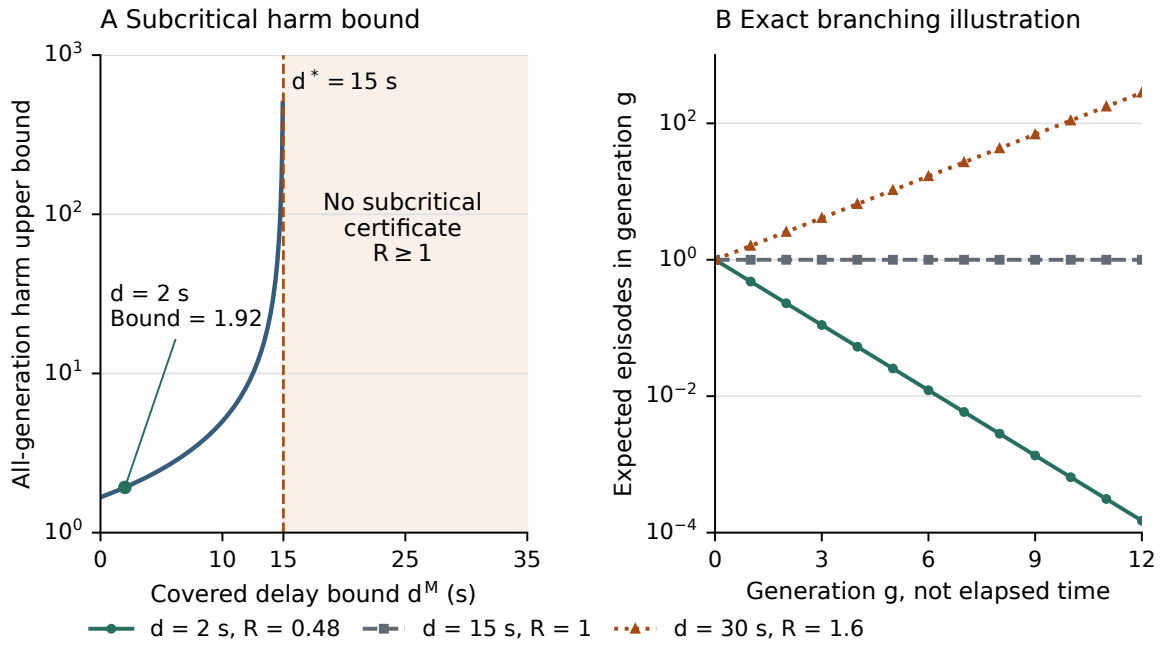
This certificate applies for every T under Propositions 5 and 6. At delays 2, 10 and 14 seconds, it gives 1.9231, 5 and 25 expected harm units. The pole marks failure of this certificate, not observed infinite loss. The second panel adds an exact independent branching construction: each episode produces a Poisson offspring count with mean R_{cas} , realised by rate- β Poisson births over deterministic lifetime $\bar{D} + d$. Generation g has mean R_{cas}^g ; the cumulative mean through G generations is:

$$\sum_{g=0}^G R_{cas}^g = \begin{cases} (1 - R_{cas}^{G+1}) / (1 - R_{cas}), & R_{cas} \neq 1, \\ G + 1, & R_{cas} = 1. \end{cases} \quad (43)$$

The branching construction specifies birth times, but the plotted generation means are not elapsed-time predictions; those require a separate time-indexed calculation. For the service annotation, a hypothetical admitted job has $Q_{\max} + w_{\max} = 100$ abstract work units, and other supported delay plus scheduling latency is one second. Hence $d \leq 1 + 100/R_{srv}$ in seconds. A reserved rate greater than $100/14 \approx 7.143$ work units per second makes this particular scalar bound subcritical. A rate of 100 supports the two-second case only while the backlog, receiver and service premises hold. These constants are illustrative, with no conversion from codec bytes to CPU work.

Containment delay and conditional cascade bounds

Illustrative $\beta = 0.04 \text{ s}^{-1}$, $\bar{D} = 10 \text{ s}$; one root, unit episode cost



$R(d^M) = \beta(\bar{D} + d^M)$; panel A uses $1/(1 - R)$ only when $R < 1$.

Reserved-service example: $d^M \leq 1 \text{ s} + 100 \text{ work units}/R_{\text{srv}}$.

$R_{\text{srv}} > 100/14 \text{ work units/s}$ is sufficient for $R < 1$ under this workload contract.

Generation counts can decay; cumulative harm cannot. No population or deployment is fitted.

An upper matrix with spectral radius at least one does not prove growth in a finite network.

Figure 4: Conditional scalar harm bound and illustrative branching generations. The shaded region has no subcritical certificate. Reserved service reduces covered delay only under bounded-backlog, service and receiver-enforcement assumptions.

For the engineering workflow, H_T counts prohibited releases through time T . A severity-weighted outcome requires declared nonnegative weights and corresponding conditional cost bounds; it is reported separately from legitimate throughput. If a certified bound is U_T , Markov’s inequality gives $\Pr(H_T \geq z) \leq \min\{1, U_T/z\}$ for $z > 0$. This bound may be too loose for an operational risk target. Common shocks can alter tail risk without changing the mean, as Section 8.2 demonstrates.

A fitted matrix is not a certificate. Deployment use requires justified conditional parameter bounds, including detection, every covered influence channel and attribution of pending effects. An entrywise dominating nonnegative matrix K^* with $\rho(K^*) < 1$ supplies a conservative route only if those bounds hold for the declared uncertainty set. Continuing external roots require a finite-horizon input model. The companion study extends the conditional analysis to continuing roots and parameter envelopes. Empirical calibration and rare-event validation remain open; model experiments do not weaken the premises of Propositions 5 and 6.

10 Validation interface and companion study

The companion paper specifies a protected protocol, exhaustively explores finite transition systems, compares stipulated hosting and workload models, and tests causal decoy constructions [22]. These completed computations examine the contracts within stated model bounds. They do not establish wire-format conformance or production timing guarantees. The following design specifies the next deployment study and the evidence it should produce.

10.1 Minimum deployment study

Use the same three-domain engineering-file workflow. Deterministic resource gates and receiver logs establish prohibited release and terminal closure before large language model (LLM) agents are introduced. Specify evaluation, invalidation, admission, dispatch, receiver effect, expiry, restart and closure as one state machine. A refinement map relates implementation traces to common-record eligibility, freshness and persistent operation accounting. Bounded checks expose counterexamples within their explored scope; successful tests alone do not prove the unrestricted implementation.

Compare independently operated mirrors carrying issuer-signed snapshots with a synchronized local ledger node. Both retain verified local state. Freeze issuer decisions, the complete authorisation predicate, accepted source age, effect semantics and incident schedule. Measure backend encoding and signature differences explicitly. Vary credential multiplicity in a separate comparison whose policy equivalence is checked. A scope-changing credential decomposition is a different policy experiment.

10.2 Outcomes and evidence requirements

The primary outcomes are prohibited releases per incident and the last accepted covered effect relative to authoritative withdrawal. Report legitimate completion by deadline and unresolved-operation duration separately. Paired trials use the whole incident as the replication unit. An excluded pilot informs a prespecified confirmatory sample size and uncertainty target. A deployment protocol must fix failure conditions and report incomplete or censored runs.

Observed delay distributions describe tested conditions. A hard service bound additionally requires enforced admission, bounded work and interference, reserved capacity, clock margins and receiver-side deadlines. Tests can refute those premises; empirical quantiles cannot replace them.

The measured connection from representation to work, supported delay and the conditional cascade matrix is the principal system result sought.

A deployment privacy study must declare the complete observer transcript and test the causal decoy lifecycle under its actual publication process. It should compare masked and unmasked variants under the same incident and allocation process. Its attack results remain distinct from the all-history conditions in Section 6. The companion provides a fixed-cohort construction and exact finite inference experiments, not continuous-issuance deployment evidence. Appendix C maps those completed model studies and the remaining requirements.

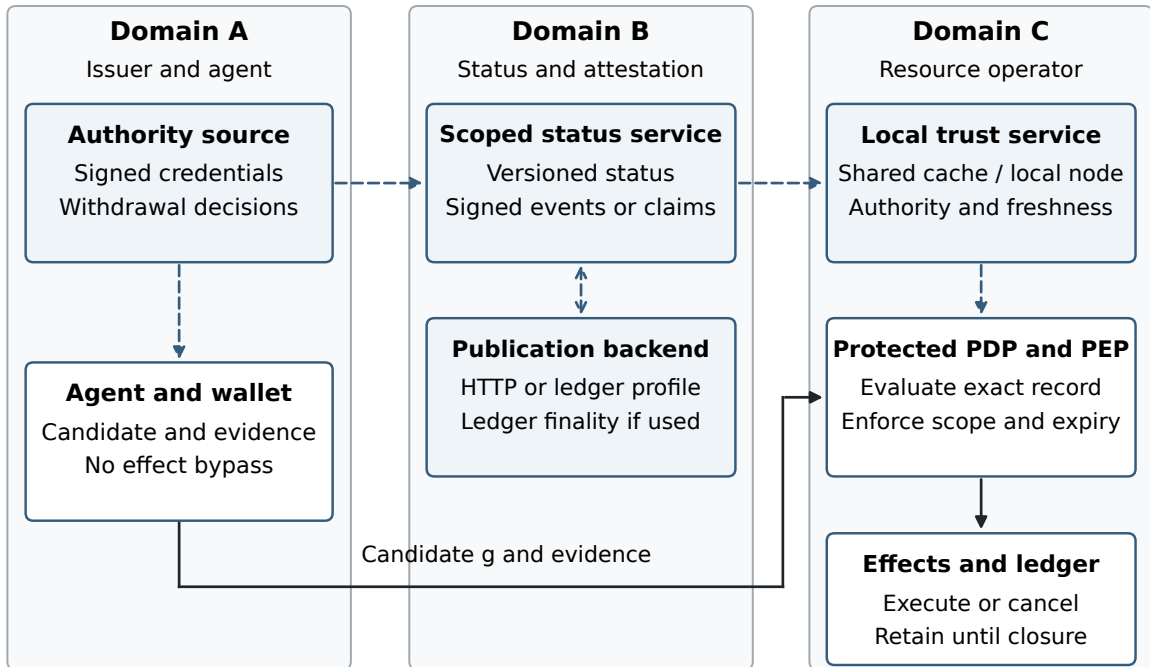
11 Revocation List Propagation in Multi-Agent Systems

11.1 Authority across three trust domains

The cross-domain protocol separates authority, evidence distribution and the protected effect. Domain A determines the scoped mandate or status. Domain B forwards authenticated evidence or supplies an explicitly recognised attestation. Domain C owns the resource and its enforcement boundary. Figure 5 shows these responsibilities and the withdrawal sequence; Section 5 defines the components.

Domain C retains its policy decision point and enforcement gate [18]. Domain B cannot refresh A’s completeness watermark by issuing a newer wrapper. If C delegates evidence evaluation to B, the profile must specify the applicability, dependency-freshness and failure-response obligations transferred to B. Authority to distribute data alone supplies none of these permissions.

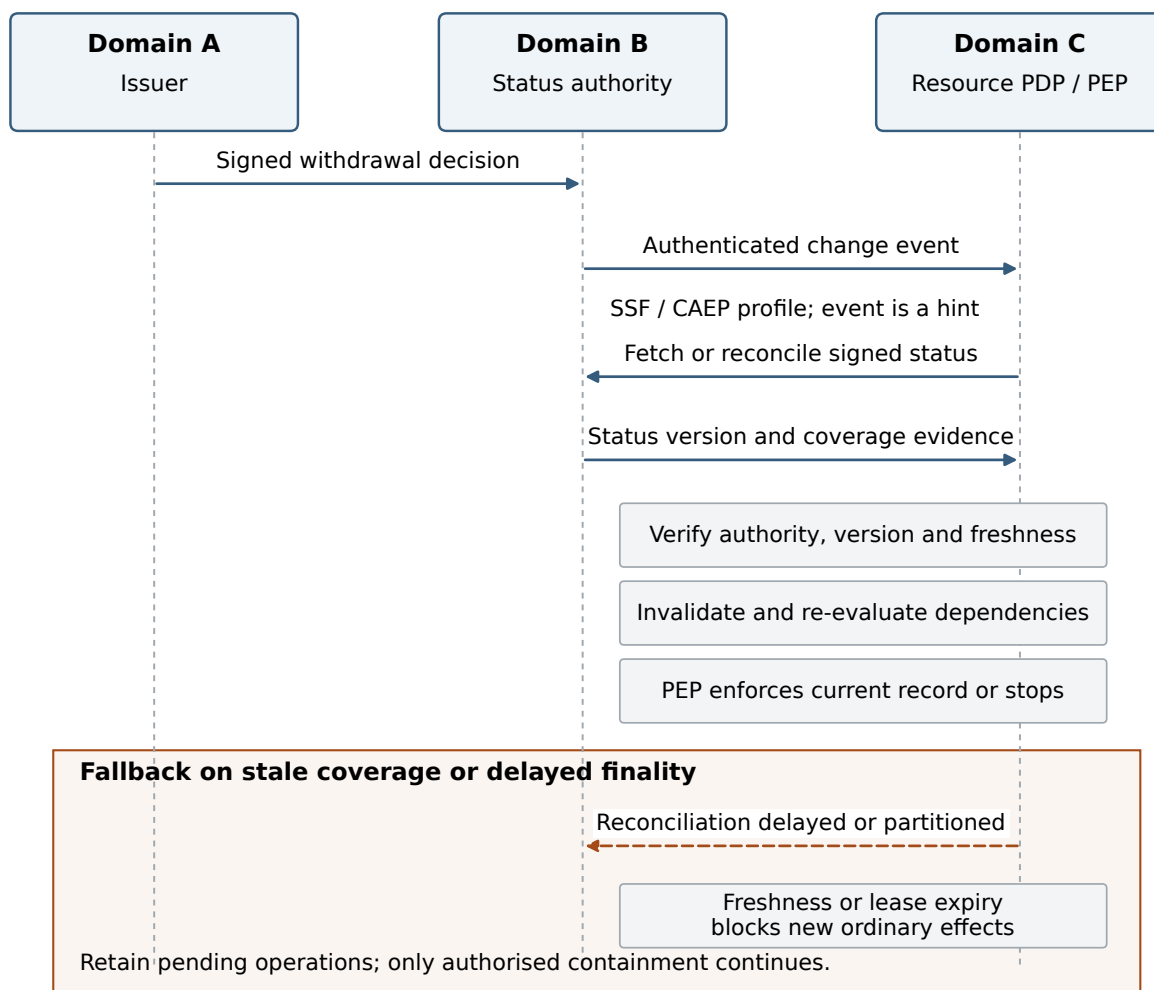
A Three domains and one resource enforcement boundary



Dashed: evidence and status. Solid: candidate, execution and closure path.

Figure 5a: Authority, evidence and enforcement across three trust domains. Domain C retains the protected resource gate and persistent operation accounting.

B Push, reconciliation and protected freshness expiry



Receipt of an event, RPC reply or ledger finality is not a PEP enforcement acknowledgement.
 Issuer recognition and status authority remain scoped by Domain C; no global allow is implied.

Figure 5b: Withdrawal and recovery. Push accelerates processing; reconciliation repairs gaps. Delayed consensus cannot extend accepted age or lease validity. Receipt and resource enforcement remain separate events.

11.2 Push pull and lease expiry

The proposed delivery profile combines authenticated change signals, retrieval of authoritative state, periodic reconciliation and protected expiry. A receiver validates the event issuer, audience, subject mapping, event type and local authority before applying a status consequence. It records receipt separately from successful validation, state installation, affected-record reevaluation and enforcement. CAEP supplies relevant change events; the mapping from those events to a particular credential allocation and action policy remains part of this profile [19, 20].

The profile treats event transport as an acceleration path. It tolerates duplicates, reordering and gaps through scoped identifiers, idempotent application and authenticated version rules. A transport acknowledgement does not prove resource enforcement [52]. SSF permits event loss during paused streams and does not retain events while disabled. Reconciliation therefore compares local state with the selected authoritative version and establishes the permitted watermark; an apparently healthy stream does not prove that every relevant event was received. Security Event Tokens (SETs) are not execution leases, and their processing must respect SSF's prohibition of the JSON Web Token (JWT) expiration claim (exp) [19].

Let $\mathcal{D}_x$ be the domains whose ordinary effects depend indispensably on withdrawn premise x , after sound policy reevaluation. For each domain d , let $\mathcal{P}_{d,x}$ contain supported delivery or expiry paths. If every path p has a justified bound b_p from the authoritative withdrawal decision to the required local enforcement, define

$$b_{d,x} = \min_{p \in \mathcal{P}_{d,x}} b_p, \quad b_x^{all} = \max_{d \in \mathcal{D}_x} b_{d,x}. \quad (44)$$

An unsupported path contributes infinity, and the minimum over an empty supported-path set is infinity. When no domain requires a new ordinary-effect restriction, the corresponding maximum is defined as zero; unresolved operations remain separately accounted. Serial stages within one path contribute their supported cumulative delay; parallel paths are alternatives. A fallback launched only after a timeout includes that timeout. The maximum describes coverage of all relevant domains, conditional on the declared domain set and enforcement semantics. It does not include detection before authoritative withdrawal or erase already accepted remote effects. To use it as an outgoing-influence bound in Section 9, the enforcing paths must cover the last accepted influence, including queued messages.

Every derived approval retains the applicable source-age limits. In the notation of Section 5, its expiry satisfies $e_g \leq \min_l (s_l + \tau_l)$ as well as its local duration and validity limits. Renewing a short lease from the same stale parent cannot restart this clock. During a partition, the remaining allowance is $H_{fresh} = \max\{0, \Delta t_{\max} - (t_0 - \omega_v) - \kappa_v\}$. A readable local replica supports operation only while this allowance and all other prerequisites remain valid.

11.3 Ordering withdrawal and execution

Each resource domain defines a protected local ordering between installation of relevant withdrawal state and authorisation of an effect. The final commit checks the same complete record, current dependency and policy generations, operation slot and episode account. A versioned compare-and-commit mechanism, or an equivalent serialised state machine, rejects an approval if a relevant premise changed after its evaluation. Cache invalidation alone is insufficient: the resource adapter must consult the protected generation before the effect escapes its control.

Let $R_{x,d}$ denote the local transition that withdraws eligibility for the affected record, and let E_o be the operation's effect-authorisation transition. If $R_{x,d} \prec_d E_o$, the old approval cannot

authorise that effect. A newly validated alternative witness may support a distinct current decision. If $E_o \prec_d R_{x,d}$, the operation is accounted for as already admitted; its cancellation and remaining effects follow the operation contract. Neither ordering makes a sender timeout evidence of cancellation.

Engineering example. The sender authorises a file-release operation, but the receiving domain queues it. The sender later times out and learns that the mandate was withdrawn. The queued operation can still execute unless the receiver enforces the agreed check, expiry or cancellation rule. A retry retains the logical operation identity, and the receiver checks that identity to prevent a duplicate release. The sender keeps the operation unresolved until authenticated evidence establishes its terminal outcome. A transport acknowledgement only confirms an earlier processing step.

This local ordering requires an identified linearisation boundary. For a release controlled inside the trusted adapter, it can coincide with the irreversible release decision. When the effect occurs at an ordinary external API, the local database transaction does not make remote acceptance atomic. The implementation must retain an operation identifier, bound execution envelope and pending outcome; the receiver must enforce the agreed expiry or cancellation semantics. Any gap between the final check and the last preventable effect belongs in the delay and assurance model. A purported global transaction spanning A, B and C requires an additional protocol and proof.

11.4 Failures and shared infrastructure

The failure response follows the freshness and service contracts in Section 5. A required expired premise stops ordinary effects under affected approvals. Separately authorised containment, accounting and recovery remain available. Deployment testing must exercise flash crowds, malformed inputs and correlated failures against the bounded admitted-work envelope.

A Byzantine publisher can issue mutually inconsistent snapshots or falsely assert a recent active state. Signatures establish provenance, and ledger ordering can expose conflicting versions, but neither establishes the truth of a completeness claim. The freshness proof requires an honest authoritative-state contract or an independently justified mechanism that supplies its premise. Equivocation evidence triggers the relying policy's suspension and investigation. No finite withdrawal guarantee is claimed against an unconstrained authority that conceals the withdrawal while continuously signing false fresh assertions.

Shared caches may deduplicate authenticated content. Tenant-specific acceptance still binds audience, authority, policy, profile, epoch and dependencies. Quotas and reserved service must cover cross-tenant interference. Cache misses, subscriber identities and invalidation logs enter the privacy transcript. Packed ledger mutation controls are profile obligations in Appendix C and the companion model. Wire-conforming mutation tests remain deployment work.

11.5 The operational trade-off

The comparison reports incremental lookup leakage $\Delta\mathcal{P}$, transmitted bytes B_{net} , accepted source age and completed receiver enforcement. Bulk refresh, event-triggered retrieval and local replication are evaluated under equal authority, freshness and effect semantics. The payload results are in Section 8 and the conditional delay-to-harm illustration is in Section 9.5. The companion study models the intervening work and service path with stated constants; deployment measurement remains necessary.

12 Implementation obligations and research priorities

12.1 Standards and experimental profiles

A deployment profile must freeze the signed representation, authority, purpose, bit order, epoch and freshness evidence. The pinned W3C baseline, draft terse representation, proposed packed-ledger method and masked-decoy experiment remain distinct profiles [12, 13, 37]. The standard baseline does not inherit the custom decoder or nonzero unissued cover entries. Appendix C states the contract; the companion paper specifies profile obligations and checks abstract mutation controls; complete signed-byte schemas and conformance vectors remain deployment work. Appendix E distinguishes proposed W3C clarifications from experimental profile changes.

ML-DSA supplies object signatures, with suite-specific sizes and verification work [21]. It does not select the credential envelope, protect transport confidentiality or establish post-quantum ledger authorisation. A deployment benchmark must fix one complete signature profile before comparisons. Broader algorithm, hybrid and transport migrations remain optional sensitivity studies.

12.2 Control selection for software engineers

Start with a protected effect, rather than a new status backend. P1 supplies the gate and finite approval. P6 preserves operation identity and unresolved outcomes across retries. P2 supplies verified local state, change signals and reconciliation. P4 is required wherever a bounded service delay is claimed, and P7 fixes accepted algorithms, keys and migration rules. These controls implement the reference architecture in design C. P3 can later reduce reevaluation work through a complete dependency index. Full reevaluation is a valid initial baseline and helps test the index. P5 remains a separate privacy experiment.

The engineering target is observable: once a relevant withdrawal is installed locally, the old approval cannot produce a later covered receiver effect, and expired evidence cannot be refreshed merely by renewing its child lease. A completed PDP evaluation is not that effect. The final resource adapter must enforce the decision at its last preventable boundary. Appendix B states the test obligations; the companion paper provides a formal protocol and executable model studies, with deployment conformance still to be established.

12.3 Telemetry and alert thresholds

Thresholds derive from the deployment's accepted-age and service contracts. Let $\omega_{v,l}$ be the accepted watermark for object l at verifier v . With clock margin κ_v , define conservative age and remaining headroom as

$$A_{v,l}(t) = t - \omega_{v,l} + \kappa_v, \quad h_{v,l}(t) = \tau_l - A_{v,l}(t). \quad (45)$$

At partition onset, the age-only residual allowance for a record is $H_v = \max\{0, \min_{l \in \mathcal{L}_g} h_{v,l}(t_0)\}$. Take the minimum again with its remaining lease, recognition, mandate and continuation validity. The clock margin includes source-clock error, verifier-clock error and drift over the claimed interval. An empty mandatory dependency set has no status-age cap, but still has the other eligibility and expiry requirements.

Ordinary admission requires $h_{v,l}(t) > 0$ for every required dependency and compliance with the other expiry limits. A warning begins before headroom falls below the supported refresh-and-application budget. When no finite refresh bound exists, expiry still supplies the operational

stop; a recent average latency does not justify extending it. Loss of the assumed clock-error bound invalidates the corresponding freshness calculation.

For containment work, let q be work already ahead of a job, w_o its bounded work, R_{srv} guaranteed service and θ scheduling latency. For a job with remaining local deadline b_o , this service envelope certifies completion by the deadline when

$$\theta + (q + w_o) / R_{srv} \leq b_o. \quad (46)$$

If this certificate fails, the declared admission restriction or another supported path must apply before the deadline. Failure of this sufficient bound does not itself prove that the actual service will miss the deadline. The quantities must use common work units and account for permitted interference. Tail-latency measurements evaluate the contract but do not establish its deterministic premise by themselves.

Cache-hit ratio is an efficiency metric; a high hit rate can conceal stale data. Report it alongside oldest accepted watermark, rejected rollback attempts, unknown-status decisions, stream gaps and receipt-to-enforcement delay. Timestamp authoritative decision, publication, receipt, validation, installation and gate action separately, with clock uncertainty. Incident-level deadline misses and legitimate work loss are separate outcomes.

For pending work, report $|P(t)|$, oldest unresolved age, operations past their receiver deadline and the conditional exposure aggregate $\sum_{o \in P(t)} q_o(t)$ when valid bounds exist. A configured queue limit follows the declared state and service model. Exceeding that limit restricts new work; it never permits deletion of unresolved records. Alerting records observable contract violations, while validation still examines violations not exposed by routine telemetry.

12.4 Where to start with one protected file release

Use one issuer, one receiving organisation and one irreversible release of a specified engineering file to a specified recipient. This narrow workflow makes the harm observable: bytes released outside the permitted scope. It also gives engineers a concrete boundary at which to block an action. Record cancellation separately from an attempted cancellation, and record lost legitimate service separately from prohibited release. The pilot establishes implementation evidence within its tested scope; it does not validate every premise of the cascade model.

Stage 1: define authority and the effect. Pin the unmasked W3C Bitstring Status List v1.0 Recommendation and its required decoding. Bind the credential to its status purpose, list and index. Specify which issuer and keys may assert each claim, which status authority may withdraw it, and how key or recognition changes affect reliance. Define an authoritative completeness watermark and its trust contract; retrieval time does not supply one. Bind the complete record to the file version, recipient, requested action, continuation, evidence and policy versions. Initially reevaluate every required dependency, including applicable prohibitions. Do not start with decoys, masking or a custom ledger decoder.

Stage 2: build the local enforcement path. Retain authenticated snapshots in a domain-local shared cache, with anti-rollback checks and explicit accepted-age limits. Keep tenant-specific decisions outside shared content reuse. Use authenticated push to accelerate retrieval, and periodic reconciliation to repair event loss. Serialize accepted changes with the receiver's final version and expiry check. Bound the check-to-effect gap. If the file adapter cannot control all released bytes atomically, state precisely when release becomes irreversible and how partial release is counted. This boundary must not be hidden inside a successful application programming interface response.

Stage 3: make expiry and recovery durable. Assign an immutable logical operation identifier (ID) before dispatch. Persist its bound contents and pending outcome; use the same ID for retries. The receiver requires a persistent duplicate-suppression and terminal-outcome contract. A timeout leaves the outcome unresolved. Enforce the earliest applicable lease, source-age and other validity limit. On expiry, stop ordinary releases. Preserve separately authorised containment and recovery. Restart restores checkpoints, unresolved operations and clock assurance before ordinary admission resumes.

Stage 4: establish the service envelope. Measure decoding, proof verification, reevaluation and final gate work separately. Reserve capacity for the withdrawal path, bound admitted work and account for blocking by other tenants. Apply Section 12.3's deadline inequality to actual work units. If detailed processing cannot meet the budget, use a separately supported broad hold or protected expiry path. A priority setting and a low average queue delay do not establish a hard bound.

Stage 5: test failures before optimising. Interleave withdrawal installation with final commitment; drop and reorder signals; isolate every refresh path; crash before and after dispatch; lose terminal receipts; retry the same operation; and withdraw issuer keys. Confirm that an earlier local invalidation blocks its old approval, stale-parent renewal cannot extend accepted age, and missing receipts never delete pending work. Record authoritative withdrawal, completed receiver enforcement and clock uncertainty separately. Add P3 only after incremental decisions agree with full reevaluation on the declared policy. Then compare signed mirrors and local ledger state under the same semantics. Test decoys and masking in an isolated branch with their full observable histories.

The following limits determine which further controls are useful. These responses reduce exposure within a stated scope; they do not remove an unsupported premise.

Table 11: Engineering limits, responses and remaining uncertainty

Boundary	Engineering response	Remaining limit
A signed issuer assertion can be false or incomplete.	Scope issuer authority; require independent evidence where justified; suspend reliance on detected conflicts; restrict high-impact actions when assurance is insufficient.	Signatures, quorums of correlated sources and ledger consensus do not establish claim truth.
A required dependency is missing.	Name authoritative discovery sources; begin with full reevaluation; compare any index against that baseline; restrict actions when required evidence is unknown.	Completeness remains relative to the source and policy contract.
Refresh paths fail while availability remains important.	Use independent distribution paths and finite cached validity; define separately authorised protective operation in advance.	No replica provides unseen updates during a total partition; extending ordinary validity changes the risk contract.
An irreversible effect has already occurred.	Reduce release scope, stage disclosure, add approval for sensitive actions and enforce expiry at the receiver.	Later revocation cannot retract disclosed information or guarantee reversal.
A burst exceeds the selective enforcement budget.	Bound admission and interference; reserve service; support a broad hold and controlled recovery.	Conservative stopping can lose legitimate work; report that cost.

Boundary	Engineering response	Remaining limit
Public status histories expose activity.	Start with scheduled shared retrieval; minimise query logs; evaluate matched decoys and masking separately against a declared observer.	Local reads do not hide public updates, and a fixed mask does not hide differences between snapshots.

12.5 Research priorities and the companion paper

The companion paper develops protocol states and finite checks in Sections 3–4, specifies profile obligations in Section 5, and reports synthetic hosting and workload results in Section 6 [22]. Its Sections 7–8 construct fixed-cohort causal decoys and test exact membership inference. Section 9 extends the conditional harm model to continuing roots and uncertainty envelopes. These results make several architectural obligations executable and expose failure traces within their stated bounds.

The next system priorities are implementation refinement, empirical service calibration, independent reproduction and validation of the source-completeness contract. The privacy priorities are continuing issuance, reserve renewal and incident-correlated withdrawals under declared side information. Masking must show a distinct incremental benefit against its unmasked control. These questions remain visible in Section 6 because the privacy propositions depend on their observation assumptions.

Rare severe harm and parameter uncertainty require deployment evidence beyond conditional mean bounds. Truthful authority and evidence completeness remain premises; omission and false-assertion tests demonstrate their limits without making them unconditional guarantees. Appendix C maps the seven questions to the companion’s actual sections. Its Appendix A supports reproduction, Appendix B supplies an implementation checklist, and Appendix C identifies evidence still needed beyond the model. Its Section 12 states the research outlook.

13 Scope and conditions for article claims

This article establishes conditional statements about common-record eligibility, membership inference, accepted-age operation and expected cascade costs, together with exact representation calculations and codec measurements. The companion develops a protected protocol and causal decoy construction and evaluates executable models. Neither article establishes production conformance, general AI safety or a deployment-wide timing certificate.

Application requires protected gates, sound authority interpretation, jointly applicable evidence, persistent accounting, supported service contracts and clocks within the declared margin. Evidence discovery is complete only relative to the named source and policy contract. A publisher that knowingly signs false fresh active state can defeat the watermark premise. An undiscovered adverse fact or an uncontrolled effect channel can defeat other premises. Neither ledger consensus nor a valid signature removes these limits.

Correlated outages may disable several delivery paths. Masked and decoy representations remain experimental. Continuous external compromise and rare severe outcomes require their own finite-horizon and uncertainty analysis. The finite stochastic model connects the work to network science; it makes no thermodynamic-limit phase-transition or quantum-neuroscience claim. The companion study tests specified constructions and model failure envelopes; deployment validation

remains necessary.

13.1 Research transparency

The byte fixtures use generated non-personal data. Their deterministic construction is specified in Appendix A. The author is the founder of Spherity GmbH, which develops identity and trust infrastructure. Architecture assessments should therefore be tested against independently implemented baselines. This article reports no production benchmark or completed large-agent experiment.

A Reproducibility of the numerical fixtures

The compression calculations use Python 3.12.14, zlib 1.3.2, $L = 131072$ bits, 30 trials, GZIP level 6 and $\text{mtime} = 0$. Each raw sparse fixture is a zeroed 16384-byte array. Python `random.Random(seed).sample(range(L), round(pL))` selects positions without replacement. Set each selected position i using byte $i // 8$ and mask $1 \ll (i \% 8)$. The two sparse counts are 131 and 1311. The byte orientation is a codec-fixture convention, not an alternative W3C decoding rule.

For each of the three one-list cases, initialise a separate generator with seed 20261008 and advance it sequentially through 30 trials. The random case calls `randbytes(16384)`. GZIP-compress the resulting bytes, apply URL-safe base64, remove trailing padding, and count one extra byte for the multibase u prefix. Record raw bytes, GZIP bytes and encoded bytes for every trial. The table reports arithmetic means. Fixed-count sparse inputs are not i.i.d. Bernoulli samples; the entropy calculation uses a separate idealised distribution.

For the forty-list masked comparison, initialise one generator with seed 20261009. Generate forty consecutive random lists in each of thirty trials and sum their sizes before averaging. For the unmasked control, initialise a separate generator with seed 20261008 and generate forty sparse $p = 0.001$ lists per trial with the same sampling rule. The first thirty buffers of the unmasked aggregate stream also occur in the single-list sparse baseline. Cases therefore are not independent statistical samples. These are deterministic snapshot controls. They do not generate or verify matched temporal histories. Compare ratios of aggregate means with the mean one-list sparse baseline; do not average per-trial ratios.

Add the illustrative 1024-byte envelope once per list after encoding. For the draft terse-profile illustration, initialise a fresh generator with seed 20261008, generate $8 \times 1024 \times 1024$ random bytes once and use the same codec. Production masks need a cryptographic random source; this deterministic generator is used solely to reproduce the measurements. Compression versions and seed streams are fixed because byte counts can vary across implementations.

Capacity checks compute $m = \text{floor}(\text{gamma} \times 131072)$ genuine entries per list and $\text{ceil}((20 \times 131072)/m)$ lists for each posterior cap. The entropy check evaluates the binary entropy function at $p = 0.001$. Ledger checks evaluate $512[1 - (1 - p)^{256}]$ at the same p . Queue examples substitute $F = 5$ and 9 at baseline utilisation 0.1. None of these calculations calibrates incident rates, issuer behaviour, gas prices or a network deadline.

The figure source package retains the original codec results and supplies a self-contained Figure 3 projection script, tabulated curve values and separate aggregate measurement markers. It uses exact rational decoy fractions before finite-list rounding. Figure 4 data are calculated from the scalar parameters in Section 9.5; they contain generation indices separately from seconds and leave the all-generation certificate undefined outside the subcritical region. These derived figure datasets add no empirical deployment or temporal-privacy measurements.

B Multi-Agent Trust and Revocation Implementation Checklist

The bounds below are deployment contracts or test invariants. A zero count in a finite test run is not a proof that the corresponding failure is impossible.

The table covers five categories: cryptographic profiles and decoy configuration; trust-domain architecture and gate mediation; freshness, event delivery and cache invalidation; persistent operation records and remote closure; and fail-closed containment and partition fallback. The first column names the category and implementation phase.

Table 12: Implementation controls and verification requirements

Component / Phase	Control & Security Requirement	Verification & Test Method	Target Metric / Bound
Crypto profiles — allocation	Bind list, index, purpose, width, epoch and authority; distinguish unused from active	Boundary, malformed-reference and uninitialised-state fixtures	Every accepted reference has an authenticated interpretation
Crypto profiles — masks	Protect master allocation; disclose only required per-credential mask; decode semantic status	Known-initialisation and holder-knowledge tests; both mask values	$B_i = Y_i \oplus r_i$; no claim of operator-exclusive r_i
Crypto profiles — decoys	Match the full declared observation law, including batches and publication	Side-information and real-only-event negative controls	Posterior cap only under the stated likelihood-ratio bound
Crypto profiles — packing	Declare bit order and enforce semantic monotonicity on every writer path	Byte/word boundary round trips; concurrent writes and replay	No lost irreversible withdrawal in the modelled traces
Crypto profiles — PQC	Select proof suite, encoding, key lifecycle and downgrade rule	Valid/invalid signature, rotation and burst tests	Measured bytes and service cost for the actual suite
Trust domains — PDP and PEP	Map policy engine, administrator and effect gate; prevent bypass	Alternative API, queue, tool and direct-resource tests	Every covered effect passes its protected gate
Trust domains — complete record	Check mandate and assurance on identical g before projection	Incompatible-continuation and changed-argument cases	Common-record and binding invariant
Trust domains — shared caches	Separate authenticated content reuse from tenant-specific decisions	Cross-tenant collision, audience and policy-change tests	No cross-tenant approval reuse; bounded interference
Trust domains — commit race	Serialize relevant withdrawal and effect-authorisation transitions	Systematically interleave invalidation, commit and dispatch	Earlier local withdrawal blocks the stale approval
Freshness — watermark	Bind authoritative completeness, not retrieval or block time	Stale heartbeat, clock failure and fresh false-assertion cases	$t - \omega_{v,l} + \kappa_v \leq \tau_l$ when accepted
Freshness — event delivery	Validate scope; apply idempotently; distinguish receipt and enforcement	Duplicate, reordered, paused and disabled streams	No assumption of complete SSF delivery or immediate enforcement
Freshness — reconciliation	Recover from event gaps using accepted authoritative state	Omit notifications; restart with an old checkpoint	Supported poll, fetch and application delay or expiry fallback

Component / Phase	Control & Security Requirement	Verification & Test Method	Target Metric / Bound
Freshness — cache invalidation	Reevaluate affected typed dependencies and alternatives	Incremental versus full-policy comparison	No missed indispensable dependency under declared completeness
Freshness — partition	Stop ordinary admission when remaining age or validity ends	Isolate all update paths and advance protected time	H_{fresh} and any stricter validity bound
Operation ledger — identity	Bind logical operation, resource, payload and idempotency namespace	Duplicate dispatch, retry and tenant-collision tests	No duplicate logical effect within the receiver's contract
Operation ledger — closure	Retain pending records until authoritative terminal evidence	Timeout, lost receipt, receiver crash and agent retirement	No timeout-based deletion; bounded-state overflow restricts admission
Operation ledger — cancellation	Distinguish requested, accepted and effective cancellation	Race execution against cancellation and expiry	Receiver-enforced deadline; unresolved exposure remains accounted
Fail-closed containment — leases	Preserve source-age cap across repeated renewal	Short-child/stale-parent renewal counterexample	$e_g \leq \min_l (s_l + \tau_l)$ plus other validity limits
Fail-closed containment — capacity	Reserve service and bound admitted work and interference	Withdrawal burst, expensive proofs and malformed inputs	$d^{service} \leq b_o$; Section 12.3
Fail-closed containment — fallback	Preserve authorised protective actions and episode accounting	Failed renewal, unavailable solver and lost clock assurance	Covered fallback; no uncharged gap or risk-account reset
Fail-closed containment — telemetry	Separate logical checks, bytes, delay, harm and legitimate loss	Trace-level attribution and incident-level uncertainty	Contract headroom, observed misses and unresolved exposure

C Implementation contracts and research allocation

C.1 Contract and remaining implementation refinement

Every covered ordinary effect must be justified by one complete record g . A protected local order covers evidence installation, dependency-generation changes and effect authorisation. Withdrawal installed before a later local effect excludes the old approval; remote effects require the receiver's corresponding contract. The protocol must preserve parent-age limits, epoch

checkpoints, unique operation identities and permanent debit records across restart. $P(t)$ retains unresolved operations until contract-valid terminal evidence arrives.

Cross-source coherence requires a declared relation among versions, applicability intervals and adverse evidence. Equal timestamps do not establish coherence. Where the required relation cannot be established, the profile restricts ordinary admission. The source contract distinguishes complete discovery within named authoritative sources from knowledge of every possible adverse fact.

The conformance package must define exact signed bytes, algorithms, key identifiers, domain separation, replay handling and bounded decoding. Separate specifications cover the pinned standards baseline, experimental masking and decoy lifecycle, and packed-ledger initialisation, atomic semantic updates, writer authority and selected finality. Each has positive, negative and boundary test vectors. These are deployment conformance requirements. The companion specifies their model-level role but does not deliver a conforming wire implementation.

C.2 Placement of the remaining questions

Table 13: Research allocation across the two articles

Research question	Role in this article	Companion paper placement
RQ1 Causal decoys	Conditional observation law and reserve costs in Sections 6–8.	Sections 7–8: fixed-cohort construction and full-transcript inference experiments; continuous issuance remains open.
RQ2 Value of masking	Mask limitations and codec cost in Sections 6 and 8.	Sections 7–8: matched mask comparison and exact inference; byte cost remains separate from privacy.
RQ3 Protocol invariants	Eligibility, freshness and ordering obligations in Sections 2, 5 and 11.	Sections 3–4: protocol and finite transition checks; Section 11: deployment guidance. Production refinement remains open.
RQ4 Scaling to harm	Separate capacity, service and harm relations in Sections 7–9.	Sections 6 and 9: synthetic work and stipulated delay, then conditional harm analysis. Empirical calibration remains open.
RQ5 Parameter and tail risk	Conditional offspring and cost premises in Section 9.	Sections 9–10: uncertainty, finite-horizon roots and inference boundaries; rare-event validation remains open.
RQ6 DLT versus mirrors	Common authority and freshness comparison in Section 5.	Sections 5–6: profile obligations and synthetic paired hosting study. Live deployment measurement remains open.
RQ7 Truth and completeness	Trusted-source and discovery boundaries in Sections 1, 3 and 13.	Sections 2–3: source and protocol contract; Sections 10 and 13: evidence and limits. No unconditional truth guarantee.

All seven questions are addressed at the model level within one bounded workflow. A negative result, an identified impossibility under the chosen observer, or a demonstrated failure of an assumed source contract is a valid outcome. Existing anonymous non-revocation mechanisms and their adoption are assessed in Section 5.3.1 and Appendix F. The companion’s Section 5.10 and Appendix F specify their unevaluated adapter and test plan; the supplied codec fixtures do not benchmark them. Layer 2 and state-channel mechanisms, broader post-quantum migration and larger agent populations remain in the companion’s Section 12 outlook. Its Appendix C states the evidence needed beyond the models.

D Complete runnable examples and outcome charts

D.1 Running the examples

This appendix prints all eight code cells of the original companion teaching notebook and adds one plotting-only cell for the existing compression results. The listings contain setup, five examples, a separate codec cell within Example 3, the added outcome chart and the final reproducibility record. Copy each listing into its own Python cell and run the cells in the printed order. The examples share variables from earlier cells. No source file, external dataset, key, credential or mounted drive is needed. The combined file `Paper_1_Appendix_D_Examples.py` contains the same listings with cell markers.

Open <https://colab.research.google.com/> and create a Python notebook, or upload the accompanying notebook through File → Upload notebook [53]. Use an ordinary runtime; a graphics processor is unnecessary. The calculations use the Python standard library. Matplotlib supplies the plots; numerical output remains available if it is not installed. The code makes no network requests. The explanatory random seeds are fixed for repeatability and must not be used as production cryptographic secrets.

All nine printed cells were executed in order in a fresh local Python process, using Python 3.12.14, zlib 1.3.2 and Matplotlib 3.10.8. The eight original cells have Python syntax trees matching the existing notebook after line wrapping, so their calculations and assertions are unchanged. The added cell plots previously computed byte counts and performs no new compression experiment. Figures D1–D5 use these outputs with publication-sized labels. No Google-hosted runtime execution, deployment result or standards-conformance test is claimed.

D.2 Setup

Run this cell first. It imports the libraries, records their versions and creates a container for the numerical results. A missing optional plotting library does not prevent the calculations from running.

```
import sys, platform, json, zlib

try:
    import matplotlib
    import matplotlib.pyplot as plt
except ImportError:
    matplotlib = None
    plt = None

RESULTS = {}
VERSIONS = {
    "python": platform.python_version(),
```

```

    "zlib_runtime": zlib.ZLIB_RUNTIME_VERSION,
    "matplotlib": (
        matplotlib.__version__
        if matplotlib
        else "not installed; plots skipped"
    ),
}
print("Environment:", VERSIONS)
print(
    "All examples use synthetic data. "
    "No network or external data is required."
)

```

D.3 Example 1: one action name can hide incompatible plans

Predict the result before running the cell. The mandate permits processing only with immediate deletion afterwards. The assurance result covers a different continuation that retains the file. Both mention the same action. Should the gate approve it? The complete record contains the action, duration, continuation and bound context.

```

action = "process_design"
context = "agent_A | resource_R | policy_v1 | evidence_v3"
delete_record = (action, 30, "delete_immediately_after_processing", context)
retain_record = (action, 30, "retain_for_further_processing", context)
mandate = {delete_record}
assurance = {retain_record}
project_actions = lambda records: {record[0] for record in records}

wrong = project_actions(mandate) & project_actions(assurance)
correct = project_actions(mandate & assurance)
print("Intersect action names only:", sorted(wrong))
print("Intersect complete records first:", sorted(correct))
assert wrong == {action} and correct == set()

# A new assurance result for the exact mandated record changes the answer.
assurance_for_both = assurance | {delete_record}
correct_after = project_actions(mandate & assurance_for_both)
assert correct_after == {action}
print("After assurance for the deletion plan:", sorted(correct_after))
RESULTS["common_record"] = {
    "wrong_accepts": True,
    "correct_accepts": False,
    "accepts_after_matching_assurance": True,
}

```

Outcome. Intersecting action names returns ['process_design'], while intersecting complete records returns an empty list. Approval becomes possible only when assurance also covers the deletion plan. This is Proposition 1's counterexample. A production gate additionally needs canonical record binding, authenticated evidence, policy evaluation and a current commit-time check; tuple equality implements none of those services.

D.4 Example 2: count real status capacity before estimating traffic

Each bundle has twenty distinct one-bit status entries. Lists contain 131,072 bits, and a chosen fraction is reserved for decoys. Genuine entries may use the remaining positions; the final list may

be partly empty. Bundles may span lists. Predict whether 50% decoys make the raw-capacity factor twenty or forty. Then compare the exact finite-list result with the approximation that ignores rounding.

```

from fractions import Fraction

def ceil_div(a, b):
    return (a + b - 1) // b

def capacity(N, L, k, decoy_share):
    d = Fraction(decoy_share)
    assert N > 0 and L > 0 and k > 0 and 0 <= d < 1
    genuine_per_list = (L * (1 - d)).numerator // (L * (1 - d)).denominator
    assert genuine_per_list > 0, "No genuine status fits in a list."
    lists = ceil_div(N * k, genuine_per_list)
    baseline_lists = ceil_div(N, L)
    return genuine_per_list, lists, lists / baseline_lists, float(k / (1 - d))

N, L, k = 131_072, 131_072, 20
capacity_rows = []
print("decoys genuine/list lists exact factor asymptotic factor")
for share in ["0", "0.25", "0.5", "0.75", "0.9", "0.95"]:
    slots, lists, exact, asymptotic = capacity(N, L, k, share)
    capacity_rows.append((float(share), slots, lists, exact, asymptotic))
    print(
        f"{float(share):5.0%} {slots:13,d} {lists:6,d} "
        f"{exact:12.2f} {asymptotic:18.2f}"
    )
assert capacity(N, L, k, "0.5") == (65_536, 40, 40.0, 40.0)
assert capacity(N, L, k, "0.9")[1] == 201
assert capacity(N, L, k, "0.95")[1] == 401
assert capacity(100, L, k, "0.5")[2] == 1.0
print("At N=100, both cases fit in one list: the exact capacity ratio is 1.")
RESULTS["capacity"] = {
    "N": N,
    "L": L,
    "k_pop": k,
    "rows": capacity_rows,
    "small_population_exact_ratio": 1.0,
}
if plt:
    fig, ax = plt.subplots(figsize=(7, 3.5))
    ax.plot(
        [100 * r[0] for r in capacity_rows],
        [r[3] for r in capacity_rows],
        "o-",
        label="Exact finite-list ratio",
    )
    ax.plot(
        [100 * r[0] for r in capacity_rows],
        [r[4] for r in capacity_rows],
        "--",
        label="Asymptotic factor",
    )
    ax.set(
        xlabel="Reserved decoy share (%)",
        ylabel="Raw capacity / baseline",
    )

```

```

        title="Capacity depends on useful positions",
    )
    ax.legend()
    ax.grid(alpha=0.25)
    fig.tight_layout()
    plt.show()

```

Outcome. For 131,072 bundles, the 50% case needs forty lists. The 90% and 95% cases need 201 and 401 lists. Their asymptotic factors are 200 and 400; integer capacity causes the difference. For only 100 bundles, both the single-status baseline and the twenty-status design with 50% decoys fit in one minimum-sized list. This is storage accounting. It is not bytes per verification, a latency multiplier or an achieved privacy level. Shared credentials require counting unique physical entries before applying the formula.

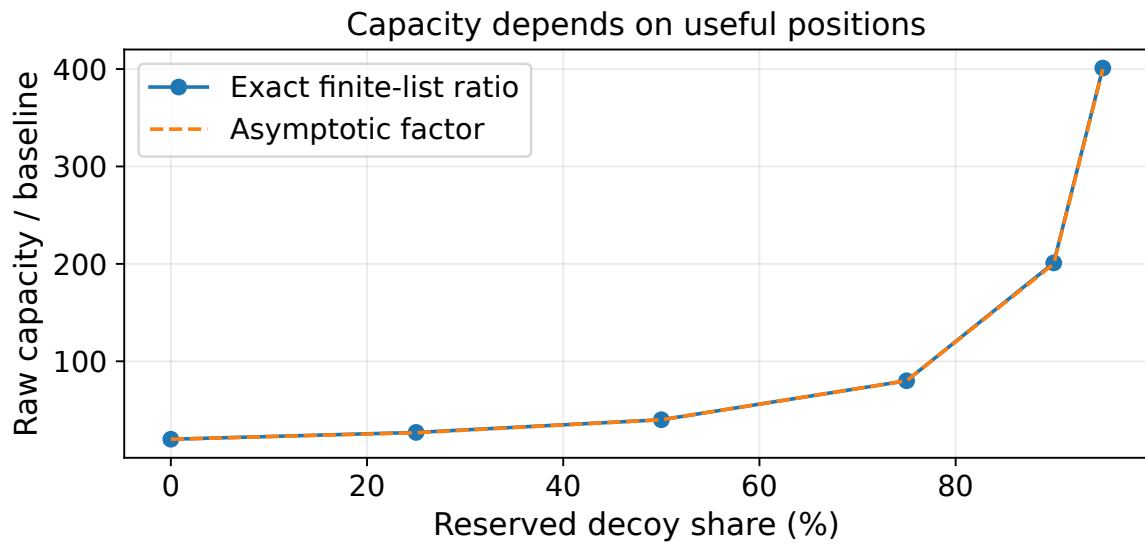


Figure D1: Raw status capacity relative to a one-status-per-bundle baseline. Both populations contain 131,072 bundles and use 131,072-bit lists; the evaluated design assigns twenty distinct statuses per bundle. Markers give finite-list ratios; the dashed line gives the asymptotic factor. The 50% case is exactly forty-fold. Decoy allocation alone does not establish the observation-law assumptions required for membership privacy.

D.5 Example 3: masking, decoys and compression

D.5.1 A fixed mask does not conceal the changes

The toy allocation has eight genuine and eight decoy positions. Unflipped positions use the same per-step event probability and can change only once. The allocation labels are printed only for teaching. Predict whether comparing two public snapshots removes the fixed mask. Also ask whether a known initially active state reveals the mask itself.

```

import random

toy_rng = random.Random(20261009) # Reproducible teaching seed, not a secret.
toy_L, toy_M, event_probability = 16, 8, 0.20
genuine = set(toy_rng.sample(range(toy_L), toy_M))
J = [int(i in genuine) for i in range(toy_L)]
r = [toy_rng.randrange(2) for _ in range(toy_L)]

```

```

x_history = [[0] * toy_L]
for tick in range(1, 4):
    previous = x_history[-1]
    x_history.append(
        [
            int(old == 1 or toy_rng.random() < event_probability)
            for old in previous
        ]
    )
y_history = [
    [mask ^ semantic for mask, semantic in zip(r, x)] for x in x_history
]

print("Teaching allocation J (not public):", "".join(map(str, J)))
print("Public snapshots:")
for tick, y in enumerate(y_history):
    print(f"t={tick}:", "".join(map(str, y)))
assert y_history[0] == r
for tick in range(1, 4):
    observed_changes = [
        a ^ b for a, b in zip(y_history[tick - 1], y_history[tick])
    ]
    real_changes = [
        a ^ b for a, b in zip(x_history[tick - 1], x_history[tick])
    ]
    assert observed_changes == real_changes
    assert all(a <= b for a, b in zip(x_history[tick - 1], x_history[tick]))
assert [a ^ b for a, b in zip(y_history[0], y_history[-1])] == x_history[-1]
assert [mask ^ wire for mask, wire in zip(r, y_history[-1])] == x_history[-1]
real_events = sum(x_history[-1][i] for i in genuine)
decoy_events = sum(x_history[-1][i] for i in range(toy_L) if i not in genuine)
print(
    "Mask cancels exactly. "
    "Initial known-zero semantics reveal the initial mask."
)
print(
    "Final teaching counts: genuine events =",
    real_events,
    "; decoy events =",
    decoy_events,
)
RESULTS["mask_toy"] = {
    "snapshots": y_history,
    "mask_cancelled": True,
    "initial_snapshot_reveals_mask": True,
    "genuine_events": real_events,
    "decoy_events": decoy_events,
}
if plt:
    fig, ax = plt.subplots(figsize=(8, 2.8))
    ax.imshow(
        y_history,
        aspect="auto",
        cmap="Greys",
        vmin=0,
        vmax=1,
        interpolation="nearest",
    )
    ax.set(
        xlabel="Public position",

```

```

        ylabel="Snapshot",
        yticks=range(4),
        xticks=range(toy_L),
        title="Public wire view: black = 1; white = 0",
    )
    fig.tight_layout()
    plt.show()

```

Outcome. Figure D2 shows the public wire snapshots. Every snapshot difference matches the semantic changes. The known all-zero semantic initialization reveals the fixed mask. The recorded seed produces three genuine events and three decoy events, but matching counts in this one realization is not the privacy condition. A real observer may also see writers, transaction types, batching and incident reports. Matching the per-position probability in this toy does not establish equality of those complete observation histories. The reserve is finite, and the printed allocation and disclosed teaching seed are not secret.

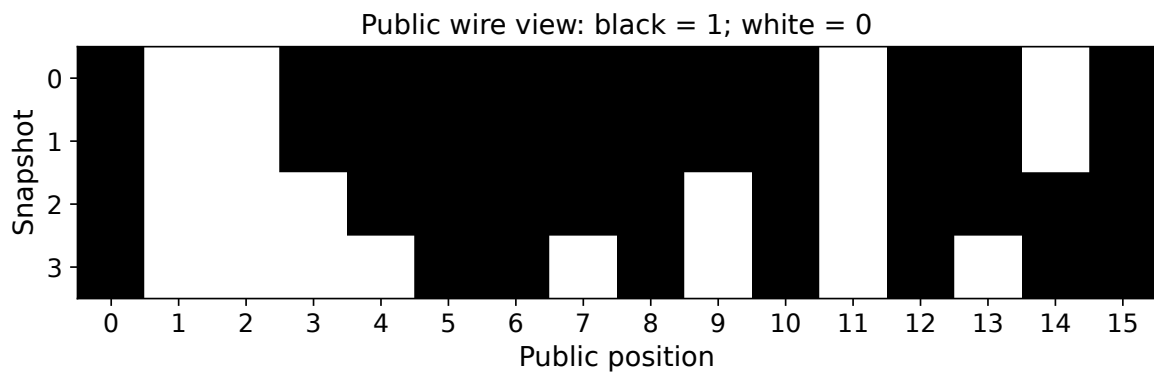


Figure D2: Public wire snapshots for the fixed-mask teaching example with sixteen positions, eight genuine entries and eight decoys. Black denotes one and white denotes zero. The public view does not display the allocation labels. Each position can make one absorbing semantic transition. Comparing snapshots cancels the fixed mask and reveals transition positions; the known all-zero semantic initialization also reveals the mask. Matching the toy event probability and the observed event counts does not establish equality of complete observation histories or deployment privacy.

D.5.2 A separate single-list GZIP fixture

This cell measures GZIP compression on one 131,072-bit list with exactly 512 set bits. A fixed random mask is applied to the same semantic list. Both raw representations contain 16,384 bytes. Predict which representation compresses better. The code checks that decompression and unmasking recover the input.

```

import gzip

codec_L = 131_072
codec_rng = random.Random(17)
sparse = bytearray(codec_L // 8)
for position in codec_rng.sample(range(codec_L), 512):
    sparse[position // 8] |= 1 << (position % 8)
raw_sparse = bytes(sparse)
raw_mask = bytes(codec_rng.getrandbits(8) for _ in range(codec_L // 8))
raw_masked = bytes(a ^ b for a, b in zip(raw_sparse, raw_mask))
compressed_sparse = gzip.compress(raw_sparse, compresslevel=6, mtime=0)
compressed_masked = gzip.compress(raw_masked, compresslevel=6, mtime=0)

```

```

assert gzip.decompress(compressed_sparse) == raw_sparse
assert gzip.decompress(compressed_masked) == raw_masked
assert bytes(a ^ b for a, b in zip(raw_masked, raw_mask)) == raw_sparse
print("Single-list raw bytes:", len(raw_sparse))
print("Sparse GZIP bytes:", len(compressed_sparse))
print("Masked GZIP bytes:", len(compressed_masked))
print(
    "Single-list GZIP size ratio:",
    round(len(compressed_masked) / len(compressed_sparse), 3),
)
RESULTS["gzip_teaching_fixture"] = {
    "L": codec_L,
    "set_bits": 512,
    "seed": 17,
    "level": 6,
    "mtime": 0,
    "raw_bytes": len(raw_sparse),
    "sparse_gzip_bytes": len(compressed_sparse),
    "masked_gzip_bytes": len(compressed_masked),
    "ratio": len(compressed_masked) / len(compressed_sparse),
}

```

Outcome in the recorded environment. The sparse GZIP payload is 1,150 bytes; the masked GZIP payload is 16,407 bytes. Their ratio is about 14.267. Figure D5 plots these outcomes. This cell counts neither base64 encoding nor credential envelopes. It is a different density, population and trial design from Appendix A, so its single-list ratio does not reproduce or replace the article's 40.03-fold and 1475.44-fold encoded population ratios. Random masking removes sparse compression savings in this fixture. Compressed length is not a privacy score, and the demonstration is not a conforming masked W3C status method. Compression versions can change exact byte counts.

D.6 Example 4: a readable local list can become too old

Connectivity is lost at time 100. The latest authoritative completeness watermark is 80, the maximum accepted age is 90 seconds, and the clock margin is 2 seconds. There are 68 seconds of remaining freshness. Predict whether repeated thirty-second lease renewals can move the absolute cutoff beyond time 168. Rereading the same local object does not change the authoritative watermark.

```

watermark, partition_start = 80.0, 100.0
max_age, clock_margin, lease_duration = 90.0, 2.0, 30.0
freshness_end = watermark + max_age - clock_margin
remaining = max(0.0, freshness_end - partition_start)

def issue_lease(now):
    if now >= freshness_end:
        # Unknown positive status: do not create an ordinary approval.
        return None
    return min(now + lease_duration, freshness_end)

def may_commit(now, expiry):
    return expiry is not None and now < expiry and now < freshness_end

print("Remaining freshness budget:", remaining, "seconds")

```

```

lease_rows = []
for now in [100.0, 129.0, 158.0, 167.0, 168.0]:
    expiry = issue_lease(now)
    lease_rows.append((now, expiry))
    print(
        f"Renewal at t={now:5.1f}: expiry={expiry}; "
        f"ordinary approval={may_commit(now, expiry)}"
    )
assert remaining == 68.0
assert issue_lease(100) == 130.0 and issue_lease(158) == 168.0
assert issue_lease(168) is None and not may_commit(168, 168)
unsafe_expiry = 167 + lease_duration
assert unsafe_expiry > freshness_end
print(
    "Unsafe renewal ignoring parent freshness would expire at", unsafe_expiry
)
print("Rereading the same object leaves watermark =", watermark)
RESULTS["freshness"] = {
    "remaining_seconds": remaining,
    "freshness_end": freshness_end,
    "lease_rows": lease_rows,
    "unsafe_renewal_expiry": unsafe_expiry,
}
if plt:
    fig, ax = plt.subplots(figsize=(7, 3.2))
    for row, (start, end) in enumerate(lease_rows[:-1]):
        ax.plot([start, end], [row, row], "o-", linewidth=3)
    ax.axvline(
        freshness_end,
        color="firebrick",
        linestyle="--",
        label="Authoritative freshness cap",
    )
    ax.set(
        xlim=(95, 200),
        ylim=(-0.5, 3.5),
        xlabel="Local time (seconds)",
        yticks=range(4),
        yticklabels=["Lease 1", "Renewal 2", "Renewal 3", "Renewal 4"],
        title="Renewal cannot move the parent freshness cap",
    )
    ax.legend(loc="lower right")
    ax.grid(axis="x", alpha=0.25)
    fig.tight_layout()
    plt.show()

```

Outcome. The first lease expires at 130. Later renewals are capped at 168, and a new lease at 168 is refused. A renewal that ignored parent freshness could incorrectly extend expiry to 197. The gate uses a strict pre-expiry check. Unknown positive status is not a factual assertion of revocation. This scalar example assumes truthful completeness, a justified clock margin and correct enforcement. It does not test consensus, compromised publishers, restart handling or pending remote effects.

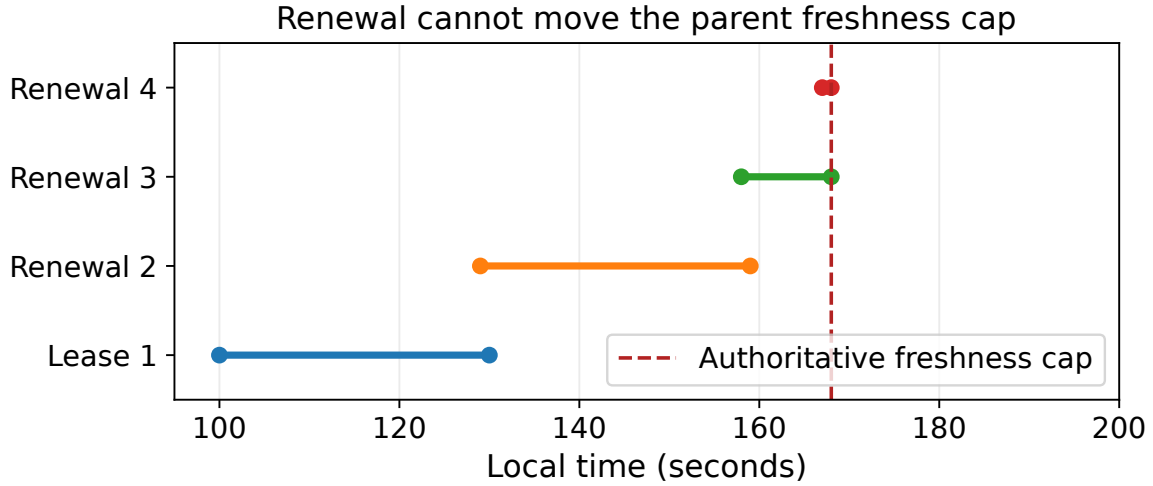


Figure D3: Lease issue times and expiry bounds under an unchanged authoritative watermark. The freshness cutoff is time 168; renewals cannot move it. Right-hand markers denote expiry boundaries, not accepted instants: the code permits ordinary effects only strictly before expiry. A readable local cache or ledger copy cannot establish the content of unseen authoritative updates during a partition.

D.7 Example 5: interpret a conditional cascade bound

The scalar illustration uses a conditional detection-and-assessment mean bound of ten seconds and an offspring-intensity bound of 0.04 per second. With one root, unit conditional episode cost and no separately pending harm, the all-generation expected-harm bound is available only when the expected offspring bound K is below one. Predict the change between delays of fourteen and ten seconds, and what the code should report at fifteen seconds.

```
import math

beta, detection_mean_bound = 0.04, 10.0

def scalar_certificate(influence_delay):
    assert influence_delay >= 0
    K = beta * (detection_mean_bound + influence_delay)
    return K, (1 / (1 - K) if K < 1 else None)

harm_rows = []
print("delay (s)      K      all-generation expected-harm upper bound")
for delay in [2.0, 10.0, 14.0, 15.0, 20.0]:
    K, bound = scalar_certificate(delay)
    harm_rows.append((delay, K, bound))
    label = f"{bound:.4f}" if bound is not None else "certificate unavailable"
    print(f"{delay:9.1f} {K:7.2f} {label}")
assert math.isclose(scalar_certificate(2)[1], 1 / 0.52)
assert math.isclose(scalar_certificate(10)[1], 5.0)
assert math.isclose(scalar_certificate(14)[1], 25.0)
assert scalar_certificate(15)[1] is None
assert sum(1.0**g for g in range(21)) == 21.0
print(
    "At K=1, an exact branching model has cumulative mean 21 "
    "through generation 20."
)
```

```

RESULTS["scalar_harm"] = {
    "beta_per_second": beta,
    "detection_mean_bound_seconds": detection_mean_bound,
    "rows": harm_rows,
    "critical_20_generation_mean": 21.0,
}
if plt:
    fig, axes = plt.subplots(1, 2, figsize=(10, 3.8))
    delays = [i / 20 for i in range(300)]
    bounds = [scalar_certificate(d)[1] for d in delays]
    axes[0].plot(delays, bounds)
    axes[0].axvline(15, color="firebrick", linestyle="--")
    axes[0].axvspan(15, 20, color="lightgray", alpha=0.5)
    axes[0].text(17.5, 14, "Certificate\nunavailable", ha="center")
    axes[0].set(
        xlim=(0, 20),
        ylim=(0, 30),
        xlabel="Covered influence delay (seconds)",
        ylabel="Conditional expected-harm upper bound",
        title="One type: vertical range limited to 30",
    )
    generations = list(range(21))
    for K in [0.48, 1.0, 1.2]:
        cumulative = [sum(K**j for j in range(g + 1)) for g in generations]
        axes[1].plot(generations, cumulative, label=f"K={K:g}")
    axes[1].set(
        xlabel="Generation (not seconds)",
        ylabel="Cumulative mean episodes",
        title="Separate exact branching construction",
    )
    axes[1].legend()
    for ax in axes:
        ax.grid(alpha=0.25)
    fig.tight_layout()
    plt.show()

```

Outcome. Supported influence delays of 2, 10 and 14 seconds give conditional bounds of about 1.9231, 5 and 25 expected harm units. At 15 seconds, the program reports that the certificate is unavailable. It does not establish infinite actual loss. The separate branching plot uses generation number, not elapsed time; at $K = 1$ its cumulative mean through generation 20 is 21. Actual deployment use requires the paper's uniform conditional rate, detection, cost and receiver-enforced influence assumptions. A file gate alone does not contain a separate uncontrolled messaging path.

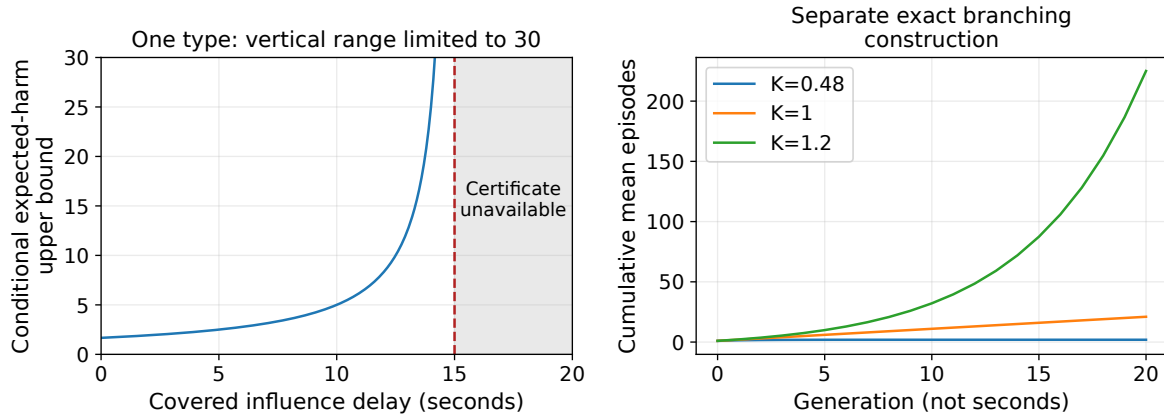


Figure D4: Left: the conditional scalar expected-harm bound versus supported covered-influence delay; values above thirty are outside the displayed vertical range. The shaded region has no subcritical certificate and is not a measured infinite-loss region. Right: cumulative means through a finite number of generations in a separately stipulated branching construction. Generation indices are not time predictions. Neither panel is fitted to an industrial deployment.

D.8 Plot the compression outcome

This additional plotting cell reads the byte counts already calculated in Section D.5.2. Both semantic representations have equal raw length. The chart separates the raw-capacity baseline from the compressed sparse and compressed masked payloads. It adds no model, trial or compression run.

```
# Plot the byte counts already computed in the codec example.
codec_sizes = [
    len(raw_sparse),
    len(compressed_sparse),
    len(compressed_masked),
]
if plt:
    fig, ax = plt.subplots(figsize=(7, 3.4))
    labels = ["Raw\n(bboth profiles)", "GZIP\nunmasked", "GZIP\nmasked"]
    ax.bar(labels, codec_sizes, color=["#777777", "#356887", "#bd6d35"])
    for position, size in enumerate(codec_sizes):
        ax.text(position, size + 400, f"{size:,}", ha="center")
    ax.set(
        ylim=(0, 19_500),
        ylabel="Bytes per list",
        title="Same raw capacity; different GZIP size",
    )
    ax.grid(axis="y", alpha=0.25)
    ax.set_axisbelow(True)
    fig.tight_layout()
    plt.show()
```

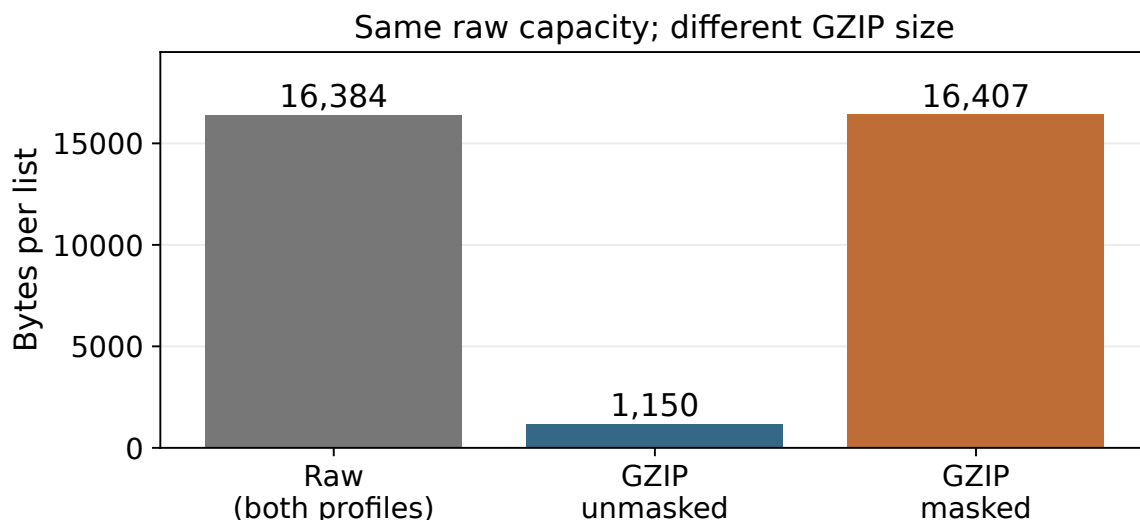


Figure D5: Single-list compression outcome for the 512-set-bit teaching fixture. The raw list has 16,384 bytes in either representation. GZIP produces 1,150 bytes for the sparse unmasked list and 16,407 bytes for its randomly masked representation. Masking removes the compression saving; the masked stream is slightly larger than its raw input because of codec overhead. These values exclude base64 encoding and envelopes and use a different fixture from the article’s population comparisons. They are not network, gas, privacy or safety measurements.

D.9 Retain the reproducibility record

The final cell prints the environment and all numerical results as a machine-readable record. It is optional for understanding the examples, but included here so the complete notebook code is available within the paper. Change one parameter at a time, predict its effect, and rerun from setup. Treat a failed assertion as a prompt to inspect the changed assumptions. These examples explain the models; they do not replace implementation tests or the companion paper’s distinct protocol and workload experiments.

```
print(json.dumps({'versions': VERSIONS, 'results': RESULTS}, indent=2))
```

E Recommendations for W3C status-list specifications

E.1 Applicable documents and the proposal boundary

This appendix proposes standards work. It does not introduce additional W3C conformance requirements. The stable implementation baseline is Bitstring Status List v1.0, a Recommendation of 15 May 2025 [12]. The primary development target is Bitstring Status List v1.1, whose latest published version at inspection on 9 October 2026 is the First Public Working Draft of 24 September 2026 [13]. The Verifiable Credentials Working Group maintains both through the same specification repository. Section numbers below refer to these dated versions.

Three forms of randomness need different treatment. Random index allocation reduces information carried by position. Decoy events add ambiguity about which unknown positions are genuine. Random initial masks change how published bits encode status. These operations have different conformance, privacy and capacity consequences. A recommendation to use one does not justify the others.

Both versions recommend random index assignment in Section 2.1. Their Section 3.3 generation algorithm starts from zero and assigns the status of issued credentials. Their non-normative Section 6.5 discusses decoys, leaves generation algorithms outside scope and warns that distinguishable decoys can reduce privacy [12, 13]. Accordingly, this article does not interpret that discussion as permission to change the normative output. Nonzero positions belonging to no issued credential need explicit standards clarification or a separately specified experimental method.

E.2 Proposed changes and required evidence

The recommendations below distinguish explanatory improvements from changes that need a new processing profile. “Experimental” means that the article has not established deployment readiness. “Clarification” means a proposed editorial or interoperability improvement; it does not imply working-group acceptance.

Table 14: Proposed standards changes and evidence requirements

Target document and section	Current baseline	Proposed amendment	Evidence and review status
Bitstring v1.1, Sections 2.1 and 6.2	Random index assignment is recommended.	Explain unpredictability, allocation secrecy and avoidance of issuance-order leakage; retain ordinary zero-based status semantics.	Allocation and side-information tests; clarification.
Bitstring v1.1, Sections 3.3 and 6.5	Issued-credential generation and non-normative decoy discussion coexist.	Clarify whether and how reserved unissued positions may acquire cover states. State the producer-conformance boundary explicitly.	Positive and negative generation fixtures; standards clarification needed.
Bitstring v1.1, Sections 6.5–6.7	Decoy distinguishability and monitoring are discussed.	Add a full-history threat model covering timing, batches, cohort births, rotation and public transaction metadata.	Matched and deliberately mismatched histories from Section 6; explanatory proposal.
Bitstring v1.1, Sections 1.3, 2.1 and 3.2–3.4	Standard readers interpret the expanded status value directly.	Keep XOR decoding outside the default method. Define a distinct method and version, authenticated parameters and unsupported-method rejection; see Section E 4.	Cross-reader and downgrade fixtures; experimental profile only.
Bitstring v1.1, Sections 2.2, 3.3–3.4 and 7.1	The current encodedList representation requires GZIP and Multibase base64url.	Evaluate an authenticated raw-byte option within a distinct experimental method or explicit future revision. Specify bit layout, length limits and rejection rules; see Section E 4.	Equal-workload byte, CPU and memory comparisons; codec confusion, downgrade and malformed input tests. Experimental proposal only.

Target document and section	Current baseline	Proposed amendment	Evidence and review status
Bitstring v1.1, Sections 2.2, 6.3–6.4 and 7.2	Refresh, caching and validity have separate roles.	Explain that retrieval privacy does not establish source freshness, completeness or resource enforcement. Include partition and stale-cache examples.	Watermark, replay and lease-expiry traces; implementation guidance.
Bitstring v1.1, Sections 2.3 and 3.5	The terse entry defines a separate sizing and conversion profile.	Report raw, compressed and signed sizes separately when adding cover. Add index-boundary and purpose-binding tests.	Profile-specific fixtures; clarification and test contribution.

The first recommendation concerns the selection of unused positions, not randomizing the status of live credentials. In an implementation, a cryptographically secure random number generator should draw indices without bias from the eligible pool, with collision handling that preserves uniqueness. A public deterministic sequence or a leaked allocation table can defeat the intended uncertainty. This is implementation guidance proposed here, not an additional quoted W3C requirement.

For irreversible revocation, the proposed decoy lifecycle makes one semantic transition and never restores that slot within the same epoch. Repeated independent toggles are a useful negative test: a position that repeatedly returns to active can identify itself as cover. Matching the average number of flips is also insufficient. The test corpus should include real-only incident bursts, batch correlations, changing issuance rates and reserve exhaustion. The observed public transcript includes object lengths, publication times, contract writers and event payloads, not only decoded bits.

E.3 Why random default wire bits are not recommended

For the masked research method, the reader computes $x_i = Y_i \oplus r_i$. An ordinary reader instead interprets Y_i directly. If $r_i = 1$, the two readers assign opposite states. Adding an unfamiliar optional field while retaining an otherwise ordinary status entry therefore risks inconsistent acceptance. An experimental implementation should use an unambiguous method identifier and reject unsupported decoding. It should authenticate the mask parameter, position, list, epoch and purpose together. Omitting the parameter must not silently select ordinary decoding.

There is also a privacy limit independent of implementation quality. If a public initial snapshot has known semantic state zero, then $Y_i(t_0) = r_i$. It discloses the mask. Even without that snapshot, comparing two versions cancels a fixed mask and reveals changed positions. More entropy in the random generator does not remove either fact. Under the matched-history membership model in Section 6, unmasked and masked variants can satisfy the same conditional privacy statement. The masked variant pays the compression cost in Section 8 without automatically gaining additional membership protection.

The proposed default is therefore standard unmasked semantics with unpredictable index assignment, authenticated retrieval and locally enforced freshness. Decoys are a separately evaluated extension. Masking should advance only if a distinct observer model demonstrates an added benefit that justifies its bytes, key or parameter handling, and interoperability cost. A

byte fixture alone cannot demonstrate that benefit.

E.4 Explicit masked status methods and codec selection

Method identification. XOR masking changes status interpretation and therefore needs an explicit experimental status method or incompatible processing profile. Random index allocation alone does not. Decoys that retain ordinary decoding raise the separate producer conformance question in Section E 1. An unknown optional masking flag is insufficient: a legacy processor could ignore it and interpret the wire bit as the semantic status. The experimental entry and list should identify one fully specified method and version, without also claiming ordinary BitstringStatusListEntry compatibility [12, 13].

Codec choice. A new compression algorithm is unnecessary. A future method could define raw packed bytes, encoded as unpadded Multibase base64url for a JSON representation, alongside a separately identified GZIP option. The selected codec and its processing rules must be authenticated. Sections 2.2 and 3.3–3.4 of the current W3C specifications require GZIP generation and expansion. Raw bytes cannot silently replace that representation in credentialSubject.encodedList while retaining the existing conformance claim [12, 13]. A distinct experimental vocabulary or an explicit future standards revision must define the alternative payload and mandatory processing rules.

Efficiency and privacy. For independent uniformly random masks, full snapshots lose the sparsity that ordinary GZIP exploits. A raw option avoids compression and decompression work, but its net benefit requires measurement. The Section 8 fixture expands from 16,384 raw bytes to 16,407 GZIP bytes. Removing GZIP saves 23 bytes per list, about 0.14%, before text encoding and envelopes. It does not recover the sparse baseline’s compression advantage or remove base64 expansion. The proposal therefore targets explicit interoperability and measurable processing costs; it supplies no additional privacy guarantee. The limitations and preferred unmasked default in Section E 3 remain unchanged.

Authenticated bindings. The list descriptor should bind the method and version, list identity, epoch, status purpose, logical bit length, entry width, bit order and codec. Fixed values may be specified by the referenced profile. Each authenticated snapshot should cover that descriptor, an ordered snapshot version, authoritative freshness evidence, validity interval and payload. The credential should bind its method, list, epoch, position, purpose and mask parameter. Snapshot authentication and decoding precede semantic unmasking. The private allocation array and per-position genuine-or-decoy labels must not be published as codec metadata. Authentication does not by itself keep a disclosed mask secret.

Reader behaviour. Under the proposed profile, an unsupported method or codec, missing parameter, inconsistent binding, expired view or detected rollback produces an error and prevents authorisation based on that status evidence. A reader must not fall back from masked semantics to ordinary bit interpretation, infer the codec by trial decoding, or ignore a mandatory status check. Define exact decoded length, padding-bit rules and bounded parsing and decompression. Successful GZIP decoding alone establishes neither semantic compatibility nor current status.

Publication policy and deltas. Fix the codec policy for a profile or epoch unless an evaluated observer model permits adaptive selection; switching according to compressibility can add a density-dependent signal. Sparse changes between fixed-mask snapshots may still compress well. An optional delta format must authenticate its base and result versions and their content bindings, preserve epoch and profile identity, detect missing updates, and provide authenticated full-snapshot recovery. Codec or publication changes must not relax the existing freshness deadline. These properties need a separate delta specification.

Review and acceptance. Submit method identification, codec selection and decoding limits as explicit proposals against Bitstring Status List v1.1 Sections 1.3, 2.1–2.2, 3.2–3.4 and 7.1 [13]. Provide cross-reader and downgrade tests, wrong-mask and wrong-epoch cases, malformed-length and decompression tests, replay and delta-gap cases, and full-transcript privacy tests. Compare complete transmitted bytes, processing time and peak memory at equal semantic workload and freshness. The current codec fixtures establish representation sizes only; they do not validate this proposed method or justify its adoption.

E.5 Related specifications and contribution route

Credential-wide privacy guidance belongs in the Verifiable Credentials Data Model. Its v2.0 Recommendation supplies the existing Sections 8.4, 8.6 and 8.11 on identifier, metadata and validation correlation [4]. For new work, the v2.1 Working Draft of 30 September 2026 places the relevant material in Appendix B, especially B.5, B.6 and B.9. Its companion Threat Model v2.1, a Group Note Draft of 27 September 2026, includes threat T5 on status and revocation lookup [54, 55]. Proposed cross-references should separate lookup privacy, unknown-position membership and issuer-activity inference. These documents are appropriate for explaining the threat; the bitstring specification remains the destination for codec and generation changes.

Recognized Entities v1.0, Sections 3.3 and 4, is a separate target for scoped recognition of status publishers and discovery dependencies [14]. It is experimental. Recognition establishes the authority accepted by policy; it does not prove that a publisher reports every withdrawal truthfully or promptly. This article proposes linking recognition withdrawal to dependency invalidation and bounded reliance, while keeping mask generation outside that document.

A review contribution should start with a narrow issue in w3c/vc-bitstring-status-list: the relationship between Section 3.3 and unissued decoy positions. A second issue can supply full-transcript privacy guidance and the runnable counterexamples from Appendix D. The distinct method and codec proposal in Section E 4 should be reviewed separately, with complete algorithms, explicit compatibility rules and adversarial fixtures. No issue has been submitted as part of this study. Acceptance would require working-group review and interoperability evidence beyond these simulations.

F Anonymous status proofs and adoption evidence

F.1 Proof binding and receiver freshness

This is a proposed interface contract, not a new accumulator construction or security theorem. Let $\mathcal{C}_{r,e}$ authenticate registry status for registry r at epoch e . The profile-specific relation **Active** means active-set membership, revoked-set non-membership, or an authenticated unrevoked leaf. An authenticated envelope binds that state to its authority, profile, version and justified completeness watermark $\omega_{r,e}$. A proof protocol must bind the same signed credential to its hidden revocation handle μ and witness ξ :

$$\begin{aligned} \exists(c, \mu, \xi, \sigma) : & \quad \text{IssuedBound}(pk_r, c, \mu, \sigma) \\ & \quad \wedge \text{Active}(\mathcal{C}_{r,e}, \mu, \xi) \\ & \quad \wedge \text{Policy}(c, g) = 1. \end{aligned} \tag{F1}$$

Here c contains the relevant credential claims; σ denotes the required issuer authentication. Bind the selected state, verifier challenge, audience and complete record g using the chosen protocol's

proved composition. An independent proof about some other unrevoked credential is insufficient. The receiver separately authenticates the state and enforces:

$$\text{StateAccepted}_v(r, e) = 1, \quad t - \omega_{r,e} + \kappa_v \leq \Delta t_{\max,r}. \quad (\text{F2})$$

A new nonce or proof time does not refresh $\omega_{r,e}$. An installed relevant withdrawal cannot be bypassed by choosing an old root that still verifies cryptographically. Across mandatory registries, take the minimum residual freshness, lease and validity budget. Section 5.4.4’s partition rule and the complete-record gate remain applicable. These are additional runtime obligations, not claims that the existing AnonCreds wire format already supplies a completeness watermark.

F.2 Lifecycle cost and invalidation work

Measure total transmitted bytes over a common horizon T , including actual replication and reuse:

$$B_{\text{alt}}(T) = B_{\text{setup}}(T) + B_{\text{state}}(T) + B_{\text{witness}}(T) + B_{\text{proof}}(T). \quad (\text{F3})$$

Count public auxiliary or path distribution once in its applicable term, root envelopes and updates in state traffic, and presentations in proof traffic. Local witness computation is a separate work metric. A compact root does not imply constant lifecycle traffic. An ordinary Merkle path has approximately $b_h \lceil \log_2 N \rceil$ hash bytes for N leaves and b_h -byte hashes; this is not a generic zero-knowledge proof-size bound. The snapshot ratios in Section 8 cannot rank these protocols.

For registry-wide invalidation, let $\mathcal{D}_{r,e}$ contain its dependent live decisions. A conservative work budget is:

$$W_{r,e} = \sum_{o \in \mathcal{D}_{r,e}} (w_o^{\text{hold}} + w_o^{\text{recheck}}). \quad (\text{F4})$$

Count each scheduled job once and bound later arrivals separately. Even one credential withdrawal can require a whole registry cohort to pause and revalidate. This is total scheduled work. The stopping service contract must reserve capacity for enforceable hold and closure jobs plus bounded interference; cooperative proof generation belongs to a separate readmission budget. Stopping must not await a holder’s new proof. Witness repair and proof generation primarily determine readmission. Test stale proofs as unavailable evidence, not as proof of revocation.

F.3 Adoption evidence and limits

This targeted review records public evidence checked on 10 October 2026. It distinguishes available services, implementation support and conformance tests; it is not a market census.

Table 15: Adoption evidence for status representations, checked 10 October 2026

Option / ecosystem	Evidence of current use or support	Limit of the evidence
BC Wallet and British Columbia court access	The government service accepts Person and Lawyer credentials. The maintained government wallet code includes AnonCreds and Indy registry dependencies [32,33].	Shows an active credential service and implementation support, not the revocation profile of every credential or production check volumes.
AnonCreds project	The 2025 annual review reports continuing v1 adoption and maintenance, while recording further work on deployment-ready v2 revocation [36].	Adoption measurement is incomplete; planned v2 work is not evidence of general production readiness.
Historical Sovrin network	The Foundation reports its dissolution in May 2025. Its January 2026 announcement describes MainNet as a read-only archive hosted on one Trinsic server [56].	Historical Sovrin adoption must not be counted as an active multi-steward update network. This does not imply that Indy or AnonCreds has ceased development.
W3C Bitstring Status List	Recommendation status and an interoperability report with eight implementation columns; test run dated 4 October 2026 [12,34].	Participation is not complete conformance or production deployment; the report includes incomplete and unsuccessful cases.
cheqd	Product documentation supports both Bitstring Status Lists and AnonCreds Status Lists through DID-linked resources [35].	Support for both profiles does not disclose customers' relative usage or verification volumes.
walt.id	Issuer2 documentation supports Bitstring Status List, StatusList2021 and Token Status List, subject to credential-format compatibility [57].	A generic bitstring or Token Status List is not evidence of W3C Bitstring Status List adoption. Product support does not quantify use.

The evidence supports coexistence. No reviewed source supplies comparable revocation-enabled credential or verification volumes from which to estimate market share. Bitstrings provide this study's interoperable baseline. Anonymous proofs are a relevant alternative when hiding the presentation's status locator warrants witness and proof management. Selection requires a matched workload and explicit observer model; a standards label or a compact root alone does not establish superiority.

References

- [1] C. Stöcker, “Governing AI That Acts Across Companies: How responsible leaders build first mover advantage through AI Safety and Alignment,” Spherity AI Safety Series executive brief, Sep. 28, 2026, sec. 5.1 and app. A.3. <https://spherity.github.io/spherity-research/Spherity-AI-Safety-Three-Paper-Series-Executive-Brief.pdf>
- [2] C. Stöcker, “Risk-Bounded Runtime Assurance for Dynamic Multi-Agent Systems under Partial Observation,” Spherity Research, 2026, secs. 9 and 14, eqs. (10)–(11). <https://spherity.github.io/spherity-research/Spherity-AI-Safety-Series-Paper-2-Runtime-Assurance.pdf>
- [3] Spherity Research, “AI Safety Three Paper Series,” research landing page, 2026. Accessed: Oct. 8, 2026. <https://spherity.github.io/spherity-research/governing-ai-that-acts-across-companies.html>
- [4] W3C, “Verifiable Credentials Data Model v2.0,” Recommendation, May 15, 2025. <https://www.w3.org/TR/vc-data-model-2.0/>
- [5] D. J. Watts, “A simple model of global cascades on random networks,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 9, pp. 5766–5771, 2002. <https://doi.org/10.1073/pnas.082090499>
- [6] S. V. Buldyrev et al., “Catastrophic cascade of failures in interdependent networks,” *Nature*, vol. 464, pp. 1025–1028, 2010. <https://doi.org/10.1038/nature08932>
- [7] C. Granell, S. Gómez, and A. Arenas, “Dynamical interplay between awareness and epidemic spreading in multiplex networks,” *Physical Review Letters*, vol. 111, art. 128701, 2013. <https://doi.org/10.1103/PhysRevLett.111.128701>
- [8] O. Diekmann, J. A. P. Heesterbeek, and M. G. Roberts, “The construction of next-generation matrices for compartmental epidemic models,” *Journal of the Royal Society Interface*, vol. 7, no. 47, pp. 873–885, 2010. <https://doi.org/10.1098/rsif.2009.0386>
- [9] D. Lee and M. Tiwari, “Prompt Infection: LLM-to-LLM Prompt Injection within Multi-Agent Systems,” arXiv:2410.07283, preprint, 2024. <https://arxiv.org/abs/2410.07283>
- [10] H. An et al., “ACIArena: Toward Unified Evaluation for Agent Cascading Injection,” in *Proceedings of ACL*, vol. 1, pp. 10049–10066, 2026. <https://doi.org/10.18653/v1/2026.acl-long.457>
- [11] R. Pang et al., “Zanzibar: Google’s Consistent, Global Authorization System,” in *USENIX Annual Technical Conference*, pp. 33–46, 2019. <https://www.usenix.org/conference/atc19/presentation/pang>
- [12] W3C, “Bitstring Status List v1.0,” Recommendation, May 15, 2025. <https://www.w3.org/TR/2025/REC-vc-bitstring-status-list-20250515/>
- [13] W3C, “Bitstring Status List v1.1,” First Public Working Draft, Sep. 24, 2026. <https://www.w3.org/TR/2026/WD-vc-bitstring-status-list-1.1-20260924/>
- [14] W3C, “Recognized Entities v1.0,” Working Draft, Sep. 6, 2026. <https://www.w3.org/TR/2026/WD-vc-recognized-entities-1.0-20260906/>
- [15] W3C, “Recognized Entities Threat Model v1.0,” Group Note Draft, Oct. 8, 2026. <https://www.w3.org/TR/2026/DNOTE-vc-recognized-entities-threat-model-1.0-20261008/>

- [16] F. Hoops, J. Gebele, and F. Matthes, “CRSet: Private Non-Interactive Verifiable Credential Revocation,” arXiv:2501.17089v3, preprint, Sep. 17, 2025. <https://arxiv.org/html/2501.17089v3>
- [17] W3C, “Data breaches, identity theft, and revocation: a real-world case study on why W3C is developing Verifiable Credential Barcodes,” press release, Oct. 7, 2026. <https://www.w3.org/press-releases/2026/data-breaches-identity-theft-and-revocation-a-real-world-case-study-on-why-w3c-is-developing-verifiable-credential-barcodes/>
- [18] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero Trust Architecture,” NIST SP 800-207, Aug. 2020. <https://doi.org/10.6028/NIST.SP.800-207>
- [19] OpenID Foundation, “Shared Signals Framework Specification 1.0,” Final Specification, Aug. 29, 2025. https://openid.net/specs/openid-sharedsignals-framework-1_0-final.html
- [20] OpenID Foundation, “Continuous Access Evaluation Profile 1.0,” Final Specification, Aug. 29, 2025. https://openid.net/specs/openid-caep-1_0-final.html
- [21] NIST, “Module-Lattice-Based Digital Signature Standard,” FIPS 204, Aug. 2024. <https://doi.org/10.6028/NIST.FIPS.204>
- [22] C. Stöcker, “Runtime Revocation in Multi-Agent Systems: Protocol Verification and Evaluation of Privacy and Containment,” companion research manuscript, Oct. 10, 2026.
- [23] S. Jaques, M. Lodder, and H. Montgomery, “ALLOSAUR: Accumulator with Low-Latency Oblivious Sublinear Anonymous credential Updates with Revocations,” in ACM ASIA CCS, 2024, doi: 10.1145/3634737.3637641. Open version: Cryptology ePrint 2022/1362. <https://eprint.iacr.org/2022/1362>
- [24] Hyperledger Indy. Indy SDK for Node.js: revocation registry and witness interfaces. Accessed 10 October 2026. Source
- [25] AnonCreds Working Group. AnonCreds Specification, version 1.0, draft. Accessed 10 October 2026. Source
- [26] A. Henzinger, M. M. Hong, H. Corrigan-Gibbs, S. Meiklejohn, and V. Vaikuntanathan, “One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval,” in USENIX Security, pp. 3889–3905, 2023. <https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger>
- [27] K. EdalatNejad, S. Faust, J. Hofmann, P.-F. Lehwalder, and T. Schneider, “Do You Need a Receipt? Anonymous Credential Revocation at Continental Scale via Private Record Certification,” in USENIX Security, pp. 3179–3198, 2026. <https://www.usenix.org/conference/usenixsecurity26/presentation/edalatnejad>
- [28] M. Thomson and C. A. Wood, “Oblivious HTTP,” RFC 9458, Jan. 2024. <https://www.rfc-editor.org/rfc/rfc9458.html>
- [29] A. Fraser and S. Schneider. A Formal Security Analysis of Hyperledger AnonCreds. IEEE European Symposium on Security and Privacy, 2025. IACR ePrint 2025/694. Source
- [30] M. Rosenberg, J. White, C. Garman, and I. Miers. zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure. IEEE Symposium on Security and Privacy, 2023. Source
- [31] W3C. Data Integrity BBS Cryptosuites v1.0. Candidate Recommendation Draft, 10 September 2026. Source

- [32] Province of British Columbia. Access to Court Materials: Access Requirements. Accessed 10 October 2026. Source
- [33] Province of British Columbia. BC Wallet Mobile, app/package.json, main branch. Accessed 10 October 2026. Source
- [34] W3C. VC Bitstring Status List Interoperability Report 1.0. Test run 4 October 2026. Accessed 10 October 2026. Source
- [35] cheqd. Product Documentation: Create Status Lists. Accessed 10 October 2026. Source
- [36] LF Decentralized Trust Technical Advisory Council. 2025 Annual Hyperledger AnonCreds. Accessed 10 October 2026. Source
- [37] P. Bolte, D. von der Bey, and L. Leifermann, “ERC-7506: Trusted Hint Registry,” Ethereum Improvement Proposals, Draft, Aug. 2023. Status checked: Oct. 9, 2026. <https://eips.ethereum.org/EIPS/eip-7506>
- [38] Ethereum documentation, “Proof of stake,” consensus and finality. Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/>
- [39] Ethereum documentation, “JSON-RPC API,” block parameters and eth_call. Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/apis/json-rpc/>
- [40] Ethereum Improvement Proposals, “EIP-1186: RPC-Method to get Merkle Proofs eth_getProof,” account and storage proof specification. Accessed: Oct. 8, 2026. <https://eips.ethereum.org/EIPS/eip-1186>
- [41] Ethereum Improvement Proposals, “ERC-5539: Revocation List Registry,” Stagnant. Status checked: Oct. 8, 2026. <https://eips.ethereum.org/EIPS/eip-5539>
- [42] IPFS documentation, “Persistence, permanence, and pinning.” Accessed: Oct. 8, 2026. <https://docs.ipfs.tech/concepts/persistence/>
- [43] IPFS documentation, “Privacy and encryption.” Accessed: Oct. 8, 2026. <https://docs.ipfs.tech/concepts/privacy-and-encryption/>
- [44] B. Laurie, E. Messeri, and R. Stradling, “Certificate Transparency Version 2.0,” RFC 9162, Dec. 2021. <https://www.rfc-editor.org/rfc/rfc9162.html>
- [45] J. Camenisch and A. Lysyanskaya, “Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials,” in CRYPTO, 2002. <https://cs.brown.edu/people/alysyans/papers/camlys02.pdf>
- [46] Ethereum documentation, “Optimistic rollups.” Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/scaling/optimistic-rollups/>
- [47] Ethereum documentation, “Zero-knowledge rollups.” Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/scaling/zk-rollups/>
- [48] Ethereum documentation, “State channels.” Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/scaling/state-channels/>
- [49] P. Herbke et al., “ShadowPath: Lookup-Private Credential Status Verification over Authenticated State,” arXiv:2608.19937v1, preprint, Aug. 20, 2026. <https://arxiv.org/pdf/2608.19937v1>

- [50] Ethereum documentation, “Nodes and clients.” Accessed: Oct. 8, 2026. <https://ethereum.org/en/developers/docs/nodes-and-clients/>
- [51] Ethereum Improvement Proposals, “EIP-2200: Structured Definitions for Net Gas Metering.” Accessed: Oct. 8, 2026. <https://eips.ethereum.org/EIPS/eip-2200>
- [52] A. Backman et al., “Push-Based Security Event Token Delivery Using HTTP,” RFC 8935, Nov. 2020. <https://www.rfc-editor.org/rfc/rfc8935.html>
- [53] Google, “Colaboratory Frequently Asked Questions,” notebook upload and runtime guidance, accessed Oct. 9, 2026. Source
- [54] W3C, “Verifiable Credentials Data Model v2.1,” Working Draft, Sep. 30, 2026. Source
- [55] W3C, “Verifiable Credentials Data Model Threat Model v2.1,” Group Note Draft, Sep. 27, 2026. Source
- [56] Sovrin Foundation. The Sovrin Foundation Has Been Dissolved—but Sovrin MainNet Remains. 20 January 2026. Source
- [57] walt.id. Issuer2 Service: Overview. Updated 26 August 2026. Accessed 10 October 2026. Source